UNICODE正常字 PDF渲染
技術報告

STK 庫存系統 — prn_to_pdf.prg 擴充

日期：2026-04-22

1. 問題背景

STK 庫存系統使用 GTWVT console mode，所有列印輸出先攔截為 PRN 文字檔，再由 PrnAsk / prn_to_pdf.prg 轉換成 PDF 檔。

系統字型分兩類：

BIG5 造字區（USRFONT）：FA41~FEFE 共 785 槽，客戶自訂字形，存於 USRFONT.BCACHE / .24M / .UMAP

UNICODE 正常字：同樣佔用 FA-FE 槽位，但字形並不存於 USRFONT.*，而是透過 UMAP 記錄 Unicode 碼後，在 PDF 轉換時另行查找字形

[!] 問題：UNICODE 正常字在 PDF 預覽中完全不顯示（空白）

測試檔案 RPT20260422UNICODE.PRN 包含 6 個 FA-FE 字元：

FEFE -> U+5803（塋） BIG5-HKSCS 專屬，標準 BIG5 無

FEFD -> U+5586（嘈） BIG5-HKSCS 專屬，標準 BIG5 無

FBC1, FBC3 BIG5 造字，有 BCACHE 資料

FEF4, FA41 UNICODE 造字區（PUA），有 BCACHE 資料

2. 根本原因分析

2.1 UNICODE 正常字不進入 USRFONT.*

GTWVT 顯示 UNICODE 正常字走的是完全不同路徑（wcharGet / UDA 系統），字形資料從不寫入 USRFONT.BCACHE 或 USRFONT.24M。因此 prn_to_pdf.prg 原有的 BCACHE -> 24M 查找鏈對這類字元完全失效。

2.2 BIG5-HKSCS 專屬字導致 UNICODE_TO_BIG5() 失敗

UMAP 機制可以將 lookupIdx 對應到 Unicode 碼點，再透過 UNICODE_TO_BIG5() 轉成標準 BIG5 雙位元組供 HPDF 輸出。

但 U+5803（塋）和 U+5586（嘈）只存在於 BIG5-HKSCS（cp950 延伸），在標準 BIG5（cp950）中不存在，UNICODE_TO_BIG5() 回傳空字串。

BIG5-HKSCS 編碼：

U+5803（塋） -> FB69 僅在 BIG5-HKSCS，標準 BIG5 查無

U+5586（嘈） -> FB46 僅在 BIG5-HKSCS，標準 BIG5 查無

2.3 原有渲染鏈不完整

擴充 UMAP 機制前，Priority 3 只有 3a 路徑：UNICODE_TO_BIG5 成功才渲染，失敗就直接 RETURN NIL（輸出空白）。需要補上 Priority 3b：BIG5 轉換失敗時，改用 GDI 離屏渲染。

3. 解決方案架構

3.1 渲染優先順序（最終版）

```
DrawUsrGlyph( nLookupIdx, nX, nY, nW, nH )
 |
 +-- Priority 1: USRFONT.BCACHE
 |    AScan s_aUsrBcache -> 8-bit 灰階 alpha
 |    -> HPDF_LoadRawImageFromMem -> HPDF_Page_DrawImage
 |
 +-- Priority 2: USRFONT.24M
 |    AScan s_aUsr24M -> 1bpp 展開為 8-bit 灰階
 |    -> HPDF_LoadRawImageFromMem -> HPDF_Page_DrawImage
 |
 +-- Priority 3: USRFONT.UMAP 回退
     lookupIdx -> s_aUsrUmap[idx+1] = Unicode 碼點
     |
     +-- 3a: UNICODE_TO_BIG5() 成功 -> 雙位元組 BIG5
     |       -> HPDF 文字輸出（使用 s_fontCJK 當前字型）
     |
     +-- 3b: UNICODE_TO_BIG5() 失敗（BIG5-HKSCS 專屬）
             GetGDIFontForCJK() -> GDI 字型名稱
             RenderUnicodeGlyph( nCP, W, H, cFont )
             -> CreateCompatibleDC(NULL) + CreateDIBSection(24bpp)
             -> CreateFontW(wFont) + TextOutW + GdiFlush
             -> BGR->灰階轉換 -> hb_retclen
             -> HPDF_LoadRawImageFromMem -> HPDF_Page_DrawImage
```

3.2 字型對照表（不硬編碼）

為避免硬編碼 GDI 字型名稱，加入 s_aCJKGDIMap 對照表，由 GetGDIFontForCJK() 根據當前 s_cCJKFontName 動態查表：

PDF CJK 字型名稱	GDI 字型名稱	說明
MingLiU	MingLiU-ExtB	細明體 -> ExtB（HKSCS 全覆蓋）
PMingLiU	PMingLiU-ExtB	新細明體 -> ExtB
MHei-Medium	Microsoft JhengHei	黑體 -> 微軟正黑體
MKai-Medium	DFKai-SB	楷體 -> 標楷體
（其他）	MingLiU-ExtB	查無對應時 fallback

4. 踩過的坑與陷阱

以下為開發過程中實際遇到的問題，每個都有明確的症狀與解決方式：

[1] hb_stackReturnItem()->type BCC32 不支援

症狀	E2288: Pointer to structure required on left side of ->
	*** 1 errors in Compile ***
	原因：BCC32 對 Harbour 內部 API 的前向宣告不同
解法	改用 int bOK=0 旗標：渲染成功後設 bOK=1；
	goto done 後以 if(!bOK) hb_retc("") 代替

[2] HDC hDC=NULL W8004 被當成錯誤

症狀	BCC32 把「宣告但未使用的變數」警告 W8004 視為錯誤
	加了 -w8004 旗標也未完全壓制
解法	加 #pragma warn -8004 在 BEGINDUMP 開頭，
	並改用 CreateCompatibleDC(NULL) 直接建離屏 DC，
	省去 GetDC / ReleaseDC，也省掉 hDC 變數

[3] hb_HexToNum() Harbour32 中不存在此函式

症狀	UNICODE.INI 整合方案（LoadUnicodeIni）在執行期呼叫
	hb_HexToNum()，但 Harbour32 無此函式 -> runtime 錯誤
	-> PDF 轉換全部亂碼
解法	放棄 UNICODE.INI 方案，改用 s_aCJKGDIMap
	Harbour 陣列對照表，無任何外部 API 依賴

[4] Python bytes 中 b';r\n' vs b';\r\n'

症狀	patch 腳本中一個 ; 後面誤寫成字母 r 而非 \r，
	導致 Harbour 編譯器 E0030 語法錯誤（行 1518-1519），
	錯誤訊息不易對應到 patch 腳本的哪一行
解法	所有換行一律明確寫 b'\r\n'（CRLF）
	並在 patch 腳本末尾加驗證（搜尋目標字串）

[5] 還原備份版本選錯

症狀	還原到 .bak_unicode_gdi_20260422_1800 後，
	Priority 3b（GDI 修補）消失，UNICODE 正常字又不顯示
	原因：該備份建立於 GDI 修補套用之前
解法	備份命名加入說明詞：.bak_gdifont_日期_時間
	還原前先 grep 目標字串確認備份內容

[6] Harbour LOCAL 只能在函式頂部宣告

症狀	inline LOCAL 在 IF 或 FOR 等可執行語句之後 -> 編譯錯誤
	（Harbour 語言規範，不同於 C 的區塊宣告）
解法	所有 LOCAL 一律提前到函式最頂端宣告，
	函式 body 中只做賦值，不宣告

[7] GDI 字型硬編碼 MingLiU-ExtB	
症狀	客戶把 CJK 字型改為 PMingLiU 或標楷體後， Priority 3b 字形仍用細明體 ExtB， 與周圍 PDF 文字字型不一致
解法	加入 s_aCJKGDIMap 對照表 + GetGDIFontForCJK() 函式 動態查表回傳 GDI 字型名， 透過 MultiByteToWideChar 傳給 CreateFontW

5. 關鍵技術說明

5.1 GDI 離屏渲染（RenderUnicodeGlyph C 函式）

C 函式嵌入 #pragma BEGINDUMP / ENDDUMP 區塊，由 BCC32 編譯：

```
1. CreateCompatibleDC(NULL)      建立記憶體 DC（不需要視窗 DC）
2. CreateDIBSection(24bpp)       建立 top-down DIB，pBits 指向像素陣列
3. PatBlt(WHITENESS)             填白底
4. CreateFontW(wFont)            建立 TrueType 字型（4th param 動態傳入）
  失敗時 fallback CreateFontW(L"MingLiU-ExtB")
5. TextOutW(hMemDC, 0, 0, wStr, 1) 渲染單一 Unicode 字元
6. GdiFlush()                    確保渲染完成
7. 逐像素 BGR -> 灰階轉換
  g = (R*77 + G*150 + B*29) >> 8   (ITU-R 601 近似加權)
8. hb_retclen(pGray, nW*nH)      回傳 Harbour 字串（二進位灰階）
```

5.2 MultiByteToWideChar 動態字型名稱

Harbour 傳入的 cGDIFont 是 ANSI 字串（hb_parc(4)），需用 MultiByteToWideChar(CP_ACP) 轉成 WCHAR 才能傳給 CreateFontW：

```
pFontA = hb_parc( 4 );
if( pFontA && *pFontA )
  MultiByteToWideChar( CP_ACP, 0, pFontA, -1, wFont, 128 );
else
  MultiByteToWideChar( CP_ACP, 0, "MingLiU-ExtB", -1, wFont, 128 );
hFont = CreateFontW( nH, 0, 0, 0, FW_NORMAL, 0, 0, 0,
                DEFAULT_CHARSET, OUT_TT_PRECIS, CLIP_DEFAULT_PRECIS,
                ANTIALIASED_QUALITY, DEFAULT_PITCH | FF_DONTCARE,
                wFont );
```

5.3 BIG5-safe Python patch 腳本規範

所有 prn_to_pdf.prg 修改透過 Python bytes-only 腳本執行，規範如下：

- 以 rb / wb 模式讀寫檔案，絕對不做 decode/encode 循環
- 中文內容先在 Python 中以 str.encode('big5') 取得位元組再拼接
- 換行一律明確寫 b'\r\n'（CRLF），避免 b';r\n' 誤字
- 每個 patch 都有 SKIP 判斷（先搜尋目標字串），保證冪等性
- patch 腳本最後做驗證輸出，確認關鍵字串存在

6. 相關檔案清單

檔案	說明
compat\prn_to_pdf.prg	主程式：DrawUsrGlyph / GetGDIFontForCJK / RenderUnicodeGlyph
build\USRFONT.UMAP	785x2 bytes LE，lookupIdx->Unicode（FEFE=U+5803, FEFD=U+5586）
build\USRFONT.BCACHE	預渲染灰階字形快取（123 entries，W=38 H=39）
build\fix_unicode_gdi.py	Priority 3b GDI 離屏渲染修補（Patch A+B）
build\fix_gdi_bcc32.py	BCC32 編譯錯誤修補（bOK flag、#pragma warn -8004）
build\fix_gdi_fontmap.py	字型對照表 s_aCJKGDIMap + GetGDIFontForCJK() 修補
build\fix_comments_big5.py	新增/修改函式加上 BIG5 中文註解（9 個 patch）
build\RPT20260422UNICODE.PRN	測試 PRN：含 UNICODE 正常字 + 造字區 + BIG5 造字

7. 結論

[OK] UNICODE 正常字（含 BIG5-HKSCS 專屬字）現已可正確顯示於 PDF

透過四層 Priority 渲染鏈，prn_to_pdf.prg 現在可以處理所有 FA-FE 槽位的字元：
造字（BCACHE/24M）、標準 UNICODE（UMAP+BIG5）、BIG5-HKSCS 專屬（GDI bitmap）。

字型一致性問題透過 s_aCJKGDIMap 對照表解決，Priority 3b GDI 渲染字型會隨客戶設定的 PDF CJK 字型（s_cCJKFontName）自動切換，不再硬編碼。

所有 PRG 修改皆透過 bytes-only Python patch 腳本進行，保持 BIG5 編碼安全，且每個 patch 具備冪等性（可重複執行不重複套用）。