

**University of Florida
Department of Electrical Engineering
EEL 5934 Intelligent Machines Design Lab**

Useful Functions in IC Libraries

serial.c

The functions in this library can be used to transfer data to/from your board using Kermit. After you load your program in IC, start your program, start Kermit, and hit "c" to enter the terminal. You can then interact with your robot **while it is running!**

One note: Although these routines have been used, there have been some problems with the newline feature. `write_int()` starts a new line, but `print()` and `write()` don't do it correctly. Brownie points to the person who fixes the problem. 😊

void init_serial()

This function initializes the SCI port to run at 9600 baud. If you want to use any of the serial port functions described here, this function must be at the beginning of your program.

Example:

```
init_serial();
```

int get_char()

Wait for a character from the serial port, and return its ASCII value.

Examples:

```
x = get_char();  
if (get_char() == 'F') fd(1);
```

void put_char(int outchar)

Write the character *outchar* to the serial port.

Examples:

```
put_char(65);  
put_char('A');
```

void write(char string[])

Write `string[]` to the serial port. The only control character supported is the newline character (`\n`) used in C.

Examples:

```
write("Hello, world.\n");  
write("one\ntwo\nthree\n");  
write("My robot is cool!");
```

void write_int(int number)

Write the integer *number* to the serial port. The number is preceded by one space and is followed by a newline.

Examples:

```
{
  int j;
  write("The value of j is");
  write_int(j);
}
```

void print(char string[], int number)

A limited version of `printf()`. Two control characters are allowed in `string[]`: newline (`\n`) and integer (`%d`). Note that one (no more, no less) integer is allowed per print command.

Examples:

```
{
  int i, j;
  print("The value of i is %d.\n", I);

  print("The value of i is %d and the value ", i);
  print("of j is %d.\n", j);
}
```

lib_rw10.c

void motor(int n, int speed)

Turns on motor *n* (where *n* is 0 or 1) at a speed which can vary from -100 to 100 where 100 is full speed forward, -100 is full speed backward, and 0 is stop. For any number in between, pulse-width modulation is used.

void fd(int n)

Turns on motor *n* to full forward. Shorthand for "motor(*n*,100)".

void bk(int n)

Turns on motor *n* to full backward. Shorthand for "motor(*n*, -100)".

void off(int n)

Turns off motor *n*. Shorthand for "motor(*n*, 0)".

void alloff() or
void ao()

Turns off all motors. Shorthand for "{off(0); off(1);}"

int analog(int port)

This function performs an analog-to-digital conversion and returns a number from 0 to 255 corresponding to the voltage on port E, pin *port*.

void beep()

If you have a piezo speaker connected, this causes a short beep.

void tone(float frequency, float length)

If you have a piezo speaker connected, this generates a tone at *frequency* Hertz for *length* seconds.

void sleep(float seconds)

Wait for *seconds* seconds.