Helicopter Rig Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 ButtonsApi

Macros

- #define NUM_POLLS 5

  Change button state only after NUM_POLLS consecutive readings have an opposite value.

Enumerations

- enum {
  BTN_UP, **BTN_DOWN**, **BTN_LEFT**, **BTN_RIGHT**,
  NUM_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.

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 NumPushes()

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

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

Parameters

button_name	one of BUT_UP, BUT_DOWN, BUT_LEFT or BUT_RIGHT

Returns

the number of pushes for the given button since last called

4.2 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.

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 GetFlightMode()

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

Get a string representation of the current flight mode.

Returns

a string representing the flight mode

4.2.2.2 PriorityTaskInit()

```
void PriorityTaskInit (
             void  )
```

Initialise the priority task sequencer.

This timer handles updating of the pid controllers and height.

4.3 HeightApi

Macros

- #define FULL_SCALE_RANGE 993

  The range of the height sensor for a 100% height reading.

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.

4.3.1 Detailed Description

4.3.2 Macro Definition Documentation

4.3.2.1 FULL_SCALE_RANGE

```
#define FULL_SCALE_RANGE 993
```

The range of the height sensor for a 100% height reading.

This translates to a 0.8 V sensor range using the 12-bit ADC peripheral.

4.3.3 Function Documentation

4.3.3.1 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.

4.3.3.2 GetHeightPercentage()

```
int32_t GetHeightPercentage (
             void  )
```

Get the current height as a percentage.

Returns

The height as a percentage.

4.3.3.3 UpdateHeight()

```
void UpdateHeight ( )
```

Trigger the current height reading to be updated.

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

4.4 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 contoller so the Main rotor start of with.
- void UpdateHeightController (uint32_t delta_t)

  Update the height controller pid loop.
- void TuneProportionalMainRotor (double gain)

  Use at own risk.

4.4.1 Detailed Description

4.4.2 Function Documentation

4.4.2.1 GetTargetHeight()

```
uint32_t GetTargetHeight (
            void  )
```

Get the target height (%).

Returns

The target height(%).

4.4.2.2 PreloadHeightController()

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

Preload the integral component of the pid contoller so the Main rotor start of with.

Parameters

control	power.

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

Parameters

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

4.4.2.3 SetTargetHeight()

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

Set the target height (%).

Parameters

height	The target height (%).

4.4.2.4 TuneProportionalMainRotor()

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

Use at own risk.

Parameters

gain	

4.4.2.5 UpdateHeightController()

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

Update the height controller pid loop.

Parameters

delta↩ _t	The update period of the height controller.

4.5 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 postitve 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.

4.5.1 Detailed Description

4.5.2 Function Documentation

4.5.2.1 PidInit()

```
void PidInit (
            PidState ∗ state )
```

Initialise the pid controller.

Parameters

state	The pid error state.

See also

PidState

4.5.2.2 PreloadPid()

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

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

Parameters

state	The pid error state.
integral_preload	The preload error.

4.5.2.3 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

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

Returns

4.6 SwitchApi

Macros

- #define NUM_POLLS 5

  Change switch state only after NUM_POLLS consecutive readings have an opposite value.

Enumerations

- enum { **SWITCH_DOWN**, **SWITCH_UP** }

Functions

- void SwitchInit (void)

  Initialise the switch.
- void UpdateSwitch ()

  Update the switch state.
- uint8_t GetSwitchEvent (void)

  Get the switch event.

4.6.1 Detailed Description

4.6.2 Function Documentation

4.6.2.1 GetSwitchEvent()

```
uint8_t GetSwitchEvent (
            void  )
```

Get the switch event.

Returns

DOWN or UP slide of the switch.

4.7 YawApi

Macros

- #define NUMBER_SLOTS 112

  The number of slots in 360 degrees of rotation.
- #define YAW_FULL_ROTATION (NUMBER_SLOTS ∗ 4)

  The number of yaw updates in 360 degrees of rotation.

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 refernce yaw has been found.
- bool YawRefFound (void)

  Check if the reference yaw has been found.

4.7.1 Detailed Description

4.7.2 Function Documentation

4.7.2.1 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

current_yaw	the current yaw

Returns

the closest reference yaw

4.7.2.2 GetYaw()

```
int32_t GetYaw (
            void  )
```

Get the current yaw.

Returns

the yaw (notches)

4.7.2.3 GetYawDegrees()

```
int32_t GetYawDegrees (
            void  )
```

Get the current yaw.

Returns

the yaw (degrees)

4.7.2.4 YawRefFound()

```
bool YawRefFound (
            void  )
```

Check if the reference yaw has been found.

Returns

true if the yaw reference has been found else false

4.8 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)
- void YawControllerInit (void)

    Initialise the yaw controller.
- void PreloadYawController (int32_t control, int32_t error)
- void UpdateYawController (uint32_t delta_t)
- void TuneProportionalTailRotor (double gain)

    Use at own risk.

4.8.1 Detailed Description

4.8.2 Function Documentation

4.8.2.1 GetTargetYaw()

```
int32_t GetTargetYaw (
            void )
```

Get the target yaw.

Returns

the target yaw.

See also

YawApi for rotation unit.

4.8.2.2 GetTargetYawDegrees()

```
int32_t GetTargetYawDegrees (
            void )
```

Get the target yaw in degrees.

Returns

The target yaw in degrees.

4.8.2.3 PreloadYawController()

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

Parameters

control	
error	

4.8.2.4 SetTargetYaw()

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

Parameters

yaw	

4.8.2.5 SetTargetYawDegrees()

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

Set the desired target yaw (degrees).

Parameters

yaw	The desired target yaw (degrees).

4.8.2.6 TuneProportionalTailRotor()

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

Use at own risk.

Parameters

gain	

4.8.2.7 UpdateYawController()

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

Parameters

delta←↩ _t	

Chapter 5

Data Structure Documentation

5.1 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.

5.1.1 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

Chapter 6

File Documentation

6.1 src/buttons.c File Reference

A module to operate the buttons.

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_gpio.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "buttons.h"
```

Macros

- #define BTN_UP_PERIPH SYSCTL_PERIPH_GPIOE

  Up button definitions.
- #define **BTN_UP_BASE** GPIO_PORTE_BASE
- #define **BTN_UP_PIN** GPIO_PIN_0
- #define **BTN_UP_DEFAULT** 0

- #define BTN_DOWN_PERIPH SYSCTL_PERIPH_GPIOD

  Down button definitions.
- #define **BTN_DOWN_BASE** GPIO_PORTD_BASE
- #define **BTN_DOWN_PIN** GPIO_PIN_2
- #define **BTN_DOWN_DEFAULT** 0

- #define BTN_LEFT_PERIPH SYSCTL_PERIPH_GPIOF

  Left button definitions.
- #define **BTN_LEFT_BASE** GPIO_PORTF_BASE
- #define **BTN_LEFT_PIN** GPIO_PIN_4
- #define **BTN_LEFT_DEFAULT** 1

- #define BTN_RIGHT_PERIPH SYSCTL_PERIPH_GPIOF

  Right button definitions.
- #define **BTN_RIGHT_BASE** GPIO_PORTF_BASE
- #define **BTN_RIGHT_PIN** GPIO_PIN_0
- #define **BTN_RIGHT_DEFAULT** 1

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.

6.1.1 Detailed Description

A module to operate the buttons.

6.2 src/buttons.h File Reference

A module to operate the buttons.

Macros

- #define NUM_POLLS 5

  Change button state only after NUM_POLLS consecutive readings have an opposite value.

Enumerations

- enum {
  BTN_UP, **BTN_DOWN**, **BTN_LEFT**, **BTN_RIGHT**,
  NUM_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.

6.2.1 Detailed Description

A module to operate the buttons.

6.3 src/flight_controller.c File Reference

Handles moving between flight modes.

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "driverlib/timer.h"
#include "utils/scheduler.h"
#include "buttons.h"
#include "flight_controller.h"
#include "height.h"
#include "height_controller.h"
#include "pwm.h"
#include "switch.h"
#include "yaw.h"
#include "yaw_controller.h"
```

Macros

- #define RATE_OF_DESCENT 35

  Rate of descent (ms per decrement of duty cycle)
- #define YAW_SAMPLE_TOLERANCE 2

  Acceptable tolerance for yaw error (rotation unit defined in yaw.h)
- #define HEIGHT_SAMPLE_TOLERANCE 1

  Acceptable tolerance for height error (%)
- #define NUM_ERROR_SAMPLES 5

  Number of samples to summate error over.

- #define TIMER_PERIPH SYSCTL_PERIPH_TIMER0

  Timer definitions.
- #define **TIMER_BASE** TIMER0_BASE
- #define **TIMER_CONFIG** TIMER_CFG_PERIODIC
- #define **TIMER_TIMER** TIMER_A
- #define **TIMER_TIMEOUT** TIMER_TIMA_TIMEOUT
- #define **TIMER_INT** INT_TIMER0A

Enumerations

- enum { **LANDED**, **INIT**, **FLYING**, **LANDING** }

Functions

- void PriorityTaskInit (void)

  Initialise the priority task sequencer.
- void PriorityTaskDisable (void)

  Disable the priority task sequencer.
- void PriorityTaskEnable (void)

  Enable the priority task sequencer.
- 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 TimerInit (void)

  Forward declarations.
- void **TimerHandler** (void)
- void **UpdateError** (void)
- void **ResetError** (void)
- bool **HasReachedTargetYaw** (void)
- bool **HasReachedTargetHeight** (void)

6.3.1 Detailed Description

Handles moving between flight modes.

For example, the heli must initiate a landing sequence if it was previously flying if the mode switch was moved to the UP position. Also, this module a convenient method to get the current flight mode.

6.4 src/flight_controller.h File Reference

Handles moving between flight modes.

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.

6.4.1 Detailed Description

Handles moving between flight modes.

For example, the heli must initiate a landing sequence if it was previously flying if the mode switch was moved to the UP position. Also, this module a convenient method to get the current flight mode.

6.5 src/height.c File Reference

Module to acquire current height via the height sensor and trigger a zero-height reading.

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_memmap.h"
#include "driverlib/adc.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "height.h"
```

Macros

- #define ADC_GPIO_BASE GPIO_PORTE_BASE

  ADC height sensor definitions.
- #define **ADC_GPIO_PIN** GPIO_PIN_4
- #define **ADC_BASE** ADC0_BASE
- #define **ADC_SEQUENCE** 3
- #define **ADC_CHANNEL** ADC_CTL_CH9
- #define **ADC_PERIPH_ADC** SYSCTL_PERIPH_ADC0
- #define **ADC_PERIPH_GPIO** SYSCTL_PERIPH_GPIOE

Functions

- void AdcHandler (void)

  The ADC interrupt handler for the height sensor.
- void HeightManagerInit ()

  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.
- int32_t GetHeight ()

  Get the current height.
- int32_t GetHeightPercentage ()

  Get the current height as a percentage.
- void UpdateHeight (void)

  Trigger the current height reading to be updated.

6.5.1 Detailed Description

Module to acquire current height via the height sensor and trigger a zero-height reading.

Provides helper methods to get height.

6.6 src/height.h File Reference

Module to acquire current height via the height sensor and trigger a zero-height reading.

Macros

- #define FULL_SCALE_RANGE 993

  The range of the height sensor for a 100% height reading.

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.

6.6.1 Detailed Description

Module to acquire current height via the height sensor and trigger a zero-height reading.

Provides helper methods to get height.

6.7 src/height_controller.h File Reference

Pid controller for the Main rotor.

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 contoller so the Main rotor start of with.
- void UpdateHeightController (uint32_t delta_t)

  Update the height controller pid loop.
- void TuneProportionalMainRotor (double gain)

  Use at own risk.

6.7.1 Detailed Description

Pid controller for the Main rotor.

6.8 src/oled_interface.c File Reference

A simple interface to the Orbit OLED library.

```
#include "../lib/libOrbitOled/OrbitOled.h"
#include "../lib/libOrbitOled/delay.h"
#include "../lib/libOrbitOled/FillPat.h"
#include "../lib/libOrbitOled/LaunchPad.h"
#include "../lib/libOrbitOled/OrbitBoosterPackDefs.h"
#include "../lib/libOrbitOled/OrbitOledChar.h"
#include "../lib/libOrbitOled/OrbitOledGrph.h"
#include "oled_interface.h"
```

Functions

- void OledStringDraw (char ∗string_ptr, uint32_t x_char, uint32_t y_char)

  Draw a string of character on the OLED display at the desired row and column.
- void OledInit (void)

  Initialise the OLED display.
- void OledClearBuffer (void)

  Clear the OLED display buffer.

6.8.1 Detailed Description

A simple interface to the Orbit OLED library.

6.8.2 Function Documentation

6.8.2.1 OledStringDraw()

```
void OledStringDraw (
            char * string_ptr,
            uint32_t x_char,
            uint32_t y_char )
```

Draw a string of character on the OLED display at the desired row and column.

Parameters

string_ptr	The string to display.
x_char	The display row.
y_char	The display column.

6.9 src/oled_interface.h File Reference

A simple interface to the Orbit OLED library.

Functions

- void OledClearBuffer (void)

  Clear the OLED display buffer.
- void OledInit (void)

  Initialise the OLED display.
- void OledStringDraw (char ∗string_ptr, uint32_t x_char, uint32_t y_char)

  Draw a string of character on the OLED display at the desired row and column.

6.9.1 Detailed Description

A simple interface to the Orbit OLED library.

6.9.2 Function Documentation

6.9.2.1 OledStringDraw()

```
void OledStringDraw (
            char * string_ptr,
            uint32_t x_char,
            uint32_t y_char )
```

Draw a string of character on the OLED display at the desired row and column.

Parameters

string_ptr	The string to display.
x_char	The display row.
y_char	The display column.

6.10 src/pid.c File Reference

Generic pid controller module.

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

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 postitve 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.

6.10.1 Detailed Description

Generic pid controller module.

6.11 src/pid.h File Reference

Generic pid controller module.

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 postitve 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.

6.11.1 Detailed Description

Generic pid controller module.

6.12 src/pwm.c File Reference

PWM module to handle the power output for the Main and Tail rotors.

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/pwm.h"
#include "driverlib/sysctl.h"
#include "pwm.h"
```

Macros

- #define PWM_MAIN_BASE PWM0_BASE

  PWM Main rotor definitions.
- #define **PWM_MAIN_GEN** PWM_GEN_3
- #define **PWM_MAIN_OUTNUM** PWM_OUT_7
- #define **PWM_MAIN_OUTBIT** PWM_OUT_7_BIT
- #define **PWM_MAIN_PERIPH_PWM** SYSCTL_PERIPH_PWM0
- #define **PWM_MAIN_PERIPH_GPIO** SYSCTL_PERIPH_GPIOC
- #define **PWM_MAIN_GPIO_BASE** GPIO_PORTC_BASE
- #define **PWM_MAIN_GPIO_CONFIG** GPIO_PC5_M0PWM7
- #define **PWM_MAIN_GPIO_PIN** GPIO_PIN_5

- #define PWM_TAIL_BASE PWM1_BASE

  PWM Tail rotor definitions.

- #define **PWM_TAIL_GEN** PWM_GEN_2
- #define **PWM_TAIL_OUTNUM** PWM_OUT_5
- #define **PWM_TAIL_OUTBIT** PWM_OUT_5_BIT
- #define **PWM_TAIL_PERIPH_PWM** SYSCTL_PERIPH_PWM1
- #define **PWM_TAIL_PERIPH_GPIO** SYSCTL_PERIPH_GPIOF
- #define **PWM_TAIL_GPIO_BASE** GPIO_PORTF_BASE
- #define **PWM_TAIL_GPIO_CONFIG** GPIO_PF1_M1PWM5
- #define **PWM_TAIL_GPIO_PIN** GPIO_PIN_1

- #define PWM_DIVIDER_CODE SYSCTL_PWMDIV_16

  General PWM definitions.
- #define **PWM_DIVIDER** 16

Functions

- void PwmInit ()

  TODO.
- void SetPwmDutyCycle (uint8_t pwm_output, uint32_t duty_cycle)
- uint32_t GetPwmDutyCycle (uint8_t pwm_output)

  TODO.
- void SetPwmState (uint8_t pwm_output, bool state)

  Set the output state for the given PWM output.
- void PwmEnable (uint8_t pwm_output)

  TODO.
- void PwmDisable (uint8_t pwm_output)

  TODO.

6.12.1 Detailed Description

PWM module to handle the power output for the Main and Tail rotors.

6.12.2 Function Documentation

6.12.2.1 GetPwmDutyCycle()

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

TODO.

Parameters

pwm_output	

Returns

6.12.2.2 PwmDisable()

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

TODO.

Parameters

pwm_output	

6.12.2.3 PwmEnable()

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

TODO.

Parameters

pwm_output	

6.12.2.4 SetPwmDutyCycle()

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

Parameters

pwm_output	
duty_cycle	

6.12.2.5 SetPwmState()

```
void SetPwmState (
            uint8_t pwm_output,
            bool state )
```

Set the output state for the given PWM output.

Parameters

pwm_output	The PWM output, either Tail or Main.
state	The output state, either true (on) or false (off);

6.13 src/pwm.h File Reference

PWM module to handle the power output for the Main and Tail rotors.

Macros

- #define PWM_FREQUENCY 200

    The frequency of the PWM output signal (Hz).

Enumerations

- enum { **MAIN_ROTOR**, **TAIL_ROTOR** }

Functions

- void PwmInit ()

    TODO.
- void SetPwmDutyCycle (uint8_t pwm_output, uint32_t duty_cycle)
- uint32_t GetPwmDutyCycle (uint8_t pwm_output)

    TODO.
- void PwmDisable (uint8_t pwm_output)

    TODO.
- void PwmEnable (uint8_t pwm_output)

    TODO.

6.13.1 Detailed Description

PWM module to handle the power output for the Main and Tail rotors.

6.13.2 Function Documentation

6.13.2.1 GetPwmDutyCycle()

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

TODO.

Parameters

pwm_output	

Returns

6.13.2.2 PwmDisable()

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

TODO.

Parameters

pwm_output	

6.13.2.3 PwmEnable()

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

TODO.

Parameters

pwm_output	

6.13.2.4 SetPwmDutyCycle()

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

Parameters

pwm_output	
duty_cycle	

6.14 src/reset.c File Reference

Soft reset module.

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "reset.h"
```

Macros

- #define RESET_PERIPH_GPIO SYSCTL_PERIPH_GPIOA

  Reset GPIO definitions.
- #define **RESET_PERIPH_BASE** GPIO_PORTA_BASE
- #define **RESET_PIN** GPIO_PIN_6
- #define **RESET_INT** INT_GPIOA

Functions

- void ResetInit (void)

  Initialise the reset module.

6.14.1 Detailed Description

Soft reset module.

6.14.2 Function Documentation

6.14.2.1 ResetInit()

```
void ResetInit (
            void )
```

Initialise the reset module.

Configures the system to reset via an interrupt on the required GPIO pin.

6.15 src/reset.h File Reference

Soft reset module.

Functions

- void ResetInit (void)

  Initialise the reset module.

6.15.1 Detailed Description

Soft reset module.

6.15.2 Function Documentation

6.15.2.1 ResetInit()

```
void ResetInit (
            void  )
```

Initialise the reset module.

Configures the system to reset via an interrupt on the required GPIO pin.

6.16 src/serial_interface.c File Reference

Serial UART interface.

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include "serial_interface.h"
```

Macros

- #define UART_BASE UART0_BASE

  UART definitions.
- #define **UART_PORT** 0
- #define **UART_GPIO_BASE** GPIO_PORTA_BASE
- #define **UART_GPIO_RX_CONFIG** GPIO_PA0_U0RX
- #define **UART_GPIO_TX_CONFIG** GPIO_PA1_U0TX
- #define **UART_GPIO_RX_PIN** GPIO_PIN_0
- #define **UART_GPIO_TX_PIN** GPIO_PIN_1
- #define **UART_PERIPH_UART** SYSCTL_PERIPH_UART0
- #define **UART_PIOSC_FREQUENCY** 16000000

Functions

- void SerialInit ()

  Initialise the UART serial interface.

6.16.1 Detailed Description

Serial UART interface.

6.17 src/serial_interface.h File Reference

Serial UART interface.

Macros

- #define BAUD_RATE 9600

  UART baud rate (Hz).

Functions

- void SerialInit ()

  Initialise the UART serial interface.

6.17.1 Detailed Description

Serial UART interface.

6.18 src/switch.c File Reference

A module to operate the mode switch.

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "switch.h"
```

Macros

- #define SWITCH_PERIPH SYSCTL_PERIPH_GPIOA

  Switch definitions.
- #define **SWITCH_BASE** GPIO_PORTA_BASE
- #define **SWITCH_PIN** GPIO_PIN_7
- #define **SWITCH_DEFAULT** 0

Functions

- void SwitchInit ()

  Initialise the switch.
- void UpdateSwitch ()

  Update the switch state.
- uint8_t GetSwitchEvent ()

  Get the switch event.

6.18.1 Detailed Description

A module to operate the mode switch.

6.19 src/switch.h File Reference

A module to operate the mode switch.

Macros

- #define NUM_POLLS 5

  Change switch state only after NUM_POLLS consecutive readings have an opposite value.

Enumerations

- enum { **SWITCH_DOWN**, **SWITCH_UP** }

Functions

- void SwitchInit (void)

  Initialise the switch.
- void UpdateSwitch ()

  Update the switch state.
- uint8_t GetSwitchEvent (void)

  Get the switch event.

6.19.1 Detailed Description

A module to operate the mode switch.

6.20 src/yaw.c File Reference

Module to handle changes in yaw and detect the reference yaw position.

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/sysctl.h"
#include "yaw.h"
```

Macros

- #define YAW_PERIPH SYSCTL_PERIPH_GPIOB

  Yaw definitions.
- #define **YAW_BASE** GPIO_PORTB_BASE
- #define **YAW_CHANNEL_A** GPIO_PIN_0
- #define **YAW_CHANNEL_B** GPIO_PIN_1
- #define **YAW_GPIO_PINS** (YAW_CHANNEL_A | YAW_CHANNEL_B)
- #define **YAW_INT** INT_GPIOB

- #define YAW_REF_PERIPH SYSCTL_PERIPH_GPIOC

  Reference yaw definitions.
- #define **YAW_REF_BASE** GPIO_PORTC_BASE
- #define **YAW_REF_PIN** GPIO_PIN_4
- #define **YAW_REF_INT** INT_GPIOC

Functions

- void YawDetectionInit (void)

  Initialises the yaw manager.
- void YawRefTrigger (void)

  Triggers an interrupt to fire when the refernce yaw has been found.
- bool YawRefFound (void)

  Check if the reference yaw has been found.
- int32_t GetYaw (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.
- int32_t GetYawDegrees (void)

  Get the current yaw.

6.20.1 Detailed Description

Module to handle changes in yaw and detect the reference yaw position.

6.21 src/yaw.h File Reference

Module to handle changes in yaw and detect the reference yaw position.

Macros

- #define NUMBER_SLOTS 112

  The number of slots in 360 degrees of rotation.
- #define YAW_FULL_ROTATION (NUMBER_SLOTS ∗ 4)

  The number of yaw updates in 360 degrees of rotation.

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 refernce yaw has been found.
- bool YawRefFound (void)

  Check if the reference yaw has been found.

6.21.1 Detailed Description

Module to handle changes in yaw and detect the reference yaw position.

6.22 src/yaw_controller.h File Reference

Pid controller for the Tail rotor.

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)
- void YawControllerInit (void)

  Initialise the yaw controller.
- void PreloadYawController (int32_t control, int32_t error)
- void UpdateYawController (uint32_t delta_t)
- void TuneProportionalTailRotor (double gain)

  Use at own risk.

6.22.1 Detailed Description

Pid controller for the Tail rotor.

Index