



Application Note

Unicode and UI Designer Studio User Guide for String Table

Revision History

Revision	Date	Author	Changes
1.0	2009/09/03	KS Hung	First draft version porting from Unicode and UI Designer User Guide
1.1	2009/11/16	KS Hung	(1) Delete the original content in chapter 12 for version 1.0. (2) Add the Winodws AP for font bitmap gereration.

目 錄

前言	4
1. Tool for Unicode	5
2. Unicode Format	5
3. PStore	8
4. Display	8
5. String Table Transform	9
(1) 將 string table 轉成 Unicode 格式的 StringTable.txt 文字檔	9
(2) 使用舊的 UI Design Studio 將 StringTable.txt 轉換成 C source code	10
(3) 使用舊的 UI Design Studio 將 font bitmap 轉換成 C source code	19
(4) 使用新的 UI Design Studio Pro 轉換 String Table 和 Font bitmap 到 C source code	26
6. UniFontGui.exe 字庫擷取工具介紹	33
7. Unicode binary file to font bitmap	35
8. 使用 String Codebook 挑選出 string table 在用的 Font	38
9. ANSI CODE to UNICODE (多媒體)	39
10. 背景、外框、主體	39
11. 16MB、32MB DRAM 中執行	40
12. 使用視窗介面程式產生 Font bitmap	43
13. Unicode Porting and MP3 Lyric	46
(1) Unicode Porting	46
(2) MP3 詞曲同步	47

前言：

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 字庫只能用在早期 draw library 的 OSD 畫法，新一代的是 Gx library，但如果要用 Unicode，仍然需要此字庫再進行轉換動作，後續會說明。

1. Tool for Unicode

使用如下的應用程式擷取 WINDOWS 的 Unicode 字庫（最後章節將說明步驟），並轉換成副檔名為 ufn 的 binary file，儲存 UFN 之前先按自動更正的 Button 來修正字型。

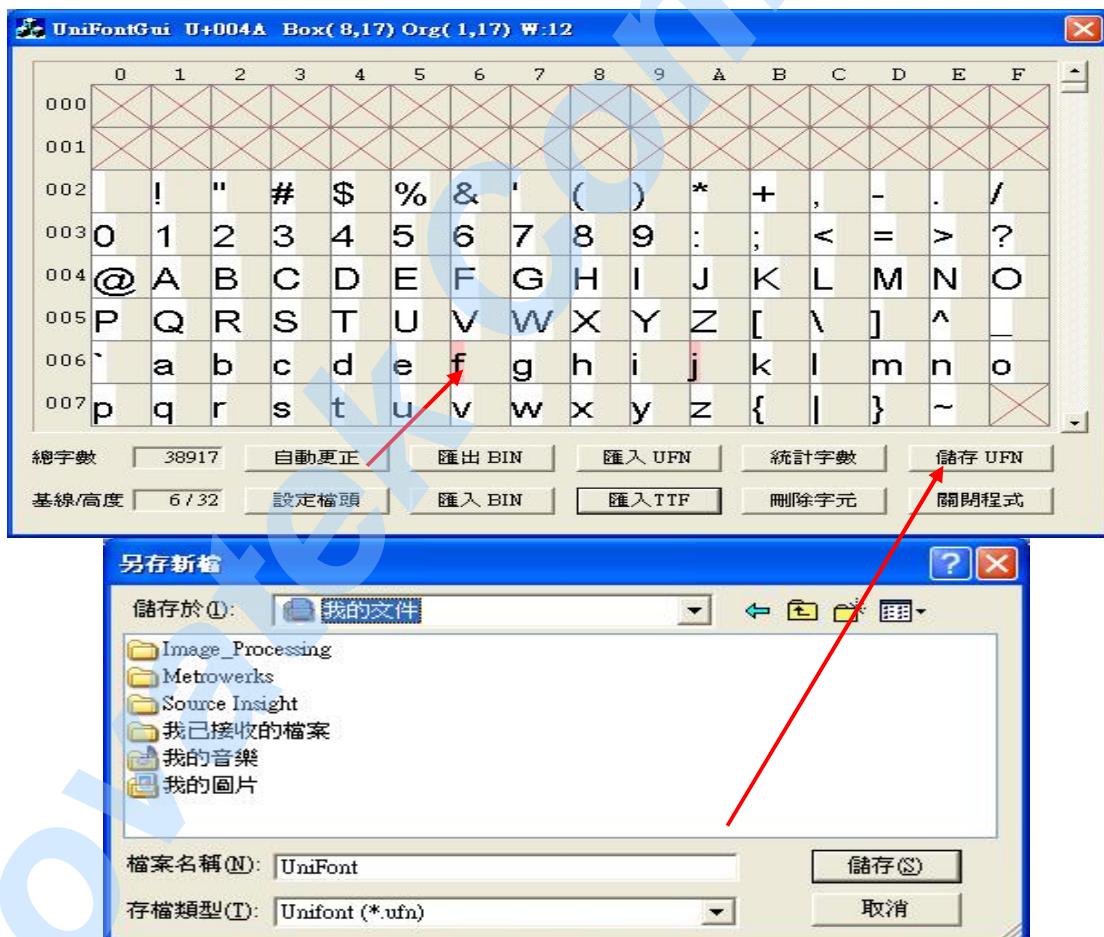


圖 1.1

2. Unicode Format

UFN 的檔案格式包含如下：**<檔頭 Header>**, **<區段表 Range Table>**,**<索引 Index Table>**,**<寬度表 Width Table>**,**<偏移表 Offset Table>**,**<字型 Glyph 0>**, **<字型 Glyph 1>**,.....,**<字型 Glyph N-1>**，使用如下的例子來說明：

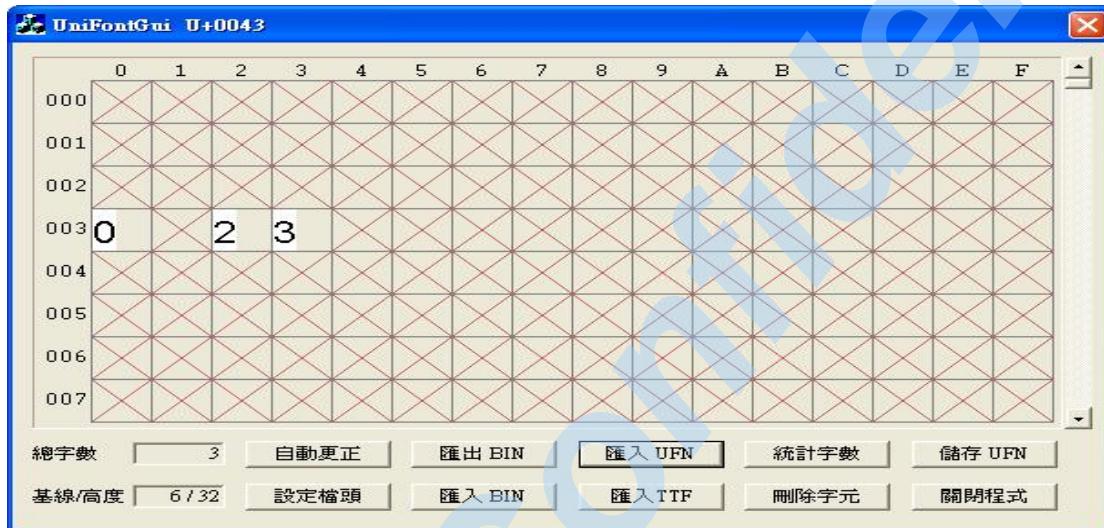


圖 2.1

(a) 檔頭 Header:

```
typedef struct tagUFN_FILE_HEADER {  
    ULONG      cbThis;           /* 檔頭大小= sizeof(UFN_FILE_HEADER) */  
    CHAR       cFaceName[28];     /* 字形名稱 */  
    SHORT      wHeight;          /* 字形高度,最大 32 */  
    SHORT      wDescent;         /* 基準線 pixel, 由最下面往上算, 例如 6 pixel 高 */  
    USHORT     wGlyphsCount;      /* 總共有多少個 Glyph */  
    USHORT     cRanges;          /* 總共有多少個區段 */  
    ULONG      FileHandle;       /* 存放字形的檔案代碼 */  
    ULONG      FontHandle;       /* 存放字形的 FontMap 代碼 */  
    UFN_RANGE* RangeTable;      /* 指向區段表,每個區段以 UFN_RANGE 結構表示 */  
    USHORT*    IndexTable;        /* 指向索引表,對應每個區段的 始 Glyph, USHORT */  
    UCHAR*    WidthTable;        /* 指向寬度表,對應每個 Glyph 的建議寬度,型別 UCHAR */  
    ULONG*    OffsetTable;        /* 指向偏移表,對應每個 Glyph 資料的偏移位址,型別 ULONG */  
    ULONG      GlyphData;         /* 第 0 個 Glyph 資料的啓始位址, 從檔案開頭啓算 */  
} UFN_FILE_HEADER, *PUFN_FILE_HEADER;
```

由圖 2.1 轉出的 Unicode 二進位檔的格式如圖 3 所示，00000000h~00000044h 為檔頭，前 4 個 byte 44 00 00 00 為檔頭大小 cbThis 共 68 bytes (這裡資料排列方式為 **LittleEndian**)，其餘 64 bytes 為上述資料結構的排列。

(b) 區段表 Range Table

區段表的定義為每一個區段第一個 unicode 的起始值，到最後一個 unicode 的結束值，在圖 2.1 中的 unicode “0”的第一個啓始值和最後一個結束值都是 0x0030; 而字型 2 和 3 的啓始值和結束值分別為 0x0032 和 0x0033，因此圖 2.1 有兩個區段表。在圖 2.2 中的 00000044h~00000047h 和 00000048H~0000004Bh 分別為第一個和第二個區段，用陣列結構表式如下<RangeTable[00000030h~00000033h] = 0x30 00 30 00>及<RangeTable [00000030h~00000033h]+4 = 0x32 00 33 00>。

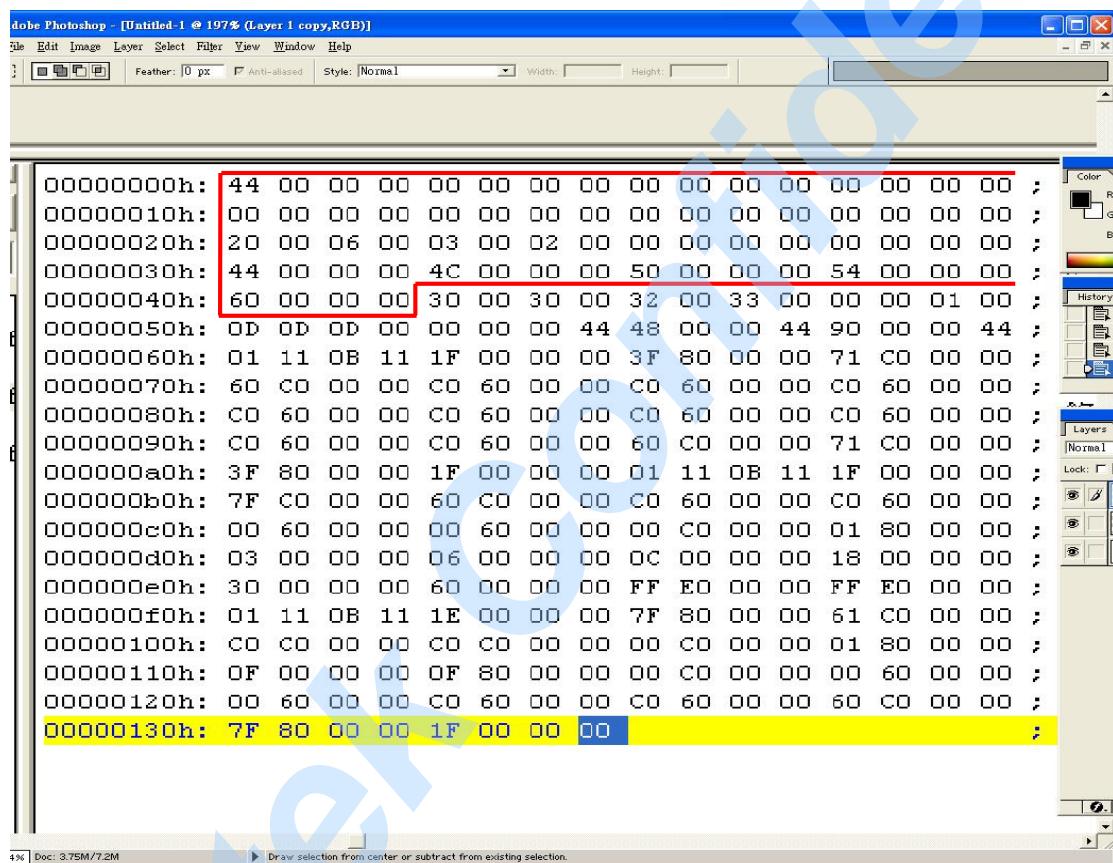


圖 2.2

(c) 索引表 Index Table

索引表所代表的為每一個區段的第一個 Unicode Glyph，有多少個區段表，就有多少個索引表。在圖 2.2 中，0000004Ch~0000004Dh 的 00 00 代表在圖 2.1 中第一個區段的第一個 Glyph ”0”; 0000004Eh~0000004Fh 的 01 00 代表在圖 2.1 中第二個區段的第一個 Glyph ”2”，第二區段的第二個 Glyph，索引表為 02 00;如果有第三個區段，則第三個區段的第一個 Glyph 索引表為 03 00，但對於應用程式轉出來的 Unicode binary file，只會表示每一個區段的第一個 Unicode Glyph。雖然索引表示是用 2 個 byte 來表示，但所佔的空間仍然是要符合 four bytes，這個例子剛好 four byte 00 00 01 00，假設今天索引表有 3 個 00 00

01 00 03 00，雖然用 6 個 byte 來表示，但在 binary file 仍然需要表示為 00 00 01 00 03 00 00
00 共 2 個 four byte。

(d) 寬度表 Width Table

在索引表之後就是寬度表，使用一個 byte 代表每一個 Glyph 的寬度，有多少個 Glyph，就有多少個寬度表，在圖 2.2 中，00000050h~00000052h 的 0D 0D 0D 依序代表在圖 2.1 中的 Glyph。雖然寬度表示是用 1 個 byte 來表示，但所佔的空間仍然是要符合 four byte，這個例子中只有 3 個 Glyph，所以只有 3 個 byte 0D 0D 0D，但在 binary file 仍然需要表示為 0D 0D 0D 00 到一個 four byte。

(e) 偏移表 Offset Table

偏移表用來指向每一個 Glyph 的資料，在記憶體中，偏移表需要靠索引表來指到啓始的位置，然後在由偏移表加上第一個 Glyph 的啓始位址，然後在指向 Glyph 資料的啓始位址。偏移表用 4 個 byte 來表示，其中最高的 byte 代表 Glyph 的資料長度(byte)，在圖 2.2 中，00000040h~00000043h 的 60 00 00 00 代表在圖 2.1 中第一個 Glyph 0 的啓始位址; 00000054~00000057 的 00 00 00 44 代表在圖 2.1 中第一個 Glyph 0 的偏移位址，其中最高 byte 0x44 是這個 Glyph 的資料長度，共有 68 個 byte，所以偏移表加上第一個 Glyph 的啓始位址，就指向 00000060h 的資料起始位址，這個資料起始位址的前 4 個 byte 分別表示這個 Glyph 的 X 座標、Y 座標、包含住 Glyph 的最小寬度、包含住 Glyph 的最大高度(如圖 2.3 所示)，爾後從 00000064h 一直到 000000A7h 共 68 個 byte 才是 Glyph 的資料長度。

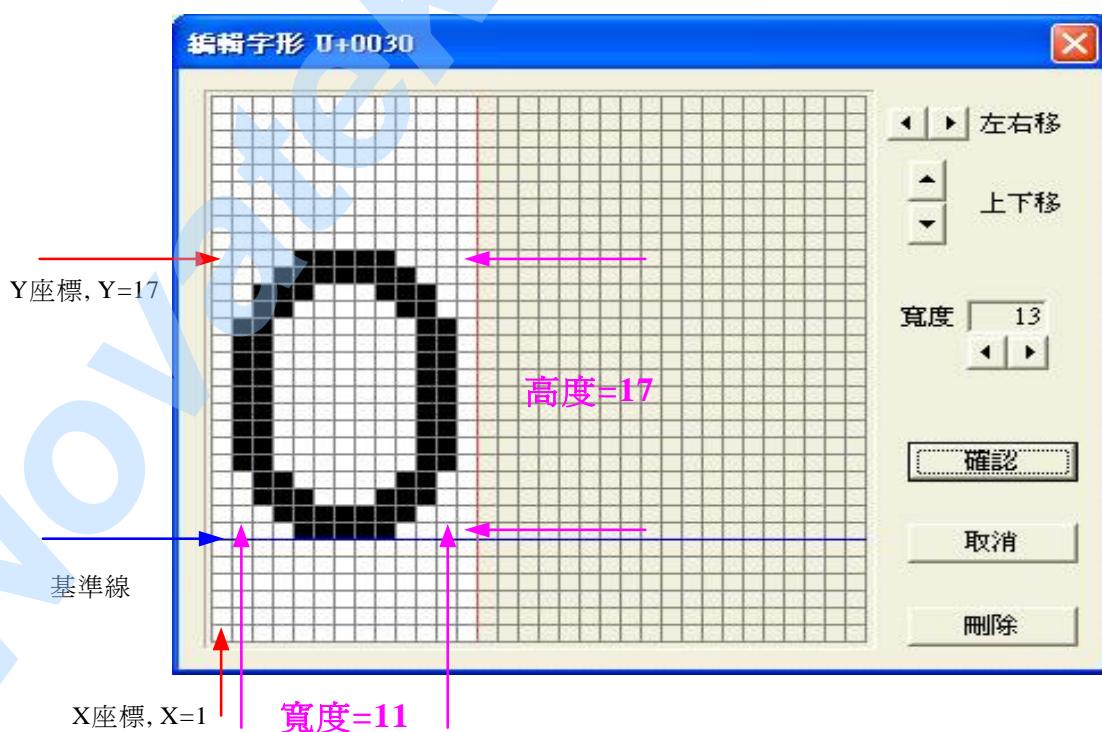


圖 2.3

(f) Glyph Information

每個 Glyph 資料都用一個掃描線來掃描，一個掃描線為 4 個 byte，最多表示 32 個 pixel，每個 byte 中的每位元從左至右對應一個 pixel，掃描的順序由 Y 座標決定，byte 的總數目為高度乘以 4，在圖 2.3 中，Y 的座標為 17，所以從這裡開始掃描，第一次掃描到的資料為 0x1F 00 00 00，對應到圖 2.2 的 00000064h~00000067h 的 1F 00 00 00。每個 Glyph 高度最高為 32，所以最多 32 條掃描線。

3. PStore

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 NAND Flash 上。這個部份可參考到 UserPStore.c 的 UserPS_ReadBinFile 和 UserPS_WriteBinFile function。另外也需要使用一塊 memory pool 來存放整個 Unicode 字庫，相關部分可參考到 SysCfg.c 和 SysCfgPrj.h (或 SysCfg.h)，目前 Unicode 字庫約 2.5MB，這是經過修改過後的，PC 端要轉成 font bitmap 仍然要用原始的 3.3MB。

4. Display

使用圖 1.1 的應用程式所轉出來的每一個 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 2.3 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 2.3 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。

相關的部份可參考到\project_name\Project\xxxDemo\SrcCode\UIDisplay\ 的 ShowStringFuncs.c ; \project_name\Project\xxxDemo\SrcCode\UIDisplay\IconsDB\ 的 OSDFont.c 、 OSDFont.h 、 OSDUnicode.c 、 OSDUnicode.h 、 OSDUnicodeApi.c 、 OSDUnicodeApi.h 、 OSDUnicodeDirectImp.c 、 OSDUnicodeDirectImp.h ; LIB\LIB_src\SubSystem\DrawLib\ 的 draw_lib.c 。

5. String Table Transform

(1) 將 string table 轉成 Unicode 格式的 StringTable.txt 文字檔

將 Excel 格式的 string table 轉成 Unicode 的 txt 文字檔，如圖 5.1 的粉紅色箭頭所示，接著要關掉圖 5.1 的檔案，否則當使用 UI designer studio 時，也無法轉出 source code 。

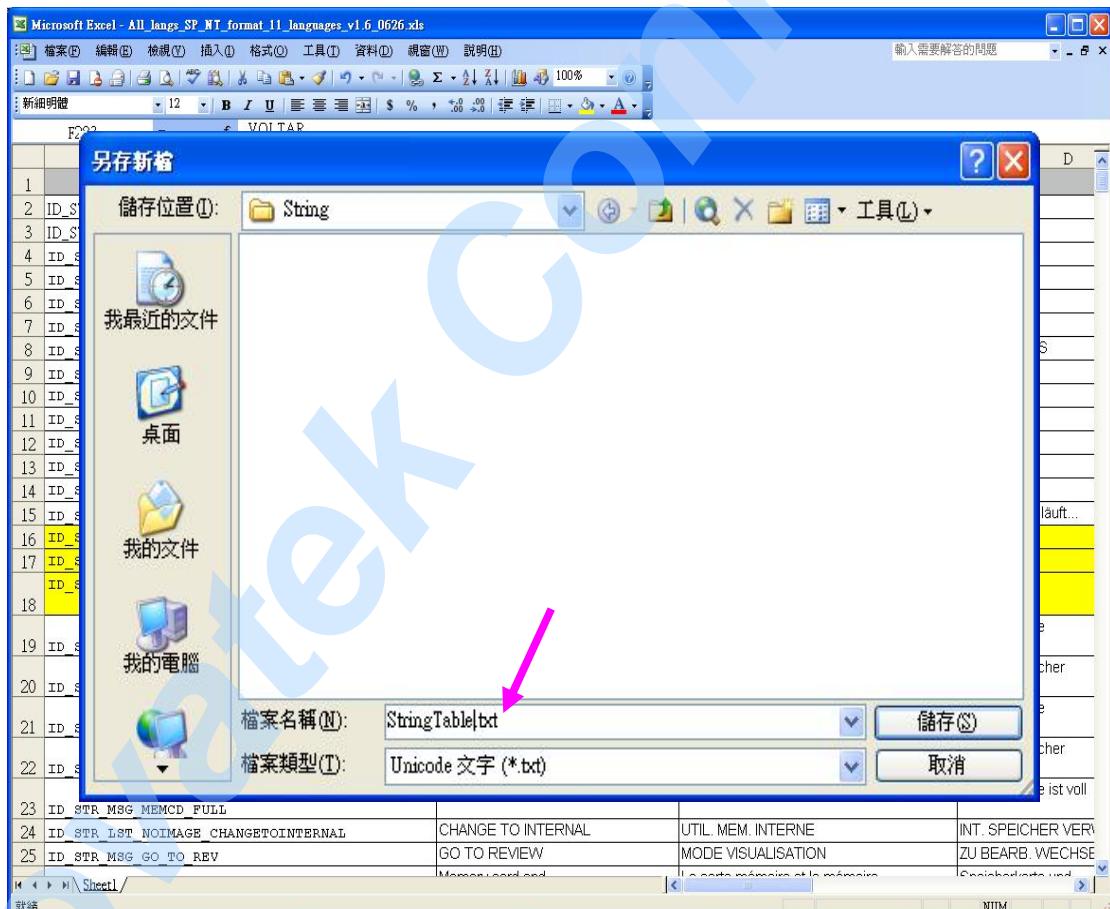


圖 5.1

	A	B	C	D
579	ID_STR_MSG_INK_LOW_ERROR	Ink low	Encre insuffisante	Tinte ist fast verbraucht
580	ID_STR_MSG_CHECK_PRINTER	Check printer	Vérifier l'imprimante	Drucker überprüfen
581	ID_STR_MSG_PRINTING	Printing...	Impression en cours...	Drucken läuft...
582	ID_STR_MSG_PRINT_COMPLETE	Printing complete	Impression terminée	Druckvorgang beendet
583	ID_STR_MSG_PRINTING_CANCELLED	Printing cancelled	Impression annulée	Druck abgebrochen
584	ID_STR_MSG_PRINTER_ERROR	Printer error	Erreur d'imprimante	Druckerfehler
585	ID_STR_DUMMY	all%_HS&@<=> ^ ~	¤=¥\$©«®	¤=¥\$©«®
586				
587				
588				
589				
590				

圖 5.2

圖 5.2 粉紅色箭頭所指的部份是 dummy string，主要可用來增加如綠色箭頭所指的特殊字元，某些應用可能需要這些特殊的字元來顯示字串。

(2) 使用舊的 UI Design Studio 將 StringTable.txt 轉換成 C source code

開啟 UI designer studio 的執行檔，畫面如圖 5.3 所示，詳細的使用可參考 Novatek NDK AN Tools - UI Designer (v2.0).doc 這份文件除了轉換 string 外，還包括 Icon 與 Font Icon 轉換，這裡僅擷取字串和 Font 部分出來說明。



圖 5.3

a) Create String Table

Start to run UIDS, and select “File/New...”

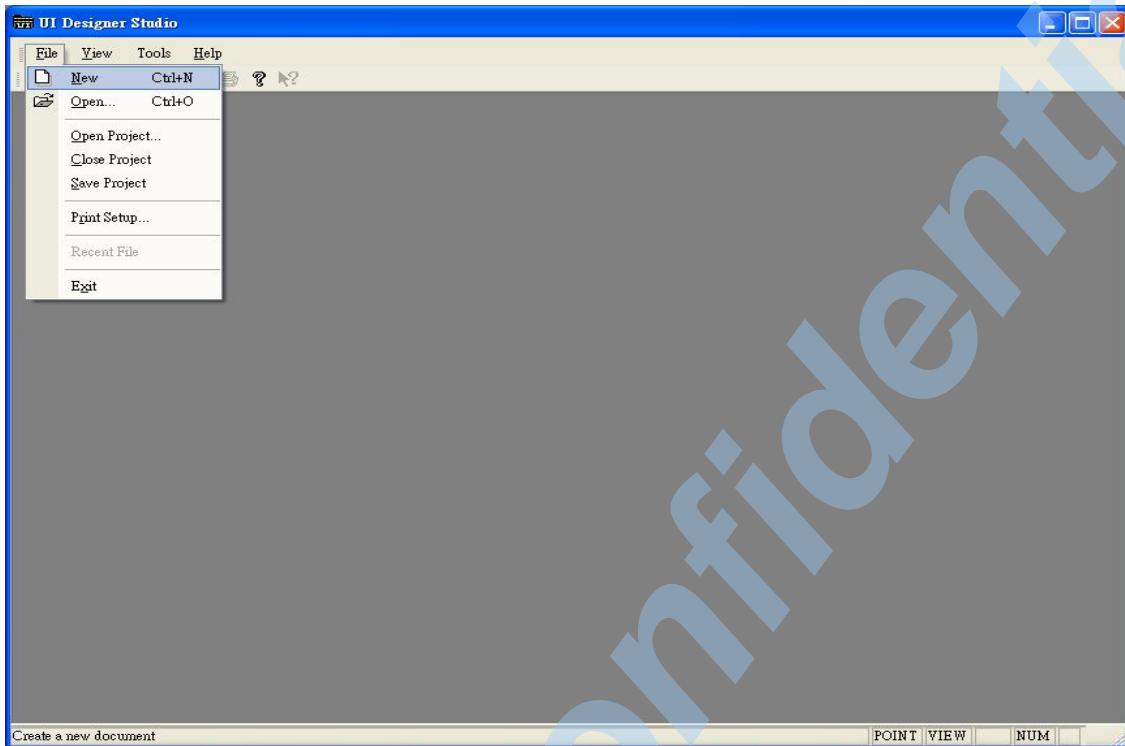


圖 5.4

Choose “String Table” for a new document, and specify filename and file path for this new document. **The “File” must be named “StringTable”.**

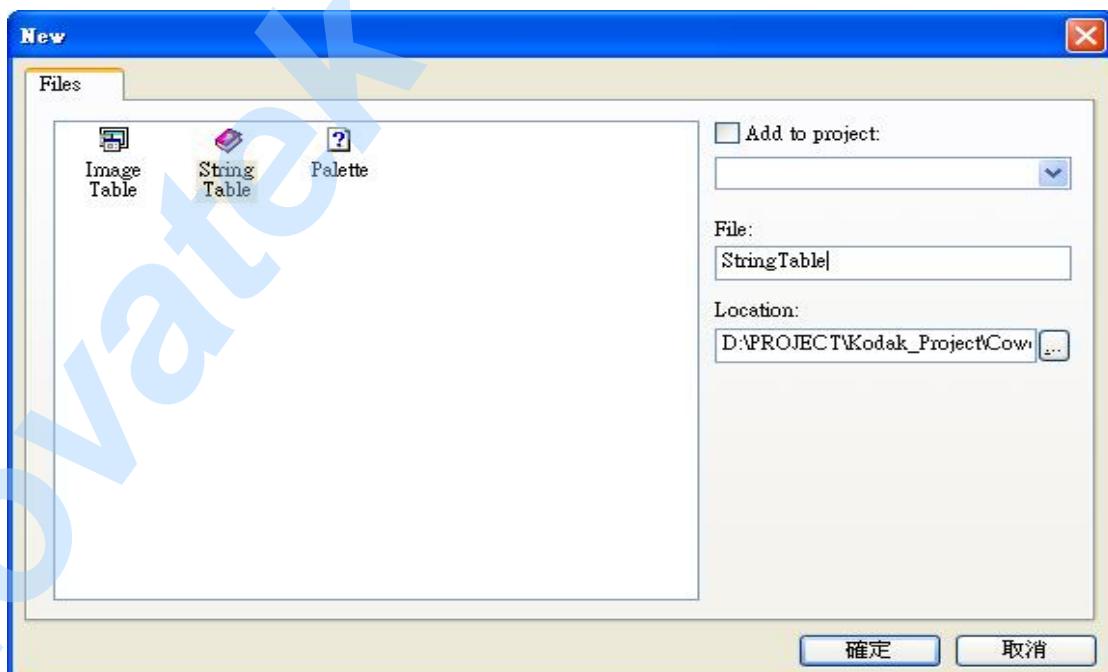


圖 5.5

b) String Table Tool UI

The screenshot shows a window titled "UI Designer Studio - [MyStringTable.xms]". The menu bar includes File, Edit, View, Item, Tools, Window, and Help. Below the menu is a toolbar with icons for New, Open, Save, Print, Find, Replace, and Help. The main area is a table titled "MyStringTable.xms" with columns: ID, Name, Info, and LANG1. A single row is present with ID "0000" and Name "LANG1". The status bar at the bottom indicates "Ready".

ID	Name	Info	LANG1
0000			

圖 5.6

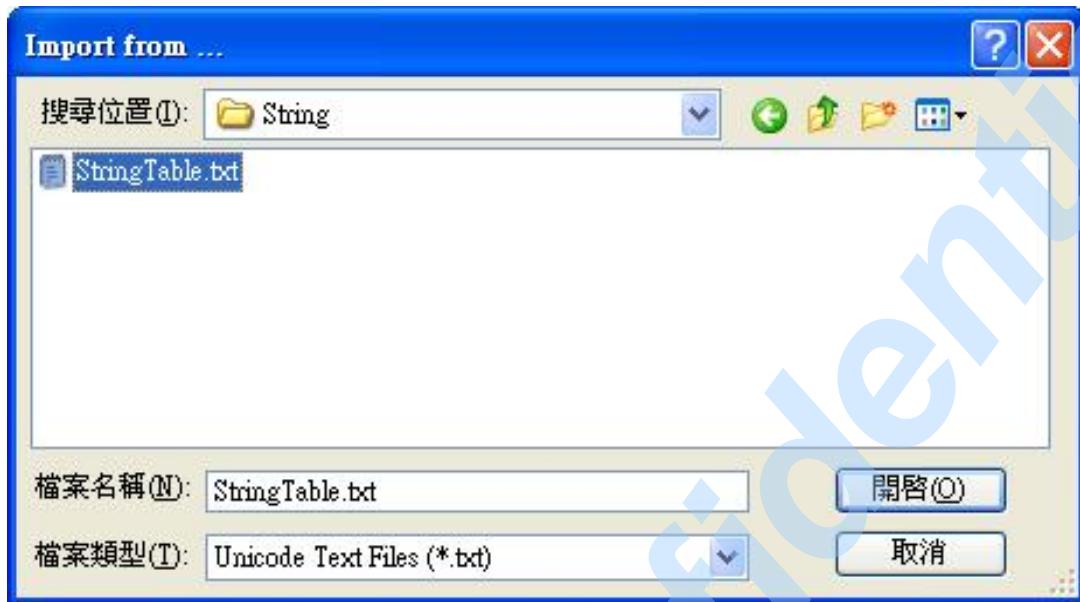
c) Right-click and select “Load String...”

The screenshot shows the same window as Figure 5.6. A right-click context menu is open over the "0000" entry in the table. The menu items include: Load String ..., Rename, Rename by first String, Rename Language Title, Move ..., Sort by Name, Compact, Clear, Clear All, and Properties... . The "Load String ..." option is highlighted. The status bar at the bottom indicates "Ready".

ID	Name	Info	LANG1
0000			

圖 5.7

d) Select your input string and press OK



**NOTE: Input TXT file is UNICODE encoded and all strings in this file
are tab character separated.**

圖 5.8

e) Setup loading options

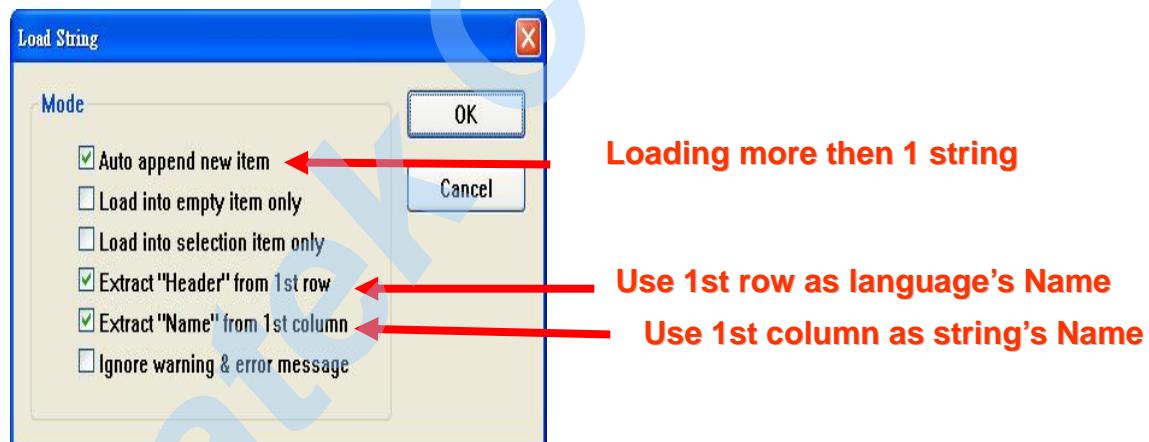


圖 5.9

f) String table is created now

UI Designer Studio - [StringTable.xms]

Language's Name

ID	Name	Info	ENGLISH	FRENCH	GERMAN	SPANISH	PORT_BRAZILIAN	CHINESE_Simplified
0000	ID_STR_NULL							
0001	ID_STR_TTL_LANGUAGE_HLP	LANGUAGE	LANGUE	SPRACHE	IDIOMA	IDIOMA	语言	
0002	ID_STR_SUM_ENGLISH	ENGLISH	ENGLISH	ENGLISH	ENGLISH	ENGLISH	ENGLISH	
0003	ID_STR_SUM_FRENCH	Français	FRANÇAIS	FRANÇAIS	FRANÇAIS	FRANÇAIS	FRANÇAIS	
0004	ID_STR_SUM_GERMAN	DEUTSCH	DEUTSCH	DEUTSCH	DEUTSCH	DEUTSCH	DEUTSCH	
0005	ID_STR_SUM_SPANISH	ESPAÑOL	ESPAÑOL	ESPAÑOL	ESPAÑOL	ESPAÑOL	ESPAÑOL	
0006	ID_STR_SUM_PORTUGUESE	PORTUGUÊS	PORTUGUÊS	PORTUGUÊS	PORTUGUÊS	PORTUGUÊS	PORTUGUÊS	
0007	ID_STR_SUM_CHINESE_SIMPLIFIED	简体中文	简体中文	简体中文	简体中文	简体中文	简体中文	
0008	ID_STR_SUM_POLISH	POLSKI	POLSKI	POLSKI	POLSKI	POLSKI	POLSKI	
0009	ID_STR_SUM_TURKISH	TÜRKÇE	TÜRKÇE	TÜRKÇE	TÜRKÇE	TÜRKÇE	TÜRKÇE	
000A	ID_STR_SUM_RUSSIAN	РУССКИЙ	РУССКИЙ	РУССКИЙ	РУССКИЙ	РУССКИЙ	РУССКИЙ	
000B	ID_STR_SUM_CZECH	ČEŠTINA	ČEŠTINA	ČEŠTINA	ČEŠTINA	ČEŠTINA	ČEŠTINA	
000C	ID_STR_SUM_GREEK	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	
000D	ID_STR_MSG_PROCESSING	Processing...	En cours...	Verarbeitung...	Procesa...	Processando...	正在处理...	
000E	ID_STR_MSG_MODE_EXPLANATION	Scenes	Motives	Escenas	Cenas	场景		
000F	ID_STR_MSG_MODE_EXPLANATION	Video	Vidéo	Vídeo	Vídeo	录像		
0010	ID_STR_LST_WHITE_BALANCE	Auto	Automatique	Automatisch	Automático	Automático	自动	
0011	ID_STR_MSG_READING_MEMORY	Readingmemory	Lecturecarte...	Speicherkart...	Leyendo...	Lendo o cart...	正在读取存...	
0012	ID_STR_MSG_READING_INTERIOR	Readinginterior	Lecturemém...	Interner Spei...	Leyendo...	Lendo a me...	正在读取内...	
0013	ID_STR_MSG_CFCARD_FULL	Memory card...	La carte mém...	Speicherkart...	Tarjeta d...	O cartão de ...	存储卡已满	
0014	ID_STR_MSG_INTERNAL_FULL	Internal mem...	La mémoire i...	Interner Spei...	Memoria...	A memória int...	内存已满	
0015	ID_STR_MSG_MEMCD_FULL	Memory card...	Carte mémoir...	Speicherkart...	Tarjeta d...	Cartão de me...	存储卡已满	
0016	ID_STR_LST_NOIMAGE_CHANGE	CHANGE TO...	UTIL. MEM. I...	INT. SPEIC...	CAMBIA...	MUDAR PA...	更改为内存	
0017	ID_STR_MSG_GO_TO_REV	GO TO REV...	MODE VISU...	ZU BEARB...	IR A RE...	IR PARA RE...	进入查看	
0018	ID_STR_MSG_REPLACE_CARD	Memory card...	La carte mém...	Speicherkart...	Tarjeta d...	Cartão de me...	存储卡和内存	
0019	ID_STR_MSG_CARD_HW_PROTOCOL	Memory card...	La carte mém...	Speicherkart...	Tarj. de ...	Cartão de me...	存储卡被写保	

圖 5.10

g) Edit String Table

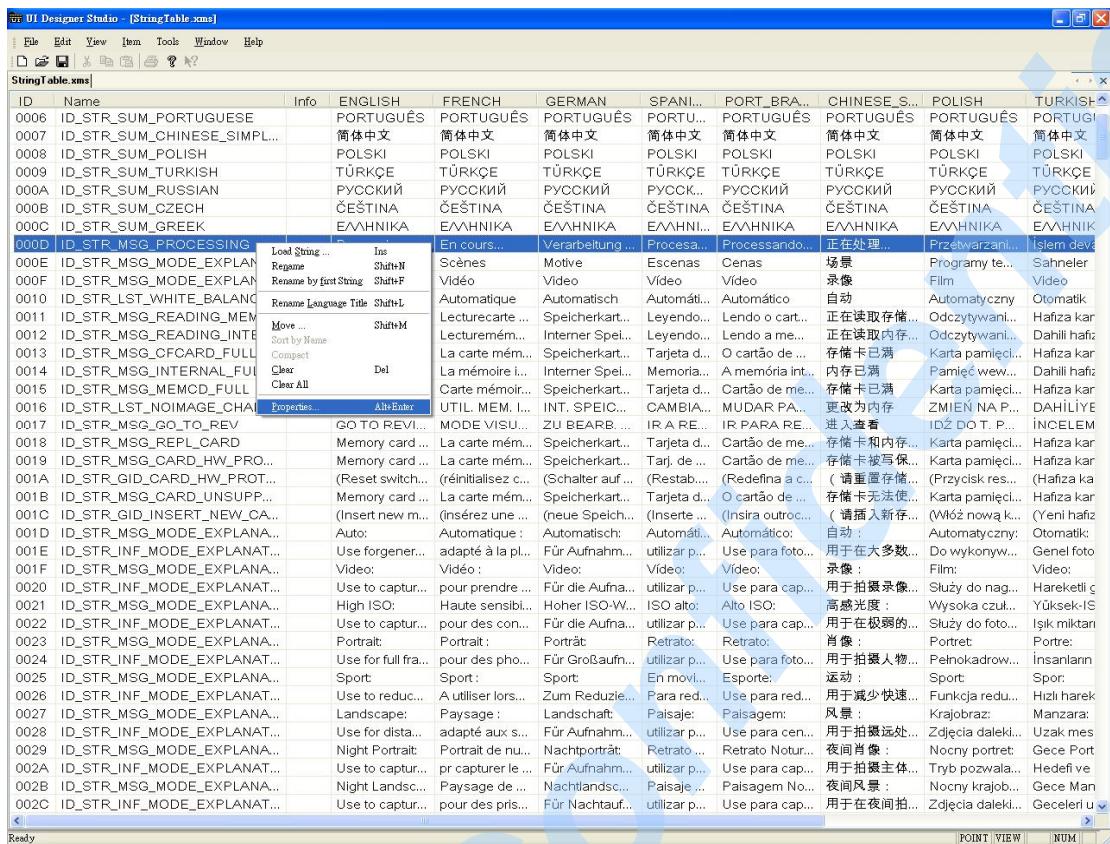
Left-click on “Name” column and edit its name

UI Designer Studio - [StringTable.xms]

ID	Name	Info	ENGLISH	FRENCH	GERMAN	SPANISH	PORT_BRAZILIAN	CHINESE_Simplified
000C	ID_STR_SUM_GREEK	EΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙ...	ΕΛΛΗΝΙΚΑ	ΕΛΛΗΝΙΚΑ	
000D	<u>ID_STR_MSG_PROCESSING</u>	Processing...	En cours...	Verarbeitung...	Procesa...	Processando...	正在处理...	
000E	ID_STR_MSG_MODE_EXPLANATION	Scenes	Motive	Escenas	Cenas	场景		
000F	ID_STR_MSG_MODE_EXPLANATION	Video	Vidéo	Vídeo	Vídeo	录像		
0010	ID_STR_LST_WHITE_BALANCE	Auto	Automatique	Automatisch	Automático	Automático	自动	
0011	ID_STR_MSG_READING_MEMORY	Readingmemory	Lecturecarte...	Speicherkart...	Leyendo...	Lendo o cart...	正在读取存...	
0012	ID_STR_MSG_READING_INTERIOR	Readinginterior	Lecturemém...	Interner Spei...	Leyendo...	Lendo a me...	正在读取内...	
0013	ID_STR_MSG_CFCARD_FULL	Memory card...	La carte mém...	Speicherkart...	Tarjeta d...	O cartão de ...	存储卡已满	
0014	ID_STR_MSG_INTERNAL_FULL	Internal mem...	La mémoire i...	Interner Spei...	Memoria...	A memória int...	内存已满	
0015	ID_STR_MSG_MEMCD_FULL	Memory card...	Carte mémoir...	Speicherkart...	Tarjeta d...	Cartão de me...	存储卡已满	
0016	ID_STR_LST_NOIMAGE_CHANGE	CHANGE TO...	UTIL. MEM. I...	INT. SPEIC...	CAMBIA...	MUDAR PA...	更改为内存	
0017	ID_STR_MSG_GO_TO_REV	GO TO REV...	MODE VISU...	ZU BEARB...	IR A RE...	IR PARA RE...	进入查看	
0018	ID_STR_MSG_REPLACE_CARD	Memory card...	La carte mém...	Speicherkart...	Tarjeta d...	Cartão de me...	存储卡和内存	
0019	ID_STR_MSG_CARD_HW_PROTOCOL	Memory card...	La carte mém...	Speicherkart...	Tarj. de ...	Cartão de me...	存储卡被写保	
001A	ID_STR_GID_CARD_HW_PROTOCOL	(Reset switch...)	(réinitialisez c...)	(Schalter auf ...)	(Restab...	(Redefina a c...	(请重置存储	
001B	ID_STR_MSG_CARD_UNSUPP...	Memory card...	La carte mém...	Speicherkart...	Tarjeta d...	O cartão de ...	存储卡无法使	
001C	ID_STR_GID_INSERT_NEW_CARD	(Insert new m...	(Insérez une n...	(neue Speich...	(Insere ...)	(Insira outr...	(请插入新存	
001D	ID_STR_MSG_MODE_EXPLANATION	Auto:	Automatique :	Automatisch:	Automát...	Automático:	自动 :	
001E	ID_STR_INF_MODE_EXPLANATION	Use for gener...	adapté à la pl...	Für Aufnahm...	utilizar p...	Use para foto...	用于在大多数	
001F	ID_STR_INF_MODE_EXPLANATION	Video:	Vidéo :	Video:	Video:	Vídeo:	录像 :	
0020	ID_STR_INF_MODE_EXPLANATION	Use to captur...	pour prendre ...	Für die Aufna...	utilizar p...	Use para cap...	用于拍摄录像	
0021	ID_STR_INF_MODE_EXPLANATION	High ISO:	Hauta sensibi...	Hoher ISO-W...	ISO alto:	Alto ISO:	高感光度 :	
0022	ID_STR_INF_MODE_EXPLANATION	Use to captur...	pour des con...	Für die Aufna...	utilizar p...	Use para cap...	用于在极弱的	
0023	ID_STR_INF_MODE_EXPLANATION	Portrait:	Portrait:	Portrait:	Retrato:	Retrato:	肖像 :	
0024	ID_STR_INF_MODE_EXPLANATION	Use for full f...	pour des pho...	Für Großaufn...	utilizar p...	Use para foto...	用于拍摄人物	
0025	ID_STR_MSG_MODE_EXPLANATION	Sport:	Sport:	Sport:	En movi...	Esporte:	运动 :	

圖 5.11

h) Right-click and select “Properties...”



The screenshot shows a UI Designer Studio window titled "UI Designer Studio - [StringTable.xml]". The main area displays a grid of strings from a "StringTable.xml" file. The columns represent various languages: ENGLISH, FRENCH, GERMAN, SPANISH, PORTUGUESE, CHINESE_Simplified, POLISH, and TURKISH. The rows contain entries such as "ID_STR_SUM_PORTUGUESE" and "ID_STR_MSG_PROCESSING". A context menu is open over the first row, with the option "Properties..." highlighted.

圖 5.12

i) Setup properties of String Table

Don't change this setting.



圖 5.13

j) Select “File\Save” to save current string table

Save String Table to a *.xms file

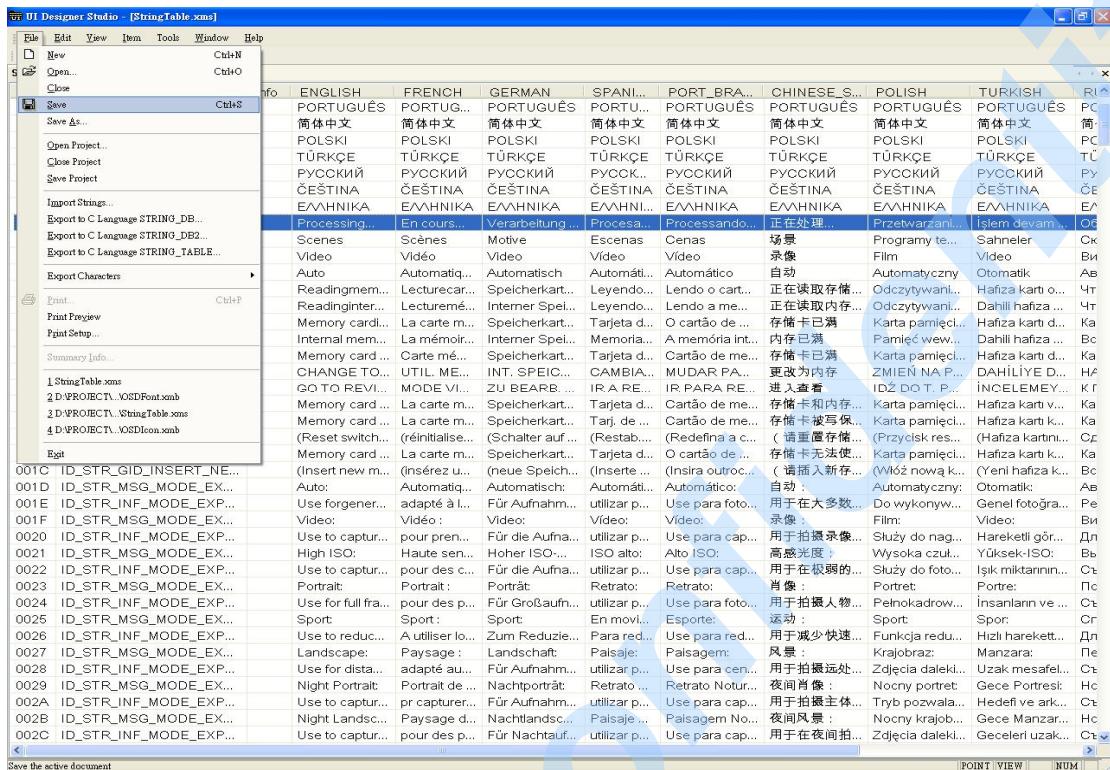


圖 5.14

k) Export String Table to source code

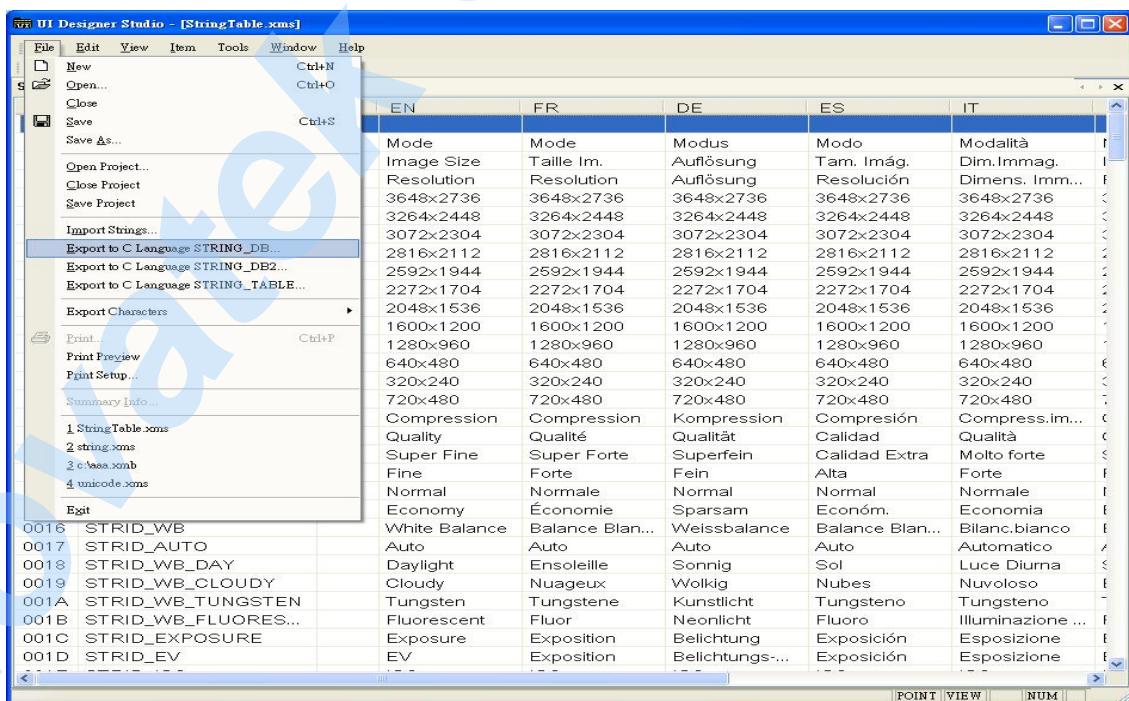


圖 5.15

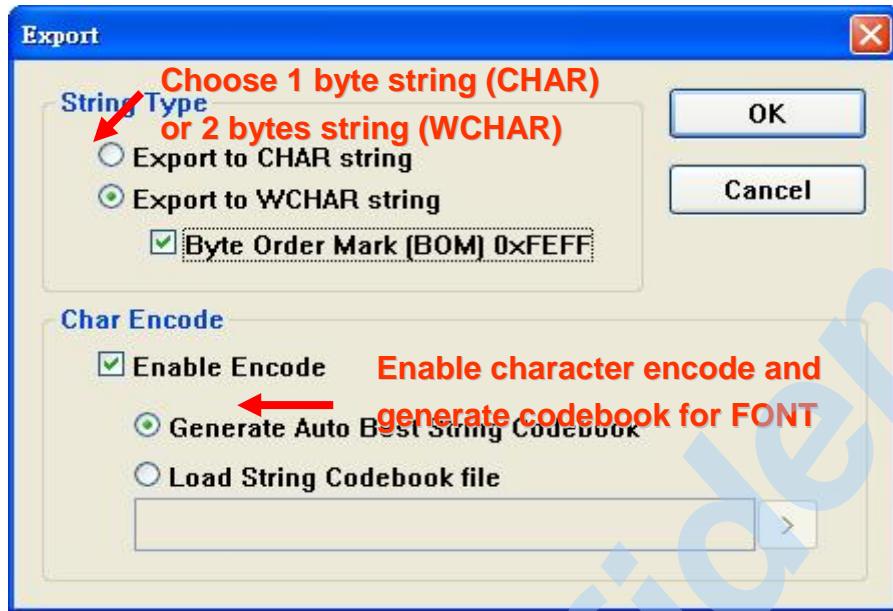


圖 5.16

在圖 17 中的 Enable Encode 對話盒，如果是要使用 Unicode 字庫的話，務必要取消掉，否則轉出的 string table 將會是只有流水號字型的字串，而不是實際的 Unicode；反之，如果不使用 Unicode 字庫則要勾選。因早期的 project 使用流水號來搜尋字串中的字型，這種方式是將會用到的字型預先定義好，可以節省記憶空間，然而當字庫需要增加時或是要顯示多媒體的字型時(例如 MP3 任意取的檔名)，在預先定義好的字型中，可能會搜尋不到，需要在另外製作。

I) Export *.c and *.h file

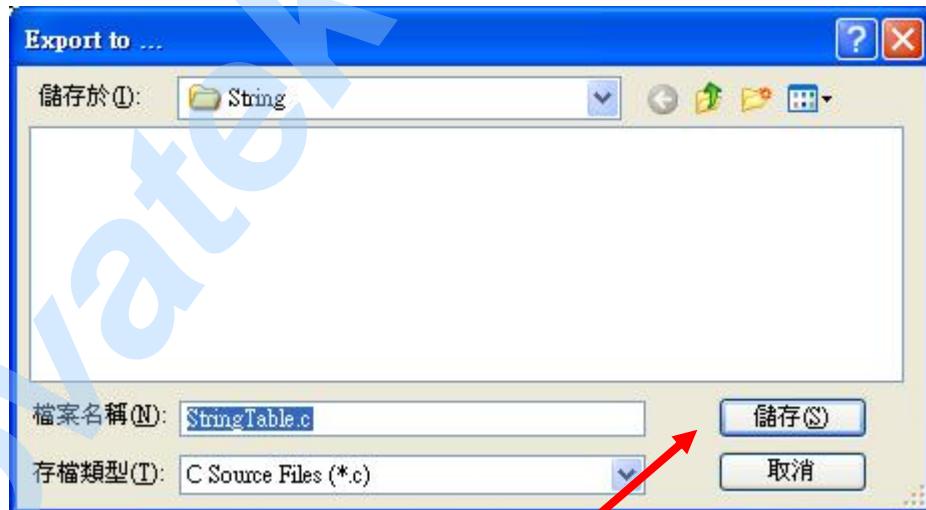


圖 5.17

Output Files:

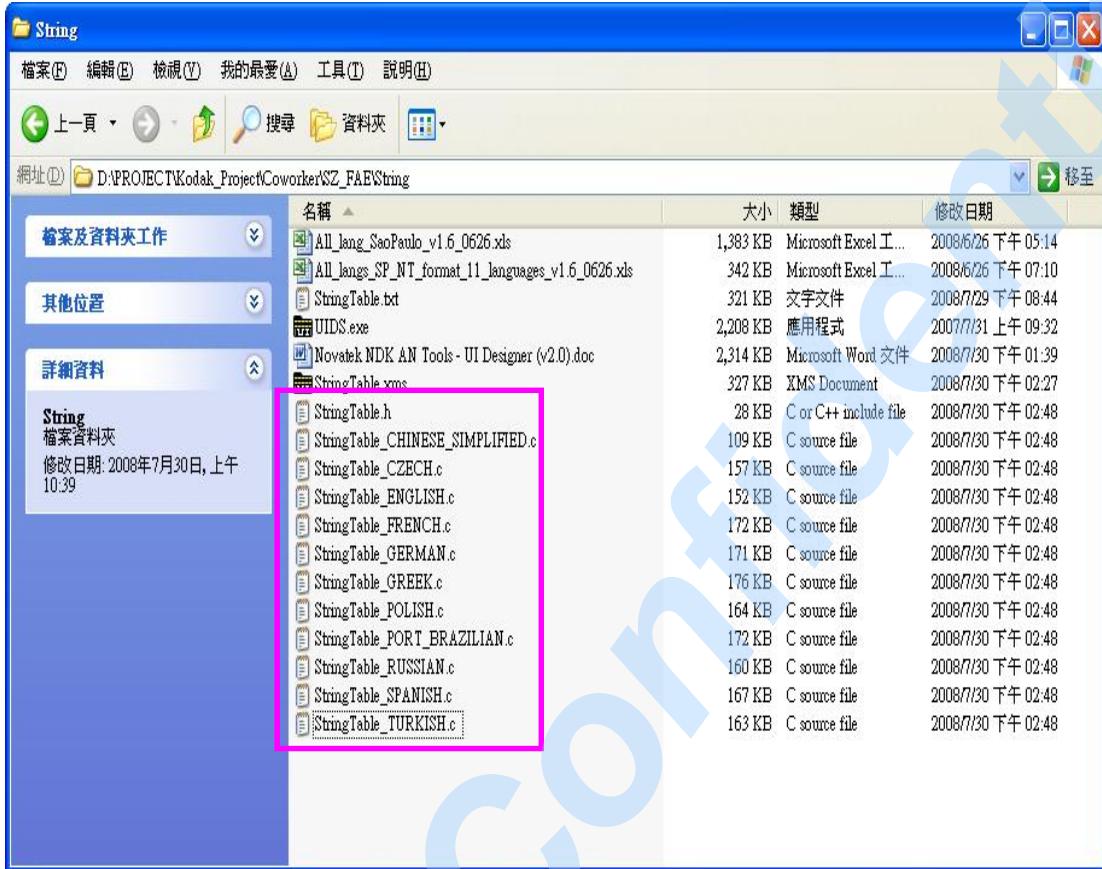


圖 5.18

m) Save codebook file



Press OK to generate codebook for FONT

Output Files: StringTable_Codebook.TXT

圖 5.19

(3) 使用舊的 UI Design Studio 將 font bitmap 轉換成 C source code

如果要使用 Unicode 字庫則不需要做此程序，因為 font 的資料組成已經在產生的 Unicode binary file 裡面，可以替代掉事先做好的 font bitmap。

a) Create Image Table as a Font

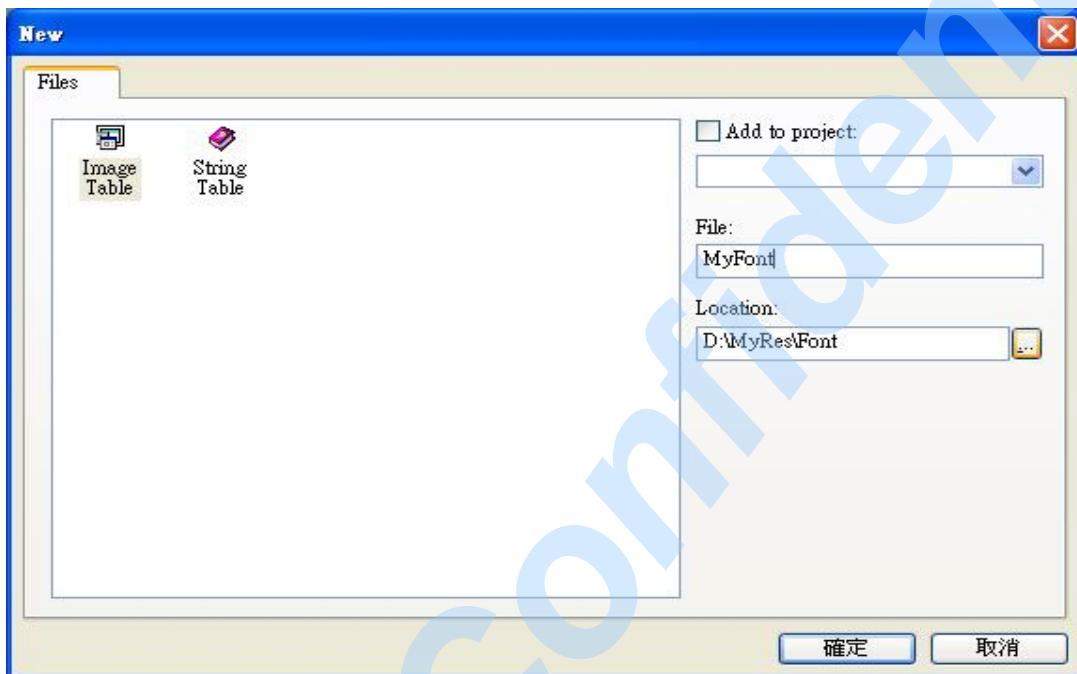


圖 5.20

ID	Name	Info	Image Source
0000			

圖 5.21

Select “File\Import Name\Import from String Codebook”

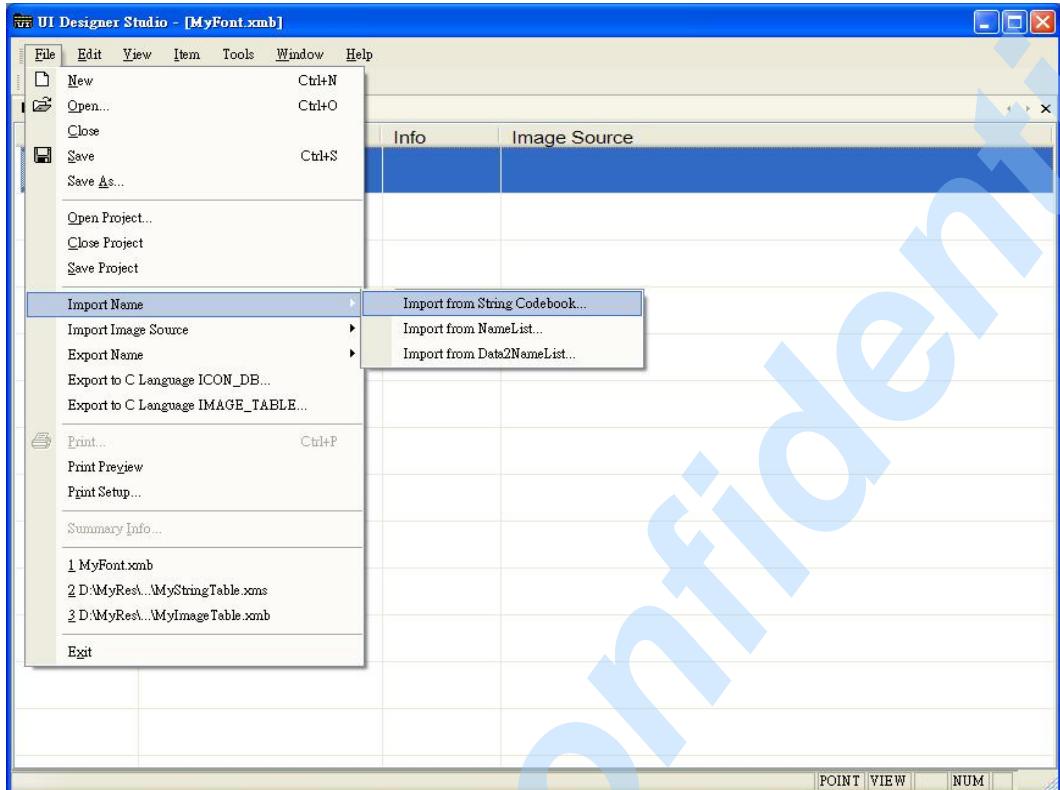


圖 5.22

Select the Codebook Generated by STRING_TABLE



NOTE: For string drawing correctly, FONT and STRING_TABLE must share the same codebook

圖 5.23

After loading codebook

UI Designer Studio - [MyFont.xmb]

File Edit View Item Tools Window Help

MyFont.xmb |

ID	Name	Info	Image Source
0030	030		
0031	031		
0032	032		
0033	033		
0034	034		
0035	035		
0036	036		
0037	037		
0038	038		
0039	039		
003A	03A		
003B			
003C			

Ready POINT VIEW NUM

This character is NOT used in String Table
This character is NOT used in String Table

Encode ID Character's original UNICODE

圖 5.24

Right-click and select “Load Image Source...”

UI Designer Studio - [MyFont.xmb]

File Edit View Item Tools Window Help

MyFont.xmb |

ID	Name	Info	Image Source
0048	0048		
0049	0049		
004A	004A		
004B	004B		
004C	004C		
004D	004D		
004E	004E		
004F	004F		
0050	0050		
0051	0051		
0052	0052		
0053	0053		
0054	0054		

Ready POINT VIEW NUM

Load Image Source ... Ins
Load Name... Shift+Ins
Rename Shift+N
Rename by ID Shift+I
Rename by Data Shift+D
Move ... Shift+M
Sort by Name
Sort by Image Source
Compact
Clear Del
Clear All
Properties... Alt+Enter

圖 5.25

Choose all source images for all characters

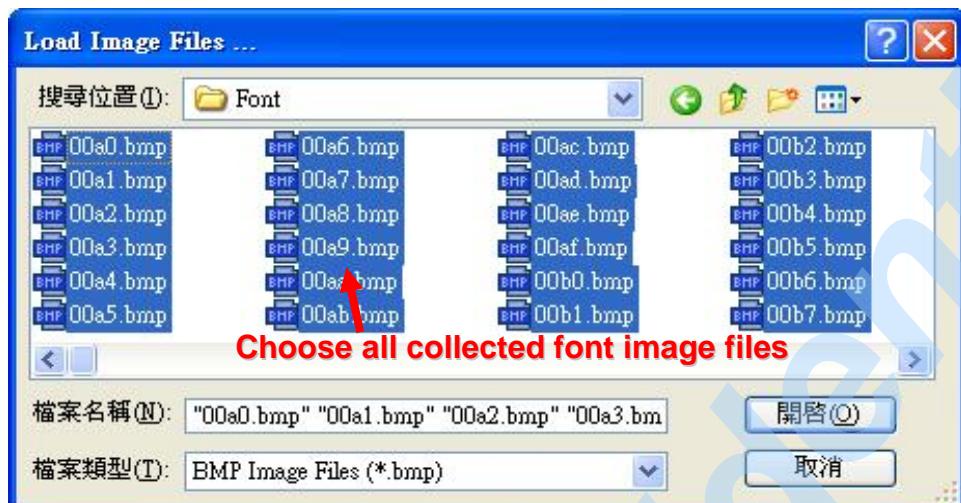


圖 5.26

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.

Setup loading options

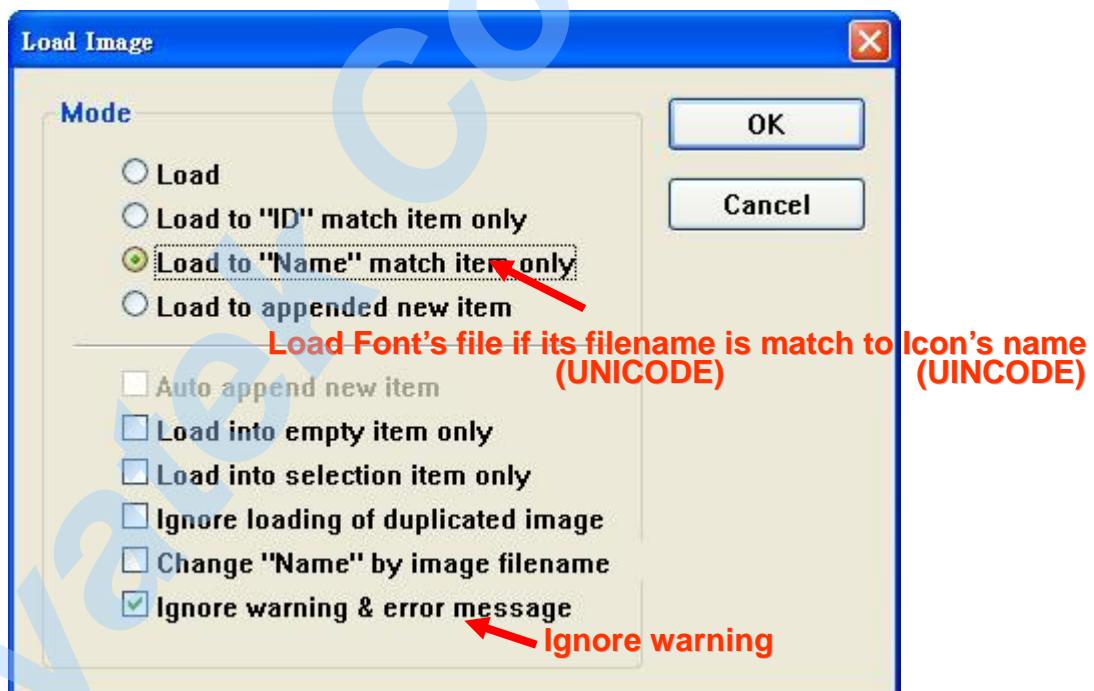


圖 5.27

After loading Font images

The screenshot shows the UI Designer Studio interface with the title bar "UI Designer Studio - [MyFont.xmb]". The menu bar includes File, Edit, View, Item, Tools, Window, and Help. Below the menu is a toolbar with icons for New, Open, Save, Print, Find, and Help. The main window is titled "MyFont.xmb" and contains a table with the following data:

ID	Name	Info	Image Source
0	0030	0030	12x24x4b... .\0030.bmp
1	0031	0031	12x24x4b... .\0031.bmp
2	0032	0032	12x24x4b... .\0032.bmp
3	0033	0033	12x24x4b... .\0033.bmp
4	0034	0034	12x24x4b... .\0034.bmp
5	0035	0035	12x24x4b... .\0035.bmp
6	0036	0036	12x24x4b... .\0036.bmp
7	0037	0037	12x24x4b... .\0037.bmp
8	0038	0038	12x24x4b... .\0038.bmp
9	0039	0039	12x24x4b... .\0039.bmp
:	003A	003A	6x24x4bpp .\003a.bmp
	003B		
	003C		

圖 5.28

Right-click and select “Properties...”

The screenshot shows the same UI Designer Studio interface as Figure 5.28. A context menu is open over the row for font image ID 0030. The menu options include:

- Load Image Source ...
- Load Name...
- Rename
- Rename by ID
- Rename by Data
- Move ...
- Sort by Name
- Sort by Image Source
- Compact
- Clear
- Clear All
- Properties...

The "Properties..." option is highlighted with a blue selection bar. The rest of the table and interface are identical to Figure 5.28.

圖 5.29

Setup properties of Image Table

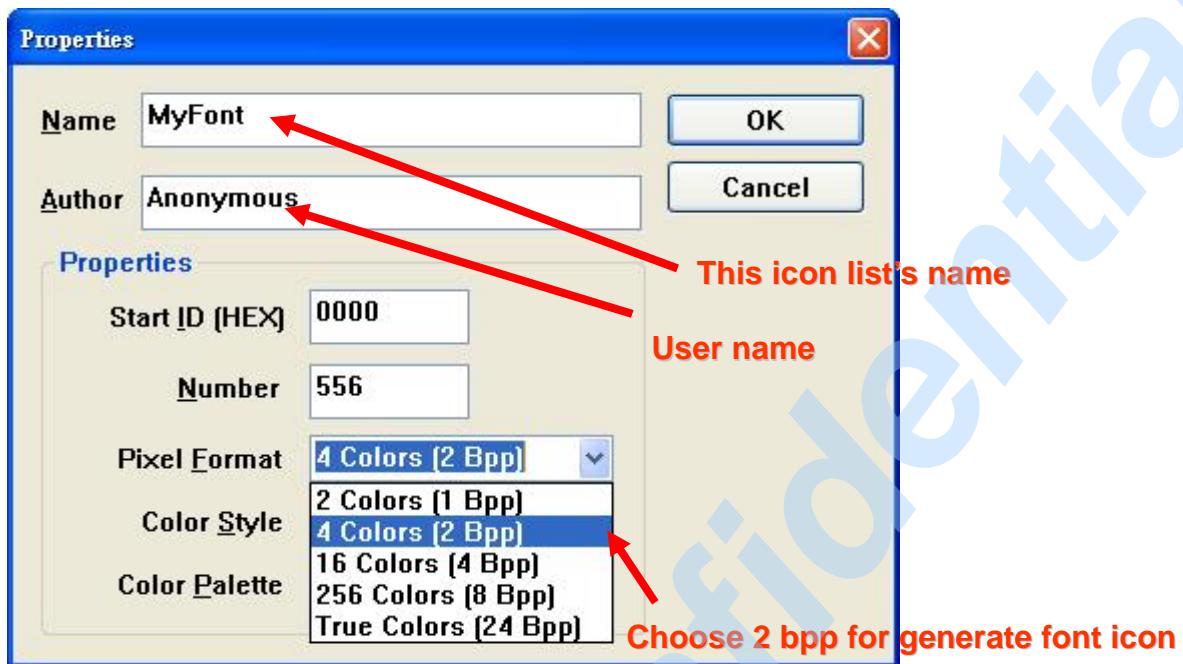


圖 5.30

在圖 5.30 中，如果使用 Unicode 轉出來的 font bitmap 只有 2 個 bit，所以必須要選 2 Colors 的格式。

b) Export font IMAGE_TABLE to source code and binary file

Select “File\Export to C Language IMAGE_TABLE...”

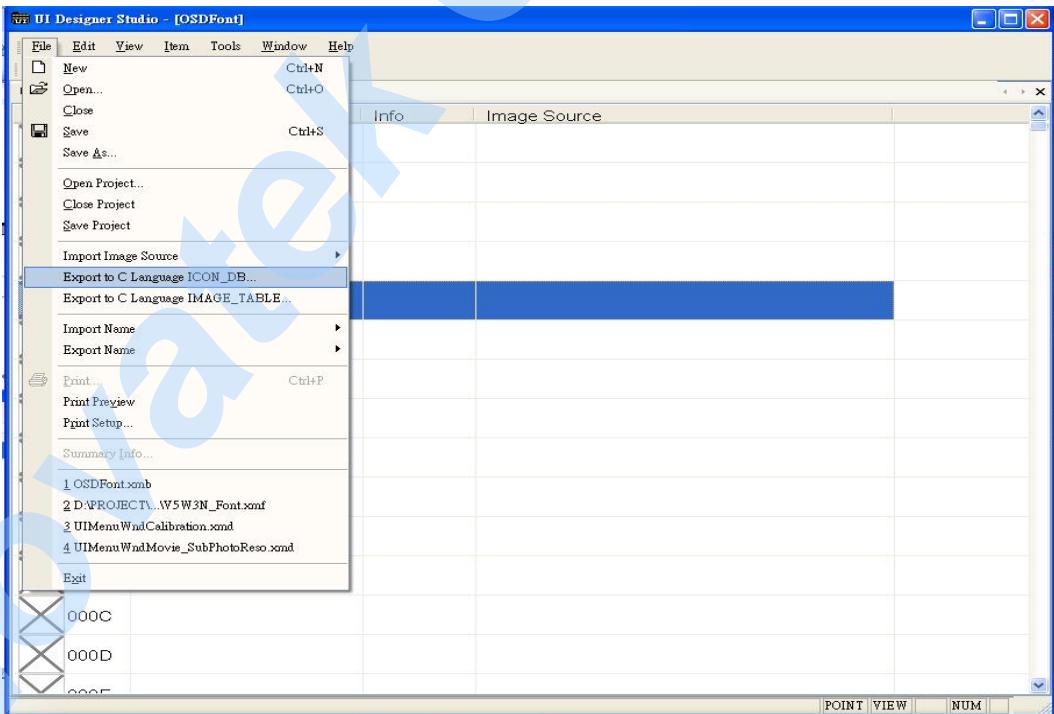


圖 5.31

Export Dialog for control Font's color

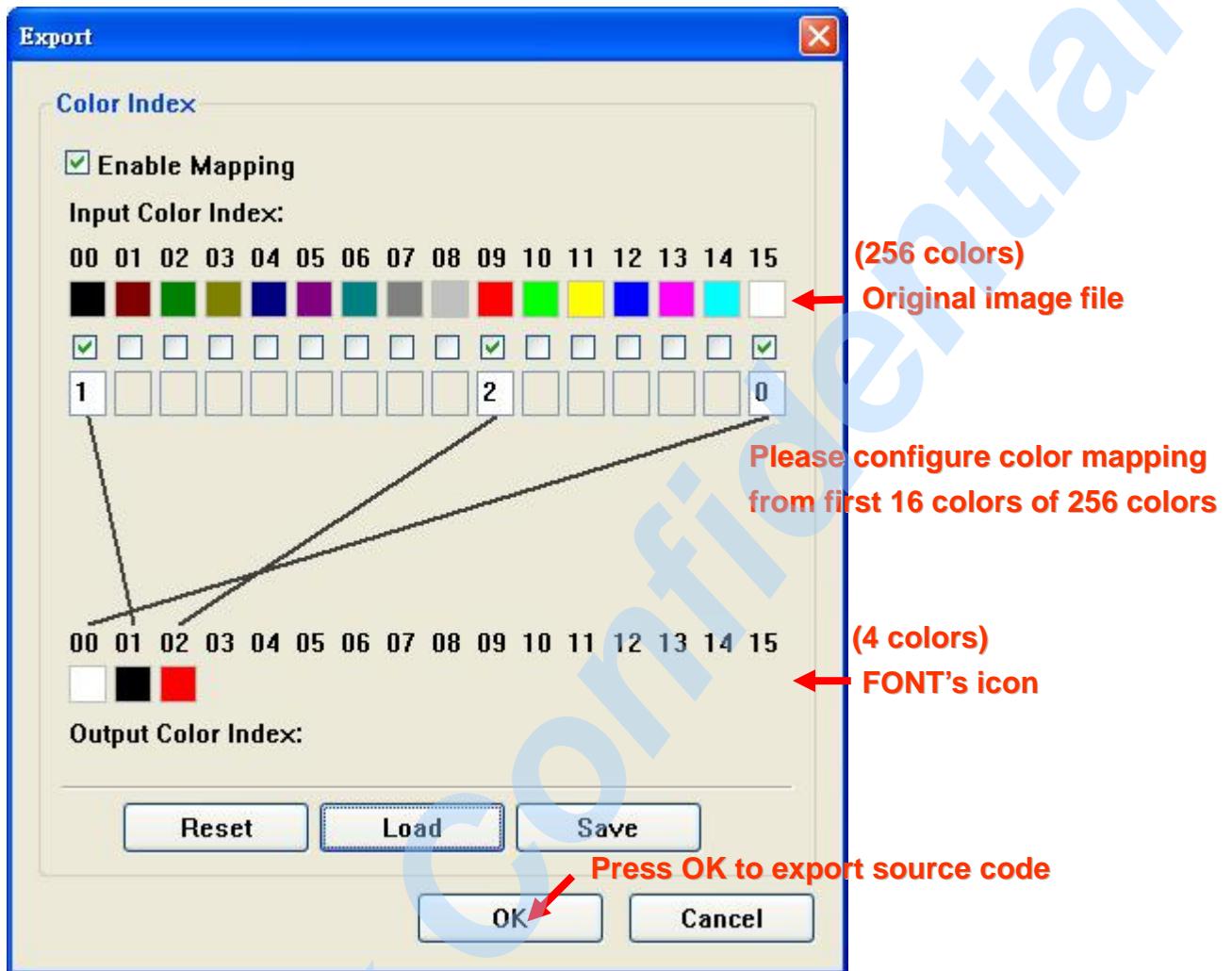


圖 5.32

在圖 5.32 中，選擇 Load 對話框，載入如下的 Font Color Mapping 之後，即會呈現。是但是 Unicode 字庫轉出的 font bitmap 只有兩個 bit, Input color index 和 Output Color Index 是一樣的，因此不需要做 mapping，但是那是目前的調色盤(Output Color Index) 白色是 0，黑色是 1，所以才不需要做轉換，如果調色盤變了，那還是得用 Font Color Mapping 來對映。最後按了 OK 鍵後，即可輸出 source code。

Index	Color Value
0	1
1	255
2	255
3	255
4	255
5	255
6	255
7	255
8	255
9	0
10	255
11	255
12	255
13	255
14	255
15	0

圖 5.33

(4) 使用新的 UI Design Studio Pro 轉換 String Table 和 Font bitmap 到 C source code

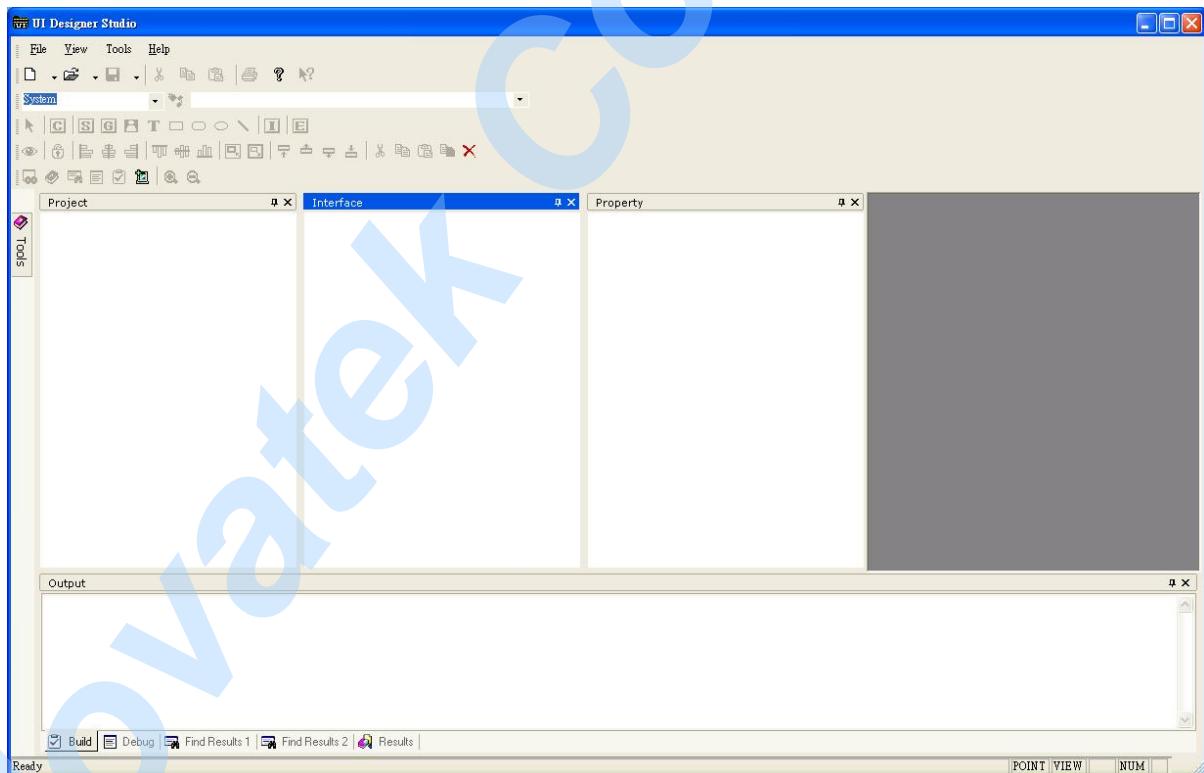
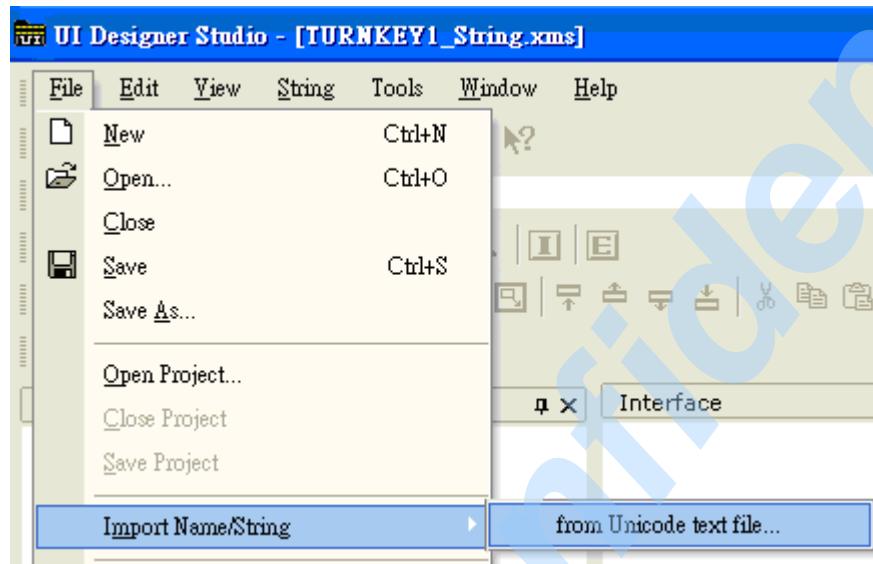
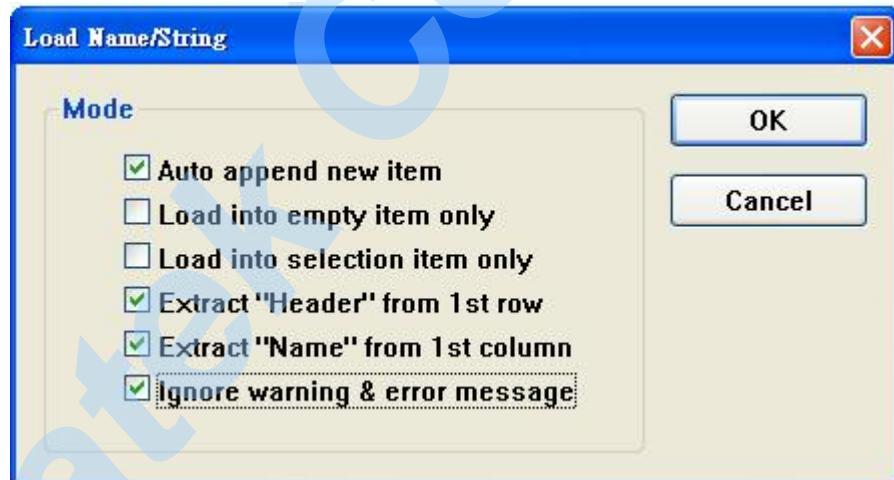


圖 5.34

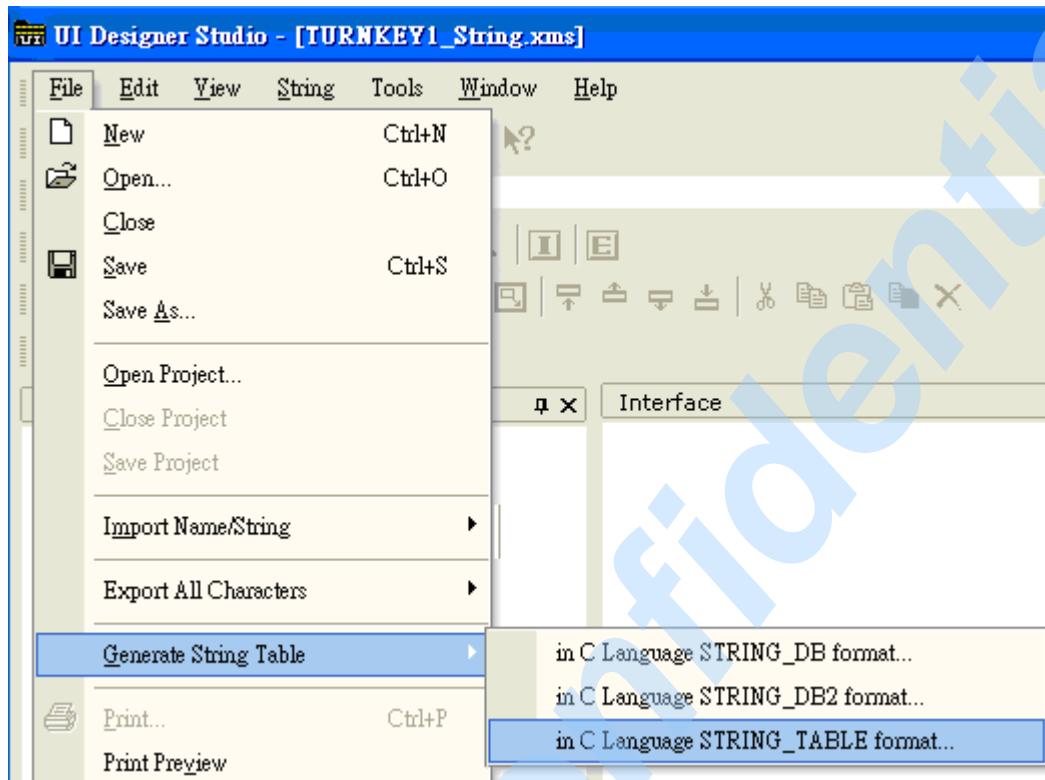
- a) 開啓\TURNKEY1_String\xmb\StringTable.xls
- 另存新檔選擇 Unicode(TXT)格式
 - 儲存到\TURNKEY1_String\xmb\StringTable_Unicode.txt
- b) 開啓 TURNKEY1_String.xms
- File → Import Name/String → from Unicode text file



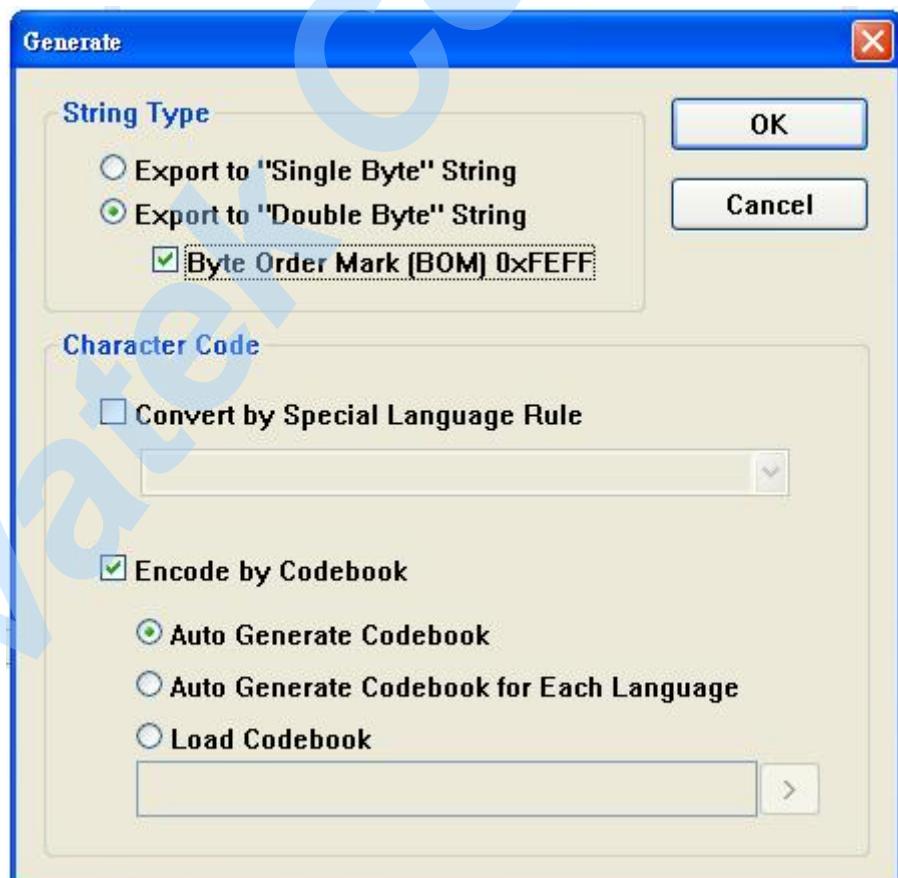
- b.2) 選擇\TURNKEY1_String\xmb\StringTable_Unicode.txt
 b.3) 出現 Load Name/String 對話框，勾選下列選項按 OK。



b.4) File→Generate String Table→in C Language STRING_TABLE format



b.5) 出現 Generate 對話框，勾選下列選項並按 OK 。



- b.6) 儲存對話框，請儲存到\TURNKEY1_String\code\TURNKEY1_String.c。
b.7) 出現下列對話框，請按「是」。

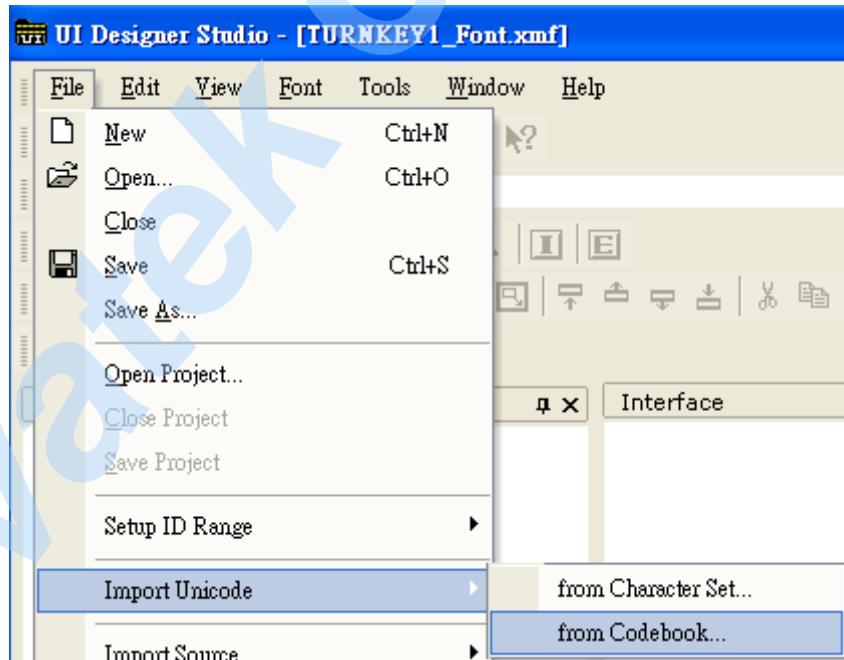


- b.8) 儲存對話框，請儲存到
TURNKEY1_String\xmb\TURNKEY1_String_Codebook.txt
b.9) File→Save
b.10) File→Exit

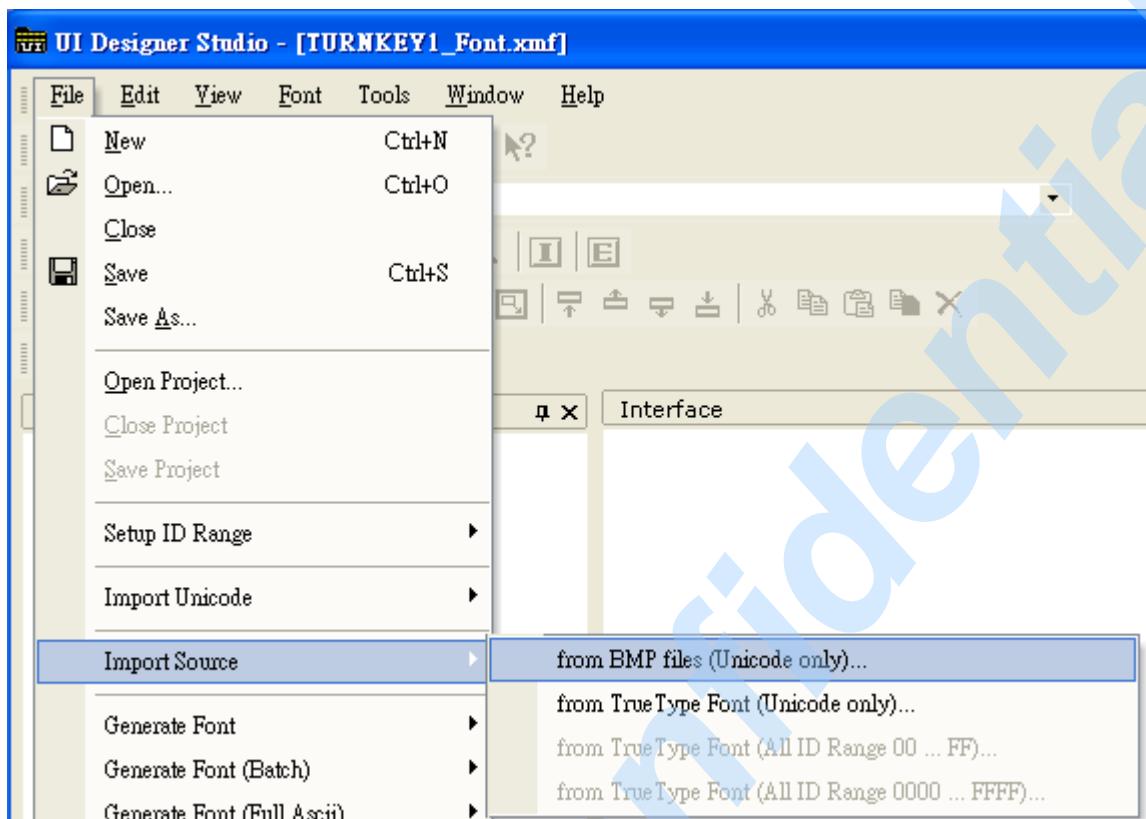
c) 開啓 TURNKEY1_Font.xmf

如果要使用 Unicode 字庫，仍然需要做此程序，但是要先使用圖 1 的工具轉出來的 Unicode binary file 要先全部轉成 font bitmap (轉換 bitmap 的工具下列會在介紹)，然後再使用 UIDS 轉成自己的 Unicode binary file code 字庫。下面介紹的轉換是只針對 String Table 會用到的 font bitmap 而已，如果需要針對多媒體(例如 MP3 功能)，則需要用 Unicode 字庫，轉換 Unicode 字庫的流程，後續會在補充在此份文件裡面。

c.1) File→ Import Unicode→from Codebook



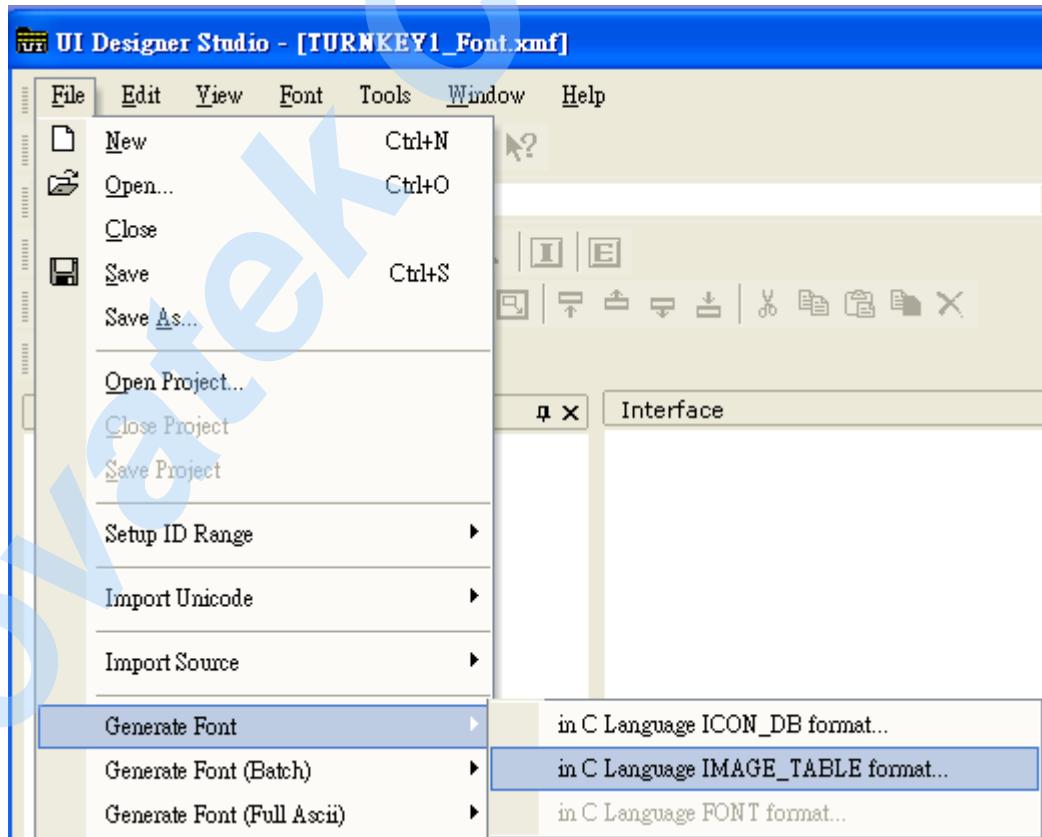
- c.2) 選擇 TURNKEY1_String\xmb\TURNKEY1_String_Codebook.txt
c.3) File→ Import Source→from BMP files



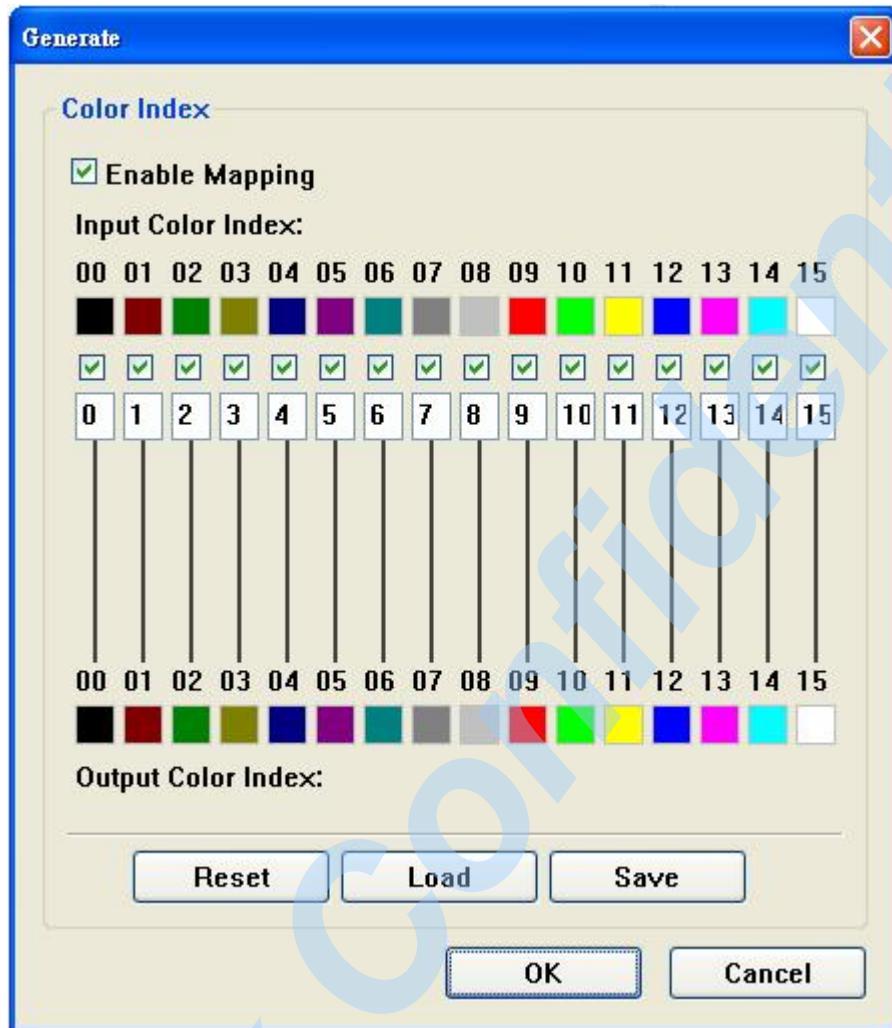
c.4) 選擇\TURNKEY1_String\font_database\中所有 BMP 檔案。

c.5) File→Save

c.6) File→Generate Font→ in C Language IMAGE_TABLE format



c.7) 出現 Generate 對話框



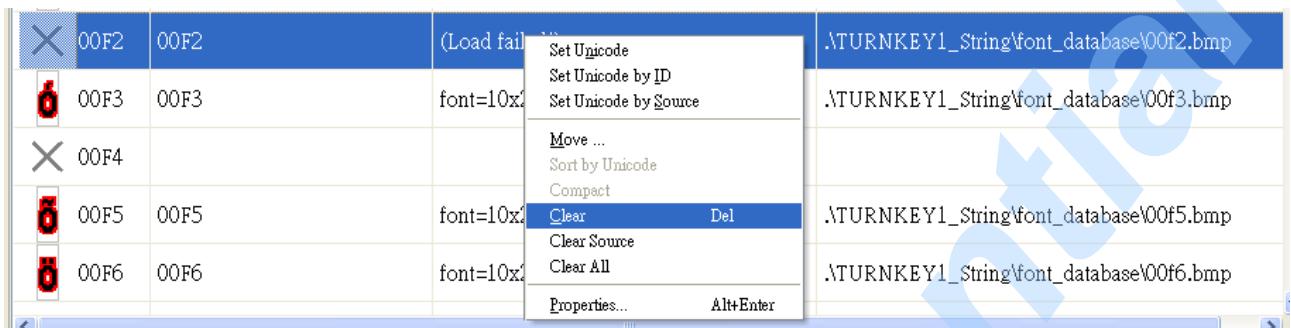
- i. Load
- ii. 選擇\TURNKEY1_String\xmb\Font_ColorMapping.TXT 並按 OK。
- iii. 儲存對話框，請儲存到
\TURNKEY1_String\code\TURNKEY1_Font.c。
- iv. 若過程中，出現下列錯誤對話框，表示找不到相對應的字型。若沒有發生錯誤，請直接跳到 3.8。



- v. 請在 UI Tool 中的尋找 Fail 的項目，如下圖。

X 00F2	00F2	(Load failed!)	\TURNKEY1_String\font_database\00f2.bmp
--------	------	----------------	---

vi. 滑鼠按右鍵，選擇「Clear」。



vii. 「Clear」之後，選項應該呈現空白如下，暫時不用理會，等相對應自行補足後，再轉一次字串。



viii. 若曾經發生找不到對應字型的錯誤，請重複 3.6 到 3.7.3。若否，請下一步驟。

乙、File→Save

丙、File→Exit

d) 複製\TURNKEY1_String\code\底下所有*.c 和*.h 檔案到

\Project\611Demo\SrcCode\UIDisplay\UIResource

e) 完成

6. UniFontGui.exe 字庫擷取工具介紹

此應用程式用來抓取 Microsoft Windows 的 font，副檔名為 ttf 的 font database，將其放在 C:\WINDOWS\Fonts 底下。如圖 6.1 所示，選擇 汇入 TTF button 進入圖 6.2。

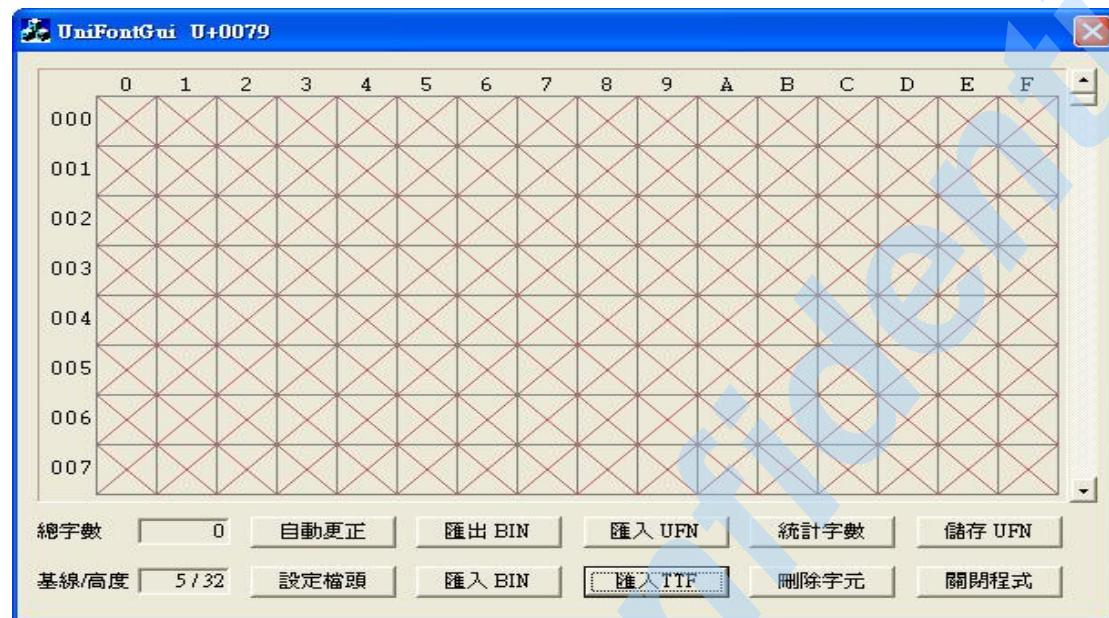


圖 6.1



圖 6.2

在圖 6.2 中，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。另外高度和寬度並非是實體字型的高度和寬度，目前這方面得靠經驗累積才能快速選擇最接近的字型。

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 6.3 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，它是華康儷黑 字形名稱。



圖 6.3

切換作業系統語系如下：

我的電腦->控制臺->地區語言及選項->



圖 6.4

7. Unicode binary file to font bitmap

從微軟作業系統所抓取的 Font (Binary file)需要轉成 bitmap 格式才能給 UI designer studio 轉換成 C source code。首先利用圖 1 的應用程式轉出副檔名爲 ufn 的 binary file 並且放到 `UnicodeToFontBMP\FontToBMP` 路徑下，在 UniFont.cpp 的 main 函數中已經有一些語系及符號，可自己增加或刪除，此外，guiFontGap 表示這個 font 的左右兩邊要留多少 Gap Line，假如左右兩邊要各留兩個 gap line，那 guiShiftRight 就要設爲 2。假如 guiShiftRight 設爲 1，那左邊會留一個 gap line，右邊會留兩個個 gap line。會這樣分的原因是字型字庫(.ttf 檔)有時候客戶可能會自行給我們，如果都用相同的 gap，那字串的 font 和 font 之間可能很靠近，會不好看。

目前只需要動到 main 函數這邊就可以轉出 bitmap，假如想瞭解 bitmap 組成格式，請自行 trace 相關的 source code。

```
int main()
{
    #if 1
        //All unicode
        printf("/*--- All Unicode ---*\n\r");
        uiFontStart = 0x0000;
        uiFontEnd   = 0xFFFF;
        guiFontGap   = 4; //Reserve two gap lines at left and right side.
        guiShiftRight = 2;
        LoadFont(uiFontStart, uiFontEnd);

    #else
        //Latin
        printf("/*--- Latin ---*\n\r");
        uiFontStart = 0x0000;
        uiFontEnd   = 0x017F;
        guiFontGap   = 4; //Reserve two gap lines at left and right side.
        guiShiftRight = 2;
        LoadFont(uiFontStart, uiFontEnd);

    //notation
    printf("/*--- notation ---*\n\r");
}
```

```
uiFontStart = 0x2000;  
uiFontEnd    = 0x20FF;  
guiFontGap   = 4; //Reserve two gap lines at left and right side.  
guiShiftRight = 2;  
LoadFont(uiFontStart, uiFontEnd);  
  
//notation  
printf("/*--- notation ---*\n\r");  
uiFontStart = 0x3000;  
uiFontEnd    = 0x303F;  
guiFontGap   = 4; //Reserve two gap lines at left and right side.  
guiShiftRight = 2;  
LoadFont(uiFontStart, uiFontEnd);  
  
//Greek  
printf("/*--- Greek ---*\n\r");  
uiFontStart = 0x0370;  
uiFontEnd    = 0x03FF;  
guiFontGap   = 3; //Left side is one gap line and right side is two gap lines.  
guiShiftRight = 1;  
LoadFont(uiFontStart, uiFontEnd);  
  
//Cyrillic (Russian)  
printf("/*--- Cyrillic (Russian) ---*\n\r");  
uiFontStart = 0x0400;  
uiFontEnd    = 0x04FF;  
guiFontGap   = 3; //Left side is one gap line and right side is two gap lines.  
guiShiftRight = 1;  
LoadFont(uiFontStart, uiFontEnd);  
  
//Chinese (Tradition and Simplified)
```

```

printf("/*--- Chinese (Tradition and Simplified) ---*\n\r");
uiFontStart = 0x4E00;
uiFontEnd   = 0x9FBF;
guiFontGap   = 4; //Reserve two gap lines at left and right side.
guiShiftRight = 2;
LoadFont(uiFontStart, uiFontEnd);

//Korean
printf("/*--- Korean ---*\n\r");
uiFontStart = 0xAC00;
uiFontEnd   = 0xD7AF;
guiFontGap   = 4; //Reserve two gap lines at left and right side.
guiShiftRight = 2;
LoadFont(uiFontStart, uiFontEnd);

//chinese notation
printf("/*--- Chinese notation ---*\n\r");
uiFontStart = 0xFF00;
uiFontEnd   = 0xFF5F;
guiFontGap   = 4; //Reserve two gap lines at left and right side.
guiShiftRight = 2;
LoadFont(uiFontStart, uiFontEnd);
#endif
return 0;

```

在 LoadFont 的函數裡面主要是找出 Unicode binary file 裡面組成 font 的資料，然後送到 BmpStructureField 這個函數去組成 bitmap，在 BmpStructureField 裡面會去修正字型的寬高，這邊的流程尚無法考慮到每一國語系的修正，仍有改善的空間。另外調色盤只用 1 個 bit 表示黑白兩顏色，而每一個 pixel 組成是用 8 個 bit 組成，bitmap 的 header 參考註解。

程式執行後，會產生所有 font bitmap 在 UnicodeToFontBMP\ FontToBMP 的路徑下，

這時候要將要複製到 UnicodeToBMP\Project_Font\Unicode_Fontr 路徑下 (不一定要此名稱,但是程式中要修改路徑)。

8. 使用 *String Codebook* 挑選出 *string table* 在用的 font

由於轉出的 font 數目過多，假如全部轉成 c source code，firmware size 將過於龐大，因此需要挑選出只針對這個 project 使用的 font，如圖 8.1 所示。圖 8.1 的產生是利用 UI designer studio 將 string stable(excel file) 轉成 c source code 後，可以另存一個 StringTable_Codebook.TXT 的檔案，此檔案就是這個 string stable 會用到的所有 font，**注意**: UI designer studio 另存的.txt 檔是 Unicode 格式，需要另存成 ANSI 格式，否則利用 fopen 去讀取時會有問題。

接下是用 Project_Font.cpp 的程式去讀入 Project_StringStable.txt (ANSI 格式，不一定要此名稱，但是程式中要修改)，然後挑選出這個 string table 所需要用到的 font bitmap，轉出的 font bitmap 就可以用 UI designer studio 轉出 OSDFont.c/OSDFont.h，目前是將選出來的 bitmap font 放到 StringTable_Font folder (可自行放到任一個 folder 中)。另外，在 Unicode_Font subfolder 的 CoveredFont subfolder 中，擺放的是經過修改過後的字形以及 Kodak 所給的 font database 有缺字並自行補上。缺字的部分請務必放到 Unicode_Font subfolder 下面，避免轉出的 OSDFont.c/OSDFont.h 會有缺字的情形。

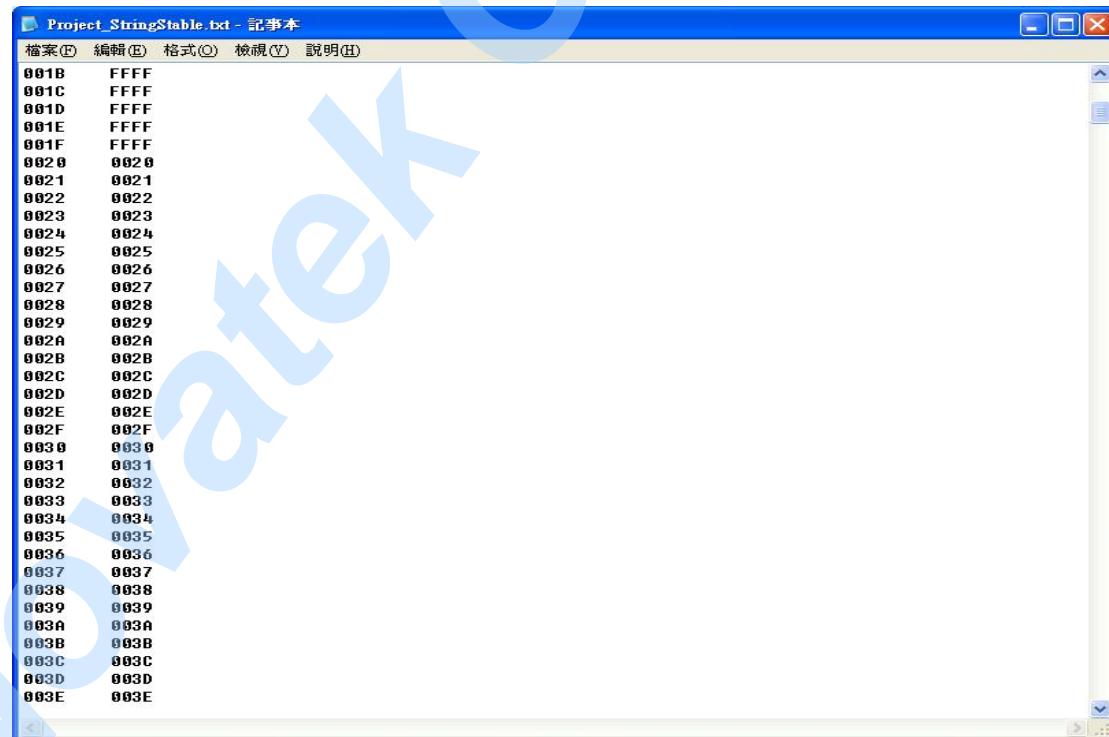


圖 8.1

9. ANSI CODE to UNICODE (多媒體)

對於 UI menu 的開發所需要用到字串，假如前面 2 個 byte 有 Unicode 的前導碼 0xFEFF，那在字串後面所跟隨的字型，都是用 Unicode 去顯示。然而，對於多媒體的名稱有可能是以 ANSI code 或是 Unicode 的命名方式，如果檔名是 Unicode 的命名，那對於各國語言方式混合的字串名稱，均可以顯示。但是，如果檔名是 ANSI code 的編碼方式，就只能顯示當前所選擇的語系，同時必須將其轉換至 Unicode，再予以顯示出來，其餘各國語言的顯示將呈現亂碼。相關的檔案請參考到 ShowStringFuncs.c 的 OsdDrawString function、CodePageApi.c 和 CodePageApi.h。

以 MP3 為例，目前的 Unicode 字庫可以顯示 MP3 檔名，只要檔名是 Unicode 格式，即使在一個字串裡面有著混合不同語系的字型，仍然可以顯示，但如果是 ANSI code 則需要轉換至 Unicode，而且只能顯示當前所選擇的語系，其餘字型將呈現亂碼。Unicode 字庫也支援詞曲同步顯示的功能，顯示的方式就看歌詞是 ANSI 或是 Unicode 編碼。後面會有章節介紹 Unicode porting 和 MP3 Lyric (詞曲同步)。

10. 背景、外框、主體

目前 project 的 font icon 繪圖方式，使用背景、外框、主體來描述一個 icon，如圖 10.1 為一個數字 0 的 icon，背景是白色，外框是紅色，主體是黑色，因此每一個 pixel 使用 2 個 pixel 來表示，然而在圖 2.3 中所取得的 Unicode pixel 資料，只有使用一個 bit 來表示，所以只有用背景和主體來表示，然而在開發 UI menu 上，因為有背景顏色來突顯所選擇的項目，所以不需要使用外框，然而在 UI flow mode 下將有可能因為背景關係，而影響到字體的顯示，例如在 playback mode 下的每一張影像都有日期、時間等，使用 Unicode 的方式顯示，因為缺少外框，所以字體有可能因為背景關係而影響到顯示，因此我們使用兩個資料結構來分別存放用 1 個 bit 和 2 個 bit 來表示 pixel 的資料。使用 2 個 bit 代表一個 pixel 只需要保留到 Unicode 的前 127 個即可 (ANSI、ASCII、UNICODE 前 127 個都是相同)，而這些資料都是事先定義好的，不可以更改到其資料結構。而使用 1 個 bit 表示一個 pixel 的，則是可以重複存取相同的資料結構，當前一個字串已被處理完之後，即可重複使用。相關的檔案可以參考到 OSDFont.c 及 OSDFont.h。

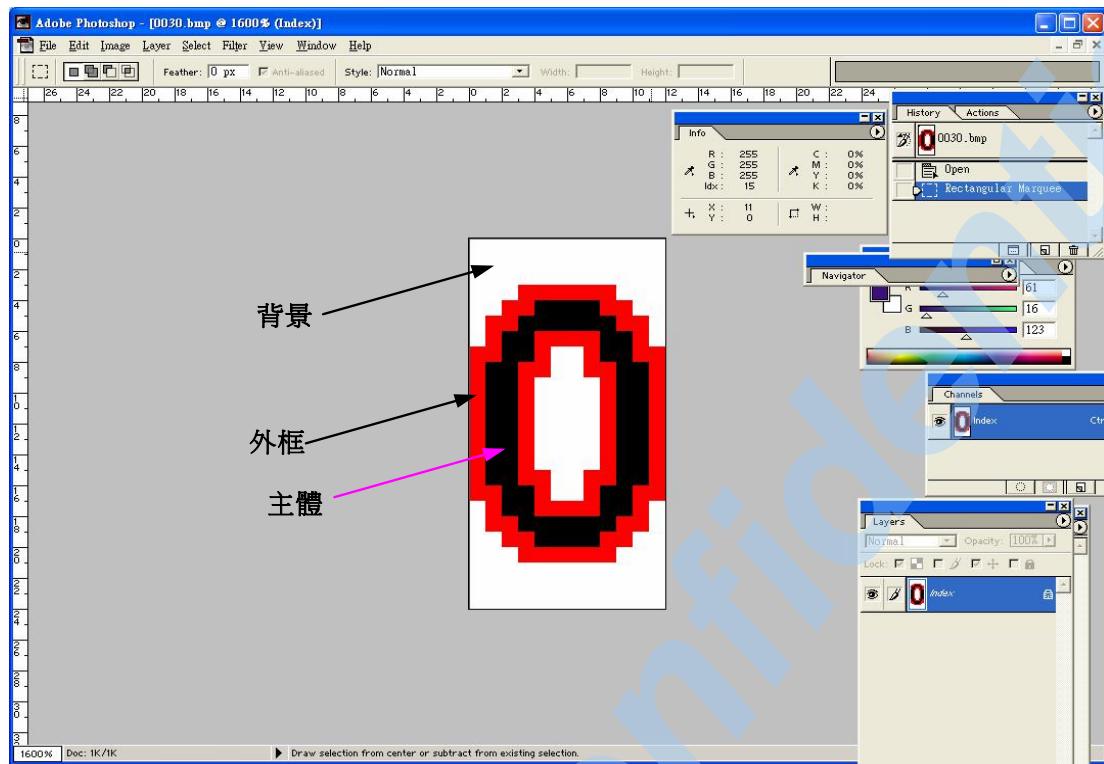


圖 10.1

11. 16MB、32MB DRAM 中執行

由圖 1.1 的應用程式所轉出來的 Unicode 字庫，其 binary file 約 3.3MB，經刪減不必要的資料後，字庫可減少至 2.5MB，對於使用 32MB DRAM 的 project 來說，可以將整各字庫全部載入到 DRAM 中執行，以加快 UI menu 的執行速度。載入字庫的動作是在 SystemInit.c 的 SystemInit function 去執行，時間約為 600ms，如果載入字庫的動作是在按了 Menu 鍵之後才去執行，那此時間將會導致第一次顯示 Menu 的速度過慢。

在使用 16MB DRAM 的 project 上來使用 Unicode 字庫，對於記憶體的使用將會很吃緊，為了在速度和記憶體使用間取得平衡，只有將 header 和相關的 table (約 192KB)，以及字庫 0x0000~0x4DFF (約 207KB, 除了中文和韓文外) 載入至 DRAM 中，以 photo mode 下簡體中文的拍照及設置兩個 page 測試，結果如圖 11.1 所顯示，在兩個 page 之間切換的速度仍然是稍微慢一點。會變慢的原因是因為使用 PStore function 到 NAND flash 所存放 Unicode 字庫的地方去搜尋字型，而每存取一個字型，PStore function 就會從 Storage 的 CARD 切換至 NAND，因此才會造成存取時間非常慢，例如在圖 11.1 的設置 page 的第一頁裡，共有 18 個中文字，因此要呼叫 PStore function 共 18 次。

為了加快速度上的顯示，改以 page 為單位，每切換到不同 page 時，Storage 會從 CARD

切換至 NAND，然後就不在切回 CARD，直到這一個 page 顯示完之後，才會切回至 CARD 上，所測試的結過如圖 11.2 所示，這種方式在 UI menu 的顯示上會加快許多。然而在 UI 中很多地方要 access card，這時候還是得記得要先切回至 card，而每個 access card 都要如此做，會非常不方便，同時也會造成 UI 開發上的困擾。

後來改以字串為單位，例如底下的設置(Page)、日期/時間(Item)、自動關機(Item)、按鍵聲音(Item)、語言設置(Item)，共 5 個字串，因此顯示這一頁時，Storage 只要切換 5 次，結果如圖 11.3 所示，其顯示速度尚可接受，而這種方式對 UI 的開發會比較方便，只要在底層 OsdDrawString function 加上 PStore 的 GetAccessRight 及 ReleaseAccessRight 即可，然而當一個 page 的字串數目變多，Storage 所需要切換的次數也將增加，顯示速度仍然會變慢。

上述的兩種方式都會面臨一種情況，就是現在 Storage 有可能連續存取 card 上的資料，然而又必須顯示一些資訊給使用者知道，而此時根本又需要切回至 NAND 去搜尋字型，同樣也會造成 UI 開發上的困擾。

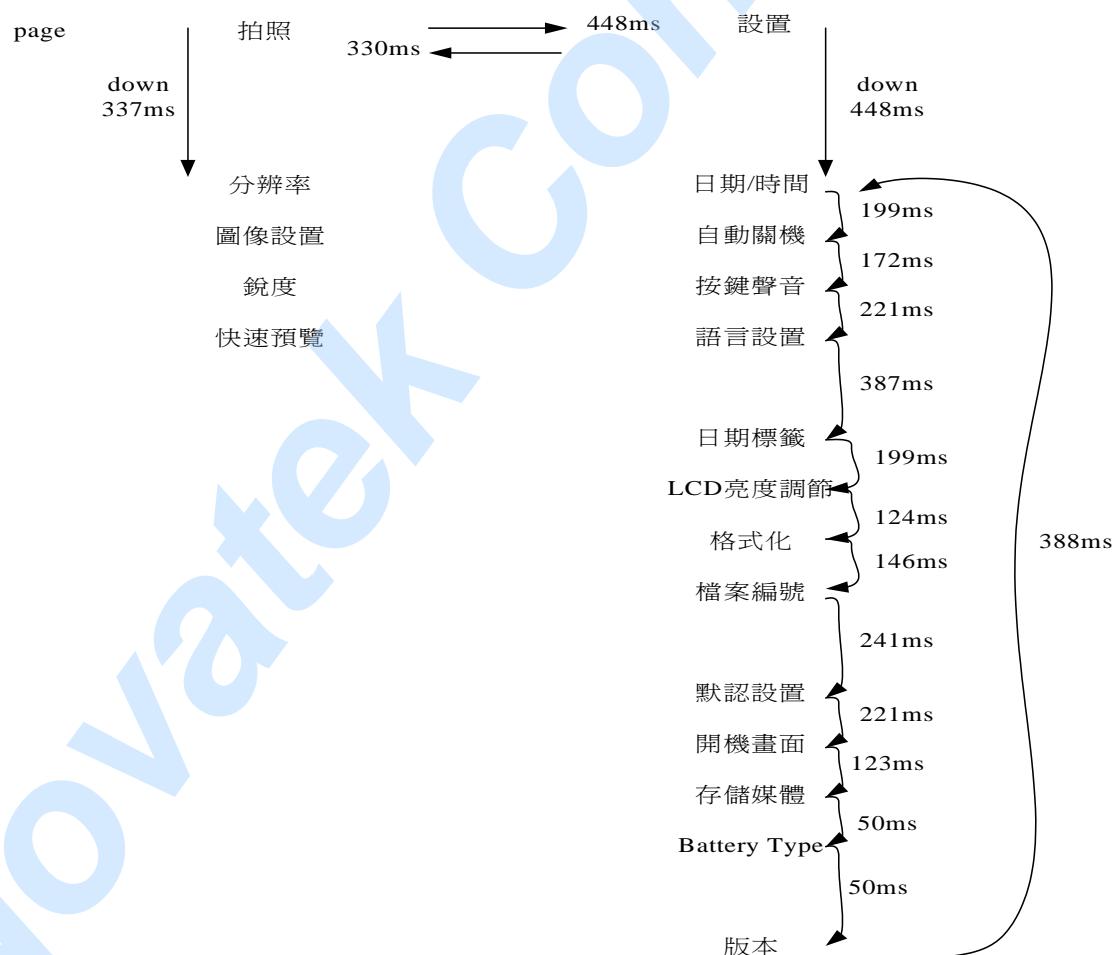


圖 11.1

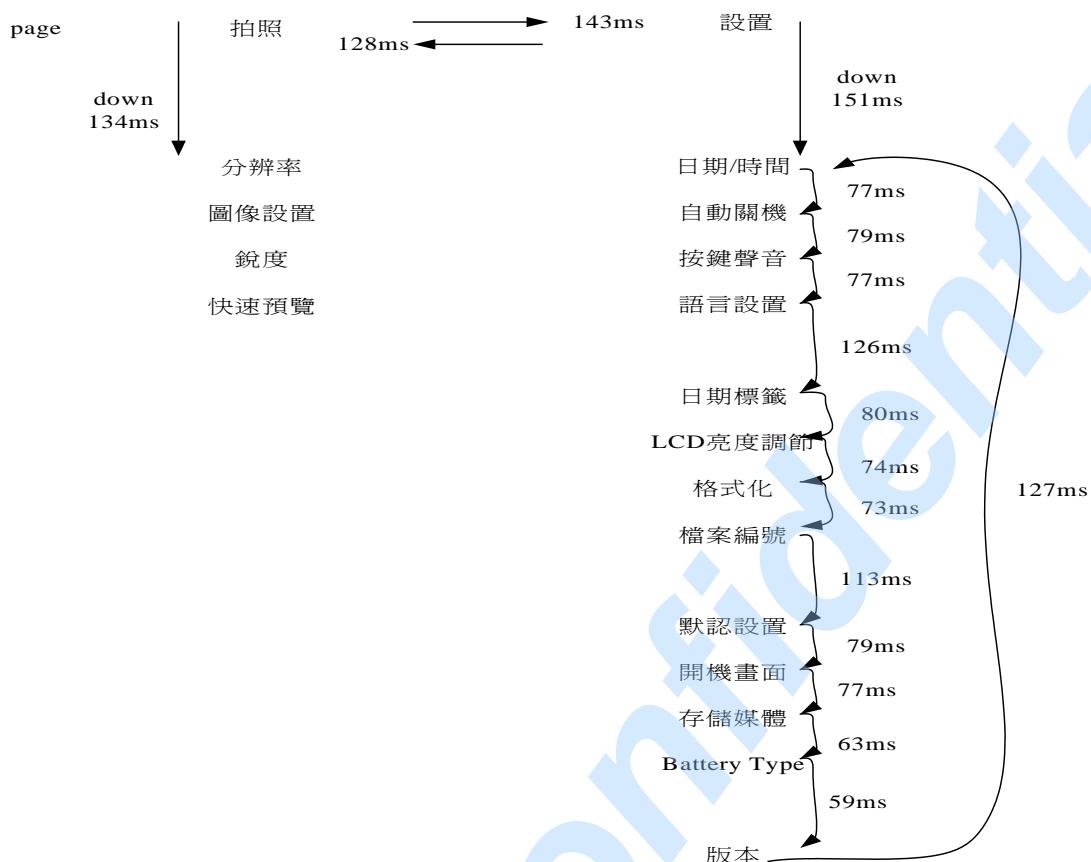


圖 11.2

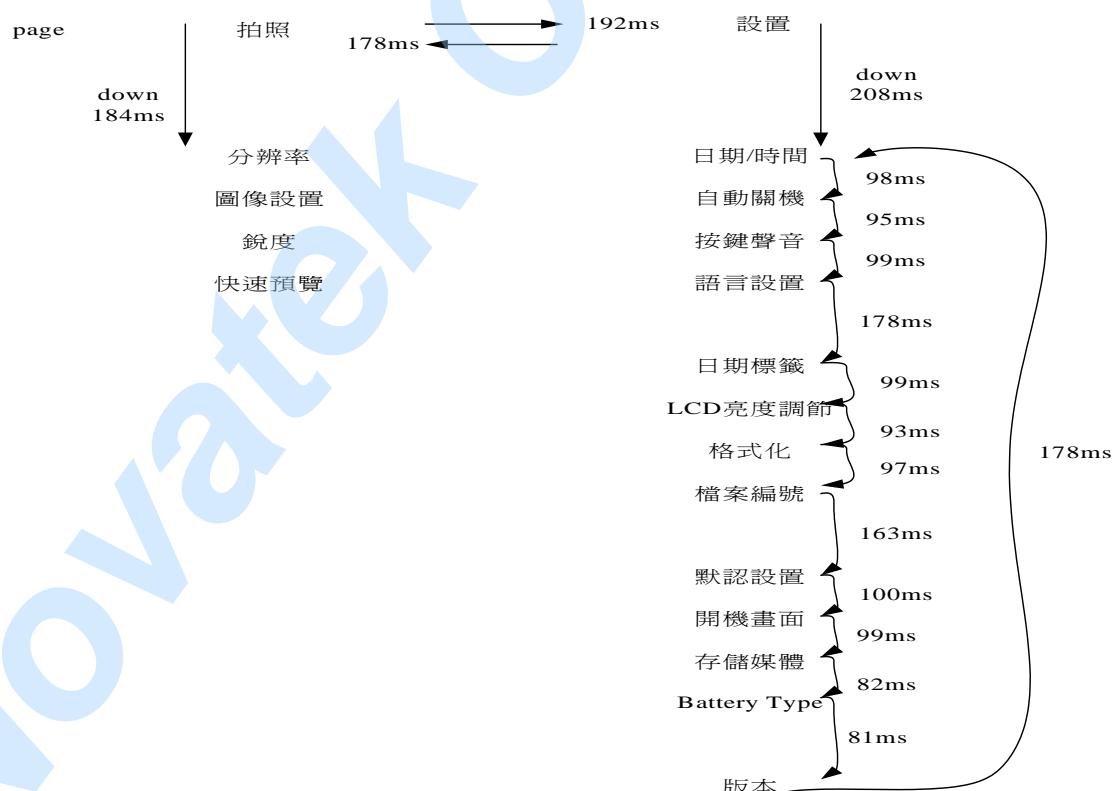


圖 11.3

12. 使用視窗介面程式產生 font bitmap

這個章節將介紹使用視窗介面程式產生 font bitmap:

- (1) 首先打開應用程式，如圖 12.1 所示。



圖 12.1

- (2) 在 open *.ufn binary file 前，先輸入高度(介於 18~25)，如果沒有輸入會產生如圖 12.2 的錯誤訊息，會提示再次輸入高度的訊息。



圖 12.2

- (3) 如果輸入的高度不在 18~24 範圍內，會提示如圖 12.3 的錯誤訊息，會提示再次輸入高度的訊息。



圖 12.3

(4) 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 12.4 所示，這表示轉出的 UFN binary file 所選擇的字型太大。

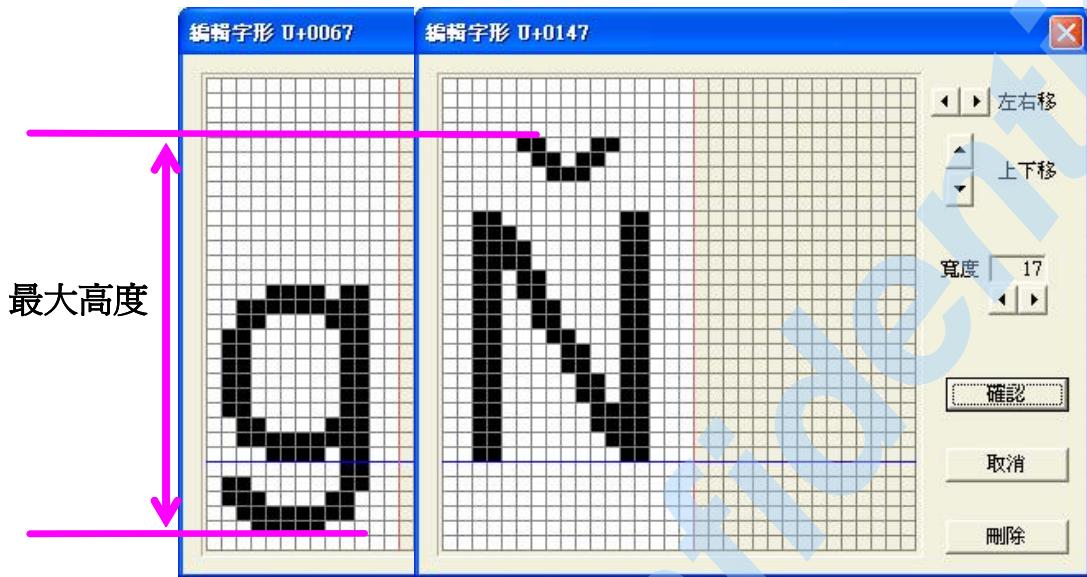


圖 12.4

(5) 輸入高度確定 OK 後，Open *.ufn binary file，如圖 12.5 所示。

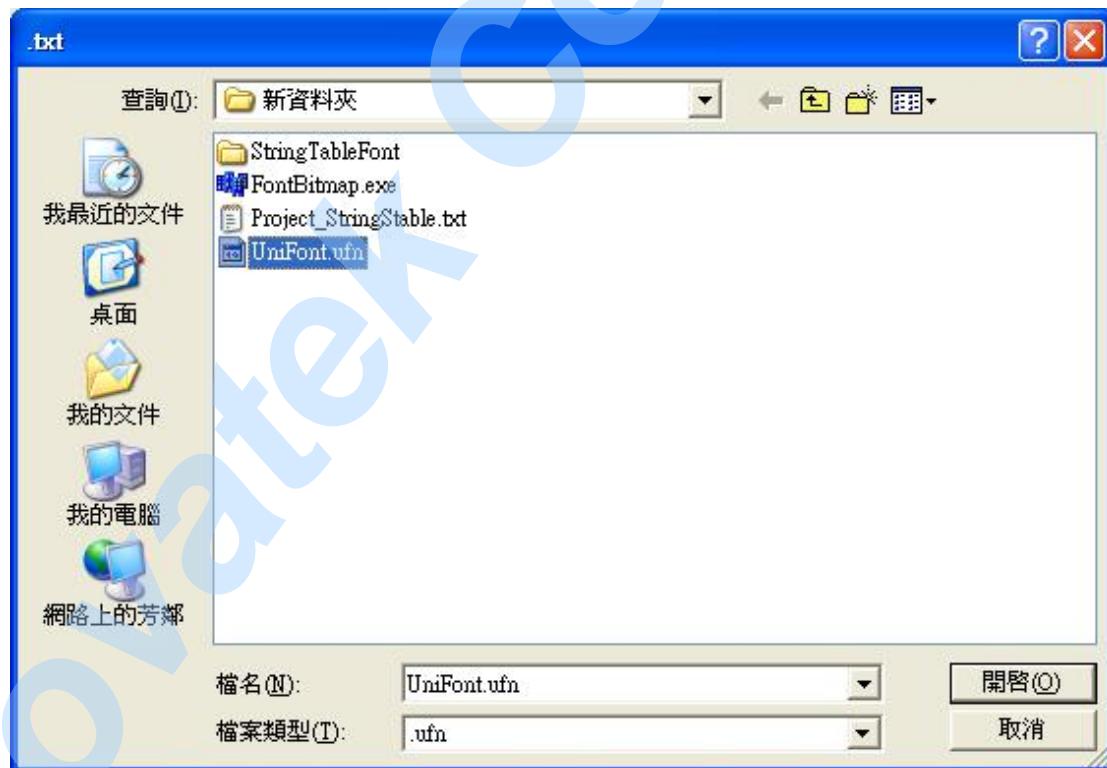


圖 12.5

(6) Load String Codebook , ANSI 或是 Unicode 格式都可以，如圖 12.6 所示。

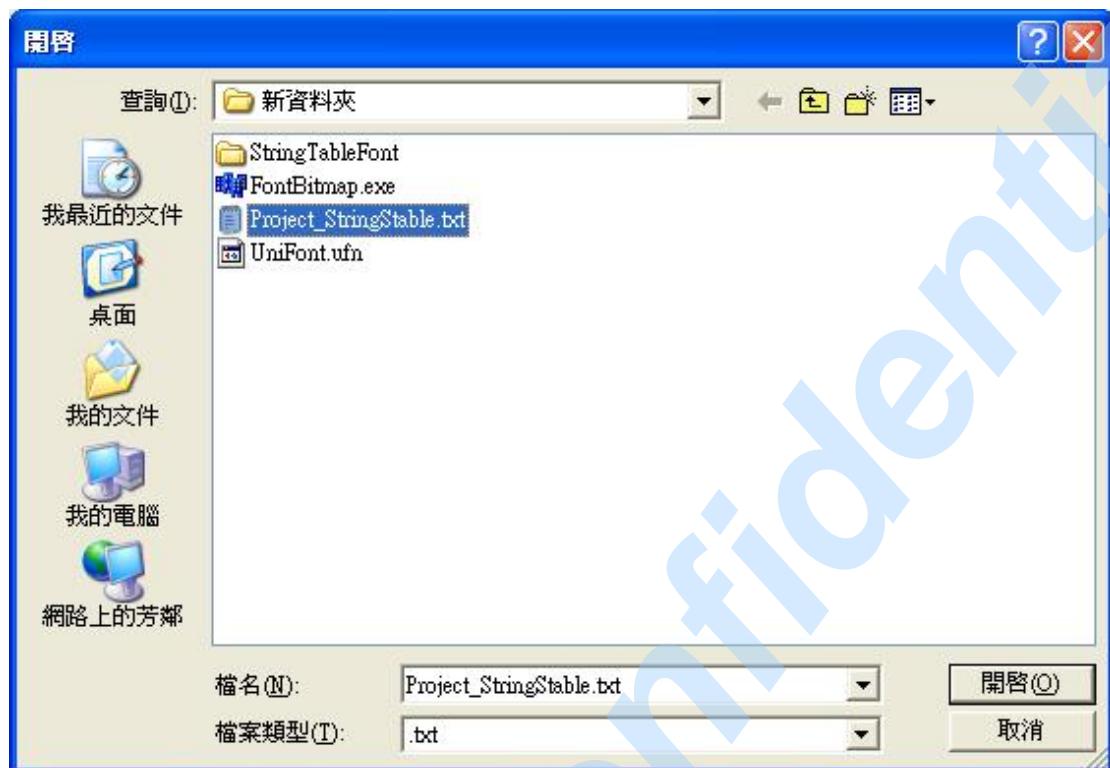


圖 12.6

(7) Export Font Bitmap , 如圖 12.7 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 。

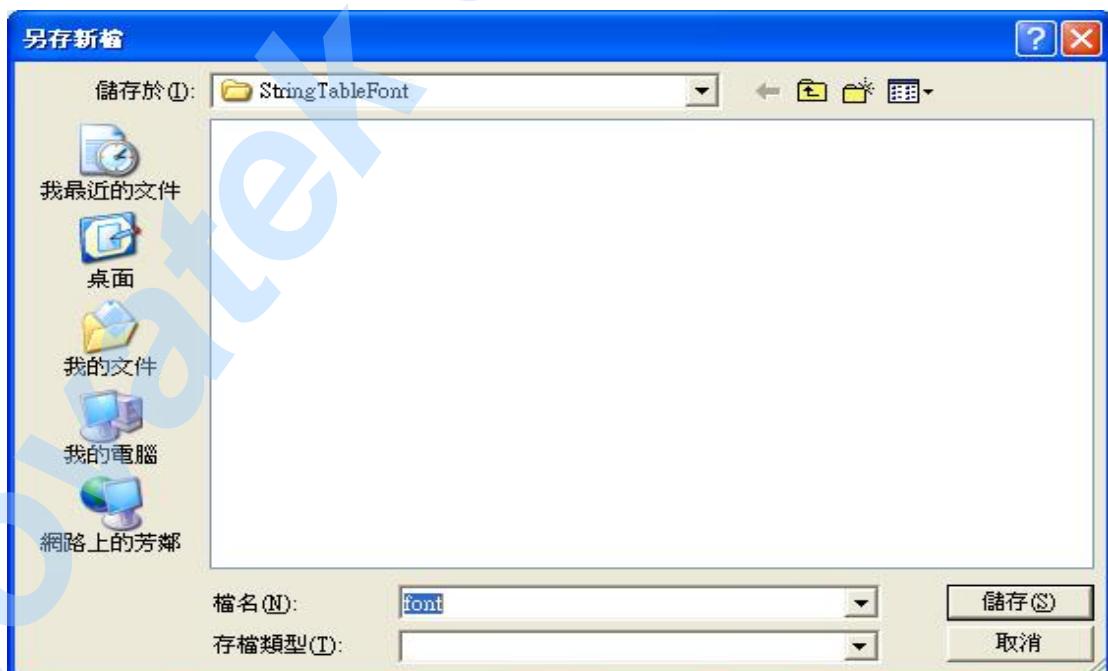


圖 12.7

13. Unicode Porting and MP3 Lyric

這裡簡單說明如何利用已存在的 Unicode binary file (Unicode.ufn, 2.5MB) 來實現 Unicode 和 MP3 詞曲同步，以 DD717(NT96613) project 的 source code 來說明，比較 DD717_Real 和 DD717_MP3 的 source code。

(1) Unicode Porting

- (1) Inlcude\Subsystem\DrawLib\draw_lib.h
- (2) Include\Type.H
- (3) LIB\LIB_src\Driver\Storage\NAND2K.c 改為 NAND_2K.c 即可 (large page)
- (4) LIB\LIB_src\Driver\Storage\NAND_PS.c 改為 NAND_PS_2K.c 即可 (large page)
- (5) LIB\LIB_src\SubSystem\DrawLib\draw_lib.c
- (6) Project\611Demo\SrcCode\SysCfg\SysCfg.c、SysCfg.h
- (7) Project\611Demo\SrcCode\SysInit\SystemInit.c
- (8) Project\611Demo\SrcCode\UIDisplay\IconsDB\ADPK\ 新增加的檔案有 UniFontApi.h、UniFontApi.c、OSDUnicodeDirectImp.h、OSDUnicodeDirectImp.c、 OSDUnicodeApi.h、OSDUnicodeApi.c、OSDUnicode.h、OSDUnicode.c、CP950.h、 CP949.h、CP936.h、CP932.h、CP874.h、CP1252.h、CP1251.h、CP1250.h、 CodePageApi.h、CodePageApi.c，其中 CP950.h、CP949.h、CP936.h、CP932.h、 CP874.h、CP1252.h、CP1251.h、CP1250.h、CodePageApi.h、CodePageApi.c 為 ANSI code 轉換至 Unicode 所需用到的相關檔案，CPxxx.h 為每一個 ANSI 語系 轉換至 Unicode 時，所用到的 Code Page Table。
- (9) Project\611Demo\SrcCode\UIDisplay\IconsDB\ADPK\OSDFont.c、OSDFont.h、 OSDIcon.c、OSDIcon.h。在 OSDFont.h 的#define UNICODE_HEADER 70 為 每一個字串所能顯示的最大字元數
- (10) Project\611Demo\SrcCode\UIFlow\ADPK\ListBox.c、ListBox.h
- (11) Project\611Demo\SrcCode\UserPStore\UserPStore.c、UserPStore.h
- (12) Project\611Demo\SrcCode\CommandTsk.c
- (13) Project\611Demo\Makefile
- (14) Project\611Demo\SrcCode\UIDisplay\OSDString\ADPK 之前都是用流水號取代 Unicode 的編碼，但這裡 StringTable.h 和 StringTable_xx.c (xx 代表每一個語系的縮寫) 需要用 UI designer studio 重新將字串直接轉換成 Unicode 的編碼。轉換流程如圖 13.1 所示：

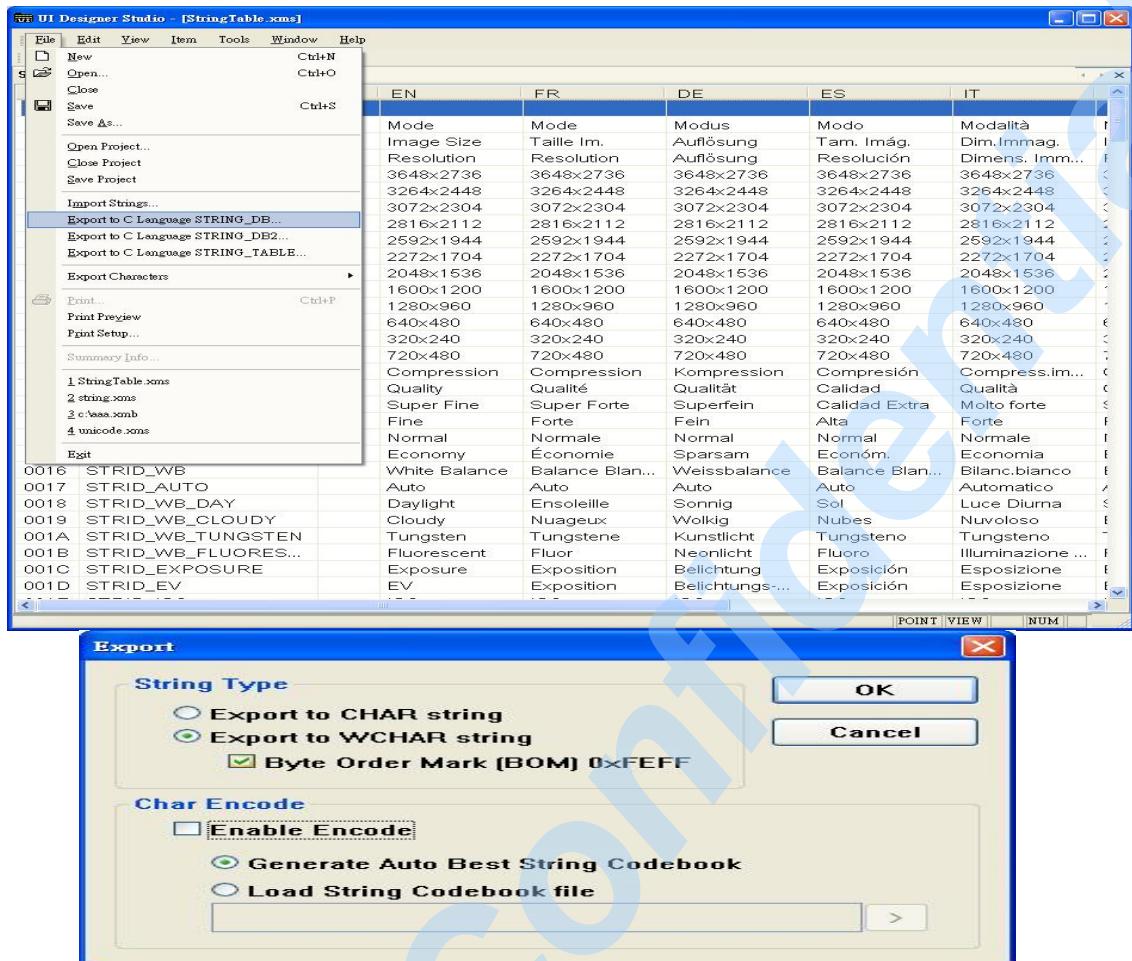


圖 13.1

(15) Project\611Demo\SrcCode\UIDisplay>ShowStringFuncs.c 裡面的 OsdCalculateString subroutine 只有針對 MP3 詞曲同步，如果不需要 MP3 詞曲同步播放功能，可將此部份拿掉。

(16) Unicode binary file 請務必使用 Unicode.ufn (2.5MB)，請將此檔案放至 SD card，在 DD717 上有兩種方式可以將 Unicode.ufn PStore 至 NAND flash。

(a) 透過 console:

>ps format

>ps unicode

(b) 進入 Engineering Mode 後，按下 Update Graph 即可。

(2) MP3 詞曲同步

- (1) Application\App_Src\Common\OSCommon.c
- (2) Application\App_Src\Mp3\MP3ShowLyricTsk.c 為新增加的檔案
- (3) Application\App_Src\Mp3\Makefile

- (4) Application\App_Src\Mp3\MP3PlayFileSysFunc.c、MP3PlayFileTsk.c、MP3PlayTsk.c
- (5) Include\Application\MP3ShowLyricTsk.h 為新增加的檔案
- (6) Include\Application\MP3PlayFileSysFunc.h、MP3PlayFileTsk.h、MP3PlayTsk.h
- (7) Include\OSCommon.h
- (8) Project\611Demo\SrcCode\UIDisplay>ShowOSDFuncs.h
- (9) Project\611Demo\SrcCode\UIDisplay>ShowStringFuncs.c 如有 MP3 詞曲同步播放功能，需加入 OsdCalculateString subroutine。
- (10) Project\611Demo\SrcCode\UIFlow\ADPK\UIFlowMP3List.c、UIFlowMP3Tsk.c
- (11) 在作歌詞顯示時，每顯示下一列歌詞前，會先呼叫 ShowOSD_DrawRect 清除目前的歌詞，而一列歌詞太長會顯示跑馬燈，但在一段時間後，會出現如下訊息：
 - ERR: ShowOSD_DrawRect> get memory FAIL!
 - ERR: NULL TempBuffer Error.
 - ERR: ShowOSD_DrawRect Failed這種情況一般都是發生在 memset 或 memcpy 時，要拷貝的資料長度已經比規劃的 buffer 還要大，造成覆蓋到 buffer 後面的記憶體資料。
- (12) 要注意到的地方有 MP3ShowLyricTsk.h 的 #define LYC_CHAR_MAX 100，OSDFont.h 的 #define UNICODE_HEADER 70，ShowStringFuncs.c 的 OsdDrawString subroutine 的 char OSDString[256]、USHORT uniWideStr[128] 和 char uniStr[256]，OsdCalculateString subroutine 的 USHORT uniWideStr[128] 和 char uniStr[256]。