# Abstract

This outline seeks to encompass the grand scheme of the Bone Project. The project is broader than just a replacement console. It will provide a framework to create a soul for the living entity that is made up of scoreboard cells that occupy the Electro-Mech Universe.

# Projects

* Bone-in Console
  + No CX Replacement currently planned
  + CX to RS232 Converter
* Scorelink Replacement
  + Wifi
* ScoreNet – Code Name Synergy
  + Internet Connectivity
  + Diagnostics
  + Google Map Display of Cells
  + Repair Logging
  + Legacy Cell Growth
    - Internet capable inline box for any old board
* Scoreboard Apps
  + Graphical Scoreboard Simulator
    - Console App
    - [Console and Scoreboard App](#_Console_and_Scoreboard)
  + Scoreboard Artwork Generator
* Scalability – Bone Enlargement
* SAGE Integration
  + CRM to ScoreNet exchange for Case Information
* Auto-Serialization
  + LX software serials logged by Bone and connected to Sales Order

# Console and Scoreboard App

Follow the Yellow Bricked Bone

GUI to simulate one cell of the ScoreNet. – Mechochondria

* + - **Mechochondria**
      * All of the elements of a location (School, Park, Church, etc.)
        + Consoles
        + Scoreboards
        + Video Elements
        + Communication Nodes

## Simulation\_Window Object Data – UI\_Simulator.py

Currently this is the \_\_main\_\_ file

Purpose is to be a GUI to control the Scoreboard object

* + Default Arguments and Key Word Arguments
    - None
    - This Example
      * sport=MPBASEBALL1, model=LX1030, color=COMPANY\_LOGO
  + Variables
    - B\_JumperChecked=False
    - boardColor=COMPANY\_LOGO
    - boardColorIndex=1
    - C\_JumperChecked=False
    - D\_JumperChecked=False
    - E\_JumperChecked=False
    - keypad3150=False
    - MMBasketball=False
    - model=LX1030
    - modelIndex=0
    - refreshLCD\_frequency=10
  + Lists
    - boardColorList
    - modelList
    - vboseList=[1,0,0]
  + Dictionaries
    - None
  + Objects
    - B\_Jumper=QCheckBox
    - backgroundGrayColor=QColor ('#E7E7E8')
    - board\_color\_radio\_buttons= radio\_button\_widget\_class RadioButtonWidget
    - bottom\_buttons = QHBoxLayout()
    - C\_Jumper=QCheckBox
    - central\_widget=QWidget
    - D\_Jumper=QCheckBox
    - E\_Jumper=QCheckBox
    - keypad=UI\_KeypadLayout.Keypad\_Layout
    - keypad\_3150\_check\_box=QCheckBox
    - keypad\_options\_layout=QVBoxLayout
    - MMBasketball\_check\_box=QCheckBox
    - model\_radio\_buttons=radio\_button\_widget\_class.RadioButtonWidget
    - modelLabel=QGraphicsTextItem
    - new\_scoreboard\_button=QPushButton
    - option\_jumpers\_layout= QVBoxLayout
    - radioFontBig= QFont('arial',30)
    - reverse\_HG\_check\_box=QCheckBox
    - save\_dictionary\_button=QPushButton
    - scene=UI\_digit.GraphicsScene
    - scoreboard=Scoreboard.Scoreboard
    - scoreboard\_view=UI\_digit.GraphicsView
    - select\_model\_button=QPushButton
    - select\_model\_layout=QVBoxLayout
    - select\_model\_widget=QWidget
    - select\_sport\_button=QPushButton
    - select\_sport\_layout\_with\_select=QVBoxLayout
    - select\_sport\_widget=QWidget
    - simulation\_grid=QGridLayout
    - sport\_options\_grid=QHBoxLayout
    - sport \_radio\_buttons=radio\_button\_widget\_class.RadioButtonWidget
    - stacked\_layout=QStackedLayout
    - view\_simulation\_widget=QWidget
  + QTimers
    - refreshLCD=(self.refreshLCD\_frequency)#milliseconds
    - refreshScoreboard=(self.refreshScoreboard\_frequency)#milliseconds
  + Methods – Public
    - None – Button driven

### Simulation\_Window – Initialization

Purpose is to be a GUI to control the Scoreboard object

* main() – Short version of major calls in gui to control the Scoreboard object
  + Simulation\_Window(QMainWindow)
    - \_\_init\_\_()
      * \_\_init\_\_(), super(Simulation\_Window)
        + create\_central\_widget()

create\_select\_sport\_layout()

select\_sport\_button.clicked.connect(create\_select\_model\_layout)

* + - create\_select\_model\_layout()
      * Config.writeSport(sport)
      * buildOptionJumpers()
      * select\_model\_button.clicked.connect(create\_keypad\_layout)
    - create\_keypad\_layout()
      * create\_scoreboard\_view()
      * instantiate\_scoreboard()
        + Scoreboard(model, color='red', driverType='LXDriver', graphicsFlag=1, parent=None, scene=scene, vboseList=vboseList, boardColor=boardColor)

Scoreboard.setKeypad(reverseHomeAndGuest, keypad3150, MMBasketball)

* + - * new\_scoreboard\_button.clicked.connect(create\_central\_widget)

## Console Object Data – Console.py

* + Default Arguments and Key Word Arguments
    - (self, vboseList=[1,0,0])
  + Variables
    - shotClockSports=False
    - verbose=1
      * #Method Name or arguments
    - verboseMore=0
      * #Deeper loop information in methods
    - verboseMost=0
      * #Crazy Deep Stuff
  + Lists
    - vboseList=[1,0,0]
  + Dictionaries
    - [configDict](#configDict)
  + Objects
    - game=Game.Game
    - keyMap = Keypad\_Mapping.[Keypad\_Mapping](#_Keypad_Mapping_Object_Data)
    - lcd = Menu\_Class.[Menu\_Event\_Handler](#_Menu_Event_Handler_Object_Data)
  + Object Dictionaries
  + threading.Timers
    - checkEventsTimer=threading.Timer(self.game.gameSettings['periodClockResolution'], self.checkEvents)
  + Methods – Public
    - setKeypad(self, reverseHomeAndGuest=False, keypad3150=False, MMBasketball=False)
    - keyPressed(self, keyPressed)

### Scoreboard Sub-Class Object Data – Scoreboard.py

This is a sub-class of Console but all elements of both are shown

* + Default Arguments and Key Word Arguments
    - (modelName='LX1030', color='red', driverType='LXDriver', graphicsFlag=0, serialInputFlag=0, parent=None, scene=None, vboseList=[1,0,0], boardColor='TIGER\_ORANGE')
    - Red Highlighted Text is with graphicsFlag=1
  + Variables
    - boardColor=TIGER\_ORANGE
    - color=red
    - driverType=LXDriver
    - graphicParent=None
    - graphicScene=None
    - graphicFlag=0
    - model=LX1030
    - serialInputFlag=0
    - shotClockSports=False
    - verbose=1
      * #Method Name or arguments
    - verboseMore=0
      * #Deeper loop information in methods
    - verboseMost=0
      * #Crazy Deep Stuff
    - boardHeight
    - boardWidth
    - qtyOfCabinets=1
  + Lists
    - addressWordList
    - driverList
    - functionList
    - headerList
    - maskID\_List
    - modelList
    - vboseList=[1,0,0]
  + Dictionaries
    - configDict
      * Dictionary built from the userConfig file
        + userConfig file is in config file format and can be edited by and text editor
        + defaultConfig and userConfig are both generated by running [Config.py](#_Config_Object_Data) in write mode
    - functionDict
      * Dictionary built from <Spreadsheets/Digits_Per_Model.csv>
    - partsDict
      * Dictionary built from <Spreadsheets/Masks_Per_Model.csv>
    - positionDict
      * Dictionary built from <Spreadsheets/Masks_Per_Model.csv>
  + Objects
    - addrMap = Address\_Mapping.[Address\_Mapping](#_Address_Mapping_Object_Data)
    - blankTest = Address\_Mapping.[Blanktest\_Mapping](#_Blanktest_Mapping_Sub-Class_Object)
    - game = [Game](#_Game_Object_Data).[Baseball](#_Baseball_Sub-Class_Object)
    - keyMap = Keypad\_Mapping.[Keypad\_Mapping](#_Keypad_Mapping_Object_Data)
    - lampTest = Address\_Mapping.[Lamptest\_Mapping](#_Lamptest_Mapping_Sub-Class_Object)
    - lcd = Menu\_Class.[Menu\_Event\_Handler](#_Menu_Event_Handler_Object_Data)
    - mp = MP\_Data\_Handler. [MP\_Data\_Handler](#_MP_Data_Handler_Object_Data)
    - board = Scoreboard.[Board](#_Board_Object_Data)
  + Object Dictionaries
    - lxDict = Driver.[LX\_Driver](#_LX_Driver_Object_Data)
    - maskDict = UI\_digit.[Mask](#_Mask_Object_Data)
    - pcbGraphicItemDict = UI\_digit.[PCB](#_PCB_Object_Data)
  + threading.Timers
    - checkEventsTimer=threading.Timer(self.game.gameSettings['periodClockResolution'], self.checkEvents)
    - menuTimer = threading.Timer(self.game.gameSettings['menuTimerDuration'], self.defaultScreen)
    - threading.Timer(self.game.configDict['splashTime'], self.Reset)
  + Methods – Public
    - setKeypad(self, reverseHomeAndGuest=False, keypad3150=False, MMBasketball=False)
    - keyPressed(self, keyPressed)

#### Scoreboard(Console) – Initialization

* + \_\_init\_\_(self, modelName='LX1030', color='red', driverType='LXDriver', graphicsFlag=0, serialInputFlag=0, parent=None, scene=None, vboseList=[1,0,0], boardColor='TIGER\_ORANGE'):
    - super(Scoreboard).\_\_init\_\_(vboseList=vboseList)
      * Console.\_\_init\_\_(vboseList=[1,0,0])
        + Game.readConfig()
        + Game. selectSportInstance(self.[configDict](#configDict)['sport'], numberOfTeams=2, MPLX3450Flag=self.[configDict](#configDict)['MPLX3450Flag'])
        + Menu\_Event\_Handler(sport=self.game.sport, splashTime=splashTime, vboseList=[1,1,0])
    - Address\_Mapping(self.game.gameData['sportType'])
    - Lamptest\_Mapping()
    - Blanktest\_Mapping()
    - MP\_Data\_Handler()
    - readDigitsPerModel()
      * Spreadsheets/Digits\_Per\_Model.csv
    - loadDrivers()
      * for driver in driverList:
        + lxDict[driver]=LX\_Driver(driver, extraJumpers=[])
    - if graphicsFlag:
      * readMasksPerModel()
        + Spreadsheets/Masks\_Per\_Model.csv
      * loadMaskAssemblies()
        + for maskID in self.maskID\_List:

Mask(maskType)

PCB(pcbSize, pcbType, self.color, maskType=maskType)

PCB. addToMask(maskType, positionRtoL, self.maskDict[maskID])

* + - * loadBoard()
        + if self.qtyOfCabinets==2:

self.board\_1=Board(self.model, self.boardColor, scaleFactor)

self.board\_2=Board(self.model, self.boardColor, scaleFactor)

if positionTopToBot==1:

self.maskDict[maskID].addToBoard(self.model, maskID, self.board\_1)

elif positionTopToBot==2:

self.maskDict[maskID].addToBoard(self.model, maskID, self.board\_2)

GraphicsItemGroup(parent=None, scene=self.graphicScene)

GraphicsItemGroup .addToGroup(self.board\_1)

GraphicsItemGroup .addToGroup(self.board\_2)

* + - * + if self.qtyOfCabinets==1:

Board(self.model, self.boardColor, scaleFactor, parent=None, scene=self.graphicScene)

Board.addToGroup(self.board\_1)

Board.addToGroup(self.board\_2)

* + - if serialInputFlag:
      * MP\_Serial\_Handler()
      * Serial\_Packet()
      * threading.Timer(0, self.serialInput).start()
      * threading.Timer(1, self.serialOutputCheck).start()
    - NOT serialInputFlag:
      * + Address\_Mapping.adjustAllBanks(game)
        + data2Drivers(self.addrMap.sendList)

for driver in self.driverList:

lxDict[driver].receive(sendList)

if graphicsFlag:

for all digits, lxDict[LXDriver].updateDisplay(LXHeader, self.pcbGraphicItemDict[function]

## Game Object Data – Game.py

The Game class is a sub-class of all sports

* + Default Arguments and Key Word Arguments
    - (numberOfTeams=2)
  + Static
    - None
  + Variables
    - guest = TEAM\_1
    - home = TEAM\_2
    - numberOfTeams=2
    - sport = GENERIC
  + Lists
    - clockList
    - teamNamesList = [‘TEAM\_1’, TEAM\_2’]
  + Dictionaries
    - [configDict](#configDict)
    - gameData
      * Dictionary built from <Spreadsheets/gameDefaultValues.csv>
    - gameSettings
      * Dictionary built from gameUserSettings file
        + gameUserSettings file is in config file format and can be edited by and text editor
      * gameDefaultSettings and gameUserSettings are both generated by running [GameDefaultSettings.py](#_GameDefaultSettings_Object_Data)
    - segmentTimerSettings
      * Dictionary built from segmentTimerUserSettings file
        + segmentTimerUserSettings file is in config file format and can be edited by and text editor
        + segmentTimerDefaultSettings and segmentTimerUserSettings are both generated by running [SegmentDefaultSettings.py](#_GameDefaultSettings_Object_Data)
  + Objects
    - etn = Not Made Yet
    - optionJumpers = option\_jumpers.OptionJumpers
  + Object Dictionaries
    - clockDict = clock.[clock](#_clock_Object_Data)
    - teamsDict=Team
  + threading.Timers
    - None
  + Methods – Public
    - reverseHomeAndGuest(self)
    - getTeamData(self, team, dataName)
    - getGameData(self, dataName)
    - getClockData(self, clockName, dataName)
    - setTeamData(self, team, dataName, value, places=2)
    - setGameData(self, dataName, value, places=2)
    - setClockData(self, clockName, dataName, value, places=2)
    - modTeamData(self, team, dataName, modulusValue=100, operator = '+', modValue = 1, places=2)
    - modGameData(self, dataName, modulusValue=100, operator = '+', modValue = 1, places=2)
    - modClockData(self, clockName, dataName, operator = '+', modulusValue=60, modValue = 1, places=2)
    - handheldButton1(self)
    - handheldButton2(self)
    - handheldButton3(self)
    - guestScorePlusTen(self)
    - guestScorePlusOne(self)
    - homeScorePlusTen(self)
    - homeScorePlusOne(self)
    - Horn(self)
    - shotHorn(self)
    - delayOfGameHorn(self)
    - hornOff(self)
    - shotHornOff(self)
    - delayOfGameHornOff(self)
    - periodClockOnOff(self)
    - minutesMinusOne(self)
    - secondsMinusOne(self)
    - secondsPlusOne(self)
    - quartersPlusOne(self)
    - periodsPlusOne(self)
    - possession(self)
    - Number\_7\_ABC(self)
    - Number\_8\_DEF(self)
    - Number\_9\_GHI(self)
    - Number\_4\_JKL(self)
    - Number\_5\_MNO(self)
    - Number\_6\_PQR(self)
    - Number\_1\_STU(self)
    - Number\_2\_VWX(self)
    - Number\_3\_YZ(self)
    - Number\_0(self)
    - clear\_(self)
    - enter\_(self)
    - setGuestScore(self)
    - setHomeScore(self)
    - setGuestFunctions(self)
    - setHomeFunctions(self)
    - shotClocks(self)
    - setClock(self)
    - playClocks(self)
    - setClockTenthSec(self)
    - tenthSecOnOff(self)
    - clockUpDown(self)
    - autoHorn(self)
    - timeOfDay(self)
    - timeOutTimer(self)
    - NewGame(self)
    - blank(self)

### Baseball Sub-Class Object Data – Game.py

The Baseball class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - periodClockReset(self)
    - clear\_FlashHit(self)
    - enter\_FlashError(self)
    - hitToggle(self)
    - errorToggle(self)
    - hitsPlusOne(self)
    - errorsPlusOne(self)
    - setSinglePitchCount(self, team)
    - setSinglePitchCountFromMenu(self)
    - teamAtBat(self)
    - singlePitchesPlusOne(self)
    - guestPitchesPlusOne(self)
    - homePitchesPlusOne(self)
    - inningsPlusOne(self)
    - ballsPlusOne(self)
    - strikesPlusOne(self)
    - outsPlusOne(self)
    - incInningTop\_Bot(self)
    - runsPlusOne(self)
    - assignError(self)
    - setPitchCounts(self)
    - setBatterNumber(self)
    - setTotalRuns(self)
    - setTotalHits(self)
    - setTotalErrors(self)
    - setRuns\_Innings(self)
    - setInningTop\_Bot(self)

### Football Sub-Class Object Data – Game.py

The Football class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - periodClockReset(self)
    - guestTimeOutsMinusOne(self)
    - homeTimeOutsMinusOne(self)
    - downsPlusOne(self)
    - yardsToGoMinusTen(self)
    - yardsToGoMinusOne(self)
    - yardsToGoReset(self)
    - setGuestTimeOuts(self)
    - setHomeTimeOuts(self)
    - setYardsToGo(self)
    - setBallOn(self)

### Soccer Sub-Class Object Data – Game.py

The Soccer class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - clear\_GuestGoal(self)
    - enter\_HomeGoal(self)
    - guestPenaltyPlusOne(self)
    - homePenaltyPlusOne(self)
    - guestShotsPlusOne(self)
    - homeShotsPlusOne(self)
    - guestKicksPlusOne(self)
    - homeKicksPlusOne(self)
    - guestSavesPlusOne(self)
    - homeSavesPlusOne(self)

### Hockey Sub-Class Object Data – Game.py

The Hockey class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - handheldButton1(self)
    - handheldButton2(self)
    - handheldButton3(self)
    - clear\_GuestGoal(self)
    - enter\_HomeGoal(self)
    - guestShotsPlusOne(self)
    - homeShotsPlusOne(self)
    - guestPenalty(self)
    - homePenalty(self)

### Basketball Sub-Class Object Data – Game.py

The Basketball class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - handheldButton1(self)
    - handheldButton2(self)
    - handheldButton3(self)
    - periodClockReset(self)
    - guestTimeOutsMinusOne(self)
    - homeTimeOutsMinusOne(self)
    - guestTeamFoulsPlusOne(self)
    - homeTeamFoulsPlusOne(self)
    - guestBonusPlusOne(self)
    - homeBonusPlusOne(self)
    - setGuestTimeOuts(self)
    - setHomeTimeOuts(self)
    - playerMatchGame(self)
    - playerFoul(self)

### Cricket Sub-Class Object Data – Game.py

The Cricket class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - oversPlusOne(self)
    - player1ScorePlusOne(self)
    - player2ScorePlusOne(self)
    - wicketsPlusOne(self)
    - setPlayer1Number(self)
    - setPlayer2Number(self)
    - setPlayer1Score(self)
    - setPlayer2Score(self)
    - setTotalScore(self)
    - setOvers(self)
    - setLastMan(self)
    - setLastWicket(self)
    - set1eInnings(self)

### Racetrack Sub-Class Object Data – Game.py

The Racetrack class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - Not Made Yet

### Stat Sub-Class Object Data – Game.py

The Stat class also contains everything in the Game class

* + Methods – Public (Methods here will replace the Game class version)
    - addPlayer(self)
    - deletePlayer(self)
    - displaySize(self)
    - editPlayer(self)
    - fouls\_digsMinusOne(self)
    - fouls\_digsPlusOne(self)
    - guest\_homeSwitch(self)
    - nextPlayer(self)
    - points\_killsMinusOne(self)
    - points\_killsPlusOne(self)
    - previousPlayer(self)
    - subPlayer(self)

## Team Object Data – Team.py

* + Default Arguments and Key Word Arguments
    - None
  + Variables
    - None
  + Dictionaries
    - configDict
    - teamData
      * Dictionary built from <Spreadsheets/teamDefaultValues.csv>
    - teamSettings
      * Dictionary built from gameUserSettings file
        + gameUserSettings file is in config file format and can be edited by and text editor
        + gameDefaultSettings and gameUserSettings are both generated by running [GameDefaultSettings.py](#_GameDefaultSettings_Object_Data) in write mode
  + Methods – Public
    - None

## OptionJumpers Object Data – option\_jumpers.py

* + Default Arguments and Key Word Arguments
  + Static
  + Variables
  + Lists
  + Dictionaries
  + Objects
  + Object Dictionaries
  + threading.Timers
  + Methods – Public

## Config Object Data – Config.py

* + Default Arguments and Key Word Arguments
    - (self, write=False, fileType='user')
  + Variables
    - fileType='user'
    - tick= system time
    - tock= system time 2
    - write=False
  + Dictionaries
    - configDict
      * Dictionary built from code
    - configFile
      * Dictionary read from any config file
    - userConfigDict
      * Dictionary built from userConfig file
  + Methods – Public
    - getDict(self)
    - user2default(self)
      * Make userConfig = defaultConfig
    - writeOptionJumpers(self, optionJumpers)
    - writeSport(self, sport)

## GameDefaultSettings Object Data – GameDefaultSettings.py

* + Default Arguments and Key Word Arguments
    - (self, write=False, fileType='user')
  + Variables
    - fileType='user'
    - tick= system time
    - tock= system time 2
    - write=False
  + Dictionaries
    - gameDefaultSettings
      * Dictionary built from code
    - gameDefaultSettingsFile
      * Dictionary read from any config file
    - gameUserSettings
      * Dictionary built from gameUserSettings file
  + Methods – Public
    - getDict(self)

## SegmentTimerDefaultSettings Object Data – SegmentTimerDefaultSettings.py

* + Default Arguments and Key Word Arguments
    - (self, write=False, fileType='user')
  + Variables
    - fileType='user'
    - tick= system time
    - tock= system time 2
    - write=False
  + Dictionaries
    - segmentTimerSettings
      * Dictionary built from code
    - segmentTimerSettingsFile
      * Dictionary read from any config file
    - segmentTimerUserSettings
      * Dictionary built from segmentTimerUserSettings file
  + Methods – Public
    - getDict(self)

## clock Object Data – clock.py

* + Default Arguments and Key Word Arguments
    - (countUp=False, maxSeconds=86399.999, resolution=0.001, hoursFlag=False, clockName='generic', internalClock=False)
  + Variables
    - \_start
    - \_stop
    - autoStop
    - blinky
    - changeTime
    - clockName
    - countUp
    - currentTime
    - days
    - daysTens
    - daysUnits
    - hours
    - hoursFlag
    - hoursTens
    - hoursUnits
    - hundredthsUnits
    - internalClock
    - maxSeconds
    - minutes
    - minutesTens
    - minutesUnits
    - next\_call
    - PM
    - resolution
    - running
    - seconds
    - secondsTens
    - secondsUnits
    - tenths\_hundredths
    - tenthsUnits
  + Lists
    - timeList
  + threading.Timers
    - refresh = threading.Timer( self.next\_call - time.time(), self.Update).start()
  + Methods – Public
    - Update(self)
    - gameDataUpdate(self, [gameData](#gameData), name='periodClock')
    - changeSeconds(self, change)
    - Start(self)
    - Stop(self)
    - Reset(self, resetValueSeconds=None)

## Keypad\_Mapping Object Data – Keypad\_Mapping.py

This object generates a map of console keys connected to that game object method

* + Status
    - Completed – 3/9/2015
  + Default Arguments and Key Word Arguments
    - (self, game, reverseHomeAndGuest=False, keypad3150=False, MMBasketball=False)
  + Variables
    - funcString=’’
    - keypad3150=False
    - keypadName
    - MMBasketball=False
    - reverseHomeAndGuest=False
    - keyPressFlag = False
  + Dictionaries
    - Keypad\_Keys
      * Built from selected row of <Spreadsheets/MP_Keypad_Layouts.csv>
  + Object Dictionaries
    - gameFuncDict
      * Dictionary of pointers to all of the selected games methods
  + Methods – Public
    - Map(self, game, keyPressed)
      * Matches grid location to function name
        + All combinations of B through F with 8 through 1
      * Calls game object method
      * Raises the keyPressFlag
        + Used by the Menu\_Class

## Menu\_Event\_Handler Object Data – Menu\_Class.py

This object performs all tasks related to the LCD Display and menus

* + Default Arguments and Key Word Arguments
    - (self, sport='MPBASEBALL1', splashTime=5, vboseList=[0,0,0])
  + Variables
    - addVariableFlag=True
    - clearFlag=False
    - col=0
    - currentData=None
    - currentMenuString=’’
    - dimmimgMenuFlag=False
    - endingMenuNumber=1
    - enterFlag=False
    - funcString=’’
    - internalRefreshFlag=False
    - lastCol=0
    - lastPlaces=1
    - lastRow=0
    - lastTeam=255
    - lastVarClock=False
    - lastVarName=’’
    - lcdTextDisplay=Blank
    - mappedFlag=False
    - menuFlag
    - menuNumber
    - menuTimerFlag
    - NewGameMenu=1
    - numberPressedFlag
    - numpadSequence=None
    - places=1
    - precisionMenuFlag=False
    - row=0
    - row1=’’
    - row2=’’
    - segmentTimerMenuFlag=False
    - splashTime=5
    - splashTimerFlag=False
    - sport
    - startFlag=False
    - startingMenuNumber=1
    - team=None
    - teamNameMenuFlag=False
    - teamNameNumpadFlag=False
    - teamNameNumpadFlagCount=0
    - teamNameString=’’
    - timerNumberGuest=1
    - timerNumberHome=1
    - varClock=None
    - varName=None
    - verbose=0
    - verboseMore=0
    - verboseMost=0
  + Lists
    - lastBlockNumList
    - numberPressedSequence
    - vboseList
  + Dictionaries
    - funcDict
    - gameSettings
    - Menu\_LCD\_Text
    - tempDict
  + Objects
    - None
  + threading.Timers
    - None (Menu Timing is a concept higher than this object)
  + Methods – Public
    - Map(self, game, funcString=‘None’)
      * Call the function to check if this funcString has a menu or only uses it in certain cases
      * Check current menu state and branch to appropriate internal method type
        + startFunc
        + selfFunc
        + clearFunc
        + enterFunc
        + numpadFunc
        + exitMenu
        + doNothing
      * Refresh the default screen if menuFlag is False
      * Return the Game object with any changes in data made

## Address\_Mapping Object Data – Address\_Mapping.py

This object generates the current data for the scoreboards addresses from the game object data

* + Default Arguments and Key Word Arguments
    - (self, sportType='Generic')
  + Variables
    - sport
    - sportType=Generic
    - verbose=False
  + Lists
    - sendList
    - sendListAddr
    - seq
  + Dictionaries
    - addressMapDict
    - [configDict](#configDict)
    - fullAddressMapDict
    - wordsDict
  + Objects
    - mp = MP\_Data\_Handler. [MP\_Data\_Handler](#_Keypad_Mapping_Object_Data)
  + Object Dictionaries
    - None
  + threading.Timers
    - None
  + Methods – Public
    - Map(self, game)
      * A map of all outputs is created based on <Spreadsheets/AddressMap.csv>
      * Flags are checked for alternate word configurations on the spreadsheet
      * Map is updated with alternate word configurations
      * Each word configuration fetches the data stored in the game object and prepares it in the MP\_Data\_Handler format
        + group, bank, word, iBit, hBit, highNibble, lowNibble, blankType, segmentData
      * The Encode method of the MP\_Data\_Handler converts this to the 2 Byte word scoreboard language and stores it in the sendList
      * External functions access the sendList to send data that has changed

### Blanktest\_Mapping Sub-Class Object Data – Address\_Mapping.py

The Blanktest\_Mapping class also contains everything in the Address\_Mapping class

This object has a map with every LED off and the Word 4 I-Bit On (Active Bit)

* + Default Arguments and Key Word Arguments
    - None
  + Methods – Public
    - Map(self, game)
      * Does nothing

### Lamptest\_Mapping Sub-Class Object Data – Address\_Mapping.py

The Lamptest\_Mapping class also contains everything in the Address\_Mapping class

This object has a map with LEDs On based on the sport

* + Default Arguments and Key Word Arguments
    - None
  + Methods – Public
    - Map(self, game)
      * Does nothing

## MP\_Data\_Handler Object Data – MP\_Data\_Handler.py

This object converts between MP\_Data\_Handler format the 2 Byte word scoreboard language

* + Default Arguments and Key Word Arguments
    - None
  + Variables
    - addr=0
    - bank=0
    - blankType=’’
    - data=0
    - group=0
    - H\_Bit=0
    - highByte=0
    - I\_Bit=0
    - lowByte=0
    - seg\_addr=0
    - segmentData=’’
    - verbose=False
    - word=0
  + Methods – Public
    - Encode (self, group, bank, word, I\_Bit, H\_Bit, highNibble, lowNibble, blankType, segmentData)
    - Decode(self, LH\_Word)

## LX\_Driver Object Data – Driver.py

This object simulates the behavior of an LX Driver. It is a sub-class of Driver.

* + Default Arguments and Key Word Arguments
    - (self, driverName='GENERIC')
  + Variables
    - bank
    - brightness
    - displayEnable=1
    - driverName=GENERIC
    - driverType=LXDriver
    - group
    - horn1=0
    - horn2=0
    - J10=0
    - J4=0
    - J5=0
    - J6=0
    - J8=0
    - J9=0
    - lastWord=0
    - legacyDimmingEnable=0
    - letters=HGFEDCBA
    - quantumDimmingTunnel=0
    - verbose=False
  + Lists
    - blankTypes=
    - extraJumpers=[]
    - headers=[J4,J5,J6,J8,J9,J10]
    - segments=[A,B,C,D,E,F,G,H]
  + Dictionaries
    - jumperDict
    - modeDict
    - outputDict
    - word1BlankingJumpersDict
    - word2BlankingJumpersDict
  + Objects
    - mp = MP\_Data\_Handler. [MP\_Data\_Handler](#_Keypad_Mapping_Object_Data)
  + threading.Timers
    - None
  + Methods – Public
    - receive(self, sendList)
      * Mimics LX Driver behavior and stores result in outputDict
      * If graphics are being used it updates segment info to match

## Board Object Data – Graphical – Scoreboard.py

This Object is a sub-class of the PyQt object GraphicsItem or GraphicsItemGroup

* + Default Arguments and Key Word Arguments
    - (self, model='LX1030', color='TIGER\_ORANGE', scaleFactor=8, parent=None, scene=None)
  + Static
    - colorDict
      * Dictionary of the key=paint color name, value=RGB hex number
  + Variables
    - color=TIGER\_ORANGE
    - boardHeight
    - boardWidth
    - model=LX1030
    - scaleFactor=8
    - verbose=False
  + Lists
    - None
  + Dictionaries
    - partsDict
    - positionDict
  + Objects
    - boundingRect=PyQt4.QtCore.QRectF(x,y,width,height)
    - color=PyQt4.QtGui.QColor
    - parent=None
    - scene=None
  + Object Dictionaries
    - None
  + Methods – Public
    - None in this subclass

### Mask Object Data – UI\_digit.py

* + Default Arguments and Key Word Arguments
    - (self, maskType=16inClock, scaleFactor=1, parent=None)
    - This Example
      * Mask('16inClock')
      * PCB('16', 'digit') .addToMask('16inClock', 1, Mask)
      * PCB('16', 'digit') .addToMask('16inClock', 2, Mask)
      * PCB('16', ‘colonDec’) .addToMask('16inClock', 3, Mask)
      * PCB('16', 'digit') .addToMask('16inClock', 4, Mask)
      * PCB('16', 'digit') .addToMask('16inClock', 5, Mask)
  + Variables
    - maskID=scorelink
    - maskHeight=18.9
    - maskRadius=0.5
    - maskType=16inClock
    - maskWidth=47.8
    - model=LX1030
    - scaleFactor=1
  + Lists
    - positionRtoL\_List=[‘1’,’3’,’2’,’5’,’4’]
  + Dictionaries
    - partsDict
    - positionDict
  + Objects
    - boundingRect = QRectF(0, 0, self.maskWidth, self.maskHeight)
    - parent=None
  + Methods – Public
    - addToBoard(self, model='LX1030', maskID='scorelink', parent=None)

#### PCB Object Data – UI\_digit.py

* + Default Arguments and Key Word Arguments
    - (self, pcbSize='18', pcbType='digit', color='amber', parent=None, maskType=None)
    - This Example
      * PCB('16', 'digit')
  + Variables
    - LED\_ON\_Color=amber
    - maskHeight=18.9
    - maskRadius=0.5
    - maskType=16inClock
    - maskWidth=47.8
    - pcbSize=16
    - pcbType=digit
  + Lists
    - ledsPerSegment
    - positionRtoL\_List
    - segmentList=[‘A’,’C’,‘B’,’E’,’D’,’G’,’F’]
  + Dictionaries
    - partsDict
    - positionDict
    - segmentDict
    - specs
  + Objects
    - boundingRect = QRectF(0, 0, self.maskWidth, self.maskHeight)
  + Object Dictionaries
    - segDict = LED
  + Methods – Public
    - addToMask(self, maskType='18inTo9', position=1, parent=None)
    - updateSegment(self, segment='A', SEG\_On=1)

##### LED Object Data – UI\_digit.py

* + Default Arguments and Key Word Arguments
    - (self, LEDs\_On=0, color='amber', scorelink=0, parent=None, pcbSize='18', pcbType='digit')
  + Static
    - ledOnSize18=.875
    - ledOnSize30=1.0625
    - ledOnSize11=.8125
    - ledOnSize3=.6875
    - ledOnSize12=.475
    - ledOnSizeBP=.425
    - ledOnSize16=.75
    - ledOnSizeElseIndoor=.6
    - ledOnSizeETN7=.8
    - ledOnSizeETN11=.825
    - ledOnSizeETN5=.525
    - ledOnSizeETN8=.545
    - ledOffSize=.25
    - AMBER\_LED="#FAA61A"
    - RED\_LED="#ED1C24"
    - GREEN\_LED="#A6CE39"
    - OFF\_LED="#434A4F"
    - WHITE="#FFFFFF"
  + Variables
    - centerAdjust=0.125
    - ledSize=0.25
    - scorelink=0
  + Objects
    - boundingRect = QRectF(-0.125,-1.125,0.25,0.25)
    - color=QColor
    - parent=None
  + Methods – Public
    - None

## RadioButtonWidget Object Data – radio\_button\_widget\_class.py

* + Default Arguments and Key Word Arguments
  + Static
  + Variables
  + Lists
  + Dictionaries
  + Objects
  + Object Dictionaries
  + threading.Timers
  + Methods – Public

## Keypad\_Layout Object Data – UI\_Keypad\_Layout.py

* + Default Arguments and Key Word Arguments
  + Static
  + Variables
  + Lists
  + Dictionaries
  + Objects
  + Object Dictionaries
  + threading.Timers
  + Methods – Public

## Object Data

* + Default Arguments and Key Word Arguments
  + Static
  + Variables
  + Lists
  + Dictionaries
  + Objects
  + Object Dictionaries
  + threading.Timers
  + Methods – Public