

User interface software documentation

Generated by Doxygen 1.8.13

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Namespace Documentation	7
4.1	command Namespace Reference	7
4.1.1	Detailed Description	7
4.2	enable_leg Namespace Reference	7
4.2.1	Detailed Description	7
4.3	error_handler Namespace Reference	8
4.3.1	Detailed Description	8
4.4	Field Namespace Reference	8
4.4.1	Detailed Description	8
4.5	leg_data_tab Namespace Reference	8
4.5.1	Detailed Description	8
4.6	main Namespace Reference	8
4.6.1	Detailed Description	9
4.7	uart_communication Namespace Reference	9
4.7.1	Detailed Description	9
4.8	user_interface Namespace Reference	9
4.8.1	Detailed Description	9

5	Class Documentation	11
5.1	command.command Class Reference	11
5.1.1	Detailed Description	11
5.1.2	Constructor & Destructor Documentation	12
5.1.2.1	__init__()	12
5.1.3	Member Function Documentation	12
5.1.3.1	__call__()	12
5.1.3.2	calc_value()	12
5.1.4	Member Data Documentation	12
5.1.4.1	command_type	13
5.2	uart_connection.connection Class Reference	13
5.2.1	Constructor & Destructor Documentation	13
5.2.1.1	__del__()	14
5.2.1.2	__init__()	14
5.2.2	Member Function Documentation	14
5.2.2.1	__call__()	14
5.2.2.2	send()	14
5.3	field.DISPLAY_FIELD Class Reference	14
5.4	enable_leg.ENABLE_LEG Class Reference	15
5.4.1	Detailed Description	15
5.4.2	Constructor & Destructor Documentation	15
5.4.2.1	__init__()	15
5.4.3	Member Function Documentation	16
5.4.3.1	get_data()	16
5.4.3.2	start()	16
5.4.3.3	stop()	16
5.5	field.ENTRY_FIELD Class Reference	16
5.5.1	Constructor & Destructor Documentation	17
5.5.1.1	__init__()	17
5.5.2	Member Function Documentation	17

5.5.2.1	<code>get_data()</code>	17
5.5.2.2	<code>start()</code>	17
5.5.2.3	<code>stop()</code>	18
5.6	<code>error_handler.ERROR_HANDLER</code> Class Reference	18
5.6.1	Detailed Description	18
5.6.2	Constructor & Destructor Documentation	18
5.6.2.1	<code>__init__()</code>	19
5.6.3	Member Function Documentation	19
5.6.3.1	<code>amplitude_error()</code>	19
5.6.3.2	<code>frequency_error()</code>	19
5.6.3.3	<code>phase_error()</code>	19
5.6.3.4	<code>pwm_frequency_error()</code>	19
5.6.3.5	<code>show_error_message()</code>	20
5.7	<code>field.FIELD</code> Class Reference	20
5.7.1	Detailed Description	20
5.7.2	Constructor & Destructor Documentation	20
5.7.2.1	<code>__init__()</code>	21
5.8	<code>user_interface.GUI</code> Class Reference	21
5.8.1	Detailed Description	22
5.8.2	Constructor & Destructor Documentation	22
5.8.2.1	<code>__init__()</code>	23
5.8.3	Member Function Documentation	23
5.8.3.1	<code>check_connection()</code>	23
5.8.3.2	<code>destroy_window()</code>	23
5.8.3.3	<code>disable_all()</code>	23
5.8.3.4	<code>enable_all()</code>	23
5.8.3.5	<code>new_data()</code>	24
5.8.3.6	<code>start_button_event()</code>	24
5.8.3.7	<code>stop_button_event()</code>	24
5.8.3.8	<code>update_all_fields()</code>	24
5.8.3.9	<code>update_button_event()</code>	24
5.9	<code>leg_data_tab.LEG_DATA_TAB</code> Class Reference	25
5.9.1	Detailed Description	25
5.9.2	Constructor & Destructor Documentation	25
5.9.2.1	<code>__init__()</code>	25
5.9.3	Member Function Documentation	26
5.9.3.1	<code>stop()</code>	26
5.9.3.2	<code>update()</code>	26

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

command	This module encodes commands into the right format	7
enable_leg	Enable leg creates the checkbox which is associated with enabling a specific leg at the top of the UI	7
error_handler	This module is responsible for the error handling of the user interface input	8
Field	This module is responsible for generating entry fields where the user can put in values and display fields where the data is being displayed	8
leg_data_tab	This module creates the data which is displayed in the different tabs	8
main	This module acts as the main for the user interface	8
uart_communication	This module sets up the serial communication between the PC and the microcontroller	9
user_interface	This module is the main graphical module of the user_interface	9

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

command.command	11
uart_connection.connection	13
enable_leg.ENABLE_LEG	15
error_handler.ERROR_HANDLER	18
user_interface.GUI	21
leg_data_tab.LEG_DATA_TAB	25
object	
field.FIELD	20
field.DISPLAY_FIELD	14
field.ENTRY_FIELD	16

Chapter 3

Class Index

3.1 Class List

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

command.command	
The command class encodes binary commands into the correct format according to the syntax of the communication protocol	11
uart_connection.connection	13
field.DISPLAY_FIELD	14
enable_leg.ENABLE_LEG	
The ENABLE_LEG class creates the checkboxes which are, in our case, present at the top of the UI	15
field.ENTRY_FIELD	16
error_handler.ERROR_HANDLER	
The ERROR_HANDLER Class returns an error message whenever the user puts in prohibited characters or values	18
field.FIELD	
The FIELD Class is the parent class to ENTRY_FIELD and DISPLAY_FIELD	20
user_interface.GUI	
The GUI class forms the main frame of the User Interface	21
leg_data_tab.LEG_DATA_TAB	
LEG_DATA_TAB creates the labels corresponding to the data which is displayed into the different tabs	25

Chapter 4

Namespace Documentation

4.1 command Namespace Reference

This module encodes commands into the right format.

Classes

- class [command](#)

The [command](#) class encodes binary commands into the correct format according to the syntax of the communication protocol.

4.1.1 Detailed Description

This module encodes commands into the right format.

4.2 enable_leg Namespace Reference

Enable leg creates the checkbox which is associated with enabling a specific leg at the top of the UI.

Classes

- class [ENABLE_LEG](#)

The [ENABLE_LEG](#) class creates the checkboxes which are, in our case, present at the top of the UI.

4.2.1 Detailed Description

Enable leg creates the checkbox which is associated with enabling a specific leg at the top of the UI.

4.3 error_handler Namespace Reference

This module is responsible for the error handling of the user interface input.

Classes

- class [ERROR_HANDLER](#)

The [ERROR_HANDLER](#) Class returns an error message whenever the user puts in prohibited characters or values.

4.3.1 Detailed Description

This module is responsible for the error handling of the user interface input.

4.4 Field Namespace Reference

This module is responsible for generating entry fields where the user can put in values and display fields where the data is being displayed.

4.4.1 Detailed Description

This module is responsible for generating entry fields where the user can put in values and display fields where the data is being displayed.

4.5 leg_data_tab Namespace Reference

This module creates the data which is displayed in the different tabs.

Classes

- class [LEG_DATA_TAB](#)

[LEG_DATA_TAB](#) creates the labels corresponding to the data which is displayed into the different tabs.

4.5.1 Detailed Description

This module creates the data which is displayed in the different tabs.

4.6 main Namespace Reference

This module acts as the main for the user interface.

Variables

- `root` = `Tk()`
- `gui` = `user_interface.GUI(root)`

4.6.1 Detailed Description

This module acts as the main for the user interface.

The only function it has is looping the user interface

4.7 `uart_communication` Namespace Reference

This module sets up the serial communication between the PC and the microcontroller.

4.7.1 Detailed Description

This module sets up the serial communication between the PC and the microcontroller.

It also sends data over the serial connection.

4.8 `user_interface` Namespace Reference

This module is the main graphical module of the `user_interface`.

Classes

- class `GUI`
The `GUI` class forms the main frame of the User Interface.

4.8.1 Detailed Description

This module is the main graphical module of the `user_interface`.

It creates the checkboxes, datalabels and dataentreis. It also creates the buttons and the tabs.

Chapter 5

Class Documentation

5.1 `command.command` Class Reference

The `command` class encodes binary commands into the correct format according to the syntax of the communication protocol.

Public Member Functions

- `def __init__` (self, com, value=0, channel=0)
The constructor can take three arguments.
- `def __call__` (self)
The call method creates the message with the startbyte being \$ and the binary representation of the command, value and the CRC checksum.
- `def calc_value` (self)
The calc_value method creates a representation of the value in bytes.
- `def crc` (self, message)
The CRC method calculates the CRC checksum.

Public Attributes

- **value**
- **type**
- **command**

Static Public Attributes

- dictionary `command_type`
command_type is a dictionary with the binary representation of the command, including if it is channel dependent or not.

5.1.1 Detailed Description

The `command` class encodes binary commands into the correct format according to the syntax of the communication protocol.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `__init__()`

```
def command.command.__init__ (
    self,
    com,
    value = 0,
    channel = 0 )
```

The constructor can take three arguments.

Com being the command, value being the value which is set to 0 if there is no value and channel being the channel, which is set to 0 if there is no channel.

5.1.3 Member Function Documentation

5.1.3.1 `__call__()`

```
def command.command.__call__ (
    self )
```

The call method creates the message with the startbyte being \$ and the binary representation of the command, value and the CRC checksum.

5.1.3.2 `calc_value()`

```
def command.command.calc_value (
    self )
```

The `calc_value` method creates a representation of the value in bytes.

5.1.4 Member Data Documentation

5.1.4.1 command_type

dictionary command.command.command_type [static]

Initial value:

```
= { 'frequency'      : [0x01,0],
    'amplitude'     : [0x10,1],
    'keyfrequency'  : [0x20,0],
    'phaseshift'    : [0x30,1],
    'info'          : [0x00,0],
    'ping'          : [0xFF,0],
    'prepare'       : [0x02,0],
    'execute'       : [0x03,0],
    'start'         : [0x04,0],
    'gather'        : [0x40,1],
    'stop'          : [0x05,0],
    'VFD'           : [0x06,0],
    'enable'        : [0x50,1]
}
```

command_type is a dictionary with the binary representation of the command, including if it is channel dependent or not.

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

- command.py

5.2 uart_connection.connection Class Reference

Public Member Functions

- def `__del__` (self)
The destructor destroys objects if they are not being used anymore.
- def `__init__` (self)
The constructors checks if there is a arduino present and connects to it if it is.
- def `__call__` (self)
The call method returns a boolean which represents if the microcontrolller is connected.
- def `send` (self, command)
The send method sends a binary formatted command over serial communication to the microcontroller.

Public Attributes

- `ser`
- `var`

5.2.1 Constructor & Destructor Documentation

5.2.1.1 `__del__()`

```
def uart_connection.connection.__del__ (
    self )
```

The destructor destroys objects if they are not being used anymore.

5.2.1.2 `__init__()`

```
def uart_connection.connection.__init__ (
    self )
```

The constructors checks if there is a arduino present and connects to it if it is.

If there is no Arduino present it displays a error message.

5.2.2 Member Function Documentation

5.2.2.1 `__call__()`

```
def uart_connection.connection.__call__ (
    self )
```

The call method returns a boolean which represents if the microcontrolller is connected.

5.2.2.2 `send()`

```
def uart_connection.connection.send (
    self,
    command )
```

The send method sends a binary formatted command over serial communication to the microcontroller.

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

- `uart_connection.py`

5.3 `field.DISPLAY_FIELD` Class Reference

Inheritance diagram for `field.DISPLAY_FIELD`:

5.4 enable_leg.ENABLE_LEG Class Reference

The [ENABLE_LEG](#) class creates the checkboxes which are, in our case, present at the top of the UI.

Public Member Functions

- `def __init__ (self, row, column, tab_nr, ID)`
The constructor takes four arguments.
- `def get_data (self)`
The get_data method returns the state of the checkbox.
- `def start (self)`
The start method disables the state of the checkbox whenever the start button is pressed.
- `def stop (self)`
The stop method enables the state of the checkbox whenever the stop button is pressed.

Public Attributes

- `tab_nr`
- `row`
- `column`
- `ID`
- `checkbutton_var`
- `checkbutton`

5.4.1 Detailed Description

The [ENABLE_LEG](#) class creates the checkboxes which are, in our case, present at the top of the UI.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 __init__()

```
def enable_leg.ENABLE_LEG.__init__ (
    self,
    row,
    column,
    tab_nr,
    ID )
```

The constructor takes four arguments.

Row and column being the geometrical location of the checkbox. tab_nr being the tab where the checkbox should be generated and ID being which leg the checkbox should be identified with

5.4.3 Member Function Documentation

5.4.3.1 `get_data()`

```
def enable_leg.ENABLE_LEG.get_data (
    self )
```

The `get_data` method returns the state of the checkbox.

5.4.3.2 `start()`

```
def enable_leg.ENABLE_LEG.start (
    self )
```

The `start` method disables the state of the checkbox whenever the start button is pressed.

5.4.3.3 `stop()`

```
def enable_leg.ENABLE_LEG.stop (
    self )
```

The `stop` method enables the state of the checkbox whenever the stop button is pressed.

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

- `enable_leg.py`

5.5 `field.ENTRY_FIELD` Class Reference

Inheritance diagram for `field.ENTRY_FIELD`:

Collaboration diagram for `field.ENTRY_FIELD`:

Public Member Functions

- `def __init__` (self, row, parameter, tab_nr, field_type, padding=(0, 0), leg_nr=0)
The constructor inherits from [FIELD](#) and creates Entries and labels corresponding to the entries.
- `def start` (self, checkbox_status=0)
The `start` method enables a checkbox if the checkbox is enabled and disables it if the checkbox is disabled.
- `def get_data` (self)
The `get_data` method returns the data which the user put into the entry if the format is correct, it displays a error message if there are prohibited characters or values put in.
- `def stop` (self)
The `stop` method deletes the content of a entry and disables it.

Public Attributes

- `parameter_entry`
- `new_data`

5.5.1 Constructor & Destructor Documentation

5.5.1.1 `__init__()`

```
def field.ENTRY_FIELD.__init__ (
    self,
    row,
    parameter,
    tab_nr,
    field_type,
    padding = (0,0),
    leg_nr = 0 )
```

The constructor inherits from `FIELD` and creates Entries and labels corresponding to the entries.

5.5.2 Member Function Documentation

5.5.2.1 `get_data()`

```
def field.ENTRY_FIELD.get_data (
    self )
```

The `get_data` method returns the data which the user put into the entry if the format is correct, it displays a error message if there are prohibited characters or values put in.

5.5.2.2 `start()`

```
def field.ENTRY_FIELD.start (
    self,
    checkbutton_status = 0 )
```

The `start` method enables a checkbutton if the checkbutton is enabled and disables it if the checkbutton is disabled.

5.5.2.3 stop()

```
def field.ENTRY_FIELD.stop (
    self )
```

The stop method deletes the content of a entry and disables it.

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

- field.py

5.6 error_handler.ERROR_HANDLER Class Reference

The [ERROR_HANDLER](#) Class returns an error message whenever the user puts in prohibited characters or values.

Public Member Functions

- `def __init__ (self, parameter, value)`
The constructor of the [ERROR_HANDLER](#) takes the parameter and value and stores them in variables.
- `def __call__ (self)`
The call method returns one of the errorchecking functions, depending on the parameter.
- `def frequency_error (self)`
The frequency_error method error checks the frequency for prohibited characters and values.
- `def pwm_frequency_error (self)`
The pwm_frequency_error method error checks the PWM frequency for prohibited characters and values.
- `def amplitude_error (self)`
The amplitude_error method error checks the amplitude for prohibited characters and values.
- `def phase_error (self)`
The phase_error method error checks the phase for prohibited characters and values.
- `def show_error_message (self)`
The show_error_message shows an error messag whenever a prohibited character or value is given in the entry.

Public Attributes

- **parameter**
- **value**
- **error**
- **message**
- **error_handler**

5.6.1 Detailed Description

The [ERROR_HANDLER](#) Class returns an error message whenever the user puts in prohibited characters or values.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 `__init__()`

```
def error_handler.ERROR_HANDLER.__init__ (
    self,
    parameter,
    value )
```

The constructor of the [ERROR_HANDLER](#) takes the parameter and value and stores them in variables.

5.6.3 Member Function Documentation

5.6.3.1 `amplitude_error()`

```
def error_handler.ERROR_HANDLER.amplitude_error (
    self )
```

The `amplitude_error` method error checks the amplitude for prohibited characters and values.

5.6.3.2 `frequency_error()`

```
def error_handler.ERROR_HANDLER.frequency_error (
    self )
```

The `frequency_error` method error checks the frequency for prohibited characters and values.

5.6.3.3 `phase_error()`

```
def error_handler.ERROR_HANDLER.phase_error (
    self )
```

The `phase_error` method error checks the phase for prohibited characters and values.

5.6.3.4 `pwm_frequency_error()`

```
def error_handler.ERROR_HANDLER.pwm_frequency_error (
    self )
```

The `pwm_frequency_error` method error checks the PWM frequency for prohibited characters and values.

5.6.3.5 show_error_message()

```
def error_handler.ERROR_HANDLER.show_error_message (
    self )
```

The show_error_message shows an error messag whenever a prohibited character or value is given in the entry.

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

- error_handler.py

5.7 field.FIELD Class Reference

The [FIELD](#) Class is the parent class to [ENTRY_FIELD](#) and [DISPLAY_FIELD](#).

Inheritance diagram for field.FIELD:

Collaboration diagram for field.FIELD:

Public Member Functions

- def [__init__](#) (self, row, parameter, tab_nr, field_type, padding=(0, 0), leg_nr=0)
The constructor takes a row for the geometrical location of the display.

Public Attributes

- **leg_nr**
- **row**
- **parameter**
- **tab_nr**
- **field_type**
- **padding**
- **parameter_unit**

5.7.1 Detailed Description

The [FIELD](#) Class is the parent class to [ENTRY_FIELD](#) and [DISPLAY_FIELD](#).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `__init__()`

```
def field.FIELD.__init__(
    self,
    row,
    parameter,
    tab_nr,
    field_type,
    padding = (0,0),
    leg_nr = 0 )
```

The constructor takes a row for the geometrical location of the display.

It takes a parameter which indicates the parameter of the field. `tab_nr` indicates in which tab the field should be generated. `Field_type` indicates if it should be a entry or a display field. `Padding` and `leg_nr` are additional arguments, `padding` creates a keepout zone between the fields if necessary and `leg_nr` the leg of which the argument belongs to.

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

- `field.py`

5.8 user_interface.GUI Class Reference

The `GUI` class forms the main frame of the User Interface.

Public Member Functions

- `def __init__(self, master)`
The constructor takes one argument, master, which is the actual tkinter main frame of the user interface.
- `def destroy_window(self)`
The method destroy_window destroys the user_interface object whenever the close program button in the menu is pressed.
- `def check_connection(self)`
The check_connection method checks if there is a connection with a microcontroller available.
- `def start_button_event(self)`
The start_button_event method creates a serial connection if available.
- `def stop_button_event(self)`
The stop_button_event method destroys the serial connection if it was present.
- `def update_button_event(self)`
The update_button_event method updates all the data labels if there is new data updated.
- `def new_data(self)`
The new_data method stores new data in a list if new data was updated.
- `def update_all_fields(self)`
The update_all_fields method updates all the data labels and entries.
- `def enable_all(self)`
The enable_all method enables the data entries of the legs that are enabled through the enable leg checkbox.
- `def disable_all(self)`
The disable_all method disables all the data entries, it sets the buttons to the standard setting.
- `def disable_tabs(self)`
The disable_tabs method cleans all the data labels in the tabs whenever stop is being pressed by the user.

Public Attributes

- master
- connection_label
- enable_leg_1
- enable_leg_2
- enable_leg_3
- enable_leg_4
- start_button
- update_button
- stop_button
- frequency_entry
- pwm_frequency_entry
- amplitude_entry
- amplitude_leg_4_entry
- phase_1_entry
- phase_2_entry
- phase_3_entry
- phase_4_entry
- frequency_display
- pwm_frequency_display
- amplitude_display
- amplitude_leg_4_display
- phase_1_display
- phase_2_display
- phase_3_display
- phase_4_display
- tab_2
- tab_3
- tab_4
- tab_5
- connection
- input_values
- new_values
- connection_available
- connection_flag
- connction_available
- new_frequency
- new_pwm_frequency
- new_amplitude
- new_phase_1
- new_phase_2
- new_phase_3
- new_phase_4
- new_amplitude_leg_4

5.8.1 Detailed Description

The [GUI](#) class forms the main frame of the User Interface.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 `__init__()`

```
def user_interface.GUI.__init__ (
    self,
    master )
```

The constructor takes one argument, `master`, which is the actual tkinter main frame of the user interface.

5.8.3 Member Function Documentation

5.8.3.1 `check_connection()`

```
def user_interface.GUI.check_connection (
    self )
```

The `check_connection` method checks if there is a connection with a microcontroller available.

If it is, it changes the display to connected, if not it stays at disconnected.

5.8.3.2 `destroy_window()`

```
def user_interface.GUI.destroy_window (
    self )
```

The method `destroy_window` destroys the [user_interface](#) object whenever the close program button in the menu is pressed.

5.8.3.3 `disable_all()`

```
def user_interface.GUI.disable_all (
    self )
```

The `disable_all` method disables all the data entries, it sets the buttons to the standard setting.

It also cleans all the data labels.

5.8.3.4 `enable_all()`

```
def user_interface.GUI.enable_all (
    self )
```

The `enable_all` method enables the data entries of the legs that are enabled through the enable leg checkbox.

5.8.3.5 new_data()

```
def user_interface.GUI.new_data (
    self )
```

The new_data method stores new data in a list if new data was updated.

5.8.3.6 start_button_event()

```
def user_interface.GUI.start_button_event (
    self )
```

The start_button_event method creates a serial connection if available.

If it is available it enables the update and start button. It also enables all the data entries.

5.8.3.7 stop_button_event()

```
def user_interface.GUI.stop_button_event (
    self )
```

The stop_button_event method destroys the serial connection if it was present.

It also disables all the data entries, cleans the data labels and disables the update and stop button.

5.8.3.8 update_all_fields()

```
def user_interface.GUI.update_all_fields (
    self )
```

The update_all_fields method updates all the data labels and entries.

It also prepares the USART message for being sent.

5.8.3.9 update_button_event()

```
def user_interface.GUI.update_button_event (
    self )
```

The update_button_event method updates all the data labels if there is new data updated.

It also sends a message to the microcontroller with the command corresponding to the data which is being updated.

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

- user_interface.py

5.9 leg_data_tab.LEG_DATA_TAB Class Reference

[LEG_DATA_TAB](#) creates the labels corresponding to the data which is displayed into the different tabs.

Public Member Functions

- `def __init__ (self, tabnr, legnr)`
The constructor takes tabnr and legnr as arguments.
- `def update (self, parameter, value)`
The update method updates datalabels if the update button is succesfully processed.
- `def stop (self)`
The stop method deletes the contents of data labels whenever the stop button is pressed.

Public Attributes

- `tabnr`
- `legnr`
- `frame`
- `frequency_label`
- `frequency_data_label`
- `pwm_frequency_label`
- `pwm_frequency_data_label`
- `amplitude_label`
- `amplitude_data_label`
- `phase_label`
- `phase_data_label`
- `new_value`
- `parameter`

5.9.1 Detailed Description

[LEG_DATA_TAB](#) creates the labels corresponding to the data which is displayed into the different tabs.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 __init__()

```
def leg_data_tab.LEG_DATA_TAB.__init__ (
    self,
    tabnr,
    legnr )
```

The constructor takes tabnr and legnr as arguments.

It creates the data labels in the tab specified and for the leg specified.

5.9.3 Member Function Documentation

5.9.3.1 stop()

```
def leg_data_tab.LEG_DATA_TAB.stop (
    self )
```

The stop method deletes the contents of data labels whenever the stop button is pressed.

5.9.3.2 update()

```
def leg_data_tab.LEG_DATA_TAB.update (
    self,
    parameter,
    value )
```

The update method updates datalabels if the update button is successfully processed.

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

- leg_data_tab.py

Index

- `__call__`
 - `command::command`, 12
 - `uart_connection::connection`, 14
 - `__del__`
 - `uart_connection::connection`, 13
 - `__init__`
 - `command::command`, 12
 - `enable_leg::ENABLE_LEG`, 15
 - `error_handler::ERROR_HANDLER`, 18
 - `field::ENTRY_FIELD`, 17
 - `field::FIELD`, 20
 - `leg_data_tab::LEG_DATA_TAB`, 25
 - `uart_connection::connection`, 14
 - `user_interface::GUI`, 22
- `amplitude_error`
 - `error_handler::ERROR_HANDLER`, 19
- `calc_value`
 - `command::command`, 12
- `check_connection`
 - `user_interface::GUI`, 23
- `command`, 7
- `command.command`, 11
- `command::command`
 - `__call__`, 12
 - `__init__`, 12
 - `calc_value`, 12
 - `command_type`, 12
- `command_type`
 - `command::command`, 12
- `destroy_window`
 - `user_interface::GUI`, 23
- `disable_all`
 - `user_interface::GUI`, 23
- `enable_all`
 - `user_interface::GUI`, 23
- `enable_leg`, 7
- `enable_leg.ENABLE_LEG`, 15
- `enable_leg::ENABLE_LEG`
 - `__init__`, 15
 - `get_data`, 16
 - `start`, 16
 - `stop`, 16
- `error_handler`, 8
- `error_handler.ERROR_HANDLER`, 18
- `error_handler::ERROR_HANDLER`
 - `__init__`, 18
 - `amplitude_error`, 19
 - `frequency_error`, 19
 - `phase_error`, 19
 - `pwm_frequency_error`, 19
 - `show_error_message`, 19
- `Field`, 8
- `field.DISPLAY_FIELD`, 14
- `field.ENTRY_FIELD`, 16
- `field.FIELD`, 20
- `field::ENTRY_FIELD`
 - `__init__`, 17
 - `get_data`, 17
 - `start`, 17
 - `stop`, 17
- `field::FIELD`
 - `__init__`, 20
- `frequency_error`
 - `error_handler::ERROR_HANDLER`, 19
- `get_data`
 - `enable_leg::ENABLE_LEG`, 16
 - `field::ENTRY_FIELD`, 17
- `leg_data_tab`, 8
- `leg_data_tab.LEG_DATA_TAB`, 25
- `leg_data_tab::LEG_DATA_TAB`
 - `__init__`, 25
 - `stop`, 26
 - `update`, 26
- `main`, 8
- `new_data`
 - `user_interface::GUI`, 23
- `phase_error`
 - `error_handler::ERROR_HANDLER`, 19
- `pwm_frequency_error`
 - `error_handler::ERROR_HANDLER`, 19
- `send`
 - `uart_connection::connection`, 14
- `show_error_message`
 - `error_handler::ERROR_HANDLER`, 19
- `start`
 - `enable_leg::ENABLE_LEG`, 16
 - `field::ENTRY_FIELD`, 17
- `start_button_event`
 - `user_interface::GUI`, 24
- `stop`

- enable_leg::ENABLE_LEG, [16](#)
 - field::ENTRY_FIELD, [17](#)
 - leg_data_tab::LEG_DATA_TAB, [26](#)
- stop_button_event
 - user_interface::GUI, [24](#)
- uart_communication, [9](#)
- uart_connection.connection, [13](#)
- uart_connection::connection
 - __call__, [14](#)
 - __del__, [13](#)
 - __init__, [14](#)
 - send, [14](#)
- update
 - leg_data_tab::LEG_DATA_TAB, [26](#)
- update_all_fields
 - user_interface::GUI, [24](#)
- update_button_event
 - user_interface::GUI, [24](#)
- user_interface, [9](#)
- user_interface.GUI, [21](#)
- user_interface::GUI
 - __init__, [22](#)
 - check_connection, [23](#)
 - destroy_window, [23](#)
 - disable_all, [23](#)
 - enable_all, [23](#)
 - new_data, [23](#)
 - start_button_event, [24](#)
 - stop_button_event, [24](#)
 - update_all_fields, [24](#)
 - update_button_event, [24](#)