



Application Note

UI Tools & UIFramework & Unicode

Revision History

Revision	Date	Author	Changes
1.0	2014/01/28	KS Hung	Draft
1.1	2014/03/19	KS Hung	Expand 32x32 font to 64x64

Content

Revision History	2
1. UI Desinger Studio 版本的介紹	5
2. UI Tools Install Guide.....	6
2.1. System Requirement	6
2.2. MSXML 4.0 Service Pack Patch	6
2.3. 新舊 UI Tool 衝突問題的解決.....	7
3. UI Tools User Guide	8
3.1. 建立一個新的 Project (.XMP).....	10
3.2. Image (Icon) Tool.....	12
3.3. Palette Tool	19
3.4. String Tool.....	24
3.5. Font Tool.....	33
3.6. Window Tool.....	42
3.6.1 UI Window 簡介及命名規則.....	42
3.6.2 Creat a new window.....	44
3.7. UI Layout 及元件簡介	46
3.7.1 Panel 元件.....	48
3.7.2 Tab 與 Button 元件.....	48
3.7.3 Static 元件	48
3.7.4 Status 元件.....	49
3.7.5 Menu 元件、簡介與程序中 Tab Menu 、擴充性的關係.....	51
3.7.6 List 元件	62
3.7.7 Scroll Bar 元件	63
3.7.8 Slide Bar 元件.....	63
3.7.9 Progress Bar 元件	53
4. UIFramework 簡介	64
4.1. Window Open and Close	64
4.2. Control View/Data/Event.....	65
4.3. Redraw	66
4.4. Send/Post/Flush Event	67
4.5. State Machine 的應用.....	68
5. Unicode 簡介	69
5.1. Tool for Unicode	69
5.2. PStore for Unicode.....	73
5.3. Unicode display in old drawing library	73
5.4. 使用視窗介面程式挑選出 Font 並轉成 bitmap.....	74

Novatek Confidential

1. UI Desinger Studio 版本的介紹

目前在 220 或是 632 (ARM)平台上開發 UI 的 Designer Studio 應該都是如圖 1.1 的版本，而圖 1.2 則是用在 650 (MIPS)平台。兩個版本是都可以互用，不過圖 1.1 的版本要用在 650 平台上，還要手動改些由 UI Tools 產生的代碼，這些檔案是 DemoKit_Palette.c 、 DemoKit_Image.c 、 DemoKit_Font.c 、 DemoKit_String_XX(各國語言).c ，都要手動去 `#include "Platform.h"` 及將 `_align(4)` 改成 `_ALIGNED(4)` 才能編譯過，因此不是很方便，故建議兩個平台各自使用。只要先按該執行檔，下次開啓 UI project (*.XMP) ，即會用該執行檔打開。

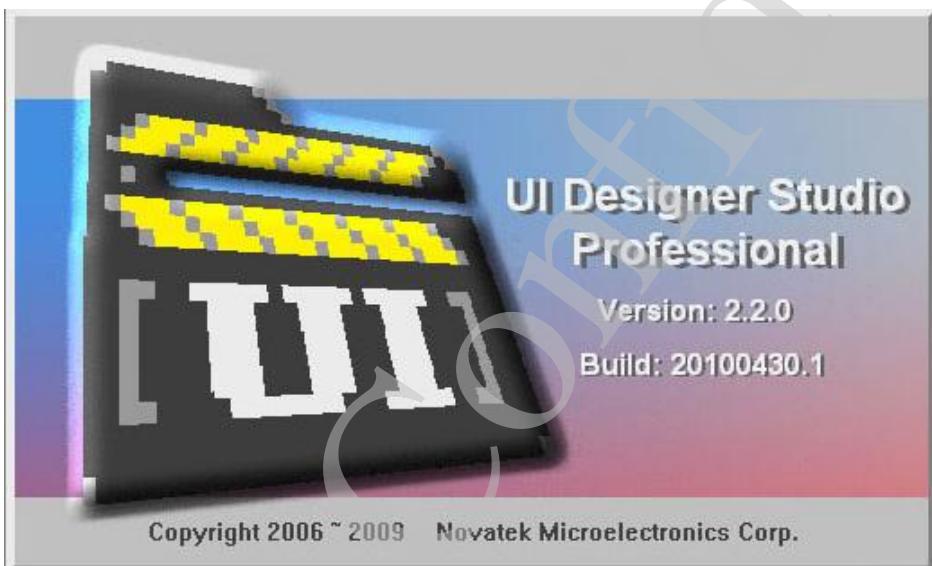


圖 1.1



圖 1.2

2. UI Tools Install Guide

2.1 System Requirement

目前這套 UI Tools 最常用的作業系統，大多在 Windows XP SP3 或是 Window 7，Windows 2000、Window Vista、Window 8 沒有測試過，但只要是 Windows 系統，基本上應該是可以的，關於需要安裝的 Patch，請參考 Chapter 2.2 的 MSXML Service Pack Patch。至於硬體方面，以現今電腦的硬體規格，大多不是問題。

2.2 MSXML 4.0 Service Pack Patch

如果電腦是 Window 2000、Windows XP (SP3 以下)、Windows 7，請安裝 MSXML 4.0 Service Pack 3 的 Patch，網址如下：

■ <http://www.microsoft.com/zh-tw/download/details.aspx?id=15697>

連接後的網站如圖 2.2.1 所示，Windows 2000、XP、Vista、7 都有支援，請選擇不同語言下載。

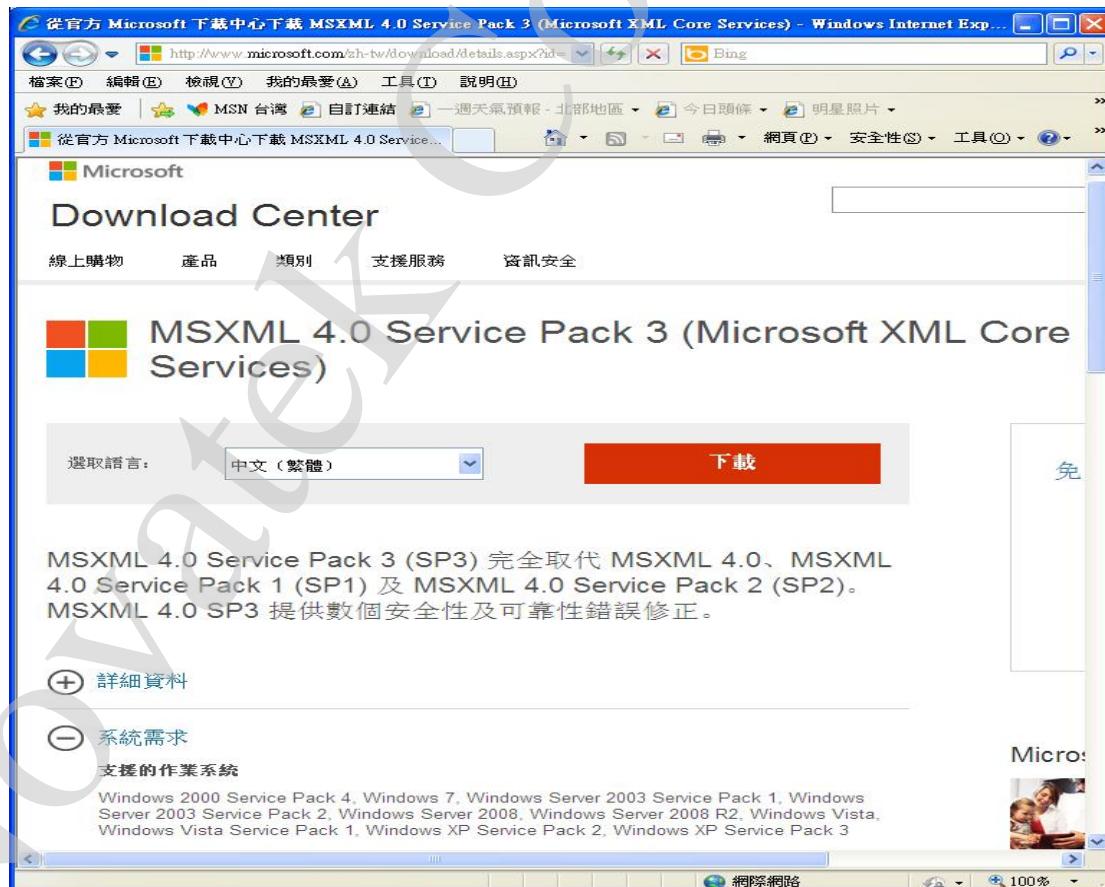


圖 2.2.1

安裝完畢後可檢查 Windows 檔案之版本的正確性：

- 檔名: **\WINDOWS\SYSTEM32\MSXML4.dll** (按滑鼠右鍵看檔案的內容), 如圖 2.2.2 所示, 如果電腦中沒有此檔案或是版本比這個還舊, 就需要安裝。
- **如果沒有更新**, 可能會遇到開檔/存檔當機的問題。

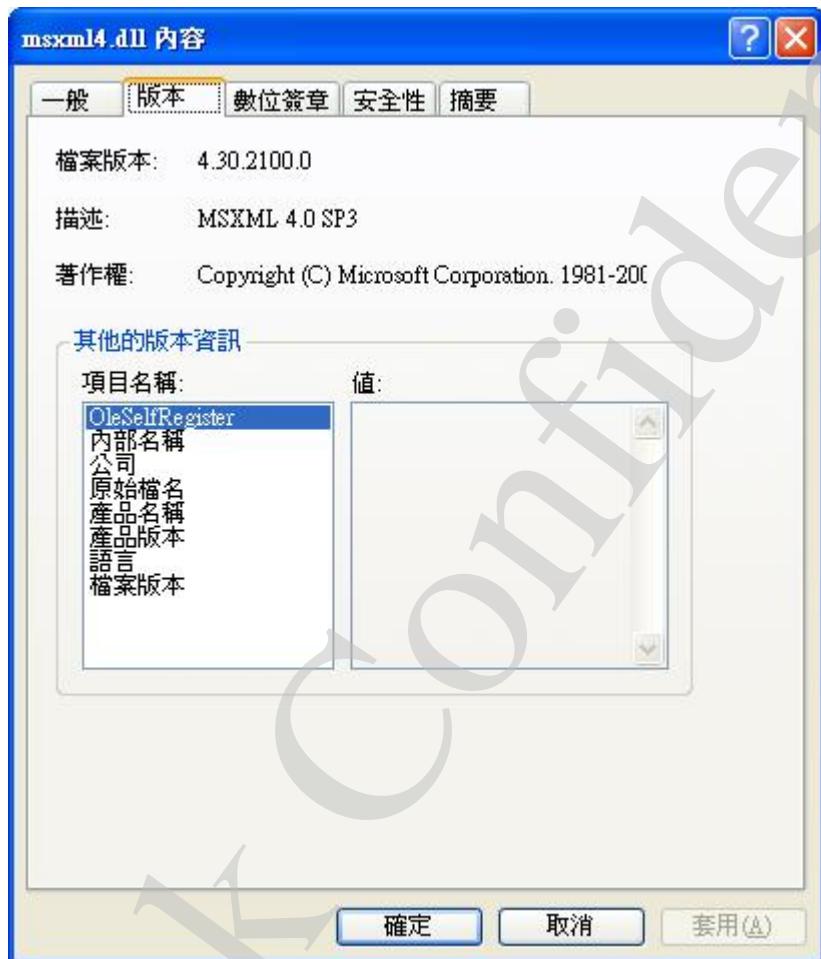


圖 2.2.2

2.3 新舊 UI 衝突問題的解決

早期的舊版 UI Tool，只能用來將 Image、Palette、String、Font 轉成 C 程序，無法畫 UI Layout，因此當你開啟新版 UI Tool 時，如果發現此工具的視窗 Layout 有亂掉的情形，或是 UI Tool 會出現錯誤訊息，然後關閉應用程式，表示該電腦可能有安裝過舊版的 UI Tool，請按下列方式處置：

- 關閉正在執行的 UI Tool
- 執行 Windows 的登錄編輯器：在桌面的左下角，選”開始\執行”，輸入”regedit”，然後按”確定”，如圖 2.3.1 和圖 2.3.2 所示。



圖 2.3.1



圖 2.3.2 輸入"regedit"開啓登錄編輯器

■ 當開啟如圖 2.3.3 的登入編輯器時，請刪除 UI_Designer 這個註冊碼，然後重新執行 UI Tools 即可。

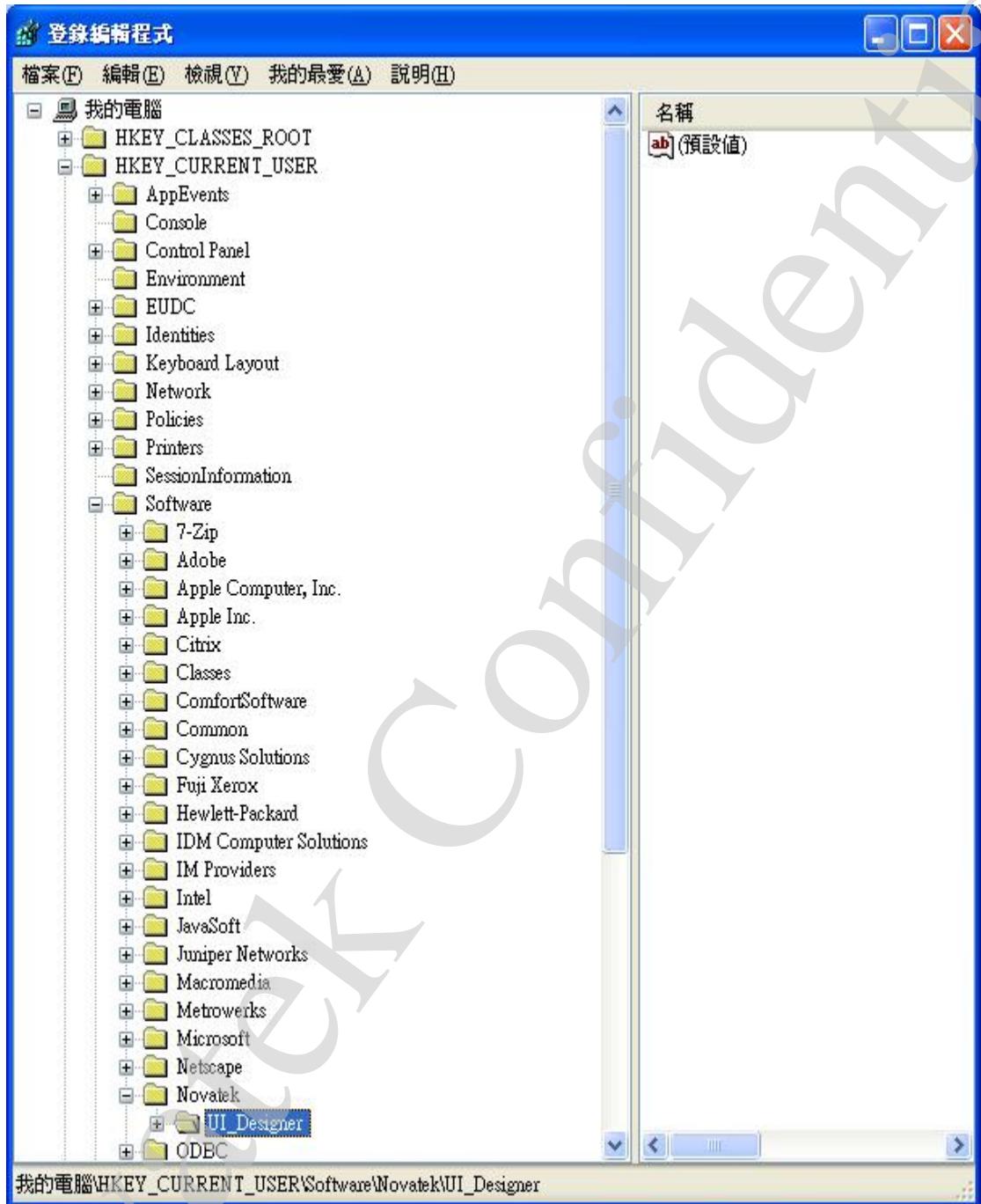


圖 2.3.3 登入編輯器，請刪除 UI_Designer 註冊碼

3. UI Tools

3.1 建立一個新的 Project (.XMP)

- Select “File\New...” (Under Resource Edit Mode)

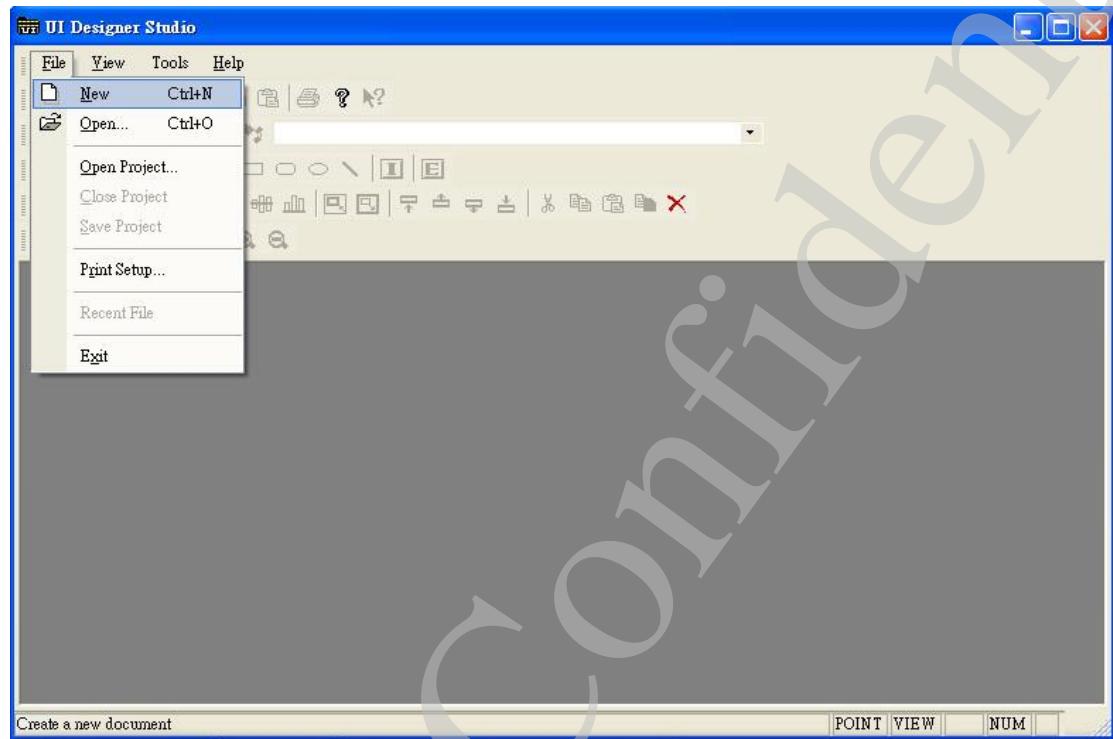


圖 3.1.1

- Press OK ◊ then, new project is “DemoKit.XMP”

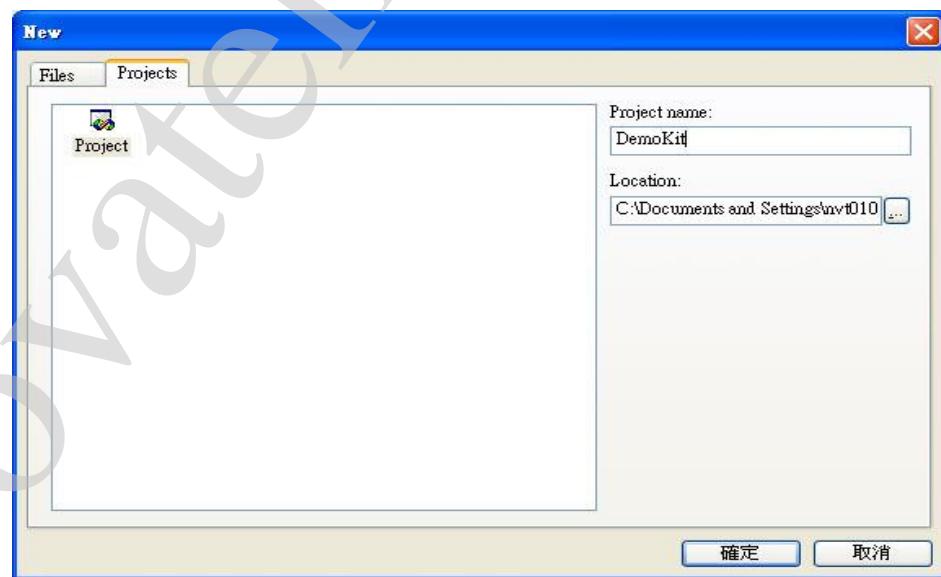


圖 3.1.2

- 建立一個 XMP project 之後會產生 DemoKit.XMP、DemoKit_Control.xmo、DemoKit_Event.xme、DemoKit_Font.xmf、DemoKit_Image.xmb、DemoKit_Palette.xmc、DemoKit_String.xms 這些檔案。
- DemoKit_Control、DemoKit_Event、DemoKit_Icon、DemoKit_Palette、DemoKit_String 這些文件夾是另外手動建立的，它是用來存放 Icon、String、JPG 背景圖、及轉成 C 語言存放的路徑。
- 其中 DemoKit_Font.xmf、DemoKit_Image.xmb、DemoKit_Palette.xmc、DemoKit_String.xms 會在後續的章節介紹如何轉成 Code



圖 3.1.3

3.2 Image (Icon) Tool

這邊說明 Image Tools 的使用，以及如何轉成 code。

- Select “File\New...” (Under Resource Edit Mode)

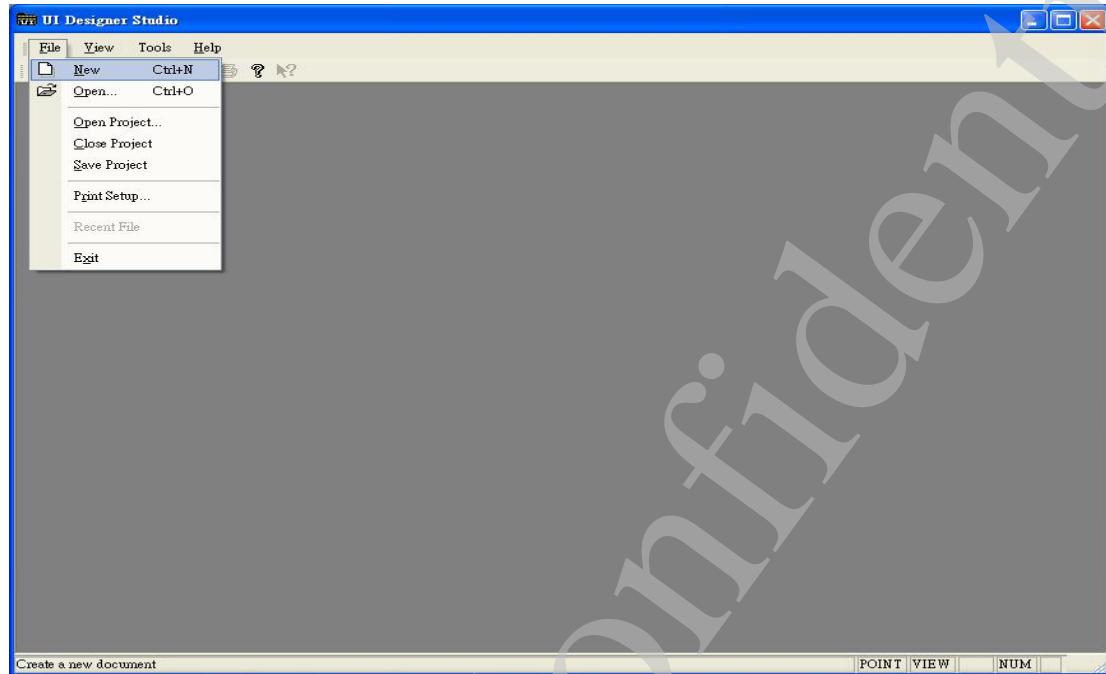


圖 3.2.1

- Choose Image Table for a new document
- Specify filename and file location for it
- Press OK ➔ then, new file is “DemoKit_Image.xmb”

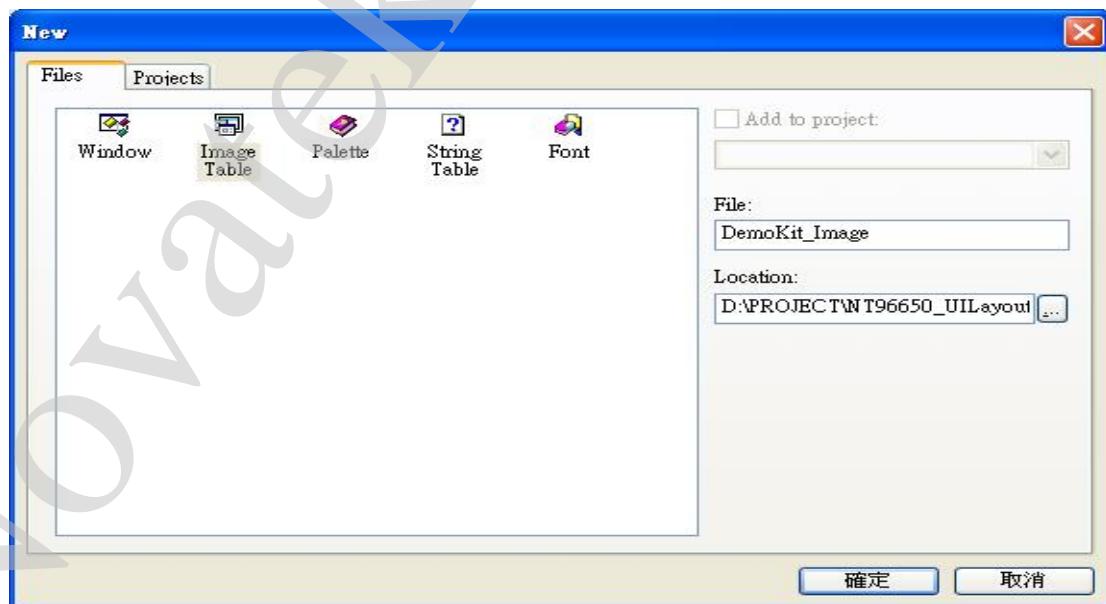


圖 3.2.2

■ After Create Image Table

- 在建立一個 XMP project 後，就會產生 XXXX_Image.xmb，圖 3.2.1 和圖 3.2.2 是說明當 xmb 不存在時，如何建立新的 xmb

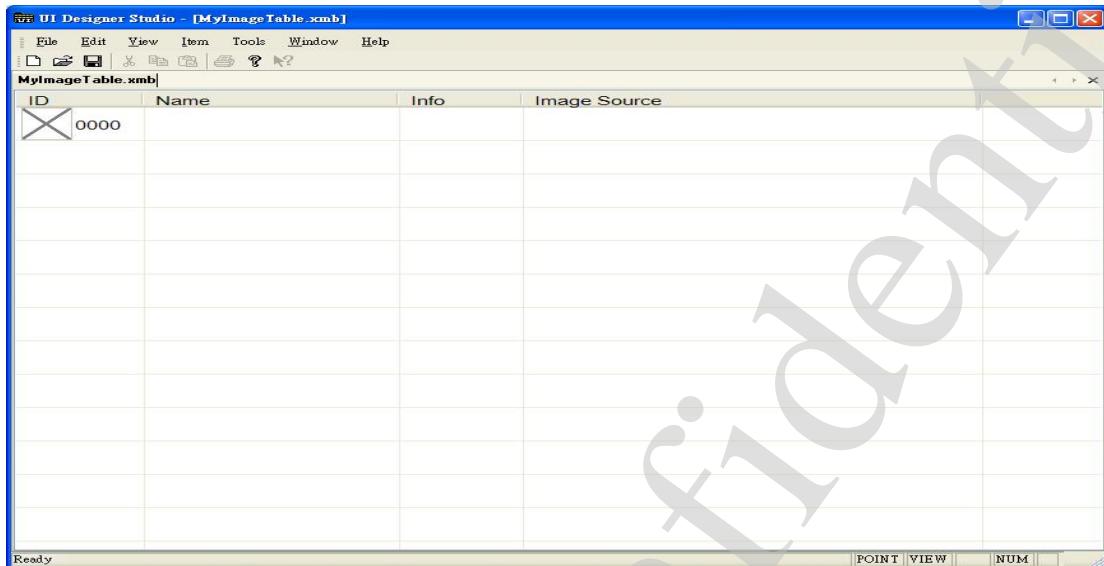


圖 3.2.3

■ Select “File\Import Source\From BMP/PNG/JPG files...”

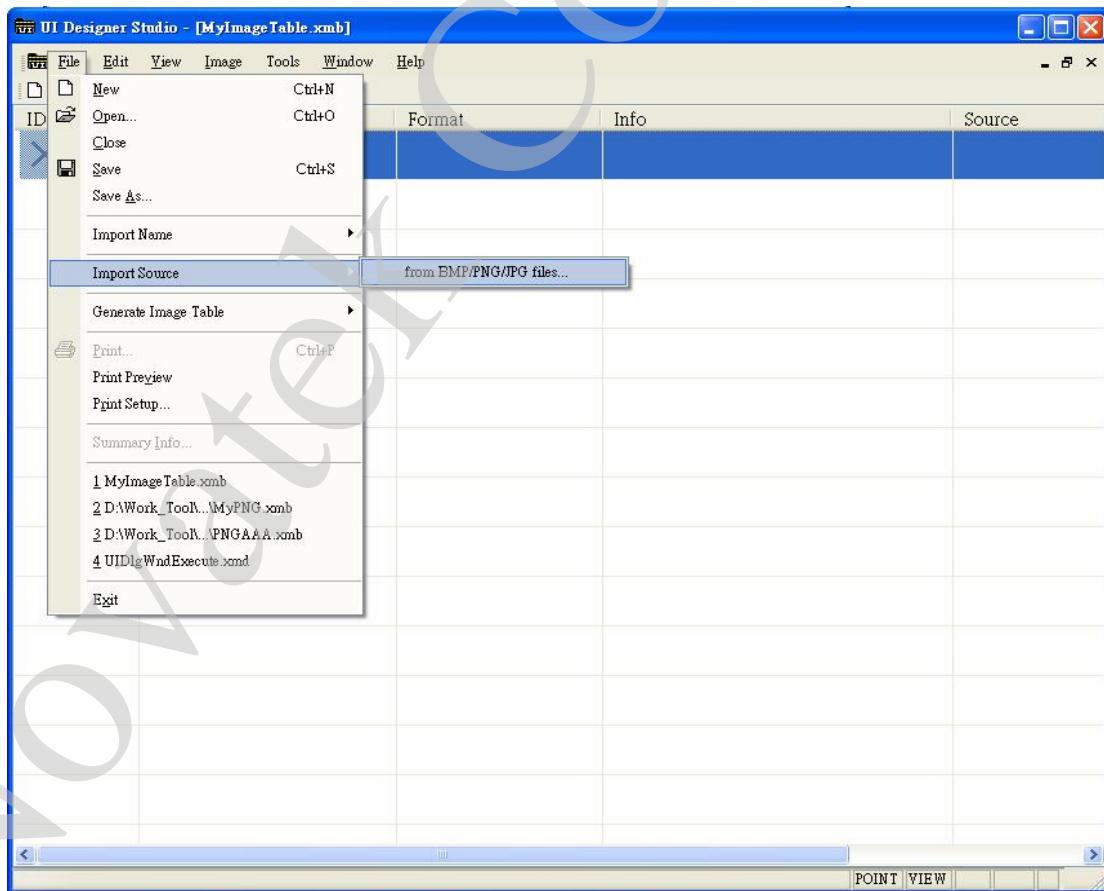


圖 3.2.4

- 請先做一個 Empty Icon，大小為 1x1 即可，顏色為調色盤指定的透明色。
- Select your input image bitmaps and press OK.
- If you want to export a 1/2/4/8 bpp image table, please load image from BMP file. (input BMP must in 8 bpp format)
- If you want to export a 24 bpp image table, please load image from JPG file.



圖 3.2.5

- 圖 3.2.6 顯示載入時的設定。”Ignore warning & error message”建議不要勾選，畢竟有這些訊息出現，表示可能有問題。

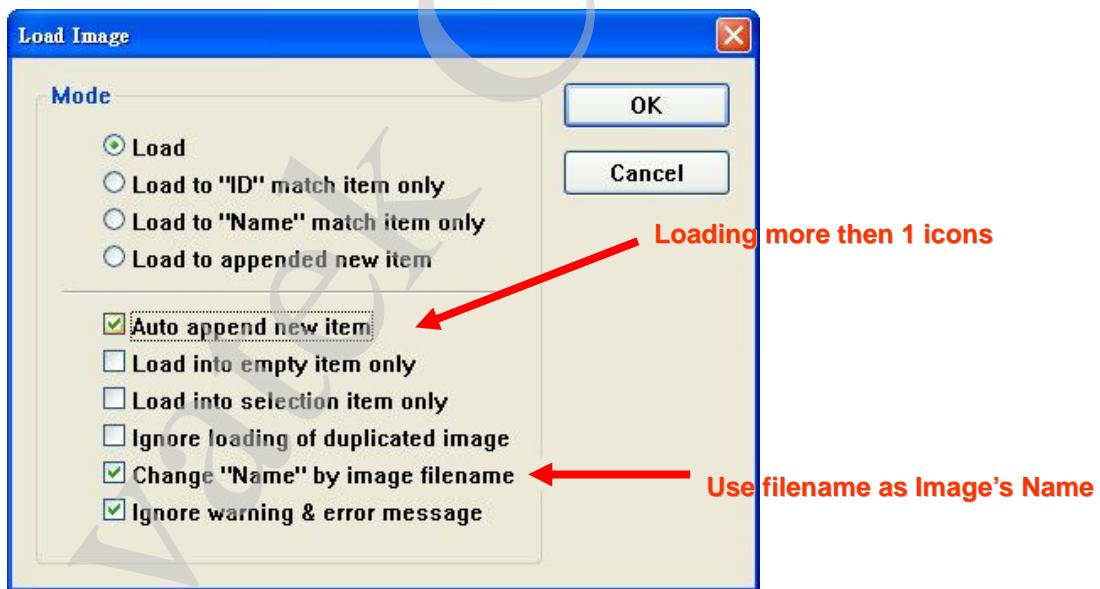
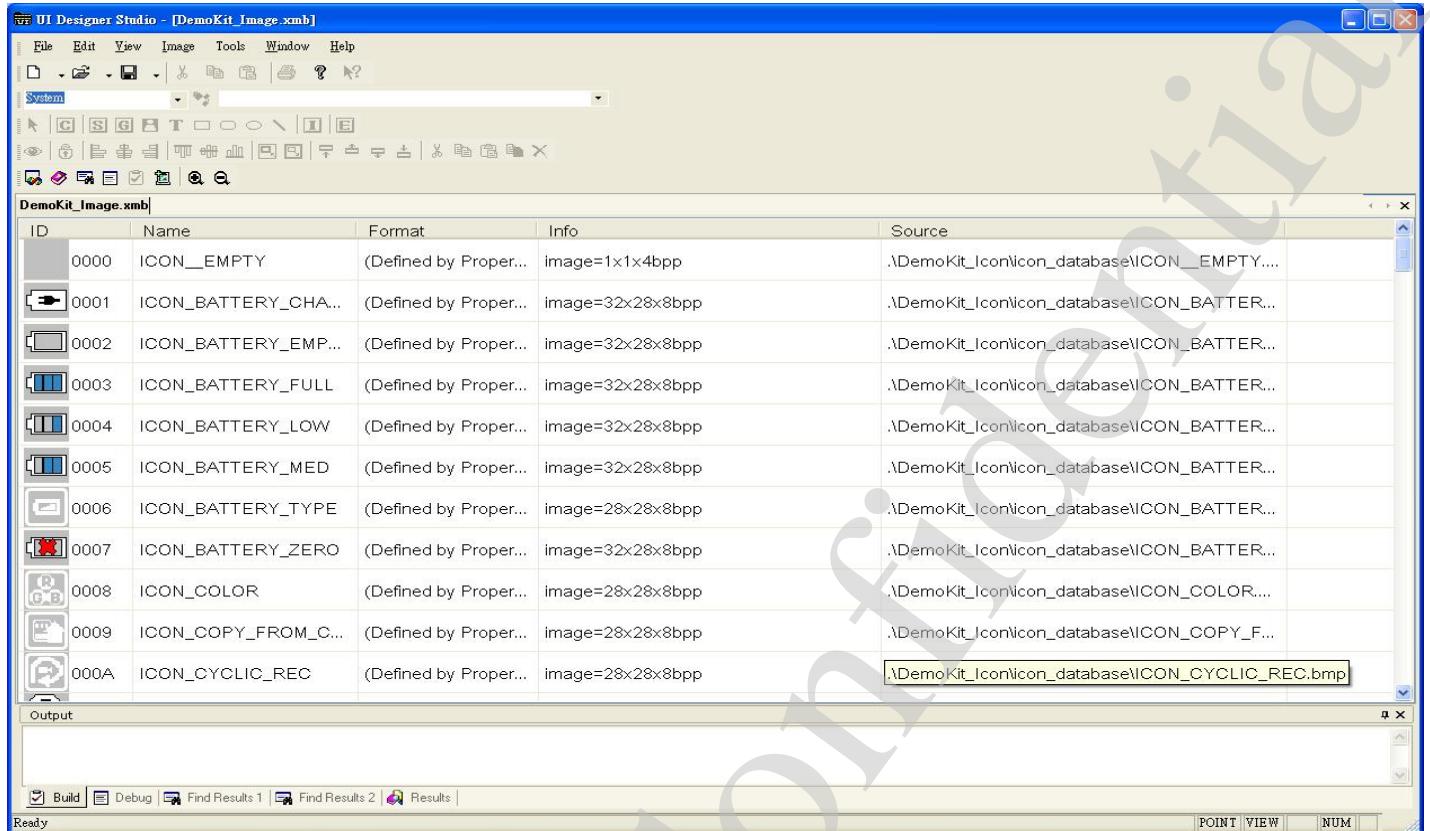


圖 3.2.6

■ After import source images



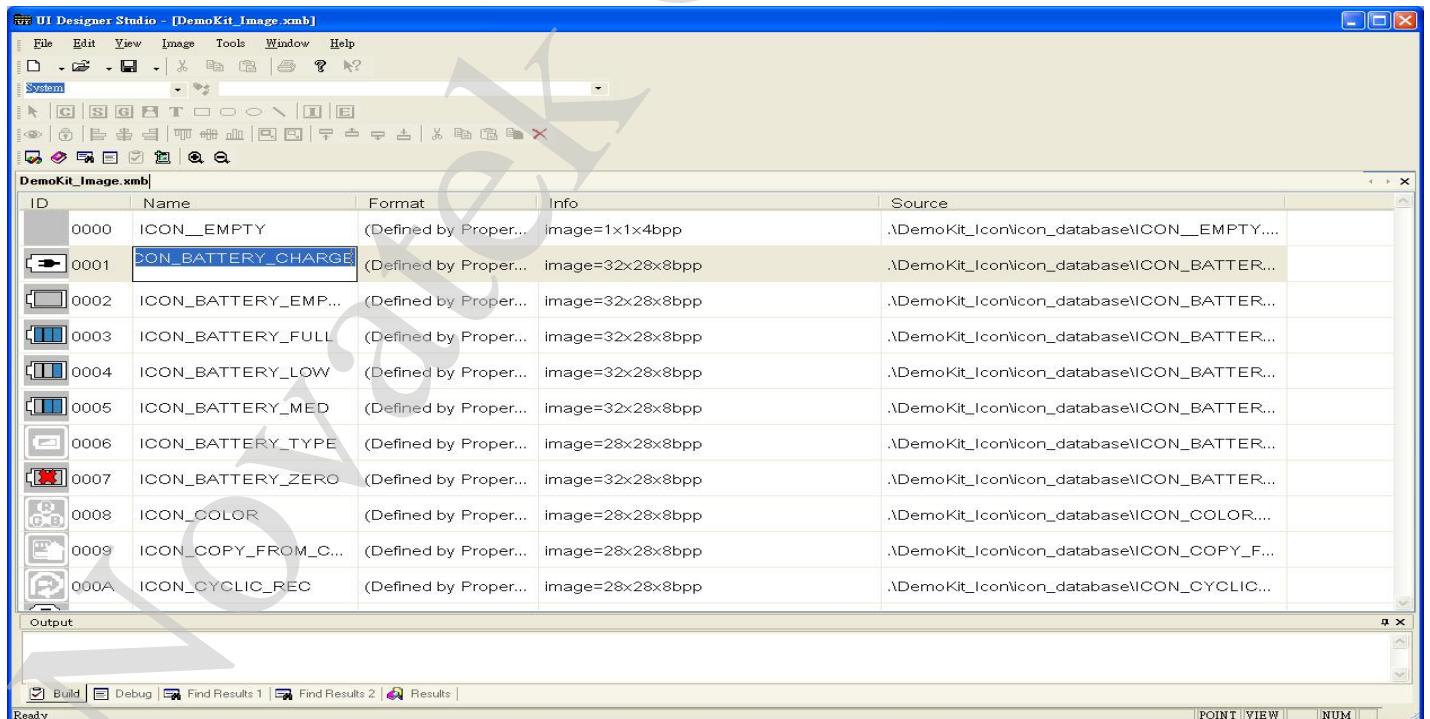
The screenshot shows the UI Designer Studio interface with the title bar "UI Designer Studio - [DemoKit_Image.xmb]". The menu bar includes File, Edit, View, Image, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Print. The main window displays a table titled "DemoKit_Image.xmb" with columns: ID, Name, Format, Info, and Source. The table lists 11 icons:

ID	Name	Format	Info	Source
0000	ICON_EMPTY	(Defined by Proper...)	image=1x1x4bpp	.\DemoKit_Icon\icon_database\ICON_EMPTY....
0001	ICON_BATTERY_CHA...	(Defined by Proper...)	image=32x28x8bpp	.\DemoKit_Icon\icon_database\ICON_BATTER...
0002	ICON_BATTERY_EMP...	(Defined by Proper...)	image=32x28x8bpp	.\DemoKit_Icon\icon_database\ICON_BATTER...
0003	ICON_BATTERY_FULL	(Defined by Proper...)	image=32x28x8bpp	.\DemoKit_Icon\icon_database\ICON_BATTER...
0004	ICON_BATTERY_LOW	(Defined by Proper...)	image=32x28x8bpp	.\DemoKit_Icon\icon_database\ICON_BATTER...
0005	ICON_BATTERY_MED	(Defined by Proper...)	image=32x28x8bpp	.\DemoKit_Icon\icon_database\ICON_BATTER...
0006	ICON_BATTERY_TYPE	(Defined by Proper...)	image=28x28x8bpp	.\DemoKit_Icon\icon_database\ICON_BATTER...
0007	ICON_BATTERY_ZERO	(Defined by Proper...)	image=32x28x8bpp	.\DemoKit_Icon\icon_database\ICON_BATTER...
0008	ICON_COLOR	(Defined by Proper...)	image=28x28x8bpp	.\DemoKit_Icon\icon_database\ICON_COLOR....
0009	ICON_COPY_FROM_C...	(Defined by Proper...)	image=28x28x8bpp	.\DemoKit_Icon\icon_database\ICON_COPY_F...
000A	ICON_CYCLIC_REC	(Defined by Proper...)	image=28x28x8bpp	.\DemoKit_Icon\icon_database\ICON_CYCLIC_REC.bmp

The bottom status bar shows "Ready", "POINT", "VIEW", and "NUM".

圖 3.2.7

■ Left-click on “Name” column and edit its name



This screenshot is identical to Figure 3.2.7, showing the UI Designer Studio interface with the same table of icons. However, the second row's "Name" cell, which previously contained "ICON_BATTERY_CHA...", now has "CON_BATTERY_CHARGE" typed over it. The rest of the table and the interface remain the same.

圖 3.2.8

■ Setup properties of Image Table

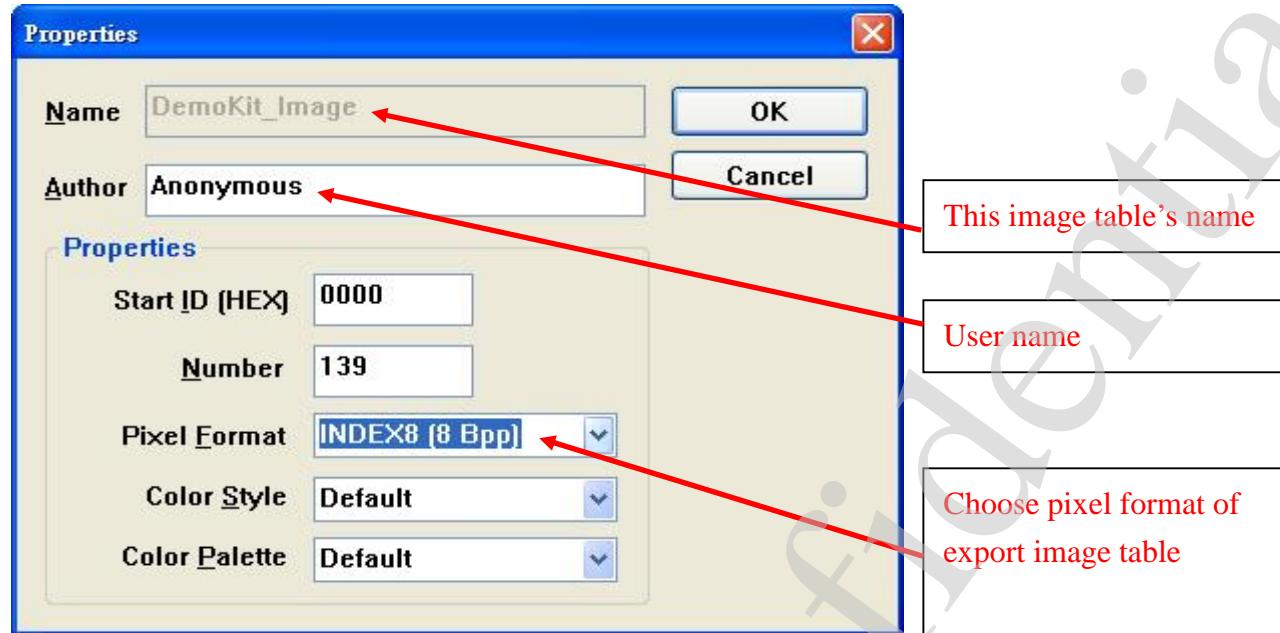


圖 3.2.9

■ Select “File\Save” to save current Image Table (*.xmb)

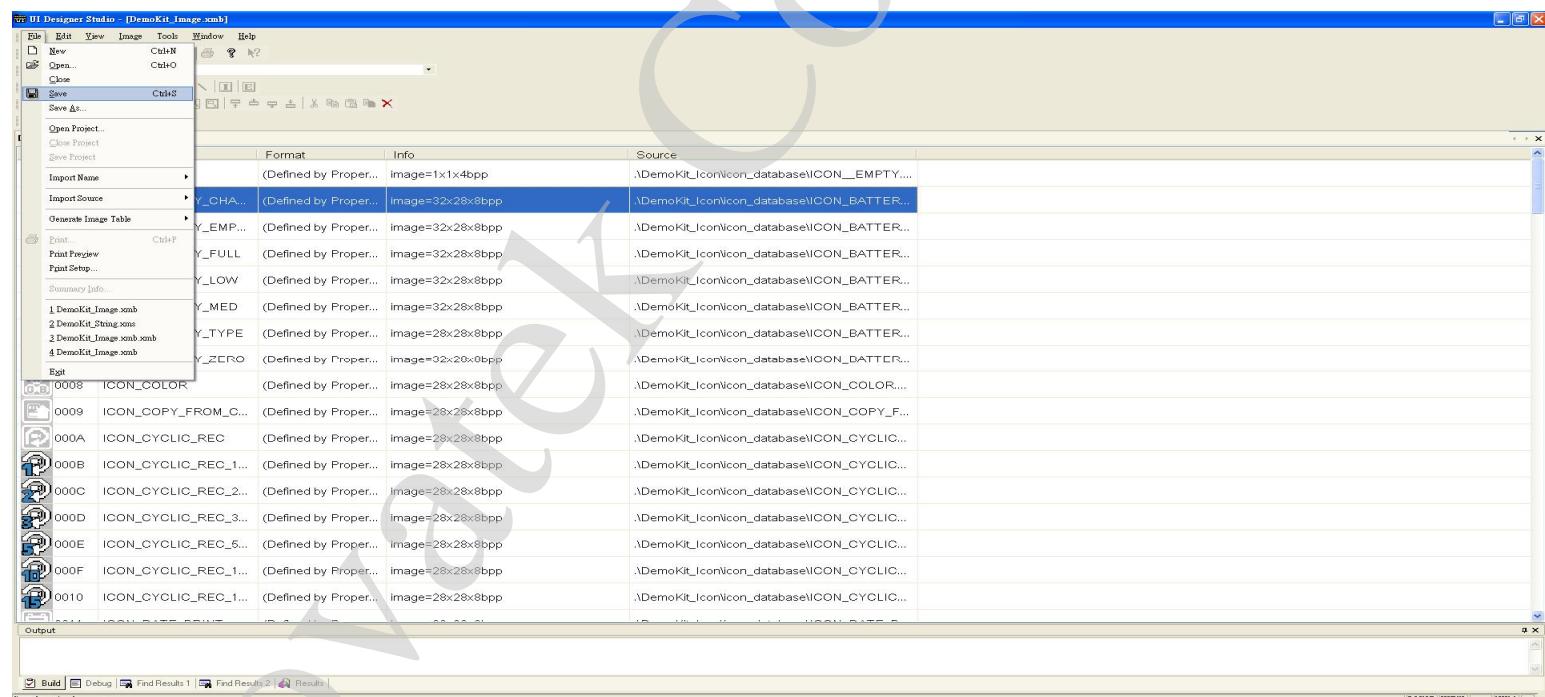


圖 3.2.10

- Select “File\Generate Image Table\in C Language IMAGE_TABLE format (NT96450, NT96630 and after)…”

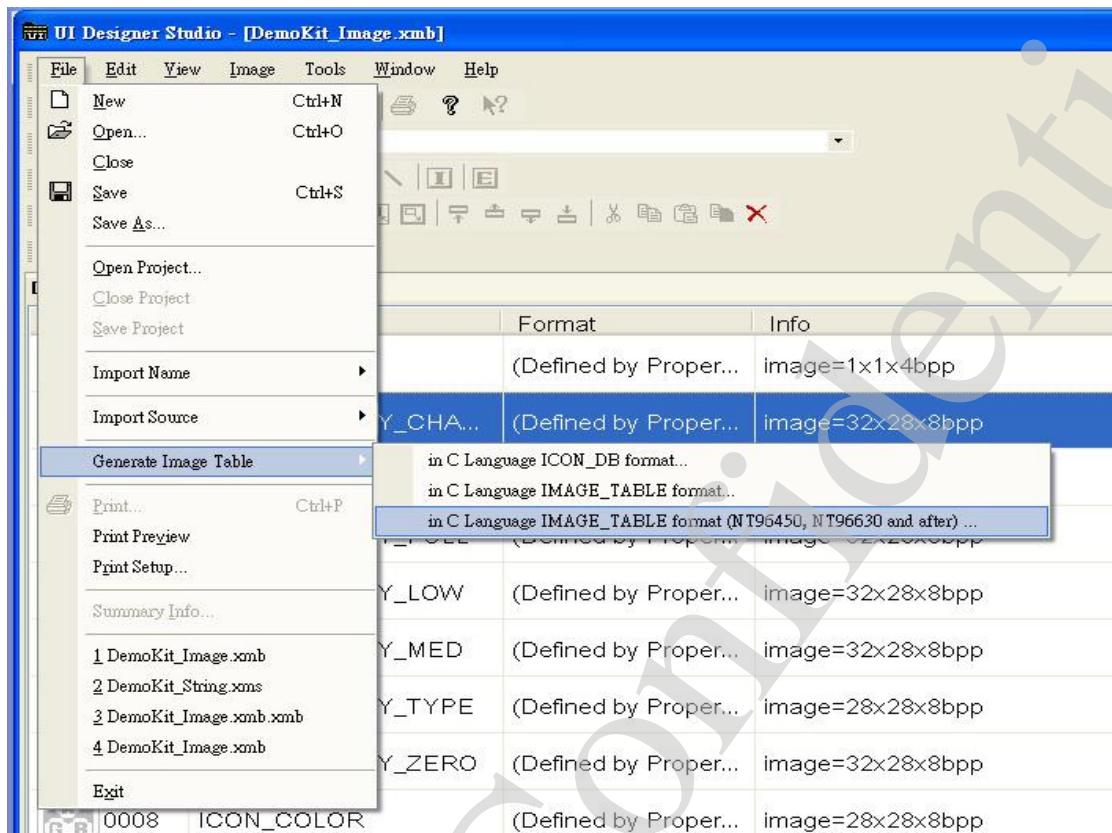


圖 3.2.11

- Export to *.c, *.h and *.bin files
- The bin file is a binary format IMAGE_TABLE. 通常不想讓 FW size 變大，就會將這個 bin file 儲存至內存 flash memory，要用時在讀取出來放至 DRAM 使用。

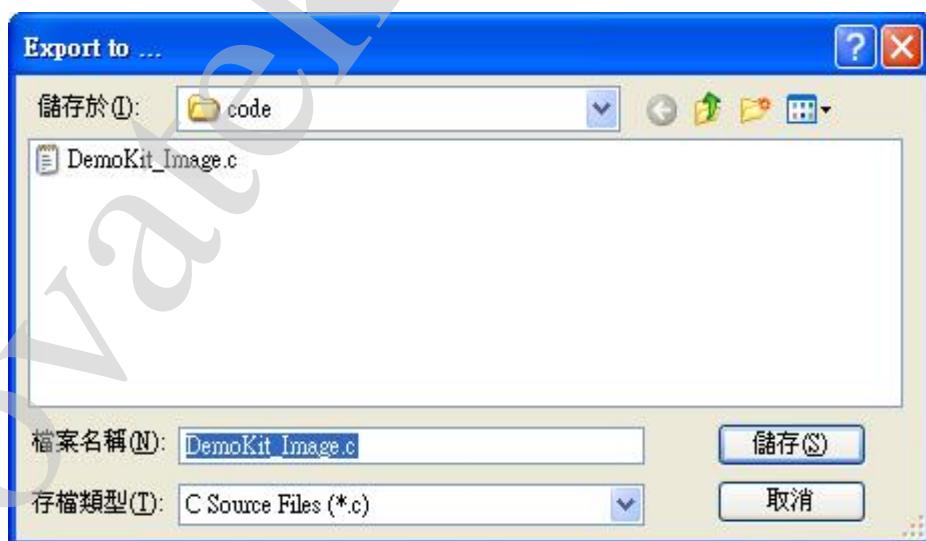


圖 3.2.12

- To use this image table in GxGfx:
 - (1) Add DemoKit_Image.h and DemoKit_Image.c to your project
 - (2) #include “DemoKit_Image.h”
 - (3) GxGfx_SetImageTable((const IMAGE_TABLE*) gDemoKit_Image);
- Question: Why UIDS say image is loading failed when I load a image table from image table file (*.xmb)?
- Answer: UIDS is not support UNICODE file path name, please do not place image table file (*.xmb) at the path with UNICODE name.

3.3 Palette Tool

這邊說明 Palette Tools 的使用，以及如何轉成 code。

- Select “File\New...” (Under Resource Edit Mode)

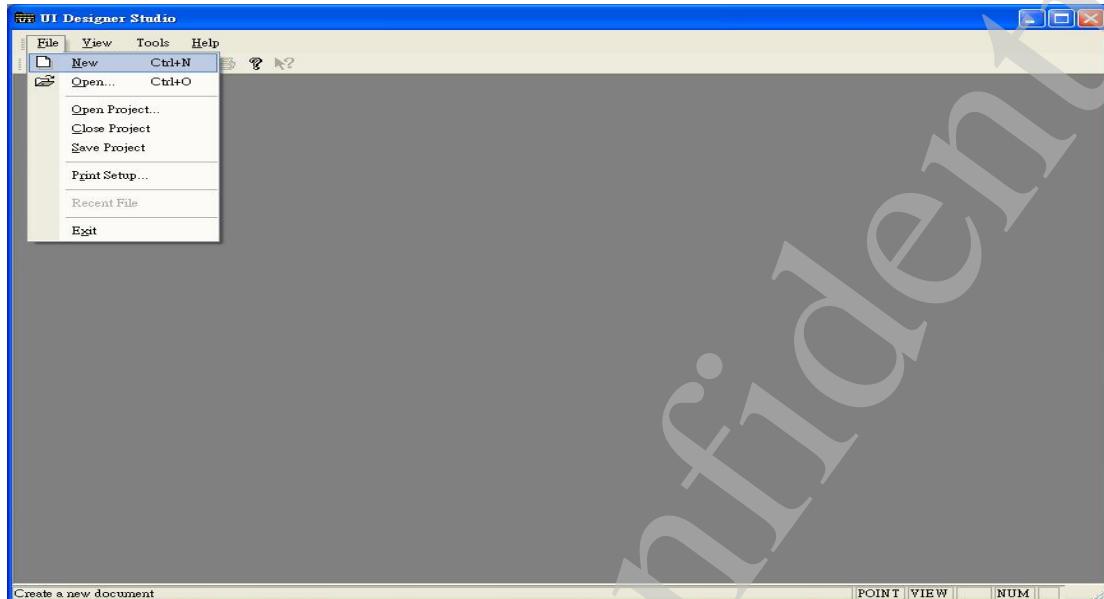


圖 3.3.1

- Choose “Palette” for a new document
- Specify filename and file location for it
- Press OK ➔ then, new file is “DemoKit_Palette.xmc”

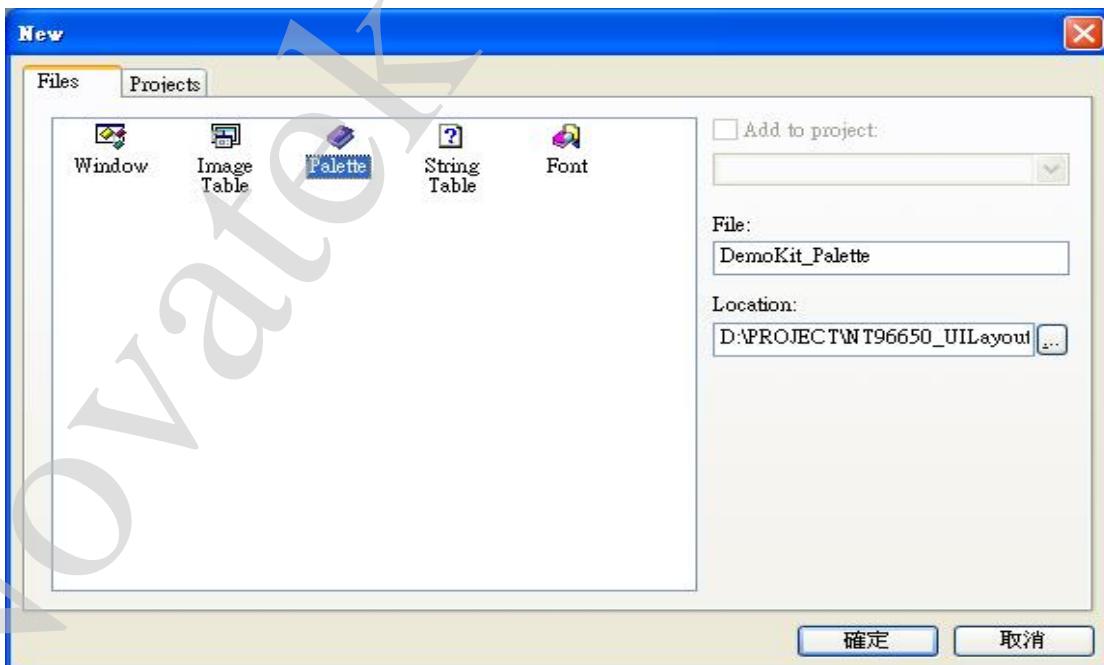


圖 3.3.2

■ After create Palette

- 在建立一個 XMP project 後，就會產生 XXXX_Palette.xmc，圖 3.3.1 和圖 3.3.2 是說明當 xmc 不存在時，如何建立新的 xmc

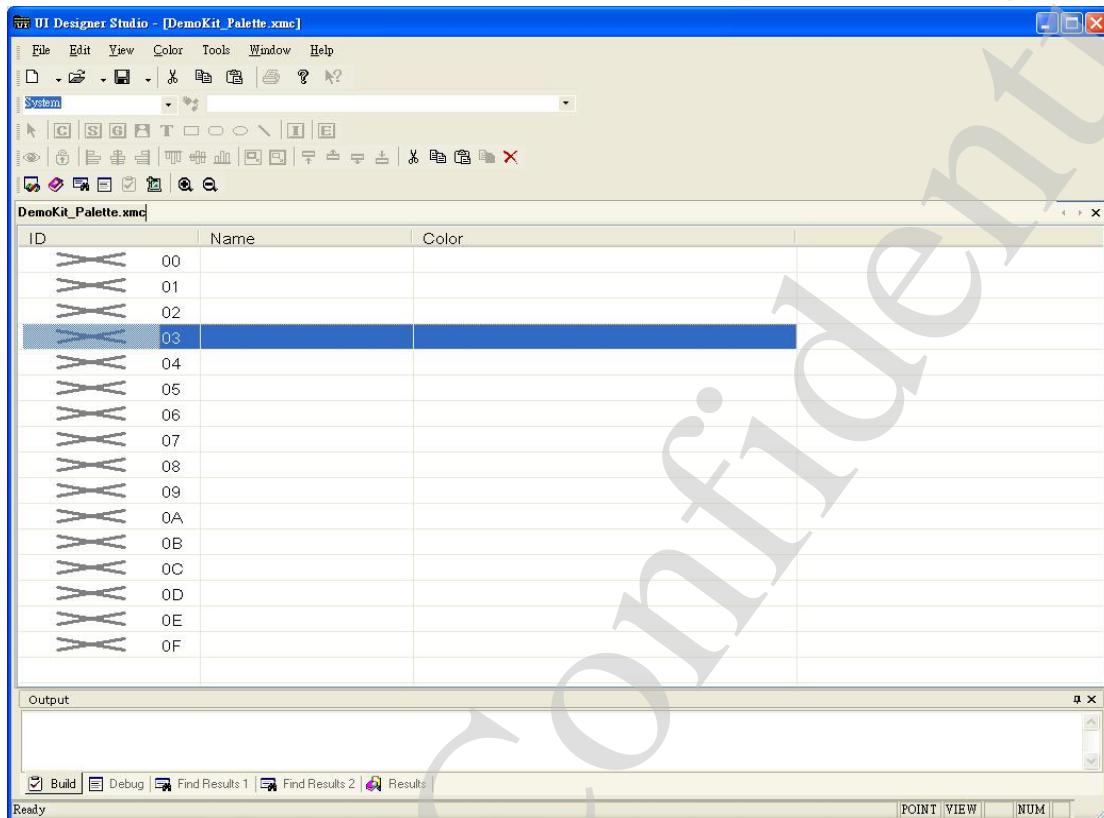


圖 3.3.3

- Right Click and select “Properties...”
- 一般選擇 256 顏色，除非使用的全部 icon 顏色低於 16 色 (可減少 code size)。
- 在選擇 256 顏色後，圖 2.15 的左邊 ID 會變成 0xFF

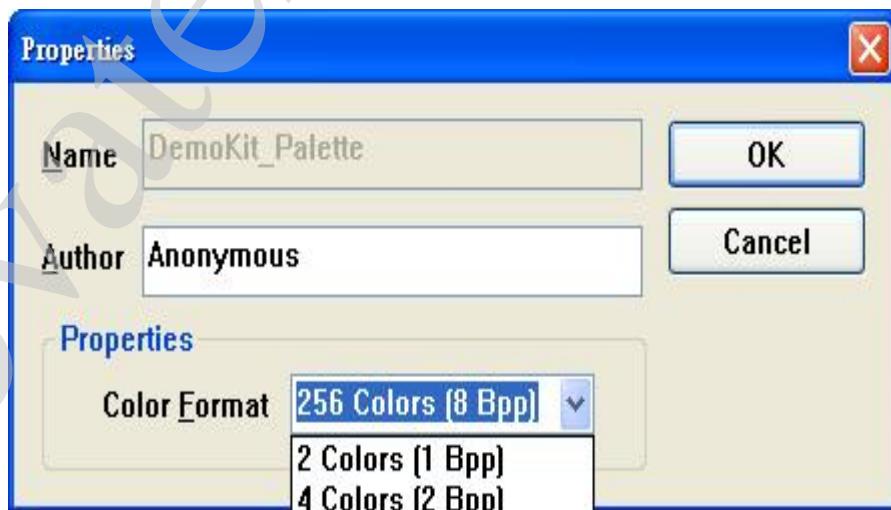


圖 3.3.4

- Select “File\Import Color\from BMP file ...”

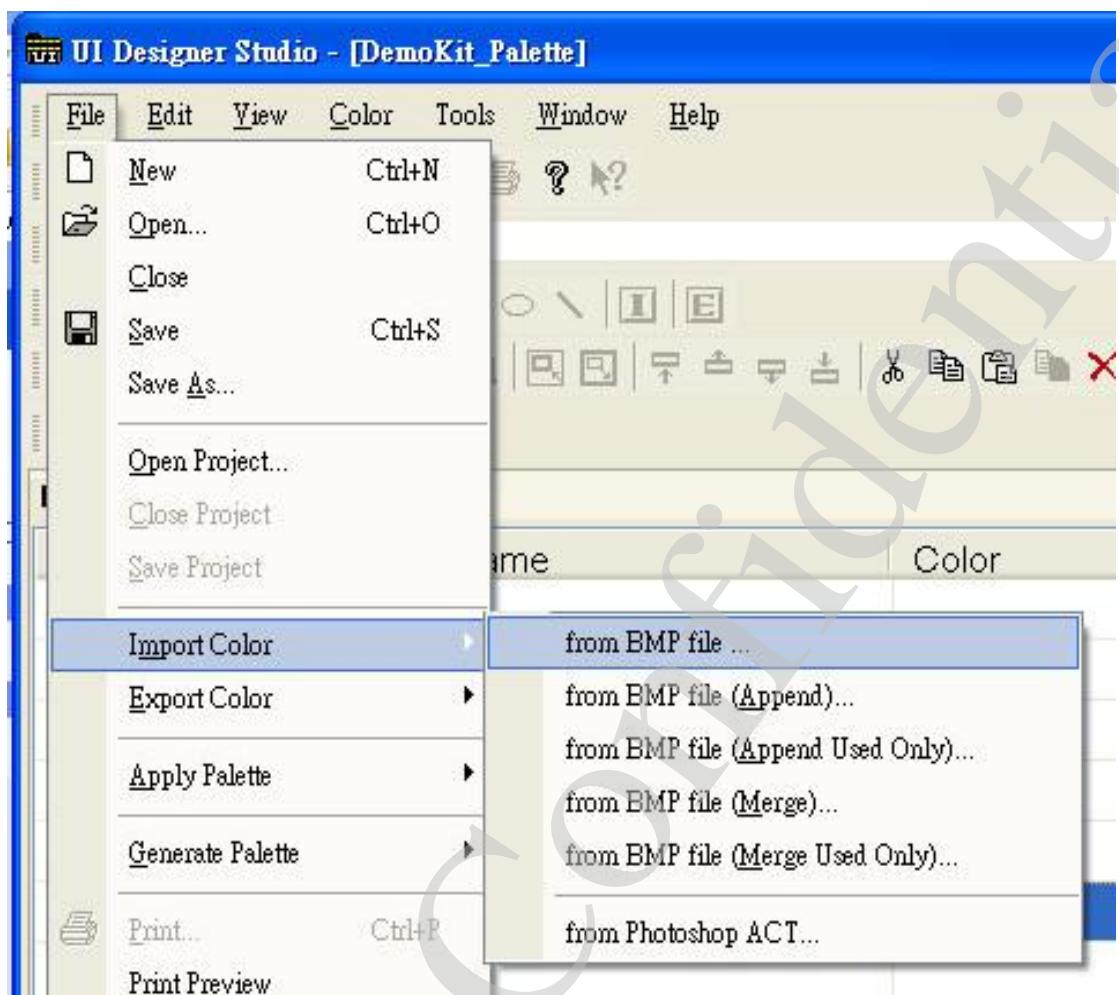


圖 3.3.5

- Click any one icon and load it

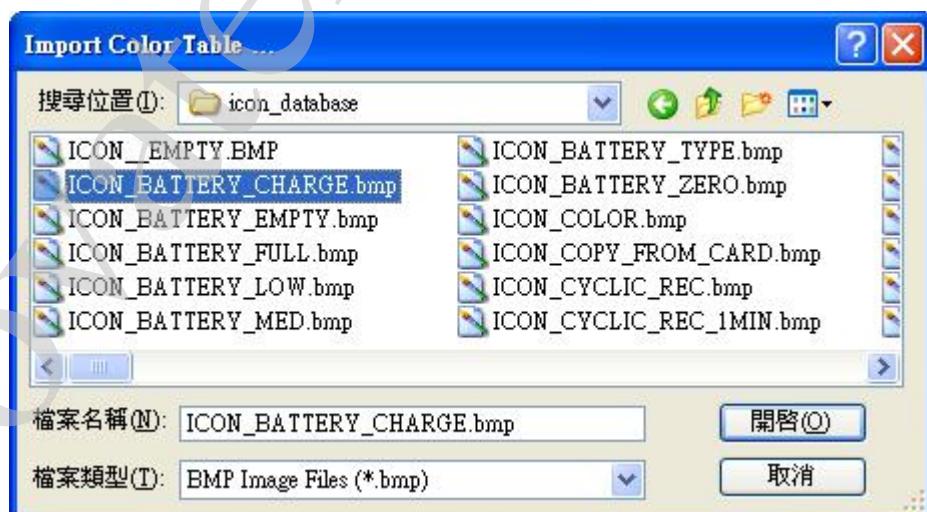


圖 3.3.6

■ After import BMP file

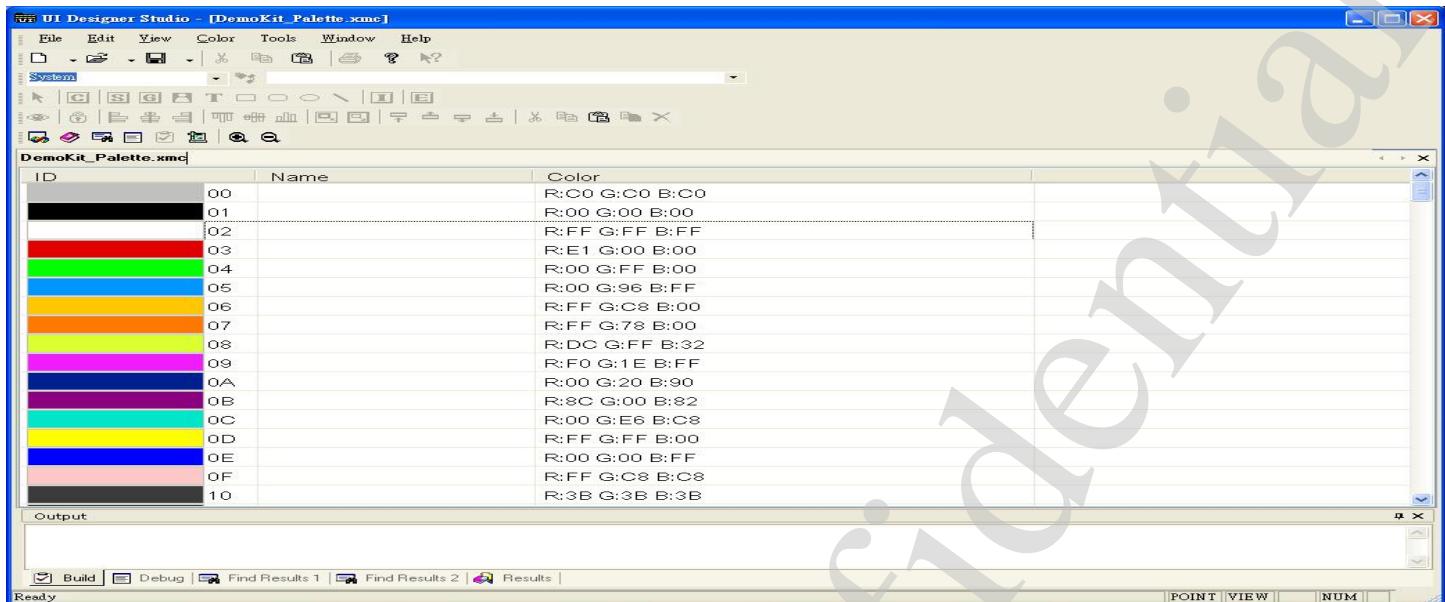


圖 3.3.7

■ Select “File\Generate Palette\in C Language PALETTE (YUVA8888) format (NT96450, and after)...”

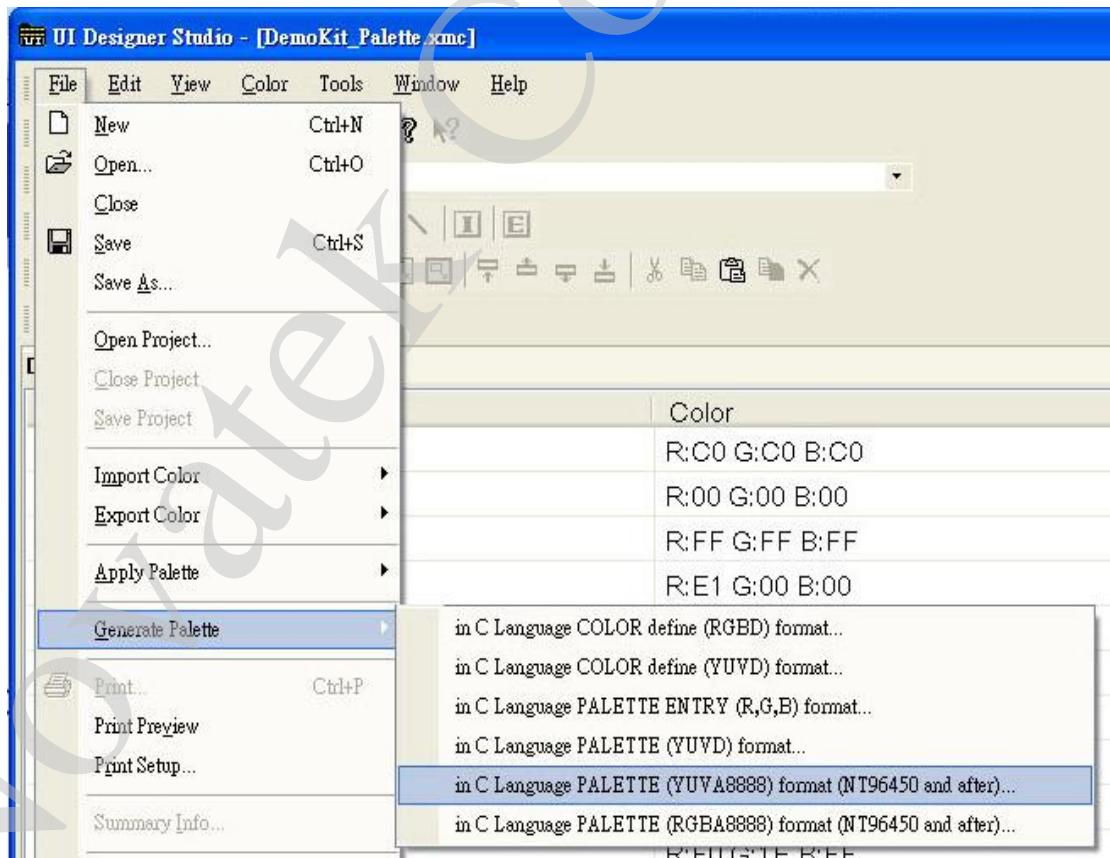


圖 3.3.8

■ To use this image table in GxGfx:

- (1) Add DemoKit_Palette.h and DemoKit_Palette.c to your project
- (2) #include “DemoKit_.h”
- (3-1)220 是 GxDisplay_SetPalette (LAYER_OSD1,0,256,gDemoKit_Palette_Palette);
- (3-2)650 是 UI_SetDisplayPalette(LAYER_OSD1,0,256,gDemoKit_Palette_Palette);

3.4 String Tool

這邊說明 String Tools 的使用，以及如何轉成 Code。

- 圖 3.4.1 是 excel 的字串表，如果需要特殊字元，可以在字串表最後新增一個 Dummy String。
- 首先要先將 excel 另存成圖 3.4.3 的 Unicode 的文字檔。
- Keep first row as language's name, and keep first column as string's name
- Select “File\Save as ..” to save this EXCEL document as an UNICODE text file (*.txt)
- 轉出 Unicode text file 後要關閉 excel，否則 String Tool 無法轉出 code。

Language's Name

A1	B	C	D	E	F	G	H	I
1 ID_XXXX	ENGLISH	FRENCH	DEUTSCH	SPANISH	ITALIAN	CHTRAD	CHSIMP	
2 IDS_NULL	0x01	0x01	0x01	0x01	0x01	0x01	0x01	
3 IDS_LANGUAGE	ENGLISH	FRANÇAIS	DEUTSCH	ESPAÑOL	ITALIANO	繁體中文	簡體中文	2 Byte
4 IDS_MENU_EXIT	Exit	Exit	Exit	Salir	Esci	Exit	Esci	
5 IDS_MENU_EXIT_2	xit	Quitter:	Ende:	Salir:	Esci:	結束:	退出:	
6 IDS_MENU_BASIC	et-up1	Régl. 1	Set-up 1	Ajuste 1	Setup 1	設定 1	设置 1	
7 IDS_MENU_CUSTOM	et-up2	Régl. 2	Set-up 2	Ajuste 2	Setup 2	設定 2	设置 2	
8 IDS_MENU_RESET	et-up3	Régl. 3	Set-up 3	Ajuste 3	Setup 3	設定 3	设置 3	
9 IDS_MENU_PICTURE	Mode1	Mode 1	Modo 1	Modo 1	Modo 1	模式 1	模式 1	
10 IDS_MENU_FUNCTION	Mode2	Mode 2	Modo 2	Modo 2	Modo 2	模式 2	模式 2	
11 IDS_MENU_AEAWB	Mode3	Mode 3	Modo 3	Modo 3	Modo 3	模式 3	模式 3	
12 IDS_MENU_PLAYBACK	playback1	Lecture1	Wiedergab	Reproducc	Visualizz	播放1	回放1	
13 IDS_MENU_PLAYBACK2	playback2	Lecture2	Wiedergab	Reproducc	Visualizz	播放2	回放2	
14 IDS_MENU_DELETE	Delete	Supprimer	Löschen	Elimina	Minimare	刪除	删除	
15 IDS_MENU_VIDEO	Video	Vidéo	Video	Video	Filmati	視訊	视频	
16 IDS_MENU_AUTO	Auto	Auto	Auto	Auto	Auto	自動	自动	
17 IDS_MENU_SCENE	Scene	PICT	BILD	PICT	PICT			
18 IDS_MENU_AV	Av	Blendenwe	Va	Av	Av	Av	Av	
19 IDS_MENU_TV2	v	Tv	Zeitwert	Vt	Tv	Tv	Tv	
20 IDS_MENU_AVTV	v / Tv	Av / Tv	El., Zeitwe Va/Vt	Av / Tv	Av / Tv	Av / Tv	Av / Tv	

String's Name

Keep first row as language's name, and keep first column as string's name

圖 3.4.1

StringTable_2013\204.xls [工作表]

A	J	K
446 STRID_WIFI	WiFi	WiFi
447 STRID_WIFI_OFF	WiFi_OFF	WiFi_OFF
448 STRID_REFRESH	Refresh	Refresh
449 STRID_WIFI_AP_MODE	AP mode	AP mode
450 STRID_WIFI_CLIENT_MODE	Client mode	Client mode
451 STRID_DUMMY	\$%&#!	\$%&#!
452		

圖 3.4.2 添加一個 Dummy 字串，可以加增特殊字元

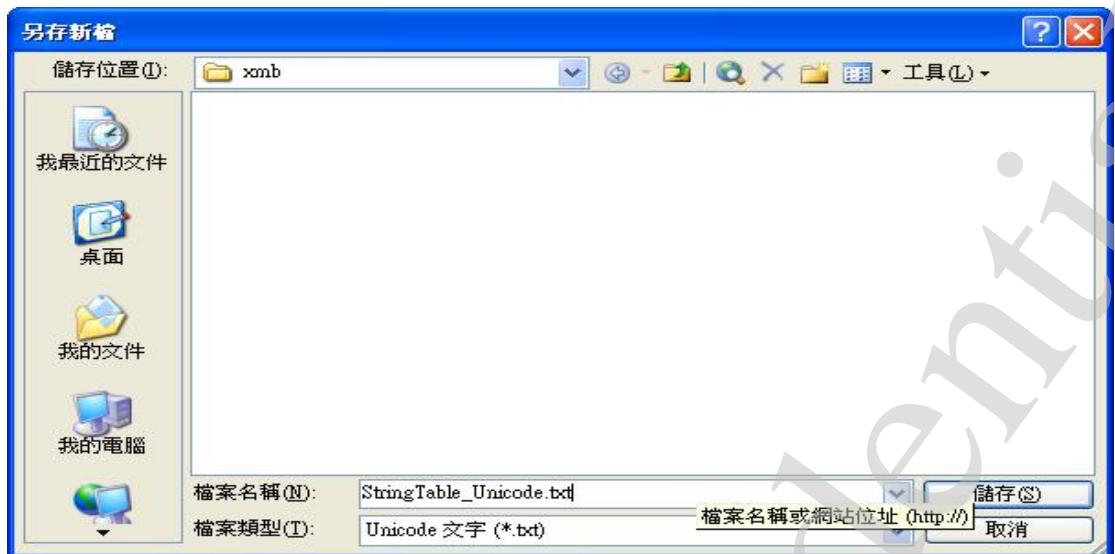


圖 3.4.3

下面開始介紹如何將 String 轉成 Code。

- Select “File\New...” (Under Resource Edit Mode)

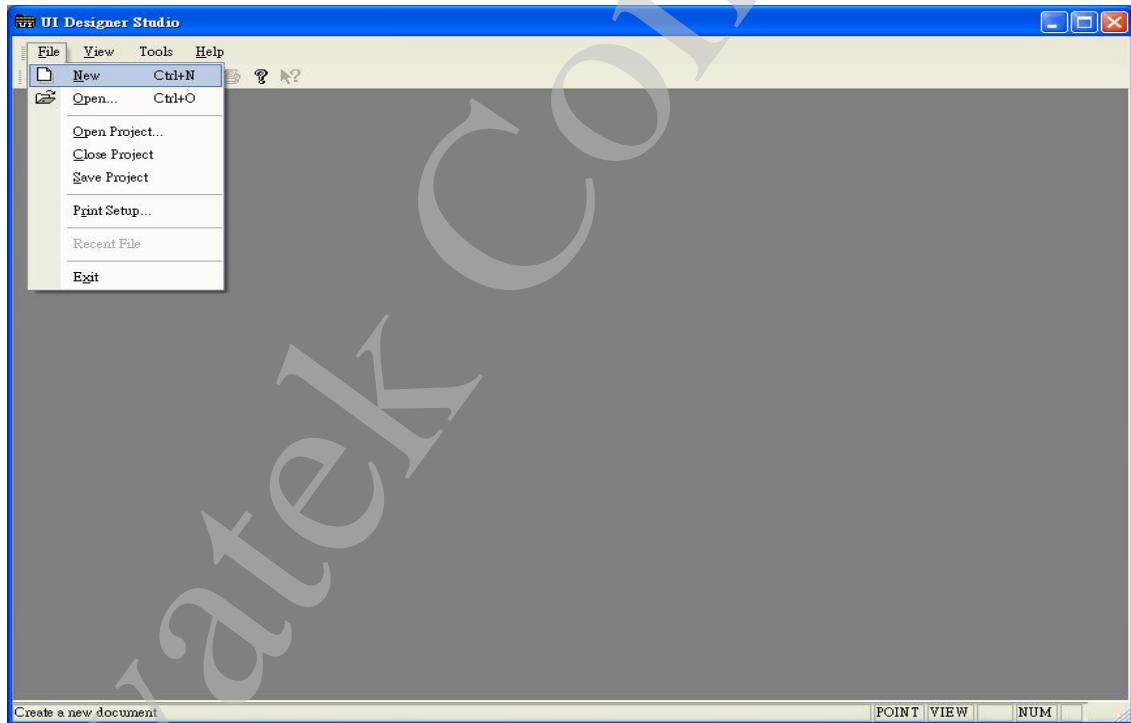


圖 3.4.4

- Choose “String Table” for a new document
- Specify filename and file location for it
- Press OK → then, new file is “DemoKit_String.xms”

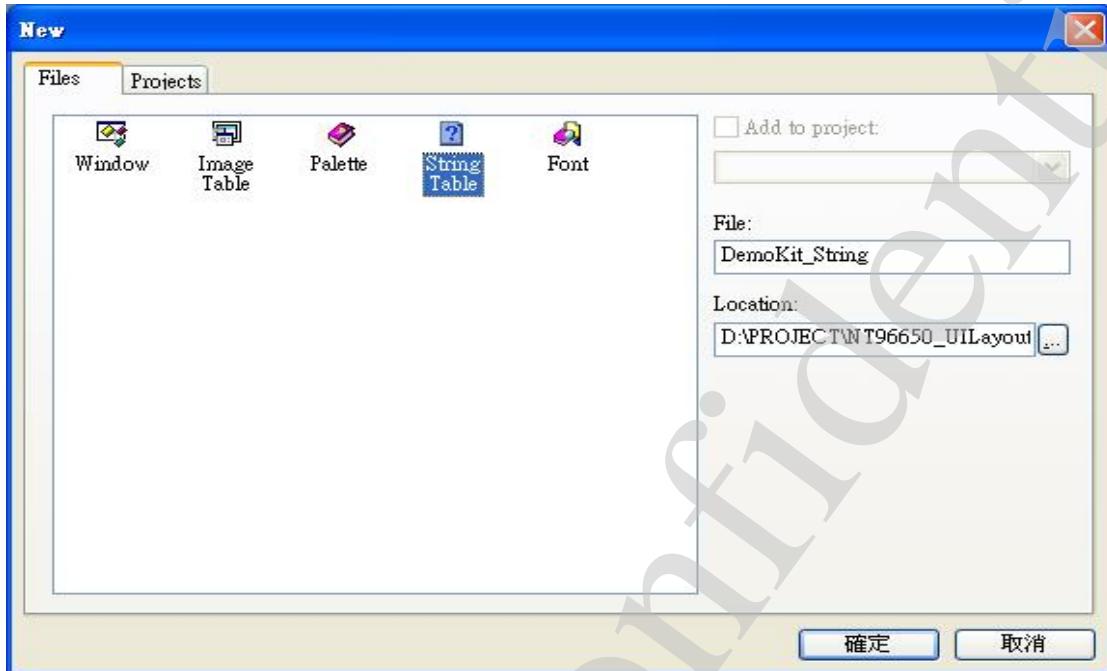


圖 3.4.5

- After create String Table
- 在建立一個 XMP project 後，就會產生 XXXX_String.xms，圖 3.4.4 和圖 3.4.5 是說明當 xms 不存在時，如何建立新的 xms

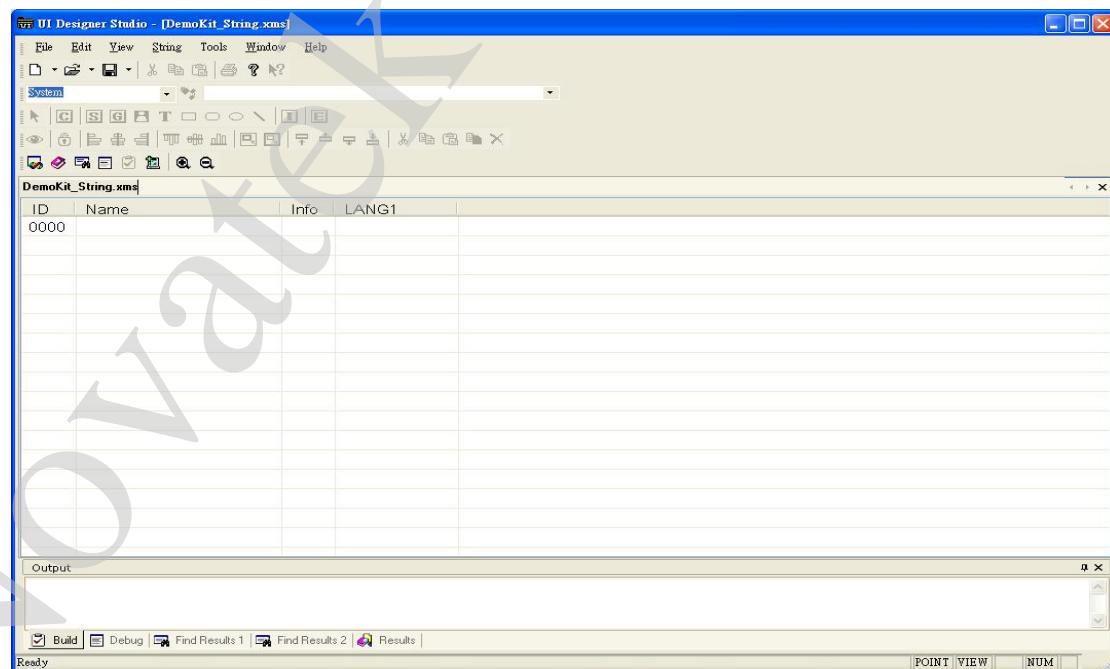


圖 3.4.6

■ Select “File\Import Name, String\from Unicode text file...”

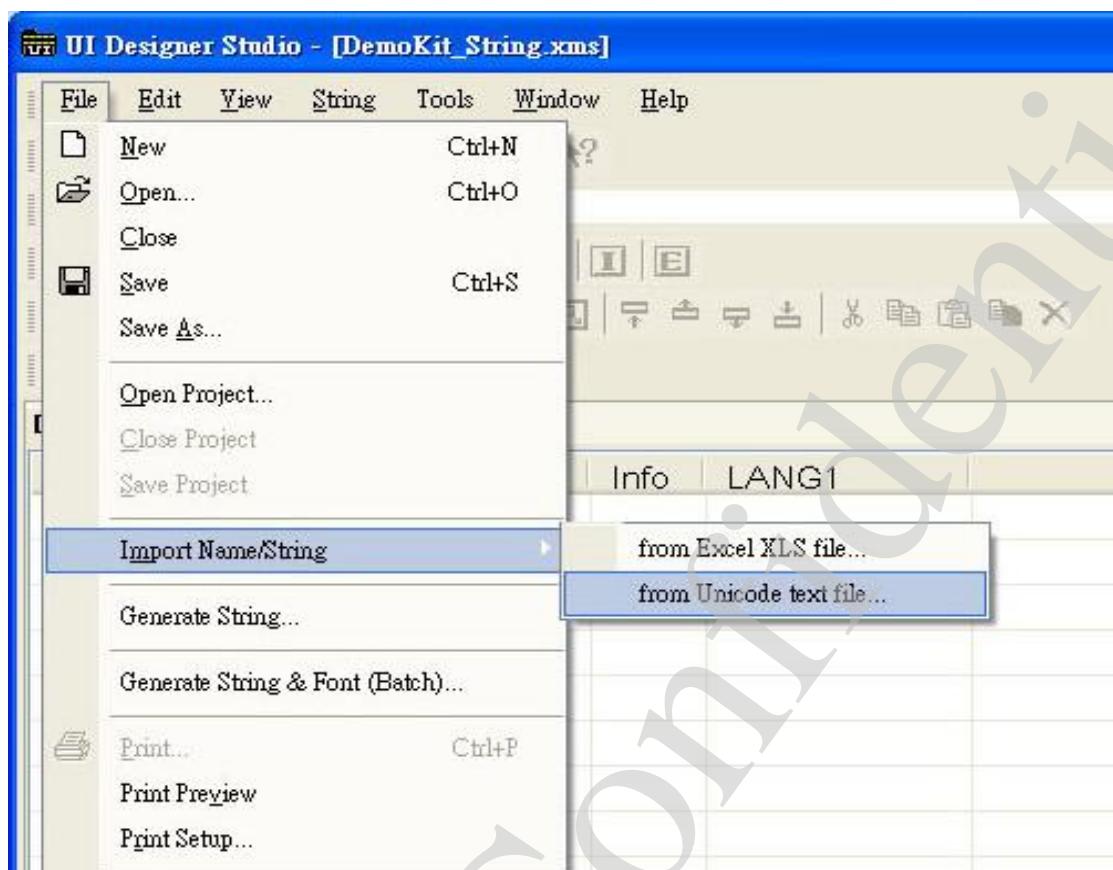


圖 3.4.7

■ Select your input string table and press OK

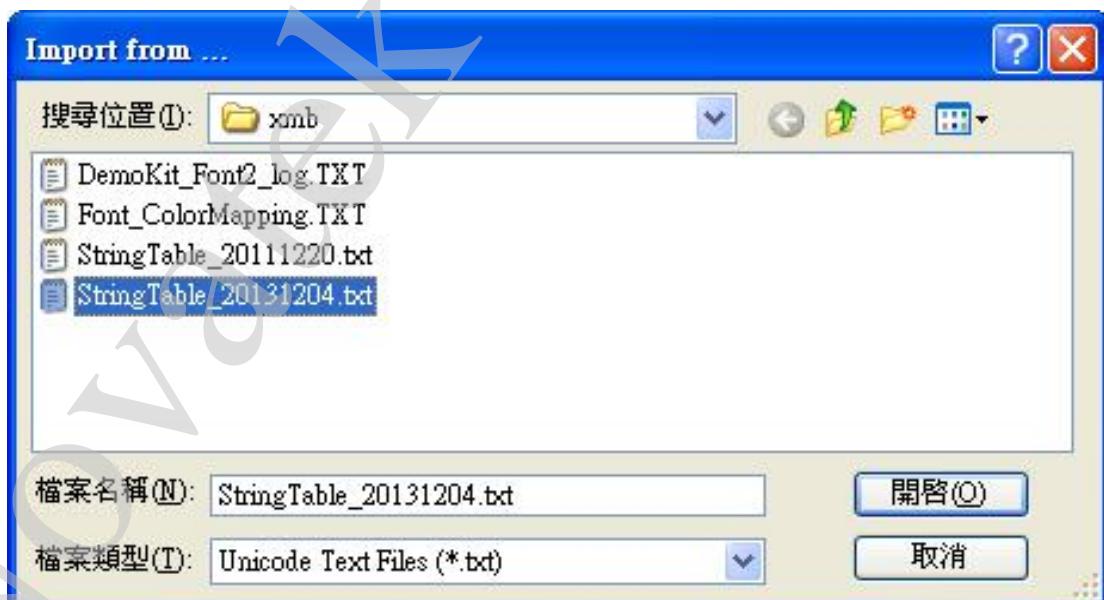


圖 3.4.8

■ 圖 3.4.9 顯示載入時的設定。“Ignore warning & error message”建議不要勾選，畢竟有這些訊息出現，表示可能有問題。

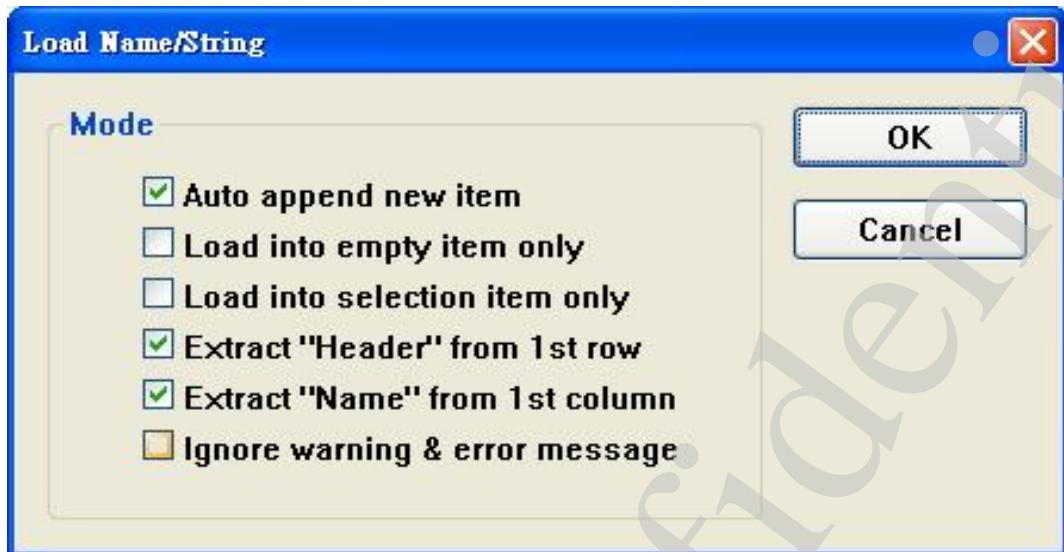


圖 3.4.9

■ After import string from Unicode text file

The screenshot shows the UI Designer Studio interface with the title bar 'UI Designer Studio - [DemoKit_String.xms]'. The menu bar includes File, Edit, View, String, Tools, Window, Help. The toolbar has various icons. The main window displays a table named 'DemoKit_String.xms'.

Language's Name

ID	Name	Info	EN	FR	DE	ES	IT	PO	SC
0000	STRID_NULL_								
0001	STRID_MODE	Mode	Mode	Modus	Modo	Modalità	Modo	拍摄模式	
0002	STRID_IMGSIZE	Image Size	Taille Im.	Auflösung	Tam. Imág.	Dim. Immag.	Image Size	分辨率	
0003	STRID_RESOLUTION	Resolution	Resolution	Auflösung	Resolución	Dimens. Imm...	Resolução	分辨率	
0004	STRID_12M	12M	12M	12M	12M	12M	12M	12M	
0005	STRID_10M	10M	10M	10M	10M	10M	10M	10M	
0006	STRID_9M	9M	9M	9M	9M	9M	9M	9M	
0007	STRID_8M	8M	8M	8M	8M	8M	8M	8M	
0008	STRID_7M	7M	7M	7M	7M	7M	7M	7M	
0009	STRID_6M	6M	6M	6M	6M	6M	6M	6M	
000A	STRID_5M	5M	5M	5M	5M	5M	5M	5M	
000B	STRID_4M	4M	4M	4M	4M	4M	4M	4M	
000C	STRID_3M	3M	3M	3M	3M	3M	3M	3M	
000D	STRID_2MHD	2MHD	2MHD	2MHD	2MHD	2MHD	2MHD	2MHD	
000E	STRID_2M	2M	2M	2M	2M	2M	2M	2M	
000F	STRID_1M	1.3M	1.3M	1.3M	1.3M	1.3M	1.3M	1.3M	
0010	STRID_VGA	VGA	VGA	VGA	VGA	VGA	VGA	VGA	
0011	STRID_QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	
0012	STRID_D1	D1	D1	D1	D1	D1	D1	D1	

ID String's Name

Output

Build Debug Find Results 1 Find Results 2 Results

圖 3.4.10

■ Left-click on “Name” column and edit its name

ID	Name	Info	EN	FR	DE	ES	IT	PO	SC
0000	STRID_NULL_		Mode	Mode	Modus	Modo	Modalità	Modo	拍摄模式
0001	STRID_MODE								
0002	STRID_IMGSIZE		Image Size	Taille Im.	Auflösung	Tam. Imág.	Dim. Immag.	Image Size	分辨率
0003	STRID_RESOLUTION		Resolution	Resolution	Auflösung	Resolución	Dimens. Imm...	Resolução	分辨率
0004	STRID_12M		12M	12M	12M	12M	12M	12M	12M
0005	STRID_10M		10M	10M	10M	10M	10M	10M	10M
0006	STRID_9M		9M	9M	9M	9M	9M	9M	9M
0007	STRID_8M		8M	8M	8M	8M	8M	8M	8M
0008	STRID_7M		7M	7M	7M	7M	7M	7M	7M
0009	STRID_6M		6M	6M	6M	6M	6M	6M	6M
000A	STRID_5M		5M	5M	5M	5M	5M	5M	5M
000B	STRID_4M		4M	4M	4M	4M	4M	4M	4M
000C	STRID_3M		3M	3M	3M	3M	3M	3M	3M
000D	STRID_2MHD		2MHD	2MHD	2MHD	2MHD	2MHD	2MHD	2MHD
000E	STRID_2M		2M	2M	2M	2M	2M	2M	2M
000F	STRID_1M		1.3M	1.3M	1.3M	1.3M	1.3M	1.3M	1.3M
0010	STRID_VGA		VGA	VGA	VGA	VGA	VGA	VGA	VGA
0011	STRID_QVGA		QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA
0012	STRID_n1		n1	n1	n1	n1	n1	n1	n1

圖 3.4.11

■ Select “File\Save” to save current String Table

Info	EN	FF
Mode	Mod	
Image Size	Ta	
Resolution	Re	
12M	12	
10M	10	
9M	9M	
8M	8M	
7M	7M	
6M	6M	
5M	5M	
4M	4M	
3M	3M	
2MHD	2M	
2M	2M	
1.3M	1.3	

圖 3.4.12

■ 圖 3.4.13 是 UI Tool 版本日期 20130322，Select “File\Generate String...”

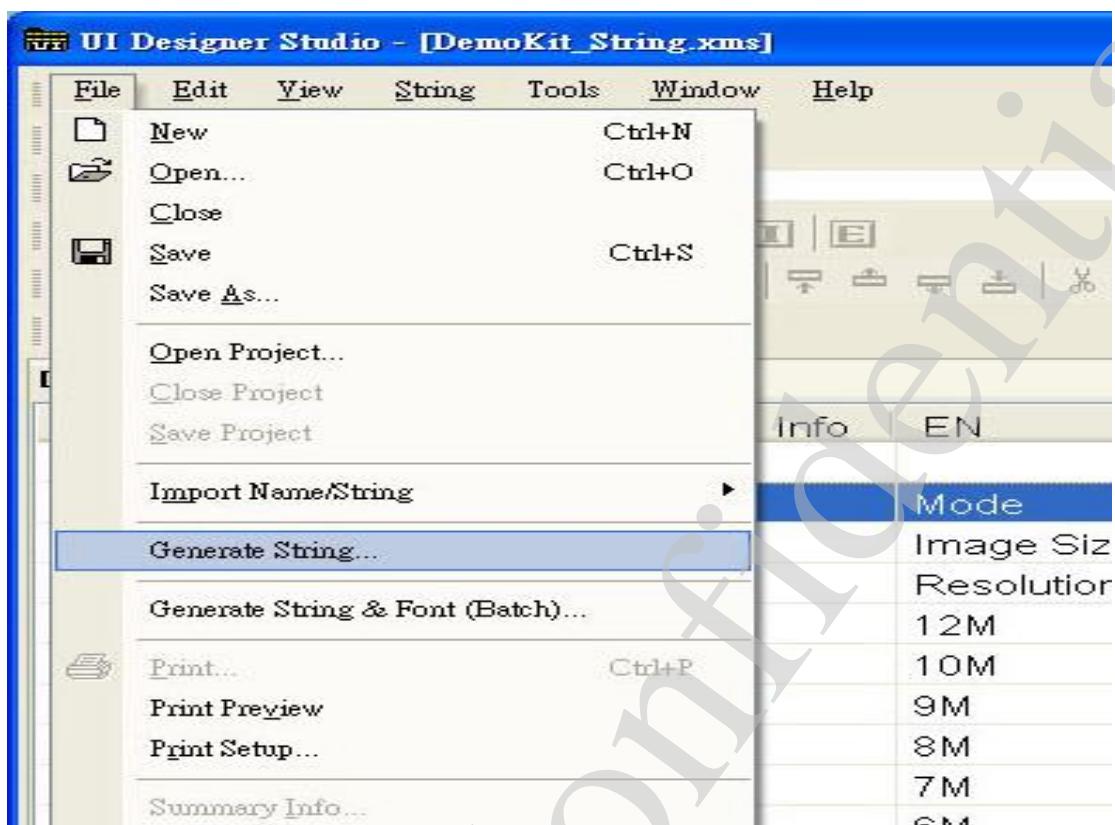


圖 3.4.13

■ 圖 3.4.14 是 UI Tool 版本日期 20100430，Select “File\Generate String Table\in C Language STRING_TABLE format...”

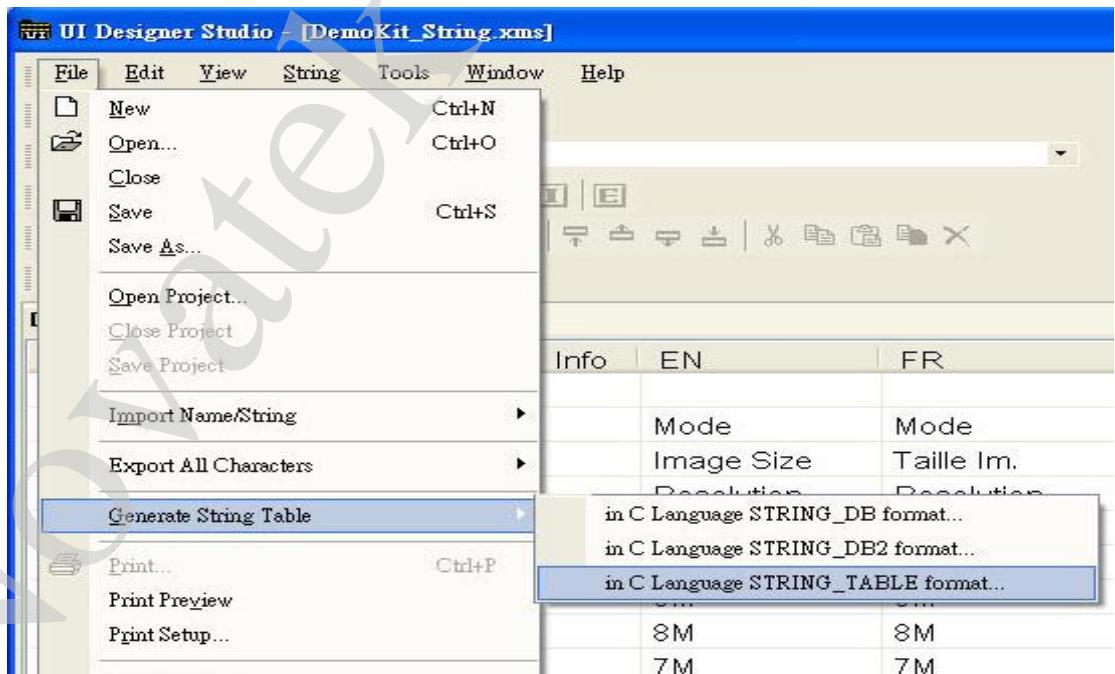


圖 3.4.14

■ Export Dialog for control string encoding , 圖 3.4.15 是 UI Tool 版本日期 20130322 , 圖 3.4.16 是 UI Tool 版本日期 20100430 。



圖 3.4.15 Export Dialog for control string encoding (UI Tool date is 20130322)

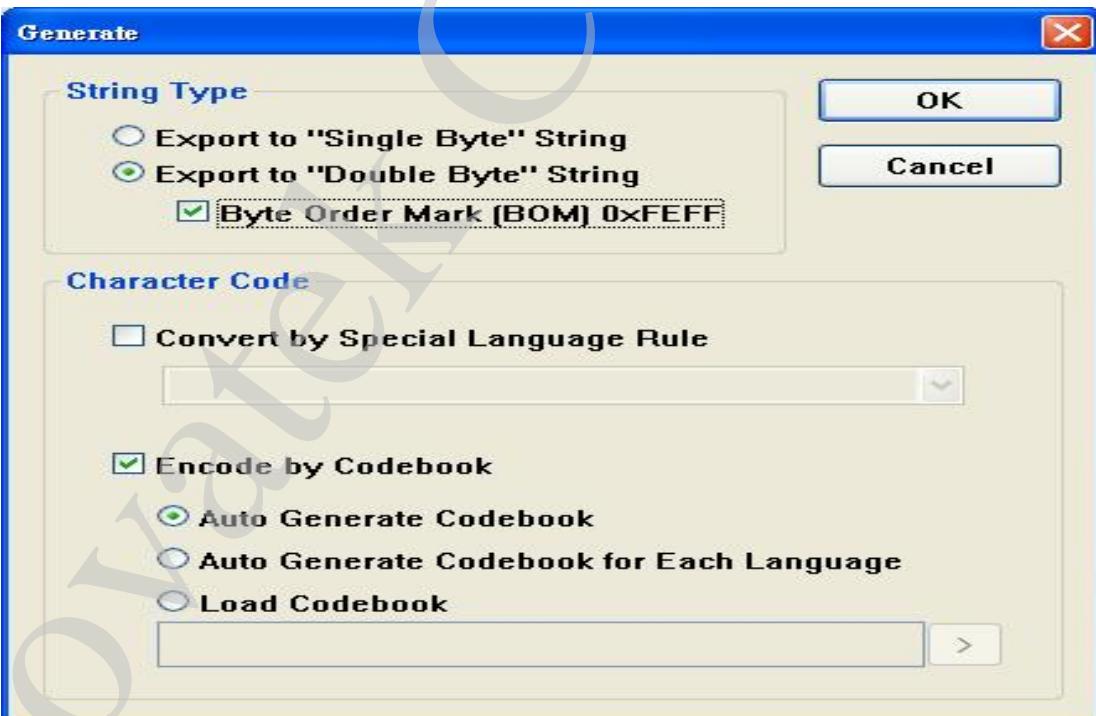


圖 3.4.16 Export Dialog for control string encoding (UI Tool date is 20100430)

- Export to *.c, *.h and *.bin files
- The bin file is a binary format STRING_TABLE. 通常不想讓 FW size 變大，就會將這個 bin file 儲存至內存 flash memory，要用時在讀取出來放至 DRAM 使用。
- 在轉換成 Code 的同時，也會產生 XXXX_String_Codebook.TXT (這邊範例是 DemoKit_String_Codebook.TXT)，這個是要給下一章節要介紹的 Font Tool 所使用的。

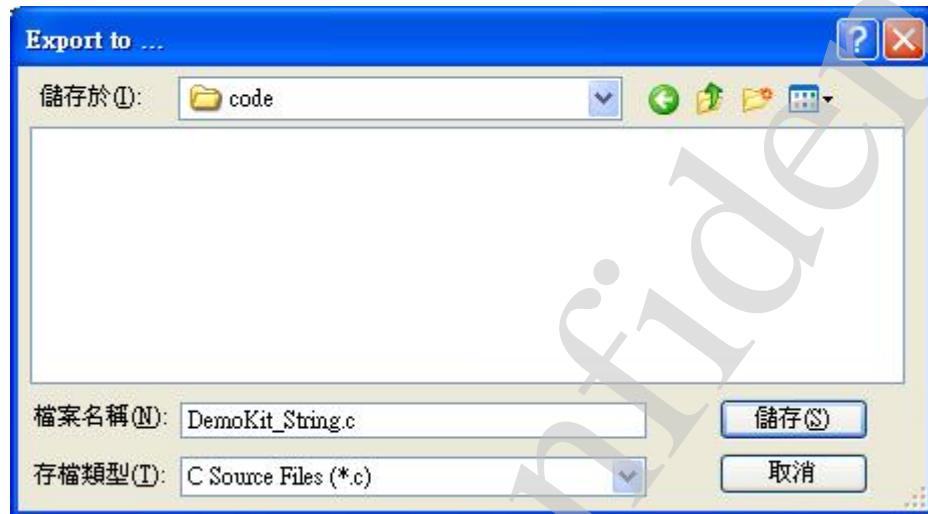


圖 3.4.17

- To use this image table in GxGfx:
 - (1) Add DemoKit_String.h and DemoKit_String.c to your project
 - (2) #include "DemoKit_String.h"
 - (3) GxGfx_SetStringTable((const STRING_TABLE*) gDemoKit_String);

3.5 Font Tool

這邊說明 Font Tools 的使用，以及如何轉成 Code。

- Select “File\New...” (Under Resource Edit Mode)

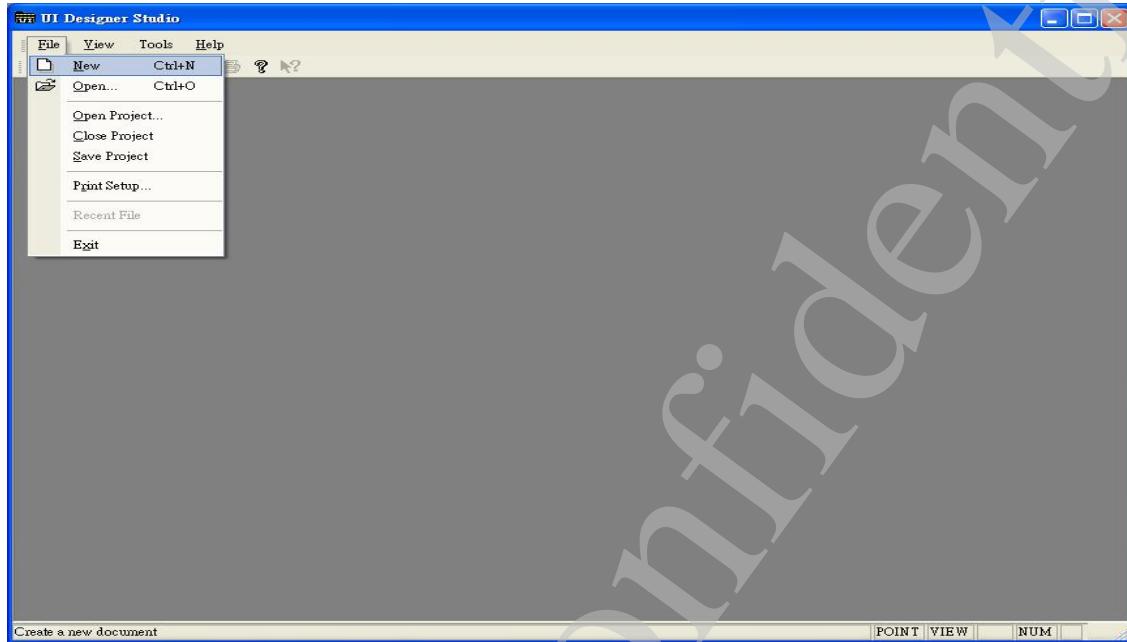


圖 3.5.1

- Choose “Font” for a new document
- Specify filename and file location for it
- Press OK → then, new file is “DemoKit_Font.xmf”



圖 3.5.2

- After create font
- 在建立一個 XMP project 後，就會產生 XXXX_Font.xmlf，圖 3.5.1 和圖 3.5.2 是說明當 xmlf 不存在時，如何建立新的 xmlf

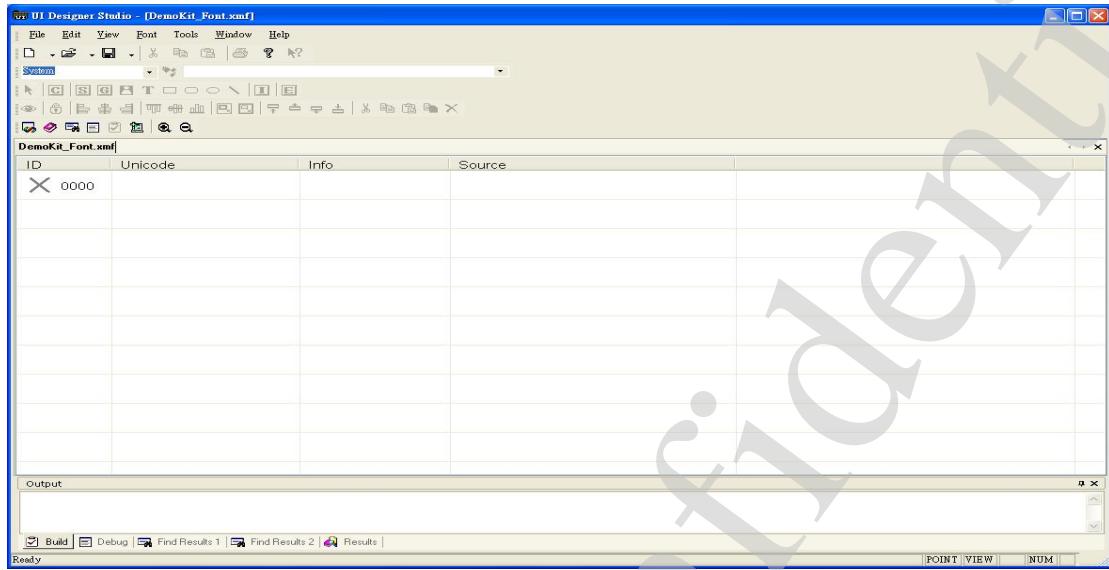


圖 3.5.3

- Select “File\Import Unicode\from Codebook”

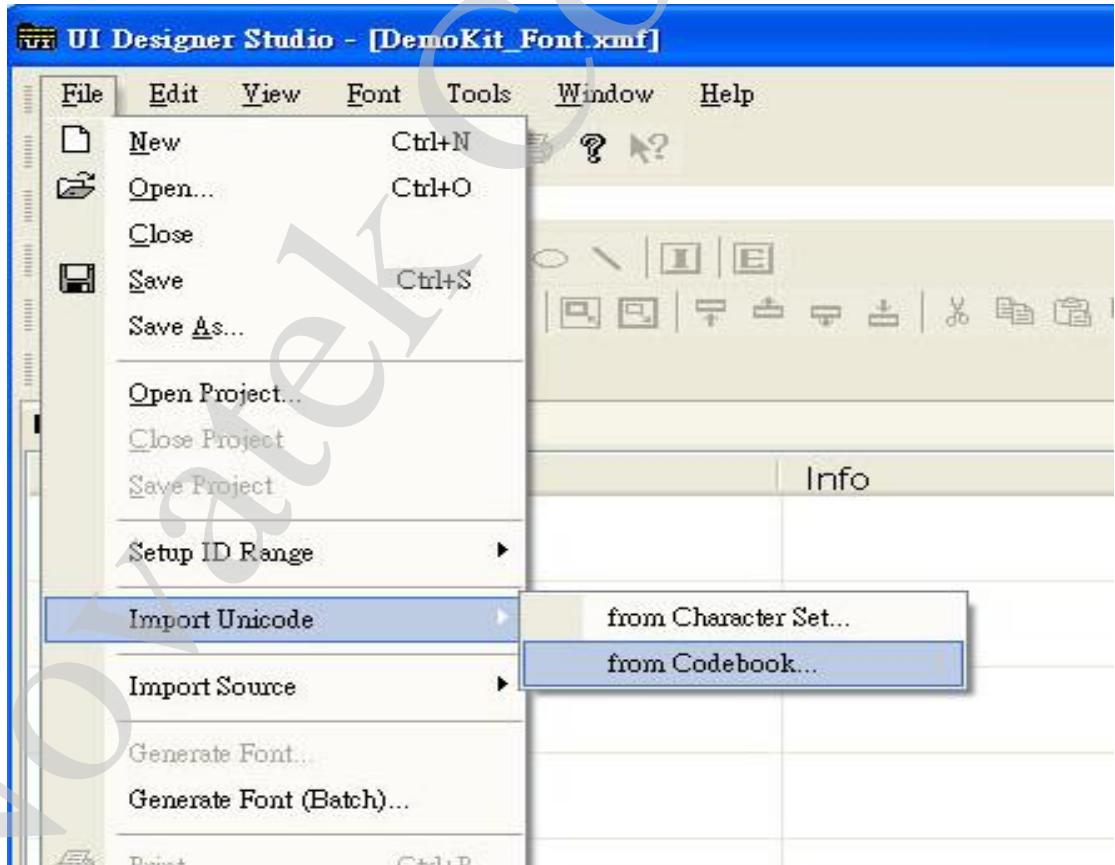


圖 3.5.4

■ Select the Codebook Generated by STRING_TABLE



圖 3.5.5

■ After loading codebook

ID	Unicode	Info	Source
X 001E			
X 001F			
X 0020	0020		
X 0021	0021		
X 0022	0022		
X 0023			
X 0024			
X 0025			
X 0026	0026		
X 0027	0027		

Encode ID Character's original UNICODE

These characters are not used in String Table

圖 3.5.6

■ Select “File\Import Source\From BMP files (Unicode only)...”

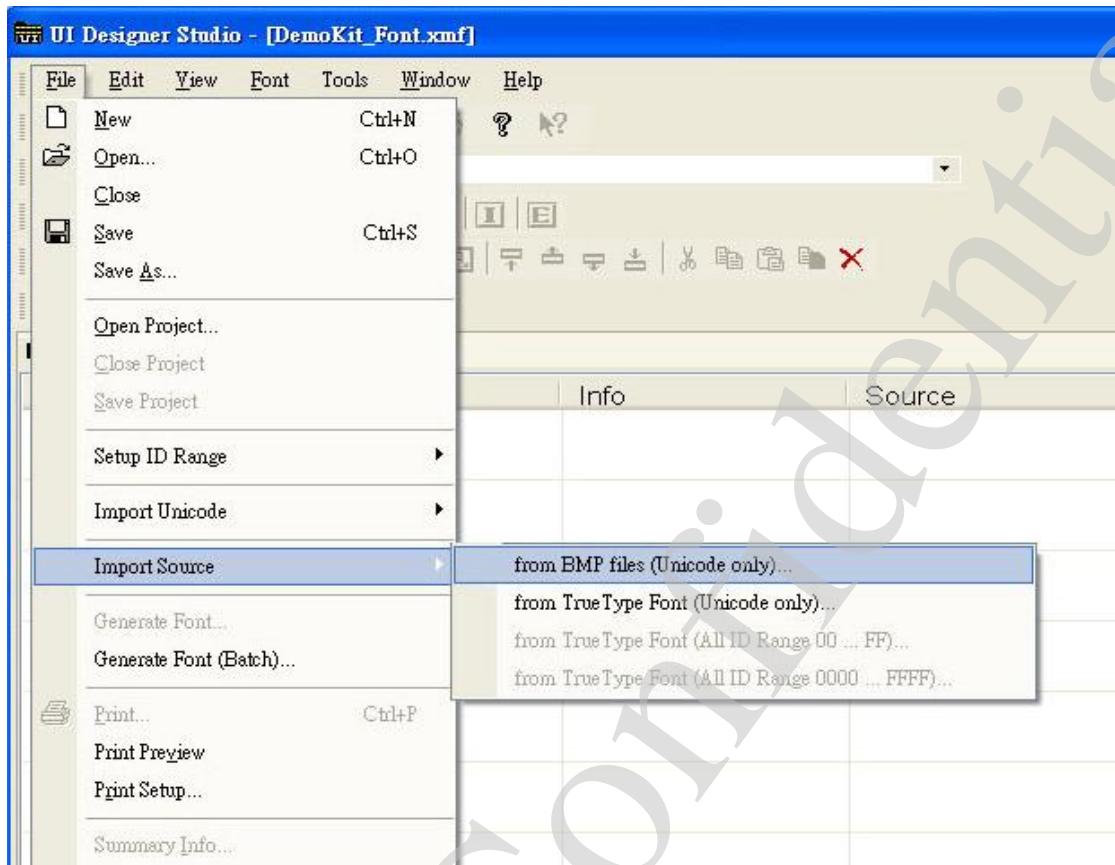
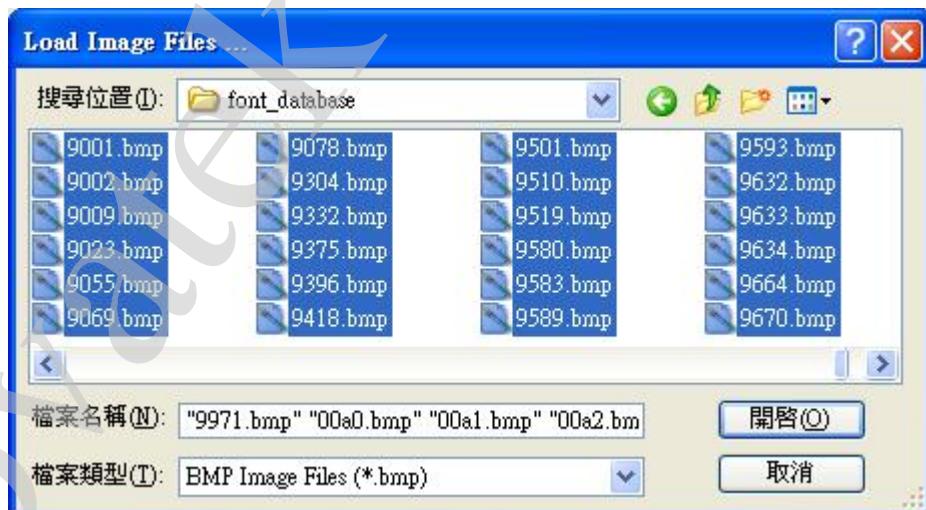


圖 3.5.7

■ Choose all source images for all characters



NOTE: Image file of each character is must be named by its UNICODE.

NOTE: Each image file must be 256 colors format and with the same palette

圖 3.5.8

■ After loading Font images

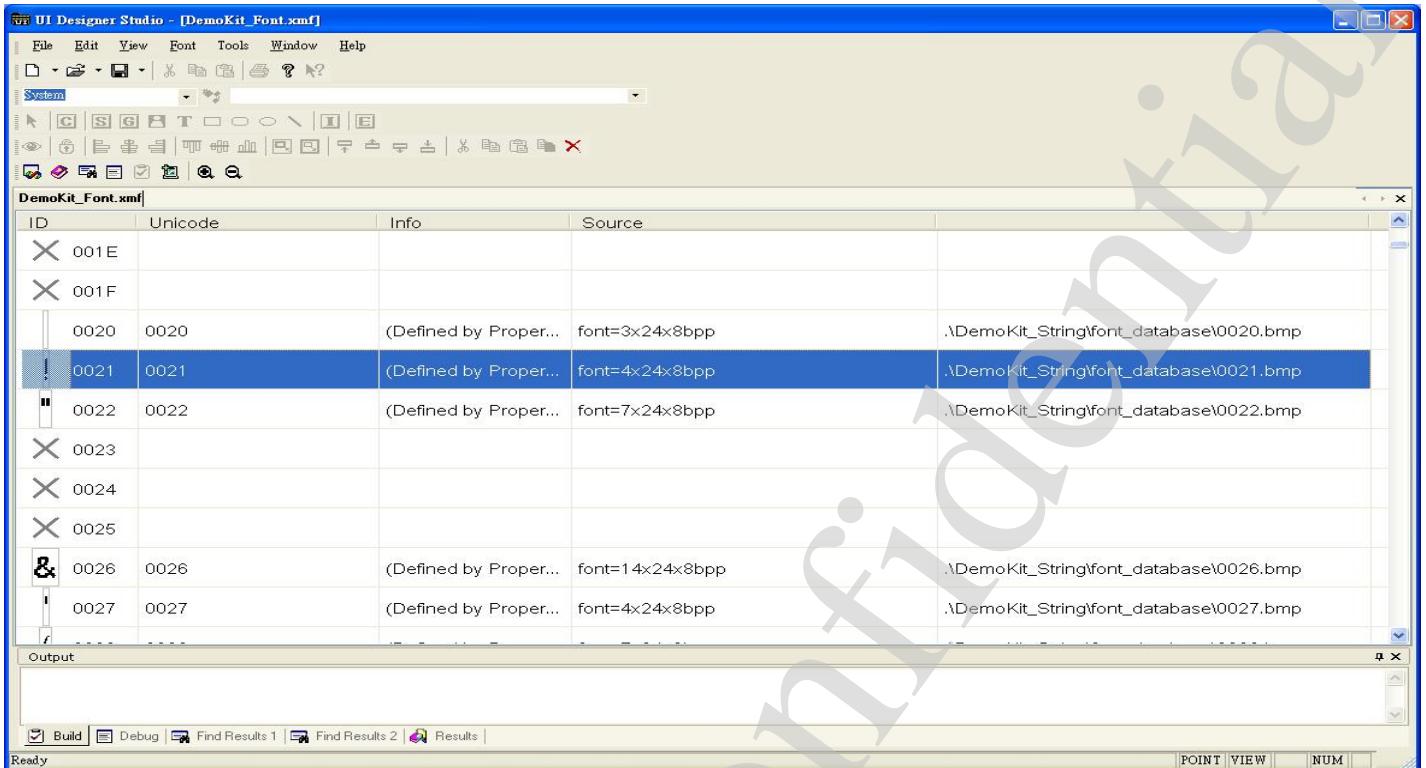


圖 3.5.9

■ Right-click and select “Properties...”

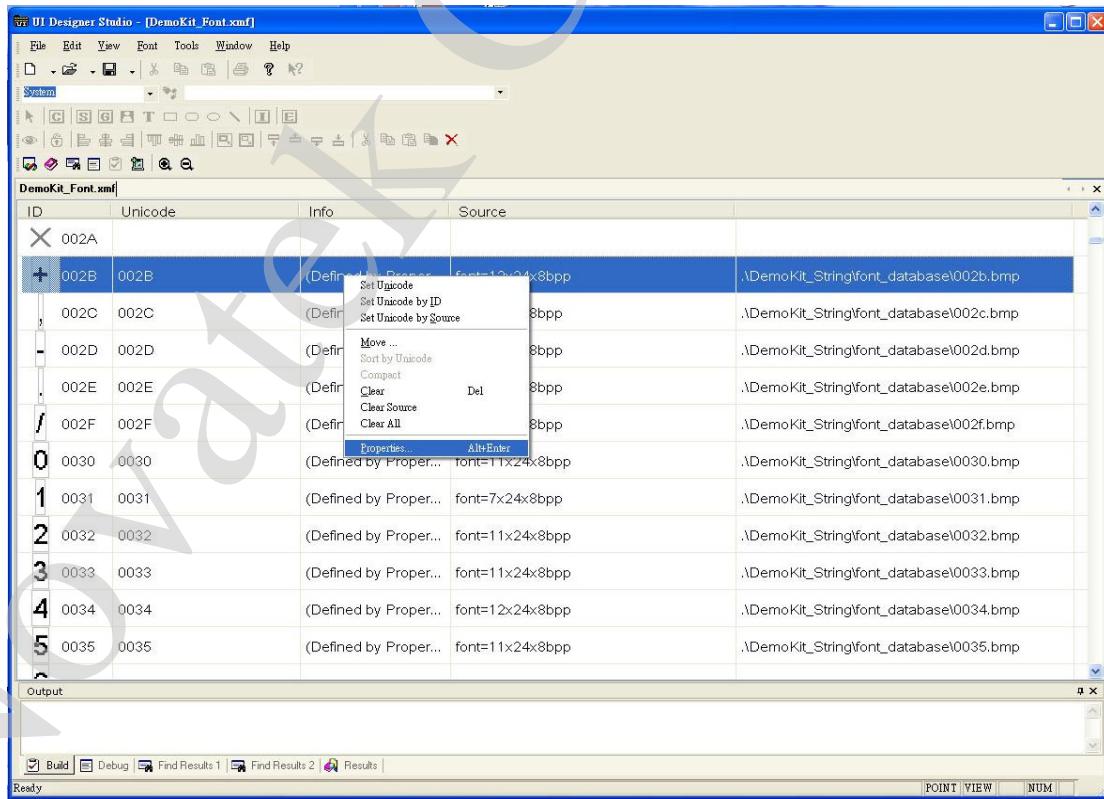


圖 3.5.10

■ Setup properties of Font

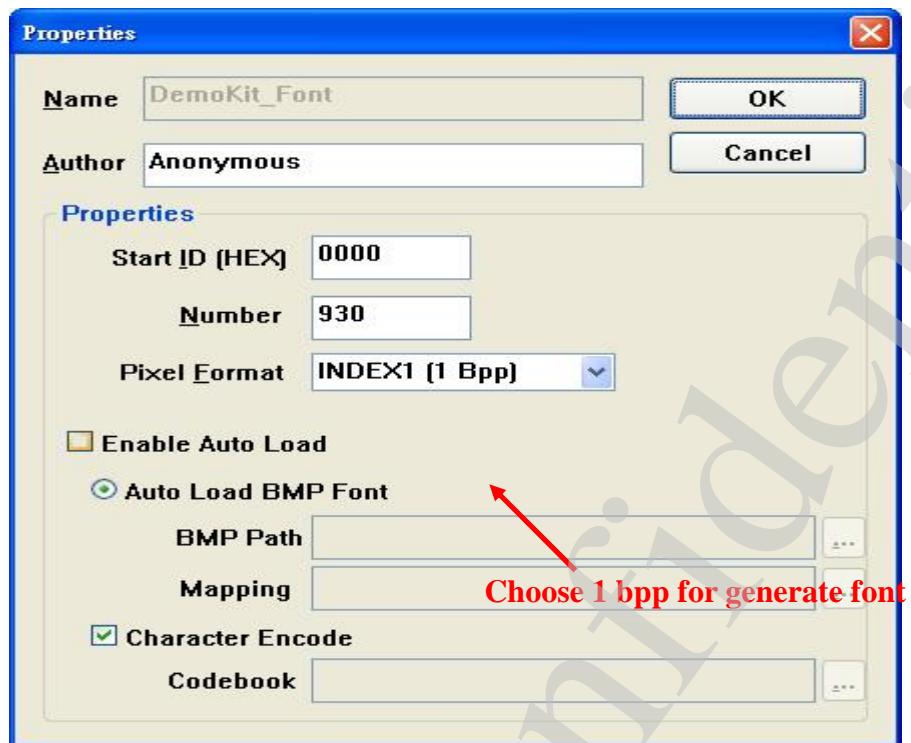


圖 3.5.11

■ 圖 3.5.12 是 UI Tool 版本日期 20130322，Select “File\Generate String...”，並且會彈出圖 2.5.13 視窗

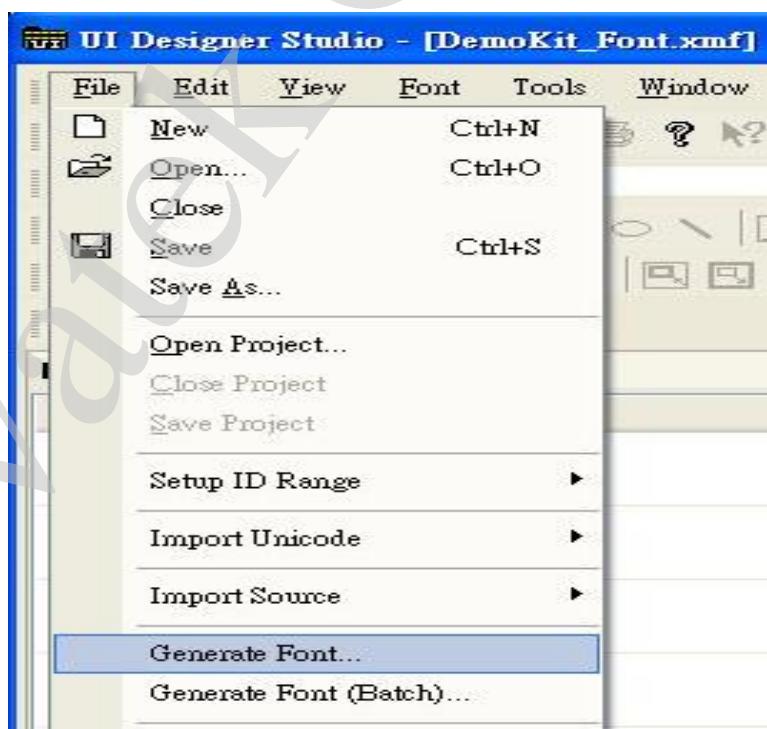


圖 3.5.12

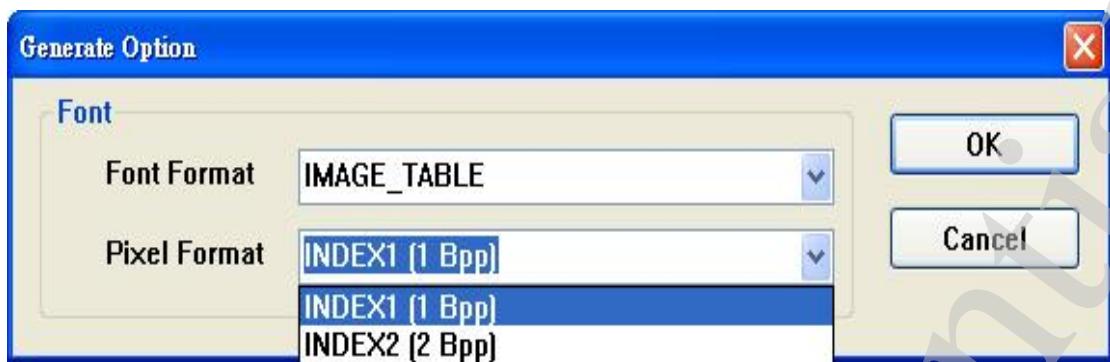


圖 3.5.13, UI Tool 版本日期 20130322 會有此視窗

- 圖 3.5.13 是 UI Tool 版本日期 20100430 , Select “File\Generate Font\in C Language IMAGE_TABLE format...”

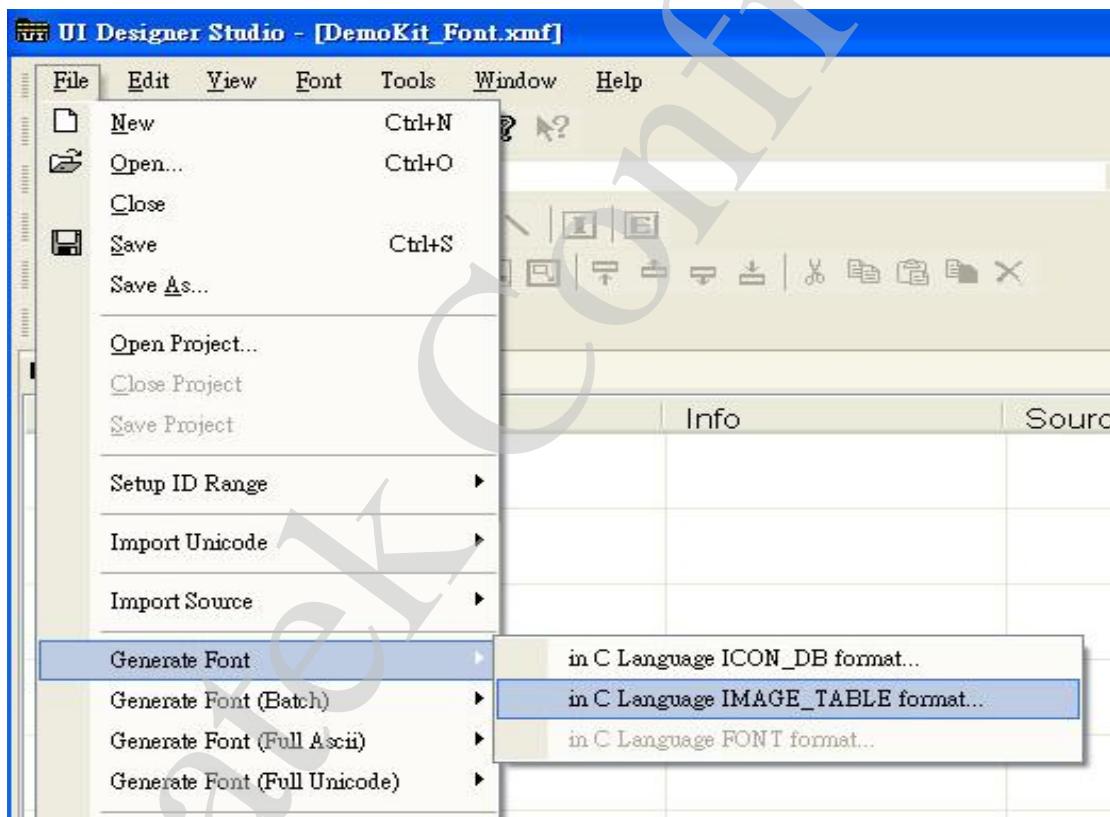


圖 3.5.14

■ 如果 Font 是由背景、主體、外框所組成，則會有圖 2.5.15 的顏色連連看視窗出現，

- 請正確指定下列這 4 種區域的顏色關係：
 - [上方] [下方]
 - 圖中的背景顏色：請對應到下方 0，(重要！一定要設定正確)
 - 圖中的外框顏色：請對應到下方 2，(如果字型圖檔沒有外框，可以不用指定)
 - 圖中的字型顏色：請對應到下方 1，(重要！一定要設定正確)
 - 圖中的底線顏色：請對應到下方 3，(目前無作用，可以不用指定)

■ 在 generate Font 的時候要記得存下上面 font color mapping 的 txt 記錄檔。

- 在顏色連連看的對話框選”Save”將此 mapping 關係存起來，取名例如 Font_ColorMapping.txt (方便下次可以直接 Load 此關係進來)

■ 選擇 OK 執行轉檔

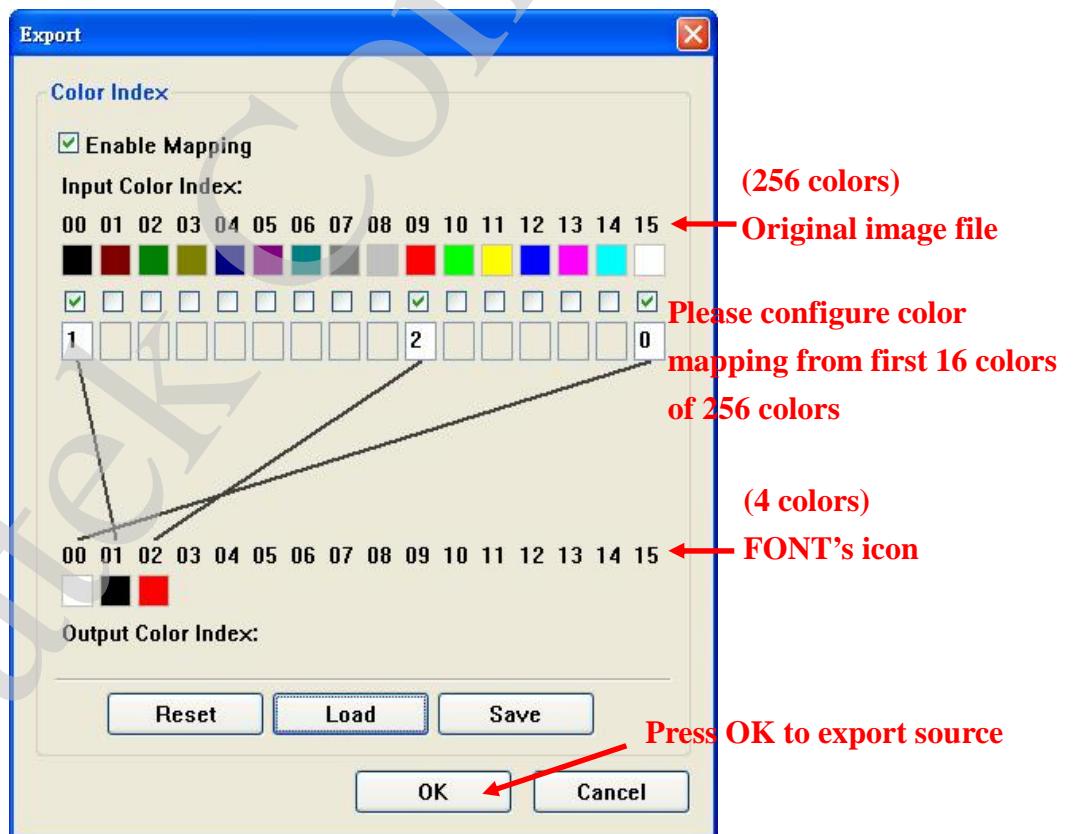


圖 3.5.15

NOTE:

User must follow this rule to mapping original input font color to match output “GxGfx” font color

“Gxgfy Font Color Define ”

00 = Background Color

01 = Font Color

02 = Shadow Color (optional)

03 = Line Color (optional)

- Export to *.c, *.h and *.bin files
- The bin file is a binary format FONT_TABLE. 通常不想讓 FW size 變大，就會將這個 bin file 儲存至內存 flash memory，要用時在讀取出來放至 DRAM 使用。

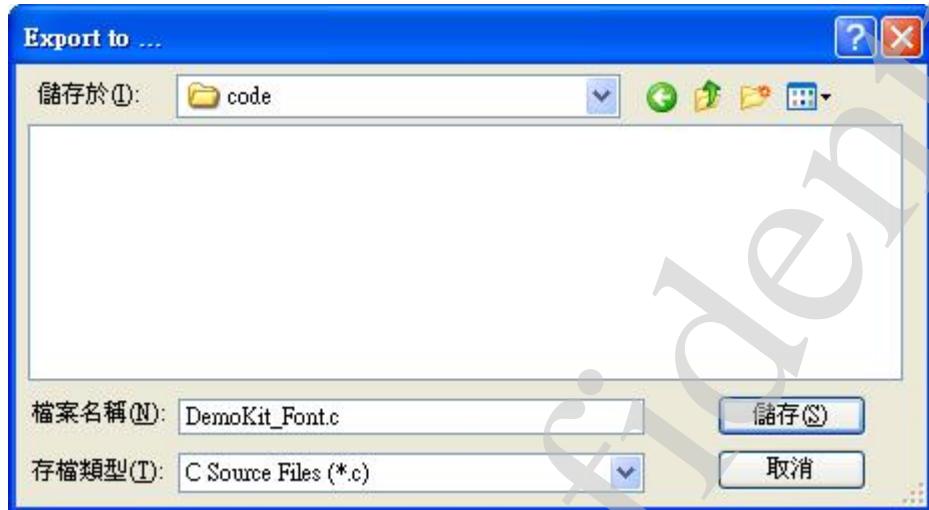


圖 3.4.17

- To use this image table in GxGfx:
 - (1) Add DemoKit_Font.h and DemoKit_Font.c to your project
 - (2) #include “DemoKit_Font.h”
 - (3) GxGfx_SetTextStroke((const FONT*) gDemoKit_Font)

3.6 Window Tool

3.6.1 UI Window 簡介與命名規則

UI Window 通常分為 UIFlow 與 UIMenu, UIFlow 就是在 Movie、Photo、Playback mode 時所看到的第一層 Window 即為 UIFlow，之後開啟菜單即為 UIMenu。通常會如圖 3.6.1 所示，會在 UI Layout 分 UIFlow 與 UIMenu 兩個主要文件夾，之後的次文件夾都放到這兩個主文件夾裡面。UIFlow 跟 UIMenu 命名的方式就如圖 3.6.1 所示，如 Movie Mode，就取名 UIFlowPhoto 和 UIMenuPhoto; 共用的部份就取名 UIFlowCommon 和 UIMenuCommon。而 Window 產生的副檔名為 xmd，就放在這些次文件夾裡面。

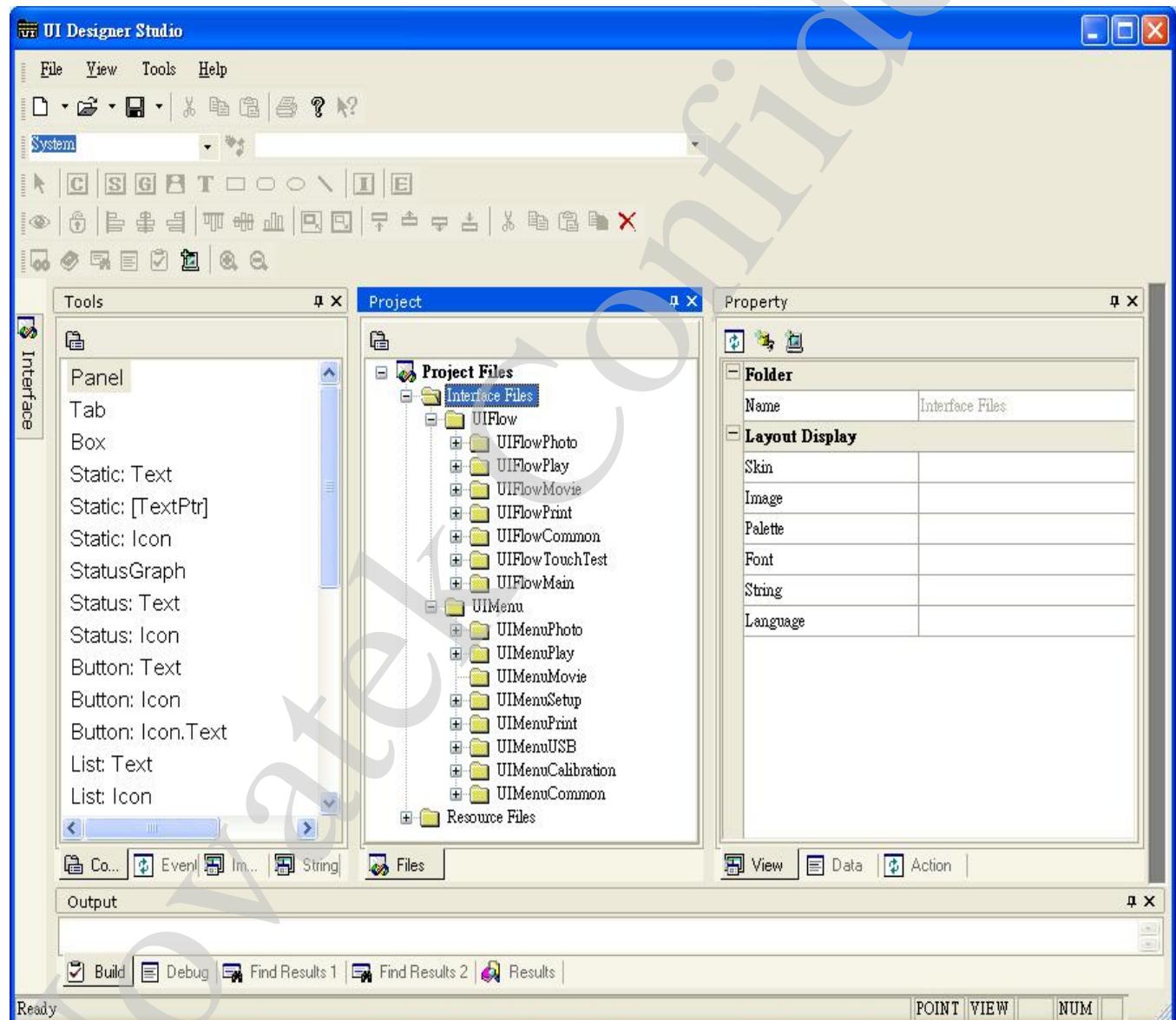


圖 3.6.1

如何在 UI Layout 新增一個 Folder?

■ 在”Interface File”上按右鍵，選擇 New Folder

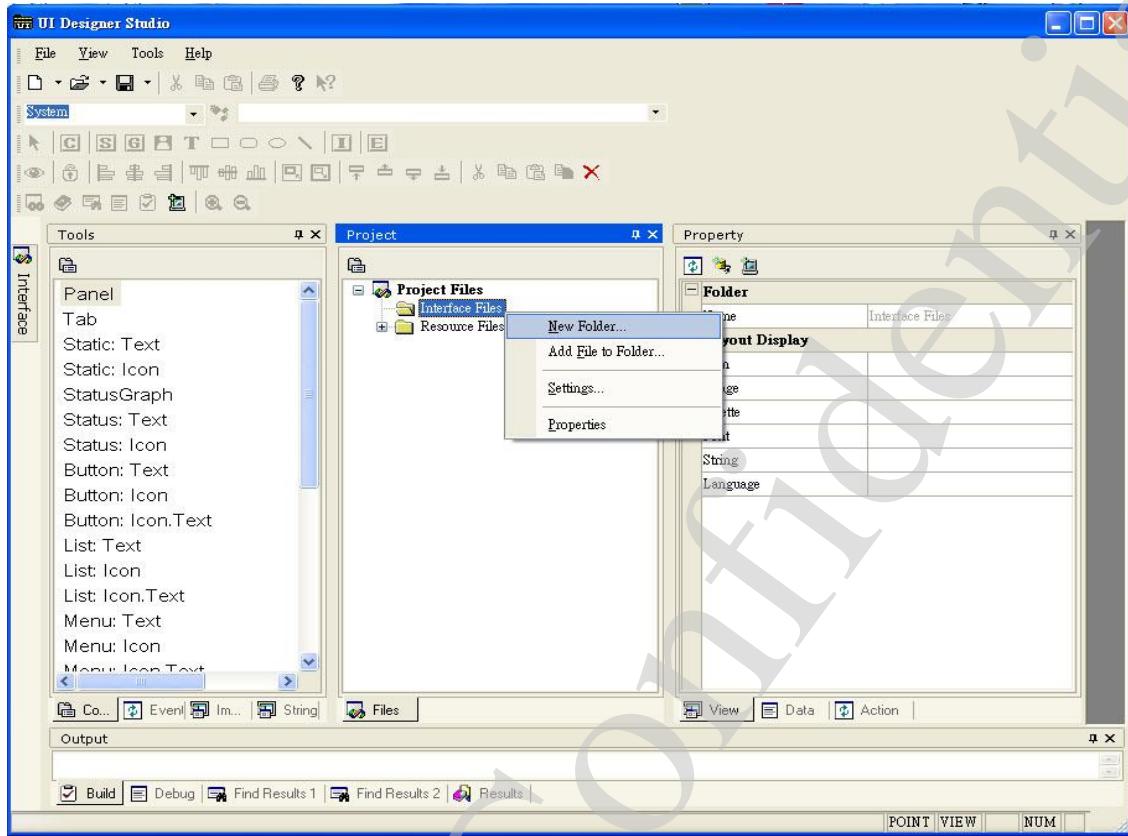


圖 3.6.2

■ 在”Name”欄位裡面修改文件名，舉例如 UIFlow

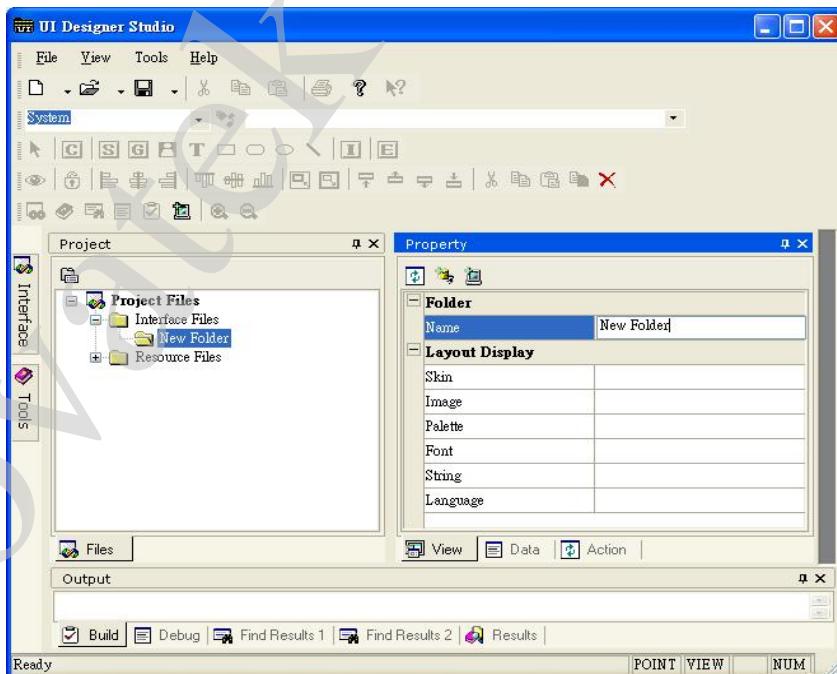


圖 3.6.3

3.6.2 Create a new window

- Select “File\New...” (Under Resource Edit Mode)

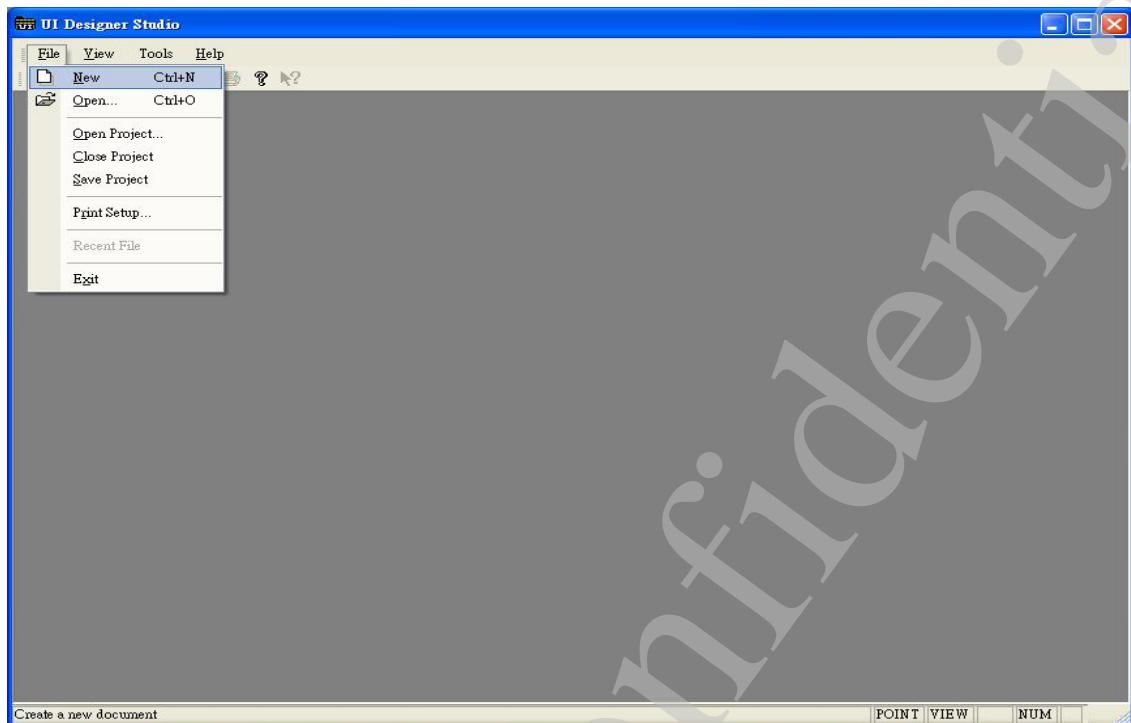


圖 3.6.1

- Choose “Window” for a new document
- Specify filename and file location for it
- Press OK → then, new file is “UIFlowWndMovie.xmd”

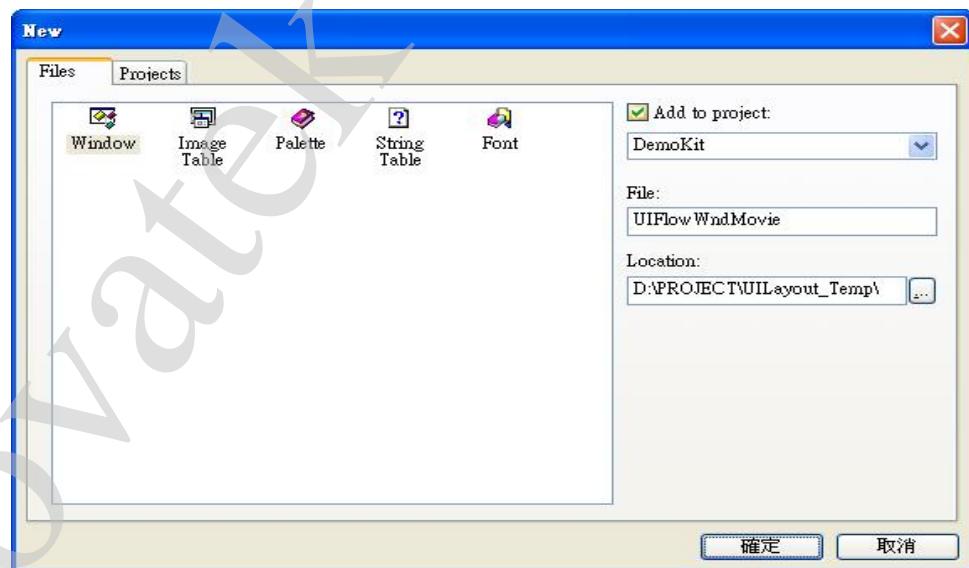


圖 3.6.2

■ Select “256 Colors”

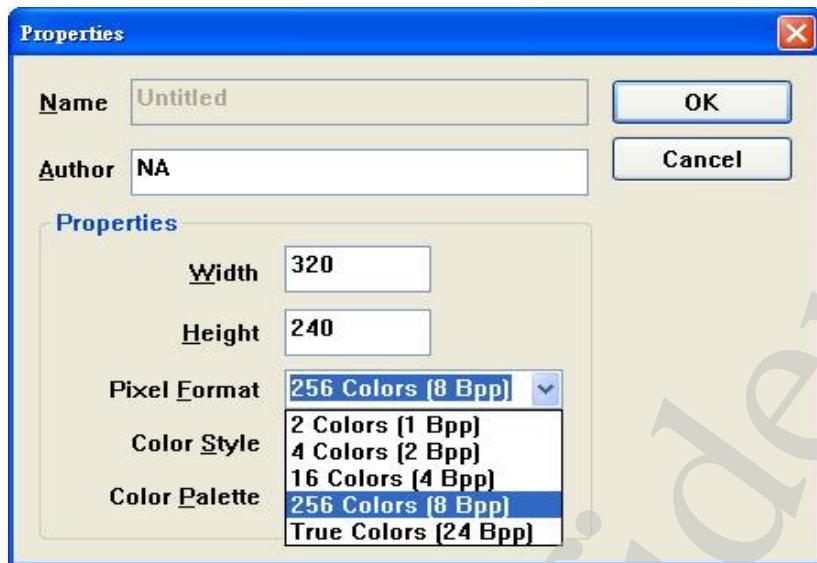


圖 3.6.3

■ 產生的 UIFlowWndMovie.xmd 檔案會在”Interface Files”路徑下，請移動到 UIFlowMovie 文件夾下。

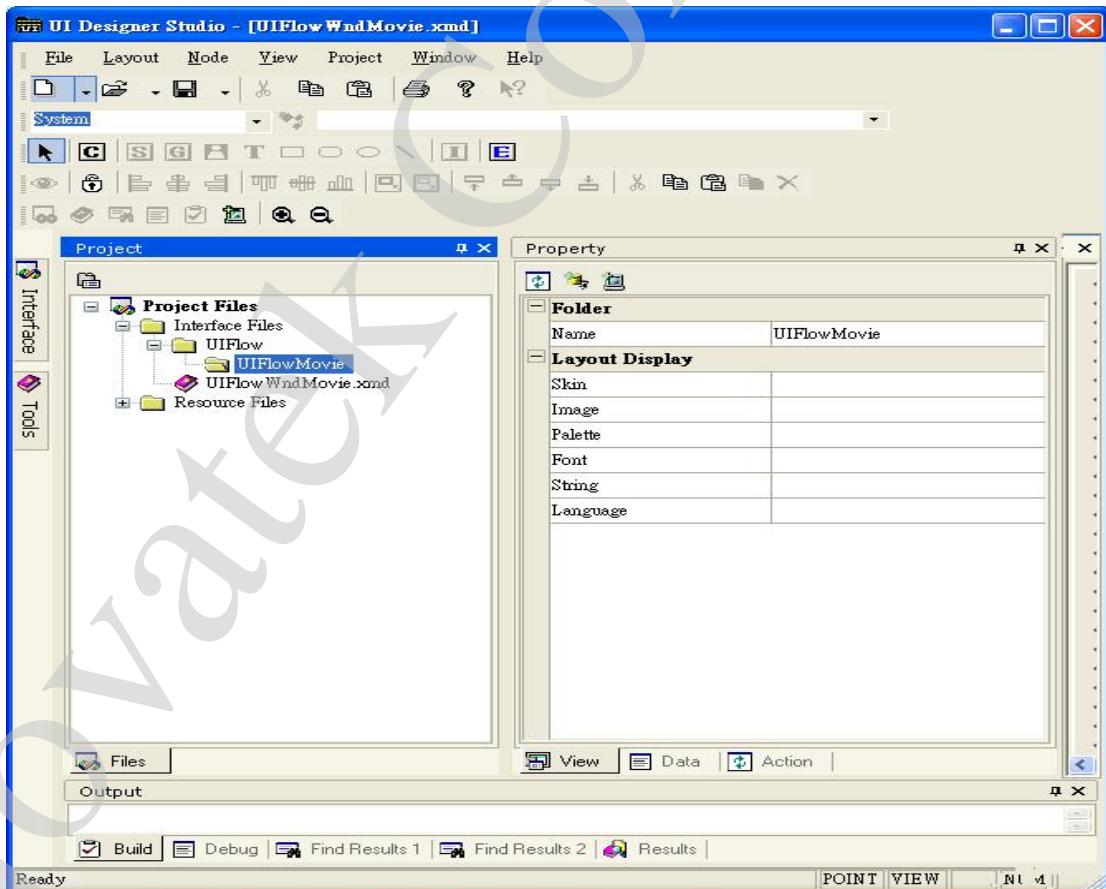


圖 3.6.4

3.7 UI Tool 及元件簡介

圖 3.7.1 就是畫 UI Layout 時會用到的所有 Tools，圖 3.7.2 即 UI Layout 畫好之後呈現在 LCD 上的 OSD layer。

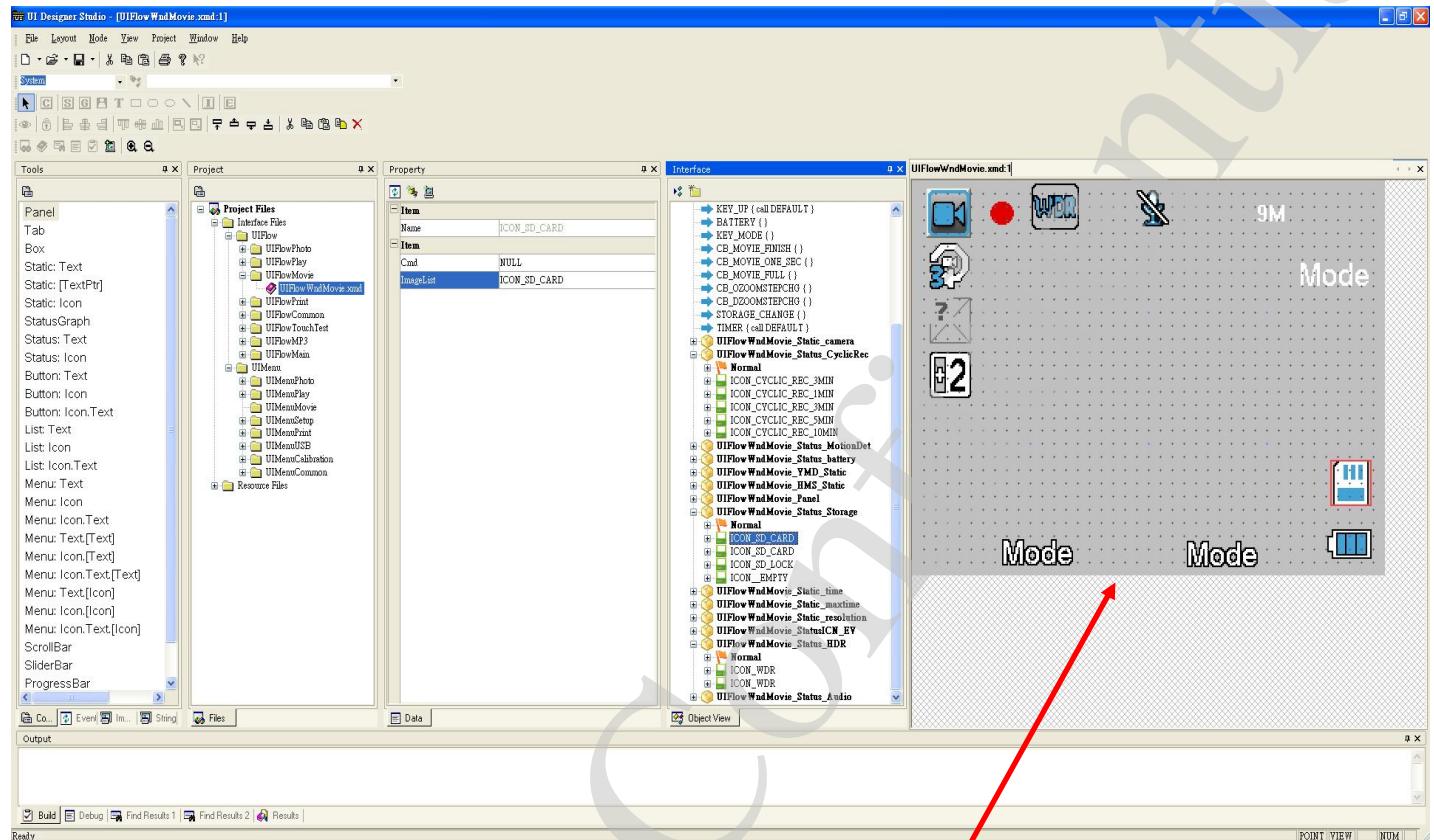


圖 3.7.1

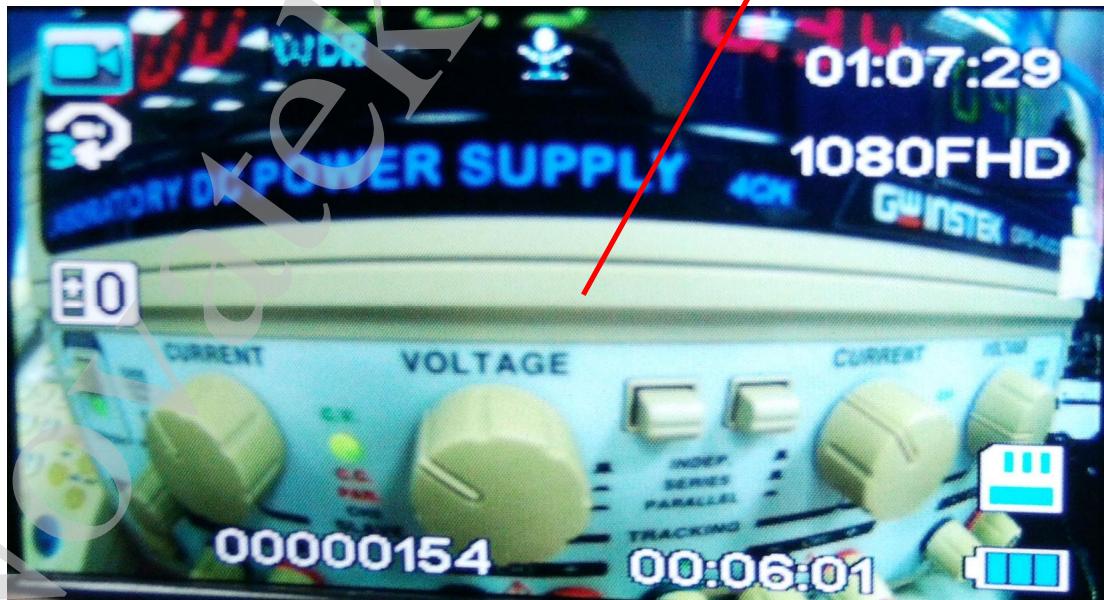


圖 3.7.2

圖 3.7.3 是 UI Layout 的元件，這邊將簡介元件的使用，不過由於 UI 是屬於多樣化，因此如何搭配這些元件的使用，還是需要視當時開發上的需求。另外我們著重在 Menu 元件的說明，因為 Menu 元件其實就包含 Skin、Icon、Text，這在 Static、Status、Button、List 等元件也都是通用的。

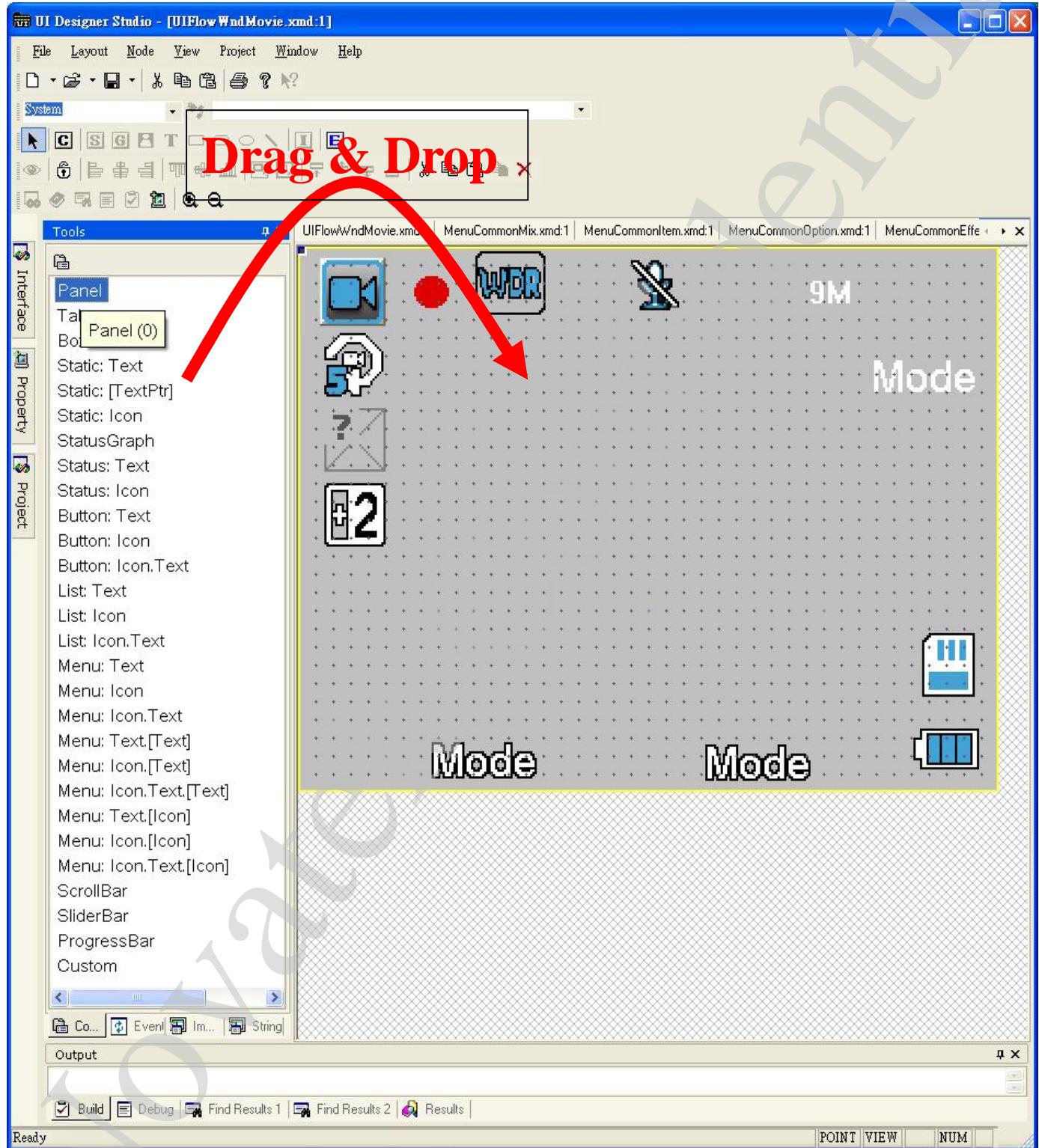


圖 3.7.3

3.7.1 Panel 元件

Panel 最典型的應用就是要控制一群元件集合的顯示或隱藏，這時候將這群元件放在 Panel 即可。命名方式建議取有意義的方式，例如假設是在 UIFlowWndMovie.xmd 下面，建議取名 UIFlowWndMovie_Panel_xxx。

3.7.2 Tab & Button 元件

Tab 通常會搭配 Button 來使用，最典型的應用就是日期跟時間的設定，如圖 3.6.2 所示，藍色框是一個 Tab 元件，裡面的年月日跟時分秒都是各自一個 Button，但上下箭頭、冒號、斜線是另外增加的，不屬於 Button。

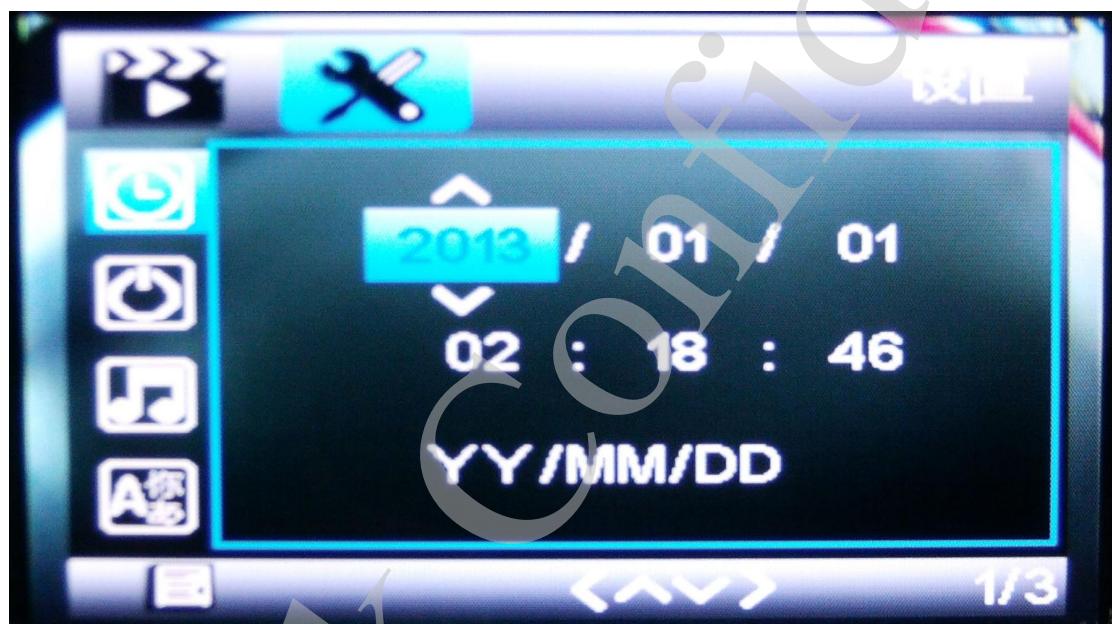


圖 3.7.5

3.7.3 Static 元件

■ Static : Text 元件

這種元件通常使用在固定顯示一個字串上，來源是 String table，如果想要更換 String Table 中的字串，只能從程式中去修改，會比較麻煩，因此如果要多種字串替換方式，則使用 status。命名方式建議取有意義的方式，例如假設是在 UIFlowWndMovie.xmd 下面，建議取名 UIFlowWndMovie_StaticTXT_xxx。

■ Static : [TextPtr] 元件

這種元件通常使用在固定一個字串，來源不一定是 String table，可以自行 assign 字串內容。命名方式建議取有意義的方式，例如假設是在 UIFlowWndMovie.xmd 下面，建

建議取名 `UIFlowWndMovie_StaticTxtPtr_xxx`。

■ Static : Icon 元件

這種元件通常使用在顯示固定一個 Icon，，如果想要更換 Icon 的話，只能從程式中去修改，會比較麻煩，因此如果要多種 Icon 替換方式，則使用 status。命名方式建議取有意義的方式，例如假設是在 `UIFlowWndMovie.xmd` 下面，建議取名 `UIFlowWndMovie_StaticICN_xxx`。

3.7.4 Status 元件

■ Status : Text 元件

這種通常使用在不同時間需顯示不同的字串訊息，如圖 3.7.6 Warning message，可以將會用的字串集合在一起，只要改變 String ID 即可改變顯示訊息。

如果要新增字串，可以點在任一字串上，按右鍵選擇”Clone”即可複製，或是按 Delete 鍵刪除字串，不過第一個字串不能被刪除，會出現圖 3.7.7 的錯誤訊息。

命名方式建議取有意義的方式，例如假設是在 `UIFlowWndWrnMsg.xmd` 下面，建議取名 `UIFlowWndWrnMsg_StatusTXT_xxx`。

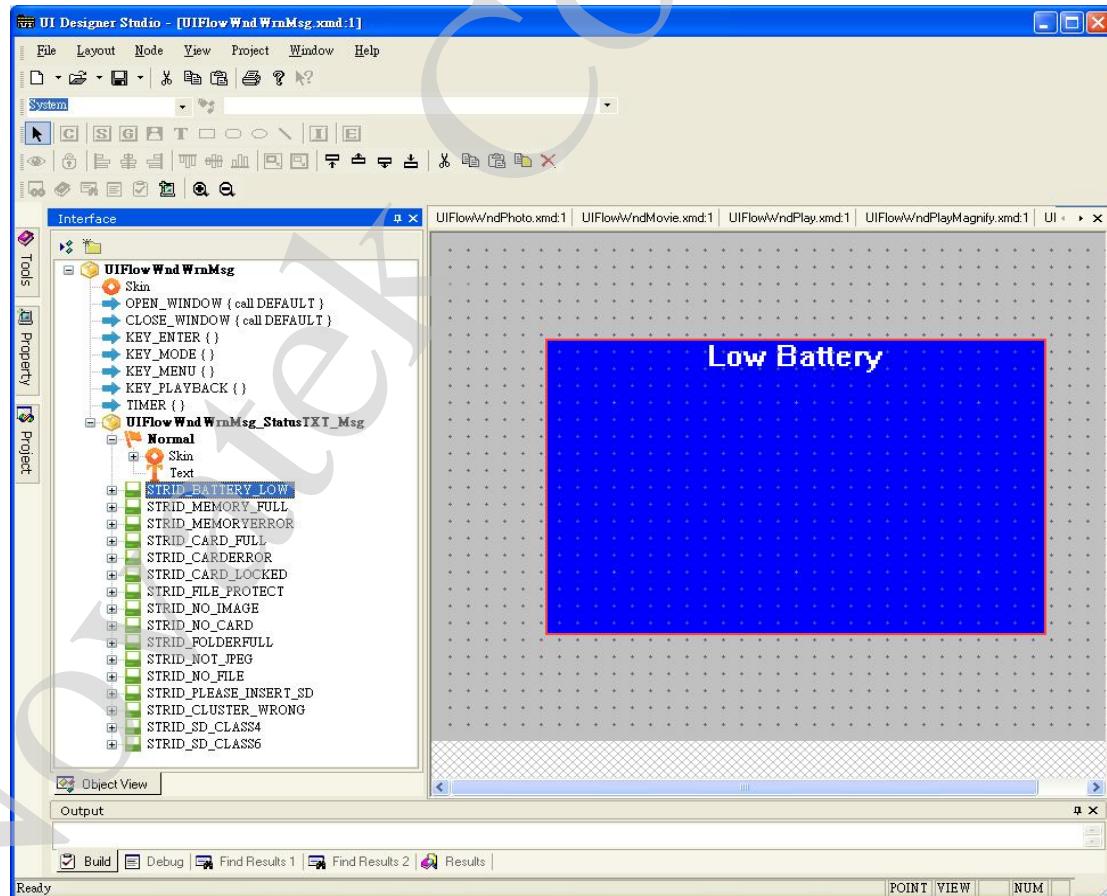


圖 3.7.6



圖 3.7.7

■ Status : Icon 元件

這種通常使用在不同時間需顯示不同的 Icon，一個典型的應用是電池電量顯示，如圖 3.7.8 所示，可以將會用的 Icon 集合在一起，只要改變 Icon ID 即可改變顯示 Icon。

如果要新增 Icon，可以點在任一 Icon 上，按右鍵選擇 Clone 即可複製，或是按 Delete 鍵刪除字串，不過第一個 Icon 不能被刪除，會出現錯誤訊息。

命名方式建議取有意義的方式，例如假設是在 UIFlowWndPhoto.xmd 下面，建議取名 UIFlowWndPhoto_StatusICN_XXX。

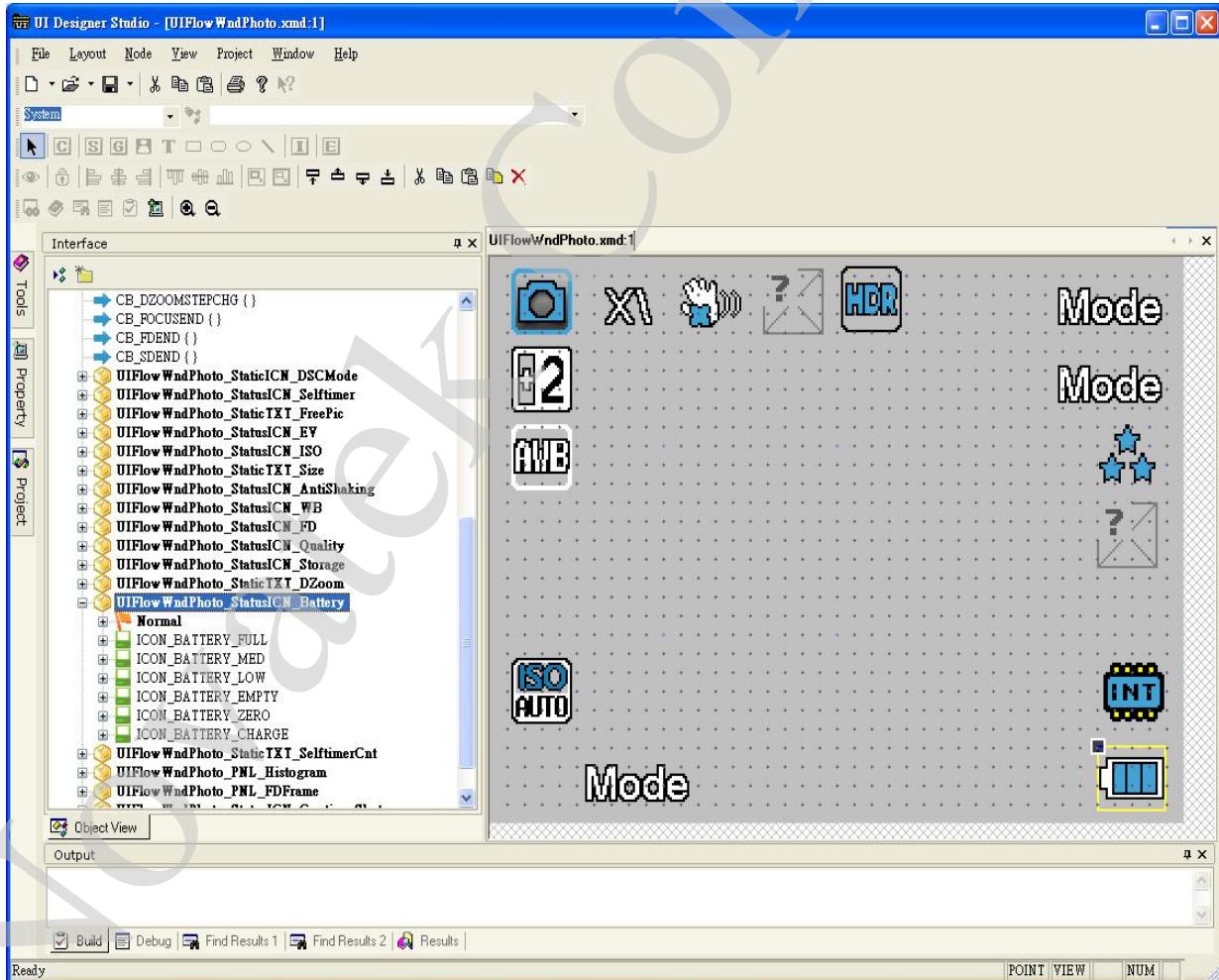


圖 3.7.8

3.7.5 Menu 元件、簡介與程序中 Tab Menu 的關係

■ Menu 簡介

由於 Menu 是整個設計的主軸，這邊將比較詳細說明 Menu 元件，採用的是 Turnkey UI。通常我們將 Menu 定義有 Page、Item、Option Menu 三種，這種的設計是屬於大部分菜單的操作都是相同的，可以使用 Common Menu Window 來設計，如下圖 3.7.9 則為 Page Menu 和 Item Menu；圖 3.7.10 則為 Option Menu。

Page Menu 有分 Movie 和 Setup menu，兩者操作起來都是相同的，所以可使用 Common Menu Window，然後分別帶入不同的 String ID 和 Icon ID 即可。Option Menu 也是相同的原理。例外的 Menu 如日期和時間設置就必須另外獨立的一個 Menu Window。



圖 3.7.9 Page & Item Menu



圖 3.7.10 Option Menu

一般我們會將 Page、Item、Option Menu 放在 UI Layout 的 UIMenuCommon 文件夾裡面，然後在分出 MenuCommonItem.cmd (包括 Page & Item)、MenuCommonOption.xmd，但是這邊舉例的 Option Menu (圖 3.7.10)它是疊在 Page & Item Menu 上面的，所以使用另外一個 MenuCommonMix.cmd Wincodw，如圖 3.7.11 所示，然後將這種三種 Menu 集合在一起。除非是像圖 3.7.12 所示，那就是一個單獨的

MenuCommonOption.xmd

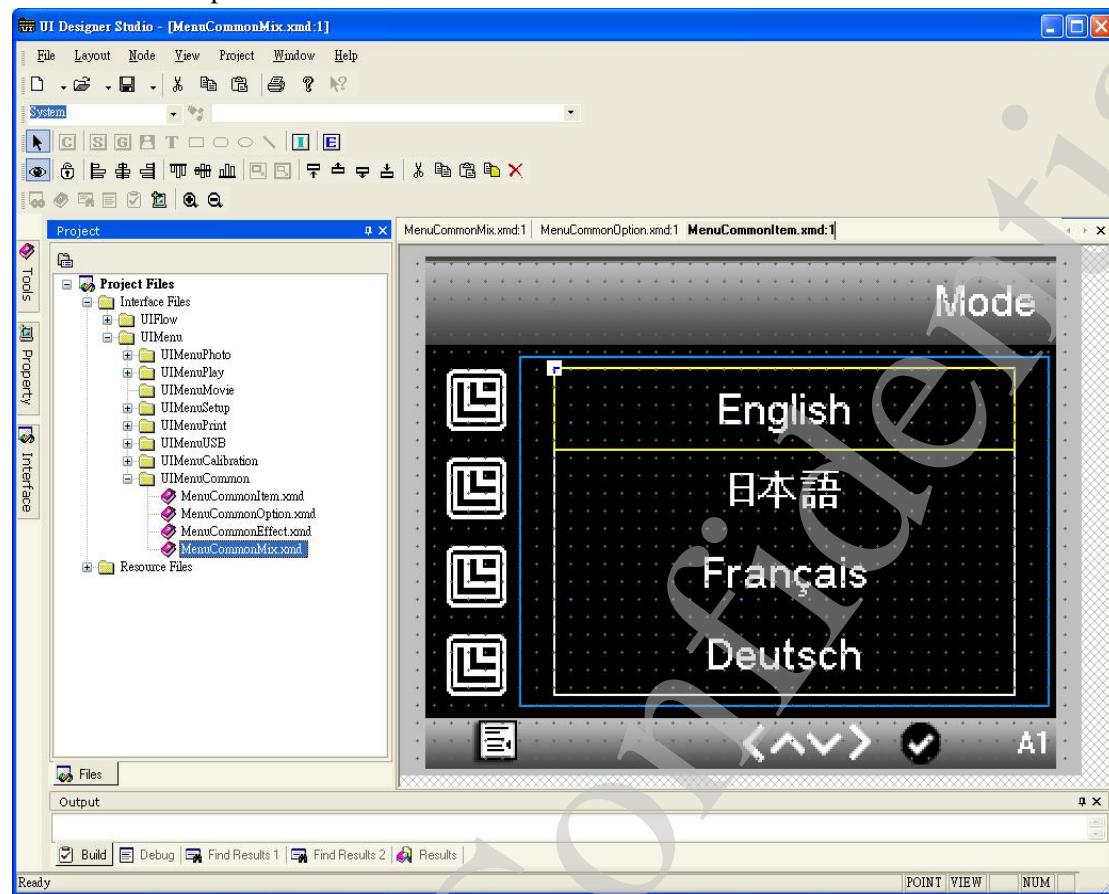


圖 3.7.11 Option Menu 疊在 Page & Item Menu 上，集合在 MenuCommonMix.xmd

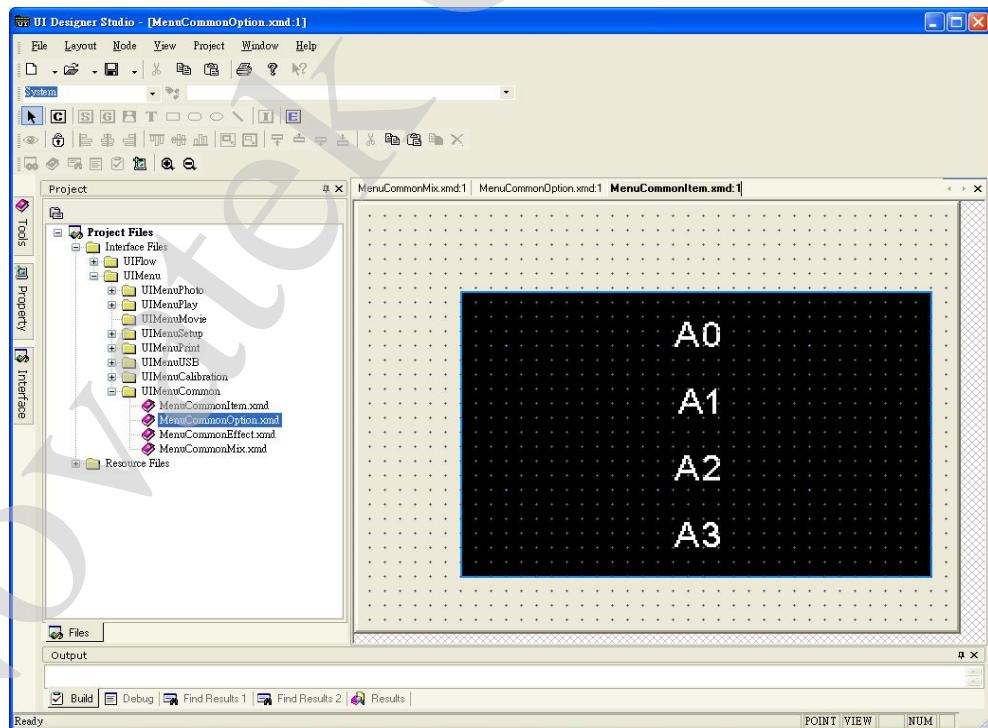


圖 3.7.12 獨立的 MenuCommonOption.xmd

■ Menu : Text 元件

- 這種元件的應用就是如下圖 3.7.13 所顯示的 Option Menu 語言選項。
- 要新增字串時，只要點在任一 STRID_XXX 並按右鍵點選”Clone”即可，但是請指定另一個 STRID，UI Tools 產生 C 程式過程不會出現錯誤，但是在編譯程式時應該會出現錯誤訊息。
- 字串數量請估算一個最大值，因為在程式裡面會計算實際數量，其餘會被 Disable。
- 屬性 View Style 裡面，每頁顯示的 View Number 是 4 個，可以點選 View Scroll 並搭配 Scroll on View Boundary 來模擬實際操作的效果。
- State Icon Source 欄位在 Menu:Text 無作用，它是一個比較特殊的應用，一般當我們選擇到該項目時，通常會用一個底色來標示出來，但是如果我們不要用底色而是要直接將原本的 ICON 換成另一種 ICON 時，可以將這個欄位設定成 IMAGE_LIST，就會自動找下一個 ICON 來顯示，不過命名是有限制的，如原本的 ICON 名稱是 ORIGINAL_ICON，那它的對應名稱必須是 ORIGINAL_ICON_S，這樣才能達到效果。因為這樣的命名在轉成 C code 時，ICON ID 才能連續。

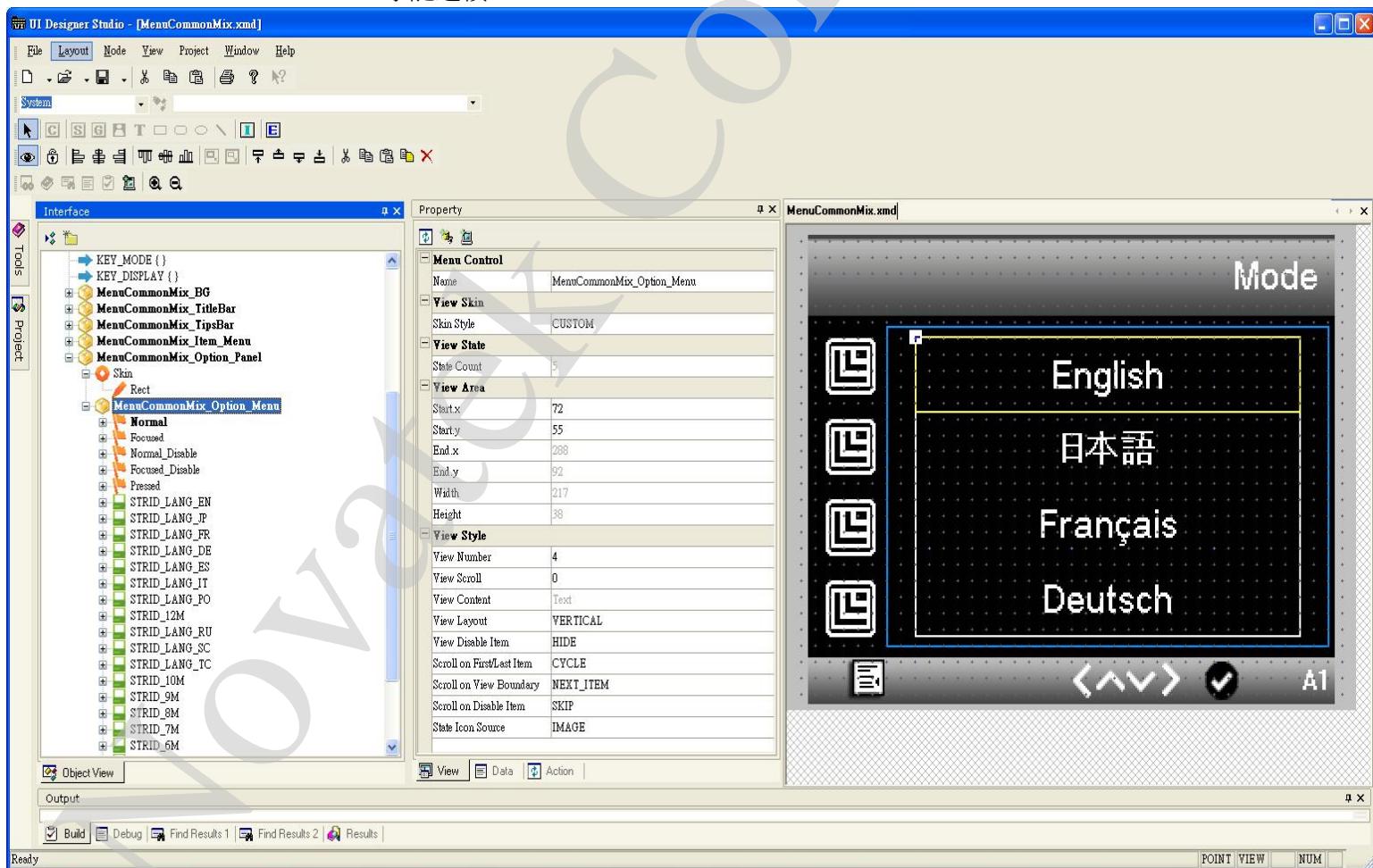


圖 3.7.13

- 圖 3.7.14 顯示 Normal、Focused、Normal_Disable、Focused_Disable、Pressed。
- Normal 和 Focused 分別代表沒有被選到或是被選到的時候。
- Normal_Disable 代表被沒有選到且這個選項是被 Disable 的，可搭配 View Style 的 View Disable Item 這個欄位來決定是被顯示或隱藏，通常會選隱藏。另外如果選擇隱藏，那 Scroll on Disable Item 這個欄位設定是無效的，因為在產生程式的時候，是會變成 SKIP 狀態。
- Focused_Disable 代表被選到且這個選項是被 Disable 的，可搭配 View Style 的 View Disable Item 這個欄位來決定是被顯示或隱藏，通常會選隱藏。另外如果選擇隱藏，那 Scroll on Disable Item 這個欄位設定是無效的，因為在產生程式的時候，是會變成 SKIP 狀態。
- 多了 Normal_Disable 和 Focused_Disable 的選項通常是依據客戶不同 Menu 的需求來設計，一般是不用到，如果顯示要設計成跟 Normal 或 Focused 相同也是可以，只是意義不大。
- Pressed 這個選項是給 Touch Panel 所使用的。

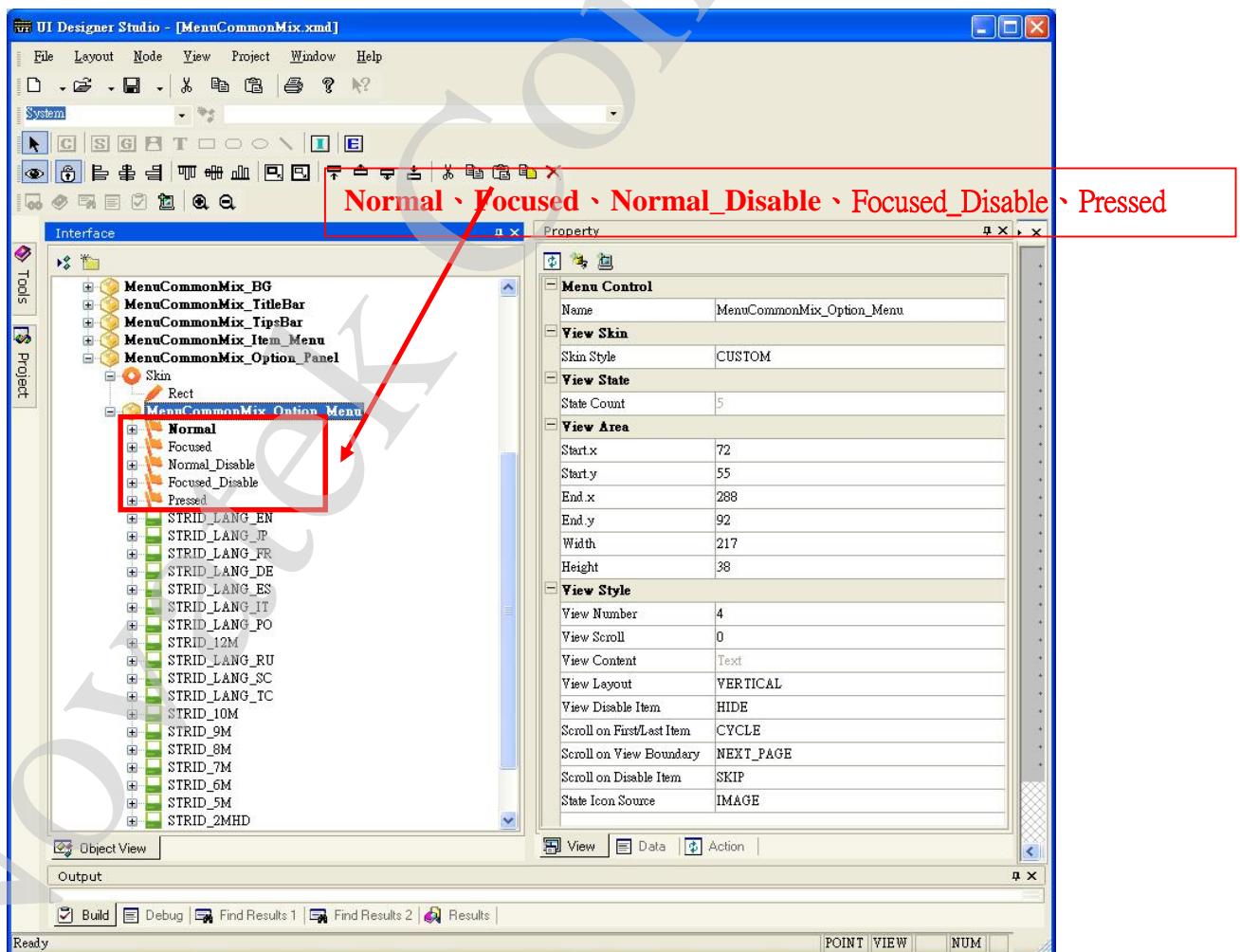


圖 3.7.14

- Normal、Focus、Normal_Disable、Focus_Normal 都有各自的 Skin 與 Text 兩種屬性，如圖 3.7.15 所示。

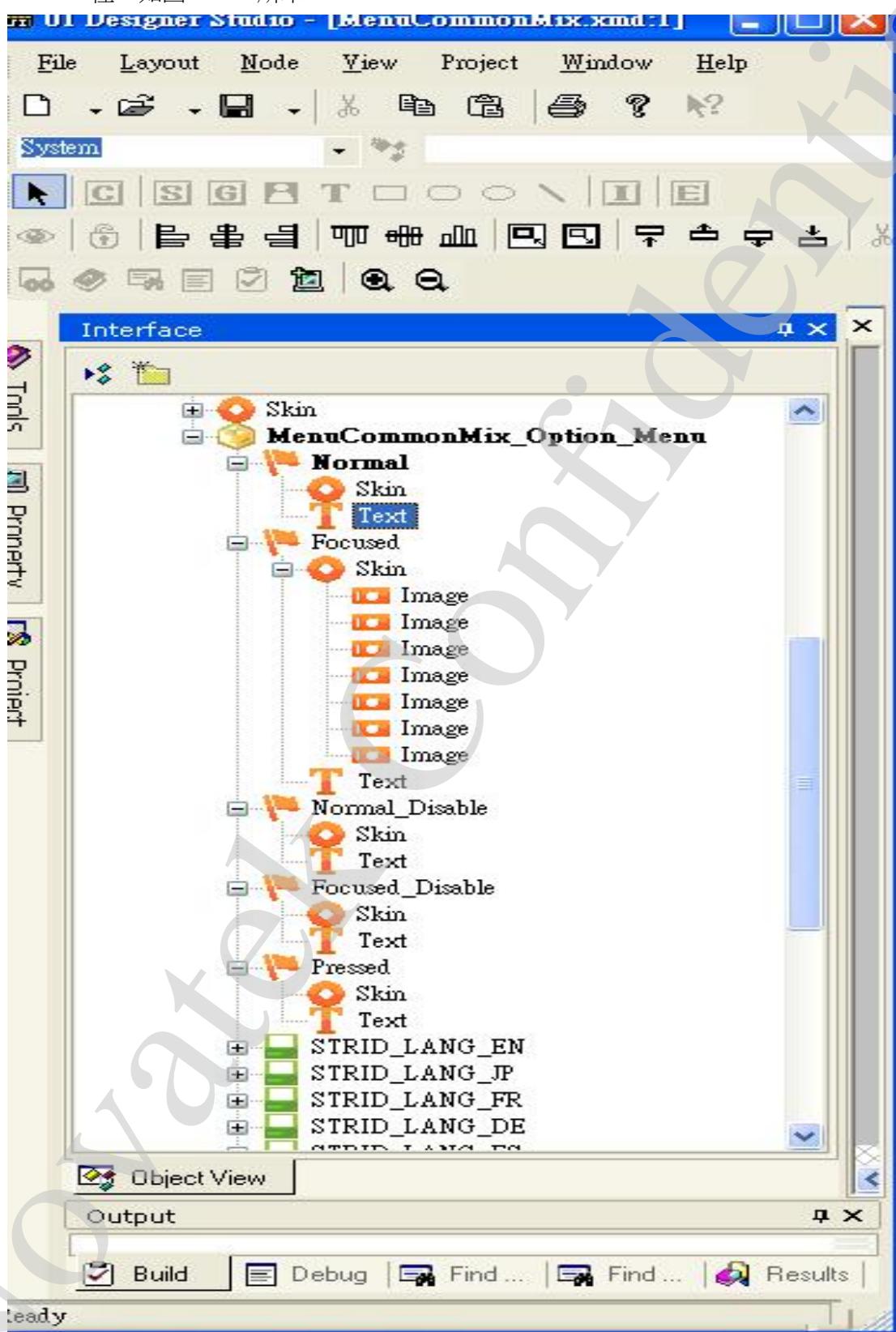


圖 3.7.15

■ Skin 就是要顯示文字之前，是否要先畫個底色、畫矩形、橢圓形、插入 Image 等，通常這只會用在 Focused 的時候，如圖 3.7.15 就是從 Normal 到 Focused 的時候，可看圖 3.7.15 的 Focused 的 Skin 下面有插入 Image。

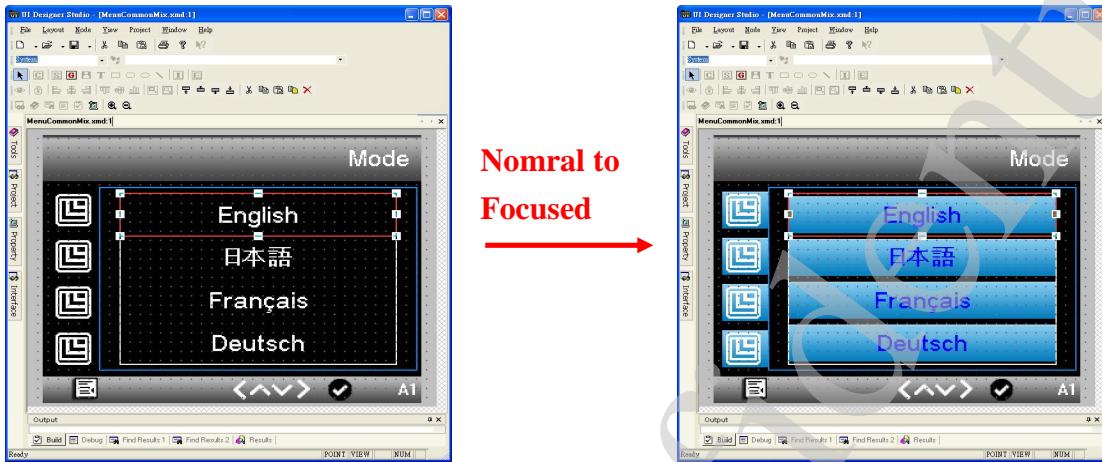


圖 3.7.15

■ 圖 3.7.16 是 Text 屬性：

- Font 可以選擇 Default 或是 DemoKit_Font，但是 DemoKit_Font 才是最終在 LCD 上看的效果，在 UI Layout 可以輸入不同 Font 看效果，但程式中只能帶一種 Font 進去。
- FontStyle、FontSize 目前設定無效，因 Code 無對應的效果。
- TextColor 可設定文字顏色。
- ShaowColor、LineColor 目前設定無效，因 Code 無對應的效果。
- Layout 可以選擇 Single Line 或是 Multiple Line，Multiple Line 就是超過 View Area 寬度設定時會自動換行顯示。
- LineHeight、LetterSpace、IndentSpace 目前設定無效，因 Code 無對應的效果。
- Clipping 可以設定 ON/OFF，設定 OFF 時就是超過 View Area 設定的寬高時，還是會顯示，只是可能會蓋到附近顯示的元件。
- Align.x 就是可以選擇文字是靠左、靠右、還是置中。
- Align.y 就是可以選擇文字是在上方、下方、還是置中。

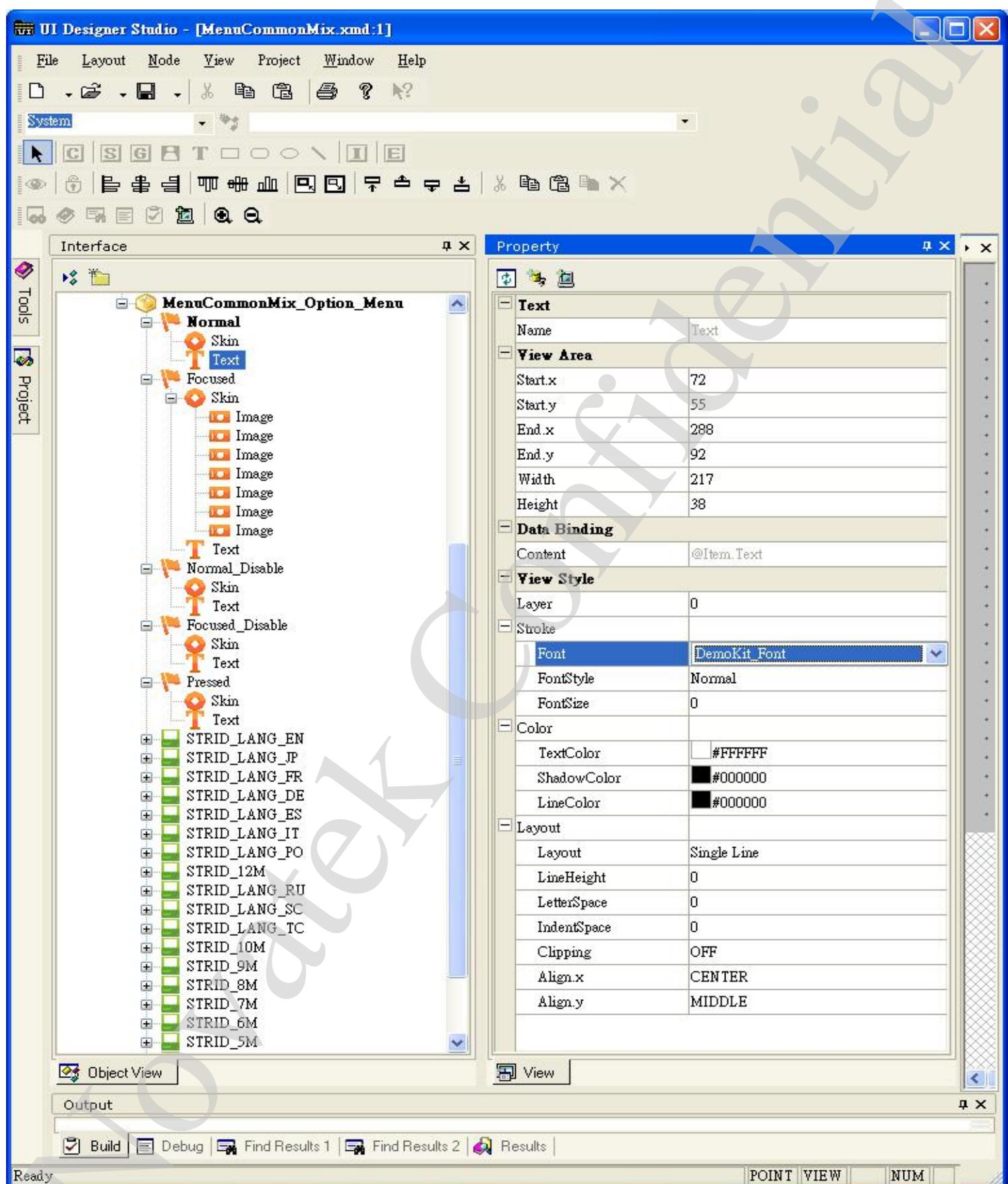


圖 3.7.16 Text Property

■ Menu : Icon.Text 元件

這個元件其實跟 Menu:Text 類似，只是 Normal、focused、Normal_Disable、Focused_Disable 和 Pressed 多了 Icon 屬性，如圖 3.7.17 所示。

- Property 的 Stroke 部分，其中 ImageEffect 欄位有分 COPY、INVERT、COLORKEY，COPY 代表將整個 ICON 複製過去；INVERT 目前無作用；COLORKEY 代表某一個顏色不畫，可由 ColorKey 這個欄位來指定顏色。
- ColorMap 欄位代表指定另一個 Color index table，例如，原本 ICON 的 Color index 0 是代表灰色，換到另一個 Color index table，index 0 可指定其它顏色。

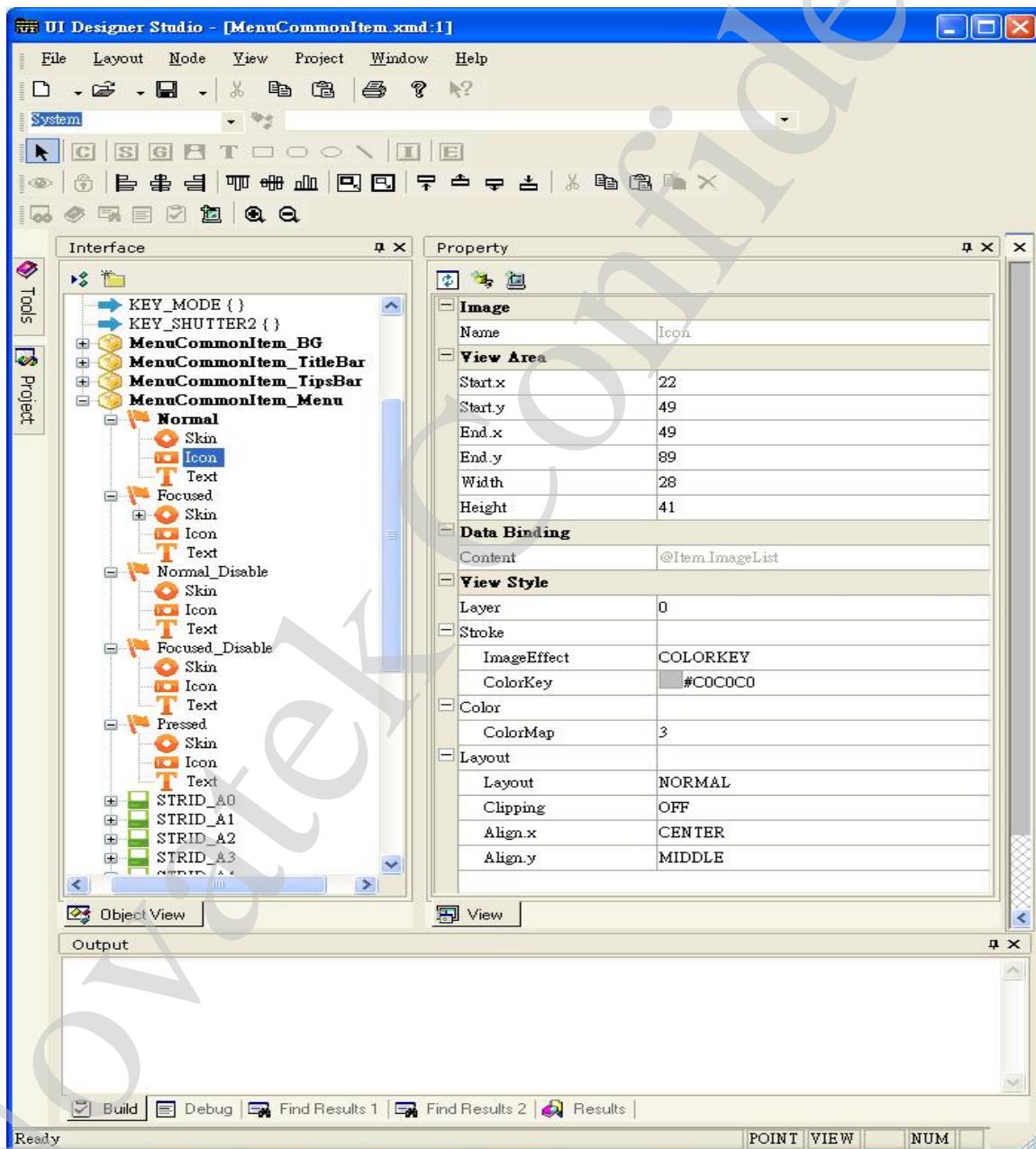


圖 3.7.17

■ Menu : Icon.Text.[Text]和Menu : Icon.Text.[Icon]元件

這兩個元件在 Normal、Focused、Normal_Disable、Focused_Disable、Pressed 就多了”TextValue”和”IconValue”屬性，如圖 3.7.18 所示。

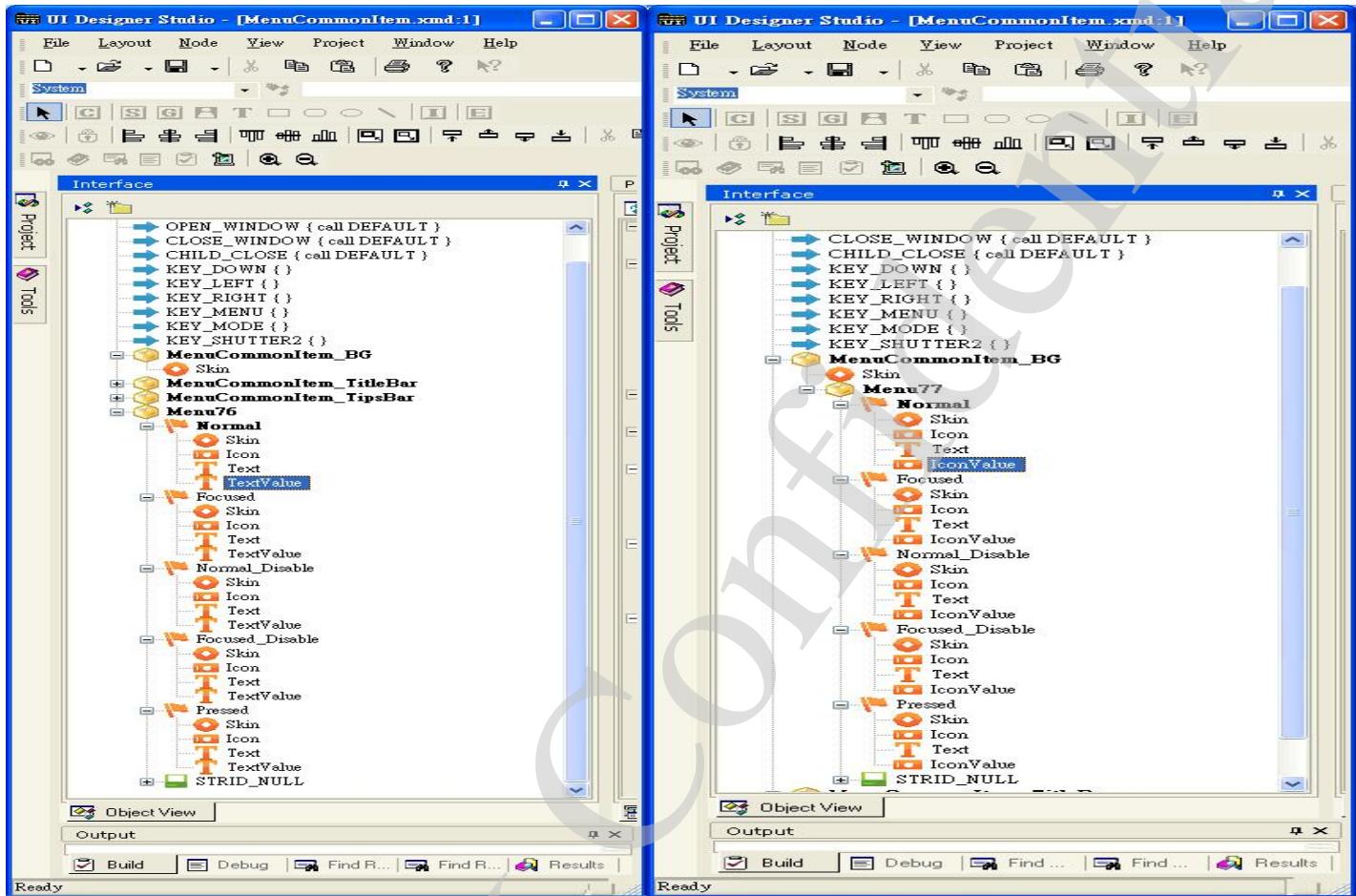


圖 3.7.18

■ 這兩個值通常不在 UI Layout 先指定 String ID 或是 Icon ID，而是在寫程序的時候，才去指定。舉例來說，想在圖 3.7.19 的 Item menu 的右邊加入文字說明或是打勾的 Icon 符號，就可以使用”TextValue”和”IconValue”來實現。



圖 3.7.19

■ **Menu:Text.[Text] 、 Menu:Icon.[Text] 、 Menu:Text.[Icon] 、 Menu:Icon.[Icon]**

這些元件跟之前介紹過的元件屬性的設定都是相同的，就依照當時菜單設計的需求，選擇適合的元件來畫 UI Layout。

■ **Menu 元件與程序中 Tab Menu 的關係**

- Tab Menu 系統是一種非常好用的結構。它可以用少數幾行 code，就把整個 menu 內容，定義得清清楚楚。這個結構是可以和 UI Framework 結合使用的。通常我們只要定義兩個 Common Menu，一個 for Item (包括 Page);一個 for Option，要用時置換其內容即可。不過可視當時 Menu 需求的情況，可以將 Item 和 Option 整合在一個 UI Window 或是將 Page 、 Item 、 Option Menu 三種都分開。Tab Menu 負責定義 Menu 內容，UI Tools 用來畫圖及定義 Menu 行為，然後在 UI Framework 架構下寫程序來達到我們想要的目標，這樣結合後，製作 Menu 將非常方便。
- TabMenu.h 是定義整個菜單系統的架構，包括 Page 、 Item 、 Option Menu 。
- TabMenu.c:
 - TM_CalcItemCount 函數只是計算每個 Page Menu 下的 Item Menu 有多少個 Item 。
 - TM_SetMenu 函數所定義的 g_pTabMenu 變數是用來記錄各個模式(Movie 、 Photo 、 Playback 、 Setup)下的的菜單，切換到不同模式需重新帶入 Menu 。
 - TM_CalcItemCount 函數只是計算每個 Page Menu 下的 Item Menu 有多少個 Item 。
 - TM_SelectTab 和 TM_ShiftTab 函數只是做更換 Page Menu 的動作。

■ **Tab Menu 系統套用到 Menu 元件**

- 按照目前 Tab Menu 系統的程序架構，我們通常將各個模式的菜單系統的 Page Menu 、 Item Menu 、 Option Menu 定義在 MenuMovie.c 、 MenuPhoto.c 、 MenuPlayback.c 、 MenuSetup..c 。
- 下面是 TabMenu.h 所定義的結構，這個結構有定義一個 Text ID 跟兩個 Icon ID ，如果都有用到，剛好就可以對應到 UI Tools 中的 Menu:Icon.Text.[Icon]。
 - 假設目前的 Common Menu 只有 Item (包括 Page Menu) 和 Option Menu ，那請參考 MenuCommonItem.c 的 MenuCommonItem_UpdateContent 函數和 MenuCommonOption_UpdateContent 函數，它只會計算實際使用的 Item 和 Option ，剩下的會被 Disable 並隱藏起來。
 - 假設目前的 Common Menu 是把 Page 、 Item 、 Option Menu 三個集合起來，可以

參考 MenuCommonMix. 的 MenuCommonMix_Item_UpdateContent 函數和 MenuCommonMix_Option_UpdateContent 函數。

```
// Page: a Page includes several Items
typedef struct _TM_PAGE
{
    UINT16    TextId;      // Tab text    對照 UI Tools Menu:Icon.Text.[Icon]
    UINT16    IconId;      // Tab icon (enable) 對照 UI Tools Menu:Icon.Text.[Icon]
    UINT16    IconIdX;     // Tab icon (disable) 對照 UI Tools Menu:Icon.Text.[Icon]
    INT16     SelItem;     // current selected Item (start with 0)
    INT16     FirstItem;   // first Item for display
    UINT16    Count;       // Item number
    TM_ITEM*  pItems;     // point to an Item
} TM_PAGE;
```

■ Tab Menu 系統結合 Menu 元件的擴充性

這邊探討一種比較特殊的 Menu 系統設計，如何將 Tab Menu 系統與 Menu 元件結合擴充來達到目的。

- 假設圖 3.7.20 的 Movie Setting 的 Page Menu 有四個 Icon 跟一個 Text，那假設下一個 Setup Setting 的 Page Menu 也是這樣的設計。那 Menu 元件就必須要用兩個，一個採用 Menu:Icon.Text.[Icon]；另一個採用 Menu:Icon.[Icon]。
- Tab Men 的 Page Menu 結構也要跟著修改，至於 Icon 要指定到哪一 Menu 元件，就是當時的需求自行分配即可。

```
typedef struct _TM_PAGE
{
    UINT16    PageId;      // Page ID
    UINT16    TextId;      // Tab text
    UINT16    IconId[4]; // Tab icon →擴充至 4 個 Icon
    INT16     SelItem;     // current selected Item (start with 0)
    INT16     FirstItem;   // first Item for display
    UINT16    Count;       // Item number
    BOOL      bCustom;    // Custom process?
    TM_ITEM*  pItems;     // point to an Item
} TM_PAGE, *PTM_PAGE;
```

- Tab Men 的 Page Menu 結構也要跟著修改，至於 Icon 要指定到哪一 Menu 元件，就是當時的需求自行指定即可。
- Item、Option Menu 的設計也是相同的原理。



圖 3.7.20 此 Menu 設計需要 1 個 Text 與 4 個 Icon

3.7.6 List 元件

List 元件的行爲基本上都跟 Menu 元件相同，就單純從 List 文字來看，如果菜單是圖 3.7.21 的設計，只有被選擇到項目才會被 Highlight，其實就可以考慮使用 List 元件來實現，同時 Icon 的變化，也可以使用前面提到的 Menu:Icon 元件 Property 的 View Style 的 State Icon Source 欄位來使用 IMAGE_LIST 來實現。

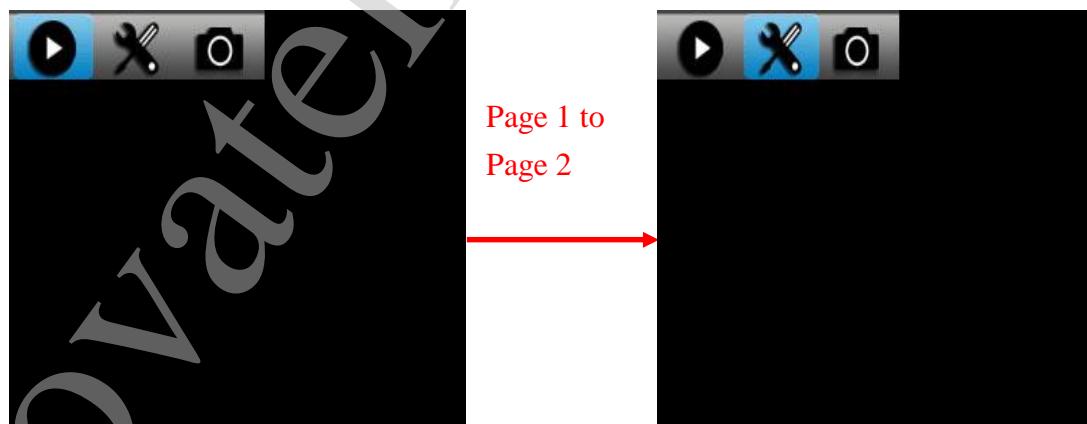


圖 3.7.21

3.7.7 Scroll Bar 元件

Scroll Bar 的功能類似 Windows 作業系統的檔案總管右邊的 Scroll Bar，典型的應用就是搭配 Menu 元件設計，如圖 3.7.22 所示。



圖 3.7.22

3.7.8 Slide Bar 元件

這個元件被應用過的地方就是在預覽模式做 zoom 的動作，變化如圖 3.7.23 所示。



圖 3.7.23

3.7.9 Progress Bar 元件

這個元件被應用過的地方就是在預覽模式做 zoom 的動作，變化如圖 3.7.24 所示。



圖 3.7.24

4. UIFramework 簡介

當使用 UI Tool 畫完 UI Layout 後，轉成 C 語言程式，是在 UIFramework 架構下執行，這邊不詳細解說運作原理，只說明上層 UI 在撰寫程序時，有哪需要注意的地方。

4.1 Window Open and Close

■ OnOpen 裡可不可以再 open window? 如果可以，要注意什麼?

可以，但要注意 onOpen 再 open window 後，會繼續執行 OnOpen 裡的動作後，才算做完這個 event，之後才 redraw。

Ex: FlowPhoto_OnOpen()

```
{  
    Ux_OpenWindow((VControl *)(&FlowSelftimerCtrl), 1, SELFTIMER_ICONCNT_10SEC);  
    UxCtrl_SetShow(&SelftimerCtrl, FALSE);  
    UxCtrl_SetShow(&EVCtrl, FALSE);  
    UxCtrl_SetShow(&FlashCtrl, FALSE); → 執行完成後才會 redraw  
}
```

■ Ux_OpenWindow 會呼叫 OnOpen, 那 close window, 除了會呼叫 OnClose, 還會呼叫什麼?

還會呼叫 parent 的 OnChildClose()

Ex: Ulflow window 上面 create menu window, close menu window 的話會先執行 menu 的 OnClose(), 再執行 Ulflow 的 OnChildClose(), 如果是直接 close Ulflow window 會怎樣? UIFramework 會自動從最上層一層層 close, 所以呼叫順序會是 menu_OnClose() -> UIFlow_OnChildClose() -> UIFlow_OnClose 另外 Ux_OpenWindow 跟 Ux_CloseWindow 這兩各 API 特別要注意的是這兩支 API 都是直接呼叫, 做完才結束。

■ 在 OnChildClosed, 如何知道這次是從 child close window 的，還是直接 close root window，這次的 OnChildClosed 只是經過，但不需重畫?

Ux_Redraw 會自己判斷這次的 close 是不是被強制關掉，而略過這一個 window 的 redraw，user 若想知道可以呼叫 Ux_IsForceCloseWindow 知道。

■ 不要在一個 OnXXEvent (如 OnKeyEvent)去 Close 目前的 Window 後，在去 open 另一個 Window，這會潛在不確定的風險。

下面的 WND open 和 close 的用法是不正確的，不能在藍色部份關掉現在的 Window 後又去 Open (紅色)另一個 Window，應該要回到 OnChildClose 之後再去

Open Window。這個動作假設在 UIFlowWndphoto 開啓 Setup Menu, 然後又開啓 Setup Format Menu, 然後又開啓 Format Confirm Menu, 但是在 open format confirm menu 之前, 已經關掉 Setup Menu 跟 Setup Format Menu, 因為這不是在 UIFlowWndphoto 去 open format confirm menu, 這樣 UIFramework status record 會有問題。

```
INT32 UIMenuWndSetupFormat_Item_OnKeyMenu(VControl *pCtrl, UINT32 paramNum,  
UINT32 *paramArray)  
{  
    INT32 format_index;  
    format_index = UxMenu_GetData(&UIMenuWndSetupFormat_ItemCtrl,MNU_CURITM);  
    Ux_CloseWindow(&UIMenuWndSetupFormatCtrl,0);  
    Ux_CloseWindow(&UIMenuWndSetupCtrl,0);  
    switch(format_index)  
    {  
        case UIMenuWndSetupFormat_Item_STRID_OK:  
            Ux_OpenWindow(&UIMenuWndSetupFormatDefaultConfirmCtrl, ,UIMenuWndSetupFo  
rmatDefaultConfirm_StatusTXT_Msg_STRID_FORMAT_QUERY);  
            break;  
        case UIMenuWndSetupFormat_Item_STRID_CANCEL:  
            break;  
    }  
    Ux_DefaultEvent(pCtrl,NVTEVT_KEY_MENU,paramNum,paramArray);  
    return NVTEVT_CONSUME;  
}
```

- 要有一個基本 Window 後 (如 move/photo/playback 都會有一個 UIFlow Window), 在去開啓另一個 Window, 不要在開啓第二個 Window 之前, 先關掉第一個基本的 Window。

4.2 Control View/Data/Event

- 如果用 **UxCtrl_SetShow** 來設定 Window Hide 會怎樣?

不會怎樣, 目前不 **support hide widow**, 要不就 **close** 這個 **window**, 要不就用一個 **Rectangle** 或使用 **Panel** 元件把畫面蓋起來, 因為 **hide window** 時, **UIFramework** 無法幫 **user** 決定要畫 **parent window**, 還是不畫, 或把畫面清空。

- 如何在 code 裡面共用相同種類元件的 skin?

A control 跟 **B control** 要共用 **skin**，可以使用 **UxCtrl_SetShowTable(B control, UxCtrl_GetShowTable(A control))**; 用 **UxCtrl_GetShowTable(A control)** 拿到 **A control** 的 **skin** 再用 **UxCtrl_SetShowTable(B control)** set **B control** 的 **skin** 另外也可以用 **UxCtrl_SetDataTable** 置換 **data** 內容。

■ 如何在 code 裡面動態修改元件的 **data**?

可以用 **UxCtrl_SetItemData** 修改單一 **data** 的內容，也可以用 **UxCtrl_SetDataTable** 置換整個 **data** 的內容。

■ 呼叫 **Ux_DefaultEvent(pCtrl, NVTEVT_XXX)**; 跟沒呼叫 **defaultEvent** 的差別是什麼？

Ux_DefaultEvent 會執行 **pCtrl** 的 **default** 行為，例如：**OnOpen** 裡面 **Ux_DefaultEvent**，會執行 **window OnOpen** 的 **default** 動作(**set all child control dirty**)，沒有呼叫 **default**，**user** 就要自己 **dirty child**，這樣 **redraw** 才會執行到 **child** 的 **redraw function**。

■ 如何在 code 裡面自己加 Event Handler?

可在 **generate code** 的 **xxx.c** 檔找到各個 **control** 的 **event table**，只要依格式加上去就可以了。

```
EVENT_BEGIN(FlowPhoto)
EVENT_ITEM(NVTEVT_KEY_UP, FlowPhoto_OnKeyUp)
EVENT_ITEM(NVTEVT_KEY_DOWN, FlowPhoto_OnKeyDown)
EVENT_ITEM(NVTEVT_CB_JPGOK, FlowPhoto_OnJpgOK)
EVENT_END
```

在 **implement FlowPhoto_OnJpgOK()**

4.3 Redraw

■ 畫面什麼時候重畫？

- (1) 每次 **Ux_PostEvent(NVTEVT_xxxx, 0)**; 做完，**UIframework** 會自動呼叫 **Ux_Redraw()**。
- (2) 如果希望這次 **PostEvent** 不要重畫可以呼叫 **UI_Set_NotFlip()**，下一次的 **redraw** 畫面就不會改變，此 **API** 為一次有效，也就是再下一次的 **redraw** 就會重畫
- (3) 如果在 **PostEvent** 外想要重新畫畫面可以自己呼叫 **Ux_Redraw()**

■ status or static control 畫出來，change state 後有上一次 icon 殘影，要 check 什麼？

通常是 **UI tool** 在畫這個 **control** 時，**Skin** 的部份，沒有 **Fill** 成透明色 (打開 **tool**，選這個 **control** 的 **skin**，修改 **property** 裡的 **Fill style -> Fill**，在選

BackColor)。

- 在每個 WND 裡面都會有一個 XXX_OnRedraw 的 event，如下面的程序，**Ux_DefaultEvent** 的一定要做，不然 UIFramework 會認為不需要做 redraw 的動作，這時候如果你需要去隱藏一個 ICON，這個隱藏動作就不會做。另外一般是不需去產生這個 function 的，除了像 Photo mode 的人臉偵測需要去畫 OSD 框，才使用 OnRedraw，其它的情形都不需要 OnRedraw function。

```
INT32 UIFlowWndPlayStill_OnRedraw(VControl *pCtrl, UINT32 paramNum, UINT32
*paramArray)
{
    Ux_DefaultEvent(pCtrl,NVTEVT_REDRAW,paramNum,paramArray);
    return NVTEVT_CONSUME;
}
```

4.4 Send/Post/Flush Event

- post event 跟 sent event 有什麼不一樣？

post event 是其他 **task** **post event** 給 **UIframework** 的 **message queue**(有 **task switch**)
send event 是直接 **send** 給 **control**，看這個 **control** 是否處理這個 **event**(沒有 **task switch**，做完才出來，相當於 **function call**)。

- post event 誰收？

UIframework 收到 **event** 後會 **dispatch** 給 **current window** (最上層的 **window**)
可以用 **Ux_GetFocusedWindow()** 知道現在最上層的 **window**，再 **print "Name"**
Ux_GetFocusedWindow(&pWnd); pWnd->Name，現在 **UIframework dispatch** 給
current window，**window** 有收這 **event** 時，就會印出 **message(ex: WarnWnd evt 2)**，
沒有印出來，表示 **current window** 沒有收這個 **event**。

- 如果 post event 後，沒動作，要 check 什麼？

(1) check **current window** 有沒有收這個 **event**。

(可以看一下 **uart message** 有沒有收到這 **event**，" **FlowPhoto evt 3**"，表示
FlowPhoto 收到 **event 3**)

Ex:post event NVTEVT_BATTERY 給 **uiflowwndphoto,battery** 沒改變，可以先看看
uiflowwndphoto 的 **event table** 是不是有註冊這個 **event**

(2) 如果有收到，**uart** 也有看到這次的 **redraw**，那就 **check function** 裡有沒有寫錯。

- **Ux_Flush Command** 的作用是？呼叫時要注意什麼？

清掉 **Message Queue** 的 **command**,因為 **UIframework** 用的是 **mail box,event** 會一個個 **Queue** 起來,如果不要執行,可以清掉 **Queue** 裡的 **event** ,常用來 **clear key event** 。

4.5 State Machine 的應用

在 Movie 、 Photo 、 Playback Mode 下都有一個 gMovData 、 gPhotoData 、 g_PlbData 的 Global Variable ,來記錄 State machine 的運作模式,如錄影狀態、拍照狀態、播放播放影片狀態、開啓 Menu 狀態、zoom 狀態、開啓 Warning Menu 狀態等等。請善用這些 State Machine 的運作來達到目標。請盡量不要呼叫 Input_SetKeyMask 函數 (650 平台) 或是 KeyScan_SetKeyMask 函數(220 平台),來控制 Enable/Disable Key 的操作,除了程式的可讀性會變差之外,也會使 Debug 時比較麻煩,同時修復一個問題後可能衍生出另一個問題。

5. Unicdoe 簡介

目前 project 中所要顯示字串的字型資料庫，都是預先作好的，當缺少所需要的字型時，必需要在製作新的字型資料庫，然而當所缺的字型非常多的時候，這將是一個非常消耗時間的人力工作。所以我們將用 Unicode 來取代這種傳統的字庫製作方式，雖然會佔掉在 Flash 儲存的空間，但可以支援所有的多國語言，對於 project 的開發將會有很大的幫助。

另外產生的 Unicode 字庫在早期 Draw library 的 OSD 畫法，會跟現在 Gx library 不同，因此 Unicode 在早期 Drawing library 的資料顯示也會簡介。如果要用在 Gx library 的顯示上，只要使用視窗工具將 String Code book 需要用到的 Font 抽取出來轉成 Gx library 使用的程式即可，這樣做法可以省下使用全部 Unicode 所造成的 FW size 變大。

5.1 Tool for Unicode

圖 5.1.1 是早期的字庫擷取應用程式是用來抓取 Microsoft Windows 的 font，副檔名為 ttf 的 font database，將其放在 C:\WINDOWS\Fonts 底下。這個工具所選擇任何字形，顯示出來的字體都是 Unicode。這個版本的 Font 只有到 32x32，對於 32x240 的 OSD font 是足夠使用的，但如果 LCD 的解析度是到了 640x480 或 720x480，那從 OSD font 320x240 scale 到 640x480，並不是很好看。因此將 Font 從 32x32 擴充到 64x64，如圖 5.1.2 所示。Button 解說如下：

- 設定檔頭和匯入 BIN 無作用。
- 將圖 5.1.1 版本的匯出 BIN button 改成輸出 Bitmap。以往要從 String code book 挑選出 Font 並轉成 Bitmap 是用另一個 API 來執行，現在整合至圖 5.1.2 的字庫擷取程式。

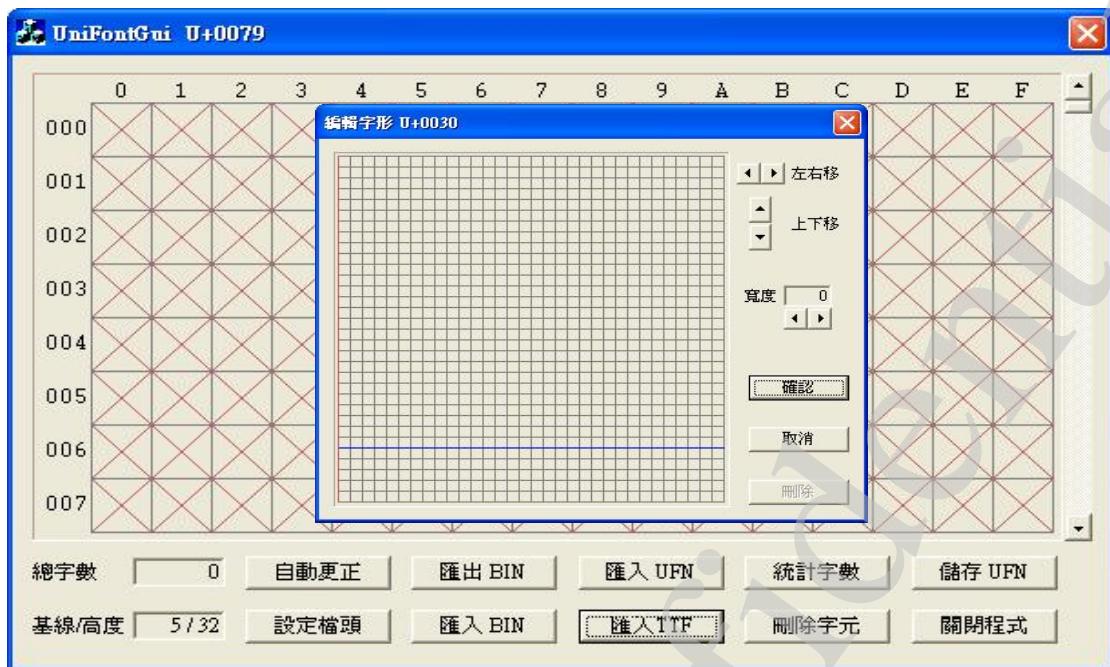


圖 5.1.1 早期的 Unicode 字庫擷取工具, 32x32 font

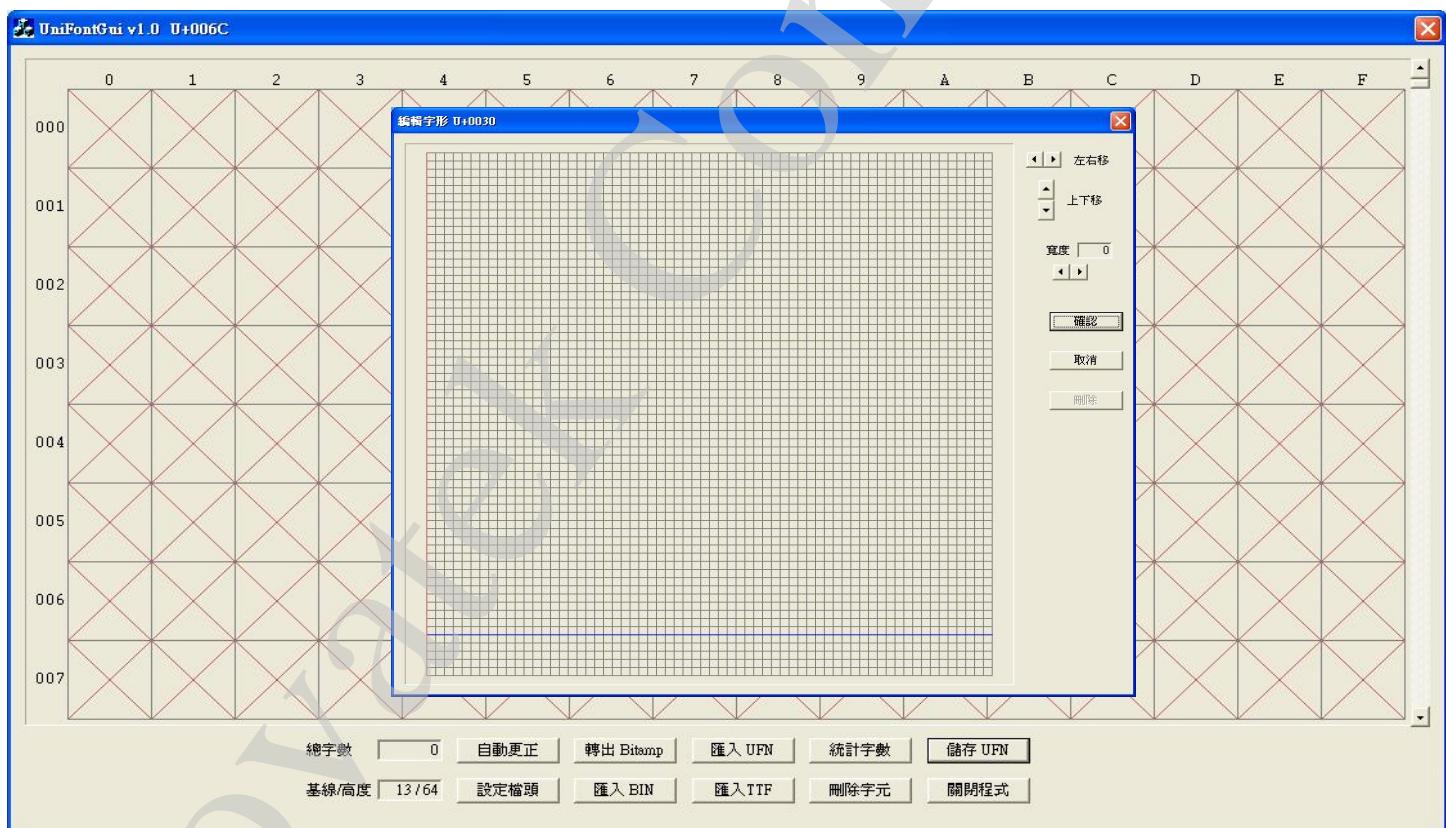


圖 5.1.2 新版 Unicode v1.0 字庫擷取工具, 64x64 font

- 按「匯入 UFN」按鈕。
- 雖然寬度、高度可以選擇，並不代表字型的實際寬度跟高度，因為一個字型的最大寬高只有 64x64，一定落在這個範圍內，所以要取用多少的寬跟高，就是累積使用的經驗來快速選擇想要的字型。
- 選擇所要的字元集(語系)以及字形名稱、高度寬度、重量(字形粗細)，此外，假如勾選強制覆蓋的 button，目前顯示的字型將會被新選擇的字型所覆蓋。



圖 5.1.3

- 關於中文部分有一個地方需要注意，例如，目前的作業系統是繁體中文，則選擇字元集為 GB2312 (簡體中文)，字形名稱將看不到完整的名稱，會顯示”？””，如圖 5.1.4 所示，名稱為”?康?黑，同時也會無法匯入字形，當轉換作業系統至簡體中文時，它是”華康儻黑”字形名稱。



圖 5.1.4

■ 切換作業系統語系: 我的電腦->控制臺->地區語言及選項->

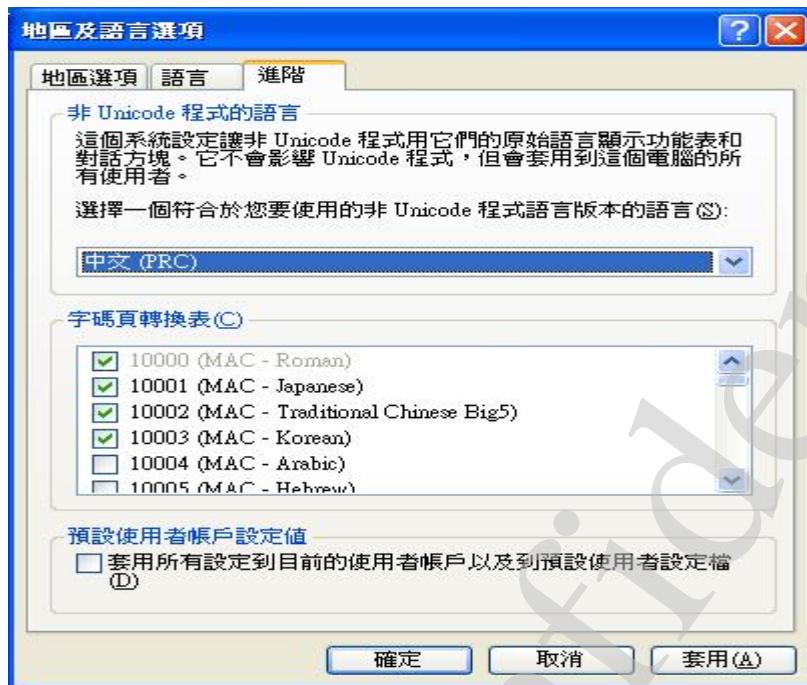


圖 5.1.5

■ 確認字型後，按自動更正的 Button 來修正字型，然後存成 UFN 的 binary file。

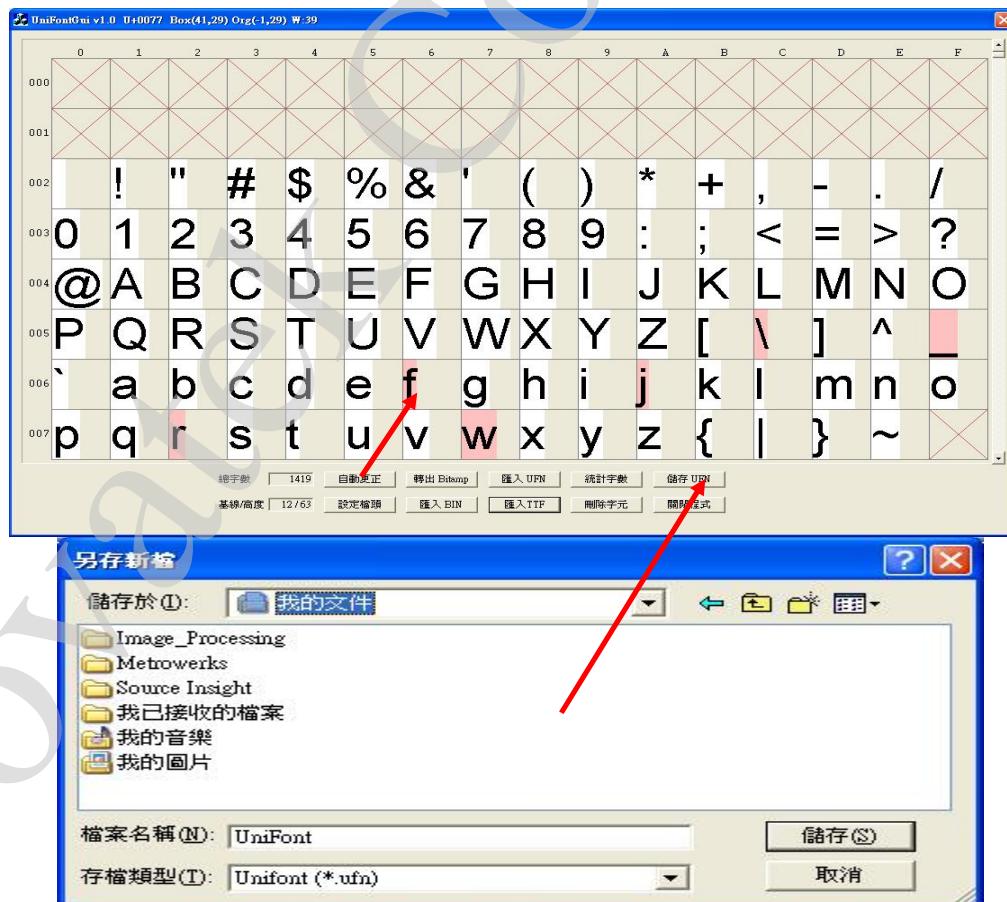


圖 5.1.6

5.2 PStore for Unicode

利用應用程式將 Unicode 轉成一個 binary file，先儲存到 SD card 上，在利用 PStore 相關的 function 將 binary file 儲存到內部 Storage 上，一般是 NAND 或是更大容量的 SPI NOR flash。另外 DRAM 也需要使用一塊 memory pool 來存放整個 Unicode 字庫。

5.3 Unicode display in old drawing library

使用字庫擷取工具所轉出來的每一個 Glyph，因為每一個 scan line 是 4 個 byte，1 個 bit 代表 1 個 pixel，然而掃掉到的每一個 scan line，有很多 bit 數不是代表實際 Glyph 的 pixel，只是為了組成 four byte 而補上的資料，而目前底層的 OSD drawing 方式是用寬度和高度去描繪出的字型，因此寬度資料有可能不是整數的 byte，所以需要額外的處理，例如字型寬度是 13 (這裡用 1 個 bit 表示 1 個 pixel)，高度是 24，代表每一列的寬度資料是 13 個 bit，所以需要用 2 個 byte 來表示，因此屬於 NULL 的 3 個 bit 就需要由下一列來補上，否則呼叫底層 OSD drawing function 去畫出時，在 panel 上看到的將是不正確的字型。

以圖 5.3.1 的 Glyph 為例子，第一列的資料為 0x1F 00 00 00，寬度為 11，因此由左至右的 11 個 bits 為有效的 pixel，剩下的 21 個 bits 就需要另外處理，而 11 個 bit 需要用 2 個 byte 來表示，因此剩下的 5 個 bit 就要由下一列來補上，依此類推，以便能顯示正確的字型。此外這些 Glyph 的寬度和高度都是實際大小，沒有考慮到邊界的問題，所以在顯示一個字串的時候，因為沒有邊界，所以所有的字型顯示將連在一起，因此看起來將不太美觀，故在實際的寬度上，左右兩邊各加一行的空白邊界，即在圖 5.3.1 中 Glyph 實際寬度由 11 增加到 13，如此一來每個字型之間將有間隔存在。

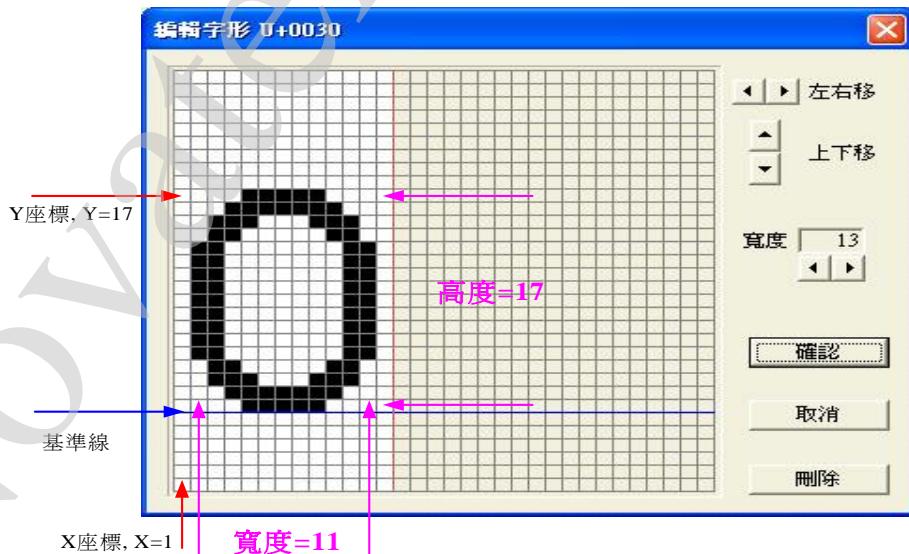


圖 5.3.1

5.4 視窗介面程式挑選出 Font 及轉成 bitmap

由於 Unicode 字庫字型過多，假如全部轉成 C 程式，FW size 將過於龐大，因此只要挑選出此 String Table 會用到的 Font 即可。

■ 圖 5.4.1 是 String table 轉成 C 程式之後，可以另存一個 DemoKit_String_Codebook.TXT 的檔案，此檔案就是這個 String stable 會用到的所有字型。



The screenshot shows a Windows Notepad window with the title 'DemoKit_String_Codebook.TXT'. The content of the window is a list of 256 entries, each consisting of a four-digit hex code followed by a single character. The characters are mostly 'FFF' or '0000', indicating they are likely placeholder or invalid characters. The list starts with '0000 FFFF' and ends with '0025 FFFF'.

Hex Value	Character
0000	FFF
0001	FFF
0002	FFF
0003	FFF
0004	FFF
0005	FFF
0006	FFF
0007	F
0008	FFF
0009	FFF
000A	0000
000B	FFF
000C	FFF
000D	F
000E	FFF
000F	FFF
0010	FFF
0011	FFF
0012	FFF
0013	FFF
0014	F
0015	FFF
0016	FFF
0017	FFF
0018	FFF
0019	FFF
001A	F
001B	FFF
001C	FFF
001D	FFF
001E	FFF
001F	FFF
0020	0000
0021	0001
0022	0022
0023	FFF
0024	FFF
0025	FFF

圖 5.4.1

■ 首先打開圖 5.4.2 的應用程式

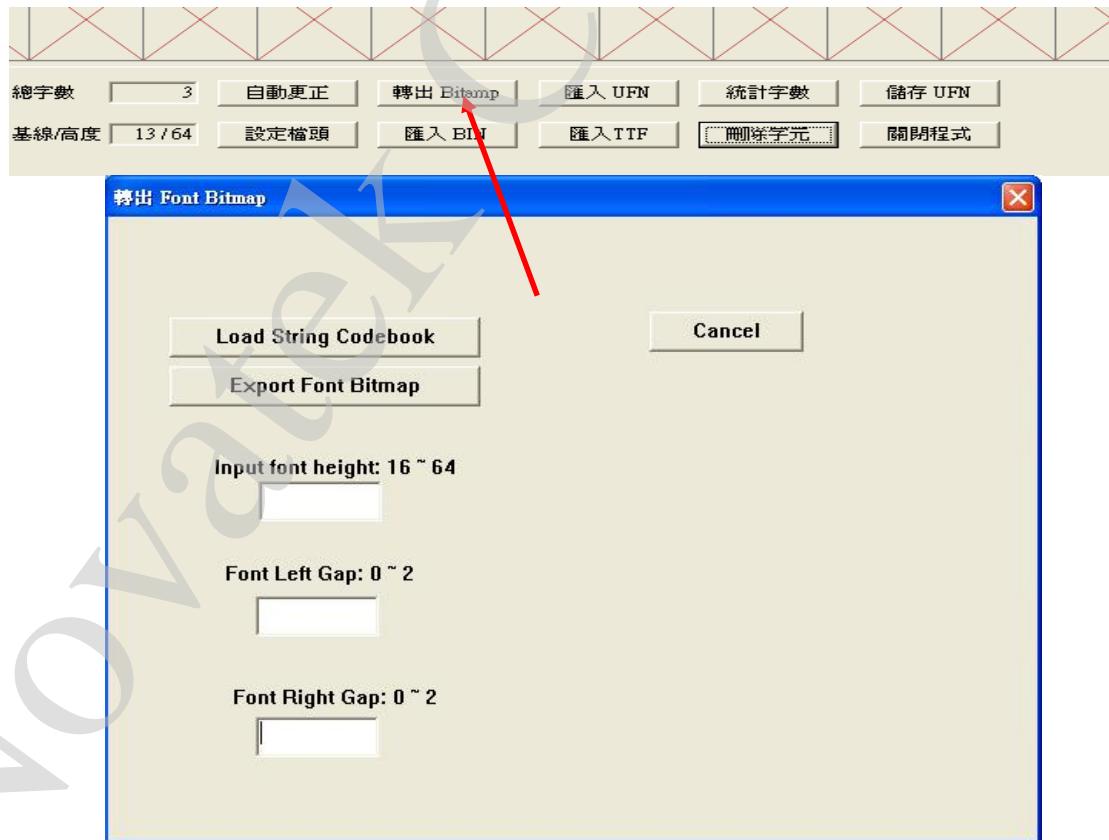


圖 5.4.2

- 要轉出 Font Bitmap，UFN binary file 一定要存在或是匯入 UFN。
- 高度沒有輸入，預設是 0，會要求輸入高度。
- Left and Right Gap 沒有輸入，預設是 0。
- 如果字型寬度已經達到 64，那設定 Left and Right Gap 無作用，預設是 0。
- 如果字型寬度已經達到 63，那設定 Left Gap 無作用，如果 Right Gap > 1 ，Right Gap 預設會是 1。
- 如果字型寬度已經達到 62，Left Gap = 1; Right Gap = 2，那 Right Gap 會預設是 1。
- 如果字型寬度已經達到 62，Left Gap = 2; Right Gap = 1，那 Left Gap 會預設是 1。
- 如果字型寬度已經達到 61，Left Gap = 2; Right Gap = 2，那 Right Gap 會預設是 1。
- 如果輸入的高度小於 font database 的最大高度，將會使用 font database 的高度。Font database 最大高度如圖 5.4.3 所示，這表示轉出的 UFN binary file 所選擇的字型太大。

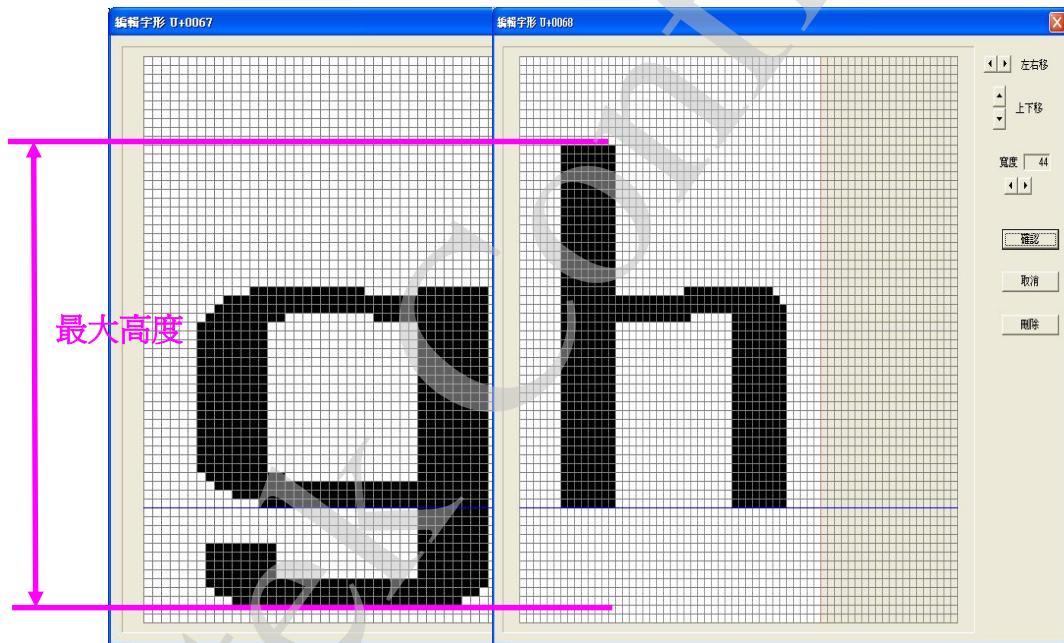


圖 5.4.3

■ Load String Codebook，ANSI 或是 Unicode 格式都可以，如圖 5.4.5 所示。

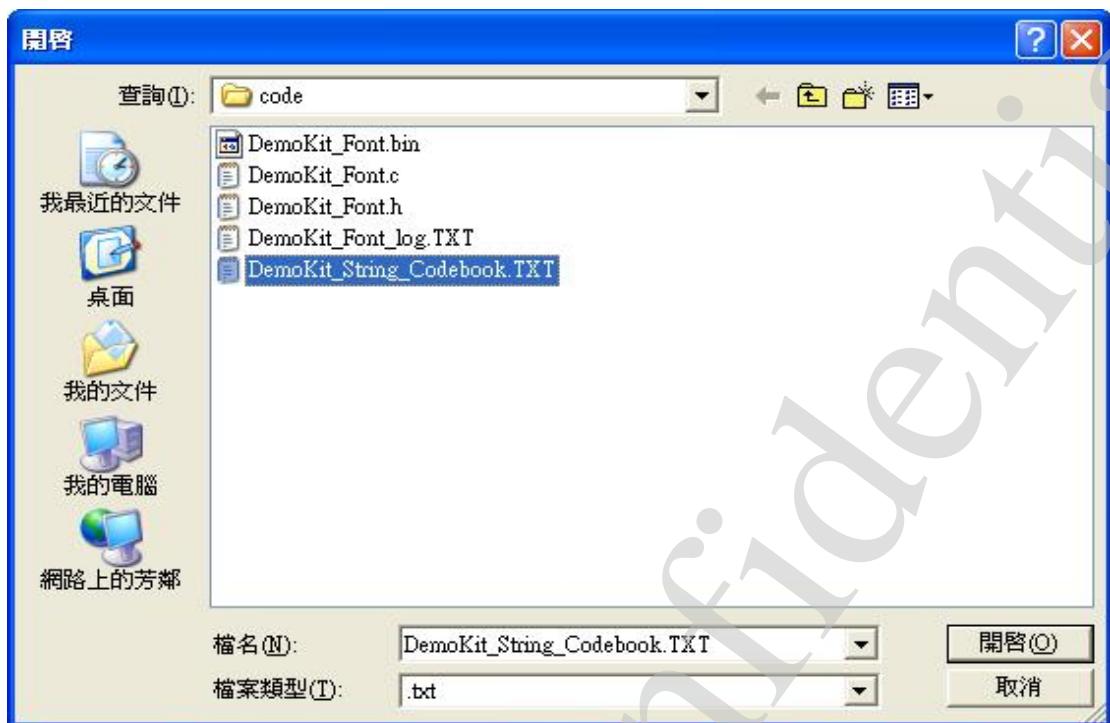


圖 5.4.5

■ Export Font Bitmap，如圖 5.4.6 所示，需要輸入任一檔名，此檔名無意義，只是為了取得一個路徑，這裡範例取名為 font。

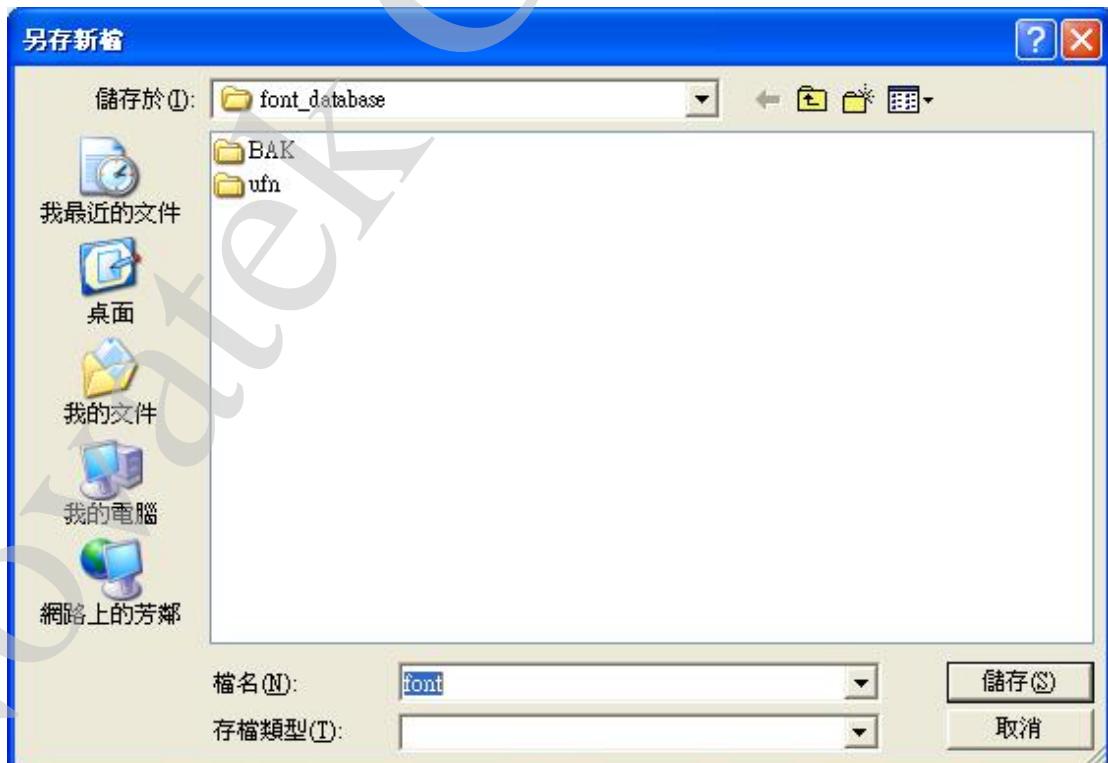


圖 5.4.6