libConsole Documentation

Release 0.9.0

ANSSI

Oct 10, 2019

CONTENTS

Contents

CHAPTER

ONE

OVERVIEW

The libconsole provides a simple interface between serial devices, such as UART or USART, and the tasks.

1.1 Limitations

This library is not designed for holding multiple serial lines concurrently.

Warning: The libconsole does not configure any hardware control flow. Deactivate this feature from your serial console client

CHAPTER

TWO

API

2.1 Initializing the console

Initialize the console is made by two functions:

```
#include "libconsole.h"

mbed_error_t console_early_init(uint8_t usart_id, uint32_t speed);
mbed_error_t console_init(void);
```

`console_early_init()` function must be called during the task *initialization phase* while the former `console_init()` must be called after, during the *nominal phase*.

Note: The `console_early_init()` can not make the whole initialization because the device is still not mapped in task's memory space.

The *usart_id* argument is the identifier of the USART device ,typically 1 to 6 on STM32F4xx SoCs.

The *speed* argument is the serial console speed in bauds (typical values are 115200, 57600, 38400, 14400 or 9200). It is possible to set any speed (such as 1Mb/s or older 1200 bps) if it's supported by the peer.

2.2 Printing data on the console

The console library provides a high level, easy to use logging function to print-out formated strings on the serial console:

```
#include "libconsole.h"

void console_log(const char *fmt, ...);
```

This function supported formated output strings the way the printf() familly functions of the EwoK libstd API handle them.

Warning: This function handle up to 128 bytes length formated output string. Longer strings are truncated

2.3 Reading a line from the console

When interacting with a peer through a serial console, a readline() function is requested. This function is a blocking function returning the command sent by the peer when a carriage return is sent.

Reading lines on the console interface is done with the following API:

```
#include "libconsole.h"

mbed_error_t console_readline(char *str, uint32_t *len, uint32_t maxlen);
```

This function is a blocking function, waiting for a command to be sent on the serial line. The command is consider as sent (and the function is unlocking its execution) when a *carriage return* is received. When this event happen, the following is done:

- The command sent is copied in the str argument, and may be truncated to maxlen if the command length is bigger than the current command length
- The current command buffer is reinitialized
- If the command copy truncate the received command, the function return MBED_ERROR_NOSTORAGE. Otherwise, the function return MBED_ERROR_NONE

2.4 Prompting

The libconsole support prompt printing on demand. The goal is to show a prompt character, followed by a space, on the console:

```
#include "libconsole.h"

void console_show_prompt(void);
```

CHAPTER

THREE

FAQ

3.1 Is it possible to declare multiple serial console in a same task ?

Not yet. The libconsole is using a single global context which does not permit to handle multiple serial lines in the same time.

3.2 Is it possible to read multiple lines at a time ?

Not at a time, but it is possible to read one line, parse it, and decide to read another one if needed (for e.g. if the last non-space character is a backslash)

3.3 Is the console USART device unmappable ?

Not by now. the serial device is mapped automatically by the kernel.