# Software Design Details

* 1. UML Diagrams for Utility Classes

The following diagrams represent the classes and methods within those classes that when called, execute tasks that will allow the Pacemaker to function. The diagrams also give insight into the permissions needed to access particular methods and variable values.

|  |
| --- |
| **class\_name** |
| \*\*variable: type |
| \*\*method: type |

|  |
| --- |
| **main()** |
| patientFirstName: private string  patientLastName: private string  patientAge: private uint18\_t  doctorNotes: private string |
|  |

|  |
| --- |
| **Pacemaker()** |
| deviceID: private const uint32\_t  replaceBattVoltage: private const float  batteryVoltage: private float  cardiacEvents: protected [Object]  leadImpedance: protected float  leadImpedanceThreshold: private float  leadOneInPin: private enum  leadTwoInPin: private enum  leadOneOutPin: private enum  leadTwoOutPin: private enum  maxVOut: private uint16\_t  comPort: private uint8\_t  txRegister: private uint8\_t  rxRegister: private uint8\_t |
| setLeadPins([enum]): protected void  getLeadPins(): protected [enum]  setMaxVOut(uint16\_t): protected void  getMaxVOut(): protected uint16\_t  setComPort(uint8\_t): protected void  getComPort(): protected uint8\_t  setTxRxReg([uint8\_t]): protected void  getTxRxReg(): protected [uint8\_t]  voltageTest(float): protected float  getCardiacEvents(): public Object  clearCardiacEvents(): private void  setLeadImpedance(float): protected void  getLeadImpedance(): protected float |

|  |
| --- |
| **Communications() extends Pacemaker** |
| i\_CommIn: [16bytes]  vraw: uint16\_t  f\_marker: uint16\_t  magnet: int  o\_CommOut: uint8\_t |
| sendEGM(): private int  initEGM(): public void |

|  |
| --- |
| **Sense() extends Pacemaker** |
| chambersSensed: private enum  activityResponse: private enum |
| setChambersSensed(enum): protected void  getChambersSensed(): public enum  setActivityResponse(enum): protected void  getActivityResponse(): public enum |

|  |
| --- |
| **Pace() extends Sense** |
| pacingState: private enum  pacingMode: private enum  hysteresis: private Boolean  hysteresisInterval: private uint16\_t  lowrateInterval: private uint16\_t  vPaceAmp: private uint16\_t  vPaceWidth: private uint16\_t  VRP: private uint16\_t  maxHeartRate: private uint8\_t  baseHeartRate: private uint8\_t |
| setPaceMode(enum): protected void  getPaceMode(): public enum  setPaceState(enum): protected void  getPaceState(): public enum  setHysteresisInterval(uint16\_t): protected void  getHysteresisInterval(): public uint16\_t  setLowRateInterval(uint16\_t): protected void  getLowRateInterval(): public uint16\_t  setvPaceAmp(uint16\_t): protected void  getvPaceAmp(): public uint16\_t  setvPaceWidth(uint16\_t): protected void  getvPaceWidth(): public uint16\_t  setVRP(uint16\_t): protected void  getVRP(): public uint16\_t  setMaxHeartRate(uint8\_t): protected void  getMaxHeartRate(): protected uint8\_t  setBaseHeartRate(uint8\_t): protected void  getMaxHeartRate(): protected uint8\_t |

* 1. Utility Classes

The following tables outline the public, private and protected methods making up each class defined above in section 3.1. Note that the *Sense* and *Communications* classes extend the *Pacemaker* class allowing them to inherit the properties defined in the Pacemaker class. The method *Pace* extends the *Sense* class in order to inherit properties of both Pacemaker and Sense.

Class 1: Pacemaker()

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Return Type** | **Description** | **Next Action (If action event triggered)** |
| setLeadPins([enum]) | void | Sets values for Lead(x)InPin,Lead(x)OutPin based on hardware GPIO requirements | None |
| getLeadPins() | [enum] | Accesses values of Lead(x)InPin,Lead(x)OutPin | None |
| setMaxVOut(uint16\_t) | void | Sets maxVOut variable to maximum safe pace amplitude based on battery capacity | None |
| getMaxVOut() | uint16\_t | Gets vale of maxVOut | None |
| setComPort(uint8\_t) | void | Sets variable comPort which stores com port for serial communication | None |
| getComPort() | uint8\_t | Gets value of comPort variable | None |
| setTxRxReg([uint8\_t]) | void | Sets hex memory locations of Tx and Rx registers storing serial buffer | None |
| getTxRxReg() | [int8\_t] | Gets array of Tx / Rx register locations | None |
| voltageTest(float) | float | Takes arg min pace amplitude and increases voltage until ERM registers P-QRS-T sequence. Returns this voltage. | None |
| getCardiacEvents() | Object | Return object containing all stored cardiac events in EEPROM | None |
| clearCardiacEvents() | void | Erases EEPROM containing stored cardiac event data | None |
| setLeadImpedance(float) | void | Used internally to sense and set value of variable leadImpedance following measurement. | If impedance measured greater than leadImpedanceThreshold, set vPaceAmp in pace class to maxVOut. Log event. |
| getLeadImpedance() | float | Gets value of leadImpedance | None |

Class 2: Sense()

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Return Type** | **Description** | **Next Action (If action event triggered)** |
| setChambersSensed(enum) | void | Takes chambers sensed as enum type {NONE, ATRIUM, VENTRICLE, DUAL} and sets value of private variable chambersSensed | None |
| getChambersSensed() | enum | Returns current value of chambersSensed | None |
| setActivityResponse(enum) | void | Takes activity response as enum type {NONE, TRIGGERED, INHIBITED, DUAL} and sets value of private variable activityResponse | None |
| getActivityResponse() | enum | Returns current value of activityResponse |  |

