

GTWVT 底層流暢化（FlushDelay 節流）功能報告

日期：2026-08-10 專案：HBSTK (Stk32 GTWVT 引擎)

一、訴求與現象

GTWVT 畫面不如 HyperDos 流暢：每一個輸出 API（每個 @say）都經 HB_GTSELF_FLUSH → Refresh → 訊息幫浦（pump），把 InvalidateRect 立即轉成同步 WM_PAINT。一個 50 個 @say 的畫面就是 50 次重繪往返，畫面「逐欄爬出來」。既有做法是逐支 PRG 優化（DispBegin/TBrowse），但本模組未來要開放給不會使用 AI 的同業——不可能要求同業逐支優化，若不處理底層，系統將「叫好不叫座」。目標：底層一次處理好，所有 PRG（含未優化的）自動流暢。

二、踩坑與陷阱

坑	症狀	防範／解法
caret 消失（兄弟實踩，debug 很久）	節流後 GET SetPos 之後的 Refresh 被跳過，沒有人更新 caret → GET/READ 游標消失、看似無法輸入	ReadKey 無條件 pump（程式即將等待輸入的點，畫面與 caret 必須正確、永不節流）+ pump 後重啟節流窗 dwLastPump
caret 停在舊內容上	節流期間若仍移動 caret，游標會停在還沒畫的空白／舊字上	Refresh 的 caret 更新故意放在 pump 分支內：沒 pump（沒畫）就不動 caret
兩版 gtwvt.c 嚴重分岔	我方 6242 行（usrfont 源碼級合併、剪貼簿影像、自訂 GTI）；兄弟 5590 行（ST087C：flushdelay、boxfont、IME 定位）。整檔覆蓋任一方向都會毀掉對方獨有功能	禁止整檔覆蓋；先函式級 diff 盤點，只精準移植 flushdelay 相關 5 小塊
GTI id 衝突風險	兄弟用 id 203；我方自訂已佔 202/210/211/213/214	確認 203 在我方空著 → 沿用同 id，兩版語意一致

錯誤的節流層（v1 計畫自省）	若節流 ProcessMessages 本身，會連鍵盤訊息（進 Keys[] 佇列）一起跳過 → 無法輸入	節流的是「輸出路徑的 pump 時機」；輸入路徑（ReadKey）永遠無條件 pump
-----------------	---	---

三、解決方法（兄弟 ST087C 方案，直接移植）

核心：HB_GTSUPER_REFRESH 照常執行（RedrawDiff → InvalidateRect 照常「標記」update region），只節流 Refresh 末端的 pump（WM_MY_UPDATE_CARET + ProcessMessages）。不 pump 期間，Windows 原生 update region 自動累積合併，下一次 pump 一次畫出整塊——不需自己維護 dirty region，程式碼極小。間隔用 GetTickCount() unsigned 減法，49.7 天溢位天生安全。

止血設計：iFlushDelay=0（預設）時完全不進新分支，執行路徑與未改版一 byte 不差。設定鍵：System.ini [GTWVT] FlushDelay → main.prg 開機 GETE 讀入 → hb_gtInfo(203, n) 推給 driver（0~500 夾限）。對 203 賦值同時強制 pump 一次，PRG 可在長運算前保證「請稍候」訊息上畫面。

FlushDelay	效果	適用
0（預設／止血）	每次 Refresh 都 pump = 原行為	出問題立即回復；保守部署預設
16 毫秒	約 60fps 上限	追求最即時手感
33 毫秒（建議）	約 30fps 上限（電視等級，肉眼無感延遲）	合併力道與手感的平衡點
50~500 毫秒	合併更強，畫面更「整塊」	特慢環境／遠端桌面實驗值

四、使用技術與用法

技術	用法／說明
Windows update region 合併	InvalidateRect 只標記；延後 DispatchMessage(WM_PAINT) 讓 region 自動 Union，一次重繪
hb_gtInfo 自訂 GTI (203)	PRG 層執行期讀寫 iFlushDelay：hb_gtInfo(203)取值、hb_gtInfo(203,n)設定＋強制 pump 一次
GetTickCount 節流窗	dwNow - dwLastPump < iFlushDelay 則跳過 pump；unsigned 減法處理 wrap
ReadKey 安全網	等待輸入前無條件 pump + caret 更新，保證畫面最新、游標正確 (caret 命脈)
Python bytes-only 修改	main.prg (BIG5 混合行尾) 與 System.ini 以 bytes 錨點置換，不做 decode/encode 循環
一鍵重編	stkgt_lib\build_StkGTWVT.bat：gtwvt.c(-O2+UNICODE) → 合併 usrfont obj → 安裝為 Harbour gtwvt.lib → 重編 Stk32

五、修改檔案清單

檔案	改動內容
gtwvt_patch\gtwvt.h	struct 加 iFlushDelay / dwLastPump (附兄弟原理註解)
gtwvt_patch\gtwvt.c	初始化顯式歸零 Refresh pump 節流分支 ReadKey 無條件 pump+重啟節流窗 InfoCustom 加 case 203
source\main.prg	hb_gtInfo(211) 之後加 hb_gtInfo(203, GETE("FlushDelay")) (開機載入)
build\System.ini	檔尾加 [GTWVT] FlushDelay=33 (本機測試值；部署預設 0)
build\ini_editor.py	INI_META「螢幕設定」group 加 FlushDelay 下拉 (0止血/16/33/50)
ini_template.json x2	build 與 dist 同步加入 FlushDelay entry (default 0)

doc\GTWVT底層流暢化計畫_20260809.md

計畫文件 v2 (兩版比對、caret 坑分析、驗收清單)

六、後續 (回公司拍板)

部署預設值：各站先 0 實測一輪，穩定後改 33 出貨給同業。長運算 PRG 套用「賦值 203 強制刷新」顯示請稍候。兄弟版另兩項 (框線伴隨字型 204、IME 組字視窗跟隨 caret) 擇期評估移植。idle 重讀 System.ini (不重開就止血) 掛載點。DBF.PRG 索引資料壓力測試 (caret 高風險場景)。