* Serial communication on pins TX/RX uses TTL logic levels (5V or 3.3V depending on the board). Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.
* Serial is used for communication between the Arduino board and a computer or other devices.
* All Arduino boards have at least one serial port (also known as a UART or USART)
* **Arduino Uno**: It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.
* You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().
* The **Arduino Mega** has three additional serial ports: **Serial1** on pins 19 (RX) and 18 (TX), **Serial2** on pins 17 (RX) and 16 (TX), **Serial3** on pins 15 (RX) and 14 (TX).
* To use them to communicate with an external TTL serial device, connect the TX pin to your device's RX pin, the RX to your device's TX pin, and the ground of your Mega to your device's ground.

Functions:

1. **if (Serial)**

#### Description

Indicates if the specified Serial port is ready.

#### Syntax

Arduino Uno:  
if (Serial)

Arduino Mega specific:   
if (Serial1)   
if (Serial2)   
if (Serial3)

#### Returns

boolean : returns true if the specified serial port is available.

## available()

#### Description

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes). available() inherits from the Stream utility class.

#### Syntax

Serial.available()

Arduino Mega only:   
Serial1.available()   
Serial2.available()   
Serial3.available()

#### Returns

the number of bytes available to read

## availableForWrite()

#### Description

Get the number of bytes (characters) available for writing in the serial buffer without blocking the write operation.

#### Syntax

Serial.availableForWrite()

Arduino Mega only:   
Serial1.availableForWrite()   
Serial2.availableForWrite()   
Serial3.availableForWrite()

#### Returns

the number of bytes available to write

## begin()

#### Description

Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200. You can, however, specify other rates - for example, to communicate over pins 0 and 1 with a component that requires a particular baud rate.

An optional second argument configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit.

#### Syntax

Serial.begin(speed)  
Serial.begin(speed, config)

Arduino Mega only:   
Serial1.begin(speed)   
Serial2.begin(speed)   
Serial3.begin(speed)   
Serial1.begin(speed, config)   
Serial2.begin(speed, config)   
Serial3.begin(speed, config)

#### Parameters

speed: in bits per second (baud) - long  
config: sets data, parity, and stop bits. Valid values are :

* SERIAL\_5N1
* SERIAL\_6N1
* SERIAL\_7N1
* SERIAL\_8N1 (the default)
* SERIAL\_5N2
* SERIAL\_6N2
* SERIAL\_7N2
* SERIAL\_8N2
* SERIAL\_5E1
* SERIAL\_6E1
* SERIAL\_7E1
* SERIAL\_8E1
* SERIAL\_5E2
* SERIAL\_6E2
* SERIAL\_7E2
* SERIAL\_8E2
* SERIAL\_5O1
* SERIAL\_6O1
* SERIAL\_7O1
* SERIAL\_8O1
* SERIAL\_5O2
* SERIAL\_6O2
* SERIAL\_7O2
* SERIAL\_8O2

#### Returns

nothing

### Example:

void setup() {  
    Serial.begin(9600); // opens serial port, sets data rate to 9600 bps  
}  
  
void loop() {}

**Arduino Mega example:**

// Arduino Mega using all four of its Serial ports   
// (Serial, Serial1, Serial2, Serial3),   
// with different baud rates:  
  
void setup(){  
  Serial.begin(9600);  
  Serial1.begin(38400);  
  Serial2.begin(19200);  
  Serial3.begin(4800);  
  
  Serial.println("Hello Computer");  
  Serial1.println("Hello Serial 1");  
  Serial2.println("Hello Serial 2");  
  Serial3.println("Hello Serial 3");  
}  
  
void loop() {}

## end()

#### Description

Disables serial communication, allowing the RX and TX pins to be used for general input and output. To re-enable serial communication, call [Serial.begin](https://www.arduino.cc/en/Serial/Begin)().

#### Syntax

Serial.end()

Arduino Mega only:   
Serial1.end()   
Serial2.end()   
Serial3.end()

## Serial.find()

#### Description

Serial.find() reads data from the serial buffer until the target string of given length is found. The function returns true if target string is found, false if it times out.

Serial.find() inherits from the [Stream](https://www.arduino.cc/en/Reference/Stream) utility class.

#### Syntax

Serial.find(target)

#### Parameters

target : the string to search for (char)

#### Returns

Boolean

## Serial.findUntil()

#### Description

Serial.findUntil() reads data from the serial buffer until a target string of given length or terminator string is found.

The function returns true if the target string is found, false if it times out.

Serial.findUntil() inherits from the [Stream](https://www.arduino.cc/en/Reference/Stream) utility class.

#### Syntax

Serial.findUntil(target, terminal)

#### Parameters

target : the string to search for (char)  
terminal : the terminal string in the search (char)

#### Returns

boolean

## flush()

#### Description

Waits for the transmission of outgoing serial data to complete. (Prior to Arduino 1.0, this instead removed any buffered incoming serial data.)

flush() inherits from the [Stream](https://www.arduino.cc/en/Reference/Stream) utility class.

#### Syntax

Serial.flush()

Arduino Mega only:   
Serial1.flush()   
Serial2.flush()   
Serial3.flush()