## Statics Strings STM32F1XX

Generated by Doxygen 1.8.18

1 Static Strings				1
1.1 Features:		 		1
1.2 Getting Started		 		1
1.2.1 Suggested names		 		1
1.2.2 First of all initialize the library		 		2
1.2.3 Creating a string		 		2
1.2.4 Also a string can created this way		 		2
1.2.5 Split a local scope string		 		2
1.2.6 Getting a substring		 		2
1.2.7 Concatenate two strings and search for a substring and a character in the result		 		2
1.2.8 Compare two equals and non equals strings		 		3
1.2.9 Transform a integer and a float to a string		 		3
1.2.10 Copy, move and clone a string		 		3
1.2.11 Concatenate and clean two strings		 		4
1.2.12 Also can be used		 		4
1.2.13 Concatenate multiple strings		 		4
1.2.14 Also can be used		 		4
1.2.15 Configure quantity and size of the memory arrays		 		5
2 Module Index				7
2.1 Modules		 		7
3 Data Structure Index				9
3.1 Data Structures	٠	 		9
4 File Index				11
4.1 File List		 		11
5 Module Documentation				13
5.1 String types size and quantity		 		13
5.1.1 Detailed Description		 		13
5.2 String types		 		14
5.2.1 Detailed Description		 		14
5.3 String status		 		15
5.3.1 Detailed Description		 		15
5.4 Error handling		 		16
5.4.1 Detailed Description		 		16
5.4.2 Variable Documentation		 		16
5.4.2.1 static_strings_error_code		 		16
5.5 Static memory arrays				17
5.5.1 Detailed Description		 		17
5.6 String descriptors		 		18
5.6.1 Detailed Description		 		18

6 Data Structure Documentation	19
6.1 static_strings_string_descriptor Struct Reference	19
6.1.1 Detailed Description	19
6.2 static_strings_string_splitter_parameters Struct Reference	19
6.2.1 Detailed Description	19
7 File Documentation	21
7.1 int_types.h File Reference	21
7.1.1 Detailed Description	21
7.2 static_strings.c File Reference	21
7.2.1 Detailed Description	23
7.2.2 Function Documentation	23
7.2.2.1 static_strings_allocate()	23
7.2.2.2 static_strings_clone()	24
7.2.2.3 static_strings_compare()	24
7.2.2.4 static_strings_concatenate()	25
7.2.2.5 static_strings_concatenate_all()	25
7.2.2.6 static_strings_concatenate_and_clean()	26
7.2.2.7 static_strings_concatenate_and_clean_all()	26
7.2.2.8 static_strings_concatenate_and_clean_both()	26
7.2.2.9 static_strings_contains_char()	27
7.2.2.10 static_strings_contains_string()	27
7.2.2.11 static_strings_copy()	28
7.2.2.12 static_strings_create_custom_string()	28
7.2.2.13 static_strings_deallocate()	29
7.2.2.14 static_strings_double_to_string()	29
7.2.2.15 static_strings_float_to_string()	29
7.2.2.16 static_strings_get_string_max_length()	30
7.2.2.17 static_strings_init()	30
7.2.2.18 static_strings_int16_to_string()	30
7.2.2.19 static_strings_int32_to_string()	31
7.2.2.20 static_strings_int8_to_string()	31
7.2.2.21 static_strings_is_line()	32
7.2.2.22 static_strings_move()	32
7.2.2.23 static_strings_save()	32
7.2.2.24 static_strings_string_splitter_get_next_token()	33
7.2.2.25 static_strings_string_splitter_set_parameters()	33
7.2.2.26 static_strings_strlen()	34
7.2.2.27 static_strings_substring()	34
7.2.2.28 static_strings_uint16_to_string()	35
7.2.2.29 static_strings_uint32_to_string()	35
7.2.2.30 static_strings_uint8_to_string()	35

55

7.2.3 Variable Documentation	. 36
7.2.3.1 static_strings_string_splitter	. 36
7.3 static_strings.h File Reference	. 36
7.3.1 Detailed Description	. 39
7.3.2 Function Documentation	. 39
7.3.2.1 static_strings_allocate()	. 40
7.3.2.2 static_strings_clone()	. 40
7.3.2.3 static_strings_compare()	. 40
7.3.2.4 static_strings_concatenate()	. 41
7.3.2.5 static_strings_concatenate_all()	. 41
7.3.2.6 static_strings_concatenate_and_clean()	. 42
7.3.2.7 static_strings_concatenate_and_clean_all()	. 42
7.3.2.8 static_strings_concatenate_and_clean_both()	. 43
7.3.2.9 static_strings_contains_char()	. 43
7.3.2.10 static_strings_contains_string()	. 43
7.3.2.11 static_strings_copy()	. 45
7.3.2.12 static_strings_create_custom_string()	. 45
7.3.2.13 static_strings_deallocate()	. 46
7.3.2.14 static_strings_double_to_string()	. 46
7.3.2.15 static_strings_float_to_string()	. 46
7.3.2.16 static_strings_get_string_max_length()	. 47
7.3.2.17 static_strings_init()	. 47
7.3.2.18 static_strings_int16_to_string()	. 47
7.3.2.19 static_strings_int32_to_string()	. 48
7.3.2.20 static_strings_int8_to_string()	. 48
7.3.2.21 static_strings_is_line()	. 49
7.3.2.22 static_strings_move()	. 49
7.3.2.23 static_strings_save()	. 49
7.3.2.24 static_strings_string_splitter_get_next_token()	. 50
7.3.2.25 static_strings_string_splitter_set_parameters()	. 50
7.3.2.26 static_strings_strlen()	. 51
7.3.2.27 static_strings_substring()	. 51
7.3.2.28 static_strings_uint16_to_string()	. 52
7.3.2.29 static_strings_uint32_to_string()	. 52
7.3.2.30 static_strings_uint8_to_string()	. 52
7.3.3 Variable Documentation	. 53
7.3.3.1 static_strings_string_splitter	

Index

# **Static Strings**

## 1.1 Features:

- Developed for the STM32F103.
- · Global scope strings.
- · No dynamic memory allocation.
- · Customizable quantity and size of string types.
- Create custom string function to create local scope strings.
- · String length function.
- String can be \0 terminated and \r\n terminated.
- · String split function.
- Fast string creation with save.
- · Low level string creation with allocate.
- · Reusable memory with deallocate.
- is\_line function.
- Substring, concatenate, concatenate and clean, concatenate all.
- Contains string, contains char and compare function.
- · Transforms integers and floats to strings
- · Get string maximum length.

## 1.2 Getting Started

## 1.2.1 Suggested names

2 Static Strings

## 1.2.2 First of all initialize the library

```
static_strings_init();
```

## 1.2.3 Creating a string

```
uint8_t test_memory[] = "Hello Word\r\n";
static_strings_string_descriptor *test = static_strings_save(test_memory);
if(test == NULL){
    Error Handling.
}
else{
    Some work.
    static_strings_deallocate(test);
}
```

DON'T FORGET TO DEALLOCATE AFTER USING.

## 1.2.4 Also a string can created this way

```
#include "string.h"
uint8_t test_memory[] = "Hello Word\r\n";
uint16_t test_length = static_strings_strlen(test_memory);
static_strings_string_descriptor *test = static_strings_allocate(test_length);
if(test == NULL){
    Error Handling.
}
else{
    memcpy(test->string,test_memory,test_length);
    test->length = test_length;
    Some work.
    static_strings_deallocate(test);
}
```

DON'T FORGET TO DEALLOCATE AFTER USING.

## 1.2.5 Split a local scope string

```
uint8_t split_memory[10] = "123,56,8\r\n";
static_strings_string_descriptor split.
static_strings_create_custom_string(&split,split_memory);
static_strings_string_descriptor *token;
static_strings_string_splitter_set_parameters(split,',');
while(static_strings_string_splitter_get_next_token(&token)) {
    HAL_UART_Transmit(&huart1,token->string,token->length,HAL_MAX_DELAY);
}
```

## 1.2.6 Getting a substring

```
uint8_t custom[10] = "123,56,89\0";
static_strings_create_custom_string(string_descriptor,custom);
static_strings_string_descriptor *substring = static_strings_substring(string_descriptor,2,8);
if(substring != NULL) {
    HAL_UART_Transmit(&huart1, substring->string, substring->length, HAL_MAX_DELAY);
    static_strings_deallocate(substring);
}
```

### 1.2.7 Concatenate two strings and search for a substring and a character in the result

```
uint8_t concatenate_at_memory[] = "Hello \0";
static_strings_string_descriptor concatenate_at;
static_strings_create_custom_string(&concatenate_at,concatenate_at_memory);
uint8_t concatenate_memory[] = "World\r\n";
static_strings_string_descriptor concatenate;
static_strings_create_custom_string(&concatenate,concatenate_memory);
static_strings_string_descriptor *concatenated;
concatenated = static_strings_concatenate(&concatenate_at,&concatenate);
if (concatenated != NULL) {
    HAL_UART_Transmit(&huart1,concatenated->string,concatenated->length,HAL_MAX_DELAY);
    if(static_strings_contains_string(concatenated,&concatenate_at)) {
        HAL_UART_Transmit(&huart1,(uint8_t *)"1\r\n",3,HAL_MAX_DELAY);
    }
}
```

1.2 Getting Started 3

```
else{
    HAL_UART_Transmit(&huart1, (uint8_t *)"0\r\n",3,HAL_MAX_DELAY);
}
if(static_strings_contains_string(concatenated,'W')) {
    HAL_UART_Transmit(&huart1, (uint8_t *)"1\r\n",3,HAL_MAX_DELAY);
}
else{
    HAL_UART_Transmit(&huart1, (uint8_t *)"0\r\n",3,HAL_MAX_DELAY);
}
static_strings_deallocate(concatenated);
```

## 1.2.8 Compare two equals and non equals strings

```
uint8_t equal_a_memory[] = "Hall\0";
static_strings_string_descriptor equal_a;
uint8_t equal_b_memory[] = "Hall\0";
static_strings_string_descriptor equal_b;
uint8_t non_equal_memory[] = "oil\0";
static_strings_string_descriptor non_equal;
static_strings_create_custom_string(&equal_a, equal_a_memory);
static_strings_create_custom_string(&equal_b, equal_b_memory);
static_strings_create_custom_string(&non_equal, non_equal_memory);
if(static_strings_create_custom_string(&non_equal, non_equal_memory);
if(static_strings_compare(&equal_a, &equal_b)){
   HAL_UART_Transmit(&huart1, (uint8_t *)"1\r\n", 3, HAL_MAX_DELAY);
}
else{
   HAL_UART_Transmit(&huart1, (uint8_t *)"0\r\n", 3, HAL_MAX_DELAY);
}
else{
   HAL_UART_Transmit(&huart1, (uint8_t *)"1\r\n", 3, HAL_MAX_DELAY);
}
else{
   HAL_UART_Transmit(&huart1, (uint8_t *)"0\r\n", 3, HAL_MAX_DELAY);
}
```

## 1.2.9 Transform a integer and a float to a string

```
static_strings_string_descriptor *var_string;
uint8_t uint8 = 200;
var_string = static_strings_uint8_to_string(uint8);
if(var_string != NULL) {
    HAL_UART_Transmit(&huart1,var_string->string,var_string->length,HAL_MAX_DELAY);
    static_strings_deallocate(var_string);
}
float float_number = 19.60232;
var_string = static_strings_float_to_string(float_number);
if(var_string != NULL) {
    HAL_UART_Transmit(&huart1,var_string->string,var_string->length,HAL_MAX_DELAY);
    static_strings_deallocate(var_string);
}
```

## 1.2.10 Copy, move and clone a string

```
static_strings_string_descriptor *copy_test_source_string = static_strings_save((uint8_t *)"I am a copy
                     test\r\n");
             if (copy_test_source_string != NULL) {
                         static_strings_string_descriptor *copy_test_target_string = static_strings_allocate(100);
                         if(static_strings_copy(copy_test_target_string,copy_test_source_string,0) != NULL){
                     HAL_UART_Transmit(&huart1,copy_test_target_string->string,copy_test_target_string->length,HAL_MAX_DELAY);
                                     static_strings_deallocate(copy_test_source_string);
                                      static_strings_deallocate(copy_test_target_string);
static\_strings\_string\_descriptor \star move\_test\_source\_string = static\_strings\_save((uint8\_t \star)"I am a move\_test\_source\_strings\_save((uint8\_t \star)"I am a move\_test\_source\_strings\_save((uint8\_t \star))"I am a move\_strings\_save((uint8\_t \star))"I am a move\_test\_source\_strings\_save((uint8\_t \star))"I am a move\_strings\_save((uint8\_t \star))"I am a move\_strings\_save((uint8\_
                     test\r\n");
if(copy_test_source_string != NULL) {
            static_strings_string_descriptor *move_test_target_string = static_strings_allocate(100);
            *move_test_target_string->string = '.';
            if(static_strings_move(move_test_target_string,move_test_source_string,1) != NULL){
                     HAL_UART_Transmit(&huart1, move_test_target_string->string, move_test_target_string->length, HAL_MAX_DELAY);
                         static strings deallocate (move test source string);
}
```

4 Static Strings

## 1.2.11 Concatenate and clean two strings

```
static_strings_string_descriptor *concatenate_at = static_strings_save((uint8_t *)"I am a ");
static_strings_string_descriptor *concatenate = static_strings_save((uint8_t *)"concatenate test\r\n");
if(concatenate_at != NULL && concatenate != NULL) {
    static_strings_string_descriptor *concatenate_string =
        static_strings_concatenate_and_clean(concatenate_at,concatenate);
    if(concatenated_string != NULL) {
        HAL_UART_Transmit(&huart1,concatenated_string->string,concatenated_string->length,HAL_MAX_DELAY);
        static_strings_deallocate(concatenate);
        static_strings_deallocate(concatenated_string);
    }
}
```

### 1.2.12 Also can be used

```
static_strings_string_descriptor *concatenate_at = static_strings_save((uint8_t *)"I am a ");
static_strings_string_descriptor *concatenate = static_strings_save((uint8_t *)"concatenate test\r\n");
if(concatenate_at != NULL && concatenate != NULL){
    static_strings_string_descriptor *concatenate_string =
        static_strings_concatenate_and_clean_both(concatenate_at,concatenate);
    if(concatenated_string != NULL) {
        HAL_UART_Transmit(&huart1,concatenated_string->string,concatenated_string->length,HAL_MAX_DELAY);
        static_strings_deallocate(concatenated_string);
    }
}
```

### 1.2.13 Concatenate multiple strings

```
static_strings_string_descriptor *one = static_strings_save((uint8_t *)"I am a ");
static_strings_string_descriptor *two = static_strings_save((uint8_t *)"concatenate all ");
static_strings_string_descriptor *three = static_strings_save((uint8_t *)"test\r\n");
if(one != NULL && two != NULL && three != NULL) {
    static_strings_string_descriptor *concatenated_string = static_strings_concatenate_all(3,one,two,three);
    if(concatenated_string != NULL) {

        HAL_UART_Transmit(&huart1,concatenated_string->string,concatenated_string->length,HAL_MAX_DELAY);
        static_strings_deallocate(one);
        static_strings_deallocate(two);
        static_strings_deallocate(three);
        static_strings_deallocate(concatenated_string);
    }
}
```

### 1.2.14 Also can be used

```
static_strings_string_descriptor *one = static_strings_save((uint8_t *)"I am a ");
static_strings_string_descriptor *two = static_strings_save((uint8_t *)"concatenate all ");
static_strings_string_descriptor *three = static_strings_save((uint8_t *)"test\r\n");
if(one != NULL && two != NULL && three != NULL) {
    static_strings_string_descriptor *concatenated_string =
        static_strings_concatenate_and_clean_all(3,one,two,three);
    if(concatenated_string != NULL) {
        HAL_UART_Transmit(&huart1,concatenated_string->string,concatenated_string->length,HAL_MAX_DELAY);
        static_strings_deallocate(concatenated_string);
    }
}
```

1.2 Getting Started 5

## 1.2.15 Configure quantity and size of the memory arrays

```
Just edit these constants in static_strings.h

#define STATIC_STRINGS_VERY_SHORT_STRING_SIZE 50

#define STATIC_STRINGS_VERY_SHORT_STRING_QUANTITY 10

#define STATIC_STRINGS_SHORT_STRING_SIZE 100

#define STATIC_STRINGS_SHORT_STRING_QUANTITY 6

#define STATIC_STRINGS_MEDIUM_STRING_QUANTITY 2

#define STATIC_STRINGS_MEDIUM_STRING_QUANTITY 2

#define STATIC_STRINGS_LONG_STRING_SIZE 500

#define STATIC_STRINGS_LONG_STRING_QUANTITY 1

#define STATIC_STRINGS_VERY_LONG_STRING_SIZE 1000

#define STATIC_STRINGS_VERY_LONG_STRING_OUANTITY 1
    #define STATIC_STRINGS_VERY_LONG_STRING_QUANTITY 1
```

6 Static Strings

# **Module Index**

## 2.1 Modules

Here is a list of all modules:

String types size and quantity	 . 1
String types	 . 1
String status	 . 1
Error handling	 . 1
Static memory arrays	 . 1
String descriptors	 . 1

8 Module Index

# **Data Structure Index**

## 3.1 Data Structures

Here are the data structures with brief descriptions:

static_strings_string_descriptor	
Meta data of a string	19
static_strings_string_splitter_parameters	
Definition of the structure to hold the parameters to static stirngs string splitter get next token	
function	19

10 Data Structure Index

# File Index

## 4.1 File List

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

int_types.h	
The fprintf() PRI[d,u,x,o,i,X][8,16,32] macros for 32 bits signed and unsigned integers	21
static_strings.c	
Strings allocation with static memory	21
static_strings.h	
Strings allocation with static memory	36

12 File Index

## **Module Documentation**

## 5.1 String types size and quantity

Constants to reserve a memory for the different types of strings according to their length.

### **Macros**

- #define STATIC\_STRINGS\_VERY\_SHORT\_STRING\_SIZE 50
- #define STATIC\_STRINGS\_VERY\_SHORT\_STRING\_QUANTITY 10
- #define STATIC\_STRINGS\_SHORT\_STRING\_SIZE 100
- #define STATIC\_STRINGS\_SHORT\_STRING\_QUANTITY 6
- #define STATIC\_STRINGS\_MEDIUM\_STRING\_SIZE 200
- #define STATIC\_STRINGS\_MEDIUM\_STRING\_QUANTITY 2
- #define STATIC\_STRINGS\_LONG\_STRING\_SIZE 500
- #define STATIC\_STRINGS\_LONG\_STRING\_QUANTITY 2
   #define STATIC STRINGS VERY LONG STRING SIZE 1500
- #define STATIC\_STRINGS\_VERY\_LONG\_STRING\_QUANTITY 2

## 5.1.1 Detailed Description

Constants to reserve a memory for the different types of strings according to their length.

14 Module Documentation

## 5.2 String types

Constants to identify the different types of strings according to their length.

## **Macros**

- #define STATIC\_STRINGS\_STRING\_TYPE\_VERY\_SHORT 0
- #define STATIC STRINGS STRING TYPE SHORT 1
- #define STATIC\_STRINGS\_STRING\_TYPE\_MEDIUM 2
- #define STATIC\_STRINGS\_STRING\_TYPE\_LONG 3
- #define STATIC\_STRINGS\_STRING\_TYPE\_VERY\_LONG 4
- #define STATIC\_STRINGS\_STRING\_TYPE\_CUSTOM 5

## 5.2.1 Detailed Description

Constants to identify the different types of strings according to their length.

5.3 String status

## 5.3 String status

Constants to define the status of a string.

## **Macros**

- #define STATIC\_STRINGS\_STRING\_STATUS\_DEALLOCATED 0
- #define STATIC\_STRINGS\_STRING\_STATUS\_ALLOCATED 1
- #define **STATIC\_STRINGS\_STRING\_STATUS\_CONSTANT** 2

## 5.3.1 Detailed Description

Constants to define the status of a string.

16 Module Documentation

## 5.4 Error handling

Error codes.

### **Macros**

- #define STATIC\_STRINGS\_ERROR\_CODE\_NO\_ERROR 0
- #define STATIC STRINGS ERROR CODE NO MEMORY AVAILABLE 1
- #define STATIC\_STRINGS\_ERROR\_CODE\_INVALID\_STRING 2
- #define STATIC STRINGS ERROR CODE STRING TOO LONG 3
- #define STATIC\_STRINGS\_ERROR\_CODE\_SUBSTRING\_START\_INDEX\_OUT\_OF\_RANGE 4
- #define STATIC\_STRINGS\_ERROR\_CODE\_SUBSTRING\_FINISH\_INDEX\_OUT\_OF\_RANGE 5
- #define STATIC\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW 6

## **Variables**

uint8\_t static\_strings\_error\_code
 Global variable to store error code.

## 5.4.1 Detailed Description

Error codes.

## 5.4.2 Variable Documentation

## 5.4.2.1 static\_strings\_error\_code

uint8\_t static\_strings\_error\_code

Global variable to store error code.

static strings error code

## 5.5 Static memory arrays

Static memory arrays to allocate strings.

## **Variables**

- uint8\_t static\_strings\_short\_string\_memory [STATIC\_STRINGS\_SHORT\_STRING\_QUANTITY][STA  $\leftarrow$  TIC\_STRINGS\_SHORT\_STRING\_SIZE]
- uint8\_t  $static\_strings\_medium\_string\_memory$  [STATIC\_STRINGS\_MEDIUM\_STRING\_QUANTI $\leftarrow$  TY][STATIC\_STRINGS\_MEDIUM\_STRING\_SIZE]
- uint8\_t static\_strings\_very\_long\_string\_memory [STATIC\_STRINGS\_VERY\_LONG\_STRING\_QUAN ← TITY][STATIC\_STRINGS\_VERY\_LONG\_STRING\_SIZE]

## 5.5.1 Detailed Description

Static memory arrays to allocate strings.

18 Module Documentation

## 5.6 String descriptors

Descriptors for all the string types.

## **Variables**

- static\_strings\_string\_descriptor static\_strings\_very\_short\_strings\_descriptors [STATIC\_STRINGS\_V ← ERY SHORT STRING QUANTITY]

- static\_strings\_string\_descriptor static\_strings\_long\_strings\_descriptors [STATIC\_STRINGS\_LONG\_← STRING\_QUANTITY]
- static\_strings\_string\_descriptor static\_strings\_very\_long\_strings\_descriptors [STATIC\_STRINGS\_V \( \to \) ERY\_LONG\_STRING\_QUANTITY]

## 5.6.1 Detailed Description

Descriptors for all the string types.

## **Data Structure Documentation**

## 6.1 static\_strings\_string\_descriptor Struct Reference

Meta data of a string.

```
#include <static_strings.h>
```

## **Data Fields**

- uint8 t \* string
- uint16\_t length
- uint8\_t type
- uint8\_t status

## 6.1.1 Detailed Description

Meta data of a string.

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

• static\_strings.h

## 6.2 static\_strings\_string\_splitter\_parameters Struct Reference

Definition of the structure to hold the parameters to static\_stirngs\_string\_splitter\_get\_next\_token function.

```
#include <static_strings.h>
```

## **Data Fields**

- static strings string descriptor \* string descriptor
- uint8\_t \* next\_token\_start
- uint8\_t delimiter

## 6.2.1 Detailed Description

Definition of the structure to hold the parameters to static\_stirngs\_string\_splitter\_get\_next\_token function.

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

· static\_strings.h

## **File Documentation**

## 7.1 int\_types.h File Reference

The fprintf() PRI[d,u,x,o,i,X][8,16,32] macros for 32 bits signed and unsigned integers.

## **Macros**

- #define PRId8 "hd"
- #define PRId16 "d"
- #define PRId32 "ld"
- #define PRIu8 "hu"
- #define PRIu16 "u"
- #define PRIu32 "lu"
- #define PRIx8 "hx"
- #define PRIx16 "x"
- #define PRIx32 "lx"
- #define PRIo8 "ho"
- #define PRIo16 "o"
- #define PRIo32 "lo"
- #define PRIi8 "hi"
- #define PRIi16 "i"
- #define PRIi32 "li"
- #define PRIX8 "hX"
- #define PRIX16 "X"
- #define PRIX32 "IX"

## 7.1.1 Detailed Description

The fprintf() PRI[d,u,x,o,i,X][8,16,32] macros for 32 bits signed and unsigned integers.

## 7.2 static\_strings.c File Reference

Strings allocation with static memory.

```
#include "static_strings.h"
```

22 File Documentation

### **Functions**

void static\_strings\_init ()

Link the descriptors with the arrays and initialize the status as deallocated. Also can be used to reset the state of all the string descriptors.

• int static\_strings\_get\_string\_max\_length (static\_strings\_string\_descriptor \*string)

get the maximum length allowed by the type of the string.

static\_strings\_string\_descriptor \* static\_strings\_copy (static\_strings\_string\_descriptor \*copy\_to, static\_strings\_string\_descriptor \*copy\_from, uint16\_t copy\_to\_offset)

Copy a string into another at determinate offset. Leaves intact the string values before the offset. Can throw STAT← IC\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW.

static\_strings\_string\_descriptor \* static\_strings\_move (static\_strings\_string\_descriptor \*move\_to, static\_strings\_string\_descriptor \*move\_to, static\_strings\_string\_string\_string\_string\_string\_string\_string\_string\_string\_string\_string\_string\_string\_string\_string\_string\_s

Move a string into another at determinate offset, if success the move\_to string is deallocated. Can throw STATIC\_← STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW. Leaves intact the string values before the offset.

 $\bullet \ \ static\_strings\_string\_descriptor * static\_strings\_clone \ (static\_strings\_string\_descriptor * static\_strings\_string\_descriptor * static\_strings\_string\_string\_string\_string\_strings\_string\_string\_string\_string\_strings\_string\_strings\_string\_strings\_string\_strings\_string\_strings\_string\_strings\_string\_strings\_st$ 

Clone a string into a new one.

static strings string descriptor \* static strings allocate (uint16 t string size)

Request memory for a string with its size, the user must copy the string with the descriptor and specify the size. Also see static\_strings\_save.

static\_strings\_string\_descriptor \* static\_strings\_save (uint8\_t \*string)

Calculate the string size, allocate memory, copy the string and set the size. String must end with  $\r$  or  $\0$ , if  $\$  is found but  $\$  n is not found, it is added, size of string include line ending but not  $\$ 0. Also see static\_strings\_allocate.

int static\_strings\_create\_custom\_string (static\_strings\_string\_descriptor \*string\_descriptor, uint8\_t \*string)

Bind the provided string descriptor with the data of a string. String must end with \r\n or \0.

void static\_strings\_deallocate (static\_strings\_string\_descriptor \*string\_descriptor)

Set the descriptor status as deallocated. Custom strings can't be deallocated.

• int static\_strings\_is\_line (static\_strings\_string\_descriptor \*string\_descriptor)

Look at the last two characters of a string to see if the string has a line ending \r\n.

uint16\_t static\_strings\_strlen (uint8\_t \*string)

Calculate the length of a string that ends with \r\n or \0, line ending is included in length. Maximum length is STAT← IC\_STRINGS\_VERY\_LONG\_STRING\_SIZE.

void static\_strings\_string\_splitter\_set\_parameters (static\_strings\_string\_descriptor \*string\_descriptor, uint8 t delimiter)

Set the parameters to the static\_strings\_string\_splitter\_get\_next\_token function.

• int static\_strings\_string\_splitter\_get\_next\_token (static\_strings\_string\_descriptor \*\*string\_descriptor)

Bind the provided string descriptor with the next token data. Can be placed in a while condition as it returns 1 if success or 0 if no token available and retrieves the token in the string\_descriptor parameter. If no delimiter the whole string is taken as token. The token is placed in a new string.

• static\_strings\_string\_descriptor \* static\_strings\_substring (static\_strings\_string\_descriptor \*string, uint16\_t start index, uint16\_t finish index)

Return a new string with the characters between the start\_index and the finish\_index. Not including the character at finish\_index. Returned string has to be deallocated. To get all the string from a start index use the length in the finish\_index.

static\_strings\_string\_descriptor \* static\_strings\_concatenate (static\_strings\_string\_descriptor \*concatenate ←
 \_at, static\_strings\_string\_descriptor \*concatenate)

Concatenate the second string at the end of the first in a new string.

• static\_strings\_string\_descriptor \* static\_strings\_concatenate\_and\_clean (static\_strings\_string\_descriptor \*concatenate\_at, static\_strings\_string\_descriptor \*concatenate)

Concatenate the second string at the end of the first in a new string and deallocate the concatenate at parameter if success.

static\_strings\_string\_descriptor \* static\_strings\_concatenate\_and\_clean\_both (static\_strings\_string\_descriptor \*concatenate\_at, static\_strings\_string\_descriptor \*concatenate)

Concatenate the second string at the end of the first in a new string and deallocate both parameters.

• static\_strings\_string\_descriptor \* static\_strings\_concatenate\_all (uint16\_t arguments\_quantity,...)

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. This function must be used careful.

static\_strings\_string\_descriptor \* static\_strings\_concatenate\_and\_clean\_all (uint16\_t arguments\_
 quantity,...)

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. All the parameters are deallocated if success. This function must be used careful.

int static\_strings\_contains\_string (static\_strings\_string\_descriptor \*search\_in, static\_strings\_string\_descriptor \*search\_for)

Search a string in other string.

• int static\_strings\_contains\_char (static\_strings\_string\_descriptor \*search\_in, uint8\_t search\_for)

Search a character in a string.

int static\_strings\_compare (static\_strings\_string\_descriptor \*compare\_string\_one, static\_strings\_string\_descriptor \*compare\_string\_two)

Compare two strings to see if they are equals.

• static\_strings\_string\_descriptor \* static\_strings\_uint8\_to\_string (uint8\_t uint8)

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \* static\_strings\_uint16\_to\_string (uint16\_t uint16)

Create a string with the value of the parameter.

• static\_strings\_string\_descriptor \* static\_strings\_uint32\_to\_string (uint32\_t uint32)

Create a string with the value of the parameter.

• static\_strings\_string\_descriptor \* static\_strings\_int8\_to\_string (int8\_t int8)

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \* static\_strings\_int16\_to\_string (int16\_t int16)

Create a string with the value of the parameter.

• static\_strings\_string\_descriptor \* static\_strings\_int32\_to\_string (int32\_t int32)

Create a string with the value of the parameter.

• static strings string descriptor \* static strings float to string (float float arg)

Create a string with the value of the parameter.

• static\_strings\_string\_descriptor \* static\_strings\_double\_to\_string (double double\_arg)

Create a string with the value of the parameter.

### **Variables**

• static\_strings\_string\_splitter\_parameters static\_strings\_string\_splitter = {NULL,'\0'}

### 7.2.1 Detailed Description

Strings allocation with static memory.

#### 7.2.2 Function Documentation

## 7.2.2.1 static\_strings\_allocate()

Request memory for a string with its size, the user must copy the string with the descriptor and specify the size. Also see static strings save.

```
static_strings_string_descriptor *static_strings_allocate(uint16_t string_size)
```

24 File Documentation

#### **Parameters**

string_size	Size of the string in
	uint16_t.

### Returns

A pointer to the string descriptor, if NULL check static\_strings\_error\_code.

## 7.2.2.2 static\_strings\_clone()

```
static\_strings\_string\_descriptor*\ static\_strings\_clone\ ( static\_strings\_string\_descriptor\ *\ clone\_from\ )
```

Clone a string into a new one.

static\_strings\_string\_descriptor \*static\_strings\_clone(static\_strings\_string\_descriptor \*clone\_from)

#### **Parameters**

e_from Pointer to the string to clone.
--

#### Returns

A pointer to the descriptor with the cloned string if success, if an error occur return NULL, check static\_
strings\_error\_code for further information.

## 7.2.2.3 static\_strings\_compare()

Compare two strings to see if they are equals.

int static\_strings\_compare(static\_strings\_string\_descriptor\* compare\_string\_one,static\_strings\_string\_descriptor\* compare\_string\_tri

## **Parameters**

compare_string_one	A pointer to the first string to compare.
compare_string_two	A pointer to the second string to compare.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

## 7.2.2.4 static\_strings\_concatenate()

Concatenate the second string at the end of the first in a new string.

static\_strings\_string\_descriptor static\_strings\_concatenate(static\_strings\_string\_descriptor concatenate\_← at,static\_strings\_string\_descriptor\* concatenate)

#### **Parameters**

concatenate← _at	A pointer to the string to concatenate at.
concatenate	A pointer to the string to concatenate at the end of the concatenate_at string.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

## 7.2.2.5 static\_strings\_concatenate\_all()

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. This function must be used careful.

static\_strings\_string\_descriptor \*static\_strings\_concatenate\_all(uint16\_t arguments\_quantity,...)

### **Parameters**

arguments_quantity	The number of strings to concatenate.
	Multiple arguments of type static_strigs_string_descriptor pointer.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

26 File Documentation

### 7.2.2.6 static\_strings\_concatenate\_and\_clean()

Concatenate the second string at the end of the first in a new string and deallocate the concatenate at parameter if success.

static\_strings\_string\_descriptor static\_strings\_concatenate\_and\_clean(static\_strings\_string\_descriptor concatenate ← at,static\_strings\_string\_descriptor \* concatenate)

#### **Parameters**

concatenate↔	A pointer to the string to concatenate at, it is deallocates if success.
_at	
concatenate	A pointer to the string to concatenate at the end of the concatenate_at string.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static strings error code.

## 7.2.2.7 static\_strings\_concatenate\_and\_clean\_all()

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. All the parameters are deallocated if success. This function must be used careful.

static\_strings\_string\_descriptor \*static\_strings\_concatenate\_all(uint16\_t arguments\_quantity,...)

### **Parameters**

arguments_quantity	The number of strings to concatenate.
	Multiple arguments of type static_strigs_string_descriptor pointer, these parameters are
	deallocated if success.

### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

## 7.2.2.8 static\_strings\_concatenate\_and\_clean\_both()

Concatenate the second string at the end of the first in a new string and deallocate both parameters.

static\_strings\_string\_descriptor static\_strings\_concatenate\_and\_clean(static\_strings\_string\_descriptor concatenate 
\_at,static\_strings\_string\_descriptor \* concatenate)

### **Parameters**

concatenate←	A pointer to the string to concatenate at, it is deallocates if success.
_at	
concatenate	A pointer to the string to concatenate at the end of the concatenate_at string, it is deallocates
	if success.

### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

## 7.2.2.9 static\_strings\_contains\_char()

Search a character in a string.

int static\_strings\_contains\_char(static\_strings\_string\_descriptor\* search\_in,uint8\_t search\_for)

## **Parameters**

search_in	A pointer to the string in which the character will be search.
search_for	The searched character.

## Returns

1 if the character is found, 0 if not.

### 7.2.2.10 static\_strings\_contains\_string()

Search a string in other string.

int static\_strings\_contains\_string(static\_strings\_string\_descriptor\* search\_in,static\_strings\_string\_descriptor\* search\_for)

28 File Documentation

#### **Parameters**

search_in	A pointer to the string in which the character will be search.
search_for	A pointer to the searched string.

#### Returns

1 if the string is found, 0 if not.

### 7.2.2.11 static\_strings\_copy()

Copy a string into another at determinate offset. Leaves intact the string values before the offset. Can throw STATIC\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW.

static\_strings\_string\_descriptor \*static\_strings\_copy(static\_strings\_string\_descriptor \*copy\_to,static\_strings\_string\_descriptor \*copy\_from,uint16\_t copy\_to\_offset)

### **Parameters**

copy_to	Pointer to the string to copy in. String must have a defined type and length before use this function
copy_from	Pointer to the string to copy from.
copy_to_offset	Start copy index.

### Returns

A pointer to the descriptor with the copied string if success, if an error occur return NULL, check static\_
strings\_error\_code for further information.

## 7.2.2.12 static\_strings\_create\_custom\_string()

Bind the provided string descriptor with the data of a string. String must end with \r\n or \0.

void static strings create custom string(static strings string descriptor \*string descriptor,uint8 t \*string)

#### **Parameters**

string_descriptor	A pointer to a string descriptor.
string	A pointer to the string to bind the descriptor.

#### Returns

Return the length of the string, if 0 check static\_strings\_error\_code.

## 7.2.2.13 static\_strings\_deallocate()

Set the descriptor status as deallocated. Custom strings can't be deallocated.

void static\_strings\_deallocate(static\_strings\_string\_descriptor \*string\_descriptor)

#### **Parameters**

## 7.2.2.14 static\_strings\_double\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_double\_to\_string(double\_double\_arg)

## **Parameters**

```
double_arg 32 bits signed float (double).
```

## Returns

A pointer to the string descriptor with the parameter as string.

## 7.2.2.15 static\_strings\_float\_to\_string()

30 File Documentation

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_float\_to\_string(float float\_arg)

### **Parameters**

```
float_arg 16 bits signed float.
```

#### Returns

A pointer to the string descriptor with the parameter as string.

## 7.2.2.16 static\_strings\_get\_string\_max\_length()

get the maximum length allowed by the type of the string.

int static\_strings\_get\_string\_max\_length(static\_strings\_string\_descriptor \*string)

#### **Parameters**

string	A pointer to a string descriptor.
--------	-----------------------------------

#### Returns

The maximum allowed length of the string as an integer.

## 7.2.2.17 static\_strings\_init()

```
void static_strings_init ( )
```

Link the descriptors with the arrays and initialize the status as deallocated. Also can be used to reset the state of all the string descriptors.

void static\_strings\_init()

## 7.2.2.18 static\_strings\_int16\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_int16\_to\_string(int16\_t int16)

#### **Parameters**

int16 16 bits signed integer.
-------------------------------

# Returns

A pointer to the string descriptor with the parameter as string.

# 7.2.2.19 static\_strings\_int32\_to\_string()

```
\begin{tabular}{ll} {\tt static\_strings\_string\_descriptor*} & {\tt static\_strings\_int32\_to\_string} & ( & int32\_t & int32 \end{tabular} ) \\ \end{tabular}
```

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_int32\_to\_string(int32\_t int32)

#### **Parameters**

int32	32 bits signed integer.

### Returns

A pointer to the string descriptor with the parameter as string.

# 7.2.2.20 static\_strings\_int8\_to\_string()

```
\begin{tabular}{ll} static\_strings\_string\_descriptor* static\_strings\_int8\_to\_string ( \\ int8\_t $int8$ ) \end{tabular}
```

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_int8\_to\_string(int8\_t int8)

# **Parameters**

```
int8 8 bits signed integer.
```

# Returns

A pointer to the string descriptor with the parameter as string.

### 7.2.2.21 static\_strings\_is\_line()

Look at the last two characters of a string to see if the string has a line ending \r\n.

int static strings is line(static strings string descriptor \*string descriptor)

#### **Parameters**

nter to the string descriptor.	string
--------------------------------	--------

#### Returns

Return 0 if the string does't have a line ending \r\n and 1 if the string has a line ending \r\n.

# 7.2.2.22 static\_strings\_move()

Move a string into another at determinate offset, if success the move\_to string is deallocated. Can throw STATI

C\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW. Leaves intact the string values before the offset.

static\_strings\_string\_descriptor \*static\_strings\_move(static\_strings\_string\_descriptor \*move\_to,static\_strings\_string\_descriptor \*move from,uint16 t move to offset)

### **Parameters**

move_to	Pointer to the string to move in. String must have a defined type and length before use this	
	function	
move_from	Pointer to the string to move from.	
move_to_offset	Start move index.	

# Returns

A pointer to the descriptor with the moved string if success, if an error occur return NULL, check static\_
strings\_error\_code for further information.

### 7.2.2.23 static\_strings\_save()

Calculate the string size, allocate memory, copy the string and set the size. String must end with \r\n or \0, if \r is found but \n is not found, it is added, size of string include line ending but not \0. Also see static\_strings\_allocate.

static\_strings\_string\_descriptor \*static\_strings\_save(uint8\_t \*string)

### **Parameters**

pointer to the string start.	string
------------------------------	--------

#### Returns

A pointer to the string descriptor, if NULL check static strings error code.

### 7.2.2.24 static\_strings\_string\_splitter\_get\_next\_token()

Bind the provided string descriptor with the next token data. Can be placed in a while condition as it returns 1 if success or 0 if no token available and retrieves the token in the string\_descriptor parameter. If no delimiter the whole string is taken as token. The token is placed in a new string.

int static\_strings\_string\_splitter\_get\_next\_token(static\_strings\_string\_descriptor \*\*string\_descriptor)

#### **Parameters**

	string_descriptor	A pointer to a pointer to a string descriptor that will contain the token.	1
--	-------------------	--	---

#### Returns

1 if success or 0 if no token is available.

# 7.2.2.25 static\_strings\_string\_splitter\_set\_parameters()

Set the parameters to the static\_strings\_string\_splitter\_get\_next\_token function.

 $\textbf{void}\ static\_strings\_string\_splitter\_set\_parameters(static\_strings\_string\_descriptor *string\_descriptor, uint8\_t\ delimiter)$ 

#### **Parameters**

string_descriptor	A pointer to the string descriptor of the string to split.
delimiter	The delimiter for the tokens.

### 7.2.2.26 static strings strlen()

Calculate the length of a string that ends with  $\n$  or  $\0$ , line ending is included in length. Maximum length is STATIC\_STRINGS\_VERY\_LONG\_STRING\_SIZE.

uint16\_t static\_strings\_strlen(uint8\_t \*string)

### **Parameters**

string	A pointer to the string.
string	A pointer to the string.

#### Returns

Length of the string in uint16\_t. If 0 check static\_strings\_error\_code.

# 7.2.2.27 static\_strings\_substring()

Return a new string with the characters between the start\_index and the finish\_index. Not including the character at finish\_index. Returned string has to be deallocated. To get all the string from a start index use the length in the finish\_index.

static\_strings\_string\_descriptor static\_strings\_substring(static\_strings\_string\_descriptor string\_descriptor,uint16

\_t start\_index,uint16\_t finish\_index)

#### **Parameters**

string_descriptor	A pointer to the string which contains the substring.
start_index	The index of the first character.
finish_index	The index of the last character, not included.

### Returns

A pointer to the string descriptor of the substring, if NULL check static strings error code.

### 7.2.2.28 static\_strings\_uint16\_to\_string()

```
 \begin{array}{c} \textbf{static\_strings\_string\_descriptor*} & \textbf{static\_strings\_uint16\_to\_string} & (\\ & \textbf{uint16\_t} & \textbf{uint16} & ) \end{array}
```

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_uint16\_to\_string(uint16\_t uint16)

#### **Parameters**

uint16 16 bits unsigned integ	er.
-------------------------------	-----

#### Returns

A pointer to the string descriptor with the parameter as string.

# 7.2.2.29 static\_strings\_uint32\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_uint32\_to\_string(uint32\_t uint32)

### **Parameters**

```
uint32 32 bits unsigned integer.
```

### Returns

A pointer to the string descriptor with the parameter as string.

# 7.2.2.30 static\_strings\_uint8\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_uint8\_to\_string(uint8\_t uint8)

#### **Parameters**

ger.
;

#### Returns

A pointer to the string descriptor with the parameter as string.

### 7.2.3 Variable Documentation

# 7.2.3.1 static\_strings\_string\_splitter

```
static\_strings\_string\_splitter\_parameters\ static\_strings\_string\_splitter\ =\ \{\texttt{NULL}, \texttt{'}\setminus \texttt{0'}\}
```

Parameters to static\_strings\_string\_splitter\_get\_next\_token function. Initialized in null and \0.

# 7.3 static\_strings.h File Reference

Strings allocation with static memory.

```
#include "stm32f1xx_hal.h"
#include "string.h"
#include "int_types.h"
#include "stdarg.h"
#include "stdio.h"
```

# **Data Structures**

• struct static\_strings\_string\_descriptor

Meta data of a string.

• struct static\_strings\_string\_splitter\_parameters

 $Definition\ of\ the\ structure\ to\ hold\ the\ parameters\ to\ static\_strings\_string\_splitter\_get\_next\_token\ function.$ 

### **Macros**

- #define STATIC\_STRINGS\_VERY\_SHORT\_STRING\_SIZE 50
- #define STATIC STRINGS VERY SHORT STRING QUANTITY 10
- #define STATIC STRINGS SHORT STRING SIZE 100
- #define STATIC STRINGS SHORT STRING QUANTITY 6
- #define STATIC STRINGS MEDIUM STRING SIZE 200
- #define STATIC\_STRINGS\_MEDIUM\_STRING\_QUANTITY 2
- #define STATIC STRINGS LONG STRING SIZE 500
- #define STATIC STRINGS LONG STRING QUANTITY 2
- #define STATIC\_STRINGS\_VERY\_LONG\_STRING\_SIZE 1500
- #define STATIC STRINGS VERY LONG STRING QUANTITY 2
- #define STATIC STRINGS STRING TYPE VERY SHORT 0
- #define STATIC STRINGS STRING TYPE SHORT 1
- #define STATIC\_STRINGS\_STRING\_TYPE\_MEDIUM 2
- #define STATIC\_STRINGS\_STRING\_TYPE\_LONG 3
- #define STATIC STRINGS STRING TYPE VERY LONG 4
- #define STATIC STRINGS STRING TYPE CUSTOM 5
- #define STATIC\_STRINGS\_STRING\_STATUS\_DEALLOCATED 0
- #define STATIC\_STRINGS\_STRING\_STATUS\_ALLOCATED 1
- #define STATIC STRINGS STRING STATUS CONSTANT 2
- #define STATIC STRINGS ERROR CODE NO ERROR 0
- #define STATIC STRINGS ERROR CODE NO MEMORY AVAILABLE 1
- #define STATIC\_STRINGS\_ERROR\_CODE\_INVALID\_STRING 2
- #define STATIC STRINGS ERROR CODE STRING TOO LONG 3
- #define STATIC STRINGS ERROR CODE SUBSTRING START INDEX OUT OF RANGE 4
- #define STATIC STRINGS ERROR CODE SUBSTRING FINISH INDEX OUT OF RANGE 5
- #define STATIC\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW 6

# **Typedefs**

- typedef struct static\_strings\_string\_descriptor static\_strings\_string\_descriptor
- typedef struct static\_strings\_string\_splitter\_parameters static\_strings\_string\_splitter\_parameters

### **Functions**

void static\_strings\_init ()

Link the descriptors with the arrays and initialize the status as deallocated. Also can be used to reset the state of all the string descriptors.

- int static\_strings\_get\_string\_max\_length (static\_strings\_string\_descriptor \*string)
  - get the maximum length allowed by the type of the string.
- static\_strings\_string\_descriptor \* static\_strings\_copy (static\_strings\_string\_descriptor \*copy\_to, static\_strings\_string\_descriptor \*copy\_from, uint16\_t copy\_to\_offset)

Copy a string into another at determinate offset. Leaves intact the string values before the offset. Can throw STAT← IC\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW.

static\_strings\_string\_descriptor \* static\_strings\_move (static\_strings\_string\_descriptor \*move\_to, static\_strings\_string\_descriptor \*move\_to, static\_strings\_string\_descriptor \*move\_from, uint16\_t move\_to\_offset)

Move a string into another at determinate offset, if success the move\_to string is deallocated. Can throw STATIC\_
STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW. Leaves intact the string values before the offset.

- static strings string descriptor \* static strings clone (static strings string descriptor \*clone from)
  - Clone a string into a new one.
- static\_strings\_string\_descriptor \* static\_strings\_allocate (uint16\_t string\_size)

Request memory for a string with its size, the user must copy the string with the descriptor and specify the size. Also see static\_strings\_save.

static strings string descriptor \* static strings save (uint8 t \*string)

Calculate the string size, allocate memory, copy the string and set the size. String must end with  $\n$  or  $\0$ , if  $\n$  is found but  $\n$  is not found, it is added, size of string include line ending but not  $\0$ . Also see static\_strings\_allocate.

int static\_strings\_create\_custom\_string (static\_strings\_string\_descriptor \*string\_descriptor, uint8\_t \*string)

Bind the provided string descriptor with the data of a string. String must end with \r\n or \0.

· void static strings deallocate (static strings string descriptor \*string descriptor)

Set the descriptor status as deallocated. Custom strings can't be deallocated.

int static\_strings\_is\_line (static\_strings\_string\_descriptor \*string\_descriptor)

Look at the last two characters of a string to see if the string has a line ending \r\n.

uint16\_t static\_strings\_strlen (uint8\_t \*string)

Calculate the length of a string that ends with \r\n or \0, line ending is included in length. Maximum length is STAT← IC\_STRINGS\_VERY\_LONG\_STRING\_SIZE.

void static\_strings\_string\_splitter\_set\_parameters (static\_strings\_string\_descriptor \*string\_descriptor, uint8 t delimiter)

Set the parameters to the static\_strings\_string\_splitter\_get\_next\_token function.

int static\_strings\_string\_splitter\_get\_next\_token (static\_strings\_string\_descriptor \*\*string\_descriptor)

Bind the provided string descriptor with the next token data. Can be placed in a while condition as it returns 1 if success or 0 if no token available and retrieves the token in the string\_descriptor parameter. If no delimiter the whole string is taken as token. The token is placed in a new string.

 static\_strings\_string\_descriptor \* static\_strings\_substring (static\_strings\_string\_descriptor \*string, uint16\_t start\_index, uint16\_t finish\_index)

Return a new string with the characters between the start\_index and the finish\_index. Not including the character at finish\_index. Returned string has to be deallocated. To get all the string from a start index use the length in the finish index.

static\_strings\_string\_descriptor \* static\_strings\_concatenate (static\_strings\_string\_descriptor \*concatenate 
 \_at, static\_strings\_string\_descriptor \*concatenate)

Concatenate the second string at the end of the first in a new string.

• static\_strings\_string\_descriptor \* static\_strings\_concatenate\_and\_clean (static\_strings\_string\_descriptor \*concatenate\_at, static\_strings\_string\_descriptor \*concatenate)

Concatenate the second string at the end of the first in a new string and deallocate the concatenate at parameter if success.

• static\_strings\_string\_descriptor \* static\_strings\_concatenate\_and\_clean\_both (static\_strings\_string\_descriptor \*concatenate\_at, static\_strings\_string\_descriptor \*concatenate)

Concatenate the second string at the end of the first in a new string and deallocate both parameters.

static strings string descriptor \* static strings concatenate all (uint16 t arguments quantity...)

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. This function must be used careful.

static\_strings\_string\_descriptor \* static\_strings\_concatenate\_and\_clean\_all (uint16\_t arguments\_← quantity,...)

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. All the parameters are deallocated if success. This function must be used careful.

• int static\_strings\_contains\_string (static\_strings\_string\_descriptor \*search\_in, static\_strings\_string\_descriptor \*search\_for)

Search a string in other string.

• int static\_strings\_contains\_char (static\_strings\_string\_descriptor \*search\_in, uint8\_t search\_for)

Search a character in a string.

int static\_strings\_compare (static\_strings\_string\_descriptor \*compare\_string\_one, static\_strings\_string\_descriptor \*compare\_string\_two)

Compare two strings to see if they are equals.

static strings string descriptor \* static strings uint8 to string (uint8 t uint8)

Create a string with the value of the parameter.

• static\_strings\_string\_descriptor \* static\_strings\_uint16\_to\_string (uint16\_t uint16)

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \* static\_strings\_uint32\_to\_string (uint32\_t uint32)

Create a string with the value of the parameter.

static strings string descriptor \* static strings int8 to string (int8 t int8)

Create a string with the value of the parameter.

• static strings string descriptor \* static strings int16 to string (int16 t int16)

Create a string with the value of the parameter.

• static\_strings\_string\_descriptor \* static\_strings\_int32\_to\_string (int32\_t int32)

Create a string with the value of the parameter.

static strings string descriptor \* static strings float to string (float float arg)

Create a string with the value of the parameter.

• static\_strings\_string\_descriptor \* static\_strings\_double\_to\_string (double double\_arg)

Create a string with the value of the parameter.

### **Variables**

· uint8 t static strings error code

Global variable to store error code.

- static\_strings\_string\_splitter\_parameters static\_strings\_string\_splitter
- uint8\_t static\_strings\_very\_short\_string\_memory [STATIC\_STRINGS\_VERY\_SHORT\_STRING\_QUA⊷ NTITY][STATIC\_STRINGS\_VERY\_SHORT\_STRING\_SIZE]
- uint8\_t static\_strings\_short\_string\_memory [STATIC\_STRINGS\_SHORT\_STRING\_QUANTITY][STAT ← IC STRINGS SHORT STRING SIZE]
- uint8\_t static\_strings\_medium\_string\_memory [STATIC\_STRINGS\_MEDIUM\_STRING\_QUANTITY][S
   — TATIC\_STRINGS\_MEDIUM\_STRING\_SIZE]
- uint8\_t static\_strings\_long\_string\_memory [STATIC\_STRINGS\_LONG\_STRING\_QUANTITY][STATIC
   — STRINGS\_LONG\_STRING\_SIZE]
- uint8\_t static\_strings\_very\_long\_string\_memory [STATIC\_STRINGS\_VERY\_LONG\_STRING\_QUAN ← TITY][STATIC\_STRINGS\_VERY\_LONG\_STRING\_SIZE]
- static\_strings\_string\_descriptor static\_strings\_short\_strings\_descriptors [STATIC\_STRINGS\_SHORT
   — STRING\_QUANTITY]
- static\_strings\_string\_descriptor static\_strings\_medium\_strings\_descriptors [STATIC\_STRINGS\_MED ∪ IUM\_STRING\_QUANTITY]

# 7.3.1 Detailed Description

Strings allocation with static memory.

# 7.3.2 Function Documentation

### 7.3.2.1 static\_strings\_allocate()

Request memory for a string with its size, the user must copy the string with the descriptor and specify the size. Also see static\_strings\_save.

static\_strings\_string\_descriptor \*static\_strings\_allocate(uint16\_t string\_size)

#### **Parameters**

string_size	Size of the string in
	uint16_t.

### Returns

A pointer to the string descriptor, if NULL check static\_strings\_error\_code.

### 7.3.2.2 static\_strings\_clone()

Clone a string into a new one.

static\_strings\_string\_descriptor \*static\_strings\_clone(static\_strings\_string\_descriptor \*clone\_from)

### **Parameters**

clone_from	Pointer to the string to clone.
------------	---------------------------------

### Returns

A pointer to the descriptor with the cloned string if success, if an error occur return NULL, check static\_ strings\_error\_code for further information.

# 7.3.2.3 static\_strings\_compare()

Compare two strings to see if they are equals.

int static\_strings\_compare(static\_strings\_string\_descriptor\* compare\_string\_one,static\_strings\_string\_descriptor\* compare\_string\_tri

#### **Parameters**

compare_string_one	A pointer to the first string to compare.
compare_string_two	A pointer to the second string to compare.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static strings error code.

# 7.3.2.4 static\_strings\_concatenate()

Concatenate the second string at the end of the first in a new string.

static\_strings\_string\_descriptor static\_strings\_concatenate(static\_strings\_string\_descriptor concatenate\_← at,static\_strings\_string\_descriptor\* concatenate)

#### **Parameters**

concatenate←	A pointer to the string to concatenate at.
_at	
concatenate	A pointer to the string to concatenate at the end of the concatenate_at string.

### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

#### 7.3.2.5 static strings concatenate all()

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. This function must be used careful.

static\_strings\_string\_descriptor \*static\_strings\_concatenate\_all(uint16\_t arguments\_quantity,...)

### **Parameters**

arguments_quantity	The number of strings to concatenate.
	Multiple arguments of type static_strigs_string_descriptor pointer.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

### 7.3.2.6 static\_strings\_concatenate\_and\_clean()

Concatenate the second string at the end of the first in a new string and deallocate the concatenate at parameter if success.

static\_strings\_string\_descriptor static\_strings\_concatenate\_and\_clean(static\_strings\_string\_descriptor concatenate 
\_at,static\_strings\_string\_descriptor \* concatenate)

#### **Parameters**

concatenate <i>←</i> _at	A pointer to the string to concatenate at, it is deallocates if success.
concatenate	A pointer to the string to concatenate at the end of the concatenate_at string.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static strings error code.

# 7.3.2.7 static\_strings\_concatenate\_and\_clean\_all()

Concatenates multiple strings in the order of the arguments, the number of arguments must be provided in the first parameter. All the parameters are deallocated if success. This function must be used careful.

static\_strings\_string\_descriptor \*static\_strings\_concatenate\_all(uint16\_t arguments\_quantity,...)

# **Parameters**

arguments_quantity	The number of strings to concatenate.
	Multiple arguments of type static_strigs_string_descriptor pointer, these parameters are deallocated if success.

### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

# 7.3.2.8 static\_strings\_concatenate\_and\_clean\_both()

Concatenate the second string at the end of the first in a new string and deallocate both parameters.

static\_strings\_string\_descriptor static\_strings\_concatenate\_and\_clean(static\_strings\_string\_descriptor concatenate ← \_\_at,static\_strings\_string\_descriptor \* concatenate)

#### **Parameters**

concatenate←	A pointer to the string to concatenate at, it is deallocates if success.
_at	
concatenate	A pointer to the string to concatenate at the end of the concatenate_at string, it is deallocates
	if success.

#### Returns

A pointer to the string descriptor with the concatenated string, if NULL check static\_strings\_error\_code.

### 7.3.2.9 static\_strings\_contains\_char()

Search a character in a string.

int static\_strings\_contains\_char(static\_strings\_string\_descriptor\* search\_in,uint8\_t search\_for)

#### **Parameters**

search_in	A pointer to the string in which the character will be search.
search_for	The searched character.

# Returns

1 if the character is found, 0 if not.

# 7.3.2.10 static\_strings\_contains\_string()

Search a string in other string.

 $int\ static\_strings\_contains\_string(static\_strings\_string\_descriptor*\ search\_in, static\_strings\_string\_descriptor*\ search\_in, static\_strings\_string\_strings\_string\_strings\_strin$ 

#### **Parameters**

search_in	A pointer to the string in which the character will be search.
search_for	A pointer to the searched string.

#### Returns

1 if the string is found, 0 if not.

# 7.3.2.11 static\_strings\_copy()

Copy a string into another at determinate offset. Leaves intact the string values before the offset. Can throw STATIC\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW.

static\_strings\_string\_descriptor \*static\_strings\_copy(static\_strings\_string\_descriptor \*copy\_to,static\_strings\_string\_descriptor \*copy\_from,uint16\_t copy\_to\_offset)

### **Parameters**

copy_to	Pointer to the string to copy in. String must have a defined type and length before use this function
copy_from	Pointer to the string to copy from.
copy_to_offset	Start copy index.

### Returns

A pointer to the descriptor with the copied string if success, if an error occur return NULL, check static\_

strings\_error\_code for further information.

# 7.3.2.12 static\_strings\_create\_custom\_string()

Bind the provided string descriptor with the data of a string. String must end with \r\n or \0.

void static\_strings\_create\_custom\_string(static\_strings\_string\_descriptor \*string\_descriptor,uint8\_t \*string)

#### **Parameters**

string_descriptor	A pointer to a string descriptor.
string	A pointer to the string to bind the descriptor.

### Returns

Return the length of the string, if 0 check static\_strings\_error\_code.

# 7.3.2.13 static\_strings\_deallocate()

Set the descriptor status as deallocated. Custom strings can't be deallocated.

void static\_strings\_deallocate(static\_strings\_string\_descriptor \*string\_descriptor)

#### **Parameters**

string_descriptor	A pointer to the string descriptor to deallocate.	
-------------------	---	--

# 7.3.2.14 static\_strings\_double\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_double\_to\_string(double\_double\_arg)

# **Parameters**

double_arg	32 bits signed float (double).

# Returns

A pointer to the string descriptor with the parameter as string.

# 7.3.2.15 static\_strings\_float\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_float\_to\_string(float float\_arg)

### **Parameters**

```
float_arg 16 bits signed float.
```

#### Returns

A pointer to the string descriptor with the parameter as string.

# 7.3.2.16 static\_strings\_get\_string\_max\_length()

get the maximum length allowed by the type of the string.

int static\_strings\_get\_string\_max\_length(static\_strings\_string\_descriptor \*string)

# **Parameters**

g A pointer to a string descriptor.	ointer to a string descriptor.	string
-------------------------------------	--------------------------------	--------

#### Returns

The maximum allowed length of the string as an integer.

# 7.3.2.17 static\_strings\_init()

```
void static_strings_init ( )
```

Link the descriptors with the arrays and initialize the status as deallocated. Also can be used to reset the state of all the string descriptors.

void static\_strings\_init()

# 7.3.2.18 static\_strings\_int16\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_int16\_to\_string(int16\_t int16)

#### **Parameters**

# Returns

A pointer to the string descriptor with the parameter as string.

# 7.3.2.19 static\_strings\_int32\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_int32\_to\_string(int32\_t int32)

### **Parameters**

int32 32 bits signed into
---------------------------

### Returns

A pointer to the string descriptor with the parameter as string.

# 7.3.2.20 static\_strings\_int8\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_int8\_to\_string(int8\_t int8)

### **Parameters**

```
int8 8 bits signed integer.
```

# Returns

A pointer to the string descriptor with the parameter as string.

### 7.3.2.21 static\_strings\_is\_line()

Look at the last two characters of a string to see if the string has a line ending \r\n.

int static strings is line(static strings string descriptor \*string descriptor)

#### **Parameters**

#### Returns

Return 0 if the string does't have a line ending \r\n and 1 if the string has a line ending \r\n.

### 7.3.2.22 static\_strings\_move()

Move a string into another at determinate offset, if success the move\_to string is deallocated. Can throw STATI

C\_STRINGS\_ERROR\_CODE\_STRING\_OVERFLOW. Leaves intact the string values before the offset.

static\_strings\_string\_descriptor \*static\_strings\_move(static\_strings\_string\_descriptor \*move\_to,static\_strings\_string\_descriptor \*move from,uint16 t move to offset)

### **Parameters**

move_to	Pointer to the string to move in. String must have a defined type and length before use this function
move_from	Pointer to the string to move from.
move_to_offset	Start move index.

# Returns

A pointer to the descriptor with the moved string if success, if an error occur return NULL, check static\_
strings\_error\_code for further information.

### 7.3.2.23 static\_strings\_save()

Calculate the string size, allocate memory, copy the string and set the size. String must end with \r\n or \0, if \r is found but \n is not found, it is added, size of string include line ending but not \0. Also see static\_strings\_allocate.

static\_strings\_string\_descriptor \*static\_strings\_save(uint8\_t \*string)

### **Parameters**

#### Returns

A pointer to the string descriptor, if NULL check static strings error code.

# 7.3.2.24 static\_strings\_string\_splitter\_get\_next\_token()

Bind the provided string descriptor with the next token data. Can be placed in a while condition as it returns 1 if success or 0 if no token available and retrieves the token in the string\_descriptor parameter. If no delimiter the whole string is taken as token. The token is placed in a new string.

int static\_strings\_string\_splitter\_get\_next\_token(static\_strings\_string\_descriptor \*\*string\_descriptor)

#### **Parameters**

string_des	criptor   A	A pointer to a pointer to a string descriptor that will contain the token.
------------	-------------	--

#### Returns

1 if success or 0 if no token is available.

# 7.3.2.25 static\_strings\_string\_splitter\_set\_parameters()

Set the parameters to the static\_strings\_string\_splitter\_get\_next\_token function.

void static\_strings\_string\_splitter\_set\_parameters(static\_strings\_string\_descriptor \*string\_descriptor,uint8\_t delimiter)

#### **Parameters**

string_descriptor	A pointer to the string descriptor of the string to split.
delimiter	The delimiter for the tokens.

# 7.3.2.26 static\_strings\_strlen()

Calculate the length of a string that ends with  $\n$  or  $\0$ , line ending is included in length. Maximum length is STATIC\_STRINGS\_VERY\_LONG\_STRING\_SIZE.

uint16\_t static\_strings\_strlen(uint8\_t \*string)

### **Parameters**

string A pointer to the strip	ng.
-------------------------------	-----

#### Returns

Length of the string in uint16\_t. If 0 check static\_strings\_error\_code.

# 7.3.2.27 static\_strings\_substring()

Return a new string with the characters between the start\_index and the finish\_index. Not including the character at finish\_index. Returned string has to be deallocated. To get all the string from a start index use the length in the finish\_index.

static\_strings\_string\_descriptor static\_strings\_substring(static\_strings\_string\_descriptor string\_descriptor,uint16

\_t start\_index,uint16\_t finish\_index)

#### **Parameters**

string_descriptor	A pointer to the string which contains the substring.
start_index	The index of the first character.
finish_index	The index of the last character, not included.

### Returns

A pointer to the string descriptor of the substring, if NULL check static strings error code.

### 7.3.2.28 static\_strings\_uint16\_to\_string()

```
 \begin{array}{c} \textbf{static\_strings\_string\_descriptor*} & \textbf{static\_strings\_uint16\_to\_string} & (\\ & \textbf{uint16\_t} & \textbf{uint16} & ) \end{array}
```

Create a string with the value of the parameter.

static strings string descriptor \*static strings uint16 to string(uint16 t uint16)

#### **Parameters**

uint16 16 bits unsigned integ	er.
-------------------------------	-----

### Returns

A pointer to the string descriptor with the parameter as string.

# 7.3.2.29 static\_strings\_uint32\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_uint32\_to\_string(uint32\_t uint32)

### **Parameters**

```
uint32 32 bits unsigned integer.
```

### Returns

A pointer to the string descriptor with the parameter as string.

# 7.3.2.30 static\_strings\_uint8\_to\_string()

Create a string with the value of the parameter.

static\_strings\_string\_descriptor \*static\_strings\_uint8\_to\_string(uint8\_t uint8)

# **Parameters**

uint8 8 bits unsigned integer.	
--------------------------------	--

# Returns

A pointer to the string descriptor with the parameter as string.

# 7.3.3 Variable Documentation

# 7.3.3.1 static\_strings\_string\_splitter

static\_strings\_string\_splitter\_parameters static\_strings\_string\_splitter

Parameters to static\_strings\_string\_splitter\_get\_next\_token function. Initialized in null and \0.

# Index

```
Error handling, 16
                                                                 static_strings_create_custom_string, 45
     static_strings_error_code, 16
                                                                 static_strings_deallocate, 46
                                                                 static strings double to string, 46
int_types.h, 21
                                                                 static_strings_float_to_string, 46
                                                                 static_strings_get_string_max_length, 47
Static memory arrays, 17
                                                                 static strings init, 47
static strings.c, 21
                                                                 static_strings_int16_to_string, 47
     static_strings_allocate, 23
                                                                 static strings int32 to string, 48
     static_strings_clone, 24
                                                                 static strings int8 to string, 48
     static_strings_compare, 24
                                                                 static_strings_is_line, 48
     static_strings_concatenate, 25
                                                                 static strings move, 49
     static strings concatenate all, 25
                                                                 static strings save, 49
     static strings concatenate and clean, 25
                                                                 static_strings_string_splitter, 53
     static strings concatenate and clean all, 26
                                                                 static_strings_string_splitter_get_next_token, 50
     static strings concatenate and clean both, 26
                                                                 static_strings_string_splitter_set_parameters, 50
     static_strings_contains_char, 27
                                                                 static_strings_strlen, 51
     static_strings_contains_string, 27
                                                                 static strings substring, 51
     static strings copy, 28
                                                                 static strings uint16 to string, 51
     static_strings_create_custom_string, 28
                                                                 static_strings_uint32_to_string, 52
     static_strings_deallocate, 29
                                                                 static strings uint8 to string, 52
     static_strings_double_to_string, 29
                                                            static strings allocate
     static_strings_float_to_string, 29
                                                                 static_strings.c, 23
     static strings get string max length, 30
                                                                 static_strings.h, 39
     static strings init, 30
                                                            static strings clone
     static strings int16 to string, 30
                                                                 static_strings.c, 24
     static strings int32 to string, 31
                                                                 static strings.h, 40
     static_strings_int8_to_string, 31
                                                            static strings compare
     static_strings_is_line, 31
                                                                 static strings.c, 24
     static strings move, 32
                                                                 static strings.h, 40
     static strings save, 32
                                                            static strings concatenate
     static_strings_string_splitter, 36
                                                                 static_strings.c, 25
     static_strings_string_splitter_get_next_token, 33
                                                                 static strings.h, 41
     static strings string splitter set parameters, 33
                                                            static_strings_concatenate_all
     static strings strlen, 34
                                                                 static_strings.c, 25
     static strings substring, 34
                                                                 static strings.h, 41
     static strings uint16 to string, 34
                                                            static strings concatenate and clean
     static strings uint32 to string, 35
                                                                 static strings.c, 25
     static_strings_uint8_to_string, 35
                                                                 static strings.h, 42
static strings.h, 36
                                                            static_strings_concatenate_and_clean_all
     static strings allocate, 39
                                                                 static strings.c, 26
     static_strings_clone, 40
                                                                 static strings.h, 42
     static_strings_compare, 40
                                                            static_strings_concatenate_and_clean_both
     static strings concatenate, 41
                                                                 static_strings.c, 26
     static strings concatenate all, 41
                                                                 static strings.h, 42
     static strings concatenate and clean, 42
                                                            static strings contains char
     static strings concatenate and clean all, 42
                                                                 static strings.c, 27
     static strings concatenate and clean both, 42
                                                                 static strings.h, 43
     static strings contains char, 43
                                                            static_strings_contains_string
     static_strings_contains_string, 43
                                                                 static_strings.c, 27
     static_strings_copy, 45
```

56 INDEX

static_strings.h, 43	static_strings.h, 51
static_strings_copy	static strings uint16 to string
static_strings.c, 28	static_strings.c, 34
static_strings.h, 45	static_strings.h, 51
static_strings_create_custom_string	static_strings_uint32_to_string
static_strings.c, 28	static strings.c, 35
static_strings.h, 45	static_strings.h, 52
static_strings.iii, 45	static_strings_uint8_to_string
	static_strings_cinto_to_string
static_strings.c, 29	static_strings.c, 33
static_strings.h, 46	
static_strings_double_to_string	String descriptors, 18
static_strings.c, 29	String status, 15
static_strings.h, 46	String types, 14
static_strings_error_code	String types size and quantity, 13
Error handling, 16	
static_strings_float_to_string	
static_strings.c, 29	
static_strings.h, 46	
static strings get string max length	
static strings.c, 30	
static strings.h, 47	
static_strings_init	
static_strings.c, 30	
static strings.h, 47	
static_strings.ii, 47 static_strings_int16_to_string	
static_strings.c, 30	
static_strings.h, 47	
static_strings_int32_to_string	
static_strings.c, 31	
static_strings.h, 48	
static_strings_int8_to_string	
static_strings.c, 31	
static_strings.h, 48	
static_strings_is_line	
static_strings.c, 31	
static_strings.h, 48	
static_strings_move	
static_strings.c, 32	
static_strings.h, 49	
static_strings_save	
static strings.c, 32	
static_strings.h, 49	
static_strings_string_descriptor, 19	
static strings string splitter	
static strings.c, 36	
static_strings.h, 53	
static_strings.ri, so static_strings_string_splitter_get_next_token	
static_strings.c, 33	
_ ·	
static_strings.h, 50	
static_strings_string_splitter_parameters, 19	
static_strings_string_splitter_set_parameters	
static_strings.c, 33	
static_strings.h, 50	
static_strings_strlen	
static_strings.c, 34	
static_strings.h, 51	
static_strings_substring	
static_strings.c, 34	