CoreS3 RTC リアルタイムクロック

CoreS3 RTC クロック関連 API およびサンプルプログラム。

サンプルプログラム

コンパイル要件

- M5Stack ボード マネージャー バージョン >= 3.2.2
- ボード選択 = M5CoreS3
- M5Unified ライブラリ バージョン >= 0.2.11

cpp

```cpp
#if defined(ARDUINO)

#define WIFI_SSID     "***********"
#define WIFI_PASSWORD "***********"
#define NTP_TIMEZONE  "UTC-8"
#define NTP_SERVER1   "0.pool.ntp.org"
#define NTP_SERVER2   "1.pool.ntp.org"
#define NTP_SERVER3   "2.pool.ntp.org"

#include <WiFi.h>

// Different versions of the framework have different SNTP header file names and
// availability.
#if __has_include(<esp_sntp.h>)
#include <esp_sntp.h>
#define SNTP_ENABLED 1
#elif __has_include(<sntp.h>)
#include <sntp.h>
#define SNTP_ENABLED 1
#endif

#endif

#ifndef SNTP_ENABLED
#define SNTP_ENABLED 0
#endif

#include <M5Unified.h>

void setup(void) {
    M5.begin();
    M5.Display.setFont(&fonts::FreeMonoBold12pt7b);

    if (!M5.Rtc.isEnabled()) {
        Serial.println("RTC not found.");
        for (;;) {
            vTaskDelay(500);
        }
    }

    Serial.println("RTC found.");

    // It is recommended to set UTC for the RTC and ESP32 internal clocks.
    /* /// setup RTC ( direct setting )
      //                      YYYY  MM  DD      hh  mm  ss
      M5.Rtc.setDateTime( { { 2021, 12, 31 }, { 12, 34, 56 } } );

    //*/

    /// setup RTC ( NTP auto setting )
```

```
51
52      M5.Display.println("WiFi Connecting...");
53
54      WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
55      while (WiFi.status() != WL_CONNECTED) {
56          Serial.print('.');
57          delay(500);
58      }
59      M5.Display.println("WiFi Connected.");
60      Serial.println("\r\n WiFi Connected.");
61
62      configTzTime(NTP_TIMEZONE, NTP_SERVER1, NTP_SERVER2, NTP_SERVER3);
63
64  #if SNTP_ENABLED
65      while (sntp_get_sync_status() != SNTP_SYNC_STATUS_COMPLETED) {
66          Serial.print('.');
67          delay(1000);
68      }
69      Serial.println("\r\n NTP Connected.");
70      M5.Display.println("NTP Connected");
71  #else
72      delay(1600);
73      struct tm timeInfo;
74      while (!getLocalTime(&timeInfo, 1000)) {
75          Serial.print('.');
76      };
77  #endif
78
79      time_t t = time(nullptr) + 1;  // Advance one second.
80      while (t > time(nullptr))
81          ;  /// Synchronization in seconds
82      M5.Rtc.setDateTime(gmtime(&t));
83      M5.Display.clear();
84  }
85
86  void loop(void) {
87      static constexpr const char* const wd[7] = {"Sun", "Mon", "Tue", "Wed",
88                                                  "Thr", "Fri", "Sat"};
89
90      delay(500);
91
92      auto dt = M5.Rtc.getDateTime();
93      Serial.printf("RTC   UTC  :%04d/%02d/%02d (%s)  %02d:%02d:%02d\r\n",
94                    dt.date.year, dt.date.month, dt.date.date,
95                    wd[dt.date.weekDay], dt.time.hours, dt.time.minutes,
96                    dt.time.seconds);
97      M5.Display.setCursor(0, 0);
98      M5.Display.printf("RTC   UTC  :%04d/%02d/%02d (%s)  %02d:%02d:%02d\n",
99                            dt.date.year, dt.date.month, dt.date.date,
100                           wd[dt.date.weekDay], dt.time.hours, dt.time.minutes,
101                           dt.time.seconds);
102
103     /// ESP32 internal timer
```

```
    auto t = time(nullptr);
    {
        auto tm = gmtime(&t);  // for UTC.
        Serial.printf("ESP32 UTC  :%04d/%02d/%02d (%s)  %02d:%02d:%02d\r\n",
                      tm->tm_year + 1900, tm->tm_mon + 1, tm->tm_mday,
                      wd[tm->tm_wday], tm->tm_hour, tm->tm_min, tm->tm_sec);
        M5.Display.printf("ESP32 UTC  :%04d/%02d/%02d (%s)  %02d:%02d:%02d\n",
                              tm->tm_year + 1900, tm->tm_mon + 1, tm->tm_mday,
                              wd[tm->tm_wday], tm->tm_hour, tm->tm_min,
                              tm->tm_sec);
    }

    {
        auto tm = localtime(&t);  // for local timezone.
        Serial.printf("ESP32 %s:%04d/%02d/%02d (%s)  %02d:%02d:%02d\r\n",
                      NTP_TIMEZONE, tm->tm_year + 1900, tm->tm_mon + 1,
                      tm->tm_mday, wd[tm->tm_wday], tm->tm_hour, tm->tm_min,
                      tm->tm_sec);
        M5.Display.printf("ESP32 %s:%04d/%02d/%02d (%s)  %02d:%02d:%02d\n",
                              NTP_TIMEZONE, tm->tm_year + 1900, tm->tm_mon + 1,
                              tm->tm_mday, wd[tm->tm_wday], tm->tm_hour,
                              tm->tm_min, tm->tm_sec);
    }
}
```

アップロードが成功した後、CoreS3 の画面に現在の RTC 時刻と ESP32 内部クロックの時刻情報が表示されます。RTC 時刻は NTP 同期で設定され、ESP32 内部クロックはシステム時刻関数で取得されます。両者は同期しているはずです。

API

CoreS3 RTC クロックセクションは M5Unified ライブラリの `RTC8563_Class` を使用しています。関連する詳細な API については、以下のドキュメントを参照してください:

- M5Unified - RTC8563 Class