聯陽半導體
ITE Tech. Inc.

iTE SDK

LED 模組開發指南

V0.9

ITE TECH. INC.

修訂記錄

修訂日期	修訂說明	頁次
2014/10/01	初建版本 V0.9	

P1
P4

目錄

1. 前言

IT9079 的 LED 模組可提供不限 1 組的 LED pin 控制，每個 LED pin 都能各自獨立設定成 ON/OFF 或是閃爍狀態。除了可使用 IT9079 本身提供的 46 組 GPIO 當作 LED 的控制 PIN 之外，還可使用外接 I/O EXTENDER CHIP 控制 LED，但兩者不能混搭使用(LED 的控制 PIN 僅限全是 GPIO PIN 或是全是 I/O EXTENDER PIN)。

ITP driver 已經包裝 LED 的 I/O 硬體控制，提供各種設定 LED ON/OFF 或是 FLICKER 的 API，讓上層應用程式使用。使用者可以選定並啟動某個 GPIO pin 或是 I/O EXTENDER PIN。

1.1 編寫目的

介紹 LED 模組之功能, 說明如何使用 LED 各種 API 來完成功能。.

ITP driver 已經包裝 LED 的硬體設定，提供各種設定 LED 的 API 讓上層應用程式使用。使用者可以選定並啟動某些 GPIO pin 或是某些 I/O ENTENDER PIN 作為 LED 的控制 PIN

1.2 適用範圍

提供 LED pin 的選定、啟動、設定 ON/OFF、設定閃爍狀態以及閃爍頻率等功能。

1.3 適用人員

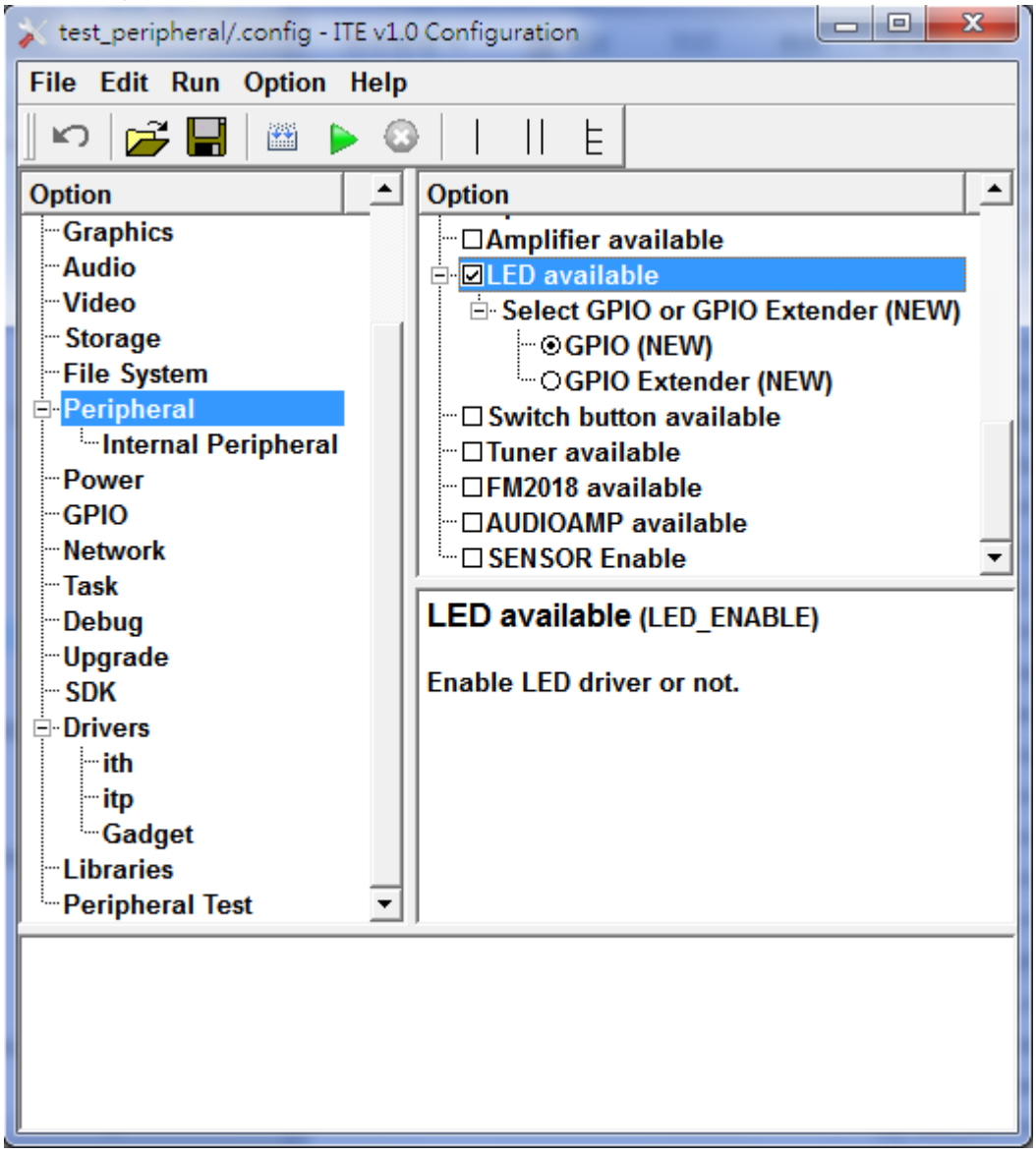
軟體應用程式開發者

2. LED模組介紹

LED 模組的原始碼放在 “sdk\driver\itp\itp_led.c”。

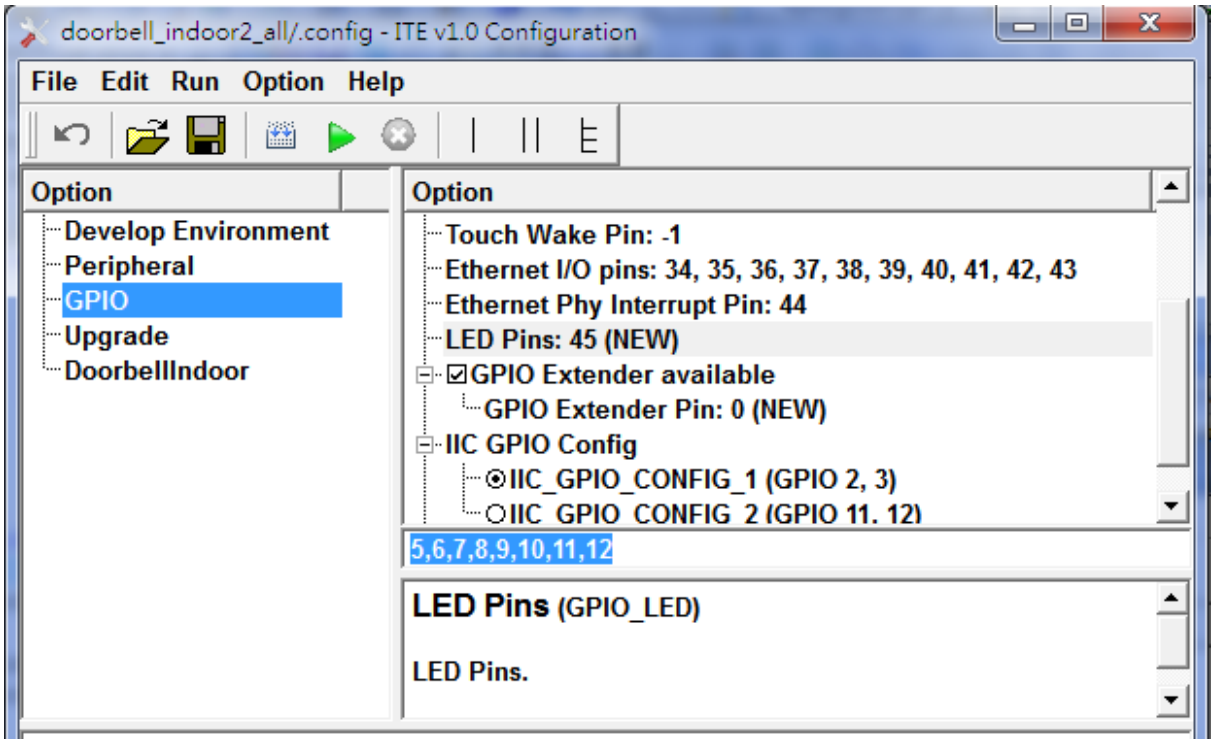
2.1 KCONFIG 設定

選用 LED 模組



LED available：啟動 LED 模組

Select GPIO or GPIO Extender：選擇使用 GPIO 或是 GPIO extender 控制 LED(默認選項是 GPIO)



LED pin：設定 LED 所使用的 GPIO pin，可選多組 GPIO。當「GPIO/IO extender」選擇 IO extender，則「LED pin(GPIO_LED)」變為 GPIO extender 的 pin number。使用者應注意 GPIO extender 的 IO pin 數目(例如 pcf8574 共有 8 個 IO pin)，以及 LED 是接在哪些 IO pin 上。

2.2 LED 模組的初始化

LED 模組的初始化相關程式，可參考 “sdk\driver\itp\itp_init_openrtos.c”或是“sdk\driver\itp\itp_init_win32.c”。

執行 LED 模組的初始化動作之前，必須先註冊 LED 模組到 ITP driver。其 API 如下

```
itpRegisterDevice(ITP_DEVICE_LED, &itpDeviceLed);
```

註冊完成之後，才能使用 ioctl()執行 LED 的初始化，其 API 如下：

ioctl(ITP_DEVICE_LED, ITP_IOCTL_INIT, NULL);

ioctl()： 參考“sdk\include\ite\itp.h” 中的“Device ioctl method”
ITP_DEVICE_LED：參考“sdk\include\ite\itp.h” 中的“device types”
ITP_IOCTL_INIT：參考“sdk\include\ite\itp.h” 中的“IOCTL definitions”

- **以上兩個函式已經含在 ITP driver 的初始化過程當中(參考 “sdk\driver\itp_init_openrtos.c”)，所以一般不必再執行此註冊函式。**

2.3 open/close LED 模組

在操作 LED 之前，必須先 OPEN LED DEVICE，OPEN 將會回傳此 LED 的 DEVICE ID

```
fd0 = open(":led:0", 0);
```

關於 device 的 open method 請參考 “sdk\include\ite\itp.h”中的

```
/**
 * Device open method.
 *
 * @param filename Device name.
 * @param oflag Type of operations allowed.
 * @param mode Permission mode.
 * @param info Device custom data.
 * @return Returns a file descriptor for the opened device. A return value of -1 indicates an error, in which case errno is set.
 */
int (*open)(const char* name, int flags, int mode, void* info);
```

此函式與標準 POSIX open()函數相容，使用方式可參考 POSIX 的說明文件。其中輸入的參數「const char* name」為 “:led:0”, “:led:1”, “:led:2”, …，視 LED 的 GPIO_LED 所定義的數目而定。GPIO_LED 的 GPIO PIN 順序即為“:led:0”, “:led:1”, “:led:2”的順序。應用方式如下所示：

```
Led0 = open(“:led:0”, 0);
Led1 = open(“:led:1”, 0);
Led2 = open(“:led:2”, 0);
```

同樣，關於 device 的 close method 請參考 “sdk\include\ite\itp.h”中的

```
/**
 * Device close method
 *
 * @param file File descriptor referring to the open device.
 * @param info Device custom data.
 * @return Returns 0 if the device was successfully closed. A return value of -1 indicates an error.
 */
int (*close)(int file, void* info);
```

此函式與標準 POSIX close()函數相容，使用方式可參考 POSIX 的說明文件。其中輸入的參數「const char* name」為 device open method 所回傳的 file descriptor。應用方式如下所示：

```
close(Led0);
close(Led1);
close(Led2);
```

LED 模組可經由以上的 open/close method，個別開啟/關閉所指定的 LED。

2.4 LED 模組的 ON/OFF 以及 flicker

關於 LED 模組的 ON/OFF 以及 flicker 皆由 ioctl()來達成。此函式與標準 POSIX ioctl()函數相容，詳細說明可參考 POSIX 的說明文件以及“sdk\include\ite\itp.h”與 LED 相關的定義。

LED ON：使用 ioctl method 配合「ITP_IOCTL_ON」的定義，可個別將 LED 設定成 ON 的狀態。關於 ioctl 的定義可參考 “sdk\include\ite\itp.h” 。調用方式如下所示：

```
ioctl(led0, ITP_IOCTL_ON, NULL);
```

LED OFF：使用 ioctl method 配合「ITP_IOCTL_OFF」的定義，可個別將 LED 設定成 OFF 的狀態。關於 ioctl 的定義可參考 “sdk\include\ite\itp.h”。 調用方式如下所示：

```
ioctl(led0, ITP_IOCTL_OFF, NULL);
```

LED FLICKER：使用 ioctl method 配合「ITP_IOCTL_FLICKER」的定義，可個別將 LED 設定成閃爍的狀態。

關於 ioctl 的定義可參考 “sdk\include\ite\itp.h” ，而第三個輸入參數是設定閃爍的週期，單位是 millisecond 。
調用方式如下所示：

ioctl(led0, ITP_IOCTL_FLICKER, (void*)33);

3. LED模組的範例

LED 的範例程式可參考以下程式，或是參考 “project\test_peripheral\test_led.c”。

```
#include <sys/ioctl.h>
#include <stdio.h>
#include <unistd.h>
#include "ite/itp.h"

int main(void* arg)
{
    ItpInit();

    int led0 = open(":led:0", 0);
    int led1 = open(":led:1", 0);
    int led2 = open(":led:2", 0);

    for (;;)
    {
        ioctl(led0, ITP_IOCTL_ON, NULL);
        ioctl(led1, ITP_IOCTL_OFF, NULL);
        ioctl(led2, ITP_IOCTL_FLICKER, (void*)50);
        sleep(1);

        ioctl(led1, ITP_IOCTL_ON, NULL);
        ioctl(led2, ITP_IOCTL_OFF, NULL);
        ioctl(led0, ITP_IOCTL_FLICKER, (void*)50);
        sleep(1);

        ioctl(led2, ITP_IOCTL_ON, NULL);
        ioctl(led0, ITP_IOCTL_OFF, NULL);
        ioctl(led1, ITP_IOCTL_FLICKER, (void*)50);
        sleep(1);
    }
    return NULL;
}
```