Erriez CRC32 library for Arduino

1.0.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

Optimized CRC32 library for Arduino

This is a target independent, flash and RAM size optimized CRC32 library for Arduino without CRC tables.

Library features

- Compatible with standard CRC32 algorithms
- Small flash footprint
- No RAM buffer allocations
- 32-bit table-less CRC32 calculation on:
  - Arduino String object
  - Single null-terminated character array
  - Single buffer
  - Multiple null-terminated character arrays
  - Multiple buffers
- No buffer alignment needed
- Buffer input length in Bytes
- Big and little endian input buffers
- Arduino C library, compatible with C and C++ applications
- CRC32 polynomial: 0xEDB88320

Hardware

This library can be used on any 8 or 32-bit target and is optimized for small flash/RAM footprints without CRC table allocations in RAM.

Examples

Arduino IDE | Examples | Erriez CRC32:

- CRC32
- tests | CRC32UnitTest

Benchmarks

Arduino IDE | Examples | Erriez CRC32:

- tests | CRC32Benchmark

Benchmark results on buffers

The benchmark results below may be different for each target and run. Lower duration means faster CRC calculation.

Board	F_CPU	4 Bytes	64 Bytes	512 Bytes	1024 Bytes
Arduino UNO (ATMega328)	16MHz	80us	1172us	9000us	17112us
ESP8266 (WeMOS D1 R2)	160MHz	3us	35us	279us	557us
ESP32 (WeMOS Lolin32)	80MHz	2us	15us	111us	223us
DUE (SAM3X8E ARM Cortex-M3)	84MHz	6us	70us	547us	1103us

Benchmarks performed with the following versions:

- Arduino IDE v1.8.5
- Arduino AVR Boards v1.6.21 (Arduino UNO)
- esp8266 v2.4.2
- Arduino SAM Boards (32-bits ARM Cortex-M3) v1.6.11

Unit tests

This library contains extensive unit tests on a various number of buffer types and lengths:

- tests | CRC32UnitTest

Library documentation

- Doxygen online HTML
- Doxygen PDF

Usage

Introduction

The library consists of four C-functions for CRC calculation:

```
1 {c++}
2 // Calculate CRC32 on null-terminated character array (string)
3 uint32_t crc32String(const char *buffer);
4 // Calculate CRC32 on one single buffer
5 uint32_t crc32Buffer(const void *buffer, size_t bufferLength);
6
7 // Calculate CRC32 on multiple buffers
8 // First crc32Update() call should start with argument crc = CRC32_INITIAL
9 // Next crc32Update() calls should set argument crc = previousCRC
10 uint32_t crc32Update(const void *buffer, size_t bufferLength, uint32_t crc);
11 // Call crc32Update() at the end after the last crc32Update()
12 uint32_t crc32Final(uint32_t crc);
```

Initialization

Only an include file is required. No object allocation or library initialization required.

```
1 {c++}
2 #include <ErriezCRC32.h>
```

Calculate CRC32 on Arduino String

```
1 {c++}
2 String msg = "Hello world String!";
3 uint32_t crc = crc32String(msg.c_str());
```

Calculate CRC32 on null-terminated character array

```
1 {c++}
2 char msg[] = "Hello world char array!";
3 uint32_t crc = crc32String(msg);
```

Calculate CRC32 on single character buffer

```
1 {c++}
2 char msg[] = "Hello world single buffer!";
3 uint32_t crc = crc32Buffer(msg, strlen(msg));
```

Calculate CRC32 on multiple characters

```
1 {c++}
2 uint32_t crc;
3
4 crc = crc32Update("Hello ", 6, CRC32_INITIAL);
5 crc = crc32Update("world ", 6, crc);
6 crc = crc32Update("multiple ", 9, crc);
7 crc = crc32Update("buffers!", 8, crc);
8 crc = crc32Final(crc);
```

Calculate CRC32 on multiple buffers

```
{c++}
const uint8_t testBuffer1[3] = { 0xEB, 0xE5, 0x51 };
const uint8_t testBuffer2[5] = { 0x87, 0x7F, 0xB8, 0x18, 0x4E };
uint32_t crc;

crc = crc32Update(testBuffer1, sizeof(testBuffer1), CRC32_INITIAL);
crc = crc32Update(testBuffer2, sizeof(testBuffer2), crc);
crc = crc32Final(crc);
```

Check CRC

```
{c++}
void checkCRC(uint32_t crcCalculated, uint32_t crcExpected)
{
    printCRC(crcCalculated);
    Serial.print(F("..."));

    if (crcCalculated == crcExpected) {
        Serial.println(F("OK"));
    } else {
        Serial.print(F("FAILED! Expected: "));
        printCRC(crcExpected);
        Serial.println(F(""));
    }
}
```

Print 32-bit CRC value

Printing a `uint32_t` value is not implemented on some targets which is a limitation of `Serial.print()` or `sprintf()`. This is a workaround to print a 32-bit hex value:

```
{c++}
void printCRC(uint32_t crc)
{
    Serial.print("0x");
    for (int8_t i = 3; i >= 0; i--) {
        uint8_t c = crc >> (i * 8);
        if (c < 0x10) {
            Serial.print("0");
        }
        Serial.print(c, HEX);
    }
}
```

CRC-calculation with Python

```
>>> import binascii
>>> hex(binascii.crc32(b'Hello world String!') & 0xffffffff)
'0x55df869b'
```

Library dependencies

- CRC32Benchmark.ino: `ErriezTimestamp.h`
- CRC32UnitTest.ino: `ArduinoUnit.h`

Library installation

Please refer to the `Wiki` page.

Other Arduino Libraries and Sketches from Erriez

- `Erriez Libraries and Sketches`

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

File Documentation

3.1 ErriezCRC32.c File Reference

Flash size optimized CRC32 library for Arduino without using CRC tables.

```
#include <string.h>
#include "ErriezCRC32.h"
```

Functions

- uint32_t crc32String (const char *buffer)

  Calculate CRC on a character string.
- uint32_t crc32Buffer (const void *buffer, size_t bufferLength)
- uint32_t crc32Update (const void *buffer, size_t bufferLength, uint32_t crc)

  Calculate CRC on multiple buffers.
- uint32_t crc32Final (uint32_t crc)

  Finalize CRC32 output.

3.1.1 Detailed Description

Flash size optimized CRC32 library for Arduino without using CRC tables.

Source: https://github.com/Erriez/ErriezCRC32 Documentation: https://erriez.github.io/ErriezCRC32

3.1.2 Function Documentation

3.1.2.1 uint32_t crc32Buffer (const void * *buffer*, size_t *bufferLength*)

This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

buffer	Input buffer.
bufferLength	Buffer length in Bytes.

Returns

32-bit CRC value.

Definition at line 66 of file ErriezCRC32.c.

3.1.2.2 uint32_t crc32Final (uint32_t *crc*)

Finalize CRC32 output.

This function should be called after crc32Update(). This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

crc	Previous CRC32 value.

Returns

32-bit CRC value.

Definition at line 113 of file ErriezCRC32.c.

3.1.2.3 uint32_t crc32String (const char * *buffer*)

Calculate CRC on a character string.

This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

buffer	Null terminated character input array.

Returns

32-bit CRC value.

Definition at line 49 of file ErriezCRC32.c.

3.1.2.4 uint32_t crc32Update (const void * *buffer*, size_t *bufferLength*, uint32_t *crc*)

Calculate CRC on multiple buffers.

This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

buffer	Input buffer. No alignment needed.
bufferLength	Input buffer length in Bytes. No alignment needed.
crc	Previous CRC value. Call with crc = CRC32_INITIAL on first calculation.

Returns

32-bit updated CRC value.

Definition at line 85 of file ErriezCRC32.c.

3.2 ErriezCRC32.h File Reference

CRC32 library for Arduino.

```
#include <inttypes.h>
#include <stddef.h>
```

Macros

- #define CRC32_INITIAL 0xFFFFFFFFUL
  CRC32 initial value.
- #define CRC32_POLYNOMIAL 0xEDB88320UL
  CRC32 polynomial.

Functions

- uint32_t crc32String (const char *buffer)
  Calculate CRC on a character string.
- uint32_t crc32Buffer (const void *buffer, size_t bufferLength)
- uint32_t crc32Update (const void *buffer, size_t bufferLength, uint32_t crc)
  Calculate CRC on multiple buffers.
- uint32_t crc32Final (uint32_t crc)
  Finalize CRC32 output.

3.2.1 Detailed Description

CRC32 library for Arduino.

Source: https://github.com/Erriez/ErriezCRC32 Documentation: https://erriez.github.io/ErriezCRC32

3.2.2 Function Documentation

3.2.2.1 uint32_t crc32Buffer (const void * *buffer*, size_t *bufferLength*)

This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

buffer	Input buffer.
bufferLength	Buffer length in Bytes.

Returns

32-bit CRC value.

Definition at line 66 of file ErriezCRC32.c.

3.2.2.2 uint32_t crc32Final (uint32_t *crc*)

Finalize CRC32 output.

This function should be called after crc32Update(). This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

crc	Previous CRC32 value.

Returns

32-bit CRC value.

Definition at line 113 of file ErriezCRC32.c.

3.2.2.3 uint32_t crc32String (const char * *buffer*)

Calculate CRC on a character string.

This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

buffer	Null terminated character input array.

Returns

32-bit CRC value.

Definition at line 49 of file ErriezCRC32.c.

3.2.2.4 uint32_t crc32Update (const void * *buffer*, size_t *bufferLength*, uint32_t *crc*)

Calculate CRC on multiple buffers.

This function is thread safe and maybe called from interrupt handlers. No buffer alignment needed.

Parameters

buffer	Input buffer. No alignment needed.
bufferLength	Input buffer length in Bytes. No alignment needed.
crc	Previous CRC value. Call with crc = CRC32_INITIAL on first calculation.

Returns

32-bit updated CRC value.

Definition at line 85 of file ErriezCRC32.c.

Index