PaperMono Display 屏幕显示

PaperMono 显示相关 API 与案例程序。

案例程序

编译要求

- M5Stack 板管理版本 >= 3.3.9
- 开发板选项 = `M5PaperMono`
- M5Unified 库版本 >= 0.2.20
- M5GFX 库版本 >= 0.2.27

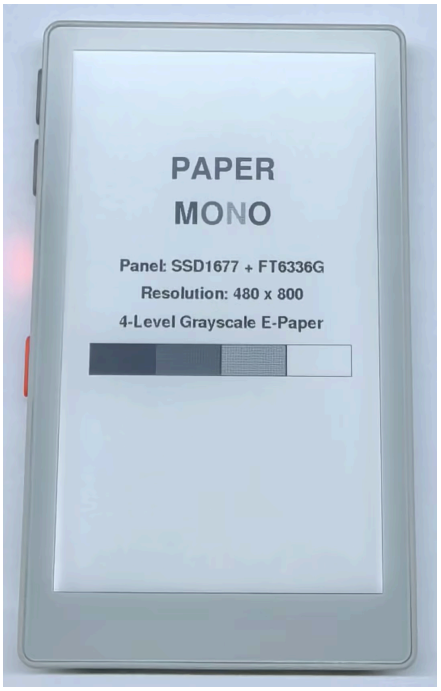
基础显示

cpp

```cpp
#include <Arduino.h>
#include <M5Unified.h>

M5Canvas canvas(&M5.Display);

static void drawBlackBorder()
{
    const int screenW = M5.Display.width();
    const int screenH = M5.Display.height();

    canvas.drawRect(0, 0, screenW, screenH, TFT_BLACK);
    canvas.drawRect(2, 2, screenW - 4, screenH - 4, TFT_BLACK);
    canvas.drawRect(4, 4, screenW - 8, screenH - 8, TFT_BLACK);
}

static int drawGrayscaleWordCentered(const char* word, int y, const uint16_t* colors, int colorCount)
{
    const int screenW = M5.Display.width();
    const int len     = strlen(word);
    int totalW        = 0;

    for (int i = 0; i < len; ++i) {
        totalW += canvas.textWidth(String(word[i]));
    }

    int x = (screenW - totalW) / 2;
    for (int i = 0; i < len; ++i) {
        canvas.setTextColor(colors[i % colorCount]);
        String ch(word[i]);
        canvas.drawString(ch, x, y);
        x += canvas.textWidth(ch);
    }
    return x;
}

static void drawMainDemo()
{
    const int screenW = M5.Display.width();
    const int screenH = M5.Display.height();

    static const uint16_t paperColors[1] = {TFT_BLACK};
    static const uint16_t monoColors[4] = {
        TFT_BLACK,
        TFT_RED,
        TFT_DARKGREY,
        TFT_BLACK
    };
    static const uint16_t grayColors[4] = {
        TFT_BLACK,
        TFT_RED,
        TFT_DARKGREY,
        TFT_WHITE
    };

    canvas.fillSprite(TFT_WHITE);
```

```
    drawBlackBorder();

    canvas.setTextDatum(top_left);
    canvas.setTextSize(1);
    canvas.setFont(&fonts::FreeSansBold24pt7b);

    drawGrayscaleWordCentered("PAPER", 168, paperColors, 1);
    drawGrayscaleWordCentered("MONO", 236, monoColors, 4);

    canvas.setFont(&fonts::FreeSansBold12pt7b);
    canvas.setTextColor(TFT_BLACK);
    canvas.setTextDatum(middle_center);
    canvas.drawString("Panel: SSD1677 + FT6336G", screenW / 2, 336);

    char resolutionBuf[32];
    snprintf(
        resolutionBuf,
        sizeof(resolutionBuf),
        "Resolution: %d x %d",
        screenW,
        screenH
    );
    canvas.drawString(resolutionBuf, screenW / 2, 376);

    canvas.drawString(
        "4-Level Grayscale E-Paper",
        screenW / 2,
        418
    );

    const int barX  = 46;
    const int barY  = 448;
    const int barW  = screenW - 92;
    const int barH  = 46;
    const int cellW = barW / 4;

    for (int i = 0; i < 4; ++i) {
        const int x = barX + i * cellW;
        const int w = (i == 3) ? (barX + barW - x) : cellW;

        canvas.fillRect(x, barY, w, barH, grayColors[i]);
        canvas.drawRect(x, barY, w, barH, TFT_BLACK);
    }

    canvas.pushSprite(0, 0);
}

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

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

```
    drawMainDemo();
}

void loop()
{
    M5.update();
    delay(100);
}
```



刷新模式

PaperMono 支持以下四种电子纸刷新模式。刷新速度越快，显示质量和残影控制通常越弱，实际刷新时间还会受图像内容影响。

cpp

```cpp
M5.Display.setEpdMode(epd_mode_t::epd_quality); // High quality, slowest refresh
M5.Display.setEpdMode(epd_mode_t::epd_text);    // Text mode, faster refresh, lower quality
M5.Display.setEpdMode(epd_mode_t::epd_fast);    // Fast refresh, lower quality
M5.Display.setEpdMode(epd_mode_t::epd_fastest); // Fastest refresh, lowest quality
```

PaperMono 的参考刷新耗时如下：

刷新模式	参考耗时
`epd_quality`	4.71 s
`epd_text`	0.45 s
`epd_fast`	0.34 s
`epd_fastest`	0.07 s

调用 `M5.Display.setEpdMode()` 后，后续的屏幕更新将使用对应模式。建议在文字界面使用 `epd_text`，在图片或需要较好灰度效果的场景使用 `epd_quality`。连续进行多次局部快速刷新后，应适时进行一次全屏高质量刷新以减轻残影。

cpp

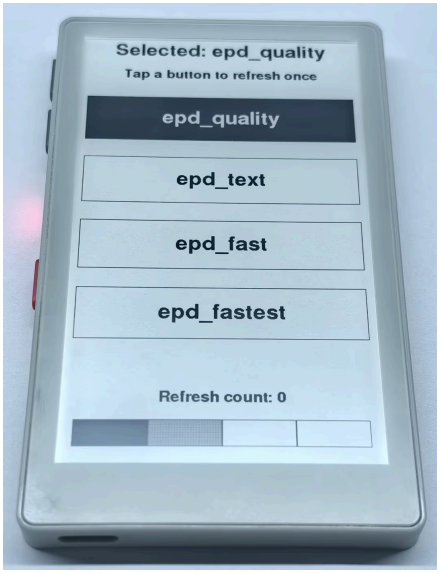
```cpp

```

```cpp
#include <M5Unified.h>
#include <M5GFX.h>

M5Canvas canvas(&M5.Display);

struct ModeOption {
    const char* name;
    epd_mode_t mode;
};

const ModeOption modeOptions[] = {
    {"epd_quality", epd_mode_t::epd_quality},
    {"epd_text", epd_mode_t::epd_text},
    {"epd_fast", epd_mode_t::epd_fast},
    {"epd_fastest", epd_mode_t::epd_fastest},
};

constexpr std::size_t MODE_COUNT = sizeof(modeOptions) / sizeof(modeOptions[0]);
constexpr int BUTTON_X = 16;
constexpr int BUTTON_Y = 130;
constexpr int BUTTON_H = 90;
constexpr int BUTTON_GAP = 34;

std::size_t selectedMode = 0;
uint32_t refreshCount = 0;
LGFX_Button modeButtons[MODE_COUNT];

int buttonWidth()
{
    return M5.Display.width() - BUTTON_X * 2;
}

void initModeButtons()
{
    const int width = buttonWidth();
    for (std::size_t i = 0; i < MODE_COUNT; ++i) {
        const int y = BUTTON_Y + static_cast<int>(i) * (BUTTON_H + BUTTON_GAP);
        modeButtons[i].initButtonUL(&M5.Lcd, BUTTON_X, y, width, BUTTON_H,
                                    TFT_BLACK, TFT_WHITE, TFT_BLACK,
                                    modeOptions[i].name, 1, 1);
    }
}

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

    canvas.fillSprite(TFT_WHITE);
    canvas.setTextDatum(middle_center);
    canvas.setFont(&fonts::FreeSansBold18pt7b);
    canvas.setTextColor(TFT_BLACK);

    char selectedText[40];
    snprintf(selectedText, sizeof(selectedText), "Selected: %s", modeOptions[selectedMode].name);
```

```
    canvas.drawString(selectedText, w / 2, 30);

    canvas.setFont(&fonts::FreeSansBold12pt7b);
    canvas.drawString("Tap a button to refresh once", w / 2, 82);

    canvas.setFont(&fonts::FreeSansBold18pt7b);

    for (std::size_t i = 0; i < MODE_COUNT; ++i) {
        const int y = BUTTON_Y + static_cast<int>(i) * (BUTTON_H + BUTTON_GAP);
        const bool selected = i == selectedMode;
        const int width = buttonWidth();

        if (selected) {
            canvas.fillRect(BUTTON_X, y, width, BUTTON_H, TFT_BLACK);
        } else {
            canvas.fillRect(BUTTON_X, y, width, BUTTON_H, TFT_WHITE);
        }
        canvas.drawRect(BUTTON_X, y, width, BUTTON_H, TFT_BLACK);
        canvas.setTextColor(selected ? TFT_WHITE : TFT_BLACK);
        canvas.drawString(modeOptions[i].name, w / 2, y + BUTTON_H / 2);
    }

    char countText[32];
    snprintf(countText, sizeof(countText), "Refresh count: %lu", (unsigned long)refreshCount);
    canvas.setFont(&fonts::FreeSansBold12pt7b);
    canvas.setTextColor(TFT_BLACK);
    canvas.drawString(countText, w / 2, h - 110);

    const int stripeY = h - 75;
    const int stripeW = (w - 48) / 4;
    const uint16_t stripeColors[] = {TFT_BLACK, TFT_DARKGREY, TFT_LIGHTGREY, TFT_WHITE};
    for (int i = 0; i < 4; ++i) {
        const int x = 24 + i * stripeW;
        canvas.fillRect(x, stripeY, stripeW, 42, stripeColors[i]);
        canvas.drawRect(x, stripeY, stripeW, 42, TFT_BLACK);
    }

    canvas.pushSprite(0, 0);
}

void selectMode(std::size_t index)
{
    selectedMode = index;
    M5.Display.setEpdMode(modeOptions[selectedMode].mode);
    ++refreshCount;
    drawScreen();
}

void setup()
{
    auto cfg = M5.config();
    cfg.clear_display = false;
    M5.begin(cfg);
    M5.Display.setRotation(0);
    M5.Display.setEpdMode(modeOptions[selectedMode].mode);
```

```
111
112     initModeButtons();
113     canvas.createSprite(M5.Display.width(), M5.Display.height());
114     drawScreen();
115 }
116
117 void loop()
118 {
119     M5.update();
120     m5::touch_point_t touchPoint;
121     const bool touched = M5.Display.getTouchRaw(&touchPoint, 1) > 0;
122
123     for (std::size_t i = 0; i < MODE_COUNT; ++i) {
124         const bool inside = touched && modeButtons[i].contains(touchPoint.x, touchPoint.y);
125         modeButtons[i].press(inside);
126         if (modeButtons[i].justPressed()) {
127             selectMode(i);
128         }
129     }
130     delay(5);
131 }
```

电子纸不适合高频连续刷新，测试不同模式时应等待当前刷新完成后再选择下一个按钮。



前光亮度

PaperMono 的前光由 M5PM1 通过 PWM 控制，可以使用 `M5.Display.setBrightness()` 设置亮度。亮度范围为 `0 ~ 255`，设置为 `0` 可关闭前光。

```
M5.Display.setBrightness(255); // 0 ~ 255
M5.Display.setBrightness(0);   // 关闭前光
```