Console Logging

afLib3 provides a set of functions to support logging to the serial console. For Arduino, use of `af_logger` is optional; existing Arduino Serial debugging methods will still work as they always have. When porting afLib3 to other MCU platforms, implementing `af_logger` will allow the library to use your MCU's logging platform to provide debugging and error output.

af_logger()

Description

A generic implementation to print debugging output to the serial console.

Syntax

There are several forms of `af_logger()`; the call you choose depends upon what type of data you intend to print. Here are the currently--available calls:

```
af_logger_format_t:
    AF_LOGGER_BIN = 2,
    AF_LOGGER_OCT = 8,
    AF_LOGGER_DEC = 10,
    AF_LOGGER_HEX = 16

void af_logger_print_value(int32_t val);
```

```
void af_logger_print_buffer(const char* val);
void af_logger_print_formatted_value(int32_t val, af_logger_format_t format);
void af_logger_println_value(int32_t val);
void af_logger_println_buffer(const char* val);
void af_logger_println_formatted_value(int32_t val, af_logger_format_t format);
```

Parameters

`val`	Value to be printed.
`format`	Optional `af_logger_format_t` to determine output format.

Note that, because `val` is declared as a signed 32-bit type, `af_logger()` will not properly print values of unsigned 32-bit numbers larger than 2147483647. If you need support for such values, you should use the standard Arduino `Serial.print()`, or implement your own logger.

Returns

None.

Examples

1 This line:

```
af_logger_println_buffer("Rebooted - Always set MCU attributes after reboot");
```

will print "**Rebooted - Always set MCU attributes after reboot**", followed by a newline.

2 On Arduino, the following two lines of code are functionally identical:

```
Serial.println( int8variable, HEX );
```

```
af_logger_println_formatted_value( int8variable, AF_LOGGER_HEX );
```

Next: *afLib Error Codes*

Updated September 25, 2019