Statics Strings STM32F1XX

Generated by Doxygen 1.8.18

1 Static Strings	1
2 Module Index	3
2.1 Modules	3
3 Data Structure Index	5
3.1 Data Structures	5
4 File Index	7
4.1 File List	7
5 Module Documentation	9
5.1 String types size and quantity	9
5.1.1 Detailed Description	9
5.2 String types	10
5.2.1 Detailed Description	10
5.3 String status	11
5.3.1 Detailed Description	
5.4 Error handling	12
5.4.1 Detailed Description	
5.4.2 Variable Documentation	12
5.4.2.1 static_strings_error_code	12
5.5 Static memory arrays	
5.5.1 Detailed Description	
5.6 String descriptors	
5.6.1 Detailed Description	14
6 Data Structure Documentation	15
6.1 static_strings_string_descriptor Struct Reference	15
6.1.1 Detailed Description	15
6.2 static_strings_string_splitter_parameters Struct Reference	15
6.2.1 Detailed Description	
7 File Documentation	17
7.1 static_strings.c File Reference	17
7.1.1 Detailed Description	
7.1.2 Function Documentation	
7.1.2.1 static_strings_allocate()	
7.1.2.2 static_strings_create_custom_string()	
7.1.2.3 static_strings_deallocate()	
7.1.2.3 static_strings_ueallocate()	
7.1.2.5 static_strings_is_line()	
7.1.2.6 static_strings_save()	
7.1.2.7 static_strings_string_splitter_get_next_token()	
otatio_otinigo_otinig_opiittoi_got_noxt_toiten()	20

7.1.2.8 static_strings_string_splitter_set_parameters()	20
7.1.2.9 static_strings_strlen()	21
7.1.3 Variable Documentation	21
7.1.3.1 static_strings_string_splitter	21
7.2 static_strings.h 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_create_custom_string()	24
7.2.2.3 static_strings_deallocate()	24
7.2.2.4 static_strings_init()	25
7.2.2.5 static_strings_is_line()	25
7.2.2.6 static_strings_save()	25
7.2.2.7 static_strings_string_splitter_get_next_token()	26
7.2.2.8 static_strings_string_splitter_set_parameters()	26
7.2.2.9 static_strings_strlen()	26
7.2.3 Variable Documentation	27
7.2.3.1 static_strings_string_splitter	27
Index	29

Static Strings

Author

Ramsés F. Pérez

Date

August 2020

Version

1.0.1

Features:

- Developed for the STM32F103.
- · Global scope strings.
- 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.
- · String split.

2 Static Strings

GETTING STARTED

Suggested names

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

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 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 == NULD){
    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);
}
```

Module Index

2.1 Modules

Here is a list of all modules:

ring types size and quantity
ring types
ring status
rror handling
atic memory arrays
ring descriptors

4 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	15
static_strings_string_splitter_parameters	
Definition of the structure to hold the parameters to static_stirngs_string_splitter_get_next_token	
function	15

6 Data Structure Index

File Index

4.1 File List

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

static_strings.c	
Strings allocation with static memory	17
static_strings.h	
Strings allocation with static memory	21

8 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.

10 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.

12 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

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.

14 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 ← 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← 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.

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_create_custom_string()

Bind the provided string descriptor with the data of a string. String must end with \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.

7.1.2.3 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.

7.1.2.4 static_strings_init()

```
void static_strings_init ( )
```

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

void static_strings_init()

7.1.2.5 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.

7.1.2.6 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

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

Returns

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

7.1.2.7 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.

void static strings string splitter set parameters(static strings string descriptor, wint8 t delimiter)

Parameters

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

Returns

1 if success or 0 if no token is available.

7.1.2.8 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, wint8_t delimiter)

Parameters

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

7.1.2.9 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.

7.1.3 Variable Documentation

7.1.3.1 static_strings_string_splitter

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

7.2 static_strings.h File Reference

Strings allocation with static memory.

```
#include "stm32f1xx_hal.h"
#include "string.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

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 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 \n or \n 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 \(\cup \C \) 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.

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_very_short_strings_descriptors [STATIC_STRINGS_V ← ERY SHORT STRING QUANTITY]
- 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]
- static_strings_string_descriptor static_strings_very_long_strings_descriptors [STATIC_STRINGS_VE ← RY_LONG_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 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.

7.2.2.3 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.
-------------------	---

7.2.2.4 static_strings_init()

```
void static_strings_init ( )
```

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

void static strings init()

7.2.2.5 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.

7.2.2.6 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 string start.

Returns

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

7.2.2.7 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.

void static_strings_string_splitter_set_parameters(static_strings_string_descriptor *string_descriptor,uint8_t delimiter)

Parameters

Returns

1 if success or 0 if no token is available.

7.2.2.8 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, wint8 t delimiter)

Parameters

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

7.2.2.9 static_strings_strlen()

Calculate the length of a string that ends with \r 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

 $Length\ of\ the\ string\ in\ uint 16_t.\ If\ 0\ check\ static_strings_error_code.$

7.2.3 Variable Documentation

7.2.3.1 static_strings_string_splitter

 $\verb|static_strings_string_splitter_parameters| | \verb|static_strings_string_splitter| |$

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

Index

```
Error handling, 12
                                                             static_strings_string_splitter_get_next_token
     static_strings_error_code, 12
                                                                  static_strings.c, 20
                                                                  static strings.h, 25
Static memory arrays, 13
                                                             static_strings_string_splitter_parameters, 15
static strings.c, 17
                                                             static_strings_string_splitter_set_parameters
     static strings allocate, 18
                                                                  static_strings.c, 20
     static_strings_create_custom_string, 18
                                                                  static strings.h, 26
     static_strings_deallocate, 19
                                                             static_strings_strlen
     static_strings_init, 19
                                                                  static_strings.c, 21
     static_strings_is_line, 19
                                                                  static strings.h, 26
     static_strings_save, 19
                                                             String descriptors, 14
     static_strings_string_splitter, 21
                                                             String status, 11
     static strings string splitter get next token, 20
                                                             String types, 10
     static_strings_string_splitter_set_parameters, 20
                                                             String types size and quantity, 9
     static_strings_strlen, 21
static_strings.h, 21
     static strings allocate, 23
     static strings create custom string, 24
     static_strings_deallocate, 24
     static_strings_init, 24
     static_strings_is_line, 25
     static strings save, 25
     static_strings_string_splitter, 27
     static strings string splitter get next token, 25
     static strings string splitter set parameters, 26
     static_strings_strlen, 26
static_strings_allocate
     static strings.c, 18
     static strings.h, 23
static_strings_create_custom_string
     static_strings.c, 18
     static_strings.h, 24
static strings deallocate
     static_strings.c, 19
     static_strings.h, 24
static strings error code
     Error handling, 12
static_strings_init
     static_strings.c, 19
     static_strings.h, 24
static_strings_is_line
     static_strings.c, 19
     static strings.h, 25
static strings save
     static strings.c, 19
     static strings.h, 25
static strings string descriptor, 15
static strings string splitter
     static_strings.c, 21
     static_strings.h, 27
```