STK 庫存系統
轉換工作技術報告

Clipper → Harbour 32-bit GTWVT Console Mode

日期：2026-04-19 ~ 2026-04-20

本次完成工作清單

1. GETSYS QuickBox — 按鍵（K_PGDN / K_ESC）被靜默吞掉，無法翻頁或取消
2. WINFORM 半邊字 — BIG5 全形字後半 byte 在 TGET INACTIVE 時消失
3. 游標殘影 & 消失 — 雙緩衝截圖時 XOR caret 被烙印，全形游標尺寸錯誤
4. Incremental Key-Repeat — 按住方向鍵/PgUp/PgDn 可流暢連續捲動，放開即停

套用檔案：TABLE、BUYSELL、DOMCMP +

INQUIRY/ENQUIRY/DOMPUR/DOMARR/FORPUR/FORARR/FORCMP/DOMSCH/DOMORD/QUOT/STOCK/DEPOSIT

1. GETSYS QuickBox 按鍵被吞掉問題

1.1 背景

QuickBox 是 GETSYS.PRG 中以 MsHitTest() 偵測滑鼠點擊 GET 欄位的機制。
當使用者在 WINFORM 等輸入畫面中，滑鼠點擊特定欄位時，
系統透過 aHitGet 陣列記錄點擊的行、欄、鍵碼等資訊並加以處理。

1.2 問題現象

在 WINFORM 輸入畫面中，按下 K_PGDN（下一頁）或 K_ESC（取消）時，
按鍵無反應——既不翻頁也不取消，游標原地不動。
透過 WRITEDEBUGLOG() 診斷確認：按鍵確實觸發，但未到達預期的 DO CASE 分支。

1.3 根本原因

問題出在 GETSYS.PRG L361-363 的 MsHitTest 點擊判斷 IF 條件過寬：

```
// 原始條件（過寬）
IF aHitGet != NIL .AND. aHitGet[5] == 0 .AND. ;
  ValType(aHitGet[4]) == "N" .AND. aHitGet[4] >= 1 .AND. aHitGet[4] <= Len(GetList)
 // 當 aHitGet[4] = K_PGDN = 3，且 Len(GetList) >= 3 時，條件成立！
 // -> 系統以為點擊了第 3 個 GET 欄位，進入 GET 切換流程
 // -> K_PGDN 被靜默吸收，不進入後續 DO CASE
```

K_PGDN 的常數值為 3，K_ESC 的某些路徑也會有類似數值落入 GET 索引範圍，
導致這些按鍵被誤判為「點擊第 N 個 GET 欄位」，靜默吸收而無任何反應。

同時，QuickBox 滑鼠觸發路徑（L380-394）缺乏 fallback 分支，
當 aHitGet 非 NIL 但不符合任何 ELSEIF 條件時，按鍵直接被丟棄。

1.4 解法（雙重修正）

修正 1：加入 GET 行號比對守衛

```
// 修正後（L361-363）：加入 GetList[aHitGet[4]]:row == aHitGet[1] 守衛
IF aHitGet != NIL .AND. aHitGet[5] == 0 .AND. ;
  ValType(aHitGet[4]) == "N" .AND. aHitGet[4] >= 1 .AND. aHitGet[4] <= Len(GetList) .AND. ;
  GetList[aHitGet[4]]:row == aHitGet[1]  // 確保 GET 行號與點擊行相符
 // K_PGDN=3：GetList[3]:row 通常與 MRow() 不同 -> 條件不成立 -> 不被誤判
```

修正 2：加入 fallback hb_keyPut

```
// L395-397：QuickBox ELSEIF 鏈末端加 fallback，
// 讓不符合任何條件的按鍵（K_PGDN、K_ESC 等）回注鍵盤佇列
ELSEIF aHitGet != NIL
 /* 其他並非要攔截的 key (K_PGDN/K_ESC) 透過 hb_keyPut 返還 */
 hb_keyPut( aHitGet[4] )
ENDIF
```

[OK] K_PGDN 翻頁、K_ESC 取消功能正常，QuickBox CTRL 系列鍵不受影響。

1.5 QuickBox 完整鍵碼處理流程（修正後）

aHitGet 狀態	條件	處理方式
NIL	MsHitTest 無命中	不做任何事
aHitGet[5] > 0	點到 dialog button	設 nCursor · exitState=GE_WRITE
aHitGet[5]==0 aHitGet[4] 為 GET 索引 GET 行號吻合	直接點擊 GET 欄位	跳到目標 GET（GE_DOWN）
aHitGet[5]==0 GETINKEYBLOCK 存在 aHitGet[4] 是 CTRL 鍵	QuickBox 區域觸發	K_CTRL_DOWN/UP：開啟搜尋框 K_CTRL_ENTER/LEFT：QuickPaste
其他（aHitGet 非 NIL）	K_PGDN、K_ESC 等	hb_keyPut 返還鍵盤

2. WINFORM 半邊字（BIG5 Trail Byte 消失）

2.1 問題現象

在 WINFORM 輸入畫面中，未取得焦點的 TGET 欄位（INACTIVE 狀態）
顯示中文全形字時，只顯示左半格，右半格空白或亂碼。
例如：「二」顯示成半個字、「點」右半消失、「喆」「堃」出現亂碼塊。

2.2 BIG5 編碼背景

BIG5 全形中文字佔 2 bytes（lead byte + trail byte），對應螢幕 2 個半形格寬。
GTWVT 在 WM_PAINT 時，每個螢幕格位（cell）各自處理。
若某格的顯示被後續操作覆蓋，trail byte 格就會消失或顯示錯誤字元。

2.3 根本原因

WINFORM.PRG L87 有一個 colorRect 操作，用來重繪輸入欄位的背景顏色：

```
// WINFORM.PRG L87 附近（簡化）
ColorRect( nRow, nCol, nRow, nCol+nLen-1, cColor ) // 重繪欄位背景
// colorRect 會整列重寫，覆蓋 TGET 已顯示的內容
// -> BIG5 全形字的 trail byte 格被空格覆蓋
// -> 視覺上只剩 lead byte 格，右半消失
```

GTWVT 的 ColorRect 是直接操作字幕緩衝區（screen buffer），
它不知道某個格位是 BIG5 trail byte，直接以空格+新色彩填入，
導致全形字被攔腰截斷。

2.4 解法

在 colorRect 之後，補畫所有 GetList 的 display()，
確保 TGET 欄位的內容是最後寫入螢幕的，覆回完整的 BIG5 全形字。

```
// WINFORM.PRG L87 修正後
ColorRect( nRow, nCol, nRow, nCol+nLen-1, cColor ) // 重繪背景

// 補畫所有 GET 欄位，確保 BIG5 trail byte 不被覆蓋
LOCAL oG
FOR EACH oG IN GetList
  oG:display()  // 重新顯示 TGET 內容（包括 trail byte 格）
NEXT
// 這樣 TGET 的顯示在 colorRect 之後，是最後寫入的
```

[OK] 所有 TGET INACTIVE 欄位的中文全形字完整顯示，不再出現半邊字。

2.5 踩坑紀錄

嘗試方向	失敗原因
在 TGET 的 display() 內修正 trail byte 偵測	GTWVT cell 層級不感知 BIG5 雙字節邊界，修改複雜且影響面廣
修改 ColorRect 跳過 trail byte 格	ColorRect 是底層 API，無法感知上層的 BIG5 內容

嘗試方向	失敗原因
只重畫 active GET	colorRect 可能覆蓋任意欄位，必須重畫所有 GetList

3. 游標殘影 & 消失修復（摘要）

詳細技術報告請見：E:\ConvertFW\HBSTK\doc\游標殘影修復技術報告_20260420.pdf
以下為重點摘要。

3.1 問題清單

問題	現象	根本原因
游標殘影	游標移動後留白色矩形	XOR caret 被雙緩衝 BitBlt 截入 off-screen buffer，貼回時烙印
游標消失（SC_NONE）	WINFORM 輸入時無游標	ScanAndDrawUda else 分支每次 WM_PAINT 都 DestroyCaret
全形游標不見	中文格游標消失或半格	GTWVT 固定建 1×CellW caret，全形需 2×CellW
INSERT 游標偏移	INSERT 游標出現在視窗頂端	SetCaretPos Y 未減游標高度 _ch

3.2 最終解法核心

```
WM_PAINT -> ScanAndDrawUda:

 1. HideCaret(hWnd)                  // 先隱藏，防止 XOR caret 被截入
 2. BitBlt(screen -> s_hdcDblBuf)      // 截取乾淨畫面
 3. DrawXxx(s_hdcDblBuf, ...)          // 畫 UDA/PUA/ASCFONT 字形
 4. BitBlt(s_hdcDblBuf -> screen)      // 一次更新，無殘影
 5a. s_nCaretStyle > 0:              // 有游標（NORMAL/INSERT）
     DestroyCaret()
     CreateCaret(hWnd, NULL,
       s_nCaretWidth * nCellW,       // 全形=2×CellW，半形=1×CellW
       _ch)                          // 高度：NORMAL=6px, INSERT=12px
     SetCaretPos(col*CellW, (row+1)*CellH - _ch)
     ShowCaret(hWnd)
 5b. s_nCaretStyle <= 0 (SC_NONE):     // 游標隱藏狀態
     ShowCaret(hWnd)                 // 讓 GTWVT caret 自然顯示
```

[OK] 殘影消除、全形/半形游標正確、INSERT 高度可設定、SC_NONE 不破壞游標。

3.3 關鍵教訓

- HideCaret/ShowCaret 必須在同一 WM_PAINT 週期內平衡，不能跨週期
- DestroyCaret 是破壞性操作，只在確定要重建不同尺寸時使用
- WM_MY_UPDATE_CARET 不可抑制——抑制即殺掉 GTWVT blink timer
- SC_NONE 是 WINFORM 期間的正常狀態，不可將其視為「異常」做清除
- stk.ini 的 CARET_NORMAL_H / CARET_INSERT_H 提供彈性調整游標高度

4. Incremental Key-Repeat（按住鍵連續捲動）

4.1 問題背景

在各訂單瀏覽模組中，使用者按住 K_UP / K_DOWN / K_PGUP / K_PGDN 時，
系統每次只移動一步，需反覆按鍵才能快速瀏覽大量資料。
目標：按住不放時自動連續移動，放開立即停止，不卡頓、不衝過頭。

4.2 TABLE.PRG 的既有機制（參考基準）

TABLE.PRG 使用 TBrowse 控制項，其 DO WHILE !oBrowse:stable 穩定化迴圈
內部呼叫 InKey()（無逾時 = 非阻塞），天然形成 incremental 效果：

```
DO WHILE !oBrowse:stable
  oBrowse:stabilize()
ENDDO
// oBrowse:stabilize() 內部呼叫 InKey(0) 非阻塞讀取
// OS 鍵盤 auto-repeat 持續注入相同按鍵 -> 迴圈持續執行
// 放開按鍵 -> 佇列清空 -> InKey() 回 0 -> 穩定化結束
```

InKey(0)（= InKey() 無參數）是非阻塞的：若鍵盤佇列有鍵則讀取並回傳，
若佇列空則立即回傳 0，不等待。這是 incremental 機制的核心。

4.3 三種代碼模式與解法

模式 A：Inline GoSerialBy（INQUIRY / ENQUIRY 等 6 個檔案，K_UP/K_DOWN）

原始結構：ShowOrderLine(.F.) → Up/Down() → GoSerialBy() → ShowOrderLine(.T.)
修改：在 GoSerialBy 後、ShowOrderLine(.T.) 前插入 DO WHILE 迴圈

```
// 修改後（K_UP 為例）
ShowOrderLine( nRow, aLen, .F. )      // 取消高亮（先執行一次）
Up( nBeginRow, @nRow, ... )           // 移動游標
GoSerialBy( aHead[ORDERNO], nSerialNo, --nSerialNo, nTotalLine )

DO WHILE nSerialNo > 1 .AND. InKey() = K_UP   // 按住期間持續執行
  ShowOrderLine( nRow, aLen, .F. )    // 取消高亮
  Up( nBeginRow, @nRow, ... )         // 移動
  GoSerialBy( aHead[ORDERNO], nSerialNo, --nSerialNo, nTotalLine )
ENDDO
ShowOrderLine( nRow, aLen, .T. )      // 放開後才更新高亮（只畫一次）
```

模式 B：KeyUpDown 函式委派（QUOT / STOCK / DEPOSIT，K_UP/K_DOWN）

原始：CASE K_UP .OR. K_DOWN → IF KeyUpDown(LastKey(), ...) → ChkBotMsg
問題：InKey() 呼叫後 LastKey() 會被改寫，需先存起來

```
CASE LastKey() = K_UP .OR. LastKey() = K_DOWN
  nLastKey := LastKey()               // 先存，InKey() 後 LastKey() 會變
  IF KeyUpDown( nLastKey, @nUpper, @nLower, @nRow, @nSerialNo, ... )
    DO WHILE InKey() = nLastKey       // 按住期間持續執行
      IF ! KeyUpDown( nLastKey, ... )
```

```
      EXIT                    // 已到邊界 -> 停止
    ENDIF
   ENDDO
   ChkBotMsg( ... )              // 放開後才更新底部訊息
   IF( MEMVAR->lHostImage, GetPicture(.T.), NIL )
  ELSE
   LOOP
  ENDIF
```

模式 C/D：KeyPageUpDown 函式委派（INQUIRY 等，K_PGUP/K_PGDN）

KeyPageUpDown 內部以 LastKey() 判斷 PGUP/PGDN 方向，
故迴圈內每次呼叫前必須先 hb_keyPut(nLastKey) 確保 LastKey() 正確：

```
CASE LastKey() = K_PGUP .OR. LastKey() = K_PGDN
 nLastKey := LastKey()                  // 存起方向鍵
 nOldSerialNo := nSerialNo
 IF KeyPageUpDown( @nUpper, @nLower, @nRow, nOldSerialNo, @nSerialNo, ... )
  DO WHILE InKey() = nLastKey            // 按住期間
   hb_keyPut( nLastKey )                // 確保 LastKey() 正確
   nOldSerialNo := nSerialNo
   IF ! KeyPageUpDown( @nUpper, ... )
    EXIT
   ENDIF
  ENDDO
  ChkBotMsg( ... )
 ENDIF
```

模式 E：Inline 獨立 CASE K_PGUP / K_PGDN（FORCMP / DOMSCH 等）

K_PGUP 和 K_PGDN 是獨立 CASE，ELSE 塊含 PageUp/PageDown + GoSerialBy。
修改：將 ELSE 塊內容包進 DO WHILE .T.，末尾以 InKey() 判斷是否繼續：

```
CASE LastKey() = K_PGUP
 IF nTotalLine = 0 .OR. nSerialNo = 1
  LOOP
 ELSE
  DO WHILE .T.
   nOldSerialNo := nSerialNo
   ShowOrderLine( cAlias, nRow, aLen, .F. )
   PageUp( nTotalLine, @nSerialNo, nBeginRow, @nRow, @nPageTop, ... )
   nRealSkip := ShowBody( cAlias, ... )
   GoSerialBy( aHead[ORDERNO], nSerialNo+nRealSkip, nSerialNo, nTotalLine )
   IF nSerialNo = 1 .OR. InKey() != K_PGUP   // 到頂 or 放開按鍵 -> 停止
    EXIT
   ENDIF
  ENDDO
  ShowOrderLine( cAlias, nRow, aLen, .T. )   // 放開後更新高亮
 ENDIF
```

5. Incremental Key-Repeat 套用檔案清單

5.1 先行完成（手工 Python 腳本）

檔案	K_UP/K_DOWN 模式	K_PGUP/K_PGDN 模式	說明
TABLE.PRG	TBrowse stabilize 迴圈（原有）	TBrowse stabilize 迴圈（原有）	參考基準，不修改
BUYSELL.PRG	B：KeyUpDown 委派	D：KeyPageUpDown + hb_keyPut	fix_buysell_incremental.py
DOMCMP.PRG	A：GoSerialBy inline	E：獨立 CASE 包 DO WHILE	fix_domcmp_incremental.py

5.2 批量套用（fix_incremental_all.py）

檔案	K_UP/K_DOWN	K_PGUP/K_PGDN	nLastKey 處理	新增行數
INQUIRY	A（GoSerialBy）	C（KeyPageUpDown, 無 ELSE）	LOCAL 加入	+21
ENQUIRY	A（GoSerialBy）	C（KeyPageUpDown, 無 ELSE）	LOCAL 加入	+21
DOMPUR	A（GoSerialBy）	C（KeyPageUpDown, 無 ELSE）	LOCAL 加入	+21
DOMARR	A（GoSerialBy）	C（KeyPageUpDown, 無 ELSE）	LOCAL 加入	+21
FORPUR	A（GoSerialBy）	C（KeyPageUpDown, 無 ELSE）	LOCAL 加入	+21
FORARR	A（GoSerialBy）	C（KeyPageUpDown, 無 ELSE）	LOCAL 加入	+21
FORCMP	A（GoSerialBy）	E（獨立 CASE）	不需要	+20
DOMSCH	A（GoSerialBy）	E（獨立 CASE）	不需要	+20
DOMORD	A（GoSerialBy）	E（獨立 CASE）	不需要	+20
QUOT	B（KeyUpDown）	D（KeyPageUpDown, 有 ELSE）	L30 已有	+13
STOCK	B（KeyUpDown）	E（獨立 CASE, ShowOrderRow）	LOCAL 加入	+16
DEPOSIT	B（KeyUpDown）	E（獨立 CASE, ShowOrderRow）	LOCAL 加入	+16

5.3 批量腳本修正過的兩個 Bug

Bug 1：Pattern A 找錯 GoSerialBy（滑鼠滾輪 vs K_UP）

INQUIRY 等檔案同時有 K_MWFORWARD（滑鼠滾輪向上）和 K_UP，
兩者都使用 GoSerialBy(--nSerialNo)。原腳本直接找第一個 --nSerialNo，
找到的是滑鼠滾輪的（行號更小），導致 CASE K_UP 在其之後，條件判斷失敗。

```
# 修正前（錯誤）
dec_g = find_line(lines, b', --nSerialNo')  # 找到 K_MWFORWARD 的！
case_up = find_line(lines, b'CASE LastKey() = K_UP', 0, dec_g)  # dec_g < K_UP -> None

# 修正後（正確）
case_up = next(i for i,l in enumerate(lines)
           if b'CASE LastKey() = K_UP' in l and b'.OR.' not in l)
dec_g = find_line(lines, b', --nSerialNo', case_up, case_up+25)  # 在 K_UP 範圍內找
```

Bug 2：Pattern E 找到 .F. 版本而非 .T. 版本

ELSE 塊內有兩行 ShowOrderLine：第一行是 .F.（取消高亮），
第二行才是 .T.（更新高亮），是 DO WHILE 的邊界。
原腳本找第一個 ShowOrderLine(，但找到的是 .F. 版本，邊界抓錯。

```
# 修正前（錯誤）
sol_t = find_line(lines, b'ShowOrderLine( ', else_idx, else_idx+25)
# 找到 ELSE 塊的第一行 ShowOrderLine(.F.)，不是想要的 .T.

# 修正後（正確）：明確搜尋含 .T. 的行
sol_t = None
for i in range(else_idx, else_idx+25):
    if b'ShowOrderLine' in lines[i] and b'.T.' in lines[i]:
        sol_t = i; break
```

6. 踩坑總表（兩天所有問題）

坑	現象	根本原因	解法
GETSYS: K_PGDN=3 誤判為 GET 索引	K_PGDN 無反應 不翻頁	aHitGet[4]>=1 且 <=Len(GetList) K_PGDN=3 符合此範圍	加 GetList[idx]:row==MRow() 守衛條件
GETSYS: ELSEIF 鏈 無 fallback	K_ESC 被靜默吸收	aHitGet 非 NIL 但不符任何 ELSEIF -> 按鍵丟棄	末端加 ELSEIF aHitGet!=NIL hb_keyPut(aHitGet[4])
WINFORM: colorRect 覆蓋 trail byte	中文右半格消失 半邊字	ColorRect 以空格填入 不感知 BIG5 雙字節邊界	colorRect 後補畫 所有 GetList:display()
XOR caret 截入 buffer	游標移動後 白色殘影	BitBlt 前 caret 已 XOR 在螢幕	BitBlt 前先 HideCaret
DestroyCaret 在 SC_NONE	WINFORM 輸入 無游標	else 分支每次 WM_PAINT 都刪 caret	else 改 ShowCaret 不主動刪游標
WM_MY_UPDATE_CARET 被抑制	游標永久消失 blink 不動	GTWVT blink timer 從不啟動	移除抑制 讓 GTWVT 正常建 caret
全形格游標 1×CellW	中文位置游標 看似消失	GTWVT 固定 1 格寬	偵測 bFullWidth CreateCaret 2×CellW
INSERT 游標 偏到頂端	INSERT 模式游標 位置錯誤	SetCaretPos Y 未減游標高度	Y = (row+1)*CellH - _ch
Incremental: 找錯 GoSerialBy	Pattern A 處理 到滑鼠滾輪行	K_MWFORWARD 與 K_UP 共用同名 GoSerialBy	先找 CASE K_UP 再在其範圍內找
Incremental: 找到 .F. 而非 .T.	Pattern E 邊界 抓錯行	ELSE 塊第一個 ShowOrderLine 是 .F.	明確搜尋 b'.T.' in line
BCC32 W8004 未用變數	編譯失敗	移除游標區塊後 switch 計算無用途	同時刪宣告與計算
Python \n 在 C 字串	BCC32 Unterminated string constant	heredoc 內 \n 被解為換行字元	改用 fputc(10, f)
hb_keyPut 缺失 (KeyPageUpDown)	PGUP/PGDN 方向 在迴圈內錯誤	InKey() 後 LastKey() 已被改寫	每次迴圈前 hb_keyPut(nLastKey)

7. 相關檔案一覽

檔案路徑	說明
source\GETSYS.PRG	GET 系統核心：QuickBox 滑鼠處理（L352-398） 已修正：K_PGDN 守衛 + hb_keyPut fallback
source\WINFORM.PRG	輸入表單：colorRect 後補畫 GetList（L87 附近） 已修正：半邊字消失
compat\usrfont.prg	雙緩衝 + ASCFONT + 游標管理（#pragma BEGINDUMP C 代碼） 已修正：殘影、游標消失、全形尺寸、INSERT 高度
source\TABLE.PRG	TBrowse 瀏覽：Incremental 參考基準（原有機制）
source\BUYSELL.PRG	買賣單：Incremental K_UP/K_DOWN（Pattern B）+ K_PGUP/K_PGDN（Pattern D） fix_buysell_incremental.py 套用
source\DOMCMP.PRG	國內比價單：Incremental K_UP/K_DOWN（Pattern A）+ K_PGUP/K_PGDN（Pattern E） fix_domcmp_incremental.py 套用
source\INQUIRY.PRG ~ DEPOSIT.PRG	12 個模組：批量套用 Incremental（Pattern A/B/C/D/E） fix_incremental_all.py 套用
build\fix_buysell_incremental.py	BUYSELL Incremental 修改腳本
build\fix_domcmp_incremental.py	DOMCMP Incremental 修改腳本
build\fix_incremental_all.py	12 個檔案批量 Incremental 修改腳本（含 5 種 Pattern）
build\stk.ini	CARET_NORMAL_H / CARET_INSERT_H 游標高度設定
doc\游標殘影修復技術報告_20260420.pdf	游標問題詳細技術報告（9 章，11 頁）

8. 結語

本報告涵蓋 2026-04-19~20 兩天的主要技術工作。
四項工作均以「最小侵入」為原則：不改變原始程式邏輯，只在關鍵位置加入處理。

8.1 工作成果摘要

工作項目	影響範圍	結果
GETSYS QuickBox 按鍵修正	GETSYS.PRG 1 個位置	K_PGDN/K_ESC 正常穿透，QuickBox 功能不受影響
WINFORM 半邊字修正	WINFORM.PRG 1 個位置	所有 TGET INACTIVE 中文全形字完整顯示
游標殘影 & 消失修復	usrfont.prg ScanAndDrawUda（C 代碼，約 30 行）	殘影消除、全形游標正確、INSERT 高度可設定
Incremental Key-Repeat	15 個 PRG 檔案（TABLE 原有 + 14 個修改）	按住方向鍵/翻頁鍵流暢連續捲動，放開即停

8.2 設計原則

- InKey(0) 非阻塞讀取是 incremental 的基礎，OS auto-repeat 提供按鍵串流
- HideCaret/ShowCaret 夾住截圖是消除 XOR caret 殘影的通用模式
- BIG5 trail byte 必須由最後寫入者負責顯示完整，不能靠底層感知
- GETSYS 按鍵處理鏈末端必須有 hb_keyPut fallback，確保所有按鍵都有出路
- 批量腳本修改 PRG 前必須先找對錨點，避免同名 pattern 被多處誤套

本報告由 Claude Code (claude-sonnet-4-6) 自動產生
2026-04-20 E:\ConvertFW\HBSTK\doc\