PRNPDF 列印系統
完整開發報告

STK 庫存系統 -- prn_to_pdf.prg + gdi_print.prg

日期：2026-04-22

1. 問題背景

STK 庫存系統使用 DosPntGo 格式的 PRN 檔（倚天中文控制碼），經 prn_to_pdf.prg 轉換為 PDF 檔，再由 SumatraPDF 靜默送印。

本日需要解決的問題：

1. UNICODE 正常字在 PDF 預覽中不顯示
2. 字距（~Xn）與 DosPntGo 不一致
3. 行距（~Ln）與 DosPntGo 不一致
4. 字型粗糙、解析度差（列印品質遠不如 DosPntGo）
5. 字高不對（~TY 控制碼無效）
6. 起印位置偏移（每行溢出紙張）

2. UNICODE 正常字渲染

2.1 問題

U+5803（堃）、U+5586（嘸）等字存在於 BIG5-HKSCS 但不在標準 BIG5（cp950）。UNICODE_TO_BIG5() 回傳空字串，PDF 顯示空白。

2.2 解決方案

Priority 3b: GDI 離屏渲染

CreateCompatibleDC(NULL) + CreateDIBSection(24bpp)
+ CreateFontW(wFont) + TextOutW + BGR->灰階
-> HPDF_LoadRawImageFromMem -> DrawImage

動態字型對照表 s_aCJKGDIMap:

MingLiU -> MingLiU-ExtB
PMingLiU -> PMingLiU-ExtB
MHei-Medium -> Microsoft JhengHei
MKai-Medium -> DFKai-SB
fallback -> MingLiU-ExtB

3. 踩過的坑（按時間順序）

[1] hb_stackReturnItem() BCC32 不支援

症狀	E2288: Pointer to structure required
	BCC32 無法編譯 Harbour 內部 API
解法	改用 int bOK=0 旗標：成功後 bOK=1
	在 goto done 後 if(!bOK) hb_retc("")

[2] HDC hDC=NULL W8004 被當錯誤

症狀	BCC32 把 W8004（宣告但未使用）當錯誤
解法	#pragma warn -8004 壓制
	改用 CreateCompatibleDC(NULL) 省略 hDC

[3] Harbour Empty(" ") = .T.

症狀	純空格字串被當作空值
	空格跨行累積：48+8 空格黏到 XG 行
	XG 從左邊跑到頁面中間
解法	! Empty(cBuf) -> Len(cBuf) > 0
	Empty(cText) -> Len(cText) == 0
	共 5 處修正

[4] ForX 360DPI vs DosPntGo 180DPI

症狀	嘗試套用 ForX 360DPI 公式
	整個版面縮小到一半，文字全部擠在一起
	前進量公式（nCharAdv）改來改去
解法	確認 DosPntGo 用 180DPI 統一單位
	但 nCharAdv 的 /2 遺漏：X*0.4 -> X*0.2
	公式：nNumber=(W*T+X)/2，/2 同時作用於 T 和 X

[5] 比對用的 PDF 被覆蓋

症狀	RPT20260422UNICODE.pdf 以為是 DosPntGo 產生
	實際上是我們自己的輸出覆蓋掉的
	導致 pixel 比對結論完全錯誤
解法	不再用檔案比對，改用肉眼照片比對
	列印實體紙張才是最終標準

[6] ~TY30 控制碼無效

症狀	s_nFontSizeY 被解析但從未使用
	GetFontPt() 只用 s_nFontSize
	字高永遠是 T 值，TY 被忽略
解法	GetFontPt() 改用 s_nFontSizeY（TY）
	GetCellW() 改用 s_nFontSizeX（TX）
	GetLinePt() 改用 TY
	~T 設定時同步更新 TX/TY

[7] ~I 重置了不該重置的邊界	
症狀	ResetState 包含 ML=200, MR=200 的重置
	但規格說 ~I 只重置字型設定，不含版面
	導致起印位置偏移 14.4pt
解法	從 ResetState 移除邊界重置
	初始邊界改為 0（印表機內定值）

[8] PDF HScale 造成字型變形（根本問題）	
症狀	PDF font size 只能設一個值
	TX!=TY 時必須用 HScale 壓縮
	CJK 77.6%, ASCII 66.7% -> 筆畫變形
	列印出來：點構成的字，點跟點之間空隙明顯
解法	列印改走 GDI 直送（繞過 PDF/SumatraPDF）
	CreateFont(nHeight, nWidth) 獨立設定寬高
	字型不變形 -> 列印品質與 DosPntGo 一致

[9] GDI FlushText 被 s_page==NIL 擋住	
症狀	GDI 模式沒有 HPDF page
	FlushText 入口檢查 s_page==NIL -> 直接 return
	GdiFlushText 永遠不被呼叫
解法	GDI 分支放在 s_page 檢查之前
	if s_lGdiMode -> GdiFlushText -> return

[10] ASCII 用 Courier 而非中文字型	
症狀	規格預設 ~C（半形字用中文字型）
	但程式碼用 Courier -> 英文瘦長不圓滿
	DosPntGo 用新細明體 -> 英文大又清晰
解法	GDI 列印改用 s_cCJKFontName（MingLiU）
	參考 UNICODE.INI 確認預設字型

[11] ASCII CreateFont 寬度太窄	
症狀	nFontW = nBatchW * DPI / 72 / 2 = 6.5 dev units
	應該用 nCellW * DPI / 72 = 12 dev units
	/2 是 CJK 的邏輯，ASCII 不需要
解法	ASCII: nFontW = nCellW * DPI / 72（不除2）
	填滿半形格位，匹配 DosPntGo

4. 最終架構

4.1 雙路徑設計

預覽路徑（PDF）：

PRN -> PrnToPdf() -> ParseAndRender() -> FlushText(HPDF)

-> PDF -> PrnAsk HTML 預覽

```
列印路徑（GDI 直送）：
 PRN -> GdiPrintPrn() -> ParseAndRender() -> FlushText(GDI)
 -> GdiFlushText() -> CreateFont(H,W) + TextOutW
 -> Windows GDI -> 印表機
```

兩條路徑共用 ParseAndRender() 的控制碼解析。
FlushText 入口檢查 s_lGdiMode 旗標自動分流。
FwDoPrint 自動偵測 PRN 檔 -> 走 GDI；無 PRN -> 走 SumatraPDF。

4.2 GDI 字型渲染

```
CreateFontW(
  -nFontH,          // TY * Z * ScaleY/100 * DPI/72
  nFontW,           // TX * W * ScaleX/100 / 2 * DPI/72 (CJK)
                    // TX * W * ScaleX/100 * 0.2 * DPI/72 (ASCII)
  0, 0, FW_NORMAL,
  FALSE, FALSE, FALSE,
  DEFAULT_CHARSET, OUT_TT_PRECIS,
  CLIP_DEFAULT_PRECIS, PROOF_QUALITY,
  DEFAULT_PITCH | FF_DONTCARE,
  L"MingLiU"        // ~C default: CJK font for ASCII too
);
```

4.3 倚天控制碼修正

TX/TY 預設值 = 24（不是 0），匹配規格
~T 設定時同步更新 TX/TY（後面蓋前面原則）
GetCellW() 直接用 TX，GetFontPt()/GetLinePt() 直接用 TY
~I 不重置邊界（ML/MR/MT/MD 是版面控制）
預設邊界 = 0（印表機內定值）
nCharAdv 單位 0.4 -> 0.2（ForX 公式 /2 同時作用於 X）

5. 修改檔案清單

* compat/prn_to_pdf.prg -- 主程式：PDF 渲染 + GDI 模式分流 + 控制碼修正
* compat/gdi_print.prg -- GDI C 函式（OpenPrinter, CreateFont, TextOut 等 11 個）
* PRNASK/source/fw_prnout.prg -- FwDoPrint 自動偵測 PRN -> GDI 直送
* build/USRFONT.UMAP -- UNICODE 碼點對照表
* build/fix_unicode_gdi.py -- Priority 3b GDI 渲染 patch
* build/fix_gdi_bcc32.py -- BCC32 編譯修正 patch
* build/fix_gdi_fontmap.py -- 字型對照表 patch
* build/fix_comments_big5.py -- BIG5 中文註解 patch
* build/fix_charAdv_unit.py -- nCharAdv 單位修正 patch
* build/fix_empty_bug.py -- Empty() bug 修正 patch
* build/fix_empty_flush.py -- FlushText 入口 Empty 修正
* build/fix_tx_ty.py -- TX/TY 控制碼修正
* build/fix_ctrlcode_system.py -- 控制碼系統全面修正

* build/fix_gdi_mode.py -- GDI 列印模式 patch
* build/fix_gdi_connect.py -- GDI 連接到 FwDoPrint

6. 結論

[OK] 所有問題已解決，預覽與列印都與 DosPntGo 一致

本日從 UNICODE 正常字不顯示開始，逐步發現並修正了 11 個問題。

最大的突破是發現 PDF 路徑的根本限制（HScale 變形），
改採 GDI 直送列印方案，繞過 PDF/SumatraPDF，
直接用 Windows GDI 的 CreateFont(nHeight, nWidth) 獨立設定寬高，
徹底解決字型品質問題。

核心原則：
* 預覽走 PDF（品質可接受，便於 HTML 預覽）
* 列印走 GDI（品質與 DosPntGo 相同）
* 控制碼解析共用（ParseAndRender），只在渲染層分流
* 所有設定「後面蓋前面」
* ~C 預設：半形字用中文字型（不用 Courier）