Baja Dashboard Widget Classes

Generated by Doxygen 1.8.17

Chapter 1

Dash_Widjet_Classes

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Widget_Classes Namespace Reference

Defines custom Widget classes for the Pyside6 python library to use in a digital dashboard with windows and linux support.

Classes

- class Fuel_Gauge

  A custom QWidget-based fuel gauge display widget.
- class Menu

  A widget that displays a scrollable menu interface with selectable modes.
- class Rotating_Image

  A QWidget that cycles through a list of PNG images at a set interval.
- class Speedometer

  A QWidget-based speedometer that visually displays a numerical speed value in MPH.
- class Tachometer

  A custom QWidget-based tachometer display widget.
- class Temp_Gauge

  A QWidget-based vertical temperature gauge with gradient fill, tick marks and a value indicator.
- class Variable_Section

  A QWidget that handles a wide variety of uses for the Bobcat Baja team.
- class Warning_Light

  A QWidget-based warning light that displays a PNG image when activated.

6.1.1 Detailed Description

Defines custom Widget classes for the Pyside6 python library to use in a digital dashboard with windows and linux support.

Some features such as the custom font will not work for anyone without the font package. These widgets were originally designed for use on the Bobcat Baja Team.

Chapter 7

Class Documentation

7.1 Widget_Classes.Fuel_Gauge Class Reference

A custom QWidget-based fuel gauge display widget.

Public Member Functions

- def __init__ (self, parent=None, max_value=100)

  Constructor for the Fuel_Gauge widget.
- def add_to_value (self, change)

  Add to the fuel gauge value.
- def update_value (self, new_value)

  Update the tachometer gauge value.
- def paintEvent (self, event)

  Handle the paint event for rendering the gauge.

Public Attributes

- value
- max_value

7.1.1 Detailed Description

A custom QWidget-based fuel gauge display widget.

This widget displays a circular fuel gauge that updates its display based on fuel level, including color-coded sections and tick marks. The percentage value of the current fuel is displayed inside the half-circle.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 __init__()

```
def Widget_Classes.Fuel_Gauge.__init__ (
              self,
              parent = None,
              max_value = 100 )
```

Constructor for the Fuel_Gauge widget.

Parameters

parent	The parent widget (optional).
max_value	The maximum value the fuel gauge can represent.

7.1.3 Member Function Documentation

7.1.3.1 add_to_value()

```
def Widget_Classes.Fuel_Gauge.add_to_value (
              self,
              change )
```

Add to the fuel gauge value.

Parameters

change	The add amount to change the current value by. The new value wraps around if it exceeds the max value.

7.1.3.2 paintEvent()

```
def Widget_Classes.Fuel_Gauge.paintEvent (
              self,
              event )
```

Handle the paint event for rendering the gauge.

This method draws the fuel gauge, gradient fill, tick marks, and text labels. It also includes a half-circle and and value/unit display. Also handles resizing.

Parameters

event	The paint event triggered by Qt.

7.1.3.3 update_value()

```
def Widget_Classes.Fuel_Gauge.update_value (
              self,
              new_value )
```

Update the tachometer gauge value.

Parameters

new_value	The new value for the gauge to be set to. The new value wraps around if it exceeds the max value.

7.1.4 Member Data Documentation

7.1.4.1 max_value

```
Widget_Classes.Fuel_Gauge.max_value
```

7.1.4.2 value

```
Widget_Classes.Fuel_Gauge.value
```

The documentation for this class was generated from the following file:

- Widget_Classes.py

7.2 Widget_Classes.Menu Class Reference

A widget that displays a scrollable menu interface with selectable modes.

Public Member Functions

- def __init__ (self, parent=None, modes=[ ])

  Constructor for the Menu widget.
- def update_value (self, move_value)

  Updates the currently selected menu state.
- def paintEvent (self, event)

  Paint event handler that draws the menu bar and its current state.

Public Attributes

- modes
- num_of_modes
- state

7.2.1 Detailed Description

A widget that displays a scrollable menu interface with selectable modes.

The Menu widget provides a simple left-right navigable menu bar that appears at the bottom of the window. It takes the middle option as the currently selected mode and displays the previous and next items in the menu list.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 __init__()

```
def Widget_Classes.Menu.__init__ (
              self,
              parent = None,
              modes = [] )
```

Constructor for the Menu widget.

Parameters

parent	The parent widget (optional).
modes	A list of mode names to cycle through in the menu.

7.2.3 Member Function Documentation

7.2.3.1 paintEvent()

```
def Widget_Classes.Menu.paintEvent (
              self,
              event )
```

Paint event handler that draws the menu bar and its current state.

Parameters

event	The QPaintEvent that triggered the paint update.

7.2.3.2 update_value()

```
def Widget_Classes.Menu.update_value (
              self,
              move_value )
```

Updates the currently selected menu state.

Parameters

move_value	The number of steps to move in the modes list (positive or negative).

7.2.4 Member Data Documentation

7.2.4.1 modes

```
Widget_Classes.Menu.modes
```

7.2.4.2 num_of_modes

```
Widget_Classes.Menu.num_of_modes
```

7.2.4.3 state

```
Widget_Classes.Menu.state
```

The documentation for this class was generated from the following file:

- Widget_Classes.py

7.3 Widget_Classes.Rotating_Image Class Reference

A QWidget that cycles through a list of PNG images at a set interval.

Public Member Functions

- def __init__ (self, parent=None, png_paths=[ ], height=100, width=100, image_time=10000)

  Constructor for the Rotating_Image widget.
- def resizeEvent (self, event)

  Handles widget resizing and scales the image accordingly.
- def change_image (self)

  Advances to the next image in the list and updates the display.

Public Attributes

- png_paths
- num_of_paths
- current_path
- timer
- label

7.3.1 Detailed Description

A QWidget that cycles through a list of PNG images at a set interval.

The Rotating_Image widget displays one image at a time from a provided list of PNG paths. It automatically rotates to the next image after a fixed time interval. The displayed image resizes dynamically with the widget while maintaining its aspect ratio.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 __init__()

```
def Widget_Classes.Rotating_Image.__init__ (
              self,
              parent = None,
              png_paths = [],
              height = 100,
              width = 100,
              image_time = 10000 )
```

Constructor for the Rotating_Image widget.

Parameters

parent	The parent widget (optional).
png_paths	List of file paths to PNG images for rotation.
height	Initial height of the image display area.
width	Initial width of the image display area.
image_time	The length of time in milliseconds that a image is displayed before it changes.

7.3.3 Member Function Documentation

7.3.3.1 change_image()

```
def Widget_Classes.Rotating_Image.change_image (
              self )
```

Advances to the next image in the list and updates the display.

This method is called automatically by a timer. It cycles through the image list and applies scaling to fit the widget's dimensions.

7.3.3.2 resizeEvent()

```
def Widget_Classes.Rotating_Image.resizeEvent (
              self,
              event )
```

Handles widget resizing and scales the image accordingly.

Parameters

event	QResizeEvent triggered when the widget is resized.

7.3.4 Member Data Documentation

7.3.4.1 current_path

```
Widget_Classes.Rotating_Image.current_path
```

7.3.4.2 label

```
Widget_Classes.Rotating_Image.label
```

7.3.4.3 num_of_paths

```
Widget_Classes.Rotating_Image.num_of_paths
```

7.3.4.4 png_paths

```
Widget_Classes.Rotating_Image.png_paths
```

7.3.4.5 timer

```
Widget_Classes.Rotating_Image.timer
```

The documentation for this class was generated from the following file:

- Widget_Classes.py

7.4 Widget_Classes.Speedometer Class Reference

A QWidget-based speedometer that visually displays a numerical speed value in MPH.

Public Member Functions

- def __init__ (self, parent=None, max_value=200)

  Constructor for the Speedometer widget.
- def add_to_value (self, change)

  Adds to the speedometer's value by a specified change.
- def update_value (self, new_value)

  Update the speedometer gauge value.
- def paintEvent (self, event)

  Handle the paint event for rendering the speedometer.

Public Attributes

- value
- max_value

7.4.1 Detailed Description

A QWidget-based speedometer that visually displays a numerical speed value in MPH.

This widget presents a digital-style speedometer readout. The speed value is updated dynamically and rendered in a custom font, centered on the widget. The display adapts its text alignment based on the number of digits in the speed.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 __init__()

```
def Widget_Classes.Speedometer.__init__ (
              self,
              parent = None,
              max_value = 200 )
```

Constructor for the Speedometer widget.

Parameters

parent	The parent widget (optional).
max_value	The maximum speed value that can be displayed.

7.4.3 Member Function Documentation

7.4.3.1 add_to_value()

```
def Widget_Classes.Speedometer.add_to_value (
              self,
              change )
```

Adds to the speedometer's value by a specified change.

Parameters

change	The amount to adjust the current speed by. Wraps around on overflow.

7.4.3.2 paintEvent()

```
def Widget_Classes.Speedometer.paintEvent (
              self,
              event )
```

Handle the paint event for rendering the speedometer.

This method draws the speedometer, which is just a text label with RPM added to the end of it.

Parameters

event	The paint event triggered by Qt.

7.4.3.3 update_value()

```
def Widget_Classes.Speedometer.update_value (
              self,
              new_value )
```

Update the speedometer gauge value.

Parameters

new_value	The new value for the gauge to be set to. The new value wraps around if it exceeds the max value.

7.4.4 Member Data Documentation

7.4.4.1 max_value

```
Widget_Classes.Speedometer.max_value
```

7.4.4.2 value

```
Widget_Classes.Speedometer.value
```

The documentation for this class was generated from the following file:

- Widget_Classes.py

7.5 Widget_Classes.Tachometer Class Reference

A custom QWidget-based tachometer display widget.

Public Member Functions

- def __init__ (self, parent=None, max_value=5000)

  Constructor for the Tachometer widget.
- def add_to_value (self, change)

  Add to the tachometer gauge value.
- def update_value (self, new_value)

  Update the tachometer gauge value.
- def paintEvent (self, event)

  Handle the paint event for rendering the gauge.

Public Attributes

- value
- max_value
- gaugeGrad

7.5.1 Detailed Description

A custom QWidget-based tachometer display widget.

This widget visually represents RPM (revolutions per minute) values using a combination of an arc and straight gauge bar. The gauge dynamically updates and fills using a gradient based on the current value.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 __init__()

```
def Widget_Classes.Tachometer.__init__ (
              self,
              parent = None,
              max_value = 5000 )
```

Constructor for the Tachometer widget.

Parameters

parent	The parent widget (optional).
max_value	The maximum RPM value shown on the tachometer.

7.5.3 Member Function Documentation

7.5.3.1 add_to_value()

```
def Widget_Classes.Tachometer.add_to_value (
              self,
              change )
```

Add to the tachometer gauge value.

Parameters

change	The add amount to change the current value by. The new value wraps around if it exceeds the max value.

7.5.3.2 paintEvent()

```
def Widget_Classes.Tachometer.paintEvent (
              self,
              event )
```

Handle the paint event for rendering the gauge.

This method draws the tachometer gauge, gradient fill, tick marks, and text labels. It also includes a half-circle and strait line and value/unit display. Also handles resizing.

Parameters

event	The paint event triggered by Qt.

7.5.3.3 update_value()

```
def Widget_Classes.Tachometer.update_value (
              self,
              new_value )
```

Update the tachometer gauge value.

Parameters

new_value	The new value for the gauge to be set to. The new value wraps around if it exceeds the max value.

7.5.4 Member Data Documentation

7.5.4.1 gaugeGrad

```
Widget_Classes.Tachometer.gaugeGrad
```

7.5.4.2 max_value

```
Widget_Classes.Tachometer.max_value
```

7.5.4.3 value

```
Widget_Classes.Tachometer.value
```

The documentation for this class was generated from the following file:

- Widget_Classes.py

7.6 Widget_Classes.Temp_Gauge Class Reference

A QWidget-based vertical temperature gauge with gradient fill, tick marks and a value indicator.

Public Member Functions

- def __init__ (self, parent=None, max_value=300)

  Constructor for the Temp_Gauge widget.
- def add_to_value (self, change)

  Add to the temperature gauge value.
- def update_value (self, new_value)

  Update the temperature gauge value.
- def paintEvent (self, event)

  Handle the paint event for rendering the gauge.

Public Attributes

- value
- max_value
- gaugeGrad

7.6.1 Detailed Description

A QWidget-based vertical temperature gauge with gradient fill, tick marks and a value indicator.

This custom widget visually represents a temperature value using a colored bar, tick marks, and numeric labels. The bar fills from green to red as the temperature increases.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 __init__()

```
def Widget_Classes.Temp_Gauge.__init__ (
              self,
              parent = None,
              max_value = 300 )
```

Constructor for the Temp_Gauge widget.

Parameters

parent	The parent widget (optional).
max_value	The maximum temperature value the gauge can represent. Scales the gauge values based on this value.

7.6.3 Member Function Documentation

7.6.3.1 add_to_value()

```
def Widget_Classes.Temp_Gauge.add_to_value (
              self,
              change )
```

Add to the temperature gauge value.

Parameters

change	The add amount to change the current value by. The new value wraps around if it exceeds the max value.

7.6.3.2 paintEvent()

```
def Widget_Classes.Temp_Gauge.paintEvent (
              self,
              event )
```

Handle the paint event for rendering the gauge.

This method draws the temperature bar, gradient fill, tick marks, and text labels. It also includes a circle and value/unit display. Also handles resizing.

Parameters

event	The paint event triggered by Qt.

7.6.3.3 update_value()

```
def Widget_Classes.Temp_Gauge.update_value (
              self,
              new_value )
```

Update the temperature gauge value.

Parameters

new_value	The new value for the gauge to be set to. The new value wraps around if it exceeds the max value.

7.6.4 Member Data Documentation

7.6.4.1 gaugeGrad

```
Widget_Classes.Temp_Gauge.gaugeGrad
```

7.6.4.2 max_value

```
Widget_Classes.Temp_Gauge.max_value
```

7.6.4.3 value

```
Widget_Classes.Temp_Gauge.value
```

The documentation for this class was generated from the following file:

- Widget_Classes.py

7.7 Widget_Classes.Variable_Section Class Reference

A QWidget that handles a wide variety of uses for the Bobcat Baja team.

Public Member Functions

- def __init__ (self, parent=None)

    Constructor for the variable section widget.
- def change_widget (self, new_widget)

    Changes the variable section to the new widget that is passed to it and redraws the new widget.
- def paintEvent (self, event)

    Paint event handler that calls the function that the variable widget is set to by its current state.
- def draw_Startup (self, painter)

    Draws the startup instructional overlay on the widget.
- def draw_Competition (self, painter)

    Draws the Competition overlay on the widget.
- def draw_Rear_Steer (self, painter)

    Draws the Rear Steer overlay on the widget.
- def draw_Two_Step (self, painter)

    Draws the two-Step RPM limit adjustment interface.
- def update_two_step (self, value)

    Updates the currently selected two-step digit by a given increment or decrement.
- def move_two_step (self)

    Handles the progression through the two-step digit selection and boundary-setting process.
- def draw_Lap_Time (self, painter)

    Draws the Lap Time overlay on the widget.
- def draw_Error (self, painter)

    Draws the Error overlay on the widget.

Public Attributes

- widget_types
- current_widget
- new_two_step_bounds
- two_step_digit_list
- selected_two_step_digit
- two_step_bound
- two_step_current_digit_values
- two_step_value_up
- two_step_bad_input
- size_factor

7.7.1 Detailed Description

A QWidget that handles a wide variety of uses for the Bobcat Baja team.

∗Bold∗ Not recommend to use as is outside of Bobcat Baja.

A set of custom widgets that have specific uses for Bobcat Baja. These currently include "Startup", "Competition", "Rear Steer", "Two-Step" and "Lap Time" although they are not all complete.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 __init__()

```
def Widget_Classes.Variable_Section.__init__ (
              self,
              parent = None )
```

Constructor for the variable section widget.

Initializes to "Startup" sub-widget.

Parameters

parent	The parent widget (optional).

7.7.3 Member Function Documentation

7.7.3.1 change_widget()

```
def Widget_Classes.Variable_Section.change_widget (
              self,
              new_widget )
```

Changes the variable section to the new widget that is passed to it and redraws the new widget.

Parameters

new_widget	The new widget to be displayed.

7.7.3.2 draw_Competition()

```
def Widget_Classes.Variable_Section.draw_Competition (
              self,
              painter )
```

Draws the Competition overlay on the widget.

Parameters

painter	QPainter object used for rendering text on the widget.

This function is not complete.

7.7.3.3 draw_Error()

```
def Widget_Classes.Variable_Section.draw_Error (
              self,
              painter )
```

Draws the Error overlay on the widget.

Parameters

painter	QPainter object used for rendering text on the widget.

This function only occurs when the widget selected doesn't have a draw function for it.

7.7.3.4 draw_Lap_Time()

```
def Widget_Classes.Variable_Section.draw_Lap_Time (
              self,
              painter )
```

Draws the Lap Time overlay on the widget.

Parameters

painter	QPainter object used for rendering text on the widget.

This function is not complete.

7.7.3.5 draw_Rear_Steer()

```
def Widget_Classes.Variable_Section.draw_Rear_Steer (
              self,
              painter )
```

Draws the Rear Steer overlay on the widget.

Parameters

painter	QPainter object used for rendering text on the widget.

This function is not complete.

7.7.3.6 draw_Startup()

```
def Widget_Classes.Variable_Section.draw_Startup (
              self,
              painter )
```

Draws the startup instructional overlay on the widget.

Parameters

painter	QPainter object used for rendering text on the widget.

This function displays startup instructions for both the HID Control and Switch Control systems. It sets the pen and font style, defines the drawing rectangle, and renders a multiline instructional string aligned to the top-left of the widget using word wrapping.

7.7.3.7 draw_Two_Step()

```
def Widget_Classes.Variable_Section.draw_Two_Step (
              self,
              painter )
```

Draws the two-Step RPM limit adjustment interface.

Parameters

painter	QPainter object used to render the visual interface.

This function draws the two-step RPM bounds adjustment screen, which allows the user to configure both the upper or lower RPM limits using four editable digits. A title is displayed to indicate the bound currently being edited, and if bad input was previously received, an error message is shown instead. Each digit is visually represented, with one digit highlighted to indicate the current selection.

7.7.3.8 move_two_step()

```
def Widget_Classes.Variable_Section.move_two_step (
              self )
```

Handles the progression through the two-step digit selection and boundary-setting process.

If the currently selected digit is the RPM label (end of the sequence), then:

- If setting the "Upper" bound, the digit values are converted into an integer and saved as the upper limit.
- If setting the "Lower" bound, the digit values are saved as the lower limit. If the lower limit is not less than the upper, a flag is raised to show bad input. Otherwise, a transition to the Competition widget is triggered.

If the current selection is not the RPM digit, moves selection to the next digit in the digit list.

Always triggers a repaint of the widget to reflect current state.

7.7.3.9 paintEvent()

```
def Widget_Classes.Variable_Section.paintEvent (
              self,
              event )
```

Paint event handler that calls the function that the variable widget is set to by its current state.

Parameters

event	The QPaintEvent that triggered the paint update.

7.7.3.10 update_two_step()

```
def Widget_Classes.Variable_Section.update_two_step (
              self,
              value )
```

Updates the currently selected two-step digit by a given increment or decrement.

Parameters

value	Integer value to apply (+1 to increment, -1 to decrement the selected digit).

This function modifies one of the four digit values used for setting the two-step RPM limit, depending on which digit is currently selected. Each digit is constrained between 0-9 by modding by 10. The function then requests a repaint of the widget to reflect the updated value.

7.7.4 Member Data Documentation

7.7.4.1 current_widget

```
Widget_Classes.Variable_Section.current_widget
```

7.7.4.2 new_two_step_bounds

```
Widget_Classes.Variable_Section.new_two_step_bounds
```

7.7.4.3 selected_two_step_digit

```
Widget_Classes.Variable_Section.selected_two_step_digit
```

7.7.4.4 size_factor

`Widget_Classes.Variable_Section.size_factor`

7.7.4.5 two_step_bad_input

`Widget_Classes.Variable_Section.two_step_bad_input`

7.7.4.6 two_step_bound

`Widget_Classes.Variable_Section.two_step_bound`

7.7.4.7 two_step_current_digit_values

`Widget_Classes.Variable_Section.two_step_current_digit_values`

7.7.4.8 two_step_digit_list

`Widget_Classes.Variable_Section.two_step_digit_list`

7.7.4.9 two_step_value_up

`Widget_Classes.Variable_Section.two_step_value_up`

7.7.4.10 widget_types

`Widget_Classes.Variable_Section.widget_types`

The documentation for this class was generated from the following file:

- Widget_Classes.py

7.8 Widget_Classes.Warning_Light Class Reference

A QWidget-based warning light that displays a PNG image when activated.

Public Member Functions

- def __init__ (self, parent=None, png_path=None, height=100, width=100)

  Constructor for the Warning_Light widget.
- def show_light (self)

  Toggles the warning light to be "On".
- def hide_light (self)

  Toggles the warning light to be "Off".
- def resizeEvent (self, event)

  Handle resize events and update label and pixmap accordingly.

Public Attributes

- png_path
- label

7.8.1 Detailed Description

A QWidget-based warning light that displays a PNG image when activated.

This widget is designed to show or hide a warning light image, useful in GUIs where visual alerts are needed. Can Be resized to fit any need when initialized.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 __init__()

```
def Widget_Classes.Warning_Light.__init__ (
              self,
              parent = None,
              png_path = None,
              height = 100,
              width = 100 )
```

Constructor for the Warning_Light widget.

Parameters

parent	The parent widget (optional).
png_path	Path to the PNG image to display as the warning light.
height	Initial height of the widget in pixels.
width	Initial width of the widget in pixels.

7.8.3 Member Function Documentation

7.8.3.1 hide_light()

```
def Widget_Classes.Warning_Light.hide_light (
              self )
```

Toggles the warning light to be "Off".

Call this function to make the warning light not visible.

7.8.3.2 resizeEvent()

```
def Widget_Classes.Warning_Light.resizeEvent (
              self,
              event )
```

Handle resize events and update label and pixmap accordingly.

This ensures the widget fills the specified area and the pixmap is scaled to fit while maintaining its aspect ratio.

Parameters

event	The resize event.

7.8.3.3 show_light()

```
def Widget_Classes.Warning_Light.show_light (
              self )
```

Toggles the warning light to be "On".

Call this function to make the warning light visible.

7.8.4 Member Data Documentation

7.8.4.1 label

```
Widget_Classes.Warning_Light.label
```

7.8.4.2 png_path

```
Widget_Classes.Warning_Light.png_path
```

The documentation for this class was generated from the following file:

- Widget_Classes.py

Chapter 8

File Documentation

8.1 README.md File Reference

8.2 Widget_Classes.py File Reference

Classes

- class Widget_Classes.Warning_Light

  A QWidget-based warning light that displays a PNG image when activated.
- class Widget_Classes.Temp_Gauge

  A QWidget-based vertical temperature gauge with gradient fill, tick marks and a value indicator.
- class Widget_Classes.Tachometer

  A custom QWidget-based tachometer display widget.
- class Widget_Classes.Fuel_Gauge

  A custom QWidget-based fuel gauge display widget.
- class Widget_Classes.Speedometer

  A QWidget-based speedometer that visually displays a numerical speed value in MPH.
- class Widget_Classes.Menu

  A widget that displays a scrollable menu interface with selectable modes.
- class Widget_Classes.Rotating_Image

  A QWidget that cycles through a list of PNG images at a set interval.
- class Widget_Classes.Variable_Section

  A QWidget that handles a wide variety of uses for the Bobcat Baja team.

Namespaces

- Widget_Classes

  Defines custom Widget classes for the Pyside6 python library to use in a digital dashboard with windows and linux support.

Index