A VCL Model of a Cardiac Pacemaker

Jérôme Leemans and Nuno Amálio
Laboratory for Advanced Software Systems
University of Luxembourg
6, rue R. Coudenhove-Kalergi
L-1359 Luxembourg

TR-LASSY-12-04

Contents

Chapter 1

Introduction

Embedded software in medical devices is becoming ubiquitous and increasing in content and complexity. Because of this complexity and the fact that many failures in existing devices have been attributed to coding errors, regulators demand rigorous assurance that software running on medical devices is safe, reliable and functions correctly [JIJ06]. It is therefore not a surprise that there has been a recent interest in applying formal methods to this domain [JIJ06, JJA10].

This document presents a visual model of a cardiac pacemaker device [Bos07b] expressed in the Visual Contract Language (VCL)[AK10, AKMG10, AGK11]. The model is built according to an informal specification [Bos07b], released into the public domain by Boston Scientific, and which constitutes a pilot case study of the software verification challenge [WLBF09].

The model presented here has been developed using VCL's tool, the Visual Contract Builder (VCB) [AGK11][1]. It is a revised version of the model presented in [Lee11]; at the time [Lee11] was developed, VCB did not support all VCL diagram types (in particular, contract diagrams).

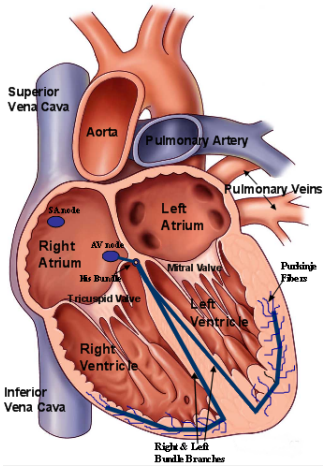


Figure 1.1: Human heart and its conduction system

During the development of the VCL model, we encountered many issues with the requirements of the pacemaker device [Bos07b]. These issues are documented in appendix A. To proceed with the modelling effort, assumptions regarding these issues were taken; these are documented in appendix B.

1.1 Heart and Pacemakers

The heart is a beating muscle that continuously pumps blood to the rest of the body. It comprises four chambers (Fig. 1.1): two *atria* (top) and two *ventricles* (bottom). The heartbeat is the rhythmic contraction of the heart's four chambers, which is stimulated by electrical signals that travel through a specific nerve pathway in the heart. These electrical signals begin at the *sinoatrial node* (SA Node), which is located in the right atrium (see Fig. 1.1). The signal then travels through the atria, causing their contraction, to reach the *atrioventricular node* (AV node), which amplifies the signal sending it through the ventricles, causing their contraction and the consequent pumping of blood into the rest of the body. The heart's conduction system functions as the body's

[1] `http://vcl.gforge.uni.lu`

own pacemaker. If it does not work properly, due to heart damage or other medical conditions, it can cause an irregular heartbeat, compromising the flow of blood to the brain and other parts of the body.

A cardiac pacemaker (PM) is an electronic device implanted in the body to regulate the heartbeat [BSS10]. It should be used when the heart's natural conduction system is damaged: either due to a sick SA node (bradycardia), or a block that prevents the signal from reaching the AV node (heart block), or other heart malfunctions [MM07]. This document presents a VCL model of a cardiac pacemaker [Bos07b]. This is a pilot case study of the software verification challenge.

1.2 The Visual Contract Language

The Visual Contract Language (VCL) [AK10, AKMG10, AGK11] is a formal language designed for the abstract description of software systems. It's modelling paradigms are set theory, object-orientation and design-by-contract (pre- and post-conditions). VCL distinguishes itself from other visual modelling languages, such as UML, by its capacity to describe predicates visually.

VCL takes a generative (or translational) approach to semantics. Currently, VCL has a Z semantics; the Z specification generated from the diagrams constituting the formal semantics of a VCL model. The Z enables formal model analysis through theorem proving. The Z specifications generated from VCL follow ZOO [APS05, Amá07], an approach to build object-oriented models in Z (a Z style).

VCL's tool, the Visual Contract Builder (VCB) [AGK11][2], an Eclipse plug-in, enables the construction of VCL diagrams, checks their well-formedness through type-checking, and generates Z specifications from them.

VCL's diagram suite comprises: *package*, *structural*, *behaviour*, *assertion* and *contract* diagrams. Package diagrams (PDs) define VCL packages, coarse-grained modules, and their dependencies with other packages (e.g. Figs. 2.1 and 4.1(a)). Structural diagrams (SDs) define structures and their relations that together make the state space of a package (e.g. Figs. 2.2 and 4.1(b)). Behaviour diagrams provide a map over the behaviour units of a package (e.g. Fig. 4.2). Assertion (or constraint) diagrams (ADs) define predicates over a single state, which can be used to define invariants and observe operations (or queries). Finally, contract diagrams (CDs) describe operations that change state through a contract (a pre- and a post-condition).

1.3 Outline

This document develops a VCL model of a cardiac pacemaker device [Bos07b], which is formally validated using snapshot analysis. This chapter introduced the case study and gave some background on VCL. The subsequent chapters are as follows:

- Chapters 2 to 18 present the packages that make the VCL model of the cardiac pacemaker.
- Chapter 19 presents the snapshot analysis performed on the Pacemaker VCL model.
- Appendix A documents the issues that we found with the requirements of the cardiac Pacemaker device documented in [Bos07b].
- Appendix B documents the assumptions that have been taken concerning the requirements issues that we found. The VCL model presented here is based on these assumptions.

[2] `http://vcl.gforge.uni.lu`

- Appendix C presents the Z generated from the VCL model developed in this document.
- Appendix D presents the ZOO toolkit (VCL's semantic domain), on which the Z specification rests.

Chapter 2

The CommonPM package

This package contains sets that are common across the different packages of the Pacemaker model. The package's PD (Fig. 2.1) defines `CommonPM` as a container package.

Figure 2.1: VCL Package diagram of package `CommonPM`

In the SD (Fig. 2.2), `BradycardiaMode` is a primary blob defining an enumerated set containing all possible operating modes of a DDD pacemaker [Bos07b, BSS10]; they are as follows: O (none), A (atrial), V (ventriculum), D (dual), T (triggered), I (inhibited), D (tracked), R (rate modulation). Other primary blobs defined this way are `Bool`, `Switch`, `BateryStatus` and `PMEvent`. Blob `BatteryStatus` holds all possible status of the device's battery as defined in [Bos07b, BSS10]; namely: `BOL` (beginning of life), `ERN` (elective replacement near), `ERT` (elective replacement time) and `ERP` (elective replacement past). Blob `PMEvent` represents internal events that need to be propagated and result in some event action to be performed by some device component.

The derived blobs `ChambersPaced`, `ChambersSensed` and `ResponseToSensing` are defined extensionally from the `BradycardiaMode` primary blob. The derived blobs `MicroSec` (microseconds, μs), `MilliSec` (milliseconds, ms), `MicroVolt` (microvolts, μv), `MilliVolt` (millivolts, mv) and `PPM` (pulses per minute) are defined from the primitive blob `Nat` (represents set of natural numbers).

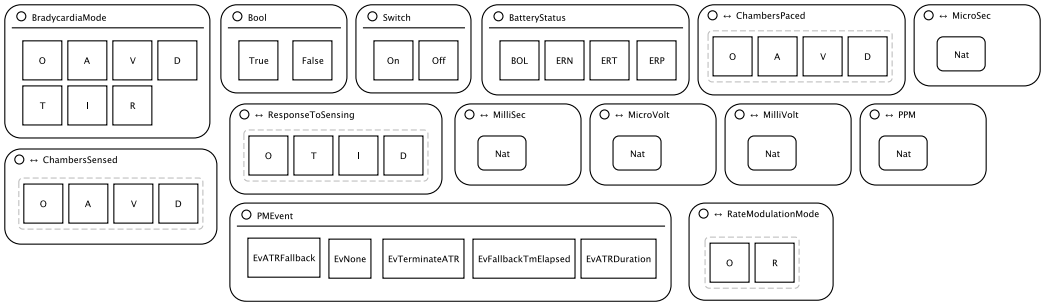


Figure 2.2: VCL Structural diagram of package `CommonPM`

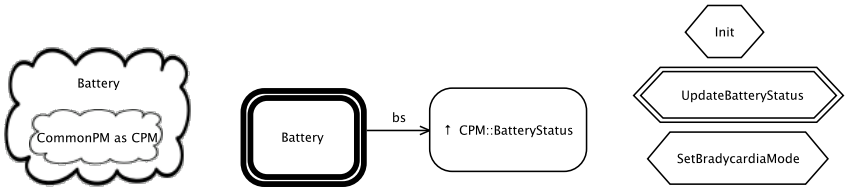
Chapter 3

The `Battery` package

3.1 State

Package `Battery` is responsible for managing the status of the device's battery. The package's PD (Fig. 3.1(a)) imports package `CommonPM` in order to access the set `BatteryStatus`.

Package `Battery`'s SD (Fig. 3.1(b)) adds the property edge `bs` of imported set `BatteryStatus` to the package blob in order to keep the current status of the battery. At any one time, the battery status may have the values: `BOL` (beginning of life), `ERN` (elective replacement near), `ERT` (elective replacement time) and `ERP` (elective replacement past).



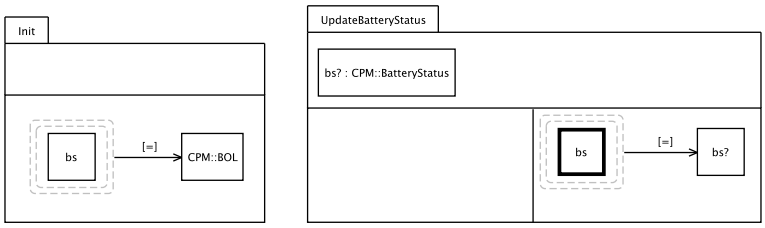
(a) Package Diagram (b) Structural Diagram (c) Behaviour Diagram

Figure 3.1: VCL Package, structural and behaviour diagram of package `Battery`

3.2 Overall behaviour

BD of package `Battery` (Fig. 3.1(c)) defines the following behavioural units:

- `Init` define the package's initialisation. Initialisation's defining AD (Fig. 3.2(a)) says that initially the battery is set to `BOL` (according to requirements assumption A7, appendix B).
- `UpdateBatteryStatus` is the operation that is called when the battery status level changes. Its definition (CD of Fig. 3.2(b)) receives the current battery status level as an input and changes the current status of the battery to the one received as input (according to requirements assumption A9, appendix B).



(a) Initialisation (b) Operation `UpdateBatteryStatus`

Figure 3.2: Assertion diagrams of Initialisation and contract diagram of `UpdateBatterStatus` in package `Battery`

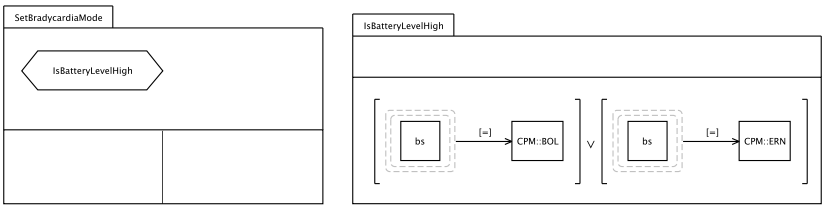


Figure 3.3: Assertion diagram of operation `SetBradycardiaMode` in package `Battery`

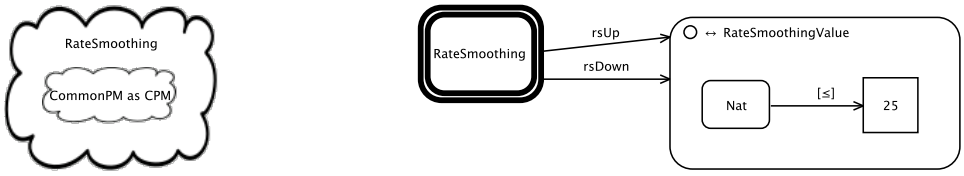
- `SetBradycardiaMode` describes the effect of this operation on this package. Its definition (CD of Fig. 3.3) describes a precondition that will affect the overall operation when composed; the precondition is described in AD `IsBatteryLevelHigh`, which says that `SetBradycardiaMode` may operate (changing the current bradycardia mode of the device) provided the battery status is `BOL` or `ERN`.

Chapter 4

The `RateSmoothing` package

Rate Smoothing is a programmable feature of Pacemakers designed to prevent sudden changes in the rhythm of pacing [BSS10]. A rate smoothing algorithm ensures gradual slowing or speeding of the pacemaker rate based on a percentage of a preceding cardiac interval [MM07, TLMB08]. Package `RateSmoothing` localises this functionality of the pacemaker device.

4.1 State



(a) Package Diagram

(b) Structural diagram

Figure 4.1: VCL Package and structural diagram of package `RateSmoothing`

PD of package `RateSmoothing` is given Fig. 4.1(a). It defines ensemble package `RateSmoothing`, saying that it imports package `CommonPM`. This import enables use of common sets defined in `CommonPM`.

Package `RateSmoothing`'s SD is given in Fig. 4.1(b). It introduces the derived blob `RateSmoothingValue`, which defines a percentage between 0 and 25 according to the "programmable parameters table" of [Bos07b, p. 34].

In Fig. 4.1(b), the property edges of package blob `RateSmoothing` define the package's state:

- `rsUp` and `rsDown` define the programmable parameters, representing, respectively, the limit for the increase and decrease in the current cardiac cycle interval.

4.2 Overall Behaviour

BD of package `RateSmoothing` (Fig. 4.2) introduces the following behavioural units:

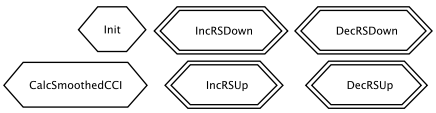


Figure 4.2: Behavioural Diagram of package `RateSmoothing`

- `Init` (Fig. 4.3) define the package's initialisation.
- `IncRSDown` (Fig. 4.4) and `DecRSDown` (Fig. 4.5) increment and decrement the programmed value of the rate smoothing down percentage.
- `IncRSUp` (Fig. 4.6) and `DecRSUp` (Fig. 4.7) increment and decrement the programmed value of the rate smoothing up percentage.
- `CalcSmoothedCCI` (Fig. 4.8) calculates a new cardiac cycle interval based on the rate smoothing settings.

4.3 Global Operations

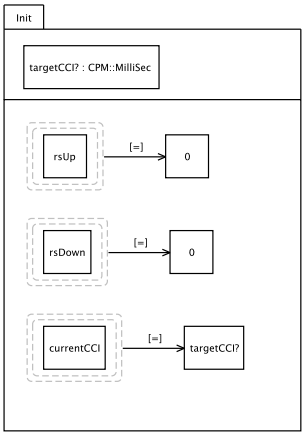


Figure 4.3: Initialisation of package `RateSmoothing`

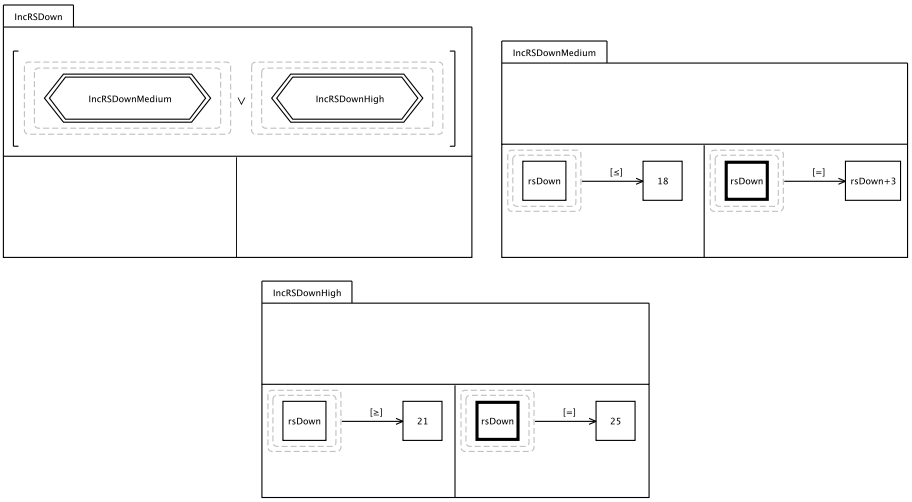


Figure 4.4: Operation `IncRSDown` of package `RateSmoothing`

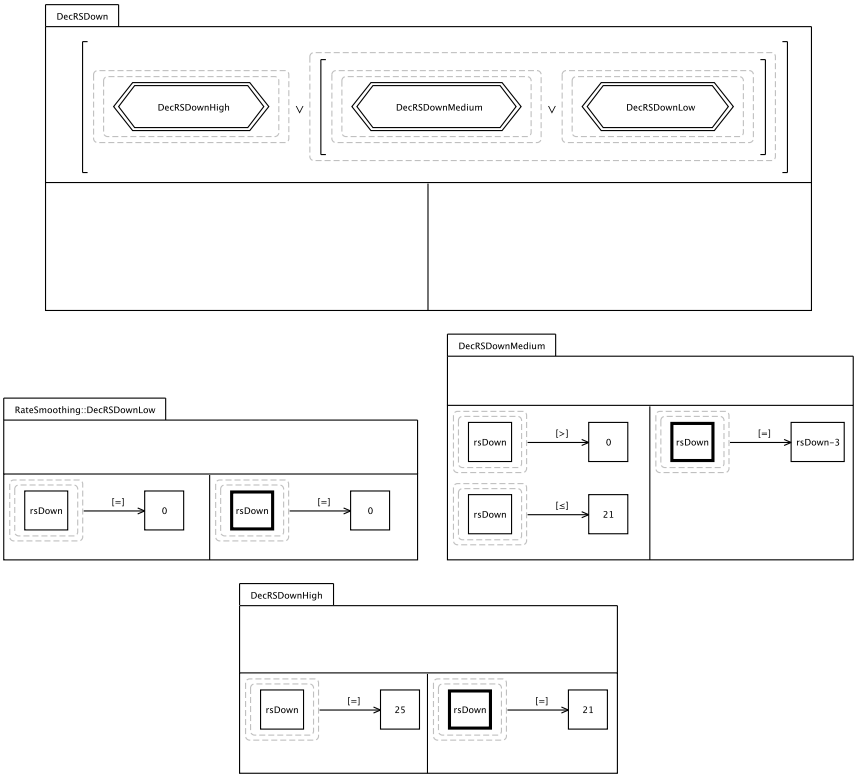


Figure 4.5: Operation `DecRSDown` of package `RateSmoothing`

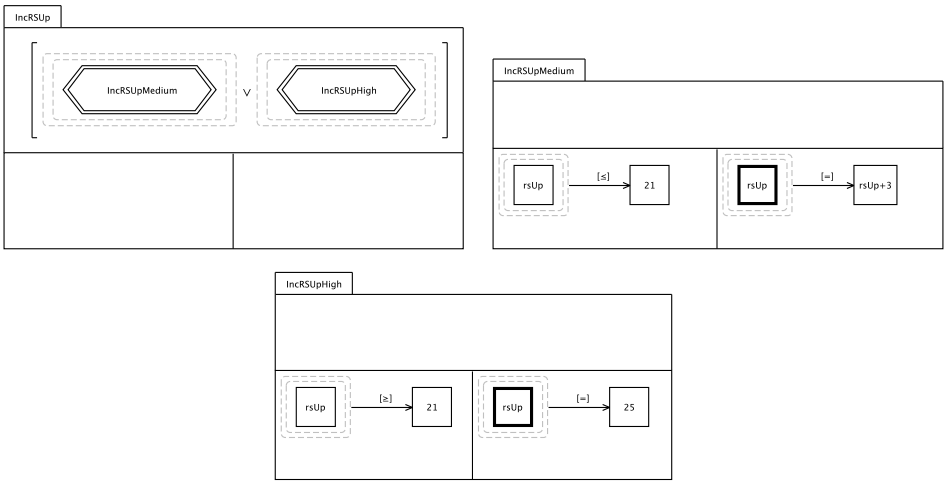


Figure 4.6: Operation `IncRSUp` of package `RateSmoothing`

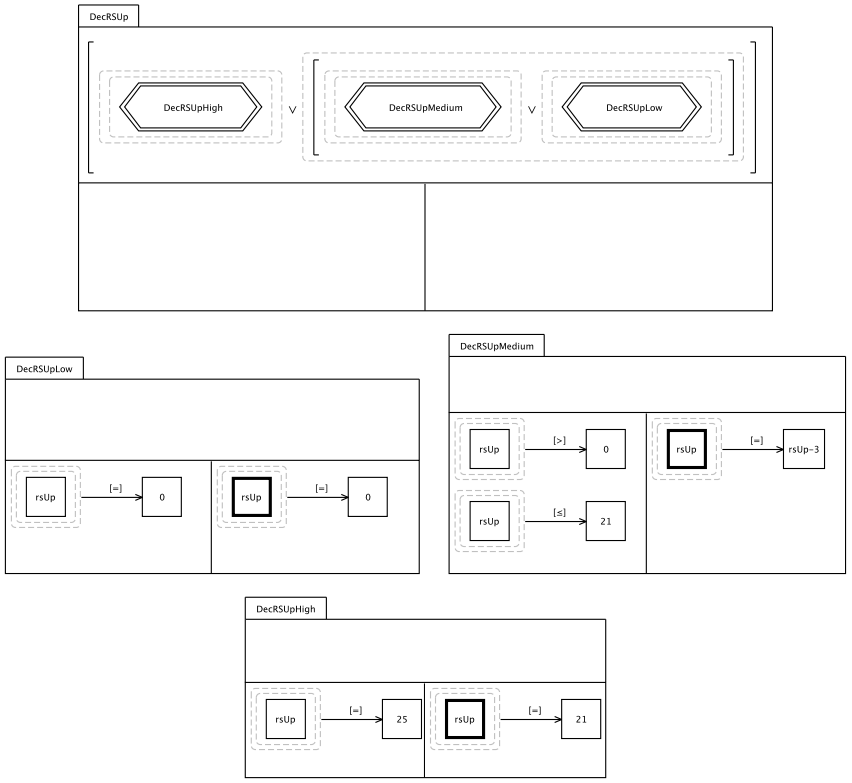


Figure 4.7: Operation `DecRSUp` of package `RateSmoothing`

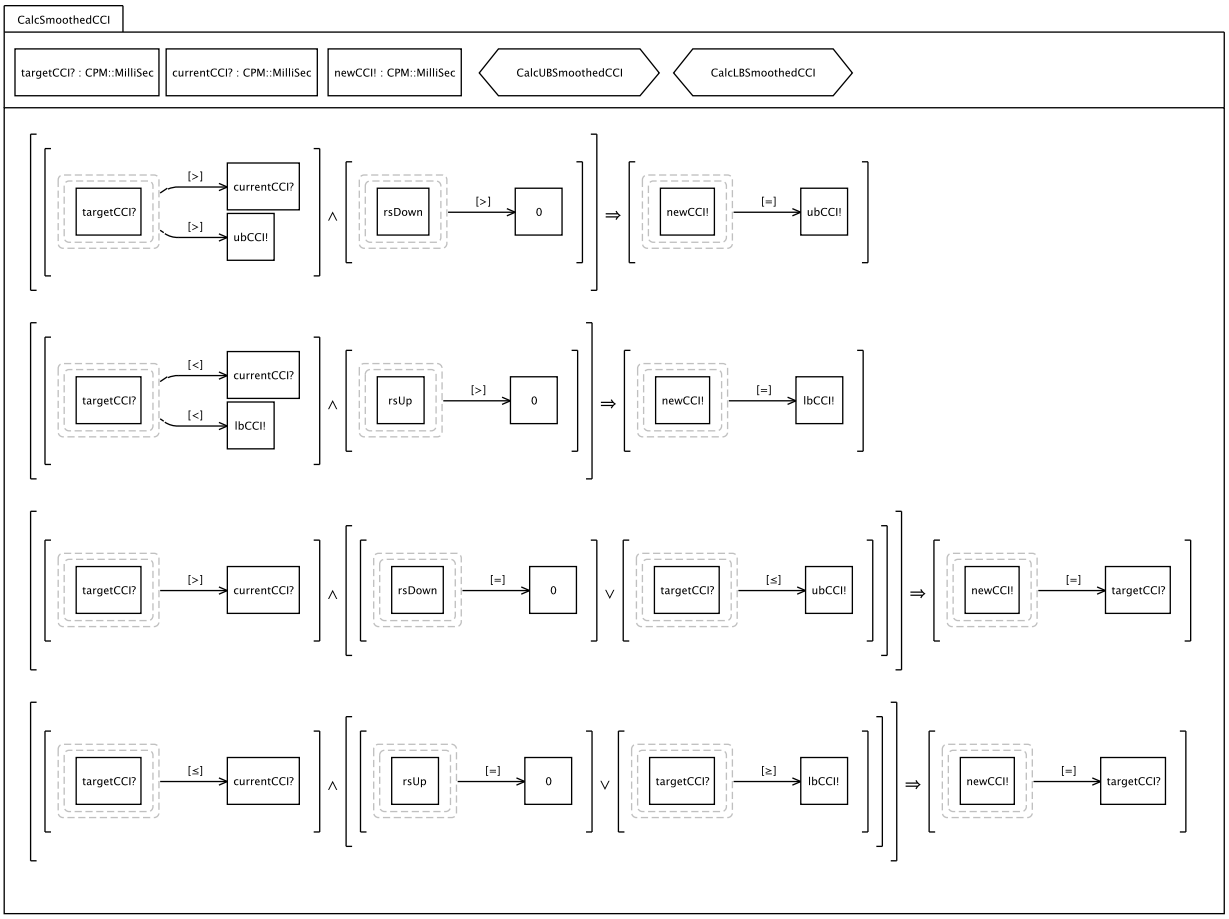


Figure 4.8: Assertion Diagram of observe operation `CalcSmoothedCCI`

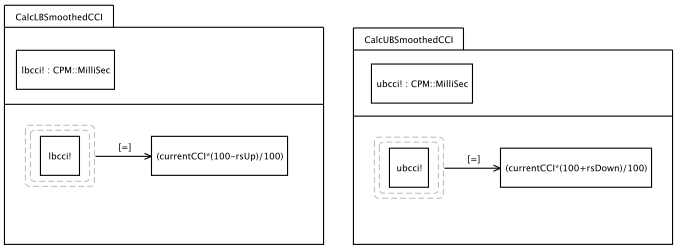


Figure 4.9: Assertion Diagrams of operations `CalcLBSmoothedCCI` and `CalcUBSmoothedCCI` of package `RateSmoothing`

Chapter 5

The `AVDelay` package

The atrioventricular (AV) Delay or atrioventricular interval (AVI) is the interval between an atrial event (either sensed or paced) and the scheduled delivery of a ventricular stimulus [MM07]. The package `AVDelay` manages this functionality of the pacemaker device.

5.1 State

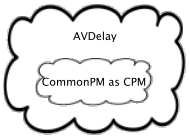


Figure 5.1: VCL Package diagram of package `AVDelay`

PD of package `AVDelay` is given Fig. 5.1. It defines ensemble package `AVDelay`, saying that it imports package `CommonPM`. This import enables use of common sets defined in `CommonPM`.

Package `AVDelay`'s SD (Fig. 5.2) introduces the following derived blobs:

- `MinDynAVDelay` represents the allowed values for the minimum AV delay as defined in the "programmable parameters table" of [Bos07b, p. 34].
- `MaxDynAVDelay` represents the allowed values of both fixed AV delay and maximum dynamic AV delay. This is defined according to "Fixed AV Delay" in the "programmable parameters table" of [Bos07b, p. 34] and results from requirements assumption A1 (appendix B), which resolves issue RI1 in the requirements (appendix A).
- `SensedAVDelayOffset` represents the allowed values of the parameter sensed AV delay offset, expressed in milliseconds and which is used to shorten the AV Delay following a tracked atrial sense. This blob is defined according to "Sensed AV Delay Offset" interval defined in the "programmable parameters table" of [Bos07b, p. 34]

The SD's package blob (Fig. 5.2) has property edges `fixedAVI` and `maxDynDelay`, `minDynDelay`, `minDynDelay`, `dynMode` and `savDelayOffset`; all programmable features of the device to be set

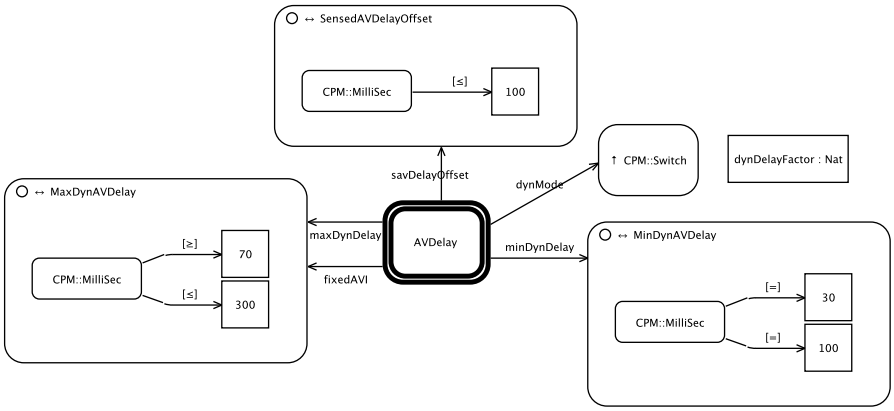


Figure 5.2: VCL structural diagram of package `AVDelay`

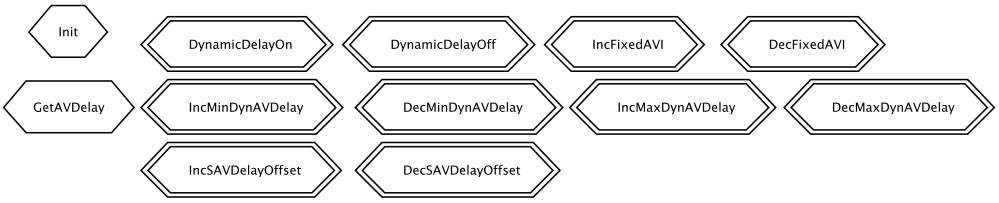


Figure 5.3: Behavioural Diagram of package `AVDelay`

by physicians. Property `dynMode` (of imported set `Switch`, which has values `On` and `Off`) indicates the dynamic mode is on or off; when off the fixed AVI is used. The constant `dynDelayFactor`, a natural number whose value is undefined, represents the "factor" (mentioned in [Bos07b, p. 29]) that is used to calculate the dynamic AV delay; this constant emerges from requirements assumption A3 (appendix B).

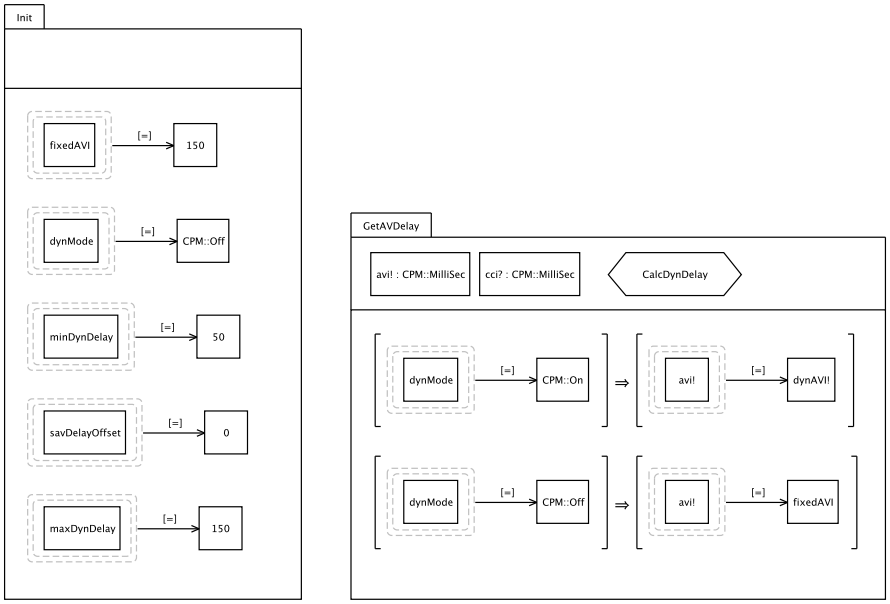
5.2 Overall Behaviour

BD of package `AVDelay` (Fig. 5.3) introduces the following behavioural units:

- `Init` (Fig. 5.4(a)) define the package's initialisation.
- `GetAVDelay` (Fig. 5.4(b)) defines an observe operation to yield the current AV delay. We consider that there is just one AV Delay, and not a sensed and paced AV Delay. This follows assumption A3 (appendix B).
- `DynamicDelayOn` and `DynamicDelayOff` turn the dynamic delay on and off.
- `IncFixedAVI` and `DecFixedAVI` increment and decrement the fixed AVI parameter (property edge `fixedAVI` above).

- `IncMinDynAvDely` and `DecMinDynAvDely` increment and decrement the parameter minimum AV Delay (property edge `minDynDelay` above).
- `IncMaxDynAVDelay` and `DecMaxDynDelay` increment and decrement the parameter maximum AV Delay (property edge `maxDynDelay` above).
- `IncSAVDelayOffset` and `DecSAVDelayOffset` increment and decrement the parameter sensed AV Delay offset (property edge `savDelayOffset` above).

5.3 Global Operations



(a) Initialisation

(b) Operation `GetAVDelay`

Figure 5.4: Assertion Diagrams of Initialisation and observe operation `GetAVDelay` in package `AVDelay`

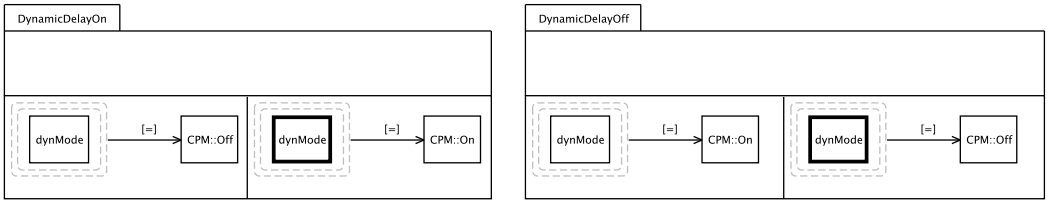


Figure 5.5: Contract Diagrams of operations `DynamicDelayOn` and `DynamicDelayOff`

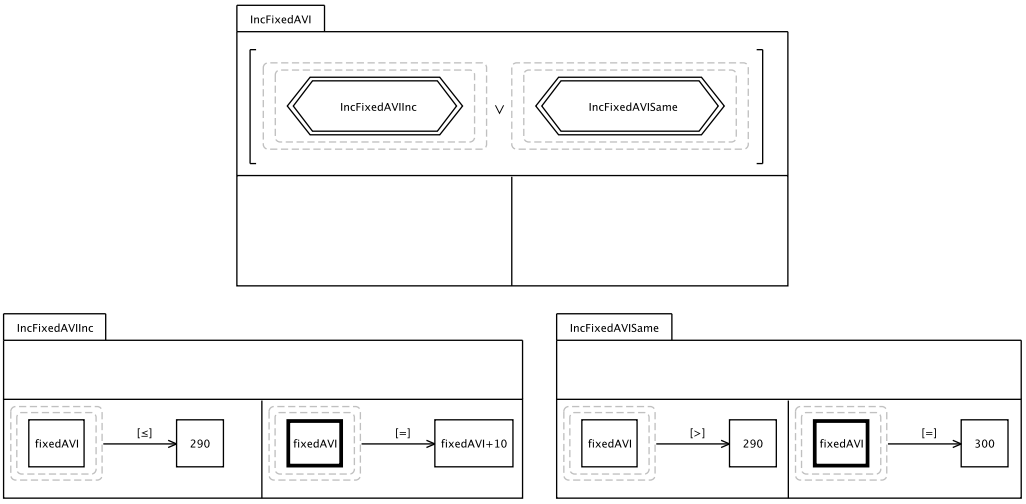


Figure 5.6: Contract Diagrams of operation `IncFixedAVI`

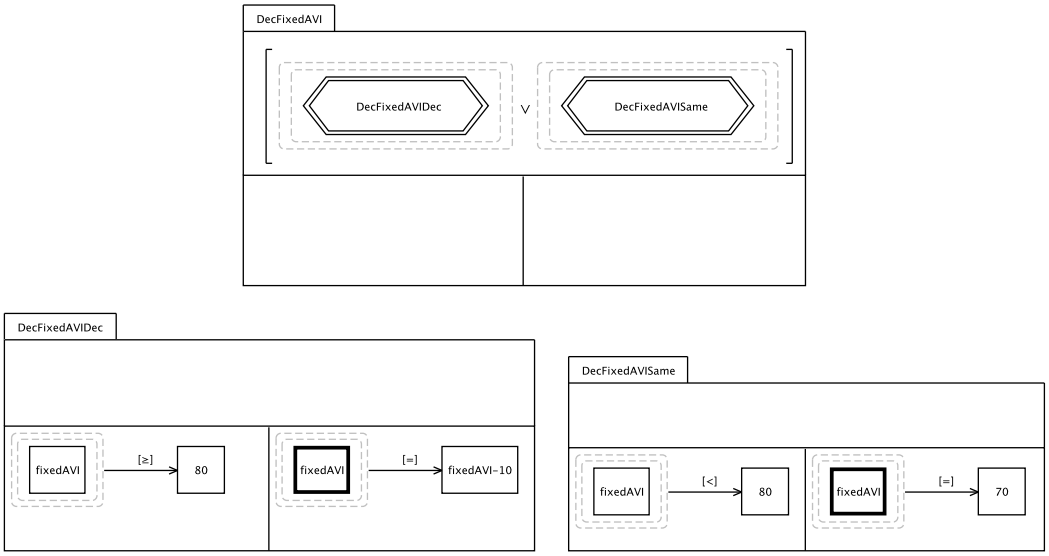


Figure 5.7: Contract Diagrams of operation `DecFixedAVI`

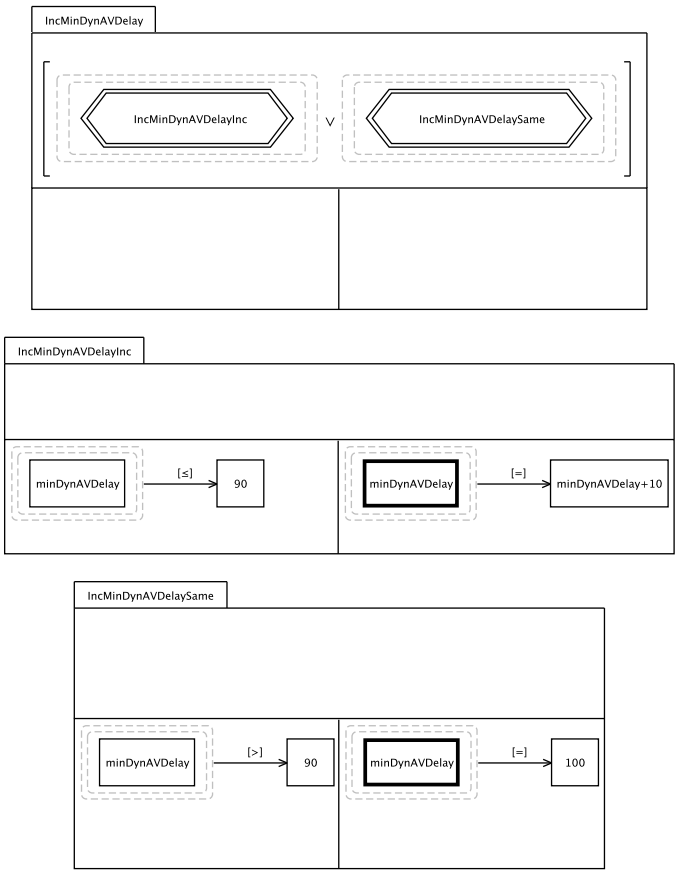


Figure 5.8: Contract Diagrams of operation `IncMinDynAVDelay`

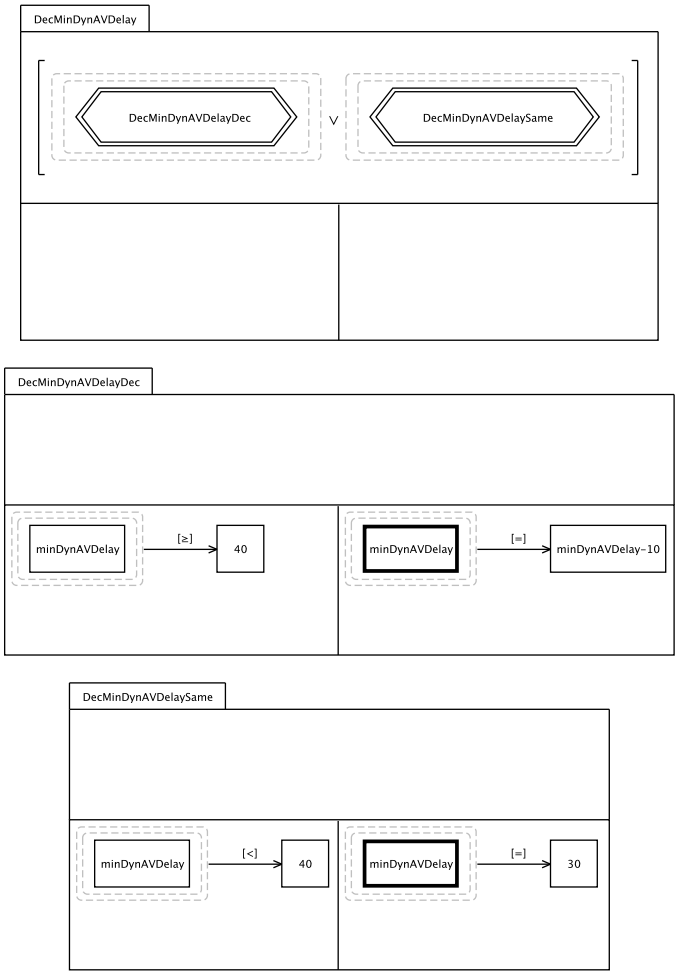


Figure 5.9: Contract Diagrams of operation `DecMinDynAVDelay`

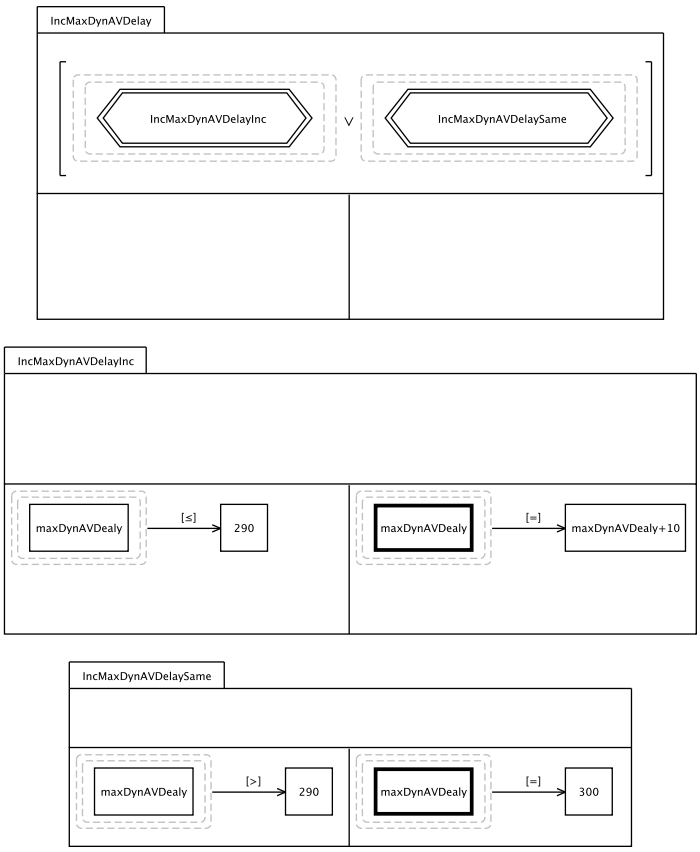


Figure 5.10: Contract Diagrams of operation `IncMaxDynAVDelay`

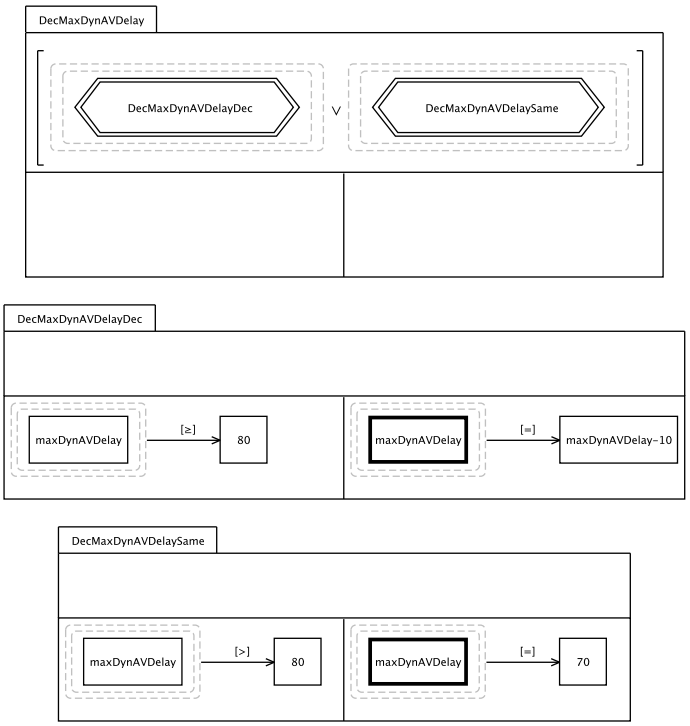


Figure 5.11: Contract Diagrams of operation `DecMaxDynAVDelay`

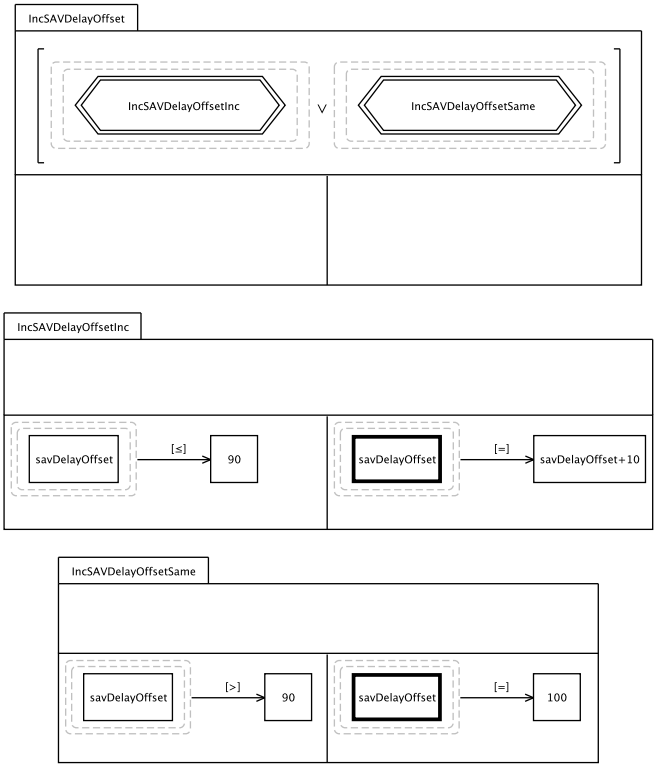


Figure 5.12: Contract Diagrams of operation `IncSAVDelayOffset`

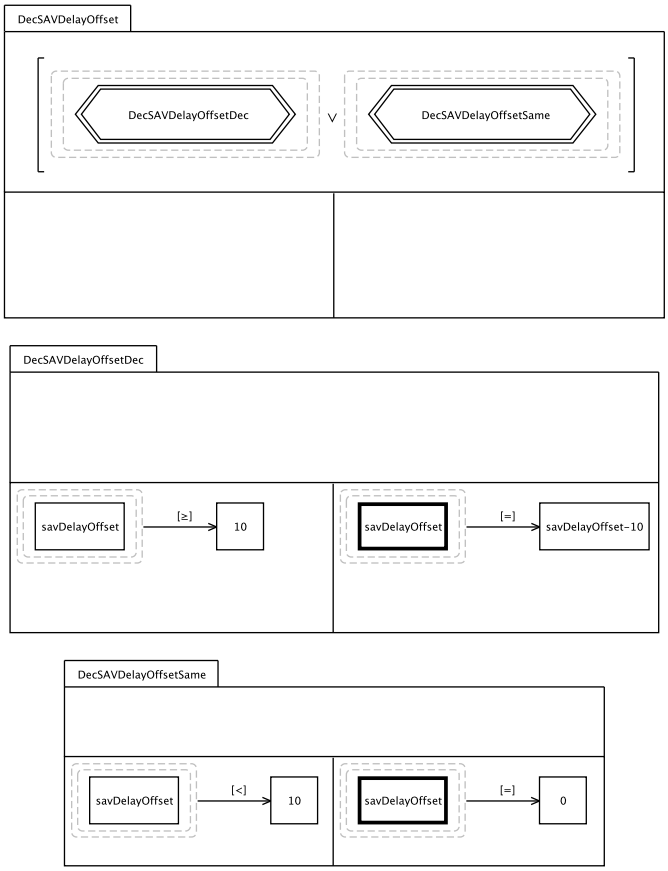


Figure 5.13: Contract Diagrams of operation `DecSAVDelayOffset`

Chapter 6

The `RateModulation` Package

Rate modulation or *rate adaptive* pacing is the ability of pacemakers to increase the pacing rate in response to physical activity or metabolic demand [MM07]. Package `RateModulation` deals with this functionality in the pacemaker device. In [Bos07b, sec. 5.7, p.32], this functionality is identified as rate adaptive pacing, which concerns the "ability to adjust the cardiac cycle in response to metabolic need as measured from body motion using an accelerometer.

6.1 State

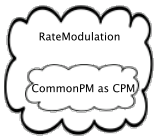


Figure 6.1: VCL Package diagram of package `RateModulation`

`RateModulation`'s PD (Fig. 6.1) defines `RateModulation` as an ensemble package, which imports container package `CommonPM`.

SD of `RateModulation` is given in Fig. 6.2. It introduces several derived blobs representing the allowed values of programmable parameters; they are as follows:

- `MaximumSensorRate` (MSR) represents the possible maximum pacing rates allowed under sensor control [Bos07b, sec. 5.7.1, p. 32]. Blob `MaximumSensorRate` defines values that represent units as paces per minute (ppm); it is defined according to the "programmable parameters table" of [Bos07b, p. 34]. This is used when the patient increases activity and the sensor responds, the heart rate then rises with a programmed response up to a maximal rate defined by the MSR [MM07].
- `ResponseFactor` is used to calculate the pacing rate at various levels of steady state patient activity; depending on the programmed value of response factor, the incremental change of rate may be higher or lower [Bos07b, sec. 5.7.3, p. 32]. This is defined according to the "programmable parameters table" of [Bos07b, p. 34].

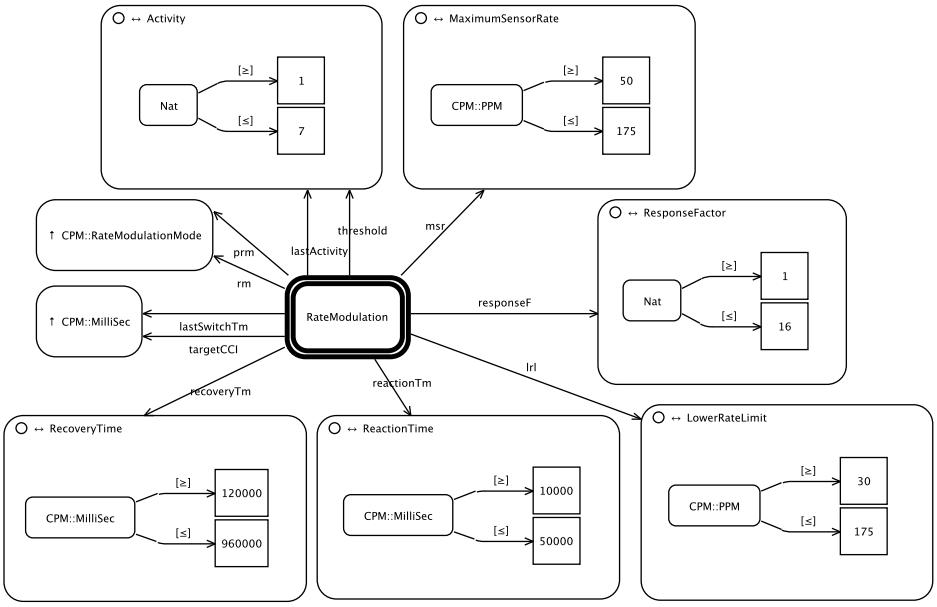


Figure 6.2: VCL structural diagram of package `RateModulation`

- `ReactionTime` is the time required to drive the rate from the lower rate limit (LRL) to MSR [Bos07b, sec. 5.7.4, p. 33]. Blob `ReactionTime` defines defines values that represent units as miliseconds (ms) and is defined according to the "programmable parameters table" of [Bos07b, p. 34].
- `RecoveryTime` is the time required for the rate to fall from MSR to LRL [Bos07b, sec. 5.7.5, p. 33]. Blob `RecoveryTime` defines defines values that represent units as miliseconds (ms) and is defined according to the "programmable parameters table" of [Bos07b, p. 34].
- `Activity` represents the various levels of patient activity as output by the accelerometer [Bos07b, sec. 5.7.2, p. 32]. This is defined according to the parameter *activity threshold* of the "programmable parameters table" of [Bos07b, p. 34] and following requirements assumption A5 (appendix B). The different values specified in [Bos07b, p. 34] are here in the model interpreted as natural numbers (1: V-Low; 2: Low; 3: Med-Low; 4: Med; 5:Med-High; 6: High; 7:V-High).
- `LowerRateLimit` represents the allowed values of *lower rate limit* (LRL); this is expressed in ppm and defined according to the "programmable parameters table" of [Bos07b, p. 34]. LRL represents the number of generator pace pulse delivered per minute (atrium or ventricle) in the absence of sensed intrinsic activity or sensor-controlled pacing at a higher rate [Bos07b, p. 28]. `LowerRateLimit`.

In Fig. 6.2, the property edges of the package blob `RateModulation` represent the state of the package. They are as follows:

- `msr`, `responseF`, `reactionTm` and `recoveryTm` represent the programmed values for parameters MSR, response factor, and reaction and recovery times.

- `rm` and `prm` represent the current and previous bradycardia mode for rate modulation.
- `threshold` and `lastActivity` are related to the accelerometer output. `threshold` is the value that enables adaptive pacing; if the accelerometer output exceeds this value then the pacemaker's rate is affected by activity data [Bos07b, sec. 5.7.2, p. 32]. `lastActivity` captures the last accelerometer output, following requirements assumption A5 (appendix B).
- `lastSwitchTm` is used by the rate-adaptation algorithm. It marks the time where there is a switch of the target cardiac cycle interval from the MSR to LRL or the other way round.

6.2 Overall Behaviour

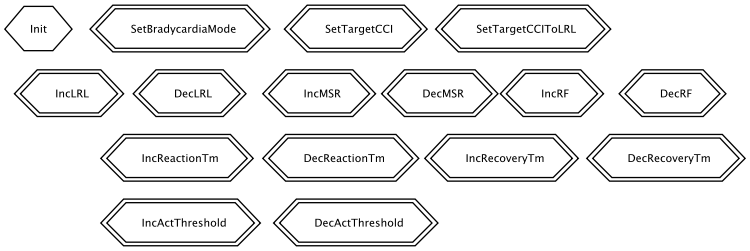


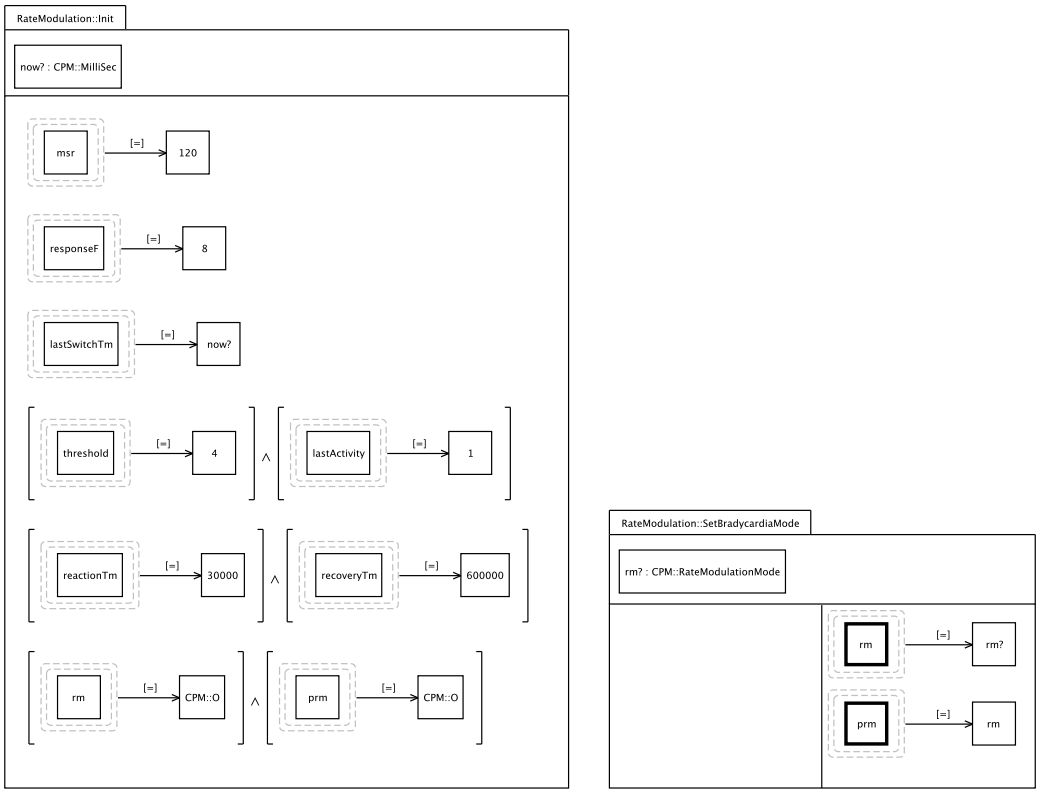
Figure 6.3: VCL behavioural diagram of package `RateModulation`

BD of package `RateModulation` is given in Fig. 6.3. It introduces the following behavioural units:

- `Init` (Fig. 6.4(a)) is the package's initialisation.
- `SetBradycardiaMode` (Fig. 6.4(b)) is the operation that sets the bradycardia mode for rate modulation.
- `SetTargetCCI` (Fig. 6.5) is the operation that sets the target cardiac cycle interval taking into account rate modulation. This involves a complex processing according to what is described in [Bos07b, sec. 5.7, p. 32]. The diagrams from figures 6.5 to 6.9 are involved in this.
- `SetTargetCCIToLRL` (Fig. 6.10) is the operation that sets the target cardiac cycle interval to the LRL.
- `IncLRL` (Fig. 6.11) and `DecLRL` (Fig. 6.12) increment and decrement the programmed value of the lower rate limit.
- `IncMSR` (Fig. 6.13) and `DecMSR` (Fig. 6.14) increment and decrement the programmed value of MSR.
- `IncRF` (Fig. 6.15) and `DecRF` (Fig. 6.16) increment and decrement the programmed value of response factor.
- `IncReactionTm` (Fig. 6.17) and `DecReactionTm` (Fig. 6.18) increment and decrement the programmed value of reaction time.

- `IncRecoveryTm` (Fig. 6.19) and `DecRecoveryTm` (Fig. 6.20) increment and decrement the programmed value of recovery time.
- `IncActThreshold` (Fig. 6.21) and `DecActThreshold` (Fig. 6.22) increment and decrement the activity threshold.

6.3 Global Operations



(a) Initialisation

(b) Operation `SetBradycardiaMode`

Figure 6.4: Package `RateModulation`: Assertion diagram of Initialisation and contract diagram of operation `SetBradycardiaMode`

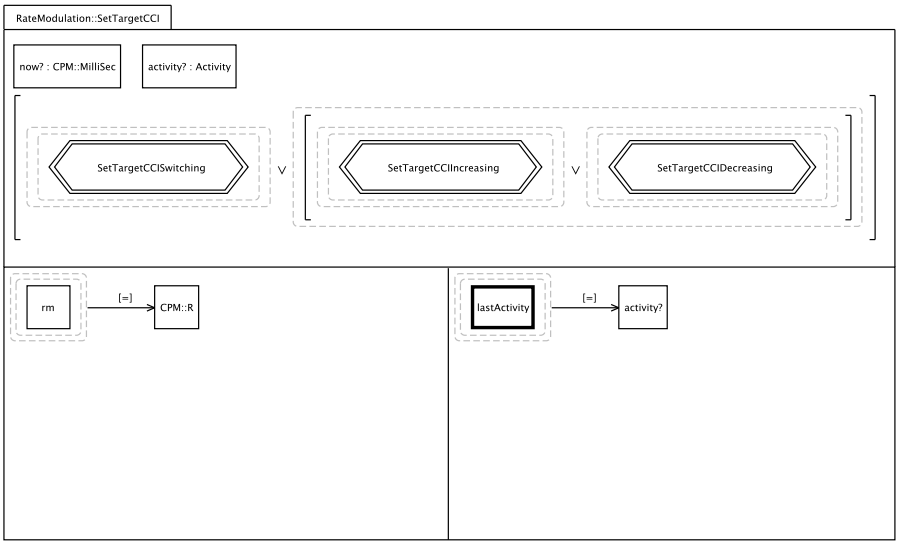


Figure 6.5: VCL contract diagram of operation `SetTargetCCI` in package `RateModulation`

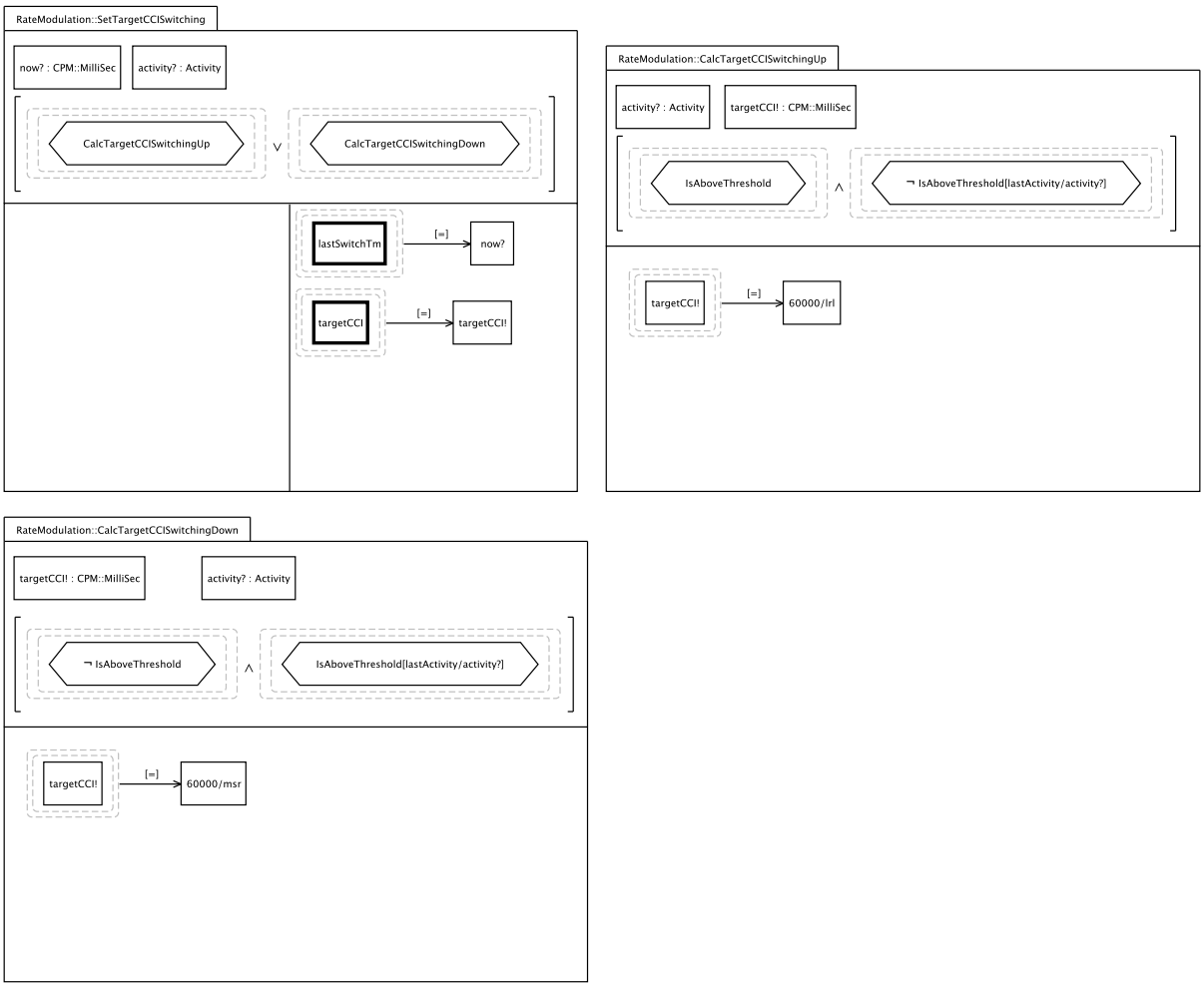


Figure 6.6: VCL contract and assertion diagrams of operation `SetTargetCCISwitching` in package `RateModulation`

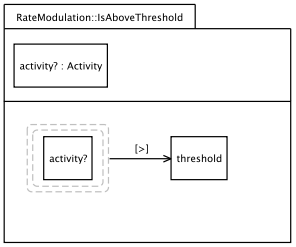


Figure 6.7: VCL assertion diagrams of predicate `IsAboveThreshold` in package `RateModulation`

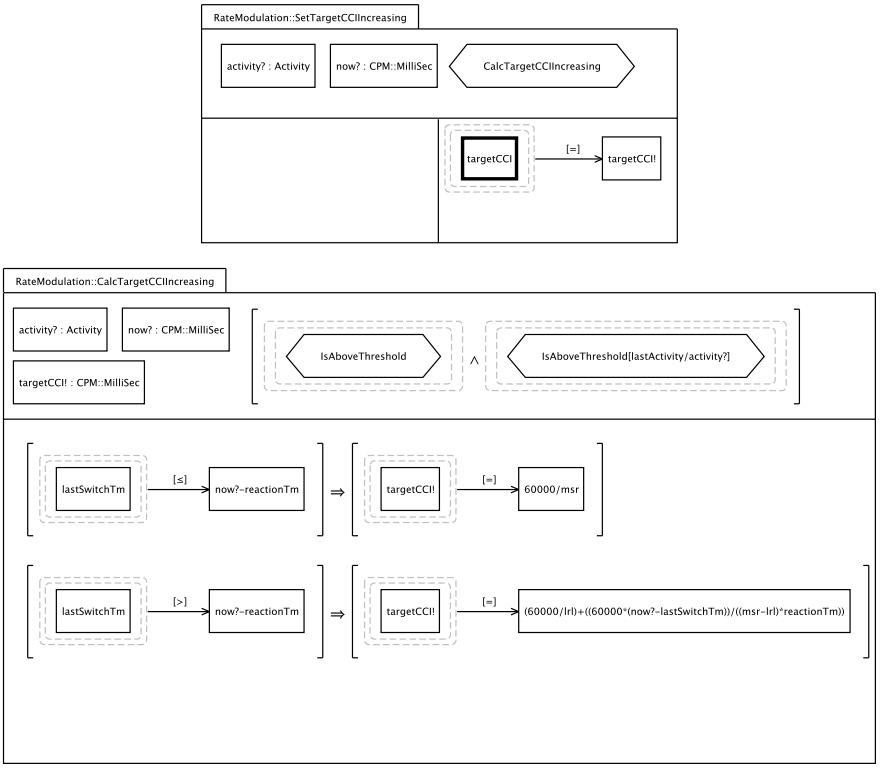


Figure 6.8: VCL contract and assertion diagrams of operation `SetTargetCCIIncreasing` in package `RateModulation`

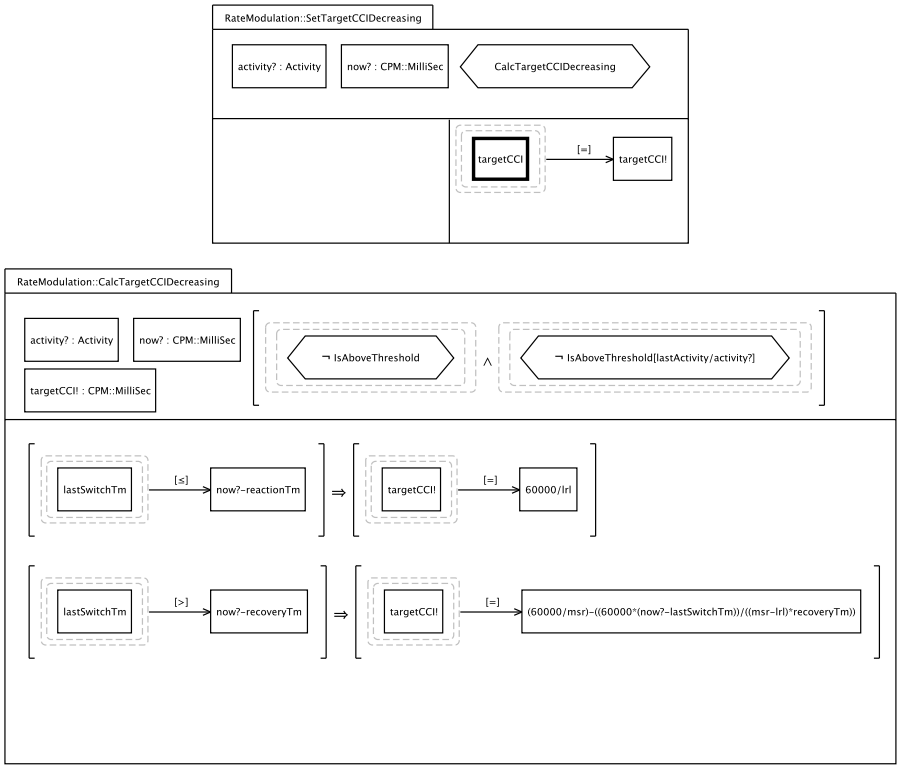


Figure 6.9: VCL contract and assertion diagrams of operation `SetTargetCCIDecreasing` in package `RateModulation`

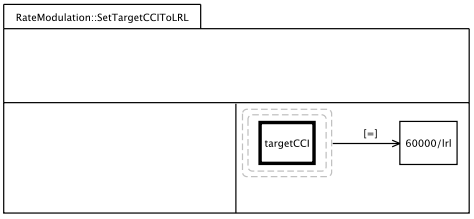


Figure 6.10: VCL contract diagram of operation `SetTargetCCIToLRL` in package `RateModulation`

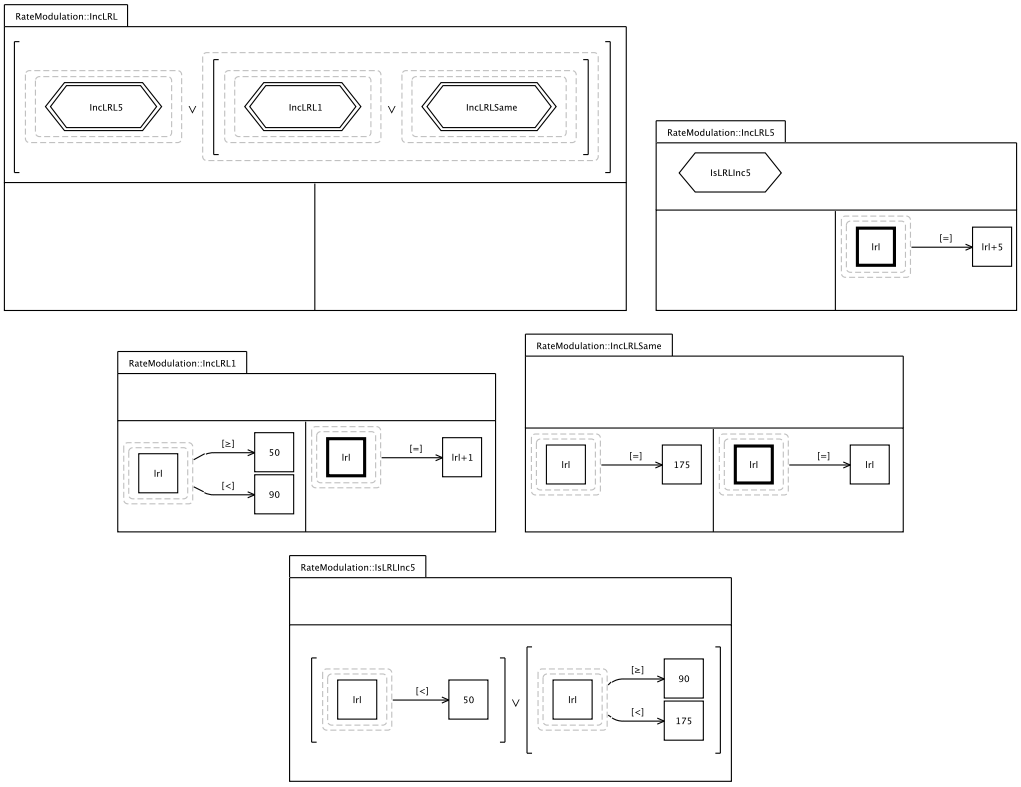


Figure 6.11: Contract and assertion diagrams of global operation `IncLRL`

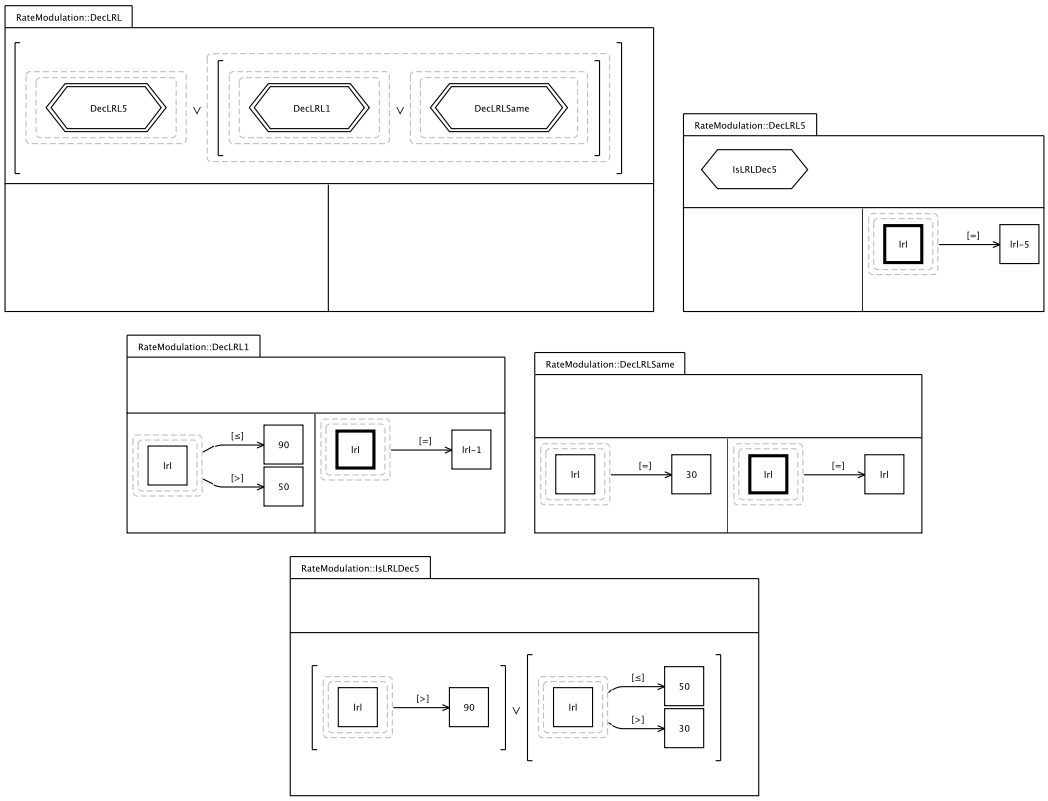


Figure 6.12: Contract and assertion diagrams of global operation `DecLRL`

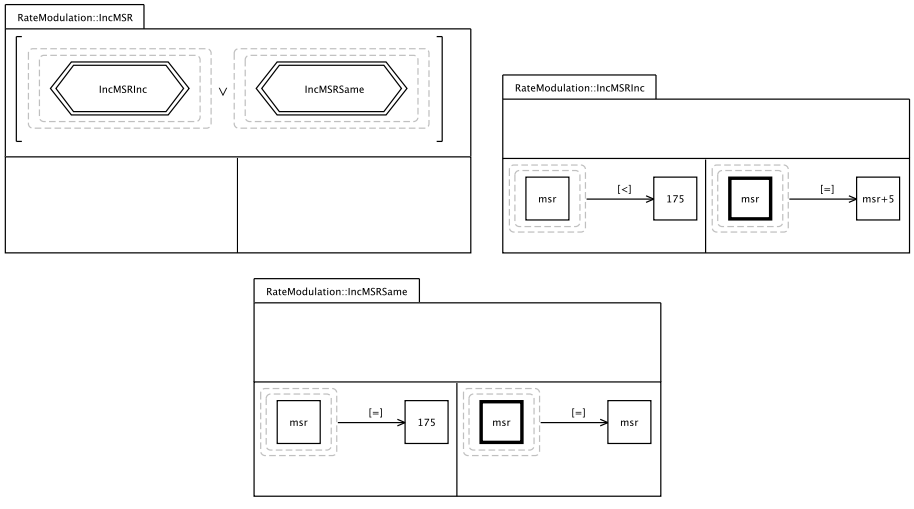


Figure 6.13: Contract diagrams of operation `IncMSR` of package `RateModulation`

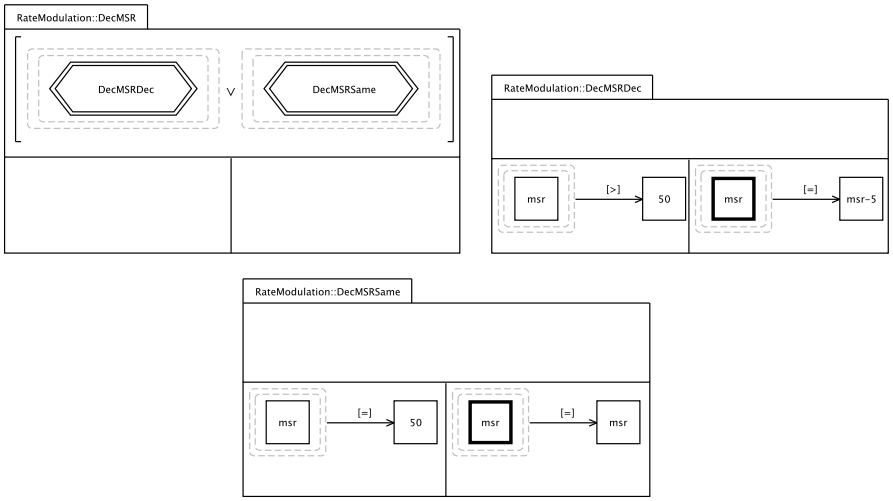


Figure 6.14: Contract diagrams of operation `DecMSR` of package `RateModulation`

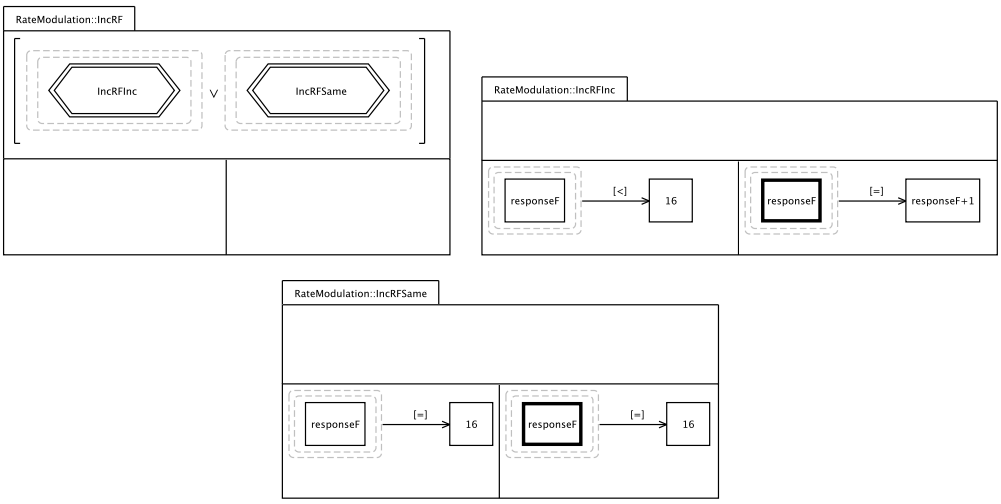


Figure 6.15: Contract diagrams of operation `IncRF` of package `RateModulation`

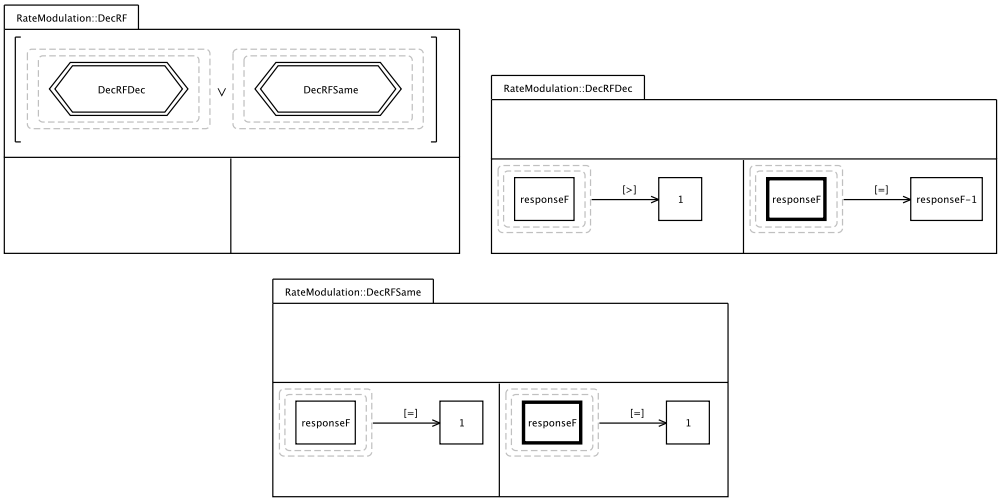


Figure 6.16: Contract diagrams of operation `DecRF` of package `RateModulation`

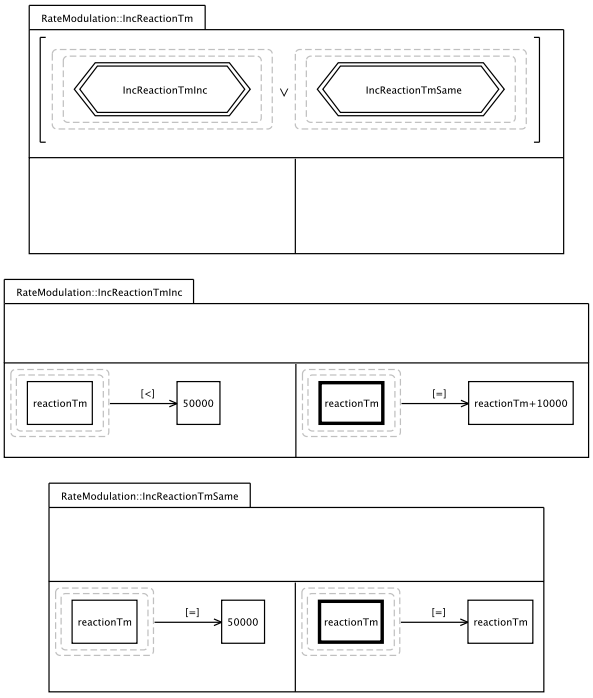


Figure 6.17: Contract diagrams of operation `IncReactionTm` of package `RateModulation`

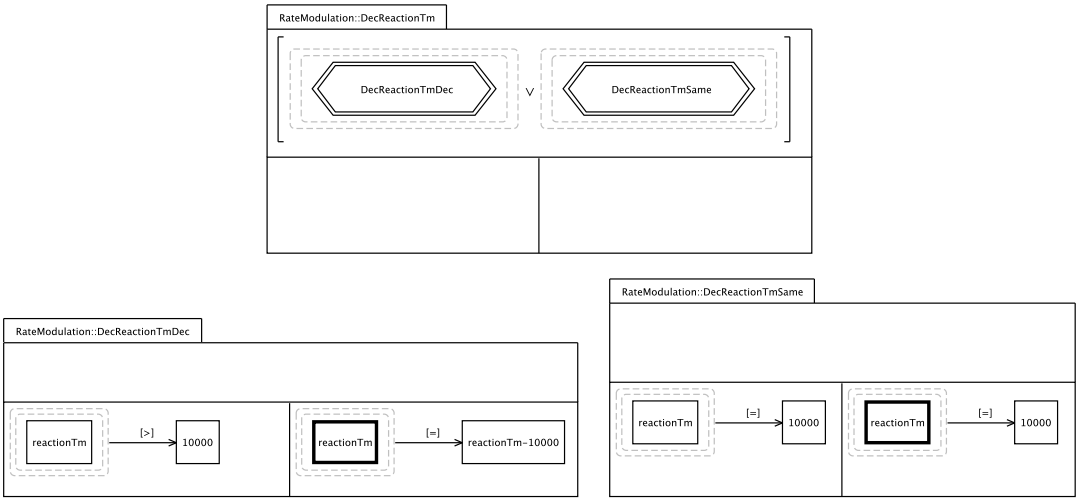


Figure 6.18: Contract diagrams of operation `DecReactionTm` of package `RateModulation`

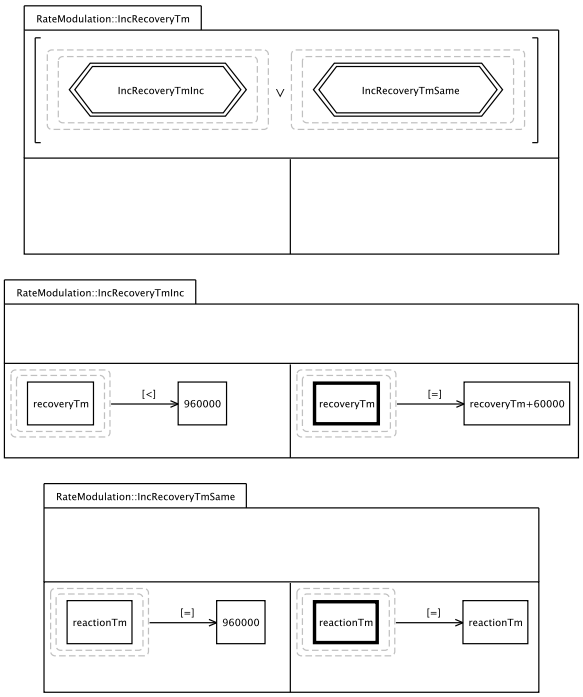


Figure 6.19: Contract diagrams of operation `IncRecoveryTm` of package `RateModulation`

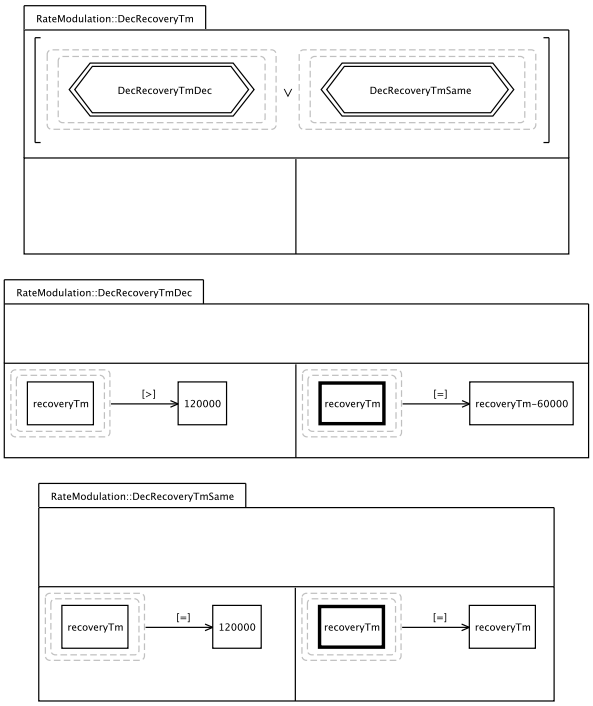


Figure 6.20: Contract diagrams of operation `DecRecoveryTm` of package `RateModulation`

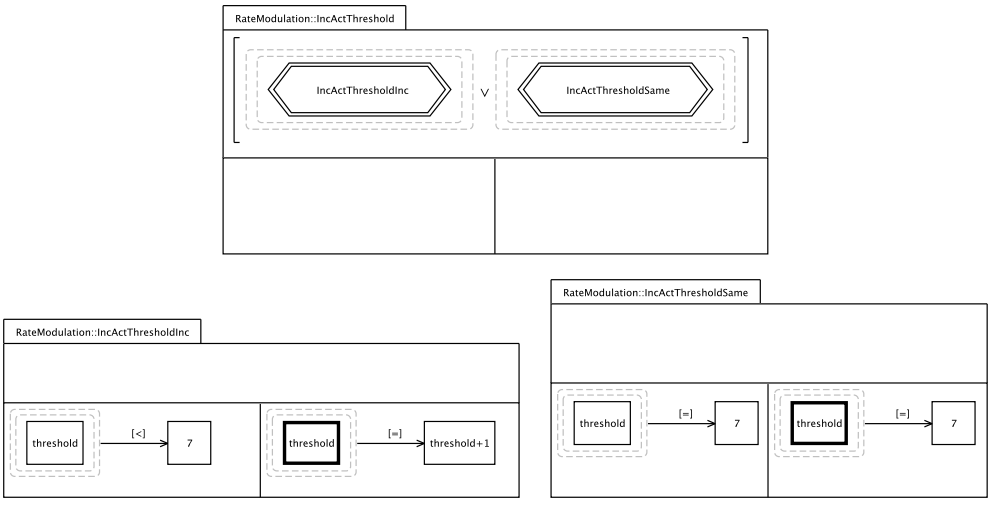


Figure 6.21: Contract diagrams of operation `IncActThreshold` of package `RateModulation`

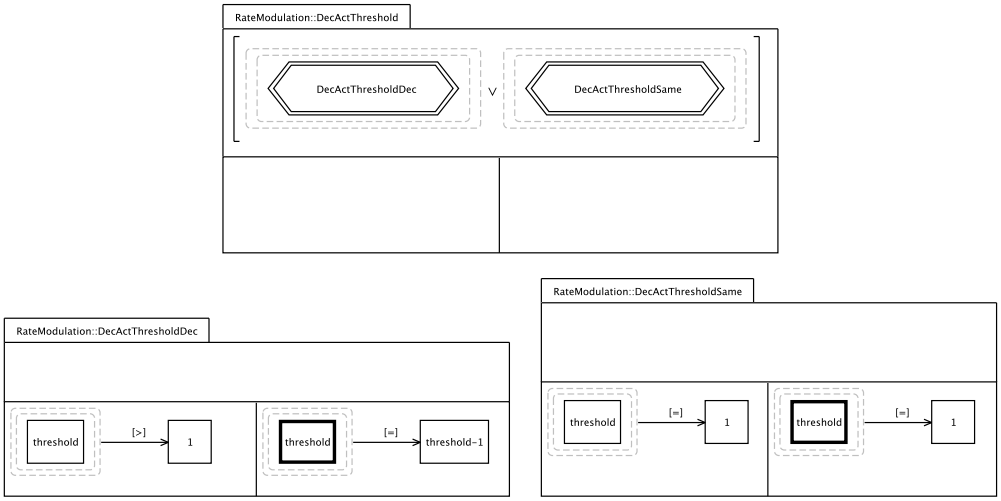


Figure 6.22: Contract diagrams of operation `DecActThreshold` of package `RateModulation`

Chapter 7

The `Pacing` package

7.1 State

Pacing is the functionality of the Pacemaker device that stimulates the heart through electric pulses. A pulse has an amplitude or peak value (a voltage) and a width or duration [BSS10]. The `Pacing` package presented here manages the pacing feature of the Pacemaker device; it is responsible for emitting electrical pulses to both the ventricular and atrial chambers of the heart.

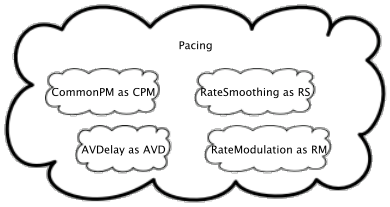


Figure 7.1: VCL Package diagram of package `Pacing`

PD of `Pacing` (Fig. 7.1) package says that it imports container package `CommonPM` and ensemble packages `RateSmoothing` and `AVDelay`, which are sub-features of pacing.

Figure 7.2 presents SD of `Pacing`, which introduces the following derived blobs:

- `PulseWidth` represents the allowed values of pulse width, which is defined from set `MicroSec` defined in `CommonPM` as described in the "programmable parameters table" of [Bos07b, p. 34]. `PulseWidth` is expressed in microseconds (μV).
- `PulseAmplitudeRegulated` represents the allowed values of the pulse amplitude during the regulated mode. It is defined according to the "programmable parameters table" of [Bos07b, p. 34]. Pulse amplitudes are expressed in millivolts (mv).
- `PulseAmplitudeUnregulated` represents the allowed values of the pulse amplitude during the unregulated mode. It is defined according to the "programmable parameters table" of [Bos07b, p. 34].

The remaining elements of the SD of Fig. 7.2 are as follows:

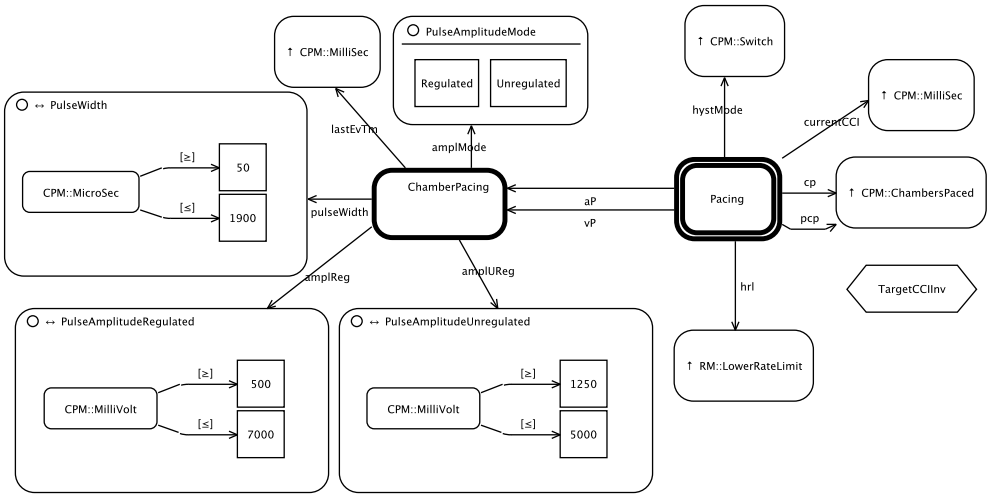


Figure 7.2: VCL structural diagram of package `Pacing`

- Domain blob `ChamberPacing` factors the pacing properties of a chamber (atrial or ventricular); this includes the programming parameters pulse width and pulse amplitude (regulated and unregulated), the last time a pacing event occurred (`lastEvTm`) and the current amplitude mode (`amplMode`, which can either be regulated or unregulated).
- Package blob `Pacing` has the following property edges: `aP` and `vP` of blob `ChamberPacing` (atrial and ventricular pacing), `lrl` (lower rate limit), `hrl` (hysteresis rate limit), `rm` and `prm` of blob `RateModulation` (current and previous rate modulation mode), `hystMode` of blob `Switch` (hysteresis mode, which can be on or off), `cp` and `pcp` of blob `ChambersPaced` (current and previous pacing mode, which can either be atrial, ventricular or dual), and `currentCCI` and `targetCCI` (current and target cardiac cycle intervals).
- Assertion `TargetCCIInv` represents an invariant describing constraint of property edge `targetCCI`, which is described in AD of Fig. 7.3. The AD describes three conditions: when rate modulation is on, the target CCI must be less than the interval calculated from the lower rate limit; when the rate modulation is off, then the target CCI is either calculated from the lower rate limit (if the hysteresis mode is off) or the hysteresis rate limit (if the hysteresis mode is on) .

7.2 Overall Behaviour

BD of `Pacing` is given in Fig. 7.4. It includes two integral extensions to bring in operations from imported packages: one brings into the new context all operations from package `RateSmoothing`, except `Init` and `CalcSmoothedCCI`, and the other all operations from package `AVDelay`, except `GetAVDelay` and `Init`.

The BD (Fig. 7.4) defines several local operations for the blob `ChamberPacing`:

- `New` (Fig. 7.5) to create a new `ChamberPacing` object.

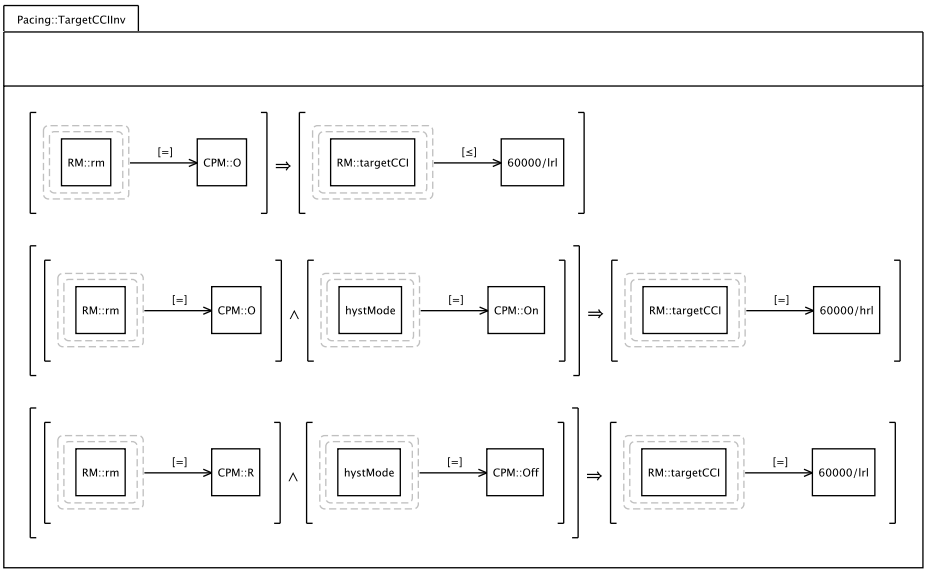


Figure 7.3: VCL assertion diagram of package `TargetCCIInv`

- `SetLastEventTm` (Fig. 7.6(a)) sets the time of the last heart event (either sensed or paced).
- `IncPulseAmplReg` (Fig. 7.7) and `DecPulseAmplReg` (Fig. 7.8) increment and decrement the regulated pulse amplitude.
- `IncPulseAmplUReg` (Fig. 7.9) and `DecPulseAmplUReg` (Fig. 7.10) increment and decrement the unregulated pulse amplitude.
- `IncPulseWidth` (Fig. 7.11) and `DecPulseWidth` (Fig. 7.12) increment and decrement the pulse width.
- Observe operations `GetLastEventTm` (Fig. 7.6(b)) gets time of the heart pacing event, `GetPulseWidth` (Fig. 7.13(a)) gets current pulse's width, and `GetPulseAmpl` (Fig. 7.13(b)) gets the pulse amplitude.

The following global operations are defined in the BD of Fig. 7.4:

- `Init` (Fig. 7.14) is the package's initialisation.
- `SetBradycardiaMode` (Fig. 7.15) sets the bradycardia mode for pacing.
- `UpdateBatteryStatus` (Fig. 7.16) does the required action for pacing when the battery status is updated.
- `IncAPulseAmplReg` (Fig. 7.17(a)) and `DecAPulseAmplReg` (Fig. 7.17(b)) to increment and decrement the regulated atrial pulse amplitude.
- `IncVPulseAmplReg` (Fig. 7.18(a)) and `DecVPulseAmplReg` (Fig. 7.18(b)) to increment and decrement the regulated ventricular pulse amplitude.

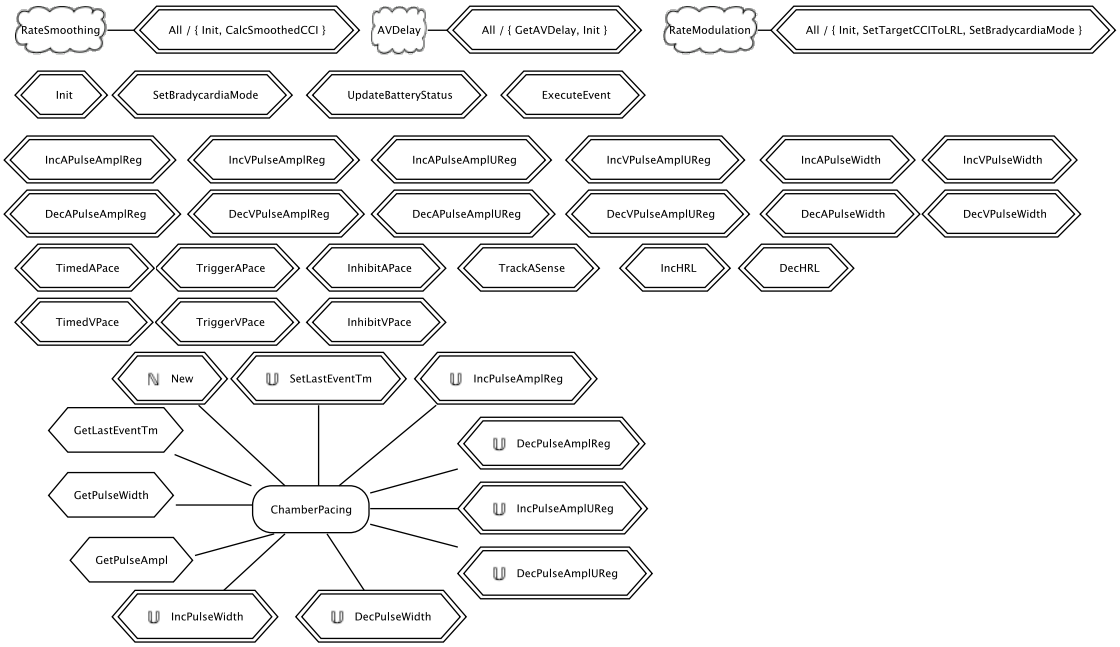


Figure 7.4: Behavioural Diagram of package `Pacing`

- `IncAPulseAmplUReg` (Fig. 7.19(a)) and `DecAPulseAmplUReg` (Fig. 7.19(b)) to increment and decrement the unregulated atrial pulse amplitude.
- `IncVPulseAmplUReg` (Fig. 7.20(a)) and `DecVPulseAmplUReg` (Fig. 7.20(a)) to increment and decrement the unregulated ventricular pulse amplitude.
- `IncAPulseWidth` (Fig. 7.21(a)) and `DecAPulseWidth` (Fig. 7.21(b)) to increment and decrement the atrial pulse width.
- `IncVPulseWidth` (Fig. 7.22(a)) and `DecVPulseWidth` (Fig. 7.22(b)) to increment and decrement the ventricular pulse width.
- `IncHRL` (Fig. 7.23) and `DecHRL` (Fig. 7.24) increment and decrement the hysteresis rate limit.
- `SetTargetCCI` (Fig. 7.28) sets the value of the state property `targetCCI`.
- `ExecuteEvent` (Fig. 7.29) executes some internal pacemaker event in the context of package `Pacing`.
- `TriggerAPace` (Fig. 7.30(a)) and `TriggerVPace` (Fig. 7.30(b)) to trigger a pace on atrial (A) and ventricular (V) chambers.
- `TimedAPace` (Fig. 7.32(a)) and `TimedVPace` (Fig. 7.32(b)) to issue a timed pace (or stimulus) in the atrial (A) and ventricular (V) chambers.

- `InhibitAPace` (Fig. 7.31(a)) and `InhibitVPace` (Fig. 7.31(b)) to inhibit or not issue a pace in the atrial (A) and ventricular (V) chambers; the event is recorded as having happened, but no stimulus is issued.
- `TimedAPace` (Fig. 7.32(a)) and `TimedVPace` (Fig. 7.32(b)) delivers a scheduled stimulus in the atrial (A) or ventricular (V) chambers.
- `TrackASense` (Fig. 7.34) is the pacing action of a tracked sensing in the atrial chamber.

7.3 Local operations of blob `ChamberPacing`

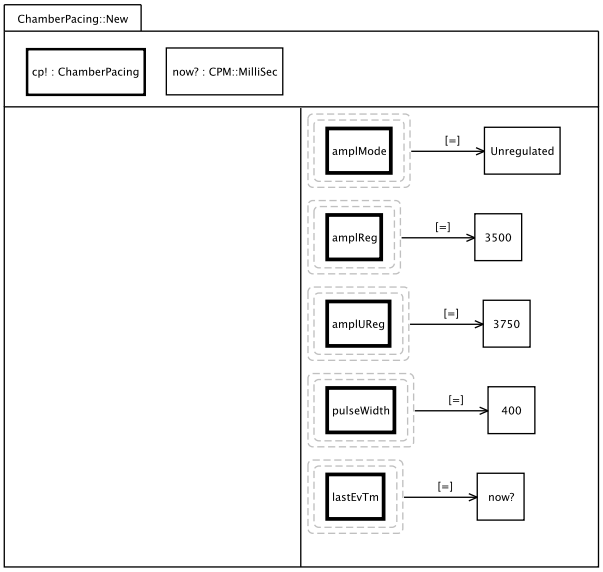
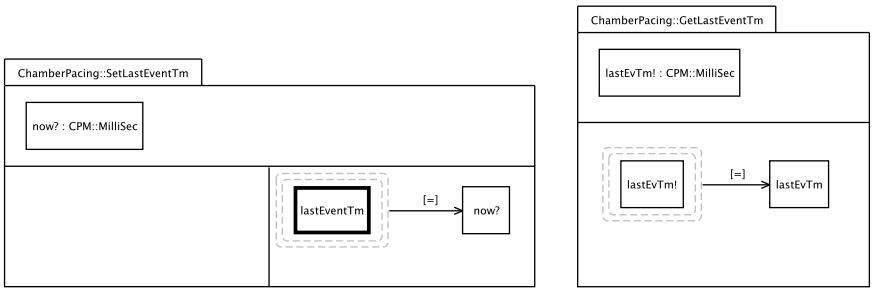


Figure 7.5: Contract diagrams of operation `New` of blob `ChamberPacing`



(a) Operation `SetLastEventTm` (b) Operation `GetLastEventTm`

Figure 7.6: Contract and assertion diagrams of operations `SetLastEventTm` and `GetLastEventTm` of blob `ChamberPacing`

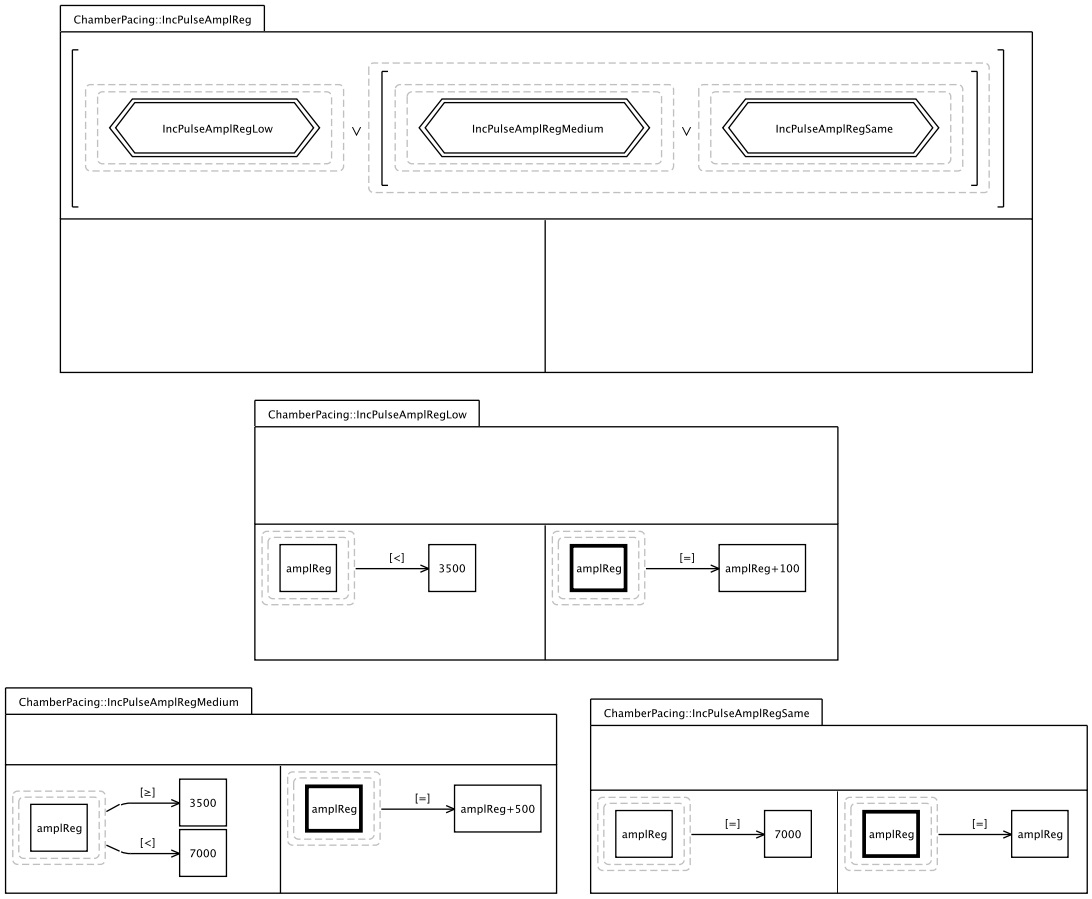


Figure 7.7: Contract diagrams of operation `IncPulseAmplReg` of blob `ChamberPacing`

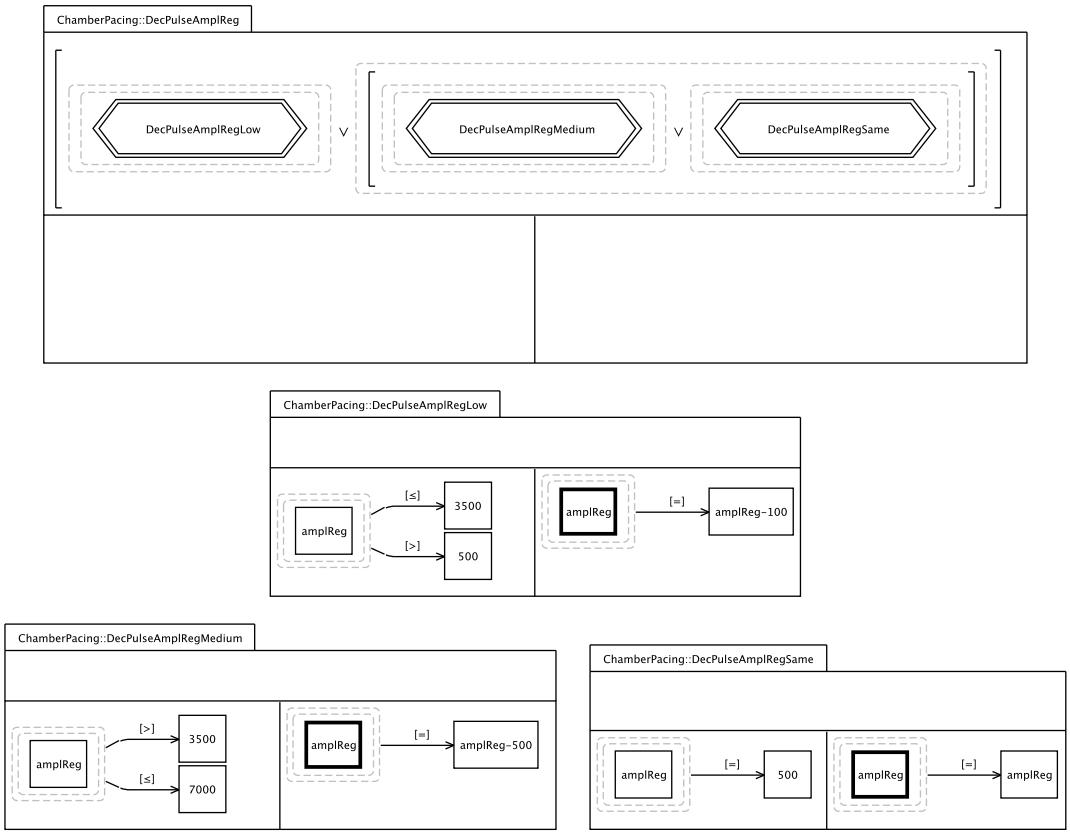


Figure 7.8: Contract diagrams of operation `DecPulseAmplReg` of blob `ChamberPacing`

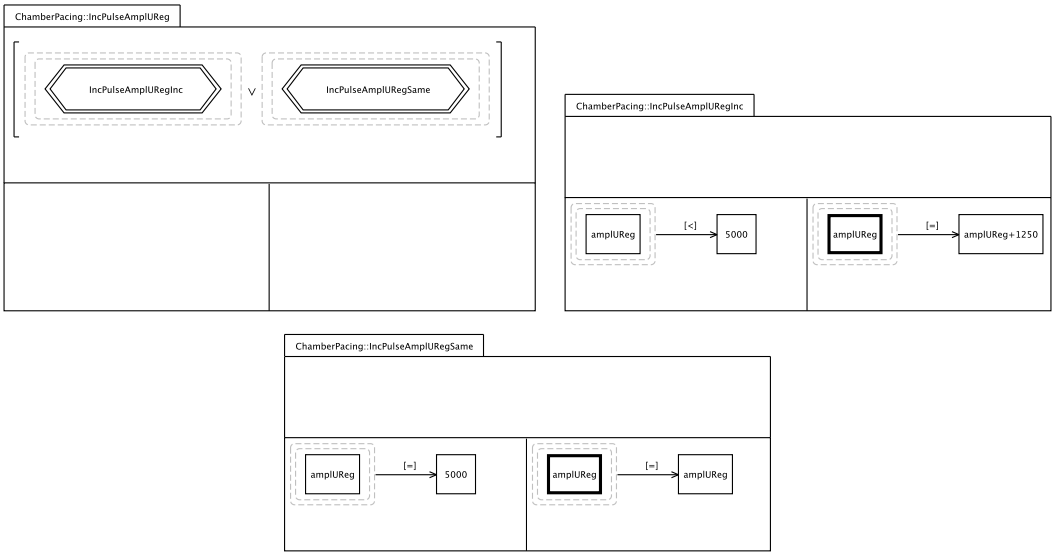


Figure 7.9: Contract diagrams of operation `IncPulseAmplUReg` of blob `ChamberPacing`

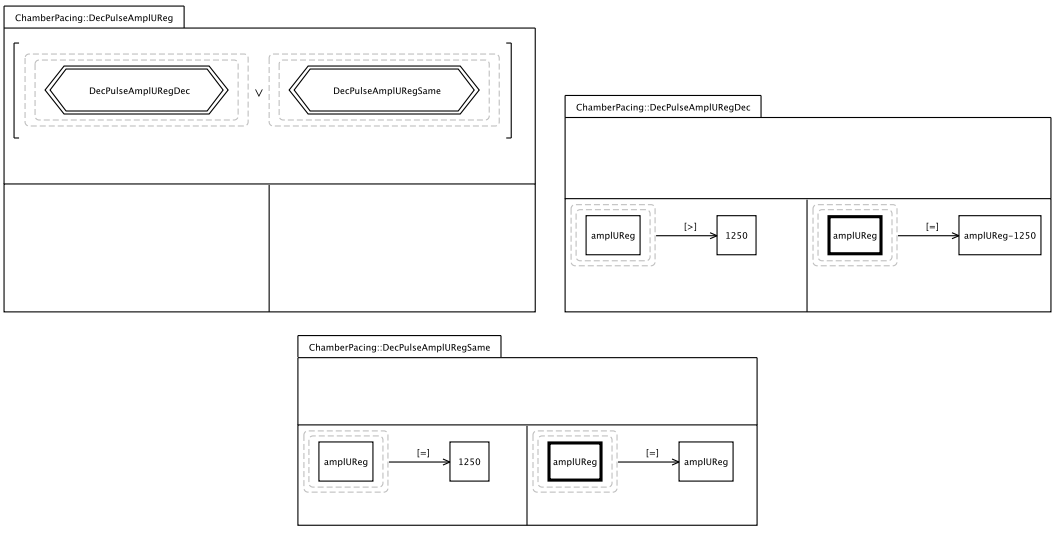


Figure 7.10: Contract diagrams of operation `DecPulseAmplUReg` of blob `ChamberPacing`

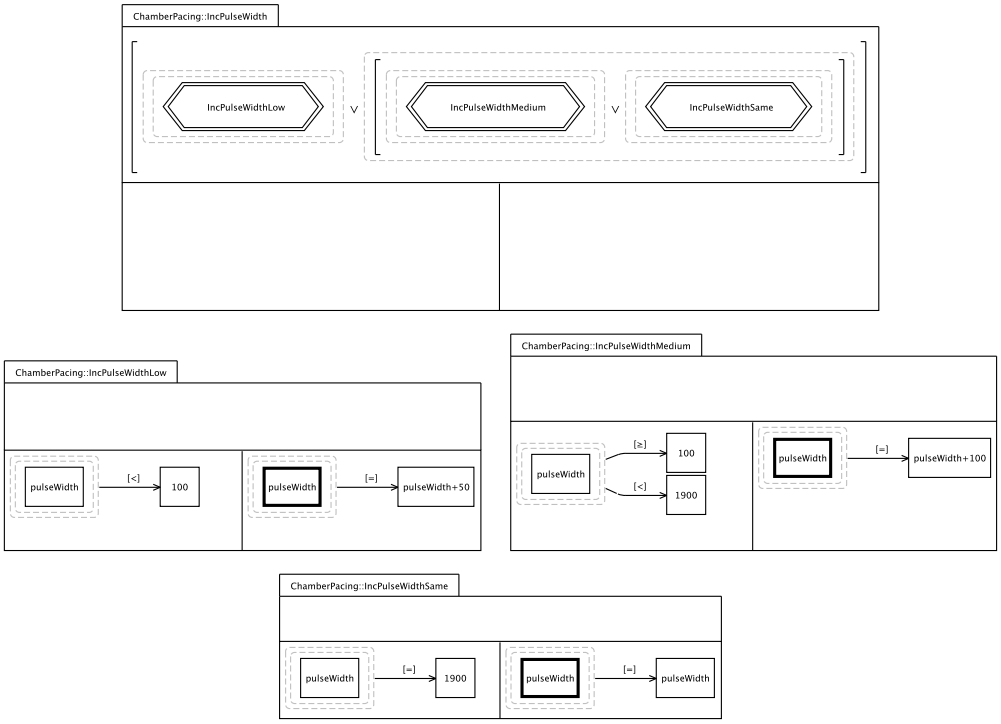


Figure 7.11: Contract diagrams of operation `IncPulseWidth` of blob `ChamberPacing`

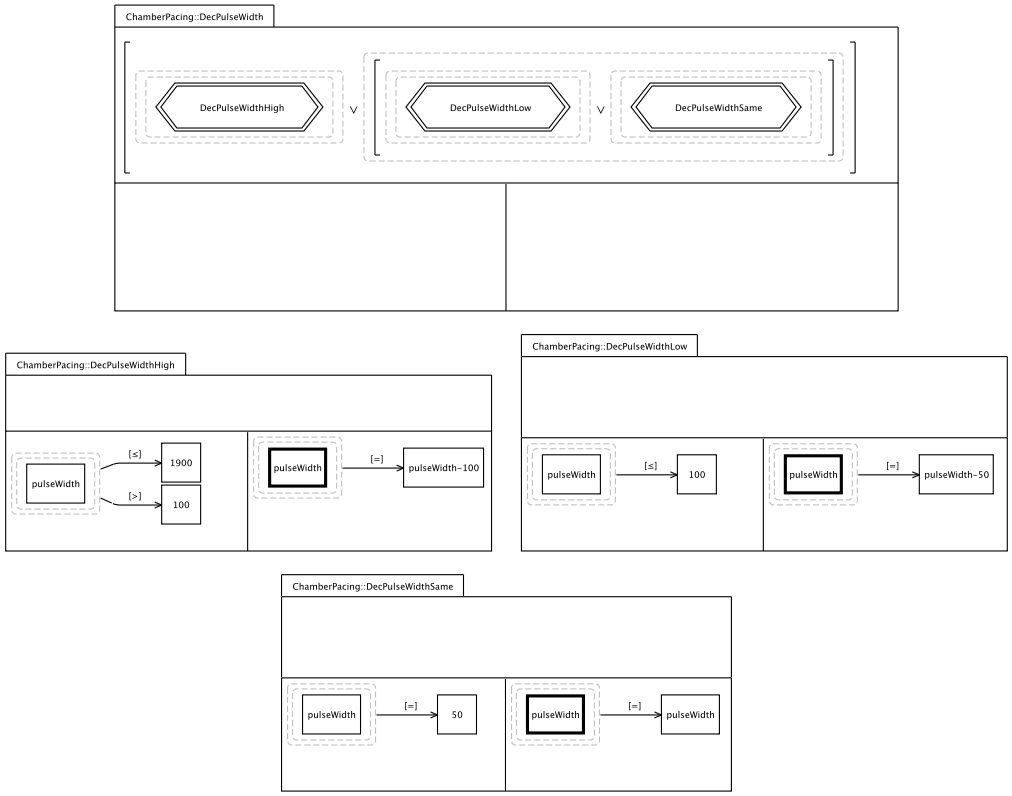
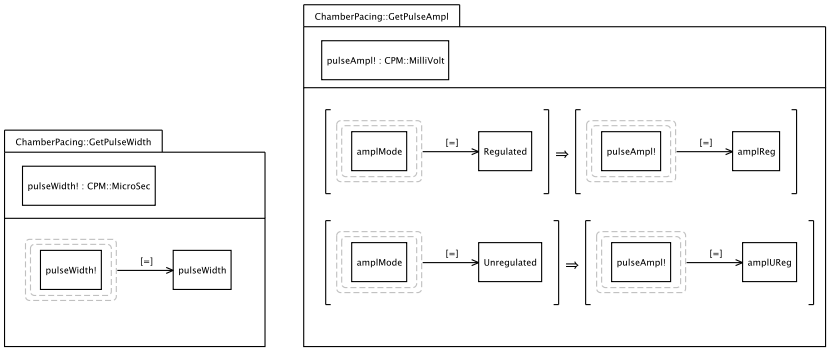


Figure 7.12: Contract diagrams of operation `DecPulseWidth` of blob `ChamberPacing`



(a) Operation `GetPulseWidth` (b) Operation `GetPulseAmpl`

Figure 7.13: Assertion diagrams of observe operations `GetPulseWidth` and `GetPulseAmpl` of blob `ChamberPacing`

7.4 Global Operations

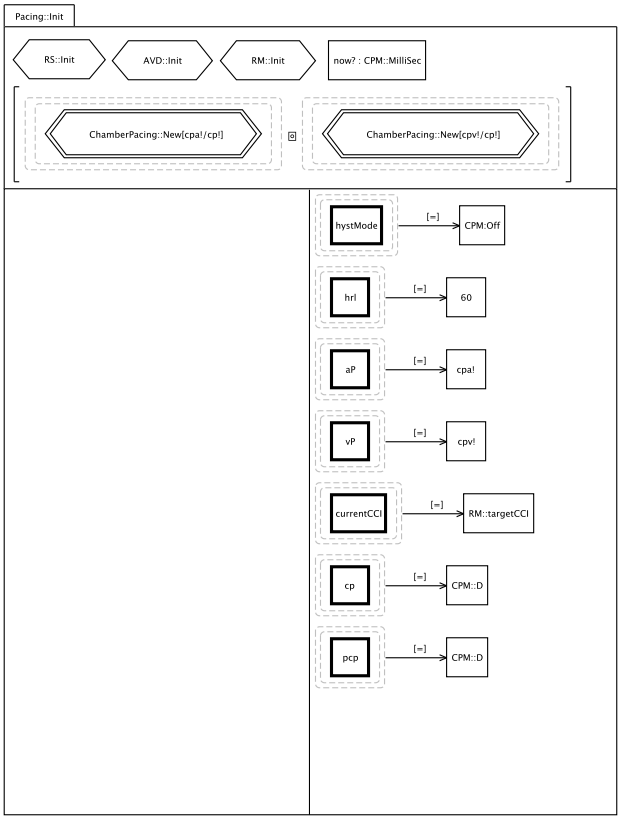


Figure 7.14: Initialisation of package `Pacing`

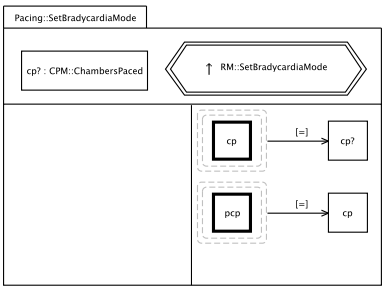


Figure 7.15: Contract Diagrams of global operation `SetBradycardiaMode`

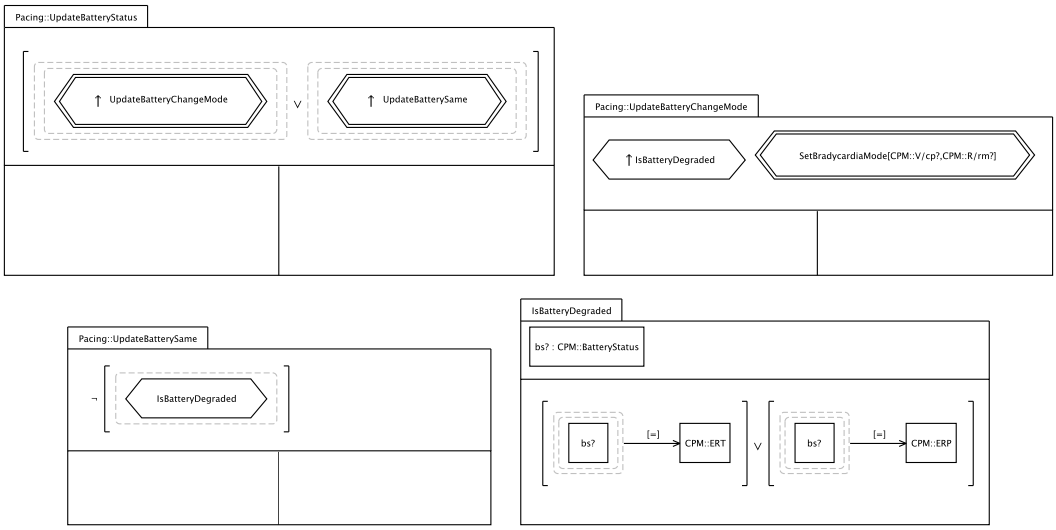
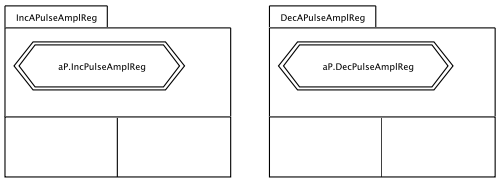


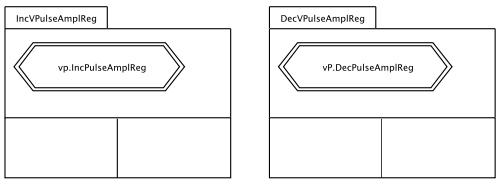
Figure 7.16: Contract Diagrams of global operation `UpdateBatteryStatus`



(a) Operation `IncAPulseAmplReg` (b) Operation `DecAPulseAmplReg`

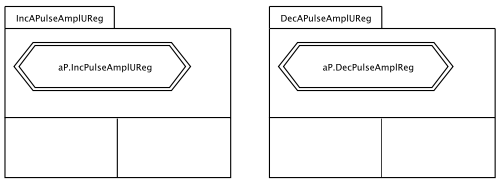
Figure 7.17: Contract Diagrams of global operations `IncAPulseAmplReg` and `DecAPulseAmplReg`

The global operation `UpdateBatteryStatus` (Fig. 7.16) considers two cases: (a) when the bradycardia mode needs to be changed because the battery becomes close to depletion (contract `UpdateBatteryChangeMode`), (b) and when there is no change (contract `UpdateBatterySame`). Contract `UpdateBatteryChangeMode` says that if the battery is close to depletion (assertion `IsBatteryDegraded`) then special mode concerning ERT is applied: operation `SetBradycardiaMode`, which paces on ventricular chamber only without rate modulation. Assertion `IsBatteryDegraded` defines a predicate indicating whether the battery is ERT or ERP. The operation `UpdateBatteryStatus` takes into account assumption A8 (appendix B); we assume that the battery always degrades, and that the only way to take it out of the ventricular mode with rate modulation turned off is by reprogramming the device, which will also require that the battery is changed.



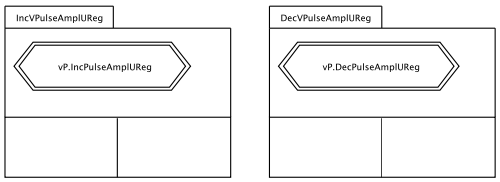
(a) Operation `IncVPulseAmplReg` (b) Operation `DecVPulseAmplReg`

Figure 7.18: Contract diagrams of global operations `IncVPulseAmplReg` and `DecVPulseAmplReg`



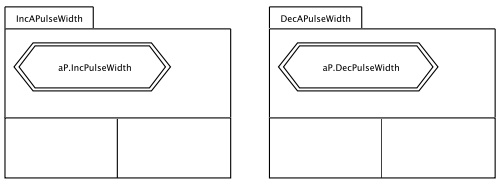
(a) Operation `IncAPulseAmplUReg` (b) Operation `DecAPulseAmplUReg`

Figure 7.19: Contract diagrams of global operations `IncAPulseAmplUReg` and `DecAPulseAmplUReg`



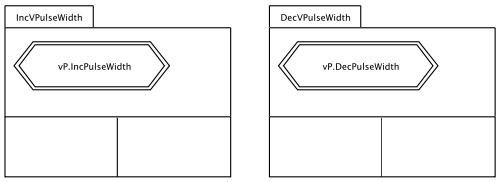
(a) Operation `IncVPulseAmplUReg` (b) Operation `DecVPulseAmplUReg`

Figure 7.20: Contract diagrams of global operations `IncVPulseAmplUReg` and `DecVPulseAmplUReg`



(a) Operation `IncAPulseWidth` (b) Operation `DecAPulseWidth`

Figure 7.21: Contract diagrams of global operations `IncAPulseWidth` and `DecAPulseWidth`



(a) Operation `IncVPulseWidth` (b) Operation `DecVPulseWidth`

Figure 7.22: Contract diagrams of global operations `IncVPulseWidth` and `DecVPulseWitdh`

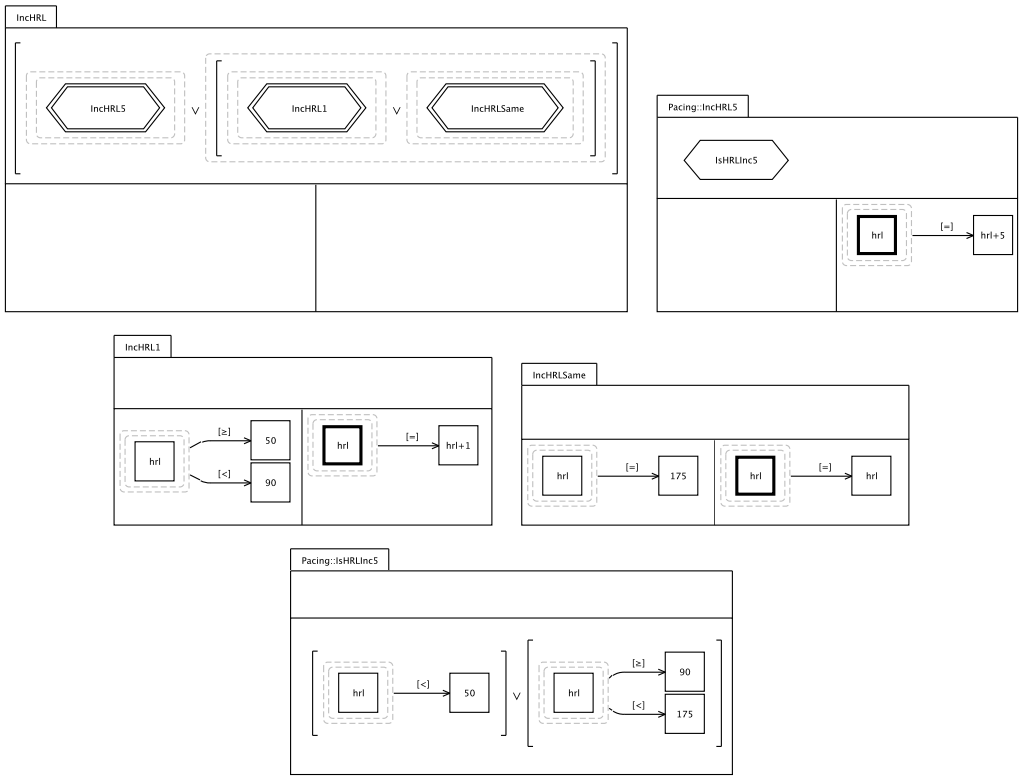


Figure 7.23: Contract and assertion diagrams of global operation `IncHRL`

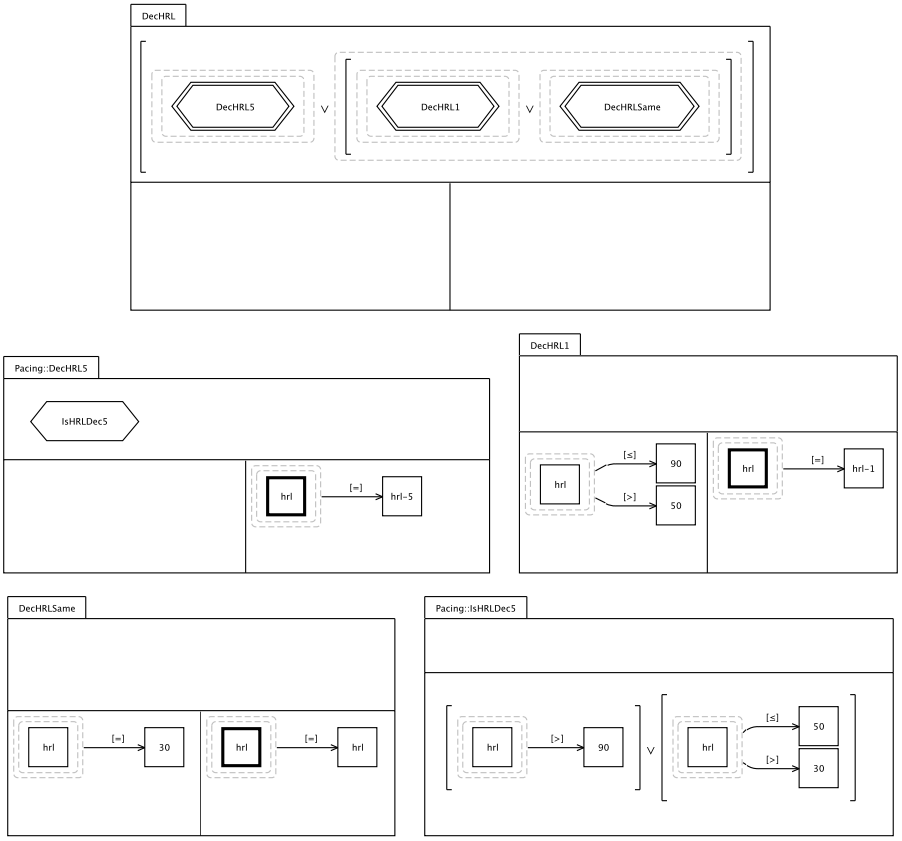


Figure 7.24: Contract and assertion diagrams of global operation `DecHRL`

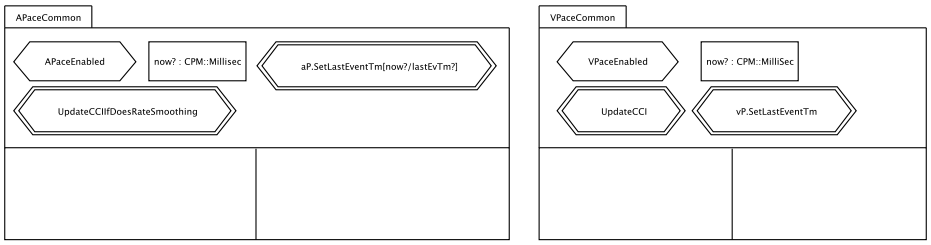


Figure 7.25: Contract diagrams of global operations `APaceCommon` and `VPaceCommon`

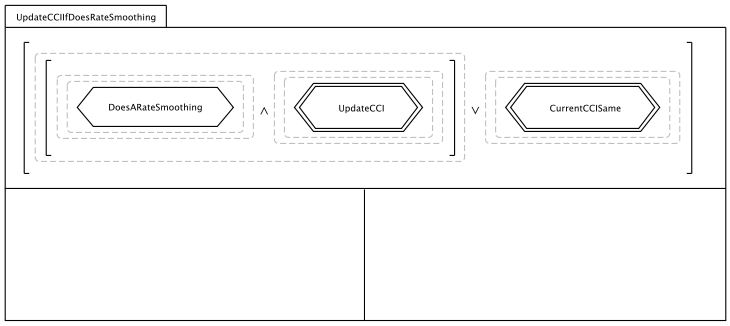


Figure 7.26: Contract diagrams of global operation `UpdateCCIIfDoesRateSmoothing`

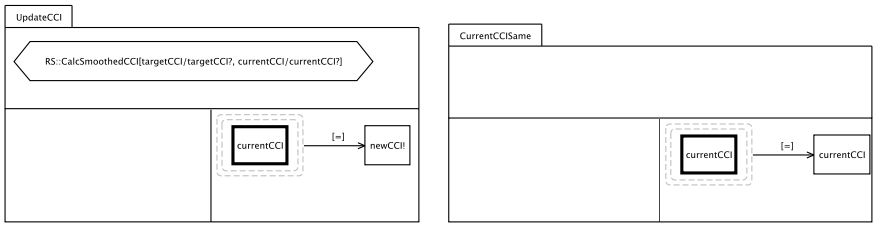


Figure 7.27: Contract diagrams of global operations `UpdateCCI` and `CurrentCCISame`

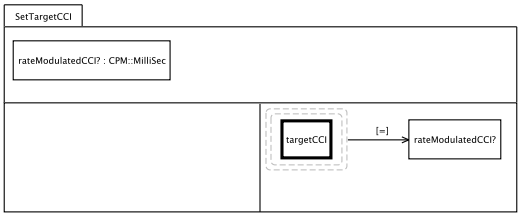


Figure 7.28: Contract diagram of global operation `SetTargetCCI`

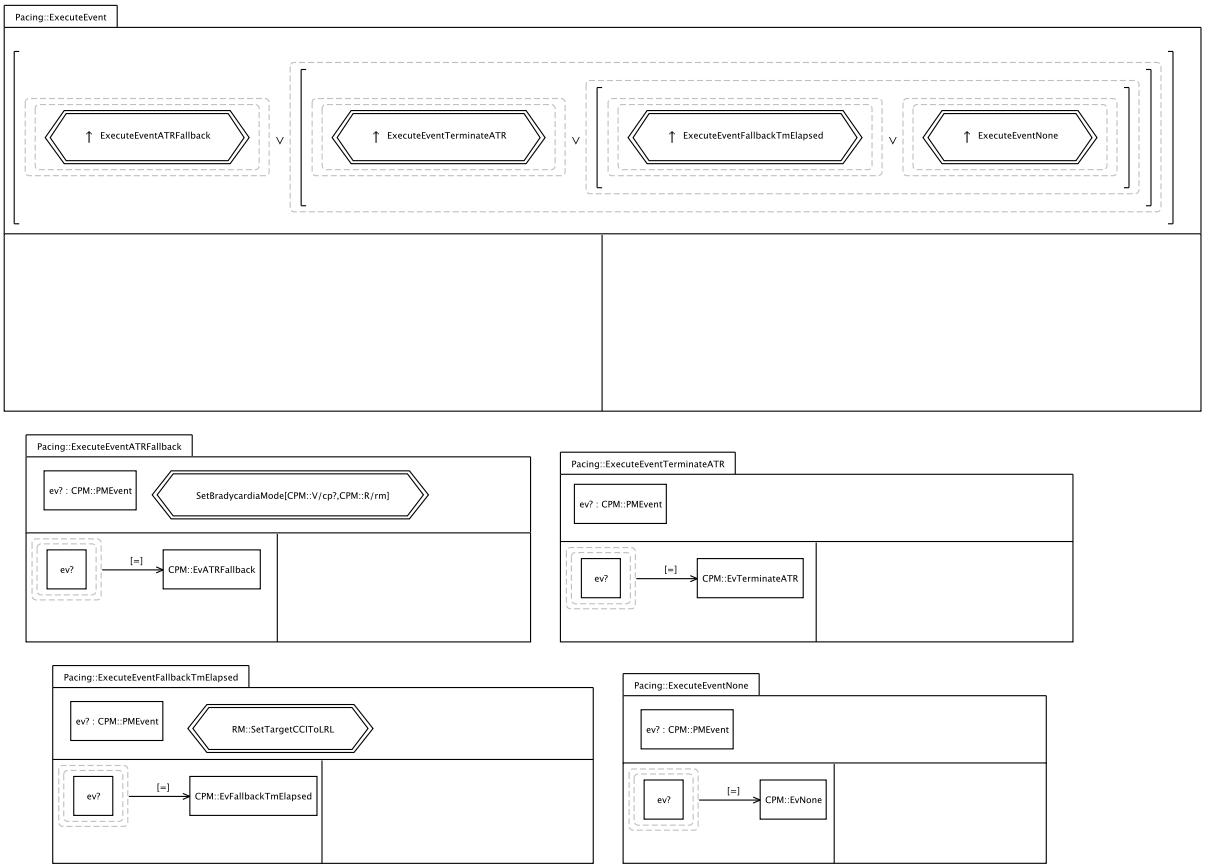
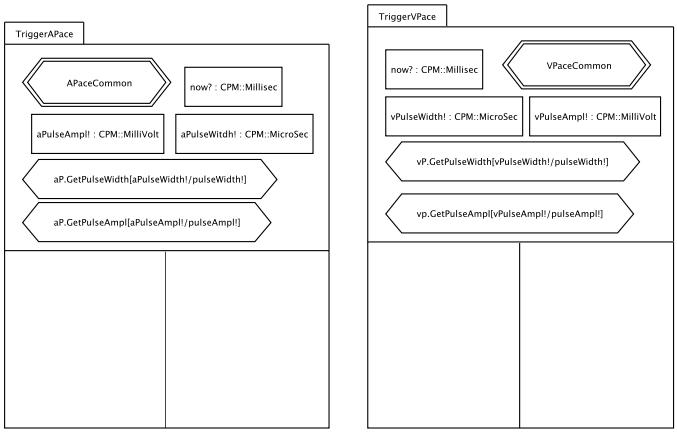
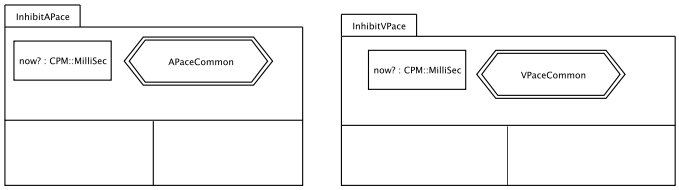


Figure 7.29: Contract diagrams of global operation `ExecuteEvent`



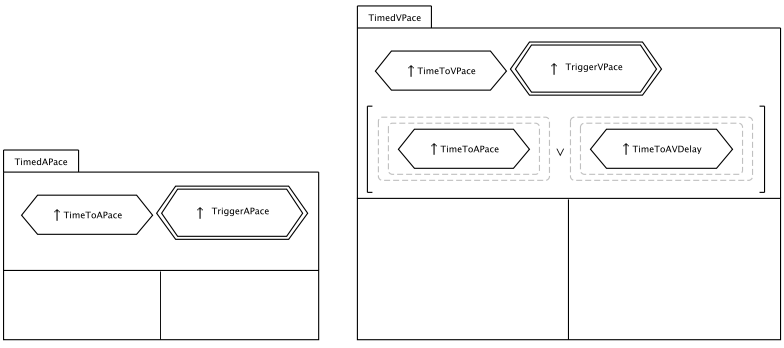
(a) Operation `TriggerAPace` (b) Operation `TriggerVPace`

Figure 7.30: Contract diagrams of global operations `TriggerAPace` and `TriggerVPace`



(a) Operation `InhibitAPace` (b) Operation `InhibitVPace`

Figure 7.31: Contract diagrams of global operations `InhibitAPace` and `InhibitVPace`



(a) Operation `TimedAPace` (b) Operation `TimedVPace`

Figure 7.32: Contract diagrams of global operations `TimedAPace` and `TimedVPace`

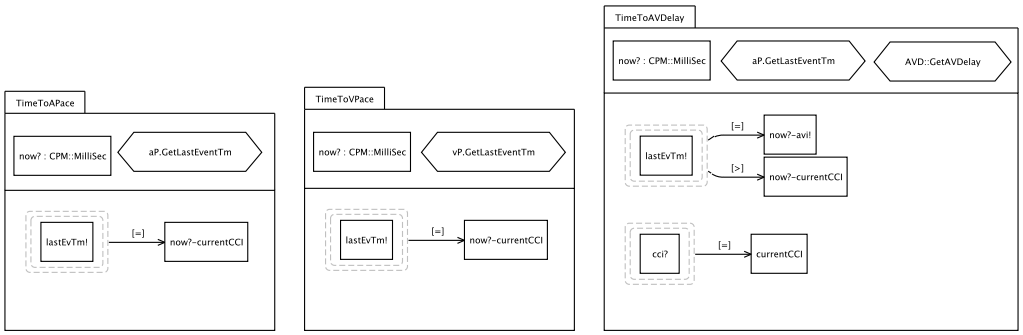


Figure 7.33: Assertion diagrams of global observe operations `TimeToAPace`, `TimeToVPace` and `TimeToAVDelay`

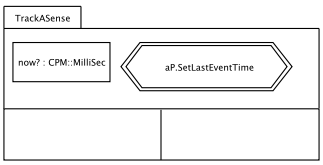


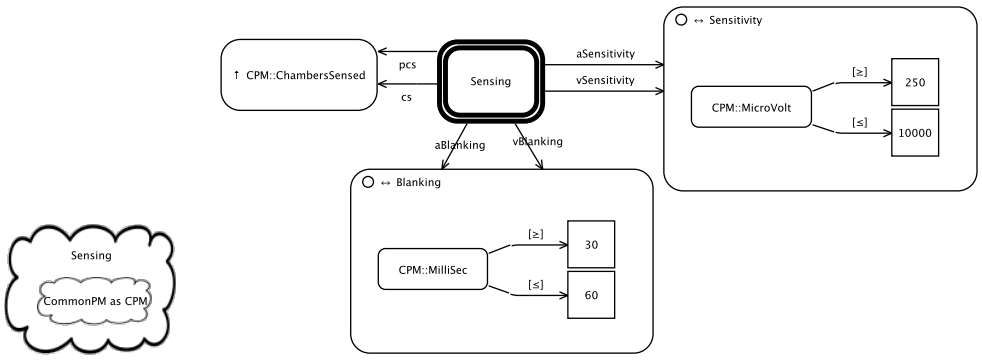
Figure 7.34: Contract diagram of global operations `TrackASense`

Chapter 8

The `Sensing` Package

8.1 State

The pacemaker device detects what is going on inside the heart by measuring the potential difference (voltage) between the two electrodes used for pacing [BSS10]. The package `Sensing` is responsible for this sensing functionality; it senses heart signals through the leads, receiving a sensed amplitude from the heart's sensors. In addition, it manages the *blanking period*: an interval of time during which the pacemaker is unable to sense (or does not respond to sensed events) [MM07].



(a) Package Diagram (b) Structural Diagram

Figure 8.1: VCL Package and structural diagram of package `Sensing`

PD of `Sensing` (Fig. 8.1(a)) package defines `Sensing` as an ensemble package; `Sensing` imports container package `CommonPM`.

Figure 8.1(b) presents the SD of package `Sensing`. It is as follows:

- Derived blob `Sensitivity` captures the allowed values of the sensitivity threshold parameter of the device, expressed in micro-volts (μV); the bounds of this set are taken from the "programmable parameters table" in [Bos07b, p. 34]. This parameter indicates the sensitivity level of sensing.

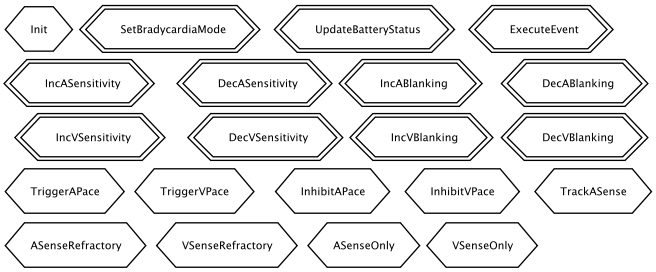


Figure 8.2: Behavioural Diagram of package `Sensing`

- Derived blob `Blanking` captures the allowed values for both atrial and ventricular blanking, as defined by the "Ventricular Blanking" interval the "programmable parameters table" in [Bos07b, p. 34]. This blob is related to requirements assumption A4 (appendix B).
- The Package blob comprises property edges to represent the current and previous heart chamber being sensed (`cs` and `pcs` of set `ChambersSensed`, which can have values atrial, ventricular or dual), the atrial and ventricular blanking periods (`aBlanking` and `vBlanking`), and the atrial and ventricular sensitivity (`aSensitivity` and `vSensitivity`).

8.2 Overall Behaviour

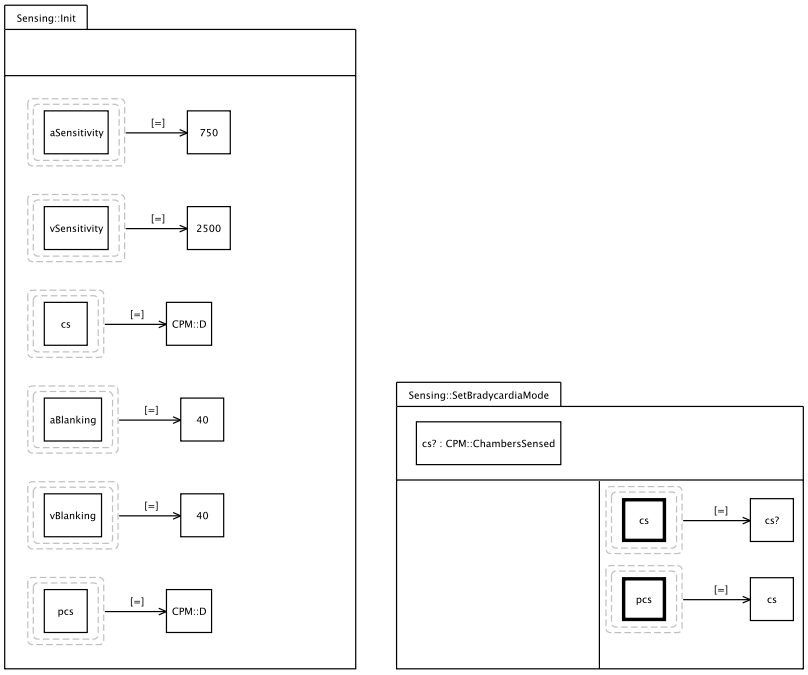
BD of package `Sensing` is given in Fig. 8.2. It introduces the following operations:

- `Init` (Fig. 8.3(a)) is the package's initialisation .
- `SetBradycardiaMode` (Fig. 8.3(b)) is the operation that sets the bradycardia mode for sensing.
- `UpdateBatteryStatus` (Fig. 8.4) does the required action for sensing when the battery status is updated.
- `ExecuteEvent` (Fig. 8.5) executes some internal pacemaker event in the context of package `Sensing`.
- `IncASensitivity` (Fig. 8.6) and `DecASensitivity` (Fig. 8.7) increment and decrement the value of atrial sensitivity.
- `IncVSensitivity` (Fig. 8.8) and `DecVSensitivity` (Fig. 8.9) increment and decrement the value of atrial sensitivity.
- `IncABlanking` (Fig. 8.10) and `DecABlanking` (Fig. 8.11) increment and decrement the value of atrial blanking.
- `IncVBlanking` (Fig. 8.12) and `DecVBlanking` (Fig. 8.13) increment and decrement the value of ventricular blanking.
- `TriggerAPace` (Fig. 8.15(a)) and `TriggerVPace` (Fig. 8.15(b)) correspond to the triggered atrial and ventricular pacing. In the triggered mode of response to sensing, a sense in a

chamber shall trigger an immediate pace in that chamber [Bos07b, p. 17]. These operations are called when this occurs. Here, package `Sensing` describes the effect of this package on these operations (it adds further pre-conditions).

- `InhibitAPace` (Fig. 8.16(a)) and `InhibitVPace` (Fig. 8.16(b)) correspond to the inhibited atrial and ventricular pacing. In the inhibited mode of response to sensing, a sense in a chamber shall inhibit a pending pace in that chamber [Bos07b, p. 17]. These operations are called when this occurs. Here, package `Sensing` describes the effect of this package on these operations (it adds further pre-conditions).
- `TrackASense` (Fig. 8.16(c)) corresponds to the tracked atrial sensing. During tracked pacing, an atrial sense shall cause a tracked ventricular pace after a programmed AV delay unless a ventricular sense was detected beforehand [Bos07b, p. 17].
- `ASenseRefractory` (Fig. 8.17(a)) and `VSenseRefractory` (Fig. 8.17(b)) correspond to an atrial and ventricular sense during the atrial or ventricular refractory period.
- `ASenseOnly` (Fig. 8.14(a)) and `VSenseOnly` (Fig. 8.14(b)) correspond to a normal atrial and ventricular sense.

8.3 Operations



(a) Initialisation (b) Operation `SetBradycardiaMode`

Figure 8.3: Package `Sensing`: Assertion diagram of Initialisation and contract diagram of operation `SetBradycardiaMode`

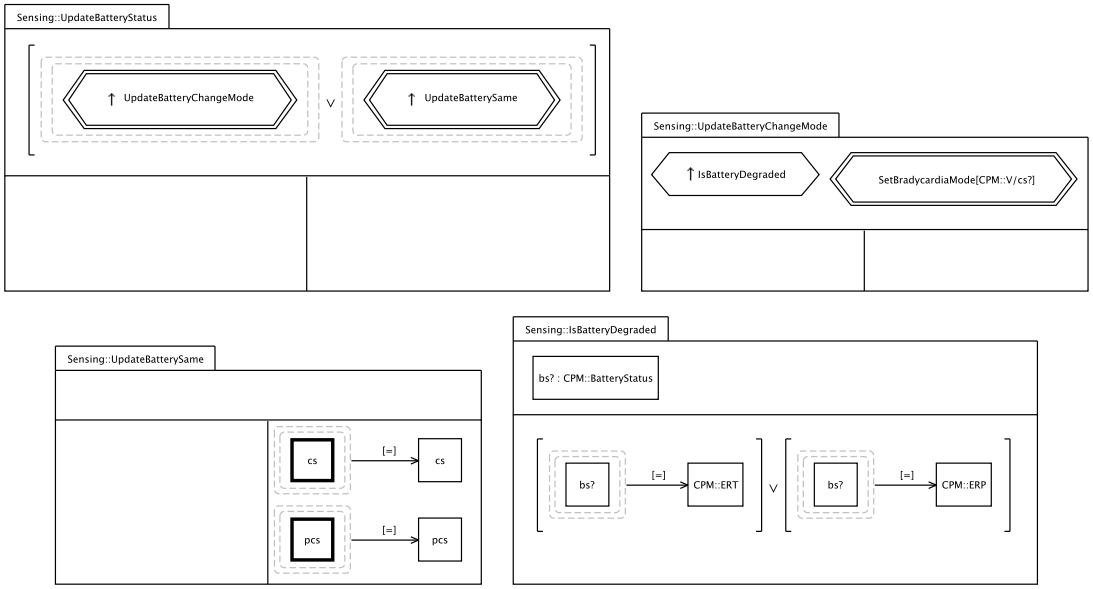


Figure 8.4: Contract Diagrams of global operation `UpdateBatteryStatus`

AD of initialisation (Fig. 8.3(a)) sets the properties of the package blob `Sensing` to the nominal values taken from the "programmable parameters table" [Bos07b, p. 34]. The only exception is the initial value of atrial blanking (not defined in [Bos07b, p. 34]); this is assumed to be as the ventricular blanking according to requirements assumption A4 (appendix B).

CD of operation `SetBradycardiaMode` (Fig. 8.3(b)) sets the property `cs` to the programmed value given as input (`cs?`).

The global operation `UpdateBatteryStatus` (Fig. 8.4) receives the new battery status as an input and considers two cases: (a) when the bradycardia mode needs to be changed because the battery becomes close to depletion (contract `UpdateBatteryChangeMode`), (b) and when there is no change (contract `UpdateBatterySame`). Contract `UpdateBatteryChangeMode` says that if the battery is close to depletion (assertion `IsBatteryDegraded`) then special mode concerning ERT is applied: only the ventricular chamber is paced and the rate modulation is turned off (postcondition). Assertion `IsBatteryDegraded` defines a predicate indicating whether the battery is ERT or ERP. The operation `UpdateBatteryStatus` takes into account assumption A8 (appendix B); we assume that the battery always degrades, and that the only way to take it out of the ventricular mode with rate modulation turned off is by reprogramming the device, which will also require that the battery is changed.

Figures 8.14(a) and 8.14(b) present the ADs of operations `ASenseOnly` and `VSenseOnly`, which are responsible for sensing in the atrial and ventricular chambers. Both operations receive an amplitude in millivolts (`aSAAmpl?` and `aSVAmpl?`), the current time in milliseconds (`now?`) and the time of the last atrial or ventricular event (`lastAEvTm?` and `lastVEvTm?`). There is a sense if the received amplitude is higher than the atrial or ventricular sensitivity (a sense signal has been detected), if the blanking period has elapsed, and if the mode allows sensing in the atrial or ventricular chambers. These operations make use of assumption A10 (appendix B), which says that sense signals are amplitudes.

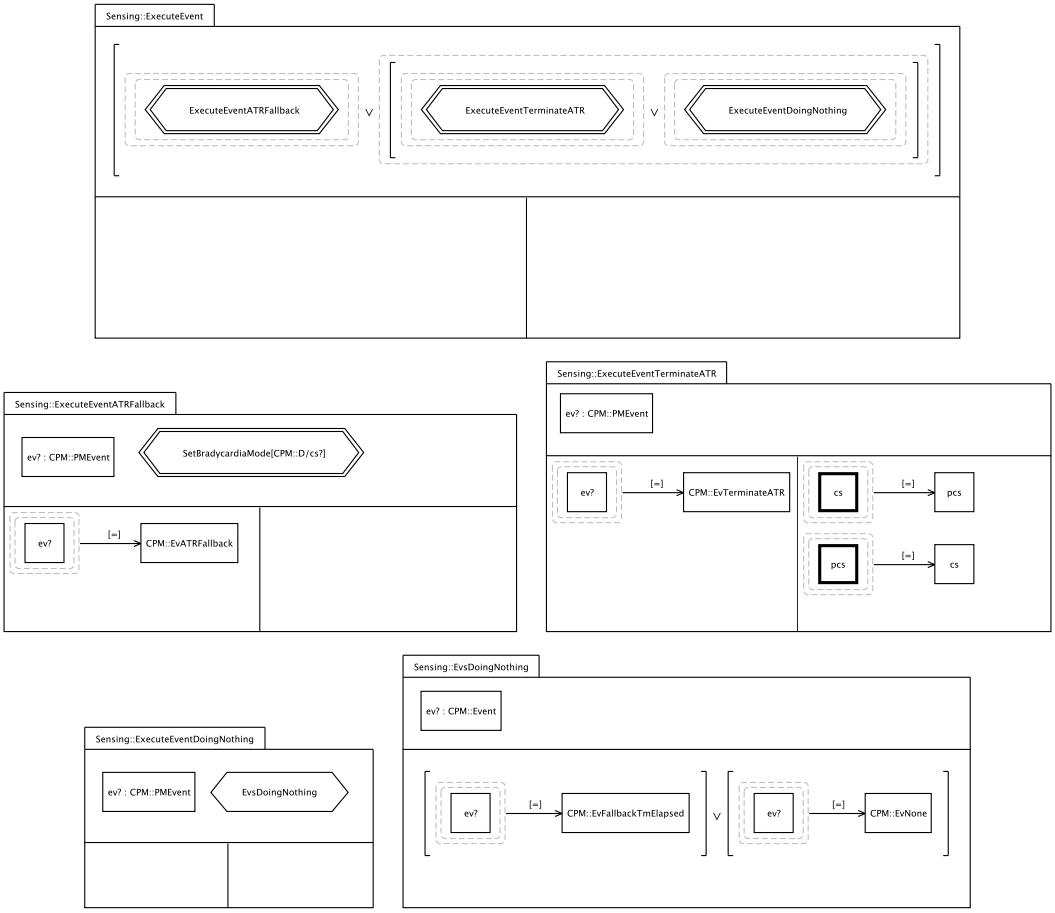


Figure 8.5: Contract Diagrams of global operation `ExecuteEvent`

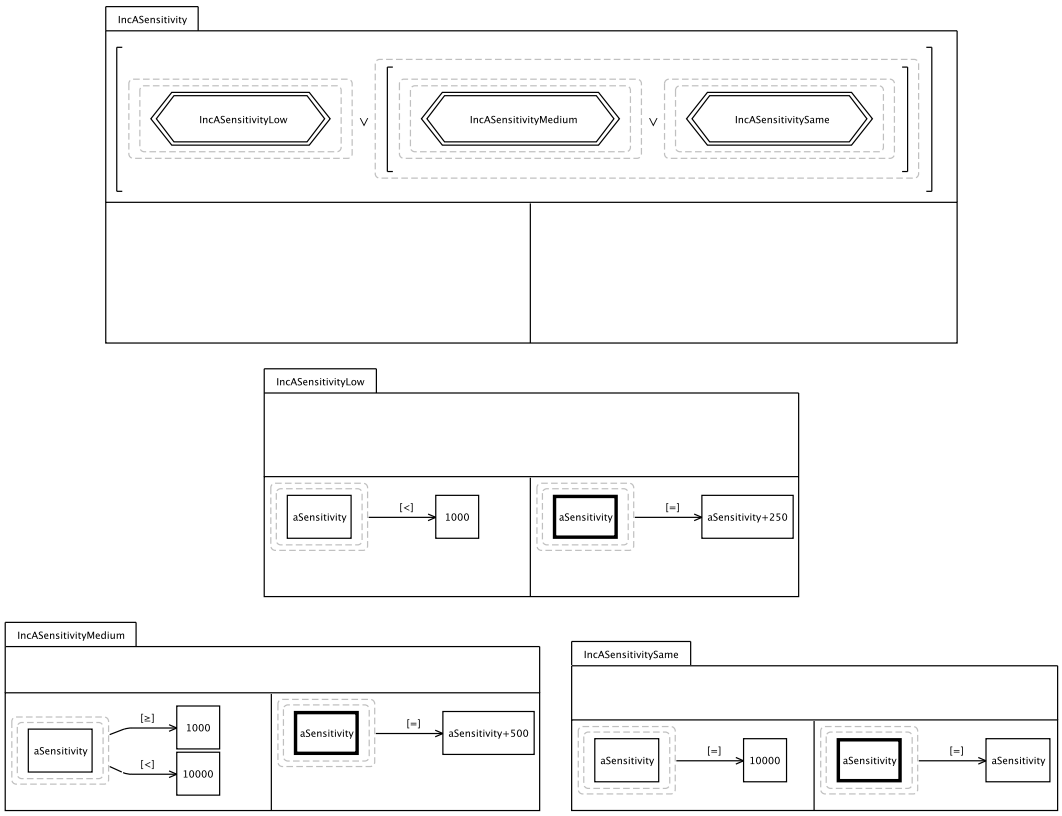


Figure 8.6: Contract Diagrams of global operation `IncASensitivity`

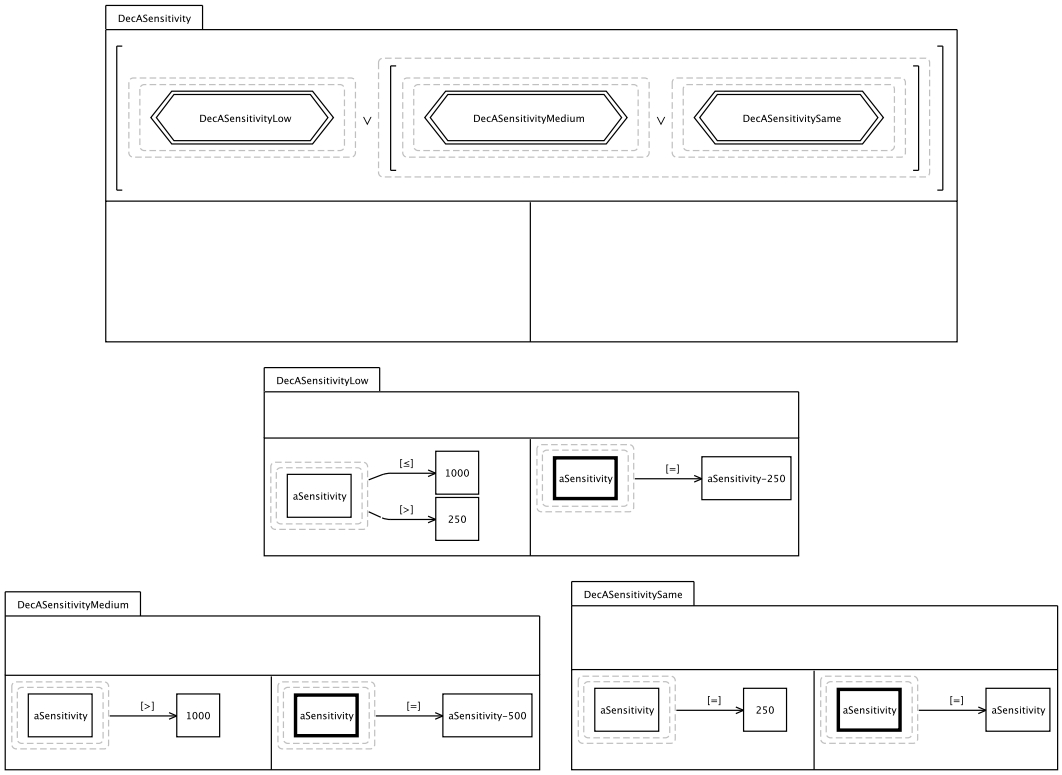


Figure 8.7: Contract Diagrams of global operation `DecASensitivity`

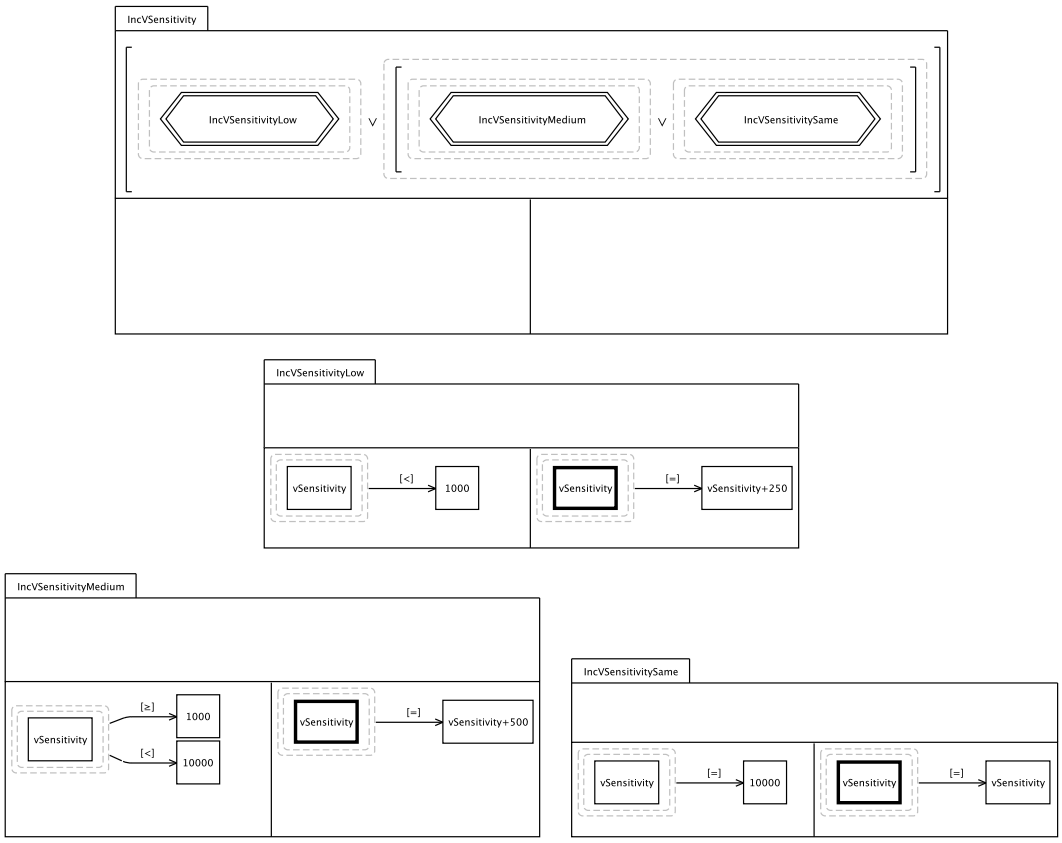


Figure 8.8: Contract Diagrams of global operation `IncVSensitivity`

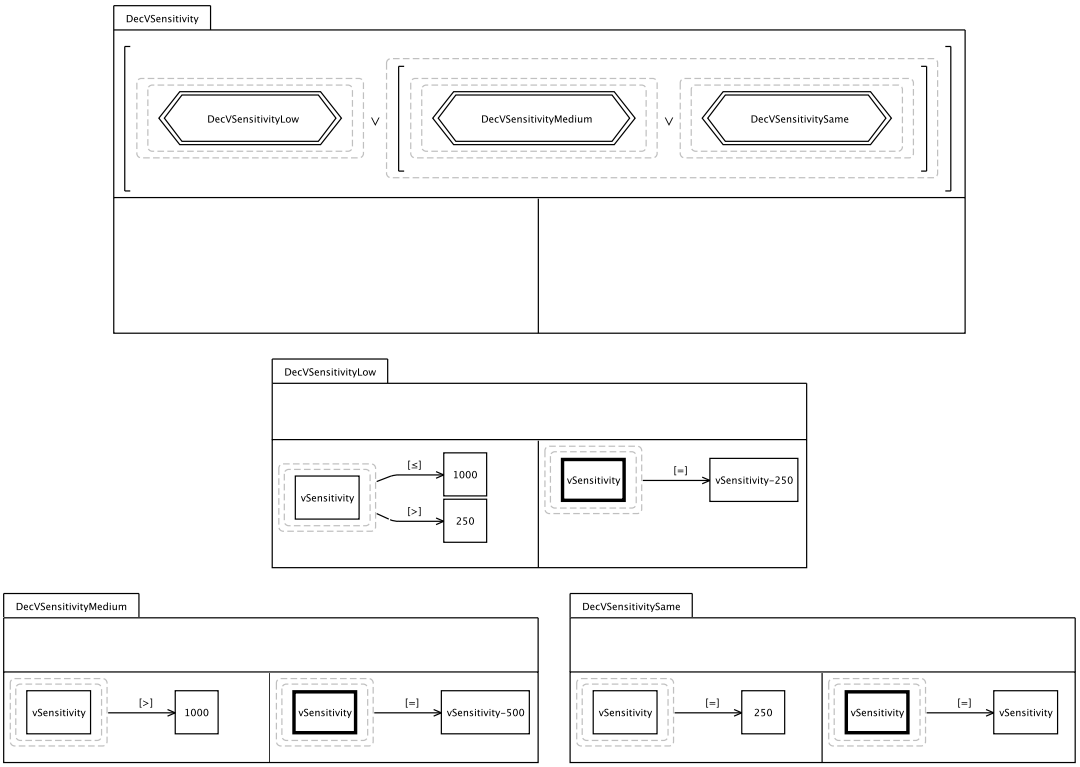


Figure 8.9: Contract Diagrams of global operation `DecVSensitivity`

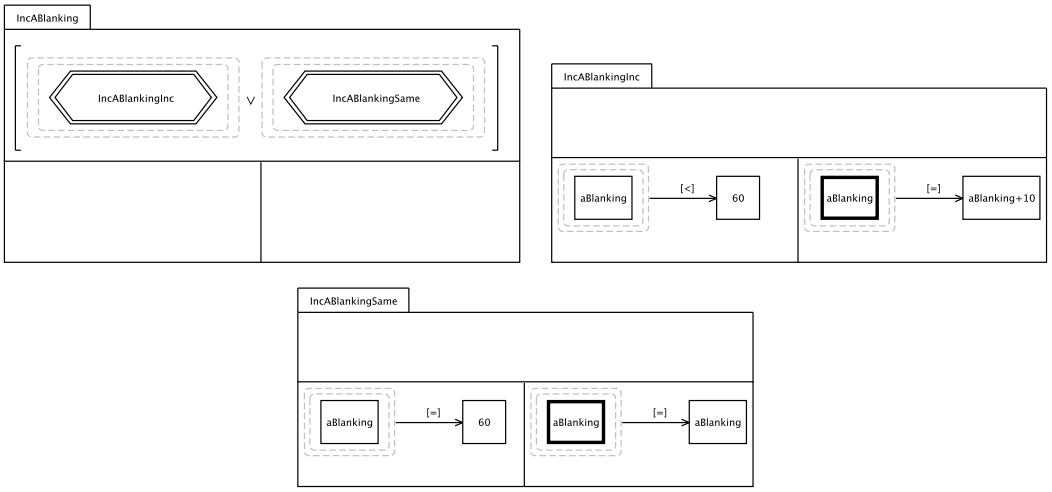


Figure 8.10: Contract Diagrams of global operation `IncABlanking`

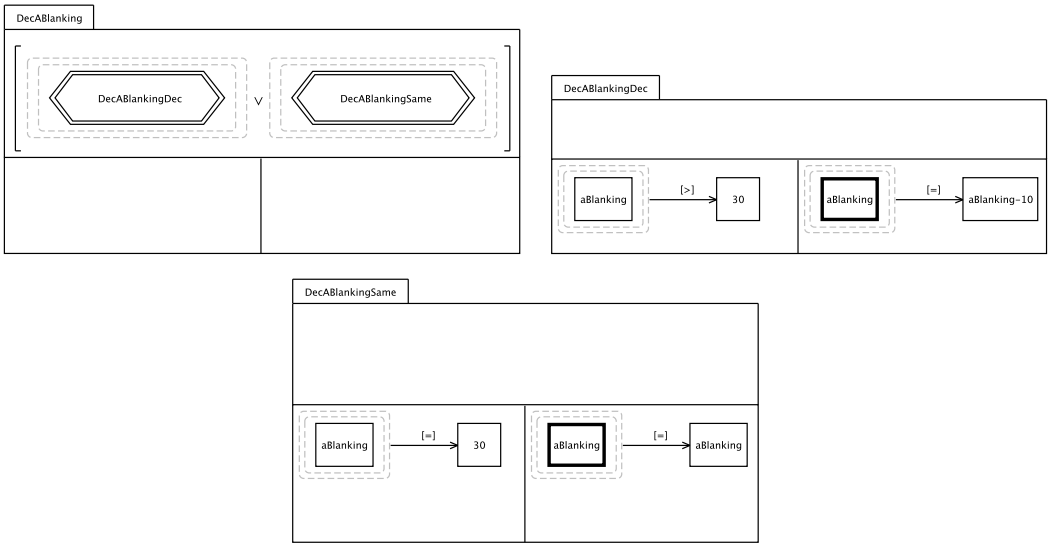


Figure 8.11: Contract Diagrams of global operation `DecABlanking`

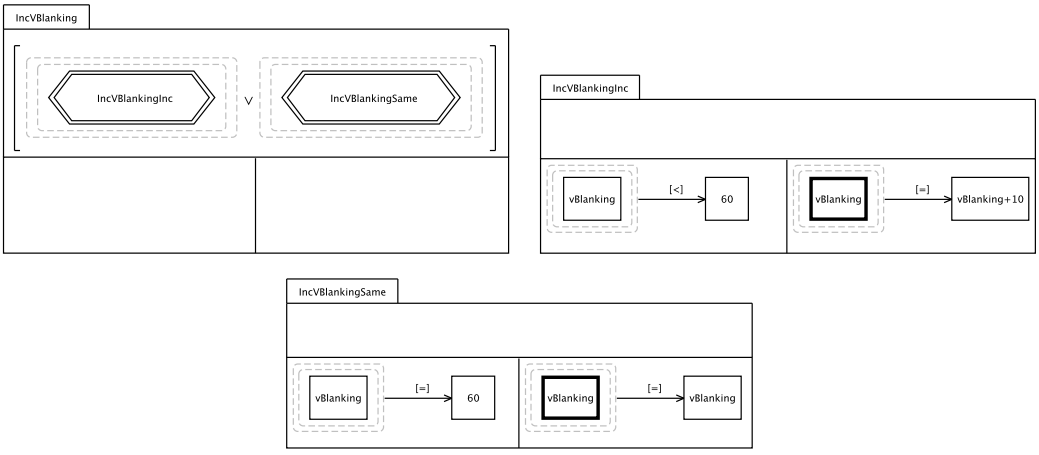


Figure 8.12: Contract Diagrams of global operation `IncVBlanking`

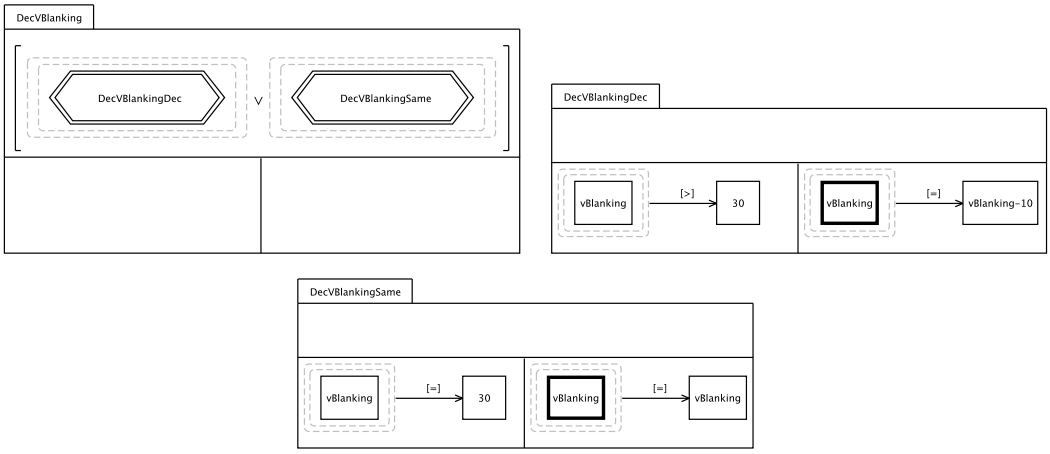
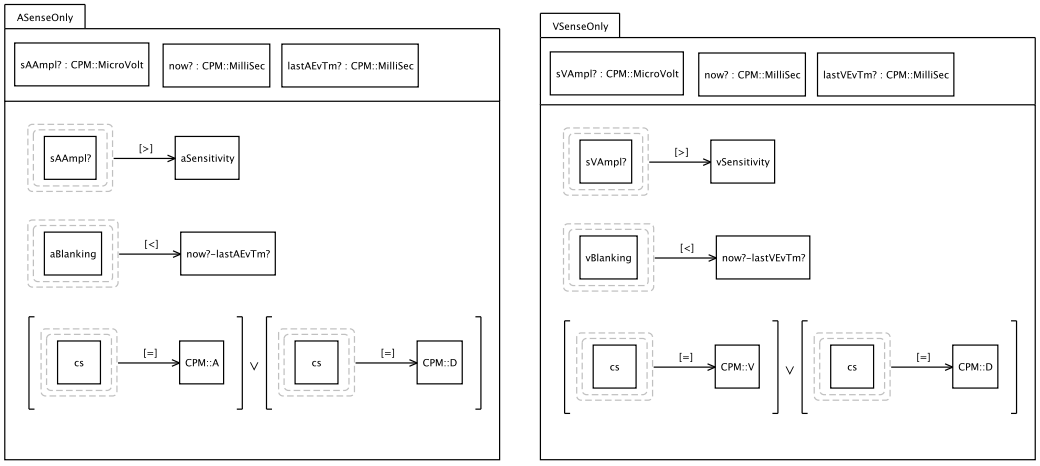
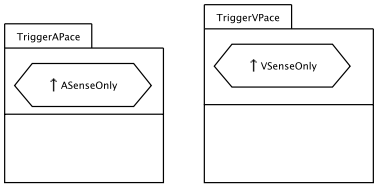


Figure 8.13: Contract Diagrams of global operation `DecVBlanking`



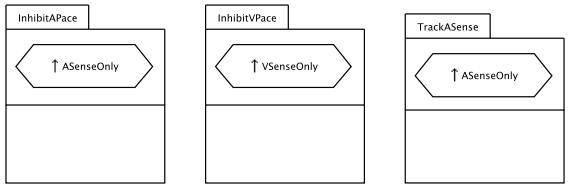
(a) `ASenseOnly` (b) `VSenseOnly`

Figure 8.14: Assertion Diagram of predicates `ASenseOnly` and `VSenseOnly`



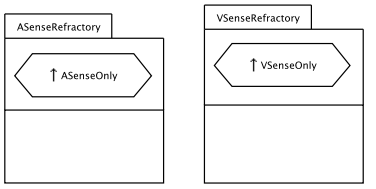
(a) `TriggerAPace` (b) `TriggerVPace`

Figure 8.15: Assertion Diagrams of operations `TriggerAPace` and `TriggerVPace`



(a) `InhibitAPace` (b) `InhibitVPace` (c) `TrackASense`

Figure 8.16: Assertion Diagrams of operations `InhibitAPace`, `InhibitVPace` and `TrackASense`



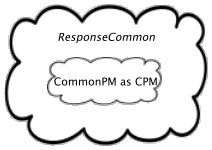
(a) `ASenseRefractory` (b) `VSenseRefractory`

Figure 8.17: Assertion Diagrams of operations `ASenseRefractory` and `VSenseRefractory`

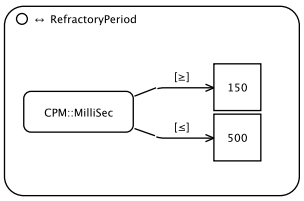
Chapter 9

The ResponseCommon Package

The `ResponseCommon` package introduces sets common across the packages that deal with the response-to-sensing functionality of the pacemaker. This is to be used by packages `AtrialResponse` and `VentricularResponse`.



(a) Package Diagram



(b) Structural Diagram

Figure 9.1: VCL Package and structural diagram of package `ResponseCommon`

`ResponseCommon`'s PD (Fig. 9.1(a)) defines `ResponseCommon` as a container package. `ResponseCommon` imports package `CommonPM` in order to access the set `MilliSec`.

The package's SD (Fig. 9.1(b)) defines derived blob `RefractoryPeriod` from the blob `MilliSec` of package `CommonPm` according to the "programmable parameters table" of [Bos07b, p. 34]: an interval between 150 and 500 milliseconds. The *refractory period* refers to the period after either a sensed or paced beat in which sensing is disabled [MM07].

Chapter 10

The `AtrialResponse` Package

10.1 State

Package `AtrialResponse` is responsible for the response-to-sensing functionality of the pacemaker in what concerns the atrial chamber.

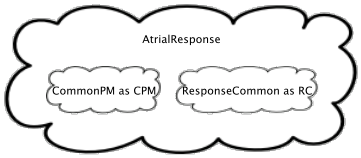


Figure 10.1: VCL Package diagram of package `AtrialResponse`

`AtrialResponse`'s PD (Fig. 10.1) defines `AtrialResponse` as an ensemble package. It imports packages `CommonPM` and `ResponseCommon`.

SD of package `AtrialResponse` (Fig. 10.2) introduces the following derived blobs:

- `UpperRateLimit` represents the allowed values of the *upper rate limit* (URL), expressed in ppm, and defined according to the "programmable parameters table" of [Bos07b, p. 34]. URL represents the maximum rate at which the paced ventricular rate will track atrial events [Bos07b, p. 29]. It represents the speed limit to control the response of the ventricular channel to sensed atrial activity [BSS10].

- `ATRFallbackTime` represents allowed values of *atrial tachycardia response (ATR) fallback time*, expressed in milliseconds (ms) and defined according to the "programmable parameters table" of [Bos07b, p. 34] (in the table the unit is expressed in minutes, which has been converted here to ms). A tachycardia is a rapid heart rate; it typically refers to a heart rate that exceeds the normal range for a resting heart rate (heart rate in an inactive or sleeping individual). ATR prevents long term pacing of a patient at unacceptably high rates during atrial tachycardia [Bos07b, p. 31]. When this happens, if the pacemaker is in the mode ATR, the Pacemaker should drop the current rate to the lower rate limit (LRL); the fallback time is the total time required to drop the rate to the LRL.

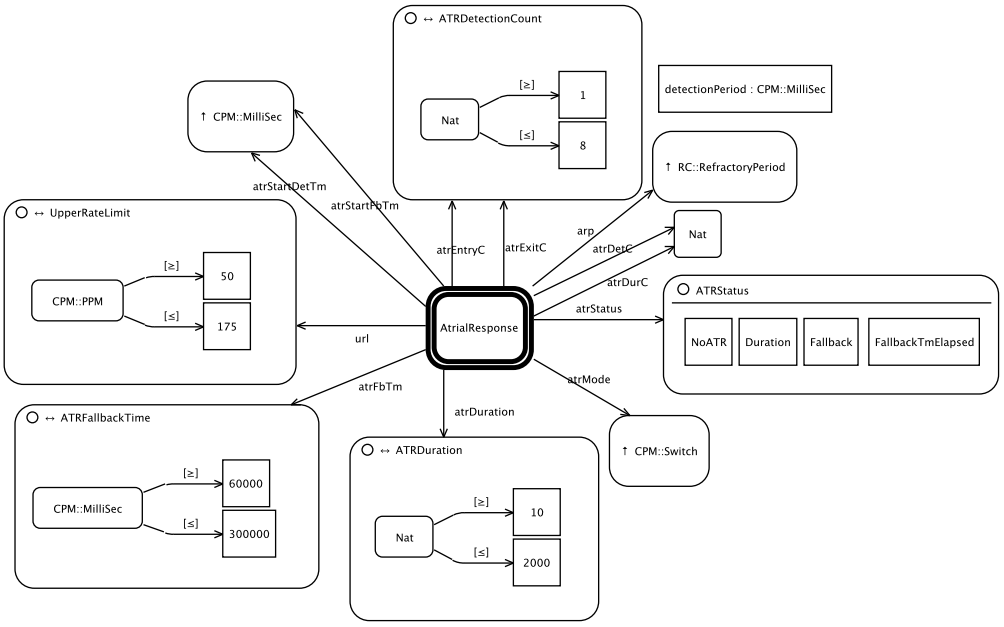


Figure 10.2: VCL structural diagram of package `AtrialResponse`

- `ATRDuration` represents the allowed values of ATR duration; it is a set derived from the natural numbers, representing the number of cardiac cycles, and defined according to the "programmable parameters table" of [Bos07b, p. 34]. ATR duration is the state the Pacemaker device gets into after an atrial tachycardia is detected; it describes a delay (in terms of number of cardiac cycles) before the device enters ATR fallback [Bos07b, p. 31].

- `ATRDetectionCount` represents the allowed values for the ATR entry and exit detection counts, which are used to detect the beginning and the end of atrial tachycardia. This is defined according to requirements assumptions A11 and A12 (appendix B).

The primary blob `ATRStatus` contains values representing the possible status of ATR. The values are: `NoATR` (not responding to atrial tachycardia), `Duration` (ATR is in the *duration* state [Bos07b, p. 31]), `Fallback` (ATR is in the *fallback* state [Bos07b, p. 32]), `FallbackTmElapsed` (the fallback time has elpased [Bos07b, p. 32]).

SD of Fig. 10.2 introduces package blob `AtrialResponse`, which has the property edges:

- `atrStatus` of blob `ATRStatus` indicates current status of ATR.

- `url`, `arp`, `atrMode`, `atrDuration`, `atrFbTm`, `atrEntryC` and `atrExitC` represent the current value for the programmable features URL [Bos07b, p. 29], atrial refractory period [Bos07b, p. 30], ATR mode (can be `On` or `Off`) [Bos07b, p. 31], ATR duration [Bos07b, p. 31], ATR fallback time [Bos07b, p. 32], and number of cardiac cycles required to detect the beginning and end of atrial tachycardia (assumptions A11 and A12, appendix B). The atrial refractory period is the programmed time interval following an atrial event during which time atrial events shall neither inhibit nor trigger pacing [Bos07b, p. 30]; available in single chamber atrial models only.

- `atrDetC`, `atrDurC`, `atrStartDetTm`, `atrStartFbTm` are used to ATR. `atrDetC` counts the current number of cycles detected for beginning or ending ATR. `atrDurC` counts the current number of cycles since duration started. `atrStartDetTm` indicates the starting time of ATR detection (either for beginning or ending ATR). `atrStartFbTm` indicates time when the device entered the fallback state.

10.2 Overall Behaviour

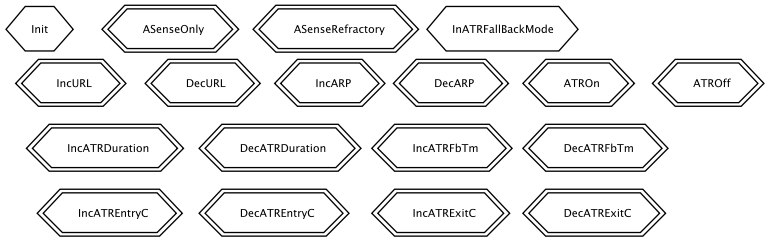


Figure 10.3: Behavioural Diagram of package `AtrialResponseResponse`

BD of package `AtrialResponse` is given in Fig. 10.3. It introduces the following behavioural units:

- `Init` is the package's initialisation (Fig. 10.4).
- `InATRFallBackMode` (Fig. 10.5) indicates whether the device is in ATR fallback mode.
- `ASenseOnly` (Fig. 10.13) and `ASenseRefractory` (Fig. 10.14) do, respectively, the normal atrial sensing and the the atrial sensing during the refractory period. These operations use the operation `PerformATR`, which is responsible for managing ATR (Fig. 10.6); this involves a complex processing according to what is described in [Bos07b, ps. 31–32]. The diagrams from figures 10.6 to 10.12 are involved in processing ATR.
- `IncURL` (Fig. 10.15) and `DecURL` (Fig. 10.16) increment and decrement the programmed value of URL.
- `IncARP` (Fig. 10.17) and `DecARP` (Fig. 10.18) increment and decrement the programmed value of atrial refractory period.
- `ATROn` (Fig. 10.19(a)) and `ATROff` (Fig. 10.19(b)) turn on and off the programmed value ATR mode.
- `IncATRDuration` (Fig. 10.20) and `DecATRDuration` (Fig. 10.21) increment and decrement the programmed value of ATR duration.
- `IncATRFbTm` (Fig. 10.22) and `DecATRFbTm` (Fig. 10.23) increment and decrement the ATR fallback time.
- `IncATREntryC` (Fig. 10.24) and `DecATREntryC` (Fig. 10.25) increment and decrement the ATR detection counter for beginning ATR.
- `IncATRExitC` (Fig. 10.26) and `DecATREntryC` (Fig. 10.27) increment and decrement the ATR detection counter for ceasing ATR.

10.3 Global Operations

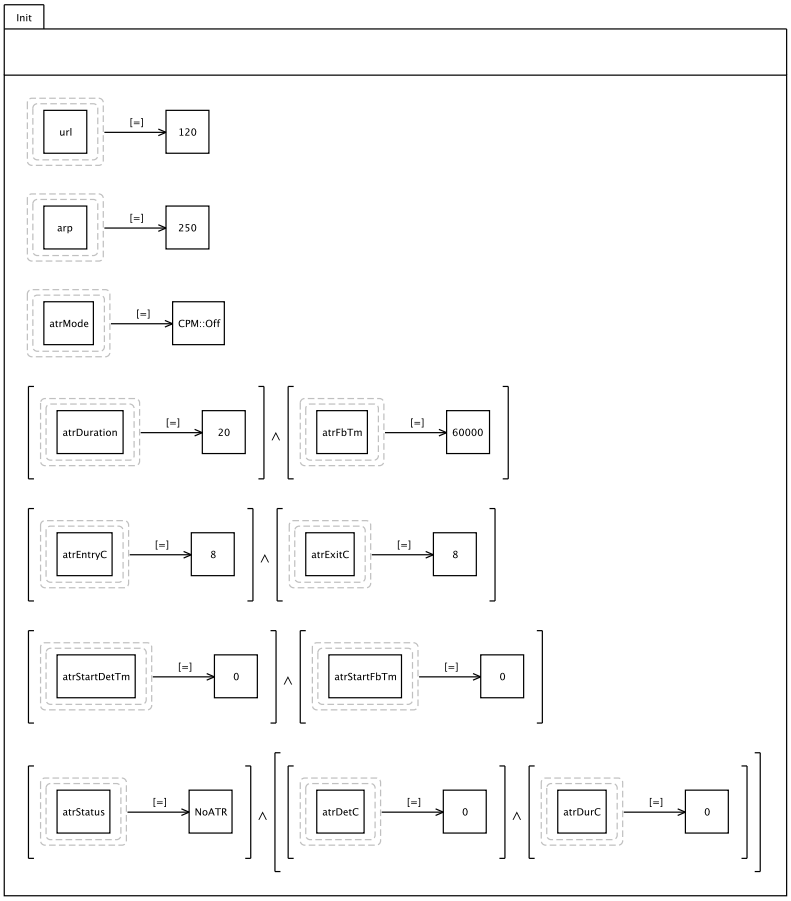


Figure 10.4: Initialisation of package `AtrialResponse`

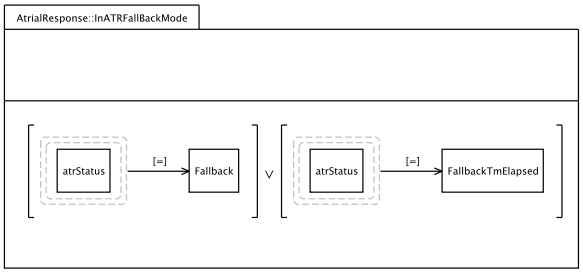


Figure 10.5: Assertion diagrams of operation `InATRFallBackMode` of package `AtrialResponse`

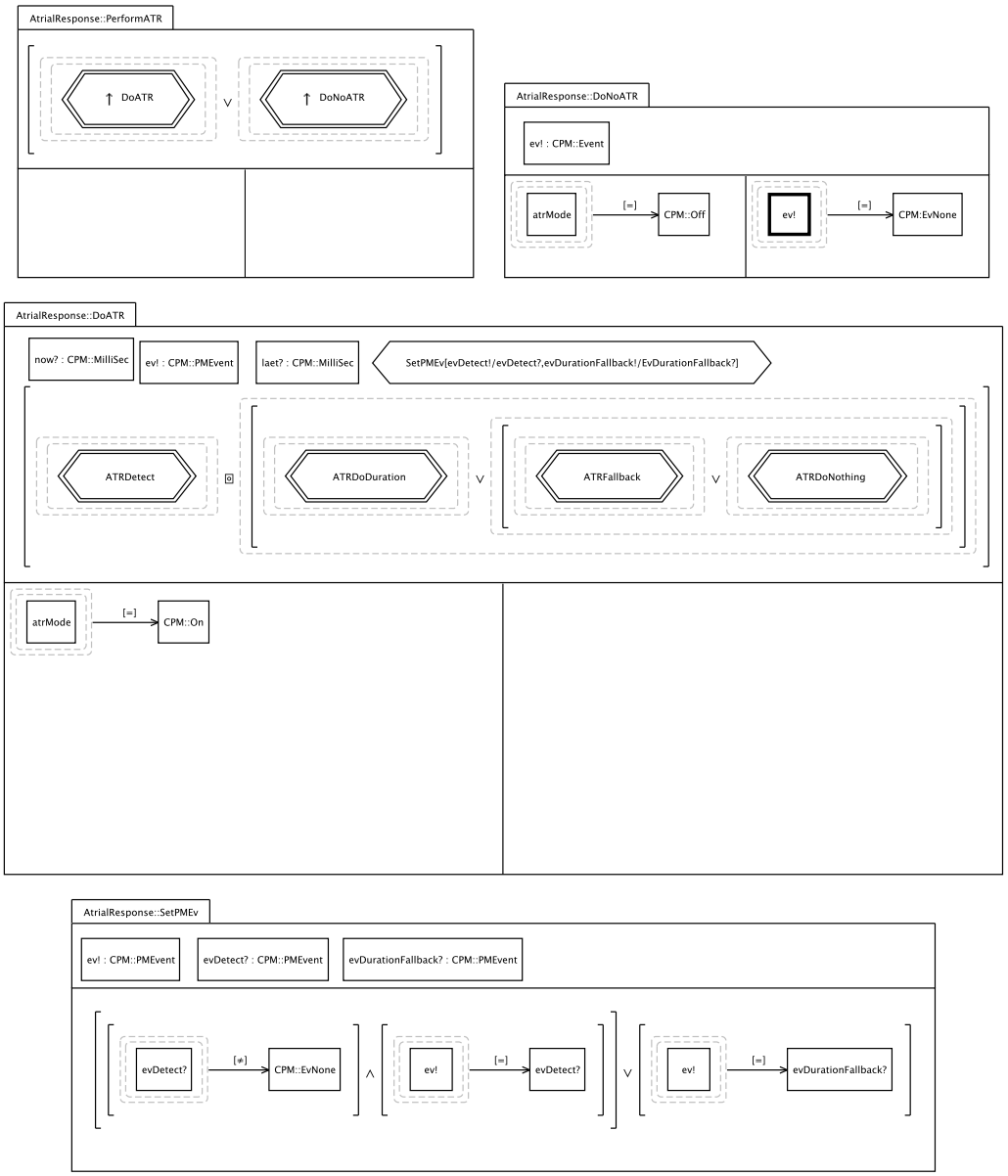


Figure 10.6: Contract diagrams of operation `PerformATR` of package `AtrialResponse`

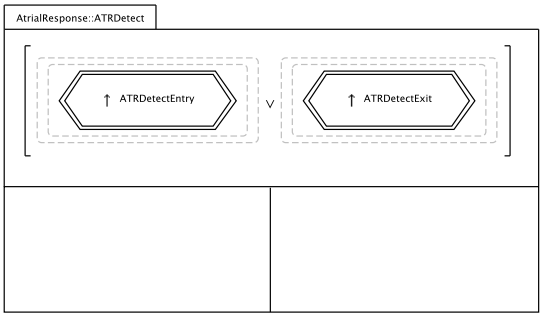


Figure 10.7: Contract diagram of operation `ATRDetect` of package `AtrialResponse`

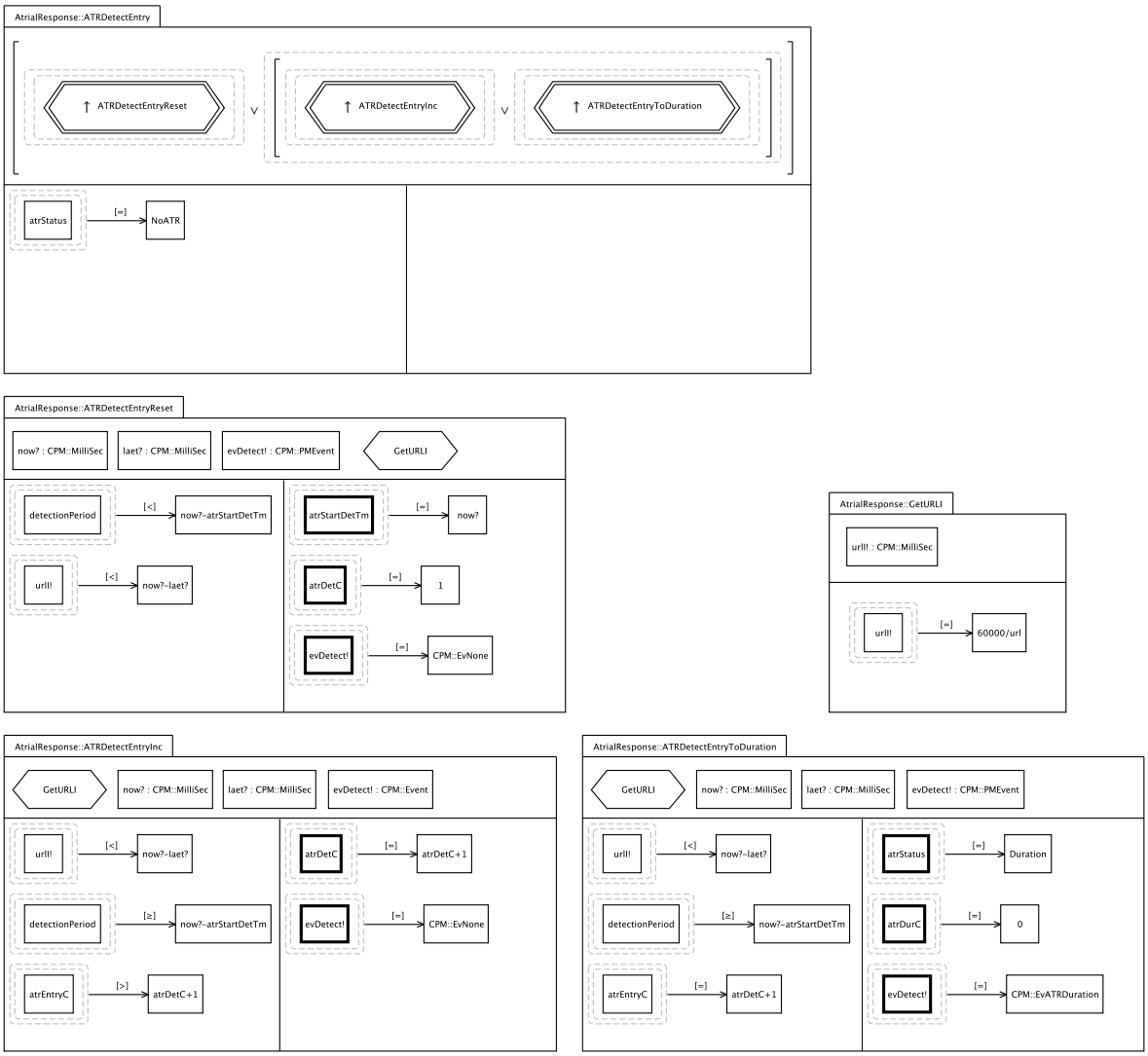


Figure 10.8: Contract diagrams of operation `ATRDetectEntry` of package `AtrialResponse`

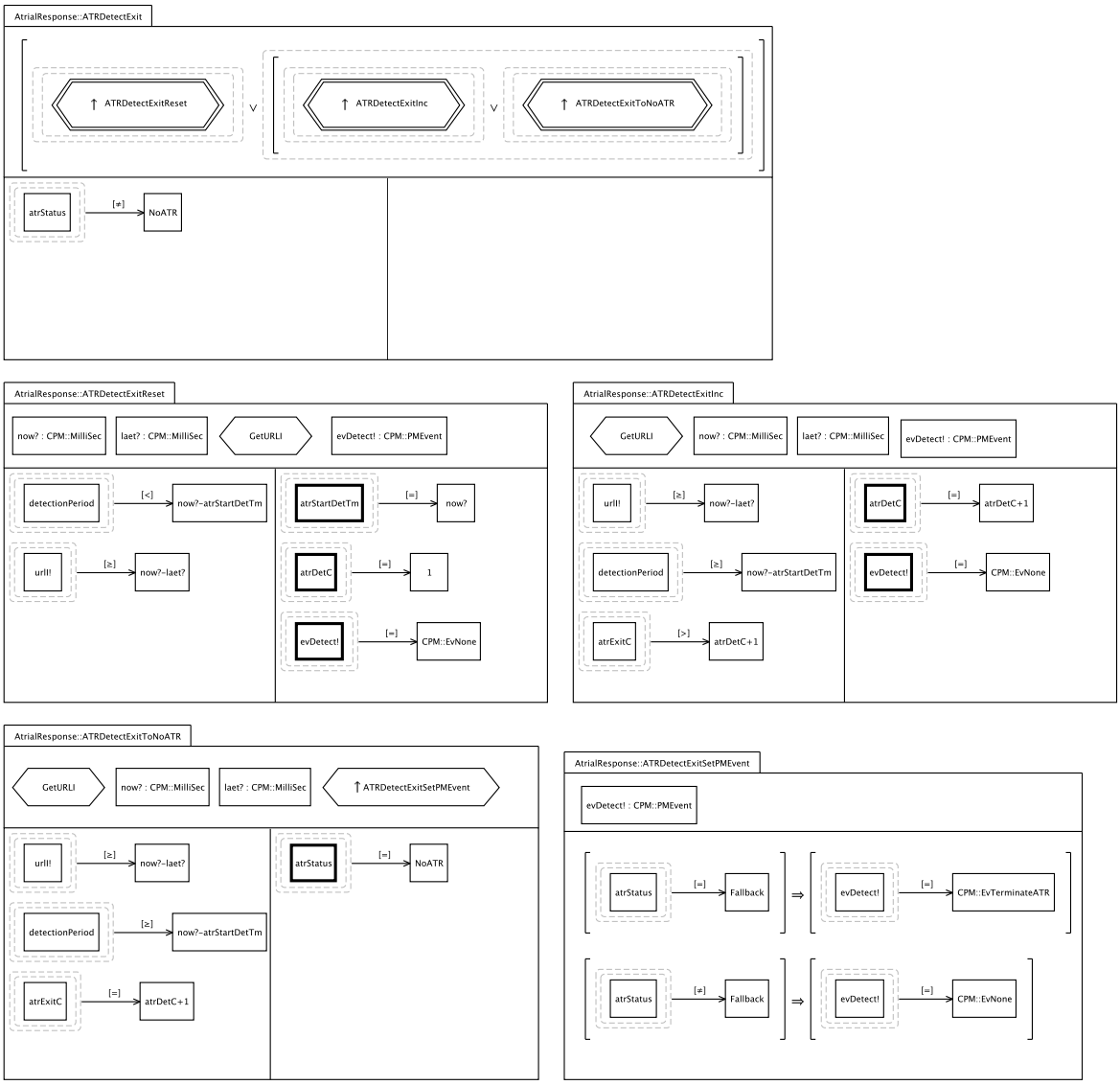


Figure 10.9: Contract diagrams of operation `ATRDetectExit` of package `AtrialResponse`

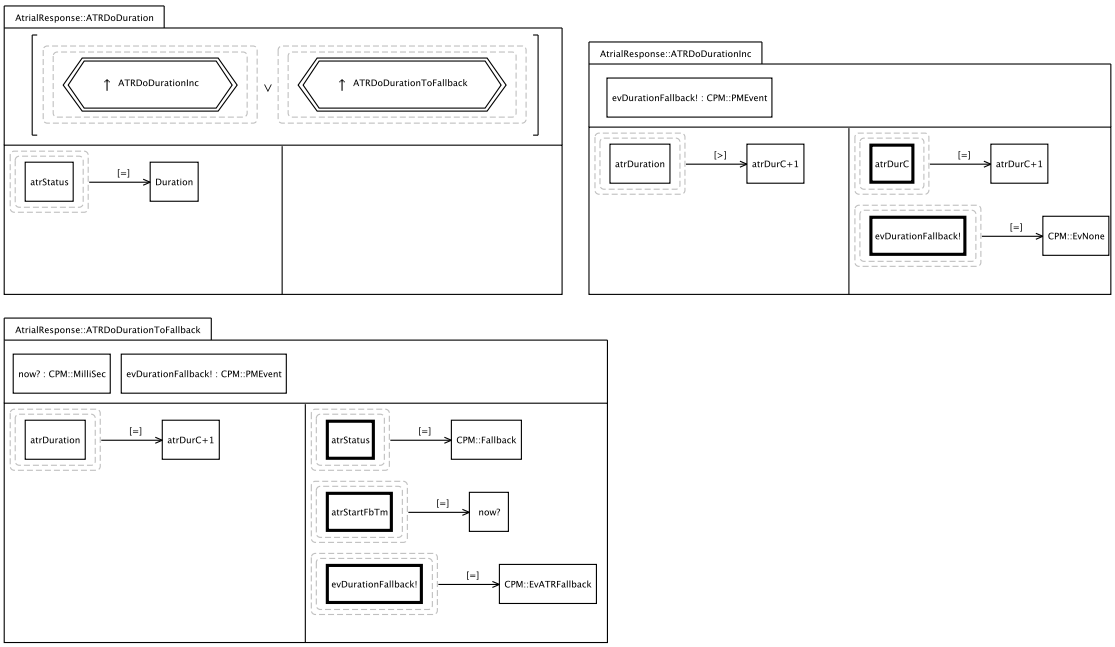


Figure 10.10: Contract diagrams of operation `ATRDoDuration` of package `AtrialResponse`

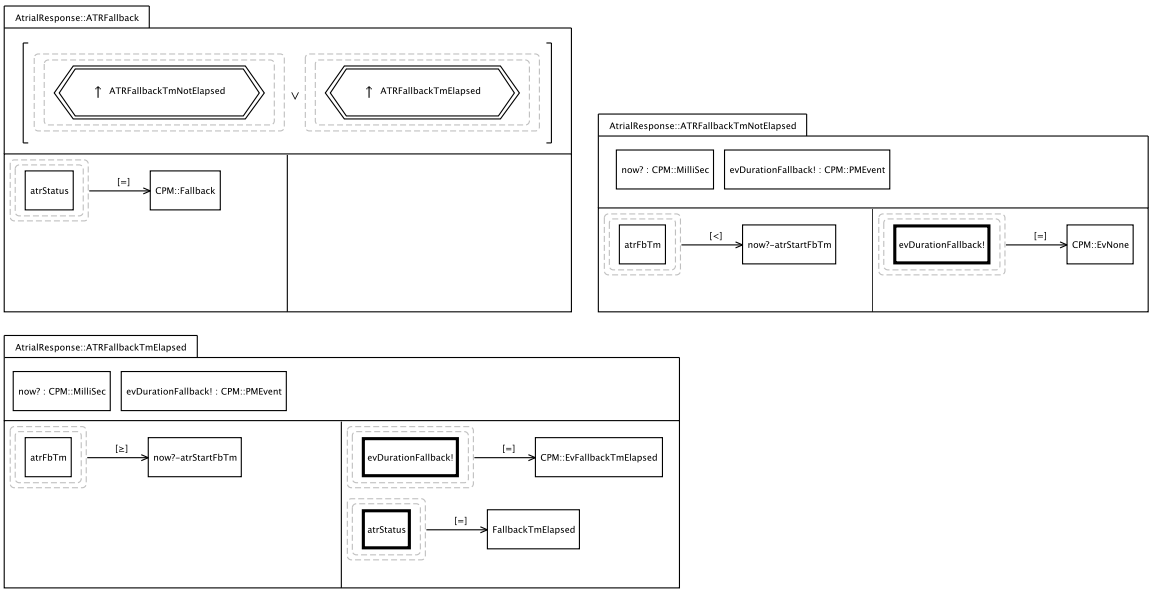


Figure 10.11: Contract diagrams of operation `ATRFallback` of package `AtrialResponse`

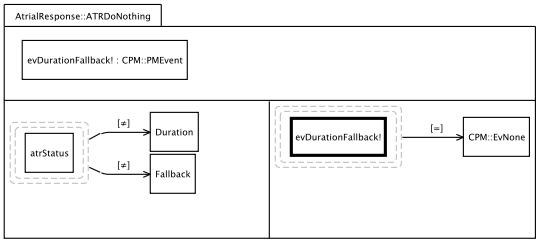


Figure 10.12: Contract diagrams of operation `ATRDoNothing` of package `AtrialResponse`

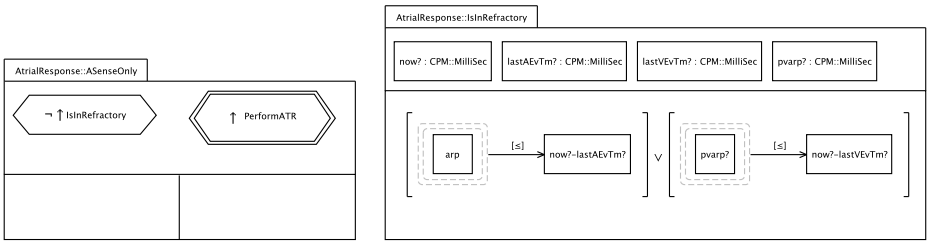


Figure 10.13: Contract and assertion diagrams of operation `ASenseOnly` of package `AtrialResponse`

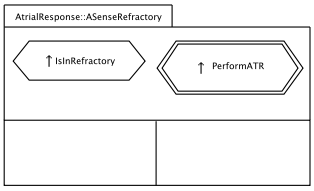


Figure 10.14: Contract diagram of operation `ASenseRefractory` of package `AtrialResponse`

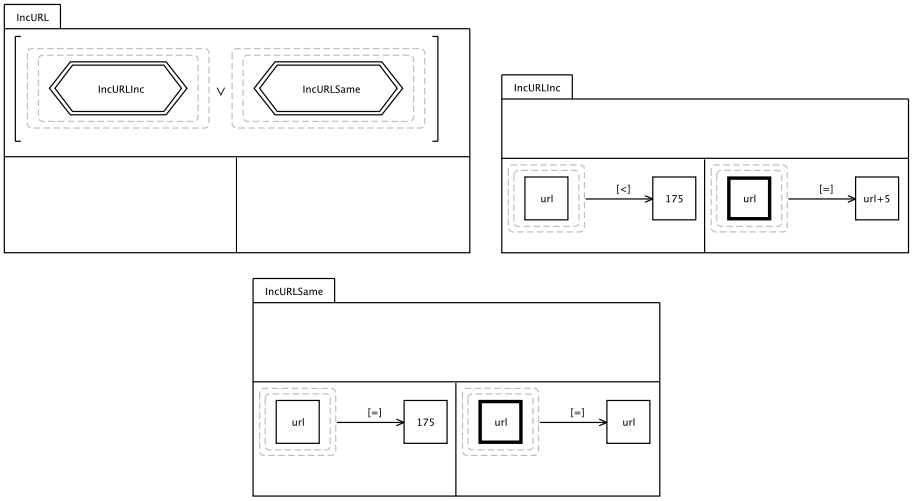


Figure 10.15: Contract diagrams of operation `IncURL` of package `AtrialResponse`

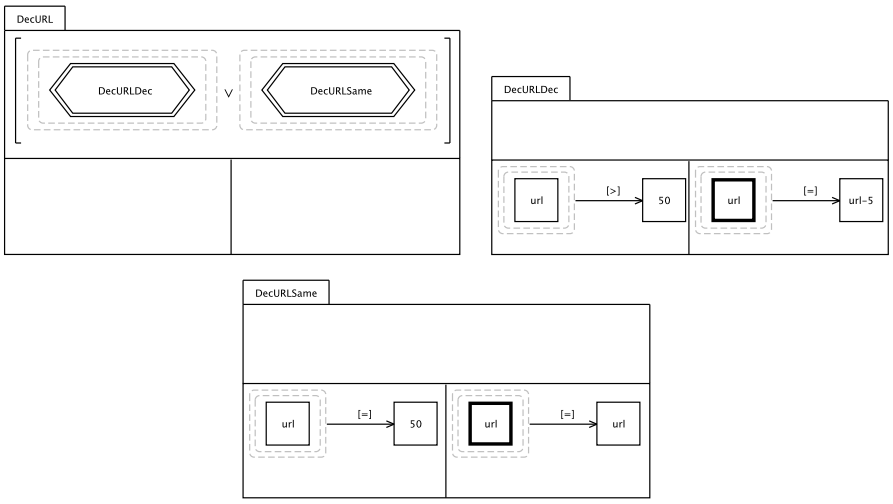


Figure 10.16: Contract diagrams of operation `DecURL` of package `AtrialResponse`

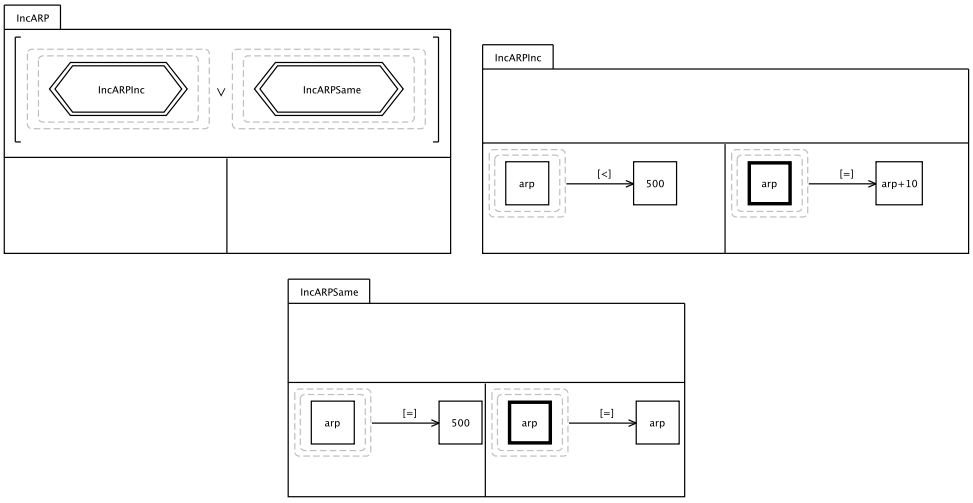


Figure 10.17: Contract diagrams of operation `IncARP` of package `AtrialResponse`

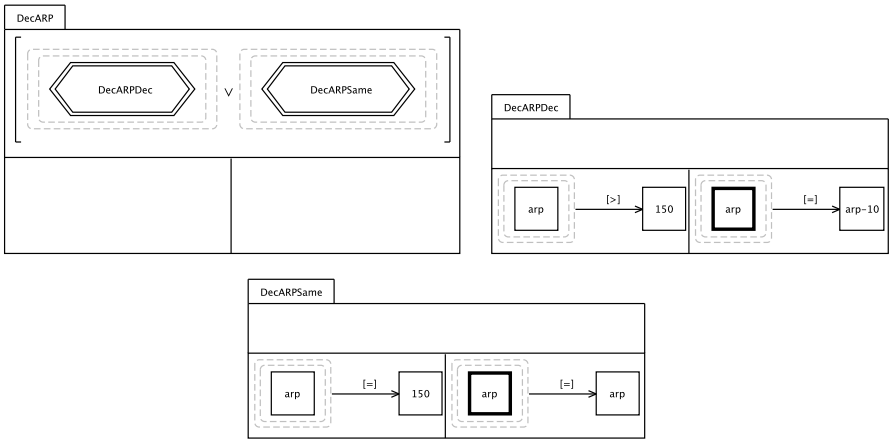
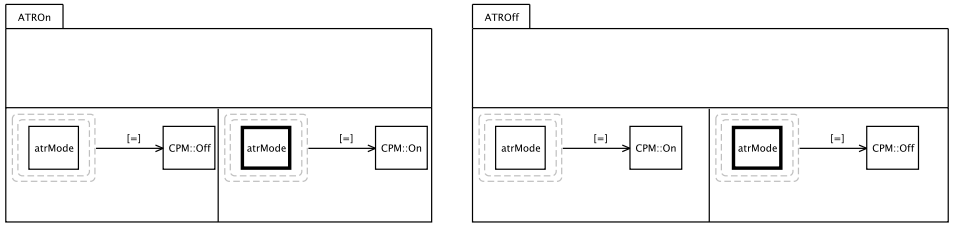


Figure 10.18: Contract diagrams of operation `DecARP` of package `AtrialResponse`



(a) `ATROn` (b) `ATROff`

Figure 10.19: Contract diagrams of operations `ATROn` and `ATROff` of package `AtrialResponse`

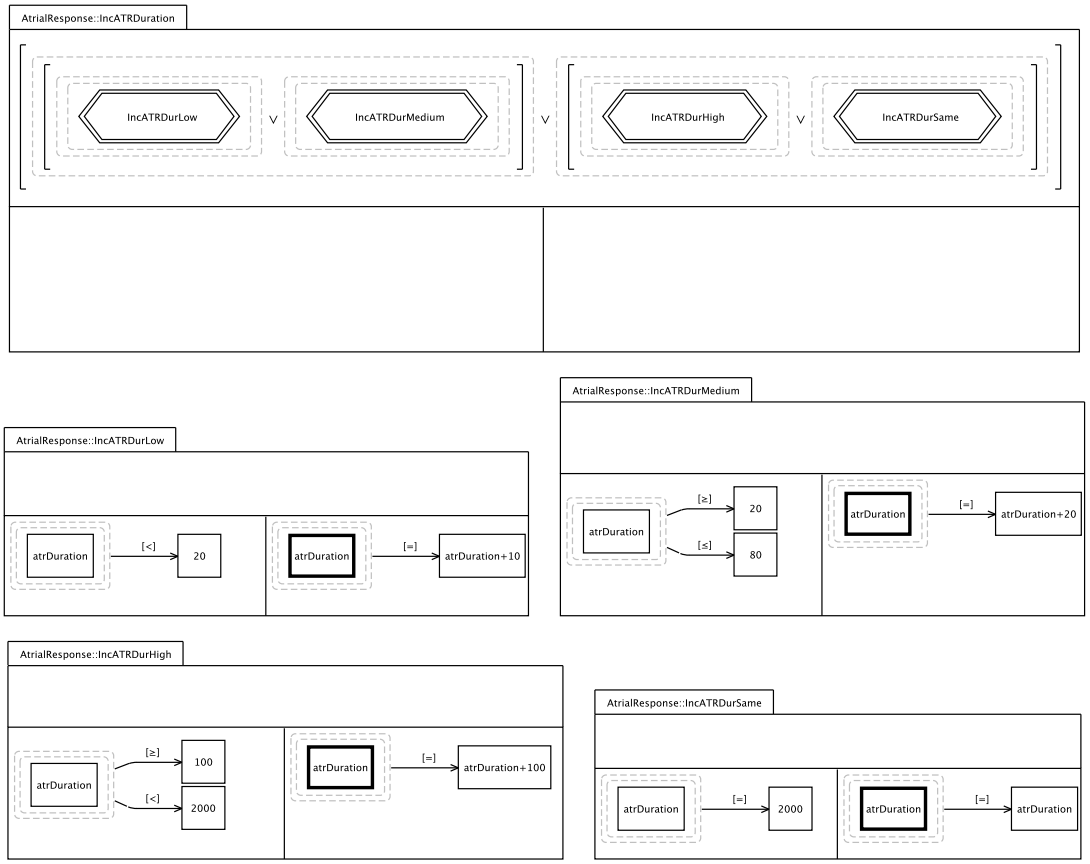


Figure 10.20: Contract diagrams of operations `IncATRDuration` of package `AtrialResponse`

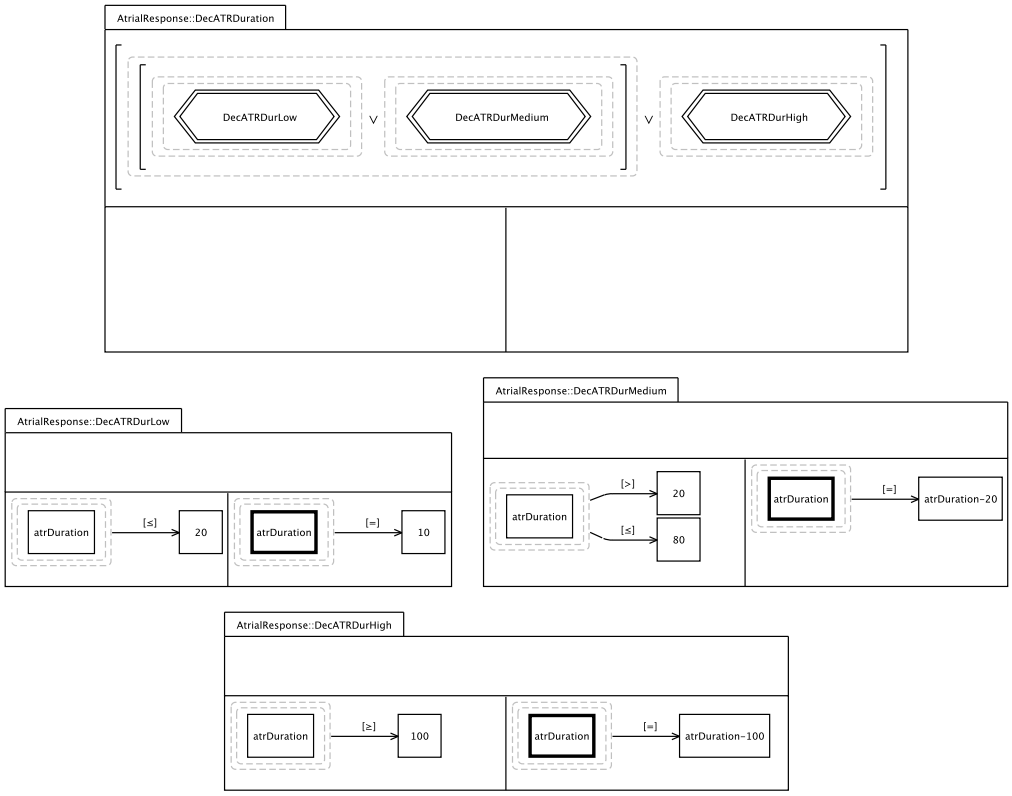


Figure 10.21: Contract diagrams of operations `DecATRDuration` of package `AtrialResponse`

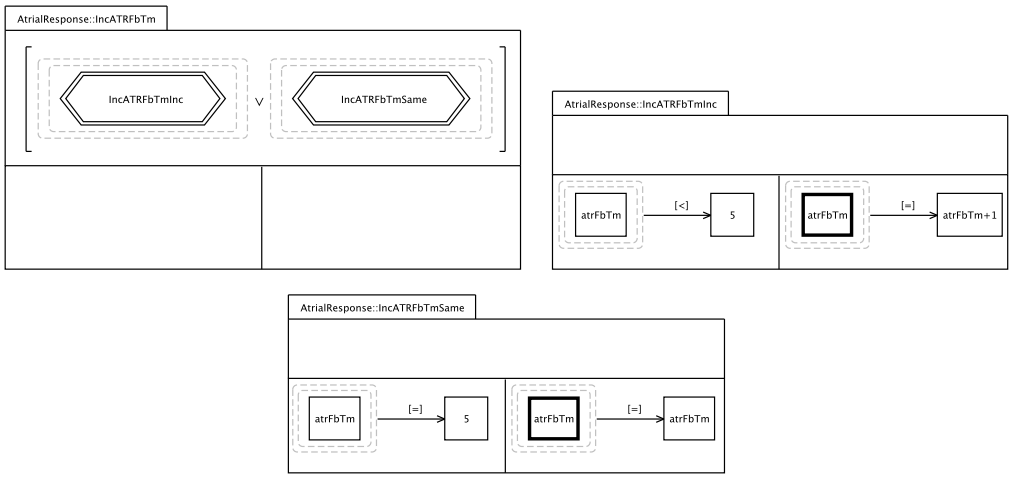


Figure 10.22: Contract diagrams of operations `IncATRFbTm` of package `AtrialResponse`

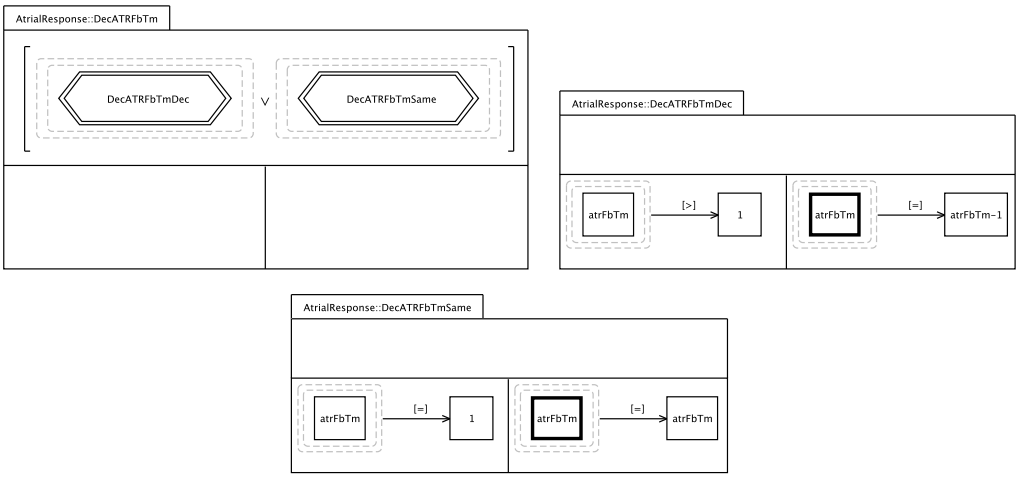


Figure 10.23: Contract diagrams of operations `DecATRFbTm` of package `AtrialResponse`

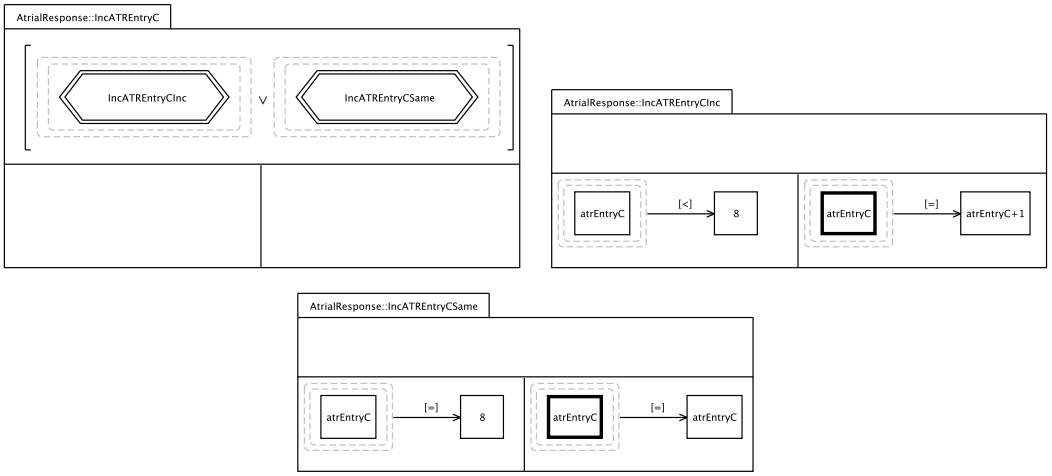


Figure 10.24: Contract diagrams of operations `IncATREntryC` of package `AtrialResponse`

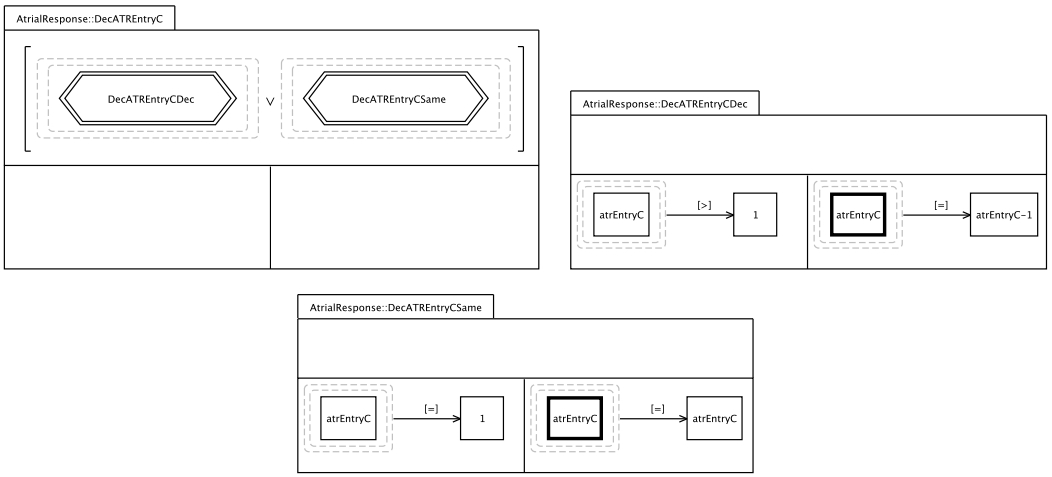


Figure 10.25: Contract diagrams of operations `DecATREntryC` of package `AtrialResponse`

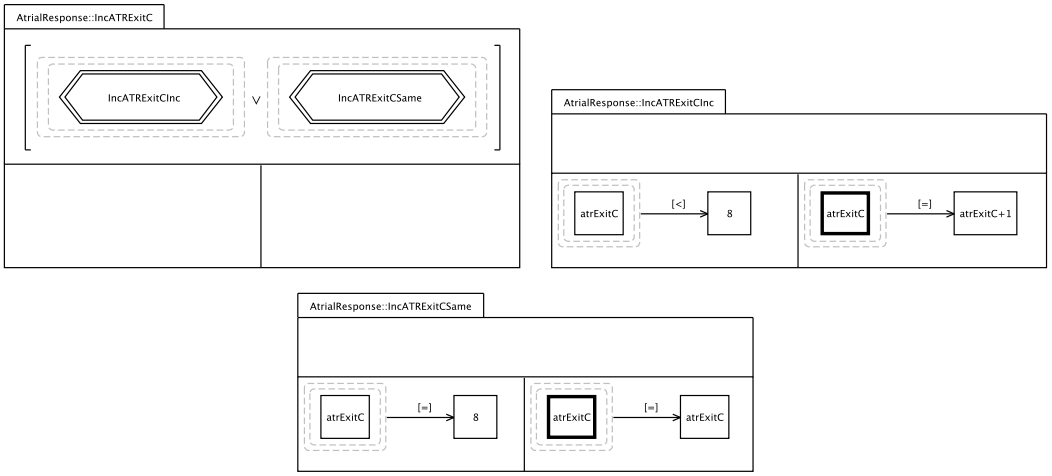


Figure 10.26: Contract diagrams of operations `IncATRExitC` of package `AtrialResponse`

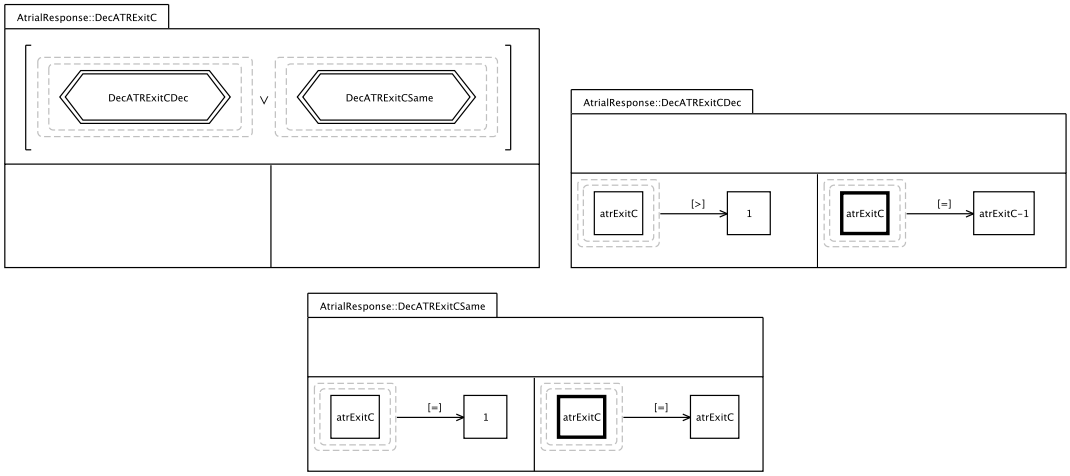


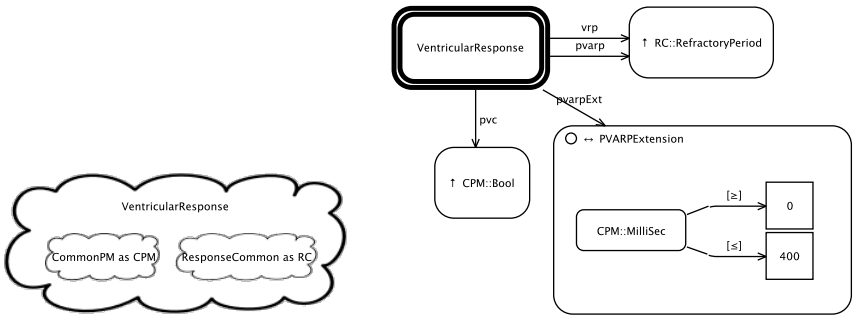
Figure 10.27: Contract diagrams of operations `DecATRExitC` of package `AtrialResponse`

Chapter 11

The `VentricularResponse` Package

11.1 State

Package `VentricularResponse` is responsible for the response-to-sensing functionality of the pacemaker in what concerns the ventricular chamber.



(a) Package Diagram (b) Structural Diagram

Figure 11.1: VCL Package and structural diagram of package `VentricularResponse`

`VentricularResponse`'s PD (Fig. 11.1(a)) defines `VentricularResponse` as an ensemble package. It imports packages `CommonPM` and `ResponseCommon`.

SD of package `VentricularResponse` is given in Fig. 11.1(b). It is as follows:

- The package blob `VentricularResponse` introduces the property edges `vrp` and `pvarp` (of blob `RefractoryPeriod` defined in `ResponseCommon`), representing the *ventricular refractory period* (VRP) and the *post ventricular refractory period* (PVARP). VRP is the programmed time interval following a ventricular event (paced or sensed) during which ventricular senses shall not inhibit nor trigger pacing [Bos07b, p. 30]; its main goal is to eliminate sensing of the T wave and identifying that, incorrectly, as a ventricular event [MM07]. PVARP is the programmable time interval following a ventricular event, when an atrial cardiac cycle event shall not inhibit nor trigger a ventricular pace [Bos07b, p. 30]; this is the period when there is no sensing in the atrium [MM07], it avoids inappropriate atrial sensing of ventricular events and retrograde P waves [BSS10].

- Property edge `pvarpext` of package blob `VentricularResponse` represents the PVARP extension. Its target is the derived blob `PVARPExtension`. Blob `PVARPExtension` represents the allowed values for PVARP (post ventricular atrial refractory period) extension expressed in milliseconds (ms) and defined according to the "programmable parameters table" of [Bos07b, p. 34]; the value 0 represents the fact that PVARP extension is disabled. PVARP extension defines a value to be used in the place of the PVARP when premature ventricular contraction occurs [Bos07b, p. 30].
- Property edge `pvc` of package blob `VentricularResponse` indicates whether a premature ventricular contraction (PVC) has been detected and the PVARP extension needs to be applied.

11.2 Overall Behaviour

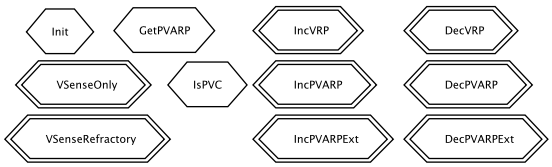
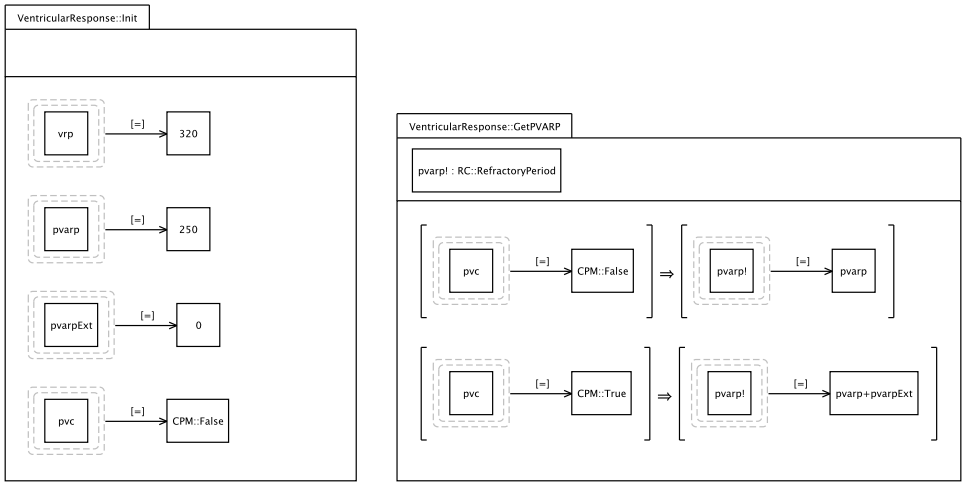


Figure 11.2: VCL behavioural diagram of package `VentricularResponse`

BD of package `AtrialResponse` is given in Fig. 11.2. It introduces the following behavioural units:

- `Init` (Fig. 11.3(a)) is the package's initialisation.
- `GetPVARP` (Fig. 11.3(b)) gets the current value of the post-ventricular atrial period. This can be the normal PVARP or the extended PVARP if a PVC has been detected.
- `IsPVC` (Fig. 11.4) indicates whether a PVC has been detected or not.
- `IncVRP` (Fig. 11.5) and `DecVRP` (Fig. 11.6) increment and decrement the programmed value of the ventricular refractory period (VRP).
- `IncPVARP` (Fig. 11.7) and `DecPVARP` (Fig. 11.8) increment and decrement the programmed value of the post ventricular atrial refractory period (PVARP).
- `IncPVARPExt` (Fig. 11.9) and `DecPVARPExt` (Fig. 11.10) increment and decrement the programmed value of the extended post ventricular atrial refractory period (PVARP).
- `VSenseOnly` (Fig. 11.11(a)) is the operation doing the normal ventricular sensing.
- `VSenseRefractory` (Fig. 11.11(b)) is the operation doing the ventricular sensing during the refractory period.

11.3 Global Operations



(a) Initialisation (b) Operation `GetPVARP`

Figure 11.3: Assertion diagrams of initialisation and `GetPVARP` of package `VentricularResponse`

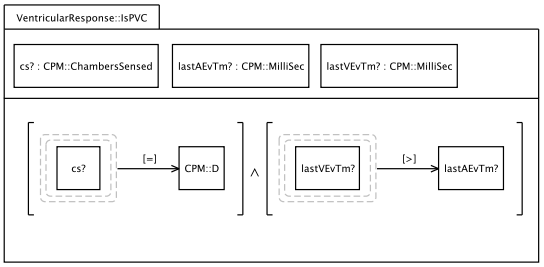


Figure 11.4: Assertion diagrams of operation `IsPVC` of package `VentricularResponse`

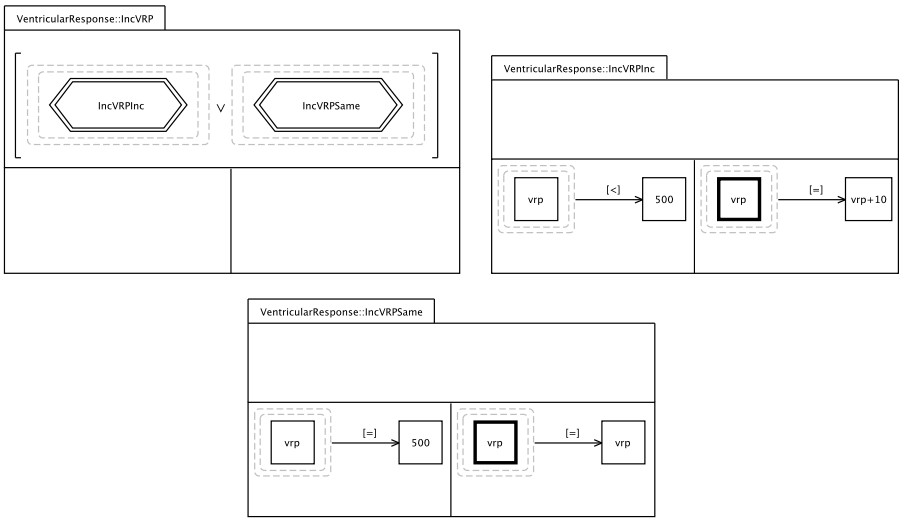


Figure 11.5: Contract diagrams of operation `IncVRP` of package `VentricularResponse`

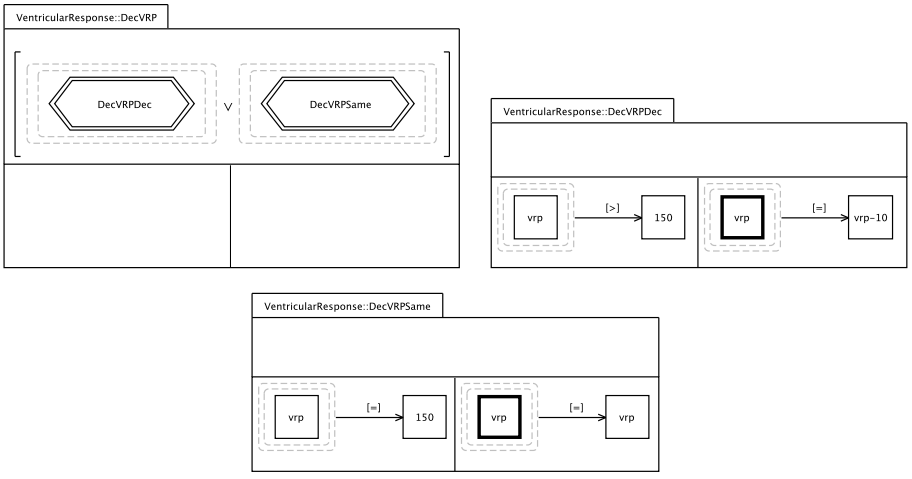


Figure 11.6: Contract diagrams of operation `DecVRP` of package `VentricularResponse`

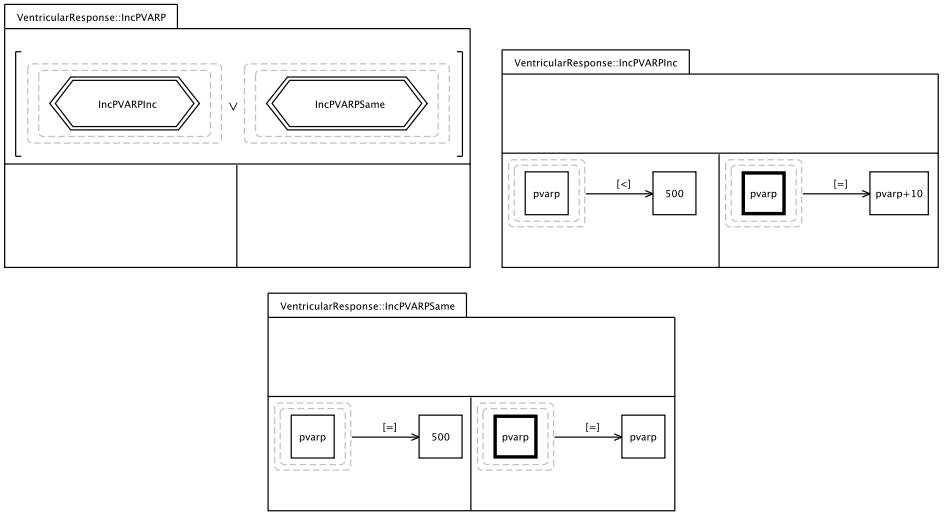


Figure 11.7: Contract diagrams of operation `IncPVARP` of package `VentricularResponse`

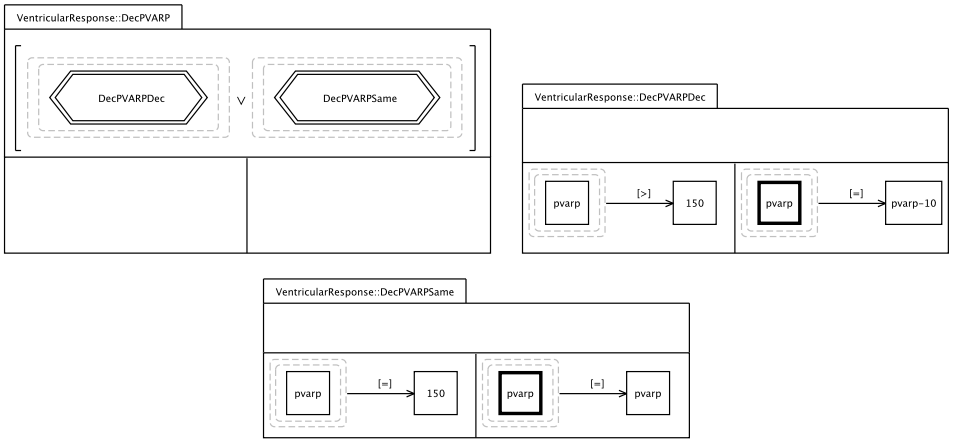


Figure 11.8: Contract diagrams of operation `DecPVARP` of package `VentricularResponse`

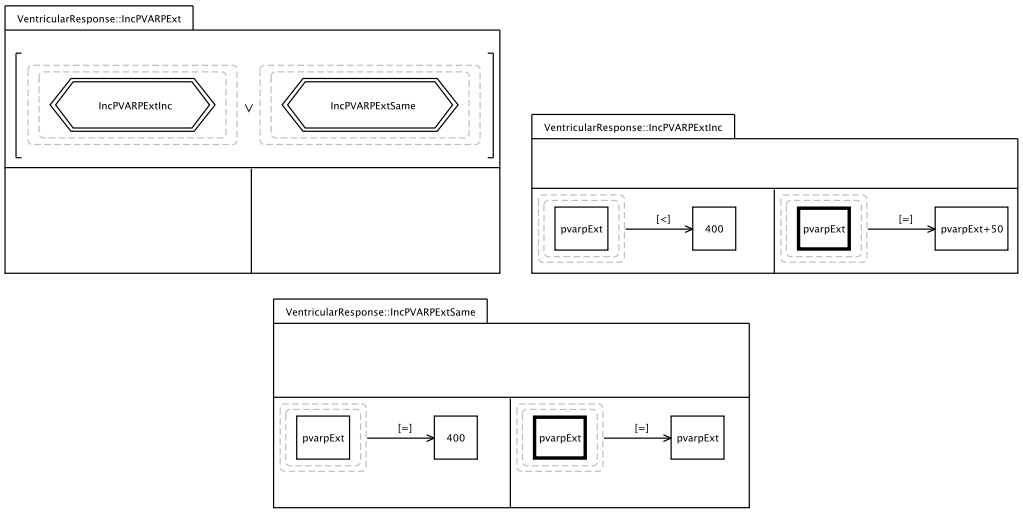


Figure 11.9: Contract diagrams of operation `IncPVARPExt` of package `VentricularResponse`

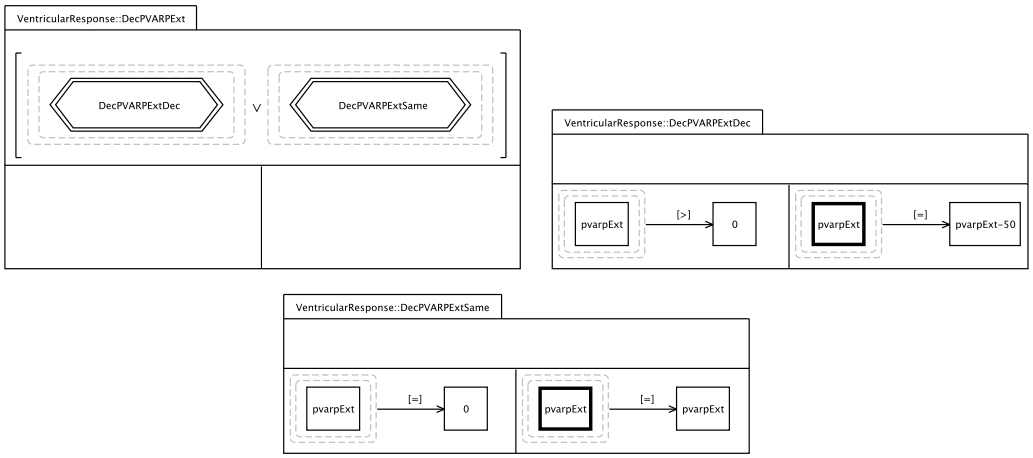
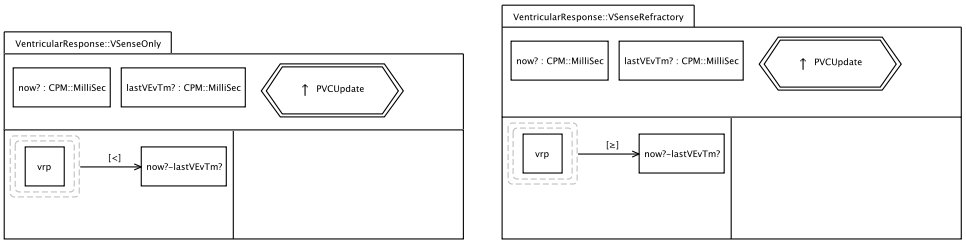


Figure 11.10: Contract diagrams of operation `DecPVARPExt` of package `VentricularResponse`



(a) Operation `VSenseOnly` (b) Operation `VSenseRefractory`

Figure 11.11: Contract diagrams of operations `VSenseOnly` and `VSenseRefractory` of package `VentricularResponse`

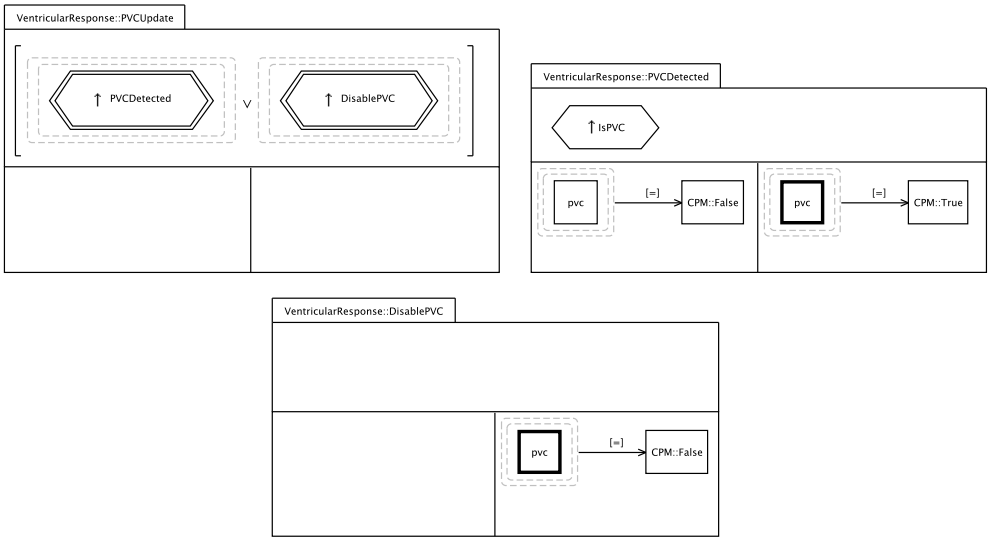


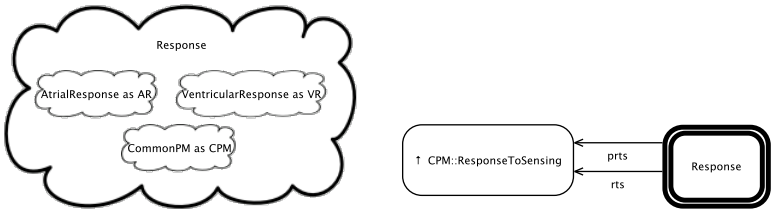
Figure 11.12: Contract diagrams of operation `PVCUpdate` of package `VentricularResponse`

Chapter 12

The `Response` package

12.1 State

Package `AtrialResponse` is responsible for the response-to-sensing functionality of the pacemaker. It puts together packages `AtrialResponse` (chapter 10) and `VentricularResponse` (chapter 11).



(a) Package Diagram (b) Structural Diagram

Figure 12.1: VCL Package and structural diagram of package `VentricularResponse`

`Response`'s PD (Fig. 12.1(a)) defines `Response` as an ensemble package. It imports packages `CommonPM`, `AtrialResponse` and `VentricularResponse`.

12.2 Overall Behaviour

BD of `Response` is given in Fig. 12.2. It includes two integral extensions to bring in operations from imported packages: one brings into the new context all operations from package `AtrialResponse`, except `Init`, `ASenseRefractory` and `ASenseOnly`; the other all operations from package `VentricularResponse`, except `Init`, `GetVARP` and `VSenseOnly`.

BD of package `Response` is given in Fig. 12.2. It introduces the following global operations:

- `Init` (Fig. 12.3(a)) is the package's initialisation.
- `SetBradycardiaMode` (Fig. 12.3(b)) is the operation that sets the bradycardia mode for response-to-sensing.

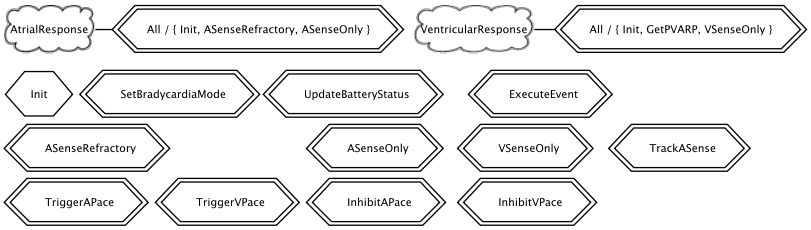
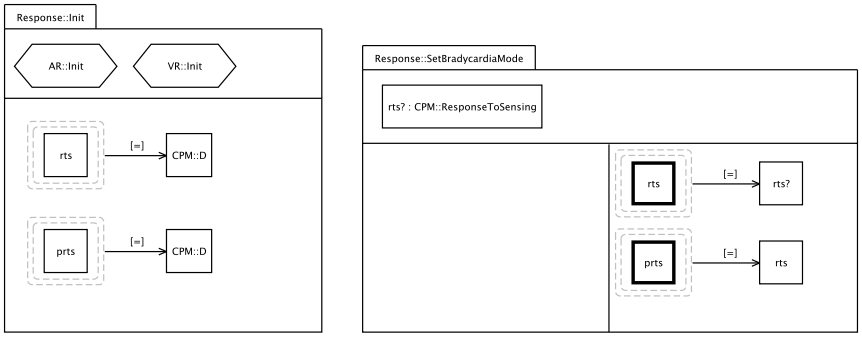


Figure 12.2: VCL behavioural diagram of package `Response`

- `UpdateBatteryStatus` (Fig. 12.4) does the required action for response-to-sensing when the battery status is updated.
- `ExecuteEvent` (Fig. 12.5) executes some internal pacemaker event in the context of response-to-sensing.
- `ASenseRefractory` (Fig. 12.6(a)) is the operation doing the atrial sensing during the refractory period.
- `ASenseOnly` (Fig. 12.6(b)) is the operation doing the normal atrial sensing.
- `VSenseOnly` (Fig. 12.7(a)) is the operation doing the normal ventricular sensing.
- `TrackASense` (Fig. 12.7(b)) corresponds to the tracked atrial sensing in response-to-sensing. During tracked pacing, an atrial sense shall cause a tracked ventricular pace after a programmed AV delay unless a ventricular sense was detected beforehand [Bos07b, p. 17].
- `TriggerAPace` (Fig. 12.8(a)) and `TriggerVPace` (Fig. 12.8(b)) correspond to the triggered atrial and ventricular pacing. In the triggered mode of response to sensing, a sense in a chamber shall trigger an immediate pace in that chamber [Bos07b, p. 17]. These operations are called when this occurs. Here, package `Response` describes the effect of this package on these operations (it adds further pre-conditions).
- `InhibitAPace` (Fig. 12.9(a)) and `InhibitVPace` (Fig. 12.9(b)) correspond to the inhibited atrial and ventricular pacing. In the inhibited mode of response to sensing, a sense in a chamber shall inhibit a pending pace in that chamber [Bos07b, p. 17]. These operations are called when this occurs. Here, package `response` describes the effect of this package on these operations (it adds further pre-conditions).

12.3 Global Operations



(a) Initialisation (b) Operation `SetBradycardiaMode`

Figure 12.3: Package `response`: Assertion diagram of Initialisation and contract diagram of operation `SetBradycardiaMode`

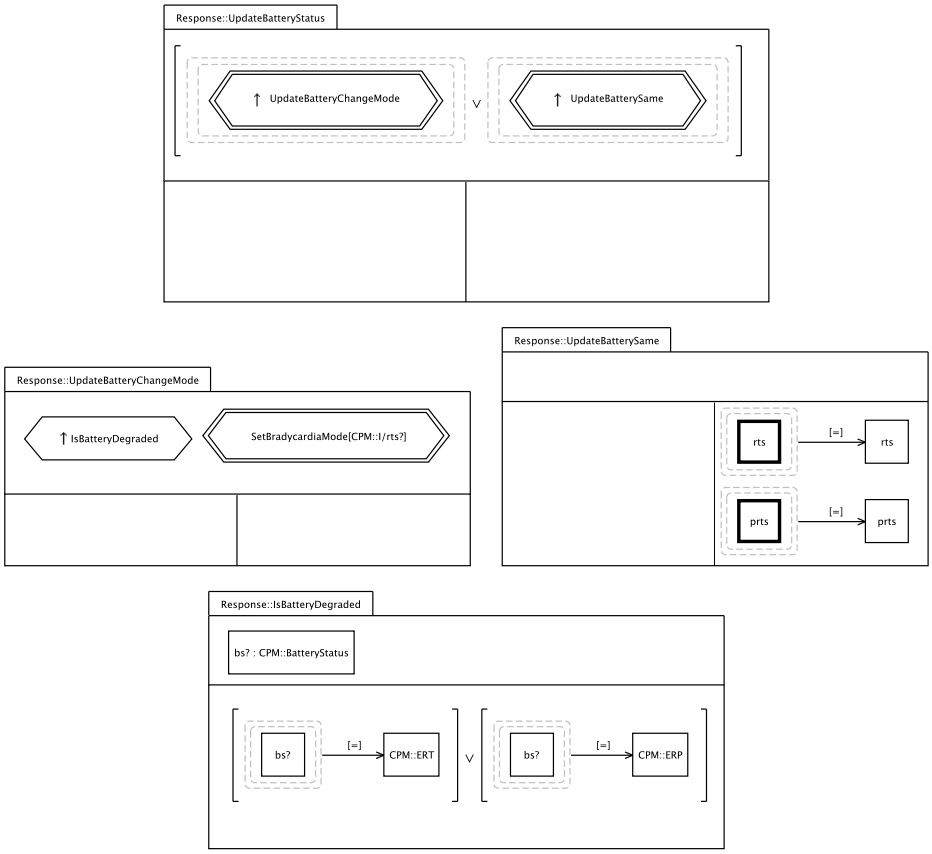


Figure 12.4: Contract Diagrams of global operation `UpdateBatteryStatus`

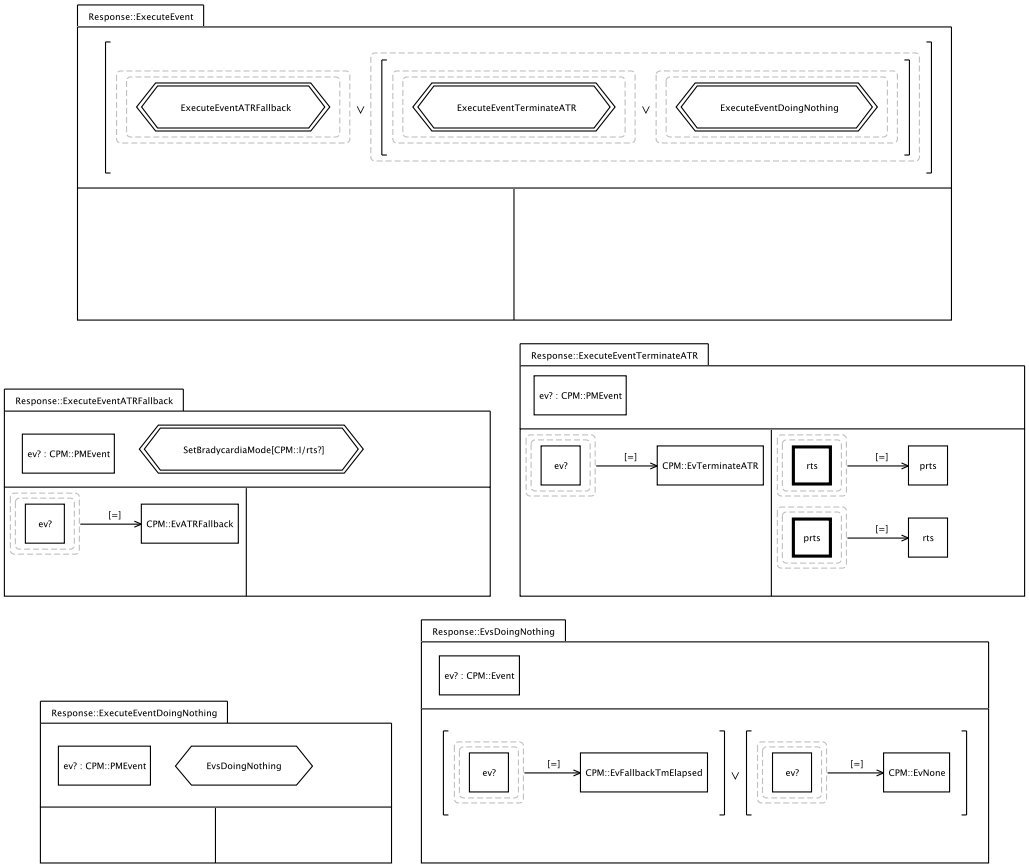
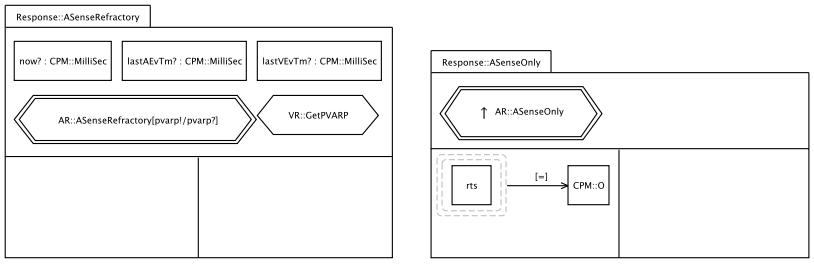
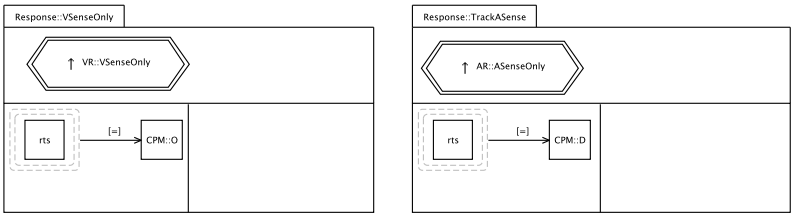


Figure 12.5: Contract Diagrams of global operation `ExecuteEvent`



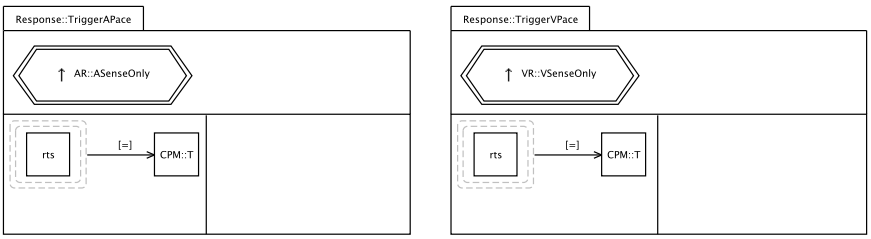
(a) Operation `ASenseRefractory` (b) Operation `ASenseOnly`

Figure 12.6: Contract diagrams of operations `ASenseRefractory` and `ASenseOnly` of package `Response`



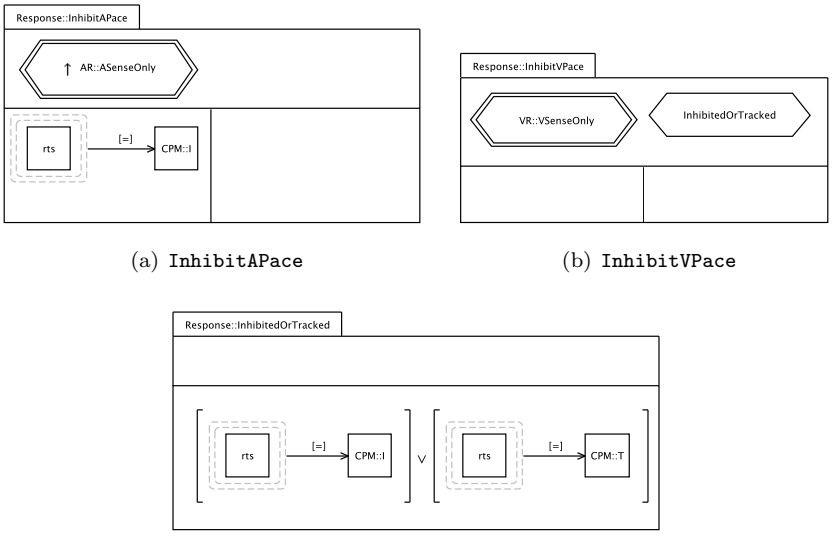
(a) Operation `VSenseOnly` (b) Operation `TrackASense`

Figure 12.7: Contract diagrams of operations `VSenseOnly` and `TrackASense` of package `Response`



(a) `TriggerAPace` (b) `TriggerVPace`

Figure 12.8: Contract diagrams of operations `TriggerAPace` and `TriggerVPace` of package `Response`



(a) `InhibitAPace` (b) `InhibitVPace`

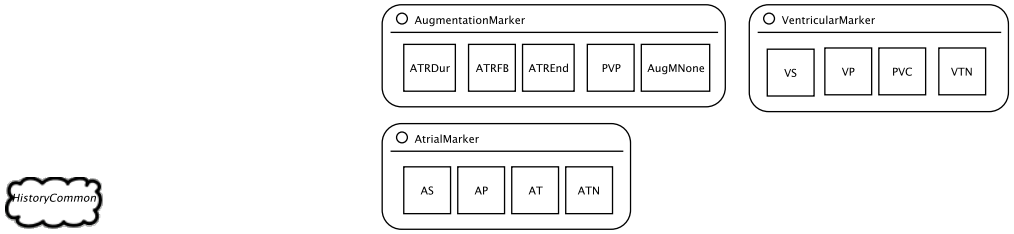
(c) `InhibitedOrTracked`

Figure 12.9: Contract and assertion diagrams of operations `InhibitAPace` and `InhibitVPace` of package `Response`

Chapter 13

The `HistoryCommon` Package

The `HistoryCommon` package introduces common set for recording event markers (History).



(a) Package Diagram

(b) Structural Diagram

Figure 13.1: VCL Package and structural diagram of package `HistoryCommon`

`HistoryCommon`'s PD (Fig. 13.1(a)) defines `HistoryCommon` as a container package.

The package's SD (Fig. 13.1(b)) defines enumerated blobs `AugmentationMarker`, `VentricularMarker` and `AtrialMarker`, which are defined according to sections 4.8.1, 4.8.2 and 4.8.3 of [Bos07b, p. 26].

Chapter 14

The `History` Package

14.1 State

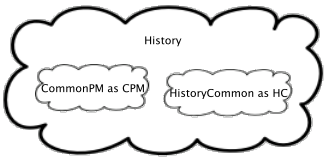


Figure 14.1: VCL Package diagram of package `History`

`History`'s PD (Fig. 14.1) defines `History` as an ensemble package, which imports container packages `CommonPM` and `HistoryPM`.

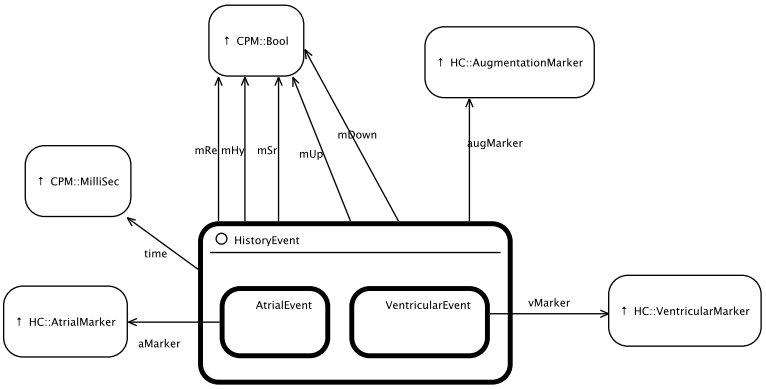


Figure 14.2: VCL structural diagram of package `History`

14.2 Overall Behaviour

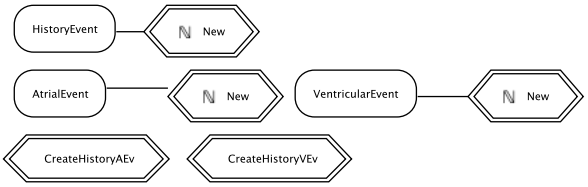


Figure 14.3: VCL behavioural diagram of package `History`

The BD Fig. 14.3 defines several local operations:

- `New` of blob `HistoryEvent` (Fig. 14.4) an abstract operation that factors the commonality of `HistoryEvent` instances.
- `New` of blob `AtrialEvent` (Fig. 14.5) to create a new `AtrialEvent` instance.
- `New` of blob `VentricularEvent` (Fig. 14.6) to create a new `VentricularEvent` instance.

The following global operations are defined in the BD of Fig. 14.3:

- `CreateHistoryAEv` (Fig. 14.7) is the global operation to create new atrial history events.
- `CreateHistoryVEv` (Fig. 14.8) is the global operation to create new ventricular history events.

14.3 Local operations of blob `HistoryEvent`

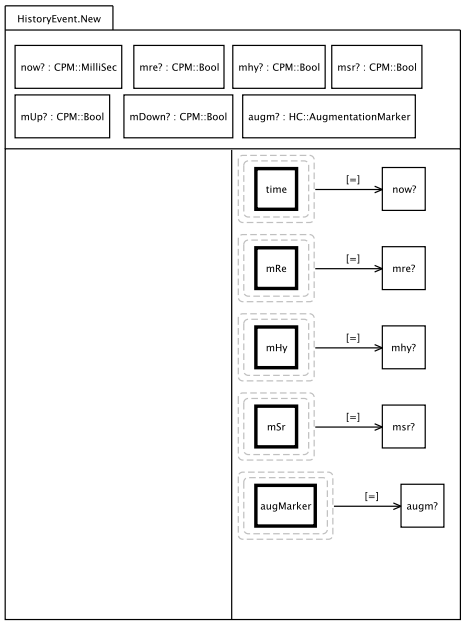


Figure 14.4: Contract diagram of operation `New` of blob `HistoryEvent`

14.4 Local operations of blob `AtrialEvent`

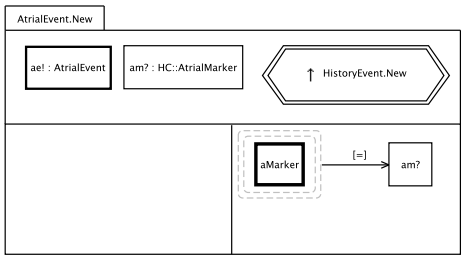


Figure 14.5: Contract diagram of operation `New` of blob `AtrialEvent`

14.5 Local operations of blob `VentricularEvent`

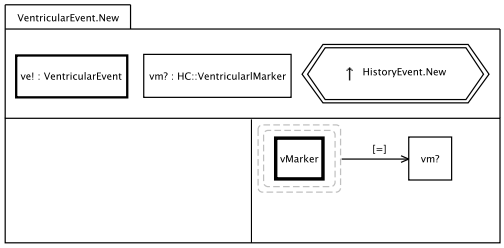


Figure 14.6: Contract diagram of operation `New` of blob `VentricularEvent`

14.6 Global Operations

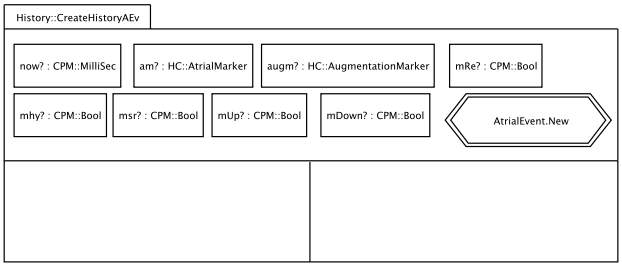


Figure 14.7: Contract diagram of operation `CreateHistoryAEv` of package `History`

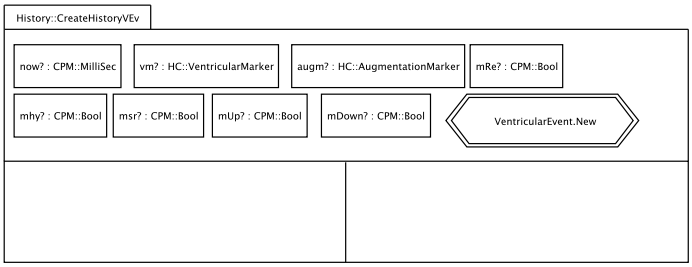


Figure 14.8: Contract diagram of operation `CreateHistoryVEv` of package `History`

Chapter 15

The PacingWithHistoryI package

The package PacingWithHistoryI extends the package Pacing (chapter 7) with an interface to enable composition with the package History (chapter 14).

15.1 State

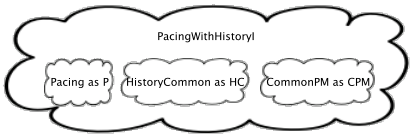


Figure 15.1: VCL Package diagram of package PacingWithHistoryI

The PD of PacingWithHistoryI is given in Fig. 15.1. It defines PacingWithHistory as an ensemble package, which imports ensemble package Pacing, and container packages HistoryCommon and CommonPM.

15.2 Overall Behaviour

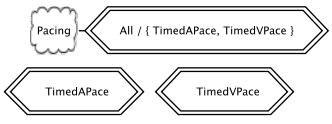


Figure 15.2: VCL behavioural diagram of package PacingWithHistory

BD of package PacingWithHistoryI is given in Fig. 15.2. It includes one integral extension to bring in operations from package Pacing, which says that all operations of Pacing except TimedAPace and TimedVPace are imported integrally.

In addition, BD of Fig. 15.2 defines two operations defined in `Pacing` that are not defined integrally: `TimedAPace` (Fig. 15.4(a)) and `TimedVPace` (Fig. 15.4(b)). These operations extend `TimedAPace` and `TimedVPace` of `Pacing` with the creation of the proper history markers for those events.

15.3 Global Operations

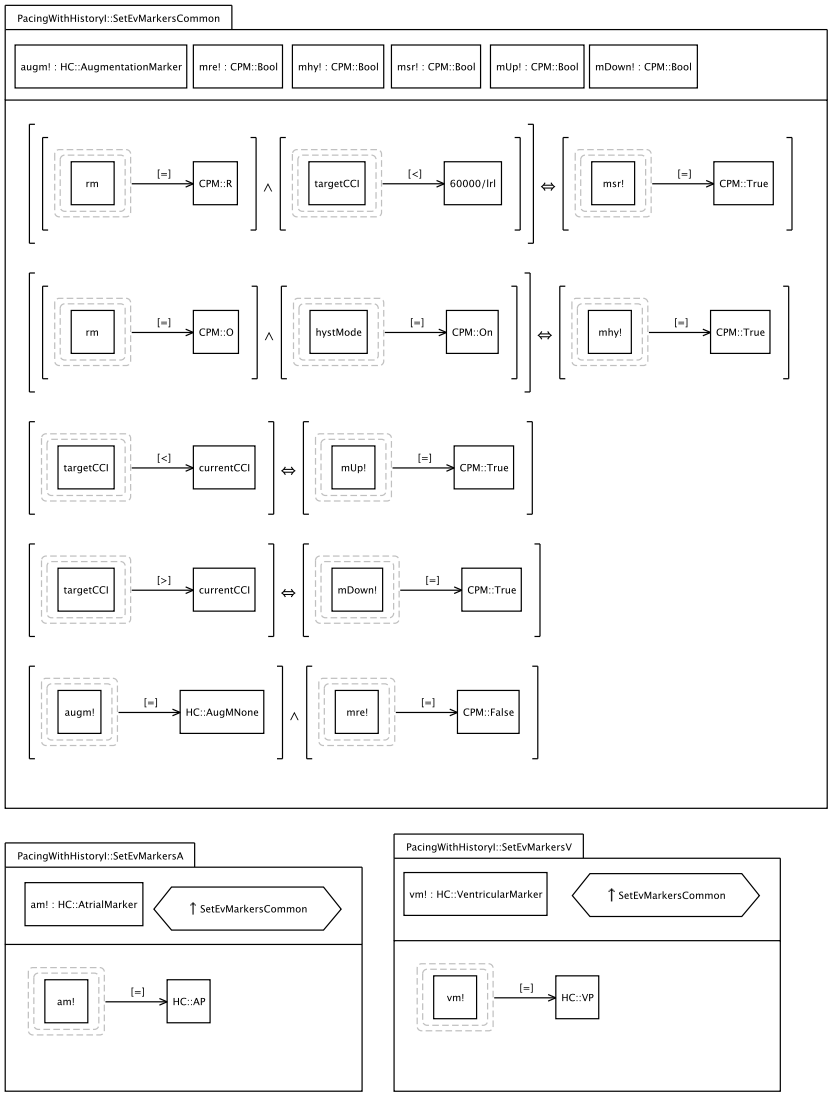
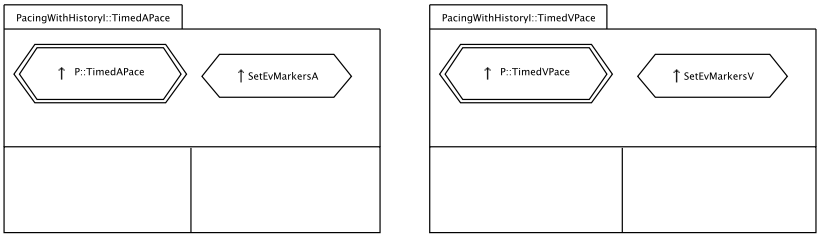


Figure 15.3: Assertion diagrams of operations `SetEvMarkersA` and `SetEvMarkersV` of package `PacingWithHistory`



(a) Operation `TimedAPace` (b) Operation `TimedVPace`

Figure 15.4: Contract diagrams of operations `TimedAPace` and `TimedVPace` of package `PacingWithHistory`

Chapter 16

The `ResponseWithHistoryI` package

The package `ResponseWithHistoryI` extends the package `Response` (chapter 12) with an interface to enable composition with the package `History` (chapter 14).

16.1 State

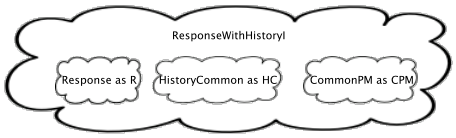


Figure 16.1: VCL Package diagram of package `ResponseWithHistoryI`

The PD of `ResponseWithHistoryI` is given in Fig. 16.1. It defines `ResponseWithHistoryI` as an ensemble package, which imports ensemble package `Response`, and container packages `HistoryCommon` and `CommonPM`.

16.2 Overall Behaviour

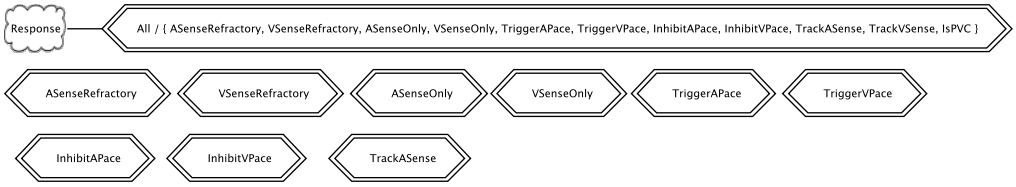


Figure 16.2: VCL behavioural diagram of package `ResponseWithHistory`

BD of package `ResponseWithHistory` is given in Fig. 16.2. It includes one integral extension to bring in operations from package `Response`, which says that all operations of `Response` except

ASenseRefractory, VSenseRefractory, ASenseOnly, VSenseOnly, TriggerAPace, TriggerVPace, InhibitAPace, InhibitVPace, TrackASense and TrackVSense are imported integrally.

In addition, BD of Fig. 15.2 defines the operations defined in Response that are not defined integrally:

- ASenseRefractory (Fig. 16.6(a)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersA (Fig. 16.3).
- VSenseRefractory (Fig. 16.6(b)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersV (Fig. 16.5).
- ASenseOnly (Fig. 16.7(a)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersA (Fig. 16.3).
- VSenseOnly (Fig. 16.7(b)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersV (Fig. 16.5).
- TriggerAPace (Fig. 16.8(a)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersA (Fig. 16.3).
- TriggerVPace (Fig. 16.8(b)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersV (Fig. 16.5).
- InhibitAPace (Fig. 16.9(a)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersA (Fig. 16.3).
- InhibitVPace (Fig. 16.9(b)) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersV (Fig. 16.5).
- TrackASense (Fig. 16.10) adds the appropriate event markers for this operation of package Response; it uses operation SetEvMarkersA (Fig. 16.3).

16.3 Global Operations

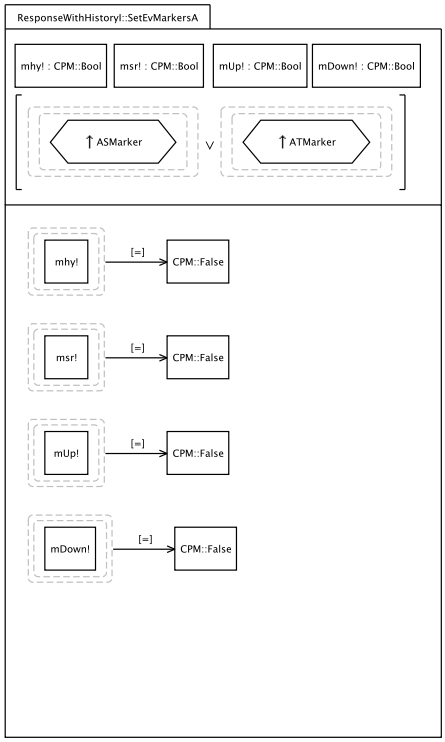
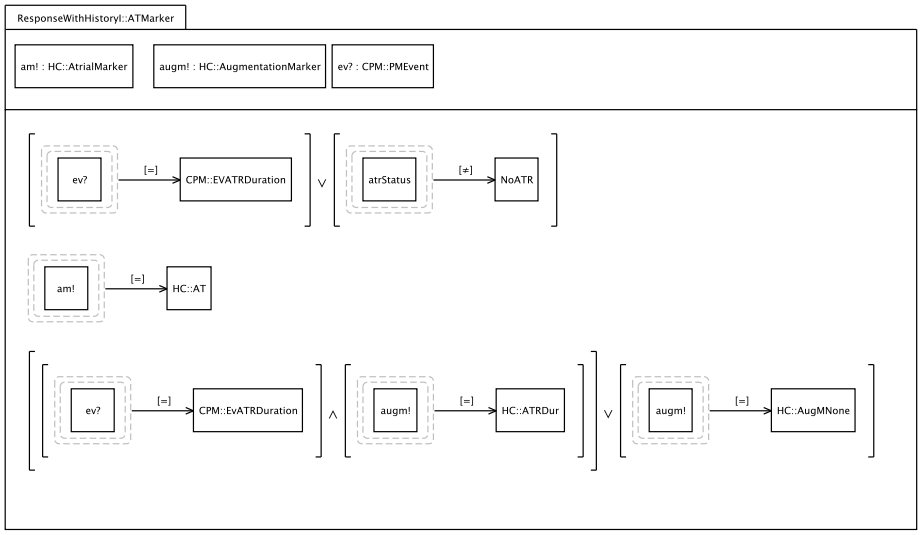
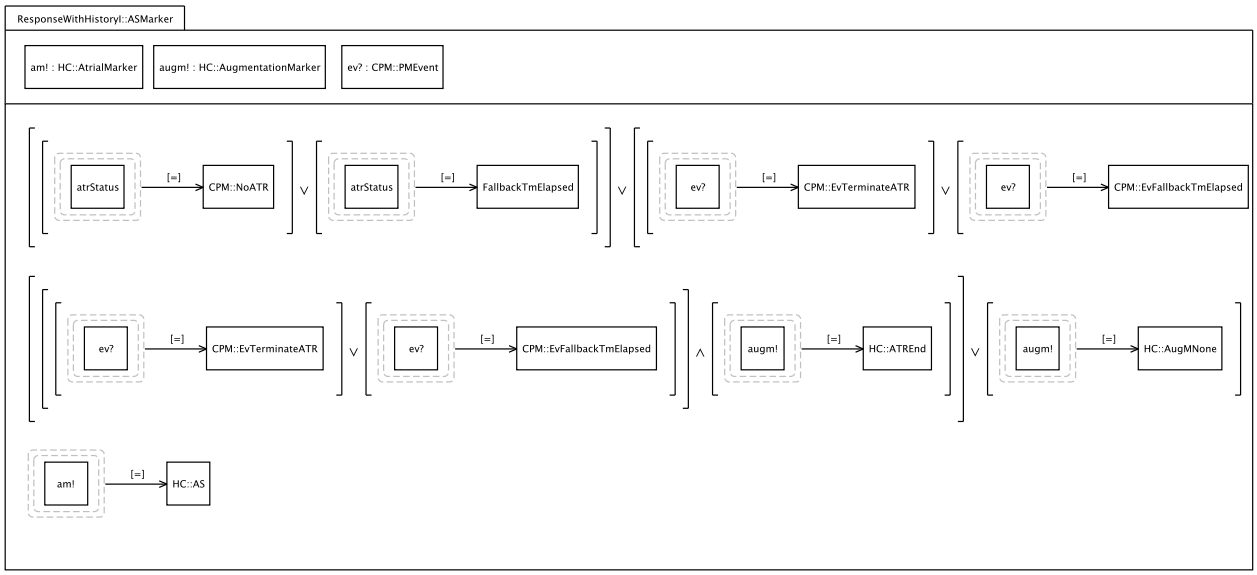


Figure 16.3: Contract diagram of operation `SetEvMarkersA` of package `ResponseWithHistoryI`



(a) `ATMarker`



(b) `ASMarker`

Figure 16.4: Assertion diagrams of operations `ATMarker` and `ASMarker` of package `ResponseWithHistoryI`

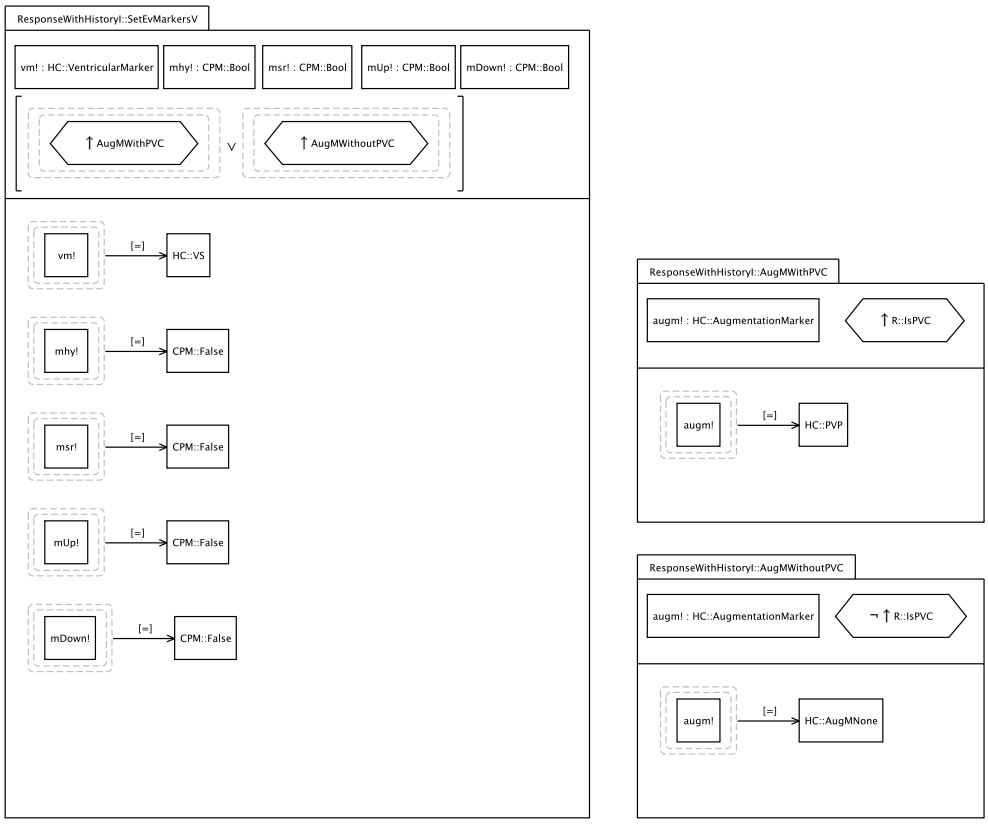
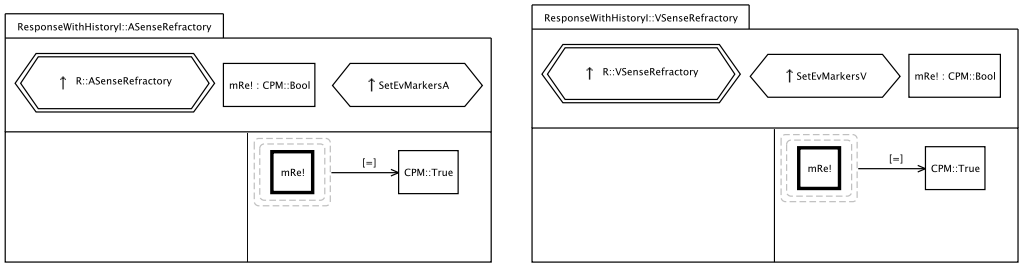
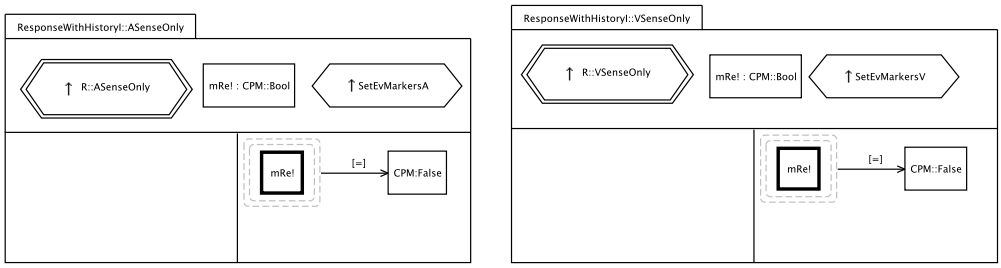


Figure 16.5: Assertion diagrams of operation `SetEvMarkersV` of package `ResponseWithHistoryI`



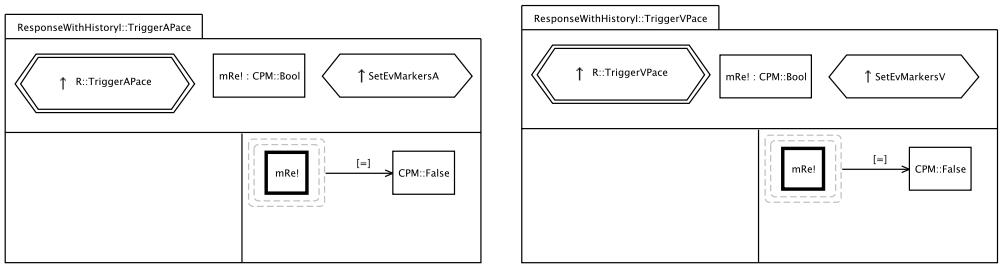
(a) `ASenseRefractory` (b) `VSenseRefractory`

Figure 16.6: Contract diagrams of operations `ASenseRefractory` and `VSenseRefractory` in package `ResponseWithHistoryI`



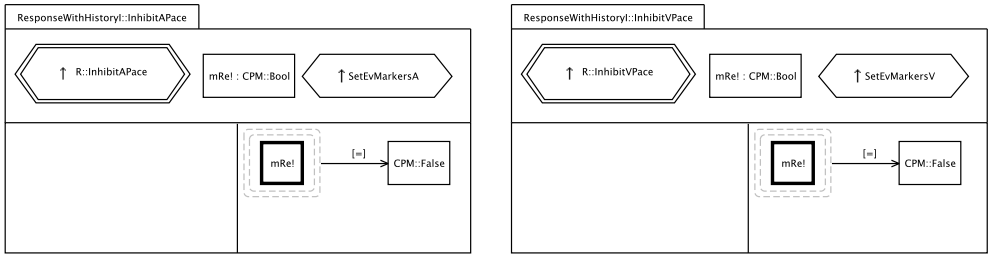
(a) `ASenseOnly` (b) `VSenseOnly`

Figure 16.7: Contract diagrams of operations `ASenseOnly` and `VSenseOnly` in package `ResponseWithHistoryI`



(a) `TriggerAPace` (b) `TriggerVPace`

Figure 16.8: Contract diagrams of operations `TriggerAPace` and `TriggerVPace` in package `ResponseWithHistoryI`



(a) `InhibitAPace` (b) `TriggerVPace`

Figure 16.9: Contract diagrams of operations `InhibitAPace` and `InhibitVPace` in package `ResponseWithHistoryI`

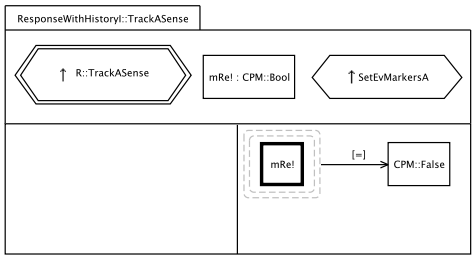


Figure 16.10: Contract diagram of operations `TrackASense` in package `ResponseWithHistoryI`

Chapter 17

The PacemakerWithoutHistory package

The package PacemakerWithoutHistory puts together sensing, pacing, response, rate modulation and battery management. This will build a package that puts together all concerns except history.

17.1 State

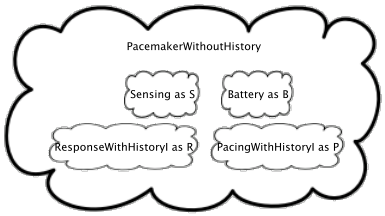


Figure 17.1: VCL Package diagram of package PacemakerWithoutHistory

The PD of PacemakerWithoutHistory is given in Fig. 17.1. It defines PacemakerWithoutHistory as an ensemble package, which imports ensemble packages Sensing, Battery, Pacing, RateModulation, ResponseWithHistoryI and PacingWithHistoryI.

17.2 Overall Behaviour

BD of package PacemakerWithoutHistory is given in Fig. 17.2. It includes one merge box to merge all operations from packages PacingWithHistoryI, Sensing, ResponseWithHistoryI, Battery and RateModulation.

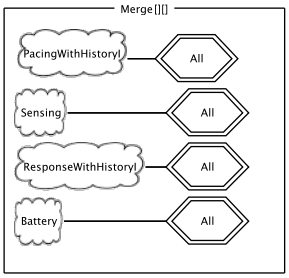


Figure 17.2: VCL behavioural diagram of package `PacemakerWithoutHistory`

Chapter 18

The Pacemaker package

The package Pacemaker puts together the concerns of sensing, pacing, response, rate modulation and battery management already composed in package PacemakerWithoutHistory with the history concern isolated in the History package.

18.1 State

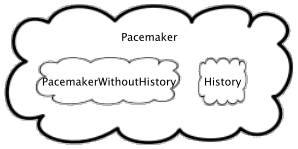


Figure 18.1: VCL Package diagram of package Pacemaker

The PD of Pacemaker is given in Fig. 18.1. It defines Pacemaker as an ensemble package, which imports ensemble packages PacemakerWithoutHistory and History.

18.2 Overall Behaviour

BD of package Pacemaker is given in Fig. 18.2. It includes one integral extension to bring operations from package PacemakerWithoutHistory; this says that all operations from package PacemakerWithoutHistory are to be imported integrally except the operations mentioned on the list associated with the All operation.

The BD of Fig. 18.2 includes two join boxes to add history to the operations that have not been imported integrally. One join box joins history for the atrial operations and the other for the ventricular operations. These operations are joined through join contracts HistoryAOp (Fig. 18.3) and HistoryVOp (Fig. 18.4).

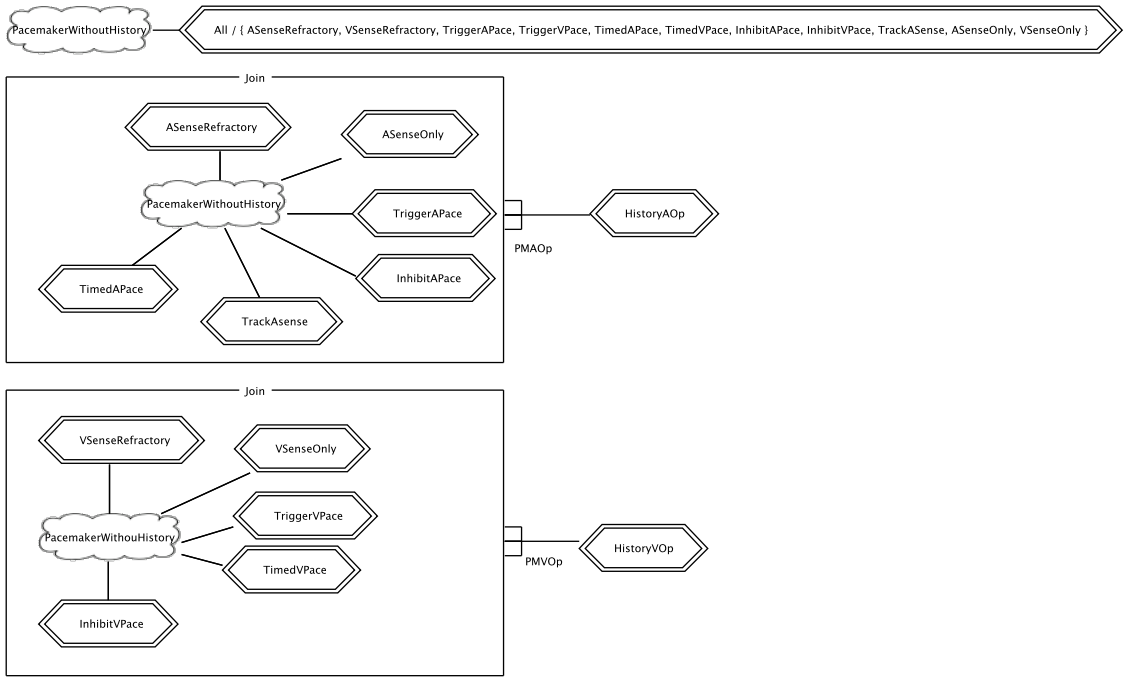


Figure 18.2: VCL behavioural diagram of package `Pacemaker`

18.3 Global Operations

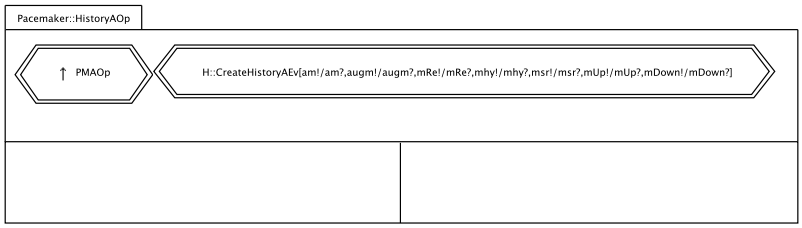


Figure 18.3: Contract diagram of operation `HistoryAOp` of package `Pacemaker`

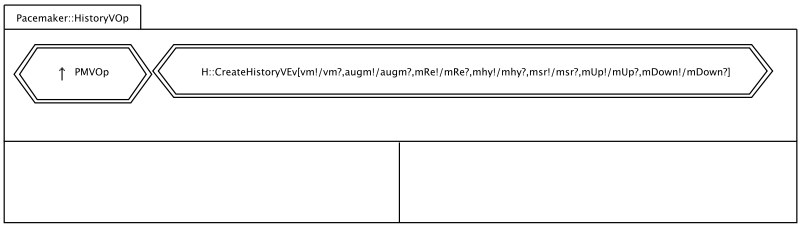
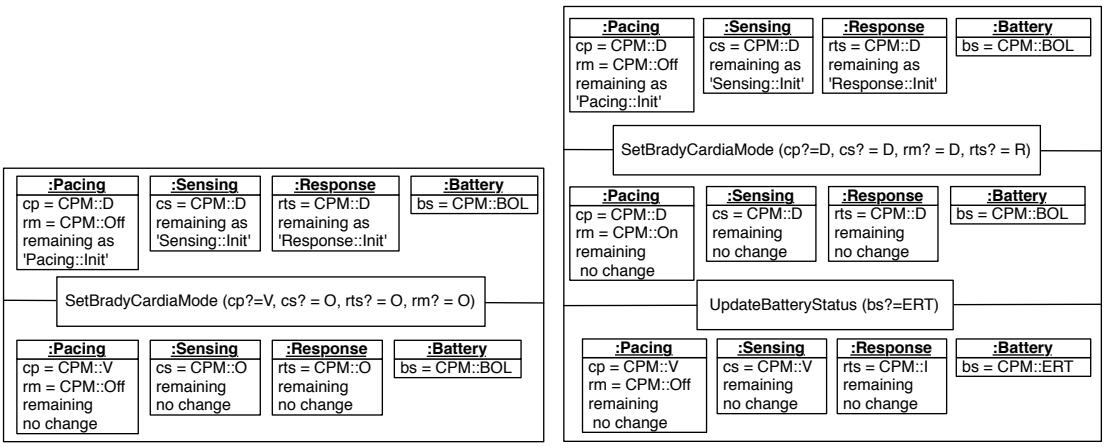


Figure 18.4: Contract diagram of operation `HistoryVOp` of package `Pacing`

Chapter 19

Snapshot Analysis

This chapter presents the snapshots that were used to analyse the pacemaker model presented in the previous chapter. This is done based on the snapshot analysis technique developed in [ASP04, Amá07].



(a) Snapshot pair: from initial `DDD` mode to `VOO`

(b) Snapshot sequence: from initial `DDD` to `DDDR` and battery `ERT`

Figure 19.1: Sample snapshots used in for model validation

Snapshot-pair of Fig. 19.1(a) simulates a change from the initial PM mode of `DDD` (dual chamber sensing and pacing) to VOO (asynchronous ventricular pacing), which is done through the operation `SetBradyCardiaMode`. The snapshot proofs for this pair are provable in the Z/Eves theorem prover, which means that the transition is valid as expected.

The snapshot sequence of Fig. 19.1(b) simulates a change of mode from `DDD` to `DDDR`, and then a situation when the battery status changes to `ERT` (Elective replacement is near), which means that the pacemaker needs to enter a degraded mode. The snapshot proofs for this snapshot sequence are provable in the Z/Eves theorem prover, which means that the sequence if valid as expected.

The snapshot sequence of Fig 19.2 simulates two tracked senses leading to the detection of a premature ventricular contraction (PVC). In the first snapshot(top), the `pvp` property is set to

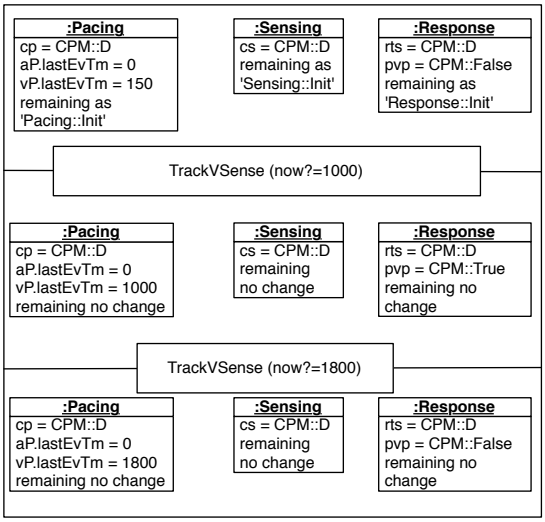


Figure 19.2: Snapshot sequence simulating two tracked senses leading to a premature ventricular contraction

`false` in `Response`, meaning that PVC has not been detected. After the first `TrackVSense`, we get the second snapshot where the `pvc` flag is set to `true` because there has been a sense and the last sense happened after a pace(see AD `IsPVC` in Fig. 11.12; $vp.lastEvTm > ap.lastEvTm$ in the first snapshot, which means that a pace did not follow a sense, as is usually expected). After the third `TrackVSense`, the flag is set to false as prescribed by the requirements.

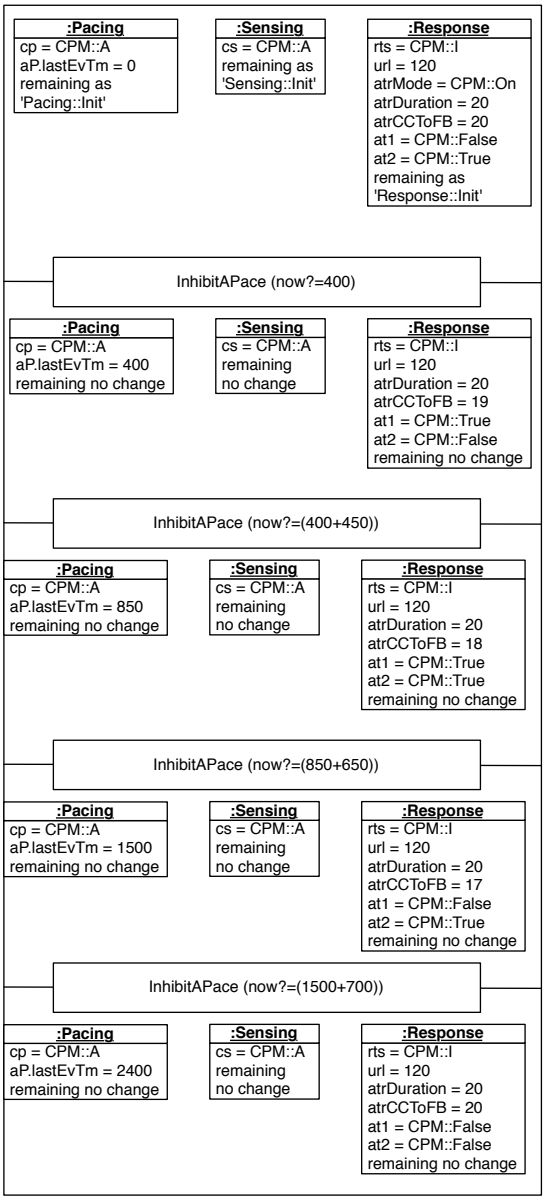


Figure 19.3: Snapshot sequence simulating the detection of atrial tachycardia

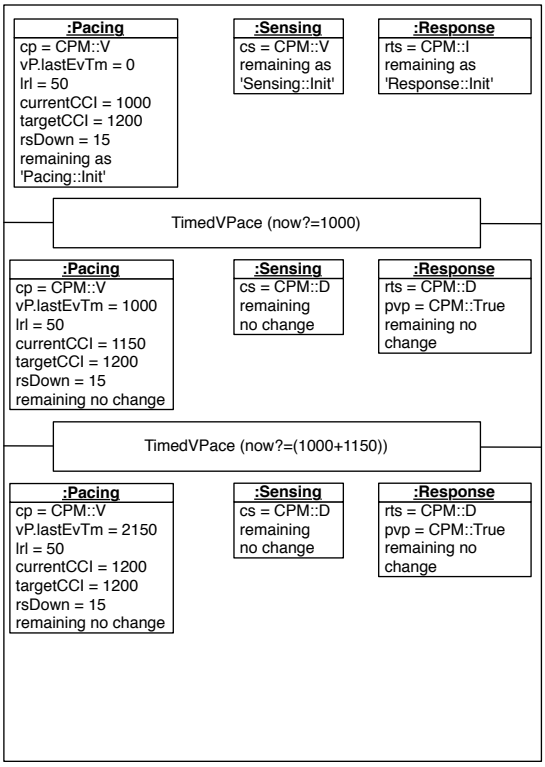


Figure 19.4: Snapshot sequence simulating rate smoothing

References

[AGK11] Nuno Amálio, Christian Glodt, and Pierre Kelsen. Building VCL models and automatically generating Z specifications from them. In *FM 2011*, LNCS. Springer, 2011.

[AK10] Nuno Amálio and Pierre Kelsen. Modular design by contract visually and formally using VCL. In *VL/HCC 2010*, 2010.

[AKMG10] Nuno Amálio, Pierre Kelsen, Qin Ma, and Christian Glodt. Using VCL as an aspect-oriented approach to requirements modelling. *TAOSD*, VII:151–199, 2010.

[Amá07] Nuno Amálio. *Generative frameworks for rigorous model-driven development*. PhD thesis, Dept. Computer Science, Univ. of York, 2007.

[APS05] Nuno Amálio, Fiona Polack, and Susan Stepney. An object-oriented structuring for Z based on views. In *ZB 2005*, volume 3455 of *LNCS*, pages 262–278. Springer, 2005.

[ASP04] Nuno Amálio, Susan Stepney, and Fiona Polack. Formal proof from UML models. In *Proc. ICFEM 2004*, volume 3308 of *LNCS*, pages 418–433. Springer, 2004.

[Bos] Boston Scientific. *ALTRUA 50 and ALTRUA 60: Multiprogrammable pacemakers*.

[Bos07a] Boston Scientific. *Advanced Timing Cycles*, 2007.

[Bos07b] Boston Scientific. Pacemaker system specification, 2007. `http://bit.ly/xWbNRM`.

[Bos07c] Boston Scientific. *Pacing Codes and Modes Concepts*, 2007.

[Bos07d] Boston Scientific. *Timing Cycles*, 2007.

[BSS10] S. Serge Barold, Roland X. Stroobandt, and Alfons F. Sinnaeve. *Cardiac Pacemakers and Resynchronization Step-by-Step: an illustrated guide*. Wiley-Blackwell, 2010.

[JIJ06] Raoul Jetley, S. Purushothaman Iyer, and Paul L. Jones. A formal methods approach to medical device review. *IEEE Computer*, 39(4), 2006.

[JJA10] Paul Jones, Raoul Jetley, and Jay Abraham. A formal methods-based verification approach to medical device software analysis. EE Times, Design, 2010. `http://bit.ly/w5guUS`.

[Lee11] Jérôme Leemans. Model a pacemaker system using VCL. Master's thesis, FSTC, Univ of Luxembourg, 2011.

[MM07] H. Weston Moses and James C. Mullin. *A practical guide to cardiac pacing*. Wolters Kluwer, 2007.

[TLMB08] Jonathon Timperley, Paul Leeson, Andrew Mitchell, and Timothy Betts. *Cardiac Pacemakers and ICDs*. Oxford University Press, 2008.

[WLBF09] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. Formal methods: Practice and experience. *ACM Computing Surveys*, 41(4):19–36, 2009.

Appendix A

Requirements Issues

The requirements of the cardiac pacemaker device modelled here are defined informally in English in [Bos07b, p. 29]. During modelling, we found many issues with the requirements, namely, omission, ambiguities and inconsistencies. These issues are summarised in table A.1.

Table A.1: Issues found in the requirements document of the Pacemaker device [Bos07b].

ID	Description
RI1	The requirements document mentions programmable minimum and maximum AV delay [Bos07b, p. 29], but the "programmable parameters table" only defines the minimum "Dynamic AV Delay" and a "Fixed AV Delay"; it does not define an interval for the allowed values of "Maximum AV Delay".
RI2	The requirements document distinguishes between paced and sensed AV Delay [Bos07b, p. 29], but there is no distinction in "programmable parameters table" [Bos07b, p. 34].
RI3	The requirements document does not provide much information about a "factor stored in device memory to create the dynamic AV delay" [Bos07b, p. 29]. Nowhere it says the value for this factor.
RI4	The requirements document misses many details regarding blanking. The requirements document only mentions "ventricular blanking" in the parameters table [Bos07b, p. 29] (it is the only mention of blanking in the document), but the requirements auxiliary documentation[1] mention both atrial and ventricular blanking.
RI5	The requirements document is not very clear on how it gets the body motion from the accelerometer, which is required for rate adaptive pacing. The requirements [Bos07b, p. 32] just say: "the device shall have the ability to adjust the cardiac cycle in response to metabolic need as measured from body motion using an accelerometer."
RI6	The requirements document is not very clear on how the rate adaptive pacing rate is calculated. No equations are provided; just a brief natural language description based on the *response factor* [Bos07b, p. 32], which is not clear to us.
RI7	The requirements document does not say what is the initial battery status. It identifies four possible status [Bos07b, p. 22], but it does not say which one is the initial one.

[1] Available at `http://sqrl.mcmaster.ca/pacemaker_docs.htm`

RI8	It is not clear what happens to the battery after it becomes depleted. If it is possible to charge it somehow, or that once depleted if it is possible that we get a status update to say that the battery becomes full again.
RI9	It is not clear how the system gets information about the battery status level of the device. The requirements [Bos07b, p. 32]just say: "the battery status for the device shall be used to predict the following battery status levels". The battery status levels are: BOL (Beginning of Life), Elective Replacement Near (ERN), Elective Replacement Time (ERT) and Elective Replacement Past (ERP). No indication is provided on how the devices reaches theses levels.
RI10	It is not clear how the device senses. All the requirements documents says about this is [Bos07b, p. 16]: "Rate Sensing shall be accomplished using bipolar electrodes and sensing circuits. All rate detection decisions shall be based in the measured cardiac cycle lengths of the sensed rhythm. Rate shall be evaluated on an interval-by-interval basis." It does not say, therefore, how sensing signals are received by the device.
RI11	The atrial tachycardia (AT) detection algorithm of [Bos07b, p. 31] is not clear. There is no information on the number of intervals between atrial senses that are faster than the URL that need to be considered for starting AT treatment ("AT onset" in [Bos07b, p. 31]). There is no information on the number of intervals between atrial senses that are slower than the URL that need to be considered for ceasing AT treatment ("AT cessation" in [Bos07b, p. 31]).There is no information on the "detection period" for starting or ceasing AT treatment [Bos07b, p. 31].
RI12	The requirements document says that the pacemaker device should switch to VVIR mode when it enters the fallback mode of atrial tachycardia detection [Bos07b, p. 33, Sec. 5.6.3]. However, this is contradictory with the requirement that says that fallback should be terminated immediately if the device detects that atrial tachycardia is over [Bos07b, p. 33, Sec. 5.6.3]. It is contradictory because cessation of atrial tachycardia is detected over the atrial channel, which is turned off in the VVIR mode. This means that the device will not be able to exit the fallback mode: it is a dead end. The consequences of this is that the patient may no longer be suffering a tachycardia, but the pacemaker device keeps responding like the patient had one.

Appendix B

Requirements Assumptions

Appendix A documents the issues that we found in the requirements document of the cardiac pacemaker device [Bos07b] modelled here. This chapter documents the requirements assumptions that we had to take concerning these issues. The assumptions are listed in table B.1. The assumptions presented here are based on the challenge's support documentation released by Boston Scientific [Bos07c, Bos07d, Bos07a], on the specification of a device (similar to the one modelled here) produced by Boston Scientific [Bos], and information on DDD pacemakers that is available in the literature [BSS10, MM07, TLMB08].

Table B.1: Requirements assumptions that resolve issues with the requirements. First column gives assumption identifier; second column gives identifier of the requirements issue; and third column describes the assumption

A1	RI1	As the requirements document does define not an interval of allowed values for the maximum AV delay, we assume that it is the same as the "fixed AV Delay" defined in the "programmable parameters table" [Bos07b, p. 34]. The rationale for this is that the most logical thing is that the maximum AV delay programmable parameter must be there (DDD pacemakers seem to have it [BSS10] and another DDD pacemaker of Boston Scientific has it [Bos]) because it would not be possible to calculate dynamic AV delays without it. We take, therefore, the values of the maximum AV delay to be the values of 'fixed AV Delay"; from [Bos] these values seem reasonable.
A2	RI2	To resolve the ambiguity between the paced and sensed AV delay, we consider that there is just one AV Delay, which is updated whenever there is an atrial event (either sensed or paced).
A3	RI3	Since we don't know the value of the "factor stored in the device memory to create the dynamic AV delay" [Bos07b, p. 29], we create in the model a natural number constant to represent this factor, where the value of this constant is left undefined.
A4	RI4	To resolve the lack of information on blanking, we decided to follow the problems's auxiliary documentation[1], which considers them. We assume that the interval of allowed values for both atrial and ventricular blanking is as defined by "ventricular blanking" in the "programmable parameters table" [Bos07b, p. 34], which are programmable by the physician.

[1] `http://sqrl.mcmaster.ca/pacemaker_docs.htm`

A5	RI5	To resolve the lack of information regarding how the accelerometer gets the body motion, we assume that the sensor will give a level of workload as defined by the activity threshold parameter of the "programmable parameters table" [Bos07b, p. 34].
A6	RI6	To resolve the lack of information regarding how rate adaptive pacing is calculated in [Bos07b, p. 34], we assume a linear equation based on LRL, MSR and response time when the rate needs to go up, and the LRL, MSR and recovery time when the rate needs to go down.
A7	RI7	We assume that the initial battery status is BOL (Beginning of Life). The device is initialized when the physician programs it; it would not make sense to put a battery in the device that is not in the state `BOL`. If the status is not correct then the operation to update the battery status is called to correct it.
A8	RI8	We assume that the battery updates tend towards degradation. We also assume that once the device is set to the special bradycardia mode after the battery reaches ERT (Elective Replacement Near), then it is not possible to change the mode unless the device is reprogrammed and the battery's level is higher than ERT.
A9	RI9	We assume that the battery sensor indicates the battery status level when it changes. The battery status level provided by the sensor is as defined in the table "battery status and therapy availability" [Bos07b, p. 22, table 3]: BOL, ERN, ERT and ERP.
A10	RI10	We assume that sense signals are amplitudes. This is consistent with pacing signals, which are also amplitudes and with what the literature says on pacemakers [BSS10]; the requirements document also say [Bos07b, p. 19]: "The device shall be capable of storing pacing thresholds and P and R wave amplitude information."
A11	RI11	We assume the programmable parameter 'Entry Count', which defines the number of atrial cycles above the URL that are required to initiate atrial tachycardia treatment. This is consistent with other Boston Scientific pacemakers that include a similar parameter, such as the Insignia [BSS10] and the ALTRUA [Bos]. The programmable values range between 1 and 8 cycles, with a nominal value of 8 (following the Insignia model [BSS10])
A12	RI11	We assume the programmable parameter 'Exit Count', which defines the number of atrial cycles below the URL that are required to terminate atrial tachycardia treatment. This is consistent with other Boston Scientific pacemakers that include a similar parameter, such as the Insignia [BSS10] and the ALTRUA [Bos]. The programmable values range between 1 and 8 cycles, with a nominal value of 8 (following the Insignia model [BSS10])
A13	RI11	Since the detection period for atrial tachycardia is not defined in [Bos07b], we introduce a constant (in milliseconds) in the model to represent this period, where the constant's value is left undefined.
A14	RI12	Having VVIR as the fallback mode means that the fallback mode will be a dead end: once it enters fallback it will not be able to exit. To solve this, we correct the fallback mode to VDIR, which is the nominal fallback mode in the Insignia [Bos07b] and ALTRUA models [Bos] (in these models the fallback mode is a programmable parameter).

Appendix C

Generated Z Specification

C.1 Preamble

section *Model_Preamble* **parents** *ZOO_Toolkit*

$CLASS ::= ChamberPacingCl \mid VentricularEventCl \mid AtrialEventCl \mid HistoryEventCl$

$$subCl : CLASS \leftrightarrow CLASS$$
$$subCl = \{(VentricularEventCl, HistoryEventCl), (AtrialEventCl, HistoryEventCl)\}$$

$$\mathbb{O} : CLASS \rightarrow \mathbb{P}_1 OBJ$$
$$\mathbb{O}x : CLASS \rightarrow \mathbb{P}_1 OBJ$$
$$\text{disjoint } \mathbb{O}x$$
$$\forall cl : CLASS \bullet \mathbb{O}\ cl = \mathbb{O}x\ cl \cup \bigcup(\mathbb{O}x\ (\!|\ (subCl^+)^\sim (\!|\{cl\}|\!)\ |\!))$$
$$\forall cl, cl' : CLASS \mid cl \mapsto cl' \in subCl \bullet \mathbb{O}\ cl \subseteq \mathbb{O}\ cl'$$

C.2 Package `CommonPM`

section *CommonPM* **parents** *ZOO_Toolkit*, *Model_Preamble*

$BradycardiaMode ::= O \mid R \mid A \mid V \mid D \mid T \mid I$

$Bool ::= True \mid False$

$Switch ::= On \mid Off$

$ChambersPaced == \{O, A, V, D\}$

$MicroSec == \mathbb{N}$

$MilliSec == \mathbb{N}$

$PPM == \mathbb{N}$

$MilliVolt == \mathbb{N}$

$MicroVolt == \mathbb{N}$

$ChambersSensed == \{O, A, V, D\}$

$ResponseToSensing == \{O, T, I, D\}$

$BatteryStatus ::= BOL \mid ERN \mid ERT \mid ERP$

$RateModulation == \{O, R\}$

C.3 Package `Battery`

section *Battery* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*

BatteryGblSt

$bs : BatteryStatus$

BatteryGbl

$BatteryGblSt$

BatteryInit

$BatteryGbl$

$bs = BOL$

BatteryIsBatteryLevelHigh

$$BatteryGbl$$

$$bs = BOL \lor bs = ERN$$

BatteryUpdateBatteryStatus

$$\Delta BatteryGbl$$
$$bs? : BatteryStatus$$

$$bs' = bs?$$

$$BatteryUpdateBatteryStatusUFr == BatteryGbl \setminus (bs)$$

$$BatteryUpdateBatteryStatusFr == \Delta BatteryGbl \land \Xi BatteryUpdateBatteryStatusUFr$$

BatteryVUpdateBatteryStatus

$$BatteryUpdateBatteryStatusFr$$
$$BatteryUpdateBatteryStatus$$

BatterySetBradycardiaMode

$$BatteryGbl$$
$$BatteryIsBatteryLevelHigh$$

BatteryVSetBradycardiaMode

$$BatterySetBradycardiaMode$$

C.4 Package `RateSmoothing`

section *RateSmoothing* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*

$$RateSmoothingValue == \{o : \mathbb{N} \mid o \leq 25\}$$

RateSmoothingGblSt

$$rsUp : RateSmoothingValue$$
$$rsDown : RateSmoothingValue$$

RateSmoothingGbl

$RateSmoothingGblSt$

RateSmoothingInit

$RateSmoothingGblSt$

$rsUp = 0$

$rsDown = 0$

RateSmoothingIncRSDownMedium

$\Delta RateSmoothingGblSt$

$rsDown \leq 18$

$rsDown' = rsDown + 3$

RateSmoothingIncRSDownHigh

$\Delta RateSmoothingGblSt$

$rsDown \geq 21$

$rsDown' = 25$

RateSmoothingIncRSDown

$\Delta RateSmoothingGbl$

$RateSmoothingIncRSDownMedium \lor RateSmoothingIncRSDownHigh$

$RateSmoothingIncRSDownUFr == RateSmoothingGbl \setminus (rsDown)$

$RateSmoothingIncRSDownFr == \Delta RateSmoothingGbl \land \Xi RateSmoothingIncRSDownUFr$

RateSmoothingVIncRSDown

$RateSmoothingIncRSDownFr$

$RateSmoothingIncRSDown$

RateSmoothingDecRSDownHigh

$\Delta RateSmoothingGblSt$

$rsDown = 25$

$rsDown' = 21$

RateSmoothingDecRSDownMedium

$\Delta RateSmoothingGblSt$

$rsDown > 0 \land rsDown \leq 21$

$rsDown' = rsDown - 3$

RateSmoothingDecRSDownLow

$\Delta RateSmoothingGblSt$

$rsDown = 0$

$rsDown' = 0$

RateSmoothingDecRSDown

$\Delta RateSmoothingGbl$

$RateSmoothingDecRSDownHigh \lor RateSmoothingDecRSDownMedium \lor RateSmoothingDecRSDownLow$

$RateSmoothingDecRSDownUFr == RateSmoothingGbl \setminus (rsDown)$

$RateSmoothingDecRSDownFr == \Delta RateSmoothingGbl \land \Xi RateSmoothingDecRSDownUFr$

RateSmoothingVDecRSDown

$RateSmoothingDecRSDownFr$

$RateSmoothingDecRSDown$

RateSmoothingIncRSUpMedium

$\Delta RateSmoothingGblSt$

$rsUp \leq 18$

$rsUp' = rsUp + 3$

RateSmoothingIncRSUpHigh

$\Delta RateSmoothingGblSt$

$rsUp \geq 21$

$rsUp' = 25$

RateSmoothingIncRSUp

$\Delta RateSmoothingGbl$

$RateSmoothingIncRSUpMedium \lor RateSmoothingIncRSUpHigh$

$RateSmoothingIncRSUpUFr == RateSmoothingGbl \setminus (rsUp)$

$RateSmoothingIncRSUpFr == \Delta RateSmoothingGbl \land \Xi RateSmoothingIncRSUpUFr$

RateSmoothingVIncRSUp

$RateSmoothingIncRSUpFr$

$RateSmoothingIncRSUp$

RateSmoothingDecRSUpHigh

$\Delta RateSmoothingGblSt$

$rsUp = 25$

$rsUp' = 21$

RateSmoothingDecRSUpMedium

$\Delta RateSmoothingGblSt$

$rsUp > 0 \land rsUp \leq 21$

$rsUp' = rsUp - 3$

RateSmoothingDecRSUpLow

$\Delta RateSmoothingGblSt$

$rsUp = 0$

$rsUp' = 0$

RateSmoothingDecRSUp

$\Delta RateSmoothingGbl$

$RateSmoothingDecRSUpHigh \lor RateSmoothingDecRSUpMedium \lor RateSmoothingDecRSUpLow$

$RateSmoothingDecRSUpUFr == RateSmoothingGbl \setminus (rsUp)$

$RateSmoothingDecRSUpFr == \Delta RateSmoothingGbl \land \Xi RateSmoothingDecRSUpUFr$

RateSmoothingVDecRSUp

$RateSmoothingDecRSUpFr$

$RateSmoothingDecRSUp$

RateSmoothingCalcUBSmoothedCCI

$RateSmoothingGblSt$

$ubCCI! : MilliSec$

$currentCCI? : MilliSec$

$ubCCI! = (currentCCI? * (100 + rsDown) \text{ div } 100)$

RateSmoothingCalcLBSmoothedCCI

$RateSmoothingGblSt$

$lbCCI! : MilliSec$

$currentCCI? : MilliSec$

$lbCCI! = (currentCCI? * (100 - rsUp) \text{ div } 100)$

RateSmoothingCalcSmoothedCCI0

$RateSmoothingGbl$

$targetCCI? : MilliSec$

$currentCCI? : MilliSec$

$newCCI! : MilliSec$

$RateSmoothingCalcUBSmoothedCCI$

$RateSmoothingCalcLBSmoothedCCI$

$(targetCCI? > currentCCI? \land targetCCI? > ubCCI! \land rsDown > 0) \Rightarrow newCCI! = ubCCI!$

$(targetCCI? < currentCCI? \land targetCCI? < lbCCI! \land rsUp > 0) \Rightarrow newCCI! = lbCCI!$

$(targetCCI? > currentCCI? \land (rsDown = 0 \lor targetCCI? \leq ubCCI!)) \Rightarrow newCCI! = targetCCI?$

$(targetCCI? \leq currentCCI? \land (rsUp = 0 \lor targetCCI? \geq lbCCI!)) \Rightarrow newCCI! = targetCCI?$

$RateSmoothingCalcSmoothedCCI == RateSmoothingCalcSmoothedCCI0 \setminus (lbCCI!, ubCCI!)$

C.5 Package `AVDelay`

section *AVDelay* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*

$dynDelayFactor : \mathbb{N}$

$MaxDynAVDelay == \{o : MilliSec \mid o \geq 70 \land o \leq 300\}$

$MinDynAVDelay == \{o : MilliSec \mid o = 30 \land o = 100\}$

$SensedAVDelayOffset == \{o : MilliSec \mid o \leq 100\}$

AVDelayGblSt

$dynMode : Switch$
$fixedAVI : MaxDynAVDelay$
$minDynDelay : MinDynAVDelay$
$savDelayOffset : SensedAVDelayOffset$
$maxDynDelay : MaxDynAVDelay$

AVDelayGbl

$AVDelayGblSt$

AVDelayInit

$AVDelayGbl$

$fixedAVI = 150$
$dynMode = Off$
$minDynDelay = 50$
$savDelayOffset = 0$
$maxDynDelay = 150$

AVDelayCalcDynDelayValue

$AVDelayGblSt$
$cdynAVI! : MilliSec$
$cci? : MilliSec$

$cdynAVI! = cci? \text{ div } dynDelayFactor$

*AVDelayCalcDynDelay*0

$$
\begin{array}{l}
AVDelayGblSt \\
dynAVI! : MilliSec \\
cci? : MilliSec \\
AVDelayCalcDynDelayValue \\
\hline
cdynAVI! \geq minDynDelay \land cdynAVI! \leq maxDynDelay \Rightarrow dynAVI! = cdynAVI! \\
cdynAVI! < minDynDelay \Rightarrow dynAVI! = minDynDelay \\
cdynAVI! > maxDynDelay \Rightarrow dynAVI! = maxDynDelay
\end{array}
$$

$$AVDelayCalcDynDelay == AVDelayCalcDynDelay0 \setminus (cdynAVI!)$$

*AVDelayGetAVDelay*0

$$
\begin{array}{l}
AVDelayGbl \\
avi! : MilliSec \\
cci? : MilliSec \\
AVDelayCalcDynDelay \\
\hline
dynMode = On \Rightarrow avi! = dynAVI! \\
dynMode = Off \Rightarrow avi! = fixedAVI
\end{array}
$$

$$AVDelayGetAVDelay == AVDelayGetAVDelay0 \setminus (dynAVI!)$$

AVDelayDynamicDelayOn

$$
\begin{array}{l}
\Delta AVDelayGbl \\
\hline
dynMode = Off \\
dynMode' = On
\end{array}
$$

$$AVDelayDynamicDelayOnUFr == AVDelayGbl \setminus (dynMode)$$

$$AVDelayDynamicDelayOnFr == \Delta AVDelayGbl \land \Xi AVDelayDynamicDelayOnUFr$$

AVDelayVDynamicDelayOn

$$
\begin{array}{l}
AVDelayDynamicDelayOnFr \\
AVDelayDynamicDelayOn
\end{array}
$$

AVDelayDynamicDelayOff

$\Delta AVDelayGbl$

$dynMode = On$

$dynMode' = Off$

$AVDelayDynamicDelayOffUFr == AVDelayGbl \setminus (dynMode)$

$AVDelayDynamicDelayOffFr == \Delta AVDelayGbl \land \Xi AVDelayDynamicDelayOffUFr$

AVDelayVDynamicDelayOff

AVDelayDynamicDelayOffFr
AVDelayDynamicDelayOff

AVDelayIncFixedAVIInc

$\Delta AVDelayGblSt$

$fixedAVI \leq 290$

$fixedAVI' = fixedAVI + 10$

AVDelayIncFixedAVISame

$\Delta AVDelayGblSt$

$fixedAVI > 290$

$fixedAVI' = 300$

AVDelayIncFixedAVI

$\Delta AVDelayGbl$

$AVDelayIncFixedAVIInc \lor AVDelayIncFixedAVISame$

$AVDelayIncFixedAVIUFr == AVDelayGbl \setminus (fixedAVI)$

$AVDelayIncFixedAVIFr == \Delta AVDelayGbl \land \Xi AVDelayIncFixedAVIUFr$

AVDelayVIncFixedAVI

AVDelayIncFixedAVIFr
AVDelayIncFixedAVI

AVDelayDecFixedAVIDec

$\Delta AVDelayGblSt$

$fixedAVI \geq 80$

$fixedAVI' = fixedAVI - 10$

AVDelayDecFixedAVISame

$\Delta AVDelayGblSt$

$fixedAVI < 80$

$fixedAVI' = 70$

AVDelayDecFixedAVI

$\Delta AVDelayGbl$

$AVDelayDecFixedAVIDec \lor AVDelayDecFixedAVISame$

$AVDelayDecFixedAVIUFr == AVDelayGbl \setminus (fixedAVI)$

$AVDelayDecFixedAVIFr == \Delta AVDelayGbl \land \Xi AVDelayDecFixedAVIUFr$

AVDelayVDecFixedAVI

$AVDelayDecFixedAVIFr$

$AVDelayDecFixedAVI$

AVDelayIncMinDynAVDelayInc

$\Delta AVDelayGblSt$

$minDynDelay \leq 90$

$minDynDelay' = minDynDelay + 10$

AVDelayIncMinDynAVDelaySame

$\Delta AVDelayGblSt$

$minDynDelay > 90$

$minDynDelay' = 100$

AVDelayIncMinDynAVDelay

$\Delta AVDelayGbl$

$AVDelayIncMinDynAVDelayInc \lor AVDelayIncMinDynAVDelaySame$

$AVDelayIncMinDynAVDelayUFr == AVDelayGbl \setminus (minDynDelay)$

$AVDelayIncMinDynAVDelayFr == \Delta AVDelayGbl \land \Xi AVDelayIncMinDynAVDelayUFr$

AVDelayVIncMinDynAVDelay

$AVDelayIncMinDynAVDelayFr$

$AVDelayIncMinDynAVDelay$

AVDelayDecMinDynAVDelayDec

$\Delta AVDelayGblSt$

$minDynDelay \geq 40$

$minDynDelay' = minDynDelay - 10$

AVDelayDecMinDynAVDelaySame

$\Delta AVDelayGblSt$

$minDynDelay < 40$

$minDynDelay' = 30$

AVDelayDecMinDynAVDelay

$\Delta AVDelayGbl$

$AVDelayDecMinDynAVDelayDec \lor AVDelayDecMinDynAVDelaySame$

$AVDelayDecMinDynAVDelayUFr == AVDelayGbl \setminus (minDynDelay)$

$AVDelayDecMinDynAVDelayFr == \Delta AVDelayGbl \land \Xi AVDelayDecMinDynAVDelayUFr$

AVDelayVDecMinDynAVDelay

$AVDelayDecMinDynAVDelayFr$

$AVDelayDecMinDynAVDelay$

AVDelayIncMaxDynAVDelayInc

$\Delta AVDelayGblSt$

$maxDynDelay \leq 290$

$maxDynDelay' = maxDynDelay + 10$

AVDelayIncMaxDynAVDelaySame

$\Delta AVDelayGblSt$

$maxDynDelay > 290$

$maxDynDelay' = 300$

AVDelayIncMaxDynAVDelay

$\Delta AVDelayGbl$

$AVDelayIncMaxDynAVDelayInc \lor AVDelayIncMaxDynAVDelaySame$

$AVDelayIncMaxDynAVDelayUFr == AVDelayGbl \setminus (maxDynDelay)$

$AVDelayIncMaxDynAVDelayFr == \Delta AVDelayGbl \land \Xi AVDelayIncMaxDynAVDelayUFr$

AVDelayVIncMaxDynAVDelay

$AVDelayIncMaxDynAVDelayFr$
$AVDelayIncMaxDynAVDelay$

AVDelayDecMaxDynAVDelayDec

$\Delta AVDelayGblSt$

$maxDynDelay \geq 80$

$maxDynDelay' = maxDynDelay - 10$

AVDelayDecMaxDynAVDelaySame

$\Delta AVDelayGblSt$

$maxDynDelay > 80$

$maxDynDelay' = 70$

AVDelayDecMaxDynAVDelay

$\Delta AVDelayGbl$

$AVDelayDecMaxDynAVDelayDec \lor AVDelayDecMaxDynAVDelaySame$

$$AVDelayDecMaxDynAVDelayUFr == AVDelayGbl \setminus (maxDynDelay)$$
$$AVDelayDecMaxDynAVDelayFr == \Delta AVDelayGbl \land \Xi AVDelayDecMaxDynAVDelayUFr$$

AVDelayVDecMaxDynAVDelay

$AVDelayDecMaxDynAVDelayFr$
$AVDelayDecMaxDynAVDelay$

AVDelayIncSAVDelayOffsetInc

$\Delta AVDelayGblSt$

$savDelayOffset \leq 90$

$savDelayOffset' = savDelayOffset + 10$

AVDelayIncSAVDelayOffsetSame

$\Delta AVDelayGblSt$

$savDelayOffset > 90$

$savDelayOffset' = 100$

AVDelayIncSAVDelayOffset

$\Delta AVDelayGbl$

$AVDelayIncSAVDelayOffsetInc \lor AVDelayIncSAVDelayOffsetSame$

$$AVDelayIncSAVDelayOffsetUFr == AVDelayGbl \setminus (savDelayOffset)$$
$$AVDelayIncSAVDelayOffsetFr == \Delta AVDelayGbl \land \Xi AVDelayIncSAVDelayOffsetUFr$$

AVDelayVIncSAVDelayOffset

$AVDelayIncSAVDelayOffsetFr$
$AVDelayIncSAVDelayOffset$

AVDelayDecSAVDelayOffsetDec

$\Delta AVDelayGblSt$

$savDelayOffset \geq 10$

$savDelayOffset' = savDelayOffset - 10$

AVDelayDecSAVDelayOffsetSame

$\Delta AVDelayGblSt$

$savDelayOffset < 10$

$savDelayOffset' = 0$

AVDelayDecSAVDelayOffset

$\Delta AVDelayGbl$

$AVDelayDecSAVDelayOffsetDec \lor AVDelayDecSAVDelayOffsetSame$

$AVDelayDecSAVDelayOffsetUFr == AVDelayGbl \setminus (savDelayOffset)$

$AVDelayDecSAVDelayOffsetFr == \Delta AVDelayGbl \land \Xi AVDelayDecSAVDelayOffsetUFr$

AVDelayVDecSAVDelayOffset

$AVDelayDecSAVDelayOffsetFr$

$AVDelayDecSAVDelayOffset$

C.6 Package `RateModulation`

section *RateModulation* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*

$MaximumSensorRate == \{o : PPM \mid o \geq 50 \land o \leq 175\}$

$ResponseFactor == \{o : \mathbb{N} \mid o \geq 1 \land o \leq 16\}$

$ReactionTime == \{o : MilliSec \mid o \geq 10000 \land o \leq 50000\}$

$RecoveryTime == \{o : MilliSec \mid o \geq 120000 \land o \leq 960000\}$

$Activity == \{o : \mathbb{N} \mid o \geq 1 \land o \leq 7\}$

$LowerRateLimit == \{o : PPM \mid o \geq 30 \land o \leq 175\}$

RateModulationGblSt

$threshold : Activity$
$msr : MaximumSensorRate$
$rm : RateModulationMode$
$responseF : ResponseFactor$
$reactionTm : ReactionTime$
$recoveryTm : RecoveryTime$
$lastSwitchTm : MilliSec$
$lastActivity : Activity$
$prm : RateModulationMode$
$targetCCI : MilliSec$
$lrl : LowerRateLimit$

RateModulationGbl

$RateModulationGblSt$

RateModulationInit

$RateModulationGbl$
$now? : MilliSec$

$msr = 120$

$responseF = 8$

$lastSwitchTm = now?$

$threshold = 4 \land lastActivity = 1$

$reactionTm = 30000 \land recoveryTm = 600000$

$rm = O \land prm = O$

$lrl = 60 \land targetCCI = 60000 \text{ div } lrl$

RateModulationSetBradycardiaMode

$\Delta RateModulationGbl$
$rm? : RateModulationMode$

$rm' = rm?$

$prm' = rm$

$RateModulationSetBradycardiaModeUFr == RateModulationGbl \setminus (rm, prm)$

$RateModulationSetBradycardiaModeFr == \Delta RateModulationGbl \land \Xi RateModulationSetBradycardiaModeUFr$

RateModulationVSetBradycardiaMode

$$RateModulationSetBradycardiaModeFr$$
$$RateModulationSetBradycardiaMode$$

RateModulationIsAboveThreshold

$$RateModulationGbl$$
$$activity? : Activity$$

$$activity? > threshold$$

RateModulationCalcTargetCCISwitchingUp

$$RateModulationGbl$$
$$activity? : Activity$$
$$targetCCI! : MilliSec$$

$$RateModulationIsAboveThreshold \land \neg\ RateModulationIsAboveThreshold[lastActivity/activity?]$$
$$targetCCI! = 60000 \text{ div } lrl$$

RateModulationCalcTargetCCISwitchingDown

$$RateModulationGbl$$
$$targetCCI! : MilliSec$$
$$activity? : Activity$$

$$\neg\ RateModulationIsAboveThreshold \land RateModulationIsAboveThreshold[lastActivity/activity?]$$
$$targetCCI! = 60000 \text{ div } msr$$

*RateModulationSetTargetCCISwitching*0

$$\Delta RateModulationGbl$$
$$now? : MilliSec$$
$$activity? : Activity$$
$$targetCCI! : MilliSec$$

$$RateModulationCalcTargetCCISwitchingUp \lor RateModulationCalcTargetCCISwitchingDown$$
$$lastSwitchTm' = now?$$
$$targetCCI' = targetCCI!$$

$$RateModulationSetTargetCCISwitching == RateModulationSetTargetCCISwitching0 \setminus (targetCCI!)$$

RateModulationCalcTargetCCIIncreasing

$RateModulationGbl$
$activity? : Activity$
$now? : MilliSec$
$targetCCI! : MilliSec$

$lastSwitchTm \leq now? - reactionTm \Rightarrow targetCCI! = 60000 \text{ div } msr$

$lastSwitchTm > now? - reactionTm \Rightarrow targetCCI! = (60000 \text{ div } lrl) + ((60000 * (now? - lastSwitchTm$

RateModulationSetTargetCCIIncreasing0

$\Delta RateModulationGbl$
$now? : MilliSec$
$activity? : Activity$
$targetCCI! : MilliSec$

$RateModulationCalcTargetCCIIncreasing$

$targetCCI' = targetCCI!$

$lastSwitchTm' = lastSwitchTm$

$RateModulationSetTargetCCIIncreasing == RateModulationSetTargetCCIIncreasing0 \setminus (targetCCI!)$

RateModulationCalcTargetCCIDecreasing

$RateModulationGbl$
$activity? : Activity$
$now? : MilliSec$
$targetCCI! : MilliSec$

$lastSwitchTm \leq now? - reactionTm \Rightarrow targetCCI! = 60000 \text{ div } lrl$

$lastSwitchTm > now? - recoveryTm \Rightarrow targetCCI! = (60000 \text{ div } msr) - ((60000 * (now? - lastSwitchT$

RateModulationSetTargetCCIDecreasing0

$\Delta RateModulationGbl$
$now? : MilliSec$
$activity? : Activity$
$targetCCI! : MilliSec$

$RateModulationCalcTargetCCIDecreasing$

$targetCCI' = targetCCI!$

$lastSwitchTm' = lastSwitchTm$

$$RateModulationSetTargetCCIDecreasing == RateModulationSetTargetCCIDecreasing0 \setminus (targetCCI!)$$

RateModulationSetTargetCCI

$\Delta RateModulationGbl$
$now? : MilliSec$
$activity? : Activity$

$RateModulationSetTargetCCISwitching \lor RateModulationSetTargetCCIIncreasing \lor RateModulationSetTarg$

$rm = R$

$lastActivity' = activity?$

$$RateModulationSetTargetCCIUFr == RateModulationGbl \setminus (lastActivity, targetCCI, lastSwitchTm)$$
$$RateModulationSetTargetCCIFr == \Delta RateModulationGbl \land \Xi RateModulationSetTargetCCIUFr$$

RateModulationVSetTargetCCI

$RateModulationSetTargetCCIFr$
$RateModulationSetTargetCCI$

RateModulationSetTargetCCIToLRL

$\Delta RateModulationGbl$

$targetCCI' = 60000 \operatorname{div} lrl$

$$RateModulationSetTargetCCIToLRLUFr == RateModulationGbl \setminus (targetCCI)$$
$$RateModulationSetTargetCCIToLRLFr == \Delta RateModulationGbl \land \Xi RateModulationSetTargetCCIToLRLUFr$$

RateModulationVSetTargetCCIToLRL

$RateModulationSetTargetCCIToLRLFr$
$RateModulationSetTargetCCIToLRL$

RateModulationIsLRLInc5

$RateModulationGblSt$

$lrl < 50 \lor lrl \geq 90 \land lrl < 175$

RateModulationIncLRL5

$\Delta RateModulationGbl$

$RateModulationIsLRLInc5$

$lrl' = lrl + 5$

RateModulationIncLRL1

$\Delta RateModulationGbl$

$lrl \geq 50 \land lrl < 90$

$lrl' = lrl + 1$

RateModulationIncLRLSame

$\Delta RateModulationGbl$

$lrl = 175$

$lrl' = lrl$

RateModulationIncLRL

$\Delta RateModulationGbl$

$RateModulationIncLRL5 \lor RateModulationIncLRL1 \lor RateModulationIncLRLSame$

$RateModulationIncLRLUFr == RateModulationGbl \setminus (lrl)$

$RateModulationIncLRLFr == \Delta RateModulationGbl \land \Xi RateModulationIncLRLUFr$

RateModulationVIncLRL

$RateModulationIncLRLFr$

$RateModulationIncLRL$

RateModulationIsLRLDec5

$RateModulationGblSt$

$lrl > 90 \lor lrl \leq 50 \land lrl > 30$

RateModulationDecLRL5

$\Delta RateModulationGbl$
$RateModulationIsLRLDec5$

$lrl' = lrl - 5$

RateModulationDecLRL1

$\Delta RateModulationGbl$

$lrl \leq 90 \land lrl > 50$

$lrl' = lrl - 1$

RateModulationDecLRLSame

$\Delta RateModulationGbl$

$lrl = 30$

$lrl' = lrl$

RateModulationDecLRL

$\Delta RateModulationGbl$

$RateModulationDecLRL5 \lor RateModulationDecLRL1 \lor RateModulationDecLRLSame$

$RateModulationDecLRLUFr == RateModulationGbl \setminus (lrl)$
$RateModulationDecLRLFr == \Delta RateModulationGbl \land \Xi RateModulationDecLRLUFr$

RateModulationVDecLRL

$RateModulationDecLRLFr$
$RateModulationDecLRL$

RateModulationIncMSRInc

$\Delta RateModulationGbl$

$msr < 175$

$msr' = msr + 5$

RateModulationIncMSRSame

$\Delta RateModulationGbl$

$msr = 175$

$msr' = msr$

$$RateModulationIncMSR == RateModulationIncMSRInc \lor RateModulationIncMSRSame$$

$$RateModulationIncMSRUFr == RateModulationGbl \setminus (msr)$$

$$RateModulationIncMSRFr == \Delta RateModulationGbl \land \Xi RateModulationIncMSRUFr$$

RateModulationVIncMSR

$$RateModulationIncMSRFr$$
$$RateModulationIncMSR$$

RateModulationDecMSRDec

$$\Delta RateModulationGbl$$

$$msr > 50$$
$$msr' = msr - 5$$

RateModulationDecMSRSame

$$\Delta RateModulationGbl$$

$$msr = 50$$
$$msr' = msr$$

$$RateModulationDecMSR == RateModulationDecMSRDec \lor RateModulationDecMSRSame$$

$$RateModulationDecMSRUFr == RateModulationGbl \setminus (msr)$$

$$RateModulationDecMSRFr == \Delta RateModulationGbl \land \Xi RateModulationDecMSRUFr$$

RateModulationVDecMSR

$$RateModulationDecMSRFr$$
$$RateModulationDecMSR$$

RateModulationIncRFInc

$\Delta RateModulationGbl$

$responseF < 16$

$responseF' = responseF + 1$

RateModulationIncRFSame

$\Delta RateModulationGbl$

$responseF = 16$

$responseF' = responseF$

$RateModulationIncRF == RateModulationIncRFInc$
$\quad \lor RateModulationIncRFSame$

$RateModulationIncRFUFr == RateModulationGbl \setminus (responseF)$

$RateModulationIncRFFr == \Delta RateModulationGbl \land \Xi RateModulationIncRFUFr$

RateModulationVIncRF

$RateModulationIncRFFr$
$RateModulationIncRF$

RateModulationDecRFDec

$\Delta RateModulationGbl$

$responseF > 1$

$responseF' = responseF - 1$

RateModulationDecRFSame

$\Delta RateModulationGbl$

$responseF = 1$

$responseF' = responseF$

$RateModulationDecRF == RateModulationDecRFDec$
$\quad \lor RateModulationDecRFSame$

$$RateModulationDecRFUFr == RateModulationGbl \setminus (responseF)$$
$$RateModulationDecRFFr == \Delta RateModulationGbl \land \Xi RateModulationDecRFUFr$$

RateModulationVDecRF

$$RateModulationDecRFFr$$
$$RateModulationDecRF$$

RateModulationIncReactionTmInc

$$\Delta RateModulationGbl$$

$$reactionTm < 50000$$
$$reactionTm' = reactionTm + 10000$$

RateModulationIncReactionTmSame

$$\Delta RateModulationGbl$$

$$reactionTm = 50000$$
$$reactionTm' = reactionTm$$

$$RateModulationIncReactionTm == RateModulationIncReactionTmInc \lor RateModulationIncReactionTmSame$$

$$RateModulationIncReactionTmUFr == RateModulationGbl \setminus (reactionTm)$$
$$RateModulationIncReactionTmFr == \Delta RateModulationGbl \land \Xi RateModulationIncReactionTmUFr$$

RateModulationVIncReactionTm

$$RateModulationIncReactionTmFr$$
$$RateModulationIncReactionTm$$

RateModulationDecReactionTmDec

$$\Delta RateModulationGbl$$

$$reactionTm > 10000$$
$$reactionTm' = reactionTm - 10000$$

RateModulationDecReactionTmSame

$\Delta RateModulationGbl$

$reactionTm = 10000$

$reactionTm' = reactionTm$

$RateModulationDecReactionTm == RateModulationDecReactionTmDec$
$\quad \lor RateModulationDecReactionTmSame$

$RateModulationDecReactionTmUFr == RateModulationGbl \setminus (reactionTm)$

$RateModulationDecReactionTmFr == \Delta RateModulationGbl \land \Xi RateModulationDecReactionTmUFr$

RateModulationVDecReactionTm

$RateModulationDecReactionTmFr$
$RateModulationDecReactionTm$

RateModulationIncRecoveryTmInc

$\Delta RateModulationGbl$

$recoveryTm < 960000$

$recoveryTm' = recoveryTm + 60000$

RateModulationIncRecoveryTmSame

$\Delta RateModulationGbl$

$recoveryTm = 960000$

$recoveryTm' = recoveryTm$

$RateModulationIncRecoveryTm == RateModulationIncRecoveryTmInc$
$\quad \lor RateModulationIncRecoveryTmSame$

$RateModulationIncRecoveryTmUFr == RateModulationGbl \setminus (recoveryTm)$

$RateModulationIncRecoveryTmFr == \Delta RateModulationGbl \land \Xi RateModulationIncRecoveryTmUFr$

RateModulationVIncRecoveryTm

$RateModulationIncRecoveryTmFr$

$RateModulationIncRecoveryTm$

RateModulationDecRecoveryTmDec

$\Delta RateModulationGbl$

$recoveryTm > 120000$

$recoveryTm' = recoveryTm - 60000$

RateModulationDecRecoveryTmSame

$\Delta RateModulationGbl$

$recoveryTm = 120000$

$recoveryTm' = recoveryTm$

$RateModulationDecRecoveryTm == RateModulationDecRecoveryTmDec$
$\quad \lor RateModulationDecRecoveryTmSame$

$RateModulationDecRecoveryTmUFr == RateModulationGbl \setminus (recoveryTm)$

$RateModulationDecRecoveryTmFr == \Delta RateModulationGbl \land \Xi RateModulationDecRecoveryTmUFr$

RateModulationVDecRecoveryTm

$RateModulationDecRecoveryTmFr$

$RateModulationDecRecoveryTm$

RateModulationIncActThresholdInc

$\Delta RateModulationGbl$

$threshold < 7$

$threshold' = threshold + 1$

RateModulationIncActThresholdSame

$\Delta RateModulationGbl$

$threshold = 7$

$threshold' = threshold$

$RateModulationIncActThreshold == RateModulationIncActThresholdInc \lor RateModulationIncActThresholdSame$

$RateModulationIncActThresholdUFr == RateModulationGbl \setminus (threshold)$

$RateModulationIncActThresholdFr == \Delta RateModulationGbl \land \Xi RateModulationIncActThresholdUFr$

RateModulationVIncActThreshold

$RateModulationIncActThresholdFr$

$RateModulationIncActThreshold$

RateModulationDecActThresholdDec

$\Delta RateModulationGbl$

$threshold > 1$

$threshold' = threshold - 1$

RateModulationDecActThresholdSame

$\Delta RateModulationGbl$

$threshold = 1$

$threshold' = threshold$

$RateModulationDecActThreshold == RateModulationDecActThresholdDec \lor RateModulationDecActThresholdSame$

$RateModulationDecActThresholdUFr == RateModulationGbl \setminus (threshold)$

$RateModulationDecActThresholdFr == \Delta RateModulationGbl \land \Xi RateModulationDecActThresholdUFr$

RateModulationVDecActThreshold

$RateModulationDecActThresholdFr$

$RateModulationDecActThreshold$

C.7 Package `Pacing`

section *Pacing* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*, *RateSmoothing*, *AVDelay*, *RateModulat*

$$PulseWidth == \{o : MicroSec \mid o \geq 50 \land o \leq 1900\}$$

$$PulseAmplitudeMode ::= Regulated \mid Unregulated$$

$$PulseAmplitudeUnregulated == \{o : MilliVolt \mid o \geq 1250 \land o = 5000\}$$

$$PulseAmplitudeRegulated == \{o : MilliVolt \mid o \geq 500 \land o \leq 7000\}$$

*ChamberPacing*0

$$amplMode : PulseAmplitudeMode$$
$$pulseWidth : PulseWidth$$
$$amplReg : PulseAmplitudeRegulated$$
$$amplUReg : PulseAmplitudeUnregulated$$
$$lastEvTm : MilliSec$$

ChamberPacing

$$ChamberPacing0$$

SChamberPacing

$$sChamberPacing : \mathbb{P}(\mathbb{O}\ ChamberPacingCl)$$
$$stChamberPacing : \mathbb{O}\ ChamberPacingCl \pfun ChamberPacing$$

$$\text{dom}\ stChamberPacing = sChamberPacing$$

SChamberPacingInit

$$SChamberPacing$$

$$sChamberPacing = \varnothing \land stChamberPacing = \varnothing$$

ChamberPacingNew

$ChamberPacing$
$now? : MilliSec$

$amplMode = Unregulated$

$amplReg = 3500$

$amplUReg = 3750$

$pulseWidth = 400$

$lastEvTm = now?$

ChamberPacingSetLastEventTm

$\Delta ChamberPacing$
$now? : MilliSec$

$amplMode' = amplMode'$

$amplReg' = amplReg'$

$amplUReg' = amplUReg'$

$pulseWidth' = pulseWidth$

$lastEvTm' = now?$

ChamberPacingIncPulseAmplRegLow

$\Delta ChamberPacing$

$amplReg < 3500$

$amplReg' = amplReg + 100$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseAmplRegMedium

$\Delta ChamberPacing$

$amplReg \geq 3500 \land amplReg < 7000$

$amplReg' = amplReg + 500$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseAmplRegSame

$\Delta ChamberPacing$

$amplReg = 7000$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseAmplReg

$\Delta ChamberPacing$

$ChamberPacingIncPulseAmplRegLow \lor ChamberPacingIncPulseAmplRegMedium \lor ChamberPacingIncPuls$

ChamberPacingDecPulseAmplRegLow

$\Delta ChamberPacing$

$amplReg > 500 \land amplReg \leq 3500$

$amplReg' = amplReg - 100$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseAmplRegMedium

$\Delta ChamberPacing$

$amplReg > 3500 \land amplReg \leq 7000$

$amplReg' = amplReg - 500$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseAmplRegSame

$\Delta ChamberPacing$

$amplReg = 500$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseAmplReg

$\Delta ChamberPacing$

$ChamberPacingDecPulseAmplRegLow \lor ChamberPacingDecPulseAmplRegMedium \lor ChamberPacingDecPu$

ChamberPacingIncPulseAmplURegInc

$\Delta ChamberPacing$

$amplUReg < 5000$

$amplUReg' = amplUReg + 1250$

$amplMode' = amplMode$

$amplReg' = amplReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseAmplURegSame

$\Delta ChamberPacing$

$amplUReg = 5000$

$amplUReg' = amplUReg$

$amplMode' = amplMode$

$amplReg' = amplReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseAmplUReg

$\Delta ChamberPacing$

$ChamberPacingIncPulseAmplURegInc \lor ChamberPacingIncPulseAmplURegSame$

ChamberPacingDecPulseAmplURegDec

$\Delta ChamberPacing$

$amplUReg > 1250$

$amplUReg' = amplUReg - 1250$

$amplMode' = amplMode$

$amplReg' = amplReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseAmplURegSame

$\Delta ChamberPacing$

$amplUReg = 1250$

$amplUReg' = amplUReg$

$amplMode' = amplMode$

$amplReg' = amplReg$

$pulseWidth' = pulseWidth$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseAmplUReg

$\Delta ChamberPacing$

$ChamberPacingDecPulseAmplURegDec \lor ChamberPacingDecPulseAmplURegSame$

ChamberPacingIncPulseWidthLow

$\Delta ChamberPacing$

$pulseWidth < 100$

$pulseWidth' = pulseWidth + 50$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseWidthMedium

$\Delta ChamberPacing$

$pulseWidth \geq 100 \land pulseWidth < 1900$

$pulseWidth' = pulseWidth + 100$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseWidthSame

$\Delta ChamberPacing$

$pulseWidth = 1900$

$pulseWidth' = pulseWidth$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$lastEvTm' = lastEvTm$

ChamberPacingIncPulseWidth

$\Delta ChamberPacing$

$ChamberPacingIncPulseWidthLow \lor ChamberPacingIncPulseWidthMedium \lor ChamberPacingIncPulseWidt$

ChamberPacingDecPulseWidthHigh

$\Delta ChamberPacing$

$pulseWidth \leq 1900 \land pulseWidth > 100$

$pulseWidth' = pulseWidth - 100$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseWidthLow

$\Delta ChamberPacing$

$pulseWidth \leq 100$

$pulseWidth' = pulseWidth - 50$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseWidthSame

$\Delta ChamberPacing$

$pulseWidth = 50$

$pulseWidth' = pulseWidth$

$amplReg' = amplReg$

$amplMode' = amplMode$

$amplUReg' = amplUReg$

$lastEvTm' = lastEvTm$

ChamberPacingDecPulseWidth

$\Delta ChamberPacing$

$ChamberPacingDecPulseWidthHigh \lor ChamberPacingDecPulseWidthLow \lor ChamberPacingDecPulseWidth$

ChamberPacingGetPulseAmpl

$ChamberPacing$
$pulseAmpl! : MilliVolt$

$amplMode = Regulated \Rightarrow pulseAmpl! = amplReg$

$amplMode = Unregulated \Rightarrow pulseAmpl! = amplUReg$

ChamberPacingGetPulseWidth

$ChamberPacing$
$pulseWidth! : MicroSec$

$pulseWidth! = pulseWidth$

ChamberPacingGetLastEventTm

$ChamberPacing$
$lastEvTm! : MilliSec$

$lastEvTm! = lastEvTm$

SChamberPacingNF

$\Delta SChamberPacing$
$ChamberPacing$
$o! : \mathbb{O}\, ChamberPacingCl$

$o! \in \mathbb{O}xChamberPacingCl \setminus sChamberPacing$

$sChamberPacing' = sChamberPacing' \cup \{o!\}$

$stChamberPacing' = stChamberPacing' \cup \{(o!, \theta\ ChamberPacing)\}$

SChamberPacingUF

$\Delta SChamberPacing$
$\Delta ChamberPacing$
$o? : \mathbb{O}x\ ChamberPacingCl$

$o? \in sChamberPacing$

$\theta\ ChamberPacing = stChamberPacing\ o?$

$sChamberPacing' = sChamberPacing$

$stChamberPacing' = stChamberPacing' \oplus \{(o?, \theta\ ChamberPacing')\}$

SChamberPacingOF

$$
\begin{array}{l}
SChamberPacing \\
ChamberPacing \\
o? : \mathbb{O}\, ChamberPacingCl \\
\hline
o? \in sChamberPacing \\
\theta\ ChamberPacing = stChamberPacing\ o?
\end{array}
$$

$$SChamberPacingNew == \exists\ ChamberPacing \bullet SChamberPacingNF[cp!/o!] \land ChamberPacingNew$$

$$SChamberPacingSetLastEventTm == \exists\ \Delta ChamberPacing \bullet SChamberPacingUF \land ChamberPacingSetLastEventTm$$

$$SChamberPacingIncPulseAmplReg == \exists\ \Delta ChamberPacing \bullet SChamberPacingUF \land ChamberPacingIncPulseAmplReg$$

$$SChamberPacingDecPulseAmplReg == \exists\ \Delta ChamberPacing \bullet SChamberPacingUF \land ChamberPacingDecPulseAmplReg$$

$$SChamberPacingIncPulseAmplUReg == \exists\ \Delta ChamberPacing \bullet SChamberPacingUF \land ChamberPacingIncPulseAmplUReg$$

$$SChamberPacingDecPulseAmplUReg == \exists\ \Delta ChamberPacing \bullet SChamberPacingUF \land ChamberPacingDecPulseAmplUReg$$

$$SChamberPacingIncPulseWidth == \exists\ \Delta ChamberPacing \bullet SChamberPacingUF \land ChamberPacingIncPulseWidth$$

$$SChamberPacingDecPulseWidth == \exists\ \Delta ChamberPacing \bullet SChamberPacingUF \land ChamberPacingDecPulseWidth$$

$$SChamberPacingGetLastEventTm == \exists\ ChamberPacing \bullet SChamberPacingOF \land ChamberPacingGetLastEventTm$$

$$SChamberPacingGetPulseWidth == \exists\ ChamberPacing \bullet SChamberPacingOF \land ChamberPacingGetPulseWidth$$

$$SChamberPacingGetPulseAmpl == \exists\ ChamberPacing \bullet SChamberPacingOF \land ChamberPacingGetPulseAmpl$$

PacingPkgSt

$hystMode : Switch$
$hrl : LowerRateLimit$
$aP : \mathbb{O}\ ChamberPacingCl$
$vP : \mathbb{O}\ ChamberPacingCl$
$cp : ChambersPaced$
$currentCCI : MilliSec$
$pcp : ChambersPaced$

PacingOnlyGblSt

$PacingPkgSt$
$SChamberPacing$

PacingGblSt

$PacingOnlyGblSt$
$AVDelayGbl$
$RateSmoothingGbl$
$RateModulationGbl$

PacingTargetCCIInv

$PacingGblSt$

$rm = O \Rightarrow targetCCI \leq 60000 \text{ div } lrl$

$(rm = O \land hystMode = On) \Rightarrow targetCCI = 60000 \text{ div } hrl$

$(rm = R \land hystMode = Off) \Rightarrow targetCCI = 60000 \text{ div } lrl$

PacingGbl

$PacingGblSt$

$PacingTargetCCIInv$

$PacingIntExtRateSmoothingFr == \Delta PacingGbl \land \Xi PacingOnlyGblSt \land \Xi AVDelayGbl \land \Xi RateModulationGbl$

PacingVIncRSDown

$PacingIntExtRateSmoothingFr$
$RateSmoothingVIncRSDown$

PacingVDecRSDown

PacingIntExtRateSmoothingFr
RateSmoothingVDecRSDown

PacingVIncRSUp

PacingIntExtRateSmoothingFr
RateSmoothingVIncRSUp

PacingVDecRSUp

PacingIntExtRateSmoothingFr
RateSmoothingVIncRSUp

$$PacingIntExtAvDelayFr == \Delta PacingGbl \land \Xi PacingOnlyGblSt \land \Xi RateSmoothingGbl \land \Xi RateModulationGbl$$

PacingVDynamicDelayOn

PacingIntExtAvDelayFr
AVDelayVDynamicDelayOn

PacingVDynamicDelayOff

PacingIntExtAvDelayFr
AVDelayVDynamicDelayOff

PacingVIncFixedAVI

PacingIntExtAvDelayFr
AVDelayIncFixedAVI

PacingVDecFixedAVI

PacingIntExtAvDelayFr
AVDelayVDecFixedAVI

PacingVIncMinDynAVDelay

PacingIntExtAvDelayFr
AVDelayVIncMinDynAVDelay

PacingVDecMinDynAVDelay

PacingIntExtAvDelayFr
AVDelayVDecMinDynAVDelay

PacingVIncMaxDynAVDelay

PacingIntExtAvDelayFr
AVDelayVIncMaxDynAVDelay

PacingVDecMaxDynAVDelay

PacingIntExtAvDelayFr
AVDelayVDecMaxDynAVDelay

PacingVIncSAVDelayOffset

PacingIntExtAvDelayFr
AVDelayVIncSAVDelayOffset

PacingVDecSAVDelayOffset

PacingIntExtAvDelayFr
AVDelayVDecSAVDelayOffset

$$PacingIntExtRateModulationFr == \Delta PacingGbl \land \Xi PacingOnlyGblSt \land \Xi RateSmoothingGbl \land \Xi AVDelayGbl$$

PacingVSetTargetCCI

PacingIntExtRateModulationFr
RateModulationVSetTargetCCI

PacingVIncLRL

PacingIntExtRateModulationFr
RateModulationVIncLRL

PacingVDecLRL

PacingIntExtRateModulationFr
RateModulationVDecLRL

PacingVIncMSR

PacingIntExtRateModulationFr
RateModulationVIncMSR

PacingVDecMSR

PacingIntExtRateModulationFr
RateModulationVDecMSR

PacingVIncRF

PacingIntExtRateModulationFr
RateModulationVIncRF

PacingVDecRF

PacingIntExtRateModulationFr
RateModulationVDecRF

PacingVIncReactionTm

PacingIntExtRateModulationFr
RateModulationVIncReactionTm

PacingVDecReactionTm

PacingIntExtRateModulationFr
RateModulationVDecReactionTm

PacingVIncRecoveryTm

PacingIntExtRateModulationFr
RateModulationVIncRecoveryTm

PacingVDecRecoveryTm

$PacingIntExtRateModulationFr$
$RateModulationVDecRecoveryTm$

PacingVIncActThreshold

$PacingIntExtRateModulationFr$
$RateModulationVIncActThreshold$

PacingVDecActThreshold

$PacingIntExtRateModulationFr$
$RateModulationVDecActThreshold$

*PacingInit*0

$PacingGbl$
$SChamberPacingInit$

PacingInit

$\Delta PacingGbl$
$PacingInit0$
$AVDelayInit$
$RateSmoothingInit$
$RateModulationInit$
$now? : MilliSec$
$SChamberPacingNew[cpa!/cp!] \fatsemi SChamberPacingNew[cpv!/cp!]$

$hystMode' = Off$

$hrl' = 60$

$currentCCI' = targetCCI$

$cp' = D$

$aP' = cpa!$

$vP' = cpv!$

PacingSetBradycardiaMode

$\Delta PacingGbl$
$cp? : ChambersPaced$
$RateModulationSetBradycardiaMode$

$cp' = cp?$

$pcp' = cp$

$PacingSetBradycardiaModeUFr == PacingGbl \setminus (cp, pcp, rm, prm)$

$PacingSetBradycardiaModeFr == \Delta PacingGbl \land \Xi PacingSetBradycardiaModeUFr$

PacingVSetBradycardiaMode

$PacingSetBradycardiaModeFr$
$PacingSetBradycardiaMode$

PacingIsBatteryDegraded

$PacingGblSt$
$bs? : BatteryStatus$

$bs? = ERT \lor bs? = ERP$

PacingUpdateBatteryChangeMode

$\Delta PacingGblSt$
$PacingIsBatteryDegraded$
$PacingSetBradycardiaMode[V/cp?, R/rm?]$

PacingUpdateBatterySame

$\Delta PacingGblSt$

$cp' = cp$

$rm' = rm$

$prm' = prm$

$pcp' = pcp$

PacingUpdateBatteryStatus

$\Delta PacingGbl$
$bs? : BatteryStatus$

$PacingUpdateBatteryChangeMode \lor PacingUpdateBatterySame$

$$PacingUpdateBatteryStatusUFr == PacingGbl \setminus (cp, pcp, rm, prm)$$

$$PacingUpdateBatteryStatusFr == \Delta PacingGbl \land \Xi PacingUpdateBatteryStatusUFr$$

PacingVUpdateBatteryStatus

$PacingUpdateBatteryStatusFr$
$PacingUpdateBatteryStatus$

PacingIncAPulseAmplReg

$\Delta PacingGbl$
$SChamberPacingIncPulseAmplReg[aP/o?]$

PacingVIncAPulseAmplReg

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingIncAPulseAmplReg$

PacingDecAPulseAmplReg

$\Delta PacingGbl$
$SChamberPacingDecPulseAmplReg[aP/o?]$

PacingVDecAPulseAmplReg

$\Delta PacingGbl$
$\Xi PacingOnlyGblSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingDecAPulseAmplReg$

PacingIncVPulseAmplReg

$\Delta PacingGbl$
$SChamberPacingIncPulseAmplReg[vP/o?]$

PacingVIncVPulseAmplReg

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingIncVPulseAmplReg$

PacingDecVPulseAmplReg

$\Delta PacingGbl$
$SChamberPacingDecPulseAmplReg[vP/o?]$

PacingVDecVPulseAmplReg

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingDecVPulseAmplReg$

PacingIncAPulseAmplUReg

$\Delta PacingGbl$
$SChamberPacingIncPulseAmplUReg[aP/o?]$

PacingVIncAPulseAmplUReg

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingIncAPulseAmplUReg$

PacingDecAPulseAmplUReg

$\Delta PacingGbl$
$SChamberPacingDecPulseAmplUReg[aP/o?]$

PacingVDecAPulseAmplUReg

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingDecAPulseAmplUReg$

PacingIncVPulseAmplUReg

$\Delta PacingGbl$
$SChamberPacingIncPulseAmplUReg[vP/o?]$

PacingVIncVPulseAmplUReg

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingIncVPulseAmplUReg$

PacingDecVPulseAmplUReg

$\Delta PacingGbl$
$SChamberPacingDecPulseAmplUReg[vP/o?]$

PacingVDecVPulseAmplUReg

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingDecVPulseAmplUReg$

PacingIncAPulseWidth

$\Delta PacingGbl$
$SChamberPacingIncPulseWidth[aP/o?]$

PacingVIncAPulseWidth

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingIncAPulseWidth$

PacingDecAPulseWidth

$\Delta PacingGbl$
$SChamberPacingDecPulseWidth[aP/o?]$

PacingVDecAPulseWidth

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingDecAPulseWidth$

PacingIncVPulseWidth

$\Delta PacingGbl$
$SChamberPacingIncPulseWidth[vP/o?]$

PacingVIncVPulseWidth

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingIncVPulseWidth$

PacingDecVPulseWidth

$\Delta PacingGbl$
$SChamberPacingDecPulseWidth[vP/o?]$

PacingVDecVPulseWidth

$\Delta PacingGbl$
$\Xi PacingPkgSt$
$\Xi AVDelayGbl$
$\Xi RateSmoothingGbl$
$PacingDecVPulseWidth$

*PacingIsHRLInc*5

$PacingGblSt$

$hrl < 50 \lor hrl \geq 90 \land hrl < 175$

*PacingIncHRL*5

$\Delta PacingGbl$
$PacingIsHRLInc5$

$hrl' = hrl + 5$

*PacingIncHRL*1

$\Delta PacingGbl$

$hrl \geq 50 \land hrl < 90$

$hrl' = hrl + 1$

PacingIncHRLSame

$\Delta PacingGbl$

$hrl = 175$

$hrl' = hrl$

PacingIncHRL

$\Delta PacingGbl$

$PacingIncHRL5 \lor PacingIncHRL1 \lor PacingIncHRLSame$

$PacingIncHRLUFr == PacingGbl \setminus (hrl)$

$PacingIncHRLFr == \Delta PacingGbl \land \Xi PacingIncHRLUFr$

PacingVIncHRL

$PacingIncHRLFr$

$PacingIncHRL$

*PacingIsHRLDec*5

$PacingGblSt$

$hrl > 90 \lor hrl \leq 50 \land hrl > 30$

*PacingDecHRL*5

$\Delta PacingGbl$

$PacingIsHRLDec5$

$hrl' = hrl - 5$

*PacingDecHRL*1

$\Delta PacingGbl$

$hrl \leq 90 \land hrl > 50$

$hrl' = hrl - 1$

PacingDecHRLSame

$\Delta PacingGbl$

$hrl = 30$

$hrl' = hrl$

PacingDecHRL

$\Delta PacingGbl$

$PacingDecHRL5 \lor PacingDecHRL1 \lor PacingDecHRLSame$

$PacingDecHRLUFr == PacingGbl \setminus (hrl)$

$PacingDecHRLFr == \Delta PacingGbl \land \Xi PacingDecHRLUFr$

PacingVDecHRL

$PacingDecHRLFr$

$PacingDecHRL$

PacingUpdateCCI0

$\Delta PacingGbl$

$RateSmoothingCalcSmoothedCCI[targetCCI/targetCCI?, currentCCI/currentCCI?]$

$currentCCI' = newCCI!$

$PacingUpdateCCI == \exists\, newCCI! : MilliSec \bullet PacingUpdateCCI0$

PacingCurrentCCISame

$\Delta PacingGbl$

$currentCCI' = currentCCI$

PacingAPaceEnabled

$PacingGbl$

$cp = A \lor cp = D$

PacingDoesARateSmoothing

$PacingGbl$
$PacingAPaceEnabled$

$\neg\ (cp\ =\ D)$

PacingUpdateCCIIfDoesRateSmoothing

$\Delta PacingGbl$

$(PacingDoesARateSmoothing \land PacingUpdateCCI) \lor PacingCurrentCCISame$

PacingAPaceCommon

$\Delta PacingGbl$
$now? : MilliSec$
$PacingAPaceEnabled$

$SChamberPacingSetLastEventTm[aP/o?]$
$PacingUpdateCCIIfDoesRateSmoothing$

PacingVPaceEnabled

$PacingGblSt$

$cp\ =\ V \lor cp\ =\ D$

PacingVPaceCommon

$\Delta PacingGbl$
$now? : MilliSec$
$PacingVPaceEnabled$

$SChamberPacingSetLastEventTm[vP/o?]$
$PacingUpdateCCI$

PacingTriggerAPace

$\Delta PacingGbl$
$now? : MilliSec$
$aPulseAmpl! : MilliVolt$
$aPulseWidth! : MicroSec$
$PacingAPaceCommon$

$SChamberPacingGetPulseWidth[aP/o?, aPulseWidth!/pulseWidth!]$

$SChamberPacingGetPulseAmpl[aP/o?, aPulseAmpl!/pulseAmpl!]$

$$PacingTriggerAPaceUFr == PacingGbl \setminus (currentCCI, sChamberPacing, stChamberPacing)$$

$$PacingTriggerAPaceFr == \Delta PacingGbl \land \Xi PacingTriggerAPaceUFr$$

PacingVTriggerAPace

$PacingTriggerAPaceFr$

$PacingTriggerAPace$

PacingTriggerVPace

$\Delta PacingGbl$

$now? : MilliSec$

$vPulseAmpl! : MilliVolt$

$vPulseWidth! : MicroSec$

$PacingVPaceCommon$

$SChamberPacingGetPulseWidth[vP/o?, vPulseWidth!/pulseWidth!]$

$SChamberPacingGetPulseAmpl[vP/o?, vPulseAmpl!/pulseAmpl!]$

$$PacingTriggerVPaceUFr == PacingGbl \setminus (currentCCI, sChamberPacing, stChamberPacing)$$

$$PacingTriggerVPaceFr == \Delta PacingGbl \land \Xi PacingTriggerVPaceUFr$$

PacingVTriggerVPace

$PacingTriggerVPaceFr$

$PacingTriggerVPace$

PacingInhibitAPace

$\Delta PacingGbl$

$now? : MilliSec$

$PacingAPaceCommon$

$$PacingInhibitAPaceUFr == PacingGbl \setminus (currentCCI, sChamberPacing, stChamberPacing)$$

$$PacingInhibitAPaceFr == \Delta PacingGbl \land \Xi PacingInhibitAPaceUFr$$

PacingVInhibitAPace

$PacingInhibitAPaceFr$

$PacingInhibitAPace$

PacingInhibitVPace

$\Delta PacingGbl$

$now? : MilliSec$

$PacingVPaceCommon$

$$PacingInhibitVPaceUFr == PacingGbl \setminus (currentCCI, sChamberPacing, stChamberPacing)$$

$$PacingInhibitVPaceFr == \Delta PacingGbl \land \Xi PacingInhibitVPaceUFr$$

PacingVInhibitVPace

$PacingInhibitVPaceFr$

$PacingInhibitVPace$

*PacingTimeToAPace*0

$PacingGblSt$

$now? : MilliSec$

$SChamberPacingGetLastEventTm[aP/o?]$

$lastEvTm! = now? - currentCCI$

$$PacingTimeToAPace == \exists\, lastEvTm! : \mathbb{Z} \bullet PacingTimeToAPace0$$

PacingTimedAPace

$\Delta PacingGbl$

$PacingTimeToAPace$

$PacingTriggerAPace$

$$PacingTimedAPaceUFr == PacingGbl \setminus (currentCCI, sChamberPacing, stChamberPacing)$$

$$PacingTimedAPaceFr == \Delta PacingGbl \land \Xi PacingTimedAPaceUFr$$

PacingVTimedAPace

$PacingTimedAPaceFr$

$PacingTimedAPace$

---

*PacingTimeToVPace*0

$PacingGblSt$

$now? : MilliSec$

$SChamberPacingGetLastEventTm[vP/o?]$

$lastEvTm! = now? - currentCCI$

$PacingTimeToVPace == \exists\, lastEvTm! : \mathbb{Z} \bullet$
$PacingTimeToVPace0$

PacingTimeToAVDelay

$PacingGblSt$

$now? : MilliSec$

$AVDelayGetAVDelay$

$SChamberPacingGetLastEventTm[aP/o?]$

$lastEvTm! = now? - avi!$

$lastEvTm! > now? - avi!$

$cci? = currentCCI$

---

*PacingTimedVPace*0

$\Delta PacingGbl$

$PacingTimeToVPace$

$PacingTriggerVPace$

$avi! : MilliSec$

$cci? : MilliSec$

$lastEvTm! : MilliSec$

$PacingTimeToAPace \lor PacingTimeToAVDelay$

$PacingTimedVPace == \exists\, avi! : \mathbb{Z};\ cci : \mathbb{Z};\ lastEvTm! : \mathbb{Z} \bullet$
$PacingTimedVPace0$

$PacingTimedVPaceUFr == PacingGbl \setminus (currentCCI, sChamberPacing, stChamberPacing)$

$PacingTimedVPaceFr == \Delta PacingGbl \land \Xi PacingTimedVPaceUFr$

PacingVTimedVPace

$PacingTimedVPaceFr$

$PacingTimedVPace$

PacingTrackASense

$\Delta PacingGbl$

$PacingTimeToVPace$

$PacingTriggerVPace$

$now? : MilliSec$

$SChamberPacingSetLastEventTm[aP/o?]$

$PacingTrackASenseUFr == PacingGbl \setminus (currentCCI, sChamberPacing, stChamberPacing)$

$PacingTrackASenseFr == \Delta PacingGbl \land \Xi PacingTrackASenseUFr$

PacingVTrackASense

$PacingTrackASenseFr$

$PacingTrackASense$

PacingExecuteEventATRFallback

$\Delta PacingGbl$

$ev? : PMEvent$

$PacingSetBradycardiaMode[V/cp?, R/rm?]$

$ev? = EvATRFallback$

$targetCCI' = targetCCI$

PacingExecuteEventTerminateATR

$\Delta PacingGbl$

$ev? : PMEvent$

$ev? = EvTerminateATR$

$cp' = pcp$

$pcp' = cp$

$rm' = prm$

$prm' = rm$

$targetCCI' = targetCCI$

PacingExecuteEventFallbackTmElapsed

$\Delta PacingGbl$
$ev? : PMEvent$
$RateModulationVSetTargetCCIToLRL$

$ev? = EvFallbackTmElapsed$

$cp' = cp$

$pcp' = pcp$

$rm' = rm$

$prm' = prm$

PacingExecuteEventNone

$\Delta PacingGbl$
$ev? : PMEvent$

$ev? = EvNone$

$cp' = cp$

$pcp' = pcp$

$rm' = rm$

$prm' = prm$

$targetCCI' = targetCCI$

$$PacingExecuteEvent == PacingExecuteEventATRFallback \\ \lor PacingExecuteEventTerminateATR \\ \lor PacingExecuteEventFallbackTmElapsed \\ \lor PacingExecuteEventNone$$

$$PacingExecuteEventUFr == PacingGbl \setminus (cp, pcp, rm, prm, targetCCI)$$
$$PacingExecuteEventFr == \Delta PacingGbl \land \Xi PacingExecuteEventUFr$$

PacingVExecuteEvent

$PacingExecuteEventFr$
$PacingExecuteEvent$

C.8 Package `Sensing`

section *Sensing* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*

$$Sensitivity == \{o : MicroVolt \mid o \geq 250 \land o \leq 10000\}$$

$$Blanking == \{o : MilliSec \mid o \geq 30 \land o \leq 60\}$$

SensingGblSt

$aSensitivity : Sensitivity$
$vSensitivity : Sensitivity$
$cs : ChambersSensed$
$aBlanking : Blanking$
$vBlanking : Blanking$
$pcs : ChambersSensed$

SensingGbl

$SensingGblSt$

SensingInit

$SensingGbl$

$aSensitivity = 750$
$vSensitivity = 2500$
$cs = D$
$aBlanking = 40$
$vBlanking = 40$
$pcs = D$

SensingSetBradycardiaMode

$\Delta SensingGbl$
$cs? : ChambersSensed$

$cs' = cs?$
$pcs' = cs$

$$SensingSetBradycardiaModeUFr == SensingGbl \setminus (cs, pcs)$$
$$SensingSetBradycardiaModeFr == \Delta SensingGbl \land \Xi SensingSetBradycardiaModeUFr$$

SensingVSetBradycardiaMode

$$SensingSetBradycardiaModeFr$$
$$SensingSetBradycardiaMode$$

SensingIsBatteryDegraded

$$SensingGblSt$$
$$bs? : BatteryStatus$$

$$bs? = ERT \lor bs? = ERP$$

SensingUpdateBatteryChangeMode

$$\Delta SensingGblSt$$
$$SensingIsBatteryDegraded$$
$$SensingSetBradycardiaMode[V / cs?]$$

SensingUpdateBatterySame

$$\Delta SensingGblSt$$

$$cs' = cs$$
$$pcs' = pcs$$

SensingUpdateBatteryStatus

$$\Delta SensingGbl$$
$$bs? : BatteryStatus$$

$$SensingUpdateBatteryChangeMode \lor SensingUpdateBatterySame$$

$$SensingUpdateBatteryStatusUFr == SensingGbl \setminus (cs, pcs)$$
$$SensingUpdateBatteryStatusFr == \Delta SensingGbl \land \Xi SensingUpdateBatteryStatusUFr$$

SensingVUpdateBatteryStatus

$$SensingUpdateBatteryStatusFr$$
$$SensingUpdateBatteryStatus$$

SensingExecuteEventATRFallback

$\Delta SensingGbl$
$ev? : PMEvent$
$SensingSetBradycardiaMode[D/cs?]$

$ev? = EvATRFallback$

SensingExecuteEventTerminateATR

$\Delta SensingGbl$
$ev? : PMEvent$

$ev? = EvTerminateATR$

$cs' = pcs$

$pcs' = cs$

SensingEvsDoingNothing

$SensingGbl$
$ev? : PMEvent$

$ev? = EvFallbackTmElapsed \lor ev? = EvNone$

SensingExecuteEventDoingNothing

$\Delta SensingGbl$
$SensingEvsDoingNothing$
$ev? : PMEvent$

$cs' = pcs$

$pcs' = cs$

$$SensingExecuteEvent == SensingExecuteEventATRFallback \lor SensingExecuteEventTerminateATR \lor SensingExecuteEventDoingNothing$$

$$SensingExecuteEventUFr == SensingGbl \setminus (cs, pcs)$$

$$SensingExecuteEventFr == \Delta SensingGbl \land \Xi SensingExecuteEventUFr$$

SensingVExecuteEvent

$SensingExecuteEventFr$
$SensingExecuteEvent$

SensingASenseOnly

$SensingGblSt$
$sAAmpl? : MicroVolt$
$now? : MilliSec$
$lastAEvTm? : MilliSec$

$sAAmpl? > aSensitivity$

$aBlanking < now? - lastAEvTm?$

$cs = A \lor cs = D$

SensingVASenseOnly

$SensingASenseOnly$

SensingTriggerAPace

$SensingGbl$
$SensingASenseOnly$

SensingVTriggerAPace

$SensingTriggerAPace$

SensingInhibitAPace

$SensingGbl$
$SensingASenseOnly$

SensingVInhibitAPace

$SensingInhibitAPace$

SensingASenseRefractory

$SensingGbl$
$SensingASenseOnly$

SensingVASenseRefractory

$SensingASenseRefractory$

SensingVSenseOnly

$SensingGblSt$
$sVAmpl? : MicroVolt$
$now? : MilliSec$
$lastVEvTm? : MilliSec$

$sVAmpl? > vSensitivity$

$vBlanking < now? - lastVEvTm?$

$cs = V \lor cs = D$

SensingVVSenseOnly

$SensingGbl$
$SensingVSenseOnly$

SensingVSenseRefractory

$SensingGbl$
$SensingVSenseOnly$

SensingVVSenseRefractory

$SensingVSenseRefractory$

SensingTriggerVPace

$SensingGbl$
$SensingVSenseOnly$

SensingVTriggerVPace

$SensingTriggerVPace$

SensingInhibitVPace

$SensingGbl$
$SensingVSenseOnly$

SensingVInhibitVPace

$$SensingInhibitVPace$$

SensingTrackASense

$$SensingGbl$$
$$SensingASenseOnly$$

SensingVTrackASense

$$SensingTrackASense$$

SensingIncASensitivityLow

$$\Delta SensingGbl$$

$$aSensitivity < 100$$
$$aSensitivity' = aSensitivity + 250$$

SensingIncASensitivityMedium

$$\Delta SensingGbl$$

$$aSensitivity \geq 1000 \land aSensitivity < 10000$$
$$aSensitivity' = aSensitivity + 500$$

SensingIncASensitivitySame

$$\Delta SensingGbl$$

$$aSensitivity = 1000$$
$$aSensitivity' = aSensitivity$$

$$SensingIncASensitivity == SensingIncASensitivityLow \lor SensingIncASensitivityMedium \lor SensingIncASensitivitySame$$

$$SensingIncASensitivityUFr == SensingGbl \setminus (aSensitivity)$$
$$SensingIncASensitivityFr == \Delta SensingGbl \land \Xi SensingIncASensitivityUFr$$

SensingVIncASensitivity

$$SensingIncASensitivityFr$$
$$SensingIncASensitivity$$

SensingDecASensitivityLow

$$\Delta SensingGbl$$

$$aSensitivity \leq 1000 \land aSensitivity > 250$$
$$aSensitivity' = aSensitivity - 250$$

SensingDecASensitivityMedium

$$\Delta SensingGbl$$

$$aSensitivity > 1000$$
$$aSensitivity' = aSensitivity - 500$$

SensingDecASensitivitySame

$$\Delta SensingGbl$$

$$aSensitivity = 250$$
$$aSensitivity' = aSensitivity$$

$$SensingDecASensitivity == SensingDecASensitivityLow$$
$$\lor\ SensingDecASensitivityMedium$$
$$\lor\ SensingDecASensitivitySame$$

$$SensingDecASensitivityUFr == SensingGbl \setminus (aSensitivity)$$
$$SensingDecASensitivityFr == \Delta SensingGbl \land \Xi SensingDecASensitivityUFr$$

SensingVDecASensitivity

$$SensingDecASensitivityFr$$
$$SensingDecASensitivity$$

SensingIncVSensitivityLow

$$\Delta SensingGbl$$

$$vSensitivity < 100$$
$$vSensitivity' = aSensitivity + 250$$

SensingIncVSensitivityMedium

$\Delta SensingGbl$

$vSensitivity \geq 1000 \land vSensitivity < 10000$

$vSensitivity' = vSensitivity + 500$

SensingIncVSensitivitySame

$\Delta SensingGbl$

$vSensitivity = 1000$

$vSensitivity' = vSensitivity$

$SensingIncVSensitivity == SensingIncVSensitivityLow$
$\lor\ SensingIncVSensitivityMedium$
$\lor\ SensingIncVSensitivitySame$

$SensingIncVSensitivityUFr == SensingGbl \setminus (vSensitivity)$

$SensingIncVSensitivityFr == \Delta SensingGbl \land \Xi SensingIncVSensitivityUFr$

SensingVIncVSensitivity

$SensingIncVSensitivityFr$
$SensingIncVSensitivity$

SensingDecVSensitivityLow

$\Delta SensingGbl$

$vSensitivity \leq 1000 \land vSensitivity > 250$

$vSensitivity' = vSensitivity - 250$

SensingDecVSensitivityMedium

$\Delta SensingGbl$

$vSensitivity > 1000$

$vSensitivity' = vSensitivity - 500$

SensingDecVSensitivitySame

$\Delta SensingGbl$

$vSensitivity = 250$

$vSensitivity' = vSensitivity$

$SensingDecVSensitivity == SensingDecVSensitivityLow$
$\quad \lor SensingDecVSensitivityMedium$
$\quad \lor SensingDecVSensitivitySame$

$SensingDecVSensitivityUFr == SensingGbl \setminus (vSensitivity)$

$SensingDecVSensitivityFr == \Delta SensingGbl \land \Xi SensingDecVSensitivityUFr$

SensingVDecVSensitivity

$SensingDecVSensitivityFr$
$SensingDecVSensitivity$

SensingIncABlankingInc

$\Delta SensingGbl$

$aBlanking < 60$

$aBlanking' = aBlanking + 10$

SensingIncABlankingSame

$\Delta SensingGbl$

$aBlanking = 60$

$aBlanking' = aBlanking$

$SensingIncABlanking == SensingIncABlankingInc$
$\quad \lor SensingIncABlankingSame$

$SensingIncABlankingUFr == SensingGbl \setminus (aBlanking)$

$SensingIncABlankingFr == \Delta SensingGbl \land \Xi SensingIncABlankingUFr$

SensingVIncABlanking

$$SensingIncABlankingFr$$
$$SensingIncABlanking$$

SensingDecABlankingDec

$$\Delta SensingGbl$$

$$aBlanking > 30$$
$$aBlanking' = aBlanking - 10$$

SensingDecABlankingSame

$$\Delta SensingGbl$$

$$aBlanking = 30$$
$$aBlanking' = aBlanking$$

$$SensingDecABlanking == SensingDecABlankingDec \lor SensingDecABlankingSame$$

$$SensingDecABlankingUFr == SensingGbl \setminus (aBlanking)$$
$$SensingDecABlankingFr == \Delta SensingGbl \land \Xi SensingDecABlankingUFr$$

SensingVDecABlanking

$$SensingDecABlankingFr$$
$$SensingDecABlanking$$

SensingIncVBlankingInc

$$\Delta SensingGbl$$

$$vBlanking < 60$$
$$vBlanking' = vBlanking + 10$$

SensingIncVBlankingSame

$\Delta SensingGbl$

$vBlanking = 60$

$vBlanking' = vBlanking$

$SensingIncVBlanking == SensingIncVBlankingInc$
$\quad \lor SensingIncVBlankingSame$

$SensingIncVBlankingUFr == SensingGbl \setminus (vBlanking)$
$SensingIncVBlankingFr == \Delta SensingGbl \land \Xi SensingIncVBlankingUFr$

SensingVIncVBlanking

$SensingIncVBlankingFr$
$SensingIncVBlanking$

SensingDecVBlankingDec

$\Delta SensingGbl$

$vBlanking > 30$

$vBlanking' = vBlanking - 10$

SensingDecVBlankingSame

$\Delta SensingGbl$

$vBlanking = 30$

$vBlanking' = vBlanking$

$SensingDecVBlanking == SensingDecVBlankingDec$
$\quad \lor SensingDecVBlankingSame$

$SensingDecVBlankingUFr == SensingGbl \setminus (vBlanking)$
$SensingDecVBlankingFr == \Delta SensingGbl \land \Xi SensingDecVBlankingUFr$

SensingVDecVBlanking

$SensingDecVBlankingFr$
$SensingDecVBlanking$

C.9 Package ResponseCommon

section *ResponseCommon* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*

$RefractoryPeriod == \{o : MilliSec \mid o \geq 150 \land o \leq 500\}$

C.10 Package AtrialResponse

section *AtrialResponse* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*, *ResponseCommon*

$\mid \quad detectionPeriod : MilliSec$

$UpperRateLimit == \{o : PPM \mid o \geq 50 \land o \leq 175\}$

$ATRDuration == \{o : \mathbb{N} \mid o \geq 10 \land o \leq 2000\}$

$ATRFallbackTime == \{o : MilliSec \mid o \geq 60000 \land o \leq 300000\}$

$ATRDetectionCount == \{o : \mathbb{N} \mid o \geq 1 \land o \leq 8\}$

$ATRStatus ::= NoATR \mid Duration \mid Fallback \mid FallbackTmElapsed$

AtrialResponseGblSt

$url : UpperRateLimit$
$arp : RefractoryPeriod$
$atrMode : Switch$
$atrDuration : ATRDuration$
$atrFbTm : ATRFallbackTime$
$atrEntryC : ATRDetectionCount$
$atrExitC : ATRDetectionCount$
$atrDetC : \mathbb{N}$
$atrStartDetTm : MilliSec$
$atrStatus : ATRStatus$
$atrDurC : \mathbb{N}$
$atrStartFbTm : MilliSec$

AtrialResponseGbl

$AtrialResponseGblSt$

AtrialResponseInit

$AtrialResponseGbl$

$url = 120$

$arp = 250$

$atrMode = Off$

$atrDuration = 20 \land atrFbTm = 60000$

$atrEntryC = 8 \land atrExitC = 8$

$atrStartDetTm = 0 \land atrStartFbTm = 0$

$atrStatus = NoATR \land (atrDetC = 0 \land atrDurC = 0)$

AtrialResponseInATRFallBackMode

$AtrialResponseGbl$

$atrStatus = Fallback \lor atrStatus = FallbackTmElapsed$

AtrialResponseVInATRFallBackMode

$AtrialResponseInATRFallBackMode$

AtrialResponseDoNoATR

$\Delta AtrialResponseGbl$

$ev! : PMEvent$

$atrMode = Off$

$ev! = EvNone$

$atrStartDetTm' = atrStartDetTm$

$atrDetC' = atrDetC$

$atrStatus' = atrStatus$

$atrDurC' = atrDurC$

AtrialResponseSetPMEv

$AtrialResponseGbl$
$ev! : PMEvent$
$evDetect? : PMEvent$
$evDurationFallback? : PMEvent$

$(evDetect? \neq EvNone \land ev! = evDetect?) \lor ev! = evDurationFallback?$

AtrialResponseGetURLI

$AtrialResponseGbl$
$urlI! : MilliSec$

$urlI! = 60000 \text{ div } url$

AtrialResponseATRDetectEntryReset

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$laet? : MilliSec$
$evDetect! : PMEvent$
$AtrialResponseGetURLI$

$detectionPeriod < now? - atrStartDetTm$

$urlI! < now? - laet?$

$atrStartDetTm' = now?$

$atrDetC' = 1$

$evDetect! = EvNone$

$atrDurC' = atrDurC$

$atrStatus' = atrStatus$

AtrialResponseATRDetectEntryInc

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$laet? : MilliSec$
$evDetect! : PMEvent$
$AtrialResponseGetURLI$

$urlI! < now? - laet?$

$detectionPeriod \geq now? - atrStartDetTm$

$atrEntryC > atrDetC + 1$

$atrDetC' = atrDetC + 1$

$evDetect! = EvNone$

$atrStartDetTm' = atrStartDetTm$

$atrDurC' = atrDurC$

$atrStatus' = atrStatus$

AtrialResponseATRDetectEntryToDuration

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$laet? : MilliSec$
$evDetect! : PMEvent$
$AtrialResponseGetURLI$

$urlI! < now? - laet?$

$detectionPeriod \geq now? - atrStartDetTm$

$atrEntryC = atrDetC + 1$

$atrStatus' = Duration$

$atrDurC' = 0$

$evDetect! = EvATRDuration$

$atrStartDetTm' = atrStartDetTm$

$atrDetC' = atrDetC$

$AtrialResponseATRDetectEntry0 == AtrialResponseATRDetectEntryReset$
$\quad \lor AtrialResponseATRDetectEntryInc$
$\quad \lor AtrialResponseATRDetectEntryToDuration$

AtrialResponseATRDetectEntry

$\Delta AtrialResponseGbl$
$AtrialResponseATRDetectEntry0$

$atrStatus = NoATR$

AtrialResponseATRDetectExitReset

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$laet? : MilliSec$
$evDetect! : PMEvent$
$AtrialResponseGetURLI$

$detectionPeriod < now? - atrStartDetTm$

$urlI! \geq now? - laet?$

$atrStartDetTm' = now?$

$atrDetC' = 1$

$evDetect! = EvNone$

$atrDurC' = atrDurC$

$atrStatus' = atrStatus$

AtrialResponseATRDetectExitInc

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$laet? : MilliSec$
$evDetect! : PMEvent$
$AtrialResponseGetURLI$

$urlI! \geq now? - laet?$

$detectionPeriod \geq now? - atrStartDetTm$

$atrExitC > atrDetC + 1$

$atrDetC' = atrDetC + 1$

$evDetect! = EvNone$

$atrStartDetTm' = atrStartDetTm$

$atrDurC' = atrDurC$

$atrStatus' = atrStatus$

AtrialResponseATRDetectExitSetPMEvent

$$
\begin{array}{l}
AtrialResponseGbl \\
evDetect! : PMEvent \\
\hline
atrStatus = Fallback \Rightarrow evDetect! = EvTerminateATR \\
atrStatus \neq Fallback \Rightarrow evDetect! = EvNone
\end{array}
$$

AtrialResponseATRDetectEntryToNoATR

$$
\begin{array}{l}
\Delta AtrialResponseGbl \\
now? : MilliSec \\
laet? : MilliSec \\
AtrialResponseGetURLI \\
AtrialResponseATRDetectExitSetPMEvent \\
\hline
urlI! \geq now? - laet? \\
detectionPeriod \geq now? - atrStartDetTm \\
atrExitC = atrDetC + 1 \\
atrStatus' = NoATR \\
atrDurC' = atrDurC \\
atrStartDetTm' = atrStartDetTm \\
atrDetC' = atrDetC
\end{array}
$$

$$
\begin{array}{l}
AtrialResponseATRDetectExit0 == AtrialResponseATRDetectExitReset \\
\quad \lor AtrialResponseATRDetectExitInc \\
\quad \lor AtrialResponseATRDetectEntryToNoATR
\end{array}
$$

AtrialResponseATRDetectExit

$$
\begin{array}{l}
\Delta AtrialResponseGbl \\
AtrialResponseATRDetectExit0 \\
\hline
atrStatus \neq NoATR
\end{array}
$$

$$
\begin{array}{l}
AtrialResponseATRDetect == AtrialResponseATRDetectEntry \\
\quad \lor AtrialResponseATRDetectExit
\end{array}
$$

AtrialResponseATRDoDurationInc

$\Delta AtrialResponseGbl$
$evDurationFallback! : PMEvent$

$atrDuration > atrDurC + 1$

$atrDurC' = atrDurC + 1$

$evDurationFallback! = EvNone$

$atrStatus' = atrStatus$

$atrStartFbTm' = atrStartFbTm$

AtrialResponseATRDoDurationToFallback

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$evDurationFallback! : PMEvent$

$atrDuration = atrDurC + 1$

$atrStatus' = Fallback$

$atrStartFbTm' = now?$

$evDurationFallback! = EvATRFallback$

$atrDurC' = atrDurC$

$AtrialResponseATRDoDuration0 == AtrialResponseATRDoDurationInc$
$\quad \lor AtrialResponseATRDoDurationToFallback$

AtrialResponseATRDoDuration

$\Delta AtrialResponseGbl$
$AtrialResponseATRDoDuration0$

$atrStatus = Duration$

AtrialResponseATRFallbackTmNotElapsed

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$evDurationFallback! : PMEvent$

$atrFbTm < now? - atrStartFbTm$

$evDurationFallback! = EvNone$

$atrStatus' = atrStatus$

AtrialResponseATRFallbackTmElapsed

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$evDurationFallback! : PMEvent$

$atrFbTm \geq now? - atrStartFbTm$

$evDurationFallback! = EvFallbackTmElapsed$

$atrStatus' = FallbackTmElapsed$

$AtrialResponseATRFallback0 == AtrialResponseATRFallbackTmNotElapsed$
$\quad \lor AtrialResponseATRFallbackTmElapsed$

AtrialResponseATRFallback

$\Delta AtrialResponseGbl$
$AtrialResponseATRFallback0$

$atrStatus = Fallback$

$atrStartFbTm' = atrStartFbTm$

$atrDurC' = atrDurC$

AtrialResponseATRDoNothing

$\Delta AtrialResponseGbl$
$evDurationFallback! : PMEvent$

$atrStatus \neq Duration \land atrStatus \neq Fallback$

$evDurationFallback! = EvNone$

$atrStatus' = atrStatus$

$atrStartFbTm' = atrStartFbTm$

$atrDurC' = atrDurC$

$AtrialResponseDoATRDisj0 == AtrialResponseATRDoDuration$
$\quad \lor AtrialResponseATRFallback$
$\quad \lor AtrialResponseATRDoNothing$

AtrialResponseDoATRDisj

$\Delta AtrialResponseGbl$
$AtrialResponseDoATRDisj0$

$atrDetC' = atrDetC$

$atrStartDetTm' = atrStartDetTm$

$AtrialResponseDoATRSemi0 == AtrialResponseATRDetect$
$\quad \fatsemi AtrialResponseDoATRDisj$

AtrialResponseDoATR0

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$ev! : PMEvent$
$laet? : MilliSec$
$AtrialResponseDoATRSemi0$
$AtrialResponseSetPMEv[evDetect!/evDetect?, evDurationFallback!/evDurationFallback?]$

$atrMode = On$

$AtrialResponseDoATR == AtrialResponseDoATR0 \setminus (evDetect!, evDurationFallback!)$

$AtrialResponsePerformATR == AtrialResponseDoATR \lor AtrialResponseDoNoATR$

$AtrialResponsePerformATRUFr == AtrialResponseGbl \setminus (atrDurC, atrStartFbTm, atrStatus, atrDetC, atrSta$
$AtrialResponsePerformATRFr == \Delta AtrialResponseGbl \land \Xi AtrialResponsePerformATRUFr$

AtrialResponseVPerformATR

$AtrialResponsePerformATRFr$
$AtrialResponsePerformATR$

AtrialResponseIsInRefractory

$AtrialResponseGbl$
$now? : MilliSec$
$lastAEvTm? : MilliSec$
$lastVEvTm? : MilliSec$
$pvarp? : MilliSec$

$arp \leq now? - lastAEvTm? \lor pvarp? \leq now? - lastVEvTm?$

AtrialResponseASenseOnly

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$lastAEvTm? : MilliSec$
$lastVEvTm? : MilliSec$
$pvarp? : MilliSec$
$AtrialResponsePerformATR$

$\neg\ AtrialResponseIsInRefractory$

$AtrialResponseASenseOnlyUFr == AtrialResponseGbl \setminus (atrDurC, atrStartFbTm, atrStatus, atrDetC, atrStart$
$AtrialResponseASenseOnlyFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseASenseOnlyUFr$

AtrialResponseVASenseOnly

$AtrialResponseASenseOnlyFr$
$AtrialResponseASenseOnly$

AtrialResponseASenseRefractory

$\Delta AtrialResponseGbl$
$now? : MilliSec$
$lastAEvTm? : MilliSec$
$lastVEvTm? : MilliSec$
$pvarp? : MilliSec$
$AtrialResponsePerformATR$

$AtrialResponseIsInRefractory$

$AtrialResponseASenseRefractoryUFr == AtrialResponseGbl \setminus (atrDurC, atrStartFbTm, atrStatus, atrDetC, at$
$AtrialResponseASenseRefractoryFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseASenseRefractoryUFr$

AtrialResponseVASenseRefractory

$AtrialResponseASenseRefractoryFr$
$AtrialResponseASenseRefractory$

AtrialResponseIncURLInc

$\Delta AtrialResponseGbl$

$url < 175$

$url' = url + 5$

AtrialResponseIncURLSame

$\Delta AtrialResponseGbl$

$url = 175$

$url' = url$

$AtrialResponseIncURLUFr == AtrialResponseGbl \setminus (url)$

$AtrialResponseIncURLFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseIncURLUFr$

$AtrialResponseIncURL == AtrialResponseIncURLInc$
$\quad \lor AtrialResponseIncURLSame$

AtrialResponseVIncURL

$AtrialResponseIncURLFr$
$AtrialResponseIncURL$

AtrialResponseDecURLDec

$\Delta AtrialResponseGbl$

$url > 50$

$url' = url - 5$

AtrialResponseDecURLSame

$\Delta AtrialResponseGbl$

$url = 50$

$url' = url$

$AtrialResponseDecURL == AtrialResponseDecURLDec$
$\quad \lor AtrialResponseDecURLSame$

$AtrialResponseDecURLUFr == AtrialResponseGbl \setminus (url)$

$AtrialResponseDecURLFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseDecURLUFr$

AtrialResponseVDecURL

$AtrialResponseDecURLFr$

$AtrialResponseDecURL$

AtrialResponseIncARPInc

$\Delta AtrialResponseGbl$

$arp < 500$

$arp' = arp + 10$

AtrialResponseIncARPSame

$\Delta AtrialResponseGbl$

$arp = 500$

$arp' = arp$

$AtrialResponseIncARPUFr == AtrialResponseGbl \setminus (arp)$

$AtrialResponseIncARPFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseIncARPUFr$

$AtrialResponseIncARP == AtrialResponseIncARPInc$
$\quad \lor AtrialResponseIncARPSame$

AtrialResponseVIncARP

$AtrialResponseIncARPFr$

$AtrialResponseIncARP$

AtrialResponseDecARPDec

$\Delta AtrialResponseGbl$

$arp > 150$

$arp' = arp - 10$

AtrialResponseDecARPSame

$\Delta AtrialResponseGbl$

$arp = 150$

$arp' = arp$

$$AtrialResponseDecARP == AtrialResponseDecARPDec \\ \lor AtrialResponseDecARPSame$$

$$AtrialResponseDecARPUFr == AtrialResponseGbl \setminus (arp)$$

$$AtrialResponseDecARPFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseDecARPUFr$$

AtrialResponseVDecARP

$$AtrialResponseDecARPFr \\ AtrialResponseDecARP$$

AtrialResponseATROn

$$\Delta AtrialResponseGbl$$

$$atrMode = Off$$

$$atrMode' = On$$

$$AtrialResponseATROnUFr == AtrialResponseGbl \setminus (atrMode)$$

$$AtrialResponseATROnFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseATROnUFr$$

AtrialResponseVATROn

$$AtrialResponseATROnFr \\ AtrialResponseATROn$$

AtrialResponseATROff

$$\Delta AtrialResponseGbl$$

$$atrMode = On$$

$$atrMode' = Off$$

$$AtrialResponseATROffUFr == AtrialResponseGbl \setminus (atrMode)$$

$$AtrialResponseATROffFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseATROnUFr$$

AtrialResponseVATROff

$AtrialResponseATROffFr$

$AtrialResponseATROff$

AtrialResponseIncATRDurLow

$\Delta AtrialResponseGbl$

$atrDuration < 20$

$atrDuration' = atrDuration + 10$

AtrialResponseIncATRDurMedium

$\Delta AtrialResponseGbl$

$atrDuration \geq 20 \land atrDuration \leq 80$

$atrDuration' = atrDuration + 20$

AtrialResponseIncATRDurHigh

$\Delta AtrialResponseGbl$

$atrDuration \geq 100 \land atrDuration < 2000$

$atrDuration' = atrDuration + 100$

AtrialResponseIncATRDurSame

$\Delta AtrialResponseGbl$

$atrDuration = 2000$

$atrDuration' = atrDuration$

$AtrialResponseIncATRDuration == AtrialResponseIncATRDurLow$
$\quad \lor AtrialResponseIncATRDurMedium \lor AtrialResponseIncATRDurHigh \lor AtrialResponseIncATRDurSame$

$AtrialResponseIncATRDurationUFr == AtrialResponseGbl \setminus (atrDuration)$

$AtrialResponseIncATRDurationFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseIncATRDurationUFr$

AtrialResponseVIncATRDuration

$$AtrialResponseIncATRDurationFr$$
$$AtrialResponseIncATRDuration$$

AtrialResponseDecATRDurLow

$$\Delta AtrialResponseGbl$$

$$atrDuration \leq 20$$
$$atrDuration' = 10$$

AtrialResponseDecATRDurMedium

$$\Delta AtrialResponseGbl$$

$$atrDuration > 20 \land atrDuration \leq 80$$
$$atrDuration' = atrDuration - 20$$

AtrialResponseDecATRDurHigh

$$\Delta AtrialResponseGbl$$

$$atrDuration \geq 100$$
$$atrDuration' = atrDuration - 100$$

$$AtrialResponseDecATRDuration == AtrialResponseDecATRDurLow \lor AtrialResponseDecATRDurMedium \lor AtrialResponseDecATRDurHigh$$

$$AtrialResponseDecATRDurationUFr == AtrialResponseGbl \setminus (atrDuration)$$
$$AtrialResponseDecATRDurationFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseDecATRDurationUFr$$

AtrialResponseVDecATRDuration

$$AtrialResponseDecATRDurationFr$$
$$AtrialResponseDecATRDuration$$

AtrialResponseIncATRFbTmInc

$$\Delta AtrialResponseGbl$$

$$atrFbTm < 5$$
$$atrFbTm' = atrFbTm + 1$$

AtrialResponseIncATRFbTmSame

$\Delta AtrialResponseGbl$

$atrFbTm = 5$

$atrFbTm' = atrFbTm$

$AtrialResponseIncATRFbTmUFr == AtrialResponseGbl \setminus (atrFbTm)$

$AtrialResponseIncATRFbTmFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseIncATRFbTmUFr$

$AtrialResponseIncATRFbTm == AtrialResponseIncATRFbTmInc$
$\quad \lor AtrialResponseIncATRFbTmSame$

AtrialResponseVIncATRFbTm

$AtrialResponseIncATRFbTmFr$
$AtrialResponseIncATRFbTm$

AtrialResponseDecATRFbTmDec

$\Delta AtrialResponseGbl$

$atrFbTm > 1$

$atrFbTm' = atrFbTm - 1$

AtrialResponseDecATRFbTmSame

$\Delta AtrialResponseGbl$

$atrFbTm = 1$

$atrFbTm' = atrFbTm$

$AtrialResponseDecATRFbTmUFr == AtrialResponseGbl \setminus (atrFbTm)$

$AtrialResponseDecATRFbTmFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseDecATRFbTmUFr$

$AtrialResponseDecATRFbTm == AtrialResponseDecATRFbTmDec$
$\quad \lor AtrialResponseDecATRFbTmSame$

AtrialResponseVDecATRFbTm

$$AtrialResponseDecATRFbTmFr$$
$$AtrialResponseDecATRFbTm$$

AtrialResponseIncATREntryCInc

$$\Delta AtrialResponseGbl$$

$$atrEntryC < 8$$
$$atrEntryC' = atrEntryC + 1$$

AtrialResponseIncATREntryCSame

$$\Delta AtrialResponseGbl$$

$$atrEntryC = 8$$
$$atrEntryC' = atrEntryC$$

$$AtrialResponseIncATREntryCUFr == AtrialResponseGbl \setminus (atrEntryC)$$

$$AtrialResponseIncATREntryCFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseIncATREntryCUFr$$

$$AtrialResponseIncATREntryC == AtrialResponseIncATREntryCInc \lor AtrialResponseIncATREntryCSame$$

AtrialResponseVIncATREntryC

$$AtrialResponseIncATREntryCFr$$
$$AtrialResponseIncATREntryC$$

AtrialResponseDecATREntryCDec

$$\Delta AtrialResponseGbl$$

$$atrEntryC > 1$$
$$atrEntryC' = atrEntryC - 1$$

AtrialResponseDecATREntryCSame

$\Delta AtrialResponseGbl$

$atrEntryC = 1$

$atrEntryC' = atrEntryC$

$AtrialResponseDecATREntryCUFr == AtrialResponseGbl \setminus (atrEntryC)$

$AtrialResponseDecATREntryCFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseDecATREntryCUFr$

$AtrialResponseDecATREntryC == AtrialResponseDecATREntryCDec$
$\lor AtrialResponseDecATREntryCSame$

AtrialResponseVDecATREntryC

$AtrialResponseDecATREntryCFr$
$AtrialResponseDecATREntryC$

AtrialResponseIncATRExitCInc

$\Delta AtrialResponseGbl$

$atrExitC < 8$

$atrExitC' = atrExitC + 1$

AtrialResponseIncATRExitCSame

$\Delta AtrialResponseGbl$

$atrExitC = 8$

$atrExitC' = atrExitC$

$AtrialResponseIncATRExitCUFr == AtrialResponseGbl \setminus (atrExitC)$

$AtrialResponseIncATRExitCFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseIncATRExitCUFr$

$AtrialResponseIncATRExitC == AtrialResponseIncATRExitCInc$
$\lor AtrialResponseIncATRExitCSame$

AtrialResponseVIncATRExitC

$AtrialResponseIncATRExitCFr$

$AtrialResponseIncATRExitC$

AtrialResponseDecATRExitCDec

$\Delta AtrialResponseGbl$

$atrExitC > 1$

$atrExitC' = atrExitC - 1$

AtrialResponseDecATRExitCSame

$\Delta AtrialResponseGbl$

$atrExitC = 1$

$atrExitC' = atrExitC$

$AtrialResponseDecATRExitCUFr == AtrialResponseGbl \setminus (atrExitC)$

$AtrialResponseDecATRExitCFr == \Delta AtrialResponseGbl \land \Xi AtrialResponseDecATRExitCUFr$

$AtrialResponseDecATRExitC == AtrialResponseDecATRExitCDec$
$\quad \lor AtrialResponseDecATRExitCSame$

AtrialResponseVDecATRExitC

$AtrialResponseDecATRExitCFr$

$AtrialResponseDecATRExitC$

C.11 Package `VentricularResponse`

section *VentricularResponse* **parents** $ZOO\_Toolkit$, $Model\_Preamble$, $CommonPM$, $ResponseCommon$

$PVARPExtension == \{o : MilliSec \mid o \geq 0 \land o \leq 400\}$

VentricularResponseGblSt

$vrp : RefractoryPeriod$

$pvarp : RefractoryPeriod$

$pvarpExt : PVARPExtension$

$pvc : Bool$

VentricularResponseGbl

$VentricularResponseGblSt$

VentricularResponseInit

$VentricularResponseGbl$

$vrp = 320$

$pvarp = 250$

$pvarpExt = 0$

$pvc = False$

VentricularResponseGetPVARP

$VentricularResponseGbl$
$pvarp! : RefractoryPeriod$

$pvc = False \Rightarrow pvarp! = pvarp$

$pvc = True \Rightarrow pvarp! = pvarp + pvarpExt$

VentricularResponseVGetPVARP

$VentricularResponseGetPVARP$

VentricularResponseIsPVC

$VentricularResponseGbl$
$cs? : ChambersSensed$
$lastAEvTm? : MilliSec$
$lastVEvTm? : MilliSec$

$cs? = D \land lastVEvTm? > lastAEvTm?$

VentricularResponseVIsPVC

$VentricularResponseIsPVC$

VentricularResponseIncVRPInc

$\Delta VentricularResponseGbl$

$vrp < 500$

$vrp' = vrp + 10$

VentricularResponseIncVRPSame

$\Delta VentricularResponseGbl$

$vrp = 500$

$vrp' = vrp$

$VentricularResponseIncVRPUFr == VentricularResponseGbl \setminus (vrp)$

$VentricularResponseIncVRPFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseIncVRPUFr$

$VentricularResponseIncVRP == VentricularResponseIncVRPInc$
$\quad \lor VentricularResponseIncVRPSame$

VentricularResponseVIncVRP

$VentricularResponseIncVRPFr$
$VentricularResponseIncVRP$

VentricularResponseDecVRPDec

$\Delta VentricularResponseGbl$

$vrp > 150$

$vrp' = vrp - 10$

VentricularResponseDecVRPSame

$\Delta VentricularResponseGbl$

$vrp = 150$

$vrp' = vrp$

$VentricularResponseDecVRPUFr == VentricularResponseGbl \setminus (vrp)$

$VentricularResponseDecVRPFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseDecVRPUFr$

$VentricularResponseDecVRP == VentricularResponseDecVRPDec$
$\quad \lor VentricularResponseDecVRPSame$

VentricularResponseVDecVRP

$$VentricularResponseDecVRPFr$$
$$VentricularResponseDecVRP$$

VentricularResponseIncPVARPInc

$$\Delta VentricularResponseGbl$$

$$pvarp < 500$$
$$pvarp' = pvarp + 10$$

VentricularResponseIncPVARPSame

$$\Delta VentricularResponseGbl$$

$$pvarp = 500$$
$$pvarp' = pvarp$$

$$VentricularResponseIncPVARPUFr == VentricularResponseGbl \setminus (pvarp)$$

$$VentricularResponseIncPVARPFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseIncPVARPUFr$$

$$VentricularResponseIncPVARP == VentricularResponseIncPVARPInc \lor VentricularResponseIncPVARPSame$$

VentricularResponseVIncPVARP

$$VentricularResponseIncPVARPFr$$
$$VentricularResponseIncPVARP$$

VentricularResponseDecPVARPDec

$$\Delta VentricularResponseGbl$$

$$pvarp > 150$$
$$pvarp' = pvarp - 10$$

VentricularResponseDecPVARPSame

$\Delta VentricularResponseGbl$

$pvarp = 150$

$pvarp' = pvarp$

$VentricularResponseDecPVARPUFr == VentricularResponseGbl \setminus (pvarp)$

$VentricularResponseDecPVARPFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseDecPVARPUFr$

$VentricularResponseDecPVARP == VentricularResponseDecPVARPDec$
$\lor\ VentricularResponseDecPVARPSame$

VentricularResponseVDecPVARP

VentricularResponseDecPVARPFr
VentricularResponseDecPVARP

VentricularResponseIncPVARPExtInc

$\Delta VentricularResponseGbl$

$pvarpExt < 400$

$pvarpExt' = pvarpExt + 50$

VentricularResponseIncPVARPExtSame

$\Delta VentricularResponseGbl$

$pvarpExt = 400$

$pvarpExt' = pvarpExt$

$VentricularResponseIncPVARPExtUFr == VentricularResponseGbl \setminus (pvarpExt)$

$VentricularResponseIncPVARPExtFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseIncPVARPExtUF$

$VentricularResponseIncPVARPExt == VentricularResponseIncPVARPExtInc$
$\lor\ VentricularResponseIncPVARPExtSame$

VentricularResponseVIncPVARPExt

$$VentricularResponseIncPVARPExtFr$$
$$VentricularResponseIncPVARPExt$$

VentricularResponseDecPVARPExtDec

$$\Delta VentricularResponseGbl$$

$$pvarpExt > 0$$
$$pvarpExt' = pvarpExt - 50$$

VentricularResponseDecPVARPExtSame

$$\Delta VentricularResponseGbl$$

$$pvarpExt = 0$$
$$pvarpExt' = pvarpExt$$

$$VentricularResponseDecPVARPExtUFr == VentricularResponseGbl \setminus (pvarpExt)$$
$$VentricularResponseDecPVARPExtFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseDecPVARPExtU$$

$$VentricularResponseDecPVARPExt == VentricularResponseDecPVARPExtDec \lor VentricularResponseDecPVARPExtSame$$

VentricularResponseVDecPVARPExt

$$VentricularResponseDecPVARPExtFr$$
$$VentricularResponseDecPVARPExt$$

VentricularResponsePVCDetected

$$\Delta VentricularResponseGbl$$
$$VentricularResponseIsPVC$$

$$pvc = False$$
$$pvc' = True$$

VentricularResponseDisablePVC

$$\Delta VentricularResponseGbl$$

$$pvc' = False$$

$$VentricularResponsePVCUPdate == VentricularResponsePVCDetected \\ \lor VentricularResponseDisablePVC$$

VentricularResponseVSenseOnly

$$\Delta VentricularResponseGbl$$
$$VentricularResponsePVCUPdate$$
$$lastVEvTm? : MilliSec$$
$$now? : MilliSec$$

$$vrp < now? - lastVEvTm?$$

$$VentricularResponseVSenseOnlyUFr == VentricularResponseGbl \setminus (pvc)$$

$$VentricularResponseVSenseOnlyFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseVSenseOnlyUFr$$

VentricularResponseVVSenseOnly

$$VentricularResponseVSenseOnlyFr$$
$$VentricularResponseVSenseOnly$$

VentricularResponseVSenseRefractory

$$\Delta VentricularResponseGbl$$
$$VentricularResponsePVCUPdate$$
$$lastVEvTm? : MilliSec$$
$$now? : MilliSec$$

$$vrp \geq now? - lastVEvTm?$$

$$VentricularResponseVSenseRefractoryUFr == VentricularResponseGbl \setminus (pvc)$$

$$VentricularResponseVSenseRefractoryFr == \Delta VentricularResponseGbl \land \Xi VentricularResponseVSenseRefract$$

VentricularResponseVVSenseRefractory

$$VentricularResponseVSenseRefractoryFr$$
$$VentricularResponseVSenseRefractory$$

C.12 Package Response

section *Response* **parents** *ZOO_Toolkit*, *Model_Preamble*, *CommonPM*, *VentricularResponse*, *AtrialResponse*

ResponsePkgSt

$rts : ResponseToSensing$

$prts : ResponseToSensing$

ResponseOnlyGblSt

$ResponsePkgSt$

ResponseGblSt

$ResponseOnlyGblSt$

$VentricularResponseGbl$

$AtrialResponseGbl$

ResponseGbl

$ResponseGblSt$

ResponseInit

$AtrialResponseInit$

$VentricularResponseInit$

$ResponseGbl$

$rts = D$

$prts = D$

$ResponseIntExtAtrialResponseFr == \Delta ResponseGbl \land \Xi ResponseOnlyGblSt \land \Xi VentricularResponseGbl$

ResponseVInATRFallBackMode

$ResponseGbl$

$AtrialResponseVInATRFallBackMode$

ResponseVIncURL

ResponseIntExtAtrialResponseFr
AtrialResponseVIncURL

ResponseVDecURL

ResponseIntExtAtrialResponseFr
AtrialResponseVDecURL

ResponseVIncARP

ResponseIntExtAtrialResponseFr
AtrialResponseVIncARP

ResponseVDecARP

ResponseIntExtAtrialResponseFr
AtrialResponseVDecARP

ResponseVATROn

ResponseIntExtAtrialResponseFr
AtrialResponseVATROn

ResponseVATROff

ResponseIntExtAtrialResponseFr
AtrialResponseVATROff

ResponseVIncATRDuration

ResponseIntExtAtrialResponseFr
AtrialResponseVIncATRDuration

ResponseVDecATRDuration

ResponseIntExtAtrialResponseFr
AtrialResponseVDecATRDuration

ResponseVIncATRFbTm

ResponseIntExtAtrialResponseFr
AtrialResponseVIncATRDuration

ResponseVDecATRFbTm

ResponseIntExtAtrialResponseFr
AtrialResponseVDecATRFbTm

ResponseVIncATREntryC

ResponseIntExtAtrialResponseFr
AtrialResponseVIncATREntryC

ResponseVDecATREntryC

ResponseIntExtAtrialResponseFr
AtrialResponseVDecATREntryC

ResponseVIncATRExitC

ResponseIntExtAtrialResponseFr
AtrialResponseVIncATRExitC

ResponseVDecATRExitC

ResponseIntExtAtrialResponseFr
AtrialResponseVDecATRExitC

$$ResponseIntExtVentricularResponseFr == \Delta ResponseGbl \land \Xi ResponseOnlyGblSt \land \Xi AtrialResponseGbl$$

ResponseVIsPVC

ResponseGbl
VentricularResponseVIsPVC

ResponseVIncVRP

ResponseIntExtVentricularResponseFr
VentricularResponseVIncVRP

ResponseVDecVRP

$ResponseIntExtVentricularResponseFr$

$VentricularResponseVIncVRP$

ResponseVIncPVARP

$ResponseIntExtVentricularResponseFr$

$VentricularResponseVIncPVARP$

ResponseVDecPVARP

$ResponseIntExtVentricularResponseFr$

$VentricularResponseVIncPVARP$

ResponseVIncPVARPExt

$ResponseIntExtVentricularResponseFr$

$VentricularResponseVIncPVARPExt$

ResponseVDecPVARPExt

$ResponseIntExtVentricularResponseFr$

$VentricularResponseVIncPVARPExt$

ResponseVVSenseRefractory

$ResponseIntExtVentricularResponseFr$

$VentricularResponseVVSenseRefractory$

ResponseSetBradycardiaMode

$\Delta ResponseGbl$

$rts? : ResponseToSensing$

$rts' = rts?$

$prts' = rts$

$ResponseSetBradycardiaModeUFr == ResponseGbl \setminus (rts, prts)$

$ResponseSetBradycardiaModeFr == \Delta ResponseGbl \land \Xi ResponseSetBradycardiaModeUFr$

ResponseVSetBradycardiaMode

$$ResponseSetBradycardiaModeFr$$
$$ResponseSetBradycardiaMode$$

ResponseIsBatteryDegraded

$$ResponseGblSt$$
$$bs? : BatteryStatus$$

$$bs? = ERT \lor bs? = ERP$$

ResponseUpdateBatteryChangeMode

$$\Delta ResponseGblSt$$
$$ResponseIsBatteryDegraded$$
$$ResponseSetBradycardiaMode[I / rts?]$$

ResponseUpdateBatterySame

$$\Delta ResponseGblSt$$

$$rts' = rts$$

$$prts' = prts$$

ResponseUpdateBatteryStatus

$$\Delta ResponseGbl$$
$$bs? : BatteryStatus$$

$$ResponseUpdateBatteryChangeMode \lor ResponseUpdateBatterySame$$

$$ResponseUpdateBatteryStatusUFr == ResponseGbl \setminus (rts, prts)$$

$$ResponseUpdateBatteryStatusFr == \Delta ResponseGbl \land \Xi ResponseUpdateBatteryStatusUFr$$

ResponseVUpdateBatteryStatus

$$ResponseUpdateBatteryStatusFr$$
$$ResponseUpdateBatteryStatus$$

ResponseExecuteEventATRFallback

$\Delta ResponseGbl$
$ev? : PMEvent$
$ResponseSetBradycardiaMode[I / rts?]$

$ev? = EvATRFallback$

ResponseExecuteEventTerminateATR

$\Delta ResponseGbl$
$ev? : PMEvent$

$ev? = EvTerminateATR$

$rts' = prts$

$prts' = rts$

ResponseEvsDoingNothing

$ResponseGbl$
$ev? : PMEvent$

$ev? = EvFallbackTmElapsed \lor ev? = EvNone$

ResponseExecuteEventDoingNothing

$\Delta ResponseGbl$
$ResponseEvsDoingNothing$
$ev? : PMEvent$

$rts' = rts$

$rts' = rts$

$$ResponseExecuteEvent == ResponseExecuteEventATRFallback \lor ResponseExecuteEventTerminateATR \lor ResponseExecuteEventDoingNothing$$

$$ResponseExecuteEventUFr == ResponseGbl \setminus (rts, prts)$$

$$ResponseExecuteEventFr == \Delta ResponseGbl \land \Xi ResponseExecuteEventUFr$$

ExecuteEvent

$ResponseExecuteEventFr$
$ResponseExecuteEvent$

ResponseASenseRefractory

$\Delta ResponseGbl$
$now? : MilliSec$
$lastAEvTm? : MilliSec$
$lastVEvTm? : MilliSec$
$VentricularResponseVGetPVARP$
$AtrialResponseVASenseRefractory[pvarp!/pvarp?]$

$ResponseASenseRefractoryUFr == ResponsePkgSt \land VentricularResponseGbl$

$ResponseASenseRefractoryFr == \Delta ResponseGbl \land \Xi ResponseExecuteEventUFr$

ResponseVASenseRefractory

$ResponseASenseRefractoryFr$
$ResponseASenseRefractory$

ResponseASenseOnly

$\Delta ResponseGbl$
$AtrialResponseVASenseOnly$

$rts = O$

$ResponseASenseOnlyUFr == ResponsePkgSt \land VentricularResponseGbl$

$ResponseASenseOnlyFr == \Delta ResponseGbl \land \Xi ResponseASenseOnlyUFr$

ResponseVASenseOnly

$ResponseASenseOnlyFr$
$ResponseASenseOnly$

ResponseVSenseOnly

$\Delta ResponseGbl$
$VentricularResponseVVSenseOnly$

$rts = O$

$ResponseVSenseOnlyUFr == ResponsePkgSt \land AtrialResponseGbl$

$ResponseVSenseOnlyFr == \Delta ResponseGbl \land \Xi ResponseVSenseOnlyUFr$

ResponseVVSenseOnly

$ResponseVSenseOnlyFr$

$ResponseVSenseOnly$

ResponseTrackASense

$\Delta ResponseGbl$

$AtrialResponseVASenseOnly$

$rts = D$

$ResponseTrackASenseUFr == ResponsePkgSt \land VentricularResponseGbl$

$ResponseTrackASenseFr == \Delta ResponseGbl \land \Xi ResponseTrackASenseUFr$

ResponseVTrackASense

$ResponseTrackASenseFr$

$ResponseTrackASense$

ResponseTriggerAPace

$\Delta ResponseGbl$

$AtrialResponseVASenseOnly$

$rts = T$

$ResponseTriggerAPaceUFr == ResponsePkgSt \land VentricularResponseGbl$

$ResponseTriggerAPaceFr == \Delta ResponseGbl \land \Xi ResponseTriggerAPaceUFr$

ResponseVTriggerAPace

$ResponseTriggerAPaceFr$

$ResponseTriggerAPace$

ResponseTriggerVPace

$\Delta ResponseGbl$

$VentricularResponseVVSenseOnly$

$rts = T$

$ResponseTriggerVPaceUFr == ResponsePkgSt \land AtrialResponseGbl$

$ResponseTriggerVPaceFr == \Delta ResponseGbl \land \Xi ResponseTriggerVPaceUFr$

ResponseVTriggerVPace

$ResponseTriggerVPaceFr$

$ResponseTriggerVPace$

ResponseInhibitAPace

$\Delta ResponseGbl$

$AtrialResponseVASenseOnly$

$rts = I$

$ResponseInhibitAPaceUFr == ResponsePkgSt \land VentricularResponseGbl$

$ResponseInhibitAPaceFr == \Delta ResponseGbl \land \Xi ResponseInhibitAPaceUFr$

ResponseVInhibitAPace

$ResponseInhibitAPaceFr$

$ResponseInhibitAPace$

ResponseInhibitedOrTracked

$ResponseGbl$

$rts = I \lor rts = T$

ResponseInhibitVPace

$\Delta ResponseGbl$

$ResponseInhibitedOrTracked$

$VentricularResponseVVSenseOnly$

$ResponseInhibitVPaceUFr == ResponsePkgSt \land AtrialResponseGbl$

$ResponseInhibitVPaceFr == \Delta ResponseGbl \land \Xi ResponseInhibitVPaceUFr$

ResponseVInhibitVPace

$ResponseInhibitVPaceFr$

$ResponseInhibitVPace$

C.13 Package `HistoryCommon`

section *HistoryCommon* **parents** *ZOO_Toolkit*, *Model_Preamble*

$AugmentationMarker ::= ATRDur \mid ATRFB \mid ATREnd \mid PVP \mid AugMNone$

$VentricularMarker ::= VS \mid VP \mid PVC \mid VTN$

$AtrialMarker ::= AS \mid AP \mid AT \mid ATN$

C.14 Package `History`

section *History* **parents** *ZOO_Toolkit*, *Model_Preamble*, *HistoryCommon*, *CommonPM*

*HistoryEvent*0

$$
\begin{array}{l}
time : MilliSec \\
augMarker : AugmentationMarker \\
mRe : Bool \\
mHy : Bool \\
mSr : Bool \\
mUp : Bool \\
mDown : Bool
\end{array}
$$

HistoryEvent

$$HistoryEvent0$$

SHistoryEvent

$$
\begin{array}{l}
sHistoryEvent : \mathbb{P}(\mathbb{O}\ HistoryEventCl) \\
stHistoryEvent : \mathbb{O}\ HistoryEventCl \pfun HistoryEvent \\
\hline
\mathrm{dom}\ stHistoryEvent = sHistoryEvent \\
sHistoryEvent \cap \mathbb{O}x\ HistoryEventCl = \varnothing
\end{array}
$$

SHistoryEventInit

$$
\begin{array}{l}
SHistoryEvent \\
\hline
sHistoryEvent = \varnothing \land stHistoryEvent = \varnothing
\end{array}
$$

HistoryEventNew

$HistoryEvent$
$now? : MilliSec$
$mre? : Bool$
$mhy? : Bool$
$msr? : Bool$
$mUp? : Bool$
$mDown? : Bool$

$time = now?$

$mRe = mre?$

$mHy = mhy?$

$mSr = msr?$

$mUp = mUp?$

$mDown = mDown?$

SHistoryEventNF

$\Delta SHistoryEvent$
$HistoryEvent$
$o! : \mathbb{O}\, HistoryEventCl$

$sHistoryEvent' = sHistoryEvent' \cup \{o!\}$

$stHistoryEvent' = stHistoryEvent' \cup \{(o!, \theta\ HistoryEvent)\}$

*AtrialEvent*0

$HistoryEvent$
$aMarker : AtrialMarker$

AtrialEvent

$AtrialEvent0$

SAtrialEvent

$sAtrialEvent : \mathbb{P}(\mathbb{O}\ AtrialEventCl)$
$stAtrialEvent : \mathbb{O}\ AtrialEventCl \pfun AtrialEvent$

$\mathrm{dom}\ stAtrialEvent = sAtrialEvent$

SAtrialEventInit

$SAtrialEvent$

$sAtrialEvent = \varnothing \land stAtrialEvent = \varnothing$

SAtrialEventIsHistoryEvent

$SHistoryEvent;\ SAtrialEvent$

$sAtrialEvent \subseteq sHistoryEvent$

$\forall\ o : sAtrialEvent \bullet$
$\quad (\lambda\ AtrialEvent \bullet \theta\ HistoryEvent)(stAtrialEvent\ o) = (stHistoryEvent\ o)$

AtrialEventNew

$AtrialEvent$
$HistoryEventNew$
$am? : AtrialMarker$

$aMarker = am?$

SAtrialEventNF

$\Delta SAtrialEvent$
$SHistoryEventNF$
$AtrialEvent$
$o! : \mathbb{O}\ AtrialEventCl$

$o! \in \mathbb{O}xAtrialEventCl \setminus sAtrialEvent$

$sAtrialEvent' = sAtrialEvent' \cup \{o!\}$

$stAtrialEvent' = stAtrialEvent' \cup \{(o!, \theta\ AtrialEvent)\}$

$SAtrialEventNew == \exists\ AtrialEvent \bullet$
$\quad SAtrialEventNF[ae!/o!] \land AtrialEventNew$

*VentricularEvent*0

$HistoryEvent$
$vMarker : VentricularMarker$

VentricularEvent

$$VentricularEvent0$$

SVentricularEvent

$$sVentricularEvent : \mathbb{P}(\mathbb{O}\ VentricularEventCl)$$
$$stVentricularEvent : \mathbb{O}\ VentricularEventCl \pfun VentricularEvent$$

$$\mathrm{dom}\ stVentricularEvent = sVentricularEvent$$

SVentricularEventInit

$$SVentricularEvent$$

$$sVentricularEvent = \varnothing \land stVentricularEvent = \varnothing$$

SVentricularEventIsHistoryEvent

$$SHistoryEvent;\ SVentricularEvent$$

$$sVentricularEvent \subseteq sHistoryEvent$$
$$\forall\ o : sVentricularEvent \bullet$$
$$(\lambda\ VentricularEvent \bullet \theta\ HistoryEvent)(stVentricularEvent\ o) = (stHistoryEvent\ o)$$

VentricularEventNew

$$VentricularEvent$$
$$HistoryEventNew$$
$$vm? : VentricularMarker$$

$$vMarker = vm?$$

SVentricularEventNF

$$\Delta SVentricularEvent$$
$$SHistoryEventNF$$
$$VentricularEvent$$
$$o! : \mathbb{O}\ VentricularEventCl$$

$$o! \in \mathbb{O}xVentricularEventCl \setminus sVentricularEvent$$
$$sVentricularEvent' = sVentricularEvent' \cup \{o!\}$$
$$stVentricularEvent' = stVentricularEvent' \cup \{(o!, \theta\ VentricularEvent)\}$$

$$SVentricularEventNew == \exists\ VentricularEvent \bullet$$
$$SVentricularEventNF[ve!/o!] \land VentricularEventNew$$

HistoryGblSt

$SVentricularEvent$
$SAtrialEvent$
$SHistoryEvent$

HistoryGbl

$HistoryGblSt$

$SAtrialEventIsHistoryEvent$

$SVentricularEventIsHistoryEvent$

*HistoryCreateHistoryAEv*0

$\Delta HistoryGbl$
$SAtrialEventNew$

$HistoryCreateHistoryAEv == HistoryCreateHistoryAEv0 \setminus (ae!)$

HistoryVCreateHistoryAEv

$\Delta HistoryGbl$
$\Xi SVentricularEvent$
$HistoryCreateHistoryAEv$

*HistoryCreateHistoryVEv*0

$\Delta HistoryGbl$
$SVentricularEventNew$

$HistoryCreateHistoryVEv == HistoryCreateHistoryVEv0 \setminus (ve!)$

HistoryVCreateHistoryVEv

$\Delta HistoryGbl$
$\Xi SAtrialEvent$
$HistoryCreateHistoryVEv$

C.15 Package PacingWithHistoryI

section *PacingWithHistoryI* **parents** *ZOO_Toolkit*, *Model_Preamble*, *HistoryCommon*, *CommonPM*, *Pacing*

PacingWithHistoryIGblSt

PacingGbl

PacingWithHistoryIGbl

PacingWithHistoryIGblSt

$PacingWithHistoryIIntExtPacingFr == \Delta PacingWithHistoryIGbl$

PacingWithHistoryIInit

PacingWithHistoryIIntExtPacingFr
PacingInit

PacingWithHistoryIVIncRSDown

PacingWithHistoryIIntExtPacingFr
PacingVIncRSDown

PacingWithHistoryIVDecRSDown

PacingWithHistoryIIntExtPacingFr
PacingVDecRSDown

PacingWithHistoryIVIncRSUp

PacingWithHistoryIIntExtPacingFr
PacingVIncRSUp

PacingWithHistoryIVDecRSUp

PacingWithHistoryIIntExtPacingFr
PacingVDecRSUp

PacingWithHistoryIVDynamicDelayOn

PacingWithHistoryIIntExtPacingFr
PacingVDynamicDelayOn

PacingWithHistoryIVDynamicDelayOff

PacingWithHistoryIIntExtPacingFr
PacingVDynamicDelayOff

PacingWithHistoryIVIncFixedAVI

PacingWithHistoryIIntExtPacingFr
PacingVIncFixedAVI

PacingWithHistoryIVDecFixedAVI

PacingWithHistoryIIntExtPacingFr
PacingVDecFixedAVI

PacingWithHistoryIVIncMinDynAVDelay

PacingWithHistoryIIntExtPacingFr
PacingVIncMinDynAVDelay

PacingWithHistoryIVDecMinDynAVDelay

PacingWithHistoryIIntExtPacingFr
PacingVDecMinDynAVDelay

PacingWithHistoryIVIncMaxDynAVDelay

PacingWithHistoryIIntExtPacingFr
PacingVIncMaxDynAVDelay

PacingWithHistoryIVDecMaxDynAVDelay

PacingWithHistoryIIntExtPacingFr
PacingVDecMaxDynAVDelay

PacingWithHistoryIVIncSAVDelayOffset
PacingWithHistoryIIntExtPacingFr
PacingVIncSAVDelayOffset

PacingWithHistoryIVDecSAVDelayOffset
PacingWithHistoryIIntExtPacingFr
PacingVDecSAVDelayOffset

PacingWithHistoryIVSetBradycardiaMode
PacingWithHistoryIIntExtPacingFr
PacingVSetBradycardiaMode

PacingWithHistoryIVUpdateBatteryStatus
PacingWithHistoryIIntExtPacingFr
PacingVUpdateBatteryStatus

PacingWithHistoryIVIncAPulseAmplReg
PacingWithHistoryIIntExtPacingFr
PacingVIncAPulseAmplReg

PacingWithHistoryIVDecAPulseAmplReg
PacingWithHistoryIIntExtPacingFr
PacingVDecAPulseAmplReg

PacingWithHistoryIVIncVPulseAmplReg
PacingWithHistoryIIntExtPacingFr
PacingVIncVPulseAmplReg

PacingWithHistoryIVDecVPulseAmplReg
PacingWithHistoryIIntExtPacingFr
PacingVDecVPulseAmplReg

PacingWithHistoryIVIncAPulseAmplUReg
PacingWithHistoryIIntExtPacingFr
PacingVIncAPulseAmplUReg

PacingWithHistoryIVDecAPulseAmplUReg
PacingWithHistoryIIntExtPacingFr
PacingVDecAPulseAmplUReg

PacingWithHistoryIVIncVPulseAmplUReg
PacingWithHistoryIIntExtPacingFr
PacingVIncVPulseAmplUReg

PacingWithHistoryIVDecVPulseAmplUReg
PacingWithHistoryIIntExtPacingFr
PacingVDecVPulseAmplUReg

PacingWithHistoryIVIncAPulseWidth
PacingWithHistoryIIntExtPacingFr
PacingVIncAPulseWidth

PacingWithHistoryIVDecAPulseWidth
PacingWithHistoryIIntExtPacingFr
PacingVDecAPulseWidth

PacingWithHistoryIVIncVPulseWidth
PacingWithHistoryIIntExtPacingFr
PacingVIncVPulseWidth

PacingWithHistoryIVDecVPulseWidth
PacingWithHistoryIIntExtPacingFr
PacingVDecVPulseWidth

PacingWithHistoryIVIncLRL

PacingWithHistoryIIntExtPacingFr
PacingVIncLRL

PacingWithHistoryIVDecLRL

PacingWithHistoryIIntExtPacingFr
PacingVDecLRL

PacingWithHistoryIVIncHRL

PacingWithHistoryIIntExtPacingFr
PacingVIncHRL

PacingWithHistoryIVDecHRL

PacingWithHistoryIIntExtPacingFr
PacingVDecHRL

PacingWithHistoryIVSetTargetCCI

PacingWithHistoryIIntExtPacingFr
PacingVSetTargetCCI

PacingWithHistoryIVTriggerAPace

PacingWithHistoryIIntExtPacingFr
PacingVTriggerAPace

PacingWithHistoryIVTriggerVPace

PacingWithHistoryIIntExtPacingFr
PacingVTriggerVPace

PacingWithHistoryIVInhibitAPace

PacingWithHistoryIIntExtPacingFr
PacingVInhibitAPace

PacingWithHistoryIVInhibitVPace

PacingWithHistoryIIntExtPacingFr
PacingVInhibitVPace

PacingWithHistoryIVTrackASense

PacingWithHistoryIIntExtPacingFr
PacingVTrackASense

PacingWithHistoryIVExecuteEvent

PacingWithHistoryIIntExtPacingFr
PacingVExecuteEvent

PacingWithHistoryISetEvMarkersCommon

PacingWithHistoryIGbl
$augm! : AugmentationMarker$
$mRe! : Bool$
$mhy! : Bool$
$msr! : Bool$
$mUp! : Bool$
$mDown! : Bool$

$(rm = R \land targetCCI < 60000 \operatorname{div} lrl) \Leftrightarrow msr! = True$

$(rm = O \land hystMode = On) \Leftrightarrow mhy! = True$

$targetCCI < currentCCI \Leftrightarrow mUp! = True$

$targetCCI > currentCCI \Leftrightarrow mDown! = True$

$augm! = AugMNone \land mRe! = False$

PacingWithHistoryISetEvMarkersA

PacingWithHistoryIGbl
PacingWithHistoryISetEvMarkersCommon
$am! : AtrialMarker$

$am! = AP$

PacingWithHistoryISetEvMarkersV

$PacingWithHistoryIGbl$
$PacingWithHistoryISetEvMarkersCommon$
$vm! : VentricularMarker$

$vm! = VP$

PacingWithHistoryIVTimedAPace

$PacingWithHistoryIGbl$
$PacingVTimedAPace$
$PacingWithHistoryISetEvMarkersA$

PacingWithHistoryIVTimedVPace

$PacingWithHistoryIGbl$
$PacingVTimedVPace$
$PacingWithHistoryISetEvMarkersV$

C.16 Package `ResponseWithHistoryI`

section *ResponseWithHistoryI* **parents** *ZOO_Toolkit*, *Model_Preamble*, *HistoryCommon*, *CommonPM*, *Respo*

ResponseWithHistoryIGblSt

$ResponseGbl$

ResponseWithHistoryIGbl

$ResponseWithHistoryIGblSt$

$ResponseWithHistoryIIntExtResponseFr == \Delta ResponseWithHistoryIGbl$

ResponseWithHistoryIInit

$ResponseWithHistoryIIntExtResponseFr$
$ResponseInit$

ResponseWithHistoryIVSetBradycardiaMode

$ResponseWithHistoryIIntExtResponseFr$
$ResponseVSetBradycardiaMode$

ResponseWithHistoryIVUpdateBatteryStatus

ResponseWithHistoryIIntExtResponseFr
ResponseVUpdateBatteryStatus

ResponseWithHistoryIVExecuteEvent

ResponseWithHistoryIIntExtResponseFr
ResponseVExecuteEvent

ResponseWithHistoryIVInATRFallBackMode

ResponseWithHistoryIGbl
ResponseVInATRFallBackMode

ResponseWithHistoryIVIncURL

ResponseWithHistoryIIntExtResponseFr
ResponseVIncURL

ResponseWithHistoryIVDecURL

ResponseWithHistoryIIntExtResponseFr
ResponseVDecURL

ResponseWithHistoryIVIncARP

ResponseWithHistoryIIntExtResponseFr
ResponseVIncARP

ResponseWithHistoryIVDecARP

ResponseWithHistoryIIntExtResponseFr
ResponseVDecARP

ResponseWithHistoryIVATROn

ResponseWithHistoryIIntExtResponseFr
ResponseVATROn

ResponseWithHistoryIVATROff

ResponseWithHistoryIIntExtResponseFr
ResponseVATROff

ResponseWithHistoryIVIncATRDuration

ResponseWithHistoryIIntExtResponseFr
ResponseVIncATRDuration

ResponseWithHistoryIVDecATRDuration

ResponseWithHistoryIIntExtResponseFr
ResponseVDecATRDuration

ResponseWithHistoryIVIncATRFbTm

ResponseWithHistoryIIntExtResponseFr
ResponseVIncATRDuration

ResponseWithHistoryIVDecATRFbTm

ResponseWithHistoryIIntExtResponseFr
ResponseVDecATRFbTm

ResponseWithHistoryIVIncATREntryC

ResponseWithHistoryIIntExtResponseFr
ResponseVIncATREntryC

ResponseWithHistoryIVDecATREntryC

ResponseWithHistoryIIntExtResponseFr
ResponseVDecATREntryC

ResponseWithHistoryIVIncATRExitC

ResponseWithHistoryIIntExtResponseFr
ResponseVIncATRExitC

ResponseWithHistoryIVDecATRExitC

ResponseWithHistoryIIntExtResponseFr
ResponseVDecATRExitC

ResponseWithHistoryIVIncVRP

ResponseWithHistoryIIntExtResponseFr
ResponseVIncVRP

ResponseWithHistoryIVDecVRP

ResponseWithHistoryIIntExtResponseFr
ResponseVDecVRP

ResponseWithHistoryIVIncPVARP

ResponseWithHistoryIIntExtResponseFr
ResponseVIncPVARP

ResponseWithHistoryIVDecPVARP

ResponseWithHistoryIIntExtResponseFr
ResponseVDecPVARP

ResponseWithHistoryIVIncPVARPExt

ResponseWithHistoryIIntExtResponseFr
ResponseVIncPVARPExt

ResponseWithHistoryIVDecPVARPExt

ResponseIntExtVentricularResponseFr
ResponseVDecPVARPExt

ResponseWithHistoryIASMarker

ResponseWithHistoryIGbl
$am! : AtrialMarker$
$augm! : AugmentationMarker$
$ev? : PMEvent$

$(atrStatus = NoATR \lor atrStatus = FallbackTmElapsed) \lor (ev? = EvTerminateATR \lor ev? = EvFallbackTm$

$((ev? = EvTerminateATR \lor ev? = EvFallbackTmElapsed) \land augm! = ATREnd) \lor augm! = AugMNone$

$am! = AS$

ResponseWithHistoryIATMarker

$ResponseWithHistoryIGbl$

$am! : AtrialMarker$

$augm! : AugmentationMarker$

$ev? : PMEvent$

$ev? = EvATRDuration \lor atrStatus \neq NoATR$

$am! = AT$

$(ev? = EvATRDuration \land augm! = ATRDur) \lor augm! = AugMNone$

$ResponseWithHistoryISetEvMarkersADF == ResponseWithHistoryIASMarker \lor ResponseWithHistoryIATMa$

ResponseWithHistoryISetEvMarkersA

$ResponseWithHistoryIGbl$

$ResponseWithHistoryISetEvMarkersADF$

$mhy! : Bool$

$msr! : Bool$

$mUp! : Bool$

$mDown! : Bool$

$mhy! = False$

$msr! = False$

$mUp! = False$

$mDown! = False$

ResponseWithHistoryIVSetEvMarkersA

$ResponseWithHistoryIGbl$

$ResponseWithHistoryISetEvMarkersA$

ResponseWithHistoryIAugMWithPVC

$ResponseWithHistoryIGbl$

$ResponseVIsPVC$

$augm! : AugmentationMarker$

$augm! = PVP$

ResponseWithHistoryIAugMWithoutPVC

$$
\begin{array}{l}
ResponseWithHistoryIGbl \\
ResponseVIsPVC \\
augm! : AugmentationMarker \\
\hline
augm! = AugMNone
\end{array}
$$

$$ResponseWithHistoryISetEvMarkersVDF == ResponseWithHistoryIAugMWithPVC \lor ResponseWithHistoryIA$$

ResponseWithHistoryISetEvMarkersV

$$
\begin{array}{l}
ResponseWithHistoryIGbl \\
ResponseWithHistoryISetEvMarkersVDF \\
vm! : VentricularMarker \\
mhy! : Bool \\
msr! : Bool \\
mUp! : Bool \\
mDown! : Bool \\
\hline
vm! = VS \\
mhy! = False \\
msr! = False \\
mUp! = False \\
mDown! = False
\end{array}
$$

ResponseWithHistoryIVSetEvMarkersV

$$
\begin{array}{l}
ResponseWithHistoryIGbl \\
ResponseWithHistoryISetEvMarkersV
\end{array}
$$

ResponseWithHistoryIASenseRefractory

$$
\begin{array}{l}
\Delta ResponseWithHistoryIGbl \\
ResponseVASenseRefractory \\
ResponseWithHistoryISetEvMarkersA \\
mre! : Bool \\
\hline
mre! = True
\end{array}
$$

$$ResponseWithHistoryIASenseRefractoryUFr == ResponseWithHistoryIGbl$$

$$ResponseWithHistoryIASenseRefractoryFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryIASenseR$$

ResponseWithHistoryIVASenseRefractory

$ResponseWithHistoryIASenseRefractoryFr$

$ResponseWithHistoryIASenseRefractory$

ResponseWithHistoryIVSenseRefractory

$\Delta ResponseWithHistoryIGbl$

$ResponseVVSenseRefractory$

$ResponseWithHistoryISetEvMarkersV$

$mre! : Bool$

$mre! = True$

$ResponseWithHistoryIVSenseRefractoryUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryIVSenseRefractoryFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryIVSenseR$

ResponseWithHistoryIVVSenseRefractory

$ResponseWithHistoryIVSenseRefractoryFr$

$ResponseWithHistoryIVSenseRefractory$

ResponseWithHistoryIASenseOnly

$\Delta ResponseWithHistoryIGbl$

$ResponseVASenseOnly$

$ResponseWithHistoryISetEvMarkersA$

$mre! : Bool$

$mre! = False$

$ResponseWithHistoryIASenseOnlyUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryIASenseOnlyFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryIASenseOnlyUF$

ResponseWithHistoryIVASenseOnly

$ResponseWithHistoryIASenseOnlyFr$

$ResponseWithHistoryIASenseOnly$

ResponseWithHistoryIVSenseOnly

$\Delta ResponseWithHistoryIGbl$
$ResponseVVSenseOnly$
$ResponseWithHistoryISetEvMarkersV$
$mre! : Bool$

$mre! = False$

$ResponseWithHistoryIVSenseOnlyUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryIVSenseOnlyFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryIVSenseOnlyUF$

ResponseWithHistoryIVVSenseOnly

$ResponseWithHistoryIVSenseOnlyFr$
$ResponseWithHistoryIVSenseOnly$

ResponseWithHistoryITriggerAPace

$\Delta ResponseWithHistoryIGbl$
$ResponseVTriggerAPace$
$ResponseWithHistoryISetEvMarkersA$
$mre! : Bool$

$mre! = False$

$ResponseWithHistoryITriggerAPaceUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryITriggerAPaceFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryITriggerAPace$

ResponseWithHistoryIVTriggerAPace

$ResponseWithHistoryITriggerAPaceFr$
$ResponseWithHistoryITriggerAPace$

ResponseWithHistoryITriggerVPace

$\Delta ResponseWithHistoryIGbl$
$ResponseVTriggerVPace$
$ResponseWithHistoryISetEvMarkersV$
$mre! : Bool$

$mre! = False$

$ResponseWithHistoryITriggerVPaceUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryITriggerVPaceFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryITriggerVPac$

ResponseWithHistoryIVTriggerVPace

$ResponseWithHistoryITriggerVPaceFr$

$ResponseWithHistoryITriggerVPace$

ResponseWithHistoryIInhibitAPace

$\Delta ResponseWithHistoryIGbl$

$ResponseVInhibitAPace$

$ResponseWithHistoryISetEvMarkersA$

$mre! : Bool$

$mre! = False$

$ResponseWithHistoryIInhibitAPaceUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryIInhibitAPaceFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryIInhibitAPaceU$

ResponseWithHistoryIVInhibitAPace

$ResponseWithHistoryIInhibitAPaceFr$

$ResponseWithHistoryIInhibitAPace$

ResponseWithHistoryIInhibitVPace

$\Delta ResponseWithHistoryIGbl$

$ResponseVInhibitVPace$

$ResponseWithHistoryISetEvMarkersV$

$mre! : Bool$

$mre! = False$

$ResponseWithHistoryIInhibitVPaceUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryIInhibitVPaceFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryIInhibitVPaceU$

ResponseWithHistoryIVInhibitVPace

ResponseWithHistoryIInhibitVPaceFr
ResponseWithHistoryIInhibitVPace

ResponseWithHistoryITrackASense

$\Delta ResponseWithHistoryIGbl$
ResponseVTrackASense
ResponseWithHistoryISetEvMarkersA
$mre! : Bool$

$mre! = False$

$ResponseWithHistoryITrackASenseUFr == ResponseWithHistoryIGbl$

$ResponseWithHistoryITrackASenseFr == \Delta ResponseWithHistoryIGbl \land \Xi ResponseWithHistoryITrackASenseU$

ResponseWithHistoryIVTrackASense

ResponseWithHistoryITrackASenseFr
ResponseWithHistoryITrackASense

C.17 Package `PacemakerWithoutHistory`

section *PacemakerWithoutHistory* **parents** *ZOO_Toolkit*, *Model_Preamble*, *PacingWithHistoryI*, *Sensing*, *Ba*

PacemakerWithoutHistoryGblSt

RateModulationGbl
BatteryGbl
SensingGbl
PacingWithHistoryIGbl
ResponseWithHistoryIGbl

PacemakerWithoutHistoryGbl

PacemakerWithoutHistoryGblSt

$PacemakerWithoutHistoryIntExtPacingWithHistoryIFr == \Delta PacemakerWithoutHistoryGbl$

PacemakerWithoutHistoryVIncRSDown

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncRSDown

PacemakerWithoutHistoryVDecRSDown

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecRSDown

PacemakerWithoutHistoryVIncRSUp

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncRSUp

PacemakerWithoutHistoryVDecRSUp

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecRSUp

PacemakerWithoutHistoryVDynamicDelayOn

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDynamicDelayOn

PacemakerWithoutHistoryVDynamicDelayOff

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDynamicDelayOff

PacemakerWithoutHistoryVIncFixedAVI

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncFixedAVI

PacemakerWithoutHistoryVDecFixedAVI

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecFixedAVI

PacemakerWithoutHistoryVIncMinDynAVDelay

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncMinDynAVDelay

PacemakerWithoutHistoryVDecMinDynAVDelay

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecMinDynAVDelay

PacemakerWithoutHistoryVIncMaxDynAVDelay

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncMaxDynAVDelay

PacemakerWithoutHistoryVDecMaxDynAVDelay

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecMaxDynAVDelay

PacemakerWithoutHistoryVIncSAVDelayOffset

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncSAVDelayOffset

PacemakerWithoutHistoryVDecSAVDelayOffset

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecSAVDelayOffset

PacemakerWithoutHistoryVIncAPulseAmplReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncAPulseAmplReg

PacemakerWithoutHistoryVDecAPulseAmplReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecAPulseAmplReg

PacemakerWithoutHistoryVIncVPulseAmplReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncVPulseAmplReg

PacemakerWithoutHistoryVDecVPulseAmplReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecVPulseAmplReg

PacemakerWithoutHistoryVIncAPulseAmplUReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncAPulseAmplUReg

PacemakerWithoutHistoryVDecAPulseAmplUReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecAPulseAmplUReg

PacemakerWithoutHistoryVIncVPulseAmplUReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncVPulseAmplUReg

PacemakerWithoutHistoryVDecVPulseAmplUReg

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecVPulseAmplUReg

PacemakerWithoutHistoryVIncAPulseWidth

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncAPulseWidth

PacemakerWithoutHistoryVDecAPulseWidth

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecAPulseWidth

PacemakerWithoutHistoryVIncVPulseWidth

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncVPulseWidth

PacemakerWithoutHistoryVDecVPulseWidth

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecVPulseWidth

PacemakerWithoutHistoryVIncLRL

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncLRL

PacemakerWithoutHistoryVDecLRL

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecLRL

PacemakerWithoutHistoryVIncHRL

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVIncHRL

PacemakerWithoutHistoryVDecHRL

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVDecHRL

PacemakerWithoutHistoryVSetTargetCCI

PacemakerWithoutHistoryIntExtPacingWithHistoryIFr
PacingWithHistoryIVSetTargetCCI

$PacemakerWithoutHistoryIntExtSensingFr == \Delta PacemakerWithoutHistoryGbl$

PacemakerWithoutHistoryVIncASensitivity

PacemakerWithoutHistoryIntExtSensingFr
SensingVIncASensitivity

PacemakerWithoutHistoryVDecASensitivity

PacemakerWithoutHistoryIntExtSensingFr
SensingVDecASensitivity

PacemakerWithoutHistoryVIncVSensitivity

PacemakerWithoutHistoryIntExtSensingFr
SensingVIncVSensitivity

PacemakerWithoutHistoryVDecVSensitivity

PacemakerWithoutHistoryIntExtSensingFr
SensingVDecVSensitivity

PacemakerWithoutHistoryVIncABlanking

PacemakerWithoutHistoryIntExtSensingFr
SensingVIncABlanking

PacemakerWithoutHistoryVDecABlanking

PacemakerWithoutHistoryIntExtSensingFr
SensingVDecABlanking

PacemakerWithoutHistoryVIncVBlanking

PacemakerWithoutHistoryIntExtSensingFr
SensingVIncVBlanking

PacemakerWithoutHistoryVDecVBlanking

PacemakerWithoutHistoryIntExtSensingFr
SensingVDecVBlanking

$PacemakerWithoutHistoryIntExtResponseWithHistoryIFr == \Delta PacemakerWithoutHistoryGbl$

PacemakerWithoutHistoryVIncURL

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncURL

PacemakerWithoutHistoryVDecURL

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecURL

PacemakerWithoutHistoryVIncARP

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncARP

PacemakerWithoutHistoryVDecARP

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecARP

PacemakerWithoutHistoryVATROn

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVATROn

PacemakerWithoutHistoryVATROff

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVATROff

PacemakerWithoutHistoryVIncATRDuration

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncATRDuration

PacemakerWithoutHistoryVDecATRDuration

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecATRDuration

PacemakerWithoutHistoryVIncATRFbTm

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncATRFbTm

PacemakerWithoutHistoryVDecATRFbTm

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecATRFbTm

PacemakerWithoutHistoryVIncATREntryC

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncATREntryC

PacemakerWithoutHistoryVDecATREntryC

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecATREntryC

PacemakerWithoutHistoryVIncATRExitC

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncATRExitC

PacemakerWithoutHistoryVDecATRExitC

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecATRExitC

PacemakerWithoutHistoryVIncVRP

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncVRP

PacemakerWithoutHistoryVDecVRP

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecVRP

PacemakerWithoutHistoryVIncPVARP

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncPVARP

PacemakerWithoutHistoryVDecPVARP

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecPVARP

PacemakerWithoutHistoryVIncPVARPExt

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVIncPVARPExt

PacemakerWithoutHistoryVDecPVARPExt

PacemakerWithoutHistoryIntExtResponseWithHistoryIFr
ResponseWithHistoryIVDecPVARPExt

$PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr == \Delta PacemakerWithoutHistoryGbl$

PacemakerWithoutHistoryInit

PacemakerWithoutHistoryGbl
PacingWithHistoryIInit
SensingInit
ResponseWithHistoryIInit
BatteryInit

PacemakerWithoutHistoryVSetBradycardiaMode

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVSetBradycardiaMode
SensingVSetBradycardiaMode
ResponseWithHistoryIVSetBradycardiaMode
BatteryVSetBradycardiaMode

PacemakerWithoutHistoryVUpdateBatteryStatus

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVUpdateBatteryStatus
SensingVUpdateBatteryStatus
ResponseWithHistoryIVUpdateBatteryStatus
BatteryVUpdateBatteryStatus

PacemakerWithoutHistoryVExecuteEvent

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVExecuteEvent
SensingVExecuteEvent
ResponseWithHistoryIVExecuteEvent

PacemakerWithoutHistoryVTriggerAPace

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVTriggerAPace
SensingVTriggerAPace
ResponseWithHistoryIVTriggerAPace

PacemakerWithoutHistoryVTriggerVPace

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVTriggerVPace
SensingVTriggerVPace
ResponseWithHistoryIVTriggerVPace

PacemakerWithoutHistoryVInhibitAPace

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVInhibitAPace
SensingVInhibitAPace
ResponseWithHistoryIVInhibitAPace

PacemakerWithoutHistoryVInhibitVPace

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVInhibitVPace
SensingVInhibitVPace
ResponseWithHistoryIVInhibitVPace

PacemakerWithoutHistoryVTrackASense

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVTrackASense
SensingVTrackASense
ResponseWithHistoryIVTrackASense

PacemakerWithoutHistoryVTimedAPace

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVTimedAPace

PacemakerWithoutHistoryVTimedVPace

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
PacingWithHistoryIVTimedVPace

PacemakerWithoutHistoryVASenseOnly

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
SensingVASenseOnly
ResponseWithHistoryIVASenseOnly

PacemakerWithoutHistoryVVSenseOnly

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
SensingVVSenseOnly
ResponseWithHistoryIVVSenseOnly

PacemakerWithoutHistoryVASenseRefractory

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
SensingVASenseRefractory
ResponseWithHistoryIVASenseRefractory

PacemakerWithoutHistoryVVSenseRefractory

PacemakerWithoutHistoryMergeExtPacingWithHistoryIFr
SensingVVSenseRefractory
ResponseWithHistoryIVVSenseRefractory

C.18 Package `Pacemaker`

section *Pacemaker* **parents** *ZOO_Toolkit*, *Model_Preamble*, *History*, *PacemakerWithoutHistory*

PacemakerGblSt

HistoryGbl
PacemakerWithoutHistoryGbl

PacemakerGbl

$PacemakerGblSt$

$PacemakerIntExtPacemakerWithoutHistoryFr == \Delta PacemakerGbl \land \Xi HistoryGbl$

PacemakerInit

$PacemakerGbl$
$PacemakerWithoutHistoryInit$

PacemakerVIncRSDown

$PacemakerIntExtPacemakerWithoutHistoryFr$
$PacemakerWithoutHistoryVIncRSDown$

PacemakerVDecRSDown

$PacemakerIntExtPacemakerWithoutHistoryFr$
$PacemakerWithoutHistoryVDecRSDown$

PacemakerVIncRSUp

$PacemakerIntExtPacemakerWithoutHistoryFr$
$PacemakerWithoutHistoryVIncRSUp$

PacemakerVDecRSUp

$PacemakerIntExtPacemakerWithoutHistoryFr$
$PacemakerWithoutHistoryVDecRSUp$

PacemakerVDynamicDelayOn

$PacemakerIntExtPacemakerWithoutHistoryFr$
$PacemakerWithoutHistoryVDynamicDelayOn$

PacemakerVDynamicDelayOff

$PacemakerIntExtPacemakerWithoutHistoryFr$
$PacemakerWithoutHistoryVDynamicDelayOff$

PacemakerVIncFixedAVI

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncFixedAVI

PacemakerVDecFixedAVI

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecFixedAVI

PacemakerVIncMinDynAVDelay

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncMinDynAVDelay

PacemakerVDecMinDynAVDelay

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecMinDynAVDelay

PacemakerVIncMaxDynAVDelay

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncMaxDynAVDelay

PacemakerVDecMaxDynAVDelay

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecMaxDynAVDelay

PacemakerVIncSAVDelayOffset

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncSAVDelayOffset

PacemakerVDecSAVDelayOffset

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecSAVDelayOffset

PacemakerVIncAPulseAmplReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncAPulseAmplReg

PacemakerVDecAPulseAmplReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecAPulseAmplReg

PacemakerVIncVPulseAmplReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncVPulseAmplReg

PacemakerVDecVPulseAmplReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecVPulseAmplReg

PacemakerVIncAPulseAmplUReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncAPulseAmplUReg

PacemakerVDecAPulseAmplUReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecAPulseAmplUReg

PacemakerVIncVPulseAmplUReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncVPulseAmplUReg

PacemakerVDecVPulseAmplUReg

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecVPulseAmplUReg

PacemakerVIncAPulseWidth

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncAPulseWidth

PacemakerVDecAPulseWidth

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecAPulseWidth

PacemakerVIncVPulseWidth

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncVPulseWidth

PacemakerVDecVPulseWidth

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecVPulseWidth

PacemakerVIncLRL

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncLRL

PacemakerVDecLRL

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecLRL

PacemakerVIncHRL

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncHRL

PacemakerVDecHRL

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecHRL

PacemakerVSetTargetCCI

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVSetTargetCCI

PacemakerVIncASensitivity

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncASensitivity

PacemakerVDecASensitivity

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecASensitivity

PacemakerVIncVSensitivity

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncVSensitivity

PacemakerVDecVSensitivity

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecVSensitivity

PacemakerVIncABlanking

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncABlanking

PacemakerVDecABlanking

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecABlanking

PacemakerVIncVBlanking

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncVBlanking

PacemakerVDecVBlanking

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecVBlanking

PacemakerVIncURL

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncURL

PacemakerVDecURL

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecURL

PacemakerVIncARP

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncARP

PacemakerVDecARP

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecARP

PacemakerVATROn

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVATROn

PacemakerVATROff

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVATROff

PacemakerVIncATRDuration

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncATRDuration

PacemakerVDecATRDuration

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecATRDuration

PacemakerVIncATRFbTm

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncATRFbTm

PacemakerVDecATRFbTm

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecATRFbTm

PacemakerVIncATREntryC

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncATREntryC

PacemakerVDecATREntryC

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecATREntryC

PacemakerVIncATRExitC

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncATRExitC

PacemakerVDecATRExitC

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecATRExitC

PacemakerVIncVRP

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncVRP

PacemakerVDecVRP

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecVRP

PacemakerVIncPVARP

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncPVARP

PacemakerVDecPVARP

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecPVARP

PacemakerVIncPVARPExt

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVIncPVARPExt

PacemakerVDecPVARPExt

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVDecPVARPExt

PacemakerVSetBradycardiaMode

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVSetBradycardiaMode

PacemakerVUpdateBatteryStatus

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVUpdateBatteryStatus

PacemakerVExecuteEvent

PacemakerIntExtPacemakerWithoutHistoryFr
PacemakerWithoutHistoryVExecuteEvent

$PacemakerJoinExt1Fr == \Delta PacemakerGbl$

PacemakerVASenseRefractory

$PacemakerJoinExt1Fr$
$PacemakerWithoutHistoryVASenseRefractory$
$HistoryCreateHistoryAEv[am!/am?, augm!/augm?, mRe!/mRe?, mhy!/mhy?, msr!/msr?, mUp!/mUp?, mDo$

PacemakerVASenseOnly

$PacemakerJoinExt1Fr$
$PacemakerWithoutHistoryVASenseOnly$
$HistoryCreateHistoryAEv[am!/am?, augm!/augm?, mRe!/mRe?, mhy!/mhy?, msr!/msr?, mUp!/mUp?, mDo$

PacemakerVTriggerAPace

$PacemakerJoinExt1Fr$
$PacemakerWithoutHistoryVTriggerAPace$
$HistoryCreateHistoryAEv[am!/am?, augm!/augm?, mRe!/mRe?, mhy!/mhy?, msr!/msr?, mUp!/mUp?, mDo$

PacemakerVInhibitAPace

$PacemakerJoinExt1Fr$
$PacemakerWithoutHistoryVInhibitAPace$
$HistoryCreateHistoryAEv[am!/am?, augm!/augm?, mRe!/mRe?, mhy!/mhy?, msr!/msr?, mUp!/mUp?, mDo$

PacemakerVTrackASense

$PacemakerJoinExt1Fr$
$PacemakerWithoutHistoryVTrackASense$
$HistoryCreateHistoryAEv[am!/am?, augm!/augm?, mRe!/mRe?, mhy!/mhy?, msr!/msr?, mUp!/mUp?, mDo$

PacemakerVTimedAPace

$PacemakerJoinExt1Fr$
$PacemakerWithoutHistoryVTimedAPace$
$HistoryCreateHistoryAEv[am!/am?, augm!/augm?, mRe!/mRe?, mhy!/mhy?, msr!/msr?, mUp!/mUp?, mDo$

$PacemakerJoinExt2Fr == \Delta PacemakerGbl$

PacemakerVVSenseRefractory

PacemakerJoinExt2Fr
PacemakerWithoutHistoryVVSenseRefractory
HistoryCreateHistoryVEv[*vm*!/*vm*?, *augm*!/*augm*?, *mRe*!/*mRe*?, *mhy*!/*mhy*?, *msr*!/*msr*?, *mUp*!/*mUp*?, *mDo*

PacemakerVVSenseOnly

PacemakerJoinExt2Fr
PacemakerWithoutHistoryVVSenseOnly
HistoryCreateHistoryVEv[*vm*!/*vm*?, *augm*!/*augm*?, *mRe*!/*mRe*?, *mhy*!/*mhy*?, *msr*!/*msr*?, *mUp*!/*mUp*?, *mDo*

PacemakerVTriggerVPace

PacemakerJoinExt2Fr
PacemakerWithoutHistoryVTriggerVPace
HistoryCreateHistoryVEv[*vm*!/*vm*?, *augm*!/*augm*?, *mRe*!/*mRe*?, *mhy*!/*mhy*?, *msr*!/*msr*?, *mUp*!/*mUp*?, *mDo*

PacemakerVTimedVPace

PacemakerJoinExt2Fr
PacemakerWithoutHistoryVTimedVPace
HistoryCreateHistoryVEv[*vm*!/*vm*?, *augm*!/*augm*?, *mRe*!/*mRe*?, *mhy*!/*mhy*?, *msr*!/*msr*?, *mUp*!/*mUp*?, *mDo*

PacemakerVInhibitVPace

PacemakerJoinExt2Fr
PacemakerWithoutHistoryVInhibitVPace
HistoryCreateHistoryVEv[*vm*!/*vm*?, *augm*!/*augm*?, *mRe*!/*mRe*?, *mhy*!/*mhy*?, *msr*!/*msr*?, *mUp*!/*mUp*?, *mDo*

Appendix D

ZOO Toolkit

section *ZOO_Toolkit* **parents** *standard_toolkit*

$[OBJ]$

relation$(opt\ \_)$

$[X]$

$$opt\ \_ : \mathbb{P}(\mathbb{P}\ X)$$
$$the : \mathbb{P}\ X \pfun X$$

$$\forall\ S : \mathbb{P}\ X \bullet opt\ S \Leftrightarrow (\exists\ x : X \bullet S = \{x\}) \lor S = \{\}$$
$$\forall\ x : X \bullet the\ \{x\} = x$$

$[L]$

$$\Sigma : (L \pfun \mathbb{Z}) \to \mathbb{Z}$$

$$\Sigma\ \{\} = 0$$
$$\forall\ l : L;\ n : \mathbb{Z} \bullet \Sigma\ \{(l, n)\} = n$$
$$\forall\ l : L;\ n : \mathbb{Z};\ S : L \pfun \mathbb{Z} \mid \neg\ l \in \mathrm{dom}\ S \bullet \Sigma\ (\{(l, n)\} \cup S) = n + \Sigma\ S$$

$MultTy ::= mm \mid mo \mid om \mid mzo \mid zom \mid mlo \mid lom \mid lolo \mid loo \mid olo \mid lozo \mid zolo \mid oo \mid zozo \mid zoo \mid ozo \mid ms \mid sm \mid ss \mid so \mid os \mid szo \mid zos$

relation$(mult\ \_)$

$[X, Y]$

$mult\_ : \mathbb{P}((X \leftrightarrow Y) \times \mathbb{P}\, X \times \mathbb{P}\, Y \times MultTy \times \mathbb{F}\, \mathbb{N} \times \mathbb{F}\, \mathbb{N})$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, mm, s_1, s_2)) \Leftrightarrow r \in sx \leftrightarrow sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, mo, s_1, s_2)) \Leftrightarrow r \in sx \rightarrow sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, om, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \rightarrow sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, mzo, s_1, s_2)) \Leftrightarrow r \in sx \pfun sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, zom, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \pfun sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, mlo, s_1, s_2)) \Leftrightarrow r \in sx \leftrightarrow sy \land \mathrm{dom}\ r = sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, lom, s_1, s_2)) \Leftrightarrow r \in sx \leftrightarrow sy \land \mathrm{ran}\ r = sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, lolo, s_1, s_2)) \Leftrightarrow r \in sx \leftrightarrow sy \land \mathrm{dom}\ r = sx \land \mathrm{ran}\ r = sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, loo, s_1, s_2)) \Leftrightarrow r \in sx \pfun sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, olo, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \pfun sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, lozo, s_1, s_2)) \Leftrightarrow r \in sx \pfun sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, zolo, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \pfun sx$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, oo, s_1, s_2)) \Leftrightarrow r \in sx \rightarrowtail sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, zozo, s_1, s_2)) \Leftrightarrow r \in sx \pinj sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, zoo, s_1, s_2)) \Leftrightarrow r \in sx \rightarrowtail sy$

$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$
$(mult(r, sx, sy, ozo, s_1, s_2)) \Leftrightarrow r^{\sim} \in sy \rightarrowtail sx$

$$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$$
$$(mult(r, sx, sy, ms, s_1, s_2)) \Leftrightarrow (mult(r, sx, sy, mm, s_1, s_2))$$
$$\land\ (\forall\, x : \mathrm{dom}\, r \bullet \#(\{x\} \lhd r) \in s_1)$$

$$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$$
$$(mult(r, sx, sy, sm, s_1, s_2)) \Leftrightarrow (mult(r, sx, sy, mm, s_1, s_2))$$
$$\land\ (\forall\, y : \mathrm{ran}\, r \bullet \#(r \rhd \{y\}) \in s_1)$$

$$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$$
$$(mult(r, sx, sy, ss, s_1, s_2)) \Leftrightarrow (mult(r, sx, sy, ms, s_1, \{\}))$$
$$\land\ (mult(r, sx, sy, sm, s_2, \{\}))$$

$$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$$
$$(mult(r, sx, sy, so, s_1, s_2)) \Leftrightarrow (mult(r, sx, sy, mo, s_1, s_2))$$
$$\land\ (mult(r, sx, sy, sm, s_1, s_2))$$

$$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$$
$$(mult(r, sx, sy, os, s_1, s_2)) \Leftrightarrow (mult(r, sx, sy, om, \{\}, \{\}))$$
$$\land\ (mult(r, sx, sy, ms, s_1, \{\}))$$

$$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$$
$$(mult(r, sx, sy, szo, s_1, s_2)) \Leftrightarrow (mult(r, sx, sy, mzo, \{\}, \{\}))$$
$$\land\ (mult(r, sx, sy, sm, s_1, \{\}))$$

$$\forall\, r : X \leftrightarrow Y;\ sx : \mathbb{P}\, X;\ sy : \mathbb{P}\, Y;\ s_1, s_2 : \mathbb{F}\, \mathbb{N} \bullet$$
$$(mult(r, sx, sy, zos, s_1, s_2)) \Leftrightarrow (mult(r, sx, sy, zom, \{\}, \{\}))$$
$$\land\ (mult(r, sx, sy, ms, s_1, \{\}))$$