

## AS7 On-Board Software

Generated by Doxygen 1.9.5



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 AS7 Namespace Reference	7
4.1.1 Enumeration Type Documentation	7
4.1.1.1 DroneCommandType	7
<b>5 Class Documentation</b>	<b>9</b>
5.1 AS7::Drone Class Reference	9
5.1.1 Constructor & Destructor Documentation	10
5.1.1.1 Drone()	10
5.1.2 Member Function Documentation	10
5.1.2.1 allowArming()	10
5.1.2.2 channelConfirm()	10
5.1.2.3 dequeueCommand()	10
5.1.2.4 disableOperatorControl()	11
5.1.2.5 droneAllowedToFly()	11
5.1.2.6 droneHasArmed()	11
5.1.2.7 droneStatus()	11
5.1.2.8 emergencyStop()	11
5.1.2.9 enableOperatorControl()	11
5.1.2.10 enqueueCommand()	11
5.1.2.11 generateEStopTx()	12
5.1.2.12 getCompassHeading()	12
5.1.2.13 getDroneArmComplete()	12
5.1.2.14 getEnableEmergencyStop()	12
5.1.2.15 getEnableOperatorControl()	12
5.1.2.16 getEStopTx()	12
5.1.2.17 getSbusRxArray()	12
5.1.2.18 getSbusTxArray()	12
5.1.2.19 getUsBack()	13
5.1.2.20 getUsDown()	13
5.1.2.21 getUsFront()	13
5.1.2.22 getUsLeft()	13
5.1.2.23 getUsRight()	13
5.1.2.24 getUsUp()	13
5.1.2.25 pause()	13

5.1.2.26 recordingEnabled()	13
5.1.2.27 resetEmergencyStop()	14
5.1.2.28 resume()	14
5.1.2.29 setDataGathering()	14
5.1.2.30 start()	14
5.1.3 Member Data Documentation	14
5.1.3.1 _dataGatheringEnabled	14
5.1.3.2 compassHeading	14
5.1.3.3 usBack	14
5.1.3.4 usDown	15
5.1.3.5 usFront	15
5.1.3.6 usLeft	15
5.1.3.7 usRight	15
5.1.3.8 usUp	15
5.2 AS7::DroneCommand Struct Reference	15
5.2.1 Member Data Documentation	16
5.2.1.1 channels	16
5.2.1.2 dataRecording	16
5.2.1.3 desc	16
5.2.1.4 duration	16
5.2.1.5 p_pt	17
5.2.1.6 p_rl	17
5.2.1.7 p_x	17
5.2.1.8 p_y	17
5.2.1.9 p_yw	17
5.2.1.10 p_z	17
5.2.1.11 rateMultiplier	17
5.2.1.12 type	17
5.2.1.13 v_pt	18
5.2.1.14 v_rl	18
5.2.1.15 v_x	18
5.2.1.16 v_y	18
5.2.1.17 v_yw	18
5.2.1.18 v_z	18
5.3 AS7::Logger Class Reference	18
5.3.1 Constructor & Destructor Documentation	19
5.3.1.1 Logger()	19
5.3.2 Member Function Documentation	19
5.3.2.1 disableSDLogging()	19
5.3.2.2 error()	19
5.3.2.3 fatal()	20
5.3.2.4 inform()	20

5.3.2.5 pause()	20
5.3.2.6 plot()	20
5.3.2.7 pushData()	20
5.3.2.8 recordData()	20
5.3.2.9 resume()	20
5.3.2.10 running()	21
5.3.2.11 setVerbosity()	21
5.3.2.12 start()	21
5.3.2.13 verbose()	21
5.3.2.14 verbosity()	21
5.3.2.15 warn()	21
<b>6 File Documentation</b>	<b>23</b>
6.1 H:/repos/Argous/src/esp32/as7-gamma/as7-gamma.ino File Reference	23
6.1.1 Macro Definition Documentation	26
6.1.1.1 ACC_FREQ	26
6.1.1.2 COLOR_ORDER [1/2]	26
6.1.1.3 COLOR_ORDER [2/2]	26
6.1.1.4 GRAVITY_CMS	26
6.1.1.5 I2C_SCL	26
6.1.1.6 I2C_SDA	27
6.1.1.7 LED_TYPE [1/2]	27
6.1.1.8 LED_TYPE [2/2]	27
6.1.1.9 SIMULATION_ENABLE	27
6.1.2 Enumeration Type Documentation	27
6.1.2.1 DroneDebugTest	27
6.1.2.2 DroneFlightMode	27
6.1.2.3 DroneState	28
6.1.3 Function Documentation	28
6.1.3.1 debug_switchModes()	28
6.1.3.2 DEFINE_GRADIENT_PALETTE()	28
6.1.3.3 loop()	28
6.1.3.4 readAccelerometer()	29
6.1.3.5 setup()	29
6.1.3.6 taskAccelerometer()	29
6.1.3.7 taskStatusLeds()	29
6.1.3.8 taskUltrasonicSensor()	29
6.1.4 Variable Documentation	29
6.1.4.1 accel_curr_millis	29
6.1.4.2 accel_delta_s	29
6.1.4.3 accel_filt_x	30
6.1.4.4 accel_filt_y	30

6.1.4.5 accel_filt_z	30
6.1.4.6 accel_index	30
6.1.4.7 accel_prev_millis	30
6.1.4.8 accel_total_x	30
6.1.4.9 accel_total_y	30
6.1.4.10 accel_total_z	30
6.1.4.11 accel_x	31
6.1.4.12 accel_x_int	31
6.1.4.13 accel_x_offset	31
6.1.4.14 accel_x_readings	31
6.1.4.15 accel_y	31
6.1.4.16 accel_y_int	31
6.1.4.17 accel_y_offset	31
6.1.4.18 accel_y_readings	31
6.1.4.19 accel_z	32
6.1.4.20 accel_z_int	32
6.1.4.21 accel_z_offset	32
6.1.4.22 accel_z_readings	32
6.1.4.23 accelData	32
6.1.4.24 accelerometerCalibrated	32
6.1.4.25 compass	32
6.1.4.26 compass_event	32
6.1.4.27 compass_heading	33
6.1.4.28 compass_x	33
6.1.4.29 compass_y	33
6.1.4.30 compass_z	33
6.1.4.31 config_file	33
6.1.4.32 currentFlightMode	33
6.1.4.33 currentState	33
6.1.4.34 debug_switchModesSemaphore	33
6.1.4.35 declinationAngle	34
6.1.4.36 DEVPAL	34
6.1.4.37 DIV_I	34
6.1.4.38 droneStateMap	34
6.1.4.39 enable_pilotSemaphore	34
6.1.4.40 enable_scribeSemaphore	34
6.1.4.41 enable_usSemaphore	35
6.1.4.42 estDronePos_x	35
6.1.4.43 estDronePos_y	35
6.1.4.44 estDronePos_z	35
6.1.4.45 filt_pos_x	35
6.1.4.46 filt_pos_y	35

6.1.4.47 filt_pos_z	35
6.1.4.48 filt_vel_x	35
6.1.4.49 filt_vel_y	36
6.1.4.50 filt_vel_z	36
6.1.4.51 FL_LED	36
6.1.4.52 FL_LEDBRIGHTNESS	36
6.1.4.53 FL_LEDNUM	36
6.1.4.54 FL_LEDPIN	36
6.1.4.55 log_file	36
6.1.4.56 logger	36
6.1.4.57 MAX_US_DISTANCE	37
6.1.4.58 nextState	37
6.1.4.59 NUM_ACCEL_READINGS	37
6.1.4.60 NUM_US_POINTS	37
6.1.4.61 NUM_US_SENSORS	37
6.1.4.62 output_file	37
6.1.4.63 raw_pos_x	37
6.1.4.64 raw_pos_y	37
6.1.4.65 raw_pos_z	38
6.1.4.66 raw_vel_x	38
6.1.4.67 raw_vel_y	38
6.1.4.68 raw_vel_z	38
6.1.4.69 recordingEnabled	38
6.1.4.70 SBUS_RXPIN	38
6.1.4.71 SBUS_TXPIN	38
6.1.4.72 sbusRxData	38
6.1.4.73 sbusTxData	39
6.1.4.74 scribe_logSemaphore	39
6.1.4.75 scribe_logStackMutex	39
6.1.4.76 scribe_msgSemaphore	39
6.1.4.77 scribe_msgStackMutex	39
6.1.4.78 Serial	39
6.1.4.79 Serial1	39
6.1.4.80 STATUS_FASTLED_PIN	39
6.1.4.81 STATUS_LED_BRIGHTNESS	40
6.1.4.82 STATUS_NUM_LEDS	40
6.1.4.83 TEMP_PIN	40
6.1.4.84 th_Accel	40
6.1.4.85 th_Comms	40
6.1.4.86 th_Switch	40
6.1.4.87 th_Ultrasonic	40
6.1.4.88 TICK_LONG	40

6.1.4.89 TICK_LONGLONG	41
6.1.4.90 TICK_MEDIUM	41
6.1.4.91 TICK_SHORT	41
6.1.4.92 TICK_US_DELAY	41
6.1.4.93 tmp_temperature	41
6.1.4.94 us_distance	41
6.1.4.95 US_ECHOPIN	41
6.1.4.96 US_ECHOPIN_0	41
6.1.4.97 US_ECHOPIN_1	42
6.1.4.98 US_ECHOPIN_2	42
6.1.4.99 US_ECHOPIN_3	42
6.1.4.100 US_ECHOPIN_4	42
6.1.4.101 US_ECHOPIN_5	42
6.1.4.102 us_rawDistance	42
6.1.4.103 US_TRIGPIN	42
6.1.4.104 US_TRIGPIN_0	42
6.1.4.105 US_TRIGPIN_1	43
6.1.4.106 US_TRIGPIN_2	43
6.1.4.107 US_TRIGPIN_3	43
6.1.4.108 US_TRIGPIN_4	43
6.1.4.109 US_TRIGPIN_5	43
6.2 H:/repos/Argous/src/esp32/as7-gamma/Drone.cpp File Reference	43
6.3 H:/repos/Argous/src/esp32/as7-gamma/Drone.h File Reference	43
6.3.1 Macro Definition Documentation	44
6.3.1.1 AS7DRONE	44
6.3.1.2 CH_ESTOP	45
6.3.1.3 CH_FLIGHTMODE	45
6.3.1.4 CH_STRAFE	45
6.3.1.5 CH_STRAIGHT	45
6.3.1.6 CH_SW1	45
6.3.1.7 CH_THROTTLE	45
6.3.1.8 CH_YAW	45
6.3.1.9 CTL_FREQ	45
6.3.1.10 DOF	46
6.3.1.11 NAV_FREQ	46
6.3.1.12 NUM_CH	46
6.3.1.13 RAMPRATE_LINEAR	46
6.3.1.14 RAMPRATE_NONE	46
6.3.1.15 RAMPRATE_PROP	46
6.3.1.16 SBUS_CHANNEL_LOWER	46
6.3.1.17 SBUS_CHANNEL_UPPER	46
6.3.1.18 STATUS_UPDATE_DELAY	47



---

6.3.1.19 THROTTLE_LIMIT . . . . .	47
6.4 Drone.h . . . . .	47
6.5 H:/repos/Argous/src/esp32/as7-gamma/SdLogger.cpp File Reference . . . . .	50
6.6 H:/repos/Argous/src/esp32/as7-gamma/SdLogger.h File Reference . . . . .	51
6.6.1 Macro Definition Documentation . . . . .	51
6.6.1.1 AS7SDLOGGING_H . . . . .	51
6.6.1.2 CS_PIN . . . . .	52
6.6.1.3 LOG_LEVEL_ERROR . . . . .	52
6.6.1.4 LOG_LEVEL_FATAL . . . . .	52
6.6.1.5 LOG_LEVEL_INFORM . . . . .	52
6.6.1.6 LOG_LEVEL_SILENT . . . . .	52
6.6.1.7 LOG_LEVEL_VERBOSE . . . . .	52
6.6.1.8 LOG_LEVEL_WARNING . . . . .	52
6.6.1.9 LOGGER_FREQ . . . . .	52
6.6.1.10 PLOTTER_ENABLE . . . . .	53
6.6.1.11 SD_DISABLED . . . . .	53
6.7 SdLogger.h . . . . .	53
<b>Index</b>	<b>55</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">AS7</a> . . . . .	<a href="#">7</a>
-------------------------------	-------------------



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AS7::Drone</a>	9
<a href="#">AS7::DroneCommand</a>	15
<a href="#">AS7::Logger</a>	18



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

H:/repos/Argous/src/esp32/as7-gamma/as7-gamma.ino . . . . .	23
H:/repos/Argous/src/esp32/as7-gamma/Drone.cpp . . . . .	43
H:/repos/Argous/src/esp32/as7-gamma/Drone.h . . . . .	43
H:/repos/Argous/src/esp32/as7-gamma/SdLogger.cpp . . . . .	50
H:/repos/Argous/src/esp32/as7-gamma/SdLogger.h . . . . .	51





## Chapter 4

# Namespace Documentation

### 4.1 AS7 Namespace Reference

#### Classes

- class [Drone](#)
- struct [DroneCommand](#)
- class [Logger](#)

#### Enumerations

- enum [DroneCommandType](#) { [Blind](#) , [Guided](#) , [Landing](#) , [Arm](#) }

#### 4.1.1 Enumeration Type Documentation

##### 4.1.1.1 DroneCommandType

enum [AS7::DroneCommandType](#)

#### Enumerator

Blind	
Guided	
Landing	
Arm	



## Chapter 5

# Class Documentation

### 5.1 AS7::Drone Class Reference

```
#include <Drone.h>
```

#### Public Member Functions

- [Drone](#) ([Logger](#) \*logger, bfs::SbusRx \*sbus\_rx, bfs::SbusTx \*sbus\_tx)
- bool [channelConfirm](#) (int16\_t channel=1, float threshold=0.7f)
- int [droneStatus](#) ()
- bool [getEnableOperatorControl](#) ()
- bool [getEnableEmergencyStop](#) ()
- bool [getDroneArmComplete](#) ()
- float [getUsFront](#) ()
- float [getUsBack](#) ()
- float [getUsLeft](#) ()
- float [getUsRight](#) ()
- float [getUsUp](#) ()
- float [getUsDown](#) ()
- float [getCompassHeading](#) ()
- void [start](#) (int core=1, int priority=3)
- void [pause](#) ()
- void [resume](#) ()
- void [enqueueCommand](#) ([DroneCommand](#) cmd)
- [DroneCommand](#) [dequeueCommand](#) ()
- void [allowArming](#) ()
- bool [droneAllowedToFly](#) () const
- bool [droneHasArmed](#) () const
- void [enableOperatorControl](#) ()
- void [disableOperatorControl](#) ()
- void [emergencyStop](#) ()
- void [resetEmergencyStop](#) ()
- void [setDataGathering](#) (bool value)
- bool [recordingEnabled](#) ()
- void [generateEStopTx](#) ()
- std::array< int16\_t, NUM\_CH > [getEStopTx](#) ()
- std::string [getSbusRxArray](#) ()
- std::string [getSbusTxArray](#) ()

## Public Attributes

- float `usFront` = 200
- float `usBack` = 200
- float `usLeft` = 200
- float `usRight` = 200
- float `usUp` = 200
- float `usDown` = 200
- float `compassHeading`
- bool `_dataGatheringEnabled` = false

## 5.1.1 Constructor & Destructor Documentation

### 5.1.1.1 Drone()

```
AS7::Drone::Drone (
    Logger * logger,
    bfs::SbusRx * sbus_rx,
    bfs::SbusTx * sbus_tx )
```

## 5.1.2 Member Function Documentation

### 5.1.2.1 allowArming()

```
void AS7::Drone::allowArming ( ) [inline]
```

### 5.1.2.2 channelConfirm()

```
bool AS7::Drone::channelConfirm (
    int16_t channel = 1,
    float threshold = 0.7f )
```

### 5.1.2.3 dequeueCommand()

```
DroneCommand AS7::Drone::dequeueCommand ( )
```

#### 5.1.2.4 disableOperatorControl()

```
void AS7::Drone::disableOperatorControl ( )
```

#### 5.1.2.5 droneAllowedToFly()

```
bool AS7::Drone::droneAllowedToFly ( ) const [inline]
```

#### 5.1.2.6 droneHasArmed()

```
bool AS7::Drone::droneHasArmed ( ) const [inline]
```

#### 5.1.2.7 droneStatus()

```
int AS7::Drone::droneStatus ( )
```

#### 5.1.2.8 emergencyStop()

```
void AS7::Drone::emergencyStop ( )
```

#### 5.1.2.9 enableOperatorControl()

```
void AS7::Drone::enableOperatorControl ( )
```

#### 5.1.2.10 enqueueCommand()

```
void AS7::Drone::enqueueCommand (
    DroneCommand cmd )
```

**5.1.2.11 generateEStopTx()**

```
void AS7::Drone::generateEStopTx ( )
```

**5.1.2.12 getCompassHeading()**

```
float AS7::Drone::getCompassHeading ( ) [inline]
```

**5.1.2.13 getDroneArmComplete()**

```
bool AS7::Drone::getDroneArmComplete ( ) [inline]
```

**5.1.2.14 getEnableEmergencyStop()**

```
bool AS7::Drone::getEnableEmergencyStop ( ) [inline]
```

**5.1.2.15 getEnableOperatorControl()**

```
bool AS7::Drone::getEnableOperatorControl ( ) [inline]
```

**5.1.2.16 getEStopTx()**

```
std::array< int16_t, NUM_CH > AS7::Drone::getEStopTx ( )
```

**5.1.2.17 getSbusRxArray()**

```
std::string AS7::Drone::getSbusRxArray ( )
```

**5.1.2.18 getSbusTxArray()**

```
std::string AS7::Drone::getSbusTxArray ( )
```

**5.1.2.19 getUsBack()**

```
float AS7::Drone::getUsBack ( ) [inline]
```

**5.1.2.20 getUsDown()**

```
float AS7::Drone::getUsDown ( ) [inline]
```

**5.1.2.21 getUsFront()**

```
float AS7::Drone::getUsFront ( ) [inline]
```

**5.1.2.22 getUsLeft()**

```
float AS7::Drone::getUsLeft ( ) [inline]
```

**5.1.2.23 getUsRight()**

```
float AS7::Drone::getUsRight ( ) [inline]
```

**5.1.2.24 getUsUp()**

```
float AS7::Drone::getUsUp ( ) [inline]
```

**5.1.2.25 pause()**

```
void AS7::Drone::pause ( )
```

**5.1.2.26 recordingEnabled()**

```
bool AS7::Drone::recordingEnabled ( ) [inline]
```

#### 5.1.2.27 resetEmergencyStop()

```
void AS7::Drone::resetEmergencyStop ( )
```

#### 5.1.2.28 resume()

```
void AS7::Drone::resume ( )
```

#### 5.1.2.29 setDataGathering()

```
void AS7::Drone::setDataGathering (
    bool value ) [inline]
```

#### 5.1.2.30 start()

```
void AS7::Drone::start (
    int core = 1,
    int priority = 3 )
```

### 5.1.3 Member Data Documentation

#### 5.1.3.1 \_dataGatheringEnabled

```
bool AS7::Drone::_dataGatheringEnabled = false
```

#### 5.1.3.2 compassHeading

```
float AS7::Drone::compassHeading
```

#### 5.1.3.3 usBack

```
float AS7::Drone::usBack = 200
```



#### 5.1.3.4 usDown

```
float AS7::Drone::usDown = 200
```

#### 5.1.3.5 usFront

```
float AS7::Drone::usFront = 200
```

#### 5.1.3.6 usLeft

```
float AS7::Drone::usLeft = 200
```

#### 5.1.3.7 usRight

```
float AS7::Drone::usRight = 200
```

#### 5.1.3.8 usUp

```
float AS7::Drone::usUp = 200
```

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

- [H:/repos/Argous/src/esp32/as7-gamma/Drone.h](#)
- [H:/repos/Argous/src/esp32/as7-gamma/Drone.cpp](#)

## 5.2 AS7::DroneCommand Struct Reference

```
#include <Drone.h>
```

## Public Attributes

- [DroneCommandType](#) type
- std::string [desc](#)
- std::array< float, [NUM\\_CH](#) > [channels](#)
- int [duration](#)
- float [rateMultiplier](#) = 1.0f
- float [p\\_x](#)
- float [p\\_y](#)
- float [p\\_z](#)
- float [v\\_x](#)
- float [v\\_y](#)
- float [v\\_z](#)
- float [p\\_rl](#)
- float [p\\_pt](#)
- float [p\\_yw](#)
- float [v\\_rl](#)
- float [v\\_pt](#)
- float [v\\_yw](#)
- bool [dataRecording](#)

## 5.2.1 Member Data Documentation

### 5.2.1.1 channels

```
std::array<float, NUM\_CH> AS7::DroneCommand::channels
```

### 5.2.1.2 dataRecording

```
bool AS7::DroneCommand::dataRecording
```

### 5.2.1.3 desc

```
std::string AS7::DroneCommand::desc
```

### 5.2.1.4 duration

```
int AS7::DroneCommand::duration
```

### 5.2.1.5 p\_pt

```
float AS7::DroneCommand::p_pt
```

### 5.2.1.6 p\_rl

```
float AS7::DroneCommand::p_rl
```

### 5.2.1.7 p\_x

```
float AS7::DroneCommand::p_x
```

### 5.2.1.8 p\_y

```
float AS7::DroneCommand::p_y
```

### 5.2.1.9 p\_yw

```
float AS7::DroneCommand::p_yw
```

### 5.2.1.10 p\_z

```
float AS7::DroneCommand::p_z
```

### 5.2.1.11 rateMultiplier

```
float AS7::DroneCommand::rateMultiplier = 1.0f
```

### 5.2.1.12 type

```
DroneCommandType AS7::DroneCommand::type
```

#### 5.2.1.13 v\_pt

```
float AS7::DroneCommand::v_pt
```

#### 5.2.1.14 v\_rl

```
float AS7::DroneCommand::v_rl
```

#### 5.2.1.15 v\_x

```
float AS7::DroneCommand::v_x
```

#### 5.2.1.16 v\_y

```
float AS7::DroneCommand::v_y
```

#### 5.2.1.17 v\_yw

```
float AS7::DroneCommand::v_yw
```

#### 5.2.1.18 v\_z

```
float AS7::DroneCommand::v_z
```

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

- [H:/repos/Argous/src/esp32/as7-gamma/Drone.h](#)

## 5.3 AS7::Logger Class Reference

```
#include <SdLogger.h>
```

## Public Member Functions

- [Logger](#) (Print \*output)
- void [start](#) (int core=1, int priority=1, int [verbosity](#)=LOG\_LEVEL\_INFORM)
- void [pause](#) ()
- void [resume](#) ()
- bool [running](#) ()
- int [verbosity](#) ()
- void [setVerbosity](#) (int [verbosity](#))
- void [recordData](#) (std::string key, float value)
- void [pushData](#) ()
- void [inform](#) (std::string message)
- void [warn](#) (std::string message)
- void [error](#) (std::string message)
- void [fatal](#) (std::string message)
- void [verbose](#) (std::string message)
- void [plot](#) (std::string message)
- void [disableSDLogging](#) ()

## 5.3.1 Constructor & Destructor Documentation

### 5.3.1.1 Logger()

```
AS7::Logger::Logger (
    Print * output )
```

## 5.3.2 Member Function Documentation

### 5.3.2.1 disableSDLogging()

```
void AS7::Logger::disableSDLogging ( )
```

### 5.3.2.2 error()

```
void AS7::Logger::error (
    std::string message )
```

#### 5.3.2.3 fatal()

```
void AS7::Logger::fatal (
    std::string message )
```

#### 5.3.2.4 inform()

```
void AS7::Logger::inform (
    std::string message )
```

#### 5.3.2.5 pause()

```
void AS7::Logger::pause ( )
```

#### 5.3.2.6 plot()

```
void AS7::Logger::plot (
    std::string message )
```

#### 5.3.2.7 pushData()

```
void AS7::Logger::pushData ( )
```

#### 5.3.2.8 recordData()

```
void AS7::Logger::recordData (
    std::string key,
    float value )
```

#### 5.3.2.9 resume()

```
void AS7::Logger::resume ( )
```

#### 5.3.2.10 running()

```
bool AS7::Logger::running ( )
```

#### 5.3.2.11 setVerbosity()

```
void AS7::Logger::setVerbosity (
    int verbosity )
```

#### 5.3.2.12 start()

```
void AS7::Logger::start (
    int core = 1,
    int priority = 1,
    int verbosity = LOG_LEVEL_INFORM )
```

#### 5.3.2.13 verbose()

```
void AS7::Logger::verbose (
    std::string message )
```

#### 5.3.2.14 verbosity()

```
int AS7::Logger::verbosity ( )
```

#### 5.3.2.15 warn()

```
void AS7::Logger::warn (
    std::string message )
```

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

- [H:/repos/Argous/src/esp32/as7-gamma/SdLogger.h](#)
- [H:/repos/Argous/src/esp32/as7-gamma/SdLogger.cpp](#)





## Chapter 6

# File Documentation

### 6.1 H:/repos/Argous/src/esp32/as7-gamma/as7-gamma.ino File Reference

```
#include <sbus.h>
#include <FastLED.h>
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Adafruit_MMC5883.h>
#include <map>
#include "SdLogger.h"
#include "Drone.h"
```

#### Macros

- #define I2C\_SDA 21
- #define I2C\_SCL 22
- #define SIMULATION\_ENABLE false
- #define LED\_TYPE WS2811
- #define COLOR\_ORDER GRB
- #define GRAVITY\_CMS 980.6
- #define ACC\_FREQ 200
- #define LED\_TYPE WS2811
- #define COLOR\_ORDER GRB

#### Enumerations

- enum DroneState {  
    Initialise , Ready , Armed , Flying ,  
    Landing , Stopped , Faulted , Debug }
- enum DroneFlightMode { OperatorControl , AutoStraightLine , ArmOnly }
- enum DroneDebugTest { SBUS\_COMMS , ARMING\_DISARMING , SD\_READ\_WRITE , LED\_RESPONSE }

## Functions

- [DEFINE\\_GRADIENT\\_PALETTE](#) (DEV\_PALLETE)
- void [readAccelerometer](#) ()
- void [taskUltrasonicSensor](#) (void \*parameters)
- void [taskAccelerometer](#) (void \*parameters)
- void [debug\\_switchModes](#) (void \*parameters)
- void [taskStatusLeds](#) (void \*parameters)
- void [setup](#) ()
- void [loop](#) ()

## Variables

- const int [STATUS\\_FASTLED\\_PIN](#) = 3
- const int [STATUS\\_NUM\\_LEDS](#) = 8
- const int [STATUS\\_LED\\_BRIGHTNESS](#) = 32
- const int [NUM\\_US\\_SENSORS](#) = 6
- const int [NUM\\_US\\_POINTS](#) = 3
- const int [MAX\\_US\\_DISTANCE](#) = 2000
- const int [US\\_TRIGPIN\\_0](#) = 32
- const int [US\\_ECHOPIN\\_0](#) = 35
- const int [US\\_TRIGPIN\\_1](#) = 25
- const int [US\\_ECHOPIN\\_1](#) = 33
- const int [US\\_TRIGPIN\\_2](#) = 12
- const int [US\\_ECHOPIN\\_2](#) = 13
- const int [US\\_TRIGPIN\\_3](#) = 14
- const int [US\\_ECHOPIN\\_3](#) = 27
- const int [US\\_TRIGPIN\\_4](#) = 0
- const int [US\\_ECHOPIN\\_4](#) = 4
- const int [US\\_TRIGPIN\\_5](#) = 2
- const int [US\\_ECHOPIN\\_5](#) = 15
- int [US\\_TRIGPIN](#) [[NUM\\_US\\_SENSORS](#)] = {}
- int [US\\_ECHOPIN](#) [[NUM\\_US\\_SENSORS](#)] = {}
- const int [FL\\_LEDPIN](#) = 3
- const int [SBUS\\_RXPIN](#) = 16
- const int [SBUS\\_TXPIN](#) = 17
- const int [TEMP\\_PIN](#) = 34
- const TickType\_t [TICK\\_US\\_DELAY](#) = 80 / portTICK\_PERIOD\_MS
- const TickType\_t [TICK\\_SHORT](#) = 200 / portTICK\_PERIOD\_MS
- const TickType\_t [TICK\\_MEDIUM](#) = 500 / portTICK\_PERIOD\_MS
- const TickType\_t [TICK\\_LONG](#) = 1000 / portTICK\_PERIOD\_MS
- const TickType\_t [TICK\\_LOGLONG](#) = 5000 / portTICK\_PERIOD\_MS
- const int [FL\\_LEDNUM](#) = 8
- const int [FL\\_LEDBRIGHTNESS](#) = 64
- CRGB [FL\\_LED](#) [[FL\\_LEDNUM](#)]
- CRGBPalette16 [DEVPAL](#) = DEV\_PALLETE
- int8\_t [accel\\_x\\_int](#)
- int8\_t [accel\\_y\\_int](#)
- int8\_t [accel\\_z\\_int](#)
- float [accel\\_total\\_x](#)
- float [accel\\_total\\_y](#)
- float [accel\\_total\\_z](#)
- const float [DIV\\_I](#) = 16
- byte [accelData](#) [3]

- const int [NUM\\_ACCEL\\_READINGS](#) = 5
- unsigned long [accel\\_prev\\_millis](#) = 0
- unsigned long [accel\\_curr\\_millis](#) = 0
- float [accel\\_delta\\_s](#) = 0
- float [accel\\_x\\_readings](#) [[NUM\\_ACCEL\\_READINGS](#)]
- float [accel\\_y\\_readings](#) [[NUM\\_ACCEL\\_READINGS](#)]
- float [accel\\_z\\_readings](#) [[NUM\\_ACCEL\\_READINGS](#)]
- int [accel\\_index](#) = 0
- float [accel\\_x](#)
- float [accel\\_y](#)
- float [accel\\_z](#)
- float [accel\\_filt\\_x](#)
- float [accel\\_filt\\_y](#)
- float [accel\\_filt\\_z](#)
- float [raw\\_vel\\_x](#) = 0
- float [raw\\_vel\\_y](#) = 0
- float [raw\\_vel\\_z](#) = 0
- float [filt\\_vel\\_x](#) = 0
- float [filt\\_vel\\_y](#) = 0
- float [filt\\_vel\\_z](#) = 0
- float [raw\\_pos\\_x](#) = 0
- float [raw\\_pos\\_y](#) = 0
- float [raw\\_pos\\_z](#) = 0
- float [filt\\_pos\\_x](#) = 0
- float [filt\\_pos\\_y](#) = 0
- float [filt\\_pos\\_z](#) = 0
- float [compass\\_x](#)
- float [compass\\_y](#)
- float [compass\\_z](#)
- float [compass\\_heading](#)
- const float [declinationAngle](#) = 0.192228625
- float [accel\\_x\\_offset](#) = 0
- float [accel\\_y\\_offset](#) = 0
- float [accel\\_z\\_offset](#) = 0
- bool [accelerometerCalibrated](#) = false
- float [estDronePos\\_x](#)
- float [estDronePos\\_y](#)
- float [estDronePos\\_z](#)
- sensors\_event\_t [compass\\_event](#)
- bfs::SbusRx sbusRx & [Serial1](#)
- std::array< int16\_t, [NUM\\_CH](#) > [sbusTxData](#)
- std::array< int16\_t, [NUM\\_CH](#) > [sbusRxData](#)
- File [config\\_file](#)
- File [output\\_file](#)
- File [log\\_file](#)
- bool [recordingEnabled](#) = false
- int [us\\_distance](#) [[NUM\\_US\\_SENSORS](#)][[NUM\\_US\\_POINTS](#)]
- int [us\\_rawDistance](#) [[NUM\\_US\\_SENSORS](#)]
- int [tmp\\_temperature](#)
- SemaphoreHandle\_t [enable\\_usSemaphore](#)
- SemaphoreHandle\_t [enable\\_pilotSemaphore](#)
- SemaphoreHandle\_t [debug\\_switchModesSemaphore](#)
- SemaphoreHandle\_t [enable\\_scribeSemaphore](#)
- SemaphoreHandle\_t [scribe\\_logSemaphore](#)
- SemaphoreHandle\_t [scribe\\_msgSemaphore](#)

- SemaphoreHandle\_t [scribe\\_logStackMutex](#)
- SemaphoreHandle\_t [scribe\\_msgStackMutex](#)
- TaskHandle\_t [th\\_Ultrasonic](#)
- TaskHandle\_t [th\\_Comms](#)
- TaskHandle\_t [th\\_Accel](#)
- TaskHandle\_t [th\\_Switch](#)
- std::map< [DroneState](#), std::string > [droneStateMap](#)
- [DroneState](#) [currentState](#) = [Initialise](#)
- [DroneState](#) [nextState](#) = [Initialise](#)
- [DroneFlightMode](#) [currentFlightMode](#) = [AutoStraightLine](#)
- [AS7::Logger](#) [logger](#) & [Serial](#)
- [AS7::Drone](#) [drone](#) & [logger](#)
- [Adafruit\\_MMC5883](#) [compass](#) = [Adafruit\\_MMC5883\(11111\)](#)

## 6.1.1 Macro Definition Documentation

### 6.1.1.1 ACC\_FREQ

```
#define ACC_FREQ 200
```

### 6.1.1.2 COLOR\_ORDER [1/2]

```
#define COLOR_ORDER GRB
```

### 6.1.1.3 COLOR\_ORDER [2/2]

```
#define COLOR_ORDER GRB
```

### 6.1.1.4 GRAVITY\_CMS

```
#define GRAVITY_CMS 980.6
```

### 6.1.1.5 I2C\_SCL

```
#define I2C_SCL 22
```

#### 6.1.1.6 I2C\_SDA

```
#define I2C_SDA 21
```

#### 6.1.1.7 LED\_TYPE [1/2]

```
#define LED_TYPE WS2811
```

#### 6.1.1.8 LED\_TYPE [2/2]

```
#define LED_TYPE WS2811
```

#### 6.1.1.9 SIMULATION\_ENABLE

```
#define SIMULATION_ENABLE false
```

### 6.1.2 Enumeration Type Documentation

#### 6.1.2.1 DroneDebugTest

```
enum DroneDebugTest
```

Enumerator

SBUS_COMMS	
ARMING_DISARMING	
SD_READ_WRITE	
LED_RESPONSE	

#### 6.1.2.2 DroneFlightMode

```
enum DroneFlightMode
```

**Enumerator**

OperatorControl	
AutoStraightLine	
ArmOnly	

**6.1.2.3 DroneState**

enum [DroneState](#)

**Enumerator**

Initialise	
Ready	
Armed	
Flying	
Landing	
Stopped	
Faulted	
Debug	

**6.1.3 Function Documentation****6.1.3.1 debug\_switchModes()**

```
void debug_switchModes (
    void * parameters )
```

**6.1.3.2 DEFINE\_GRADIENT\_PALETTE()**

```
DEFINE_GRADIENT_PALETTE (
    DEV_PALLETE )
```

**6.1.3.3 loop()**

```
void loop ( )
```

#### 6.1.3.4 readAccelerometer()

```
void readAccelerometer ( )
```

#### 6.1.3.5 setup()

```
void setup ( )
```

#### 6.1.3.6 taskAccelerometer()

```
void taskAccelerometer (
    void * parameters )
```

#### 6.1.3.7 taskStatusLeds()

```
void taskStatusLeds (
    void * parameters )
```

#### 6.1.3.8 taskUltrasonicSensor()

```
void taskUltrasonicSensor (
    void * parameters )
```

### 6.1.4 Variable Documentation

#### 6.1.4.1 accel\_curr\_millis

```
unsigned long accel_curr_millis = 0
```

#### 6.1.4.2 accel\_delta\_s

```
float accel_delta_s = 0
```

#### 6.1.4.3 accel\_filt\_x

```
float accel_filt_x
```

#### 6.1.4.4 accel\_filt\_y

```
float accel_filt_y
```

#### 6.1.4.5 accel\_filt\_z

```
float accel_filt_z
```

#### 6.1.4.6 accel\_index

```
int accel_index = 0
```

#### 6.1.4.7 accel\_prev\_millis

```
unsigned long accel_prev_millis = 0
```

#### 6.1.4.8 accel\_total\_x

```
float accel_total_x
```

#### 6.1.4.9 accel\_total\_y

```
float accel_total_y
```

#### 6.1.4.10 accel\_total\_z

```
float accel_total_z
```



#### 6.1.4.11 accel\_x

```
float accel_x
```

#### 6.1.4.12 accel\_x\_int

```
int8_t accel_x_int
```

#### 6.1.4.13 accel\_x\_offset

```
float accel_x_offset = 0
```

#### 6.1.4.14 accel\_x\_readings

```
float accel_x_readings[NUM\_ACCEL\_READINGS]
```

#### 6.1.4.15 accel\_y

```
float accel_y
```

#### 6.1.4.16 accel\_y\_int

```
int8_t accel_y_int
```

#### 6.1.4.17 accel\_y\_offset

```
float accel_y_offset = 0
```

#### 6.1.4.18 accel\_y\_readings

```
float accel_y_readings[NUM\_ACCEL\_READINGS]
```

#### 6.1.4.19 accel\_z

```
float accel_z
```

#### 6.1.4.20 accel\_z\_int

```
int8_t accel_z_int
```

#### 6.1.4.21 accel\_z\_offset

```
float accel_z_offset = 0
```

#### 6.1.4.22 accel\_z\_readings

```
float accel_z_readings[NUM\_ACCEL\_READINGS]
```

#### 6.1.4.23 accelData

```
byte accelData[3]
```

#### 6.1.4.24 accelerometerCalibrated

```
bool accelerometerCalibrated = false
```

#### 6.1.4.25 compass

```
Adafruit_MMC5883 compass = Adafruit_MMC5883(11111)
```

#### 6.1.4.26 compass\_event

```
sensors_event_t compass_event
```

#### 6.1.4.27 compass\_heading

```
float compass_heading
```

#### 6.1.4.28 compass\_x

```
float compass_x
```

#### 6.1.4.29 compass\_y

```
float compass_y
```

#### 6.1.4.30 compass\_z

```
float compass_z
```

#### 6.1.4.31 config\_file

```
File config_file
```

#### 6.1.4.32 currentFlightMode

```
DroneFlightMode currentFlightMode = AutoStraightLine
```

#### 6.1.4.33 currentState

```
DroneState currentState = Initialise
```

#### 6.1.4.34 debug\_switchModesSemaphore

```
SemaphoreHandle_t debug_switchModesSemaphore
```

#### 6.1.4.35 declinationAngle

```
const float declinationAngle = 0.192228625
```

#### 6.1.4.36 DEVPAL

```
CRGBPalettet16 DEVPAL = DEV_PALLETE
```

#### 6.1.4.37 DIV\_I

```
const float DIV_I =16
```

#### 6.1.4.38 droneStateMap

```
std::map<DroneState, std::string> droneStateMap
```

##### Initial value:

```
= {  
    {Initialise, "Initialise"},  
    {Ready, "Ready"},  
    {Armed, "Armed"},  
    {Flying, "Flying"},  
    {Landing, "Landing"},  
    {Stopped, "Stopped"},  
    {Faulted, "Faulted"},  
    {Debug, "Debug"}  
}
```

#### 6.1.4.39 enable\_pilotSemaphore

```
SemaphoreHandle_t enable_pilotSemaphore
```

#### 6.1.4.40 enable\_scribeSemaphore

```
SemaphoreHandle_t enable_scribeSemaphore
```

#### 6.1.4.41 enable\_usSemaphore

```
SemaphoreHandle_t enable_usSemaphore
```

#### 6.1.4.42 estDronePos\_x

```
float estDronePos_x
```

#### 6.1.4.43 estDronePos\_y

```
float estDronePos_y
```

#### 6.1.4.44 estDronePos\_z

```
float estDronePos_z
```

#### 6.1.4.45 filt\_pos\_x

```
float filt_pos_x = 0
```

#### 6.1.4.46 filt\_pos\_y

```
float filt_pos_y = 0
```

#### 6.1.4.47 filt\_pos\_z

```
float filt_pos_z = 0
```

#### 6.1.4.48 filt\_vel\_x

```
float filt_vel_x = 0
```

#### 6.1.4.49 `filt_vel_y`

```
float filt_vel_y = 0
```

#### 6.1.4.50 `filt_vel_z`

```
float filt_vel_z = 0
```

#### 6.1.4.51 `FL_LED`

```
CRGB FL_LED[FL_LEDNUM]
```

#### 6.1.4.52 `FL_LEDBRIGHTNESS`

```
const int FL_LEDBRIGHTNESS = 64
```

#### 6.1.4.53 `FL_LEDNUM`

```
const int FL_LEDNUM = 8
```

#### 6.1.4.54 `FL_LEDPIN`

```
const int FL_LEDPIN = 3
```

#### 6.1.4.55 `log_file`

```
File log_file
```

#### 6.1.4.56 `logger`

```
AS7::Drone drone& logger
```

#### 6.1.4.57 MAX\_US\_DISTANCE

```
const int MAX_US_DISTANCE = 2000
```

#### 6.1.4.58 nextState

```
DroneState nextState = Initialise
```

#### 6.1.4.59 NUM\_ACCEL\_READINGS

```
const int NUM_ACCEL_READINGS = 5
```

#### 6.1.4.60 NUM\_US\_POINTS

```
const int NUM_US_POINTS = 3
```

#### 6.1.4.61 NUM\_US\_SENSORS

```
const int NUM_US_SENSORS = 6
```

#### 6.1.4.62 output\_file

```
File output_file
```

#### 6.1.4.63 raw\_pos\_x

```
float raw_pos_x = 0
```

#### 6.1.4.64 raw\_pos\_y

```
float raw_pos_y = 0
```

#### 6.1.4.65 raw\_pos\_z

```
float raw_pos_z = 0
```

#### 6.1.4.66 raw\_vel\_x

```
float raw_vel_x = 0
```

#### 6.1.4.67 raw\_vel\_y

```
float raw_vel_y = 0
```

#### 6.1.4.68 raw\_vel\_z

```
float raw_vel_z = 0
```

#### 6.1.4.69 recordingEnabled

```
bool recordingEnabled = false
```

#### 6.1.4.70 SBUS\_RXPIN

```
const int SBUS_RXPIN = 16
```

#### 6.1.4.71 SBUS\_TXPIN

```
const int SBUS_TXPIN = 17
```

#### 6.1.4.72 sbusRxData

```
std::array<int16_t, NUM_CH> sbusRxData
```



#### 6.1.4.73 sbusTxData

```
std::array<int16_t, NUM_CH> sbusTxData
```

#### 6.1.4.74 scribe\_logSemaphore

```
SemaphoreHandle_t scribe_logSemaphore
```

#### 6.1.4.75 scribe\_logStackMutex

```
SemaphoreHandle_t scribe_logStackMutex
```

#### 6.1.4.76 scribe\_msgSemaphore

```
SemaphoreHandle_t scribe_msgSemaphore
```

#### 6.1.4.77 scribe\_msgStackMutex

```
SemaphoreHandle_t scribe_msgStackMutex
```

#### 6.1.4.78 Serial

```
AS7::Logger logger& Serial
```

#### 6.1.4.79 Serial1

```
bfs::SbusTx sbusTx & Serial1
```

#### 6.1.4.80 STATUS\_FASTLED\_PIN

```
const int STATUS_FASTLED_PIN = 3
```

#### 6.1.4.81 STATUS\_LED\_BRIGHTNESS

```
const int STATUS_LED_BRIGHTNESS = 32
```

#### 6.1.4.82 STATUS\_NUM\_LEDS

```
const int STATUS_NUM_LEDS = 8
```

#### 6.1.4.83 TEMP\_PIN

```
const int TEMP_PIN = 34
```

#### 6.1.4.84 th\_Accel

```
TaskHandle_t th_Accel
```

#### 6.1.4.85 th\_Comms

```
TaskHandle_t th_Comms
```

#### 6.1.4.86 th\_Switch

```
TaskHandle_t th_Switch
```

#### 6.1.4.87 th\_Ultrasonic

```
TaskHandle_t th_Ultrasonic
```

#### 6.1.4.88 TICK\_LONG

```
const TickType_t TICK_LONG = 1000 / portTICK_PERIOD_MS
```

#### 6.1.4.89 TICK\_LONGLONG

```
const TickType_t TICK_LONGLONG = 5000 / portTICK_PERIOD_MS
```

#### 6.1.4.90 TICK\_MEDIUM

```
const TickType_t TICK_MEDIUM = 500 / portTICK_PERIOD_MS
```

#### 6.1.4.91 TICK\_SHORT

```
const TickType_t TICK_SHORT = 200 / portTICK_PERIOD_MS
```

#### 6.1.4.92 TICK\_US\_DELAY

```
const TickType_t TICK_US_DELAY = 80 / portTICK_PERIOD_MS
```

#### 6.1.4.93 tmp\_temperature

```
int tmp_temperature
```

#### 6.1.4.94 us\_distance

```
int us_distance[NUM\_US\_SENSORS][NUM\_US\_POINTS]
```

#### 6.1.4.95 US\_ECHOPIN

```
int US_ECHOPIN[NUM\_US\_SENSORS] = {}
```

#### 6.1.4.96 US\_ECHOPIN\_0

```
const int US_ECHOPIN_0 = 35
```

#### 6.1.4.97 US\_ECHOPIN\_1

```
const int US_ECHOPIN_1 = 33
```

#### 6.1.4.98 US\_ECHOPIN\_2

```
const int US_ECHOPIN_2 = 13
```

#### 6.1.4.99 US\_ECHOPIN\_3

```
const int US_ECHOPIN_3 = 27
```

#### 6.1.4.100 US\_ECHOPIN\_4

```
const int US_ECHOPIN_4 = 4
```

#### 6.1.4.101 US\_ECHOPIN\_5

```
const int US_ECHOPIN_5 = 15
```

#### 6.1.4.102 us\_rawDistance

```
int us_rawDistance[NUM\_US\_SENSORS]
```

#### 6.1.4.103 US\_TRIGPIN

```
int US_TRIGPIN[NUM\_US\_SENSORS] = {}
```

#### 6.1.4.104 US\_TRIGPIN\_0

```
const int US_TRIGPIN_0 = 32
```

#### 6.1.4.105 US\_TRIGPIN\_1

```
const int US_TRIGPIN_1 = 25
```

#### 6.1.4.106 US\_TRIGPIN\_2

```
const int US_TRIGPIN_2 = 12
```

#### 6.1.4.107 US\_TRIGPIN\_3

```
const int US_TRIGPIN_3 = 14
```

#### 6.1.4.108 US\_TRIGPIN\_4

```
const int US_TRIGPIN_4 = 0
```

#### 6.1.4.109 US\_TRIGPIN\_5

```
const int US_TRIGPIN_5 = 2
```

## 6.2 H:/repos/Argous/src/esp32/as7-gamma/Drone.cpp File Reference

```
#include "Drone.h"
```

### Namespaces

- namespace [AS7](#)

## 6.3 H:/repos/Argous/src/esp32/as7-gamma/Drone.h File Reference

```
#include <Arduino.h>
#include <sbus.h>
#include <vector>
#include "SdLogger.h"
```

## Classes

- struct [AS7::DroneCommand](#)
- class [AS7::Drone](#)

## Namespaces

- namespace [AS7](#)

## Macros

- `#define` [AS7DRONE](#)
- `#define` [SBUS\\_CHANNEL\\_LOWER](#) 0
- `#define` [SBUS\\_CHANNEL\\_UPPER](#) 2056
- `#define` [NUM\\_CH](#) 16
- `#define` [DOF](#) 6
- `#define` [NAV\\_FREQ](#) 250
- `#define` [CTL\\_FREQ](#) 1000
- `#define` [CH\\_THROTTLE](#) 2
- `#define` [CH\\_YAW](#) 3
- `#define` [CH\\_STRAIGHT](#) 1
- `#define` [CH\\_STRAFE](#) 0
- `#define` [CH\\_SW1](#) 7
- `#define` [CH\\_FLIGHTMODE](#) 4
- `#define` [CH\\_ESTOP](#) 6
- `#define` [STATUS\\_UPDATE\\_DELAY](#) 250
- `#define` [THROTTLE\\_LIMIT](#) 0.5f
- `#define` [RAMPRATE\\_NONE](#) 0
- `#define` [RAMPRATE\\_LINEAR](#) 1
- `#define` [RAMPRATE\\_PROP](#) 2

## Enumerations

- enum [AS7::DroneCommandType](#) { [AS7::Blind](#) , [AS7::Guided](#) , [AS7::Landing](#) , [AS7::Arm](#) }

### 6.3.1 Macro Definition Documentation

#### 6.3.1.1 AS7DRONE

```
#define AS7DRONE
```

TODO: THIS BANNER

[AS7](#) Drone This class encapsulates talking to [AS7](#)

The main program controls the state that [AS7](#) is in This class represents the inputs/outputs for [AS7](#)

#### 6.3.1.2 CH\_ESTOP

```
#define CH_ESTOP 6
```

#### 6.3.1.3 CH\_FLIGHTMODE

```
#define CH_FLIGHTMODE 4
```

#### 6.3.1.4 CH\_STRAFE

```
#define CH_STRAFE 0
```

#### 6.3.1.5 CH\_STRAIGHT

```
#define CH_STRAIGHT 1
```

#### 6.3.1.6 CH\_SW1

```
#define CH_SW1 7
```

#### 6.3.1.7 CH\_THROTTLE

```
#define CH_THROTTLE 2
```

#### 6.3.1.8 CH\_YAW

```
#define CH_YAW 3
```

#### 6.3.1.9 CTL\_FREQ

```
#define CTL_FREQ 1000
```

**6.3.1.10 DOF**

```
#define DOF 6
```

**6.3.1.11 NAV\_FREQ**

```
#define NAV_FREQ 250
```

**6.3.1.12 NUM\_CH**

```
#define NUM_CH 16
```

**6.3.1.13 RAMPRATE\_LINEAR**

```
#define RAMPRATE_LINEAR 1
```

**6.3.1.14 RAMPRATE\_NONE**

```
#define RAMPRATE_NONE 0
```

**6.3.1.15 RAMPRATE\_PROP**

```
#define RAMPRATE_PROP 2
```

**6.3.1.16 SBUS\_CHANNEL\_LOWER**

```
#define SBUS_CHANNEL_LOWER 0
```

**6.3.1.17 SBUS\_CHANNEL\_UPPER**

```
#define SBUS_CHANNEL_UPPER 2056
```



## 6.3.1.18 STATUS\_UPDATE\_DELAY

```
#define STATUS_UPDATE_DELAY 250
```

## 6.3.1.19 THROTTLE\_LIMIT

```
#define THROTTLE_LIMIT 0.5f
```

## 6.4 Drone.h

[Go to the documentation of this file.](#)

```
1
14 #pragma once
15 #ifndef AS7DRONE
16 #define AS7DRONE
17
18 #include <Arduino.h>
19 #include <sbus.h> // SBUS Communication Library with FC
20 #include <vector>
21
22 #include "SdLogger.h" // Use the SD Logger to share messages
23
24 #define SBUS_CHANNEL_LOWER 0 // Default lower bound for sbus channels
25 #define SBUS_CHANNEL_UPPER 2056 // Default upper bound for sbus channels
26 #define NUM_CH 16 // Number of SBUS channels. Always 16. Equivalent to
    bfs::SbusRx::NUM_CH()
27 #define DOF 6 // Degrees of freedom for the drone. 0-5 represent x, y, z, roll
    (rl), pitch (pt), yaw (yw) (Euler ZYX Convention)
28
29 // Thread update frequencies (to reduce starvation of the watchdog)
30 // Feed your watchdogs, people! They do important household chores!
31 #define NAV_FREQ 250 // Navigation update rate, Hz (Default: 250 Hz)
32 #define CTL_FREQ 1000 // Controller update rate, Hz (Default: 250 Hz)
33
34 // Channel definitions
35 // These channels index from ZERO. Ch[0] = CH1!
36 #define CH_THROTTLE 2 // Left stick y axis (starts from 0)
37 #define CH_YAW 3 // Left stick x axis
38 #define CH_STRAIGHT 1 // Right stick y axis
39 #define CH_STRAFE 0 // Right stick x axis
40 // #define CH_BUTTON1 6 // Button on middle of controller
41 #define CH_SW1 7 // Right toggle switch
42 #define CH_FLIGHTMODE 4 // Left toggle switch
43 #define CH_ESTOP 6 // Other button?
44
45 #define STATUS_UPDATE_DELAY 250 // Number of updates to wait before sending controller status
46
47 #define THROTTLE_LIMIT 0.5f
48
49 // Cv = Control Value, Pv = Present Value. Use CV to control PV
50 #define RAMPRATE_NONE 0 // No ramp rate. Pv = Cv
51 #define RAMPRATE_LINEAR 1 // Linear ramp rate, Pv += Constant until Pv > Cv
52 #define RAMPRATE_PROP 2 // Proportional Ramp Rate Pv = (Cv - Pv) * Constant
53
54 namespace AS7
55 {
56     enum DroneCommandType {Blind, Guided, Landing, Arm};
57
58     // Drone Command Structure/Format
59     // Two types: Blind and Guided, set by enum DroneCommandType
60     // Blind commands only refer to the channel array and duration
61     // Guided commands will attempt to use on-board sensors to control the drone
62     //
63     // Landing is a special type where the drone will lower thrust just below its known weight to land
64     // If possible, it will use the bottom sensors to guide landing
65     // Landing is equivalent to setting v_y to some pre-defined value in blind mode.
66     //
67     // Arm is a special type that will send the arming command to the drone.
68     // As SBUS is one-way at the moment, it's not possible to *know* if the drone is armed but we can
    generally assume
69     // that after a certain duration, the drone is armed.
```

```

70 // Because of this, there is no way to know if the drone is disarmed as it will disarm automatically
    from the FC
71 // More advanced implementations using two-way SBUS or ideally MAVLink can get around this
    limitation.
72 //
73 // Drone commands are enqueued to the drone, and will be executed FIFO.
74 //
75 // Note: Blind Commands can also be used as buttons and inputs to the FC, not just for navigation
76 //
77 typedef struct {
78     DroneCommandType type; // Blind = Purely a drone command, Guided = Assisted with
    sensors
79     std::string desc; // A description for the logger
80     std::array<float, NUM_CH> channels; // A float for each channel from (-1, 1)
81     int duration; // in ms
82     float rateMultiplier = 1.0f; // Can be thought of as "aggressiveness" of controls
83
84     // Prefix P = Position; V = Velocity. Units on per-member basis and are *convention* (not
    checked)
85     float p_x; // Position to hold (some distance unit tbc)
86     float p_y;
87     float p_z;
88     float v_x; // Velocity to hold (-1 to 1) floating point value
89     float v_y; // e.g. 0.1 is equivalent to 10% forward thrust
90     float v_z;
91     float p_rl; // Not directly controllable on a drone
92     float p_pt; // Not directly controllable on a drone
93     float p_yw;
94     float v_rl; // Not directly controllable on a drone
95     float v_pt; // Not directly controllable on a drone
96     float v_yw;
97     bool dataRecording; // indicates if the drone should record data, used to synchronise sensors and
    current drone state
98 } DroneCommand;
99
100 class Drone
101 {
102 private:
103     TaskHandle_t thDrone;
104     TaskHandle_t thRemote;
105     static void startNavTask(void*); // Task implementation for classes
106     static void startCtlTask(void*);
107     void navigationTask(void* parameters); // The threaded task
108     void controllerTask(void* parameters);
109
110     std::queue<DroneCommand> _droneCommandQueue;
111
112     SemaphoreHandle_t _semDroneEnableMutex; // Enables/Disables main drone task
113     SemaphoreHandle_t _semControlEnableMutex; // Enables/Disables main drone task
114     SemaphoreHandle_t getSemDroneEnableMutex(); // Returns the enable mutex
115     SemaphoreHandle_t getSemControlEnableMutex(); // Returns the enable mutex
116
117     SemaphoreHandle_t _semTxChMutex; // Mutex Lock for the tx data array
118     SemaphoreHandle_t getTxChMutex(); // Returns the write mutex for threading implementation
119
120     SemaphoreHandle_t _semRxChMutex; // Mutex Lock for the rx data array
121     SemaphoreHandle_t getRxChMutex(); // Returns the write mutex for threading implementation
122
123     SemaphoreHandle_t _semCommandQueueMutex; // Mutex lock for drone command queue
124     SemaphoreHandle_t getCommandQueueMutex();
125
126     Logger* _logger;
127     Logger* getLogger();
128
129     bfs::SbusRx* _sbusRx; // SBUS Receive Channel Object
130     bfs::SbusTx* _sbusTx; // SBUS Transmit Channel Object
131
132     std::array<int16_t, NUM_CH> _sbusRxData; // Array of data received from the Radio Control
133     std::array<int16_t, NUM_CH> _sbusTxData; // Array of data to transmit to the Flight
    Controller
134
135     std::array<bool, NUM_CH> _sbusAbsChannels; // When true, the channel is (0, 1). Otherwise
    channels default to (-1, 1).
136     std::array<bool, NUM_CH> getSbusAbsChannels(); // Returns an array which defines if a channel
    is absolute (true) or not (false).
137     void generateAbsChannels(); // Sets the default Absolute Channels
138
139     int _controllerStatusCount = 0;
140     bool getControllerStatusCount();
141
142     // Channels that will be transmitted to the drone
143     std::array<int16_t, NUM_CH> _sbusTxChLower; // Lower bounds for SBUS Transmit channels
144     std::array<int16_t, NUM_CH> _sbusTxChUpper; // Upper bounds for SBUS Transmit Channels
145
146     // Channels that are received from the transmitter
147     std::array<int16_t, NUM_CH> _sbusRxChLower; // Lower bounds for SBUS Receiver channels

```

```

149     std::array<int16_t, NUM_CH> _sbusRxChUpper;    // Upper bounds for SBUS Receiver Channels
150
151     std::array<int16_t, NUM_CH> _sbusEStopTx;      // Data to be written in case of an E-Stop. Not
all channels are to be zeroed!
152
153     // Methods for getting and setting data for task theads
154     inline std::array<int16_t, NUM_CH> getSbusRxData() {return _sbusRxData; }
155     inline std::array<int16_t, NUM_CH> getSbusTxData() {return _sbusTxData; }
156     void setSbusRxData(std::array<int16_t, NUM_CH> data);
157     void setSbusTxData(std::array<int16_t, NUM_CH> data);
158
159     void initUpperLowerBoundArrays();    // Sets UBound and LBound array to default
160
161     void writeChannel(int16_t value, int8_t ch);    // writes the value into the sbus transmit
channel
162     void writeRxChannel_f(float value, int8_t ch);    // writes the value into the sbus transmit
channel
163     void writeTxChannel_f(float value, int8_t ch);    // writes the value into the sbus transmit
channel
164
165     int16_t convRxChannel_i(float value, int8_t ch);    // Returns the adjusted int16_t value for that
channel
166     float convRxChannel_f(int16_t value, int8_t ch);    // Returns the adjusted int16_t value for that
channel
167
168     int16_t convTxChannel_i(float value, int8_t ch);    // Returns the adjusted int16_t value for that
channel
169     float convTxChannel_f(int16_t value, int8_t ch);    // Returns the adjusted int16_t value for that
channel
170
171     int16_t readRxChannel(int16_t ch);                // Reads the value from the channel
172     float readRxChannel_f(int16_t ch);                // Reads the floating point value from the
channel, adjusted for upper and lower bounds
173
174     int16_t readTxChannel(int16_t ch);                // Reads the current value being written to the
controller
175     float readTxChannel_f(int16_t ch);                // Returns the current value as a floating point
number
176
177     // Helper/Utility functions
178     float clamp(float value, float lbound, float ubound);    // Returns
values inside of upper bound and lower bound.
179     std::string formatSbusArray(std::array<int16_t, NUM_CH> chData);    // Returns the
channels in a formatted string
180     float rampValue(float value, float target = 0, float rate = 0, int rampRateType =
RAMPRATE_LINEAR);    // Returns the next ramped value depending on ramp type
181
182     // Combines rampValue with current channel data for ramping.
183     void rampChannel(float target, int8_t ch, float rate, int rampRateType = RAMPRATE_LINEAR);
184
185     bfs::SbusRx* getSbusRx();    // Returns SBUS RX object for task implementation
186     bfs::SbusTx* getSbusTx();    // Returns SBUS TX object for task implementation
187
188
189     bool _hasArmed = false;    // Remembers if the drone has undergone an arming process
190     bool _armingAllowed = false;    // Set by main program. Once allowed, drone will start
processing instructions
191     inline void setDroneHasArmed() {_hasArmed = true;}
192     inline bool getDroneHasArmed() {return _hasArmed; }
193
194     bool _droneCommandsStarted = false;    // Indicates if the drone has started processing
commands
195     bool _droneCommandsCompleted = false;    // Indicates that there are no commands left (queue is
empty)
196     bool _droneHasActiveCommand = false;
197     inline bool getHasActiveComamnd()const {return _droneHasActiveCommand;}
198     inline void setHasActiveCommand(bool value) {_droneHasActiveCommand = value;}
199     inline void setDroneCommandsStarted() {_droneCommandsStarted = true;}    // Latching
check that the drone has started commands
200     inline void setDroneCommandsCompleted() {_droneCommandsCompleted = true;}    // Latching
check that the rone has completed commands. If commands haven't started, then this indicates empty.
201     inline bool nextCommandAvailable() {return _droneCommandQueue.size() != 0;}    // Checks if
command queue size is not equal to zero.
202
203     bool _running = false;    // Indicates current state of main drone task
204     bool _enableOperatorControl = false;    // When enabled, remote control commands are passed
directly to drone from RX to TX
205     bool _enableEmergencyStop = false;    // When enabled, all TX channels are set to 0
206     bool _armingComplete = false;
207
208     public:
209
210     Drone(Logger *logger, bfs::SbusRx* sbus_rx, bfs::SbusTx* sbus_tx);
211     bool channelConfirm(int16_t channel=1, float threshold=0.7f);    // Returns true if the channel
above threshold. e.g. button press
212
213     // Drone status is indicated by an int, though it could be indicated by an enum later on with

```

```

DEFINES
214 // Current statuses could be drone_starting, drone_waitin0gdrone_in_progress, drone_estop,
    drone_operator_over, drone_finished
215 // Not sure how the internal mechanism could work -- this could be a bunmch of bools with
    increasnig preference for noe another another? maybe we shouldn't even consider status inside the
    drone *command* class
216 int droneStatus();
217
218 inline bool getEnableOperatorControl() {return _enableOperatorControl; } // Returns
operator control bit
219 inline bool getEnableEmergencyStop() {return _enableEmergencyStop; } // Returns e-stop
bit
220 inline bool getDroneArmComplete() {return _armingComplete; }
221
222 // Assume everything is far so we don't shock the drone into a position
223
224 float usFront = 200; // Distance measurement from the front US sensor
225 float usBack = 200; // Distance measurement from the back US sensor
226 float usLeft = 200; // Distance measurement from the left US sensor
227 float usRight = 200; // Distance measurement from the right US sensor
228 float usUp = 200; // Distance measurement from the upwards US sensor
229 float usDown = 200; // Distance measurement from the downwards US sensor
230
231 inline float getUsFront() {return usFront; };
232 inline float getUsBack() {return usBack; };
233 inline float getUsLeft() {return usLeft; };
234 inline float getUsRight() {return usRight; };
235 inline float getUsUp() {return usUp; };
236 inline float getUsDown() {return usDown; };
237
238 float compassHeading;
239 inline float getCompassHeading() {return compassHeading; };
240
241
242
243 // Main thread control
244 // Operates both the controller and navigator threads
245 void start(int core=1, int priority=3);
246 void pause();
247 void resume();
248
249 void enqueueCommand(DroneCommand cmd); // Adds command to drone queue
250 DroneCommand dequeueCommand(); // Remove drone command, returns command from queue
251
252 inline void allowArming() {_armingAllowed = true;} // Allows drone to start processing commands
253 inline bool droneAllowedToFly()const {return _armingAllowed;} // Returns _armingAllowed bit
254 inline bool droneHasArmed()const {return _hasArmed;} // Returns if dorn has armed
previously
255
256 void enableOperatorControl(); // Enables pass-through from RX to TX. Latching
257 void disableOperatorControl();
258
259 void emergencyStop();
260 void resetEmergencyStop();
261
262 bool _dataGatheringEnabled = false;
263 inline void setDataGathering(bool value) {_dataGatheringEnabled = value; }
264 inline bool recordingEnabled() {return _dataGatheringEnabled;} // Returns true if the current
drone command is requesting data gathering
265
266 void generateEStopTx();
267 std::array<int16_t, NUM_CH> getEStopTx();
268
269 std::string getSbusRxArray();
270 std::string getSbusTxArray();
271 };
272 }
273
274
275 #endif

```

## 6.5 H:/repos/Argous/src/esp32/as7-gamma/SdLogger.cpp File Reference

```
#include "SdLogger.h"
```

### Namespaces

- namespace [AS7](#)

## 6.6 H:/repos/Argous/src/esp32/as7-gamma/SdLogger.h File Reference

```
#include <Arduino.h>
#include <queue>
#include <SPI.h>
#include <SD.h>
#include <map>
```

### Classes

- class [AS7::Logger](#)

### Namespaces

- namespace [AS7](#)

### Macros

- #define [AS7SDLOGGING\\_H](#)
- #define [LOG\\_LEVEL\\_SILENT](#) 0
- #define [LOG\\_LEVEL\\_FATAL](#) 1
- #define [LOG\\_LEVEL\\_ERROR](#) 2
- #define [LOG\\_LEVEL\\_WARNING](#) 3
- #define [LOG\\_LEVEL\\_INFORM](#) 4
- #define [LOG\\_LEVEL\\_VERBOSE](#) 5
- #define [CS\\_PIN](#) 5
- #define [LOGGER\\_FREQ](#) 100
- #define [PLOTTER\\_ENABLE](#) false
- #define [SD\\_DISABLED](#) false

### 6.6.1 Macro Definition Documentation

#### 6.6.1.1 AS7SDLOGGING\_H

```
#define AS7SDLOGGING_H
```

TODO: THIS BANNER

This is [AS7](#)'s Logger. It prints to serial and SD card. The goal of this logger is to tie logging and comms into one location and delegate to a thread Read this for getters and setters inside freertos <https://forums.freertos.org/t/freertos-task-in-a-c-class/13984>

#### 6.6.1.2 CS\_PIN

```
#define CS_PIN 5
```

#### 6.6.1.3 LOG\_LEVEL\_ERROR

```
#define LOG_LEVEL_ERROR 2
```

#### 6.6.1.4 LOG\_LEVEL\_FATAL

```
#define LOG_LEVEL_FATAL 1
```

#### 6.6.1.5 LOG\_LEVEL\_INFORM

```
#define LOG_LEVEL_INFORM 4
```

#### 6.6.1.6 LOG\_LEVEL\_SILENT

```
#define LOG_LEVEL_SILENT 0
```

#### 6.6.1.7 LOG\_LEVEL\_VERBOSE

```
#define LOG_LEVEL_VERBOSE 5
```

#### 6.6.1.8 LOG\_LEVEL\_WARNING

```
#define LOG_LEVEL_WARNING 3
```

#### 6.6.1.9 LOGGER\_FREQ

```
#define LOGGER_FREQ 100
```

## 6.6.1.10 PLOTTER\_ENABLE

```
#define PLOTTER_ENABLE false
```

## 6.6.1.11 SD\_DISABLED

```
#define SD_DISABLED false
```

## 6.7 SdLogger.h

[Go to the documentation of this file.](#)

```
1
2
3
4
5
6
7
8 #pragma once
9 #ifndef AS7SDLOGGING_H
10 #define AS7SDLOGGING_H
11
12 #include <Arduino.h>
13 #include <queue>
14 #include <SPI.h>
15 #include <SD.h>
16
17 #include <map>
18
19 // Defines the maximum level of messages sent through the serial port
20 // e.g. a level of WARNING (3) will only allow warnings, errors, and fatal issues to be sent to the
    serial port.
21 #define LOG_LEVEL_SILENT 0
22 #define LOG_LEVEL_FATAL 1
23 #define LOG_LEVEL_ERROR 2
24 #define LOG_LEVEL_WARNING 3
25 #define LOG_LEVEL_INFORM 4
26 #define LOG_LEVEL_VERBOSE 5
27 #define CS_PIN 5
28
29 #define LOGGER_FREQ 100 // Update rate in Hertz
30
31 #define PLOTTER_ENABLE false // only prints plots, for testing.
32 #define SD_DISABLED false
33
34 namespace AS7
35 {
36     class Logger
37     {
38     private:
39
40         std::queue<std::string> _msg_Queue;
41         std::queue<std::string> _log_Queue;
42         // Used for reading the message Queues for the scribe task
43         SemaphoreHandle_t _sem_log;
44         SemaphoreHandle_t _sem_msg;
45
46         // Mutex for the message Queues
47         SemaphoreHandle_t _sem_logQueueMutex;
48         SemaphoreHandle_t _sem_msgQueueMutex;
49
50         SemaphoreHandle_t _sem_dataMutex;
51         SemaphoreHandle_t _sem_dataEnqMutex;
52
53         // Sem for Enabling/Disabling Task
54         SemaphoreHandle_t _sem_enableMutex;
55         bool _running = false; // tracks if the thread is running or stopped
56         bool _sdEnabled = false;
57         bool _sdDetected = false;
58
59         std::map<std::string, float> _activeData; // Data that is actively written to
60         std::map<std::string, float> _enqueuedData; // Used as a buffer before being written
61         bool _hasEnqueuedData = false;
62
63         File _logFile;
64         File _dataFile;
65         File _configFile;
66
67     };
68 }
```

```

67     File getLogFile();
68     File getDataFile();
69     File getConfigFile();
70
71     void openLogFile();      // Opens Log file on SD in Append
72     void openDataFile();    // Opens Data file on SD in Append
73     void closeLogFile();    // Closes Log File
74     void closeDataFile();   // Closes Data File
75
76     const std::string _logFileLocation = "/as7.log"; // Location of the logging file, includes
extension. Use .c_str() to for SD library
77     const std::string _dataFileLocation = "/data.csv"; // Location of the data file, includes
extension. Use .c_str() to for SD library
78
79     Print* _printer;
80
81     Print* getPrinter();
82     int _verbosity = LOG_LEVEL_INFORM;
83
84     std::string getTestMessage();
85     std::queue<std::string> getMsgQueue();
86     std::queue<std::string> getLogQueue();
87
88     SemaphoreHandle_t getSemLog();
89     SemaphoreHandle_t getSemMsg();
90     SemaphoreHandle_t getSemLogQueueMutex();
91     SemaphoreHandle_t getSemMsgQueueMutex();
92
93     inline SemaphoreHandle_t getSemDataMutex() {return _sem_dataMutex; }
94     inline SemaphoreHandle_t getSemDataEnqMutex() {return _sem_dataEnqMutex; }
95
96     SemaphoreHandle_t getSemEnableMutex();
97
98
99     // Adds a message to be recorded to the SD card
100    // diagnostics probably? will flush out later
101    void enqueueMsg(std::string message);
102
103    // Adds a message to be sent to the console and onto the SD card
104    void enqueueLog(std::string message, int verbosity);
105
106    // for PLY generation, will need to be flushed out.
107    // String since we can also send header information
108    void enqueuePnt(std::string points);
109
110    std::string dequeueLog();
111
112    // Implementation to start FreeRTOS tasks in classes
113    static void startTaskImpl(void*);
114    TaskHandle_t th_logger;
115    void mainTask(void* parameters);
116
117    void initialiseSD();
118
119    public:
120    Logger(Print* output);
121    void start(int core=1, int priority=1, int verbosity=LOG_LEVEL_INFORM);
122    void pause();
123    void resume();
124
125    bool running();
126    int verbosity();
127
128    void setVerbosity(int verbosity);
129
130    void recordData(std::string key, float value);
131    void pushData();
132
133    // The main logging tasks
134    void inform(std::string message);
135    void warn(std::string message);
136    void error(std::string message);
137    void fatal(std::string message);
138    void verbose(std::string message);
139
140    void plot(std::string message);
141
142    void disableSDLogging(); // Disables SD Logging, even if SD Card is attached.
143
144    };
145 }
146
147
148
149 #endif

```



# Index

`_dataGatheringEnabled`  
AS7::Drone, 14

`ACC_FREQ`  
as7-gamma.ino, 26  
`accel_curr_millis`  
as7-gamma.ino, 29  
`accel_delta_s`  
as7-gamma.ino, 29  
`accel_filt_x`  
as7-gamma.ino, 29  
`accel_filt_y`  
as7-gamma.ino, 30  
`accel_filt_z`  
as7-gamma.ino, 30  
`accel_index`  
as7-gamma.ino, 30  
`accel_prev_millis`  
as7-gamma.ino, 30  
`accel_total_x`  
as7-gamma.ino, 30  
`accel_total_y`  
as7-gamma.ino, 30  
`accel_total_z`  
as7-gamma.ino, 30  
`accel_x`  
as7-gamma.ino, 30  
`accel_x_int`  
as7-gamma.ino, 31  
`accel_x_offset`  
as7-gamma.ino, 31  
`accel_x_readings`  
as7-gamma.ino, 31  
`accel_y`  
as7-gamma.ino, 31  
`accel_y_int`  
as7-gamma.ino, 31  
`accel_y_offset`  
as7-gamma.ino, 31  
`accel_y_readings`  
as7-gamma.ino, 31  
`accel_z`  
as7-gamma.ino, 31  
`accel_z_int`  
as7-gamma.ino, 32  
`accel_z_offset`  
as7-gamma.ino, 32  
`accel_z_readings`  
as7-gamma.ino, 32  
`accelData`

as7-gamma.ino, 32  
`accelerometerCalibrated`  
as7-gamma.ino, 32  
`allowArming`  
AS7::Drone, 10  
`Arm`  
AS7, 7  
`Armed`  
as7-gamma.ino, 28  
`ARMING_DISARMING`  
as7-gamma.ino, 27  
`ArmOnly`  
as7-gamma.ino, 28  
`AS7`, 7  
Arm, 7  
Blind, 7  
DroneCommandType, 7  
Guided, 7  
Landing, 7  
as7-gamma.ino  
ACC\_FREQ, 26  
accel\_curr\_millis, 29  
accel\_delta\_s, 29  
accel\_filt\_x, 29  
accel\_filt\_y, 30  
accel\_filt\_z, 30  
accel\_index, 30  
accel\_prev\_millis, 30  
accel\_total\_x, 30  
accel\_total\_y, 30  
accel\_total\_z, 30  
accel\_x, 30  
accel\_x\_int, 31  
accel\_x\_offset, 31  
accel\_x\_readings, 31  
accel\_y, 31  
accel\_y\_int, 31  
accel\_y\_offset, 31  
accel\_y\_readings, 31  
accel\_z, 31  
accel\_z\_int, 32  
accel\_z\_offset, 32  
accel\_z\_readings, 32  
accelData, 32  
accelerometerCalibrated, 32  
Armed, 28  
ARMING\_DISARMING, 27  
ArmOnly, 28  
AutoStraightLine, 28

COLOR\_ORDER, 26  
 compass, 32  
 compass\_event, 32  
 compass\_heading, 32  
 compass\_x, 33  
 compass\_y, 33  
 compass\_z, 33  
 config\_file, 33  
 currentFlightMode, 33  
 currentState, 33  
 Debug, 28  
 debug\_switchModes, 28  
 debug\_switchModesSemaphore, 33  
 declinationAngle, 33  
 DEFINE\_GRADIENT\_PALETTE, 28  
 DEVPAL, 34  
 DIV\_I, 34  
 DroneDebugTest, 27  
 DroneFlightMode, 27  
 DroneState, 28  
 droneStateMap, 34  
 enable\_pilotSemaphore, 34  
 enable\_scribeSemaphore, 34  
 enable\_usSemaphore, 34  
 estDronePos\_x, 35  
 estDronePos\_y, 35  
 estDronePos\_z, 35  
 Faulted, 28  
 filt\_pos\_x, 35  
 filt\_pos\_y, 35  
 filt\_pos\_z, 35  
 filt\_vel\_x, 35  
 filt\_vel\_y, 35  
 filt\_vel\_z, 36  
 FL\_LED, 36  
 FL\_LEDBRIGHTNESS, 36  
 FL\_LEDNUM, 36  
 FL\_LEDPIN, 36  
 Flying, 28  
 GRAVITY\_CMS, 26  
 I2C\_SCL, 26  
 I2C\_SDA, 26  
 Initialise, 28  
 Landing, 28  
 LED\_RESPONSE, 27  
 LED\_TYPE, 27  
 log\_file, 36  
 logger, 36  
 loop, 28  
 MAX\_US\_DISTANCE, 36  
 nextState, 37  
 NUM\_ACCEL\_READINGS, 37  
 NUM\_US\_POINTS, 37  
 NUM\_US\_SENSORS, 37  
 OperatorControl, 28  
 output\_file, 37  
 raw\_pos\_x, 37  
 raw\_pos\_y, 37  
 raw\_pos\_z, 37  
 raw\_vel\_x, 38  
 raw\_vel\_y, 38  
 raw\_vel\_z, 38  
 readAccelerometer, 28  
 Ready, 28  
 recordingEnabled, 38  
 SBUS\_COMMS, 27  
 SBUS\_RXPIN, 38  
 SBUS\_TXPIN, 38  
 sbusRxData, 38  
 sbusTxData, 38  
 scribe\_logSemaphore, 39  
 scribe\_logStackMutex, 39  
 scribe\_msgSemaphore, 39  
 scribe\_msgStackMutex, 39  
 SD\_READ\_WRITE, 27  
 Serial, 39  
 Serial1, 39  
 setup, 29  
 SIMULATION\_ENABLE, 27  
 STATUS\_FASTLED\_PIN, 39  
 STATUS\_LED\_BRIGHTNESS, 39  
 STATUS\_NUM\_LEDS, 40  
 Stopped, 28  
 taskAccelerometer, 29  
 taskStatusLeds, 29  
 taskUltrasonicSensor, 29  
 TEMP\_PIN, 40  
 th\_Accel, 40  
 th\_Comms, 40  
 th\_Switch, 40  
 th\_Ultrasonic, 40  
 TICK\_LONG, 40  
 TICK\_LONGLONG, 40  
 TICK\_MEDIUM, 41  
 TICK\_SHORT, 41  
 TICK\_US\_DELAY, 41  
 tmp\_temperature, 41  
 us\_distance, 41  
 US\_ECHOPIN, 41  
 US\_ECHOPIN\_0, 41  
 US\_ECHOPIN\_1, 41  
 US\_ECHOPIN\_2, 42  
 US\_ECHOPIN\_3, 42  
 US\_ECHOPIN\_4, 42  
 US\_ECHOPIN\_5, 42  
 us\_rawDistance, 42  
 US\_TRIGPIN, 42  
 US\_TRIGPIN\_0, 42  
 US\_TRIGPIN\_1, 42  
 US\_TRIGPIN\_2, 43  
 US\_TRIGPIN\_3, 43  
 US\_TRIGPIN\_4, 43  
 US\_TRIGPIN\_5, 43  
 AS7::Drone, 9  
 \_dataGatheringEnabled, 14  
 allowArming, 10

- channelConfirm, [10](#)
- compassHeading, [14](#)
- dequeueCommand, [10](#)
- disableOperatorControl, [10](#)
- Drone, [10](#)
- droneAllowedToFly, [11](#)
- droneHasArmed, [11](#)
- droneStatus, [11](#)
- emergencyStop, [11](#)
- enableOperatorControl, [11](#)
- enqueueCommand, [11](#)
- generateEStopTx, [11](#)
- getCompassHeading, [12](#)
- getDroneArmComplete, [12](#)
- getEnableEmergencyStop, [12](#)
- getEnableOperatorControl, [12](#)
- getEStopTx, [12](#)
- getSbusRxArray, [12](#)
- getSbusTxArray, [12](#)
- getUsBack, [12](#)
- getUsDown, [13](#)
- getUsFront, [13](#)
- getUsLeft, [13](#)
- getUsRight, [13](#)
- getUsUp, [13](#)
- pause, [13](#)
- recordingEnabled, [13](#)
- resetEmergencyStop, [13](#)
- resume, [14](#)
- setDataGathering, [14](#)
- start, [14](#)
- usBack, [14](#)
- usDown, [14](#)
- usFront, [15](#)
- usLeft, [15](#)
- usRight, [15](#)
- usUp, [15](#)
- AS7::DroneCommand, [15](#)
  - channels, [16](#)
  - dataRecording, [16](#)
  - desc, [16](#)
  - duration, [16](#)
  - p\_pt, [16](#)
  - p\_rl, [17](#)
  - p\_x, [17](#)
  - p\_y, [17](#)
  - p\_yw, [17](#)
  - p\_z, [17](#)
  - rateMultiplier, [17](#)
  - type, [17](#)
  - v\_pt, [17](#)
  - v\_rl, [18](#)
  - v\_x, [18](#)
  - v\_y, [18](#)
  - v\_yw, [18](#)
  - v\_z, [18](#)
- AS7::Logger, [18](#)
  - disableSDLogging, [19](#)
  - error, [19](#)
  - fatal, [19](#)
  - inform, [20](#)
  - Logger, [19](#)
  - pause, [20](#)
  - plot, [20](#)
  - pushData, [20](#)
  - recordData, [20](#)
  - resume, [20](#)
  - running, [20](#)
  - setVerbosity, [21](#)
  - start, [21](#)
  - verbose, [21](#)
  - verbosity, [21](#)
  - warn, [21](#)
- AS7DRONE
  - Drone.h, [44](#)
- AS7SDLOGGING\_H
  - SdLogger.h, [51](#)
- AutoStraightLine
  - as7-gamma.ino, [28](#)
- Blind
  - AS7, [7](#)
- CH\_ESTOP
  - Drone.h, [44](#)
- CH\_FLIGHTMODE
  - Drone.h, [45](#)
- CH\_STRAFE
  - Drone.h, [45](#)
- CH\_STRAIGHT
  - Drone.h, [45](#)
- CH\_SW1
  - Drone.h, [45](#)
- CH\_THROTTLE
  - Drone.h, [45](#)
- CH\_YAW
  - Drone.h, [45](#)
- channelConfirm
  - AS7::Drone, [10](#)
- channels
  - AS7::DroneCommand, [16](#)
- COLOR\_ORDER
  - as7-gamma.ino, [26](#)
- compass
  - as7-gamma.ino, [32](#)
- compass\_event
  - as7-gamma.ino, [32](#)
- compass\_heading
  - as7-gamma.ino, [32](#)
- compass\_x
  - as7-gamma.ino, [33](#)
- compass\_y
  - as7-gamma.ino, [33](#)
- compass\_z
  - as7-gamma.ino, [33](#)
- compassHeading
  - AS7::Drone, [14](#)

- config\_file
  - as7-gamma.ino, 33
- CS\_PIN
  - SdLogger.h, 51
- CTL\_FREQ
  - Drone.h, 45
- currentFlightMode
  - as7-gamma.ino, 33
- currentState
  - as7-gamma.ino, 33
- dataRecording
  - AS7::DroneCommand, 16
- Debug
  - as7-gamma.ino, 28
- debug\_switchModes
  - as7-gamma.ino, 28
- debug\_switchModesSemaphore
  - as7-gamma.ino, 33
- declinationAngle
  - as7-gamma.ino, 33
- DEFINE\_GRADIENT\_PALETTE
  - as7-gamma.ino, 28
- dequeueCommand
  - AS7::Drone, 10
- desc
  - AS7::DroneCommand, 16
- DEVPAL
  - as7-gamma.ino, 34
- disableOperatorControl
  - AS7::Drone, 10
- disableSDLogging
  - AS7::Logger, 19
- DIV\_I
  - as7-gamma.ino, 34
- DOF
  - Drone.h, 45
- Drone
  - AS7::Drone, 10
- Drone.h
  - AS7DRONE, 44
  - CH\_ESTOP, 44
  - CH\_FLIGHTMODE, 45
  - CH\_STRAFE, 45
  - CH\_STRAIGHT, 45
  - CH\_SW1, 45
  - CH\_THROTTLE, 45
  - CH\_YAW, 45
  - CTL\_FREQ, 45
  - DOF, 45
  - NAV\_FREQ, 46
  - NUM\_CH, 46
  - RAMP\_RATE\_LINEAR, 46
  - RAMP\_RATE\_NONE, 46
  - RAMP\_RATE\_PROP, 46
  - SBUS\_CHANNEL\_LOWER, 46
  - SBUS\_CHANNEL\_UPPER, 46
  - STATUS\_UPDATE\_DELAY, 46
  - THROTTLE\_LIMIT, 47
- droneAllowedToFly
  - AS7::Drone, 11
- DroneCommandType
  - AS7, 7
- DroneDebugTest
  - as7-gamma.ino, 27
- DroneFlightMode
  - as7-gamma.ino, 27
- droneHasArmed
  - AS7::Drone, 11
- DroneState
  - as7-gamma.ino, 28
- droneStateMap
  - as7-gamma.ino, 34
- droneStatus
  - AS7::Drone, 11
- duration
  - AS7::DroneCommand, 16
- emergencyStop
  - AS7::Drone, 11
- enable\_pilotSemaphore
  - as7-gamma.ino, 34
- enable\_scribeSemaphore
  - as7-gamma.ino, 34
- enable\_usSemaphore
  - as7-gamma.ino, 34
- enableOperatorControl
  - AS7::Drone, 11
- enqueueCommand
  - AS7::Drone, 11
- error
  - AS7::Logger, 19
- estDronePos\_x
  - as7-gamma.ino, 35
- estDronePos\_y
  - as7-gamma.ino, 35
- estDronePos\_z
  - as7-gamma.ino, 35
- fatal
  - AS7::Logger, 19
- Faulted
  - as7-gamma.ino, 28
- filt\_pos\_x
  - as7-gamma.ino, 35
- filt\_pos\_y
  - as7-gamma.ino, 35
- filt\_pos\_z
  - as7-gamma.ino, 35
- filt\_vel\_x
  - as7-gamma.ino, 35
- filt\_vel\_y
  - as7-gamma.ino, 35
- filt\_vel\_z
  - as7-gamma.ino, 36
- FL\_LED
  - as7-gamma.ino, 36
- FL\_LEDBRIGHTNESS

- as7-gamma.ino, 36
- FL\_LEDNUM
  - as7-gamma.ino, 36
- FL\_LEDPIN
  - as7-gamma.ino, 36
- Flying
  - as7-gamma.ino, 28
- generateEStopTx
  - AS7::Drone, 11
- getCompassHeading
  - AS7::Drone, 12
- getDroneArmComplete
  - AS7::Drone, 12
- getEnableEmergencyStop
  - AS7::Drone, 12
- getEnableOperatorControl
  - AS7::Drone, 12
- getEStopTx
  - AS7::Drone, 12
- getSbusRxArray
  - AS7::Drone, 12
- getSbusTxArray
  - AS7::Drone, 12
- getUsBack
  - AS7::Drone, 12
- getUsDown
  - AS7::Drone, 13
- getUsFront
  - AS7::Drone, 13
- getUsLeft
  - AS7::Drone, 13
- getUsRight
  - AS7::Drone, 13
- getUsUp
  - AS7::Drone, 13
- GRAVITY\_CMS
  - as7-gamma.ino, 26
- Guided
  - AS7, 7
- H:/repos/Argous/src/esp32/as7-gamma/as7-gamma.ino, 23
- H:/repos/Argous/src/esp32/as7-gamma/Drone.cpp, 43
- H:/repos/Argous/src/esp32/as7-gamma/Drone.h, 43, 47
- H:/repos/Argous/src/esp32/as7-gamma/SdLogger.cpp, 50
- H:/repos/Argous/src/esp32/as7-gamma/SdLogger.h, 51, 53
- I2C\_SCL
  - as7-gamma.ino, 26
- I2C\_SDA
  - as7-gamma.ino, 26
- inform
  - AS7::Logger, 20
- Initialise
  - as7-gamma.ino, 28
- Landing
  - AS7, 7
  - as7-gamma.ino, 28
- LED\_RESPONSE
  - as7-gamma.ino, 27
- LED\_TYPE
  - as7-gamma.ino, 27
- log\_file
  - as7-gamma.ino, 36
- LOG\_LEVEL\_ERROR
  - SdLogger.h, 52
- LOG\_LEVEL\_FATAL
  - SdLogger.h, 52
- LOG\_LEVEL\_INFORM
  - SdLogger.h, 52
- LOG\_LEVEL\_SILENT
  - SdLogger.h, 52
- LOG\_LEVEL\_VERBOSE
  - SdLogger.h, 52
- LOG\_LEVEL\_WARNING
  - SdLogger.h, 52
- Logger
  - AS7::Logger, 19
- logger
  - as7-gamma.ino, 36
- LOGGER\_FREQ
  - SdLogger.h, 52
- loop
  - as7-gamma.ino, 28
- MAX\_US\_DISTANCE
  - as7-gamma.ino, 36
- NAV\_FREQ
  - Drone.h, 46
- nextState
  - as7-gamma.ino, 37
- NUM\_ACCEL\_READINGS
  - as7-gamma.ino, 37
- NUM\_CH
  - Drone.h, 46
- NUM\_US\_POINTS
  - as7-gamma.ino, 37
- NUM\_US\_SENSORS
  - as7-gamma.ino, 37
- OperatorControl
  - as7-gamma.ino, 28
- output\_file
  - as7-gamma.ino, 37
- p\_pt
  - AS7::DroneCommand, 16
- p\_rl
  - AS7::DroneCommand, 17
- p\_x
  - AS7::DroneCommand, 17
- p\_y
  - AS7::DroneCommand, 17

p\_yw  
     AS7::DroneCommand, 17  
 p\_z  
     AS7::DroneCommand, 17  
 pause  
     AS7::Drone, 13  
     AS7::Logger, 20  
 plot  
     AS7::Logger, 20  
 PLOTTER\_ENABLE  
     SdLogger.h, 52  
 pushData  
     AS7::Logger, 20  
  
 RAMPRATE\_LINEAR  
     Drone.h, 46  
 RAMPRATE\_NONE  
     Drone.h, 46  
 RAMPRATE\_PROP  
     Drone.h, 46  
 rateMultiplier  
     AS7::DroneCommand, 17  
 raw\_pos\_x  
     as7-gamma.ino, 37  
 raw\_pos\_y  
     as7-gamma.ino, 37  
 raw\_pos\_z  
     as7-gamma.ino, 37  
 raw\_vel\_x  
     as7-gamma.ino, 38  
 raw\_vel\_y  
     as7-gamma.ino, 38  
 raw\_vel\_z  
     as7-gamma.ino, 38  
 readAccelerometer  
     as7-gamma.ino, 28  
 Ready  
     as7-gamma.ino, 28  
 recordData  
     AS7::Logger, 20  
 recordingEnabled  
     as7-gamma.ino, 38  
     AS7::Drone, 13  
 resetEmergencyStop  
     AS7::Drone, 13  
 resume  
     AS7::Drone, 14  
     AS7::Logger, 20  
 running  
     AS7::Logger, 20  
  
 SBUS\_CHANNEL\_LOWER  
     Drone.h, 46  
 SBUS\_CHANNEL\_UPPER  
     Drone.h, 46  
 SBUS\_COMMS  
     as7-gamma.ino, 27  
 SBUS\_RXPIN  
     as7-gamma.ino, 38  
  
 SBUS\_TXPIN  
     as7-gamma.ino, 38  
 sbusRxData  
     as7-gamma.ino, 38  
 sbusTxData  
     as7-gamma.ino, 38  
 scribe\_logSemaphore  
     as7-gamma.ino, 39  
 scribe\_logStackMutex  
     as7-gamma.ino, 39  
 scribe\_msgSemaphore  
     as7-gamma.ino, 39  
 scribe\_msgStackMutex  
     as7-gamma.ino, 39  
 SD\_DISABLED  
     SdLogger.h, 53  
 SD\_READ\_WRITE  
     as7-gamma.ino, 27  
 SdLogger.h  
     AS7SDLOGGING\_H, 51  
     CS\_PIN, 51  
     LOG\_LEVEL\_ERROR, 52  
     LOG\_LEVEL\_FATAL, 52  
     LOG\_LEVEL\_INFORM, 52  
     LOG\_LEVEL\_SILENT, 52  
     LOG\_LEVEL\_VERBOSE, 52  
     LOG\_LEVEL\_WARNING, 52  
     LOGGER\_FREQ, 52  
     PLOTTER\_ENABLE, 52  
     SD\_DISABLED, 53  
 Serial  
     as7-gamma.ino, 39  
 Serial1  
     as7-gamma.ino, 39  
 setDataGathering  
     AS7::Drone, 14  
 setup  
     as7-gamma.ino, 29  
 setVerbosity  
     AS7::Logger, 21  
 SIMULATION\_ENABLE  
     as7-gamma.ino, 27  
 start  
     AS7::Drone, 14  
     AS7::Logger, 21  
 STATUS\_FASTLED\_PIN  
     as7-gamma.ino, 39  
 STATUS\_LED\_BRIGHTNESS  
     as7-gamma.ino, 39  
 STATUS\_NUM\_LEDS  
     as7-gamma.ino, 40  
 STATUS\_UPDATE\_DELAY  
     Drone.h, 46  
 Stopped  
     as7-gamma.ino, 28  
  
 taskAccelerometer  
     as7-gamma.ino, 29  
 taskStatusLeds

- as7-gamma.ino, [29](#)
- taskUltrasonicSensor
  - as7-gamma.ino, [29](#)
- TEMP\_PIN
  - as7-gamma.ino, [40](#)
- th\_Accel
  - as7-gamma.ino, [40](#)
- th\_Comms
  - as7-gamma.ino, [40](#)
- th\_Switch
  - as7-gamma.ino, [40](#)
- th\_Ultrasonic
  - as7-gamma.ino, [40](#)
- THROTTLE\_LIMIT
  - Drone.h, [47](#)
- TICK\_LONG
  - as7-gamma.ino, [40](#)
- TICK\_LONGLONG
  - as7-gamma.ino, [40](#)
- TICK\_MEDIUM
  - as7-gamma.ino, [41](#)
- TICK\_SHORT
  - as7-gamma.ino, [41](#)
- TICK\_US\_DELAY
  - as7-gamma.ino, [41](#)
- tmp\_temperature
  - as7-gamma.ino, [41](#)
- type
  - AS7::DroneCommand, [17](#)
- us\_distance
  - as7-gamma.ino, [41](#)
- US\_ECHOPIN
  - as7-gamma.ino, [41](#)
- US\_ECHOPIN\_0
  - as7-gamma.ino, [41](#)
- US\_ECHOPIN\_1
  - as7-gamma.ino, [41](#)
- US\_ECHOPIN\_2
  - as7-gamma.ino, [42](#)
- US\_ECHOPIN\_3
  - as7-gamma.ino, [42](#)
- US\_ECHOPIN\_4
  - as7-gamma.ino, [42](#)
- US\_ECHOPIN\_5
  - as7-gamma.ino, [42](#)
- us\_rawDistance
  - as7-gamma.ino, [42](#)
- US\_TRIGPIN
  - as7-gamma.ino, [42](#)
- US\_TRIGPIN\_0
  - as7-gamma.ino, [42](#)
- US\_TRIGPIN\_1
  - as7-gamma.ino, [42](#)
- US\_TRIGPIN\_2
  - as7-gamma.ino, [43](#)
- US\_TRIGPIN\_3
  - as7-gamma.ino, [43](#)
- US\_TRIGPIN\_4
  - as7-gamma.ino, [43](#)
- US\_TRIGPIN\_5
  - as7-gamma.ino, [43](#)
- usBack
  - AS7::Drone, [14](#)
- usDown
  - AS7::Drone, [14](#)
- usFront
  - AS7::Drone, [15](#)
- usLeft
  - AS7::Drone, [15](#)
- usRight
  - AS7::Drone, [15](#)
- usUp
  - AS7::Drone, [15](#)
- v\_pt
  - AS7::DroneCommand, [17](#)
- v\_rl
  - AS7::DroneCommand, [18](#)
- v\_x
  - AS7::DroneCommand, [18](#)
- v\_y
  - AS7::DroneCommand, [18](#)
- v\_yw
  - AS7::DroneCommand, [18](#)
- v\_z
  - AS7::DroneCommand, [18](#)
- verbose
  - AS7::Logger, [21](#)
- verbosity
  - AS7::Logger, [21](#)
- warn
  - AS7::Logger, [21](#)