

**Serial Library**  
**for the**  
**Atmel AVR Microcontroller**

Software Library Version  
June 13, 2005

Andrew J. Blauch  
School of Engineering



**DISCLAIMER:**

This software is provided “as is” and without any express or implied warranties, including, without limitation, the implied warranties of merchantability and fitness for a particular purpose.

**COMPATABILITY:**

This software has been created and tested using the following development systems:

Compiler(s):

GCC AVR compiler version 3.4.1

Processor(s):

Atmel ATMega32

Evaluation Board(s):

In-house evaluation board, School of Engineering, GVSU

Copyright © 2005  
School of Engineering  
Grand Valley State University

**Table of Contents**

1	Overview .....	1
2	Specifications and I/O Usage .....	2
3	Function Summary.....	3
3.1	Configuration Functions .....	3
3.2	Input Functions .....	3
3.3	Output Functions.....	3
3.4	String Functions .....	3
4	Function Documentation.....	4
4.1	Configuration Functions .....	5
4.2	Input Functions .....	6
4.3	Output Functions.....	9
4.4	String Functions .....	12

## 1 Overview

The SERIAL library allows a programmer to quickly and easily transfer and receive data across the serial port. To use the library functions, include the `serial.h` header file in the source code and link with the `libserial.a` library file. If using the **gccavr** batch file, simply use the **-l serial** option on the command line to link your source code with the library.

The library contains of a basic set of serial I/O functions. The functionality and form of the library functions follow that of the standard I/O C library functions. The library also contains a set of string functions along with some advanced serial I/O functions. The string functions provide basic conversion capabilities between upper and lower case and between ASCII and signed/unsigned representation of numeric values. The advanced serial I/O functions combine the basic serial functions and string functions to provide the capability to transfer and receive numeric values across the serial port.

The serial functions require that the `Startup()` function be called to initialize the serial I/O peripheral/parameters. This must occur prior to using any of the serial functions. The `Cleanup()` function should be called to disable/un-initialize the serial I/O peripheral/parameters. Failure to call these two functions will result in sporadic and incorrect operation. This library can be used for developing boot code.

## 2 Specifications and I/O Usage

The Serial library functions use the SCI peripheral on the AVR. The SCI peripheral is initialized by calling the startup function. The following is a list of the pin usage and fixed serial communication settings.

Transmit signal: PD1 (digital output)

Receive signal: PD0 (digital input)

38400 BAUD, 8 data bits, no parity, 1 stop bit, no flow control

## 3 Function Summary

### 3.1 Configuration Functions

```
void SerialSetup(void)
void SerialCleanup(void)
```

### 3.2 Input Functions

```
char input(void)
char getch(void)
char getche(void)
char * gets(char *ptr, unsigned int len)
int getint(void)
unsigned int getuint(void)
unsigned char gethex8(void)
unsigned int gethex16(void)
```

### 3.3 Output Functions

```
void output(char data)
void putch(char data)
void puts(char *ptr)
void putint(int data)
void putuint(unsigned int data)
void puthex8(unsigned char data)
void puthex16(unsigned int data)
```

### 3.4 String Functions

```
char toupper(char data)
char tolower(char data)
char *strupr(char *ptr)
char *strlwr(char *ptr)
int ctoh(char data)
unsigned atou(char *ptr)
int atoi(char *ptr)
char htoc(int data)
char * utoa(unsigned value, char *ptr)
char * itoa(int value, char *ptr)
```

## **4 Function Documentation**

## 4.1 Configuration Functions

---

**void SerialSetup(void)**

**Description:**

Performs all necessary initialization for serial communication library functions. This function must be called prior to calling any other serial library functions. This function need only be called once. Refer to I/O usage for communication parameter settings.

**Parameters:**

None

**Return Value:**

None

**Related Functions:**

SerialCleanup

---

**void SerialCleanup(void)**

**Description:**

Performs all necessary un-initialization for serial communication library functions. This function need only be called once at the very end of the program.

**Parameters:**

None

**Return Value:**

None

**Related Functions:**

SerialSetup

## 4.2 Input Functions

---

`char input(void)`

**Description:**

Reads one byte from serial port (no echo). This function returns immediately, regardless of whether or not anything has been received.

**Parameters:**

None

**Return Value:**

Received value or 0 if nothing has been received.

**Related Functions:**

output  
getch

---

`char getch(void)`

**Description:**

Reads one byte from serial port (no echo). This function does not return until data has been received. A received CR () character is automatically translated into a LF () character.

**Parameters:**

None

**Return Value:**

Received value.

**Related Functions:**

input  
getche  
putch

---

`char getche(void)`

**Description:**

Reads one byte from serial port (with echo). This function does not return until data has been received.

**Parameters:**

None

**Return Value:**

Received value.

**Related Functions:**

input  
getch

---

**char \* gets(char \*ptr, unsigned int len)****Description:**

Reads string from serial port (with echo). This function does not return until a CRLF is received or the specified length has been reached.

**Parameters:**

ptr                    pointer to buffer in which to place received string  
len                    maximum length of buffer

**Return Value:**

Pointer to received string (same as ptr parameter).

**Related Functions:**

puts

---

**int getint(void)****Description:**

Gets a 16-bit signed decimal formatted number from the serial port (with echo). Valid range is -32768...32768. This function does not return until a CRLF is received. Conversion of the received string is terminated by the first invalid numeric character.

**Parameters:**

None

**Return Value:**

16-bit signed value of received number

**Related Functions:**

putint  
gethex8  
gethex16  
getuint

---

**unsigned int getuint(void)****Description:**

Gets a 16-bit unsigned decimal formatted number from the serial port (with echo). Valid range is 0...65535. This function does not return until a CRLF is received. Conversion of the received string is terminated by the first invalid numeric character.

**Parameters:**

None

**Return Value:**

16-bit unsigned value of received number

**Related Functions:**

putuint  
gethex8  
gethex16  
getint

---

**unsigned char gethex8(void)****Description:**

Gets an 8-bit hexadecimal formatted number from the serial port (with echo). Do not enter hexadecimal notation specifier (i.e. enter DE not 0xDE or \$DE). Valid range is 00...FF. This function does not return until a CRLF is received. Conversion of the received string is terminated by the first invalid numeric character.

**Parameters:**

None

**Return Value:**

8-bit value of received number

**Related Functions:**

puthex8  
gethex16  
getint  
getuint

---

**unsigned int gethex16(void)****Description:**

Gets a 16-bit hexadecimal formatted number from the serial port (with echo). Do not enter hexadecimal notation specifier (i.e. enter DE not 0xDE or \$DE). Valid range is 0000...FFFF. This function does not return until a CRLF is received. Conversion of the received string is terminated by the first invalid numeric character.

**Parameters:**

None

**Return Value:**

16-bit value of received number

**Related Functions:**

puthex16  
gethex8  
getint  
getuint

## 4.3 Output Functions

---

`void output(char data)`

**Description:**

Outputs one byte to serial port.

**Parameters:**

data                    one byte value to transmit

**Return Value:**

None

**Related Functions:**

input  
putch

---

`void putch(char data)`

**Description:**

Outputs one byte to serial port. The LF () character is automatically translated into a CRLF () combination for transmission.

**Parameters:**

data                    one byte value to transmit

**Return Value:**

None

**Related Functions:**

getch  
puts

---

`void puts(char *ptr)`

**Description:**

Outputs string to serial port.

**Parameters:**

ptr                    pointer to string to be transmitted

**Return Value:**

None

**Related Functions:**

gets  
putch

---

```
void putint(int data)
```

**Description:**

Outputs decimal ASCII representation of signed number to serial port. Number is not padded with zeros (variable 1 to 5 digit display).

**Parameters:**

data                    signed number to be transmitted

**Return Value:**

None

**Related Functions:**

getint  
puthex8  
puthex16  
putuint

---

```
void putuint(unsigned int data)
```

**Description:**

Outputs decimal ASCII representation of unsigned number to serial port. Number is not padded with zeros (variable 1 to 5 digit display).

**Parameters:**

data                    unsigned number to be transmitted

**Return Value:**

None

**Related Functions:**

getuint  
puthex8  
puthex16  
putint

---

```
void puthex8(unsigned char data)
```

**Description:**

Outputs hexadecimal ASCII representation of 8 bit number to serial port. Number is left-padded with zeros (fixed 2 digit display).

**Parameters:**

data                    8 bit number to be transmitted

**Return Value:**

None

**Related Functions:**

gethex8  
puthex16  
putint  
putuint

---

`void puthex16(unsigned int data)`

**Description:**

Outputs hexadecimal ASCII representation of 16 bit number to serial port. Number is left-padded with zeros (fixed 4 digit display).

**Parameters:**

data                    16 bit number to be transmitted

**Return Value:**

None

**Related Functions:**

gethex16  
puthex8  
putint  
putuint

## 4.4 String Functions

---

**char toupper(char data)**

**Description:**

Converts character from lowercase to uppercase.

**Parameters:**

data                      character to convert

**Return Value:**

Converted character

**Related Functions:**

tolower  
strupr  
strlwr

---

**char tolower(char data)**

**Description:**

Converts character from uppercase to lowercase.

**Parameters:**

data                      character to convert

**Return Value:**

Converted character

**Related Functions:**

toupper  
strupr  
strlwr

---

**char \* strupr(char \*ptr)**

**Description:**

Converts string from lowercase to uppercase.

**Parameters:**

ptr                      pointer to string

**Return Value:**

Pointer to string (same as ptr parameter).

**Related Functions:**

tolower  
toupper  
strlwr

---

**char \* strtolwr(char \*ptr)**

**Description:**

Converts string from uppercase to lowercase.

**Parameters:**

ptr                    pointer to string

**Return Value:**

Pointer to string (same as ptr parameter).

**Related Functions:**

tolower  
toupper  
strupr

---

**int ctoh(char data)**

**Description:**

Converts hex character ['0'...'F'] to integer [0...15]. Conversion is case-insensitive.

**Parameters:**

data                    character to convert

**Return Value:**

Converted value

**Related Functions:**

htoc  
atou  
atoi  
utoa  
itoa

---

**unsigned atoi(char \*ptr)**

**Description:**

Converts ASCII string to unsigned integer value.

**Parameters:**

ptr                    pointer to string to convert

**Return Value:**

Converted value

**Related Functions:**

ctoh  
htoc  
atoi  
utoa  
itoa

---

`int atoi(char *ptr)`

**Description:**

Converts ASCII string to integer value.

**Parameters:**

ptr                    pointer to string to convert

**Return Value:**

Converted value

**Related Functions:**

ctoh  
htoc  
atou  
utoa  
itoa

---

`char htoc(int data)`

**Description:**

Converts integer [0...15] to hex character ['0'...'F']. Conversion is in uppercase.

**Parameters:**

data                    integer to convert

**Return Value:**

Converted character

**Related Functions:**

ctoh  
atou  
atoi  
utoa  
itoa

---

```
char * utoa(unsigned value, char *ptr)
```

**Description:**

Converts unsigned integer value to ASCII string format.

**Parameters:**

value	value to convert
ptr	pointer to string in which to place result

**Return Value:**

Pointer to converted string (same as ptr parameter).

**Related Functions:**

ctoh  
htoc  
atou  
atoi  
itoa

---

```
char * itoa(int value, char *ptr)
```

**Description:**

Converts integer value to ASCII string format.

**Parameters:**

value	value to convert
ptr	pointer to string in which to place result

**Return Value:**

Pointer to converted string (same as ptr parameter).

**Related Functions:**

ctoh  
htoc  
atou  
atoi  
utoa