



Micropython - WiFiBoy Module

[[Micropython for WiFiBoy OK:ESP32 v2.04 \(Released in March 2020\)](#)]

WiFiBoy Module for Micropython Documentation v1.01TW

一、WiFiBoy Module 模組簡介

WiFiBoy 模組是專為 WiFiBoy 系列玩學機設計的擴充功能集，包括基本的LCD繪圖功能、GPIO按鍵處理，以及一個有趣且適合程式學習入門的小龜繪圖系統，還有一個進階的 BLIT 遊戲快顯系統，可以用來製作一些高效率顯示的互動應用程式。

在 WiFiBoy Micropython 啟動之後，會自動執行 **import wifiboy as wb** 的初始化指令，用戶只要使用 wb 縮寫即可使用 WiFiBoy Module，例如清除螢幕的指令就是 **wb.cls()**。

二、WiFiBoy Module 模組指令介紹

基本繪圖、按鍵處理系統

```
wb.init() - 啟動 WiFiBoy Module 系統
wb.cls([color]) - 清除 LCD 螢幕
wb.box(x, y, w, h, color) - 在 LCD 上畫一個實心的彩色方塊
wb.line (x1, y1, x2, y2, color[, width]) - 在 LCD 上畫出一條彩色的線
wb.circle(x, y, r, color [, width]) - 在 LCD 上畫出一個彩色的空心圓
wb.pix(x, y, color) - 在 LCD 上畫出一個彩色的點
wb.win(x1, y1, x2, y2) - 設定一個可以填色的範圍(自動換行)
wb.pushpix(color) - 在一個範圍內自動填色
wb.str(str, x, y[, font[, size]]) - 在 LCD 上顯示一個彩色的英數字串
wb.colors(color1, color2) - 設定文字字串的颜色
wb.img(x, y, w, h, buffer) - 將一個 bytearray 的圖像快速顯示出來
wb.showbuf(buffer) - 將一個全螢幕的 bytearray 圖像快速顯示出來
```

```
wb.showbmp(file) - 將一個全螢幕的 JPEG 圖像快速顯示出來
wb.showjpg(file) - 將一個全螢幕的 16-bit BMP 格式圖像快速顯示出來
wb.rand([start, end] | [range]) - 產生一個隨機整數亂數（硬體產生）
wb.getkey() - 讀取按鍵的組合值
```

小龜繪圖指令集

```
wb.ttreset() - 讓小龜回家（小龜開始）
wb.ttpos(x, y) - 移動小龜到某個座標（不產生足跡）
wb.ttgo(step) - 小龜前進 step 步
wb.ttrots(degree) - 小龜旋轉 degree 度
wb.ttpenup(mode) - 小龜畫筆提起或放下
wb.ttcOLOR(color) - 小龜畫筆顏色
wb.ttwIDTH(width) - 小龜畫筆的粗細
```

BLIT 遊戲快顯繪圖系統

```
wb.blit() - 將 off-screen 圖像快顯出來
wb.blitbuf([address, data]) - 清除或填寫 off-screen 記憶體
wb.blitimg(sprite, x, y) - 以 8x8 為單位搬移一個 8 x 8 的圖元到 off-screen 記憶體
wb.blitusr(sprite, x, y) - 以 8x8 為單位搬移一個 8 x 8 的自定義圖元到 off-screen 記憶體
wb.setusr(buf, len) - 設定用戶自訂義圖庫
wb.blitpal([num, pal]) - 設定 256 個調色盤
wb.blitstr(str, x, y) - 搬移內建圖庫字型區域的字串
wb.blitfree() - 釋放 off-screen 的記憶體
```

基本繪圖、按鍵處理系統

wb.init() - 啟動 WiFiBoy Module 系統

- 這個指令在 WiFiBoy 開機啟動時就會自動執行，用戶不須使用這個指令。

wb.cls([color]) - 清除 LCD 螢幕

- [color] 表示可以省略這個參數。
- 如果沒有設定 color 值，則預設為黑色值 0。
- 顏色值的範圍是整數 0 ~ 65535，每個數值都對應到不同的顏色。數值規則定義如下：

顏色數值的規則是：**RRRRR**(5bits)+**GGGggg**(6bits)+**BBBBB**(5bits) 組成的 16bits 兩個位元組，再進行位元組交換後的數值。也就是 **RRRRRGGG gggBBBB**，交換後的數值即為 **gggBBBBB RRRRRGGG** 兩個八位元組數值。

例如紅色為 **RRRR RGGG gggB BBBB** = **1111 1000 0000 0000**，紅色位元都是 1，綠色位元與藍色位元都為 0。
也就是組成十六進位的 **[F800]**，位元組交換後得到紅色顏色值為 **[00F8]** = **248**。

再看一個綠色值的範例：**0000 0111 1110 0000** = **[07E0]**，交換後的綠色顏色值即為 **[E007]**，轉換為十進位就是 **57351**。

- 以下是幾種常見顏色的 WiFiBoy Module 預設名稱與顏色值對照表：

BLACK -- 0	NAVY -- 3840	DARKGREEN -- 57347	DARKCYAN -- 61187
MAROON -- 120	PURPLE -- 3960	OLIVE -- 57467	LIGHTGREY -- 6342
DARKGREY -- 61307	BLUE -- 7936	GREEN -- 57351	CYAN -- 65287
RED -- 248	MAGENTA -- 8184	YELLOW -- 57599	WHITE -- 65535
ORANGE -- 8445	PINK -- 8184	GREENYELLOW -- 58799	

- WiFiBoy Module 支援上表的 19 種顏色名稱定義，我們可以使用顏色的英文大寫名稱來取得顏色值，例如 **wb.cls(wb.WHITE)** 即可清除螢幕為白色。
- 如果我們有個創意，想要使用隨機顏色來改變螢幕顏色，可以使用「亂數函數」**wb.rand(65536)** 取得任何一個 0~65535 的顏色值。所以結合清除螢幕的值令，就可以組合成一個「讓螢幕隨機變色」的指令：**wb.cls(wb.rand(65536))**。

wb.box(x, y, w, h, color) - 在 LCD 上畫出一個實心的彩色方塊

- (x, y) 是方塊左上角在螢幕的座標。WiFiBoy LCD 螢幕的原點(0,0)是在左上角。

WiFiBoy玩學機的不同版本的螢幕大小定義如下：

```
* Mini: 128 x 128
* Edu/Classic: 128 x 160
* Pro: 240 x 320
* OK:ESP32: 160 x 128
```

- (w, h) 是方塊的寬度與高度。超出螢幕的部分會自動被忽略。
- color 顏色值範圍 0-65535。顏色值規則請參考 **wb.cls()** 的說明。
- 如果 x, y 給的數值是在螢幕外的座標，也是可以畫出方塊的。例如 **wb.box(-10, -10, 20, 20, wb.RED)**，會在 LCD 螢幕左上角畫出一個 10 x 10 的紅色方塊。雖然方塊設定值為 20 x 20，但因為方塊左上角起點座標是在螢幕外面，所以只能畫出看得見的部位。
- 實心方塊的範例：**wb.box(10, 10, 100, 100, wb.YELLOW)**

wb.line(x1, y1, x2, y2, color[, width]) - 在 LCD 上畫出一條彩色的線

- [width] 是線條的寬度，範圍是 1~16。這個寬度參數可以省略，預設值為1。
- x1, y1 是線段的起點座標，可以是任何座標，超出螢幕也沒關係。
- x2, y2 是線段的終點座標，同樣可以是任何座標，超出螢幕也沒關係。
- color 顏色值範圍 0-65535。顏色值規則請參考 **wb.cls()** 的說明。
- 彩色線條的範例：**wb.line(0, 0, 100, 100, wb.RED)**
- 彩色線條的另一個範例：**wb.line(120, 10, 20, 100, wb.GREEN, 3)**
- 我們可以試試一個在螢幕上畫出隨機線條的程式範例：

```
while True:
    x1 = wb.rand(100)
    y1 = wb.rand(100)
    x2 = wb.rand(100)
    y2 = wb.rand(100)
    color = wb.rand(65536)
    width = wb.rand(1,3)
    wb.line(x1, y1, x2, y2, color, width)
```

wb.circle(x, y, r, color [, width]) - 在 LCD 上畫出一個彩色的空心圓

- [width] 是線條的寬度，範圍是 1~16。這個寬度參數可以省略，預設值為1。
- x, y 是圓心座標，可以是任何座標，超出螢幕也沒關係。
- r 是圓圈的半徑，可以是任何數值，超出螢幕也沒關係。
- color 顏色值範圍 0-65535。顏色值規則請參考 **wb.cls()** 的說明。
- 彩色圓圈的範例：**wb.circle(30, 30, 20, wb.RED, 2)**
- 彩色圓圈的另一個範例：**wb.circle(0, 0, 50, wb.YELLOW)**

wb.pix(x, y, color) - 在 LCD 上畫出一個彩色的點

- x, y 是點的座標。
- color 顏色值範圍 0-65535。顏色值的定義規則請參考 **wb.cls()** 的說明。
- 範例：**wb.pix(10, 10, wb.WHITE)**

wb.win(x1, y1, x2, y2) - 設定一個可以填色的範圍(自動換行)

- x1, y1 是範圍的左上角座標。
- x2, y2 是範圍的右下角座標。
- 在這個方框範圍內，可以使用配套的 **wb.pushpix(color)** 填色指令，從左上角向右循序填入圖樣的每個顏色，遇到最右邊時會自動換下一行的左邊繼續填色。超過範圍後的 **wb.pushpix()**，就不能再顯示顏色了。
- **wb.win()** 的使用範例 - 在 (0,0) ~ (99,99) 的矩形範圍內隨意填色：

```
wb.win(0, 0, 99, 99)
for i in range(10000):
    wb.pushpix(wb.rand())
```

- 這個指令可以將網路上的圖片串流，設定正確的矩形範圍，循序下載並填入顏色以顯示圖片。

wb.pushpix(color) - 在一個範圍內自動填色

- color 顏色值範圍 0-65535。顏色值的定義規則請參考 **wb.cls()** 的說明。
- 填色是基於 **wb.win()** 定義的矩形範圍自動循序填色。從左上角向右循序填入圖樣的每個顏色，遇到最右邊時會自動換下一行的左邊繼續填色。超過範圍後的 **wb.pushpix()**，就不能再顯示顏色了。
- 具體的使用範例請參照前項 **wb.win()** 的配套指令。

wb.str(str, x, y[, font[, size]]) - 在 LCD 上顯示一個彩色的英數符字串

- str 是一個文字字串。（僅支持英文、數字與符號的 ASCII 字串。）
- x, y 是字串顯示在 LCD 螢幕上的左上角座標。
- font 是字型類別，可以是 1~5。可以省略，預設值為 1。
- size 是字型的放大倍率，可以是 1~8。可以省略，預設值為 1。
- 要注意的是字串的顏色設定方式，是由 **wb.colors(color1, color2)** 所定義的。color1 是字的顏色，color2 是背景的顏色。如果 color2 等於 color1，則背景就會是透空的。
- 範例：**wb.str("Hello, WiFiBoy!", 5, 5, 1, 3)**
- 範例：**wb.str("Hello, 12345", 55, 55)**

wb.colors(color1, color2) - 設定文字字串的顏色

- color1 是字的顏色，color2 是背景的顏色。
- 如果 color2 等於 color1，則文字顯示時的背景就會是透空的。
- color1, color2 的顏色值範圍 0-65535。顏色值的定義規則請參考 **wb.cls()** 的說明。

wb.img(x, y, w, h, buffer) - 將一個 bytearray 的圖像快速顯示出來

- x, y 是圖像要顯示在 LCD 螢幕上的左上角座標。
- (w, h) 是圖像要顯示的寬度與高度。
- buffer 是一個 bytearray。也就是一個個的點的颜色資料。注意：每個點都是 16bit 顏色，也就是需要兩個 bytes 來定義一個點的颜色。

```
# 製作一個10x10的亂數圖塊，共計100個點，需要200個bytes的顏色值
image = bytearray([wb.rand(256) for i in range(200)])

for i in range(100):
    wb.img(i, 10, 10, 10, image) # 向右移動顯示
    time.sleep(0.05)
    wb.box(i, 10, 10, 10, 0)
```

- 從網路或磁碟上取得一張圖片的原始點資料，可以用這個指令直接快速顯示出來。
- 如果是全螢幕大小的圖型資料文件，可以直接用 **wb.showbuf(buf)**, **wb.showjpg(file)** 或 **wb.showbmp(file)** 快速顯示。

wb.showbuf(buffer) - 將一個全螢幕的 bytearray 圖像快速顯示出來

- buffer 是一個 bytearray 的全螢幕圖像資料。

wb.showbmp(file) - 將一個全螢幕的 JPEG 圖像快速顯示出來

- file 是一個全螢幕的 jpeg 格式圖像檔案路徑。

wb.showjpg(file) - 將一個全螢幕的 16-bit BMP 格式圖像快速顯示出來

- file 是一個全螢幕的 BMP 格式圖像檔案路徑。

wb.rand([start, end] | [range]) - 產生一個隨機整數亂數（硬體產生）

- start, end 是整數亂數的範圍（包含 start 與 end）
- range 是整數亂數的數量（範圍是從 0 到 range-1）
- 如果省略參數，就直接產生一個 32bit 的整數亂數，範圍是 -2147483648 ~ 2147483647。
- 這個整數亂數是利用 WiFiBoy 硬體亂數產生器的數值，有較佳的亂度與產生效率。

wb.getkey() - 讀取按鍵的組合值

- 依據按鍵現況產生鍵值，規則如下：

```
A:1, B:2, R:4, L:8, D:16, U:32, Menu:64
```

- 取得的鍵值是每個按鍵是否按下去的鍵值總和。（按下去是鍵值，放開是0）
- 例如按下 A 鍵，讀取到的值是 1。按下 L 鍵，讀取到的值是 8。
- 如果同時按下 A 鍵與 B 鍵，則讀取到 3。因為 A=1, B=2，相加等於 3。
- 如果上下左右同時按下，讀取到的值就是 $4+8+16+32 = 60$ 。可以用這個程式試試：

```
while True:
    print(wb.getkey())
```

小龜繪圖指令集

小龜繪圖是經典的程式入門學習工具，透過小龜移動與旋轉的過程畫出圖像，可以學習基本的程式運算邏輯與迴圈的應用。

標準的小龜繪圖範例是一個「圓形排列的20個圓圈」，對初學者而言，這複雜的圖像看起來是非常不容易的程式任務。但只要經過一系列的引導練習，任何人都可以順利操控一隻小龜，運用迴圈技術，逐步畫出這個圖像，並能理解這個程式設計的原理。而更驚人的是，學習者接著就可以開始創作出許多自己發明的美妙規則圖像，非常有趣。

wb.ttreset() - 讓小龜回家（小龜開始）

- 小龜的座標回到螢幕中央 (80, 64)。
- 小龜面向角度 0 度（向上）。
- 小龜畫筆設為白色且放下，也就是移動小龜時會畫出白色的足跡。

wb.ttpos(x, y) - 移動小龜到某個座標（不產生足跡）

- x, y 是小龜的新座標。

wb.ttgo(step) - 小龜前進 step 步

- step 是移動的距離，單位是LCD的點距。可以是負值，也就是倒退。
- 注意：倒退時，小龜面向的角度維持不變，沒有轉向。

wb.ttrots(degree) - 小龜旋轉 degree 度

- 小龜原地順時針旋轉 degree 度(360度為一圈)。
- degree 可以是負值，也就是逆時針旋轉 degree 度。

wb.ttpenup(mode) - 小龜畫筆提起或放下

- mode 是 1: 提起 或 0: 放下。
- 小龜畫筆如果放下，前進或後退時就會留下軌跡。
- 小龜畫筆如果提起，前進或後退時就不會留下軌跡。

wb.ttcolor(color) - 小龜畫筆顏色

- color 的顏色值範圍 0-65535。顏色值的定義規則請參考 **wb.cls()** 的說明。

wb.ttwidth(width) - 小龜畫筆的粗細

- width 的範圍是 1~16。

小龜繪圖範例一：畫出一個正方形（四邊形，這是最基本的小龜圖像，請一定要理解）

```
wb.cls()
wb.ttreset()
for count in range(4):
    wb.ttgo(50)
    wb.ttrots(90)
```

小龜繪圖範例二：畫出一個正五邊形（不要看答案，想想看，畫得出來嗎？）


```
wb.cls()
wb.ttrreset()
for count in range(5):
    wb.ttgo(40)
    wb.ttrrot(72)
```

小龜繪圖範例三：畫出一個五角星星（不要看答案，想想看，畫得出來嗎？）

```
wb.cls()
wb.ttrreset()
for count in range(5):
    wb.ttgo(40)
    wb.ttrrot(144)
```

小龜繪圖範例四：畫出一個10邊形（不要看答案，想想看，畫得出來嗎？）

```
wb.cls()
wb.ttrreset()
for count in range(10):
    wb.ttgo(20)
    wb.ttrrot(36)
```

小龜繪圖範例五：畫出一個圓形（不要看答案，想想看，畫得出來嗎？）

```
wb.cls()
wb.ttrreset()
for count in range(30):
    wb.ttgo(8)
    wb.ttrrot(12)
```

小龜繪圖範例六：畫出三個旋轉120度的正方形（看一下這個雙層迴圈，能理解嗎？）

```
wb.cls()
wb.ttrreset()
for count2 in range(3):
    for count in range(4):
        wb.ttgo(40)
        wb.ttrrot(90)
    wb.ttrrot(120) # 注意這行程式，是向前退四格的
```

小龜繪圖範例七：畫出三個旋轉120度的星星（不看答案，想得出來嗎？）


```

wb.cls()
wb.ttrreset()
for count2 in range(3):
    for count in range(5):
        wb.ttgo(40)
        wb.ttrrot(144)
wb.ttrrot(120) # 注意這行程式，是向前退四格的

```

小龜繪圖範例八：畫出20個旋轉排列的圓圈（挑戰題，能做得出來嗎？）

```

wb.cls()
wb.ttrreset()
for count2 in range(20):
    for count in range(30):
        wb.ttgo(6)
        wb.ttrrot(12)
wb.ttrrot(18) # 注意這行程式，是向前退四格的

```

小龜繪圖範例九：畫出20個旋轉排列的隨機顏色圓圈

```

wb.cls()
wb.ttrreset()
for count2 in range(20):
    wb.ttcolor(wb.rand())
    for count in range(30):
        wb.ttgo(6)
        wb.ttrrot(12)
wb.ttrrot(18) # 注意這行程式，是向前退四格的

```

小龜繪圖範例十：猜猜看，這個程式會畫出什麼樣的圖案呢？

```

wb.cls()
wb.ttrreset()
wb.ttrpos(80,34); wb.ttrrot(-42); wb.ttcolor(wb.RED)
step = 5
for i in range(60):
    step += 1
    for j in range(4):
        wb.ttgo(step); wb.ttrrot(90)
    wb.ttrrot(3)
for i in range(60):
    step -= 1
    for j in range(4):
        wb.ttgo(step); wb.ttrrot(90)
    wb.ttrrot(3)

```

BLIT 遊戲快顯繪圖系統

BLIT 是一個非常強大好用的遊戲快顯系統。為了快速處理遊戲圖像顯示，BLIT 系統以搬移整塊圖元 (sprite) 到一塊記憶體組合成遊戲畫面，來加速螢幕的繪圖處理。

BLIT 系統的顯示原理，是在記憶體中預留一塊全螢幕 buffer，稱為 off-screen buffer。當用戶對這塊記憶體填好想要顯示的圖之後，快顯系統可以一次將這張圖送上 LCD 螢幕，減少繪圖期間塗塗改改的次數，可以大幅減少繪圖時間，也可以讓螢幕更新時不會有擦抹的閃爍問題。

在 WiFiBoy 的系統中，已經內建一個 8bit color 的遊戲圖像集(sprite map)，是一個 128×512 點的 8bit 顏色圖庫。用戶可以從這個圖像集，擷取一塊小圖（例如8×8的一個遊戲角色），以非常快的速度貼入 off-screen buffer。貼完數十個這類型的小圖像之後，可以直接以 wb.blit() 指令，查找調色盤（對應256種顏色），一次把圖快顯出來。

這個內建的圖像集，可以透過這個程式顯示出來：

```
wb.blitpal()
while True:
    for i in range(0,80):
        wb.blitbuf()
        wb.blitimg(i*16, 0,0,160,128)
        wb.blit()
        time.sleep(0.1)
```

當然，如果您有自己的遊戲圖像要顯示，也可以利用 wb.setusr() 製作一個自己的圖像集(sprite map)，這個時候就需要使用 blitusr() 來搬移圖像。

wb.blit() - 將 off-screen 圖像快顯出來

- 將 off-screen 記憶體的內容，查找調色盤，做出完整 16-bit 圖像，再送上 LCD 快顯。

wb.blitbuf([address, data]) - 清除或填寫 off-screen 記憶體

- address 是 off-screen 的資料位址，範圍是 0-20479
- data 是一個 0-255 的 8-bit 顏色值。（對應 16-bit 調色盤的顏色）
- 如果省略 adress, data，則自動將整個 off-screen buffer 全部清除為 0。

wb.blitimg(sprite, x, y) - 以8×8為單位搬移 8 x 8 的圖元到 off-screen 記憶體

- sprite 是以 8×8 為單位，從圖庫左上角向右數，到底換一行繼續數的圖元編號值。
- x, y 是搬到 off-screen 的目的地座標。
- wb.blitimg() 有四種呼叫方式，請繼續查看後面的其他呼叫方式。

wb.blitimg(sprite, x, y, n) - 以8×8為單位搬移 n x n 的圖元到 off-screen 記憶體

- sprite 是以 8×8 為單位，從圖庫左上角向右數，到底換一行繼續數的圖元編號值。

- x, y 是搬到 off-screen 的目的地座標。
- 這個搬移方式不以 8×8 為單位搬移，可以是任意的 n x n 圖元。
- wb.blitimg() 有四種呼叫方式，請查看前後的其他呼叫方式。

wb.blitimg(sprite, x, y, w, h) - 以8×8為單位搬移 w x h 的圖元到 off-screen 記憶體

- sprite 是以 8×8 為單位，從圖庫左上角向右數，到底換一行繼續數的圖元編號值。
- x, y 是搬到 off-screen 的目的地座標。
- 這個搬移方式不以 8×8 為單位搬移，是搬移任意的 w x h 圖元。
- wb.blitimg() 有四種呼叫方式，請查看前後的其他呼叫方式。

wb.blitimg(sx, sy, x, y, w, h) - 從任意起點搬移 w x h 的圖元到 off-screen 記憶體

- sx, sy 是從圖庫的 sx, sy 為左上角搬移一個 w x h 的方塊。
- x, y 是搬到 off-screen 的目的地座標。
- 這個搬移方式不以 8×8 為單位搬移，是搬移任意的 w x h 圖元。
- wb.blitimg() 有四種呼叫方式，請看前面的其他呼叫方式。

wb.blitusr(sprite, x, y) - 以8×8為單位搬移 8 x 8 的自定義圖元到 off-screen 記憶體

- sprite 是以 8×8 為單位，從圖庫左上角向右數，到底換一行繼續數的圖元編號值。
- x, y 是搬到 off-screen 的目的地座標。
- wb.blitusr() 與 wb.blitimg() 類似，只是圖庫為 wb.setusr() 自定義。

wb.blitusr(sprite, x, y, n) - 以8×8為單位搬移 n x n 的自定義圖元到 off-screen 記憶體

- sprite 是以 8×8 為單位，從圖庫左上角向右數，到底換一行繼續數的圖元編號值。
- x, y 是搬到 off-screen 的目的地座標。
- 這個搬移方式不以 8×8 為單位搬移，可以是任意的 n x n 圖元。
- wb.blitusr() 與 wb.blitimg() 類似，只是圖庫為 wb.setusr() 自定義。

wb.blitusr(sprite, x, y, w, h) - 以8×8為單位搬移 w x h 的自定義圖元到 off-screen 記憶體

- sprite 是以 8×8 為單位，從圖庫左上角向右數，到底換一行繼續數的圖元編號值。
- x, y 是搬到 off-screen 的目的地座標。
- 這個搬移方式不以 8×8 為單位搬移，是搬移任意的 w x h 圖元。
- wb.blitusr() 與 wb.blitimg() 類似，只是圖庫為 wb.setusr() 自定義。

wb.blitusr(sx, sy, x, y, w, h) - 從任意起點搬移 w x h 的自定義圖元到 off-screen 記憶體

- sx, sy 是從圖庫的 sx, sy 為左上角搬移一個 w x h 的方塊。
- x, y 是搬到 off-screen 的目的地座標。
- 這個搬移方式不以 8×8 為單位搬移，是搬移任意的 w x h 圖元。
- wb.blitusr() 與 wb.blitimg() 類似，只是圖庫為 wb.setusr() 自定義。

wb.setusr(buf, len) - 設定用戶自訂義圖庫

- 設定一塊 bytearray buf，長度 len 為自定義圖庫。

wb.blitpal([num, pal]) - 設定256個調色盤

- num 是 8-bit 顏色號碼。
- pal 是 16-bit 顏色值。
- 如果省略 num, pal，則自動設定為系統調色盤顏色值。

wb.blitstr(str, x, y) - 搬移內建圖庫字型區域的字串

- 將 str 字串內容，搬移到 off-screen 的 x, y 座標上。

wb.blitfree - 釋放 off-screen 的記憶體

- 只有在應用程式執行時需要的記憶體不足時，且用不到 off-screen 時，可透過釋放記憶體的方式，取得部分被佔用的記憶體。這個記憶體大小為 20 KBytes。
- 注意這個記憶體一旦釋放，就無法再要回來了。只有重新啟動系統 (RESET) 才能重新啟用 off-screen 記憶體。

BLIT 快顯系統的範例：

小蜜蜂程式（42行精簡版）

```
01 wb.blitpal() # 設定調色盤
02 star=[]; starspeed=[] # 預備星空特效
03 for i in range(0, 40): # 設定 40 個星星
04     star.append(wb.rand() % 20480) # 隨意產生一個座標點
05     starspeed.append((wb.rand() % 4)+1) # 隨意產生星興的速度(1-4)
06 x = 50; bt = btx = ct = dx = sc = 0; dir = 2; spn=[] # 一些遊戲初始值
07 for i in range(0,18): spn.append(1) # 設定 18 隻小蜜蜂
08 while True:
09     wb.blitbuf() # 清除 off-screen
10     for j in range(0,40):
11         wb.blitbuf(star[j],255) # 畫出 40 個星星 (顏色為255, 白色)
12         star[j] = star[j]+160*starspeed[j] # 調整星星位置 (依據星星速度)
```

```

13     if (star[j]>20480): star[j] = star[j] - 20480 # 如果超出螢幕，捲到上面
14 if wb.getkey() & 4 == 4: # 右鍵, x+2
15     x += 2
16     if x > 144: x = 144 # 到右邊緣就停住
17 if wb.getkey() & 8 == 8: # 左鍵, x-2
18     x -= 2
19     if x < 0: x = 0 # 到左邊緣就停住
20 if wb.getkey() & 1 == 1: # A鍵
21     if bt==0: bt = 112; btx = x+6 # 發射子彈
22 if wb.getkey() & 2 == 2: # B鍵
23     for i in range(0,18): spn[i]=1 # 小蜜蜂全部復活
24 if bt > 0: wb.blitimg(1, btx, bt); bt -= 2 # 子彈飛
25 ct += 1 # 設置一個動畫用的 counter
26 if (ct %4)==0: # counter 數 4 次就移動
27     dx += dir
28     if dx>64: dir=-2; dx=64
29     if dx<1: dir=2; dx=1
30 for j in range(32,97,48): wb.blitimg(176, dx+j-16, 12, 16) # 大魔鬼
31 for i in range(0,18): # 小蜜蜂處理
32     sx = dx+(i%6)*16; sy = (i//6)*16+30 # 移動
33     if (spn[i]==1): # 這隻小蜜蜂是活的嗎?
34         if ((ct+i)%40<20): wb.blitimg(144, sx, sy, 16) # 小蜜蜂動作1
35         else: wb.blitimg(146, sx, sy, 16) # 小蜜蜂動作2
36         if (bt-sy>0) and (bt-sy<16) and (btx-sx>0) and (btx-sx<13): # 擊中
37             spn[i]=20; bt = 0; sc+=100 # 加100分, 子彈消失, 小蜜蜂準備爆炸
38         elif (spn[i]>7): wb.blitimg(121, sx, sy, 16); spn[i]-=1 # 爆炸特效
39         elif (spn[i]>2): wb.blitimg(119, sx, sy, 16); spn[i]-=1 # 爆炸特效
40 wb.blitimg(115, x, 112, 16) # 顯示主角戰鬥機
41 wb.blitstr("SCORE %5s"%(str(sc)), 20, 0) # 顯示分數
42 wb.blit() # 將遊戲畫面顯示到 LCD

```