Table of Contents

Overview

API

JSLK HAL Kernel API

The following functions are provided by the kernel's Hardware Abstraction Layer (HAL). This functions should only be used in kernel mode by including header `<hal.h>` . This article is being written with focus on x86 with JSLK kernel version 0.0.6.1 pre-alpha/debug. You can also view this document online.

Screen I/O

The following functions and types are used to interact with the kernel console. To work with this functions the HAL also provides the `vga_color` type which can be set to any of the following VGA colors:

- **vga_black**
- **vga_blue**
- **vga_green**
- **vga_cyan**
- **vga_red**
- **vga_magenta**
- **vga_brown**
- **vga_light_grey**
- **vga_grey**
- **vga_light_blue**
- **vga_light_green**
- **vga_light_cyan**
- **vga_light_red**
- **vga_light_magenta**
- **vga_light_brown**
- **vga_white**

`getColor(vga_color text, vga_color background)`

`getColor()` takes 2 VGA colors as its arguments and returns a `uint8_t` which can then be passed to other functions as the color set to use.

Example code:

```
#include <hal.h>
#include <stdint.h>

int main() {
    uint8_t colorSet = getColor(vga_white, vga_light_grey);
    setConsoleColor(colorSet);
    return 0;
}
```

Result:

```
The console colors change.
```

`setConsoleColor(uint8_t color)`

`setConsoleColor()` can be used to change the color set used by the console. Unless you clear the console, the changes can only be seen when new text is printed to the screen.

Example code:

```
#include <hal.h>
#include <stdint.h>

int main() {
    uint8_t colorSet = getColor(vga_white, vga_blue);
    setConsoleColor(colorSet);
    clear_console();
    return 0;
}
```

Result:

```
The console colors change.
```

This function takes a `uint8_t` with the color set to use as its argument and has no return value.

`clear_console()`

`clear_console()` erases all the contents of the console and updates its color attributes if necessary.

Example code:

```
#include <hal.h>

int main() {
    clear_console();
    return 0;
}
```

Result:

```
The console is cleared.
```

This function takes no arguments and doesn't have a return value.

`kprint(string c)`

`kprint()` allows you to print strings to the kernel console. A valid console must be active to use this function.

Example code:

```
#include <hal.h>

int main() {
    kprint("Hello World \n");
    return 0;
}
```

Output:

```
Hello World
```

The function takes the text to print as a `string` as its first argument and has no return value.

`kputc(char c)`

`kputc()` allows you to print a character to the kernel console. A valid console must be active to use this function.

Example code:

```
#include <hal.h>
#include <stdint.h>

int main() {
    char buffer[] = {'H', 'e', 'l', 'l', 'o'};
    for (size_t i = 0; i < 5; i++)
        kputc(buffer[i]);
    return 0;
}
```

Output:

```
Hello
```

The function takes the character to print as a `char` as its first argument and has no return value.

`writeStyledString(string c, uint8_t color)`

`writeStyledString()` allows you to print a string with different color attributes to the kernel console. A valid console must be active to use this function.

Example code:

```
#include <hal.h>
#include <stdint.h>

int main() {
    uint8_t stringColor = getColor(vga_red, vga_green);
    writeStyledString("This is a string with style \n", stringColor);
    return 0;
}
```

Output:

```
This is a string with style.
```

The function takes the text to print as a `string` as its first argument, a `uint8_t` with a color set and has no return value.

`consolePutChar(char c, uint8_t color, size_t x, size_t y)`

`consolePutChar()` allows you to print a character to a specified index in the console.

> Note: y can't be smaller than one or greater than 25, and x can't be smaller than 0 or greater than 80.

Example code:

```
#include <hal.h>
#include <stdint.h>

int main() {
    uint8_t charColor = getColor(vga_red, vga_green);
    consolePutChar('a', charColor, 3, 2);
    return 0;
}
```

Output:

```
// blank line
  a
```

The function takes the character to print as a `char` as its first argument, a `uint8_t` with a color set as its second argument, index x as its third and index y as its fourth. It has no return value.