

WayFindX

Version 1.0

Generated by Doxygen 1.10.0



<b>1 WayFindX</b>	<b>1</b>
1.1 Key Features	1
1.2 Hardware Components	1
1.3 Software Interface	2
1.4 Challenges Faced	2
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 File Documentation</b>	<b>5</b>
3.1 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ds/ds.c File Reference	5
3.1.1 Function Documentation	5
3.1.1.1 ds_clear()	5
3.1.1.2 ds_init()	5
3.1.1.3 ds_print_string()	6
3.2 ds.c	6
3.3 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ds/ds.h File Reference	6
3.3.1 Detailed Description	7
3.3.2 Macro Definition Documentation	7
3.3.2.1 MAX_COL	7
3.3.2.2 MAX_ROWS	7
3.3.2.3 SPACES	8
3.3.3 Function Documentation	8
3.3.3.1 ds_clear()	8
3.3.3.2 ds_init()	8
3.3.3.3 ds_print_string()	8
3.4 ds.h	9
3.5 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ir/ir.c File Reference	9
3.5.1 Function Documentation	9
3.5.1.1 ir_init()	9
3.5.1.2 ISR() [1/2]	10
3.5.1.3 ISR() [2/2]	10
3.5.2 Variable Documentation	10
3.5.2.1 ir_sec_counter	10
3.5.2.2 ir_trigger_1hz_flag_g	10
3.6 ir.c	10
3.7 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ir/ir.h File Reference	11
3.7.1 Detailed Description	12
3.7.2 Function Documentation	12
3.7.2.1 ir_init()	12
3.7.3 Variable Documentation	12
3.7.3.1 ir_sec_counter	12
3.7.3.2 ir_trigger_1hz_flag_g	12

3.8 ir.h	13
3.9 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/main.c File Reference	13
3.9.1 Macro Definition Documentation	13
3.9.1.1 F_CPU	13
3.9.2 Function Documentation	14
3.9.2.1 main()	14
3.9.2.2 startup()	14
3.9.2.3 task_1hz()	14
3.9.2.4 update_display()	14
3.10 main.c	15
3.11 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/nf/nf.c File Reference	17
3.11.1 Macro Definition Documentation	18
3.11.1.1 F_CPU	18
3.11.1.2 GGA_TYPE	19
3.11.1.3 GLL_TYPE	19
3.11.1.4 GSA_TYPE	19
3.11.1.5 GSV_TYPE	19
3.11.1.6 MSS_TYPE	19
3.11.1.7 NMEA_MSG_ID_SIZE	19
3.11.1.8 OK_TO_SEND_TYPE	19
3.11.1.9 RMC_TYPE	19
3.11.1.10 UART_BAUD_RATE	20
3.11.1.11 VTG_TYPE	20
3.11.1.12 ZDA_TYPE	20
3.11.2 Function Documentation	20
3.11.2.1 convertNMEAtoLLA()	20
3.11.2.2 get_serial_char()	20
3.11.2.3 nf_clear_nav_strings()	21
3.11.2.4 nf_init()	21
3.11.2.5 read_nmea_msg_raw()	21
3.11.3 Variable Documentation	21
3.11.3.1 altitudeLLA_float	21
3.11.3.2 altitudeLLA_str	21
3.11.3.3 ew_indicator	22
3.11.3.4 hdop	22
3.11.3.5 latitude	22
3.11.3.6 latitudeLLA_float	22
3.11.3.7 latitudeLLA_str	22
3.11.3.8 longitude	22
3.11.3.9 longitudeLLA_float	23
3.11.3.10 longitudeLLA_str	23
3.11.3.11 msl_altitude	23

3.11.3.12 ns_indicator . . . . .	23
3.11.3.13 position_fix_indicator . . . . .	23
3.11.3.14 satellites_used . . . . .	23
3.11.3.15 speed . . . . .	24
3.11.3.16 utc_time . . . . .	24
3.12 nf.c . . . . .	24
3.13 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/nf/nf.h File Reference . . . . .	28
3.13.1 Detailed Description . . . . .	29
3.13.2 Macro Definition Documentation . . . . .	29
3.13.2.1 NF_INIT_FAILURE . . . . .	29
3.13.2.2 NF_INIT_SUCCESS . . . . .	29
3.13.3 Function Documentation . . . . .	30
3.13.3.1 convertNMEAtoLLA() . . . . .	30
3.13.3.2 get_serial_char() . . . . .	30
3.13.3.3 nf_init() . . . . .	31
3.13.3.4 read_nmea_msg_raw() . . . . .	31
3.14 nf.h . . . . .	31
3.15 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/nf/nf_types.h File Reference . . . . .	32
3.15.1 Macro Definition Documentation . . . . .	32
3.15.1.1 GGA_ALTITUDE_BUFFER_SIZE . . . . .	32
3.15.1.2 GGA_HDOP_BUFFER_SIZE . . . . .	33
3.15.1.3 GGA_INDICATOR_SIZE . . . . .	33
3.15.1.4 GGA_LAT_BUFFER_SIZE . . . . .	33
3.15.1.5 GGA_LONG_BUFFER_SIZE . . . . .	33
3.15.1.6 GGA_NUM_INDICATORS_ACTIVE . . . . .	33
3.15.1.7 GGA_NUM_ITEMS_ACTIVE . . . . .	33
3.15.1.8 GGA_SIZE . . . . .	34
3.15.1.9 GGA_SV_USD_BUFFER_SIZE . . . . .	34
3.15.1.10 GGA_UTC_BUFFER_SIZE . . . . .	34
3.15.1.11 LLA_ALT_BUFFER_SIZE . . . . .	34
3.15.1.12 LLA_LAT_BUFFER_SIZE . . . . .	34
3.15.1.13 LLA_LONG_BUFFER_SIZE . . . . .	34
3.15.1.14 VTG_SPEED_BUFER_SIZE . . . . .	35
3.15.2 Variable Documentation . . . . .	35
3.15.2.1 altitudeLLA_float . . . . .	35
3.15.2.2 altitudeLLA_str . . . . .	35
3.15.2.3 ew_indicator . . . . .	35
3.15.2.4 hdop . . . . .	35
3.15.2.5 latitude . . . . .	35
3.15.2.6 latitudeLLA_float . . . . .	36
3.15.2.7 latitudeLLA_str . . . . .	36
3.15.2.8 longitude . . . . .	36

3.15.2.9 longitudeLLA_float . . . . .	36
3.15.2.10 longitudeLLA_str . . . . .	36
3.15.2.11 msl_altitude . . . . .	36
3.15.2.12 ns_indicator . . . . .	37
3.15.2.13 position_fix_indicator . . . . .	37
3.15.2.14 satellites_used . . . . .	37
3.15.2.15 speed . . . . .	37
3.15.2.16 utc_time . . . . .	37
3.16 nf_types.h . . . . .	38
3.17 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ut/ut_types.h File Reference . . . . .	38
3.17.1 Detailed Description . . . . .	39
3.17.2 Macro Definition Documentation . . . . .	39
3.17.2.1 false . . . . .	39
3.17.2.2 true . . . . .	39
3.17.3 Typedef Documentation . . . . .	39
3.17.3.1 boolean_t . . . . .	39
3.17.3.2 int8_t . . . . .	40
3.17.3.3 uint16_t . . . . .	40
3.17.3.4 uint8_t . . . . .	40
3.18 ut_types.h . . . . .	40
3.19 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ut/utilities.c File Reference . . . . .	40
3.19.1 Function Documentation . . . . .	41
3.19.1.1 is_button_pressed() . . . . .	41
3.19.1.2 ut_convert_lat_float_to_string() . . . . .	42
3.19.1.3 ut_convert_long_float_to_string() . . . . .	42
3.19.1.4 ut_init() . . . . .	42
3.19.1.5 ut_load_from_non_vol() . . . . .	43
3.19.1.6 ut_poll_btns() . . . . .	43
3.19.1.7 ut_write_to_non_vol() . . . . .	43
3.19.2 Variable Documentation . . . . .	44
3.19.2.1 ut_lat_mem_floats . . . . .	44
3.19.2.2 ut_lat_mem_str . . . . .	44
3.19.2.3 ut_long_mem_floats . . . . .	44
3.19.2.4 ut_long_mem_str . . . . .	44
3.19.2.5 ut_memory_Oidx . . . . .	44
3.19.2.6 ut_mode . . . . .	44
3.19.2.7 ut_operation . . . . .	45
3.20 utilities.c . . . . .	45
3.21 G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ut/utilities.h File Reference . . . . .	48
3.21.1 Detailed Description . . . . .	49
3.21.2 Macro Definition Documentation . . . . .	49
3.21.2.1 ACTION_BTN . . . . .	49

3.21.2.2 CLEAR_OP . . . . .	49
3.21.2.3 CLEAR_STR . . . . .	49
3.21.2.4 MAX_MEM_INDEX . . . . .	49
3.21.2.5 MEM_SELECT_BTN . . . . .	50
3.21.2.6 MODE_SELECT_BTN . . . . .	50
3.21.2.7 NAV_MODE . . . . .	50
3.21.2.8 NOP . . . . .	50
3.21.2.9 NUM_BUTTONS . . . . .	50
3.21.2.10 NUM_OPERATIONS . . . . .	50
3.21.2.11 ON_TIME_THRESHOLD . . . . .	51
3.21.2.12 OP_SELECT_BTN . . . . .	51
3.21.2.13 RESET_OP . . . . .	51
3.21.2.14 RESET_STR . . . . .	51
3.21.2.15 RESET_TIME_THRESHOLD . . . . .	51
3.21.2.16 SAVE_OP . . . . .	51
3.21.2.17 SAVE_STR . . . . .	52
3.21.2.18 STAT_MODE . . . . .	52
3.21.3 Function Documentation . . . . .	52
3.21.3.1 ut_init() . . . . .	52
3.21.3.2 ut_poll_btns() . . . . .	52
3.21.4 Variable Documentation . . . . .	53
3.21.4.1 ut_lat_mem_floats . . . . .	53
3.21.4.2 ut_lat_mem_str . . . . .	53
3.21.4.3 ut_long_mem_floats . . . . .	53
3.21.4.4 ut_long_mem_str . . . . .	53
3.21.4.5 ut_memory_Oidx . . . . .	53
3.21.4.6 ut_mode . . . . .	54
3.21.4.7 ut_operation . . . . .	54
3.22 utilities.h . . . . .	54

<b>Index</b>	<b>55</b>
--------------	-----------





# Chapter 1

## WayFindX

Synopsis of the WayFindX Embedded Systems Project.

Synopsis of the WayFindX Embedded Systems Project.

WayFindX is a handheld GPS receiver project developed by Avengers Assembly, consisting of Bradley Johnson and Abele Atresso. The project aims to create a portable navigation device that enhances outdoor navigation capabilities for users such as hikers, adventurers, and outdoor enthusiasts. WayFindX provides essential features including position and velocity tracking, storage of multiple locations, and distance calculation to selected stored positions.

### 1.1 Key Features

- Display of user position and velocity information
- Storage of multiple locations for navigation
- Calculation of distance to selected stored positions
- Intuitive user interface with control buttons and LCD display

### 1.2 Hardware Components

The project utilizes the following hardware components:

- Arduino microcontroller (ATMega328p)
- HD44780 Controlled 20x4 LCD for display
- Control Stick for user input
- NEO-6M GPS module for positioning
- Power Switch and 9V battery for power management
- Additional components for connectivity
- Custom-designed 3D printed case for enclosure

## 1.3 Software Interface

WayFindX provides a user-friendly interface with control buttons for navigation and interaction. The software is designed to display essential information such as current position, stored locations, and operational modes on the LCD display. Users can cycle through saved positions, select operations, and execute actions with ease.

## 1.4 Challenges Faced

The project encountered various challenges including serial communication, interfacing with new ICs, soldering, and enclosure design. Overcoming these challenges required careful planning, experimentation, and troubleshooting.

For detailed documentation and code implementation, refer to the project source code and comments.\

### Version

1.0

### Copyright

(C) 2024 Bradley Johnson and Abele Atresso

## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

G:/Project_Files/WayFindX/wfx_sw/wfx_sw/main.c . . . . .	13
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ds/ds.c . . . . .	5
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ds/ds.h	
Header file containing common functions and definitions for the display 'computer software component' or CSC . . . . .	6
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ir/ir.c . . . . .	9
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ir/ir.h	
Header file containing common functions and definitions for the interrupt system 'computer software component' or CSC . . . . .	11
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/nf/nf.c . . . . .	17
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/nf/nf.h	
Header file containing common functions and definitions for the navigation fetch (NV) CSC . . . . .	28
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/nf/nf_types.h . . . . .	32
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ut/ut_types.h	
Header file containing typedefs for common types . . . . .	38
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ut/utilities.c . . . . .	40
G:/Project_Files/WayFindX/wfx_sw/wfx_sw/ut/utilities.h	
Header file containing common utility functions and definitions . . . . .	48



## Chapter 3

# File Documentation

### 3.1 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ds/ds.c File Reference

```
#include "ds.h"
#include "../lib/lcd.h"
#include "../ut/ut_types.h"
#include <avr/io.h>
```

#### Functions

- void [ds\\_print\\_string](#) (char \*inputString, int size, [uint8\\_t](#) row)  
*Prints a string on the LCD display.*
- void [ds\\_init](#) ()  
*Initializes the LCD display. This function initializes the LCD display and turns on the display.*
- void [ds\\_clear](#) ()  
*Clears the LCD display. This function clears the LCD display.*

#### 3.1.1 Function Documentation

##### 3.1.1.1 [ds\\_clear\(\)](#)

```
void ds_clear ( )
```

Clears the LCD display. This function clears the LCD display.

Clears the display.

Definition at line [44](#) of file [ds.c](#).

##### 3.1.1.2 [ds\\_init\(\)](#)

```
void ds_init ( )
```

Initializes the LCD display. This function initializes the LCD display and turns on the display.

Uses library to initialize display. If DEBUG is defined, cursor will blink. Otherwise, cursor will be off.

Definition at line [33](#) of file [ds.c](#).

### 3.1.1.3 ds\_print\_string()

```
void ds_print_string (
    char * inputString,
    int size,
    uint8_t row )
```

Prints a string on the LCD display.

#### Parameters

<i>inputString</i>	Pointer to the string to be printed.
<i>size</i>	Size of the string to be printed.
<i>row</i>	indicating what row to place the cursor

Definition at line 14 of file [ds.c](#).

## 3.2 ds.c

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

```
00001 #include "ds.h"
00002 #include "../lib/lcd.h"
00003 #include "../ut/ut_types.h"
00004 #include <avr/io.h>
00005
00006
00014 void ds_print_string(char * inputString, int size, uint8_t row)
00015 {
00016     //Make sure size is legit
00017     if ((size >= 0) && (size <= MAX_COL) && (row >= 0) && (row < MAX_ROWS))
00018     {
00019         lcd_gotoxy((uint8_t)0, row); //send to second line
00020
00021         for (int i = 0; i < size; i++)
00022         {
00023             lcd_putc(inputString[i]); //put i'th char on display
00024         }
00025     }
00026     return;
00027 }
00028
00033 void ds_init()
00034 {
00035     lcd_init(LCD_DISP_ON);
00036     lcd_clrscr();
00037     return;
00038 }
00039
00044 void ds_clear(){
00045     lcd_clrscr();
00046 }
```

## 3.3 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ds/ds.h File Reference

Header file containing common functions and definitions for the display 'computer software component' or CSC.

```
#include "../ut/ut_types.h"
```

## Macros

- `#define MAX_ROWS 4`
- `#define MAX_COL 20`
- `#define SPACES " "`

## Functions

- void `ds_print_string` (char \*inputString, int size, `uint8_t` row)  
*Prints a string on the LCD display.*
- void `ds_init` ()  
*Uses library to initialize display. If DEBUG is defined, cursor will blink. Otherwise, cursor will be off.*
- void `ds_clear` ()  
*Clears the display.*

### 3.3.1 Detailed Description

Header file containing common functions and definitions for the display 'computer software component' or CSC.

This file provides declarations for common utility functions and definitions associated with the LCD display.

#### Version

1.0

#### Copyright

(C) 2024 Bradley Johnson and Abele Atresso

Definition in file [ds.h](#).

### 3.3.2 Macro Definition Documentation

#### 3.3.2.1 MAX\_COL

```
#define MAX_COL 20
```

Maximum number of columns on the LCD display

Definition at line 14 of file [ds.h](#).

#### 3.3.2.2 MAX\_ROWS

```
#define MAX_ROWS 4
```

Maximum number of rows on the LCD display

Definition at line 13 of file [ds.h](#).

### 3.3.2.3 SPACES

```
#define SPACES " "
```

String of spaces used for clearing the LCD display

Definition at line 15 of file [ds.h](#).

## 3.3.3 Function Documentation

### 3.3.3.1 ds\_clear()

```
void ds_clear ( )
```

Clears the display.

Clears the display.

Definition at line 44 of file [ds.c](#).

### 3.3.3.2 ds\_init()

```
void ds_init ( )
```

Uses library to initialize display. If DEBUG is defined, cursor will blink. Otherwise, cursor will be off.

#### Returns

none

Uses library to initialize display. If DEBUG is defined, cursor will blink. Otherwise, cursor will be off.

Definition at line 33 of file [ds.c](#).

### 3.3.3.3 ds\_print\_string()

```
void ds_print_string (
    char * inputString,
    int size,
    uint8_t row )
```

Prints a string on the LCD display.

#### Parameters

<i>inputString</i>	Pointer to the string to be printed.
<i>size</i>	Size of the string to be printed.
<i>row</i>	Row index where the cursor will be placed.
<i>inputString</i>	Pointer to the string to be printed.
<i>size</i>	Size of the string to be printed.
<i>row</i>	indicating what row to place the cursor



Definition at line 14 of file ds.c.

## 3.4 ds.h

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

```
00001
00010 #ifndef DS_H_
00011 #define DS_H_
00012
00013 #define MAX_ROWS 4
00014 #define MAX_COL 20
00015 #define SPACES "
00018 #include "../ut/ut_types.h"
00019
00027 void ds_print_string(char * inputString, int size, uint8_t row);
00028
00035 void ds_init();
00036
00040 void ds_clear();
00041
00042
00043 #endif /* DS_H_ */
```

## 3.5 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ir/ir.c File Reference

```
#include "ir.h"
#include <avr/interrupt.h>
#include "../ut/utilities.h"
```

### Functions

- [ISR](#) (TIMER0\_OVF\_vect)
- [ISR](#) (TIMER2\_OVF\_vect)
- void [ir\\_init](#) ()

*Initializes interrupt system functionality.*

### Variables

- [uint16\\_t ir\\_sec\\_counter](#) = 0
- [boolean\\_t ir\\_trigger\\_1hz\\_flag\\_g](#) = false

### 3.5.1 Function Documentation

#### 3.5.1.1 ir\_init()

```
void ir_init ( )
```

Initializes interrupt system functionality.

This function configures Timer2 for time management and Timer 0 for background button polling.

Definition at line 38 of file [ir.c](#).

### 3.5.1.2 ISR() [1/2]

```
ISR (
    TIMER0_OVF_vect )
```

Definition at line 16 of file [ir.c](#).

### 3.5.1.3 ISR() [2/2]

```
ISR (
    TIMER2_OVF_vect )
```

Definition at line 22 of file [ir.c](#).

## 3.5.2 Variable Documentation

### 3.5.2.1 ir\_sec\_counter

```
uint16_t ir_sec_counter = 0
```

Seconds counter for indicating TTFF

Definition at line 6 of file [ir.c](#).

### 3.5.2.2 ir\_trigger\_1hz\_flag\_g

```
boolean_t ir_trigger_1hz_flag_g = false
```

Global flag indicating 1Hz trigger

Definition at line 7 of file [ir.c](#).

## 3.6 ir.c

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

```
00001 #include "ir.h"
00002 #include <avr/interrupt.h>
00003 #include "../ut/utilities.h"
00004
00005 //global
00006 uint16_t ir_sec_counter = 0;
00007 boolean_t ir_trigger_1hz_flag_g = false;
00011 //local static
00012 static uint8_t timer2_overflow_counter;
00013
00014
00015 // Interrupt Service Routine for Timer0 overflow: Poll all four buttons in background set state if we
    have polled enough times
00016 ISR(TIMER0_OVF_vect) {
00017     ut_poll_btns();
00018     TCNT0 = 0;
00019 }
00020
00021 // Interrupt Service Routine for Timer2 overflow: Used to trip trigger flag at 1Hz
00022 ISR(TIMER2_OVF_vect) {
00023     timer2_overflow_counter++;
00024     if (timer2_overflow_counter >= 15){
```

```

00025         timer2_overflow_counter = 0;
00026         ir_sec_counter++;
00027         if (!ir_trigger_lhz_flag_g){
00028             ir_trigger_lhz_flag_g = true;
00029         }
00030     }
00031 }
00032
00038 void ir_init()
00039 {
00040     cli();
00041     //-----
00042     //timer 0 for background button polling
00043     OCR0A = 2;          // Set TOP (maximum value for counter)
00044
00045     TCNT0 = 0x00;       //set counter to zero
00046     TCCR0A = 0x00;      //set TCCR1A to zero before configuring timer0
00047     TCCR0B = 0x00;
00048     // TOP is defined as OCR0A when WGM2:0 = 5
00049
00050     // Set pre-scaler to /8
00051     TCCR0B |= (1 << CS01);
00052     // Enable Timer0 Overflow Interrupt
00053     TIMSK0 |= (1 << TOIE0);
00054     // Initialize Timer0 count
00055     TCNT0 = 0;
00056
00057
00058     //Set up timer2 for 1 hz task management
00059     // Set Timer2 in normal mode (WGM22:0 = 0)
00060     TCCR2A = 0x00;
00061     TCCR2B = 0x00;
00062     // Set pre-scaler to clk/1024 for smoothing
00063     TCCR2B |= (1 << CS22) | (1 << CS21) | (1 << CS20); // Pre-scaler = 64
00064     // Enable Timer2 Overflow Interrupt
00065     TIMSK2 |= (1 << TOIE2);
00066
00067
00068     sei(); // Enable interrupts.
00069 }

```

## 3.7 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ir/ir.h File Reference

Header file containing common functions and definitions for the interrupt system 'computer software component" or CSC.

```
#include "../ut/ut_types.h"
```

### Functions

- void [ir\\_init\(\)](#)  
*Initializes interrupt system functionality.*

### Variables

- [boolean\\_t ir\\_trigger\\_1hz\\_flag\\_g](#)
- [uint16\\_t ir\\_sec\\_counter](#)

### 3.7.1 Detailed Description

Header file containing common functions and definitions for the interrupt system "computer software component" or CSC.

This file provides declarations for common utility functions and definitions associated with the interrupt system.

#### Version

1.0

#### Copyright

(C) 2024 Bradley Johnson and Abele Atresso

Definition in file [ir.h](#).

### 3.7.2 Function Documentation

#### 3.7.2.1 ir\_init()

```
void ir_init ( )
```

Initializes interrupt system functionality.

This function configures Timer2 for time management and Timer 0 for background button polling.

Definition at line [38](#) of file [ir.c](#).

### 3.7.3 Variable Documentation

#### 3.7.3.1 ir\_sec\_counter

```
uint16_t ir_sec_counter [extern]
```

Seconds counter for indicating TTFF

Definition at line [6](#) of file [ir.c](#).

#### 3.7.3.2 ir\_trigger\_1hz\_flag\_g

```
boolean_t ir_trigger_1hz_flag_g [extern]
```

Global flag indicating 1Hz trigger

Definition at line [7](#) of file [ir.c](#).

## 3.8 ir.h

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

```
00001
00010 #ifndef IR_H_
00011 #define IR_H_
00012
00013 #include "../ut/ut_types.h"
00014
00015 //globals
00016 extern boolean_t ir_trigger_lhz_flag_g;
00017 extern uint16_t ir_sec_counter;
00024 void ir_init();
00025
00026
00027 #endif /* IR_H_ */
```

## 3.9 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/main.c File Reference

```
#include <avr/power.h>
#include <avr/io.h>
#include <util/delay.h>
#include "ds/ds.h"
#include "ir/ir.h"
#include "nf/nf.h"
#include "nf/nf_types.h"
#include "ut/utilities.h"
#include "ut/ut_types.h"
#include <string.h>
```

### Macros

- #define [F\\_CPU](#) 4000000UL

### Functions

- void [startup](#) ()  
*Performs startup initialization.*
- void [task\\_1hz](#) ()  
*Executes tasks that should occur every 1Hz.*
- void [update\\_display](#) ()  
*Updates the display with relevant information.*
- int [main](#) (void)  
*Main loop function.*

## 3.9.1 Macro Definition Documentation

### 3.9.1.1 F\_CPU

```
#define F_CPU 4000000UL
```

Define the CPU frequency to 4MHz.

Definition at line [46](#) of file [main.c](#).

## 3.9.2 Function Documentation

### 3.9.2.1 main()

```
int main (
    void )
```

Main loop function.

This function controls the program flow and will not return. It initializes the peripherals, clears the display, and continuously updates the system behavior.

Definition at line 73 of file [main.c](#).

### 3.9.2.2 startup()

```
void startup ( )
```

Performs startup initialization.

< Include display-related functions. < Include interrupt routines. < Include navigation fetch functions < Include utility functions. < Include common type definitions.

This function initializes various peripherals and components during startup, such as the display, interrupt routines, navigation fetch, and utilities. < Initialize display CSC.

< Initialize interrupt routines.

<Initialize navigation fetch CSC.

< Initialize utilities CSC.

Definition at line 93 of file [main.c](#).

### 3.9.2.3 task\_1hz()

```
void task_1hz ( )
```

Executes tasks that should occur every 1Hz.

This function is called once per second and performs tasks such as updating the display.

Definition at line 130 of file [main.c](#).

### 3.9.2.4 update\_display()

```
void update_display ( )
```

Updates the display with relevant information.

This function updates the display with information such as GPS coordinates, mode, operation, and memory status.

Definition at line 147 of file [main.c](#).

## 3.10 main.c

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

```

00001
00045 #ifndef F_CPU
00046 #define F_CPU 4000000UL
00047 #endif
00048
00049 //PORT Pin 2 PD2
00050 #include <avr/power.h>
00051 #include <avr/io.h>
00052 #include <util/delay.h>
00053
00054 #include "ds/ds.h"
00055 #include "ir/ir.h"
00056 #include "nf/nf.h"
00057 #include "nf/nf_types.h"
00058 #include "ut/utilities.h"
00059 #include "ut/ut_types.h"
00060 #include <string.h>
00061
00062 void startup();
00063 void task_lhz();
00064 void update_display();
00065
00066
00073 int main(void)
00074 {
00075     startup();
00076     /* Main loop */
00077     while (1){
00078         //read_nmea_msg
00079         read_nmea_msg_raw();
00080         if (ir_trigger_lhz_flag_g == true){
00081             task_lhz();
00082             ir_trigger_lhz_flag_g = false;
00083         }
00084     }
00085 }
00086
00087
00093 void startup(){
00094     //Initialize
00095     // Set clock pre-scaler to divide by 4
00096     clock_prescale_set(clock_div_4);
00097
00098     // Initialize computer software components (CSC's)
00099     ds_init();
00100     {
00101         char* welcome = "- - - - ~~~~ - - - -";
00102         ds_print_string(welcome, MAX_COL, 0);
00103     }
00104     { //Welcome Screen - limited scope
00105         char* welcome = "- - - WayFindX - - -";
00106         ds_print_string(welcome, MAX_COL, 1);
00107     }
00108     {
00109         char* welcome = "- - - - ~~~~ - - - -";
00110         ds_print_string(welcome, MAX_COL, 2);
00111     }
00112     _delay_ms(0.1f);
00113
00114     ir_init();
00115     _delay_ms(0.1f);
00116     if (nf_init()){
00117         char* err = " Nav init failure ";
00118         ds_print_string(err, MAX_COL, 1);
00119         while(1){};
00120     }
00121     _delay_ms(0.6f);
00122     ut_init();
00123 }
00124
00130 void task_lhz(){
00131     update_display();
00132     //Condition where USART if out of sync with NEO6-M
00133     if ((position_fix_indicator[0] == '1') && (speed[0] == ' ') && (speed[1] == ' ')){
00134         /* Re-Initialize navigation fetch CSC. */
00135         do {
00136             char* err = "Nav module off sync!";
00137             ds_print_string(err, MAX_COL, 1);
00138         } while(nf_init());
00139     }
00140 }
00141

```

```

00147 void update_display(){
00148     char line0[MAX_COL] = SPACES;
00149     char line1[MAX_COL] = SPACES;
00150     char line2[MAX_COL] = SPACES;
00151     char line3[MAX_COL] = SPACES;
00152
00153     if (utc_time[0] == ' '){ //Until we solve for time
00154         char* temp = "Acquiring Satellites";
00155         for (int i = 0; i < MAX_COL; i++){
00156             line3[i] = temp[i];
00157         }
00158     } else if (position_fix_indicator[0] != '1'){ //Display time once we solve for time
00159         line2[0] = 'U';
00160         line2[1] = 'T';
00161         line2[2] = 'C';
00162         line2[3] = ':';
00163         for (int i=0; i < GGA.UTC_BUFFER_SIZE; i++){
00164             line2[4+i] = utc_time[i];
00165         }
00166
00167         char* temp = "Getting PVT Solution";
00168         for (int i = 0; i < MAX_COL; i++){
00169             line3[i] = temp[i];
00170         }
00171
00172     } else if (position_fix_indicator[0] == '1') { //Once we get a fix, go into normal operation
00173         convertNMEAtoLLA();
00174
00175         //Mode agnostic parts
00176         //line0
00177         for (int i = 0; i < LLA_LAT_BUFFER_SIZE; i++){
00178             line0[i] = latitudeLLA_str[i];
00179         }
00180
00181         line0[19] = ut_mode + '0';
00182         line0[18] = ':';
00183         line0[17] = 'e';
00184         line0[16] = 'd';
00185         line0[15] = 'o';
00186         line0[14] = 'M';
00187
00188         for (int i = 0; i < LLA_LONG_BUFFER_SIZE; i++){
00189             line1[i] = longitudeLLA_str[i];
00190         }
00191
00192         //mode-specific parts
00193         if (ut_mode == STAT_MODE){
00194             //line1
00195             line1[MAX_COL-8] = 'H';
00196             line1[MAX_COL-7] = 'D';
00197             line1[MAX_COL-6] = 'O';
00198             line1[MAX_COL-5] = 'P';
00199             line1[MAX_COL-4] = ':';
00200             line1[MAX_COL-3] = hdop[0];
00201             line1[MAX_COL-2] = hdop[1];
00202             line1[MAX_COL-1] = hdop[2];
00203
00204             //line2
00205             line2[0] = 'U';
00206             line2[1] = 'T';
00207             line2[2] = 'C';
00208             line2[3] = ':';
00209             for (int i=0; i < GGA.UTC_BUFFER_SIZE; i++){
00210                 line2[4+i] = utc_time[i];
00211             }
00212
00213             line2[MAX_COL-1] = satellites_used[1];
00214             line2[MAX_COL-2] = satellites_used[0];
00215             line2[MAX_COL-3] = ':';
00216             line2[MAX_COL-4] = 'V';
00217             line2[MAX_COL-5] = 'S';
00218             line2[MAX_COL-6] = '#';
00219
00220             //line3
00221             line3[0] = 'V';
00222             line3[1] = 'e';
00223             line3[2] = 'l';
00224             line3[3] = ':';
00225             //put vel here once we get it done TODO
00226             for (int i =0; i < 6; i++){
00227                 line3[4+i] = speed[i];
00228             }
00229
00230             line3[MAX_COL-9] = 'A';
00231             line3[MAX_COL-8] = 'l';
00232             line3[MAX_COL-7] = 't';
00233             for (int i = 0; i < 6; i ++){

```



```

00234         line3[MAX_COL-6+i] = msl_altitude[i];
00235     }
00236
00237     }else { //if mode != STAT_MODE
00238         //line 1
00239         switch (ut_operation){
00240             case SAVE_OP:
00241                 strncpy(line1+(MAX_COL-5), SAVE_STR, 5);
00242                 break;
00243             case CLEAR_OP:
00244                 strncpy(line1+(MAX_COL-5), CLEAR_STR, 5);
00245                 break;
00246             case RESET_OP:
00247                 strncpy(line1+(MAX_COL-5), RESET_STR, 5);
00248                 break;
00249             default:
00250                 strncpy(line1+(MAX_COL-5), "Error", 5);
00251                 break;
00252         }
00253         //line2
00254         line2[MAX_COL-4] = 'M';
00255         line2[MAX_COL-3] = 'e';
00256         line2[MAX_COL-2] = 'm';
00257         line2[MAX_COL-1] = ut_memory_0idx + '0';
00258
00259         for (int i = 0; i < LLA_LAT_BUFFER_SIZE-2; i++){
00260             line2[i] = ut_lat_mem_str[i];
00261         }
00262         for (int i = 0; i < LLA_LONG_BUFFER_SIZE; i++){
00263             line3[i] = ut_long_mem_str[i];
00264         }
00265
00266         //line3
00267         line3[MAX_COL-10]= 'D';
00268         line3[MAX_COL-9]= 'i';
00269         line3[MAX_COL-8]= 's';
00270         line3[MAX_COL-7]= 't';
00271
00272
00273     } //end mode checks
00274 } //end else
00275
00276 ds_print_string(line0, MAX_COL, 0);
00277 ds_print_string(line1, MAX_COL, 1);
00278 ds_print_string(line2, MAX_COL, 2);
00279 ds_print_string(line3, MAX_COL, 3);
00280
00281
00282 } //end update display

```

## 3.11 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/nf/nf.c File Reference

```

#include <avr/interrupt.h>
#include <string.h>
#include <stdlib.h>
#include <stdint-gcc.h>
#include "nf.h"
#include "nf_types.h"
#include "../lib/uart.h"
#include "../ds/ds.h"
#include "../ut/utilities.h"

```

### Macros

- #define F\_CPU 4000000UL
- #define UART\_BAUD\_RATE 9600
- #define NMEA\_MSG\_ID\_SIZE 5
- #define GGA\_TYPE "GPGGA"
- #define GLL\_TYPE "GPGLL"

- `#define GSA_TYPE "GPGSA"`
- `#define GSV_TYPE "GPGSV"`
- `#define MSS_TYPE "GPMSS"`
- `#define RMC_TYPE "GPRMC"`
- `#define VTG_TYPE "GPVTG"`
- `#define ZDA_TYPE "GPZDA"`
- `#define OK_TO_SEND_TYPE "PSRF1"`

## Functions

- `void nf_clear_nav_strings ()`  
*Clear all navigation strings except UTC time.*
- `uint8_t nf_init ()`  
*Initialize the navigation fetch module. This function initializes UART communication and clears navigation strings.*
- `void get_serial_char (char *outputchar)`  
*Read a single character from the serial communication. This function blocks until a character is received.*
- `void read_nmea_msg_raw ()`  
*Read a raw NMEA message from the serial communication. This function reads a raw NMEA message and processes GGA and VTG messages. Additionally, it polls buttons between messages.*
- `void convertNMEAtoLLA ()`  
*Convert NMEA format coordinates to Latitude, Longitude, and Altitude (LLA) format. This function converts NMEA format coordinates to LLA format and stores them in global variables.*

## Variables

- `char utc_time [GGA_UTC_BUFFER_SIZE]`
- `char latitude [GGA_LAT_BUFFER_SIZE]`
- `char ns_indicator [GGA_INDICATOR_SIZE]`
- `char longitude [GGA_LONG_BUFFER_SIZE]`
- `char ew_indicator [GGA_INDICATOR_SIZE]`
- `char position_fix_indicator [GGA_INDICATOR_SIZE]`
- `char satellites_used [GGA_SV_USD_BUFFER_SIZE]`
- `char hdop [GGA_HDOP_BUFFER_SIZE]`
- `char msl_altitude [GGA_ALTITUDE_BUFFER_SIZE]`
- `char speed [VTG_SPEED_BUFFER_SIZE]`
- `float latitudeLLA_float`
- `float longitudeLLA_float`
- `float altitudeLLA_float`
- `char latitudeLLA_str [LLA_LAT_BUFFER_SIZE]`
- `char longitudeLLA_str [LLA_LONG_BUFFER_SIZE]`
- `char altitudeLLA_str [LLA_ALT_BUFFER_SIZE]`

## 3.11.1 Macro Definition Documentation

### 3.11.1.1 F\_CPU

```
#define F_CPU 4000000UL
```

Define the CPU frequency to 4MHz.

Definition at line 2 of file `nf.c`.

### 3.11.1.2 GGA\_TYPE

```
#define GGA_TYPE "GPGGA"
```

Definition at line 20 of file [nf.c](#).

### 3.11.1.3 GLL\_TYPE

```
#define GLL_TYPE "GPGLL"
```

Definition at line 21 of file [nf.c](#).

### 3.11.1.4 GSA\_TYPE

```
#define GSA_TYPE "GPGSA"
```

Definition at line 22 of file [nf.c](#).

### 3.11.1.5 GSV\_TYPE

```
#define GSV_TYPE "GPGSV"
```

Definition at line 23 of file [nf.c](#).

### 3.11.1.6 MSS\_TYPE

```
#define MSS_TYPE "GPMSS"
```

Definition at line 24 of file [nf.c](#).

### 3.11.1.7 NMEA\_MSG\_ID\_SIZE

```
#define NMEA_MSG_ID_SIZE 5
```

Definition at line 17 of file [nf.c](#).

### 3.11.1.8 OK\_TO\_SEND\_TYPE

```
#define OK_TO_SEND_TYPE "PSRF1"
```

Definition at line 28 of file [nf.c](#).

### 3.11.1.9 RMC\_TYPE

```
#define RMC_TYPE "GPRMC"
```

Definition at line 25 of file [nf.c](#).

### 3.11.1.10 UART\_BAUD\_RATE

```
#define UART_BAUD_RATE 9600
```

Definition at line 16 of file [nf.c](#).

### 3.11.1.11 VTG\_TYPE

```
#define VTG_TYPE "GPVTG"
```

Definition at line 26 of file [nf.c](#).

### 3.11.1.12 ZDA\_TYPE

```
#define ZDA_TYPE "GPZDA"
```

Definition at line 27 of file [nf.c](#).

## 3.11.2 Function Documentation

### 3.11.2.1 convertNMEAtoLLA()

```
void convertNMEAtoLLA ( )
```

Convert NMEA format coordinates to Latitude, Longitude, and Altitude (LLA) format. This function converts NMEA format coordinates to LLA format and stores them in global variables.

Convert NMEA format coordinates to Latitude, Longitude, and Altitude (LLA) format.

Definition at line 334 of file [nf.c](#).

### 3.11.2.2 get\_serial\_char()

```
void get_serial_char (
    char * outputchar )
```

Read a single character from the serial communication. This function blocks until a character is received.

Read a single character from the serial communication.

#### Parameters

<i>outputchar</i>	Pointer to the character variable to store the read character.
-------------------	--

Definition at line 114 of file [nf.c](#).

### 3.11.2.3 `nf_clear_nav_strings()`

```
void nf_clear_nav_strings ( )
```

Clear all navigation strings except UTC time.

Definition at line 60 of file [nf.c](#).

### 3.11.2.4 `nf_init()`

```
uint8_t nf_init ( )
```

Initialize the navigation fetch module. This function initializes UART communication and clears navigation strings.

Initialize the navigation fetch module.

#### Returns

0 if initialization is successful, otherwise returns 1.

Definition at line 82 of file [nf.c](#).

### 3.11.2.5 `read_nmea_msg_raw()`

```
void read_nmea_msg_raw ( )
```

Read a raw NMEA message from the serial communication. This function reads a raw NMEA message and processes GGA and VTG messages. Additionally, it polls buttons between messages.

Read a raw NMEA message from the serial communication and extracts relevant data.

Definition at line 176 of file [nf.c](#).

## 3.11.3 Variable Documentation

### 3.11.3.1 `altitudeLLA_float`

```
float altitudeLLA_float
```

Altitude in meters

Definition at line 45 of file [nf.c](#).

### 3.11.3.2 `altitudeLLA_str`

```
char altitudeLLA_str[LLA_ALT_BUFFER_SIZE]
```

Altitude in meters

Definition at line 48 of file [nf.c](#).

### 3.11.3.3 ew\_indicator

```
char ew_indicator[GGA_INDICATOR_SIZE]
```

E/W Indicator, 'E' for east or 'W' for west

Definition at line 36 of file [nf.c](#).

### 3.11.3.4 hdop

```
char hdop[GGA_HDOP_BUFFER_SIZE]
```

HDOP (Horizontal Dilution of Precision), e.g., "1.0"

Definition at line 39 of file [nf.c](#).

### 3.11.3.5 latitude

```
char latitude[GGA_LAT_BUFFER_SIZE]
```

Latitude, e.g., "3723.2475"

Definition at line 33 of file [nf.c](#).

### 3.11.3.6 latitudeLLA\_float

```
float latitudeLLA_float
```

Latitude in degrees

Definition at line 43 of file [nf.c](#).

### 3.11.3.7 latitudeLLA\_str

```
char latitudeLLA_str[LLA_LAT_BUFFER_SIZE]
```

Latitude in degrees

Definition at line 46 of file [nf.c](#).

### 3.11.3.8 longitude

```
char longitude[GGA_LONG_BUFFER_SIZE]
```

Longitude, e.g., "12158.3416"

Definition at line 35 of file [nf.c](#).

### 3.11.3.9 longitudeLLA\_float

```
float longitudeLLA_float
```

Longitude in degrees

Definition at line 44 of file [nf.c](#).

### 3.11.3.10 longitudeLLA\_str

```
char longitudeLLA_str[LLA_LONG_BUFFER_SIZE]
```

Longitude in degrees

Definition at line 47 of file [nf.c](#).

### 3.11.3.11 msl\_altitude

```
char msl_altitude[GGA_ALTITUDE_BUFFER_SIZE]
```

Mean Sea Level Altitude, e.g., "1.0"

Definition at line 40 of file [nf.c](#).

### 3.11.3.12 ns\_indicator

```
char ns_indicator[GGA_INDICATOR_SIZE]
```

N/S Indicator, 'N' for north or 'S' for south

Definition at line 34 of file [nf.c](#).

### 3.11.3.13 position\_fix\_indicator

```
char position_fix_indicator[GGA_INDICATOR_SIZE]
```

Position Fix Indicator, see Table 1-4

Definition at line 37 of file [nf.c](#).

### 3.11.3.14 satellites\_used

```
char satellites_used[GGA_SV_USD_BUFFER_SIZE]
```

Satellites Used, range 0 to 12 eg 07

Definition at line 38 of file [nf.c](#).

### 3.11.3.15 speed

```
char speed[VTG_SPEED_BUFER_SIZE]
```

Speed, e.g., "0.0"

Definition at line 41 of file [nf.c](#).

### 3.11.3.16 utc\_time

```
char utc_time[GGA_UTC_BUFFER_SIZE]
```

UTC Time, e.g., "161229.487"

Definition at line 32 of file [nf.c](#).

## 3.12 nf.c

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

```
00001 #ifndef F_CPU
00002 #define F_CPU 4000000UL
00003 #endif
00004
00005 #include <avr/interrupt.h>
00006 #include <string.h>
00007 #include <stdlib.h>
00008 #include <stdint-gcc.h>
00009 #include "nf.h"
00010 #include "nf_types.h"
00011 #include "../lib/uart.h"
00012 #include "../ds/ds.h"
00013 #include "../ut/utilities.h"
00014
00015 //defines
00016 #define UART_BAUD_RATE 9600
00017 #define NMEA_MSG_ID_SIZE 5
00018
00019 //NMEA message IDs from NMEA documentation
00020 #define GGA_TYPE "GPGGA"
00021 #define GLL_TYPE "GPGLL"
00022 #define GSA_TYPE "GPGSA"
00023 #define GSV_TYPE "GPGSV"
00024 #define MSS_TYPE "GPMSS"
00025 #define RMC_TYPE "GPRMC"
00026 #define VTG_TYPE "GPVTG"
00027 #define ZDA_TYPE "GPZDA"
00028 #define OK_TO_SEND_TYPE "PSRF1"
00029
00030 //global
00031 //GGA MESSAGE
00032 char utc_time[GGA_UTC_BUFFER_SIZE];
00033 char latitude[GGA_LAT_BUFFER_SIZE];
00034 char ns_indicator[GGA_INDICATOR_SIZE];
00035 char longitude[GGA_LONG_BUFFER_SIZE];
00036 char ew_indicator[GGA_INDICATOR_SIZE];
00037 char position_fix_indicator[GGA_INDICATOR_SIZE];
00038 char satellites_used[GGA_SV_USD_BUFFER_SIZE];
00039 char hdop[GGA_HDOP_BUFFER_SIZE];
00040 char msl_altitude[GGA_ALTITUDE_BUFFER_SIZE];
00041 char speed[VTG_SPEED_BUFER_SIZE];
00043 float latitudeLLA_float;
00044 float longitudeLLA_float;
00045 float altitudeLLA_float;
00046 char latitudeLLA_str[LLA_LAT_BUFFER_SIZE];
00047 char longitudeLLA_str[LLA_LONG_BUFFER_SIZE];
00048 char altitudeLLA_str[LLA_ALT_BUFFER_SIZE];
00050 //local static
00051 static char nmea_msg_id_buffer[NMEA_MSG_ID_SIZE];
00052 static char gga_msg_buffer[GGA_SIZE];
00055 //function definitions
00056
```



```

00060 void nf_clear_nav_strings() {
00061     memset(nmea_msg_id_buffer, 0, sizeof(char)*NMEA_MSG_ID_SIZE); //zeroize
00062     memset(gga_msg_buffer, ' ', sizeof(char)*GGA_SIZE); //zeroize msg buffer
00063     memset(latitude, ' ', GGA_LAT_BUFFER_SIZE * sizeof(char));
00064     memset(ns_indicator, ' ', GGA_INDICATOR_SIZE * sizeof(char));
00065     memset(longitude, ' ', GGA_LONG_BUFFER_SIZE * sizeof(char));
00066     memset(ew_indicator, ' ', GGA_INDICATOR_SIZE * sizeof(char));
00067     memset(position_fix_indicator, ' ', GGA_INDICATOR_SIZE * sizeof(char));
00068     memset(satellites_used, ' ', GGA_SV_USD_BUFFER_SIZE * sizeof(char));
00069     memset(hdop, ' ', GGA_HDOP_BUFFER_SIZE * sizeof(char));
00070     memset(msl_altitude, ' ', GGA_ALTITUDE_BUFFER_SIZE * sizeof(char));
00071
00072     memset(latitudeLLA_str, ' ', LLA_LAT_BUFFER_SIZE * sizeof(char));
00073     memset(longitudeLLA_str, ' ', LLA_LONG_BUFFER_SIZE * sizeof(char));
00074     memset(altitudeLLA_str, ' ', LLA_ALT_BUFFER_SIZE * sizeof(char));
00075 }
00076
00082 uint8_t nf_init() {
00083     cli();
00084     /*
00085     * Initialize UART library, pass baudrate and AVR cpu clock
00086     * with the macro
00087     * UART_BAUD_SELECT() (normal speed mode )
00088     * or
00089     * UART_BAUD_SELECT_DOUBLE_SPEED() ( double speed mode)
00090     */
00091     uart_init( UART_BAUD_SELECT(UART_BAUD_RATE,F_CPU) );
00092
00093     // Initialize the arrays within gga_msg
00094     memset(utc_time, ' ', GGA_UTC_BUFFER_SIZE * sizeof(char));
00095     memset(speed, ' ', VTG_SPEED_BUFER_SIZE * sizeof(char));
00096
00097     nf_clear_nav_strings();
00098
00099     latitudeLLA_float = 0;
00100     longitudeLLA_float = 0;
00101     altitudeLLA_float = 0;
00102
00103
00104
00105     sei(); //UART is interrupt based;
00106     return NF_INIT_SUCCESS;
00107 }
00108
00114 void get_serial_char(char* outputchar){
00115     unsigned int c;
00116     while (1){
00117         /*
00118         * Get received character from ringbuffer
00119         * uart_getc() returns in the lower byte the received character and
00120         * in the higher byte (bitmask) the last receive error
00121         * UART_NO_DATA is returned when no data is available.
00122         */
00123         c = uart_getc();
00124         if (!( c & UART_NO_DATA ))
00125         {
00126             /*
00127             * new data available from UART
00128             * check for Frame or Overrun error
00129             */
00130             #ifdef __DEBUG__
00131                 if ( c & UART_FRAME_ERROR )
00132                 {
00133                     /* Framing Error detected, i.e no stop bit detected */
00134                     #ifdef _DEBUG_
00135                         char* output = "NF Frame Error! ";
00136                         ds_print_string(output, 16, 0);
00137                     #endif
00138                 }
00139                 if ( c & UART_OVERRUN_ERROR )
00140                 {
00141                     /*
00142                     * Overrun, a character already present in the UART UDR register was
00143                     * not read by the interrupt handler before the next character arrived,
00144                     * one or more received characters have been dropped
00145                     */
00146                     char* output = "OR ER";
00147                     ds_print_string(output, 16, 0);
00148                 }
00149                 if ( c & UART_BUFFER_OVERFLOW )
00150                 {
00151                     /*
00152                     * We are not reading the receive buffer fast enough,
00153                     * one or more received character have been dropped
00154                     */
00155                     char* output = "OF ER";
00156

```

```

00157         ds_print_string(output, 5, 0);
00158     }
00159 #endif
00160
00161     /*
00162     * send received character back
00163     */
00164     *outputchar = (unsigned char)c;
00165     return;
00166 }
00167 }
00168
00169 }
00170
00176 void read_nmea_msg_raw(){
00177     //nf_clear_nav_strings();
00178     char tempChar;
00179     do {
00180         get_serial_char(&tempChar);
00181     } while (tempChar != '$');
00182
00183     for (int i = 0; i < NMEA_MSG_ID_SIZE; i++){
00184         get_serial_char(nmea_msg_id_buffer + i);
00185     }
00186     get_serial_char(&tempChar); //eat comma
00187
00188     int counter = 0; //common counter to save memory alloc time
00189     //-----
00190     //only need GGA and VTG for requirements. Drop others
00191     if(strcmp(nmea_msg_id_buffer, GGA_TYPE) == 0){
00192         //scrape raw
00193         for (counter = 0; counter < GGA_SIZE; counter++){
00194             get_serial_char(gga_msg_buffer+counter);
00195         }
00196         //process message
00197         int offset = 0;
00198         //grab UTC if available
00199         if (gga_msg_buffer[offset] != ','){
00200             for (int i = 0; i < GGA_UTC_BUFFER_SIZE; i++){
00201                 utc_time[i] = gga_msg_buffer[offset++];
00202             }
00203             //ds_print_string(utc_time, GGA_UTC_BUFFER_SIZE, 0);
00204         }else{ //otherwise skip comma from if statement
00205             offset++;
00206         }
00207         offset++; // skip comma
00208
00209         //grab LAT if available
00210         if (gga_msg_buffer[offset] != ','){
00211             for (int i = 0; i < GGA_LAT_BUFFER_SIZE; i++){
00212                 latitude[i] = gga_msg_buffer[offset++];
00213             }
00214             //ds_print_string(latitude, GGA_LAT_BUFFER_SIZE, 0);
00215         }else{ //otherwise skip comma from if statement
00216             offset++;
00217         }
00218         offset++; // skip comma
00219
00220         //grab NS indicator if available
00221         if (gga_msg_buffer[offset] != ','){
00222             ns_indicator[0] = gga_msg_buffer[offset++];
00223             //ds_print_string(ew_indicator, GGA_INDICATOR_SIZE, 1);
00224         }else{ //otherwise skip comma from if statement
00225             offset++;
00226         }
00227         offset++; // skip comma
00228
00229         //grab LONG if available
00230         if (gga_msg_buffer[offset] != ','){
00231             for (int i = 0; i < GGA_LONG_BUFFER_SIZE; i++){
00232                 longitude[i] = gga_msg_buffer[offset++];
00233             }
00234             //ds_print_string(longitude, GGA_LONG_BUFFER_SIZE, 1);
00235         }else{ //otherwise skip comma from if statement
00236             offset++;
00237         }
00238         offset++; // skip comma
00239
00240         //grab EW indicator if available
00241         if (gga_msg_buffer[offset] != ','){
00242             for (int i = 0; i < GGA_INDICATOR_SIZE; i++){
00243                 ew_indicator[i] = gga_msg_buffer[offset++];
00244             }
00245             //ds_print_string(ew_indicator, GGA_INDICATOR_SIZE, 1);
00246         }else{ //otherwise skip comma from if statement
00247             offset++;
00248         }
00249         offset++; // skip comma
00250
00251         //grab EW indicator if available
00252         if (gga_msg_buffer[offset] != ','){
00253             for (int i = 0; i < GGA_INDICATOR_SIZE; i++){
00254                 ew_indicator[i] = gga_msg_buffer[offset++];
00255             }
00256             //ds_print_string(ew_indicator, GGA_INDICATOR_SIZE, 1);
00257         }else{ //otherwise skip comma from if statement
00258             offset++;
00259         }
00260         offset++; // skip comma
00261     }
00262 }

```

```

00254         offset++;
00255     }
00256     offset++; // skip comma
00257
00258
00260     //grab FIX indicator if available
00261     if (gga_msg_buffer[offset] != ','){
00262         for (int i =0; i < GGA_INDICATOR_SIZE; i++){
00263             position_fix_indicator[i] = gga_msg_buffer[offset++];
00264         }
00265         //ds_print_string(position_fix_indicator, GGA_INDICATOR_SIZE, 1);
00266     }else{ //otherwise skip comma from if statement
00267         offset++;
00268     }
00269     offset++; // skip comma
00270
00272     //grab NUM_SV if available
00273     if (gga_msg_buffer[offset] != ','){
00274         for (int i =0; i < GGA_SV_USD_BUFFER_SIZE; i++){
00275             satellites_used[i] = gga_msg_buffer[offset++];
00276         }
00277         //ds_print_string(satellites_used, GGA_SV_USD_BUFFER_SIZE, 1);
00278     }else{ //otherwise skip comma from if statement
00279         offset++;
00280     }
00281     offset++; // skip comma
00282
00284     //grab HDOP if available
00285     if (gga_msg_buffer[offset] != ','){
00286         for (int i =0; i < GGA_HDOP_BUFFER_SIZE; i++){
00287             hdop[i] = gga_msg_buffer[offset++];
00288         }
00289         //ds_print_string(hdop, GGA_HDOP_BUFFER_SIZE, 0);
00290     }else{ //otherwise skip comma from if statement
00291         offset++;
00292     }
00293     offset++;
00294     //end of known sizes. Alt is dynamic
00295     tempChar = 0;
00296     int i = 0;
00297     get_serial_char(&tempChar); //eat comma
00298     get_serial_char(&tempChar);
00299     while ((tempChar != ',') && (i < GGA_ALTITUDE_BUFFER_SIZE)){
00300         msl_altitude[i++] = tempChar;
00301         get_serial_char(&tempChar);
00302     }
00303
00304
00305 } //END GGA
00306 else if (strcmp(nmea_msg_id_buffer, VTG_TYPE) == 0){
00307     for (uint8_t comma_counter = 6; comma_counter >0; comma_counter--){
00308         do{
00309             get_serial_char(&tempChar);
00310         } while (tempChar != ',');
00311     }
00312     memset(speed, ' ', VTG_SPEED_BUFER_SIZE * sizeof(char));
00313     //should be at speed now
00314     int i = 0;
00315     get_serial_char(&tempChar); //eat comma
00316     while ((tempChar != ',') && (i < VTG_SPEED_BUFER_SIZE)){
00317         speed[i++] = tempChar;
00318         get_serial_char(&tempChar);
00319     }
00320
00321 } //end VTG msg
00322
00323 //poll buttons between messages
00324 //ut_poll_btns();
00325 }
00326
00327
00328
00329
00334 void convertNMEAtoLLA() {
00335     // Convert latitude from NMEA format to degrees
00336     double deg = 0.0;
00337     double min = 0.0;
00338
00339     // Convert the latitude from degrees and minutes to degree decimal
00340     char tempBuffer[9] = {0};
00341     for (int i = 0; i < 8; i++){
00342         tempBuffer[i] = latitude[2+i];
00343     }
00344     min = atof(tempBuffer);
00345
00346     deg += (1.0 * (latitude[GGA_LAT_BUFFER_SIZE - 9] - '0'));
00347     deg += (10.0 * (latitude[GGA_LAT_BUFFER_SIZE - 10] - '0'));

```

```

00348
00349     latitudeLLA_float = deg + (min / 60.0f); // Combine degrees and minutes
00350
00351
00352     // Extract integer and decimal parts
00353     uint16_t integer_part = (uint16_t)latitudeLLA_float;
00354     float fractional_part = latitudeLLA_float - integer_part;
00355     uint32_t decimal_part = (uint32_t)(fractional_part * 100000);
00356
00357     // Convert integer part to string
00358     latitudeLLA_str[1] = '0' + ((integer_part / 10) % 10); // Tens
00359     latitudeLLA_str[2] = '0' + (integer_part % 10); // Ones
00360
00361     // Decimal point
00362     latitudeLLA_str[3] = '.';
00363
00364     // Convert decimal part to string
00365     latitudeLLA_str[4] = '0' + ((decimal_part / 10000) % 10); // Ten-thousands
00366     latitudeLLA_str[5] = '0' + ((decimal_part / 1000) % 10); // Thousands
00367     latitudeLLA_str[6] = '0' + ((decimal_part / 100) % 10); // Hundreds
00368     latitudeLLA_str[7] = '0' + ((decimal_part / 10) % 10); // Tens
00369     latitudeLLA_str[8] = '0' + (decimal_part % 10); // Ones
00370
00371     if (ns_indicator[0] == 'S'){
00372         latitudeLLA_str[0] = '-';
00373         latitudeLLA_float *= -1;
00374     } else{
00375         latitudeLLA_str[0] = '+';
00376     }
00377
00378
00379     //long
00380     deg = 0.0;
00381     min = 0.0;
00382
00383     // Convert the longitude from degrees and minutes to degree decimal
00384     char tempBuffer2[9] = {0};
00385     for (int i = 0; i < 8; i++){
00386         tempBuffer2[i] = longitude[3+i];
00387     }
00388     min = atof(tempBuffer2);
00389
00390     deg += (1.0 * (longitude[2] - '0'));
00391     deg += (10.0 * (longitude[1] - '0'));
00392     deg += (100.0 * (longitude[0] - '0'));
00393
00394     longitudeLLA_float = deg + (min / 60.0f); // Combine degrees and minutes
00395
00396     // Extract integer and decimal parts
00397     integer_part = (uint16_t)longitudeLLA_float;
00398     fractional_part = longitudeLLA_float - integer_part;
00399     decimal_part = (uint32_t)(fractional_part * 100000);
00400
00401     // Convert integer part to string
00402     longitudeLLA_str[1] = '0' + ((integer_part / 100) % 10); // Hundreds
00403     longitudeLLA_str[2] = '0' + ((integer_part / 10) % 10); // Tens
00404     longitudeLLA_str[3] = '0' + (integer_part % 10); // Ones
00405
00406     // Decimal point
00407     longitudeLLA_str[4] = '.';
00408
00409     // Convert decimal part to string
00410     longitudeLLA_str[5] = '0' + ((decimal_part / 10000) % 10); // Ten-thousands
00411     longitudeLLA_str[6] = '0' + ((decimal_part / 1000) % 10); // Thousands
00412     longitudeLLA_str[7] = '0' + ((decimal_part / 100) % 10); // Hundreds
00413     longitudeLLA_str[8] = '0' + ((decimal_part / 10) % 10); // Tens
00414     longitudeLLA_str[9] = '0' + (decimal_part % 10); // Ones
00415
00416     if (ns_indicator[0] == 'S'){
00417         longitudeLLA_str[0] = '-';
00418         longitudeLLA_float *= -1;
00419     } else{
00420         longitudeLLA_str[0] = '+';
00421     }
00422
00423     altitudeLLA_float = atof(msl_altitude);
00424 }

```

### 3.13 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/nf/nf.h File Reference

Header file containing common functions and definitions for the navigation fetch (NV) CSC.

```
#include "../ut/ut_types.h"
#include "../ds/ds.h"
```

## Macros

- `#define NF_INIT_SUCCESS 0`
- `#define NF_INIT_FAILURE 1`

## Functions

- `uint8_t nf_init ()`  
*Initialize the navigation fetch module.*
- `void get_serial_char (char *outputchar)`  
*Read a single character from the serial communication.*
- `void read_nmea_msg_raw ()`  
*Read a raw NMEA message from the serial communication and extracts relevant data.*
- `void convertNMEAtoLLA ()`  
*Convert NMEA format coordinates to Latitude, Longitude, and Altitude (LLA) format.*

### 3.13.1 Detailed Description

Header file containing common functions and definitions for the navigation fetch (NV) CSC.

This file provides declarations for common utility functions and definitions associated with NV

Definition in file [nf.h](#).

### 3.13.2 Macro Definition Documentation

#### 3.13.2.1 NF\_INIT\_FAILURE

```
#define NF_INIT_FAILURE 1
```

Definition at line 15 of file [nf.h](#).

#### 3.13.2.2 NF\_INIT\_SUCCESS

```
#define NF_INIT_SUCCESS 0
```

Definition at line 14 of file [nf.h](#).

### 3.13.3 Function Documentation

#### 3.13.3.1 `convertNMEAtoLLA()`

```
void convertNMEAtoLLA ( )
```

Convert NMEA format coordinates to Latitude, Longitude, and Altitude (LLA) format.

Convert NMEA format coordinates to Latitude, Longitude, and Altitude (LLA) format.

Definition at line [334](#) of file [nf.c](#).

#### 3.13.3.2 `get_serial_char()`

```
void get_serial_char (
    char * outputchar )
```

Read a single character from the serial communication.

**Parameters**

<i>outputchar</i>	Pointer to the character variable to store the read character.
-------------------	--

Read a single character from the serial communication.

**Parameters**

<i>outputchar</i>	Pointer to the character variable to store the read character.
-------------------	--

Definition at line 114 of file [nf.c](#).

**3.13.3.3 nf\_init()**

```
uint8_t nf_init ( )
```

Initialize the navigation fetch module.

**Returns**

0 if initialization is successful

Initialize the navigation fetch module.

**Returns**

0 if initialization is successful, otherwise returns 1.

Definition at line 82 of file [nf.c](#).

**3.13.3.4 read\_nmea\_msg\_raw()**

```
void read_nmea_msg_raw ( )
```

Read a raw NMEA message from the serial communication and extracts relevant data.

Read a raw NMEA message from the serial communication and extracts relevant data.

Definition at line 176 of file [nf.c](#).

**3.14 nf.h**

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

```
00001
00008 #ifndef NF_H_
00009 #define NF_H_
00010
00011 #include "../ut/ut_types.h"
00012 #include "../ds/ds.h"
00013
00014 #define NF_INIT_SUCCESS 0
00015 #define NF_INIT_FAILURE 1
00016
00017
00022 uint8_t nf_init();
00023
00028 void get_serial_char(char* outputchar);
00029
00033 void read_nmea_msg_raw();
00034
00038 void convertNMEAtoLLA();
00039
00040
00041 #endif /* NF_H_ */
```

## 3.15 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/nf/nf\_types.h File Reference

### Macros

- `#define GGA_INDICATOR_SIZE 1`
- `#define LLA_LONG_BUFFER_SIZE 10`
- `#define LLA_LAT_BUFFER_SIZE 11`
- `#define LLA_ALT_BUFFER_SIZE 6`
- `#define GGA_UTC_BUFFER_SIZE 9`
- `#define GGA_LAT_BUFFER_SIZE 10`
- `#define GGA_LONG_BUFFER_SIZE 11`
- `#define GGA_SV_USD_BUFFER_SIZE 2`
- `#define GGA_HDOP_BUFFER_SIZE 3`
- `#define GGA_NUM_INDICATORS_ACTIVE 3`
- `#define GGA_NUM_ITEMS_ACTIVE 8`
- `#define GGA_ALTITUDE_BUFFER_SIZE 7`
- `#define VTG_SPEED_BUFFER_SIZE 7`
- `#define GGA_SIZE (GGA_UTC_BUFFER_SIZE + GGA_LAT_BUFFER_SIZE + GGA_LONG_BUFFER_SIZE + GGA_SV_USD_BUFFER_SIZE + GGA_HDOP_BUFFER_SIZE + GGA_NUM_INDICATORS_ACTIVE + GGA_NUM_ITEMS_ACTIVE)`

### Variables

- `char utc_time [GGA_UTC_BUFFER_SIZE]`
- `char latitude [GGA_LAT_BUFFER_SIZE]`
- `char ns_indicator [GGA_INDICATOR_SIZE]`
- `char longitude [GGA_LONG_BUFFER_SIZE]`
- `char ew_indicator [GGA_INDICATOR_SIZE]`
- `char position_fix_indicator [GGA_INDICATOR_SIZE]`
- `char satellites_used [GGA_SV_USD_BUFFER_SIZE]`
- `char hdop [GGA_HDOP_BUFFER_SIZE]`
- `char msl_altitude [GGA_ALTITUDE_BUFFER_SIZE]`
- `float latitudeLLA_float`
- `float longitudeLLA_float`
- `float altitudeLLA_float`
- `char latitudeLLA_str [LLA_LAT_BUFFER_SIZE]`
- `char longitudeLLA_str [LLA_LONG_BUFFER_SIZE]`
- `char altitudeLLA_str [LLA_ALT_BUFFER_SIZE]`
- `char speed [VTG_SPEED_BUFFER_SIZE]`

### 3.15.1 Macro Definition Documentation

#### 3.15.1.1 GGA\_ALTITUDE\_BUFFER\_SIZE

```
#define GGA_ALTITUDE_BUFFER_SIZE 7
```

Size of the buffer for storing altitude in the GGA message

Definition at line 26 of file [nf\\_types.h](#).



#### 3.15.1.2 GGA\_HDOP\_BUFFER\_SIZE

```
#define GGA_HDOP_BUFFER_SIZE 3
```

Size of the buffer for storing HDOP in the GGA message

Definition at line 23 of file [nf\\_types.h](#).

#### 3.15.1.3 GGA\_INDICATOR\_SIZE

```
#define GGA_INDICATOR_SIZE 1
```

Size of the N/S and E/W indicators in the GGA message

Definition at line 12 of file [nf\\_types.h](#).

#### 3.15.1.4 GGA\_LAT\_BUFFER\_SIZE

```
#define GGA_LAT_BUFFER_SIZE 10
```

Size of the buffer for storing latitude in the GGA message

Definition at line 20 of file [nf\\_types.h](#).

#### 3.15.1.5 GGA\_LONG\_BUFFER\_SIZE

```
#define GGA_LONG_BUFFER_SIZE 11
```

Size of the buffer for storing longitude in the GGA message

Definition at line 21 of file [nf\\_types.h](#).

#### 3.15.1.6 GGA\_NUM\_INDICATORS\_ACTIVE

```
#define GGA_NUM_INDICATORS_ACTIVE 3
```

Number of active indicators in the GGA message

Definition at line 24 of file [nf\\_types.h](#).

#### 3.15.1.7 GGA\_NUM\_ITEMS\_ACTIVE

```
#define GGA_NUM_ITEMS_ACTIVE 8
```

Number of active items in the GGA message

Definition at line 25 of file [nf\\_types.h](#).

### 3.15.1.8 GGA\_SIZE

```
#define GGA_SIZE (GGA.UTC_BUFFER_SIZE + GGA.LAT_BUFFER_SIZE + GGA.LONG_BUFFER_SIZE + GGA.SV_USD_BUFFER_SIZE  
+ GGA.HDOP_BUFFER_SIZE + GGA.NUM_INDICATORS_ACTIVE + GGA.NUM_ITEMS_ACTIVE)
```

Total size of the GGA message

Definition at line 29 of file [nf\\_types.h](#).

### 3.15.1.9 GGA\_SV\_USD\_BUFFER\_SIZE

```
#define GGA_SV_USD_BUFFER_SIZE 2
```

Size of the buffer for storing satellites used in the GGA message

Definition at line 22 of file [nf\\_types.h](#).

### 3.15.1.10 GGA.UTC\_BUFFER\_SIZE

```
#define GGA.UTC_BUFFER_SIZE 9
```

Size of the buffer for storing UTC time in the GGA message

Definition at line 19 of file [nf\\_types.h](#).

### 3.15.1.11 LLA\_ALT\_BUFFER\_SIZE

```
#define LLA_ALT_BUFFER_SIZE 6
```

Size of the buffer for storing altitude in LLA format

Definition at line 16 of file [nf\\_types.h](#).

### 3.15.1.12 LLA\_LAT\_BUFFER\_SIZE

```
#define LLA_LAT_BUFFER_SIZE 11
```

Size of the buffer for storing latitude in LLA format

Definition at line 15 of file [nf\\_types.h](#).

### 3.15.1.13 LLA\_LONG\_BUFFER\_SIZE

```
#define LLA_LONG_BUFFER_SIZE 10
```

Size of the buffer for storing longitude in LLA format

Definition at line 14 of file [nf\\_types.h](#).

### 3.15.1.14 VTG\_SPEED\_BUFER\_SIZE

```
#define VTG_SPEED_BUFER_SIZE 7
```

Size of the buffer for storing speed in the VTG message

Definition at line 27 of file [nf\\_types.h](#).

## 3.15.2 Variable Documentation

### 3.15.2.1 altitudeLLA\_float

```
float altitudeLLA_float [extern]
```

Altitude in meters

Definition at line 45 of file [nf.c](#).

### 3.15.2.2 altitudeLLA\_str

```
char altitudeLLA_str[LLA_ALT_BUFFER_SIZE] [extern]
```

Altitude in meters

Definition at line 48 of file [nf.c](#).

### 3.15.2.3 ew\_indicator

```
char ew_indicator[GGA_INDICATOR_SIZE] [extern]
```

E/W Indicator, 'E' for east or 'W' for west

Definition at line 36 of file [nf.c](#).

### 3.15.2.4 hdop

```
char hdop[GGA_HDOP_BUFFER_SIZE] [extern]
```

HDOP (Horizontal Dilution of Precision), e.g., "1.0"

Definition at line 39 of file [nf.c](#).

### 3.15.2.5 latitude

```
char latitude[GGA_LAT_BUFFER_SIZE] [extern]
```

Latitude, e.g., "3723.24756"

Latitude, e.g., "3723.2475"

Definition at line 33 of file [nf.c](#).

### 3.15.2.6 latitudeLLA\_float

```
float latitudeLLA_float [extern]
```

Latitude in degrees

Definition at line 43 of file [nf.c](#).

### 3.15.2.7 latitudeLLA\_str

```
char latitudeLLA_str[LLA_LAT_BUFFER_SIZE] [extern]
```

Latitude in degrees

Definition at line 46 of file [nf.c](#).

### 3.15.2.8 longitude

```
char longitude[GGA_LONG_BUFFER_SIZE] [extern]
```

Longitude, e.g., "12158.34166"

Longitude, e.g., "12158.3416"

Definition at line 35 of file [nf.c](#).

### 3.15.2.9 longitudeLLA\_float

```
float longitudeLLA_float [extern]
```

Longitude in degrees

Definition at line 44 of file [nf.c](#).

### 3.15.2.10 longitudeLLA\_str

```
char longitudeLLA_str[LLA_LONG_BUFFER_SIZE] [extern]
```

Longitude in degrees

Definition at line 47 of file [nf.c](#).

### 3.15.2.11 msl\_altitude

```
char msl_altitude[GGA_ALTITUDE_BUFFER_SIZE] [extern]
```

Altitude above sea level in meters one decimal of precision.

Mean Sea Level Altitude, e.g., "1.0"

Definition at line 40 of file [nf.c](#).

### 3.15.2.12 ns\_indicator

```
char ns_indicator[GGA_INDICATOR_SIZE] [extern]
```

N/S Indicator, 'N' for north or 'S' for south

Definition at line 34 of file [nf.c](#).

### 3.15.2.13 position\_fix\_indicator

```
char position_fix_indicator[GGA_INDICATOR_SIZE] [extern]
```

Position Fix Indicator, see Table 1-4

Definition at line 37 of file [nf.c](#).

### 3.15.2.14 satellites\_used

```
char satellites_used[GGA_SV_USD_BUFFER_SIZE] [extern]
```

Satellites Used, range 0 to 12 eg 07

Definition at line 38 of file [nf.c](#).

### 3.15.2.15 speed

```
char speed[VTG_SPEED_BUFER_SIZE] [extern]
```

Speed in km/hr

Speed, e.g., "0.0"

Definition at line 41 of file [nf.c](#).

### 3.15.2.16 utc\_time

```
char utc_time[GGA_UTC_BUFFER_SIZE] [extern]
```

UTC Time, e.g., "161229.487"

Definition at line 32 of file [nf.c](#).

## 3.16 nf\_types.h

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

```

00001 /*
00002  * nf_types.h
00003  * @brief Header file containing common types and definitions for the navigation fetch (NV) CSC.
00004  *
00005  * This file provides definitions for common data types and sizes associated with the NV CSC.
00006  */
00007
00008
00009 #ifndef NF_TYPES_H_
00010 #define NF_TYPES_H_
00011
00012 #define GGA_INDICATOR_SIZE 1
00014 #define LLA_LONG_BUFFER_SIZE 10
00015 #define LLA_LAT_BUFFER_SIZE 11
00016 #define LLA_ALT_BUFFER_SIZE 6
00018 // Size definitions for components of the GGA message
00019 #define GGA_UTC_BUFFER_SIZE 9
00020 #define GGA_LAT_BUFFER_SIZE 10
00021 #define GGA_LONG_BUFFER_SIZE 11
00022 #define GGA_SV_USD_BUFFER_SIZE 2
00023 #define GGA_HDOP_BUFFER_SIZE 3
00024 #define GGA_NUM_INDICATORS_ACTIVE 3
00025 #define GGA_NUM_ITEMS_ACTIVE 8
00026 #define GGA_ALTITUDE_BUFFER_SIZE 7
00027 #define VTG_SPEED_BUFFER_SIZE 7
00029 #define GGA_SIZE (GGA_UTC_BUFFER_SIZE + GGA_LAT_BUFFER_SIZE + GGA_LONG_BUFFER_SIZE +
    GGA_SV_USD_BUFFER_SIZE + GGA_HDOP_BUFFER_SIZE + GGA_NUM_INDICATORS_ACTIVE + GGA_NUM_ITEMS_ACTIVE)
00031 extern char utc_time[GGA_UTC_BUFFER_SIZE];
00032 extern char latitude[GGA_LAT_BUFFER_SIZE];
00033 extern char ns_indicator[GGA_INDICATOR_SIZE];
00034 extern char longitude[GGA_LONG_BUFFER_SIZE];
00035 extern char ew_indicator[GGA_INDICATOR_SIZE];
00036 extern char position_fix_indicator[GGA_INDICATOR_SIZE];
00037 extern char satellites_used[GGA_SV_USD_BUFFER_SIZE];
00038 extern char hdop[GGA_HDOP_BUFFER_SIZE];
00039 extern char msl_altitude[GGA_ALTITUDE_BUFFER_SIZE];
00041 extern float latitudeLLA_float;
00042 extern float longitudeLLA_float;
00043 extern float altitudeLLA_float;
00045 extern char latitudeLLA_str[LLA_LAT_BUFFER_SIZE];
00046 extern char longitudeLLA_str[LLA_LONG_BUFFER_SIZE];
00047 extern char altitudeLLA_str[LLA_ALT_BUFFER_SIZE];
00049 extern char speed[VTG_SPEED_BUFFER_SIZE];
00051 #endif /* NF_TYPES_H_ */

```

## 3.17 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ut/ut\_types.h File Reference

Header file containing typedefs for common types.

### Macros

- `#define false ((boolean_t)0)`  
*Definition of false as boolean value 0.*
- `#define true ((boolean_t)1)`  
*Definition of true as boolean value 1.*

### Typedefs

- typedef unsigned char `uint8_t`  
*8-bit unsigned integer type.*
- typedef signed char `int8_t`  
*8-bit signed integer type.*
- typedef unsigned int `uint16_t`  
*16-bit unsigned integer type.*
- typedef unsigned char `boolean_t`  
*Boolean type (true/false).*

### 3.17.1 Detailed Description

Header file containing typedefs for common types.

This file provides typedefs for common types to improve code readability and portability.

#### Version

1.0

#### Copyright

(C) 2024 Bradley Johnson and Abele Atresso

Definition in file [ut\\_types.h](#).

### 3.17.2 Macro Definition Documentation

#### 3.17.2.1 false

```
#define false ((boolean_t)0)
```

Definition of false as boolean value 0.

Definition at line 37 of file [ut\\_types.h](#).

#### 3.17.2.2 true

```
#define true ((boolean_t)1)
```

Definition of true as boolean value 1.

Definition at line 42 of file [ut\\_types.h](#).

### 3.17.3 Typedef Documentation

#### 3.17.3.1 boolean\_t

```
typedef unsigned char boolean_t
```

Boolean type (true/false).

Definition at line 32 of file [ut\\_types.h](#).

### 3.17.3.2 int8\_t

```
typedef signed char int8_t
```

8-bit signed integer type.

Definition at line 22 of file [ut\\_types.h](#).

### 3.17.3.3 uint16\_t

```
typedef unsigned int uint16_t
```

16-bit unsigned integer type.

Definition at line 27 of file [ut\\_types.h](#).

### 3.17.3.4 uint8\_t

```
typedef unsigned char uint8_t
```

8-bit unsigned integer type.

Definition at line 17 of file [ut\\_types.h](#).

## 3.18 ut\_types.h

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

```
00001
00011 #ifndef UT_TYPES_H_
00012 #define UT_TYPES_H_
00013
00017 typedef unsigned char  uint8_t;
00018
00022 typedef signed char    int8_t;
00023
00027 typedef unsigned int   uint16_t;
00028
00032 typedef unsigned char  boolean_t;
00033
00037 #define false ((boolean_t)0)
00038
00042 #define true  ((boolean_t)1)
00043
00044 #endif /* UT_TYPES_H_ */
```

## 3.19 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ut/utilities.c File Reference

```
#include <avr/io.h>
#include "utilities.h"
```



## Functions

- `boolean_t is_button_pressed` (volatile `uint8_t` \*port, `uint8_t` pin)  
*Checks if a button is pressed.*
- void `ut_load_from_non_vol` (`uint8_t` index, float \*longitude, float \*latitude)  
*Loads position from non-volatile memory.*
- void `ut_write_to_non_vol` (`uint8_t` index)  
*Writes to non-volatile memory.*
- void `ut_convert_lat_float_to_string` (float lat\_float, char \*lat\_string)  
*Converts latitude from float to string.*
- void `ut_convert_long_float_to_string` (float long\_float, char \*long\_string)  
*Converts longitude from float to string.*
- void `ut_init` ()  
*Initializes the pins for buttons, loads from SD card, and initializes stored locations on startup.*
- void `ut_poll_btns` ()  
*Checks the status of each button sequentially.*

## Variables

- `boolean_t` ut\_mode
- `uint8_t` ut\_operation
- `uint8_t` ut\_memory\_0idx
- float `ut_lat_mem_floats` [MAX\_MEM\_INDEX]
- float `ut_long_mem_floats` [MAX\_MEM\_INDEX]
- char `ut_lat_mem_str` [LLA\_LAT\_BUFFER\_SIZE]
- char `ut_long_mem_str` [LLA\_LONG\_BUFFER\_SIZE]

## 3.19.1 Function Documentation

### 3.19.1.1 is\_button\_pressed()

```
boolean_t is_button_pressed (
    volatile uint8_t * port,
    uint8_t pin )
```

Checks if a button is pressed.

Simple utility function to poll an individual button. keeping for maintainability if button needs switched to different port or pin.

This function checks whether a specified button is pressed.

#### Parameters

<i>port</i>	Pointer to the port register.
<i>pin</i>	The pin number of the button.

#### Returns

true if the button is pressed, false otherwise.

Definition at line 289 of file [utilities.c](#).

### 3.19.1.2 ut\_convert\_lat\_float\_to\_string()

```
void ut_convert_lat_float_to_string (
    float lat_float,
    char * lat_string )
```

Converts latitude from float to string.

This function converts a floating-point latitude value to a string.

#### Parameters

<i>lat_float</i>	The latitude value as a float.
<i>lat_string</i>	Pointer to the string to store the converted latitude.

Definition at line 67 of file [utilities.c](#).

### 3.19.1.3 ut\_convert\_long\_float\_to\_string()

```
void ut_convert_long_float_to_string (
    float long_float,
    char * long_string )
```

Converts longitude from float to string.

This function converts a floating-point longitude value to a string.

#### Parameters

<i>long_float</i>	The longitude value as a float.
<i>long_string</i>	Pointer to the string to store the converted longitude.

Definition at line 105 of file [utilities.c](#).

### 3.19.1.4 ut\_init()

```
void ut_init ( )
```

Initializes the pins for buttons, loads from SD card, and initializes stored locations on startup.

This function initializes the pins for buttons, loads data from an SD card, and initializes stored locations on startup.

Definition at line 142 of file [utilities.c](#).

### 3.19.1.5 ut\_load\_from\_non\_vol()

```
void ut_load_from_non_vol (
    uint8_t index,
    float * longitude,
    float * latitude )
```

Loads position from non-volatile memory.

This function loads the longitude and latitude from non-volatile memory.

#### Parameters

<i>index</i>	The index of the memory location.
<i>longitude</i>	Pointer to store the longitude value.
<i>latitude</i>	Pointer to store the latitude value.

Definition at line 42 of file [utilities.c](#).

### 3.19.1.6 ut\_poll\_btns()

```
void ut_poll_btns ( )
```

Checks the status of each button sequentially.

Checks status of each button sequentially. Action triggers on button release. Is to be called in background via interrupt. Post-polling action takes place in the following order: MODE\_SELECT\_BTN, MEM\_SELECT\_BTN, OP\_SELECT\_BTN.

This function checks the status of each button sequentially. Action triggers on button release. It is to be called in the background via interrupt. Post-polling action takes place in the following order: MODE\_SELECT\_BTN, MEM\_SELECT\_BTN, OP\_SELECT\_BTN.

Definition at line 189 of file [utilities.c](#).

### 3.19.1.7 ut\_write\_to\_non\_vol()

```
void ut_write_to_non_vol (
    uint8_t index )
```

Writes to non-volatile memory.

This function writes data to non-volatile memory based on the provided index.

#### Parameters

<i>index</i>	The index of the memory location.
--------------	-----------------------------------

Definition at line 55 of file [utilities.c](#).

### 3.19.2 Variable Documentation

#### 3.19.2.1 ut\_lat\_mem\_floats

```
float ut_lat_mem_floats[MAX_MEM_INDEX]
```

Array to store latitude

Definition at line 8 of file [utilities.c](#).

#### 3.19.2.2 ut\_lat\_mem\_str

```
char ut_lat_mem_str[LLA_LAT_BUFFER_SIZE]
```

String to store latitude

Definition at line 10 of file [utilities.c](#).

#### 3.19.2.3 ut\_long\_mem\_floats

```
float ut_long_mem_floats[MAX_MEM_INDEX]
```

Array to store longitude

Definition at line 9 of file [utilities.c](#).

#### 3.19.2.4 ut\_long\_mem\_str

```
char ut_long_mem_str[LLA_LONG_BUFFER_SIZE]
```

String to store longitude

Definition at line 11 of file [utilities.c](#).

#### 3.19.2.5 ut\_memory\_0idx

```
uint8_t ut_memory_0idx
```

Index for memory

Definition at line 7 of file [utilities.c](#).

#### 3.19.2.6 ut\_mode

```
boolean_t ut_mode
```

Current mode

Definition at line 5 of file [utilities.c](#).

## 3.19.2.7 ut\_operation

```
uint8_t ut_operation
```

Current operation

Definition at line 6 of file [utilities.c](#).

## 3.20 utilities.c

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

```
00001 #include <avr/io.h>
00002 #include "utilities.h"
00003
00004 //global variables
00005 boolean_t ut_mode;
00006 uint8_t ut_operation;
00007 uint8_t ut_memory_0idx;
00008 float ut_lat_mem_floats[MAX_MEM_INDEX];
00009 float ut_long_mem_floats[MAX_MEM_INDEX];
00010 char ut_lat_mem_str[LLA_LAT_BUFFER_SIZE];
00011 char ut_long_mem_str[LLA_LONG_BUFFER_SIZE];
00014 //local static variables
00015 static uint16_t num_button_polls;
00016 static uint16_t btn_on_time[NUM_BUTTONS];
00017 static uint8_t btn_off_time[NUM_BUTTONS];
00018 static boolean_t prev_state[NUM_BUTTONS];
00019 static boolean_t btn_state[NUM_BUTTONS];
00021 //local functions
00031 boolean_t is_button_pressed(volatile uint8_t *port, uint8_t pin);
00032
00042 void ut_load_from_non_vol(uint8_t index, float* longitude, float* latitude){
00043     //TODO, load ith element of array from non-vol... or whole array at once if possible?
00044     *longitude = 0.0f;
00045     *latitude = 0.0f;
00046 }
00047
00055 void ut_write_to_non_vol(uint8_t index){
00056     //TODO, look at index and read from ut_long_mem_floats and ut_lat_mem_floats
00057 }
00058
00067 void ut_convert_lat_float_to_string(float lat_float, char* lat_string){
00068     //passed by value; will not change float from where function is called
00069     if (lat_float < 0){
00070         lat_string[0] = '-';
00071         lat_float *= -1; //make positive for integer math
00072     } else{
00073         lat_string[0] = '+';
00074     }
00075
00076     // Extract integer and decimal parts
00077     uint16_t integer_part = (uint16_t)(lat_float);
00078     float fractional_part = lat_float - integer_part;
00079     uint32_t decimal_part = (uint32_t)(fractional_part * 100000);
00080
00082     // Convert integer part to string
00083     lat_string[1] = '0' + ((integer_part / 10) % 10); // Tens
00084     lat_string[2] = '0' + (integer_part % 10); // Ones
00085
00086     // Decimal point
00087     lat_string[3] = '.';
00088
00089     // Convert decimal part to string
00090     lat_string[4] = '0' + ((decimal_part / 10000) % 10); // Ten-thousands
00091     lat_string[5] = '0' + ((decimal_part / 1000) % 10); // Thousands
00092     lat_string[6] = '0' + ((decimal_part / 100) % 10); // Hundreds
00093     lat_string[7] = '0' + ((decimal_part / 10) % 10); // Tens
00094     lat_string[8] = '0' + (decimal_part % 10); // Ones
00095 }
00096
00105 void ut_convert_long_float_to_string(float long_float, char* long_string){
00106     //passed by value; will not change float from where function is called
00107     if (long_float < 0){
00108         long_string[0] = '-';
00109         long_float *= -1; //make positive for integer math
00110     } else{
```

```

00111     long_string[0] = '+';
00112 }
00113
00114 // Extract integer and decimal parts
00115 uint16_t integer_part = (uint16_t)(long_float);
00116 float fractional_part = long_float - integer_part;
00117 uint32_t decimal_part = (uint32_t)(fractional_part * 100000);
00118
00119
00120 // Convert integer part to string
00121 long_string[1] = '0' + ((integer_part / 100) % 10); // Hundreds
00122 long_string[2] = '0' + ((integer_part / 10) % 10); // Tens
00123 long_string[3] = '0' + (integer_part % 10); // Ones
00124
00125 // Decimal point
00126 long_string[4] = '.';
00127
00128 // Convert decimal part to string
00129 long_string[5] = '0' + ((decimal_part / 10000) % 10); // Ten-thousands
00130 long_string[6] = '0' + ((decimal_part / 1000) % 10); // Thousands
00131 long_string[7] = '0' + ((decimal_part / 100) % 10); // Hundreds
00132 long_string[8] = '0' + ((decimal_part / 10) % 10); // Tens
00133 long_string[9] = '0' + (decimal_part % 10); // Ones
00134 }
00135
00136
00142 void ut_init()
00143 {
00144     //read from SD card
00145     for (int i = 0; i < MAX_MEM_INDEX; i++){
00146         ut_load_from_non_vol(i, ut_long_mem_floats+i, ut_lat_mem_floats+i);
00147     }
00148
00149     //initialize globals
00150     ut_mode = NAV_MODE;
00151     ut_operation = SAVE_OP;
00152     ut_memory_0idx = 0;
00153
00154     ut_convert_lat_float_to_string(ut_lat_mem_floats[ut_memory_0idx], ut_lat_mem_str);
00155     ut_convert_long_float_to_string(ut_long_mem_floats[ut_memory_0idx], ut_long_mem_str);
00156
00157     //init local static
00158     num_button_polls = 0;
00159     for (int i = 0; i < NUM_BUTTONS; i++){
00160         btn_on_time[i] = 0;
00161         btn_off_time[i] = 0;
00162         prev_state[i] = false;
00163         btn_state[i] = false;
00164     }
00165 }
00166
00167
00168 // Set PD2, PD3, PD4, PD5 pins as inputs
00169 DDRC &= ~(1 << ACTION_BTN);
00170 DDRC &= ~(1 << OP_SELECT_BTN);
00171 DDRC &= ~(1 << MEM_SELECT_BTN);
00172 DDRC &= ~(1 << MODE_SELECT_BTN);
00173
00174 // Enable pull-up resistors for PC0, PC1, PC2, PC3 pins
00175 PORTC |= (1 << ACTION_BTN);
00176 PORTC |= (1 << OP_SELECT_BTN);
00177 PORTC |= (1 << MEM_SELECT_BTN);
00178 PORTC |= (1 << MODE_SELECT_BTN);
00179
00180 return;
00181 }
00182
00189 void ut_poll_btns(){
00190     // Iterate through each button
00191     for (int i = 0; i < NUM_BUTTONS; i++){
00192         prev_state[i] = btn_state[i];
00193
00194         if (is_button_pressed(&PINC, i)) {
00195             // Increment button logic high count if pressed
00196             btn_on_time[i]++;
00197             btn_off_time[i] = 0;
00198             // Check if button press threshold is reached
00199             if (btn_on_time[i] >= ON_TIME_THRESHOLD) {
00200                 btn_state[i] = true; // Set button state to pressed
00201                 btn_on_time[i] = 0;
00202                 btn_off_time[i] = 0;
00203             }
00204         }else{
00205             btn_off_time[i]++;
00206             if (btn_off_time[i] >= RESET_TIME_THRESHOLD){
00207                 btn_state[i] = false; // Set button state to released

```

```

00209         btn_on_time[i] = 0;
00210         btn_off_time[i] = 0;
00211     }
00212 }
00213
00214
00215
00216 } //end for loop
00217
00218
00219 // Check for virtual short and take action accordingly
00220 if (!(btn_state[MODE_SELECT_BTN]) && (prev_state[MODE_SELECT_BTN])) {
00221     // Mode select button pressed
00222     ut_mode ^= 1; //toggle mode
00223 } else if ((ut_mode != STAT_MODE) && !(btn_state[MEM_SELECT_BTN]) && (prev_state[MEM_SELECT_BTN]))
{
00224     // Memory select button pressed
00225     ut_memory_0idx = (ut_memory_0idx + 1)%MAX_MEM_INDEX; //cycle memory index selected
00226     //update strings to reflect selected mem location
00227     ut_convert_lat_float_to_string(ut_lat_mem_floats[ut_memory_0idx], ut_lat_mem_str);
00228     ut_convert_long_float_to_string(ut_long_mem_floats[ut_memory_0idx], ut_long_mem_str);
00229
00230 } else if ((ut_mode != STAT_MODE) && !(btn_state[OP_SELECT_BTN]) && (prev_state[OP_SELECT_BTN])) {
00231     // Operation select button pressed
00232     ut_operation = (ut_operation + 1)%NUM_OPERATIONS; //cycle operation selected
00233
00234 } else if ((ut_mode != STAT_MODE) && !(btn_state[ACTION_BTN]) && (prev_state[ACTION_BTN])) {
00235     // Action button pressed
00236     //TODO Implement action for action button press
00237     switch (ut_operation){
00238         case SAVE_OP:
00239             //Load into global array
00240             ut_lat_mem_floats[ut_memory_0idx] = latitudeLLA_float;
00241             ut_long_mem_floats[ut_memory_0idx] = longitudeLLA_float;
00242
00243             //update string
00244             ut_convert_lat_float_to_string(ut_lat_mem_floats[ut_memory_0idx], ut_lat_mem_str);
00245             ut_convert_long_float_to_string(ut_long_mem_floats[ut_memory_0idx], ut_long_mem_str);
00246
00247             //write to SD card
00248             ut_write_to_non_vol(ut_memory_0idx);
00249
00250             break;
00251         case CLEAR_OP:
00252             //Load into global array
00253             ut_lat_mem_floats[ut_memory_0idx] = 0.0f;
00254             ut_long_mem_floats[ut_memory_0idx] = 0.0f;
00255
00256             //update string
00257             ut_convert_lat_float_to_string(ut_lat_mem_floats[ut_memory_0idx], ut_lat_mem_str);
00258             ut_convert_long_float_to_string(ut_long_mem_floats[ut_memory_0idx], ut_long_mem_str);
00259
00260             //write to SD card
00261             ut_write_to_non_vol(ut_memory_0idx);
00262
00263             break;
00264         case RESET_OP:
00265             for (int i = 0; i < MAX_MEM_INDEX; i++){
00266                 //Load into global array
00267                 ut_lat_mem_floats[i] = 0.0f;
00268                 ut_long_mem_floats[i] = 0.0f;
00269
00270                 //update string
00271                 ut_convert_lat_float_to_string(ut_lat_mem_floats[i], ut_lat_mem_str);
00272                 ut_convert_long_float_to_string(ut_long_mem_floats[i], ut_long_mem_str);
00273
00274                 //write to SD card
00275                 ut_write_to_non_vol(i);
00276             }
00277             break;
00278         default: //not reachable; error will show on screen (see main)
00279             break;
00280     }
00281 }
00282 }
00283
00284 //local function definition
00285 boolean_t is_button_pressed(volatile uint8_t *port, uint8_t pin) {
00290     // Check if the button is pressed (logic low)
00291     if (!(*port & (1 << pin))) {
00292         return true;
00293     }
00294     return false;
00295 }

```

### 3.21 G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ut/utilities.h File Reference

Header file containing common utility functions and definitions.

```
#include "ut_types.h"
#include "../nf/nf_types.h"
```

#### Macros

- `#define NOP` `asm("nop");`
- `#define STAT_MODE` 1
- `#define NAV_MODE` 0
- `#define NUM_OPERATIONS` 3
- `#define SAVE_OP` 0
- `#define CLEAR_OP` 1
- `#define RESET_OP` 2
- `#define MAX_MEM_INDEX` 10
- `#define SAVE_STR` "SAVE"
- `#define CLEAR_STR` "CLEAR"
- `#define RESET_STR` "RESET"
- `#define NUM_BUTTONS` 4
- `#define MODE_SELECT_BTN` PINC0
- `#define MEM_SELECT_BTN` PINC1
- `#define OP_SELECT_BTN` PINC2
- `#define ACTION_BTN` PINC3
- `#define ON_TIME_THRESHOLD` (`uint16_t`)100
- `#define RESET_TIME_THRESHOLD` (`uint16_t`)30

#### Functions

- void `ut_init` ()  
*Initializes the pins for buttons, loads from SD card, and initializes stored locations on startup.*
- void `ut_poll_btns` ()  
*Checks status of each button sequentially. Action triggers on button release. Is to be called in background via interrupt. Post-polling action takes place in the following order: MODE\_SELECT\_BTN, MEM\_SELECT\_BTN, OP\_SELECT\_BTN.*

#### Variables

- `boolean_t` `ut_mode`
- `uint8_t` `ut_operation`
- `uint8_t` `ut_memory_0idx`
- `float` `ut_lat_mem_floats` [`MAX_MEM_INDEX`]
- `float` `ut_long_mem_floats` [`MAX_MEM_INDEX`]
- `char` `ut_lat_mem_str` [`LLA_LAT_BUFFER_SIZE`]
- `char` `ut_long_mem_str` [`LLA_LONG_BUFFER_SIZE`]



### 3.21.1 Detailed Description

Header file containing common utility functions and definitions.

This file provides declarations for common utility functions and definitions that can be used throughout the project.

#### Version

1.0

#### Copyright

(C) 2024 Bradley Johnson and Abele Atresso

Definition in file [utilities.h](#).

### 3.21.2 Macro Definition Documentation

#### 3.21.2.1 ACTION\_BTN

```
#define ACTION_BTN PINC3
```

Action button pin.

Definition at line 36 of file [utilities.h](#).

#### 3.21.2.2 CLEAR\_OP

```
#define CLEAR_OP 1
```

Clear operation index.

Definition at line 24 of file [utilities.h](#).

#### 3.21.2.3 CLEAR\_STR

```
#define CLEAR_STR "CLEAR"
```

Clear operation string.

Definition at line 29 of file [utilities.h](#).

#### 3.21.2.4 MAX\_MEM\_INDEX

```
#define MAX_MEM_INDEX 10
```

Maximum memory index.

Definition at line 27 of file [utilities.h](#).

### 3.21.2.5 MEM\_SELECT\_BTN

```
#define MEM_SELECT_BTN PINC1
```

Memory select button pin.

Definition at line 34 of file [utilities.h](#).

### 3.21.2.6 MODE\_SELECT\_BTN

```
#define MODE_SELECT_BTN PINC0
```

Mode select button pin.

Definition at line 33 of file [utilities.h](#).

### 3.21.2.7 NAV\_MODE

```
#define NAV_MODE 0
```

Mode indicating navigation.

Definition at line 20 of file [utilities.h](#).

### 3.21.2.8 NOP

```
#define NOP asm("nop");
```

No operation macro.

Definition at line 17 of file [utilities.h](#).

### 3.21.2.9 NUM\_BUTTONS

```
#define NUM_BUTTONS 4
```

Number of buttons.

Definition at line 32 of file [utilities.h](#).

### 3.21.2.10 NUM\_OPERATIONS

```
#define NUM_OPERATIONS 3
```

Number of available operations.

Definition at line 22 of file [utilities.h](#).

### 3.21.2.11 ON\_TIME\_THRESHHOLD

```
#define ON_TIME_THRESHHOLD (uint16_t)100
```

Button press threshold time.

Definition at line 38 of file [utilities.h](#).

### 3.21.2.12 OP\_SELECT\_BTN

```
#define OP_SELECT_BTN PINC2
```

Operation select button pin.

Definition at line 35 of file [utilities.h](#).

### 3.21.2.13 RESET\_OP

```
#define RESET_OP 2
```

Reset operation index.

Definition at line 25 of file [utilities.h](#).

### 3.21.2.14 RESET\_STR

```
#define RESET_STR "RESET"
```

Reset operation string.

Definition at line 30 of file [utilities.h](#).

### 3.21.2.15 RESET\_TIME\_THRESHHOLD

```
#define RESET_TIME_THRESHHOLD (uint16_t)30
```

Button release threshold time.

Definition at line 39 of file [utilities.h](#).

### 3.21.2.16 SAVE\_OP

```
#define SAVE_OP 0
```

Save operation index.

Definition at line 23 of file [utilities.h](#).

### 3.21.2.17 SAVE\_STR

```
#define SAVE_STR " SAVE"
```

Save operation string.

Definition at line 28 of file [utilities.h](#).

### 3.21.2.18 STAT\_MODE

```
#define STAT_MODE 1
```

Mode indicating status.

Definition at line 19 of file [utilities.h](#).

## 3.21.3 Function Documentation

### 3.21.3.1 ut\_init()

```
void ut_init ( )
```

Initializes the pins for buttons, loads from SD card, and initializes stored locations on startup.

This function initializes the pins for buttons, loads data from an SD card, and initializes stored locations on startup.

Definition at line 142 of file [utilities.c](#).

### 3.21.3.2 ut\_poll\_btns()

```
void ut_poll_btns ( )
```

Checks status of each button sequentially. Action triggers on button release. Is to be called in background via interrupt. Post-polling action takes place in the following order: MODE\_SELECT\_BTN, MEM\_SELECT\_BTN, OP↔\_SELECT\_BTN.

Checks status of each button sequentially. Action triggers on button release. Is to be called in background via interrupt. Post-polling action takes place in the following order: MODE\_SELECT\_BTN, MEM\_SELECT\_BTN, OP↔\_SELECT\_BTN.

This function checks the status of each button sequentially. Action triggers on button release. It is to be called in the background via interrupt. Post-polling action takes place in the following order: MODE\_SELECT\_BTN, MEM↔\_SELECT\_BTN, OP\_SELECT\_BTN.

Definition at line 189 of file [utilities.c](#).

### 3.21.4 Variable Documentation

#### 3.21.4.1 ut\_lat\_mem\_floats

```
float ut_lat_mem_floats[MAX_MEM_INDEX] [extern]
```

Array to store latitude memory floats.

Array to store latitude

Definition at line 8 of file [utilities.c](#).

#### 3.21.4.2 ut\_lat\_mem\_str

```
char ut_lat_mem_str[LLA_LAT_BUFFER_SIZE] [extern]
```

Array to store latitude memory strings.

String to store latitude

Definition at line 10 of file [utilities.c](#).

#### 3.21.4.3 ut\_long\_mem\_floats

```
float ut_long_mem_floats[MAX_MEM_INDEX] [extern]
```

Array to store longitude memory floats.

Array to store longitude

Definition at line 9 of file [utilities.c](#).

#### 3.21.4.4 ut\_long\_mem\_str

```
char ut_long_mem_str[LLA_LONG_BUFFER_SIZE] [extern]
```

Array to store longitude memory strings.

String to store longitude

Definition at line 11 of file [utilities.c](#).

#### 3.21.4.5 ut\_memory\_Oidx

```
uint8_t ut_memory_Oidx [extern]
```

Current memory index.

Index for memory

Definition at line 7 of file [utilities.c](#).

### 3.21.4.6 ut\_mode

```
boolean_t ut_mode [extern]
```

Current mode indicator.

Current mode

Definition at line 5 of file [utilities.c](#).

### 3.21.4.7 ut\_operation

```
uint8_t ut_operation [extern]
```

Current operation index.

Current operation

Definition at line 6 of file [utilities.c](#).

## 3.22 utilities.h

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

```
00001
00011 #ifndef UTILITIES_H
00012 #define UTILITIES_H
00013
00014 #include "ut_types.h"
00015 #include "../nf/nf_types.h"
00016
00017 #define NOP asm("nop");
00019 #define STAT_MODE 1
00020 #define NAV_MODE 0
00022 #define NUM_OPERATIONS 3
00023 #define SAVE_OP 0
00024 #define CLEAR_OP 1
00025 #define RESET_OP 2
00027 #define MAX_MEM_INDEX 10
00028 #define SAVE_STR "SAVE"
00029 #define CLEAR_STR "CLEAR"
00030 #define RESET_STR "RESET"
00032 #define NUM_BUTTONS 4
00033 #define MODE_SELECT_BTN PINC0
00034 #define MEM_SELECT_BTN PINC1
00035 #define OP_SELECT_BTN PINC2
00036 #define ACTION_BTN PINC3
00038 #define ON_TIME_THRESHOLD (uint16_t)100
00039 #define RESET_TIME_THRESHOLD (uint16_t)30
00041 extern boolean_t ut_mode;
00042 extern uint8_t ut_operation;
00043 extern uint8_t ut_memory_0idx;
00044 extern float ut_lat_mem_floats[MAX_MEM_INDEX];
00045 extern float ut_long_mem_floats[MAX_MEM_INDEX];
00046 extern char ut_lat_mem_str[LLA_LAT_BUFFER_SIZE];
00047 extern char ut_long_mem_str[LLA_LONG_BUFFER_SIZE];
00052 void ut_init();
00053
00059 void ut_poll_btns();
00060
00061
00062 #endif /* UTILITIES_H */
```

# Index

ACTION\_BTN  
    utilities.h, 49  
altitudeLLA\_float  
    nf.c, 21  
    nf\_types.h, 35  
altitudeLLA\_str  
    nf.c, 21  
    nf\_types.h, 35  
  
boolean\_t  
    ut\_types.h, 39  
  
CLEAR\_OP  
    utilities.h, 49  
CLEAR\_STR  
    utilities.h, 49  
convertNMEAtoLLA  
    nf.c, 20  
    nf.h, 30  
  
ds.c  
    ds\_clear, 5  
    ds\_init, 5  
    ds\_print\_string, 5  
ds.h  
    ds\_clear, 8  
    ds\_init, 8  
    ds\_print\_string, 8  
    MAX\_COL, 7  
    MAX\_ROWS, 7  
    SPACES, 7  
ds\_clear  
    ds.c, 5  
    ds.h, 8  
ds\_init  
    ds.c, 5  
    ds.h, 8  
ds\_print\_string  
    ds.c, 5  
    ds.h, 8  
  
ew\_indicator  
    nf.c, 21  
    nf\_types.h, 35  
  
F\_CPU  
    main.c, 13  
    nf.c, 18  
false  
    ut\_types.h, 39  
  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ds/ds.c, 5, 6  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ds/ds.h, 6, 9  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ir/ir.c, 9, 10  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ir/ir.h, 11, 13  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/main.c, 13,  
    15  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/nf/nf.c, 17,  
    24  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/nf/nf.h, 28,  
    31  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/nf/nf\_types.h,  
    32, 38  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ut/ut\_types.h,  
    38, 40  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ut/utilities.c,  
    40, 45  
G:/Project\_Files/WayFindX/wfx\_sw/wfx\_sw/ut/utilities.h,  
    48, 54  
get\_serial\_char  
    nf.c, 20  
    nf.h, 30  
GGA\_ALTITUDE\_BUFFER\_SIZE  
    nf\_types.h, 32  
GGA\_HDOP\_BUFFER\_SIZE  
    nf\_types.h, 32  
GGA\_INDICATOR\_SIZE  
    nf\_types.h, 33  
GGA\_LAT\_BUFFER\_SIZE  
    nf\_types.h, 33  
GGA\_LONG\_BUFFER\_SIZE  
    nf\_types.h, 33  
GGA\_NUM\_INDICATORS\_ACTIVE  
    nf\_types.h, 33  
GGA\_NUM\_ITEMS\_ACTIVE  
    nf\_types.h, 33  
GGA\_SIZE  
    nf\_types.h, 33  
GGA\_SV\_USD\_BUFFER\_SIZE  
    nf\_types.h, 34  
GGA\_TYPE  
    nf.c, 18  
GGA.UTC\_BUFFER\_SIZE  
    nf\_types.h, 34  
GLL\_TYPE  
    nf.c, 19  
GSA\_TYPE  
    nf.c, 19  
GSV\_TYPE  
    nf.c, 19

- hdop
  - nf.c, [22](#)
  - nf\_types.h, [35](#)
- int8\_t
  - ut\_types.h, [39](#)
- ir.c
  - ir\_init, [9](#)
  - ir\_sec\_counter, [10](#)
  - ir\_trigger\_1hz\_flag\_g, [10](#)
  - ISR, [9](#), [10](#)
- ir.h
  - ir\_init, [12](#)
  - ir\_sec\_counter, [12](#)
  - ir\_trigger\_1hz\_flag\_g, [12](#)
- ir\_init
  - ir.c, [9](#)
  - ir.h, [12](#)
- ir\_sec\_counter
  - ir.c, [10](#)
  - ir.h, [12](#)
- ir\_trigger\_1hz\_flag\_g
  - ir.c, [10](#)
  - ir.h, [12](#)
- is\_button\_pressed
  - utilities.c, [41](#)
- ISR
  - ir.c, [9](#), [10](#)
- latitude
  - nf.c, [22](#)
  - nf\_types.h, [35](#)
- latitudeLLA\_float
  - nf.c, [22](#)
  - nf\_types.h, [35](#)
- latitudeLLA\_str
  - nf.c, [22](#)
  - nf\_types.h, [36](#)
- LLA\_ALT\_BUFFER\_SIZE
  - nf\_types.h, [34](#)
- LLA\_LAT\_BUFFER\_SIZE
  - nf\_types.h, [34](#)
- LLA\_LONG\_BUFFER\_SIZE
  - nf\_types.h, [34](#)
- longitude
  - nf.c, [22](#)
  - nf\_types.h, [36](#)
- longitudeLLA\_float
  - nf.c, [22](#)
  - nf\_types.h, [36](#)
- longitudeLLA\_str
  - nf.c, [23](#)
  - nf\_types.h, [36](#)
- main
  - main.c, [14](#)
- main.c
  - F\_CPU, [13](#)
  - main, [14](#)
  - startup, [14](#)
  - task\_1hz, [14](#)
  - update\_display, [14](#)
- MAX\_COL
  - ds.h, [7](#)
- MAX\_MEM\_INDEX
  - utilities.h, [49](#)
- MAX\_ROWS
  - ds.h, [7](#)
- MEM\_SELECT\_BTN
  - utilities.h, [49](#)
- MODE\_SELECT\_BTN
  - utilities.h, [50](#)
- msl\_altitude
  - nf.c, [23](#)
  - nf\_types.h, [36](#)
- MSS\_TYPE
  - nf.c, [19](#)
- NAV\_MODE
  - utilities.h, [50](#)
- nf.c
  - altitudeLLA\_float, [21](#)
  - altitudeLLA\_str, [21](#)
  - convertNMEAtoLLA, [20](#)
  - ew\_indicator, [21](#)
  - F\_CPU, [18](#)
  - get\_serial\_char, [20](#)
  - GGA\_TYPE, [18](#)
  - GLL\_TYPE, [19](#)
  - GSA\_TYPE, [19](#)
  - GSV\_TYPE, [19](#)
  - hdop, [22](#)
  - latitude, [22](#)
  - latitudeLLA\_float, [22](#)
  - latitudeLLA\_str, [22](#)
  - longitude, [22](#)
  - longitudeLLA\_float, [22](#)
  - longitudeLLA\_str, [23](#)
  - msl\_altitude, [23](#)
  - MSS\_TYPE, [19](#)
  - nf\_clear\_nav\_strings, [20](#)
  - nf\_init, [21](#)
  - NMEA\_MSG\_ID\_SIZE, [19](#)
  - ns\_indicator, [23](#)
  - OK\_TO\_SEND\_TYPE, [19](#)
  - position\_fix\_indicator, [23](#)
  - read\_nmea\_msg\_raw, [21](#)
  - RMC\_TYPE, [19](#)
  - satellites\_used, [23](#)
  - speed, [23](#)
  - UART\_BAUD\_RATE, [19](#)
  - utc\_time, [24](#)
  - VTG\_TYPE, [20](#)
  - ZDA\_TYPE, [20](#)
- nf.h
  - convertNMEAtoLLA, [30](#)
  - get\_serial\_char, [30](#)
  - nf\_init, [31](#)



- NF\_INIT\_FAILURE, 29
- NF\_INIT\_SUCCESS, 29
- read\_nmea\_msg\_raw, 31
- nf\_clear\_nav\_strings
  - nf.c, 20
- nf\_init
  - nf.c, 21
  - nf.h, 31
- NF\_INIT\_FAILURE
  - nf.h, 29
- NF\_INIT\_SUCCESS
  - nf.h, 29
- nf\_types.h
  - altitudeLLA\_float, 35
  - altitudeLLA\_str, 35
  - ew\_indicator, 35
  - GGA\_ALTITUDE\_BUFFER\_SIZE, 32
  - GGA\_HDOP\_BUFFER\_SIZE, 32
  - GGA\_INDICATOR\_SIZE, 33
  - GGA\_LAT\_BUFFER\_SIZE, 33
  - GGA\_LONG\_BUFFER\_SIZE, 33
  - GGA\_NUM\_INDICATORS\_ACTIVE, 33
  - GGA\_NUM\_ITEMS\_ACTIVE, 33
  - GGA\_SIZE, 33
  - GGA\_SV\_USD\_BUFFER\_SIZE, 34
  - GGA\_UTC\_BUFFER\_SIZE, 34
  - hdop, 35
  - latitude, 35
  - latitudeLLA\_float, 35
  - latitudeLLA\_str, 36
  - LLA\_ALT\_BUFFER\_SIZE, 34
  - LLA\_LAT\_BUFFER\_SIZE, 34
  - LLA\_LONG\_BUFFER\_SIZE, 34
  - longitude, 36
  - longitudeLLA\_float, 36
  - longitudeLLA\_str, 36
  - msl\_altitude, 36
  - ns\_indicator, 36
  - position\_fix\_indicator, 37
  - satellites\_used, 37
  - speed, 37
  - utc\_time, 37
  - VTG\_SPEED\_BUFFER\_SIZE, 34
- NMEA\_MSG\_ID\_SIZE
  - nf.c, 19
- NOP
  - utilities.h, 50
- ns\_indicator
  - nf.c, 23
  - nf\_types.h, 36
- NUM\_BUTTONS
  - utilities.h, 50
- NUM\_OPERATIONS
  - utilities.h, 50
- OK\_TO\_SEND\_TYPE
  - nf.c, 19
- ON\_TIME\_THRESHOLD
  - utilities.h, 50
- OP\_SELECT\_BTN
  - utilities.h, 51
- position\_fix\_indicator
  - nf.c, 23
  - nf\_types.h, 37
- read\_nmea\_msg\_raw
  - nf.c, 21
  - nf.h, 31
- RESET\_OP
  - utilities.h, 51
- RESET\_STR
  - utilities.h, 51
- RESET\_TIME\_THRESHOLD
  - utilities.h, 51
- RMC\_TYPE
  - nf.c, 19
- satellites\_used
  - nf.c, 23
  - nf\_types.h, 37
- SAVE\_OP
  - utilities.h, 51
- SAVE\_STR
  - utilities.h, 51
- SPACES
  - ds.h, 7
- speed
  - nf.c, 23
  - nf\_types.h, 37
- startup
  - main.c, 14
- STAT\_MODE
  - utilities.h, 52
- task\_1hz
  - main.c, 14
- true
  - ut\_types.h, 39
- UART\_BAUD\_RATE
  - nf.c, 19
- uint16\_t
  - ut\_types.h, 40
- uint8\_t
  - ut\_types.h, 40
- update\_display
  - main.c, 14
- ut\_convert\_lat\_float\_to\_string
  - utilities.c, 42
- ut\_convert\_long\_float\_to\_string
  - utilities.c, 42
- ut\_init
  - utilities.c, 42
  - utilities.h, 52
- ut\_lat\_mem\_floats
  - utilities.c, 44
  - utilities.h, 53

- ut\_lat\_mem\_str
  - utilities.c, [44](#)
  - utilities.h, [53](#)
- ut\_load\_from\_non\_vol
  - utilities.c, [42](#)
- ut\_long\_mem\_floats
  - utilities.c, [44](#)
  - utilities.h, [53](#)
- ut\_long\_mem\_str
  - utilities.c, [44](#)
  - utilities.h, [53](#)
- ut\_memory\_0idx
  - utilities.c, [44](#)
  - utilities.h, [53](#)
- ut\_mode
  - utilities.c, [44](#)
  - utilities.h, [53](#)
- ut\_operation
  - utilities.c, [44](#)
  - utilities.h, [54](#)
- ut\_poll\_btns
  - utilities.c, [43](#)
  - utilities.h, [52](#)
- ut\_types.h
  - boolean\_t, [39](#)
  - false, [39](#)
  - int8\_t, [39](#)
  - true, [39](#)
  - uint16\_t, [40](#)
  - uint8\_t, [40](#)
- ut\_write\_to\_non\_vol
  - utilities.c, [43](#)
- utc\_time
  - nf.c, [24](#)
  - nf\_types.h, [37](#)
- utilities.c
  - is\_button\_pressed, [41](#)
  - ut\_convert\_lat\_float\_to\_string, [42](#)
  - ut\_convert\_long\_float\_to\_string, [42](#)
  - ut\_init, [42](#)
  - ut\_lat\_mem\_floats, [44](#)
  - ut\_lat\_mem\_str, [44](#)
  - ut\_load\_from\_non\_vol, [42](#)
  - ut\_long\_mem\_floats, [44](#)
  - ut\_long\_mem\_str, [44](#)
  - ut\_memory\_0idx, [44](#)
  - ut\_mode, [44](#)
  - ut\_operation, [44](#)
  - ut\_poll\_btns, [43](#)
  - ut\_write\_to\_non\_vol, [43](#)
- utilities.h
  - ACTION\_BTN, [49](#)
  - CLEAR\_OP, [49](#)
  - CLEAR\_STR, [49](#)
  - MAX\_MEM\_INDEX, [49](#)
  - MEM\_SELECT\_BTN, [49](#)
  - MODE\_SELECT\_BTN, [50](#)
  - NAV\_MODE, [50](#)
  - NOP, [50](#)
  - NUM\_BUTTONS, [50](#)
  - NUM\_OPERATIONS, [50](#)
  - ON\_TIME\_THRESHOLD, [50](#)
  - OP\_SELECT\_BTN, [51](#)
  - RESET\_OP, [51](#)
  - RESET\_STR, [51](#)
  - RESET\_TIME\_THRESHOLD, [51](#)
  - SAVE\_OP, [51](#)
  - SAVE\_STR, [51](#)
  - STAT\_MODE, [52](#)
  - ut\_init, [52](#)
  - ut\_lat\_mem\_floats, [53](#)
  - ut\_lat\_mem\_str, [53](#)
  - ut\_long\_mem\_floats, [53](#)
  - ut\_long\_mem\_str, [53](#)
  - ut\_memory\_0idx, [53](#)
  - ut\_mode, [53](#)
  - ut\_operation, [54](#)
  - ut\_poll\_btns, [52](#)
- VTG\_SPEED\_BUFER\_SIZE
  - nf\_types.h, [34](#)
- VTG\_TYPE
  - nf.c, [20](#)
- WayFindX, [1](#)
- ZDA\_TYPE
  - nf.c, [20](#)