微算機實驗報告

Final Project

姓名：仇健安
系級：電機系
學號：111511239
Demo 時間：2025/06/19

一、FINAL 介紹與目的：

這個期末專題，我設計了使用 8051 實作的音樂播放器系統。選擇這個題目，是因為流程剛好可以很好的結合整個學期學習的內容，包含 Table、Memory、Timer、LCD 顯示等重要的功能與困難的元件。

我打造一套使用者可以互動操作的音樂播放器，並實現「按鍵輸入 → 顯示狀態 → 播放音樂 → 隨時卡歌」的完整流程。

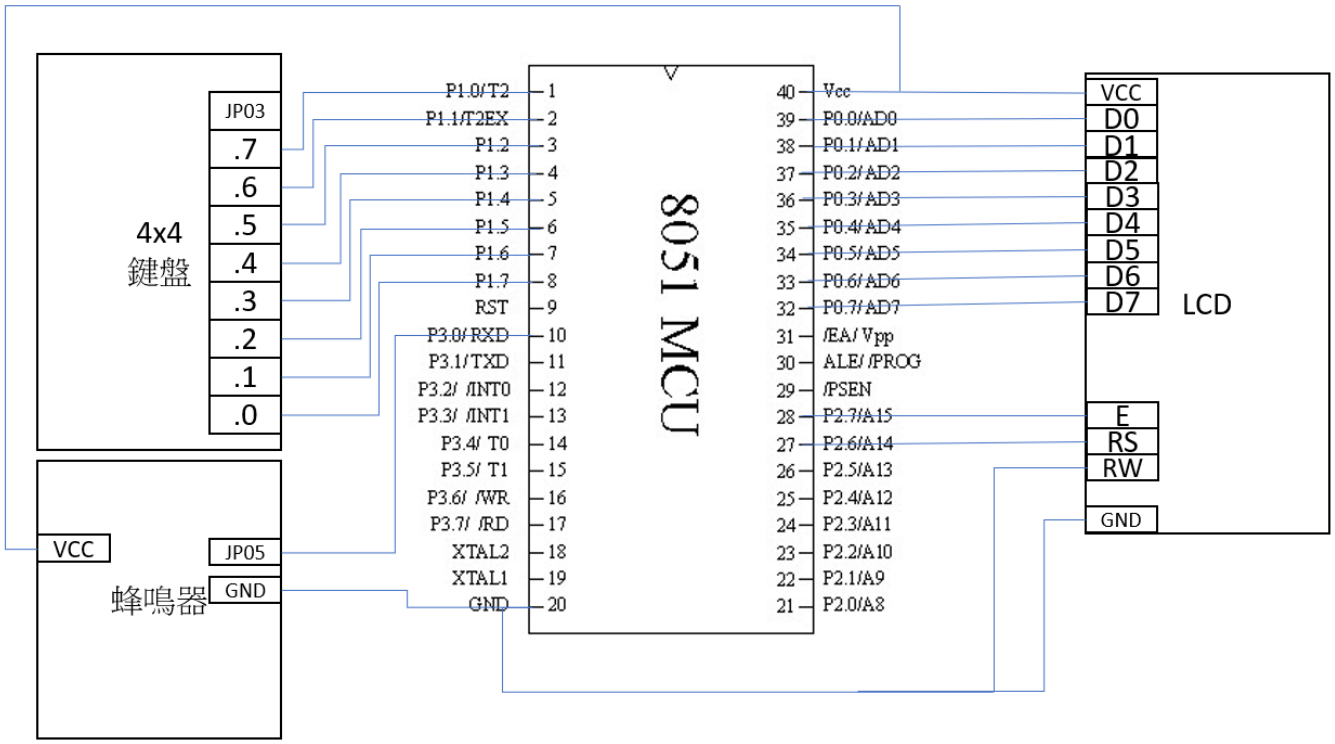
我的設計階段性目標如下：

- **LCD 正確初始化並顯示 welcome message**。
- **鍵盤輸入最多兩位數歌曲代碼**，用來選擇歌曲(1~99)。
- **LCD 顯示輸入狀態**並在按下撥放後顯示歌曲名稱，提供使用者回饋。
- **播放多首旋律**（如《小蜜蜂》、《小星星》、《天空之城》）。
- **新增更多音階**（如#Fa、高音 Do~Mi）
- **實作卡歌功能**，讓使用者在播放過程中可立即中斷音樂。

在實作過程中，我運用 8051 的多項功能模組，包括：

- **4x4 矩陣式鍵盤掃描**，以列掃描方式偵測使用者按鍵，並轉換為 ASCII 字元處理。
- **LCD 顯示模組**，搭配自訂指令顯示歡迎畫面、歌曲輸入提示與播放狀態。
- **蜂鳴器播放音樂**，透過 Timer 0 產生方波，控制 P3.0 輸出對應頻率，實現不同音階的播放。
- **音符對應表**（Song Table）以編碼方式儲存旋律，再由播放子程式逐一讀取並播放。
- **卡歌控制設計**，原本規劃使用 INT0 中斷方式達成播放中斷，後來想說直接整合在鍵盤中，改為在音樂撥放迴圈掃描鍵盤達成即時控制，減少再連接一個彈跳開關。

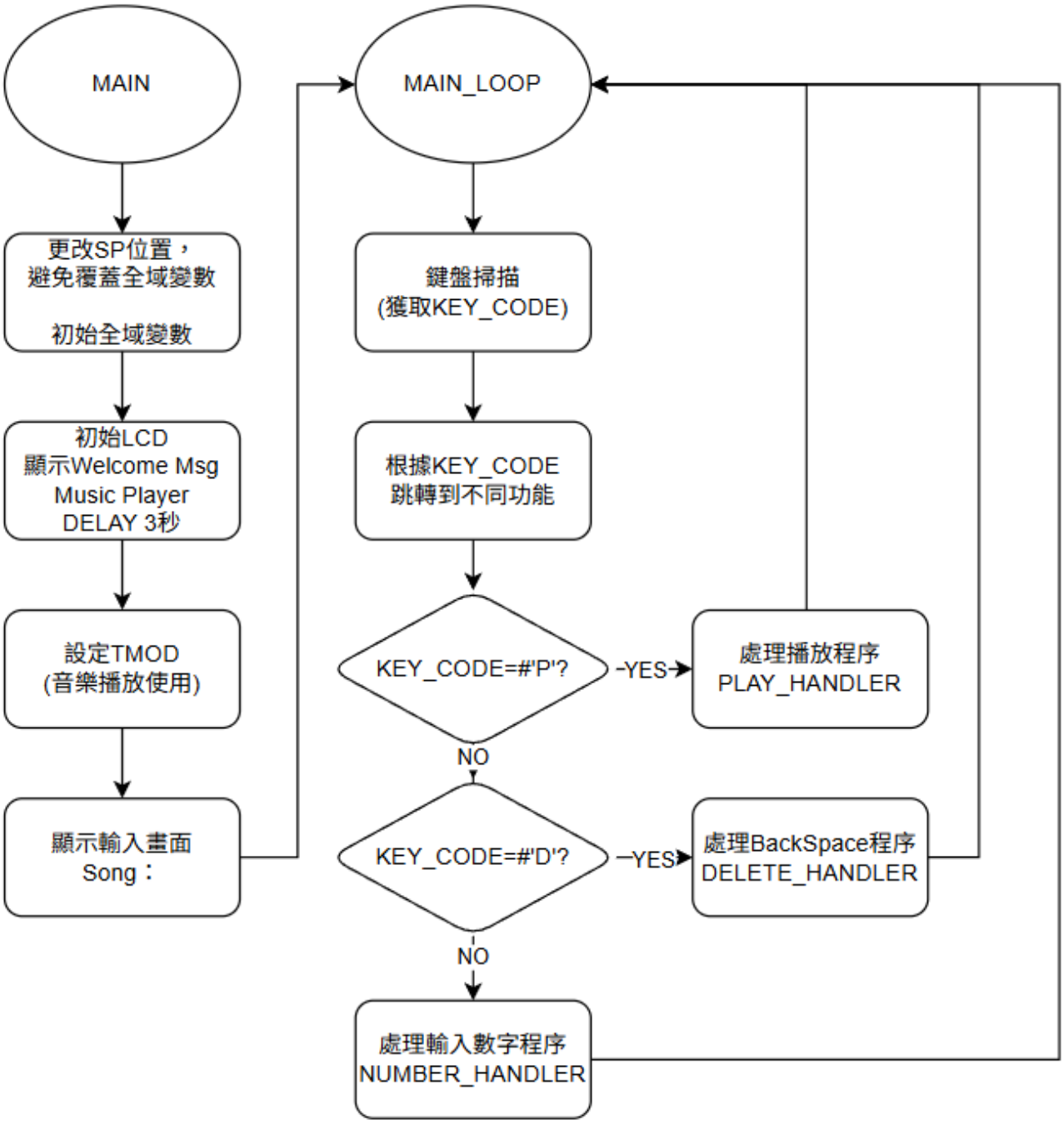
二、硬體架構：



元件	功能說明	接至 8051 腳位	備註
蜂鳴器	音樂播放輸出	P3.0	使用 Timer 0 產生方波控制頻率
LCD 顯示器	資料匯流排 DB0~DB7	P0.0 ~ P0.7	資料雙向傳輸
	RS（指令/資料選擇）	P2.6	RS=0: 指令；RS=1: 資料
	E（啟用）	P2.7	LCD 動作觸發腳位
	RW（讀寫選擇）	GND	設計僅寫入 LCD，RW 接地
矩陣鍵盤	4x4 鍵盤 - 列掃描 (Row)	P1.4 ~ P1.7	輸出低電位逐列掃描
	4x4 鍵盤 - 行讀取 (Col)	P1.0 ~ P1.3	讀取按鍵輸入電位
所有元件電源	系統供電	VCC / GND	5V 直流供電

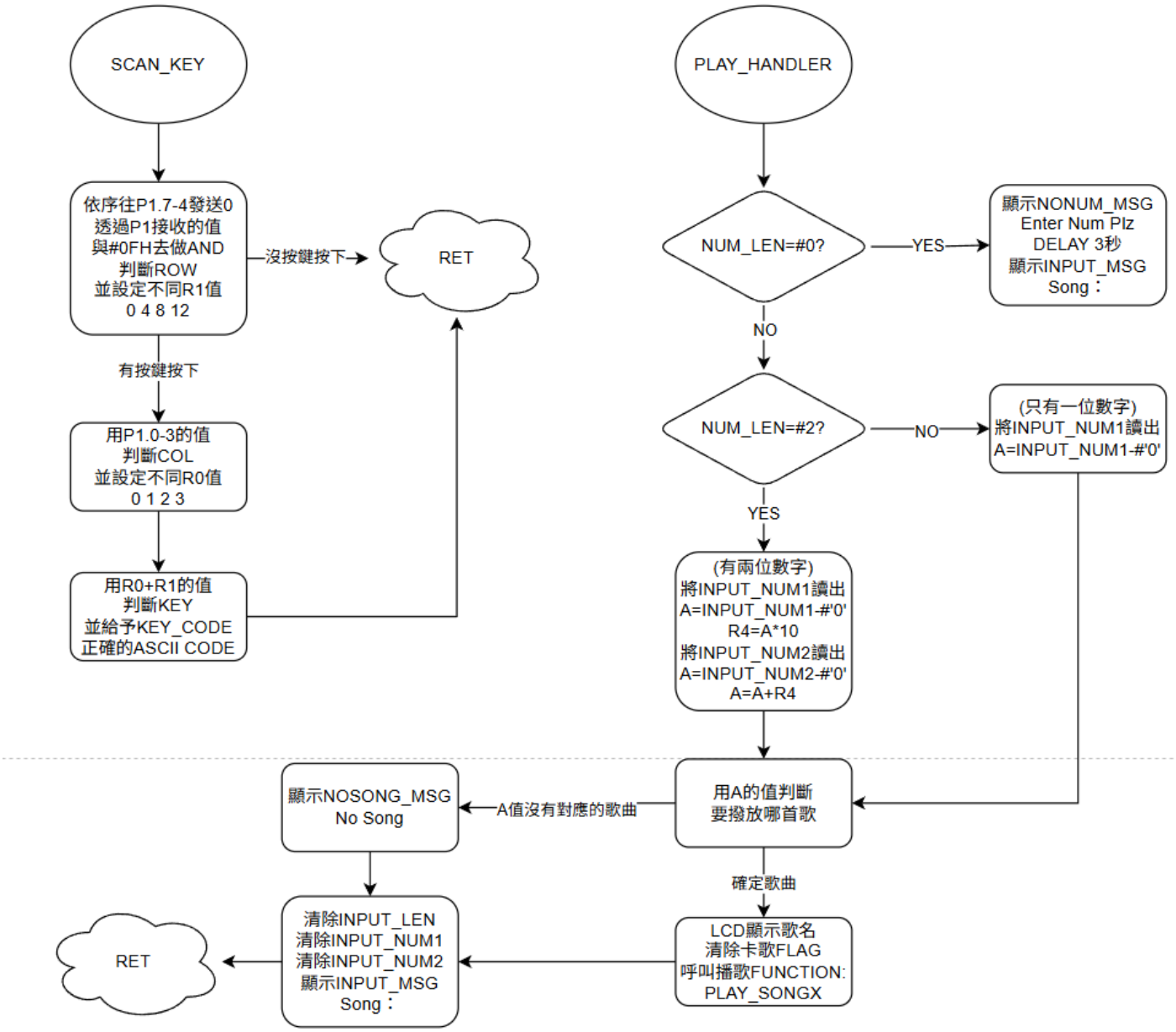
三、程式流程圖：

主要迴圈流程圖：

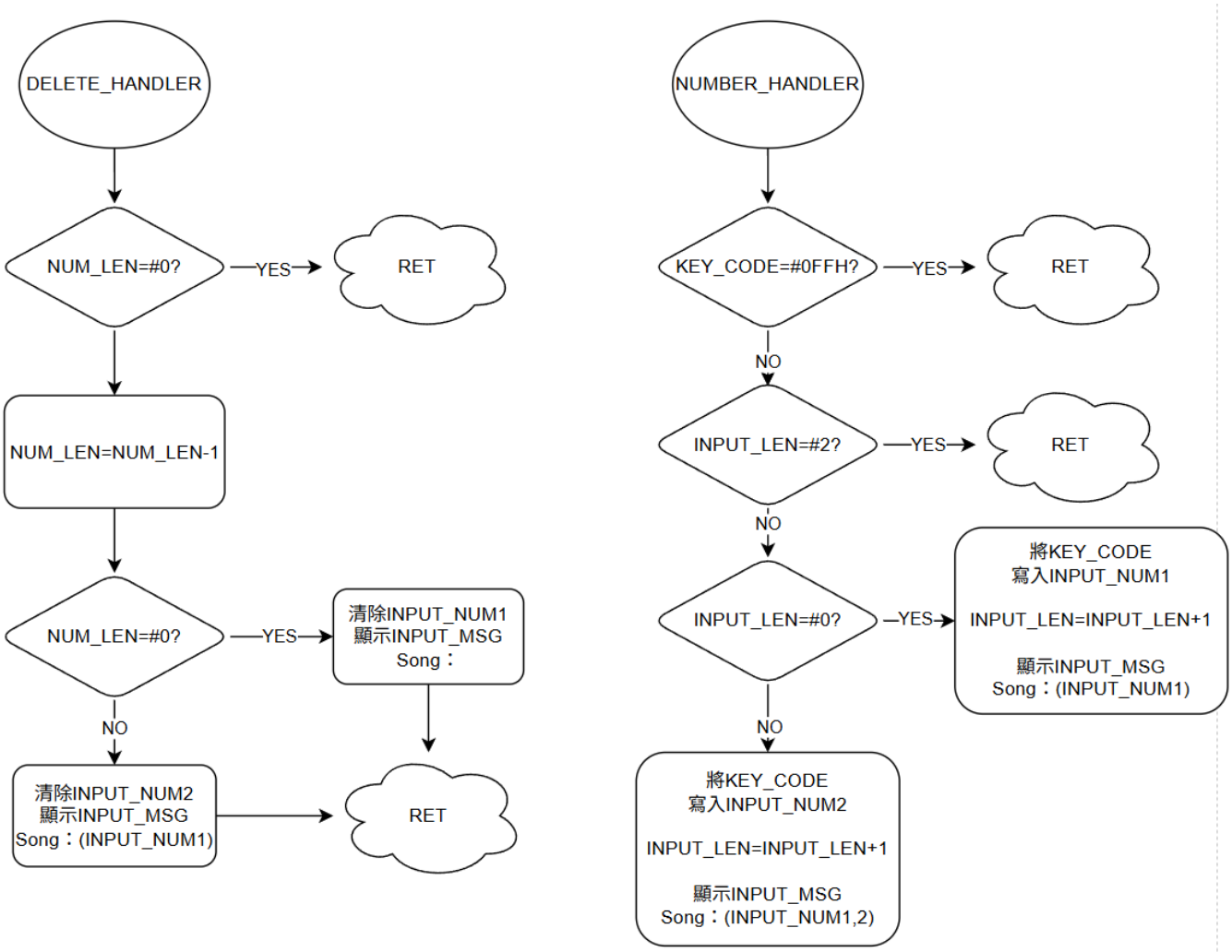


副程式流程圖：

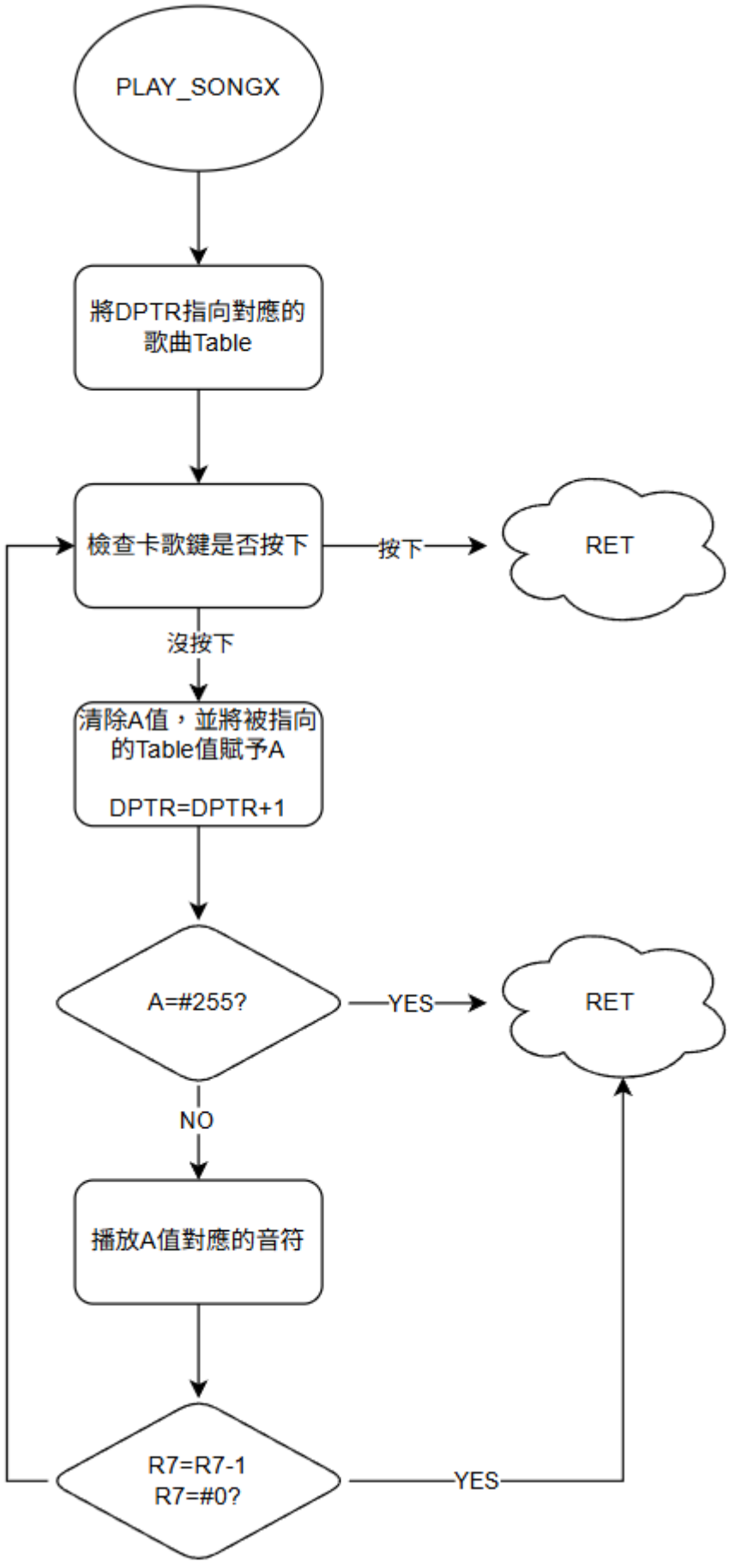
鍵盤掃描以及音樂撥放(PLAY_HANDLER)副程式：



刪除鍵處理(DELETE_HANDLER)以及數字處理(NUMBER_HANDLER)副程式：



歌曲播放(PLAY_SONGX)副程式:



四、問題與討論：

無

五、程式碼與註解：

主程式：

```
; 8051 音樂播放器完整程式 (含完整選歌、播放、刪除號碼、卡歌功能)

;==========================;
; 系統進入點與中斷向量設定 ;
;==========================;

ORG 0000H
AJMP MAIN                ; 開機後直接跳至 MAIN 執行主程式

;ORG 0003H
;AJMP EX0_ISR            ; INT0 外部中斷入口（卡歌功能，目前已註解）

ORG 0050H                ; 程式碼從 0050H 開始編寫

;==========================;
;          MAIN            ;
;==========================;

MAIN:
    ;-----------------------------------------------------
    ; 1) 堆疊與 I/O 初始化
    ;-----------------------------------------------------
    MOV SP, #5FH                ; 設定堆疊指標從 0x60 開始，避免與變數空間衝突

    MOV P0, #0FFH               ; 設定 P0 為全高電位（開汲極），需外接上拉電阻
    MOV P1, #0FFH               ; 設定 P1 為全高，用於矩陣式鍵盤掃描
    MOV P2, #0FFH               ; 設定 P2 為全高，用於 LCD 的 RS/E 和資料傳輸

    ;-----------------------------------------------------
    ; 2) 初始化 LCD 並顯示歡迎畫面
    ;-----------------------------------------------------
    CALL LCD_INIT               ; 初始化 LCD
    CALL LCD_SHOW_WELCOME       ; 顯示歡迎訊息
    CALL DELAY3SEC              ; 延遲 3 秒讓使用者看清楚畫面

    ;-----------------------------------------------------
    ; 3) 清除所有全域變數（輸入值與旗標）
```

```
    ;---------------------------------------------------
    MOV A, #00H
    MOV R0, #INPUT_NUM1
    MOV @R0, A                  ; INPUT_NUM1 ← 0
    INC R0
    MOV @R0, A                  ; INPUT_NUM2 ← 0
    INC R0
    MOV @R0, A                  ; INPUT_LEN ← 0（目前已輸入幾位）
    INC R0
    MOV @R0, A                  ; CANCEL_FLAG ← 0（卡歌旗標）
    INC R0
    MOV @R0, A                  ; SONG_CODE ← 0（歌曲編號）

    ;---------------------------------------------------
    ; 4) 中斷設定（暫時關閉 INT0）
    ;---------------------------------------------------
    ;SETB IT0                  ; 設定 INT0 為邊緣觸發（負緣）
    ;SETB EX0                   ; 開啟 INT0 中斷允許
    ;SETB EA                    ; 全域中斷允許（打開總開關）

    MOV TMOD, #00000000B        ; Timer 模式清除（預備音符播放用）
    SETB P3.0                  ; 設定 P3.0 為輸出腳（用來控制蜂鳴器）

    ;---------------------------------------------------
    ; 5) 顯示輸入畫面（提示使用者輸入歌曲代碼）
    ;---------------------------------------------------
    CALL LCD_SHOW_INPUT         ; 顯示 "Song:" 畫面與輸入數字

    ;---------------------------------------------------
    ; 6) 主迴圈：持續掃描鍵盤並執行對應指令
    ;---------------------------------------------------
MAIN_LOOP:
    CALL SCAN_KEY               ; 執行矩陣鍵盤掃描程序，結果儲存於 KEY_CODE

    MOV R0, #KEY_CODE
    MOV A, @R0                  ; 讀取鍵盤結果

    CJNE A, #'P', CHK_DEL       ; 若非播放鍵 'P'，跳至檢查刪除鍵
        CALL PLAY_HANDLER       ; 播放音樂
        SJMP AFTER_INPUT        ; 處理完回到主迴圈
```

```
CHK_DEL:
    CJNE A, #'D', CHK_NUM      ; 若非刪除鍵 'D'，跳至檢查是否為數字鍵
        CALL DELETE_HANDLER    ; 處理刪除輸入數字
        SJMP AFTER_INPUT

;CHK_CAN:                      ; 卡歌功能（目前註解）
;    CJNE A, #'C', CHK_NUM
;        MOV A, #01H
;        MOV R0, #CANCEL_FLAG
;        MOV @R0, A            ; 將 CANCEL_FLAG 設為 1（用於中斷播放）
;        SJMP AFTER_INPUT

CHK_NUM:
    CALL NUMBER_HANDLER        ; 若為數字鍵，處理輸入流程
    CALL DELAY100MS            ; 消除彈跳與多次觸發

AFTER_INPUT:
    SJMP MAIN_LOOP             ; 回到主迴圈持續等待使用者操作
```

鍵盤以及KEY_CODE賦值副程式：

```
;=========================;
; 鍵盤掃描模組 (4x4矩陣鍵盤) ;
;=========================;

;-------------------------;
; SCAN_KEY: 掃描整個鍵盤列與行
; 若偵測到有按鍵，會將對應的 ASCII 存入 KEY_CODE
; 若無任何按鍵，KEY_CODE ← 0FFH
;-------------------------;

SCAN_KEY:

ROW1:
    MOV P1, #7FH               ; 將 P1.7 置為 0（其餘為1）→ 啟用第1列（Row1）
    ACALL DEBOUNCE             ; 抖動處理
    MOV A, P1                  ; 讀取當前輸入
    ANL A, #0FH                ; 只保留低 4 bits（Column 資訊）
    MOV R1, #0                 ; 記錄 row_offset = 0
    CJNE A, #0FH, COL_SCAN     ; 若有任何 Column 被拉低，跳轉到 COL_SCAN

ROW2:
```

```
    MOV P1, #0BFH              ; 啟用 Row2（P1.6 = 0）
    ACALL DEBOUNCE
    MOV A, P1
    ANL A, #0FH
    MOV R1, #4                 ; row_offset = 4
    CJNE A, #0FH, COL_SCAN

ROW3:
    MOV P1, #0DFH              ; 啟用 Row3（P1.5 = 0）
    ACALL DEBOUNCE
    MOV A, P1
    ANL A, #0FH
    MOV R1, #8                 ; row_offset = 8
    CJNE A, #0FH, COL_SCAN

ROW4:
    MOV P1, #0EFH              ; 啟用 Row4（P1.4 = 0）
    ACALL DEBOUNCE
    MOV A, P1
    ANL A, #0FH
    MOV R1, #12                ; row_offset = 12
    CJNE A, #0FH, COL_SCAN

    ; 若四行都沒有按鍵，將 KEY_CODE 設為無效碼 0xFF
    MOV A, #0FFH
    MOV R0, #KEY_CODE
    MOV @R0, A
    RET

;-------------------------;
; COL_SCAN:
; 根據當前列的 Column 結果判定是哪個 Column
;-------------------------;

COL_SCAN:
    CJNE A, #0EH, COL2         ; 若為 1110B → Col0 被拉低
    MOV R0, #0                 ; col_offset = 0
    SJMP CALC_KEY

COL2:
    CJNE A, #0DH, COL3         ; 1101B → Col1 被拉低
```

```
    MOV R0, #1
    SJMP CALC_KEY

COL3:
    CJNE A, #0BH, COL4          ; 1011B → Col2 被拉低
    MOV R0, #2
    SJMP CALC_KEY

COL4:
    CJNE A, #07H, NO_KEY        ; 0111B → Col3 被拉低
    MOV R0, #3

;-------------------------;
; CALC_KEY:
; 根據 R1=row_offset 與 R0=col_offset
; 得到對應按鍵編號 (0~15)，再轉為 ASCII
;-------------------------;

CALC_KEY:
    MOV A, R1
    ADD A, R0                   ; A = row_offset + col_offset
    MOV R2, A                   ; R2 暫存為按鍵代號 (0~15)

    ; 開始轉換編號到 ASCII 字碼
    CJNE R2, #0, KEY_1
    MOV A, #'1'
    SJMP STORE_KEY

KEY_1: CJNE R2, #1, KEY_2
    MOV A, #'2'
    SJMP STORE_KEY

KEY_2: CJNE R2, #2, KEY_3
    MOV A, #'3'
    SJMP STORE_KEY

KEY_3: CJNE R2, #3, KEY_DEL
    MOV A, #'D'                 ; D = 刪除鍵
    SJMP STORE_KEY

KEY_DEL: CJNE R2, #4, KEY_5
```

```
        MOV A, #'4'
        SJMP STORE_KEY

KEY_5: CJNE R2, #5, KEY_6
        MOV A, #'5'
        SJMP STORE_KEY

KEY_6: CJNE R2, #6, KEY_PLAY
        MOV A, #'6'
        SJMP STORE_KEY

KEY_PLAY: CJNE R2, #7, KEY_7
        MOV A, #'P'                     ; P = 播放鍵
        SJMP STORE_KEY

KEY_7: CJNE R2, #8, KEY_8
        MOV A, #'7'
        SJMP STORE_KEY

KEY_8: CJNE R2, #9, KEY_9
        MOV A, #'8'
        SJMP STORE_KEY

KEY_9: CJNE R2, #10, KEY_CANCEL
        MOV A, #'9'
        SJMP STORE_KEY

KEY_CANCEL: CJNE R2, #12, KEY_0
        MOV A, #'C'                     ; C = 卡歌鍵（取消鍵）
        SJMP STORE_KEY

KEY_0: CJNE R2, #13, NO_KEY
        MOV A, #'0'

;-------------------------;
; 將轉換後結果寫入 KEY_CODE
;-------------------------;

STORE_KEY:
        MOV R0, #KEY_CODE
        MOV @R0, A
```

```
    RET

;-------------------------;
; DEBOUNCE:
; 鍵盤防彈跳延遲（雙層延遲迴圈）
;-------------------------;

DEBOUNCE:
    MOV R5, #100
DB1:
    MOV R6, #100
DB2:
    DJNZ R6, DB2
    DJNZ R5, DB1
    RET
```

數字輸入、刪除按鍵、播歌副程式：

```
;==========================;
; 數字輸入模組              ;
;==========================;

NUMBER_HANDLER:

    ;-------------------------------------------------
    ; 若目前沒有任何有效按鍵（KEY_CODE = 0FFH），直接返回
    ;-------------------------------------------------
    MOV R0, #KEY_CODE
    MOV A, @R0
    CJNE A, #0FFH, CHECK_LEN   ; 若不是 0xFF，表示有按鍵，繼續檢查輸入長度
    RET                        ; 否則直接返回（不處理）

CHECK_LEN:
    MOV R0, #INPUT_LEN
    MOV A, @R0
    CJNE A, #2, ADD_DIGIT      ; 若輸入長度未滿 2，允許新增
    RET                        ; 否則已輸入兩碼，不再接受新數字

ADD_DIGIT:
    MOV R0, #INPUT_LEN
    MOV A, @R0
    CJNE A, #0, ADD_SECOND     ; 若 INPUT_LEN 為 0，表示輸入的是第一碼
                               ; 否則跳去 ADD_SECOND，處理第二碼
```

```
    ;------------------------------------------------
    ; 處理第一碼：寫入 INPUT_NUM1
    ;------------------------------------------------
    MOV R0, #KEY_CODE
    MOV A, @R0                  ; 取得按鍵 ASCII 碼
    MOV R0, #INPUT_NUM1
    MOV @R0, A                  ; 寫入 INPUT_NUM1

    ; 增加 INPUT_LEN
    MOV R0, #INPUT_LEN
    MOV A, @R0
    INC A
    MOV @R0, A

    ACALL LCD_SHOW_INPUT        ; 更新 LCD 顯示目前輸入的內容
    RET

ADD_SECOND:
    ;------------------------------------------------
    ; 處理第二碼：寫入 INPUT_NUM2
    ;------------------------------------------------
    MOV R0, #KEY_CODE
    MOV A, @R0
    MOV R0, #INPUT_NUM2
    MOV @R0, A                  ; 寫入 INPUT_NUM2

    ; 增加 INPUT_LEN
    MOV R0, #INPUT_LEN
    MOV A, @R0
    INC A
    MOV @R0, A

    ACALL LCD_SHOW_INPUT        ; 更新 LCD 顯示
    RET


;==========================;
; 刪除鍵處理模組           ;
;==========================;
```

```
DELETE_HANDLER:

    ;------------------------------------------------
    ; 若 INPUT_LEN 為 0，表示沒有輸入任何數字，直接返回
    ;------------------------------------------------
    MOV R0, #INPUT_LEN
    MOV A, @R0
    CJNE A, #0, DELETE_EXIST   ; 若 >0，表示有東西可以刪
    RET

DELETE_EXIST:
    ;------------------------------------------------
    ; 減少 INPUT_LEN，並依照剩下的長度清除對應的變數
    ;------------------------------------------------
    MOV R0, #INPUT_LEN
    MOV A, @R0
    DEC A                          ; 減去一碼
    MOV @R0, A

    CJNE A, #0, DEL_SECOND      ; 若現在長度是 1，則只需清除第二碼
                                ; 若現在變成 0，則清除第一碼

    ; 清除 INPUT_NUM1（只剩下第一碼）
    MOV A, #0
    MOV R0, #INPUT_NUM1
    MOV @R0, A

    ACALL LCD_SHOW_INPUT
    RET

DEL_SECOND:
    ; 清除 INPUT_NUM2（原本已輸入兩碼，刪除後剩第一碼）
    MOV A, #0
    MOV R0, #INPUT_NUM2
    MOV @R0, A

    ACALL LCD_SHOW_INPUT
    RET
;=========================;
; 播放控制模組                ;
;=========================;
```

```
PLAY_HANDLER:

    ;------------------------------------------------
    ; 檢查是否有輸入任何數字
    ;------------------------------------------------
    MOV R0, #INPUT_LEN
    MOV A, @R0
    CJNE A, #0, CHECK_FIRST      ; 若 INPUT_LEN ≠ 0 則繼續往下判斷

    ; 若沒有輸入任何數字，顯示錯誤訊息 "No Number"
    ACALL LCD_CLEAR
    MOV DPTR, #NONUM_MSG
    ACALL LCD_PRINT
    CALL   DELAY3SEC

    ; 顯示提示畫面 "INPUT:"
    ACALL LCD_CLEAR
    MOV DPTR,   #INPUT_MSG
    ACALL LCD_PRINT
    CALL   DELAY3SEC
    RET                          ; 提早結束播放流程


CHECK_FIRST:
    ;------------------------------------------------
    ; 若輸入兩碼，需將兩碼組合成整數（十位與個位）
    ;------------------------------------------------
    CJNE A, #2, COMBINE_NUM      ; 若不是兩碼（只有一碼），跳去單碼處理

    ; 取 INPUT_NUM1 並轉為十位數（減去 '0' 轉成數值）
    MOV R0, #INPUT_NUM1
    MOV A, @R0
    SUBB A, #'0'

    MOV B, #10
    MUL AB                       ; A × 10 → A
    MOV R4, A                    ; 暫存十位數結果

    ; 取 INPUT_NUM2 並轉為數值
    MOV R0, #INPUT_NUM2
```

```
    MOV A, @R0
    CLR C
    SUBB A, #'0'

    ; 組合為兩位整數：十位 + 個位
    ADD A, R4
    SJMP CHECK_SONG                 ; 跳至歌曲判斷區

COMBINE_NUM:
    ;------------------------------------------------
    ; 處理只有一碼輸入的情況（直接轉成數值）
    ;------------------------------------------------
    MOV R0, #INPUT_NUM1
    MOV A, @R0
    CLR C
    SUBB A, #'0'



CHECK_SONG:
    ;------------------------------------------------
    ; A =  最終歌曲編號，寫入 SONG_CODE
    ;------------------------------------------------
    MOV R0, #SONG_CODE
    MOV @R0, A

    ; 根據歌曲代碼選擇播放對應歌曲

    ;------------------- SONG 1 -------------------
    CJNE A, #1, CHECK_SONG2
    ACALL LCD_SHOW_TITLE1          ; 顯示歌曲1標題

    ; 重設 CANCEL_FLAG = 0（尚未取消）
    MOV A, #0
    MOV R0, #CANCEL_FLAG
    MOV @R0, A

    ACALL PLAY_SONG1
    SJMP FINISH_PLAY

    ;------------------- SONG 2 -------------------
CHECK_SONG2:
```

```
    CJNE A, #2, CHECK_SONG3
    ACALL LCD_SHOW_TITLE2

    MOV A, #0
    MOV R0, #CANCEL_FLAG
    MOV @R0, A

    ACALL PLAY_SONG2
    SJMP FINISH_PLAY

    ;------------------- SONG 3 -------------------
CHECK_SONG3:
    CJNE A, #3, NO_SONG
    ACALL LCD_SHOW_TITLE3

    MOV A, #0
    MOV R0, #CANCEL_FLAG
    MOV @R0, A

    ACALL PLAY_SONG3
    SJMP FINISH_PLAY

NO_SONG:
    ; 若輸入的代碼不是已定義的歌曲（非 1, 2, 3）
    ACALL LCD_CLEAR
    MOV DPTR, #NOSONG_MSG
    ACALL LCD_PRINT
    CALL   DELAY3SEC
    ACALL WAIT_DELAY              ; 額外停頓，防止畫面閃爍

FINISH_PLAY:
    ;-----------------------------------------------
    ; 播放結束後清除輸入資料
    ;-----------------------------------------------
    MOV A, #0

    MOV R0, #INPUT_LEN
    MOV @R0, A

    MOV R0, #INPUT_NUM1
    MOV @R0, A
```

```
    MOV R0, #INPUT_NUM2
    MOV @R0, A

    ACALL LCD_SHOW_INPUT          ; 顯示回到 INPUT 輸入畫面
    RET


;==========================;
; 播放過程中檢查是否按下取消鍵 'C'
;==========================;
CHECK_CANCEL_KEY:
    ACALL SCAN_KEY                ; 重新掃描鍵盤
    MOV R0, #KEY_CODE
    MOV A, @R0
    CJNE A, #'C', EXIT_CHECK      ; 若非 'C' 鍵則離開
        MOV A, #1
        MOV R0, #CANCEL_FLAG
        MOV @R0, A                ; 將 CANCEL_FLAG 設為 1
EXIT_CHECK:
    RET


;==========================;
; 額外延遲子程序 (播放錯誤訊息後停頓)
;==========================;
WAIT_DELAY:
    MOV R7, #50
W1: MOV R6, #255
W2: DJNZ R6, W2
    DJNZ R7, W1
    RET
```

播歌副程式：

```
;==========================;
; 播放歌曲模組 (含卡歌功能);
;==========================;

;------------------------------------------------;
; 播放歌曲 1：小蜜蜂；
;------------------------------------------------;
```

```
PLAY_SONG1:
    MOV DPTR, #BEE_TABLE        ; 將歌曲資料表指標設為 BEE_TABLE
    MOV R7, #13                 ; 播放長度（音符數），共 13 個

SONG1_LOOP:
    ACALL CHECK_CANCEL_KEY      ; 每個音符前檢查是否按下取消鍵
    MOV R0, #CANCEL_FLAG
    MOV A, @R0
    CJNE A, #0, STOP_PLAY       ; 若 CANCEL_FLAG ≠ 0，跳至停止播放

    CLR A
    MOVC A, @A+DPTR             ; 從 BEE_TABLE 取出目前音符編號（透過 DPTR）
    INC DPTR                    ; 移到下一個音符位置
    CJNE A, #255, SONG1_PLAY    ; 若還沒遇到結尾標記 255，繼續播放
    RET                         ; 否則代表播放完畢，結束

SONG1_PLAY:
    ACALL PLAY_NOTE             ; 播放對應音符（A 內含音符編號）
    ACALL NOTE_DELAY            ; 音符播放之間的延遲
    DJNZ R7, SONG1_LOOP         ; 播完一個音後遞減計數器，若未為0，繼續播下一音
    RET



;-----------------------------------------------;
; 播放歌曲 2：小星星；
;-----------------------------------------------;
PLAY_SONG2:
    MOV DPTR, #STAR_TABLE       ; 將歌曲資料表指標設為 STAR_TABLE
    MOV R7, #14                 ; 小星星音符數為 14

SONG2_LOOP:
    ACALL CHECK_CANCEL_KEY
    MOV R0, #CANCEL_FLAG
    MOV A, @R0
    CJNE A, #0, STOP_PLAY

    CLR A
    MOVC A, @A+DPTR
    INC DPTR
    CJNE A, #255, SONG2_PLAY
    RET
```

```
SONG2_PLAY:
    ACALL PLAY_NOTE
    ACALL NOTE_DELAY
    DJNZ R7, SONG2_LOOP
    RET


;------------------------------------------------;
; 播放歌曲 3：天空之城；
;------------------------------------------------;
PLAY_SONG3:
    MOV DPTR, #SKY_TABLE        ; 設定 DPTR 為天空之城的音符表
    MOV R7, #100                ; 最大播放音符數暫設為 100

SONG3_LOOP:
    ACALL CHECK_CANCEL_KEY
    MOV R0, #CANCEL_FLAG
    MOV A, @R0
    CJNE A, #0, STOP_PLAY

    CLR A
    MOVC A, @A+DPTR
    INC DPTR
    CJNE A, #255, SONG3_PLAY
    RET

SONG3_PLAY:
    ACALL PLAY_NOTE
    ACALL NOTE_DELAY
    DJNZ R7, SONG3_LOOP
    RET


;------------------------------------------------;
; 停止播放歌曲（當 CANCEL_FLAG 被設定）；
;------------------------------------------------;
STOP_PLAY:
    RET
```

LCD模組相關副程式：

```
;=========================;
; LCD初始化與顯示模組          ;
;=========================;

; --- LCD 腳位定義 ---
LCDBUS   EQU P0              ; 資料匯流排 DB0~DB7 → P0.0~P0.7
RS       EQU P2.6            ; 資料/指令選擇：RS=1為資料、RS=0為指令
ENABLE   EQU P2.7            ; 啟用訊號 E：由高變低觸發 LCD 接收
; ※ R/W 腳需直接接 GND（僅寫入模式）

; --- LCD 初始化流程 ---
LCD_INIT:
    ; LCD 上電穩定時間，需延遲 >15ms
    ACALL POWER_ON_DELAY

    ; Function Set (38H = 8-bit, 2-line, 5x8 dots)
    MOV A,#00111000B
    ACALL LCD_WRIR           ; 傳送 Function Set 指令
    ACALL DELAY5MS           ; 等待 LCD 處理

    ; 顯示設定：開啟顯示、不顯示游標、不閃爍
    MOV A,#00001100B
    ACALL LCD_WRIR
    ACALL DELAY5MS

    ; 清除顯示畫面
    MOV A,#00000001B
    ACALL LCD_WRIR
    ACALL DELAY5MS

    ; 輸入模式設定：游標自動右移、畫面不移動
    MOV A,#00000110B
    ACALL LCD_WRIR
    ACALL DELAY5MS

    ; 設定 DDRAM 起始地址為 0（第一列開頭）
    MOV A,#10000000B
    CALL LCD_WRIR
    CALL DELAY5MS
```

```
    ; 回歸原點指令
    MOV A,#00000010B
    CALL LCD_WRIR
    CALL DELAY5MS
    RET

; --- 清除畫面 ---
LCD_CLEAR:
    MOV A,#00000001B          ; Clear display 指令
    ACALL LCD_WRIR
    ACALL DELAY5MS
    RET

; --- 顯示開機歡迎訊息 ---
LCD_SHOW_WELCOME:
    ACALL LCD_CLEAR
    MOV DPTR,#WELCOME_MSG   ; 指向歡迎字串陣列
    ACALL LCD_PRINT           ; 顯示整段字串
    RET

; --- 顯示輸入提示與目前輸入的號碼 ---
LCD_SHOW_INPUT:
    ACALL LCD_CLEAR
    MOV DPTR, #INPUT_MSG    ; 顯示 "SONG:" 字串
    ACALL LCD_PRINT

    MOV R0, #INPUT_LEN      ; 判斷目前輸入長度
    MOV A, @R0
    CJNE A, #0, PRINT_NUM1 ; 若輸入數為 0，直接跳出
    RET

PRINT_NUM1:
    MOV R0, #INPUT_NUM1
    MOV A, @R0
    ACALL LCD_WRDR            ; 顯示第一位數（ASCII 碼）

    MOV R0, #INPUT_LEN
    MOV A, @R0
    CJNE A, #2, LSI_EXIT    ; 若長度 ≠ 2，表示第二位尚未輸入
    MOV R0, #INPUT_NUM2
    MOV A, @R0
```

```
    ACALL LCD_WRDR          ; 顯示第二位數
LSI_EXIT:
    RET

; --- 顯示三首歌對應標題 ---
LCD_SHOW_TITLE1:
    ACALL LCD_CLEAR
    MOV DPTR,#SONG1_MSG
    ACALL LCD_PRINT
    RET

LCD_SHOW_TITLE2:
    ACALL LCD_CLEAR
    MOV DPTR,#SONG2_MSG
    ACALL LCD_PRINT
    RET

LCD_SHOW_TITLE3:
    ACALL LCD_CLEAR
    MOV DPTR,#SONG3_MSG
    ACALL LCD_PRINT
    RET

; --- LCD 顯示字串副程式（以 0 為終止）---
LCD_PRINT:
    CLR A
    MOVC A,@A+DPTR          ; 取出字元
    JZ  LP_END              ; 若遇到 0 結束符號，跳出
    ACALL LCD_WRDR          ; 顯示字元
    ACALL DELAY5MS
    INC DPTR
    SJMP LCD_PRINT
LP_END:
    RET

; ===== 寫入 LCD 指令 =====
LCD_WRIR:
    CLR  RS                 ; 指令模式
    SETB ENABLE             ; E 拉高
    MOV  LCDBUS, A          ; 指令寫入匯流排
    ACALL DELAY1MS
```

```
    ACALL DELAY1MS
    MOV R7, #50
WRIR_WAIT1:
    DJNZ R7, WRIR_WAIT1
    CLR ENABLE              ; E 拉低觸發 LCD 接收
    CALL DELAY1MS
    MOV R7, #25
WRIR_WAIT2:
    DJNZ R7, WRIR_WAIT2
    RET

; ===== 寫入 LCD 資料 =====
LCD_WRDR:
    SETB RS                 ; 資料模式
    SETB ENABLE
    MOV LCDBUS, A           ; 資料寫入匯流排
    ACALL DELAY1MS
    ACALL DELAY1MS
    MOV R7, #50
WRDR_WAIT1:
    DJNZ R7, WRDR_WAIT1
    CLR ENABLE              ; E 拉低觸發
    CALL DELAY1MS
    MOV R7, #25
WRDR_WAIT2:
    DJNZ R7, WRDR_WAIT2
    RET

; ===== 各種延遲副程式 =====
POWER_ON_DELAY:
    MOV R7,#255
POD_L1:
    MOV R6,#255
POD_L2:
    DJNZ R6,POD_L2
    DJNZ R7,POD_L1
    RET

DELAY5MS:
    ACALL DELAY1MS
    ACALL DELAY1MS
```

```
    ACALL DELAY1MS
    ACALL DELAY1MS
    ACALL DELAY1MS
    RET

DELAY1MS:
    MOV R6,#109
D1_L1:
    MOV R7,#26
D1_L2:
    DJNZ R7,D1_L2
    DJNZ R6,D1_L1
    RET
```

音符播放副程式：

```
;=========================;
; 音符播放模組
;=========================;

;-----------------------------------------------------------
; PLAY_NOTE：接收音符編號 (0~13)，並對應播放指定音符
;-----------------------------------------------------------
PLAY_NOTE:
    CJNE A, #0, NOTE_1
    ACALL PLAY_LSO        ; 低音 So
    RET
NOTE_1: CJNE A, #1, NOTE_2
    ACALL PLAY_LLA        ; 低音 La
    RET
NOTE_2: CJNE A, #2, NOTE_3
    ACALL PLAY_LSI        ; 低音 Si
    RET
NOTE_3: CJNE A, #3, NOTE_4
    ACALL PLAY_DO         ; 中音 Do
    RET
NOTE_4: CJNE A, #4, NOTE_5
    ACALL PLAY_RE         ; 中音 Re
    RET
NOTE_5: CJNE A, #5, NOTE_6
    ACALL PLAY_MI         ; 中音 Mi
    RET
```

```
NOTE_6: CJNE A, #6, NOTE_7
        ACALL PLAY_FA          ; 中音 Fa
        RET
NOTE_7: CJNE A, #7, NOTE_8
        ACALL PLAY_SO          ; 中音 So
        RET
NOTE_8: CJNE A, #8, NOTE_9
        ACALL PLAY_LA          ; 中音 La
        RET
NOTE_9: CJNE A, #9, NOTE_10
        ACALL PLAY_SI          ; 中音 Si
        RET
NOTE_10: CJNE A, #10, NOTE_11
        ACALL PLAY_HDO         ; 高音 Do
        RET
NOTE_11: CJNE A, #11, NOTE_12
        ACALL PLAY_HRE         ; 高音 Re
        RET
NOTE_12: CJNE A, #12, NOTE_13
        ACALL PLAY_HMI         ; 高音 Mi
        RET
NOTE_13:
        ACALL PLAY_SFA         ; 升音 Fa（新加）
        RET

;-----------------------------------------------------------
; 每個音符使用 Timer0 + P3.0 方波輸出播放，約 440~1000Hz
;-----------------------------------------------------------

; 範例：低音 So
PLAY_LSO:
        MOV TL0, #194
        MOV TH0, #230
        MOV R3, #3             ; 外圈：3次循環
        SETB TR0
PLSO_LOOP: MOV R2, #200 ; 內圈：200次方波輸出
PLSO_LOOP1: ACALL INVERT_LSO
        DJNZ R2, PLSO_LOOP1
        DJNZ R3, PLSO_LOOP
        CLR TR0
        RET
```

```
INVERT_LSO:
    CPL P3.0              ; 反轉蜂鳴器輸出
    JNB TF0, $            ; 等待 Timer0 溢位
    CLR TF0
    MOV TL0, #194         ; 重設 Timer 值
    MOV TH0, #230
    RET

; 其餘音符與此邏輯相同

; 低 La (LLa)
PLAY_LLA:
    MOV TL0, #249
    MOV TH0, #228
    MOV R3, #3
    SETB TR0
PLLA_LOOP: MOV R2, #200
PLLA_LOOP1: ACALL INVERT_LLA
    DJNZ R2, PLLA_LOOP1
    DJNZ R3, PLLA_LOOP
    CLR TR0
    RET

PLAY_DO:
    MOV TL0, #12
    MOV TH0, #196
    MOV R3, #3
    SETB TR0
PDLOOP: MOV R2, #130
PDLOOP1: ACALL INVERTDO
        DJNZ R2, PDLOOP1
        DJNZ R3, PDLOOP
        CLR TR0
        RET

PLAY_RE:
    MOV TL0, #28
    MOV TH0, #202
    MOV R3, #3
    SETB TR0
```

```
PRLOOP: MOV R2, #196
PRLOOP1: ACALL INVERTRE
        DJNZ R2, PRLOOP1
        DJNZ R3, PRLOOP
        CLR TR0
        RET

PLAY_MI:
    MOV TL0, #21
    MOV TH0, #208
    MOV R3, #3
    SETB TR0
PMLOOP: MOV R2, #220
PMLOOP1: ACALL INVERTMI
        DJNZ R2, PMLOOP1
        DJNZ R3, PMLOOP
        CLR TR0
        RET

PLAY_FA:
    MOV TL0, #8
    MOV TH0, #211
    MOV R3, #3
    SETB TR0
PFLOOP: MOV R2, #233
PFLOOP1: ACALL INVERTFA
        DJNZ R2, PFLOOP1
        DJNZ R3, PFLOOP
        CLR TR0
        RET

PLAY_SO:
    MOV TL0, #5
    MOV TH0, #216
    MOV R3, #4
    SETB TR0
PSLOOP: MOV R2, #196
PSLOOP1: ACALL INVERTSO
        DJNZ R2, PSLOOP1
        DJNZ R3, PSLOOP
        CLR TR0
```

```
        RET

PLAY_LA:
    MOV TL0, #16
    MOV TH0, #220
    MOV R3, #4
    SETB TR0
PLLOOP: MOV R2, #220
PLLOOP1: ACALL INVERTLA
        DJNZ R2, PLLOOP1
        DJNZ R3, PLLOOP
        CLR TR0
        RET

PLAY_SI:
    MOV TL0, #12
    MOV TH0, #224
    MOV R3, #4
    SETB TR0
PSILOOP: MOV R2, #247
PSILOOP1: ACALL INVERTSI
         DJNZ R2, PSILOOP1
         DJNZ R3, PSILOOP
         CLR TR0
         RET

; 高音 Do (HDo)
PLAY_HDO:
    MOV TL0, #247
    MOV TH0, #225
    MOV R3, #4
    SETB TR0
PHDO_LOOP: MOV R2, #220
PHDO_LOOP1: ACALL INVERT_HDO
    DJNZ R2, PHDO_LOOP1
    DJNZ R3, PHDO_LOOP
    CLR TR0
    RET

; 高音 Re (HRe)
PLAY_HRE:
```

```
    MOV TL0, #66
    MOV TH0, #229
    MOV R3, #4
    SETB TR0
PHRE_LOOP: MOV R2, #220
PHRE_LOOP1: ACALL INVERT_HRE
    DJNZ R2, PHRE_LOOP1
    DJNZ R3, PHRE_LOOP
    CLR TR0
    RET

; 高音 Mi
PLAY_HMI:
    MOV TL0, #46
    MOV TH0, #232
    MOV R3, #4
    SETB TR0
HMI_LOOP: MOV R2, #220
HMI_LOOP1: ACALL INVERT_HMI
    DJNZ R2, HMI_LOOP1
    DJNZ R3, HMI_LOOP
    CLR TR0
    RET

; 升 Fa 音（index = 13）
PLAY_SFA:
    MOV TL0, #0
    MOV TH0, #212
    MOV R3, #3
    SETB TR0
SFA_LOOP: MOV R2, #220
SFA_LOOP1: ACALL INVERT_SFA
    DJNZ R2, SFA_LOOP1
    DJNZ R3, SFA_LOOP
    CLR TR0
    RET

INVERT_LLA:
    CPL P3.0
    JNB TF0, $
    CLR TF0
```

```
        MOV TL0, #249
        MOV TH0, #228
        RET

; 低 Si (LSi)
PLAY_LSI:
        MOV TL0, #224
        MOV TH0, #227
        MOV R3, #3
        SETB TR0
PLSI_LOOP: MOV R2, #200
PLSI_LOOP1: ACALL INVERT_LSI
        DJNZ R2, PLSI_LOOP1
        DJNZ R3, PLSI_LOOP
        CLR TR0
        RET
;=========== 各音階方波產生 ===========
INVERT_LSI:
        CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #224
        MOV TH0, #227
        RET

INVERTDO: CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #12
        MOV TH0, #196
        RET

INVERTRE: CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #28
        MOV TH0, #202
        RET

INVERTMI: CPL P3.0
        JNB TF0, $
```

```
        CLR TF0
        MOV TL0, #21
        MOV TH0, #208
        RET

INVERTFA: CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #8
        MOV TH0, #211
        RET

INVERTSO: CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #5
        MOV TH0, #216
        RET

INVERTLA: CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #16
        MOV TH0, #220
        RET

INVERTSI: CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #12
        MOV TH0, #224
        RET

INVERT_HDO:
        CPL P3.0
        JNB TF0, $
        CLR TF0
        MOV TL0, #247
        MOV TH0, #225
        RET
```

```
INVERT_HRE:
    CPL P3.0
    JNB TF0, $
    CLR TF0
    MOV TL0, #66
    MOV TH0, #229
    RET

INVERT_HMI: CPL P3.0
    JNB TF0, $
    CLR TF0
    MOV TL0, #46
    MOV TH0, #232
    RET

INVERT_SFA: CPL P3.0
    JNB TF0, $
    CLR TF0
    MOV TL0, #0
    MOV TH0, #212
    RET
;-------------------------------------------------------
; 音符播放後的停頓（延遲），防止連音
;-------------------------------------------------------
NOTE_DELAY:
    MOV R5, #100
ND1: MOV R6, #100
ND2: DJNZ R6, ND2
    DJNZ R5, ND1
    RET
```

變數、TABLE與延遲相關副程式：

```
;---------------------------
; 延遲約 20ms：呼叫 4 次 DELAY5MS
;---------------------------
DELAY20MS:
    ACALL DELAY5MS
    ACALL DELAY5MS
    ACALL DELAY5MS
    ACALL DELAY5MS
    RET
```

```
;---------------------------
; 延遲約 100ms：呼叫 5 次 DELAY20MS
;---------------------------
DELAY100MS:
    ACALL DELAY20MS
    ACALL DELAY20MS
    ACALL DELAY20MS
    ACALL DELAY20MS
    ACALL DELAY20MS
    RET

;---------------------------
; 延遲約 1 秒：呼叫 10 次 DELAY100MS
;---------------------------
DELAY1SEC:
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    ACALL DELAY100MS
    RET

;---------------------------
; 延遲約 3 秒：呼叫 3 次 DELAY1SEC
;---------------------------
DELAY3SEC:
    ACALL DELAY1SEC
    ACALL DELAY1SEC
    ACALL DELAY1SEC
    RET
;-----------------------------------
; 小蜜蜂 BEE_TABLE：共 13 音符 + 結束
;-----------------------------------
BEE_TABLE:
    DB 7,5,5,6,4,4,3,4,5,6,7,7,7,255
```

```
;-----------------------------------
; 小星星 STAR_TABLE：共 14 音符 + 結束
;-----------------------------------
STAR_TABLE:
    DB 3,3,7,7,8,8,7,6,6,5,5,4,4,3,255

;-----------------------------------
; Sky Castle 長曲：多段旋律，結尾 255 表結束
;-----------------------------------
SKY_TABLE:
    DB 8,9,10, 9,10,12, 9
    DB 5,8, 7,8,10, 7
    DB 5,6, 5,6,10, 5
    DB 10,9, 13,13,9, 9
    DB 8,9,10, 9,10,12, 9
    DB 5,8, 7,8,10, 7
    DB 5,6,10,9,10,11,12,10
    DB 10,9,8,9,7,8
    DB 255

WELCOME_MSG: DB "Music Player", 0      ; 開機歡迎畫面
INPUT_MSG:   DB "Song:", 0             ; 輸入提示
NONUM_MSG:   DB "Enter Num Plz", 0     ; 無輸入提示
SONG1_MSG:   DB "Little Bee", 0        ; 小蜜蜂歌名
SONG2_MSG:   DB "Twinkle Star", 0      ; 小星星歌名
SONG3_MSG:   DB "Sky Castle", 0        ; Sky Castle 歌名
NOSONG_MSG:  DB "No Song", 0           ; 輸入代碼無對應歌曲

ORG 0020H              ; RAM 記憶體區段起始位址
KEY_CODE:      DS 1 ; 鍵盤讀入的 ASCII 值
INPUT_NUM1:    DS 1 ; 輸入第一位數
INPUT_NUM2:    DS 1 ; 輸入第二位數
INPUT_LEN:     DS 1 ; 目前輸入幾碼 (0,1,2)
CANCEL_FLAG: DS 1 ; 卡歌中斷旗標 (1=中止播放)
SONG_CODE:     DS 1 ; 最終轉換後的歌曲代碼 (0~n)
```

六、心得：

期末專題心得感想：

在這次的期末專題中，我設計並實作一套具備選曲與播放功能的音樂播放器。這個專題由我自行發想與規劃，從系統功能設計、程式架構建立、硬體接線配置，到最終的實作整合，過程中不僅驗證了我整學期的學習成果，也讓我學會如何獨立完成一個具備互動界面與多項功能的嵌入式應用系統。

專題的核心功能包括：數字鍵盤輸入選曲代碼、LCD 顯示當前輸入與播放資訊、播放不同旋律的歌曲（如《小蜜蜂》、《小星星》、《天空之城》等），並具備刪除數字、取消播放（卡歌）等操作功能。這些功能的實現涉及了 Timer 計時器的準確控制、LCD 指令流程、鍵盤矩陣掃描、音符資料表解析與聲音播放模組設計等多項技術。

一開始遇到最大的挑戰是音階對應的 Timer 設定，由於每一個音符頻率不同，我需要精確計算 TH0/TL0 的初值，並搭配適當的 delay 迴圈來模擬長短音的播放效果。中間也遇到過 LCD 顯示亂碼、鍵盤輸入無反應等問題，但這些 Debug 過程讓我養成了更縝密的檢查與除錯習慣，也學會從程式與電路雙方面檢查原因。

此外，我也特別設計了模組化的程式架構，將每個功能區塊如：鍵盤掃描、數字處理、播放邏輯、LCD 顯示、音符播放等獨立出來，這讓程式碼更有可讀性與維護性。實作過程中我反覆測試各種邏輯狀況，包括歌曲播放中途按下取消、輸入錯誤數字後刪除等情境，確保整體流程穩定且使用者友善。

這次的專題不僅讓我更熟悉 8051 系統的使用，更重要的是讓我體驗到從「功能需求」轉化為「具體實作」的過程，也讓我首次真正感受到做出一個可以實際使用的產品的成就感。

Notes:

1. 內容字體大小為 12，間距為單行間距
2. 中文字字體為標楷體
3. 英文字和阿拉伯數字為 Times New Roman
4. 嚴禁抄襲，抄襲者以 0 分計算
5. 請於報告左上角附上照片
6. 每次實驗課繳交上次實驗結報