國立交通大學

資訊學院　資訊學程

碩 士 論 文

多 個 螢 幕 媒 體 設 備 之 實 作

The DPF Multi-Panel Controller

研 究 生：廖家麟

指導教授：林一平　教授

羅悅臣　先生

中 華 民 國 一 百 年 一 月

多 個 螢 幕 媒 體 設 備 之 實 作

The DPF Multi-Panel Controller

研 究 生：廖家麟　　Student：Chia-Lin Liao

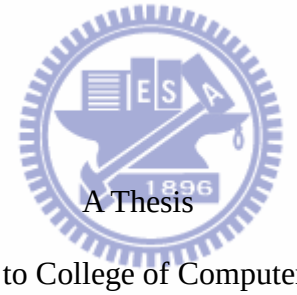
指導教授：林一平　　Advisor：Yi-Bing Lin

羅悅臣　　Yue-Chen Luo

國 立 交 通 大 學

資訊學院　資訊學程

碩 士 論 文



A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of science

in

Computer Science

January 2011

Hsinchu, Taiwan, Republic of China

多個螢幕媒體設備之實作

學生：廖家麟　　指導教授：林一平　教授

國立交通大學

資訊學院　資訊學程碩士班

中文摘要

目前市面數位相框的功能設計為單一畫面顯示照片。欲讓使用者同時欣賞豐富的色彩及影像，可將多台數位相框彼此串接，以達到多個畫面顯示之功能。

本研究以數位相框為例實作多個螢幕串接之媒體設備。我們透過 USB 連接線將多台數位相框彼此連接起來及實作 USB 主機端(USB Host) 及 USB 裝置端(USB Device) 兩種角色的程式，並以自訂的協定來互相彼此溝通，以達到照片可直接在多個數位相框螢幕上顯示之目的。本實作基於整體成本考量下，必須選用低成本的相關硬體，如 CPU(MIPS Clock 96MHz)、Memory(Clock 96MHz；8MB SDRAM)、JPEG Decoder(Clock 48MHz)。然而另外考慮的限制是，近日數位相機的高解析度技術不斷提昇，影像檔案越來越大，效能表現較有限制。

在上述相關限制下，若要考量提升整體效率，我們必須在軟體上做相對的調整。本實作將兩台數位相框串聯，以第一台數位相框為主，因 JPEG 檔案解壓縮成原始影像資料的時間較為耗時，故將 JPEG 檔案解壓縮成螢幕尺寸之原始影像資料作為傳輸檔案，以加快傳輸的速度，並可將螢幕尺寸之原始影像資料直接顯示在第二台數位相框螢幕上，不需經過相關解壓縮過程，以增加顯示的效率。

The DPF Multi-Panel Controller

Student：Chia-Lin Liao Advisor：Dr. Yi-Bing Lin

Mr. Yue-Chen Luo

Degree Program of Computer Science

National Chiao Tung University

ABSTRACT

Current market design of DPF features a single screen to display photos. However, to allow users to simultaneously enjoy the rich colors and images, numerous DPFs are cascaded to achieve the multiple screen display function.

In this research, DPF is taken as an example and demonstrates the DPF multi-panel controller. Many DPFs are linked together through USB cable. DPF plays the roles as both USB Host and USB Device. The researcher controls the software of USB Host and USB Device, and makes them communicate through vendor protocols, so that the pictures can be displayed directly in a number of DPFs' screen. For the overall cost of the implementation considerations, inexpensive hardware is utilized, such as CPU (MIPS Clock 96MHz), Memory (Clock 96 MHz, 8MB SDRAM), JPEG Decoder (Clock 48 MHz). In addition, another limitation is that, recently, with the improvement of the DPF high-resolution, the image sizes are bigger and bigger, which leads to the limited performance.

Under the related limitation stated above, in order to improve the overall performance, the software must be adjusted relatively. In the experiment, two DPFs are cascaded with the first DPF as the base. It is time-consuming to decode JPEG files into the raw data. Hence, JPEG files are decoded into the scaled-down raw data as the transmission files to accelerate the transfer rate. This scaled-down raw data is shown directly on the second DPF screen without any related decoding process to increase the performance.

誌謝

首先非常誠摯地感謝林一平教授與羅悅臣先生。由於二位的細心指導與專業建議，使我得以順利圓滿完成此篇碩士論文。在林教授嚴格殷切的要求與再三修正潤飾之下，讓我學習到論文寫作的嚴謹與做學問研究的方法；在羅先生的指導協助中，讓我獲得許多專業知識與實作技巧。此外，口試委員楊舜仁教授在口試中給與指正與建議，使得論文能更加完備，併此致謝。

同時也要感謝皇甫建君學長、邱鈺真同學及實驗室同伴們給予相當協助與建議。

最後感謝我的家人與朋友們在論文寫作期間給予莫大的鼓勵與支持。

目錄

圖目錄

表目錄

第一章 緒論

近年來數位影像產品已逐漸普及於日常生活，不但可以增加人們在親子互動、友誼傳遞等感情交流，同時使彼此之間的關係更加密切。目前市面上的數位相框產品僅單純提供照片欣賞。

要擴大數位相框的多樣應用，必須增加其通訊功能，如能透過傳輸的媒介如通用序列匯流排(Universal Serial Bus；USB[1])、WiFi、藍牙(Bluetooth)等，使用螢幕媒體設備如數位相框(Digital Photo Frame;DPF)、數位相機(Digital Still Camera;DSC)、個人數位助理(Personal Digital Assistant;PDA)等設備傳輸照片，讓影像直接傳至另一個螢幕媒體設備上，達到多個螢幕媒體設備的連接(如圖 1.1)，更進一步可做相關特效之處理。如此透過 USB 連接線將多個數位相框連結起來，後續自動將照片由左至右方向依序傳輸至下一台數位相框，以達到影像動態傳輸的效果(如圖 1.2)。

圖 1.1 多個螢幕媒體設備連接(經由 USB Cable)

圖 1.2 多個螢幕媒體設備連接之特效

以下就數位相框之軟硬體設備架構分別說明如下。

1.1 數位相框之硬體架構

數位相框基本的硬體設備架構包括六個元件，逐一說明如下(如圖 1.3)。

(1) 直接記憶體存取控制器(Direct Memory Access Controller;DMA Controller；如圖 1.3(a))是協助記憶體資料的搬移，可節省中央處理器(Central Processing Unit；CPU)資源。

(2) JPEG 解碼器(JPEG Decoder；如圖 1.3(b))負責將 JPEG 檔案解壓縮成原始影像資料(Raw data)。

(3) 影像顯示單元控制器(Image Display Unit Controller；IDU Controller；如圖 1.3(c))負責將影像顯示至液晶顯示器(Liquid Crystal Display；LCD)螢幕上。

(4) 通用序列匯排流控制器(Universal Serial Bus Controller；USB Controller；如圖 1.3(d))負責不同裝置之間的 USB 資料傳輸。數位相框可扮演兩種角色，一種為 USB 主機端(USB Host)，主動式負責資料傳輸控制相關流程，如數位相框連接 USB Storage Device(如隨身碟)，便可主動進行資料存取；另一種為 USB 裝置端(USB Device)，被動式聽從 USB Host 控制，如數位相框連接 PC，並由 PC 端主動存取資料。

(5) 記憶體(Memory；如圖 1.3(e))用來存放主程式，主程式可用來控制整體運作流程。

(6) 尺寸縮放控制器(Scaler；如圖 1.3(f))負責可將原始影像資料以寬高非等比例縮減大小。

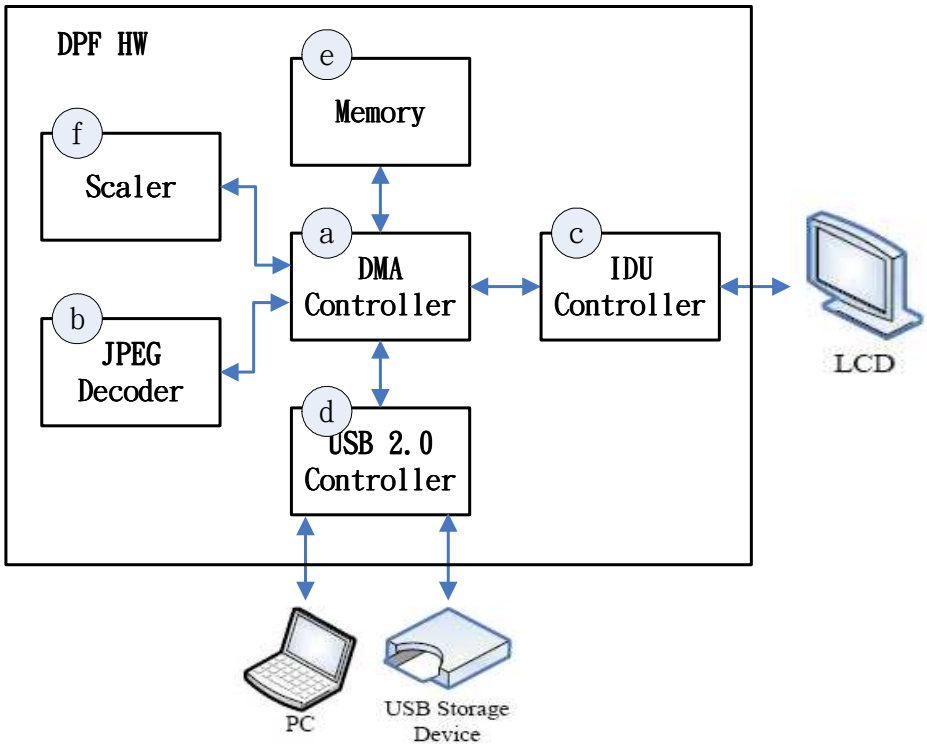


圖 1.3　數位相框之硬體架構

如欲將 JPEG 檔案(File-Size:14 MB / Resolution : 4492 x 3328)透過 USB 介面顯示至 LCD 上，該流程則逐一說明如下。首先向系統要一塊 1MB 大小的記憶體(如圖 1.3(e))後，使用 USB2.0 Controller(如圖 1.3(d))透過 DMA Controller (如圖 1.3(a))將 JPEG 圖檔檔案依序多次搬移至記憶體。每當讀取 1MB 記憶體完成時，JPEG Decoder(如圖 1.3(b))會使用 DMA 方式，將該記憶體上的 JPEG 檔案搬移至 JPEG Decoder 內，再轉換為原始影像資料(Raw Data)，且該 JPEG Decoder 僅支援將解析度依 1/2 或 1/4 等比降低，故依該 LCD 解析(LCD Resolution : 480 x 234)選擇 1/4 等比降低解析(Resolution : 1123 x 832)。每當 JPEG Decoder 解壓縮完 1MB 資料後，也會使用 DMA 方式依序多次搬移至另一塊記憶體約 2MB 記憶體內(須大於 1123 x 832 x 2 Bytes;如圖 1.3(e))。待 JPEG Decoder 全部解完後，會得到一份原始影像資料，但並未符合該 LCD 解析，故需使用 Scaler(如圖 1.3(f))來將原始影像資料依需求縮放至 LCD 解析，所以也使用 DMA 方式將原始影像資料搬至 Scaler 內作縮放後，再透過 DMA

搬至 IDU(如圖 1.3(c))，即可顯示至 LCD 上，則完成將影像顯示至 LCD 上。

1.2 數位相框之軟體架構

數位相框基本的軟體架構如圖 1.4 所示，以兩台數位相框為例，其中一台數位相框為 USB Host[1][2]角色，另一台數位相框則為 USB Device[1]角色。首先介紹每層架構部分，包括功能層(Function Layer)、裝置層(USB Device Layer)及裝置匯流介面層(USB Bus Interface Layer)。

- 功能層(Function Layer;如圖 1.4(a))依據 USB Device 的類別，則 USB Host 及 USB Device 構成彼此做相對的功能處理的架構層。
- 裝置層(USB Device Layer；如圖 1.4(b))負責將相關資料與 USB 格式的資料之間做轉換的架構層。
- 裝置匯流介面層(USB Bus Interface Layer；如圖 1.4(c))實際控制 USB 訊號傳輸的架構層。

其次介紹 USB Host 部分，包括 USB 用戶軟體(Client Software)、USB 系統軟體(USB System Software)及 USB 主機端控制器(USB Host Controller)。

- USB 用戶軟體(Client Software;如圖 1.4(d))負責處理不同 USB 類別的應用。USB 類別可分為人機介面類別(Human Interface Device; HID)及儲存類別(Mass-Storage Device Class;MSDC)等，一般常見的 HID[3]類別為滑鼠、鍵盤等設備；Mass-Storage[4][5][6][7]類別如 USB 隨身碟、USB 式硬碟等設備。
- USB 系統軟體(USB System Software；如圖 1.4(e))負責將上層 USB 用戶軟體傳遞的資料轉換為 USB 格式的資料，並控制 USB 主機端控制器(USB Host Controller) 將相關資料傳遞至 USB Device。

- USB 主機端控制器(USB Host Controller；如圖 1.4(f))負責 USB Host 端實體資料傳輸。

最後介紹 USB Device 部分，包括功能模組(Function)、USB 邏輯裝置(USB Logical Device)及 USB 匯流介面(USB Bus Interface)，逐一說明如下。

- 功能模組(Function；如圖 1.4(i))負責處理 USB Device 端傳送及接收後的資料，並依據不同 USB 類別進行相對應的處理。
- USB 邏輯裝置(USB Logical Device；如圖 1.4(h))負責傳送及接收 USB Device 端的資料，其功能相當於 USB Host 端的 USB 系統軟體(如圖 1.4(e))。
- USB 匯流介面(USB Bus Interface；如圖 1.4(g))，負責 USB Device 端實體資料傳輸，其功能相當於 USB Host 端的 USB 主機控制器(如圖 1.4(f))。

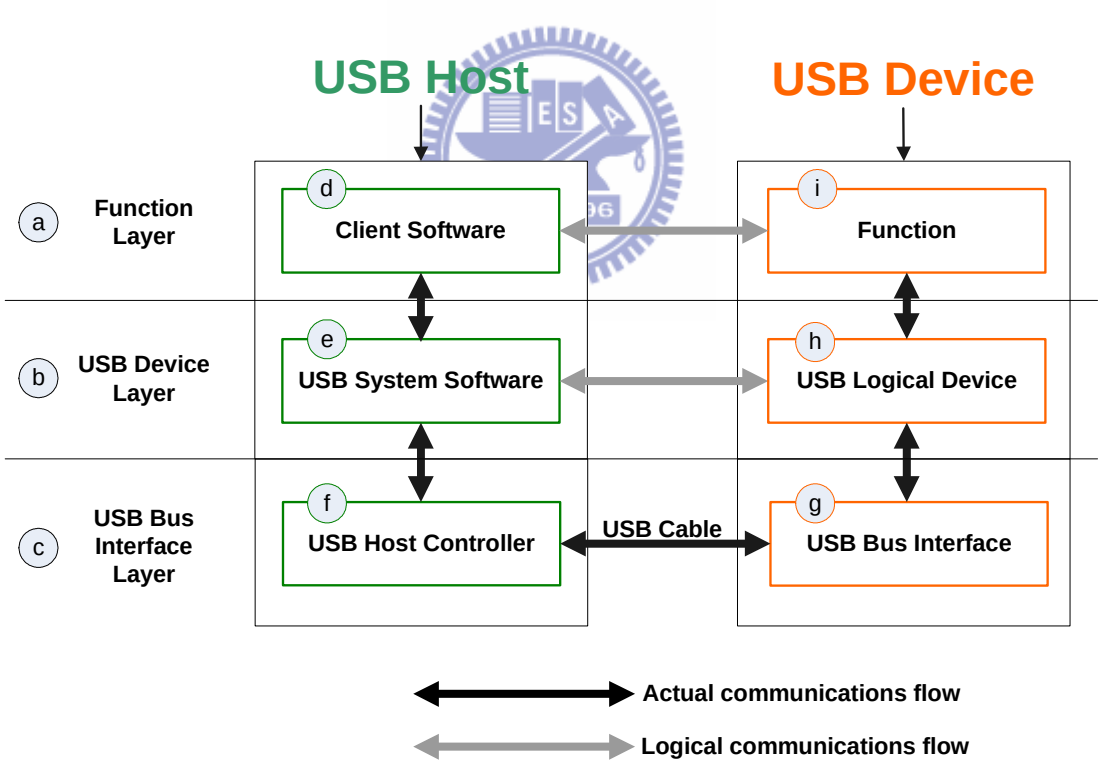


圖 1.4 數位相框之軟體架構

1.3 研究動機與目的

根據第 1.1 節的架構，目前市面數位相框皆為單一螢幕顯示照片。如能透過相關傳輸設備(如第 1.2 節描述之 USB)，讓多台數位相框彼此連接，可以達到多個畫面顯示之功能。

本實作基於整體成本考量下，必須選用低成本的相關硬體，如 CPU(MIPS Clock 96MHz)、Memory(Clock 96MHz；8MB SDRAM)、JPEG Decoder(Clock 48MHz)，然而另外必須考慮的限制是，近日數位相機的高解析度技術不斷提昇，影像檔案越來越大，效能表現較有限制。

在上述相關限制下，若要考量提升整體效率，必須在軟體上做相對的調整。本實作將兩台數位相框串聯，以第一台數位相框為主，將 JPEG 檔案解壓縮成原始影像資料作為傳輸檔案，隨後將之直接顯示在第二台數位相框螢幕上，不需經過相關解壓縮過程，可增加顯示的效率。本論文實作 USB 主機端(USB Host) 及 USB 裝置端(USB Device) 兩種角色的程式，並以自訂的協定來互相彼此溝通，以達到照片動態傳輸之功能。最後實際測量該實作效能。

1.4 論文章節說明

本論文後續的章節架構說明如下。第二章為 USB 介紹，著重在敘述一般基本 USB 的知識，以讓讀者能了解 USB 的相關協定。第三章說明數位相框的傳輸機制，與此機制中各程式的名稱與功用。第四章再詳細說明多個螢幕媒體設備傳輸照片之實作方式。第五章進行多個螢幕媒體設備之效能評估。第六章為綜合本論文的結論與未來展望。

第二章 USB 介紹

USB 為通用串列匯流排(Universal Serial Bus) 是一種標準的連接介面，其優點在於可使電腦同時將多種不同的週邊設備加以連接，最高可支援 127 個週邊設備，並且具有能提供高速傳輸速度的優勢。在 USB 2.0 的規範[1]中，USB 低速(Low Speed；LS)裝置的傳輸速度可達到 1.5Mbps，USB 全速(Full Speed；FS)裝置的傳輸速度可高達 12Mbps 及 USB 高速(High Speed；HS)裝置的傳輸速度更高達 480Mbps。

2.1 USB 傳輸模式

USB 傳輸模式[1][8][9]總共為四種，包括控制型傳輸(Control Transfers)、中斷型資料傳輸(Interrupt Data Transfers)、巨量型資料傳輸(Bulk Data Transfers)及等時型資料傳輸(Isochronous Data Transfers)，逐一說明如下。

- 控制型傳輸(Control Transfers)提供 USB Host 傳送命令給 USB Device 及查詢 USB Device 狀態…等動作，所有 USB Host 及 USB Device 皆必須具備控制型傳輸模式。當一個 USB Device 連接上 USB Host 時，USB Host 會透過控制型傳輸模式得知 USB Device 設備資訊，讓 USB Host 能識別 USB Device 裝置所支援的傳輸模式，以便 USB Host 後續能正確與 USB Device 溝通及傳輸資料。

- 中斷型資料傳輸(Interrupt Data Transfers)提供快速傳送正確的資料。以 USB 高速裝置為例，每次傳輸資料量最大可高達 1024 Bytes，但如需達到 125 微秒高速且正確的資料傳輸，則 USB 2.0 的規範建議每次傳輸量約為 64 Bytes。目前常見之中斷型資料傳輸為低速設備，例如 USB 滑鼠及 USB 鍵盤…等裝置的資料傳輸。在 USB 規格中，中斷型資料傳輸以週期性輪詢(Polling)的方式完成，亦即

USB Host 以週期性輪詢的方式詢問 USB Device，以便得知 USB Device 是否需要傳送資料給 USB Host。當傳輸過程中，因錯誤而發生傳送失敗的情況，可以在下一次輪詢期間重新再傳送一次。

- 巨量型資料傳輸(Bulk Data Transfers)提供大量傳送正確的資料。以 USB 高速裝置為例，每次傳輸資料量最大可高達 512 Bytes，但傳輸速度則以 USB Device 能正確地接收資料的時間為依據，例如印表機或掃描器…等裝置的資料傳輸。巨量型資料傳輸模式下資料必須準確地加以傳輸，所以對於資料在傳輸速度上並無時效限制。

- 等時型資料傳輸(Isochronous Data Transfers)提供快速傳送資料。以 USB 高速裝置為例，每次傳輸資料量最大可高達 1024 Bytes，並可達到 125 微秒高速傳輸，例如 USB 攝影機或 USB 耳機…等多媒體設備的資料傳輸。當一個 USB Device 連接上 USB Host 時，彼此透過控制型傳輸模式，互相約定彼此相同的傳輸頻寬，以確保發送與接收的速度能互相配合。本傳輸方式中，容許資料在傳輸上的遺失，並且不會進行重送。

2.2 USB 描述元

當一個新的 USB Device 經由 USB Cable 連接 USB Host 時，USB Host 必須透過 USB 描述元(Descriptor) 來了解這一個新連接 USB Device 的基本特性，並且尋找符合這個 USB Device 的驅動程式。USB 描述元包括裝置描述元[1](Device Descriptor)、組態描述元[1] (Configuration Descriptor)、介面描述元[1] (Interface Descriptor)、端點描述元[1] (Endpoint Descriptor)、字串描述元[1] (String Descriptor)，逐一說明如下。

- 裝置描述元(Device Descriptor)讓 USB Host 可以透過裝置描述元得到 USB Device 的特性，包括這一個裝置描述元長度、裝置描述元種類、所依據的 USB 規範版本、裝置類別資訊、裝置通訊協定、控制型端點的封包最大傳輸量、製造商代碼(VID)、產品代碼(PID)、裝置韌體版本、製造商字串索引、產品字串索引、序號字串索引與組態描述元總數。其格式列表說明如下(如表 2.1)。

表 2.1　裝置描述元(Device Descriptor)

欄位	大小(BYTE)	說明
bLength	1	裝置描述元長度
bDescriptorType	1	裝置描述元種類
bcdUSB	2	USB 規範版本
bDeviceClass	1	裝置類別資訊
bDeviceSubClass	1	
bDeviceProtocol	1	裝置通訊協定
bMaxPacketSize0	1	控制型端點的封包最大傳輸量
idVendor	2	製造商代碼(VID)
idProduct	2	產品代碼(PID)
bcdDevice	2	裝置韌體版本
iManufacturer	1	製造商字串索引
iProduct	1	產品字串索引
iSerialNumber	1	序號字串索引
bNumConfigurations	1	組態描述元總數

- 組態描述元(Configuration Descriptor)包含一至多組的組態描述元，不同於裝置描述元限定只能有一組。包括這個組態描述元長度、組態描述元種類、總長度(含組態描述元、介面描述元及端點描述元)、介面描述元總數、組態描述元值、組態描述元索引、組態描述元屬性(自我供電功能、遠端喚醒功能等)與設備所需最大電流量。其格式列表說明如下(如表 2.2)。

表 2.2　組態描述元(Configuration Descriptor)

欄位	大小(BYTE)	說明
bLength	1	組態描述元長度
bDescriptorType	1	組態描述元種類
wTotalLength	2	總長度(含組態描述元、介面描述元及端點描述元)
bNumInterface	1	介面描述元總數
bConfigurationValue	1	組態描述元值
iConfiguration	1	組態描述元字串索引
bmAttributes	1	組態描述元屬性
bMaxPower	1	設備所需最大的電流量

- 介面描述元(Interface Descriptor)與組態描述元相同可以包含一至多組的介面描述元。包括這個介面描述元長度、介面描述元種類、介面描述元索引、替代設定、端點描述元總數、介面類別資訊、介面通訊協定與介面描述元字串索引。其格式列表說明如下(如表 2.3)。

表 2.3　介面描述元(Interface Descriptor)

欄位	大小(BYTE)	說明
bLength	1	介面描述元長度
bDescriptorType	1	介面描述元種類
bInterfaceNumber	1	介面描述元索引
bAlternateSetting	1	替代設定
bNumEndpoints	1	端點描述元總數
bInterfaceClass	1	介面類別資訊
bInterfaceSubClass	1	
bInterfaceProtocol	1	介面通訊協定
iInterface	1	介面描述元字串索引

- 端點描述元(Endpoint Descriptor)包括這個端點描述元長度、端點描述元種類、端點描述元位置及方向(輸入/輸出)、端點描述元傳輸模式(控制、中斷、巨量、等時)、端點描述元最大傳輸量(單位:BYTE)與輪詢間隔。其格式列表說明如下(如表 2.4)。

表 2.4　端點描述元(Endpoint Descriptor)

欄位	大小(BYTE)	說明
bLength	1	端點描述元長度
bDescriptorType	1	端點描述元種類
bEndpointAddress	1	端點描述元位置及方向
bmAttributes	1	端點描述元傳輸模式
wMaxPacketSize	2	端點描述元最大傳輸量
bInterval	1	輪詢間隔

- 字串描述元(String Descriptor)提供人們可看懂的文字，在 USB 的描述元中此描述元是可有可無的，並不一定要提供給 USB Host，如果沒有提供字串描述元的話，則必須要將其他的描述元中的字串索引值設定成 0，代表並沒有字串可以提供使用。包括這個字串描述元長度、字串描述元種類與字串描述元字串內容。其格式列表說明如下(如表 2.5)。

表 2.5　字串描述元(String Descriptor)

欄位	大小(BYTE)	說明
bLength	1	字串描述元長度
bDescriptorType	1	字串描述元種類
Content	N	字串描述元字串內容

第三章 數位相框傳輸介紹

本章以數位相框為例，介紹 USB Host 及 USB Device 之間的傳輸機制及軟體模組，並說明其傳輸方法。

3.1 數位相框傳輸機制

本節考慮兩台串聯的數位相框之照片傳輸。其中一台數位相框為 USB Host 角色，另一台為 USB Device 角色，兩台數位相框間之傳輸架構[1]如圖 3.1 所示。首先介紹 USB Host 部分，包括 USB 用戶軟體(Client Software)、USB 主機端驅動程式(USB Host Driver)、USB 主機端控制器驅動程式(USB Host Controller Driver)、USB 主機端控制器(USB Host Controller)及序列傳輸介面(SIE)，逐一說明如下。

- USB 用戶軟體(Client Software;如圖 3.1(a))負責處理不同 USB 類別的應用。USB 類別可分為人機介面類別(Human Interface Device; HID)及儲存類別(Mass-Storage)等，一般常見的 HID 類別為滑鼠、鍵盤等設備；Mass-Storage 類別如 USB 隨身碟、USB 式硬碟等設備。因一個 USB Device 設備會包含多個 USB 介面(USB Interface x；如圖 3.1(h))，所以 USB 用戶軟體必須知道每一個 USB 介面的類別及資訊，以便能使用恰當的 USB 介面與 USB Device 溝通及傳輸。
- USB 主機端驅動程式(USB Host Driver；如圖 3.1(b))負責將上層 USB 用戶軟體資料轉換為 USB 類別格式的資料，並作為 Client Software 及 USB Host Controller Driver 之間的介面。
- USB 主機端控制器驅動程式(USB Host Controller Driver；如圖 3.1(c))負責提供存取 USB 主機端控制器的介面，讓 USB Host Driver 透過該介面控制 USB 硬體，以便傳送及接收 USB 封包資料。
- USB 主機端控制器(USB Host Controller；如圖 3.1(d))負責控制 USB 主機

端實體資料的傳輸。

- 序列傳輸介面(Serial Interface Engine；SIE；如圖 3.1(e))負責實際序列訊號傳輸的介面。

USB Device 包括功能介面(Interface)、端點位置(Endpoint)及序列傳輸介面(SIE)，逐一說明如下。

- 功能介面(Interface；如圖 3.1(h))負責處理各種不同 USB 介面編號的資料應用傳輸。因一個 USB Device 設備會包含多個 USB 介面，所以 USB Device 端必須詳細說明該介面的資訊，以便讓 USB Host 端能使用恰當的 USB 介面與 USB Device 溝通及傳輸。
- 端點位置(Endpoint；如圖 3.1(g))負責處理各種不同端點位置的資料傳輸。例如 USB Host 初期識別 USB Device 的 USB 類別，係透過控制型端點(Endpoint Zero)彼此溝通，以得知 USB Device 的類別及端點位置編號等資訊。後續 USB Host 會依據 USB Device 類別，使用其端點位置編號與 USB Device 溝通及傳輸資料。
- 序列傳輸介面(Serial Interface Engine；SIE；如圖 3.1(f))負責實際序列訊號傳輸的介面。其功能與 USB Host 端的序列傳輸介面(如圖 3.1(e))相同。

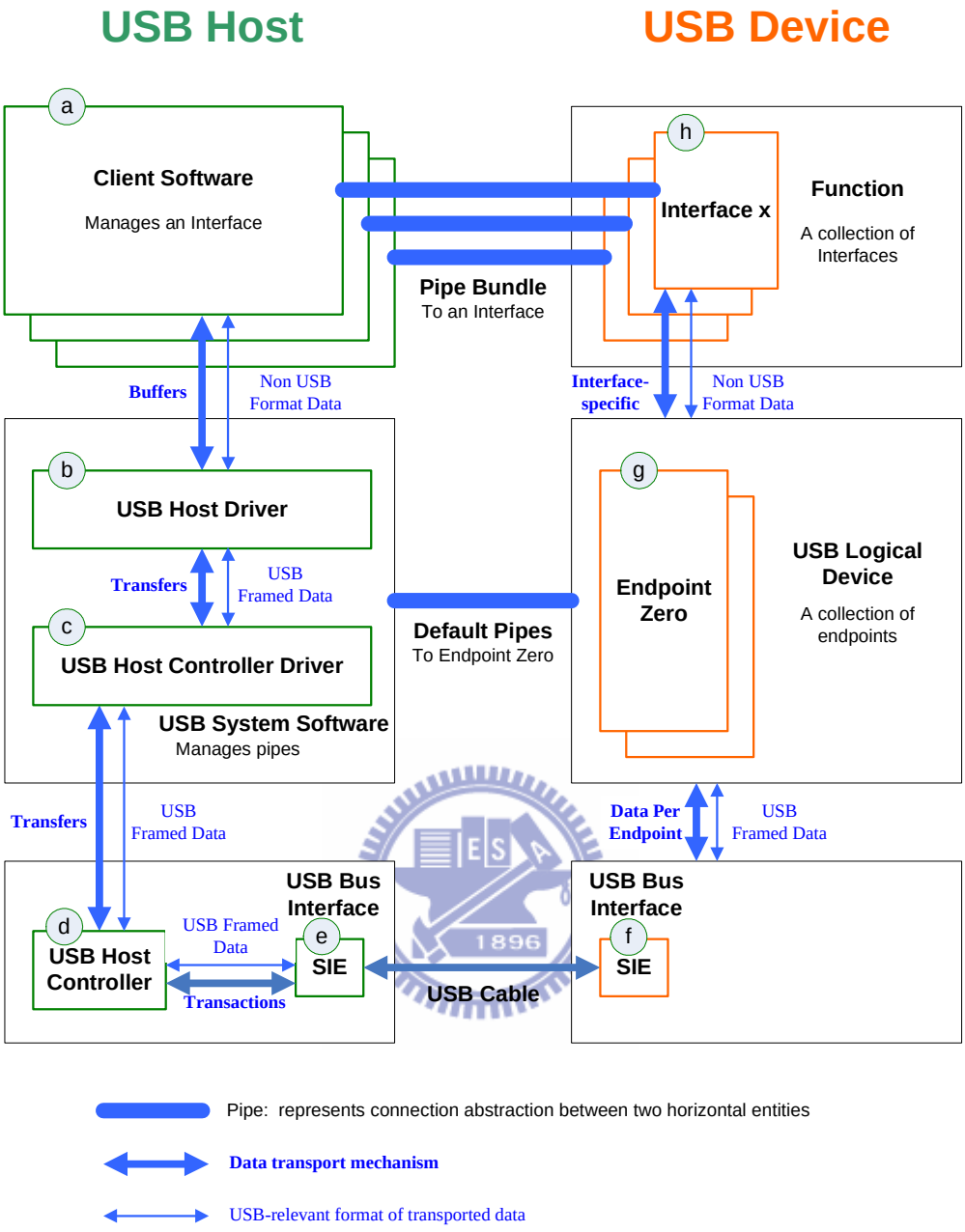


圖 3.1　數位相框之傳輸架構

3.2 數位相框運作流程

數位相框運作說明 USB Host 如何傳輸數據到 USB Device，其步驟分成三個部分：裝置描述元傳輸流程、原始影像資料傳輸流程及變更畫面命令流程，詳述如下。

3.2.1 取得裝置描述元流程

圖 3.2 繪出取得裝置描述元(Get Descriptor)流程，說明 USB Host 如何使用 Get Descriptor 方式，得知 USB Device 的類別，以便後續 USB Host 及 USB Device 以該類別的命令進行彼此溝通及控制。其步驟逐一說明如下。

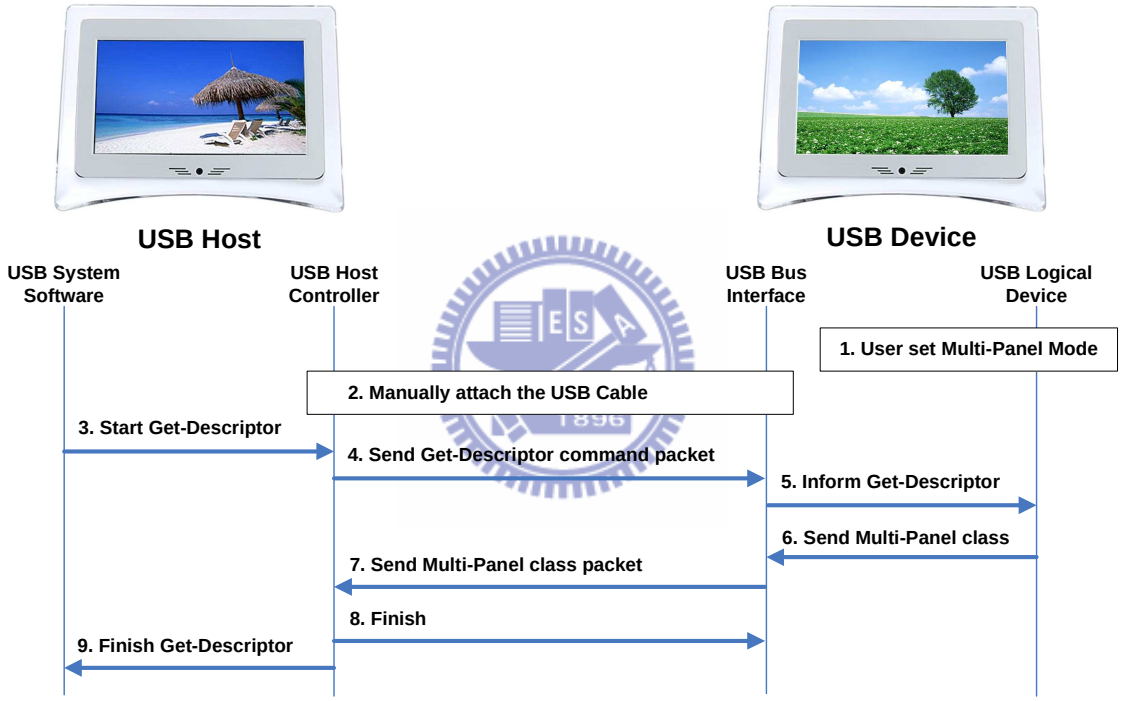


圖 3.2　裝置描述元傳輸流程

步驟一：使用者將 USB Device 設定為 Multi-Panel 的模式。

步驟二：手動以 USB 線(USB Cable)將兩台數位相框連接。

步驟三：USB Host 的 USB System Software 將 Get Descriptor 命令轉換為 USB 封包格式後，通知 USB Host Controller 可以開始傳送 USB 封包。

步驟四：USB 主機端控制器(USB Host Controller)實際傳送 USB 封包至 USB

Device。

步驟五：當 USB Device 的 USB Bus Interface 收到 USB 封包後，送交給 USB Logical Device。

步驟六：USB Logical Device 經分析 USB 封包後，得知 USB Host 需識別 USB Device 的類別，故使用者將裝置描述元(如表 2.1)設定為 Multi-Panel 類別(如表 3.1)並轉換為 USB 封包格式後，通知 USB Bus Interface 可以開始傳送 USB 封包。

步驟七：USB Bus Interface 實際傳送 USB 封包至 USB Host。

步驟八：當 USB Host 的 USB 主機端控制器收到 USB 封包並且確認資料正確後，會回覆已完成訊息給 USB Device。

步驟九：USB 主機端控制器通知 USB System Software 已順利完成裝置描述元的傳輸流程，同時 USB Host 亦可得知 USB Device 為 Multi-Panel 類別。

表 3.1　Multi-Panel 類別的裝置描述元

欄位	數值	說明
bLength	0x12	裝置描述元長度 18 Bytes
bDescriptorType	0x01	裝置種類(DEVICE)
bcdUSB	0x0200	USB2.0 規範
bDeviceClass	0xFF	Multi-Panel 類別
bDeviceSubClass	0x00	
bDeviceProtocol	0x00	無裝置通訊協定
bMaxPacketSize0	0x40	控制型端點的封包最大傳輸量為 64Bytes
idVendor	0x0851	製造商
idProduct	0x1544	產品代碼
bcdDevice	0x0200	裝置韌體版本 2.0
iManufacturer	0x01	製造商字串索引
iProduct	0x02	產品字串索引
iSerialNumber	0x03	序號字串索引
bNumConfigurations	0x01	一個組態描述元

3.2.2 原始影像資料傳輸流程

圖 3.3 繪出原始影像資料傳輸(Send Raw Data Transfer)流程，說明 USB Host 如何傳輸原始影像資料給 USB Device。其步驟逐一說明如下。

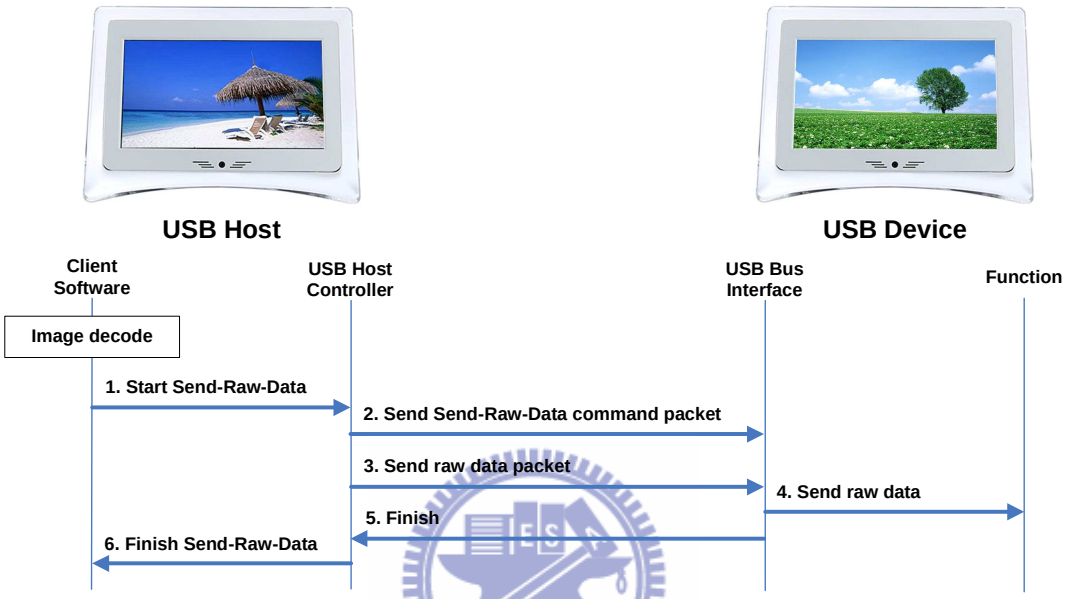


圖 3.3 原始影像資料傳輸流程

步驟一：相片經解壓縮為影像原始檔後，由 USB Host 的 Client Software 將傳送影像原始檔的命令轉換為 USB 封包格式，並通知 USB 主機端控制器(USB Host Controller)可以開始傳送 USB 封包。

步驟二：USB 主機端控制器實際傳送 USB 封包至 USB Device。

步驟三：USB Device 的 USB Bus Interface 收到 USB 封包後，會開始接收 USB Host 傳來原始影像的實際資料。

步驟四：USB Device 的 Function 收到 USB Host 傳送的影像原始檔後，再將該影像原始檔之內容複製到螢幕對應之頁面內。

步驟五：當上述步驟四完成並確認資料正確後，USB Device 會回覆已完成訊息給 USB Host。

步驟六：USB Host 的 USB 主機端控制器通知 Client Software 已順利完成

原始影像資料傳輸流程。

3.2.3 變更畫面命令流程

圖 3.4 繪出變更畫面命令(Change Window Command)流程，說明 USB Host 如何通知 USB Device 變更新畫面。其步驟逐一說明如下。

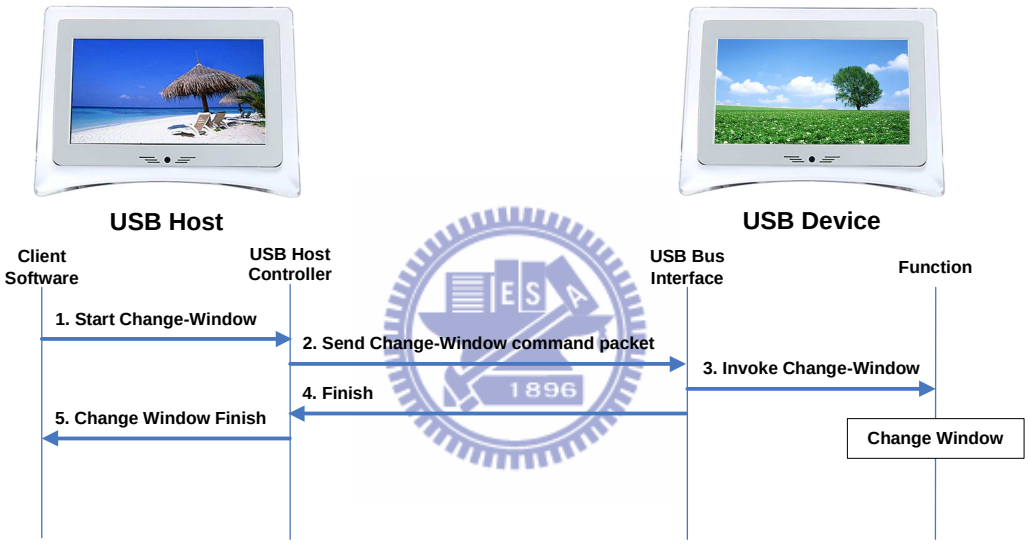


圖 3.4　變更畫面命令流程

步驟一：經原始影像資料傳輸流程完成後，由 USB Host 的 Client Software 將更換頁面的命令轉換為 USB 封包格式，並通知 USB 主機端控制器(USB Host Controller)可以開始傳送 USB 封包。

步驟二：USB 主機端控制器實際傳送 USB 封包至 USB Device。

步驟三：USB Device 的 USB Bus Interface 收到 USB 封包後，會通知 Function 更換新的頁面，即可在螢幕上顯示該影像。

步驟四：當上述步驟三完成後，USB Device 會回覆已完成訊息給 USB Host。

步驟五：USB Host 的 USB 主機端控制器通知 Client Software 已順利完成

變更畫面流程。

3.3 數位相框軟體模組

首先對於事件標記(Event Flag[10])及訊息(Message[10])之定義加以說明如下：

- 事件標記(Event Flag；如圖 3.5)係一個 32 位元向量(Bit Vector)，每位元代表一個事件(Event)是否發生。如行程 A(Task A)通知行程 B(Task B)某事件發生，首先行程 A 會將事件標記內之事件的位元值設定為 1 時，作業系統(Operating System)即可通知行程 B 接收該事件標記並加以處理。

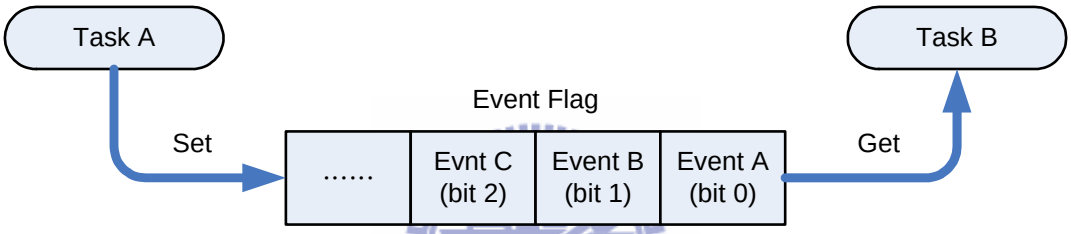


圖 3.5 事件標記說明

- 訊息(Message；如圖 3.6)係大小可變形式，可為 Byte、Integer、Long 等型態，如行程 A(Task A)傳遞訊息給行程 B(Task B)，首先行程 A 會將該訊息放置訊息佇列(Message Queue)內時，作業系統會將該訊息以先進先出(FIFO)方式通知行程 B 接收並加以處理。

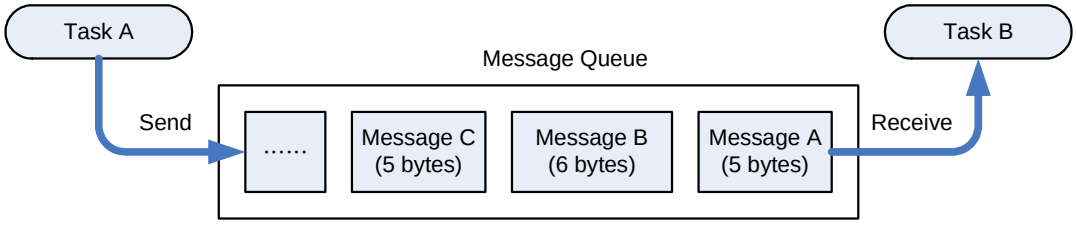


圖 3.6　訊息說明

數位相框軟體模組(如圖 3.7)是依據數位相框傳輸架構圖(如圖 3.1)，將 USB Host

及 USB Device 以軟體程式名稱方式表示。

USB Host 端的命令處理程式(CommandProcess；如圖 3.7(a))係呼叫 USB 主機端命令程式(UsbHostCommand；如圖 3.7(b))，將一般的資料轉換為 USB 封包格式，再以事件標記(Event Flag)通知 USB 主機端驅動任務程(UsbHostDriverTask；如圖 3.7(c))，其目的是透過 USB 主機端硬體啟動程式(UsbHostSendActive；如圖 3.7(d))使 USB 主機端控制器(USB Host Controller;如圖 3.7(e))傳送 USB 封包至 USB Device 端。當 USB Device 端的 USB Bus Interface (如圖 3.7(f))收到 USB 封包後，會立刻產生 USB 中斷(USB Interrupt)，進而呼叫所對應的中斷程式(UsbDeviceIsr；如圖 3.7(g))進行處理，並以訊息(Message)方式通知 USB 裝置端設備任務程式(UsbDeviceTask；如圖 3.7(h))，其目的是透過 USB 裝置端處理程式(UsbDeviceHandler；如圖 3.7(i))將 USB 封包轉換為一般命令，並依據該命令存取系統資源(System Resource；如圖 3.7(j))以取得 USB Host 端所需的資料，爾後再經由 USB 裝置端行動程式(UsbDeviceAction；如圖 3.7(k))使資料透過 USB Bus Interface 回覆 USB Host。當 USB Host 端的 USB 主機端控制器收到 USB 封包後，會立刻產生 USB 中斷(USB Interrupt)，進而呼叫所對應的中斷程式(UsbHostIsr；如圖 3.7(l))進行處理，並依序以事件方式通知命令處理程式其已完成此次傳輸流程，以便可再進行下一次的傳輸流程。

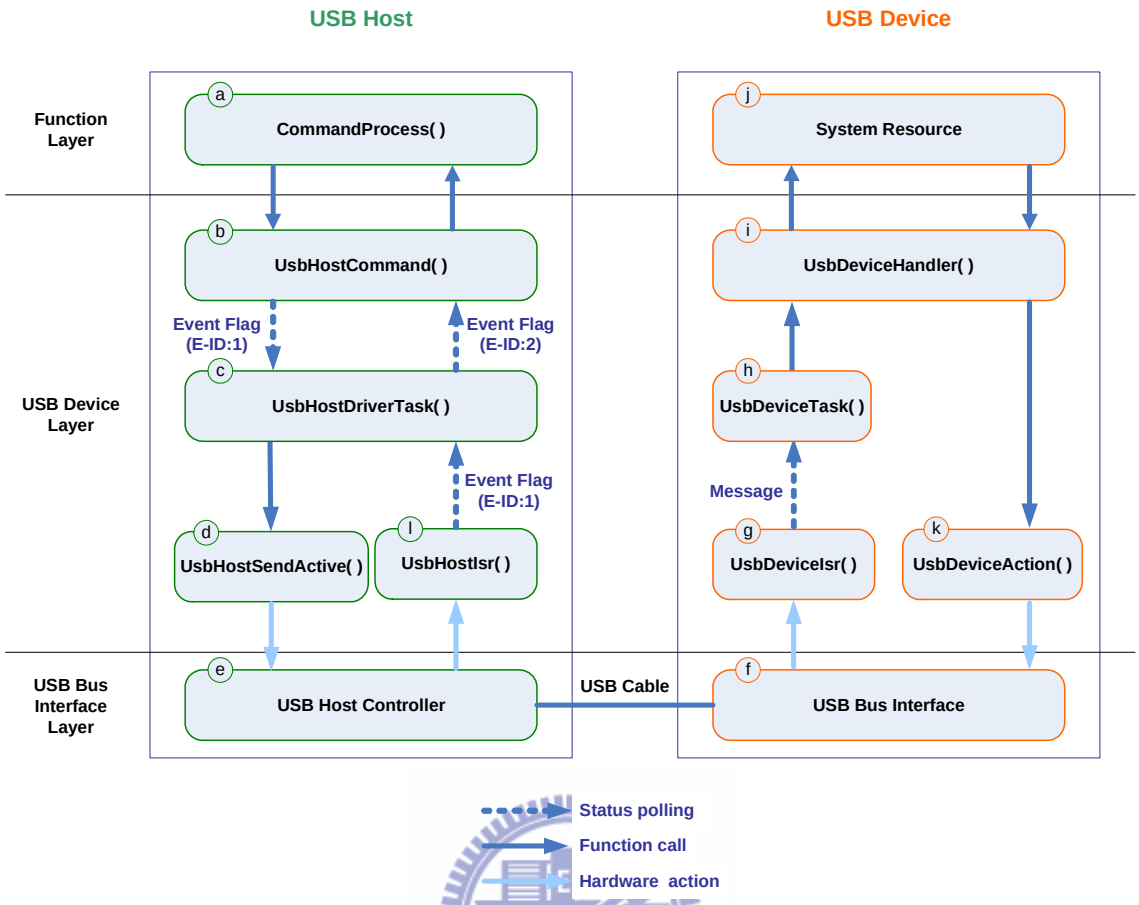


圖 3.7　數位相框軟體模組及訊息/事件流程

3.4 數位相框傳輸方法

本節先以兩台串聯的數位相框為主，一台數位相框為 USB Host 角色，另一台為 USB Device 角色，彼此傳輸方法為傳輸方法一及傳輸方法二(即本文處理方法)。傳輸方法一係由 USB Host 直接將相片檔案傳至 USB Device 端，再經由解壓縮後顯示至螢幕上(如圖 3.8)；傳輸方法二則由 USB Host 直接將解壓縮後的原始影像資料傳至 USB Device 端加以顯示(如圖 3.9)。

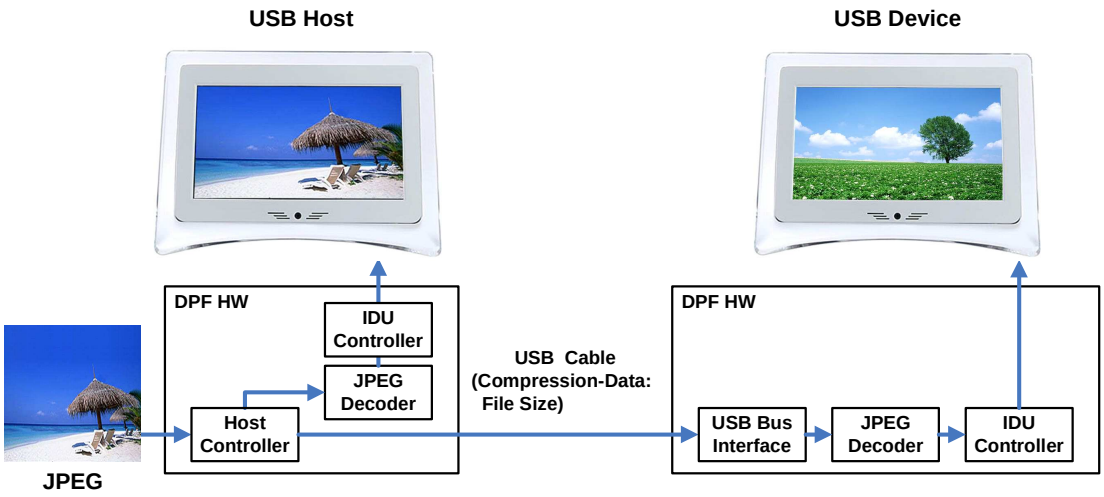


圖 3.8　傳輸方法一

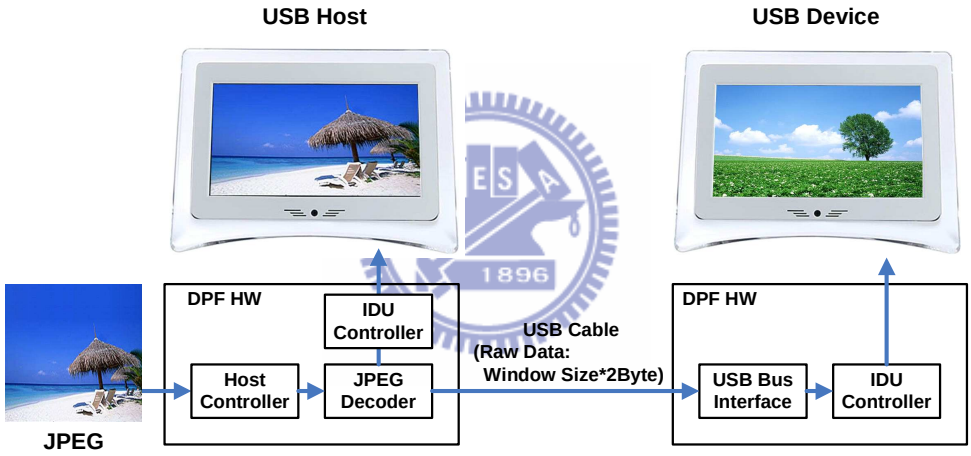


圖 3.9　傳輸方法二

(1) 傳輸方法一(如圖 3.8)由 USB Host 透過 USB Cable 將 JPEG 影像檔案(Compression-Data:File Size)傳輸至 USB Device 端後，再經由 JPEG 影像解碼器(JPEG Decoder)將之解壓縮為原始影像資料後，可提供給影像顯示單元控制器(Image Display Unit Controller；IDU Controller)顯示至螢幕上。

(2) 傳輸方法二(如圖 3.9)由 USB Host 端的 JPEG 影像解碼器(JPEG Decoder)將 JPEG 影像檔案解壓縮為螢幕尺寸之原始影像資料(Scaled-down raw data：Window Size x 2 Bytes)後，可傳輸至 USB Device 端，並且直接提供給 IDU

Controller 處理，最後影像就會呈現在螢幕上。

第四章 多個螢幕媒體設備傳輸相片之實作

本章介紹多個螢幕媒體設備之間互相傳輸相片的軟體實作(如圖 4.1)，以兩台串聯的數位相框為例，可由第一台數位相框透過 USB 連接線將相片由左至右方向依序傳輸至第二台數位相框，以達到影像動態傳輸的效果。本實驗的開發設備包括四個部分：

(1) 數位相框(DPF)兩台(含電源供應器)為主要開發設備。

(2) 除錯轉接板兩套可將數位相框的序列傳輸埠(RS232)介面轉為 USB 介面，並透過 USB 傳輸線連接電腦，使除錯的訊息顯示到電腦螢幕上。

(3) 標準 USB2.0 連接線(USB Cable)三條分別與數位相框及電腦連接，其中一條用於連接兩台數位相框，另外兩條則分別用於連接兩台數位相框的除錯轉接板與電腦。

(4) 開發程式一套係採用美普思科技公司(MIPS Technologies, Inc)的 MIPS[11]處理器平台開發軟體。

圖 4.1 多個螢幕媒體設備相片傳輸特效實作成果

本章以虛擬程式碼 (Pseudo Code)方式說明以上四部分的程式運作。

4.1 USB Host 程式說明

USB Host 程式逐一說明如下：

- 命令處理程式(CommandProcess；如圖 4.2(a))使用迴圈方式不斷的等待事件發生，並依據事件值執行相關動作(UsbHostCommand；如圖 4.2(b))，例如 Multi-Panel 類別的應用命令 VENDOR_SEND_CMD(參見第 3.2.2 節) 與 VENDOR_CHG_CMD(參見第 3.2.3 節)。C 語言程式碼請參考圖 4.3。

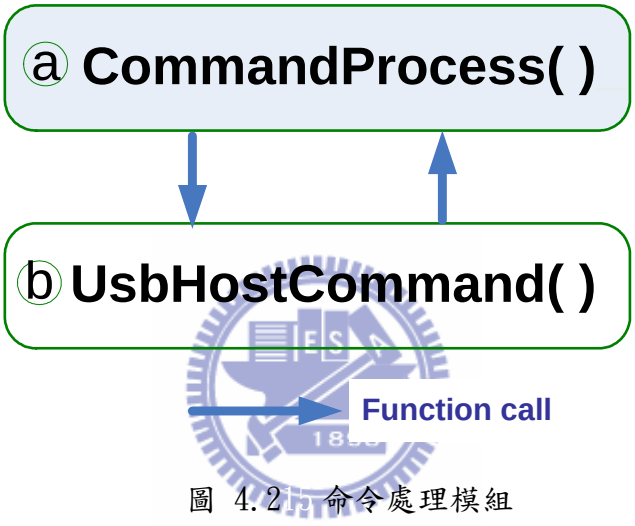


圖 4.2 命令處理模組

```
1   WORD CommandProcess (WORD wCmd)
2   {
3       WORD wRetValue = PASS;
4       switch (wCmd){
5           case READ_CMD:
6               wRetValue = UsbhMsdcCommand(SCSI_READ_10);
7               break;
8           case WRITE_CMD:
9               wRetValue = UsbhMsdcCommand(SCSI_WRITE_10);
10              break;
11          case VENDOR_SEND_CMD:
12              wRetValue = UsbhMsdcCommand(MPXMP_SEND_DATA);
13              break;
14          case VENDOR_CHG_CMD:
15              wRetValue = UsbhMsdcCommand(MPXMP_CHG_WIN);
16              break;
17          default:
18              MP_DEBUG("-E- INVALID CMD");
19              wRetValue = FAIL;
20              break;
21      }
22      return wRetValue;
23  }
```

圖 4.3　命令處理程式碼

Line 3. 宣告 wRetValue 變數以儲存回傳值，俾提供呼叫者判斷成功與否。

Lines 5–10. 此為 Mass-Storage 類別之讀取資料(READ_CMD)與寫入資料(WRITE_CMD)應用命令，將之以傳輸命令提供給 USB 主機端命令程式(UsbHostCommand)。

Lines 11–16. 此為 Multi-Panel 類別之 VENDOR_SEND_CMD 與 VENDOR_CHG_CMD 應用命令，將之以傳輸命令提供給 USB 主機端命令程式(UsbHostCommand)。

Lines 11–20. 如果 USB 傳輸命令非為上述等情形，則顯示警告訊息。

- USB 主機端命令程式(UsbHostCommand；如圖 4.4(b))負責將傳輸的命令轉為 USB 格式，並以事件方式通知 USB 主機端驅動任務程式(UsbHostDriverTask；如圖 4.4(c))開始執行命令，後續再通知命令處理程式(CommandProcess；如圖 4.4(a))該命令執行成功或失敗。C 語言程式碼請參考圖 4.5。

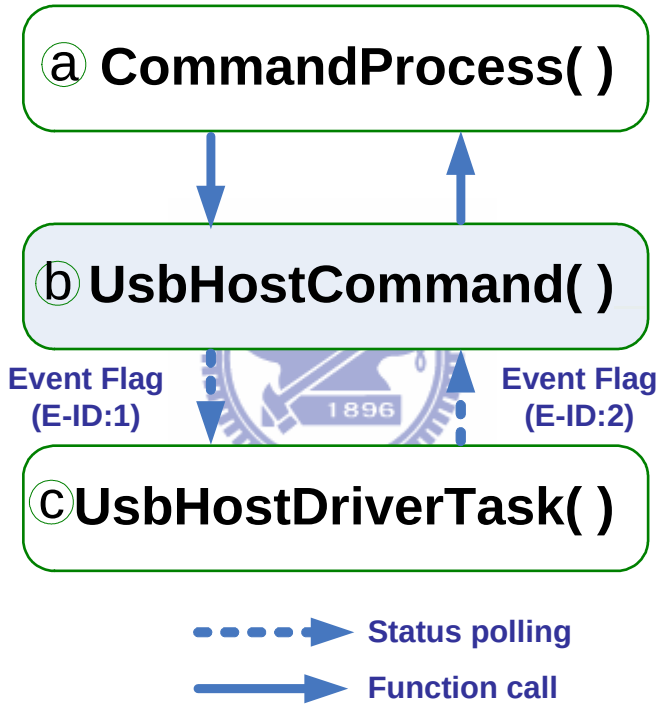


圖 4.4　USB 主機端命令模組

```
1   WORD UsbHostCommand(BYTE bCmd)
2   {
3       DWORD dwEventFlag = 0;
4       WORD   wRetValue = OS_STATUS_OK;
5       UsbBuilder(bCmd);
6       SetCurUsbState(COMMAND_STATE);
7       EventSet(USB_HOST_DRIVER_ID, EVENT_ACTIVE);
8       wRetValue = EventWaitWithTO(USB_HOST_COMMAND_ID,
9                                   &dwEventFlag, TIME_OUT_THRESS_SEC);
10      if(wRetValue == OS_STATUS_OK && dwEventFlag == EVENT_PASS)
11          return USB_NO_ERROR;
12      else
13          return USB_ERROR;
14  }
```

圖 4.5　USB 主機端命令程式碼

表 4.1　USB 主機端驅動任務程式事件說明(E-ID：1)

BIT	Event	Description
0	EVENT_PLUGIN	連接設備事件
1	EVENT_PLUGOUT	拔除設備事件
2	EVENT_ACTIVE	開始傳輸事件
3	EVENT_IOC	完成傳輸事件
4	EVENT_FINISH	整體完成事件
5	EVENT_ERROR	整體失敗事件

表 4.2　USB 主機端命令處理程式事件說明(E-ID：2)

BIT	Event	Description
0	EVENT_PASS	成功事件
1	EVENT_FAIL	失敗事件

Line 3. 宣告 dwEventFlag 變數以儲存事件值，俾提供後續判斷相關事件。該變數每一位元代表之事件如表 4.2 所示。

Line 4. 宣告 wRetValue 變數以儲存事件狀態值，俾提供後續判斷事件成功與否。

Lines 5 and 6. 呼叫 UsbCmdBuilder 程式將傳輸的命令轉為 USB 格式，並將目前 USB 狀態設定為命令狀態(COMMAND_STATE)。

Line 7. 將開始傳輸事件(EVENT_ACTIVE；如表 4.1)提供給 USB 主機端驅動任務程式(UsbHostDriverTask)。

Lines 8–12. 等待 USB 主機端驅動任務程式完成整體傳輸動作。如三秒內完成並收到成功事件(EVENT_PASS；如表 4.2)，則回傳 USB 正確(USB_NO_ERROR)，反之，則回傳 USB 錯誤(USB_ERROR)。

- USB 主機端驅動任務程式(UsbHostDriverTask;如圖 4.6(b))使用迴圈方式不斷的等待事件發生。當收到從 USB 主機端中斷服務程式(UsbHostIsr；如圖 4.6(d))或 USB 主機端命令程式(UsbHostCommand；如圖 4.6(a))所發送相關事件時，會依據該事件執行相對應的動作，如 USB 主機端硬體啟動程式(UsbHostSendActive；如圖 4.6(c))或 UsbHostCommand。C 語言程式碼請參考圖 4.7。

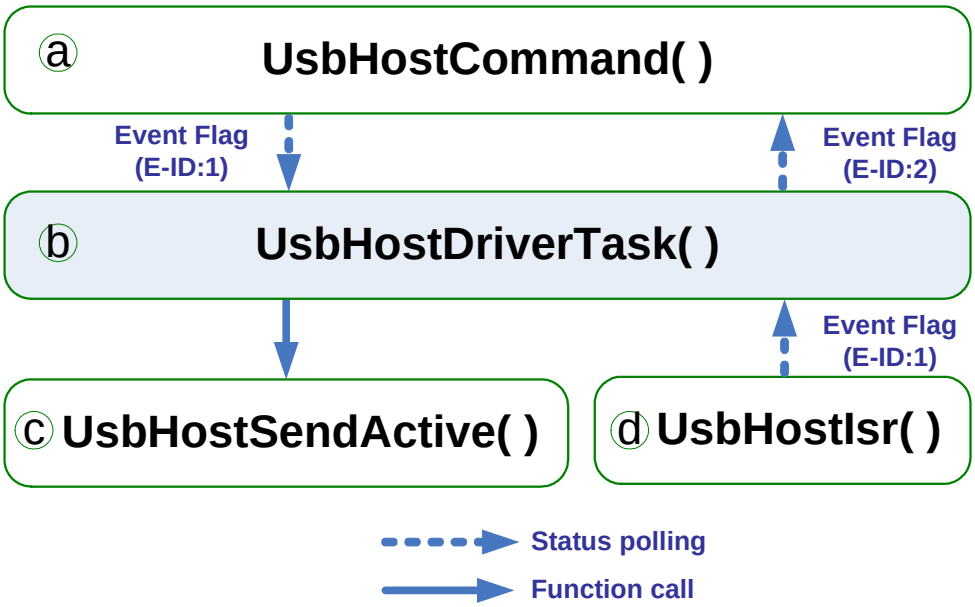


圖 4.6　USB 主機端驅動任務模組

```
1   void UsbHostDriverTask(void)
2   {
3       DWORD dwEventFlag = 0;
4       while (1) {
5           EventWait(USB_HOST_DRIVER_ID, &dwEventFlag);
6           If (dwEventFlag & EVENT_PLUGIN)
7               PlugInAction( );
8           If (dwEventFlag & EVENT_PLUGOUT)
9               PlugOutAction ( );
10          If (dwEventFlag & EVENT_ACTIVE)
11              UsbHostSendActive(ACTION_ACTIVE);
12          If (dwEventFlag & EVENT_IOC)
13              UsbHostSendActive(ACTION_IOC);
14          If (dwEventFlag & EVENT_FINISH)
15              EventSet(USB_HOST_COMMAND_ID, EVENT_PASS);
16          If (dwEventFlag & EVENT_ERROR)
17              EventSet(USB_HOST_COMMAND_ID, EVENT_FAIL);
18      }
19  }
```

圖 4.7　USB 主機端驅動任務程式碼

Line 3. 宣告 dwEventFlag 變數以儲存事件值，俾提供後續判斷相關事件。

Line 5. 無窮迴圈內不斷等待該事件發生並將事件值設定 dwEventFlag 變數。

Lines 6 and 7. 如果 dwEventFlag 變數為連接設備事件(EVENT_PLUGIN；如表 4.1)，則呼叫 PlugInAction 程式處理連接設備的後續動作。

Lines 8 and 9. 如果 dwEventFlag 變數為拔除設備事件(EVENT_PLUGOUT)，則呼叫 PlugOutAction 程式處理拔除設備的後續動作。

Lines 10 and 11. 如果 dwEventFlag 變數為開始傳輸事件(EVENT_ACTIVE)，則呼叫 USB 主機端硬體啟動程式(UsbHostSendActive)開始執行硬體啟動之動作。

Lines 12 and 13. 如果 dwEventFlag 變數為完成傳輸事件(EVENT_IOC)，則呼叫 USB 主機端硬體啟動程式(UsbHostSendActive)執行硬體完成之動作。

Lines 14 and 15. 如果 dwEventFlag 變數為整體完成事件(EVENT_FINISH)，則提供成功事件給 USB 主機端命令程式(UsbHostCommand)。

Lines 16 and 17. 如果 dwEventFlag 變數為整體失敗事件(EVENT_ERROR)，則提供失敗事件給 USB 主機端命令程式(UsbHostCommand)。

- USB 主機端硬體啟動程式(UsbHostSendActive；如圖 4.8(b))經由 UsbHostDriverTask(如圖 4.8(a))呼叫並依據不同的傳輸狀態(亦即 dwState 可以為 Command、Data 或 Ack)，控制 USB 主機端控制器(USB Host Controller；如圖 4.8(c))傳送及接收 USB 封包資料。C 語言程式碼請參考圖 4.9。

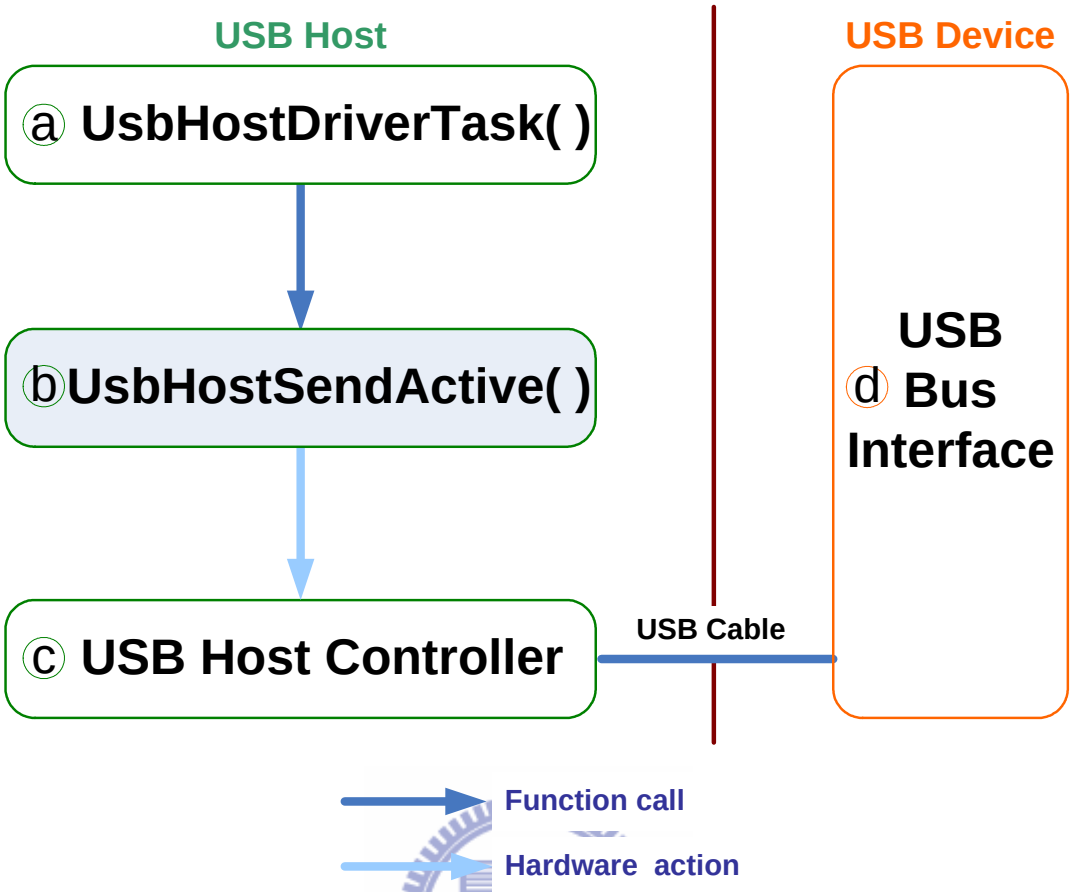


圖 4.82 USB 主機端硬體啟動模組

```
1   void UsbHostSendActive(BYTE bAction)
2   {
3       DWORD dwState = 0;
4       dwState = GetCurUsbState( );
5       switch (bAction) {
6           case ACTION_ACTIVE:
7               switch (dwState) {
8                   case COMMAND_STATE:
9                       SendCommand( );
10                      break;
11                  case DATA_IN_STATE:
12                      ReceiveData( );
13                      break;
14                  case DATA_OUT_STATE:
15                      SendData( );
16                      break;
```

```
17                case ACK_STATE:
18                    ReceiveAck( );
19                    break;
20                default:
21                    MP_DEBUG("-E- ACTIVE Invalid State");
22                    EventSet(USB_HOST_DRIVER_ID, EVENT_ERROR);
23                    break;
24            }
25            break;
26
27        case ACTION_IOC:
28            switch (dwState) {
29                case COMMAND_STATE:
30                    if(DataInSize > 0)
31                        SetCurUsbState(DATA_IN_STATE);
32                    else
33                        SetCurUsbState(DATA_OUT_STATE);
34                    EventSet(USB_HOST_DRIVER_ID, EVENT_ACTIVE);
35                    break;
36                case DATA_IN_STATE:
37                    SetCurUsbState(ACK_STATE);
38                    EventSet(USB_HOST_DRIVER_ID, EVENT_ACTIVE);
39                    break;
40                case DATA_OUT_STATE:
41                    SetCurUsbState(ACK_STATE);
42                    EventSet(USB_HOST_DRIVER_ID, EVENT_ACTIVE);
43                    break;
44                case ACK_STATE:
45                    EventSet(USB_HOST_DRIVER_ID, EVENT_FINISH);
46                    break;
47                default:
48                    MP_DEBUG("-E- IOC INVALID STATE");
49                    EventSet(USB_HOST_DRIVER_ID, EVENT_ERROR);
50                    break;
51            }
52            break;
53
54        default:
```

```
55              MP_DEBUG("-E- INVALID ACTION:0x%x", bAction);
56              EventSet(USB_HOST_DRIVER_ID, EVENT_ERROR);
57              break;
58      }
59  }
```

圖 4.9　USB 主機端硬體啟動程式碼

Line 3. 宣告 dwState 變數以儲存目前 USB 傳輸狀態值，俾提供後續判斷相關狀態。

Line 4. 將目前 USB 傳輸狀態值儲存到 dwState 變數。

Line 5. 參數 bAction 提供判斷相關動作，如執行硬體啟動(ACTION_ACTIVE)或收到硬體完成(ACTION_IOC)之動作。

Lines 8–10. 如果 USB 傳輸狀態(dwStat)為命令狀態(COMMAND_STATE)，則呼叫 SendCommand 程式設定 USB 主機端控制器(USB Host Controller)暫存器並將命令傳輸到相連接的另一台數位相框。

Lines 11–13. 如果 USB 傳輸狀態為接收資料狀態(DATA_IN_STATE)，則呼叫 ReceiveData 程式設定 USB Host Controller 暫存器並接收來自相連接的另一台數位相框資料。

Lines 14–16. 如果 USB 傳輸狀態為傳送資料狀態(DATA_OUT_STATE)，則呼叫 SendData 程式設定 USB Host Controller 暫存器並將資料傳送到相連接的另一台數位相框。

Lines 17–19. 如果 USB 傳輸狀態為回應狀態(ACK_STATE)，則呼叫 ReceiveAck 程式設定 USB 主機端控制器之暫存器並接收來自相連接另一台數位相框的完成傳輸流程回應通知。

Lines 20–23. 如果 USB 傳輸狀態非為上述等情形，則顯示警告訊息並提供整體失敗事件給 USB 主機端驅動任務程式(UsbHostDriverTask)。

Lines 29–35. 如果目前 USB 狀態為命令狀態(COMMAND_STATE)且需接收資

料(DataInSize > 0)，則將之設定為接收資料狀態(DATA_IN_STATE)；反之，則設定為傳送資料狀態(DATA_OUT_STATE)。爾後再提供開始傳輸事件(EVENT_ACTIVE；如表 4.1)給 USB 主機端驅動任務程式。

Lines 36–39. 如果目前 USB 狀態為接收資料狀態，則將之設定為回覆狀態(ACK_STATE)。爾後再提供開始傳輸事件給 USB 主機端驅動任務程式。

Lines 40–43. 如果目前 USB 狀態為傳送資料狀態，則將之設定為回覆狀態。爾後再提供開始傳輸事件給 USB 主機端驅動任務程式。

Lines 44–46. 如果目前 USB 狀態為回應狀態，則將之以完成傳輸事件(EVENT_FINISH)提供給 USB 主機端驅動任務程式。

Lines 47–50. 如果目前 USB 狀態非為上述等情形，則顯示警告訊息並提供整體失敗事件給 USB 主機端驅動任務程式(UsbHostDriverTask)。

Lines 54–57. 如果 USB 動作非為上述等情形，則顯示警告訊息並提供整體失敗事件給 USB 主機端驅動任務程式(UsbHostDriverTask)。

- USB 主機端中斷服務程式(UsbHostIsr；如圖 4.10(b))負責接收經由 USB Device 端的 USB 匯流介面(USB Bus Interface；如圖 4.10(d))至 USB Host 端的 USB 主機端控制器(USB Host Controller；如圖 4.10(c))所產生的中斷，並將 USB Host 相關中斷以事件方式通知 USB 主機端驅動任務程式(UsbHostDriverTask；如圖 4.10(a))。C 語言程式碼請參考圖 4.11。

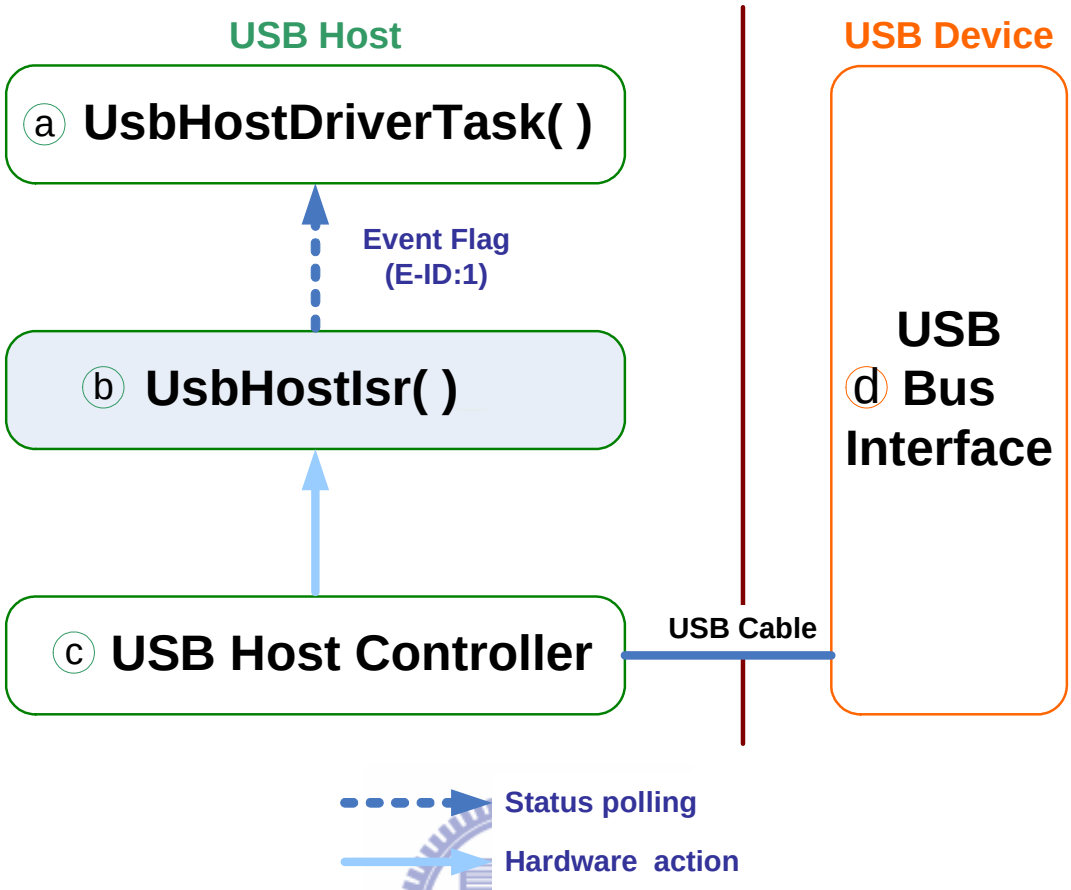


圖 4.10 USB 主機端中斷服務模組

```
1.  void UsbHostIsr(void)
2.  {
3.      WORD   wIntStatus = 0;
4.      DWORD  dwState = 0;
5.      wIntStatus = ReadUsbIntStatus();
6.      if (wIntStatus & CompletionOfTransaction)
7.          EventSet(USB_HOST_DRIVER_ID, EVENT_IOC);
8.      if (wIntStatus & DevicePlugIn)
9.          EventSet(USB_HOST_DRIVER_ID, EVENT_PLUGIN);
10.     if (wIntStatus & DevicePlugOut)
11.         EventSet(USB_HOST_DRIVER_ID, EVENT_PLUGOUT);
}
```

圖 4.11 USB 主機端中斷程式碼

Line 3. 宣告 wIntStatus 變數以儲存 USB Host 中斷的狀態值，俾提供後續

判斷相關狀態。

Line 4. 宣告 dwState 變數以儲存目前 USB 傳輸狀態值，俾提供後續判斷相關狀態。

Line 5. 讀取 USB Host 中斷的狀態並儲存至 wIntStatus 變數。

Lines 6 and 7. 如果 USB Host 中斷的狀態為傳輸完成狀態(CompletionOf Transaction)，則將之以完成傳輸事件(EVENT_IOC)提供給 USB 主機端驅動任務程式處理。

Lines 8 and 9. 如果 USB Host 中斷的狀態為連接設備狀態(DevicePlugIn)，則將之以連接設備事件(EVENT_PLUGIN)提供給 USB 主機端驅動任務程式處理。

Lines 10 and 11. 如果 USB Host 中斷的狀態為拔除設備狀態(DevicePlug Out)，則將之以拔除設備事件(EVENT_PLUGOUT)提供給 USB 主機端驅動任務程式處理。

4.2 USB Device 程式說明

USB Device 程式部分，逐一說明如下：

- USB 裝置端中斷服務程式(UsbDeviceIsr；如圖 4.12(b))負責接收經由 USB Host 端的 USB 主機端控制器(USB Host Controller；如圖 4.12(d))至 USB Device 端的 USB 匯流介面(USB Bus Interface；如圖 4.12(c))所產生的中斷，並將 USB Device 觸發的中斷訊號以訊息結構方式通知 USB 裝置端設備任務程式(UsbDeviceTask；如圖 4.12(a))。C 語言程式碼請參考圖 4.13。

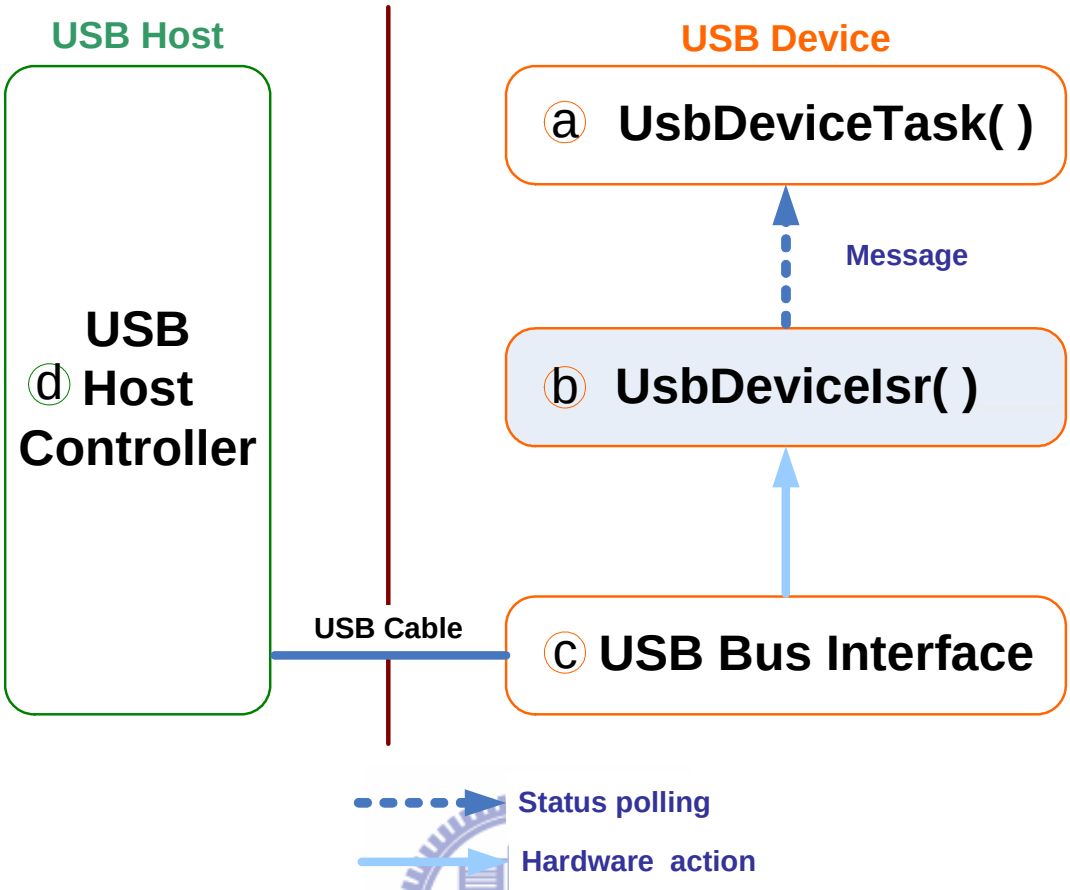


圖 4.12 USB 裝置端中斷服務模組

```
1   typedef struct {
2       DWORD dwArgument1;
3       DWORD dwArgument2;
4       DWORD dwArgument3;
5       DWORD dwArgument4;
6   } ST_MESSAGE_CONTENT;
7
8   void UsbDevIsr(void)
9   {
10      ST_MESSAGE  stMessage;
11      SystemIntDisable(UsbDevice);
12      stMessage.Argument1 = ReadUsbIntStatus( );
13      MessageSend(USB_DEVICE_ID, &stMessage);
14  }
```

圖 4.13 USB 裝置端中斷服務程式碼

Lines 1–6. 定義訊息結構類型。訊息結構內包涵參數一(dwArgument1)至參數四(dwArgument4)，俾藉參數傳遞資訊。

Line 10. 宣告訊息結構類型變數(stMessage)以儲存 USB Device 中斷值，俾提供給 USB 裝置端設備任務程式(UsbDeviceTask)處理。

Line 11. 系統停止 USB Device 中斷。

Lines 12 and13. 將 USB Device 中斷狀態儲存至訊息結構的參數一內，爾後再提供給 USB 裝置端設備任務程式處理。

- USB 裝置端設備任務程式(UsbDeviceTask；如圖 4.14(b))使用迴圈方式不斷地接收來自於 UsbDeviceIsr(如圖 4.14(c))的訊息時，會依據該事件執行相對應的動作(UsbDeviceHandler;如圖 4.14(a))。C 語言程式碼請參考圖 4.15。

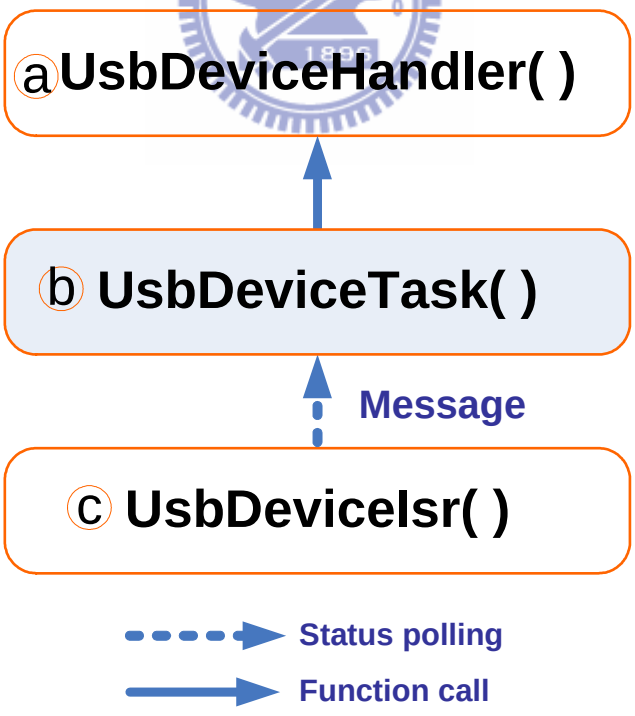


圖 4.14 USB 裝置端設備任務模組

```
1   void UsbDeviceTask(void)
2   {
3       ST_MESSAGE  stMessage;
4       while(1)
5       {
6           MessageReceive(USB_DEVICE_ID, &stMessage);
7           UsbDeviceHandler(stMessage.Argument1);
8           SystemIntEnable (UsbDevice);
9       }
10  }
```

圖 4.15 USB 裝置端設備任務程式碼

Line 3. 宣告訊息結構類型變數(stMessage)以儲存來自於 USB 裝置端中斷服務程式(UsbDevIsr)的訊息，俾提供給 USB 裝置端處理程式(UsbDeviceHandler)處理。

Lines 6 and 7. 在無窮迴圈內，不斷地接收訊息。爾後再以訊息結構的參數一(stMessage.Argument1)提供給 USB 裝置端處理程式處理。

Line 8. 系統啟動 USB Device 中斷。

- USB 裝置端處理程式(UsbDeviceHandler;如圖 4.16(b))經由 USB 裝置端設備任務程式(UsbDeviceTask；如圖 4.16(c))呼叫，進而處理資料接收或傳送。如需系統資源(System Resource；如圖 4.16(a)；例如檔案資料)，則會向其存取資料，並依據 USB Host 的需求執行接收或傳送的動作(UsbDeviceAction；如圖 4.16(d))。C 語言程式碼請參考圖 4.17。

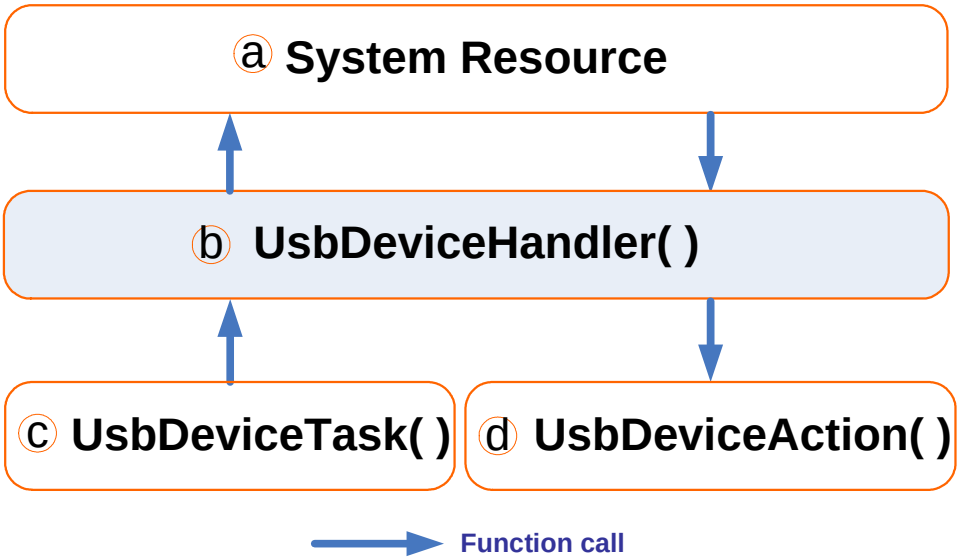


圖 4.16 USB 裝置端處理模組

```
1   void UsbDeviceHandler(BYTE bUsbIntStatus)
2   {
3       BYTE* pbBuff;
4       if (bUsbIntStatus & USB_DATA_IN)  {
5           Read pbBuff from System Resource Buffer Action
6           UsbDeviceAction(UsbDataIn, pbBuff);    }
7       if (bUsbIntStatus & USB_DATA_OUT)  {
8           UsbDeviceAction(UsbDataOut, pbBuff);
9           Write pbBuff to System Resource Buffer Action  }
10  }
```

圖 4.17 USB 裝置端處理程式碼

Line 3. 宣告記憶體空間指標(pbBuff)，俾提供資料存取。

Lines 4–6. 如果 USB 中斷狀態為 USB Device 需傳送資料至 USB Host 時(如 USB_DATA_IN)，可由 USB 裝置端行動程式(UsbDeviceAction)將系統資源提供給 USB Host。

Lines 7–9. 如果 USB 中斷狀態為 USB Host 需傳送資料至 USB Device 時(如 USB_DATA_OUT)，可由 USB 裝置端行動程式將之提供至 USB Device 的系統資源內。

- USB 裝置端行動程式(UsbDeviceAction；如圖 4.18(b))係經由 USB 裝置端處理程式(UsbDeviceHandler;如圖 4.18(a))呼叫，負責控制 USB 匯流介面(USB Bus Interface;如圖 4.18(c))作實際資料的接收或傳送。例如 USB Device 需傳送資料至 USB Host 時，則由 USB Device Action 程式將資料寫入 USB Bus Interface 內的硬體記憶暫存區(USB FIFO)，爾後再送至 USB Host 端(如圖 4.18(d))。C 語言程式碼請參考圖 4.19。

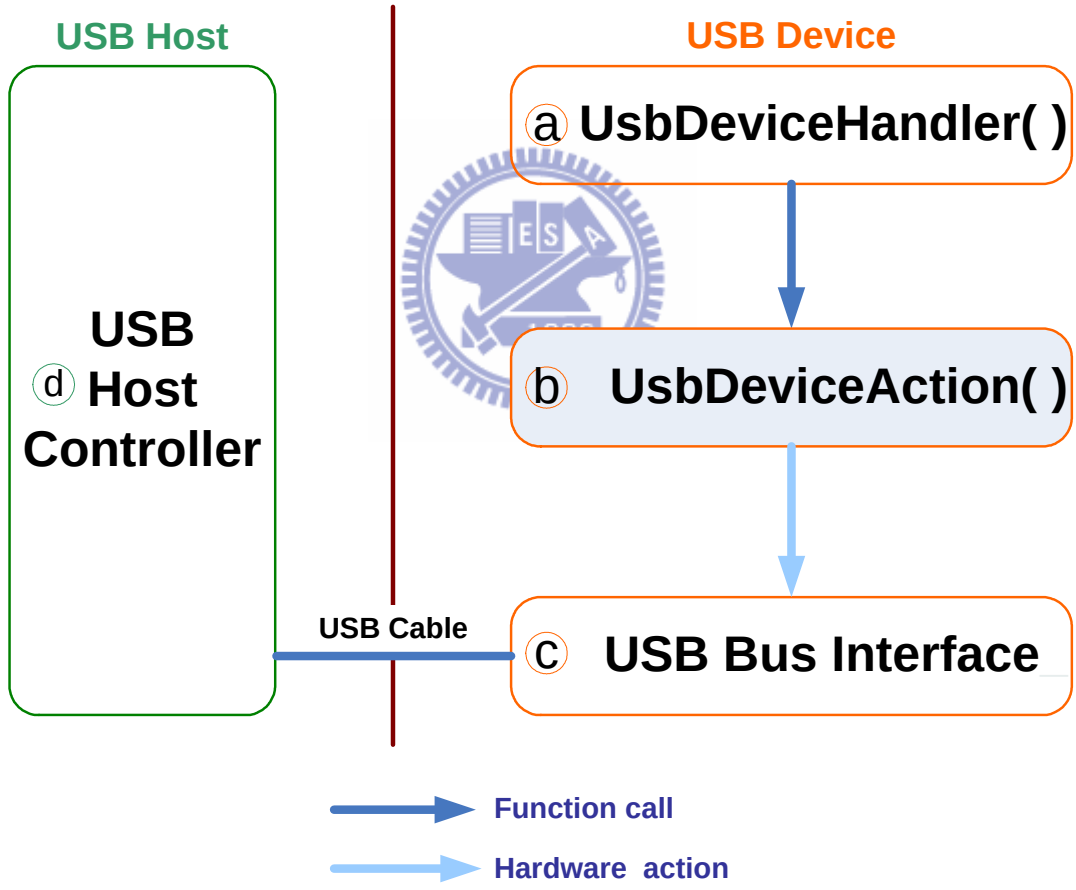


圖 4.18 USB 裝置端行動模組

```
1   void UsbDeviceAction(BYTE UsbDataIO, BTYE* pbData)
2   {
3       switch(UsbDataIO)
4       {
5           case UsbDataIn:
6               WriteUsbFifo(pbData);
7               break;
8           case UsbDataOut:
9               ReadUsbFifo(pbData);
10              break;
11          default:
12              MP_DEBUG("-E- UsbDeviceAction Invalid Data IO");
13              break;
14      }
15  }
```

圖 4.19 USB 裝置端行動程式碼

Lines 5–7. 如果 USB 資料方向為 USB Device 需傳送資料至 USB Host 時(如 UsbDataIn)，可由 WriteUsbFifo 程式將資料(如 pbData)寫入硬體記憶暫存區(USB FIFO)，爾後再送到 USB Host。

Lines 8–10. 如果 USB 資料方向為 USB Host 需傳送資料至 USB Device 時(如 UsbDataOut)，可由 ReadUsbFifo 程式從硬體記憶暫存區讀取資料至 USB Device。

Lines 11–13. 如果 USB 資料方向非為上述等情形，則顯示警告訊息。

第五章 多個螢幕媒體設備之效能

本章探討多個螢幕媒體傳輸之效能，根據第 3 節的兩種傳輸方法作整體效能的評估。

5.1 運算式說明

傳輸方法一(Method 1)所需花費總時間(Total Time)的運算式包含兩次將 JPEG 檔案解壓縮成原始影像資料的時間(Decode Time)及 USB 資料傳輸時間(USB Time A)，而 USB 資料傳輸時間為檔案大小(File Size)除以 USB 傳輸率(USB Trans Rate)。傳輸方法二(Method 2)所需花費總時間的效能運算式包含一次將 JPEG 檔案解壓縮成原始影像資料的時間(Decode Time)及 USB 資料傳輸時間(USB Time B)，而 USB 資料傳輸時間為影像解碼後的原始資料(Window Size x 2 Bytes)除以 USB 傳輸率(USB Trans Rate)。其兩種傳輸方法效能的運算式說明如下：

$$\text{Method 1 Total Time} = \text{Decode Time} + \text{USB Time A} + \text{Decode Time}$$

$$\text{Method 2 Total Time} = \text{Decode Time} + \text{USB Time B}$$

其中傳輸方法一及傳輸方法二之 USB 資料傳輸時間表示如下：

$$\text{Method 1 USB Time A} = \frac{\text{File Size}}{\text{USB Trans Rate}}$$

$$\text{Method 2 USB Time B} = \frac{\text{Window Size} \times 2}{\text{USB Trans Rate}}$$

因上述兩種傳輸方法將 JPEG 檔案解壓縮成原始影像資料的時間相同，可簡化如下：

$$\text{Method 1 Total Time} = \frac{\text{File Size}}{\text{USB Trans Rate}} + \text{Decode Time}$$

$$\text{Method 2 Total Time} = \frac{\text{Window Size} \times 2}{\text{USB Trans Rate}}$$

上述兩種運算式經簡化後，則以 USB 資料傳輸時間及解壓縮成原始影像資料的時間來分析傳輸方法一及傳輸方法二的差異。

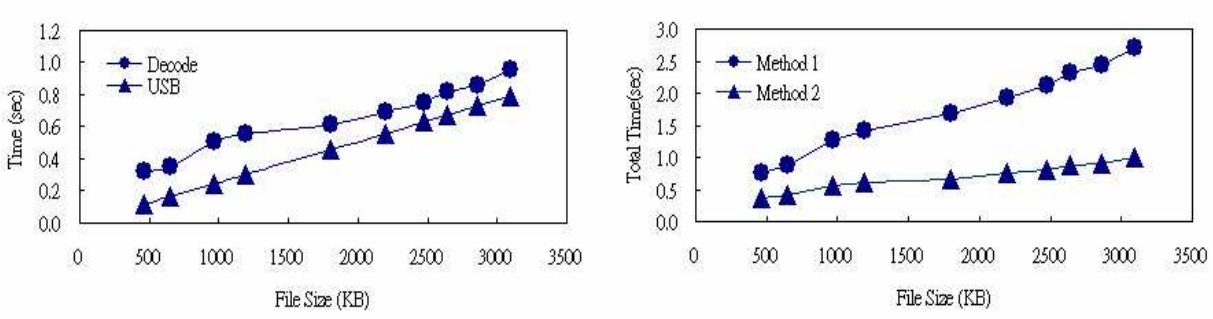
5.2 實際效能分析

本節依據不同照片的影像解析度(Resolution)及檔案大小[File Size(KB)]，比較兩種傳輸方法所需花費總時間[Total Time(sec)](如表 5.1)，總時間包含將 JPEG 檔案解壓縮成原始影像資料的時間[Decode Time(sec)]及 USB 資料傳輸時間[USB Time(sec)]。傳輸方法一(Method 1)需花費兩次 JPEG 檔案解壓縮成原始影像資料的時間及影像檔案較大則傳輸效率較差之關係，因此相較之下，則以傳輸方法二(Method 2)較佳。

表 5.1　不同傳輸大小比較

Image Information					Method 1		Method 2		Result
ID	Resolution	File Size(KB)	Scaled-down Raw Data Size(KB)	Decode Time(sec)	USB Time A(sec)	Total Time = 2 x Decode Time + USB Time A(sec)	USB Time B(sec)	Total Time = Decode Time + USB Time B(sec)	Total Time of Method1 - Method2(sec)
1	1200x900	463	219.375	0.318	0.118	0.754	0.056	0.374	0.380
2	1600x1200	651	219.375	0.351	0.166	0.868	0.056	0.407	0.461
3	2048x1536	968	219.375	0.505	0.247	1.257	0.056	0.561	0.696
4	2272x1704	1191	219.375	0.560	0.304	1.424	0.056	0.616	0.808
5	3072x2048	1800	219.375	0.612	0.459	1.683	0.056	0.668	1.015
6	3456x2304	2197	219.375	0.689	0.561	1.939	0.056	0.745	1.194
7	3696x2448	2478	219.375	0.747	0.633	2.127	0.056	0.803	1.324
8	3648x2736	2642	219.375	0.822	0.674	2.318	0.056	0.878	1.440
9	4064x2709	2860	219.375	0.855	0.730	2.440	0.056	0.911	1.529
10	4272x2848	3095	219.375	0.953	0.790	2.696	0.056	1.009	1.687

JPEG 檔案解壓縮成原始影像資料的時間(如圖 5.1(a)之 Decode)及 USB 資料傳輸時間(如圖 5.1(a)之 USB)皆會隨著檔案大小[File Size(KB)]增加而增長。同樣地，整體傳輸所需花費的總時間[Total Time(sec)]亦是如此。依照傳輸方法比較圖(如圖 5.1(b))可得知傳輸方法二(Method 2)所需花費總時間比傳輸方法一(Method 1)短(如表 5.1 之 Result)。



(a) USB 傳輸時間及影像解碼　　(b) 傳輸方法比較

圖 5.1 傳輸時間及方法

本論文以數位相框解析 480 x 234 為例，則 Window Size x 2 Bytes 資料大小為 480 x 234 x 2 Bytes ÷ 1024 = 219.375 KB(如表 5.1 之 Scaled-down Raw Data Size)，因傳輸方法一需多一次解壓縮成原始影像資料的時間(Decode Time)，所以檔案大小(File Size)大於 219.375 KB 者，則建議使用傳輸方法二，較能縮短時間，提高整體應用效能。

第六章 結論與未來展望

目前市面數位相框皆為單一螢幕顯示照片，經由本論文所實作之 USB 主機端(USB Host)及 USB 裝置端(USB Device)兩種角色的程式，並以自訂的通訊協定，讓多台數位相框連接並彼此溝通，以達到多個螢幕媒體設備之照片動態傳輸功能。基於整體成本考量下，為提升整體效率，故在軟體上做相對的調整，並根據第五章之實作效能分析後，發現 JPEG 檔案解壓縮成原始影像資料的時間較為耗時，故如傳輸方法二直接將解壓縮成原始影像資料透過 USB 方式直接傳輸，使串接的第二台螢幕媒體設備可節省解壓縮時間，則傳輸方法二的整體效率較佳。

為探索多個螢幕媒體設備的未來發展方向及應用，以下提供幾個研究方向如下：

(a) 在不同大小的螢幕彼此傳輸效能的情形。如將解析度較大(例如:600x480)的數位相框的相片以螢幕尺寸之原始影像資料(Scaled-down raw data)傳輸至解析度較小(例如:480x234)的數位相框，並執行一次影像縮小的動作，但該動作較不影響整體效能。另外如將解析度較小(例如: 480x234)的數位相框的相片以螢幕尺寸之原始影像資料(Scaled-down raw data)傳至解析度較大(例如:600x480)的數位相框時，則可使用插補點方式，在相臨之像素間插入相似的像素，以避免解析度大幅降低。

(a) 本研究方向未來可應用至無線等傳輸介面，例如無線通用序列匯流排(Wireless USB)、無線(WiFi)與藍牙(Bluetooth)等介面，以達到傳輸更加便利，應用更為廣泛之目的。

參考文獻

[1] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips, "Universal Serial Bus Specification, Revision 2.0", April 27, 2000.

[2] Intel Corporation, "Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 1.0", March 12, 2002.

[3] USB Implementers Forum, "Device Class Definition for Human Interface Device (HID) Firmware Specification, Revision 1.11", June 27, 2001.

[4] USB Implementers Forum, "Universal Serial Bus Mass Storage Class Specification Overview, Revision 1.2", June 23, 2003.

[5] USB Implementers Forum, "Universal Serial Bus Mass Storage Class Control/Bulk/Interrupt (CBI) Transport, Revision 1.1", June 23, 2003.

[6] USB Implementers Forum, "Universal Serial Bus Mass Storage Class Bulk-Only Transport, Revision 1.0", September 31, 1999.

[7] USB Implementers Forum, "Universal Serial Bus Mass Storage Class UFI Command Specification, Revision 1.0", December 14, 1998.

[8] 許永和，"USB2.0高速週邊裝置設計之實務應用"， 全華科技圖書股份有限公司，民國八十七年。

[9] 馮育新，"整合 USB 之嵌入式系統設計"， 國立中正大學， 碩士論文， 民國 91 年。

[10] Hiroaki Takada, "uITRON4.0 Specification, Ver.4.00.00", June, 1999.

[11] MIPS, "MIPS32 Architecture For Programmers", June 9, 2003.