```
;;      Jetton 錢包數據結構處理
;;      繁體中文註解版 by Y.C.
;;      所有中文均經過人腦編譯，不存在 AI 製造的奇怪語法，並可直接複製貼上取代原有的 jetton-uilts.fc 服用。



#include "workchain.fc";
const int STATUS_SIZE = 4; ;; 定義錢包狀態的佔用空間



;; 把數據包裝成 cell 用作更新 Jetton Wallet 的狀態
cell pack_jetton_wallet_data(int status, int balance, slice owner_address, slice jetton_master_address) inline
{
        return begin_cell() ;; 開始建構 cell
        .store_uint(status, STATUS_SIZE) ;; 儲存 status 狀態，佔用 4 bits （STATUS_SIZE）
        .store_coins(balance) ;; Jetton Token 餘額數量
        .store_slice(owner_address) ;; 錢包主權者地址
        .store_slice(jetton_master_address) ;; Jetton 主合約地址
        .end_cell(); ;; 結束 cell 建構
}



;; 把數據包裝成 cell 可用作代表或更新 Jetton Wallet 的狀態
cell calculate_jetton_wallet_state_init(slice owner_address, slice jetton_master_address, cell
jetton_wallet_code) inline {
        return begin_cell() ;; 開始建構 cell
        .store_uint(0, 2) ;; 兩個 flag bit (0b00)，分別表示split_depth & special
        .store_maybe_ref(jetton_wallet_code) ;; 裝進錢包代碼
        .store_maybe_ref( ;; 裝進錢包數據
            pack_jetton_wallet_data(
                0, ;; Status -
                        ;; 0: Uninitialized - 表示錢包尚未初始化
                        ;; 1: Active - 表示錢包已初始化並處於活動狀態
                        ;; 2: Frozen - 表示錢包已凍結，無法進行操作
                0, ;; Jetton Token 的 balance
                owner_address, ;; 持有者地址
                jetton_master_address) ;; Jetton 主合約地址
        )
        .store_uint(0, 1) ;; 以一個 0 bit 表示不存有額外字典引用
        .end_cell(); ;; 結束 cell 建構
}



;; 計算 Jetton 合約（賬戶）地址
slice calculate_jetton_wallet_address(cell state_init) inline {
        return begin_cell() ;; 開始建構 cell
        .store_uint(4, 3) ;; 將數字 4 存儲為二進制 - 0b100 定義地址類型為 addr_std
                        ;; 000 (0)：無地址（addr_none），表示地址未定義或不存在。
                        ;; 001 (1)：外部地址（addr_extern），表示地址不存在 TON 區塊鏈內。
```

```
                        ;; 010 (2)：標準地址（addr_std），通常用於表示 TON 區塊鏈內的標準地址。
                        ;; 011 (3)：變長地址（addr_var），表示變長格式的地址。
                        ;; 100 (4)：標準地址（addr_std），使用 3 位標籤，其中 100 被用來表示標準的 TON 區塊鏈地址，並
                        確保不使用 anycast 功能，意味地址僅指定特定節點，而不是一組節點。
        .store_int(MY_WORKCHAIN, 8) ;; 將工作鏈的值 (8 bits) 儲存到 cell
        .store_uint(cell_hash(state_init), 256) ;; 計算並存儲 state_init 的 hash
        .end_cell() ;; 結束 cell 建構
        .begin_parse(); ;; 轉化剛才建構的 cell，返回一個 slice
}



;; 計算 Jetton 用戶合約（賬戶）地址
slice calculate_user_jetton_wallet_address(slice owner_address, slice jetton_master_address, cell
jetton_wallet_code) inline {
   return calculate_jetton_wallet_address(calculate_jetton_wallet_state_init(owner_address,
jetton_master_address, jetton_wallet_code));
}



;; 用以規限及管理 payload 轉發的格式與操作
() check_either_forward_payload(slice s) impure inline {
        if (s.preload_uint(1)) {
                ;; 根據 stdlib.fc 解說： int preload_uint(slice s, int len) asm "PLDUX";
                ;; 從 slice s 預讀 1 bit 來判斷 ref 中是否存在有效 payload，不改變 slice 的狀態
            (int remain_bits, int remain_refs) = slice_bits_refs(s);
                ;; 根據 stdlib.fc 解說： (int, int) slice_bits_refs(slice s) asm "SBITREFS";
                ;; 回傳位元數據量和 cell 引用的數量
            throw_unless(error::invalid_message, (remain_refs == 1) & (remain_bits == 1));
                ;; 檢查是否恰好只剩一個引用和 bit flag，嚴格檢查確保數據的格式符合預期結構：
                ;;     remain_refs == 1：表示 slice 必須引用一個子單元（子 cell）
                ;;     remain_bits == 1：必須有且僅有一個 bit flag 標識狀態或條件
        }
                ;; 當 forward_payload 在 slice 中時，它的結構和內容則不受特定限制，
                ;; 可包含任意數量的 bits & refs，在這情況下 s 的格式是靈活的，
                ;; 所以工程師可因應情況新增 "else" 操作進行客制化。
}
```