

USRFONT 徹底重構計畫手冊

STK 庫存系統 — Harbour GTWVT Console 版

日期	2026-05-12
撰寫	Claude + 開發者協同產出
版本	1.0（出差前定稿）
用途	出差回來後依此手冊逐步執行
核心問題	狀況 1 work、狀況 2 fail —— 為什麼，怎麼修
設計理念	USRFONT 自帶向量字、自我成長、零 OS 依賴

目錄

章節	內容
1	問題現象
2	為什麼會有 USRFONT 系統（設計初衷）
3	系統現有的 FontDirect 四種模式
4	錯誤探索歷程（七波失敗）
5	關鍵洞察：FD=2 在狀況 2 也 fail
6	真因鎖定：USRFONT_GETBIG5 Step 2 依賴 CP950
7	徹底重構計畫
8	逐步執行清單（A-F 階段）
9	附錄：相關檔案／指令

1. 問題現象

兩台機器同樣執行 W:\S002\Stk32.exe，行為不同：

機器	狀態	結果
狀況 1	曾裝華康字型 → 解除 → 重開機	所有造字（鍵盤圖示等）正常顯示
狀況 2	從未裝過華康字型	大部分造字抓不到，並且顯示奇怪的中文字

關鍵觀察：兩台 EXE、TTF、UMAP

完全相同，差異只能在「機器本身狀態」。

2. 為什麼會有 USRFONT 系統（設計初衷）

從 DOS Clipper 時代演進到 Harbour Windows 時代，USRFONT

系統承擔的四大用途：

用途	說明	舉例
1. 承接倚天舊資	DOS Clipper 時代的 DBF 內含倚天系統造字A40A碼，指向某處，以備能繼續顯示	
2. 使用者造字	Big5 沒有的字（如「堃」），使用者用造字程式畫出，放在 USRFONT.24M字檔	
3. 鍵盤按鍵圖示	系統內所有特殊符號（Esc / F1-F12 / INS / DEL / ... / USFontRegBig5RegDel等）取得	
4. IME 輸入 + 使用者透過 Windows 內碼輸入法	（輸入字彙輸入也能既連型，沒有的即取code，	

核心精神

- 自帶：所有字型、對映表都打包在 EXE 內或 deploy 檔內
- 自我成長：使用者輸入新字 → DynAllocSlot 動態配位 → 寫入 UMAP → 下次啟動載入
- 向量字：USRFONT.TTF 提供向量 glyph，跨解析度、可列印
- 零 OS 依賴：不靠華康、不靠 Windows EUDC、不靠任何外部字型

3. 系統現有的 FontDirect 四種模式

位置：System.ini 內 FontDirect= 設定。

FD	名稱	字型來源	OS 依賴	螢幕品質	列印品質	跨解析度
0	auto	偵測華康：有→GD高；(無華康或BCACHE)視情況	高	高	高	中
1	force font	強制用 OS 華康字型	高 (必裝華康)	高	高	高
2	force BCACHE	預烘焙 alpha bitmap	只對固定解析度 (點陣放大)	只對固定解析度 (點陣放大)	只對固定解析度 (點陣放大)	只對固定解析度 (點陣放大)
3	embedded TTF	USRFONT.TTF (向墨字，自帶)	高	高	高	高

評估結論

- FD=1：違背自帶設計（必裝華康），實務上不可用
- FD=2：BCACHE 是點陣，列印與解析度切換時品質差
- FD=0：邏輯複雜，仍依賴 HuaKang detect 或 BCACHE
- FD=3：完全自帶向量字，符合所有設計初衷

但保守起見，不立即移除 0/1/2/3，先新增 FD=4 為新模式，運作成熟後再淘汰。

4. 錯誤探索歷程（七波失敗）

為了便於日後回顧避免重蹈覆轍，逐項記錄。

第一波：gtwvt.c hb_big5_wcharGet UDA 攔截

假設	推測：Big5 → Unicode 轉換在這條路徑出錯，加攔截搶在前面
結果	加 debug log 後發現該路徑完全沒被呼叫 → 修錯地方
教訓 / 處置	回退。學到：先驗證路徑會被走到再動。

第二波：CreateFontW CharSet 從 DEFAULT_CHARSET 改 CHINESEBIG5_CHARSET

假設	推測：嚴格 charset 能讓 OS 認得 FR_PRIVATE CJK 名稱字型
結果	狀況 1 也壞了！USRFONT.TTF 的 OS/2 沒宣告 BIG5 codepage，被嚴格 charsets 攔截
教訓 / 處置	回退到 DEFAULT_CHARSET。學到：「假設要對照成功與失敗兩端」——解法必須考慮兩端

第三波：USRFONT.TTF 改名為 ASCII (USRFONT-STK)

假設	推測：CJK face name 可能讓 CreateFontW 比對失敗
結果	改名後 gi / 像素數完全不變 → 字型本來就找得到，問題不在 face name
教訓 / 處置	保留改名（無副作用）。學到：log 顯示 actual=requested 不代表真載入該字

第四波：USRFONT.TTF OS/2 advisory fields 復原

假設	推測：瘦身時清空了 ulUnicodeRange / ulCodePageRange，OS 視為 decorative
結果	從 24MB 原版復原後狀況 2 仍不正常
教訓 / 處置	保留復原（無副作用）。學到：OS/2 advisory 不影響 FR_PRIVATE 私有字型

第五波：HKCU\EUDC\950 registry

假設	推測：曾裝過華康在 HKCU 留下 EUDC.TTE 對映，狀況 2 沒此 registry
結果	寫 set_eudc_950.bat 部署 registry 到狀況 2，仍不正常
教訓 / 處置	registry 不是真因。學到：實測比推測重要。

第六波：日誌深度分析

假設	對比兩台 hex log 與像素數
結果	狀況 1 capture 14+ 個 PUA，狀況 2 只 capture 3 個 → 大部分 PUA 沒進登
教訓 / 處置	導向真因（USRFONT_GETBIG5 階段失敗）。

第七波：FD=2 在狀況 2 也 fail 的關鍵測試

假設	使用者測試 FD=2 BCACHE 模式（理論上不依賴字型）
結果	狀況 1 work、狀況 2 仍 fail → 證明 bug 在「字型渲染」之前的「Big5 對
教訓 / 處置	確認真因在 USRFONT_GETBIG5 ，不在任何 FD 渲染路徑。

5. 關鍵洞察：FD=2 在狀況 2 也 fail

FD=2 BCACHE 不依賴 OS 字型，理論上應該不受機器差異影響。但狀況 2 仍 fail。

含義

Bug 不在「字型渲染」這層（FD=1/2/3 渲染邏輯都正常），而在更上游的「Big5 Unicode 對映」這層。

資料流分析

PRG: KY_INS = UsrFontGetBig5(0xED61)

↓ Step 1 ■ s_aDynUnicode ■■■

↓ Step 2 WideCharToMultiByte(950) ■■ EDxx → ■■ 2 ■■ → ■ ""

KY_INS = "" ■■■■■■

↓

PRG ■■■■■■■■■■ Big5 UDA cell

↓

■■ buffer ■ PUA ■■ → ScanAndDrawUda ■■■■■■

↓

■■ FD=1/2/3 ■■■

所有 FD 模式都卡在 USRFONT_GETBIG5 Step 2 — 那是真正的瓶頸。

6. 真因鎖定：USRFONT_GETBIG5 依賴 CP950 反查

現有邏輯 (usrfont_api.c::USRFONT_GETBIG5)

Step 1: `s_aDynUnicode[] == wc` → `Big5`

Step 2: `WideCharToMultiByte(950, wc, ...)` → OS CP950

Step 3: → `""`

為什麼狀況 1 work、狀況 2 fail

機器	CP950 對 EDxx/ECxx 反查	Step 2 結果
狀況 1 (曾裝華康可能有：HKCU\EUDC\950 殘留、Traditional Chinese Big5 語言包、其他 CJK)		成功
狀況 2 (從未裝過這些 CJK 系統元件 → CP950 只支援標準 E000-E310 映射)		失敗

為什麼狀況 2 還是有 3 個 PUA work (ED66 / E010 / E00F)

- E010 / E00F：在 USRFONT.24M CMEX tag 範圍內 (cmexTag < 0xE315)，USRFONT_LOAD 時填入 s_aBig5Lookup，Step 1 命中。
- ED66：可能來自其他路徑 (PRG 直接寫 Big5 byte，wcharGet 走 CP950 反查並寫入 s_aDynUnicode；或 build 時 USRFONT.UMAP 已預存)

其他 EDxx/ECxx 大圖示因兩條路都不通，回 `""` → 底部訊息空缺 → 顯示其他 CJK 殘留

7. 徹底重構計畫

總原則

- 保留現有 FD=0/1/2/3 完全不動，新增 FD=4 為新模式
- FD=4 走「完全自帶 + 自我成長 + 向量字 + 零 OS 依賴」路徑
- FD=4 運作成熟（穩定數週／月）後，再逐一移除 FD=0/1/2/3

FD=4 設計重點

元件	FD=4 處理方式
USRFONT.TTF	使用嵌入的向量字（同 FD=3），不靠華康
USRFONT.UMAP	不再 ExtractOneRes 覆蓋（overwrite=0），讓 runtime 自我成長
USRFONT_GETBIG5	改為「Step 1 查 UMAP，找不到直接 DynAllocSlot」，不走 CP950
DynAllocSlot	從 FE 倒序找 FA-FE 空位，s_aDynUnicode[slot] = wc，SaveDynMap
預烘焙	建置時用 Python 把 CustomChar.ch 內所有 UsrFontGetBig5(0xFFFF)

8. 逐步執行清單（出差回來後做）

階段 A：清理先前無效修改

#	工作	檔案	說明
A1	回退 gtwvt.c UDA 攔截	gtwvt5patch\gtwvt.c	回退到第一波探索前的版本
A2	保留 USRFONT.TTF 改名	USRFONT\USRFONT.TTF	無副作用，且修了 nameID 6 殘骸
A3	保留 OS/2 fields 復原	build\USRFONT.TTF	無副作用
A4	清掉 set_eudc_950.bat	build\set_eudc_950.bat	不再需要

階段 B：新增 FontDirect=4 模式

#	工作	檔案
B1	新增 FD=4 case 處理：載入 USRFONT.TTF + ASCEFONT.TTF，改 gtwvt.c LoadFontByFontDirect	source/main.prg
B2	同步加 FD=4 分支到 main.prg	source/main.prg
B3	修改 USRFONT_GETBIG5：偵測 g_bFd4Mode，若為 1 則跳過 Step.2 CP950，直接走 Direct	c_code/usrfont_api.c::STK_EXTRACT_RE
B4	USRFONT.UMAP overwrite=1 → 0	c_code/usrfont_api.c::STK_EXTRACT_RE

階段 C：預烘焙 USRFONT.UMAP（強化首次啟動）

#	工作	檔案
C1	寫 Python 腳本：掃 source/*.PRG + compat61.BRG 內所有 USRFONT.UMAP，找出所有新建的 Font	source/main.prg
C2	腳本邏輯：每個 PUA codepoint 配 Big5 slow 碼上從 FE 倒序，避開 USRFONT.24M 佔用	source/main.prg
C3	產出 build\USRFONT.UMAP (785 × 2 = 1570 個 raw array)	source/main.prg
C4	把腳本整進 _rebuild_claude.bat (rebuild 前自動跑)	source/main.prg

階段 D：測試與驗證

#	驗證項目	預期結果
D1	System.ini 設 FontDirect=3 (舊路徑)	原本正常 1-跑仍正常 (不退化)
D2	System.ini 設 FontDirect=3, 狀況 2	原本壞 → 仍壞 (FD=3 不變)
D3	System.ini 設 FontDirect=4, 狀況 1	跑常顯示底部按鍵列圖示
D4	System.ini 設 FontDirect=4, 狀況 2	跑常顯示底部按鍵列圖示 真正的 fix
D5	兩台 USRFONT.UMAP 比對	預烘焙部分完全相同; runtime DynAllocSlot
D6	列印測試 (PDF 預覽 + GDI 列印)	所有 PUA 圖示列印清晰 (向量字)
D7	使用者輸入 'UECE1 等 IME 內碼字元	可輸入、顯示、列印 (既有功能不退化)

階段 E：移除 FD=2 BCACHE (FD=4 穩定後執行)

理由：BCACHE 點陣字、列印品質爛、跨解析度爛、檔案多邏輯雜。FD=1 保留 (萬一出狀況裝華康即可恢復)。

#	工作	檔案
E1	移除 main.prg 內 FD=2 分支; FD=0 無華康時加進 FD=3 路徑	main.prg
E2	移除 usrfont.prg LoadFontByFontDirect 內的 FDs2f60 (OTHERWISE) 分支	usrfont.prg
E3	移除 stk32.rc 內 BCACHE resource (idb201/8103230d-310)	stk32.rc
E4	刪除 build/bcache/ 整個目錄	build/bcache/
E5	移除 STK_EXTRACT_RES 內 USRFONT.BCACHECodeAssFonttBpACHE 解壓	usrfont.bcache
E6	移除 c_code 的 BCACHE 持久化邏輯 (LoadUserFontUmap_refSaveUserFontUmap)	c_code
E7	移除相關 BCACHE 變數 (s_szUfntUmapFilecodesusbfondicty)	usrfont.h
E8	移除 BCACHE 預烘焙腳本 (若有)	tools/ + _rebuild_claude.bat

階段 F：FD=4 改名回 FD=2 (語意接管)

FD=2 BCACHE 已淘汰，2 號位騰出。FD=4 是新時代的「自帶自我成長」，為簡化使用者認知，改名回 FD=2 接管該編號的語意。

#	工作
F1	main.prg / usrfont.prg：將所有 FD=4 分支改為 FD=2
F2	usrfont_api.c：g_bFd4Mode → g_bFd2Mode（或改用更語意的名稱如 g_bSelfGrowMode）
F3	System.ini 範例值更新為 FontDirect=2（預設）
F4	PDF 文件、UpdateList、Memory 同步更新「FD=2 新語意」
F5	最終狀態：FD=0（auto） / FD=1（force HuaKang，救援用） / FD=2（自帶自我成長，主體）

最終藍圖

階段	FontDirect 模式組合	說明
現在	0 / 1 / 2(BCACHE) / 3	原狀態
過渡 1	0 / 1 / 2(BCACHE) / 3 / 4(新)	新增 FD=4
過渡 2	0 / 1 / 3 / 4(新)	移除 FD=2 BCACHE
最終	0 / 1 / 2(新, 自帶自我成長) / 3 / 4	FD=4 改名接管 FD=2

9. 附錄

9.1 關鍵程式碼位置

函式 / 段落	位置	用途
USRFONT_GETBIG5	c_code/usrfont_api.c:800-844	PRG → Big5 (PUA→Big5 對映入口)
STK_EXTRACT_RES	c_code/usrfont_api.c:1038-1058	啟動時解壓 EXE 內嵌資源到磁碟
DynAllocSlot	c_code/usrfont_c.c:240-263	動態配 FA-FE slot
LoadDynMap / SaveDynMap	c_code/usrfont_c.c:80-128	USRFONT.UMAP 持久化讀寫
LoadFontByFontDirect	compat/usrfont.prg:342-380	FontDirect 模式分支
main.prg FontDirect 處理	source/main.prg:177-200	FontDirect 設定入口
CustomChar.ch	source/CustomChar.ch	所有 KY_xxx 鍵盤圖示 PUA codepoint
USRFONT.TTF resource	build/stk32.rc:35 (id=109)	嵌入 EXE 的向量字
USRFONT.UMAP resource	build/stk32.rc:7 (id=102)	嵌入 EXE 的對映表

9.2 必要備份指令

```
cp usrfont_api.c usrfont_api.c.bak_before_fd4_YYYYMMDD
cp main.prg main.prg.bak_before_fd4_YYYYMMDD
cp usrfont.prg usrfont.prg.bak_before_fd4_YYYYMMDD
cp build/USRFONT.UMAP build/USRFONT.UMAP.bak_before_prebake_YYYYMMDD
```

9.3 Memory 已存項目（避免重蹈覆轍）

Memory 條目	提醒
feedback_hypothesis_consistency	解法必須能同時解釋「為什麼 A work、B 壞」
feedback_createfontw_default_charset	USRFONT.TTF CreateFontW 必須用 DEFAULT_CHARSET

9.4 出差期間禁止動作

- 不要刪除 W:\S002 內任何檔案（log / TTF / UMAP / EXE 仍是診斷資料）
- 不要重新編譯（current build 是診斷版，含 hex log）

- 不要修改 `c_code/usrfont_*.c` (修改前需先讀本手冊)

—— 計畫手冊結束 ——

出差回來後，請從第 8 章「逐步執行清單」階段 A 開始。

如有疑問，逐章回顧 1-7 章背景脈絡。