

← GO BACK

Learn

Arduino Ecosystem



Microcontrollers



Programming



Electronics



Communication



Hardware Design



Built-In Libraries



PDM Library

I2S Library

EEPROM Library

SoftwareSerial Library

Contributors

Home / Learn / EEPROM Library

ON THIS PAGE

EEPROM Library

Documentation for usage of the EEPROM library.

EEPROM is a memory whose values are kept when the board is powered off.

Author • Arduino

Last revision • 02/02/2024

Examples

Functions



read()



Description

Syntax

Parameters

Returns

Example

write()



Description

Syntax

Parameters

Returns

Example

Help

memory whose values are kept when the board is turned off (like a tiny hard drive). This library enables you to read and write those bytes.

The supported micro-controllers on the various Arduino and Genuino boards have different amounts of EEPROM: 1024 bytes on the ATmega328P, 512 bytes on the ATmega168 and ATmega8, 4 KB (4096 bytes) on the ATmega1280 and ATmega2560. The Arduino and Genuino 101 boards have an emulated EEPROM space of 1024 bytes.

To use this library

COPY

```
1 #include <EEPROM.h>
```

Examples

To see a list of examples for the EEPROM library, click the link below:

Functions

read()

Description

Reads a byte from the EEPROM. Locations that have never been written to have the value of 255.

Syntax

COPY

```
1 EEPROM.read(address)
```

Parameters

address: the location to read from, starting from 0 (int)

the value stored in that location (byte)

Example

```
1  #include <EEPROM.h>
2
3  int a = 0;
4  int value;
5
6  void setup()
7  {
8      Serial.begin(9600);
9  }
10
11 void loop()
12 {
13     value = EEPROM.read(a);
14
15     Serial.print(a);
16     Serial.print("\t");
17     Serial.print(value);
18     Serial.println();
19
20     a = a + 1;
21
22     if (a == 512)
23         a = 0;
24
25     delay(500);
26 }
```

write()

Write a byte to the EEPROM.

Syntax

COPY

```
1 EEPROM.write(address, value)
```

Parameters

address: the location to write to, starting from 0
(int)

value: the value to write, from 0 to 255 (byte)

Returns

none

Note: An EEPROM write takes 3.3 ms to complete. The EEPROM memory has a specified life of 100,000 write/erase cycles, so you may

Example

[COPY](#)

```
1 #include <EEPROM.h>
2
3 void setup()
4 {
5     for (int i = 0; i < 255; i++)
6         EEPROM.write(i, i);
7 }
8
9 void loop()
10 {
11 }
```

update()

Description

Write a byte to the EEPROM. The value is written only if differs from the one already saved at the same address.

COPY

```
1 EEPROM.update(address, value)
```

Parameters

address: the location to write to, starting from 0 (int)

value: the value to write, from 0 to 255 (byte)

Returns

none

Note: An EEPROM write takes 3.3 ms to complete. The EEPROM memory has a specified life of 100,000 write/erase cycles, so using this function instead of write() can save cycles if the written data does not change often

Example


```
1 #include <EEPROM.h>
2
3 void setup()
4 {
5     for (int i = 0; i < 255; i++) {
6         // this performs as EEPROM.wr
7         EEPROM.update(i, i);
8     }
9     for (int i = 0; i < 255; i++) {
10        // write value "12" to cell 3
11        // will not write the cell th
12        EEPROM.update(3, 12);
13    }
14 }
15
16 void loop()
17 {
18 }
```

get()

Description

Read any data type or object from the EEPROM.

```
1 EEPROM.get(address, data)
```

Parameters

address: the location to read from, starting from 0 (int)

data: the data to read, can be a primitive type (eg. float) or a custom struct

Returns

A reference to the data passed in

Example

```

1  #include <EEPROM.h>
2
3  struct MyObject{
4    float field1;
5    byte field2;
6    char name[10];
7  };
8
9  void setup(){
10
11    float f = 0.00f;    //Variable t
12    int eeAddress = 0; //EEPROM add
13
14    Serial.begin( 9600 );
15    while (!Serial) {
16      ; // wait for serial port to
17    }
18    Serial.print( "Read float from
19
20    //Get the float data from the E
21    EEPROM.get( eeAddress, f );
22    Serial.println( f, 3 ); //This
23
24    // get() can be used with custo
25    eeAddress = sizeof(float); //Mo
26    MyObject customVar; //Variable
27    EEPROM.get( eeAddress, customVa
28
29    Serial.println( "Read custom ob
30    Serial.println( customVar.field

```

Description

Write any data type or object to the EEPROM.

Syntax

COPY

```
1 EEPROM.put(address, data)
```

Parameters

address: the location to write to, starting from 0
(int)

data: the data to write, can be a primitive type
(eg. float) or a custom struct

Returns

A reference to the data passed in

value if it didn't change.

Example

```
1 #include <EEPROM.h>
2
3 struct MyObject {
4     float field1;
5     byte field2;
6     char name[10];
7 };
8
9 void setup() {
10
11     Serial.begin(9600);
12     while (!Serial) {
13         ; // wait for serial port to
14     }
15
16     float f = 123.456f; //Variable
17     int eeAddress = 0;  //Location
18
19
20     //One simple call, with the add
21     EEPROM.put(eeAddress, f);
22
23     Serial.println("Written float d
24
25     /** Put is designed for use wit
26
27     //Data to store.
28     MyObject customVar = {
29         3.14f,
```

Description

This operator allows using the identifier `EEPROM` like an array. EEPROM cells can be read and written directly using this method.

Syntax

COPY

```
1 EEPROM[address]
```

Parameters

address: the location to read/write from, starting from 0 (int)

Returns

A reference to the EEPROM cell

```

1  #include <EEPROM.h>
2
3  void setup(){
4
5      unsigned char val;
6
7      //Read first EEPROM cell.
8      val = EEPROM[ 0 ];
9
10     //Write first EEPROM cell.
11     EEPROM[ 0 ] = val;
12
13     //Compare contents
14     if( val == EEPROM[ 0 ] ){
15         //Do something...
16     }
17 }
18
19 void loop(){ /* Empty loop */ }
```

length()

This function returns an unsigned int containing the number of cells in the EEPROM.

This function returns an `unsigned int` containing the number of cells in the EEPROM.

Syntax

COPY

```
1 EEPROM.length()
```

Returns

Number of cells in the EEPROM as an `unsigned int`.

Suggested changes

The content on docs.arduino.cc is facilitated through a public [GitHub repository](#). You can read more on how to

Need support? License

Help Center
Ask the Arduino Forum
Discover Arduino
Discord

The Arduino documentation is licensed under the [Creative Commons Attribution-Share Alike 4.0](#) license.

