

# Erriez Serial Terminal library for Arduino

1.1.2

Generated by Doxygen 1.8.17



---

<b>1 Serial Terminal library for Arduino</b>	<b>1</b>
<b>2 Class Index</b>	<b>5</b>
2.1 Class List . . . . .	5
<b>3 File Index</b>	<b>7</b>
3.1 File List . . . . .	7
<b>4 Class Documentation</b>	<b>9</b>
4.1 SerialTerminal Class Reference . . . . .	9
4.1.1 Detailed Description . . . . .	9
4.1.2 Constructor & Destructor Documentation . . . . .	10
4.1.2.1 SerialTerminal() . . . . .	10
4.1.3 Member Function Documentation . . . . .	10
4.1.3.1 addCommand() . . . . .	10
4.1.3.2 getNext() . . . . .	10
4.1.3.3 getRemaining() . . . . .	11
4.1.3.4 readSerial() . . . . .	11
4.1.3.5 setDefaultHandler() . . . . .	11
4.1.3.6 setPostCommandHandler() . . . . .	12
4.1.3.7 setSerialEcho() . . . . .	12
<b>5 File Documentation</b>	<b>13</b>
5.1 src/ErriezSerialTerminal.cpp File Reference . . . . .	13
5.1.1 Detailed Description . . . . .	13
5.2 src/ErriezSerialTerminal.h File Reference . . . . .	14
5.2.1 Detailed Description . . . . .	14
<b>Index</b>	<b>15</b>



## 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 |

- [ErriezSerialTerminal](#)

## 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

```
{c++}
#include <ErriezSerialTerminal.h>
// Newline character '\r' or '\n'
char newlineChar = '\n';
// Separator character between commands and arguments
char delimiterChar = ' ';
// Create serial terminal object
SerialTerminal term(newlineChar, delimiterChar);
void setup()
{
    // Initialize serial port
    Serial.begin(115200);

    // Initialize the built-in LED
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, LOW);
}
```

### 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:

```
{c++}
void setup()
{
    ...
    // Add command callback handlers
    term.addCommand("?", cmdHelp);
    term.addCommand("help", cmdHelp);
    term.addCommand("on", cmdLedOn);
    term.addCommand("off", cmdLedOff);
}
void cmdHelp()
{
    // Print usage
    Serial.println(F("Serial terminal usage:"));
    Serial.println(F("  help or ?      Print this usage"));
    Serial.println(F("  on             Turn LED on"));
    Serial.println(F("  off            Turn LED off"));
}
void cmdLedOn()
{
    // Turn LED on
    Serial.println(F("LED on"));
    digitalWrite(LED_BUILTIN, HIGH);
}
void cmdLedOff()
{
    // Turn LED off
    Serial.println(F("LED off"));
    digitalWrite(LED_BUILTIN, LOW);
}
```

### Set default handler

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

```
{c++}
void setup()
{
    ...
    // Set default handler for unknown commands
    term.setDefaultHandler(unknownCommand);
}
void unknownCommand(const char *command)
{
    // Print unknown command
}
```

```

    Serial.print(F("Unknown command: "));
    Serial.println(command);
}

```

## Read from serial port

Read from the serial port in the main loop:

```

{c++}
void loop()
{
    // Read from serial port and handle command callbacks
    term.readSerial();
}

```

## Get next argument

Get pointer to next argument in serial receive buffer:

```

{c++}
char *arg;
// Get next argument
arg = term.getNext();
if (arg != NULL) {
    Serial.print(F("Argument: "));
    Serial.println(arg);
} else {
    Serial.println(F("No argument"));
}

```

## Get remaining characters

Get pointer to remaining characters in serial receive buffer:

```

{c++}
char *arg;
// Get remaining characters
arg = term.getRemaining();
if (arg != NULL) {
    Serial.print(F("Remaining: "));
    Serial.println(arg);
}

```

## Clear buffer

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

```

{c++}
term.clearBuffer();

```

## Enable/Disable Character Echoing

Optional: Allow for any entered characters to be printed back to the Serial interface. This is useful for terminal programs like PuTTY.

```

{c++}
term.setSerialEcho(true); //Enable Character Echoing

```

## Set Post Command Handler

Optional: Add a function to be called AFTER a command has been handled.

```

{c++}
void setup()
{
    ...
    // Set handler to be run AFTER a command has been handled.
    term.setDefaultHandler(postCommandHandler);
}
void setPostCommandHandler()
{
    // Print '>' for a primitive user UI
    Serial.print(F("> "));
}

```

## 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:

<a href="#">SerialTerminal</a>	
<a href="#">SerialTerminal</a> class	9



## Chapter 3

# File Index

### 3.1 File List

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

src/ <a href="#">ErriezSerialTerminal.cpp</a>	
Serial terminal library for Arduino . . . . .	13
src/ <a href="#">ErriezSerialTerminal.h</a>	
Serial terminal library for Arduino . . . . .	14



## 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 [setSerialEcho](#) (bool doEcho)  
*Set the control state to echo any printable chars to the console.*
- void [setPostCommandHandler](#) (void(\*function)(void))  
*Set post command handler callback for after all handled 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::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 addCommand()

```
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 <code>command</code> .

Definition at line 66 of file `ErriezSerialTerminal.cpp`.

### 4.1.3.2 getNext()

```
char * SerialTerminal::getNext ( )
```

Get next argument.

**Returns**

Address: Pointer to next argument  
NULL: No argument available

Definition at line 194 of file ErriezSerialTerminal.cpp.

**4.1.3.3 getRemaining()**

```
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 205 of file ErriezSerialTerminal.cpp.

**4.1.3.4 readSerial()**

```
void SerialTerminal::readSerial ( )
```

Read from serial port.

Process command when newline character has been received.

Definition at line 123 of file ErriezSerialTerminal.cpp.

**4.1.3.5 setDefaultHandler()**

```
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 113 of file ErriezSerialTerminal.cpp.

#### 4.1.3.6 setPostCommandHandler()

```
void SerialTerminal::setPostCommandHandler (
    void(*) (void) function )
```

Set post command handler callback for after all handled commands.

Store post command handler which will be called when after executing any handled command.

##### Parameters

<i>function</i>	Address of the callback function. This function will be called after any handled event.
-----------------	---

Definition at line 100 of file ErriezSerialTerminal.cpp.

#### 4.1.3.7 setSerialEcho()

```
void SerialTerminal::setSerialEcho (
    bool doEcho )
```

Set the control state to echo any printable chars to the console.

Set the control state to echo any printable chars to the console.

##### Parameters

<i>doEcho</i>	Should all printable chars be echoed to the serial console?
---------------	---

Definition at line 88 of file ErriezSerialTerminal.cpp.

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

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



## Chapter 5

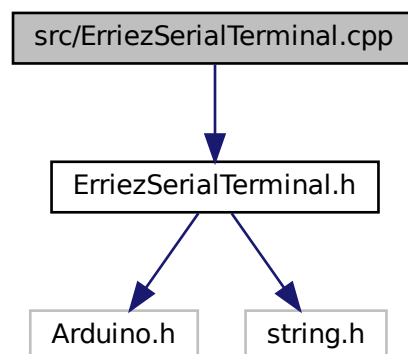
# File Documentation

### 5.1 src/ErriezSerialTerminal.cpp File Reference

Serial terminal library for Arduino.

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

Include dependency graph for ErriezSerialTerminal.cpp:



#### 5.1.1 Detailed Description

Serial terminal library for Arduino.

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

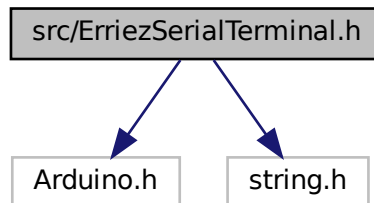
## 5.2 src/ErriezSerialTerminal.h File Reference

Serial terminal library for Arduino.

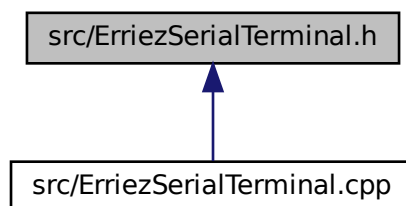
```
#include <Arduino.h>
```

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

Include dependency graph for ErriezSerialTerminal.h:



This graph shows which files directly or indirectly include this file:



### 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](#)
- getNext
  - SerialTerminal, [10](#)
- getRemaining
  - SerialTerminal, [11](#)
- readSerial
  - SerialTerminal, [11](#)
- SerialTerminal, [9](#)
  - addCommand, [10](#)
  - getNext, [10](#)
  - getRemaining, [11](#)
  - readSerial, [11](#)
  - SerialTerminal, [10](#)
  - setDefaultHandler, [11](#)
  - setPostCommandHandler, [12](#)
  - setSerialEcho, [12](#)
- setDefaultHandler
  - SerialTerminal, [11](#)
- setPostCommandHandler
  - SerialTerminal, [12](#)
- setSerialEcho
  - SerialTerminal, [12](#)
- src/ErriezSerialTerminal.cpp, [13](#)
- src/ErriezSerialTerminal.h, [14](#)