Statics Strings

STM32F1XX

Generated by Doxygen 1.8.18

Chapter 1

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.

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 == 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);
}
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

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.

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

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.

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_very_short_string_memory** [STATIC_STRINGS_VERY_SHORT_STRING_QU↩ANTITY][STATIC_STRINGS_VERY_SHORT_STRING_SIZE]
- uint8_t **static_strings_short_string_memory** [STATIC_STRINGS_SHORT_STRING_QUANTITY][STA↩TIC_STRINGS_SHORT_STRING_SIZE]
- uint8_t **static_strings_medium_string_memory** [STATIC_STRINGS_MEDIUM_STRING_QUANTI↩TY][STATIC_STRINGS_MEDIUM_STRING_SIZE]
- uint8_t **static_strings_long_string_memory** [STATIC_STRINGS_LONG_STRING_QUANTITY][STATI↩C_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]

5.5.1 Detailed Description

Static memory arrays to allocate strings.

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_short_strings_descriptors** [STATIC_STRINGS_SHOR↩ T_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.

Chapter 6

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

Chapter 7

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\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←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()

```
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_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()

```
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_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()

```
void static_strings_deallocate (
            static_strings_string_descriptor * string_descriptor )
```

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()

```
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.

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()

```
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.

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.

7.1.2.7 static_strings_string_splitter_get_next_token()

```
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.

void static_strings_string_splitter_set_parameters(static_strings_string_descriptor ∗string_descriptor,uint8_t delimiter)

Parameters

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

Returns

1 if success or 0 if no token is available.

7.1.2.8 static_strings_string_splitter_set_parameters()

```
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.

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.

7.1.2.9 static_strings_strlen()

```
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 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'}
```

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"
```

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\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←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_long_strings_descriptors** [STATIC_STRINGS_LONG_S←TRING_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()

```
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_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()

```
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_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()

```
void static_strings_deallocate (
            static_strings_string_descriptor * string_descriptor )
```

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()

```
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.

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()

```
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.

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()

```
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.

void static_strings_string_splitter_set_parameters(static_strings_string_descriptor ∗string_descriptor,uint8_t delimiter)

Parameters

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

Returns

1 if success or 0 if no token is available.

7.2.2.8 static_strings_string_splitter_set_parameters()

```
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.

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.

7.2.2.9 static_strings_strlen()

```
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 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.

7.2.3 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