Statics Strings STM32F1XX

Generated by Doxygen 1.8.18

1 Static Strings	1
2 Module Index	5
2.1 Modules	5
3 Data Structure Index	7
3.1 Data Structures	7
4 File Index	9
4.1 File List	9
5 Module Documentation	11
5.1 String types size and quantity	11
5.1.1 Detailed Description	
5.2 String types	12
5.2.1 Detailed Description	12
5.3 String status	13
5.3.1 Detailed Description	
5.4 Error handling	14
5.4.1 Detailed Description	
5.4.2 Variable Documentation	
5.4.2.1 static_strings_error_code	14
5.5 Static memory arrays	
5.5.1 Detailed Description	
5.6 String descriptors	
5.6.1 Detailed Description	16
6 Data Structure Documentation	17
6.1 static_strings_string_descriptor Struct Reference	17
6.1.1 Detailed Description	17
6.2 static_strings_string_splitter_parameters Struct Reference	17
6.2.1 Detailed Description	
7 File Documentation	19
7.1 static_strings.c File Reference	
7.1.1 Detailed Description	
7.1.2 Function Documentation	
7.1.2.1 static_strings_allocate()	
7.1.2.2 static_strings_compare()	
7.1.2.3 static_strings_concatenate()	
7.1.2.4 static_strings_contains_char()	
7.1.2.5 static_strings_contains_string()	
7.1.2.6 static_strings_create_custom_string()	
7.1.2.7 static_strings_deallocate()	
///.z./ otatio_oti/iigo_dodi/otato()	20

7.1.2.8 static_strings_double_to_string()	23
7.1.2.9 static_strings_float_to_string()	24
7.1.2.10 static_strings_init()	24
7.1.2.11 static_strings_int16_to_string()	24
7.1.2.12 static_strings_int32_to_string()	25
7.1.2.13 static_strings_int8_to_string()	25
7.1.2.14 static_strings_is_line()	25
7.1.2.15 static_strings_save()	26
7.1.2.16 static_strings_string_splitter_get_next_token()	26
7.1.2.17 static_strings_string_splitter_set_parameters()	27
7.1.2.18 static_strings_strlen()	27
7.1.2.19 static_strings_substring()	27
7.1.2.20 static_strings_uint16_to_string()	28
7.1.2.21 static_strings_uint32_to_string()	28
7.1.2.22 static_strings_uint8_to_string()	29
7.1.3 Variable Documentation	29
7.1.3.1 static_strings_string_splitter	29
7.2 static_strings.h File Reference	29
7.2.1 Detailed Description	32
7.2.2 Function Documentation	32
7.2.2.1 static_strings_allocate()	32
7.2.2.2 static_strings_compare()	32
7.2.2.3 static_strings_concatenate()	33
7.2.2.4 static_strings_contains_char()	33
7.2.2.5 static_strings_contains_string()	34
7.2.2.6 static_strings_create_custom_string()	34
7.2.2.7 static_strings_deallocate()	35
7.2.2.8 static_strings_double_to_string()	35
7.2.2.9 static_strings_float_to_string()	35
7.2.2.10 static_strings_init()	36
7.2.2.11 static_strings_int16_to_string()	36
7.2.2.12 static_strings_int32_to_string()	36
7.2.2.13 static_strings_int8_to_string()	37
7.2.2.14 static_strings_is_line()	37
7.2.2.15 static_strings_save()	38
7.2.2.16 static_strings_string_splitter_get_next_token()	38
7.2.2.17 static_strings_string_splitter_set_parameters()	38
7.2.2.18 static_strings_strlen()	39
7.2.2.19 static_strings_substring()	39
7.2.2.20 static_strings_uint16_to_string()	40
7.2.2.21 static_strings_uint32_to_string()	40
7.2.2.22 static_strings_uint8_to_string()	40

		iii
	7.2.3 Variable Documentation	41
	7.2.3.1 static_strings_string_splitter	41
Index		43

Static Strings

Author

Ramsés F. Pérez

Date

August 2020

Version

1.0.1

Features:

- Developed for the STM32F103.
- · Global scope strings.
- · Configurable quantity and size of the memory arrays.
- No dynamic memory allocation.
- Customizable quantity and length 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, contains string, contains char and compare function.
- Transforms integers and floats to strings

2 Static Strings

GETTING STARTED

Suggested names

```
static_strings_string_descriptor string_name;
uint8_t string_name_memory[];
```

First of all initialize the library

```
static_strings_init();
```

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.

Also a string can be 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.

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);
}
```

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);
}
```

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);
    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);
```

Compare two equals and non equals strings

```
uint8_t equal_a_memory[] = "Hall\0";
static_strings_string_descriptor equ
uint8_t equal_b_memory[] = "Hall\0";
                                    equal a:
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);
\verb|static_strings_create_custom_string|(\&equal\_b,equal\_b\_memory)|;
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);
if(static_strings_compare(&equal_a,&non_equal)){
  HAL_UART_Transmit(&huart1, (uint8_t *) "1\r\n", 3, HAL_MAX_DELAY);
  HAL_UART_Transmit(&huart1, (uint8_t *) "0\r\n", 3, HAL_MAX_DELAY);
```

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);
}
```

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_MEDIUM_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 1
#define STATIC_STRINGS_VERY_LONG_STRING_SIZE 1000
#define STATIC_STRINGS_VERY_LONG_STRING_QUANTITY 1
```

4 Static Strings

Module Index

2.1 Modules

Here is a list of all modules:

tring types size and quantity $\ldots\ldots\ldots\ldots\ldots$. 1
tring types	. 12
tring status	. 13
rror handling	. 14
tatic memory arrays	. 15
tring descriptors	. 16

6 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	17
static_strings_string_splitter_parameters	
Definition of the structure to hold the parameters to static_stirngs_string_splitter_get_next_token	
function	17

8 Data Structure Index

File Index

4.1 File List

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

int_types.h	 ??
static_strings.c	
Strings allocation with static memory	 19
static_strings.h	
Strings allocation with static memory	 29

10 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 1
 #define STATIC STRINGS VERY LONG STRING SIZE 1000
- #define STATIC_STRINGS_VERY_LONG_STRING_QUANTITY 1

5.1.1 Detailed Description

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

12 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.

14 Module Documentation

5.4 Error handling

Error codes.

Macros

- #define STATIC_STRINGS_ERROR_CODE_NO_MEMORY_AVAILABLE 0
- #define STATIC STRINGS ERROR CODE INVALID STRING 1
- #define STATIC_STRINGS_ERROR_CODE_STRING_TOO_LONG 2
- #define STATIC_STRINGS_ERROR_CODE_SUBSTRING_START_INDEX_OUT_OF_RANGE 3
- #define STATIC_STRINGS_ERROR_CODE_SUBSTRING_FINISH_INDEX_OUT_OF_RANGE 4

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.

16 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_medium_strings_descriptors [STATIC_STRINGS_ME ← DIUM_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 static_strings.c File Reference

Strings allocation with static memory.

```
#include "static strings.h"
```

Functions

· void static strings init ()

Link the descriptors with the arrays and initialize the status as deallocated.

• 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 \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.

• 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 \(\cup \text{IC_STRINGS_VERY_LONG_STRING_SIZE.} \)

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. To get all the string from a start index use the length in the finish_index.

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'}

7.1.1 Detailed Description

Strings allocation with static memory.

7.1.2 Function Documentation

7.1.2.1 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.

7.1.2.2 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.

7.1.2.3 static_strings_concatenate()

Concatenate the second string at the end of the first in a new string. To get all the string from a start index use the length in the finish_index.

 $static_strings_string_descriptor \qquad static_strings_concatenate(static_strings_string_descriptor \qquad concatenate_\leftrightarrow \\ at, static_strings_string_descriptor * concatenate)$

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.

7.1.2.4 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.

7.1.2.5 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)

Parameters

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

Returns

1 if the string is found, 0 if not.

7.1.2.6 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.

7.1.2.7 static_strings_deallocate()

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

 $\textbf{void} \ static_strings_deallocate(static_strings_string_descriptor) * string_descriptor)$

Parameters

string_descrip	A pointer to the string descriptor to deallocat	e.
----------------	---	----

7.1.2.8 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.

7.1.2.9 static_strings_float_to_string()

```
{\tt static\_strings\_string\_descriptor* static\_strings\_float\_to\_string \ (} \\ {\tt float\_arg} \ )
```

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.
	1

Returns

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

7.1.2.10 static_strings_init()

```
void static_strings_init ( )
```

Link the descriptors with the arrays and initialize the status as deallocated.

void static_strings_init()

7.1.2.11 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.

7.1.2.12 static_strings_int32_to_string()

```
 \begin{array}{c} \textbf{static\_strings\_string\_descriptor* static\_strings\_int32\_to\_string (} \\ \textbf{int32\_t } int32 \ ) \end{array}
```

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.

7.1.2.13 static_strings_int8_to_string()

```
\begin{tabular}{ll} {\tt static\_strings\_string\_descriptor*} & {\tt static\_strings\_int8\_to\_string} & ( \\ & {\tt int8\_t} & {\tt 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.

7.1.2.14 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

Suring A pointer to the string descriptor.	string	A pointer to the string descriptor.
--	--------	-------------------------------------

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.

7.1.2.15 static strings save()

Calculate the string size, allocate memory, copy the string and set the size. String must end with \n 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.

static_strings_string_descriptor *static_strings_save(uint8_t *string)

Parameters

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

Returns

A pointer to the string descriptor, if NULL check static_strings_error_code.

7.1.2.16 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

atala a da a adata a	A contract of the state of the
strina aescriptor	A pointer to a pointer to a string descriptor that will contain the token.
cumg_accompten	The point of to a point of to a ouring accompton that will contain the torte

Returns

1 if success or 0 if no token is available.

7.1.2.17 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.

7.1.2.18 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.	
---------------------------------	--

Returns

Length of the string in uint16_t. If 0 check static_strings_error_code.

7.1.2.19 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.

7.1.2.20 static_strings_uint16_to_string()

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 integer.
--------	---------------------------

Returns

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

7.1.2.21 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

ſ	uint32	32 bits unsigned integer.
---	--------	---------------------------

Returns

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

7.1.2.22 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.

7.1.3 Variable Documentation

7.1.3.1 static_strings_string_splitter

```
static_strings_string_splitter_parameters static_strings_string_splitter = {NULL,'\0'}
```

Parameters to static_strings_string_splitter_get_next_token function. Initialized in null and \0.

7.2 static_strings.h File Reference

Strings allocation with static memory.

```
#include "stm32f1xx_hal.h"
#include "string.h"
#include "int_types.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_stirngs_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 1
- #define STATIC STRINGS VERY LONG STRING SIZE 1000
- #define STATIC_STRINGS_VERY_LONG_STRING_QUANTITY 1
- #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 MEMORY AVAILABLE 0
- #define STATIC_STRINGS_ERROR_CODE_INVALID_STRING 1
- #define STATIC_STRINGS_ERROR_CODE_STRING_TOO_LONG 2
- #define STATIC STRINGS ERROR CODE SUBSTRING START INDEX OUT OF RANGE 3
- #define STATIC_STRINGS_ERROR_CODE_SUBSTRING_FINISH_INDEX_OUT_OF_RANGE 4

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.

• 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\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.

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 \(\cdots \) 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. To get all the string from a start index use the length in the finish index.

• 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]

7.2.1 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)

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 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.

7.2.2.3 static_strings_concatenate()

Concatenate the second string at the end of the first in a new string. To get all the string from a start index use the length in the finish_index.

 $static_strings_string_descriptor \qquad static_strings_concatenate(static_strings_string_descriptor \qquad concatenate_\leftrightarrow \\ at, static_strings_string_descriptor* concatenate)$

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.

7.2.2.4 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.

7.2.2.5 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)

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.

7.2.2.6 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.

7.2.2.7 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.

7.2.2.8 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.

7.2.2.9 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

Returns

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

7.2.2.10 static_strings_init()

```
void static_strings_init ( )
```

Link the descriptors with the arrays and initialize the status as deallocated.

void static_strings_init()

7.2.2.11 static_strings_int16_to_string()

```
\begin{tabular}{ll} {\tt static\_strings\_string\_descriptor*} & {\tt static\_strings\_int16\_to\_string} & ( \\ & {\tt int16\_t} & {\tt int16} & ) \\ \end{tabular}
```

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.

7.2.2.12 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 integer.
-------	-------------------------

Returns

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

7.2.2.13 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.

7.2.2.14 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

```
string A pointer to the string descriptor.
```

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.

7.2.2.15 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

string A pointer to the s

Returns

A pointer to the string descriptor, if NULL check static_strings_error_code.

7.2.2.16 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

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

Returns

1 if success or 0 if no token is available.

7.2.2.17 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.

7.2.2.18 static_strings_strlen()

Calculate the length of a string that ends with \r\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.
--------	--------------------------

Returns

Length of the string in uint16_t. If 0 check static_strings_error_code.

7.2.2.19 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.

7.2.2.20 static_strings_uint16_to_string()

```
 \begin{array}{c} \textbf{static\_strings\_string\_descriptor*} \ \ \textbf{static\_strings\_uint16\_to\_string} \ \ (\\ \textbf{uint16\_t} \ \ 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 integer.
--------	---------------------------

Returns

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

7.2.2.21 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.

7.2.2.22 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.

7.2.3 Variable Documentation

7.2.3.1 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, 14	static_strings_uint32_to_string, 40
static_strings_error_code, 14	static_strings_uint8_to_string, 40
	static_strings_allocate
Static memory arrays, 15	static_strings.c, 20
static_strings.c, 19	static_strings.h, 32
static_strings_allocate, 20	static_strings_compare
static_strings_compare, 21	static_strings.c, 21
static_strings_concatenate, 21	static_strings.h, 32
static_strings_contains_char, 22	static strings concatenate
static_strings_contains_string, 22	static_strings.c, 21
static_strings_create_custom_string, 22	static_strings.h, 33
static_strings_deallocate, 23	static_strings_contains_char
static_strings_double_to_string, 23	static_strings.c, 22
static_strings_float_to_string, 24	static_strings.h, 33
static_strings_init, 24	static_strings_contains_string
static_strings_int16_to_string, 24	static_strings.c, 22
static_strings_int32_to_string, 25	static_strings.h, 34
static_strings_int8_to_string, 25	static_strings_create_custom_string
static_strings_is_line, 25	static_strings.c, 22
static_strings_save, 26	static_strings.b, 34
static_strings_string_splitter, 29	static_strings_deallocate
static_strings_string_splitter_get_next_token, 26	static_strings_deallocate
static_strings_string_splitter_set_parameters, 26	static_strings.c, 25 static_strings.h, 35
static_strings_strlen, 27	
static_strings_substring, 27	static_strings_double_to_string
static_strings_uint16_to_string, 28	static_strings.c, 23
static_strings_uint32_to_string, 28	static_strings.h, 35
static_strings_uint8_to_string, 28	static_strings_error_code
static_strings.h, 29	Error handling, 14
static_strings_allocate, 32	static_strings_float_to_string
static_strings_compare, 32	static_strings.c, 24
static_strings_concatenate, 33	static_strings.h, 35
static_strings_contains_char, 33	static_strings_init
static_strings_contains_string, 34	static_strings.c, 24
static_strings_create_custom_string, 34	static_strings.h, 36
static_strings_deallocate, 35	static_strings_int16_to_string
static_strings_double_to_string, 35	static_strings.c, 24
static_strings_float_to_string, 35	static_strings.h, 36
static_strings_init, 36	static_strings_int32_to_string
static_strings_int16_to_string, 36	static_strings.c, 25
static_strings_int32_to_string, 36	static_strings.h, 36
static_strings_int8_to_string, 37	static_strings_int8_to_string
static_strings_is_line, 37	static_strings.c, 25
static_strings_save, 37	static_strings.h, 37
static_strings_string_splitter, 41	static_strings_is_line
static_strings_string_splitter_get_next_token, 38	static_strings.c, 25
static_strings_string_splitter_set_parameters, 38	static_strings.h, 37
static_strings_strlen, 39	static_strings_save
static_strings_substring, 39	static_strings.c, 26
static_strings_uint16_to_string, 40	static_strings.h, 37

44 INDEX

```
static_strings_string_descriptor, 17
static_strings_string_splitter
     static_strings.c, 29
     static_strings.h, 41
static_strings_string_splitter_get_next_token
     static strings.c, 26
     static_strings.h, 38
static_strings_string_splitter_parameters, 17
static_strings_string_splitter_set_parameters
     static strings.c, 26
     static_strings.h, 38
static_strings_strlen
     static_strings.c, 27
     static_strings.h, 39
static_strings_substring
     static_strings.c, 27
     static strings.h, 39
static_strings_uint16_to_string
     static_strings.c, 28
     static_strings.h, 40
static_strings_uint32_to_string
     static_strings.c, 28
     static_strings.h, 40
static_strings_uint8_to_string
     static_strings.c, 28
     static_strings.h, 40
String descriptors, 16
String status, 13
String types, 12
String types size and quantity, 11
```