Serial

Набор функций **Serial** служит для связи устройства Ардуино с компьютером или другими устройствами, поддерживающими последовательный интерфейс обмена данными. Все платы Arduino имеют хотя бы один последовательный порт (UART, иногда называют USART). Для обмена данными **Serial** используют цифровые порты ввод/вывода 0 (RX) и 1 (TX), а также USB порт.

Serial.begin()

Инициирует последовательное соединение и задает скорость передачи данных в бит/с (бод). Для обмена данными с компьютером используйте следующие значения: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 или 115200. При соединение через порты вход/выхода 0 и 1 могут быть использованы другие значения скорости, требуемые устройством с которым будет осуществляться обмен данными.

Синтаксис

Serial.begin (speed)

Параметры

speed: скорость в бит/с

Пример

```
void setup() {
    Serial.begin(9600); // открывает последовательный порт, устанавливает скорость 9600 бит/с
}
```

Serial.end()

Закрывает последовательное соединение, порты RX и TX освобождаются и могут быть использованы для ввод/вывода. Для восстановления последовательного соединения используйте Serial.begin().

Синтаксис

Serial.end()

Параметры

нет

Serial.available()

Функция получает количество байт(символов) доступных для чтения из последовательного интерфейса связи.

Синтаксис

Serial.available()

```
    if (Serial.available() > 0) {  //если есть доступные данные
        // считываем байт
```

Serial.read()

Считывает очередной доступный байт из буфера последовательного соединения.

Синтаксис

Serial.read()

```
incomingByte = Serial.read();
```

Возвращает значение типа char или код из таблицы ASCI

```
int x = Serial.parseInt();
```

Возвращает значение типа int

Serial.print()

Синтаксис

Serial.print(val)
Serial.print(val, format)

Параметры

val: данные для передачи через последовательное соединение

format: базис для целых чисел или количество знаков после запятой для вещественных

Передает данные через последовательный порт как ASCII текст. Эта функция может принимать различные типы данных. Так целые числа выводятся соответствующими им символами ASCII. Вещественные выводятся с помощью двух ASCII символов, для целой и дробной части. Байты передаются как символ с соответствующим номером. Симоволы и строки отсылаются как есть.
Пример:
Serial.print(78) передается как "78"
Serial.print(1.23456) передается как "1.23"
Serial.print(byte(78)) передается как "N" (т.к. в таблице ASCII "N" под 78 номером)
Serial.print('N') передается как "N"
Serial.print("Hello world.") передается как "Hello world."
С помощью второго опционально параметра можно задать базис (систему счисления) для чисел. Допустимые значения BYTE, BIN (двоичный), OCT (восьмиричный), DEC (десятеричный), HEX (шестнадцатеричный). Для вещественных (дробных) чисел второй параметр задает количество знаков после запятой. Пример:
Serial.print(78, BYTE) выводит "N"
Serial.print(78, BIN) выводит "1001110"
Serial.print(78, OCT) выводит "116"
Serial.print(78, DEC) выводит "78"
Serial.print(78, HEX) выводит "4E"
Serial.println(1.23456, 0) выводит "1"
Serial.println(1.23456, 2) выводит "1.23"
Serial.println(1.23456, 4) выводит "1.2346"

Serial.println()

Передает данные через последовательное соединение как ASCII текст с следующим за ним символом переноса строки (ASCII символ 13 или '\r') и символом новой строки (ASCII 10 или '\n'). Параметры и типы данных для этой функции такие же, как и для Serial.print().

Синтаксис

Serial.println(val)
Serial.println(val, format)

Параметры

val: данные для передачи через последовательное соединение

format: базис для целых чисел или количество знаков после запятой для вещественных

Serial.write()

Функция передает данные как бинарный код через последовательное соединение. Данные послаются как один или серия байтов. Для того, чтобы передать данные как символы следует использовать другую функцию print().

Синтаксис

Serial.write(val)
Serial.write(str)
Serial.write(buf, len)

Параметры

val: один байт

str: строка как серия байт

buf: массив байт

len: длина массива