GTWVT CJK 字元顯示錯誤修正技術報告

BIG5 中文字半寬顯示問題的診斷與修復

專案：Clipper → Harbour 32-bit Console 轉換 (STK 庫存管理系統)

終端驅動：GTWVT (Harbour Windows Virtual Terminal)

撰寫日期：2026年4月10日

關鍵修正檔案：`E:\ConvertFW\HBSTK\gtwvt_patch\gtwvt.c`

目錄

1. 問題描述
2. 系統架構背景
3. 初步假設與排除
4. 偵錯過程：加入 Debug Log
5. 日誌分析：找到根本原因
6. 根本原因說明
7. 修正方案
8. 修正效果對照
9. 建置流程
10. 來龍去脈完整敘事
11. 技術摘要

一、問題描述

在將 Clipper 5.2 庫存管理系統轉換為 Harbour 32-bit console 程式後，以 **GTWVT 終端驅動**執行時，BIG5 中文字（華康楷書 W5A 字型）顯示異常：

> **症狀：**每個中文字後面有一個多餘的空格，整排文字被「撐開」，看起來像每個字都是半寬字，旁邊帶一個空格。
>
> 例如：「請輸入使用」應顯示為連續字串，卻顯示為「請 輸 入 使 用」。

相比之下，**造字（PUA User-Defined Characters）**顯示完全正常，字寬與排列都正確。這個對比成為後來找到根本原因的關鍵線索。

作業系統	Windows 10 繁體中文版
終端驅動	GTWVT (Harbour contrib)
字型	HuaKang 楷書 W5A（HKKaiW5A.TTF）
字碼頁	BIG5 (HB_CPTERM=BIG5, HB_CODEPAGE=BIG5)
視窗尺寸	80 欄 × 25 列，視窗寬度 1520px
字型高度	39px
計算格寬	1520 / 80 = 19px（半格寬）

二、系統架構背景

理解這個問題需要先了解 GTWVT 的字型管理機制：

```
GTWVT 字型建立流程：

  hb_gt_wvt_FitSize()
  ├─ 計算 fontWidth = maxWidth / COLS = 1520 / 80 = 19
  ├─ 設定 PTEXTSIZE.x = 19  (半格寬 = ASCII 字元寬度)
  └─ 呼叫 hb_gt_wvt_GetFont( lfHeight=39, iWidth=19 )
       └─ LOGFONT.lfWidth = iWidth = 19   ← 【BUG 在這裡】
            └─ CreateFontIndirect()
                 └─ GDI 強制所有字元平均寬度 = 19px
                      ├─ ASCII 'A' → 約 10px（正常，<19px）
```

```
                    └─ CJK '中' → 被壓縮成 19px（本應 39px！）

PaintText CJK 字元處理：
  ├─ 計算 FixedSize[i] = charCells × PTEXTSIZE.x = 2 × 19 = 38px
  ├─ FixedSize[i+1] = 0  (trail byte 佔位)
  └─ ExtTextOut(..., FixedSize[])
       ├─ 位置 i    ：保留 38px 空間
       ├─ 實際字形  ：只有 19px 寬（被 GDI 壓縮了！）
       └─ 結果      ：字形 19px + 空白 19px = 看起來像半寬字！
```

```
造字（ScanAndDrawUda）獨立繪製流程：

  ScanAndDrawUda()
  ├─ 從 BMP 讀取字形點陣圖
  ├─ GlyphRect_w = s_nCellW × 2 = 19 × 2 = 38px
  └─ StretchBlt( ..., 38px, 39px )   ← 直接操作像素，完全不受 lfWidth 影響！
       └─ 正確繪製 38px 寬的字形

結論：造字正常 ← 因為繞過 GDI 字型系統
      華康字異常 ← 因為被 GDI lfWidth=19 壓縮
```

關鍵資料結構

名稱	說明	正確值	Bug 時的值
`PTEXTSIZE.x`	半格寬（ASCII 字元寬度 / 半個 CJK 格）	19px	19px（正確）
`LOGFONT.lfWidth`	GDI 字型強制平均字寬	0（自然寬度）	19px（錯誤！）
CJK 字形實際寬度	GetTextExtentPoint32W 量測	39px	19px（被壓縮）
`FixedSize[CJK]`	ExtTextOut 的 CJK 字元步進寬度	38px（2×19）	38px（正確）
最終視覺效果	字形+空白	38px 全填滿	19px 字形 + 19px 空白

三、初步假設與排除

在找到真正原因之前，先後排除了以下假設：

假設 A：FixedSize[] 計算錯誤

懷疑 `PaintText()` 對 CJK 字元的步進寬度計算有誤，設成了 1×PTEXTSIZE.x 而非 2×PTEXTSIZE.x。
排除：加入 Debug 後確認 FixedSize=38（正確的 2×19）。

假設 B：PTEXTSIZE.x 設定錯誤

懷疑 `GTWVT_SETCELLWIDTH` 沒有正確設定 PTEXTSIZE.x，導致後續計算偏差。
排除：SETCELLWIDTH 在 UsrFontInit 時期呼叫，彼時視窗尚未建立（hWnd==NULL），所以必然失敗。但 FitSize 在視窗建立後才執行，並正確設定 PTEXTSIZE.x=19。

假設 C：isWideChar() 判斷不正確

懷疑 CJK 字元的雙寬判定邏輯有誤，把某些字誤判為單寬。
排除：造字（PUA E000-F8FF）和普通 CJK 都在 isWideChar 的判斷範圍內，且造字正常，所以問題不在 isWideChar。

假設 D：ScanAndDrawUda 與 PaintText 的 nCellW 不同步

懷疑造字繪製用的 nCellW 與 PTEXTSIZE.x 數值不一致，導致造字看起來正常但實際上座標錯誤。
排除：加入 s_nCellW 同步機制後，確認兩者都是 19px，且造字確實正常。

真正原因：LOGFONT.lfWidth 壓縮字形

GetFont Debug 直接量測 CJK 字形寬度：
`[GetFont#4] lfW=19 CJK_w=19` vs `[GetFont#2] lfW=0 CJK_w=39`
確認 lfWidth=19 時 GDI 將 CJK 字壓縮成 19px，lfWidth=0 時自然寬度為 39px。

四、偵錯過程：加入 Debug Log

4.1 Debug 基礎設施

在 `gtwvt.c` 中加入統一的日誌函式，寫入檔案同時輸出 OutputDebugString：

```
/* 字型偵錯日誌：同時寫入 font_debug.log 及 OutputDebugString */
static void hb_gt_wvt_FontDebugLog( const char * szMsg )
{
    FILE * fp = fopen( "font_debug.log", "a" );
    if( fp )
    {
        fprintf( fp, "%s\n", szMsg );
        fclose( fp );
    }
    OutputDebugStringA( szMsg );
}
```

4.2 GetFont 量測點

在 `hb_gt_wvt_GetFont()` 建立字型後，立即量測 CJK 和 ASCII 字元的實際像素寬度：

```
static int s_nGetFontLog = 0;
HFONT hRetFont = CreateFontIndirect( &logfont );

if( hRetFont && s_nGetFontLog < 10 )
{
    HDC   hdcT  = GetDC( NULL );
    SIZE  szCJK, szAsc;
    WCHAR wcCJK = 0x4E2D;  /* 中 */
    WCHAR wcAsc = 'A';
    HFONT hOldT = SelectObject( hdcT, hRetFont );

    GetTextExtentPoint32W( hdcT, &wcCJK, 1, &szCJK );
    GetTextExtentPoint32W( hdcT, &wcAsc, 1, &szAsc );

    SelectObject( hdcT, hOldT );
    ReleaseDC( NULL, hdcT );

    int nCJKHalf = szCJK.cx / 2;
    sprintf( szDbg,
        "[GetFont#%d] lfH=%d lfW=%d CJK_w=%d Asc_w=%d => CJK_half=%d"
        " (PTEXTSIZE.x will be Asc or CJK/2)",
        s_nGetFontLog, logfont.lfHeight, logfont.lfWidth,
        szCJK.cx, szAsc.cx, nCJKHalf );
    hb_gt_wvt_FontDebugLog( szDbg );
    s_nGetFontLog++;
}
```

4.3 PaintText CJK 量測點

在 `PaintText()` 中記錄前 5 個 CJK 字元的 FixedSize 分配：

```
static int s_nPTDbg = 0;
if( charCells == 2 )
{
    FixedSize[ i ] = charCells * PTEXTSIZE.x;  /* = 38px */
    FixedSize[ i + 1 ] = 0;

    if( s_nPTDbg < 5 )
    {
        sprintf( szDbg,
            "[PaintText] CJK U+%04X FixedSize=%d PTEXTSIZE.x=%d"
            " (glyph compressed? check GetFont log)",
            wc, FixedSize[ i ], PTEXTSIZE.x );
        hb_gt_wvt_FontDebugLog( szDbg );
        s_nPTDbg++;
    }
}
```

五、日誌分析：找到根本原因

執行程式後取得的 `font_debug.log`（關鍵段落）：

```
[GetFont#0] lfH=39 lfW=10 CJK_w=10 Asc_w=5   => CJK_half=5
[GetFont#1] lfH=39 lfW=19 CJK_w=19 Asc_w=10  => CJK_half=9
[USRFONT_MEASURE] fontH=39 CJK_w=39 CJK_half=19 Asc_w=20 nResult=19
[GetFont#2] lfH=39 lfW=0  CJK_w=39 Asc_w=20  => CJK_half=19  ← lfWidth=0 正常！
[SETCELLWIDTH] GetFont FAILED nW=19
[GetFont#3] lfH=39 lfW=0  CJK_w=39 Asc_w=20  => CJK_half=19
[GetFont#4] lfH=39 lfW=19 CJK_w=19 Asc_w=10  => CJK_half=9   ← lfWidth=19 被壓縮！
[FitSize] fontWidth_input=19 PTEXTSIZE.x=19 COLS=80 maxWidth=1520
[FitSize_Actual] CJK_w=19 Asc_w=10 PTEXTSIZE.x=19 FixedSize4CJK(2x)=38 CJK_half=9
[GetFont#5] lfH=39 lfW=19 CJK_w=19 Asc_w=10  => CJK_half=9
[ScanAndDrawUda] nCellW=19 nCellH=39 => GlyphRect_w=38
[PaintText] CJK U+8ACB FixedSize=38 PTEXTSIZE.x=19
[PaintText] CJK U+8F38 FixedSize=38 PTEXTSIZE.x=19
```

關鍵對比如下：

呼叫場合	lfWidth 傳入	CJK 實際量測寬度	結果
USRFONT_MEASUREFONT（量測用）	0	39px	✅ 正確

GetFont#2（SETCELLWIDTH）	0	39px	✅ 正確
GetFont#4（FitSize 呼叫）	**19**	**19px**	**❌ 被壓縮！**
GetFont#5（正式使用字型）	19	19px	❌ 被壓縮！

診斷結論：
`FitSize()` 計算出 `fontWidth = maxWidth / COLS = 1520 / 80 = 19`，並將此值作為 `iWidth=19` 傳給 `GetFont()`。
GetFont 將 `logfont.lfWidth = 19`，GDI 強制所有字元平均寬度為 19px，CJK 字（自然寬度 39px）被壓縮成 19px。

PaintText 為 CJK 保留了 38px 空間（FixedSize=38），但字形只有 19px，剩餘 19px 成為空白。

六、根本原因說明

LOGFONT.lfWidth 的語意

Windows API 文件對 `LOGFONT.lfWidth` 的說明：

> *"lfWidth: The average width, in logical units, of characters in the font. If lfWidth is zero, the aspect ratio of the device is matched against the digitization aspect ratio of the available fonts to find the closest match, determined by the absolute value of the difference."*

重要：當 lfWidth > 0 時，GDI 會**水平縮放**整個字型以達到指定的平均字元寬度。 這對等寬的 ASCII 字型很合理，但對 BIG5 雙位元組字型（CJK 字寬 ≈ 2× ASCII 字寬）會造成嚴重問題：

- GDI 目標平均寬度 = 19px（fontWidth）
- BIG5 字型實際平均寬度受 CJK 字拉高（因為 CJK 字占大多數）
- GDI 縮放整個字型使平均接近 19px → CJK 字從 39px 縮成約 19px
- ASCII 字對應縮成約 10px

設計問題：`FitSize()` 的 fontWidth 計算的是「每個 console cell 的寬度」，即 ASCII 字元的寬度（半格）。BIG5/GTWVT 架構下，一個 CJK 字佔用 2 個 cell，所以 CJK 字形的自然寬度應該是 2×cellWidth = 38px，而不是 cellWidth = 19px。

`lfWidth` 不能用來限制每個 cell 的寬度——它是字型整體的水平縮放，不可能讓 CJK 字和 ASCII 字同時都適合。**正確做法是 lfWidth=0，讓 GDI 依字型高度自然決定字寬，再靠 `ExtTextOut` 的 `FixedSize[]` 陣列控制**

每個字元的步進寬度。

七、修正方案

修正位置：`E:\ConvertFW\HBSTK\gtwvt_patch\gtwvt.c`，函式 `hb_gt_wvt_GetFont()`

修正前（錯誤）

```
/* 原始碼 — lfWidth 使用傳入的 iWidth，導致 GDI 壓縮 CJK 字形 */
logfont.lfWidth = iWidth <= 0 ? 0 : iWidth;
```

修正後（正確）

```
/* CJK fix: 永遠使用 lfWidth=0，讓 GDI 依字型高度自然決定字寬。
 * 不可用 iWidth>0 強制壓縮，否則 BIG5 CJK 字（自然寬度 39px）
 * 被壓縮成 19px，與 ExtTextOut FixedSize 的 38px 不符，
 * 導致每個 CJK 字後出現 19px 空白（看起來像半寬字）。
 * 步進寬度的控制完全由 PaintText 的 FixedSize[] 陣列負責。     */
logfont.lfWidth = 0;
```

修改量：1 行（將賦值從 `iWidth <= 0 ? 0 : iWidth` 改為常數 `0`）

> **為何這樣改是安全的？**
>
> `lfWidth=0` 是 Win32 LOGFONT 的預設安全值，表示「依裝置長寬比自動選擇」。步進寬度（字元間距）完全由 `ExtTextOut` 的第 9 個參數 `lpDx[]`（即 FixedSize[]）控制，與 lfWidth 無關。因此，lfWidth=0 不會破壞原有的等寬排列邏輯，只是讓 CJK 字形能以自然寬度渲染。

八、修正效果對照

項目	修正前（lfWidth=19）	修正後（lfWidth=0）
CJK 字形實際寬度	19px（被壓縮）	39px（自然寬度）
ASCII 字形實際寬度	10px	20px

FixedSize[CJK]	38px（步進正確，但字形不足）	38px（步進=字形，完全填滿）
CJK 字後空白	19px（38-19=19px 空白）	0px（無空白）
視覺效果	王 先 生 白 小 姐（字與字之間有空隙）	王先生白小姐（緊密正常）
造字（PUA）顯示	正常（ScanAndDrawUda 不受影響）	正常（同前）
ASCII 顯示	正常（FixedSize=PTEXTSIZE.x=19）	正常（同前）

九、建置流程

GTWVT 是 Harbour 的 C 語言模組，修改後需要二步驟重建：

步驟 1：重新編譯 gtwvt.c 並安裝 lib

```
REM E:\ConvertFW\HBSTK\gtwvt_patch\build_now.bat
bcc32 -c -6 -O2 -tW -DWIN32 -DWINVER=0x0500 ^
      -I"E:\Harbour32\include" ^
      -I"E:\bcc7732\include" ^
      -I"E:\bcc7732\include\windows\sdk" ^
      gtwvt.c

tlib gtwvt.lib +gtwvt.obj /P64

copy /Y gtwvt.lib "E:\Harbour32\lib\win\bcc\gtwvt.lib"
echo === gtwvt.lib updated ===
```

步驟 2：重新連結 Stk.exe

```
REM E:\ConvertFW\HBSTK\build\rebuild_now.bat
set HB_USER_CFLAGS=-I"E:\bcc7732\include\windows\sdk"
hbmk2 stk32.hbp -comp=bcc -rebuild 1> log\out2.log 2> log\err2.log
echo === Stk.exe rebuild done ===
```

步驟 3：執行

```
REM 必須使用 RUN.BAT，不可直接執行 Stk.exe
REM 原因：Stk.exe 依賴 loc_dbf、loc_sys 等環境變數
REM 直接執行會導致 \Spool.dbf (DOS Error 2) 錯誤
call RUN.BAT
```

> **常見陷阱：**若修改 gtwvt.c 後直接執行 Stk.exe，效果不會改變。 必須先執行 `build_now.bat` 更新 gtwvt.lib， 再執行 `rebuild_now.bat` 重新連結，才能看到修正效果。

十、來龍去脈完整敘事

10.1 問題的開始

STK 庫存管理系統原本是 1990 年代的 Clipper 5.2 DOS 程式，經過多年使用後， 決定將其移植到 Harbour（Clipper 的開源繼承者），在 Windows 上以 GTWVT 終端模擬器執行， 保留所有 DOS 時代的 @ SAY/GET、TBrowse 等 UI，不改變任何業務邏輯。

移植工作完成大半後，測試時發現一個令人困惑的問題： 畫面上的中文字每個都有一個多餘的空格，「請輸入使用者密碼」看起來變成「請 輸 入 使 用 者 密 碼」。 但奇怪的是，程式中使用的「造字」（公司自訂的 PUA 點陣圖漢字）卻完全正常顯示。

10.2 造字正常、普通中文不正常——這個對比是關鍵

這個現象很奇特。造字和普通中文字都是「雙寬字元」，都應該佔用 2 個 console cell。 如果問題是 PTEXTSIZE.x 設定錯誤，兩者應該都異常； 如果問題是 isWideChar() 判斷錯誤，也應該兩者都受影響。

為何造字正常？翻查 gtwvt.c 的原始碼，發現一個關鍵差異：

- **普通中文字**：由 `PaintText()` 透過 `ExtTextOut()` 使用 GDI 字型渲染
- **造字**：由 `ScanAndDrawUda()` 讀取 BMP 點陣圖，用 `StretchBlt()` 直接貼圖

造字完全繞過了 GDI 字型系統！如果問題出在字型建立階段（例如 LOGFONT 參數錯誤）， 只會影響 GDI 渲染的普通中文字，不會影響造字。

10.3 加入 Debug Log 的過程

在字型相關的每個關鍵位置都加入了 Debug Log： `GetFont()`、`FitSize()`、`PaintText()`、`USRFONT_MEASUREFONT()`、`ScanAndDrawUda()`。

但是第一次加完 Debug、重建 Stk.exe 後，`font_debug.log` 卻沒有出現。 排查後發現：`build_now.bat`（編譯 gtwvt.c）和 `rebuild_now.bat`（連結 Stk.exe）是兩個獨立步驟。第一次只執行了 rebuild_now.bat，gtwvt.lib 仍然是舊版本（時間戳差了 1.5 小時），所以 Debug 程式碼根本沒被編譯進去。

執行 build_now.bat 後重建，log 檔終於出現。

10.4 Log 讀出的訊息

日誌中最關鍵的兩行：

```
[GetFont#2] lfH=39 lfW=0  CJK_w=39 Asc_w=20  => CJK_half=19
[GetFont#4] lfH=39 lfW=19 CJK_w=19 Asc_w=10  => CJK_half=9
```

同一個字型名稱、同樣的 lfHeight=39，唯一的差別是 `lfWidth`：

- `lfWidth=0`：CJK 字形 39px（正確的自然寬度）
- `lfWidth=19`：CJK 字形 19px（被 GDI 水平壓縮了！）

同時看到 `FitSize_Actual` 的日誌：`CJK_w=19 PTEXTSIZE.x=19 FixedSize4CJK(2x)=38 CJK_half=9` —— FixedSize 是 38px（正確），但字形只有 19px，兩者不匹配，空出 19px 的空白。

謎題解開了。

10.5 修正與驗證

找到原因後，修正只需一行：將 `logfont.lfWidth = iWidth <= 0 ? 0 : iWidth` 改為 `logfont.lfWidth = 0`。

依序執行 build_now.bat → rebuild_now.bat → RUN.BAT， 畫面上「王先生」、「白小姐」等中文字緊密正常地顯示，問題解決。

十一、技術摘要

項目	內容
問題類型	Win32 GDI 字型水平壓縮導致 CJK 字形半寬
根本原因	`LOGFONT.lfWidth = 19`（cellWidth）傳入 GDI，強制所有字元平均寬度 = 19px，CJK 字被壓縮

修正	`logfont.lfWidth = 0`（一行修改）
修正原理	lfWidth=0 讓 GDI 以字型自然寬度渲染，步進由 ExtTextOut FixedSize[] 控制，兩者相符無空白
偵錯關鍵線索	造字（PUA BMP）顯示正常 → 問題在 GDI 字型層，不在 FixedSize/isWideChar 邏輯
修正檔案	`E:\ConvertFW\HBSTK\gtwvt_patch\gtwvt.c`，函式 `hb_gt_wvt_GetFont()`
影響範圍	僅影響 BIG5/CJK 雙寬字型；純 ASCII 字型因為字元都是單寬，lfWidth=cellWidth 是正確的

✅ **問題已完全解決。**
BIG5 中文字（華康楷書 W5A）在 GTWVT 終端現在正常顯示為全寬字元， 造字（PUA）亦維持正常。一行修改，根除 Win32 GDI 字型壓縮問題。