DosPntGo PRN 引擎

使用慘痛經驗心得報告

給後人參考，願你少踩幾顆地雷

HBSTK 庫存系統開發組 · 2026-04-12

目錄

前言　引擎背景與文件說明

第一坑　初始化必須放在同一行

第二坑　頁尾渲染觸發換頁（引擎Bug）

第三坑　教學範例文字被引擎執行

第四坑　~[JM] 置中+格式變數同行→亂碼

第五坑　BIG5中沒有全形波浪號～

第六坑　簡體字在BIG5/ETen無字形

第七坑　相鄰背景色行互相重疊癌字

第八坑　BOX()/SEC()參數的隱藏規則

第九坑　~I後的字全變亂碼

第十坑　BCC不在PATH，hbmk2靜默敗

總結　黃金守則速查表

本文件由gen_report_pdf.py自動產生 · 2026-04-12 07:21:25

前 言

本 引擎是以 Harbour 32-bit + libharu（hbhpdf） 實作 的 PRN→PDF 轉換核心， 用途 是把 Clipper 5.2 時代 的點陣式 列 印 輸出 轉換 為 PDF 文 件。

PRN 檔 是純文字（BIG5/CP950 編 碼），以 ~ 波 浪 號為控制 碼前綴， 格式 看似 簡單，但有大 量隱藏地 雷。以 下 是 開發過 程 中 踩過 的每一 顆。

!! 文件稀少，錯誤訊息模糊，唯一可靠的除錯方式是直接讀引擎源碼

!! 遇到奇怪現象時，先查 prn_to_pdf.prg 源碼，不要鑽牛角尖

第一坑 初始化必須放在同一行

>> 症狀

第1 頁沒有 頁首、沒有 頁 碼， 第2 頁空 白，內容從 第3 頁才 開始出 現。

>> 根因

引 擎在 遇到 第一 個換 行符 號(\n) 時就 呼叫 NewPdfPage() 建立 第 1 頁。若此 時頁首 ~{UT;...} 或頁尾 ~{DT;...} 尚未定 義， 第 1 頁 就永遠沒有 頁首頁尾，無 論後 面怎麼 設定 都補不 回來 。

!! 錯誤寫法：邊距和頁首分開寫在不同行

```
~[MT;500];~[MD;800];     ( 第一 行)
(換 行)                  ← 這裡 已建立 第1 頁!
~{UT; 頁首;              ( 第二 行 ← 已經太 晚 ！)
```

>> 正確寫法：邊距 + 頁首 + 頁尾，合為同一行，同一個 P() 呼叫

```
~[MT;500];~[MD;800];~[ML;600];~[MR;500];~{UT;...};~{DT;...};
```

>> 教訓

鐵律： 初始化 設定 必 須全 部合為 同一 行。

Python 寫法：多 個字 串 串接 用 P(a + b + c)， 確保 只有一 個換 行。

MD（下邊 界） 建 議少 設800（約 20mm）， 可確保 頁尾有 足夠空 間。

第二坑　頁尾渲染觸發換頁（引擎 Bug）

>> 症狀

即使初始化正確，仍然出現：第 2 頁空白、第 3 頁才有內容。頁尾文字跳到了空白的第 2 頁，而不是第 1 頁底部。

>> 根因

頁尾渲染位置 s_nY = MilPt(MD) / 2，底部邊距檢查 s_nY < MilPt(MD)。

當 MD=500 時：

```
頁尾Y = MilPt(500)/2 = 18pt（在邊距36pt以下）
FlushText() 偵測到 Y < 邊距→ 呼叫NewPdfPage() → 建立空白第2頁!
渲染頁尾的過程中，意外地又建立了一個新空白頁。
```

!! 修法：FlushText() 加上 s_lInHdrFtr 守衛旗標

```
// 原本（有Bug）：
IF s_nPageNum > 0 .AND. s_nY < MilPt( s_nMB )

// 修正後：頁首/尾渲染時絕不觸發換頁
IF s_nPageNum > 0 .AND. s_nY < MilPt(s_nMB) .AND. ! s_lInHdrFtr
```

>> 教訓

MD 設小（< 500）時容易出現。

建議標準值：~[MD;800]；（約 20mm），可確保頁尾有足夠空間。

第三坑　教學範例文字被引擎執行

>> 症狀

程式 Crash：Access Violation 在 HPDF_PAGE_SETWIDTH。錯誤發生在換頁時完全看不出原因在哪。

>> 根因

在教學手冊的BOX() 裡寫了控制碼範例：

```
BOX(" 設定：~[?100; 高度; 寬度];  ...")
```

引擎看到 ~[?100; 高度; 寬度] 就實際執行「設定自訂紙張，把「高度」和「寬度」（BIG5中文）用 Val() 轉成 0，下次換頁呼叫HPDF_Page_SetWidth(page, 0) → Crash！

!! BOX()/SEC() 參數內，絕對不能含有任何 ~ 控制碼

>> 解法：在引擎加入 ~~ 跳脫機制（兩個波浪號輸出一個真實的 ~）

```
// prn_to_pdf.prg ParseAndRender() 中新增：
IF SubStr( cText, nPos+1, 1 ) == s_cCtrl
   cBuf += s_cCtrl   //  輸出一個真實的~字元
```

```
    nPos += 2           //  跳過兩 個~~
ENDIF
```

>> Python 產生 PRN 的寫法

```
def SH(t): return t.replace("~", "~~")  // Show： 顯示 用，不執行
P(BOX(SH("~[?100; 高度; 寬度];")))          // 正 確: 輸出 可見的~~
```

第四坑 ~[JM] 置中 + 格式變數同行 → 亂碼

>> 症狀

頁尾的頁碼或日期顯示亂碼，或位置嚴重偏移。只要含有格式變數的頁尾套用了置中對齊，必然出現。

>> 根因

~[JM] 置中對齊需要在渲染前計算整行文字總寬度，才能決定起始 X。但 ~[FPAGE0]、~[FDATESHORT] 等格式變數在渲染時才展開，引擎無法預先知道展開後的文字長度，計算出錯。

!! 錯誤寫法：置中行混用格式變數

```
~{DT;~[JM]; 日期：~[FDATESHORT];  第[FPAGE0]; 頁;
```

>> 正確寫法：靠左（預設），格式變數才能正常展開

```
~{DT;~[FDATESHORT];  第[FPAGE0]; 頁;
```

若要靠右：用 ~[JR] 而非 ~[JM]，格式變數展開後靠右排列，不影響寬度計算。

第五坑 BIG5 中沒有全形波浪號 ~

>> 症狀

想在 PDF 輸出 ～（全形波浪號 U+FF5E）當作 ~ 的視覺替代，結果全部顯示為 ？或全空白。

>> 根因

```
Python encode('big5') 不支援 U+FF5E（全形波浪號）
→ errors='replace' 換成 ?（0x3F）
```

嘗試改用 CP950：

```
CP950 將 U+FF5E 編碼為 \xa1\xe3，看似正確，
但 libharu 使用 ETen-B5-H 的 MingLiU 字型，
\xa1\xe3 在該字型無對應字形 → 顯示為空白。
```

>> 最終解法：用 ~~ 跳脫機制顯示真實的 ~，完全迴避字型問題

PRN 中寫 ~~（兩個 ASCII 波浪號），引擎輸出一個真實的 ~。雖然是半形，但完全避開字型/編碼問題，顯示絕對正確。

第六坑 簡體字在 BIG5/ETen 無字形

台灣 BIG5（ETen-B5-H）只收錄繁體中文字集。Python 源碼中若不小心打了簡體字，在 PDF 顯示為？或空白。

常見踩坑字：

内(U+5185 簡體)	內(U+5167 繁 體)	(裡 面)
处(U+5904 簡體)	處 (U+8655 繁 體)	(地方)
体(U+4F53 簡體)	體(U+9AD4 繁 體)	(身 體/ 整體)
为 (U+4E3A 簡體)	為 (U+70BA 繁 體)	(為了)

!! 所有 BIG5 輸出文字必須使用繁體中文，使用簡體輸入法的開發者須特別注意

第七坑　相鄰背景色行互相重疊遮字

>> 症狀

多 個 ~OBn 背景色連續行，每行底部 1~2pt 被下一行背景蓋住，上下文字相互遮掩，難以閱讀。

>> 根因（數學計算）

```
行距        = nPt + s_nLineSpace × 0.4
            = 8.8 + 2.4 = 11.2pt   (T22, 預設L6)
背景矩形高  = nPt + 4 = 12.8pt     (原始程式碼)
重疊量      = 12.8 - 11.2 = 1.6pt  ← 這就是遮字的原因
```

>> 修法：FlushText() 中的背景矩形高度改為 nPt+2，偏移改為 -1

```
// 原本：高度 nPt+4，底部偏移 -2
HPDF_Page_Rectangle( s_page, nDrawX, s_nY-2, nTW, nPt+4 )

// 修正後：高度 nPt+2，底部偏移 -1（與行距留 0.4pt 間隙）
HPDF_Page_Rectangle( s_page, nDrawX, s_nY-1, nTW, nPt+2 )
```

驗算：修正後高度 10.8pt < 行距 11.2pt，相鄰行之間留 0.4pt 間隙，不重疊。

注意：若使用 ~L0;（緊縮行距），行距等於字高，背景仍會緊密相接。

第八坑　BOX()/SEC() 參數的隱藏規則

BOX 和 SEC 函式本身包含 ~OB（背景色）控制碼，這些碼被引擎執行。傳入的參數字中若又包含 ~ 控制碼，引擎會在錯誤時機執行它們。

!! 常見誤用，後果嚴重：

```
× BOX("~PA4; = ~[?2];")  →  執行！紙張被切換為 A4
× BOX("~[R;50;50];")     →  執行！整體縮小 50%
× BOX("~{UT; 新標題")    →  執行！覆蓋掉整份文件頁首
× BOX("~[MD;0];")        →  執行！下邊界設為 0 →  頁尾消失
```

>> 正確寫法：BOX/SEC 參數一律用 SH() 轉換，~~ 只顯示不執行

```
OK  BOX(SH("~PA4; = ~[?2];"))
OK  BOX(SH("~[R;50;50];"))
OK  BOX(SH("~{UT; 新標題"))
OK  BOX(SH("~[MD;0];"))
```

第九坑 ~I 後的字全變亂碼

~I; 是「還原所有排版設定」控制碼。但若在 ~I 後面沒有加 ; 就接其他控制碼:

```
~IT24L21;   ← 錯誤: ParseCtrlBlock 只處理 I 這一個字元
  →  回傳 nPos+1，指向 I 的下一個字元
  → T24L21; 被當成普通文字印到頁面上！
```

>> 正確寫法：~I; 後面加分號，再接其他碼

```
~I;~T24;~L6;   ← 正確: ~I; 結束，~T24; 開始新命令
```

>> 教訓

所有控制碼後面都要加 ; 結尾，不要省略。

連寫多個控制碼時: ~09;~S2;~T22; ← 每個都要; 結尾。

第十坑 BCC 不在 PATH，hbmk2 靜默失敗

hbmk2 編譯PRG 成功，但 C 編譯階段回傳 -1 且無具體錯誤訊息:

```
hbmk2[prnpdf]: Error: Running C/C++ compiler. -1
```

根因: hbmk2 呼叫bcc32.exe 時使用命令名稱非完整路徑，若 E:\BCC7732\bin 不在 PATH 環境變數中，找不到 bcc32.exe，hbmk2 只說「無法執行編譯器」，不說找不到檔案。

>> 解法：每次編譯前先設 PATH

```
set PATH=%PATH%;E:\BCC7732\bin
//  或在 go32.bat / Stk.ini 中永久設定
```

總結　黃金守則速查表

#	守則	違反後果
1	初始化(邊距 頁首 頁尾) 同一行	第一頁無 頁首頁碼
2	MD 至少 設800	頁尾觸發換 頁,空 白頁
3	BOX/SEC 參 數不 含~ ， 用SH()	Crash 或執行錯 誤控制 碼
4	~[JM] 行不 混格式 變 數(FPAGE/FDATE)	頁碼日期 亂碼
5	所有文字 用繁 體中文 (不 用 簡體)	字元 顯示 ？
6	顯示 ~ 字元 用~~ 跳脫	控制 碼被執行 或字元 消失
7	連續背 景色行: 矩形 高度不超過行 距	相鄰行背 景重疊您字
8	~I; 後面必 須加 ; 再接其 他碼	後續字元 當文字 印出
9	BCC 加入 PATH 後再執行hbmk2	靜默 敗,不知原 因
10	每 個控制 碼都要加 ; 結 尾,不 要省 略	下一 個控制 碼當文字 輸出

後 記

這份引擎從 Clipper 5.2 時代移植過來，諸 行為是為了 相容舊版點陣印表機 的控制 碼而 設計的文 件稀少，錯 誤訊息 模糊，唯一 可靠的除錯方式 是直接 讀prn_to_pdf.prg 的原 碼。

每一坑 都是真 實踩過 的，每一 個修法 都經過 驗證。希望 這份報告能讓後人 少踩幾 顆地雷，早點 把工作做完，早點回家。

本文件由gen_report_pdf.py 自動產生 · 2026-04-12 07:21:25