GTWVT + USRFONT + ASCFONT 合併

細部實作計畫

2026-05-06

總覽

Phase	方案	內容	風險	工期
1	多 OBJ 合併	gtwvt.obj + usrfont.obj + threshold.obj → gtwvt.lib	低	30 分鐘
2A	Paint callback	gtwvt.c WM_PAINT 尾端加 paint callback	低	1 小時
2B	Char callback	gtwvt.c WM_CHAR/IME_CHAR 加 char callback	中	1 小時
2C	Timer callback	gtwvt.c 加 WM_TIMER case + timer callback	低	30 分鐘
2D	移除 subclassing	usrfont_wndproc.c 改用 callback，移除 SetWindowLongPtrW	中	2 小時
3	extern → pWVT	全域變數改為 pWVT 結構體成員	高	1 天

Phase 1：多 OBJ 合併進 gtwvt.lib

目標：usrfont_c.c 和 threshold_c.c 從 Stk32.exe 搬到 gtwvt.lib，stk32.hbp 不再列出這兩個 C 檔。

步驟 1.1：備份

```
cp E:\Harbour32\lib\win\bcc\gtwvt.lib
E:\Harbour32\lib\win\bcc\gtwvt.lib.bak before merge
cp E:\ConvertFW\HBSTK\gtwvt patch\build gtwvt.bat
E:\ConvertFW\HBSTK\gtwvt_patch\build_gtwvt.bat.bak
```

步驟 1.2：修改 build_gtwvt.bat

在現有 gtwvt.c 編譯之後，加入 usrfont_c.c 和 threshold_c.c 的編譯：

```
REM === □□□□□ gtwvt.c ===
bcc32 -c -6 -O2 -d -q -w-sig -w-aus ... ^
-ogtwvt.obj gtwvt.c
REM === □□□□□ usrfont c.c ===
bcc32 -c -6 -O2 -d -q -w-sig -w-aus ... ^
-I"..\c code" ^
-ousrfont.obj ..\c_code\usrfont_c.c
REM === □□□□□ threshold c.c ===
bcc32 -c -6 -O2 -d -q ... ^
-othreshold.obj ..\c_code\threshold_c.c
REM === □□□tlib □□□□ obj ===
tlib gtwvt.lib +gtwvt.obj +usrfont.obj +threshold.obj /P64
```

注意：usrfont_c.c 需要 -I"..\c_code" 才能找到 usrfont_int.h 和 #include 的子檔

注意：BCC ICE 風險 — 如果 usrfont_c.c 單獨編譯 ICE，改用 -O1 或加 dummy.c

步驟 1.3：修改 stk32.hbp

移除第 211-212 行（usrfont_c.c 和 threshold_c.c）：

```
# □□□□□□
# ..\c code\usrfont c.c
# ..\c_code\threshold_c.c
```

保留不動：

```
-gtwvt # □□□□□ gtwvt.lib
..\compat\usrfont.prg # □□□Harbour wrapper
..\c code\wintitle c.c # □□□□□□□
..\c_code\printer_c.c # □□□□□□□
```

步驟 1.4：編譯與安裝

```
cd E:\ConvertFW\HBSTK\gtwvt_patch
```

```
build gtwvt.bat # □□□□ obj → gtwvt.lib
copy gtwvt.lib E:\Harbour32\lib\win\bcc\gtwvt.lib
cd E:\ConvertFW\HBSTK\build
hbmk2 stk32.hbp -comp=bcc -rebuild
```

步驟 1.5：驗證

- [ ] Stk32.exe 正常啟動
- [ ] 造字（UDA）正確顯示
- [ ] ASCFONT 框線/符號正確顯示
- [ ] scrollbar 上下箭頭 CHR(20)/CHR(127) 正確
- [ ] hover 功能正常
- [ ] 滑鼠點擊正常
- [ ] 游標不閃爍（caret tail 修正仍有效）
- [ ] MessageBox 按鈕 hover 正常
- [ ] PickList scrollbar 拖動正常
- [ ] 主選單滑鼠自由選取正常

步驟 1.6：退回方案

```
copy E:\Harbour32\lib\win\bcc\gtwvt.lib.bak before merge
E:\Harbour32\lib\win\bcc\gtwvt.lib
# □□ stk32.hbp □□ usrfont c.c / threshold c.c
hbmk2 stk32.hbp -comp=bcc -rebuild
```

Phase 2A：Paint Callback

目標：gtwvt.c 在 WM_PAINT 處理完畢後呼叫 callback，取代 UsrFontWndProc 攔截 WM_PAINT。

步驟 2A.1：gtwvt.h 加型別宣告

```
/* Paint callback: gtwvt PaintText □□□□□ */
typedef void (*HB GT PAINT CALLBACK)( void );
/* □□/□□ paint callback */
extern void hb gt wvt SetPaintCallback(
HB_GT_PAINT_CALLBACK pFunc );
```

步驟 2A.2：gtwvt.c 加 static 變數和函式

```
/* □□□□□static □□ */
static HB GT PAINT CALLBACK
s pPaintCallback = NULL;
void hb gt wvt SetPaintCallback(
HB GT PAINT CALLBACK pFunc )
{
s pPaintCallback = pFunc;
}
```

步驟 2A.3：gtwvt.c WM_PAINT 加 callback 呼叫

位置：gtwvt.c 第 3624-3626 行

```
/* □□□ */
case WM PAINT:
if( GetUpdateRect( hWnd, NULL, FALSE ) )
hb gt wvt PaintText( pWVT );
return 0;
/* □□□ */
case WM PAINT:
if( GetUpdateRect( hWnd, NULL, FALSE ) )
{
hb gt wvt PaintText( pWVT );
if( s pPaintCallback )
s pPaintCallback();
}
return 0;
```

步驟 2A.4：usrfont 改用 callback 註冊

usrfont_api.c 的 USRFONT_HOOKWND 改為呼叫：

```
hb_gt_wvt_SetPaintCallback( ScanAndDrawUda );
```

usrfont_wndproc.c 的 WM_PAINT case 移除（改由 gtwvt 呼叫）。

Phase 2B：Char Callback（PUA 攔截）

目標：攔截 WM_CHAR/WM_IME_CHAR 中的 PUA 字元（E311-F8FF），分配 FA-FE slot。

步驟 2B.1：gtwvt.h 加型別

```
/* Char callback: WM CHAR/IME CHAR □□□□□ */
/* □□ TRUE = □□□□□□□□□□□ */
typedef BOOL (*HB GT CHAR CALLBACK)(
HWND hWnd, UINT msg,
WPARAM *pwParam );
extern void hb gt wvt SetCharCallback(
HB_GT_CHAR_CALLBACK pFunc );
```

步驟 2B.2：gtwvt.c WM_CHAR 加 callback

位置：gtwvt.c 第 3658-3675 行（WM_CHAR case）和第 3689 行（WM_IME_CHAR case）

```
case WM CHAR:
case WM SYSCHAR:
{
/* □□□char callback □□ PUA */
if( s pCharCallback )
{
if( s pCharCallback(hWnd, message,
&wParam;) )
; /* wParam □□□□ */
}
/* □□□□ */
if( (unsigned) wParam >= 128 )
hb gt wvt HandleWideChar(pWVT,
(HB WCHAR) wParam);
...
```

步驟 2B.3：usrfont 的 char callback

將 usrfont_wndproc.c 第 57-72 行的 WM_CHAR/WM_IME_CHAR 邏輯封裝為 callback：

```
static BOOL UsrFontCharCallback(
HWND hWnd, UINT msg, WPARAM *pwp )
{
WCHAR wc = (WCHAR) *pwp;
if( wc > 0xE310 && wc <= 0xF8FF )
{
```

```
int slot = DynAllocSlot( wc );
if( slot >= 0 )
*pwp = (WPARAM)(0xE000 + slot);
}
return FALSE; /* □□□□□□ */
}
```

注意：callback 回傳 FALSE 表示「已修改 wParam 但仍需 gtwvt 預設處理」

注意：回傳 TRUE 表示「完全攔截，gtwvt 不處理」

Phase 2C：Timer Callback

目標：gtwvt.c 處理 WM_TIMER，讓 usrfont 不需自己攔截 WM_TIMER。

步驟 2C.1：gtwvt.c 加 WM_TIMER case

目前 gtwvt.c 沒有 WM_TIMER case。加入：

```
case WM TIMER:
if( s pTimerCallback )
s pTimerCallback( hWnd, wParam );
return 0;
```

步驟 2C.2：usrfont timer callback

```
static void UsrFontTimerCallback(
HWND hWnd, UINT PTR idTimer )
{
if( idTimer == 9988 )
{
KillTimer( hWnd, 9988 );
InvalidateRect( hWnd, NULL, FALSE );
}
}
```

Phase 2D：移除 SetWindowLongPtrW

Phase 2A-2C 全部完成後，UsrFontWndProc 已無需攔截任何訊息。可以移除 subclassing。

步驟 2D.1：修改 USRFONT_HOOKWND

```
HB FUNC( USRFONT HOOKWND )
{
/* □□□□ EnumWindows + subclassing */
/* □□□□□□ callback */
hb gt wvt SetPaintCallback(
ScanAndDrawUda );
hb gt wvt SetCharCallback(
UsrFontCharCallback );
hb gt wvt SetTimerCallback(
UsrFontTimerCallback );
g bHookPaint = 1;
hb retl( HB TRUE );
}
```

步驟 2D.2：移除不再需要的程式碼

檔案	移除內容	說明
usrfont_wndproc.c	UsrFontWndProc 整個函式	改為三個 callback 函式
usrfont_wndproc.c	s_pOrigWndProc / s_bInsidePaint	不再需要 subclassing 狀態
usrfont_api.c	FindProcessWindowProc / EnumWindows	不再需要搜索視窗
usrfont_api.c	SetWindowLongPtrW / CallWindowProcW	不再需要子類別化

步驟 2D.3：保留的必要邏輯

原位置	邏輯	新位置
WndProc WM_PAINT	ghost detection（比對 s_aDrawnCells）	paint callback 內
WndProc WM_PAINT	ScanAndDrawUda() 呼叫	paint callback 內
WndProc WM_CHAR	DynAllocSlot PUA 分配	char callback 內
WndProc WM_TIMER	KillTimer(9988) + InvalidateRect	timer callback 內
WndProc WM_KEYDOWN	SetTimer(9988, 50)	移到 paint callback 或保留 keydown callback
WndProc WM_KILLFOCUS	s_nCaretRow/Col = -1	考慮加 focus callback 或用 gtwvt KillCaret
WndProc WM_ERASEBKGND	return 1（防背景清除）	gtwvt.c 直接加判斷

Phase 2D（續）：特殊訊息處理

VK_INSERT（WM_KEYDOWN）處理

UsrFontWndProc 在 VK_INSERT 時設定 50ms timer 同步游標高度。有兩個選擇：

選擇 A：加第四個 callback（keydown callback）— 但 overkill

選擇 B（推薦）：直接在 gtwvt.c WM_KEYDOWN 加入 VK_INSERT timer 邏輯（~3 行）

```
case WM KEYDOWN:
case WM SYSKEYDOWN:
/* □□□VK INSERT □ 50ms □□□□□□□□□□*/
if( wParam == VK INSERT && s pPaintCallback )
SetTimer( hWnd, 9988, 50, NULL );
/* □□ IME □□... */
```

WM_ERASEBKGND 處理

UsrFontWndProc 在字型載入後攔截 WM_ERASEBKGND 回傳 1（防止背景清除閃爍）。

直接加到 gtwvt.c 的 WndProc：

```
case WM ERASEBKGND:
if( s pPaintCallback ) /* usrfont □□□ */
return 1; /* □□□□□□ */
break; /* □□ DefWindowProc */
```

WM_KILLFOCUS / WM_SETFOCUS 處理

原有 usrfont 重設 caret 狀態（s_nCaretRow/Col = -1）。gtwvt 本來就有 KillCaret/SetFocus 處理（第 3633-3637 行），不需額外 callback。只需確認 ScanAndDrawUda 的 caret 重建邏輯能配合 gtwvt 的 UpdateCaret。

Phase 3：extern → pWVT 結構體整合

這是最乾淨的最終架構，但改動最大。建議 Phase 1+2 穩定後再考慮。

3.1 目標

將 7 個 extern 全域變數改為 HB_GTWVT 結構體（pWVT）成員。ScanAndDrawUda 透過 callback 參數接收 pWVT 指標。

3.2 修改項目

現有 extern	新增 pWVT 成員	說明
g_nCellW	pWVT->PTEXTSIZE.x	已存在，直接用
g_nCellH	pWVT->PTEXTSIZE.y	已存在，直接用
g_nMarginLeft	pWVT->MarginLeft	已存在，直接用
g_nMarginTop	pWVT->MarginTop	已存在，直接用
g_nFontLfWidth	pWVT->fontWidth	已存在（或加新成員）
g_bHookPaint	pWVT->bHookPaint	新增 BOOL 成員
g_hPaintBuf	pWVT->hdcPaintBuf	已存在，直接用
g_bCaretHidden	pWVT->CaretHidden	已存在（BOOL → int）

3.3 paint callback 簽名變更

```
/* Phase 2 □ callback □□ */
typedef void (*HB_GT_PAINT_CALLBACK)( void );
/* Phase 3 □□□□ pWVT */
typedef void (*HB GT PAINT CALLBACK)(
PHB_GTWVT pWVT );
```

3.4 影響範圍

檔案	修改內容
gtwvt.h	加 bHookPaint 成員到 HB_GTWVT 結構體；修改 callback 型別
gtwvt.c	PaintText 不再寫 extern 變數，直接用 pWVT 成員；callback 傳 pWVT
usrfont_c.c	移除 g_nCellW 等全域變數定義
usrfont_scan.c	ScanAndDrawUda 改為接收 pWVT 參數，從 pWVT 讀取 cell metrics
usrfont_int.h	加 #include "gtwvt.h"（需要 PHB_GTWVT 型別）

風險：usrfont_int.h 加 #include "gtwvt.h" 會引入大量定義，可能造成 BCC ICE

備案：不 include gtwvt.h，改為在 usrfont_int.h 前向宣告 struct _HB_GTWVT

附錄 A：各 Phase 的檔案修改矩陣

檔案	Ph1	Ph2A	Ph2B	Ph2C	Ph2D	Ph3
build_gtwvt.bat	Y					
stk32.hbp	Y					
gtwvt.h		Y	Y	Y		Y
gtwvt.c (WM_PAINT)		Y				Y
gtwvt.c (WM_CHAR)			Y			
gtwvt.c (WM_TIMER)				Y		
gtwvt.c (WM_KEYDOWN)					Y	
gtwvt.c (WM_ERASEBKGND)					Y	
usrfont_wndproc.c					Y	
usrfont_api.c					Y	
usrfont_c.c						Y
usrfont_scan.c						Y
usrfont_int.h						Y

附錄 B：各 Phase 完成後的測試清單

測試項目	Phase 1	Phase 2	Phase 3
Stk32 啟動正常	Y	Y	Y
造字（UDA）正確顯示	Y	Y	Y
ASCFONT 框線/符號正確	Y	Y	Y
游標不閃爍（caret tail）	Y	Y	Y
PickList scrollbar 拖動	Y	Y	Y
MessageBox hover	Y	Y	Y
主選單滑鼠自由選取	Y	Y	Y
DIALOG 右鍵 ESC	Y	Y	Y
INSERT 鍵游標高度切換	Y	Y	Y
IME 中文輸入（注音/倉頡）	Y	Y	Y
PUA 罕用字輸入		Y	Y
視窗最大化/還原	Y	Y	Y
WriteSys 退出速度	Y	Y	Y

附錄 C：風險與對策摘要

風險	Phase	機率	對策
BCC ICE（usrfont_c.c 單獨編）	1	中	改 -O1 或加 dummy.c 一起編
符號重複定義	1	低	各 obj 為獨立編譯單元，static 不衝突
callback 時序	2A	低	PaintText 完成後才呼叫，與現有 subclass 一致
PUA wParam 修改競爭	2B	中	callback 在 HandleWideChar 之前呼叫
Timer ID 衝突	2C	低	9988 已與現有 timer 不衝突
移除 subclass 後遺漏	2D	中	逐一比對 UsrFontWndProc 每個 case
usrfont_int.h include gtwvt.h BCC ICE	3	高	改用前向宣告避開