mcu-uuid-common

Jun 02, 2021

Contents

CHAPTER 1

Description

Microcontroller common utilities library

CHAPTER 2

Purpose

The primary purpose of this library is to maintain a common 64-bit uptime in milliseconds with overflow handling, as long as the loop function is called regularly.

CHAPTER 3

Contents

3.1 Usage

```
#include <uuid/common.h>
```

Call `uuid::loop()` regularly and then call `uuid::get_uptime_ms()` when the uptime is required.

3.1.1 Example

```
#include <Arduino.h>
#include <uuid/common.h>

class Uptime: public Printable {
public:
        Uptime(uint64_t timestamp_ms = uuid::get_uptime_ms())
                        : timestamp_ms_(timestamp_ms) {

        }

        size_t printTo(Print &print) const override {
                unsigned long days;
                unsigned int hours, minutes, seconds, milliseconds;
                uint64_t timestamp_ms = timestamp_ms_;

                days = timestamp_ms / 86400000UL;
                timestamp_ms %= 86400000UL;

                hours = timestamp_ms / 3600000UL;
                timestamp_ms %= 3600000UL;

                minutes = timestamp_ms / 60000UL;
                timestamp_ms %= 60000UL;
```

(continues on next page)

(continued from previous page)

```
                seconds = timestamp_ms / 1000UL;
                timestamp_ms %= 1000UL;

                milliseconds = timestamp_ms;

                size_t len = print.print(F("Uptime: "));
                len += print.print(days);
                len += print.print('+');

                char time[2 + 1 /* hours */
                                + 2 + 1 /* minutes */
                                + 2 + 1 /* seconds */
                                + 3 /* milliseconds */
                                + 1] = { 0 };
                int ret = snprintf_P(time, sizeof(time), PSTR("%02u:%02u:%02u.%03u"),
                                hours, minutes, seconds, milliseconds);
                if (ret > 0) {
                        len += print.print(time);
                }
                return len;
        }

private:
        uint64_t timestamp_ms_;
};

void setup() {
        Serial.begin(115200);
}

void loop() {
        uuid::loop();

        Serial.println(Uptime());

        delay(1000);
}
```

Output

```
Uptime: 0+00:00:00.000
Uptime: 0+00:00:01.000
Uptime: 0+00:00:02.000
Uptime: 0+00:00:03.000
Uptime: 0+00:00:04.000
Uptime: 0+00:00:05.000
Uptime: 0+00:00:06.000
Uptime: 0+00:00:07.000
Uptime: 0+00:00:08.000
Uptime: 0+00:00:09.000
Uptime: 0+00:00:10.000
```

CHAPTER 4

Resources

4.1 Change log

4.1.1 Unreleased

4.1.2 1.1.0 – 2019-09-15

Printable to string functions.

Addded

- Functions to output a `Printable` to a `std::string`.

Changed

- Put each function in a separate file to improve linker behaviour.

4.1.3 1.0.2 – 2019-08-16

Fix example.

Fixed

- Example now prints the time part of the uptime instead of ignoring it.

4.1.4 1.0.1 – 2019-08-15

Update example and add tests.

Added

- Test builds of the example.
- Unit tests of the uptime overflow handling.

Changed

- Exclude the test directory from exports of the library source code.

Fixed

- Remove use of `Serial.printf()` from the example (which does not exist in the standard Arduino library).

4.1.5 1.0.0 – 2019-08-11

First stable release.

Added

- Function `void uuid::loop()` that should be called regularly.
- Function `uint64_t uuid::get_uptime_ms()` to get the current uptime as a 64-bit milliseconds value.
- Function `std::string uuid::read_flash_string(const __FlashStringHelper *flash_str)` to convert flash strings to `std::string`.
- Type `uuid::flash_string_vector` for a `std::vector` of flash strings.