SeaSense

Generated by Doxygen 1.8.11

Contents

1 SeaSense Arduino Library **1**

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 CLI_CMD Struct Reference 7

4.1.1 Detailed Description 7

4.2 SeaSense Class Reference 7

4.2.1 Detailed Description 8

4.2.2 Constructor & Destructor Documentation 8

4.2.2.1 SeaSense() 8

4.2.3 Member Function Documentation 8

4.2.3.1 BluetoothClient() 8

4.2.3.2 CollectData() 8

4.2.3.3 Initialize() 8

4.2.3.4 ReadAnalogPin(int pin) 9

5 File Documentation **11**

5.1 Cli.cpp File Reference . . . 11

5.1.1 Detailed Description . . . 12

5.1.2 Macro Definition Documentation . . . 13

5.1.2.1 CLI_CMD . . . 13

5.1.2.2 CLI_CMD . . . 13

5.1.2.3 CLI_CORE_CMD_LIST . . . 13

5.1.3 Function Documentation . . . 13

5.1.3.1 cli_help(int argc, char ∗argv[]) . . . 13

5.1.3.2 cli_log_app(int argc, char ∗argv[]) . . . 14

5.1.3.3 cli_log_data(int argc, char ∗argv[]) . . . 14

5.1.3.4 cli_log_file(int argc, char ∗argv[]) . . . 14

5.1.3.5 cli_rtc_get(int argc, char ∗argv[]) . . . 14

5.1.3.6 cli_rtc_set(int argc, char ∗argv[]) . . . 15

5.1.3.7 cli_sd_append(int argc, char ∗argv[]) . . . 15

5.1.3.8 cli_sd_cat(int argc, char ∗argv[]) . . . 15

5.1.3.9 cli_sd_create(int argc, char ∗argv[]) . . . 15

5.1.3.10 cli_sd_dd(int argc, char ∗argv[]) . . . 16

5.1.3.11 cli_sd_del(int argc, char ∗argv[]) . . . 16

5.1.3.12 cli_sd_init(int argc, char ∗argv[]) . . . 16

5.1.3.13 cli_sd_ls(int argc, char ∗argv[]) . . . 16

5.1.3.14 cli_test(int argc, char ∗argv[]) . . . 17

5.1.3.15 cli_wdt_reset(int argc, char ∗argv[]) . . . 17

5.1.3.16 dangerzone() . . . 17

5.1.3.17 dumpCSV(File dir, int numTabs) . . . 17

5.1.3.18 dumpCSVinfo(File dir, int numTabs, char ∗dirName) . . . 18

5.1.3.19 isCSV(char ∗filename) . . . 18

5.1.3.20 newFile(int filenum, char ∗directory) . . . 18

5.1.3.21 numCSVfiles(File dir, int numTabs) . . . 19
5.1.3.22 printDirectory(File dir, int numTabs) . . . 19
5.1.3.23 processCMD(char *command, int size) . . . 19
5.1.3.24 rmSubFiles(File dir) . . . 20
5.1.4 Variable Documentation . . . 20
5.1.4.1 cli_cmds . . . 20
5.2 Cli.h File Reference . . . 20
5.2.1 Detailed Description . . . 21
5.2.2 Function Documentation . . . 21
5.2.2.1 processCMD(char *command, int size) . . . 21
5.3 dataCollection.cpp File Reference . . . 22
5.3.1 Detailed Description . . . 23
5.3.2 Function Documentation . . . 23
5.3.2.1 getAccel() . . . 23
5.3.2.2 getADCreadings() . . . 23
5.3.2.3 getGyro() . . . 23
5.3.2.4 getInternals() . . . 23
5.3.2.5 getLight() . . . 24
5.3.2.6 getMag() . . . 24
5.3.2.7 getTime() . . . 24
5.3.2.8 resetADC() . . . 24
5.4 dataCollection.h File Reference . . . 25
5.4.1 Detailed Description . . . 25
5.4.2 Function Documentation . . . 26
5.4.2.1 getAccel() . . . 26
5.4.2.2 getADCreadings() . . . 26
5.4.2.3 getGyro() . . . 26
5.4.2.4 getInternals() . . . 26

5.4.2.5 getLight() . . . 26
5.4.2.6 getMag() . . . 26
5.4.2.7 getTime() . . . 27
5.5 display.h File Reference . . . 27
5.5.1 Detailed Description . . . 28
5.5.2 Function Documentation . . . 28
5.5.2.1 drawArrow(int degrees) . . . 28
5.5.2.2 drawBatInd() . . . 28
5.6 globals.h File Reference . . . 28
5.6.1 Detailed Description . . . 30
5.6.2 Macro Definition Documentation . . . 31
5.6.2.1 _TP . . . 31
5.6.2.2 ADC_BUFFER_SIZE . . . 31
5.6.2.3 BT_BAUDRATE . . . 31
5.6.2.4 BT_PWR . . . 31
5.6.2.5 DECLINATION_ANGLE . . . 31
5.6.2.6 LEDpin . . . 31
5.6.2.7 LOW_PWR_DEPTH . . . 31
5.6.2.8 LPM_WAKE . . . 31
5.6.2.9 MAX_CLI_ARGV . . . 31
5.6.2.10 MAX_INPUT_SIZE . . . 32
5.6.2.11 NUM_ADC_CHANNELS . . . 32
5.6.2.12 SD_CS . . . 32
5.6.3 Variable Documentation . . . 32
5.6.3.1 AccelX . . . 32
5.6.3.2 adc_ready . . . 32
5.6.3.3 adcBuf . . . 32
5.6.3.4 app_logData . . . 32

5.6.3.5 Cond . . . 33
5.6.3.6 Depth . . . 33
5.6.3.7 GyroX . . . 33
5.6.3.8 Head . . . 33
5.6.3.9 Light . . . 33
5.6.3.10 logData . . . 34
5.6.3.11 lowPowerLogging . . . 34
5.6.3.12 noSD . . . 34
5.6.3.13 PresInt . . . 34
5.6.3.14 RTC_AUTOSET . . . 34
5.6.3.15 sd_logData . . . 34
5.6.3.16 SDfile . . . 34
5.6.3.17 Temp . . . 34
5.6.3.18 TempInt . . . 35
5.6.3.19 Timestamp . . . 35
5.6.3.20 vBat . . . 35
5.7 SeaSense.cpp File Reference . . . 35
5.7.1 Detailed Description . . . 37
5.7.2 Function Documentation . . . 37
5.7.2.1 ISR(TIMER1_COMPA_vect) . . . 37
5.7.2.2 ISR(TIMER5_OVF_vect) . . . 37
5.7.2.3 ISR(ADC_vect) . . . 37
5.7.2.4 lpmWake() . . . 38
5.7.2.5 printAppData() . . . 38
5.7.2.6 printFileData() . . . 38
5.7.2.7 printOLEDdata() . . . 38
5.7.2.8 printVerboseData() . . . 38
5.7.3 Variable Documentation . . . 38

5.7.3.1 AccelX . . . 38
5.7.3.2 adc_ready . . . 38
5.7.3.3 adcBuf . . . 39
5.7.3.4 app_logData . . . 39
5.7.3.5 Cond . . . 39
5.7.3.6 Depth . . . 39
5.7.3.7 GyroX . . . 39
5.7.3.8 Head . . . 39
5.7.3.9 Light . . . 40
5.7.3.10 logData . . . 40
5.7.3.11 lowPowerLogging . . . 40
5.7.3.12 noSD . . . 40
5.7.3.13 PresInt . . . 40
5.7.3.14 RTC_AUTOSET . . . 40
5.7.3.15 sd_logData . . . 40
5.7.3.16 SDfile . . . 40
5.7.3.17 Temp . . . 41
5.7.3.18 TempInt . . . 41
5.7.3.19 Timestamp . . . 41
5.7.3.20 vBat . . . 41
5.8 SeaSense.h File Reference . . . 41
5.8.1 Detailed Description . . . 42

Index **43**

Chapter 1

SeaSense Arduino Library

Created by Georges Gauthier - glgauthier@wpi.edu

SeaSense is an underwater sensor package created for educational outreach. The SeaSense package is capable of logging data from various sensors, as well as displaying and wirelessly transmitting gathered data via Bluetooth.

This library relies on some extremely useful functions contained in the following Adafruit libraries:

- RTClib
- Adafruit_Sensor
- Adafruit HMC5883L Magnetometer Driver
- Adafruit ADXL345 Accelerometer Driver
- Adafruit GFX Library
- Adafruit SSD1306 Library

The software license found in license.txt applies to the above libraries

For more detailed documentation on this library, please refer to refman.pdf

Chapter 2

Class Index

2.1 Class List

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

CLI_CMD 7
SeaSense 7

Chapter 3

File Index

3.1 File List

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

Cli.cpp 11
Cli.h 20
dataCollection.cpp 22
dataCollection.h 25
display.h 27
globals.h 28
SeaSense.cpp 35
SeaSense.h 41

Chapter 4

Class Documentation

4.1 CLI_CMD Struct Reference

Public Attributes

- char * **name**
- char * **description**
- void(* **cli_function**)(int argc, char *argv[])

4.1.1 Detailed Description

Format for CLI commands added to CLI_CORE_CMD_LIST

Parameters

name	Function name, as entered in the command line interface
description	Function description (shown when calling the help command)
cli_function	Local function in Cli.cpp corresponding to the given command name and description

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

- Cli.cpp

4.2 SeaSense Class Reference

```
#include <SeaSense.h>
```

Public Member Functions

- SeaSense ()
- void Initialize ()
- void BluetoothClient ()
- void CollectData ()
- int ReadAnalogPin (int pin)

4.2.1 Detailed Description

SeaSense Arduino library class.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 SeaSense::SeaSense ( )

Preliminary initialization function for the sensor suite. This function runs when the line "SeaSense <name>; is called in an arduino sketch.

4.2.3 Member Function Documentation

4.2.3.1 void SeaSense::BluetoothClient ( )

Reads in new characters from the bluetooth serial port and parses them as an input string upon detecting a carriage return. Tested working with PuTTY, Arduino's serial monitor, and the Android app

See also

processCMD(char *command, int size)

4.2.3.2 void SeaSense::CollectData ( )

Lowest priority code used for sending pre-gathered sensor data to a serial log or file and clearing the ADC for more interrupts. This code should be run from the main loop of your arduino sketch

4.2.3.3 void SeaSense::Initialize ( )

Main initialization function for the sensor suite.

- used to configure all I/O and ISRs.
  - initializes serial comms on the bluetooth port (serial1).
  - configures Timer5 for hardware edge counting (light sensor).
  - configures a 10Hz Timer1 interrupt (for writing data to serial/SD).
  - configures an ADC interrupt routine.
  - initializes the SD card and RTC.

4.2.3.4 int SeaSense::ReadAnalogPin (int *pin*)

Replacement for Arduino's analogRead so that analog pins can be read indipendently from the ADC ISR. This function will disable interrupts, store the current ADC register settings, read from the given pin, and then revert the ADC register settings and re-enable the ADC isr. THIS CODE IS EXPERIMENTAL AND NOT TESTED FULLY WORKING

- semi based on analogRead source code: http://garretlab.web.fc2.com/en/arduino/inside/arduino/wiri
_analog.c/analogRead.html

Parameters

pin	A given analog pin

Returns

A 10-bit integer corresponding to the analog voltage on the given pin.

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

- SeaSense.h
- SeaSense.cpp

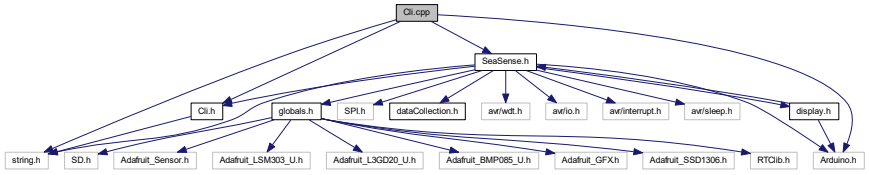
Chapter 5

File Documentation

5.1 Cli.cpp File Reference

```
#include "Arduino.h"
#include "Cli.h"
#include "SeaSense.h"
#include <string.h>
```

Include dependency graph for Cli.cpp:



Classes

- struct CLI_CMD

Macros

- #define CLI_CORE_CMD_LIST
- #define CLI_CMD(cmd, desc, func) void func(int argc, char ∗argv[ ]);
- #define CLI_CMD(cmd, desc, func) {cmd, desc, func},

Typedefs

- typedef struct CLI_CMD **cli_cmd_t**

Functions

- void printDirectory (File dir, int numTabs)
- char * newFile (int filenum, char *directory)
- void dumpCSV (File dir, int numTabs)
- bool isCSV (char *filename)
- void rmSubFiles (File dir)
- void numCSVfiles (File dir, int numTabs)
- void dumpCSVinfo (File dir, int numTabs, char *dirName)
- void dangerzone ()
- void processCMD (char *command, int size)
- void cli_help (int argc, char *argv[])
- void cli_test (int argc, char *argv[])
- void cli_rtc_get (int argc, char *argv[])
- void cli_rtc_set (int argc, char *argv[])
- void cli_sd_init (int argc, char *argv[])
- void cli_sd_ls (int argc, char *argv[])
- void cli_sd_cat (int argc, char *argv[])
- void cli_sd_dd (int argc, char *argv[])
- void cli_sd_append (int argc, char *argv[])
- void cli_sd_create (int argc, char *argv[])
- void cli_sd_del (int argc, char *argv[])
- void cli_log_data (int argc, char *argv[])
- void cli_log_app (int argc, char *argv[])
- void cli_log_file (int argc, char *argv[])
- void cli_wdt_reset (int argc, char *argv[])

Variables

- int **_numCSV** = 0
- CLI_CORE_CMD_LIST cli_cmd_t cli_cmds [ ]
- int **num_cli_cmds** = sizeof(cli_cmds)/sizeof(cli_cmds[0])

5.1.1 Detailed Description

Contains all functions for handling command line input

Author

Georges Gauthier, glgauthier@wpi.edu
Eugene Chabot (original CLI code for PIC24FJ64GB004)
John DiCecco (original CLI code for PIC24FJ64GB004)

Date

May-July 2016

5.1.2 Macro Definition Documentation

5.1.2.1 #define CLI_CMD(*cmd, desc, func*) void func(int argc, char ∗argv[ ]);

Generate function prototypes for all CLI_CMD structs in CLI_CORE_CMD_LIST

5.1.2.2 #define CLI_CMD(*cmd, desc, func*) {cmd, desc, func},

Generate function prototypes for all CLI_CMD structs in CLI_CORE_CMD_LIST

5.1.2.3 #define CLI_CORE_CMD_LIST

Value:

```
CLI_CMD("help", "Show a list of commands", cli_help) \
        CLI_CMD("test", "A test command", cli_test) \
        /* Realtime Clock commands. */ \
        CLI_CMD("rtc_get", "Get the RTC time", cli_rtc_get) \
        CLI_CMD("rtc_set", "Set the RTC time", cli_rtc_set) \
        /* SD card debugging commands. */ \
        CLI_CMD("sd_init", "Initialize the SD card", cli_sd_init) \
        CLI_CMD("sd_ls", "List all files on the SD card", cli_sd_ls) \
        CLI_CMD("sd_cat", "Dump a file", cli_sd_cat) \
        CLI_CMD("sd_dd", "Dump all .csv files", cli_sd_dd) \
        CLI_CMD("sd_append", "Append a comment to a file", cli_sd_append) \
        CLI_CMD("sd_create", "Create a file", cli_sd_create) \
        CLI_CMD("sd_del", "Delete a file or folder of files", cli_sd_del) \
        /* Data logging commands */ \
        CLI_CMD("log", "Log sensor data to command line.", cli_log_data) \
        CLI_CMD("logapp", "Log sensor data to the andriod app", cli_log_app) \
        CLI_CMD("logfile", "Log sensor data to file.", cli_log_file) \
        /* Misc. commands*/ \
        CLI_CMD("reset", "Reset the SeaSense (BE CAREFUL - KILLS ALL PROCESSES)",
      cli_wdt_reset) \
```

Generate a list of CLI_CMD structs for all commands visible from the help menu. Note that passing "text" for each command name and description is a depreciated conversion from a string constant to a char∗ and will result in compiler warnings. However, this method does save a lot of space and is more readable. In the future this may need to be fixed; proper syntax would be to use strcpy() or something similar.

5.1.3 Function Documentation

5.1.3.1 void cli_help (int *argc*, char ∗ *argv[ ]*)

Print out all available commands within CLI_CORE_CMD_LIST and their descriptions

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.2 void cli_log_app (int *argc,* char * *argv[ ]*)

Log data to the bluetooth serial port in a machine-recognizable format

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.3 void cli_log_data (int *argc,* char * *argv[ ]*)

Log data to the bluetooth serial port in a human-readable format

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.4 void cli_log_file (int *argc,* char * *argv[ ]*)

Create a new datafile on the SD card and begin logging data to it. Files recorded on the SD card will be saved in folders corresponding to the date of the recording. Filenames will correspond to YYMMDD##.CSV, where ## is the current number of files recorded on the corresponding date.

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

See also

newFile(int filenum, char* directory)

5.1.3.5 void cli_rtc_get (int *argc,* char * *argv[ ]*)

Print the current time

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.6 void cli_rtc_set (int *argc*, char * *argv[ ]*)

Set the RTC to a given time and date Input from command line should follow "rtc_set yyyy/mm/dd hh:mm:ss"

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

See also

RTC_AUTOSET

5.1.3.7 void cli_sd_append (int *argc*, char * *argv[ ]*)

Append a comment to a file

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing text to be written to file

5.1.3.8 void cli_sd_cat (int *argc*, char * *argv[ ]*)

Print out all data contained in a file. Proper syntax is sd_cat FILENAME.EXT or sd_cat FOLDERNAME/FILENAME.EXT

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.9 void cli_sd_create (int *argc*, char * *argv[ ]*)

Create a new file on the SD card

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.10 void cli_sd_dd (int *argc*, char * *argv[ ]*)

Dump all .csv files on the SD card

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

See also

dumpCSV(File dir, int numTabs)

5.1.3.11 void cli_sd_del (int *argc*, char * *argv[ ]*)

Delete a file or directory of files from the SD card

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

See also

rmSubFiles(File dir)

5.1.3.12 void cli_sd_init (int *argc*, char * *argv[ ]*)

Attempt to re-initialize the SD card. Note that the SD library doesn't have the ability to terminate an already-initialized SD card, so attempting an sd_init after already initializing the card will throw an error. I can't find a workaround to this :(

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.13 void cli_sd_ls (int *argc*, char * *argv[ ]*)

Print all files and directories saved on the SD card

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

See also

printDirectory(File dir, int numTabs)

5.1.3.14 void cli_test (int *argc,* char * *argv[ ]*)

Print a test string to illustrate the use of the command line client. Also prints each argv and its corresponding argc number to demonstrate input parsing

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.15 void cli_wdt_reset (int *argc,* char * *argv[ ]*)

Reset the microcontroller by enabling the watchdog timer and letting it overflow. Note that for the Seeeduino Mega version I've commented out the watchdog timer config and replaced it with a jump to the first address in program memory. Due to the bootloader on the 2560 of the seeeduino board, the micro will end up in a permanent WDT overflow loop if you enable the watchdog timer. This is NOT an issue on the blueduino board.

Parameters

argc	Number of space-separated arguments entered via the command line interface
argv	Character array containing argument vectors

5.1.3.16 void dangerzone ( )

Oh wait, I had something for this

5.1.3.17 void dumpCSV (File *dir,* int *numTabs*)

Recursively search through each dir on the SD card and dump the contents of all .CSV files contained within.

Parameters

dir	Top directory to begin searching within.
numTabs	Depth within the top directory to begin searching at

5.1.3.18 void dumpCSVinfo (File *dir,* int *numTabs,* char * *dirName*)

Recursively print the file size and name of each CSV file contained on the sd card Used for the android app.

Parameters

dir	Top directory to begin searching within.
numTabs	Depth within the top directory to begin searching at
dirName	name of the directory being searched (used in recursion to create a fullpath string for each filename)

See also

processCMD(char *command, int size)

5.1.3.19 bool isCSV (char * *filename*)

Check to see if a file is .csv format

Parameters

filename	8.3 Filename

Returns

true or false

5.1.3.20 char * newFile (int *filenum,* char * *directory*)

Scans SD card for a filename in the format of YYMMDDxx.csv contained within the given directory. This function will recursively search the directory until it hits a filename in sequence (xx = 00->99) that doesn't exist, and will return a full path to said filename.

Parameters

filenum	Sequential recording number
directory	File directory (8.3 name formatted as YYYYMMDD/)

Returns

Pointer to a new 8.3 filename containing YYMMDD and a number (00-99)

See also

cli_log_file(int argc, char ∗argv[ ])

5.1.3.21 void numCSVfiles (File *dir,* int *numTabs*)

Recursively print the number of CSV files contained on the sd card to the bluetooth serial port. Used for the android app.

Parameters

dir	Top directory to begin searching within.
numTabs	Depth within the top directory to begin searching at

See also

processCMD(char ∗command, int size)

5.1.3.22 void printDirectory (File *dir,* int *numTabs*)

Recursively print all files/folders/filesizes starting from the given directory.

Parameters

dir	Top directory of the filesystem being iterated through
numTabs	Depth to start searching the given dir at

5.1.3.23 void processCMD (char ∗ *command,* int *size*)

Parses an incoming command string into argv and argc, searches CLI_CMD_LIST for a corresponding command, and excecutes said command.

Parameters

command	Pointer to a character array
size	Number of characters in the array being pointed to

See also

SeaSense::BluetoothClient()
CLI_CORE_CMD_LIST

5.1.3.24 void rmSubFiles (File *dir*)

Delete all files within a directory. Created because SD.delete() doesn't work on a full dir.

Parameters

dir	Directory containing files

5.1.4 Variable Documentation

5.1.4.1 CLI_CORE_CMD_LIST cli_cmd_t cli_cmds[ ]

Initial value:

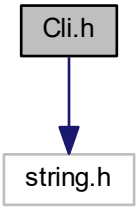
```
= {

#define CLI_CMD(cmd, desc, func)
    CLI_CORE_CMD_LIST
}
```

Array containing all CLI_CMD structs

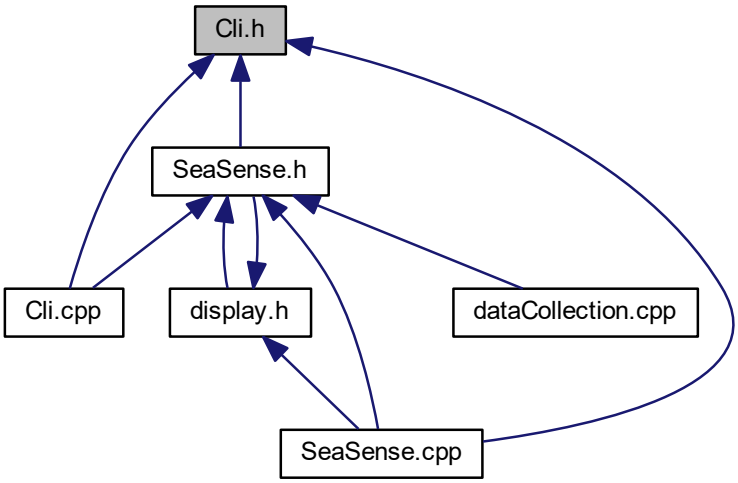
5.2 Cli.h File Reference

```
#include <string.h>
```

Include dependency graph for Cli.h:



This graph shows which files directly or indirectly include this file:



Functions

- void processCMD (char *command, int size)

5.2.1 Detailed Description

The cli.h file provides hooks for parsing command line inputs

Author

Georges Gauthier, glgauthier@wpi.edu

Date

May-July 2016

5.2.2 Function Documentation

5.2.2.1 void processCMD (char * *command,* int *size*)

Parses an incoming command string into argv and argc, searches CLI_CMD_LIST for a corresponding command, and excecutes said command.

Parameters

command	Pointer to a character array
size	Number of characters in the array being pointed to

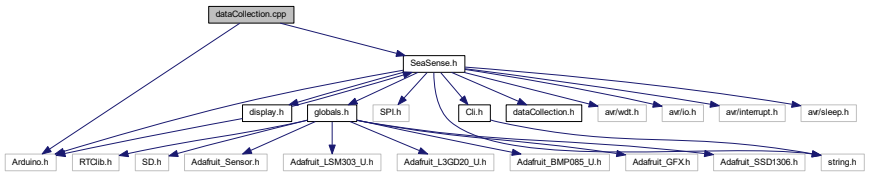
See also

SeaSense::BluetoothClient()
CLI_CORE_CMD_LIST

5.3 dataCollection.cpp File Reference

```
#include "Arduino.h"
#include "SeaSense.h"
```

Include dependency graph for dataCollection.cpp:



Functions

- void resetADC ()
- void getTime ()
- void getLight ()
- void getADCreadings ()
- void getMag ()
- void getAccel ()
- void getGyro ()
- void getInternals ()

Variables

- int **carryOut** = 0

5.3.1 Detailed Description

low-priority code for updating global variables with new sensor readings, as well as code for resetting the ADC for a new conversion sequence

Author

Georges Gauthier, glgauthier@wpi.edu

Date

May-July 2016

5.3.2 Function Documentation

5.3.2.1 void getAccel ( )

Gets the current accelerometer readings and stores them in globally accessible variables

5.3.2.2 void getADCreadings ( )

Average an ADC reading, perform any necessary conversions, store the converted value in the sensor variable corresponding to adc_channel, and reset the ADC registers for a new set of conversions on a different channel. Currently this function contains conversions for temperature, pressure, conductivity, and battery voltage.

See also

resetADC()

5.3.2.3 void getGyro ( )

Gets the current gyroscope readings and stored them in globally accessible variables

5.3.2.4 void getInternals ( )

Gets the current internal temperature and pressure and stores them in global variables. Note that this function will return on entry if _TP is #undef in globals.h

5.3.2.5 void getLight ( )

Put the current Light Sensor reading from the hardware pulse counter in global var Light.

See also

Light
SeaSense::Initialize()
ISR(TIMER5_OVF_vect)

5.3.2.6 void getMag ( )

Gets the current magnetic heading and converts it to a compass angle between 0 and 360 degrees, with the declination angle accounted for.

See also

DECLINATION_ANGLE

5.3.2.7 void getTime ( )

Put the current RTC timestamp into global var Timestamp.

See also

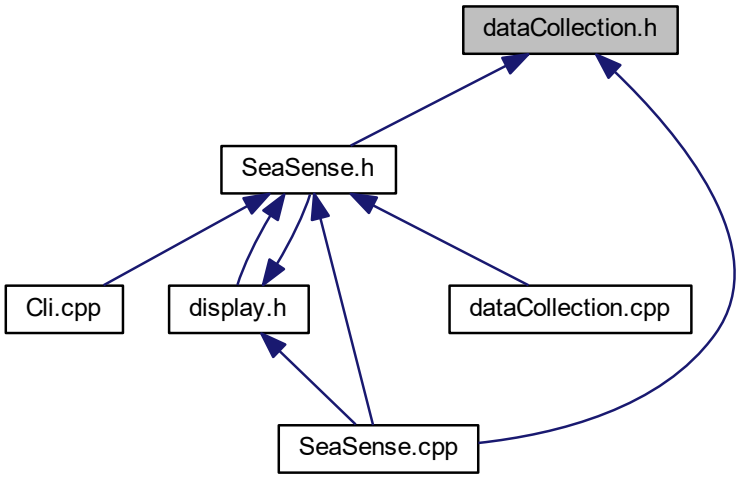
Timestamp

5.3.2.8 void resetADC ( )

This function resets the ADC for new conversions starting with channel 10 and buffer index 0

5.4 dataCollection.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void getTime ()
- void getLight ()
- void getADCreadings ()
- void getMag ()
- void getAccel ()
- void getGyro ()
- void getInternals ()

5.4.1 Detailed Description

Header containing prototypes for all functions related to collecting sensor data.

Author

Georges Gauthier, glgauthier@wpi.edu

Date

May-July 2016

5.4.2 Function Documentation

5.4.2.1 void getAccel ( )

Gets the current accelerometer readings and stores them in globally accessible variables

5.4.2.2 void getADCreadings ( )

Average an ADC reading, perform any necessary conversions, store the converted value in the sensor variable corresponding to adc_channel, and reset the ADC registers for a new set of conversions on a different channel. Currently this function contains conversions for temperature, pressure, conductivity, and battery voltage.

See also

resetADC()

5.4.2.3 void getGyro ( )

Gets the current gyroscope readings and stored them in globally accessible variables

5.4.2.4 void getInternals ( )

Gets the current internal temperature and pressure and stores them in global variables. Note that this function will return on entry if _TP is #undef in globals.h

5.4.2.5 void getLight ( )

Put the current Light Sensor reading from the hardware pulse counter in global var Light.

See also

Light
SeaSense::Initialize()
ISR(TIMER5_OVF_vect)

5.4.2.6 void getMag ( )

Gets the current magnetic heading and converts it to a compass angle between 0 and 360 degrees, with the declination angle accounted for.

See also

DECLINATION_ANGLE

5.4.2.7 void getTime ( )

Put the current RTC timestamp into global var Timestamp.

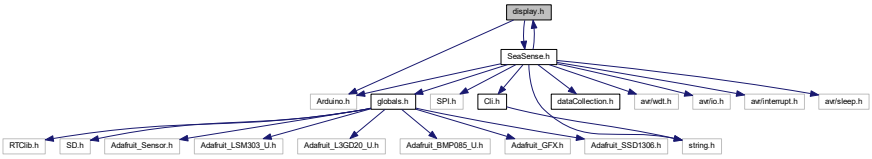
See also

Timestamp

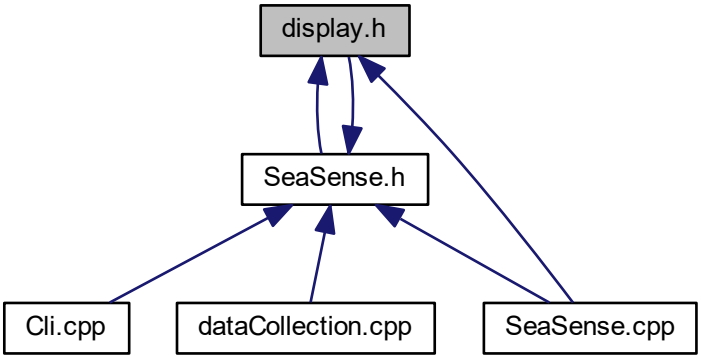
5.5 display.h File Reference

```
#include "Arduino.h"
#include "SeaSense.h"
```

Include dependency graph for display.h:



This graph shows which files directly or indirectly include this file:



Functions

- void drawArrow (int degrees)
- void drawBatInd ()

5.5.1 Detailed Description

Header containing data and function prototypes for display graphics.

Author

Georges Gauthier, glgauthier@wpi.edu

Date

May-July 2016

5.5.2 Function Documentation

5.5.2.1 void drawArrow (int *degrees*)

Draw an arrow on the display compass representing the current direction that the IMU faces

Parameters

degrees	Current compass heading in degrees.

Returns

Draws line on display.

5.5.2.2 void drawBatInd ( )

Populate the empty battery icon on the display with a bar representing how full the battery is Note that this function populates the battery indicator based on the value of global vBat.

See also

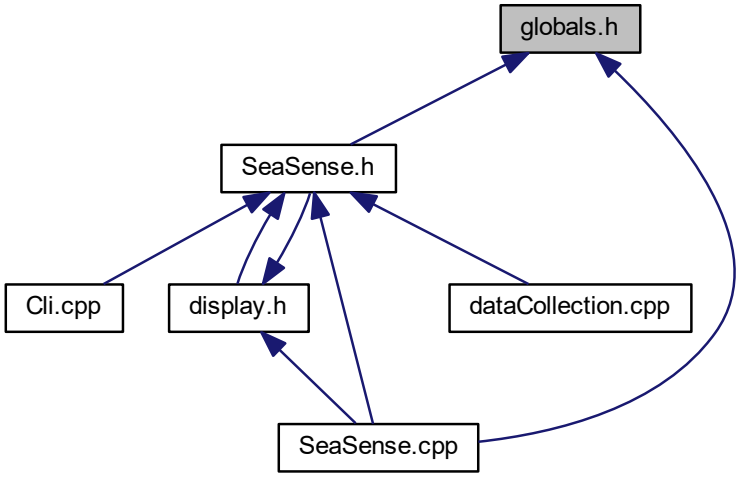
vBat

5.6 globals.h File Reference

```
#include "RTClib.h"
#include "SD.h"
#include "Adafruit_Sensor.h"
#include "Adafruit_LSM303_U.h"
#include "Adafruit_L3GD20_U.h"
#include "Adafruit_BMP085_U.h"
#include "Adafruit_GFX.h"
#include "Adafruit_SSD1306.h"
```

Include dependency graph for globals.h:

This graph shows which files directly or indirectly include this file:



Macros

- #define SD_CS 4
- #define _TP
- #define DECLINATION_ANGLE 0.2516f
- #define BT_PWR 38
- #define BT_BAUDRATE 115200
- #define MAX_INPUT_SIZE 80
- #define MAX_CLI_ARGV 10
- #define NUM_ADC_CHANNELS 4
- #define ADC_BUFFER_SIZE 100
- #define **OLED_MOSI** 6
- #define **OLED_CLK** 5
- #define **OLED_DC** 49
- #define **OLED_CS** 7
- #define **OLED_RESET** 48
- #define LEDpin 3
- #define LPM_WAKE 2
- #define LOW_PWR_DEPTH -5

Variables

- boolean RTC_AUTOSET
- RTC_DS1307 **rtc**
- File SDfile
- Adafruit_LSM303_Accel_Unified **accel**
- Adafruit_LSM303_Mag_Unified **mag**
- Adafruit_L3GD20_Unified **gyro**
- Adafruit_BMP085_Unified **bmp**
- float TempInt
- float PresInt
- Adafruit_SSD1306 **display**
- int adcBuf [ADC_BUFFER_SIZE]
- byte **adc_channel**
- byte **adc_pos**
- boolean adc_ready
- boolean lowPowerLogging
- boolean noSD
- boolean logData
- boolean sd_logData
- boolean app_logData
- char Timestamp [9]
- double Temp
- unsigned int Depth
- int Cond
- unsigned long Light
- int **carryOut**
- int Head
- float AccelX
- float **AccelY**
- float **AccelZ**
- float GyroX
- float **GyroY**
- float **GyroZ**
- int vBat

5.6.1 Detailed Description

Contains global variable storage and configuration.

Globals.h contains globally accessible variables that are used in multiple source files. These include, but are not limited to, configuration for various sensors and storage for processed sensor readings. Note that support for the legacy light sensor can be toggled in this file, as well as internal temperature and pressure readings, and the choice of a GY80 or Adafruit 9 or 10 DOF IMU.

Author

Georges Gauthier, glgauthier@wpi.edu

Date

May-July 2016

5.6.2 Macro Definition Documentation

5.6.2.1 #define _TP

Internal temp/pressure (optional, change to #undef to disable)

5.6.2.2 #define ADC_BUFFER_SIZE 100

Number of individual ADC samples to be taken from each channel

5.6.2.3 #define BT_BAUDRATE 115200

Baud rate of the BlueSmirf

5.6.2.4 #define BT_PWR 38

Pin connected to the 5V pin of the BlueSmirf. Using this configuration allows for the microcontroller to manually turn the module on/off to save power.

5.6.2.5 #define DECLINATION_ANGLE 0.2516f

http://www.magnetic-declination.com/

5.6.2.6 #define LEDpin 3

Indicator LED for data logging

5.6.2.7 #define LOW_PWR_DEPTH -5

Minimum depth required to enable low power logging mode (i.e. display and bluetooth are turned off).

5.6.2.8 #define LPM_WAKE 2

Low power mode wake pin. An external interrupt on this pin is configured to wake the atmega2560 from sleep mode. This allows for users to wake the arduino using the hall effect sensor

5.6.2.9 #define MAX_CLI_ARGV 10

CLI input argument limit (space separated)

5.6.2.10 #define MAX_INPUT_SIZE 80

CLI input buffer character limit

5.6.2.11 #define NUM_ADC_CHANNELS 4

Number of ADC channels to be read from by the sequential ISR setup. Note: channels 10:(NUM_ADC_CHANNELS-1) will be read into the ADC buffer by an ISR

5.6.2.12 #define SD_CS 4

SD card ChipSelect

5.6.3 Variable Documentation

5.6.3.1 float AccelX

Current accelerometer reading.

See also

getAccel()

5.6.3.2 boolean adc_ready

Indicator that adcBuf is full and ready to be read from

See also

ISR(ADC_vect)
getADCreadings()

5.6.3.3 int adcBuf[ADC_BUFFER_SIZE]

ADC sample buffer - fills from one ADC channel, is averaged, and then filled by the next ADC channel, etc, etc

5.6.3.4 boolean app_logData

Current status of data logging to the android app

5.6.3.5 int Cond

Current conductivity.

See also

getADCreadings()

5.6.3.6 unsigned int Depth

Current depth in cm.

See also

getADCreadings()

5.6.3.7 float GyroX

Current gyroscope reading.

See also

getGyro()

5.6.3.8 int Head

Current heading in degrees.

See also

getMag()

5.6.3.9 unsigned long Light

Current light sensor reading in lux.

See also

getLight()
SeaSense::Initialize()
ISR(TIMER5_OVF_vect)

5.6.3.10 boolean logData

Current status of verbose data logging to the Bluetooth port

5.6.3.11 boolean lowPowerLogging

Current status of low power logging (true=enabled)

5.6.3.12 boolean noSD

Current status of SD card slot (true = no SD card found)

5.6.3.13 float PresInt

Current reading from internal pressure sensor

5.6.3.14 boolean RTC_AUTOSET

Set to true to set RTC time to compile time. Set to false to allow for manual configuration of the RTC using the rtc_set command.

5.6.3.15 boolean sd_logData

Current status of data logging to the SD card

5.6.3.16 File SDfile

Current file being logged to

5.6.3.17 double Temp

Current temperature in deg C.

See also

getADCreadings()

5.6.3.18 float TempInt

Current reading from internal temperature sensor

5.6.3.19 char Timestamp[9]

Character array containing the current time, updated once per second

5.6.3.20 int vBat

Current battery voltage as read through the ADC.

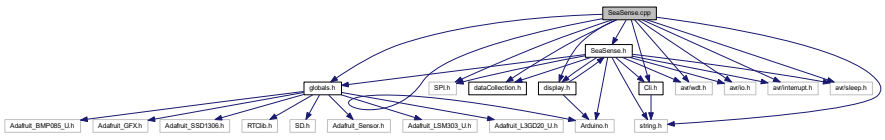
See also

getADCreadings()
drawBatInd()

5.7 SeaSense.cpp File Reference

```
#include "Arduino.h"
#include "globals.h"
#include "SeaSense.h"
#include "SPI.h"
#include "Cli.h"
#include "dataCollection.h"
#include "display.h"
#include "avr/wdt.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <string.h>
```

Include dependency graph for SeaSense.cpp:



Macros

- #define **_UNSIGNED**(X) ((X) + 32768)

Functions

- void printVerboseData ()
- void printAppData ()
- void printFileData ()
- void printOLEDdata ()
- void lpmWake ()
- Adafruit_SSD1306 **display** (OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS)
- ISR (TIMER1_COMPA_vect)
- ISR (TIMER5_OVF_vect)
- ISR (ADC_vect)

Variables

- boolean RTC_AUTOSET
- boolean logData
- boolean sd_logData
- boolean app_logData
- boolean noSD
- boolean adc_ready
- boolean lowPowerLogging
- char **cli_rxBuf** [MAX_INPUT_SIZE]
- byte **adc_channel** = 10
- int adcBuf [ADC_BUFFER_SIZE]
- byte **adc_pos** = 0
- int **_count**
- char Timestamp [9]
- double Temp = 0.0
- unsigned int Depth = 0
- int Cond = 0
- unsigned long Light = 0
- int Head = 0
- float AccelX = 0
- float **AccelY** = 0
- float **AccelZ** = 0
- float GyroX = 0
- float **GyroY** = 0
- float **GyroZ** = 0
- int vBat
- Sd2Card **card**
- File SDfile
- RTC_DS1307 **rtc**
- Adafruit_LSM303_Accel_Unified **accel**
- Adafruit_LSM303_Mag_Unified **mag**
- Adafruit_L3GD20_Unified **gyro**
- Adafruit_BMP085_Unified **bmp**
- float TempInt
- float PresInt

5.7.1 Detailed Description

File containing all main source code for the SeaSense Arduino lib. All externally accessible library functions are contained within this file, along with all register configurations and ISRs.

Author

Georges Gauthier, glgauthier@wpi.edu

Date

May-July 2016

5.7.2 Function Documentation

5.7.2.1 ISR (TIMER1_COMPA_vect)

Timer interrupt called once every 100mS

- checks to see if any type of serial or file logging is enabled and acts accordingly.

See also

SeaSense::Initialize()

5.7.2.2 ISR (TIMER5_OVF_vect)

Timer overflow for edge count interrupt (used with light sensor). This ISR will occur if the hardware edge counter doesn't detect any new rising edges before TIMER5 overflows

5.7.2.3 ISR (ADC_vect)

ADC hardware interrupt routine See the following for some useful information:

- https://bennthomsen.wordpress.com/arduino/peripherals/analogue-input/
- http://www.avrfreaks.net/forum/sampling-multiple-adc-channels
- Users' guide pg 283

The ADC is configured to run until the adc buffer has been filled based on ADC_BUFFER_SIZE. The ADC data is processed, and the ADC ISR is then re-enabled externally for a different ADC channel as specified in dataCollection.cpp. Note that to add additional ADC channels to read from, you need to modify the globals for the number of ADCs, as well as the getADCreadings() and resetADC() functions in dataCollection.cpp

5.7.2.4 void lpmWake ( )

Called automatically when waking back up from low power mode. NOTE that this *function* is called multiple times on wake, so there isn't any internal way to run a process a single time unless you use globals

5.7.2.5 void printAppData ( )

Print sensor readings to the android app

5.7.2.6 void printFileData ( )

Print sensor readings to a file on the SD card

5.7.2.7 void printOLEDdata ( )

Update content shown on the OLED display

5.7.2.8 void printVerboseData ( )

print user-readable sensor data to the bluetooth port

5.7.3 Variable Documentation

5.7.3.1 float AccelX = 0

Current accelerometer reading.

See also

getAccel()

5.7.3.2 boolean adc_ready

Indicator that adcBuf is full and ready to be read from

See also

ISR(ADC_vect)
getADCreadings()

5.7.3.3 int adcBuf[ADC_BUFFER_SIZE]

ADC sample buffer - fills from one ADC channel, is averaged, and then filled by the next ADC channel, etc, etc

5.7.3.4 boolean app_logData

Current status of data logging to the android app

5.7.3.5 int Cond = 0

Current conductivity.

See also

getADCreadings()

5.7.3.6 unsigned int Depth = 0

Current depth in cm.

See also

getADCreadings()

5.7.3.7 float GyroX = 0

Current gyroscope reading.

See also

getGyro()

5.7.3.8 int Head = 0

Current heading in degrees.

See also

getMag()

5.7.3.9 unsigned long Light = 0

Current light sensor reading in lux.

See also

getLight()
SeaSense::Initialize()
ISR(TIMER5_OVF_vect)

5.7.3.10 boolean logData

Current status of verbose data logging to the Bluetooth port

5.7.3.11 boolean lowPowerLogging

Current status of low power logging (true=enabled)

5.7.3.12 boolean noSD

Current status of SD card slot (true = no SD card found)

5.7.3.13 float PresInt

Current reading from internal pressure sensor

5.7.3.14 boolean RTC_AUTOSET

Set to true to set RTC time to compile time. Set to false to allow for manual configuration of the RTC using the rtc_set command.

5.7.3.15 boolean sd_logData

Current status of data logging to the SD card

5.7.3.16 File SDfile

Current file being logged to

5.7.3.17 double Temp = 0.0

Current temperature in deg C.

See also

getADCreadings()

5.7.3.18 float TempInt

Current reading from internal temperature sensor

5.7.3.19 char Timestamp[9]

Character array containing the current time, updated once per second

5.7.3.20 int vBat

Current battery voltage as read through the ADC.

See also

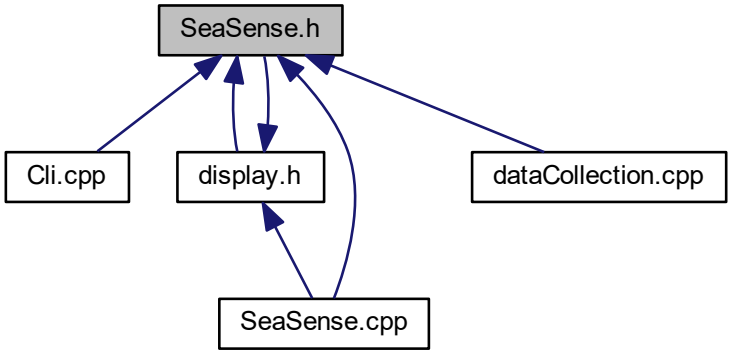
getADCreadings()
drawBatInd()

5.8 SeaSense.h File Reference

```
#include "Arduino.h"
#include "globals.h"
#include "SPI.h"
#include "Cli.h"
#include "dataCollection.h"
#include "display.h"
#include "avr/wdt.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <string.h>
```

Include dependency graph for SeaSense.h:

This graph shows which files directly or indirectly include this file:



Classes

- class SeaSense

5.8.1 Detailed Description

Header for file containing all main source code for the SeaSense Arduino lib. All externally accessible library functions are contained within this file, along with all register configurations and ISRs.

Author

Georges Gauthier, glgauthier@wpi.edu

Date

May-July 2016

Index

_TP
    globals.h, 31

ADC_BUFFER_SIZE
    globals.h, 31
AccelX
    globals.h, 32
    SeaSense.cpp, 38
adc_ready
    globals.h, 32
    SeaSense.cpp, 38
adcBuf
    globals.h, 32
    SeaSense.cpp, 38
app_logData
    globals.h, 32
    SeaSense.cpp, 39

BT_BAUDRATE
    globals.h, 31
BT_PWR
    globals.h, 31
BluetoothClient
    SeaSense, 8

CLI_CMD, 7
    Cli.cpp, 13
CLI_CORE_CMD_LIST
    Cli.cpp, 13
Cli.cpp, 11
    CLI_CMD, 13
    CLI_CORE_CMD_LIST, 13
    cli_cmds, 20
    cli_help, 13
    cli_log_app, 14
    cli_log_data, 14
    cli_log_file, 14
    cli_rtc_get, 14
    cli_rtc_set, 14
    cli_sd_append, 15
    cli_sd_cat, 15
    cli_sd_create, 15
    cli_sd_dd, 15
    cli_sd_del, 16
    cli_sd_init, 16
    cli_sd_ls, 16
    cli_test, 17
    cli_wdt_reset, 17
    dangerzone, 17
    dumpCSVinfo, 18
    dumpCSV, 17
    isCSV, 18
    newFile, 18
    numCSVfiles, 19
    printDirectory, 19
    processCMD, 19
    rmSubFiles, 20
Cli.h, 20
    processCMD, 21
cli_cmds
    Cli.cpp, 20
cli_help
    Cli.cpp, 13
cli_log_app
    Cli.cpp, 14
cli_log_data
    Cli.cpp, 14
cli_log_file
    Cli.cpp, 14
cli_rtc_get
    Cli.cpp, 14
cli_rtc_set
    Cli.cpp, 14
cli_sd_append
    Cli.cpp, 15
cli_sd_cat
    Cli.cpp, 15
cli_sd_create
    Cli.cpp, 15
cli_sd_dd
    Cli.cpp, 15
cli_sd_del
    Cli.cpp, 16
cli_sd_init
    Cli.cpp, 16
cli_sd_ls
    Cli.cpp, 16
cli_test
    Cli.cpp, 17
cli_wdt_reset
    Cli.cpp, 17

CollectData
SeaSense, 8
Cond
globals.h, 32
SeaSense.cpp, 39

DECLINATION_ANGLE
globals.h, 31
dangerzone
Cli.cpp, 17
dataCollection.cpp, 22
getADCreadings, 23
getAccel, 23
getGyro, 23
getInternals, 23
getLight, 23
getMag, 24
getTime, 24
resetADC, 24
dataCollection.h, 25
getADCreadings, 26
getAccel, 26
getGyro, 26
getInternals, 26
getLight, 26
getMag, 26
getTime, 26
Depth
globals.h, 33
SeaSense.cpp, 39
display.h, 27
drawArrow, 28
drawBatInd, 28
drawArrow
display.h, 28
drawBatInd
display.h, 28
dumpCSVinfo
Cli.cpp, 18
dumpCSV
Cli.cpp, 17

getADCreadings
dataCollection.cpp, 23
dataCollection.h, 26
getAccel
dataCollection.cpp, 23
dataCollection.h, 26
getGyro
dataCollection.cpp, 23
dataCollection.h, 26
getInternals
dataCollection.cpp, 23
dataCollection.h, 26
getLight
dataCollection.cpp, 23
dataCollection.h, 26
getMag
dataCollection.cpp, 24
dataCollection.h, 26
getTime
dataCollection.cpp, 24
dataCollection.h, 26
globals.h, 28
_TP, 31
ADC_BUFFER_SIZE, 31
AccelX, 32
adc_ready, 32
adcBuf, 32
app_logData, 32
BT_BAUDRATE, 31
BT_PWR, 31
Cond, 32
DECLINATION_ANGLE, 31
Depth, 33
GyroX, 33
Head, 33
LEDpin, 31
LOW_PWR_DEPTH, 31
LPM_WAKE, 31
Light, 33
logData, 33
lowPowerLogging, 34
MAX_CLI_ARGV, 31
MAX_INPUT_SIZE, 31
NUM_ADC_CHANNELS, 32
noSD, 34
PresInt, 34
RTC_AUTOSET, 34
SD_CS, 32
SDfile, 34
sd_logData, 34
Temp, 34
TempInt, 34
Timestamp, 35
vBat, 35
GyroX
globals.h, 33
SeaSense.cpp, 39

Head
globals.h, 33
SeaSense.cpp, 39

ISR
SeaSense.cpp, 37
Initialize
SeaSense, 8
isCSV
Cli.cpp, 18

LEDpin
globals.h, 31
LOW_PWR_DEPTH
globals.h, 31
LPM_WAKE
globals.h, 31
Light
globals.h, 33
SeaSense.cpp, 39
logData
globals.h, 33
SeaSense.cpp, 40
lowPowerLogging
globals.h, 34
SeaSense.cpp, 40
lpmWake
SeaSense.cpp, 37

MAX_CLI_ARGV
globals.h, 31
MAX_INPUT_SIZE
globals.h, 31

NUM_ADC_CHANNELS
globals.h, 32
newFile
Cli.cpp, 18
noSD
globals.h, 34
SeaSense.cpp, 40
numCSVfiles
Cli.cpp, 19

PresInt
globals.h, 34
SeaSense.cpp, 40
printAppData
SeaSense.cpp, 38
printDirectory
Cli.cpp, 19
printFileData
SeaSense.cpp, 38
printOLEDdata
SeaSense.cpp, 38
printVerboseData
SeaSense.cpp, 38
processCMD
Cli.cpp, 19
Cli.h, 21

RTC_AUTOSET
globals.h, 34
SeaSense.cpp, 40
ReadAnalogPin
SeaSense, 8
resetADC
dataCollection.cpp, 24
rmSubFiles
Cli.cpp, 20

SD_CS
globals.h, 32
SDfile
globals.h, 34
SeaSense.cpp, 40
sd_logData
globals.h, 34
SeaSense.cpp, 40
SeaSense, 7
BluetoothClient, 8
CollectData, 8
Initialize, 8
ReadAnalogPin, 8
SeaSense, 8
SeaSense.cpp, 35
AccelX, 38
adc_ready, 38
adcBuf, 38
app_logData, 39
Cond, 39
Depth, 39
GyroX, 39
Head, 39
ISR, 37
Light, 39
logData, 40
lowPowerLogging, 40
lpmWake, 37
noSD, 40
PresInt, 40
printAppData, 38
printFileData, 38
printOLEDdata, 38
printVerboseData, 38
RTC_AUTOSET, 40
SDfile, 40
sd_logData, 40
Temp, 40
TempInt, 41
Timestamp, 41
vBat, 41
SeaSense.h, 41

Temp
globals.h, 34
SeaSense.cpp, 40
TempInt
globals.h, 34
SeaSense.cpp, 41
Timestamp

globals.h, 35
SeaSense.cpp, 41

vBat
globals.h, 35
SeaSense.cpp, 41