mcu-uuid-log

Simon Arlott

Apr 09, 2024

CONTENTS

CHAPTER

ONE

DESCRIPTION

Microcontroller logging framework library

CHAPTER

TWO

PURPOSE

Provides a framework for handling log messages. Thread-safe on the ESP32 but cannot be used from an interrupt context.

CHAPTER

THREE

DEPENDENCIES

- mcu-uuid-common

Refer to the `library.json` file for more details.

CHAPTER

FOUR

CONTENTS

4.1 Usage

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

Create a `uuid::log::Logger` and call its functions for logging messages.

Create a class that implements `uuid::log::Handler` and registers to handle log messages. Output those messages by whatever means is appropriate for your application (e.g. serial console, over the WiFi network, by email).

There is no option to exclude compilation of debug-level messages, the expectation is that these will be disabled at runtime. If your program cannot fit in flash with the debug messages enabled or run with sufficient performance then there's no benefit in having them.

The performance impact of disabled debug logging can be limited by using the `Logger::enabled(LogLevel)` function (but messages will not be formatted if the level isn't enabled).

4.1.1 Example

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

class SerialLogHandler: public uuid::log::Handler {
public:
        SerialLogHandler() = default;

        void start() {
                uuid::log::Logger::register_handler(this, uuid::log::Level::ALL);
        }

        /*
         * It is not recommended to directly output from this function,
         * this is only a simple example. Messages should normally be
         * queued for later output when the application is less busy.
         */
        void operator<<(std::shared_ptr<uuid::log::Message> message) {
                char temp[100] = { 0 };

                int ret = snprintf_P(temp, sizeof(temp), PSTR("%s %c [%S] %s\r\n"),
```

(continues on next page)

(continued from previous page)

```
                        uuid::log::format_timestamp_ms(message->uptime_ms).c_str(),
                        uuid::log::format_level_char(message->level),
                        message->name, message->text.c_str());

                if (ret > 0) {
                        Serial.print(temp);
                }
        }
};

static SerialLogHandler log_handler;

void setup() {
        static uuid::log::Logger logger{F("setup")};

        Serial.begin(115200);

        log_handler.start();

        logger.info(F("Application started"));
}

void loop() {
        static uuid::log::Logger logger{F("loop")};
        static unsigned int i = 0;

        uuid::loop();

        logger.debug(F("Hello %u World!"), i++);

        delay(1000);
}
```

Output

```
0+00:00:00.000 I [setup] Application started
0+00:00:00.000 D [loop] Hello 0 World!
0+00:00:01.000 D [loop] Hello 1 World!
0+00:00:02.000 D [loop] Hello 2 World!
0+00:00:03.000 D [loop] Hello 3 World!
0+00:00:04.000 D [loop] Hello 4 World!
0+00:00:05.000 D [loop] Hello 5 World!
0+00:00:06.000 D [loop] Hello 6 World!
0+00:00:07.000 D [loop] Hello 7 World!
0+00:00:08.000 D [loop] Hello 8 World!
0+00:00:09.000 D [loop] Hello 9 World!
0+00:00:10.000 D [loop] Hello 10 World!
```

4.2 Related Libraries

- mcu-uuid-syslog is a log handler that sends messages to a syslog server.

CHAPTER

FIVE

RESOURCES

5.1 Change log

5.1.1 Unreleased

5.1.2 3.1.0 – 2024-03-17

Support forwarding plain and formatted messages to the `Logger`.

Added

- Make the `vlog()` functions public so that a variable number of arguments can be passed directly to the logger.
- Add `logp()` functions that allow plain text to be passed directly to the logger without using formatting.

5.1.3 3.0.0 – 2022-11-03

Support setting a level on each `Logger`.

Added

- Support setting a level on each `Logger`.
- Function to get the global log level.
- Function to get the effective log level on a `Logger`.
- Support logging messages at a specific level without providing the facility.

Changed

- Make `PrintHandler` more efficient by removing the log message before printing it.
- Use `PSTR_ALIGN` for flash strings.

5.1.4 2.3.0 – 2022-10-25

Be thread-safe where possible.

Added

- Function to get a `Logger`'s default facility for new messages.
- Indicate whether this version of the library is thread-safe or not (`UUID_LOG_THREAD_SAFE` and `uuid::log::thread_safe`).

Changed

- Make the library thread-safe when supported by the platform.

5.1.5 2.2.0 – 2022-01-29

Add a basic log handler for the Print interface.

Added

- Basic log handler for writing messages to any object supporting the `Print` interface (`PrintHandler`).

5.1.6 2.1.4 – 2021-06-02

Fix for compile failure with newer GCC in espressif8266 3.0.0.

Fixed

- Compile failure with newer GCC in espressif8266 3.0.0 using `reinterpret_cast` for a `constexpr` value.

5.1.7 2.1.3 – 2021-04-18

Upgrade to PlatformIO 5.

Changed

- Use PlatformIO 5 dependency specification.

5.1.8 2.1.2 – 2020-01-17

Fixes for uncontrolled ordering of static object lifetimes.

Changed

- Automatically unregister handlers when they are destroyed.

Fixed

- Make registration of log handlers safe during static initialization and unregistration safe during static deinitialization.

5.1.9 2.1.1 – 2019-09-15

Fix to use less memory.

Fixed

- Use a flash string for the format string in `format_timestamp_ms()`.

5.1.10 2.1.0 – 2019-09-07

Add functions to get levels, format them as strings and parse them

Added

- Functions to get a list of all levels, format them as uppercase or lowercase strings and then parse them back again.

5.1.11 2.0.4 – 2019-08-26

Fix uptime format string.

Changed

- Put formatting functions in a separate file to improve linker behaviour.

Fixed

- Uptime format string should use the unsigned conversion specifier for all values.

5.1.12 2.0.3 – 2019-08-17

Make `get_log_level()` accept a `const Handler`.

Fixed

- Make `get_log_level()` accept a `const Handler`.

5.1.13 2.0.2 – 2019-08-17

Make logging functions `const` so that a `const Logger` can be used.

Changed

- Make all of the logging functions `const`.

5.1.14 2.0.1 – 2019-08-16

Fix ESP32 builds and the example.

Fixed

- Workaround incorrect definition of `FPSTR()` on ESP32 (#1371).
- Remove use of `Serial.printf_P()` from the example (which does not exist in the standard Arduino library).
- Add missing `Serial.begin()` to the example.

5.1.15 2.0.0 – 2019-08-12

Improve names of types, data members and interfaces.

Changed

- The names of `Message` data members no longer have trailing underscores.
- Rename log `Receiver` type to `Handler`.
- Use operator << instead of `add_log_message()` as the interface for log handlers.

5.1.16 1.0.1 – 2019-08-12

Fix uptime text formatting.

Changed

- The size of the `Level` and `Facility` enums is now 1 byte.

Fixed

- The width of the days part when formatting uptimes now has an upper bound of 10 instead of a lower bound of 10.

5.1.17 1.0.0 – 2019-08-11

First stable release.

Added

- Support for logging messages at all syslog levels and facilities.
- Support for registering receivers to handle log messages.