

Erriez Serial Terminal library for Arduino

1.1.1

Generated by Doxygen 1.8.11

Contents

1	Serial Terminal library for Arduino	1
2	Class Index	5
2.1	Class List	5
3	File Index	7
3.1	File List	7
4	Class Documentation	9
4.1	SerialTerminal Class Reference	9
4.1.1	Detailed Description	9
4.1.2	Constructor & Destructor Documentation	9
4.1.2.1	SerialTerminal(char newlineChar='\n', char delimiterChar=' ')	9
4.1.3	Member Function Documentation	10
4.1.3.1	addCommand(const char *command, void(*function)())	10
4.1.3.2	getNext()	10
4.1.3.3	getRemaining()	10
4.1.3.4	readSerial()	11
4.1.3.5	setDefaultHandler(void(*function)(const char *))	11
5	File Documentation	13
5.1	ErriezSerialTerminal.cpp File Reference	13
5.1.1	Detailed Description	13
5.2	ErriezSerialTerminal.h File Reference	13
5.2.1	Detailed Description	13
	Index	15

Chapter 1

Serial Terminal library for Arduino

This is a universal Serial Terminal library for Arduino to parse ASCII commands and arguments.

Hardware

Any Arduino hardware with a serial port.

Examples

Arduino IDE | Examples | Erriez Serial Terminal |

- [SerialTerminal](#)

Documentation

- [Online HTML](#)
- [Download PDF](#)

Usage

Initialization

Create a Serial Terminal object. This can be initialized with optional newline and delimiter characters.

Default newline character: `'\n'` Default delimiter character: `Space`

```

1 {c++}
2 #include <ErriezSerialTerminal.h>
3
4 // Newline character '\r' or '\n'
5 char newlineChar = '\n';
6 // Separator character between commands and arguments
7 char delimiterChar = ' ';
8
9 // Create serial terminal object
10 SerialTerminal term(newlineChar, delimiterChar);
11
12
13 void setup()
14 {
15     // Initialize serial port
16     Serial.begin(115200);
17
18     // Initialize the built-in LED
19     pinMode(LED_BUILTIN, OUTPUT);
20     digitalWrite(LED_BUILTIN, LOW);
21 }

```

Register new commands

Commands must be registered at startup with a corresponding `callback handler` . This registers the command only, excluding arguments.

The callback handler will be called when the command has been received including the newline character.

An example of registering multiple commands:

```

1 {c++}
2 void setup()
3 {
4     ...
5
6     // Add command callback handlers
7     term.addCommand("?", cmdHelp);
8     term.addCommand("help", cmdHelp);
9     term.addCommand("on", cmdLedOn);
10    term.addCommand("off", cmdLedOff);
11 }
12
13 void cmdHelp()
14 {
15     // Print usage
16     Serial.println(F("Serial terminal usage:"));
17     Serial.println(F("  help or ?      Print this usage"));
18     Serial.println(F("  on             Turn LED on"));
19     Serial.println(F("  off            Turn LED off"));
20 }
21
22 void cmdLedOn()
23 {
24     // Turn LED on
25     Serial.println(F("LED on"));
26     digitalWrite(LED_BUILTIN, HIGH);
27 }
28
29 void cmdLedOff()
30 {
31     // Turn LED off
32     Serial.println(F("LED off"));
33     digitalWrite(LED_BUILTIN, LOW);
34 }

```

Set default handler

Optional: The default handler will be called when the command is not recognized.

```

1 {c++}
2 void setup()
3 {
4     ...
5
6     // Set default handler for unknown commands

```

```
7     term.setDefaultHandler(unknownCommand);
8 }
9
10 void unknownCommand(const char *command)
11 {
12     // Print unknown command
13     Serial.print(F("Unknown command: "));
14     Serial.println(command);
15 }
```

Read from serial port

Read from the serial port in the main loop:

```
1 {c++}
2 void loop()
3 {
4     // Read from serial port and handle command callbacks
5     term.readSerial();
6 }
```

Get next argument

Get pointer to next argument in serial receive buffer:

```
1 {c++}
2 char *arg;
3
4 // Get next argument
5 arg = term.getNext();
6 if (arg != NULL) {
7     Serial.print(F("Argument: "));
8     Serial.println(arg);
9 } else {
10     Serial.println(F("No argument"));
11 }
```

Get remaining characters

Get pointer to remaining characters in serial receive buffer:

```
1 {c++}
2 char *arg;
3
4 // Get remaining characters
5 arg = term.getRemaining();
6 if (arg != NULL) {
7     Serial.print(F("Remaining: "));
8     Serial.println(arg);
9 }
```

Clear buffer

Optional: The serial receive buffer can be cleared with the following call:

```
1 {c++}
2 term.clearBuffer();
```

Library configuration

SerialTerminal.h contains the following configuration macro's:

- `ST_RX_BUFFER_SIZE` : The default serial receive buffer size is 32 Bytes. This includes the command and arguments, excluding the `'\0'` character.
- `ST_NUM_COMMAND_CHARS`: The default number of command characters is 8 Bytes, excluding the `'\0'` character.

Library dependencies

- None.

Library installation

Please refer to the [Wiki](#) page.

Other Arduino Libraries and Sketches from Erriez

- [Erriez Libraries and Sketches](#)

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

SerialTerminal	
SerialTerminal class	9

Chapter 3

File Index

3.1 File List

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

ErriezSerialTerminal.cpp	
Serial terminal library for Arduino	13
ErriezSerialTerminal.h	
Serial terminal library for Arduino	13

Chapter 4

Class Documentation

4.1 SerialTerminal Class Reference

[SerialTerminal](#) class.

```
#include <ErriezSerialTerminal.h>
```

Public Member Functions

- [SerialTerminal](#) (char newlineChar='\n', char delimiterChar=' ')
SerialTerminal constructor.
- void [addCommand](#) (const char *command, void(*function)())
Add command with callback handler.
- void [setDefaultHandler](#) (void(*function)(const char *))
Set default callback handler for unknown commands.
- void [readSerial](#) ()
Read from serial port.
- void [clearBuffer](#) ()
Clear serial receive buffer.
- char * [getNext](#) ()
Get next argument.
- char * [getRemaining](#) ()
Get all remaining characters from serial buffer.

4.1.1 Detailed Description

[SerialTerminal](#) class.

Definition at line 52 of file ErriezSerialTerminal.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 [SerialTerminal::SerialTerminal](#) (char *newlineChar* = '\n', char *delimiterChar* = ' ') [explicit]

[SerialTerminal](#) constructor.

Parameters

<i>newlineChar</i>	Newline character ' <code>\r</code> ' or ' <code>\n</code> '. Default: ' <code>\n</code> '.
<i>delimiterChar</i>	Delimiter separator character between commands and arguments. Default: space.

Definition at line 44 of file ErriezSerialTerminal.cpp.

4.1.3 Member Function Documentation

4.1.3.1 `void SerialTerminal::addCommand (const char * command, void(*)() function)`

Add command with callback handler.

Parameters

<i>command</i>	Register a null-terminated ASCII command.
<i>function</i>	The function to be called when receiving the <i>command</i> .

Definition at line 66 of file ErriezSerialTerminal.cpp.

4.1.3.2 `char * SerialTerminal::getNext ()`

Get next argument.

Returns

Address: Pointer to next argument
NULL: No argument available

Definition at line 154 of file ErriezSerialTerminal.cpp.

4.1.3.3 `char * SerialTerminal::getRemaining ()`

Get all remaining characters from serial buffer.

Returns

Address: Pointer to remaining characters in serial receive buffer.
NULL: No remaining data available.

Definition at line 165 of file ErriezSerialTerminal.cpp.

4.1.3.4 void SerialTerminal::readSerial ()

Read from serial port.

Process command when newline character has been received.

Definition at line 98 of file ErriezSerialTerminal.cpp.

4.1.3.5 void SerialTerminal::setDefaultHandler (void(*) (const char *) *function*)

Set default callback handler for unknown commands.

Store default callback handler which will be called when receiving an unknown command.

Parameters

<i>function</i>	Address of the default handler. This function will be called when the command is not recognized. The parameter contains the first ASCII command.
-----------------	---

Definition at line 88 of file ErriezSerialTerminal.cpp.

The documentation for this class was generated from the following files:

- [ErriezSerialTerminal.h](#)
- [ErriezSerialTerminal.cpp](#)

Chapter 5

File Documentation

5.1 ErriezSerialTerminal.cpp File Reference

Serial terminal library for Arduino.

```
#include "ErriezSerialTerminal.h"
```

5.1.1 Detailed Description

Serial terminal library for Arduino.

Source: <https://github.com/Erriez/ErriezSerialTerminal> Documentation: <https://erriez.github.io/ErriezSerialTerminal>

5.2 ErriezSerialTerminal.h File Reference

Serial terminal library for Arduino.

```
#include <Arduino.h>
#include <string.h>
```

Classes

- class [SerialTerminal](#)
SerialTerminal class.

Macros

- #define [ST_RX_BUFFER_SIZE](#) 32
Size of the serial receive buffer in bytes (Maximum length of one command plus arguments)
- #define [ST_NUM_COMMAND_CHARS](#) 8
Number of command characters.

5.2.1 Detailed Description

Serial terminal library for Arduino.

Source: <https://github.com/Erriez/ErriezSerialTerminal> Documentation: <https://erriez.github.io/ErriezSerialTerminal>

Index

- addCommand
 - SerialTerminal, [10](#)
- ErriezSerialTerminal.cpp, [13](#)
- ErriezSerialTerminal.h, [13](#)
- getNext
 - SerialTerminal, [10](#)
- getRemaining
 - SerialTerminal, [10](#)
- readSerial
 - SerialTerminal, [10](#)
- SerialTerminal, [9](#)
 - addCommand, [10](#)
 - getNext, [10](#)
 - getRemaining, [10](#)
 - readSerial, [10](#)
 - SerialTerminal, [9](#)
 - setDefaultHandler, [11](#)
- setDefaultHandler
 - SerialTerminal, [11](#)