PaperColor M5PM1 電源管理

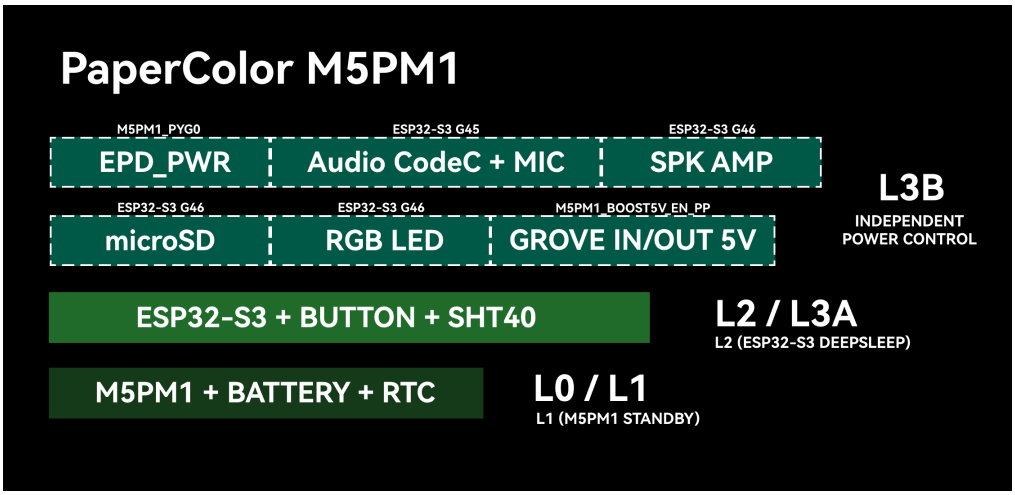
1. 多段電源スイッチ設計

PaperColor は内部に M5PM1 を搭載しており、ハードウェア回路と連携して多段電源スイッチを実現しています。異なるレベルの電源スイッチが対応する周辺機器やインターフェースの給電を制御します。ユーザーは動作要件に応じて電源レベルを切り替え、未使用の周辺機器の電源を遮断することで、システム全体の低消費電力化を実現できます。

M5PM1 ドライバライブラリ

M5PM1 ドライバライブラリを使用することで、M5PM1 のピン機能を簡単に設定でき、低消費電力ウェイクアップや周辺機器の電源スイッチ制御に利用できます。

- M5PM1 Arduino Library



注意事項

PaperColor の多段電源スイッチ設計は直列構造ではありません。L1 ~ L3B スイッチの電源入力はすべて L0（SYS_VBUS）レベルから供給されており、前段の電源からではないため、各レベルの電源スイッチを独立して制御できます。上記の分類は周辺機器の給電と消費電力に基づいて区分されています。M5PM1 の起動後、L1、L2、L3A は自動的に有効になります（デフォルトで `DCDC3V3_EN_PP`、`LDO3V3_EN_PP`、`CHG_EN_PP` がオン）。

L0 / L1

この電源レベルでは、バッテリーが M5PM1 および RTC への給電を維持します。バッテリー残量がある限り、この電源レベルは常に維持され、M5PM1 は基本的な電源ボタンによるオン/オフ操作をサポートします。

L0 レベルは M5PM1 がシャットダウン状態であることを示します。

L1 レベルでは、M5PM1 はスタンバイ状態です。

L2 / L3A

この電源レベルでは、ESP32-S3 メインコントローラ、SHT40I 温湿度センサー、およびユーザーボタンのプルアップ抵抗への給電（3V3_L2）が有効になります。

ESP32-S3 がスリープ状態のとき、電源は L2 レベルです。 ESP32-S3 が動作状態のとき、電源は L3A レベルです。

ESP32-S3 は M5PM1 をスリープに移行させることで、自身の給電を遮断できます（L2→L1/L0）。

L3B

この電源レベルでは、M5PM1 と ESP32-S3 GPIO を通じてオンボード周辺機器の給電をさらに制御できます。

ESP32S3R8	G45
ES8311 & ES7210 Power (CODEC_3V3_L3B)	AUDIO_PWR_EN

ESP32S3R8	G46
AW8737A	SPK_EN

- G46(SPK_EN): スピーカーアンプ有効化
- G45(AUDIO_PWR_EN): オーディオコーデックチップおよびマイク給電

M5PM1	DCDC3V3_EN_PP	LDO3V3_EN_PP	BOOST5V_EN_PP
3V3_L2	PY_MPWR_EN		
RGB LED		PY_RGB_PWR_EN	
Grove			PY_GROVE_OUT_EN

- DCDC3V3_EN_PP(PY_MPWR_EN): 3V3_L2 レベル電源スイッチ
- LDO3V3_EN_PP(PY_RGB_PWR_EN): RGB LED 給電スイッチ
- BOOST5V_EN_PP(PY_GROVE_OUT_EN): Grove 拡張インターフェース給電方向制御

M5PM1	PYG0	PYG2	PYG4	PYG3	PYG1
E-Paper	PY_EPD_EN				
RTC		RTC_IRQ			
microSD			PY_SD_DET_EN	PY_SD_PWR_EN	CARD_DEC

- PYG0(PY_EPD_EN): 電子ペーパーディスプレイ給電有効化
- PYG2(RTC_IRQ): RTC 割り込み信号
- PYG3(PY_SD_PWR_EN): microSD モジュール給電
- PYG4(PY_SD_DET_EN): microSD 検出機能有効化、プルアップ有効
- PYG1(CARD_DEC): microSD 挿入検出

2. M5PM1 スリープ

手動スリープ

M5PM1 はプログラムから手動でスリープ状態に移行させることで、システム全体の消費電力を低減できます。デフォルト状態では、スリープを直接設定すると L0 レベルの電源に戻り、M5PM1 と RTC のみが給電を維持します。

```
pm1.shutdown();
```

一部の特殊な使用シーン（例：ESP32-S3 SoC スリープモード）では、M5PM1 をスリープに移行させて消費電力を低減しつつ、一部のレベルの周辺機器給電を維持し、ウェイクアップソースやステート保持に利用できます。

このようなアプリケーションでは、M5PM1 がスリープモードに入る前に、対応するレベルの電源スイッチピンの状態を設定し、ステート保持を有効化する必要があります。

I2C アイドルスリープ

M5PM1 は I2C アイドル通信による自動スリープ機能の設定をサポートしており、システム全体の消費電力を低減できます。スリープ状態に入った後、ESP32-S3 と M5PM1 の最初の通信は M5PM1 のウェイクアップに使用されるため通信失敗となり、有効な通信はウェイクアップ後の次回通信からとなります。

```
m5pm1_err_t setI2cSleepTime(uint8_t seconds);
```

3. M5PM1 Timer タイマー

M5PM1 はタイマー機能の設定をサポートしており、カウント終了後に電源オン、電源オフ、リセットなどの対応する操作を実行できます。

```
m5pm1_err_t timerSet(uint32_t seconds, m5pm1_tim_action_t action);
```

```
typedef enum {
    M5PM1_TIM_ACTION_STOP = 0b000,     // 停止，無動作
                                       // Stop, no action
    M5PM1_TIM_ACTION_FLAG = 0b001,     // フラグのみ設定
                                       // Set flag only
    M5PM1_TIM_ACTION_REBOOT = 0b010,   // システムリセット
                                       // System reboot
    M5PM1_TIM_ACTION_POWERON = 0b011,  // 電源オン
                                       // Power on
    M5PM1_TIM_ACTION_POWEROFF = 0b100  // 電源オフ
                                       // Power off
} m5pm1_tim_action_t;
```

タイマー電源オン / 電源オフのサンプル

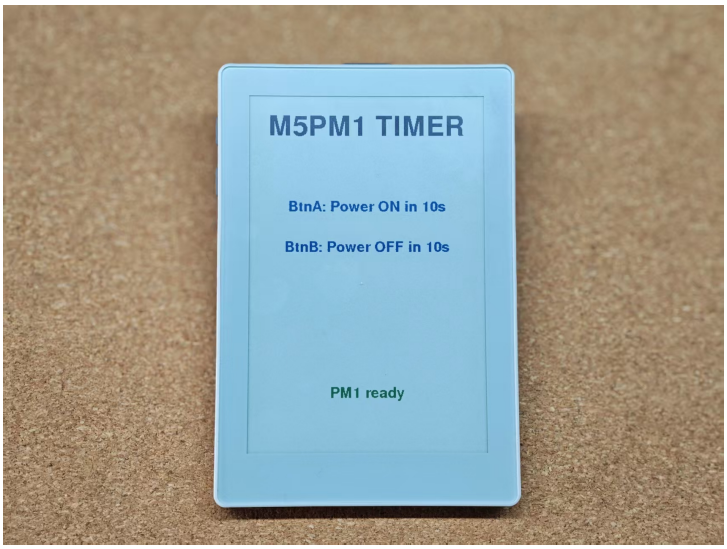
サンプル説明：デバイス起動後、ボタン A を1 回押すと 10 秒タイマーで電源オンイベントを設定し、その後デバイスはシャットダウンしてウェイクアップを待ちます。ボタン B を1 回押すと、10 秒タイマーで M5PM1 のシャットダウンを設定します（電源ボタンを1 回押すと再起動できます）。

cpp

```cpp
#include <M5Unified.h>
#include <M5PM1.h>
#include <Wire.h>
#include <Adafruit_NeoPixel.h>

static constexpr uint32_t POWERON_SECONDS = 10;
static constexpr uint32_t POWEROFF_SECONDS = 10;
static constexpr uint8_t LED_PIN = 21;
static constexpr uint8_t LED_COUNT = 2;

M5PM1 pm1;
M5Canvas canvas(&M5.Display);
Adafruit_NeoPixel pixels(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

static bool pm1_ready = false;

static void setAllLeds(uint32_t color)
{
    for (uint8_t i = 0; i < LED_COUNT; ++i) {
        pixels.setPixelColor(i, color);
    }
    pixels.show();
}

static void drawScreen(const char* status)
{
    const int w = M5.Display.width();
    const int h = M5.Display.height();

    canvas.fillSprite(WHITE);
    canvas.setTextDatum(top_center);
    canvas.setTextColor(BLACK);

    canvas.setFont(&fonts::FreeSansBold24pt7b);
    canvas.drawString("M5PM1 TIMER", w / 2, 34);

    canvas.setFont(&fonts::FreeSansBold12pt7b);
    canvas.setTextColor(BLUE);
    canvas.drawString("BtnA: Power ON in 10s", w / 2, 180);
    canvas.drawString("BtnB: Power OFF in 10s", w / 2, 248);

    canvas.setTextColor(pm1_ready ? GREEN : RED);
    canvas.drawString(status, w / 2, h - 110);

    canvas.pushSprite(0, 0);
}

void setup(void)
{
    auto cfg = M5.config();
```

```
    cfg.clear_display = false;
    M5.begin(cfg);

    Serial.begin(115200);
    M5.Display.setEpdMode(epd_mode_t::epd_quality);

    canvas.createSprite(M5.Display.width(), M5.Display.height());

    m5pm1_err_t err = pm1.begin(&M5.In_I2C, M5PM1_DEFAULT_ADDR, M5PM1_I2C_FREQ_100K);
    pm1_ready       = (err == M5PM1_OK);

    if (pm1_ready) {
        pm1.setLdoEnable(true);
    }

    pixels.begin();
    pixels.setBrightness(60);
    setAllLeds(0);

    if (pm1_ready) {
        Serial.println("PM1 init ok");
        drawScreen("PM1 ready");
        setAllLeds(pixels.Color(0, 255, 0));
    } else {
        Serial.printf("PM1 init failed: %d\n", (int)err);
        drawScreen("PM1 init failed");
        setAllLeds(pixels.Color(255, 0, 0));
    }
}

void loop(void)
{
    M5.update();

    if (!pm1_ready) {
        delay(20);
        return;
    }

    if (M5.BtnA.wasPressed()) {
        Serial.printf("BtnA: timer power on in %lu s\n", POWERON_SECONDS);
        setAllLeds(pixels.Color(255, 255, 0));
        m5pm1_err_t err = pm1.timerSet(POWERON_SECONDS, M5PM1_TIM_ACTION_POWERON);
        if (err != M5PM1_OK) {
            Serial.printf("timerSet POWERON failed: %d\n", (int)err);
            drawScreen("BtnA failed");
            setAllLeds(pixels.Color(255, 0, 0));
        }
        delay(1000);
        pm1.shutdown();
    }

    if (M5.BtnB.wasPressed()) {
```

```
104         Serial.printf("BtnB: timer poweroff in %lu s\n", POWEROFF_SECONDS);
105         setAllLeds(pixels.Color(0, 0, 255));
106         m5pm1_err_t err = pm1.timerSet(POWEROFF_SECONDS, M5PM1_TIM_ACTION_POWEROFF);
107         if (err != M5PM1_OK) {
108             Serial.printf("timerSet POWEROFF failed: %d\n", (int)err);
109             drawScreen("BtnB failed");
110             setAllLeds(pixels.Color(255, 0, 0));
111         }
112     }
113
114     delay(20);
115 }
```



4. RTC M5PM1 ウェイクアップ

デバイス起動後、ボタン A を1 回押すと RTC タイマーウェイクアップを設定し、M5PM1 をスリープ状態に移行させます。この時、システムは最低の L0 レベル給電のみを維持し、IMU と M5PM1 の動作を維持します。RTC がウェイクアップ信号をトリガーすると、M5PM1 が再起動します。M5PM1 のウェイクアップ後、L1、L2、L3A の電源投入シーケンスが再度実行され、ESP32-S3 は初期化を再実行します。

cpp

```cpp
#include <M5Unified.h>
#include <M5PM1.h>
#include <Wire.h>
#include <Adafruit_NeoPixel.h>

static constexpr int RTC_WAKE_AFTER_SECONDS = 20;

M5Canvas canvas(&M5.Display);
M5PM1 pm1;
Adafruit_NeoPixel pixels(2, 21, NEO_GRB + NEO_KHZ800);

static bool pm1_ready = false;

static void setAllLeds(uint32_t color)
{
    pixels.setPixelColor(0, color);
    pixels.setPixelColor(1, color);
    pixels.show();
}

static void drawScreen(const char* status, uint32_t status_color)
{
    const int w = M5.Display.width();
    const int h = M5.Display.height();

    canvas.fillSprite(WHITE);
    canvas.setTextDatum(top_center);

    canvas.setFont(&fonts::FreeSansBold24pt7b);
    canvas.setTextColor(BLACK);
    canvas.drawString("RTC Wake PM1", w / 2, 40);

    canvas.setFont(&fonts::FreeSansBold18pt7b);
    canvas.setTextColor(RED);
    canvas.drawString("BtnA: Shutdown", w / 2, 160);

    canvas.setTextColor(BLUE);
    canvas.drawString("RTC Alarm", w / 2, 270);
    canvas.drawString("wakes PM1 after 10s", w / 2, 316);

    canvas.setTextColor(status_color);
    canvas.drawString(status, w / 2, 430);

    canvas.pushSprite(0, 0);
}

void setup(void)
{
    auto cfg          = M5.config();
    cfg.clear_display = false;
```

```
    M5.begin(cfg);
    Serial.begin(115200);

    M5.Display.setEpdMode(epd_mode_t::epd_quality);
    canvas.createSprite(M5.Display.width(), M5.Display.height());

    pixels.begin();
    pixels.setBrightness(80);
    setAllLeds(0);

    m5pm1_err_t err = pm1.begin(&M5.In_I2C, M5PM1_DEFAULT_ADDR, M5PM1_I2C_FREQ_100K);
    pm1_ready       = (err == M5PM1_OK);

    if (pm1_ready) {
        pm1.setLdoEnable(true);
        setAllLeds(pixels.Color(0, 255, 0));
        drawScreen("PM1 ready", GREEN);
        Serial.println("PM1 initialization successful");
        pm1.gpioSetWakeEnable(M5PM1_GPIO_NUM_2, true);
        pm1.gpioSetWakeEdge(M5PM1_GPIO_NUM_2, M5PM1_GPIO_WAKE_FALLING);  // Falling edge

    } else {
        setAllLeds(pixels.Color(255, 0, 0));
        drawScreen("PM1 init failed", RED);
        Serial.printf("PM1 initialization failed, error code: %d\n", err);
    }
}

void loop(void)
{
    M5.update();

    if (!pm1_ready) {
        delay(20);
        return;
    }

    if (M5.BtnA.wasPressed()) {
        M5.Rtc.disableIRQ();
        M5.Rtc.clearIRQ();
        M5.Rtc.setAlarmIRQ(RTC_WAKE_AFTER_SECONDS);

        for (int i = 0; i < 3; i++) {
            setAllLeds(pixels.Color(255, 0, 0));
            delay(200);
            setAllLeds(0);
            delay(200);
        }
        pm1.shutdown();
    }

    delay(20);
}
```


RTC Wake PM1
BtnA: Shutdown
RTC Alarm
wakes PM1 after 10s
PM1 ready