

Helicopter Rig Project

Generated by Doxygen 1.8.13

Contents

1	Module Index	2
1.1	Modules	2
2	Data Structure Index	2
2.1	Data Structures	2
3	Module Documentation	2
3.1	ButtonsApi	2
3.1.1	Detailed Description	3
3.1.2	Enumeration Type Documentation	3
3.1.3	Function Documentation	3
3.2	FlightController	5
3.2.1	Detailed Description	5
3.2.2	Function Documentation	5
3.3	HeightApi	6
3.3.1	Detailed Description	6
3.3.2	Function Documentation	6
3.4	HeightController	7
3.4.1	Detailed Description	7
3.4.2	Function Documentation	7
3.5	PidController	9
3.5.1	Detailed Description	9
3.5.2	Function Documentation	9
3.6	PwmOutput	11
3.6.1	Detailed Description	11
3.6.2	Enumeration Type Documentation	11
3.6.3	Function Documentation	11
3.7	SwitchApi	13
3.7.1	Detailed Description	13
3.7.2	Enumeration Type Documentation	13
3.7.3	Function Documentation	13
3.8	YawApi	14
3.8.1	Detailed Description	14
3.8.2	Function Documentation	14
3.9	YawController	16
3.9.1	Detailed Description	16
3.9.2	Function Documentation	16

4 Data Structure Documentation	19
4.1 PidState Struct Reference	19
4.1.1 Detailed Description	19
Index	21

1 Module Index

1.1 Modules

Here is a list of all modules:

ButtonsApi	2
FlightController	5
HeightApi	6
PidController	9
HeightController	7
YawController	16
PwmOutput	11
SwitchApi	13
YawApi	14

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

PidState	
A structure to accumulate the error and store the previous error for use by the pid controller	19

3 Module Documentation

3.1 ButtonsApi

Enumerations

- enum `Button` {
 `BTN_UP`, `BTN_DOWN`, `BTN_LEFT`, `BTN_RIGHT`,
 `NUM_BUTTONS` }
 The buttons.

Functions

- void [ButtonsInit](#) (void)
Initialise the buttons.
- void [UpdateButtons](#) ()
Update all of the buttons and their state.
- uint8_t [NumPushes](#) (uint8_t button_name)
Gets the number of pushes for a given button and resets the push count.
- void [ResetPushes](#) (void)
Reset the push count for all buttons.

3.1.1 Detailed Description

3.1.2 Enumeration Type Documentation

3.1.2.1 Button

enum [Button](#)

The buttons.

Enumerator

BTN_UP	The UP button.
BTN_DOWN	The DOWN button.
BTN_LEFT	The LEFT button.
BTN_RIGHT	The RIGHT button.
NUM_BUTTONS	The total number of buttons.

3.1.3 Function Documentation

3.1.3.1 NumPushes()

```
uint8_t NumPushes (  
    uint8_t button_name )
```

Gets the number of pushes for a given button and resets the push count.

Parameters

<i>button_name</i>	One of BTN_UP , BTN_DOWN , BTN_LEFT , or BTN_RIGHT .
--------------------	----------------------------------------------------------------------------------------------------------------------

Returns

The number of pushes for the given button since this function was last called.

3.2 FlightController

Functions

- void [FlightControllerInit](#) (void)
Initialise the flight controller module.
- void [UpdateFlightMode](#) ()
U.
- const char * [GetFlightMode](#) (void)
Get a string representation of the current flight mode.
- void [PriorityTaskInit](#) (void)
Initialise the priority task sequencer.
- void [PriorityTaskEnable](#) (void)
Enable the priority task sequencer.
- void [PriorityTaskDisable](#) (void)
Disable the priority task sequencer.

3.2.1 Detailed Description

3.2.2 Function Documentation

3.2.2.1 [GetFlightMode\(\)](#)

```
const char* GetFlightMode (  
    void )
```

Get a string representation of the current flight mode.

Returns

a string representing the flight mode

3.2.2.2 [PriorityTaskInit\(\)](#)

```
void PriorityTaskInit (  
    void )
```

Initialise the priority task sequencer.

This timer handles updating of the pid controllers and height.

3.3 HeightApi

Functions

- `int32_t GetHeight (void)`
Get the current height.
- `int32_t GetHeightPercentage (void)`
Get the current height as a percentage.
- `void UpdateHeight ()`
Trigger the current height reading to be updated.
- `void HeightManagerInit (void)`
Initialise the height sensor peripherals and ports.
- `void ZeroHeightTrigger (void)`
Trigger a zero height reading to be used as a reference for subsequent height readings.

3.3.1 Detailed Description

3.3.2 Function Documentation

3.3.2.1 GetHeight()

```
int32_t GetHeight (  
    void )
```

Get the current height.

Retrieve the sensor reading after it has been offset the zeroed sensor reading.

Returns

The height.

3.3.2.2 GetHeightPercentage()

```
int32_t GetHeightPercentage (  
    void )
```

Get the current height as a percentage.

Returns

The height as a percentage.

3.3.2.3 UpdateHeight()

```
void UpdateHeight ( )
```

Trigger the current height reading to be updated.

No longer used in favour of direct triggering via timer timeout flag.

3.4 HeightController

Functions

- void [SetTargetHeight](#) (uint32_t height)
Set the target height (%).
- uint32_t [GetTargetHeight](#) (void)
Get the target height (%).
- void [HeightControllerInit](#) (void)
Initialise the height controller.
- void [PreloadHeightController](#) (int32_t control, int32_t error)
Preload the integral component of the pid controller so the Main rotor starts of with `control` power.
- void [UpdateHeightController](#) (uint32_t delta_t)
Update the height controller pid loop.
- void [TuneProportionalMainRotor](#) (double gain)
Use at own risk.

3.4.1 Detailed Description

3.4.2 Function Documentation

3.4.2.1 [GetTargetHeight\(\)](#)

```
uint32_t GetTargetHeight (  
    void )
```

Get the target height (%).

Returns

The target height(%).

3.4.2.2 [PreloadHeightController\(\)](#)

```
void PreloadHeightController (  
    int32_t control,  
    int32_t error )
```

Preload the integral component of the pid controller so the Main rotor starts of with `control` power.

This was to improves rise time of the helicopter by boosting the Main rotor.

Parameters

<i>control</i>	The immediate control power desired by the Main rotor
<i>error</i>	The absolute difference between current height and target height.

3.4.2.3 SetTargetHeight()

```
void SetTargetHeight (
    uint32_t height )
```

Set the target height (%).

Parameters

<i>height</i>	The target height (%).
---------------	------------------------

3.4.2.4 TuneProportionalMainRotor()

```
void TuneProportionalMainRotor (
    double gain )
```

Use at own risk.

Parameters

<i>gain</i>	Proportial gain.
-------------	------------------

3.4.2.5 UpdateHeightController()

```
void UpdateHeightController (
    uint32_t delta_t )
```

Update the height controller pid loop.

Parameters

<i>delta_t</i>	The update period of the height controller.
----------------	---------------------------------------------

3.5 PidController

Modules

- [HeightController](#)
- [YawController](#)

Data Structures

- struct [PidState](#)

A structure to accumulate the error and store the previous error for use by the pid controller.

Functions

- void [PidInit](#) ([PidState](#) *state)
Initialise the pid controller.
- void [PreloadPid](#) ([PidState](#) *state, int32_t integral_preload)
Preload the integral component of the pid state with a positive or negative error to reduce integration time.
- int32_t [UpdatePid](#) ([PidState](#) *state, int32_t error, uint32_t delta_t, double proportional_gain, double integral_gain, double derivative_gain)
Update the pid controller loop.

3.5.1 Detailed Description

3.5.2 Function Documentation

3.5.2.1 PidInit()

```
void PidInit (  
    PidState * state )
```

Initialise the pid controller.

Parameters

<i>state</i>	The pid error state.
--------------	----------------------

See also

[PidState](#)

3.5.2.2 PreloadPid()

```
void PreloadPid (  
    PidState * state,  
    int32_t integral_preload )
```

Preload the integral component of the pid state with a positive or negative error to reduce integration time.

Parameters

<i>state</i>	The pid error state.
<i>integral_preload</i>	The preload error.

3.5.2.3 UpdatePid()

```
int32_t UpdatePid (  
    PidState * state,  
    int32_t error,  
    uint32_t delta_t,  
    double proportional_gain,  
    double integral_gain,  
    double derivative_gain )
```

Update the pid controller loop.

Parameters

<i>state</i>	The pid error state.
<i>error</i>	The current error.
<i>delta_t</i>	The update period of the pid controller.
<i>proportional_gain</i>	The proportional gain constant.
<i>integral_gain</i>	The integral gain constant.
<i>derivative_gain</i>	The derivative gain constant.

Returns

3.6 PwmOutput

Enumerations

- enum [PwmOutput](#) { [MAIN_ROTOR](#), [TAIL_ROTOR](#) }
An enumeration for determining which PWM output to configure.

Functions

- void [PwmInit](#) ()
Initialise both of the pwm outputs.
- void [SetPwmDutyCycle](#) (uint8_t pwm_output, uint32_t duty_cycle)
Set the duty cycle of the PWM output in the range 2-98%.
- uint32_t [GetPwmDutyCycle](#) (uint8_t pwm_output)
Get the current PWM duty cycle for the given PWM output.
- void [PwmDisable](#) (uint8_t pwm_output)
Disable the given PWM output.
- void [PwmEnable](#) (uint8_t pwm_output)
Enable the given PWM output.

3.6.1 Detailed Description

3.6.2 Enumeration Type Documentation

3.6.2.1 PwmOutput

enum [PwmOutput](#)

An enumeration for determining which PWM output to configure.

Enumerator

MAIN_ROTOR	The Main rotor PWM output.
TAIL_ROTOR	The Tail rotor PWM output.

3.6.3 Function Documentation

3.6.3.1 GetPwmDutyCycle()

```
uint32_t GetPwmDutyCycle (  
    uint8_t pwm_output )
```

Get the current PWM duty cycle for the given PWM output.

Parameters

<i>pwm_output</i>	The PWM output.
-------------------	-----------------

Returns

The current PWM duty cycle as percentage.

3.6.3.2 PwmDisable()

```
void PwmDisable (
    uint8_t pwm_output )
```

Disable the given PWM output.

Parameters

<i>pwm_output</i>	The PWM output to configure.
-------------------	------------------------------

3.6.3.3 PwmEnable()

```
void PwmEnable (
    uint8_t pwm_output )
```

Enable the given PWM output.

Parameters

<i>pwm_output</i>	The PWM output to configure.
-------------------	------------------------------

3.6.3.4 SetPwmDutyCycle()

```
void SetPwmDutyCycle (
    uint8_t pwm_output,
    uint32_t duty_cycle )
```

Set the duty cycle of the PWM output in the range 2-98%.

Parameters

<i>pwm_output</i>	The PWM output to configure.
<i>duty_cycle</i>	The desired duty cycle, in the range 2-98%.

3.7 SwitchApi

Enumerations

- enum [SwitchState](#) { [SWITCH_DOWN](#), [SWITCH_UP](#) }
An enumeration of the states of the slider switch.

Functions

- void [SwitchInit](#) (void)
Initialise the switch.
- void [UpdateSwitch](#) ()
Update the switch state.
- uint8_t [GetSwitchEvent](#) (void)
Get the latest switch event.

3.7.1 Detailed Description

3.7.2 Enumeration Type Documentation

3.7.2.1 SwitchState

enum [SwitchState](#)

An enumeration of the states of the slider switch.

Enumerator

SWITCH_DOWN	The switch is in the downwards position.
SWITCH_UP	The switch is in the upwards position.

3.7.3 Function Documentation

3.7.3.1 GetSwitchEvent()

```
uint8_t GetSwitchEvent (  
    void )
```

Get the latest switch event.

Returns

DOWN or UP slide of the switch.

3.8 YawApi

Functions

- void [YawDetectionInit](#) (void)
Initialises the yaw manager.
- int32_t [GetYaw](#) (void)
Get the current yaw.
- int32_t [GetYawDegrees](#) (void)
Get the current yaw.
- int32_t [GetClosestYawRef](#) (int32_t current_yaw)
Helper function to return the closest yaw such that the helicopter is facing towards the camera.
- void [YawRefTrigger](#) (void)
Triggers an interrupt to fire when the reference yaw has been found.
- bool [YawRefFound](#) (void)
Check if the reference yaw has been found.

3.8.1 Detailed Description

3.8.2 Function Documentation

3.8.2.1 GetClosestYawRef()

```
int32_t GetClosestYawRef (
    int32_t current_yaw )
```

Helper function to return the closest yaw such that the helicopter is facing towards the camera.

Parameters

<i>current_yaw</i>	the current yaw
--------------------	-----------------

Returns

the closest reference yaw

3.8.2.2 GetYaw()

```
int32_t GetYaw (
    void )
```

Get the current yaw.

Returns

the yaw (notches)

3.8.2.3 GetYawDegrees()

```
int32_t GetYawDegrees (
    void )
```

Get the current yaw.

Returns

the yaw (degrees)

3.8.2.4 YawRefFound()

```
bool YawRefFound (
    void )
```

Check if the reference yaw has been found.

Returns

true if the yaw reference has been found else false

3.9 YawController

Functions

- `int32_t GetTargetYawDegrees` (void)
Get the target yaw in degrees.
- `void SetTargetYawDegrees` (int32_t yaw)
Set the desired target yaw (degrees).
- `int32_t GetTargetYaw` (void)
Get the target yaw.
- `void SetTargetYaw` (int32_t yaw)
Set the desired target yaw.
- `void YawControllerInit` (void)
Initialise the yaw controller.
- `void PreloadYawController` (int32_t control, int32_t error)
Preload the integral component of the pid controller so the Tail rotor starts of with `control` power.
- `void UpdateYawController` (uint32_t delta_t)
Update the yaw controller pid loop.
- `void TuneProportionalTailRotor` (double gain)
Use at own risk.

3.9.1 Detailed Description

3.9.2 Function Documentation

3.9.2.1 GetTargetYaw()

```
int32_t GetTargetYaw (  
    void )
```

Get the target yaw.

Returns

the target yaw.

See also

[YawApi](#) for rotation unit.

3.9.2.2 GetTargetYawDegrees()

```
int32_t GetTargetYawDegrees (
    void )
```

Get the target yaw in degrees.

Returns

The target yaw in degrees.

3.9.2.3 PreloadYawController()

```
void PreloadYawController (
    int32_t control,
    int32_t error )
```

Preload the integral component of the pid controller so the Tail rotor starts of with `control` power.

Can be used in conjunction with [PreloadHeightController](#) to improve the rise time of the helicopter.

Parameters

<i>control</i>	The immediate control power desired by the Tail rotor
<i>error</i>	The absolute difference between current yaw and target yaw.

3.9.2.4 SetTargetYaw()

```
void SetTargetYaw (
    int32_t yaw )
```

Set the desired target yaw.

Parameters

<i>yaw</i>	The desired target yaw.
------------	-------------------------

3.9.2.5 SetTargetYawDegrees()

```
void SetTargetYawDegrees (
    int32_t yaw )
```

Set the desired target yaw (degrees).

Parameters

<i>yaw</i>	The desired target yaw (degrees).
------------	-----------------------------------

3.9.2.6 TuneProportionalTailRotor()

```
void TuneProportionalTailRotor (
    double gain )
```

Use at own risk.

Parameters

<i>gain</i>	Proportial gain.
-------------	------------------

3.9.2.7 UpdateYawController()

```
void UpdateYawController (
    uint32_t delta_t )
```

Update the yaw controller pid loop.

Parameters

<i>delta_t</i>	The update period of the yaw controller.
----------------	------------------------------------------

4 Data Structure Documentation

4.1 PidState Struct Reference

A structure to accumulate the error and store the previous error for use by the pid controller.

```
#include <pid.h>
```

Data Fields

- `int32_t error_previous`
The previous error.
- `int32_t error_integrated`
The accumulated error.

4.1.1 Detailed Description

A structure to accumulate the error and store the previous error for use by the pid controller.

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

- `src/pid.h`

Index

- Button
 - ButtonsApi, [3](#)
- ButtonsApi, [2](#)
 - Button, [3](#)
 - NumPushes, [3](#)
- FlightController, [5](#)
 - GetFlightMode, [5](#)
 - PriorityTaskInit, [5](#)
- GetClosestYawRef
 - YawApi, [14](#)
- GetFlightMode
 - FlightController, [5](#)
- GetHeight
 - HeightApi, [6](#)
- GetHeightPercentage
 - HeightApi, [6](#)
- GetPwmDutyCycle
 - PwmOutput, [11](#)
- GetSwitchEvent
 - SwitchApi, [13](#)
- GetTargetHeight
 - HeightController, [7](#)
- GetTargetYaw
 - YawController, [16](#)
- GetTargetYawDegrees
 - YawController, [16](#)
- GetYaw
 - YawApi, [14](#)
- GetYawDegrees
 - YawApi, [14](#)
- HeightApi, [6](#)
 - GetHeight, [6](#)
 - GetHeightPercentage, [6](#)
 - UpdateHeight, [6](#)
- HeightController, [7](#)
 - GetTargetHeight, [7](#)
 - PreloadHeightController, [7](#)
 - SetTargetHeight, [8](#)
 - TuneProportionalMainRotor, [8](#)
 - UpdateHeightController, [8](#)
- NumPushes
 - ButtonsApi, [3](#)
- PidController, [9](#)
 - PidInit, [9](#)
 - PreloadPid, [9](#)
 - UpdatePid, [10](#)
- PidInit
 - PidController, [9](#)
- PidState, [19](#)
- PreloadHeightController
 - HeightController, [7](#)
- PreloadPid
 - PidController, [9](#)
- PreloadYawController
 - YawController, [17](#)
- PriorityTaskInit
 - FlightController, [5](#)
- PwmDisable
 - PwmOutput, [12](#)
- PwmEnable
 - PwmOutput, [12](#)
- PwmOutput, [11](#)
 - GetPwmDutyCycle, [11](#)
 - PwmDisable, [12](#)
 - PwmEnable, [12](#)
 - PwmOutput, [11](#)
 - SetPwmDutyCycle, [12](#)
- SetPwmDutyCycle
 - PwmOutput, [12](#)
- SetTargetHeight
 - HeightController, [8](#)
- SetTargetYaw
 - YawController, [17](#)
- SetTargetYawDegrees
 - YawController, [17](#)
- SwitchApi, [13](#)
 - GetSwitchEvent, [13](#)
 - SwitchState, [13](#)
- SwitchState
 - SwitchApi, [13](#)
- TuneProportionalMainRotor
 - HeightController, [8](#)
- TuneProportionalTailRotor
 - YawController, [18](#)
- UpdateHeight
 - HeightApi, [6](#)
- UpdateHeightController
 - HeightController, [8](#)
- UpdatePid
 - PidController, [10](#)
- UpdateYawController
 - YawController, [18](#)
- YawApi, [14](#)
 - GetClosestYawRef, [14](#)
 - GetYaw, [14](#)
 - GetYawDegrees, [14](#)
 - YawRefFound, [15](#)
- YawController, [16](#)
 - GetTargetYaw, [16](#)
 - GetTargetYawDegrees, [16](#)
 - PreloadYawController, [17](#)
 - SetTargetYaw, [17](#)
 - SetTargetYawDegrees, [17](#)

TuneProportionalTailRotor, [18](#)
UpdateYawController, [18](#)
YawRefFound
YawApi, [15](#)