Arduino Serial Class Documentation

Serial.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 (http://arduino.cc/en/Reference/Stream) utility class.

Syntax:

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

Parameters:

none

Returns:

the number of bytes available to read

Example:

```
int incomingByte = 0; // for incoming serial data
void setup()
	{
		Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
	}
void loop()
	{
		// send data only when you receive data:
		if (Serial.available() > 0)
			{
				// read the incoming byte:
				incomingByte = Serial.read();
				// say what you got:
				Serial.print("I received: ");
				Serial.println(incomingByte, DEC);
			}
	}
```

Arduino Mega example:

```
void setup()

	{
		Serial.begin(9600); Serial1.begin(9600);
	}
void loop()
	{
		// read from port 0, send to port 1:
		if (Serial.available())
			{
				int inByte = Serial.read();
				Serial1.print(inByte, BYTE);
			}
```

```
        // read from port 1, send to port 0:
        if (Serial1.available())
              {
                    int inByte = Serial1.read();
                    Serial.print(inByte, BYTE);
              }
}
```

Serial.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() {}
```

Serial.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 (http://arduino.cc/en/Serial/Begin)().

Syntax:

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

Parameters:

none

Returns:

nothing

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 (http://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 (http://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

Serial.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 (http://arduino.cc/en/Reference/Stream) utility class.

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

Parameters:
none

Returns:
nothing

Serial.parseFloat() Description

Serial.parseFloat() returns the first valid floating point number from the Serial buffer. Characters that are not digits (or the minus sign) are skipped. parseFloat() is terminated by the first character that is not a floating point number.

Serial.parseFloat() inherits from the Stream (http://arduino.cc/en/Reference/Stream) utility class.

Syntax:
Serial.parseFloat()

Parameters:
none

Returns:
float

Serial.parseInt() Description

Looks for the next valid integer in the incoming serial stream. parseInt() inherits from the Stream (http://arduino.cc/en/Reference/Stream) utility class.

In particular:

- Initial characters that are not digits or a minus sign, are skipped;
- Parsing stops when no characters have been read for a configurable time-out value, or a non-digit is read;
- If no valid digits were read when the time-out (see Serial.setTimeout (http://arduino.cc/en/Serial/SetTimeout)()) occurs, 0 is returned;

Syntax:

Serial.parseInt() Serial.parseInt(char skipChar)
Arduino Mega only: Serial1.parseInt() Serial2.parseInt() Serial3.parseInt()

Parameters:

skipChar: used to skip the indicated char in the search. Used for example to skip thousands divider.

Returns:

long : the next valid integer

Serial.peek() Description

Returns the next byte (character) of incoming serial data without removing it from the internal serial buffer. That is, successive calls to peek() will return the same character, as will the next call to read(). peek() inherits from the Stream (http://arduino.cc/en/Reference/Stream) utility class.

Syntax:

Serial.peek()
Arduino Mega only: Serial1.peek() Serial2.peek() Serial3.peek()

Parameters:

None

Returns:

the first byte of incoming serial data available (or -1 if no data is available) - int

Serial.print() Description

Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. For example:

- Serial.print(78) gives "78"
- Serial.print(1.23456) gives "1.23"

- Serial.print('N') gives "N"
- Serial.print("Hello world.") gives "Hello world." An optional second parameter specifies the base (format) to use; permitted values are BIN (binary, or base 2), OCT (octal, or base 8), DEC (decimal, or base 10), HEX (hexadecimal, or base 16). For floating point numbers, this parameter specifies the number of decimal places to use. For example:
- Serial.print(78, BIN) gives "1001110"
- Serial.print(78, OCT) gives "116"
- Serial.print(78, DEC) gives "78"
- Serial.print(78, HEX) gives "4E"
- Serial.println(1.23456, 0) gives "1"
- Serial.println(1.23456, 2) gives "1.23"
- Serial.println(1.23456, 4) gives "1.2346"

  You can pass flash-memory based strings to Serial.print() by wrapping them with F(). For example :

- Serial.print(F("Hello World"))

To send a single byte, use Serial.write (http://arduino.cc/en/Serial/Write)().

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

Parameters
val: the value to print - any data type
format: specifies the number base (for integral data types) or number of decimal places (for floating point types)

Returns
size_t (long): print() returns the number of bytes written, though reading that number is optional

Example:

```
/*
Uses a FOR loop for data and prints a number in various formats.
*/
int x = 0; // variable
void setup()
        {
                Serial.begin(9600);
        }
void loop()
        {
                // print labels Serial.print("NO FORMAT"); Serial.print("\t");
                Serial.print("DEC"); Serial.print("\t");
                Serial.print("HEX"); Serial.print("\t");
                Serial.print("OCT"); Serial.print("\t");
                Serial.print("BIN"); Serial.print("\t");
                for(x=0; x< 64; x++)
                        {
                        }
        }
```

Serial.println() Description

Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). This command takes the same forms as Serial.print (http://arduino.cc/en/Serial/Print)().

Syntax:

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

Parameters:

val: the value to print - any data type
format: specifies the number base (for integral data types) or number of decimal places (for floating point types)

Returns:

size_t (long): println() returns the number of bytes written, though reading that number is optional

Example:

```
/*
  Analog input
 reads an analog input on analog in 0, prints the value out.
 created 24 March 2006
 by Tom Igoe
 */
int analogValue = 0; // variable to hold the analog value
void setup()
	{
		// open the serial port at 9600 bps:
		Serial.begin(9600);
	}
void loop()
	{
		// read the analog input on pin 0:
		analogValue = analogRead(0);
		// print it out in many formats:
		Serial.println(analogValue);
		Serial.println(analogValue, DEC); // print as an ASCII-encoded decimal
		Serial.println(analogValue, HEX); // print as an ASCII-encoded hexadecimal
		Serial.println(analogValue, OCT); // print as an ASCII-encoded octal
		Serial.println(analogValue, BIN); // print as an ASCII-encoded binary
		// delay 10 milliseconds before the next reading:
		delay(10);
	}
```

Serial.read() Description

Reads incoming serial data. read() inherits from the Stream (http://arduino.cc/en/Reference/Stream) utility class.

Syntax:

Serial.read()

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

Parameters:
None

Returns:
the first byte of incoming serial data available (or -1 if no data is available) - int

Example:
```
int S
incomingByte = 0;  // for incoming serial data
void setup()
        {
                Serial.begin(9600);  // opens serial port, sets data rate to 9600 bps
        }
void loop()
        {
         // send data only when you receive data:
        if (Serial.available() > 0)
                {
                        // read the incoming byte:
                        incomingByte = Serial.read();
                        // say what you got:
                        Serial.print("I received: ");
                        Serial.println(incomingByte, DEC);


                }
        }
```

Serial.readBytes() Description
Serial.readBytes() reads characters from the serial port into a buffer. The function terminates if the determined length has been read, or it times out (see Serial.setTimeout (http://arduino.cc/en/Serial/SetTimeout)()).
Serial.readBytes() returns the number of characters placed in the buffer. A 0 means no valid data was found.
Serial.readBytes() inherits from the Stream (http://arduino.cc/en/Reference/Stream) utility class.

Syntax:
Serial.readBytes(buffer, length)

Parameters:
buffer: the buffer to store the bytes in (char[] or byte[])
length : the number of bytes to read (int)

Returns:
byte

Serial.readBytesUntil() Description

Serial.readBytesUntil() reads characters from the serial buffer into an array. The function terminates if the terminator character is detected, the determined length has been read, or it times out (see Serial.setTimeout (http://arduino.cc/en/Serial/SetTimeout)()).
Serial.readBytesUntil() returns the number of characters read into the buffer. A 0 means no valid data was found.
Serial.readBytesUntil() inherits from the Stream (http://arduino.cc/en/Reference/Stream) utility class.

Syntax:
Serial.readBytesUntil(character, buffer, length)

Parameters:
character : the character to search for (char)
buffer: the buffer to store the bytes in (char[] or byte[])
length : the number of bytes to read (int)

Returns:
byte

Serial.readString() Description

Serial.readString() reads characters from the serial buffer into a string. The function terminates if it times out (see setTimeout (http://arduino.cc/en/Serial/SetTimeout)()).
This function is part of the Stream class, and is called by any class that inherits from it (Wire, Serial, etc). See the Stream class (http://arduino.cc/en/Reference/Stream) main page for more information.

Syntax:
Serial.readString()

Parameters:
none

Returns:
A string read from the serial buffer

Serial.readStringUntil() Description

readStringUntil() reads characters from the serial buffer into a string. The function terminates if the terminator character is detected or it times out (see setTimeout (http://arduino.cc/en/Serial/SetTimeout)()).
This function is part of the Stream class, and is called by any class that inherits from it (Wire, Serial, etc). See the Stream class (http://arduino.cc/en/Reference/Stream) main page for more information.

Syntax:
Serial.readString(terminator)

Parameters:
terminator : the character to search for (char)

Returns:
The entire string read from the serial buffer, until the terminator character is detected

serialEvent() Description
Called when data is available. Use Serial.read() to capture this data.
NB : Currently, serialEvent() is not compatible with the Esplora, Leonardo, or Micro Syntax

```
void serialEvent()
        {
                //statements
        }
```

Arduino Mega only:

```
void serialEvent1()
        {
                //statements
        }
void serialEvent2()
        {
                //statements
        }
void serialEvent3()
        {
                //statements
        }
```

Parameters:
statements: any valid statements

Serial.setTimeout() Description
Serial.setTimeout() sets the maximum milliseconds to wait for serial data when using Serial.readBytesUntil (http://arduino.cc/en/Serial/ReadBytesUntil)() or Serial.readBytes (http://arduino.cc/en/Serial/ReadBytes)(). It defaults to 1000 milliseconds.
Serial.setTimeout() inherits from the Stream (http://arduino.cc/en/Reference/Stream) utility class.

Syntax:
Serial.setTimeout(time)

Parameters:
time : timeout duration in milliseconds (long).

Parameters:
None

Serial.write() Description

Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print (http://arduino.cc/en/Serial/Print)() function instead.

Syntax:

Serial.write(val) Serial.write(str) Serial.write(buf, len)
Arduino Mega also supports: Serial1, Serial2, Serial3 (in place of Serial)

Parameters:

val: a value to send as a single byte
str: a string to send as a series of bytes
buf: an array to send as a series of bytes
len: the length of the buffer

Returns:

byte

write() will return the number of bytes written, though reading that number is optional

Example:

```
void setup()
        {
                Serial.begin(9600);
        }
void loop()
        {
                Serial.write(45); // send a byte with the value 45
                int bytesSent = Serial.write("hello"); //send the string "hello" and return the length of the string.
        }
```