九齊科技股份有限公司
Nyquest Technology Co., Ltd.

使用手冊

NY7 系列 *Q-Code* 範例

PowerSpeech Format Programmer

Version 1.0
May 27, 2014

改 版 記 錄

版本	日期	內 容 描 述	修正頁
1.0	2014/5/27	新發佈。	-

目　錄

1 簡介

隨著科技的進步，電子產品在不斷更新。在這種充滿競爭的社會，九齊科技始終秉持著誠信、精確、效率的經營理念，爲客戶提供高品質及高附加價值的語音 IC，並提供優質的服務。爲使客戶能夠更方便快捷的使用九齊 IC，九齊科技針對 NY7 系列 IC 開發了一套軟體工具 — *Q-Code NY7*。透過 *Q-Code NY7*，初級工程師只需要進行簡單的培訓，在短時間內就可以完成某一產品的開發。本篇將介紹實際應用中的一些基本功能，以及相關的範例，讓初學者能夠大概瞭解到 *Q-Code NY7* 的基本功能和應用。

1.1 內容

1 簡介

第一章主要介紹 *Q-Code NY7*，以及要用到的相關軟體和硬體工具。

2 Q-Code NY7 基本功能介紹

第二章主要是 *Q-Code NY7* 基本功能介紹（如何實現按鍵播放 Voice 或 Midi）。

3 應用實例

第三章主要是 *Q-Code NY7* 的應用實例，實例中結合了 *Q-Code NY7* 常用到的一些功能，還有一些特殊功能的應用，例如：用暫存器播放聲音、ADSR 功能、Action 功能等。

1.2 何謂*Q-Code NY7*？

Q-Code NY7 爲九齊科技股份有限公司所提供，它是針對九齊科技的 NY7 系列而開發的軟體工具。它提供簡易的圖形處理介面來完成應用程式的開發。無需瞭解硬體結構和組合語言的編寫，工程師依然可以利用 *Q-Code NY7* 強大的功能來完成應用程式的開發，簡化了產品開發流程，提高了產品開發效率。對於初級工程師只需要進行簡單的培訓，在短時間內就可以完成某一產品的開發。

您可以聯繫九齊科技來獲得 *Q-Code NY7* 的安裝程式檔，按兩下執行檔後進入程式安裝嚮導，然後依照畫面指示就可輕鬆完成安裝。

1.3 相關軟體工具

在使用 *Q-Code NY7* 編寫程式時，我們不僅僅只要用到這一個工具，我們還需要用到幾個相關的工具。例如，當我們在編寫完程式時，就需要編譯產生一個 .bin 和 .htm 檔(這是投 Code 的重要檔)，此時，我們就需要用到 *NYASM*，它應該在安裝 *Q-Code NY7* 時同步安裝。

在使用 *Q-Code* 時還可能用到以下一些相關工具：

Q-Midi：此軟體包含了音色、包絡編輯、音色自動分析、即時音色模擬等功能，可以快速、輕鬆地利用滑鼠進行包絡的編輯、完成音色檔與包絡的合成，而完成使用者專屬的音色庫.idb2，另外提供轉換.mid 和音色檔成爲.md3。

Q-Writer：此軟體是用來將 .bin 檔燒錄在 Flash Demo Board 或者 NY_Romter 上以供驗證。

Q-Visio：此軟體是信號編輯軟體，可以產生 Vio 和 Vix 格式的檔，其中 Vio 檔可以添加到 *Q-Code_NY7* 的 Action 區段中，*Q-Code_NY7* 自動將信號編碼轉化成資料表供自己使用。

注意：在安裝 Q-Code NY7 時，使用者最好是將用到的相關工具都同步安裝，以上工具的使用，請參考相關的使用手冊。

1.4 相關硬體工具

當完成程式編寫並編譯後，使用者可以將程式下載至 ICE 上進行模擬，或者透過燒錄器 (*Q-Writer*) 燒錄至 Demo 板上進行演示。

何謂 ICE？ICE 是 In Circuit Emulator 的縮寫，中文叫模擬器。使用者只需要將 ICE 透過 USB 連接至個人電腦，便可以將編譯好的程式下載至 ICE 進行模擬。(有關 ICE 硬體安裝請參考 *NYIDE* 使用手冊)

何謂*Q-Writer*？*Q-Writer* 是一個圖形介面的燒錄系統，讓使用者能夠快速的將程式產生的.bin 檔燒錄至FDB（Flash Demo Board）演示板中，或者下載到Romter 驗證，也可直接透過*Q-Writer* 直接燒錄至OTP 來進行驗證。使用者可以用FDB_Writer 燒錄，也可以用*Q-Writer* 燒錄（有關*Q-Writer* 的軟體和硬體安裝及使用，請參考*Q-Writer* 使用手冊）

FDB_Writer：如下圖所示。

Q-FDB_Writer：如下圖所示。

NY7_Romter 如下圖所示。

2 基本功能介紹

2.1 基本 Option 選項設定與 I/O Pin 設定

2.1.1 基本 Option 選項設定

ICBody：根據實際應用選擇適當的 ICBody。

例. ICBody = NY7A065A

Client：必須填入客戶名稱，否則編譯將不能透過。（此舉是用來保護程式開發者的所有權益）

例. Client = Nyquest

Voltage：選擇IC實際應用的工作電壓，一般爲3.0V或是4.5V。(頻率參考電壓)

例. Voltage = 3.0V

例. Voltage = 4.5V

System Clock：系統頻率設定，Clock=2 MHz / 4 MHz。（Default = 4MHz）

例. Clock = 4MHz

Voice Output：NY7A系列支援PWM+DAC，NY7B/NY7C系列支援DAC與Push-Pull。

例. Voice Output = PWM

例. Voice Output = DAC

例. Voice Output = PWM+DAC

例. Voice Output = Push-Pull

Reset：外部強制復位IC的腳位，Reset pin可選擇有Pull-High電阻或是Floating。（Default = Disable）

例. Reset = Pull-High

IR：IR即紅外線發射及接收功能，系統已預設特定的I/O作爲發射或接收腳，但需使用者指定IR編碼資料的長度與IR載波頻率。

IR Mode：選擇TX or RX 腳位。

IR Data Length：選擇IR編碼長度。

IR Frequency：選擇IR載波頻率。

IR Carry：選擇由NPN或者PNP電晶體來驅動IR LED。

例.

IR_Mode = TX+RX

IR_Bit = 8

IR_Freq = 38.5K

IR_Carry = High

FD：Flash with Dynamic 隨著音量的大小來閃動。可以選擇大於1/2或者是3/4音量來閃動，並且可以設定輸出型態爲Drive或是Sink的方式。（D爲Drive，S爲Sink。Default = Disable）

例. FD = 1/2 Drive

例. FD = 3/4 Sink

Power On Trigger：可以選擇是否要有Power on Trigger功能。（Default = Enable）

例. PowerOnTrigger = Disable

Debounce：使用者可以選擇按鍵的debounce時間，時間範圍：1 ~ 1000ms。（Default = 16ms）

例. Debounce = 16 ms

Interrupt Service：Action、Melody及Timer 3 種中斷服務。

例. InterruptService = Action

例. InterruptService = Melody

例. InterruptService = Timer

Time Base：TimeBase = 0.128/0.256/1.024 ms 。（Default = 0.256ms）

例. TimeBase = 0.256ms

Fade Out Location：樂曲音符淡出（Fade Out Location）是防止聲音突然結束而產生雜音的機制，可設定範圍值1~15。（Default = 5）

例. Fade Out Location = 5

2.1.2 I/O Pin 設定

Input 模式：

I/O 狀態	Input 模式			預設值
Mode	P	F	PF	P
State	LT	HT		LT

P：有上拉電阻的輸入模式（Input mode with Pull-High Resistor）。

F：無上拉電阻的輸入模式（Input mode with Floating）。

PF：上拉電阻控制模式（Pull-High Resistor Control）。設定該模式後，上拉電阻可由IO暫存器來控制打開或是關閉。

LT：低電位觸發（Low Trigger）。

HT：高電位觸發（High Trigger）。

注意：PF：LT 模式下僅能接受低電位電壓來喚醒 IC。

Input State：

格式	按鍵觸發型態	注 釋
X	忽略	無動作。
Path1	下降沿	當輸入腳從高電位到低電位時，執行Path1路徑。
/Path2	上升沿	當輸入腳從低電位到高電位時，執行Path2路徑。
Path1/Path2	下降沿&上升沿	當輸入腳從高電位到低電位時，執行Path1路徑。 當輸入腳從低電位到高電位時，執行Path2路徑。

Output 模式：

I/O 狀態	Output 模式			預設值
Current	N	LS	CS	N
State	L	H		L

N：一般電流輸出（Normal current output）。

LS：大流入電流（Large Sink current output）。

CS：定流入電流（Constant Sink current output）。

L：初始輸出電壓爲低電位（Initial Low）。

H：初始輸出電壓爲高電位（Initial High）。

Output State：

輸出值	注　釋
X	表示該腳位維持原來的電位。
0	表示該腳位輸出低電位。
1	表示該腳位輸出高電位。
FD	表示隨播放的聲音音量大小變化(Flash with Dynamic)。
An	表示該腳位所對應到的ActionLabel爲哪一個（該功能僅對於播放整個VIO檔有效）。

2.2 按鍵觸發模式

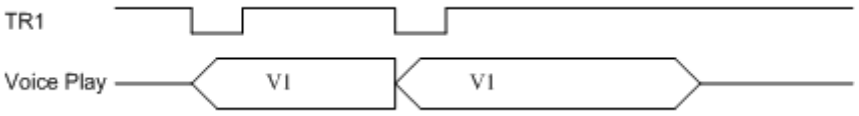
按鍵觸發模式即 Trigger Mode。在 *Q-Code NY7* 中，使用者可以對按鍵進行不同的操作來實現不同的功能。此處，我們將介紹 7 組按鍵觸發模式。

觸發模式縮寫	觸發模式	說明
E/U/R	Edge mode，Unhold，Retrigger	邊沿觸發，非保持，可重複觸發。
E/U/I	Edge mode，Unhold，Irretrigger	邊沿觸發，非保持，不可重複觸發。
L/U/R	Level mode，Unhold，Retrigger	電位觸發，非保持，可重複觸發。
L/U/I	Level mode，Unhold，Irretrigger	電位觸發，非保持，不可重複觸發。
E/H/X	Edge mode，Hold，X	邊沿觸發，保持。
L/H/X	Level mode，Hold，X	電位觸發，保持。
Toggle On/Off	Toggle On/Off	邊沿觸發，ON/OFF 鍵。

2.2.1 Edge mode, Unhold, Retrigger

Edge mode，Unhold，Retrigger（縮寫爲：E/U/R），即：邊沿觸發，非保持，可重複觸發。

功能：壓下按鍵後開始播放歌曲，使用者可以在歌曲播放中再 trigger，每一次 trigger 只會播放一次歌曲。



例. PA0 (E/U/R) = V1

[Input State]

;	PA0	PA1	PA2	PA3
TR1_Enable	TR1	**X**	**X**	**X**

[Path]

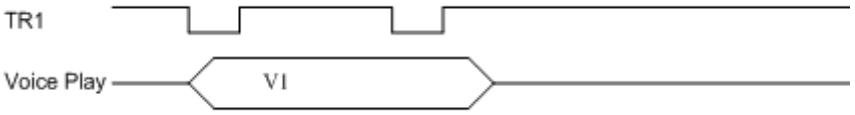
PowerOn: **TR1_Enable**, **IC_Init**

TR1: **PlayV**(CH0,$V1), **IC_Sleep**

2.2.2 Edge mode, Unhold, Irretrigger

Edge mode，Unhold，Irretrigger（縮寫爲：E/U/I），即：邊沿觸發，非保持，不可重複觸發。

功能：壓下按鍵後開始播放歌曲，使用者不可以在第一次歌曲播放中再 trigger，必須等到第一次播放的歌曲播放完畢後才能夠再 trigger，每一次 trigger 只會播放一次歌曲。



例. PA0 (E/U/I) = V1

[Input State]

;	PA0	PA1	PA2	PA3
TR1_Enable:	TR1	**X**	**X**	**X**
TR1_Disable:	**X**	**X**	**X**	**X**

[Path]

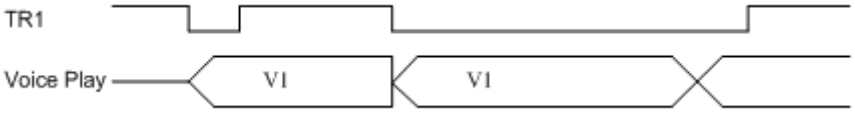
PowerOn: **TR1_Enable**, **IC_Init**

TR1: **TR1_Disable**, **PlayV**(CH0,$V1), **TR1_Enable**, **IC_Sleep**

2.2.3 Level mode, Unhold, Retrigger

Level mode，Unhold，Retrigger（縮寫爲：L/U/R），即：電位觸發，非保持，可重複觸發。

功能：壓下按鍵後開始播放歌曲，使用者可以在歌曲播放中再 trigger，如果按鍵一直被按著則歌曲會一直重複播放。



例. PA0 (L/U/R) = V1

[Input State]

;	PA0	PA1	PA2	PA3
TR1_Enable:	TR1F/TR1R	**X**	**X**	**X**

[Path]

PowerOn: **TR1_Enable, IC_Init**

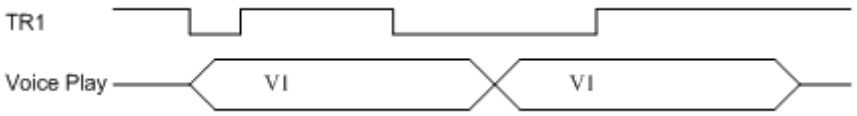
TR1F: **R0**=1, **PlayV**(CH0,$V1) ,**R0**=1?TR1F, **IC_Sleep**

TR1R: **R0**=0

2.2.4 Level mode, Unhold, Irretrigger

Level mode，Unhold，Irretrigger（縮寫爲：L/U/I），即：電位觸發，非保持，不可重複觸發。

功能：壓下按鍵後開始播放歌曲，使用者不可以在歌曲播放中再 trigger，如果按鍵一直被按著則歌曲會一直重複播放，按鍵放開後必須等到歌曲播放完畢後才能夠 trigger。



例. PA0 (L/U/I) = V1

[Input State]

;	PA0	PA1	PA2	PA3
TR1_Enable:	TR1F/TR1R	**X**	**X**	**X**
TR1_Disable:	TR1F1/TR1R	**X**	**X**	**X**

[Path]

PowerOn: **TR1_Enable**, **IC_Init**

TR1F: **TR1_Disable**, **R0**=1, **PlayV**(CH0,$V1,), **R0**=1?TR1F, **TR1_Enable**, **IC_Sleep**

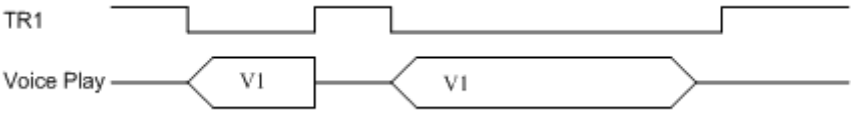
TR1R: **R0**=0

TR1F1: **R0**=1

2.2.5 Edge mode, Hold, X

Edge mode，Hold，X（縮寫爲：E/H/X），即：邊沿觸發，保持。

功能：壓下按鍵後開始播放歌曲，放開按鍵後歌曲將會停止播放，每一次的 trigger 只會播放一次歌曲。



例. PA0 (E/H/X) = V1

[Input State]

;	PA0	PA1	PA2	PA3
TR1_Enable:	TR1F/TR1R	**X**	**X**	**X**

[Path]

PowerOn: **TR1_Enable**, **IC_Init**

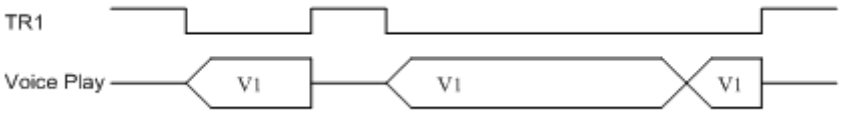
TR1F:**PlayV**(CH0,$V1,)

TR1R: **IC_Sleep**

2.2.6 Level mode, Hold, X

Level mode，Hold，X（縮寫爲：L/H/X），即：電位觸發，保持。

功能：壓下按鍵後開始播放歌曲，當按鍵被按著時歌曲會一直重複播放，放開按鍵後歌曲將會停止播放。



例: PA0 (L/H/X) = V1

[Input State]

;	PA0	PA1	PA2	PA3
TR1_Enable:	TR1F/TR1R	**X**	**X**	**X**

[Path]

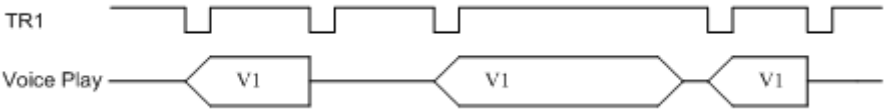
PowerOn: **TR1_Enable**, **IC_Init**

TR1F: **R0**=1,**PlayV**(CH1,$V1), **R0**=1?TR1F, **IC_Sleep**

TR1R: **R0**=0, **IC_Sleep**

2.2.7 Toggle On/Off

功能：按下按鍵後開始播放歌曲，當按鍵再次按下時，歌曲停止，如果再次按下按鍵時，歌曲又開始播放，如此重複。



例: PA0 (E/U/R, On/Off) = V1

[Input State]

;	PA0	PA1	PA2	PA3
TR1_Enable:	TR1R	**X**	**X**	**X**
TR1_Disable :	OFF	**X**	**X**	**X**

[Path]

PowerOn: **TR1_Enable**, **IC_Init**

TR1R: **TR1_Disable**, **PlayV**(CH0,$V1), **TR1_Enable**, **IC_Sleep**

OFF: **TR1_Enable**, **IC_Sleep**

2.3 輸出狀態的設置及切換

輸出狀態，即輸出口的狀態。每個輸出口會根據設置的輸出狀態執行相應的動作。不同的輸出狀態間可以自由切換，例如，當開始的輸出狀態爲 Output_0 時，經過一段時間的延時後，輸出狀態可以切換成 Output_1，即可以寫成：Output_0, Delay(200ms), Output_1。輸出狀態可以放在前景中，也可以放在背景中。

例.

```
[Output State]
;              PB0     PB1     PB2     PB3     ; 設置輸出狀態
LED_OFF:       1       1       1       1
LED_PB1:       1       0       X       X
LED_PB0:       0       1       X       X
LED_PB2:       X       X       0       0
LED_PB3:       X       X       1       1
LED_Action:    A1      A2      A3      A4
LED_FD:        FD      FD      FD      FD
[Path]
PowerOn: Input_0, LED_OFF
TR1: LED_Action, PlayV(CH0,$V0), LED_OFF   ; 按下TR1播放Voice，同時輸出Action的動作，聲音結
                                           ; 束後，輸出狀態切換到LED_OFF。
TR2: LED_FD, PlayV(CH1,$V0), LED_OFF       ; 按下TR2播放Voice，同時輸出FD的動作，聲音結束
                                           ; 後，輸出狀態切換到LED_OFF。
TR3: PlayV(CH2,$V0)&[BG1,BG2], All_End     ; 按下TR3播放Voice，同時，執行BG1和BG2的動作
TR4: StopBG, All_End                       ; 停止所有動作。
All_End: LED_OFF, IC_Sleep                 ; 停止所有動作，並進入睡眠狀態。
[Background1]
BG1: LED_PB1, Delay(167ms), LED_PB0, Delay(167ms), BG1   ; PB0和PB1作3Hz交錯閃爍。
[Background2]
BG2: LED_PB2, Delay(500ms), LED_PB3, Delay(500ms), BG2   ; PB2 和 PB3 作 1Hz 同步閃爍。
```

2.4 播放 Voice

對於NY7A系列IC，總共有8個I/O，使用者可根據實際功能，分配I/O腳位。NY7A系列IC提供8通道的語音/MIDI合成功能，所有的通道或部分的通道可同時播放語音或MIDI。

2.4.1 添加Voice

當功能要求播放 Voice 時，使用者需要在 Voice File 段落下添加。可直接添加 .wav 檔和 .nyq 檔，或者經過 *Voice_Encoder* 編碼之後的 .v7x 檔。

例.

```
[Voice File]
V0 = bird6k.wav
```

2.4.2 功能實現

對於一個 *Q-Code* 程式，要想實現功能，有一個段落必不可少，那就是 Path 段落，每一個程式功能的實現，都需要在這個段落下完成。Path 段落由 "PowerOn" 開始，即 IC 一上電要執行的動作，之後才能執行其它的動作。如果有按鍵功能時，上電時就必須先寫輸入狀態，此時 IC 才會掃描該輸入狀態下的按鍵，按下按鍵才會執行相應的功能，否則按鍵無效。(輸入狀態在 Input State 段落下設定)

例.

```
[Path]
PowerOn: Input_0
TR1: PlayV(CH0,$V0)
```

2.5 播放 Melody

2.5.1 添加 .md3 檔

當功能要求播放 MIDI 時，使用者不能在程式中直接添加 .mid 檔，而需要將 .mid 檔經過 *Q-Midi* 轉換之後，產生的 .md3 檔才能在 *Q-Code NY7* 中添加。(.mid 檔的轉換請參考 *Q-MIDI* 使用手冊)。使用者可在**[Melody Database]**段落下添加.md3 檔。

例.

```
[Melody Database]

Demo.md3

; m0 = EPiano_C3B5.mid, 1 channel
```

2.5.2 功能實現

如果要播放一首 Midi，使用者可直接寫播放 Midi 的指令即可。

例.

```
[Path]
PowerOn: Input_0
TR1: PlayM($M0)
```

2.6 NY7 播放 Voice 和 Midi 的範例

功能介紹

```
;*********************************************************************************************
```

4 個按鍵：TR1, TR2, TR3, TR4　　(PA0, PA1, PA2, PA3 按鍵都是 EUR 功能。)

4 個輸出：PB0, PB1, PB2, PB3　　(每一個輸出口接一個燈，所有的燈都是低電位有效。)

功能說明：

TR1(PA0)：按下該按鍵，播放第一首 Wave，播放聲音的過程中，PB0，PB1，PB2，PB3 對應的 LED 作流水燈閃爍。

TR2(PA1)：按下該按鍵，播放第二首 Wave，播放聲音的過程中，PB0，PB1，PB2，PB3 隨音量大小閃爍。

TR3(PA2)：按下該按鍵，播放第一首 MIDI，播放聲音的過程中，PB0，PB1，PB2，PB3 對應的 LED 作流水燈閃爍。

TR4(PA3)：按下該按鍵，播放第二首 MIDI，播放聲音的過程中，PB0 與 PB1，PB2 與 PB3 作 3Hz 對閃。

```
;*********************************************************************************
;Q-Code 程式
;=============================================================
[Option]
ICBody = NY7C110A
Client = Nyquest
Voltage = 3.0V
Voice Output = Push-Pull
FD = 1/2S
InterruptService = Melody
[Voice File]
V0=Wave\Example1.wav
V1=Wave\Example2.wav
[Melody Database]
Midi\Demo.md3
; m0 = _new_DeckthehallsA_1G_Cm1.mid, 6 channel
; m1 = _new_first2_1H_m60_Cm3.mid, 5 channel
; m2 = _new_Hark_1a.mid, 4 channel
; m3 = _new_JingleBells_1.mid, 4 channel
; m4 = _new_Joytotheworld_2.mid, 5 channel
; m5 = _new_OXmasTree_1a.mid, 4 channel
; m6 = _new_SilentNighta.mid, 4 channel
; m7 = _WeWish_1G_c.mid, 6 channel
[I/O Pin]
;       Input Pin
;           No.     1     2     3     4
;           Name    PA0   PA1   PA2   PA3
;           Key     TR1   TR2   TR3   TR4
;       Output Pin
;           No.     1     2     3     4     5     6     7     8     9     10    11    12
;           Name    PB0   PB1   PB2   PB3   PC0   PC1   PC2   PC3   PD0   PD1   PD2
PD3
Direct = 4
Input_Mode = [P:LT P:LT P:LT P:LT]
```

```
Output_Mode = [CS:H CS:H CS:H CS:H N:L N:L N:L N:L N:L N:L N:L N:L]
[Input State]
TR_Enable: TR1     TR2     TR3     TR4
[Output State]
LED_OFF:  1     1     1     1
LED0_ON:  0     1     1     1
LED1_ON:  1     0     1     1
LED2_ON:  1     1     0     1
LED3_ON:  1     1     1     0
LED_FD:    FD    FD    FD    FD
LED01_ON: 0     1     0     1
LED23_ON: 1     0     1     0
[Path]
PowerOn:IC_Init
;---------------------
TR1:
  LED_Output,          ; 背景 LED 流水燈閃爍
  PlayV(CH0,$V0),      ; 播放 Example1.wav
  IC_Sleep
;---------------------
TR2:
  LED_FD,              ; 背景 LED 隨音量大小閃爍
  PlayV(CH0,$V1),      ; 播放 Example2.wav
  IC_Sleep
;---------------------
TR3:
  LED_Output,          ; 背景 LED 流水燈閃爍
  PlayM($M0),          ; 播放 Midi0
  IC_Sleep
;---------------------
TR4:
  LED_3Hz,             ; PB0與PB1，PB2與PB3作 3Hz 互閃
  PlayM($M1),          ; 播放 Midi1
  IC_Sleep
;---------------------
IC_Init:
  Delay(50ms),
  TR_Enable,           ; 開啓所有按鍵
```

```
  IC_Sleep
;---------------------
IC_Sleep:
  LED_OFF,          ; 關閉 LED
  TR_Enable,        ; 開啓所有按鍵
  STOP              ; 關閉前景、背景和 Play、Delay 等，IC 進入睡眠模式。
;---------------------
[Background1]
LED_OutPut:
  LED0_ON,          ; 點亮 PB0 LED
  Delay(200ms),
  LED1_ON,          ; 點亮 PB1 LED
  Delay(200ms),
  LED2_ON,          ; 點亮 PB2 LED
  Delay(200ms),
  LED3_ON,          ; 點亮 PB3 LED
  Delay(200ms),
  LED_Output        ; LED 流水迴圈
;---------------------
LED_3Hz:
  LED01_ON,         ; 點亮 PB0, PB2 LED
  Delay(167ms),
  LED23_ON,         ; 點亮 PB1, PB3 LED
  Delay(167ms),
  LED_3Hz           ; LED 作 3Hz 互閃
;---------------------
```

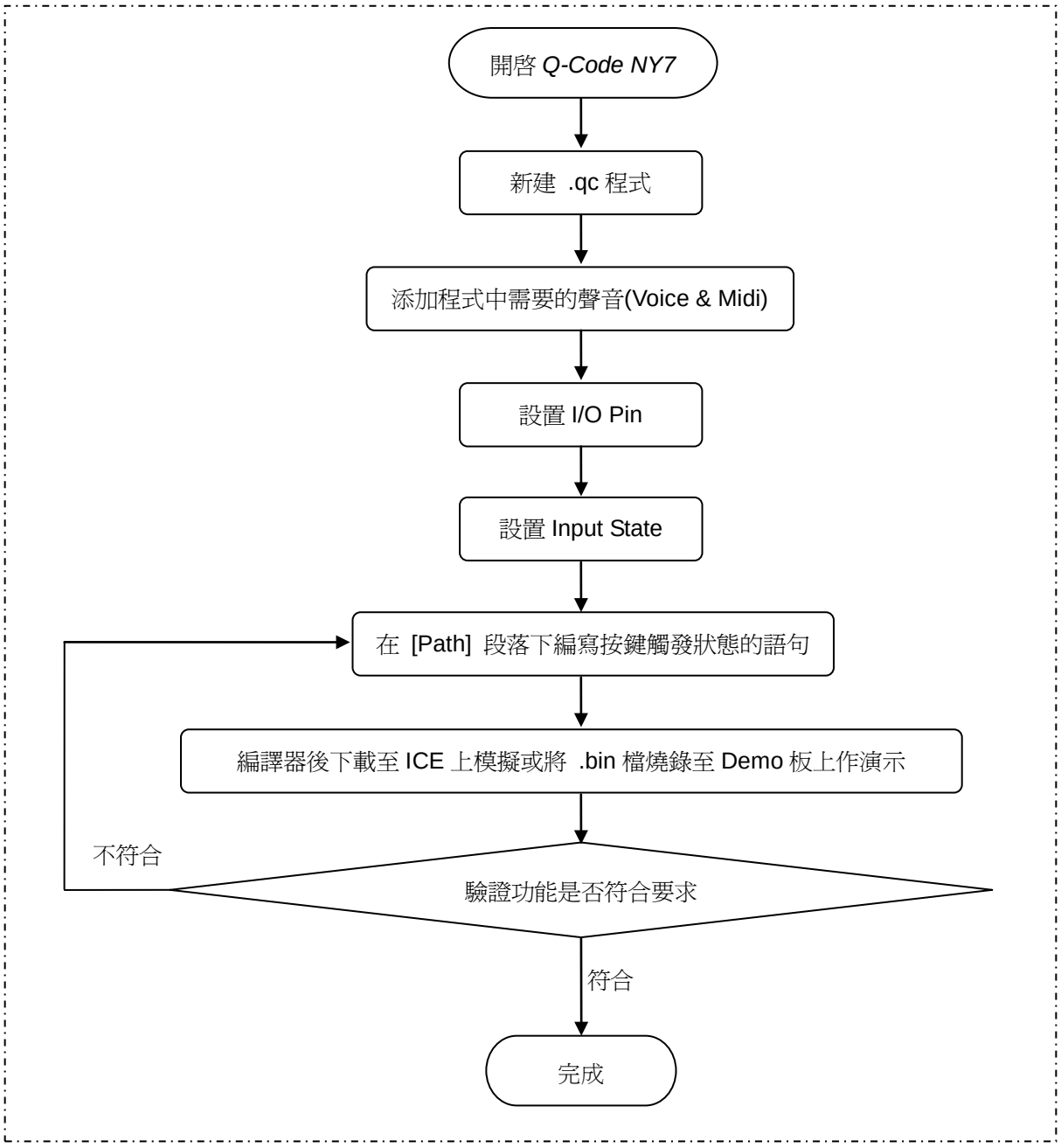
3 *Q-Code NY7* 具體應用實例

3.1 按鍵觸發模式組合

3.1.1 功能介紹

功能說明：利用按鍵觸發模式，實現不同的功能。	
E/U/R；E/U/I；L/U/R；L/U/I；E/H/X；L/H/X；Toggle On/Off	
7個按鍵 (PA0，PA1，PA2，PA3，PB0，PB1，PB2)：	
TR1 鍵 (PA0)	E/U/R: 按下該按鍵後開始播放Example.wav，在歌曲播放中可以再次觸發該按鍵。
TR2 鍵 (PA1)	E/U/I: 按下該按鍵後開始播放Example.wav，在歌曲播放中不可以再次觸發該按鍵，必須等到歌曲播放完畢後才能夠再次觸發。
TR3 鍵 (PA2)	L/U/R: 按下該按鍵後開始播放Example.wav，在歌曲播放中可以再次觸發該按鍵，如果按鍵一直被按著則歌曲會一直重複播放。
TR4 鍵 (PA3)	L/U/I: 按下該按鍵後開始播放Example.wav，在歌曲播放中不可以再次觸發該按鍵，如果按鍵一直被按著則歌曲會一直重複播放。
TR5 鍵 (PB0)	E/H/X: 按下該按鍵後開始播放Example midi0，放開按鍵後歌曲將會立即停止播放，每一次的觸發只會播放一次歌曲。
TR6 鍵 (PB1)	L/H/X: 按下該按鍵後開始播放Example midi1，如果按鍵一直被按著則歌曲會一直重複播放，放開按鍵後歌曲將會停止播放。
TR7 鍵 (PB2)	Toggle On/Off: 按下該按鍵後開始播放Example midi2，當按鍵再次按下時，歌曲將會停止播放，如果再次按下按鍵時，歌曲又開始播放，如此重複。

3.1.2 設計流程圖


開啓 Q-Code NY7
新建 .qc 程式
添加程式中需要的聲音(Voice & Midi)
設置 I/O Pin
設置 Input State
在 [Path] 段落下編寫按鍵觸發狀態的語句
編譯器後下載至 ICE 上模擬或將 .bin 檔燒錄至 Demo 板上作演示
不符合
驗證功能是否符合要求
符合
完成

3.1.3 Q-Code 程式

```
;----------------------------
; 基本Option選項設定
[Option]
ICBody = NY7B087A
Client = Nyquest
Voltage = 3.0V
Voice Output = Push-Pull
PowerOnTrigger = Enable
Debounce = 16ms
FD = 3/4S
InterruptService  =  Melody
Gain = 66%
;----------------------------
; 添加音源檔區段
[Voice File]
V0 = Wave\Example.wav
;----------------------------
; 添加 Action 檔區段
[Action File]
VIO1 = Action\Multicolour.vio
;----------------------------
; 添加 Midi 檔區段
[Melody Database]
Midi\Demo.md3
; m0 = _new_DeckthehallsA_1G_Cm1.mid, 6 channel
; m1 = _new_first2_1H_m60_Cm3.mid, 5 channel
; m2 = _new_Hark_1a.mid, 4 channel
; m3 = _new_JingleBells_1.mid, 4 channel
; m4 = _new_Joytotheworld_2.mid, 5 channel
; m5 = _new_OXmasTree_1a.mid, 4 channel
; m6 = _new_SilentNighta.mid, 4 channel
; m7 = _WeWish_1G_c.mid, 6 channel
;----------------------------
; I/O 設定區段
[I/O Pin]
;          Input Pin
```

```
;              No.      1      2      3      4      5      6      7      8
;              Name     PA0    PA1    PA2    PA3    PB0    PB1    PB2    PB3
;              Key      TR1    TR2    TR3    TR4    TR5    TR6    TR7    TR8
;         Output Pin
;              No.      1      2      3      4      5      6      7      8
;              Name     PC0    PC1    PC2    PC3    PD0    PD1    PD2    PD3
;
Direct = 8
Input_Mode = [P:LT P:LT P:LT P:LT P:LT P:LT P:LT P:LT]
Output_Mode = [CS:L CS:L CS:L CS:L N:L N:L N:L N:L]
;---------------------------
; 按鍵輸入狀態設定區段
[Input State]
TR1_Enable:      TR1     X       X       X       X       X       X
TR2_Enable:      X       TR2     X       X       X       X       X
TR2_Disable:     X       X       X       X       X       X       X
TR3_FR_Enable:   X       X       TR3F/TR3R       X       X       X       X
TR4_FR_Enable1:  X       X       X       TR4F1/TR4R      X       X       X
TR4_FR_Enable2:  X       X       X       TR4F2/TR4R      X       X       X
TR5_FR_Enable:   X       X       X       X       TR5F/TR5R       X       X
TR6_FR_Enable:   X       X       X       X       X       TR6F/TR6R       X
TR7_OFF:         X       X       X       X       X       X       OFF
All_KeyEnable:   TR1     TR2     TR3F    TR4F1   TR5F    TR6F    TR7
;---------------------------
; I/O  輸出狀態設定區段
[Output State]
LED_OFF:    1       1       1       1
LED0_ON:    0       1       1       1
LED1_ON:    1       0       1       1
LED2_ON:    1       1       0       1
LED3_ON:    1       1       1       0
LED_FD:     FD      FD      FD      FD
LED_Action: A1      A2      A3      X
;---------------------
[Symbol]
Key_Flag = R0                   ; 按鍵壓下釋放標誌
;Key_Flag.0 = R0.0              ; TR3  按鍵壓下釋放標誌
;Key_Flag.1 = R0.1              ; TR4  按鍵壓下釋放標誌
```

```
;Key_Flag.2 = R0.2              ; TR6 按鍵壓下釋放標誌
;----------------------
; 副程式
[Subroutine]
Set_Voice_Channel:
    CHMode(2)                  ; 開啓 2 個 channel(提升播放品質）
Set_Melody_Channel:
    CHMode(8)                  ; 開啓 8 個 channel(播放多 Channel Melody）
;----------------------
; Main 程式
[Path]
PowerOn: IC_Init
;----------------------
IC_Init:
    LED_OFF,                   ; 關閉 LED
    Delay(50ms),
    All_KeyEnable,             ; 開啓所有按鍵
;   CHMode(2),                 ; 開啓 2 個 channel(提升播放品質）
    IC_Sleep
;----------------------
; E/U/R
TR1:
    TR1_Enable,                ; 僅允許 TR1 可以重複觸發
    LED_Output,                ; 開啓背景LED流水燈閃爍
    Set_Voice_Channel,
    PlayV(CH0,$V0),            ; 播放 Example.Wav
    IC_Sleep
;----------------------
; E/U/I
TR2:
    TR2_Disable,               ; 禁止 TR2 觸發，直到Voice播放結束
    LED_FD,                    ; LED隨音量閃爍
    Set_Voice_Channel,
    PlayV(CH0,$V0),            ; 播放 Example.Wav
    IC_Sleep
;----------------------
; L/U/R
TR3F:
```

```
    TR3_FR_Enable,          ; 允許 TR3 按鍵壓下釋放都回應
    Key_Flag.0=1,           ; 設置 TR3 按鍵壓下標誌
    LED_Action,             ; Action 隨音樂輸出，PB0、PB1、PB2 組成RGB全彩LED輸出。
    PlayAS(CH2,$VIO1),
    Set_Voice_Channel,
    PlayV(CH0,$V0),         ; 播放 Example.Wav
    Key_Flag.0=1?           ; TR3 被壓下？
    TR3F,                   ; 是，迴圈播放
    IC_Sleep
TR3R:
    Key_Flag.0=0            ; 清除按鍵釋放標誌
;--------------------
; L/U/I
TR4F1:
    TR4_FR_Enable2,         ; 允許播放過程再次觸發，但是不打斷當前播放
    Key_Flag.1=1,           ; 設置 TR4 按鍵壓下標誌
    LED_Action,             ; Action 隨音樂輸出，PB0、PB1、PB2 組成RGB全彩LED輸出。
    PlayAS(CH2,$VIO1),
    Set_Voice_Channel,
    PlayV(CH0,$V0),         ; 播放 Example.Wav
    Key_Flag.1=1?           ; TR4 被壓下？
    TR4F1,                  ; 是，迴圈播放
    IC_Sleep
TR4R:
    Key_Flag.1=0            ; 設置 TR4 按鍵釋放標誌
TR4F2:
    Key_Flag.1=1            ; 設置 TR4 按鍵壓下標誌
;--------------------
; E/H/X
TR5F:
    TR5_FR_Enable,          ; 允許 TR5 按鍵有效
    LED_Action,             ; Action 隨音樂輸出，PB0、PB1、PB2 組成RGB全彩LED輸出。
    PlayAS(CH2,$VIO1),
    Set_Melody_Channel,
    PlayM($M0),             ; 播放 Example midi0
    IC_Sleep
TR5R:
    IC_Sleep
```

```
;----------------------
; L/H/X
TR6F:
    TR6_FR_Enable,          ; 允許 TR6 按鍵有效
    Key_Flag.2=1,           ; 設置 TR6 按鍵壓下標誌
    LED_Output,             ; 開啓背景LED流水燈閃爍
    Set_Melody_Channel,
    PlayM($M1),             ; 播放 Example midi1
    Key_Flag.2=1?           ; TR6 被壓下？
    TR6F,                   ; 是，迴圈播放
    IC_Sleep
TR6R:
    Key_Flag.2=0,           ; 設置 TR6 按鍵釋放標誌
    IC_Sleep
;----------------------
; ON/OFF
TR7:
    TR7_OFF,                ; 按鍵再次壓下，結束播放
    LED_Output,             ; 開啓背景LED流水燈閃爍
    Set_Melody_Channel,
    PlayM($M2),             ; 播放 Example midi2
    IC_Sleep
OFF:
    IC_Sleep
;----------------------
IC_Sleep:
    Key_Flag = 0,           ; 清除所有按鍵標誌
    LED_OFF,                ; 關閉 LED
    All_KeyEnable,          ; 開啓所有按鍵
    STOP                    ; 關閉前景、背景和 Play、Delay 等，IC 進入睡眠模式。
;----------------------
[Background1]
LED_OutPut:
    LED0_ON,                ; 點亮 PB0 LED
    Delay(200ms),
    LED1_ON,                ; 點亮 PB1 LED
    Delay(200ms),
    LED2_ON,                ; 點亮 PB2 LED
```

```
    Delay(200ms),
    LED3_ON,                ; 點亮 PB3 LED
    Delay(200ms),
    LED_Output              ; LED 流水迴圈
;--------------------
[Background2]
; Null
```

4 附錄

Q-Code 指令清單及說明如下。

4.1 Arithmetic Command（算數邏輯指令）

4-bit Arithmetic Command				
Ri = data	Ri.n = 1	Ri.n = 0	Ri = Rj	Ri.n = Rj.n
Ri = RandomL	Ri = RandomH	!Ri	!Ri.n	Ri++
Ri--	Ri = Rj + Rk	Ri = Rj - Rk	Ri = Rj | Rk	Ri = Rj ^ Rk
Ri = Rj & Rk	Ri = Ri + data	Ri = Rj – data	Ri = Rj | data	Ri = Rj ^ data
Ri = Rj & data	Ri = data + Rj	Ri = data - Rj	[Ri, Rj] = Rk * Rl	[Ri, Rj] = Rk * data
[Ri, Rj] = data * Rk	[Ri, Rj] = Rk / Rl	[Ri, Rj] = Rk / data	[Ri, Rj] = data / Rk	Ri = Rj << n
Ri = Rj >> n	Ri = Rj << Rk	Ri = Rj >> Rk	-	-
8-bit Arithmetic Command				
Ri = XjL	Ri = XjH	Ri.n = XjL.n	Ri.n = XjH.n	Ri.n = Xj.n
XiL = Rj	XiH = Rj	XiL.n = Rj.n	XiH.n = Rj.n	!Xi
!Xi.n	Xi++	Xi--	Xi = data	Xi = Xj
Xi.n = 0	Xi.n = 1	Xi.n = Xj.n	Xi = Random	Xi = Xj + Xk
Xi = Xj - Xk	Xi = Xj | Xk	Xi = Xj ^ Xk	Xi = Xj & Xk	Xi = Xi + data
Xi = Xj – data	Xi = Xj | data	Xi = Xj ^ data	Xi = Xj & data	Xi = data + Xj
Xi = data - Xj	[Xi, Xj] = Xk * Xl	[Xi, Xj] = Xk * data	[Xi, Xj] = data * Xk	[Xi, Xj] = Xk / Xl
[Xi, Xj] = Xk / data	[Xi, Xj] = data / Xk	Xi = Xj << n	Xi = Xj >> n	Xi = Xj << Rk
Xi = Xj >> Rk	-	-	-	-

Ex1: R0=0x4, R1=R0 ；賦值操作, R0=0x4, R1=0x4

Ex2: R0.1=1, R0.0=0 ；賦值操作，表示 R0 的第 1 位爲 1, R0 的第 0 位爲 0

Ex3: R0.2=1, R1=0x8, R1.1=R0.2 ；賦值操作, R1 最終的值爲 0xA

Ex4: R0=RandomL, R1=RandomH ；賦值操作，如果 RandomL=1, RandomH=2，則 R0=1, R1=2

Ex5: !R0, !R0.3 ；取反運算，如果 R0=1，則!R0=0xE，再執行!R0.3, R0=0x6

Ex6: R0++, R0-- ；變數的增減，如果 R0=1，則執行 R0++後, R0=2，再執行
；R0--後, R0=1

Ex7: R2=R0 + R1, R2=R0 - R1 ；暫存器的加、減運算，如果 R0=9, R1=3, R2=R0+R1=12,
；R2=R0-R1=6

Ex8: R2=R0 + 5, R2=R0 - 8, R2=3 + R0, R2=9 - R0 ；暫存器的加、減運算，如果 R0=9, R2=R0+5=14,
；R2=R0-8=1, R2=3+R0=12, R2=9-R0=0

Ex9: R2=R0 & R1, R3=R0 | R1, R4=R0 ^ R1 ；暫存器的與、或、異或運算，如果 R0=1, R1=3,

；則 R2=1(1 & 3), R3=3 (1 | 3), R4=2 (1 ^ 3)

Ex10: R0=R0 & 5, R0=R0 | 2, R0=R0 ^ 8 ；暫存器的與、或、異或運算，如果 R0=15 則

；R0(5)=R0&5(15&5), R0(7)=R0|2(5| 2), R0(15)=R0^ 8(7^8)

Ex11: [R1, R0]=R0 * R1, [R1, R0]=R0 * 5, [R1, R0]=2 * R0, [R1, R0]=R0 / R1, [R1, R0]=R0 / 5, [R1, R0]=12 / R0

；1) 4 位的乘法運算 [R1, R0]=R0 * R1, R0 乘以 R1，把高四位放在 R1，低四位放在 R0

；2) 4 位的乘法運算 [R1, R0]=R0 * 5, R0 乘以 5，把高四位放在 R1，低四位放在 R0

；3) 4 位的乘法運算 [R1, R0]=2 * R0, 2 乘以 R0，把高四位放在 R1，低四位放在 R0

；4) 4 位的除法運算 [R1, R0]=R0 / R1, R0 除以 R1，把商放在 R1，餘數放在 R0

；5) 4 位的除法運算 [R1, R0]=R0 / 5, R0 除以 5，把商放在 R1，餘數放在 R0

；6) 4 位的除法運算 [R1, R0]=12 / R0, 12 除以 R0，把商放在 R1，餘數放在 R0

Ex12: R0=R0 << 2, R0=R0 >> 2, R0=R0 << R1, R0=R0 >> R1

；1) 4 位暫存器的左移右移運算, R0=R0 << 2，左移 2 位; R0=R0 >> 2，右移 2 位

；2) 4 位暫存器的左移右移運算, R0=R0 << R1，左移 R1 位; R0=R0 >> R1，右移 R1 位

Ex13: R0=X2L, R1=X2H ；賦值操作，如果 X2=0x56，則 R0=0x6, R1=0x5

Ex14: X2L=R0, X2H=R1 ；賦值操作，如果 R0=0xA, R1=0x3，則 X2=0x3A

Ex15: R0.1=X2L.1, R0.3=X2H.3 ；賦值操作，如果 X2=0x54, R0=0xA，則執行指令後 R0=0

Ex16: X2L.0=R0.0, X2H.2=R0.2 ；賦值操作，如果 X2=0x54, R0=0x1，則執行指令後 X2=0x15

Ex17: R0.1=X2.3 ；賦值操作，如果 X2=0x39, R0=0xC，則執行指令後 R0=0xE

Ex18: !X0, !X0.3 ；取反運算，如果 X0=1，則!X0=0xE, !X0.3, X0.3=0

Ex19: X0++, X0-- ；變數的增減，如果 X0=1，則執行 X0++後, X0=2，再執行 X0--後, X0=1

Ex20: X0=0x4, X1=X0 ；賦值操作，執行指令後 X1=0x4

Ex21: X0.1=1, X0.0=0 ；賦值操作，表示 X0 的第 1 位爲 1, X0 的第 0 位爲 0

Ex22: X0.7=X1.5 ；賦值操作，如果 X0=0x39, X1=0x7C，則執行指令後 X0=0xB9

Ex23: X0=Random ；賦值操作，如果亂數 Random 爲 0xD2，則 X0=0xD2

Ex24: X2=X0 + X1, X2=X1 – X0 ；暫存器的加、減運算，如果 X0=0x29, X1=0x33, X2=X0+X1=0x5C,

；X2=X1-X0=0xA

Ex25: X2=X0 + 5, X2=X0 - 8, X2=3 + X0, X2=0x86 - X0

；暫存器的加、減運算，如果 X0=0x34, X2=X0+0x5=0x39,

；X2=X0-0x8=0x2C, X2=0x3+X0=0x37, X2=0x86-X0=0x52

Ex26: X2=X0 & X1, X3=X0 | X1, X4=X0 ^ X1

；暫存器的與、或、異或運算，如果 X0=1, X1=3，則 X2=1(1& 3),

；X3=3 (1| 3), X4=2 (1^ 3)

Ex27: X0=X0 & 5, X0=X0 | 2, X0=X0 ^ 8

；暫存器的與、或、異或運算，如果 X0=15，則 X0(5)=X0&5(15&5),

；X0(7)=X0|2(5| 2), X0(15)=X0^ 8(7^8)

Ex28: [X1,X0]=X0 * X1, [X1,X0]=X0 * 5, [X1,X0]=5 * X0 , [X1,X0]=X0 / X1, [X1,X0]=X0 / 5, [X1,X0]=12 / X0

；1) 8 位的乘法運算 [X1,X0]=X0 * X1, X0 乘以 X1，把高八位放在 X1，低八位放在 X0

；2) 8 位的乘法運算 [X1,X0]=X0 * 5, X0 乘以 5，把高八位放在 X1，低八位放在 X0

；3) 8 位的乘法運算 [X1,X0]=5 * X0, 5 乘以 X0，把高八位放在 X1，低八位放在 X0

；4) 8 位的除法運算 [X1,X0]=X0 / X1, X0 除以 X1，把商放在 X1，餘數放在 X0

；5) 8 位的除法運算 [X1,X0]=X0 / 5, X0 除以 5，把商放在 X1，餘數放在 X0

；6) 8 位的除法運算 [X1,X0]=12 / X0, 12 除以 X0，把商放在 X1，餘數放在 X0

Ex29: X0=X0 << 2, X0=X0 >> 2, X0=X0 << R2, X0=X0 >> R2

；1) 8 位暫存器的左移右移運算, X0=X0 << 2，左移 2 位; X0=X0 >>2，右移 2 位

；2) 8 位暫存器的左移右移運算, X0=X0 << R2，左移 R2 位; X0=X0 >>R2，右移 R2 位

4.2 Condition Jump Command（條件跳轉指令）

4-bit Condition Jump Command				
Ri = Rj?Path	Ri != Rj?Path	Ri >= Rj?Path	Ri <= Rj?Path	Ri > Rj?Path
Ri < Rj?Path	Ri = data?Path	Ri != data?Path	Ri >= data?Path	Ri <= data?Path
Ri > data?Path	Ri < data?Path	Ri.n = 0?Path	Ri.n != 0?Path	Ri.n = 1?Path
Ri.n != 1?Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	Px[1 X 0 X]?Path	Voice(CH)?Path	PauseV(CH)?Path
@Melody?Path	@PauseM?Path	Vol = n?Path	Vol !=n ?Path	Delay(n)?Path
Action(CH)?Path	CheckSum?Path	-	-	-
RandomL = data?Path		RandomH = data?Path		
Switch(Ri) = [Path0, Path1, Path2,..Path15]		Switch(Px) = [Path0, Path1, Path2,..Path15]		
Switch(Px[d X d X]) = [Path0, Path1, Path2….Path15]				
8-bit Condition Jump Command				
Xi = Xj?Path	Xi != Xj?Path	Xi >= Xj?Path	Xi <= Xj?Path	Xi > Xj?Path
Xi < Xj?Path	Xi = data?Path	Xi != data?Path	Xi >= data?Path	Xi<=data?Path
Xi > data?Path	Xi < data?Path	Xi.n = 0?Path	Xi.n != 0?Path	Xi.n = 1?Path
Xi.n != 1?Path	-	-	-	-
Random = data?Path		Random != data?Path		
Switch(Xi)=[Path0, Path1, Path2,..Path255]		Switch(Random)=[Path0, Path1,Path2,..Path255]		

Ex1: R0=R1?Path1, Path2 ；R0 等於 R1 程式跳至 Path1，否則程式跳至 Path2

Ex2: R0!=R1?Path1, Path2 ；R0 不等於 R1 程式跳至 Path1，否則程式跳至 Path2

Ex3: R0>=R1?Path1, Path2 ；R0 大於或等於 R1 程式跳至 Path1，否則程式跳至 Path2

Ex4: R0<=R1?Path1, Path2 ；R0 小於或等於 R1 程式跳至 Path1，否則程式跳至 Path2

Ex5: R0>R1?Path1, Path2 ；R0 大於 R1 程式跳至 Path1，否則程式跳至 Path2

Ex6: R0<R1?Path1, Path2 ；R0 小於 R1 程式跳至 Path1，否則程式跳至 Path2

Ex7: R0=data?Path1, Path2 ；R0 等於 data 程式跳至 Path1，否則程式跳至 Path2

Ex8: R0!=data?Path1, Path2 ；R0 不等於 data 程式跳至 Path1，否則程式跳至 Path2

```
Ex9: R0>=data?Path1, Path2          ; R0 大於或等於 data 程式跳至 Path1, 否則程式跳至 Path2
Ex10: R0<=data?Path1, Path2         ; R0 小於或等於 data 程式跳至 Path1, 否則程式跳至 Path2
Ex11: R0>data?Path1, Path2          ; R0 大於 data 程式跳至 Path1, 否則程式跳至 Path2
Ex12: R0<data?Path1, Path2          ; R0 小於 data 程式跳至 Path1, 否則程式跳至 Path2
Ex13: R0.1=0?Path1, Path2           ; R0.1 等於 0 則程式跳至 Path1, 否則程式跳至 Path2
Ex14: R0.1!=0?Path1, Path2          ; R0.1 不等於 0 則程式跳至 Path1, 否則程式跳至 Path2
Ex15: R0.1=1?Path1, Path2           ; R0.1 等於 1 則程式跳至 Path1, 否則程式跳至 Path2
Ex16: R0.1!=1?Path1, Path2          ; R0.1 不等於 1 則程式跳至 Path1, 否則程式跳至 Path2
Ex17: PB=3?Path1, Path2             ; PB 等於 3 程式跳至 Path1, 否則程式跳至 Path2
Ex18: PB!=3?Path1, Path2            ; PB 不等於 3 程式跳至 Path1, 否則程式跳至 Path2
Ex19: PB.1=0?Path1, Path2           ; PB.1 等於 0 則程式跳至 Path1, 否則程式跳至 Path2
Ex20: PB.1!=0?Path1, Path2          ; PB.1 不等於 0 則程式跳至 Path1, 否則程式跳至 Path2
Ex21: PB.1=1?Path1, Path2           ; PB.1 等於 1 則程式跳至 Path1, 否則程式跳至 Path2
Ex22: PB.1!=1?Path1, Path2          ; PB.1 不等於 1 則程式跳至 Path1, 否則程式跳至 Path2
Ex23: PB[ 1 0 0 1 ]?Path1, Path2    ; PB 等於 0x9 則程式跳至 Path1, 否則程式跳至 Path2
Ex24: Voice(0)?Path1, Path2         ; 判斷 CH0 是否在播放 Voice, 有則程式跳至 Path1,
                                    ; 反之, 程式跳至 Path2
Ex25: PauseV(0)?Path1, Path2        ; 判斷 CH0 是否在暫停播放 Voice, 有則程式跳至 Path1,
                                    ; 反之, 程式跳至 Path2
Ex26: @Melody?Path1, Path2          ; 判斷程式是否在播放 Melody, 有則程式跳至 Path1,
                                    ; 反之, 程式跳至 Path2
Ex27: @PauseM?Path1, Path2          ; 判斷程式是否在暫停播放 Melody, 有則程式跳至 Path1,
                                    ; 反之, 程式跳至 Path2
Ex28: Vol=5?Path1, Path2            ; 判斷音量是否等於 5, 等於則程式跳轉至 Path1,
                                    ; 反之, 程式跳轉至 Path2
Ex29: Vol!=5?Path1, Path2           ; 判斷音量是否不等於 5, 不等於則程式跳轉至 Path1,
                                    ; 反之, 程式跳轉至 Path2
Ex30: Delay(0)?Path1, Path2         ; 判斷程式前景是否有 Delay 延時, 有則程式跳至 Path1,
                                    ; 反之, 跳至 Path2
                                    ; n=0 前景, n=1 背景 1, n= 2 背景 2, 不指定 n 判斷所有路徑
Ex31: Action(1)?Path1, Path2        ; 判斷程式 Action 是在 CH1 播放, 是則程式跳至 Path1,
                                    ; 反之, 程式跳至 Path2
                                    ; ch 代表第幾通道播放, 不指定 ch 判斷所有通道
Ex32: CheckSum?Path1, Path2         ; 判斷程式的 CheckSum 是否正確, 正確則程式跳至 Path1,
                                    ; 反之, 程式跳至 Path2
Ex33: RandomL=0x6?Path1,Path2       ; 判斷亂數的低四位是否等於 0x6, 等於則跳轉至 Path1,
                                    ; 反之, 跳轉至 Path2
Ex34: RandomH=0xB?Path1,Path2       ; 判斷亂數的高四位是否等於 0xB, 等於則跳轉至 Path1,
```

；反之，跳轉至 Path2

Ex35: Switch(R0)=[Path0, Path1, Path2,..Path15]

；R0=2，此時程式執行 Path2

Ex36: Switch(PA)=[Path0, Path1, Path2,..Path15]

；如果 PA=2，此時程式執行 Path2

Ex37: Switch(PA[d d d d])=[Path0, Path1, Path2,..Path15]

；如果 PA=0xF，此時程式執行 Path15

Ex38: X0=X1?Path1, Path2 ；X0 等於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex39: X0!=X1?Path1, Path2 ；X0 不等於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex40: X0>=X1?Path1, Path2 ；X0 大於或等於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex41: X0<=X1?Path1, Path2 ；X0 小於或等於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex42: X0>X1?Path1, Path2 ；X0 大於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex43: X0<X1?Path1, Path2 ；X0 小於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex44: X0=0xDE?Path1, Path2 ；X0 等於 0xDE 程式跳至 Path1，否則程式跳至 Path2

Ex45: X0!=0x54?Path1, Path2 ；X0 不等於 0x54 程式跳至 Path1，否則程式跳至 Path2

Ex46: X0>=0xC?Path1, Path2 ；X0 大於或等於 0xC 程式跳至 Path1，否則程式跳至 Path2

Ex47: X0<=0x91?Path1, Path2 ；X0 小於或等於 0x91 程式跳至 Path1，否則程式跳至 Path2

Ex48: X0>0x3D?Path1, Path2 ；X0 大於 0x3D 程式跳至 Path1，否則程式跳至 Path2

Ex49: X0<0xF3?Path1, Path2 ；X0 小於 0xF3 程式跳至 Path1，否則程式跳至 Path2

Ex50: X0.1=0?Path1, Path2 ；X0.1 等於 0 則程式跳至 Path1，否則程式跳至 Path2

Ex51: X0.1=1?Path1, Path2 ；X0.1 等於 1 則程式跳至 Path1，否則程式跳至 Path2

Ex52: X0.1!=0?Path1, Path2 ；X0.1 不等於 0 則程式跳至 Path1，否則程式跳至 Path2

Ex53: X0.1!=1?Path1, Path2 ；X0.1 不等於 1 則程式跳至 Path1，否則程式跳至 Path2

Ex54: Random=250?Path, Path2 ；如果 Random 等於 250，則程式執行 Path1，否則執行 Path2

Ex55: Random!=250?Path, Path2 ；如果 Random 不等於 250，則程式執行 Path1，否則執行 Path2

Ex56: Switch(X0)=[Path0, Path1, Path2,..Path255]

；X0=220，此時程式執行 Path220

Ex57: Switch(Random)=[Path0, Path1, Path2,..Path255]

；Random=220，此時程式執行 Path220

4.3 I/O Command（I/O指令）

4-bit I/O Command				
Ri = Px	Ri.n = Px.n	Px = data	Px = Ri	Px = Py
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Ri = PxM	PxM = Ri	PxM.n = 0	PxM.n = 1
PxM = data	Px = Py + Ri	Px = Py + data	Px = Py - Rj	Px = Py - data
Px = Py | Ri	Px = Py | data	Px = Py ^ Ri	Px = Py ^ data	Px = Py & Ri
Px = Py & data	Ri = Px + Rj	Ri = Px + data	Ri = Px - Rj	Ri = Px – data
Ri = Px | Rj	Ri = Px | data	Ri = Px ^ Rj	Ri = Px ^ data	Ri = Px & Rj
Ri = Px & data	Px = [1 x 0 FD Q]	Px.n= 1KHz(time)	-	-
8-bit I/O Command				
XiL = Px	XiH = Px	Xi = [Px, Py]	Px = XiL	Px = XiH
[Px, Py] = Xi	-	-	-	-

Ex1: R0=PA ；將 PA 的值讀入 R0

Ex2: R0.1=PA.1 ；將 PA.1 的值讀入 R0.1

Ex3: PB=0x5 ；將 PB 輸出 0x5

Ex4: PB=R1 ；將 PB 輸出 R1 的值，若 R1=0x4, PB 輸出 0x4

Ex5: PB=PA ；將 PB 輸出 PA 的值，若 PA=0x1, PB 輸出 0x1

Ex6: !PB ；將 PB 的值取反輸出

Ex7: !PB.2 ；將 PB.2 的值取反輸出

Ex8: PB.3 =0 ；將 PB.3 輸出 0

Ex9: PB.3 =1 ；將 PB.3 輸出 1

Ex10: R0=PBM, PAM=R0, ；先將 PBM 埠方向暫存器的值讀入到 R0,
；再將 R0 的值賦給 PAM

Ex11: PBM=0x5, PBM.0=0, PBM.1=1 ；先將 PB 設置成輸出，再透過指令 PBM=0x5，將 PB.0, PB.2 設置
；成輸入，再透過 PBM.0 把 PB.0 設置成輸出,透過 PBM.1 把 PB.1
；設置成輸入（PBM 等於 0 的位爲輸出，等於 1 的位爲輸入）

Ex12: PB=0, R1=8, PB=PB+R1, PB=PB+2 ；將 PB 輸出 0x0, PB=PB+R1,PB 輸出 8, PB=PB+2,PB 又輸出 10

Ex13: PB=14, R1=8, PB=PB-R1, PB=PB-3 ；將 PB 輸出 0xE, PB=PB-R1, PB 輸出 6, PB=PB-3, PB 輸出 3

Ex14: R1=4, PB=9, PB=PB | R1, PB=PB | 6 ；PB 與 R1 進行或運算後輸出 0xD, PB 與 6 進行或運算後輸出 0xF

Ex15: R1=4, PB=5, PB=PB^R1, PB=PB^6 ；PB 與 R1 進行異或運算後輸出 1, PB 與 6 進行異或運算後輸出 3

Ex16: R1=6, PB=7, PB=PB&R1, PB=PB&5 ；PB 與 R1 進行與運算後輸出 6, PB 與 5 進行與運算後輸出 5

Ex17: PB=9, R1=4, R0=PB+R1, R0=PB+1 ；執行 R0=PB+R1 後, R0=13，執行 R0=PB+1 後, R0=10

Ex18: PB=9, R1=4, R0=PB-R1, R0=PB-8 ；執行 R0=PB-R1 後, R0=5，執行 R0=PB-8 後, R0=1

Ex19: R1=6, PB=12, R0=PB | R1, R0=PB | 1 ；執行 R0=PB | R1 後, R0=14，執行 R0=PB | 1 後, R0=13

Ex20: R1=4, PB=8, R0=PB ^ R1, R0=PB ^ 3 ；執行 R0=PB ^ R1 後, R0=12, 執行 R0=PB ^ 3 後, R0=11

Ex21: R1=12, PB=7, R0=PB&R1, R0=PB&11 ；執行 R0=PB & R1 後, R0=4, 執行 R0=PB & 11 後, R0=3

Ex22: PB=[x 0 FD Q] ；將 PB.0 QIO 輸出, 將 PB.1 FD 輸出 將 PB.2 輸出 0, 將 PB.3 保持不變

Ex23: PB.0=1KHz(3) ；將 PB.0 輸出 3 秒 1KHz 的方波

Ex24: X0L=PA, X0H=PB ；將 PA 的值讀入 X0L, PB 的值讀入 X0H

Ex25: PA=X0L, PB =X0H ；將 X0L 的值寫入 PA 口, 將 X0H 的值寫入 PB

Ex26: X0=[PA, PB], [PA, PB]=X0 ；將 PA、PB 的值讀入 X0, X0 低位讀入 PB 的值, X0 高位讀入 PA 的值, 將 X0 的值寫入 PA、PB, X0 低位的值寫入 PB, X0 高位的值寫入 PA

4.4 Path Command（路徑指令）

Path Command				
ASM	BG(BG1,BG2)	Break	StopFG	StopBG
StopBG1	StopBG2	Macro	Subroutine	Label(Pathname)

Ex1: ASM, Macro, Subroutine

；1) ASM 是插入彙編的介面

；2) Macro 是巨集的介面

；3) Subroutine 是副程式的介面

Ex2: BG(BG1, BG2) ；打開背景 1, 背景 2

Ex3: BG(BG1, OFF) ；打開背景 1, 關閉背景 2

Ex4: BG(OFF, BG2) ；關閉背景 1, 打開背景 2

Ex5: Break ；結束當前前景運行的 Delay, Voice, Action, 運行下一條指令

Ex6: StopFG ；停止所有前景運行的 Delay, Voice, Action 等動作

Ex7: StopBG ；停止所有背景運行的 Delay, Voice, Action 等動作

Ex8: StopBG1 ；停止所有背景 1 運行的 Delay, Voice, Action 等動作

Ex9: StopBG2 ；停止所有背景 2 運行的 Delay, Voice, Action 等動作

Ex10: PlayV($V0), Label(Loop), PlayV($V1), Loop ；迴圈播放 PlayV($v1)

4.5 Voice Command（語音指令）

Voice Command				
PlayV	PlayVS	ChMode(n)	PauseV(CH)	ResumeV(CH)
StopV(CH)	-	-	-	-

Ex1: PlayV($V0, 6K)*2 & [BG1, BG2] ；播放 V0 (Voice) 兩遍，且調用背景 1 和背景 2

Ex2: PlayVS($V0, 6K), BG1, BG2 ；播放 V0 (Voice) 兩遍，且調用背景 1 和背景 2

Ex3: ChMode(4), PlayV(CH1, $V1) ；開放 4 個通道，預定使用通道 1 來播放 V1

Ex4: PauseV(0) ；暫停 CH0 的 Voice 的播放

Ex5: ResumeV(0) ；恢復 CH0 的 Voice 的播放

Ex6: StopV(0) ；停止 CH0 的 Voice 的播放

4.6 Melody Command（音樂指令）

Melody Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM
StopM	Tempo = n	Tempo++	Tempo--	Tempo(Ri:Rj)
Mute_On(Ch)	Mute_Off(Ch)	SingleNote_On	Singlenote_Off	SinglePlay

Ex1: PlayM($M0)*2 & [BG1, BG2] ；播放 M0 (MIDI) 兩遍，且調用背景 1 和背景 2

Ex2: PlayMS($M0), BG1, BG2 ；播放 M0 (MIDI)，且調用背景 1 和背景 2

Ex3: WaitMN, PlayM($M0) ；等其它的 MIDI 播放完後再播放 M0 (MIDI)

Ex4: PauseM ；暫停 MIDI 的播放

Ex5: ResumeM ；恢復 MIDI 的播放

Ex6: StopM ；停止所有 MIDI 的播放

Ex7: Tempo = 100 ；將播放的 MIDI 的 Tempo 改爲 100，詳細 Tempo List，
；請查閱 *Q-Code* NY7 UM

Ex8: Tempo = 100, Tempo ++ ；Tempo++，播放的 MIDI 的 Tempo 改爲 102

Ex9: Tempo = 100, Tempo -- ；Tempo--，播放的 MIDI 的 Tempo 改爲 98

Ex10: X0=100, Tempo (R1:R0) ；將播放的 MIDI 的 Tempo 改爲 100

Ex11: Mute_On(0) ；把 CH0 的 MIDI 靜音, CH 代表 MIDI 的通道，不指定 CH 表示所有通道

Ex12: Mute_OFF(0) ；把 CH0 靜音的 MIDI 恢復, CH 代表 MIDI 的通道，不指定 CH 表示所有通道

Ex13: SingleNote_On ；開啓 One-Key-One-Note 功能

Ex14: Singlenote_Off ；關閉 One-Key-One-Note 功能

Ex15: SinglePlay ；播放一個音符

4.7 Keyboard Command（鍵盤指令）

Keyboard Command				
Instrument(i)	NoteOn	-	-	-

Ex1: Instrument ；用於設定 Keyboard 音色

Ex2: NoteOn ；播放一個音符

4.8 Volume Command（音量指令）

Volume Command				
Vol_Max	Vol_Min	Vol = n	Vol = Ri	Ri = Vol
Vol++	Vol--	Px = Vol	-	-

Ex1: Vol_Max ；將輸出音量調到最大 (最大爲 15)

Ex2: Vol_Min ；將輸出音量調到最小 (最小爲 0)

Ex3: Vol=7 ；將輸出音量設置爲 7

Ex4: Vol=R0 ；音量由 R0 控制

Ex5: R0=Vol ；將目前的音量設定取出, 存入 R0

Ex6: Vol++ ；將目前的音量設定遞加一級

Ex7: Vol-- ；將目前的音量設定遞減一級

Ex8: PB=Vol ；將目前 Volume 的階數輸出到 PB

4.9 Table Command（查表指令）

Table Command	
TableL(TableName, Rx, Ry, Ri)	TableM(TableName, Rx, Ry, Ri)
TableH(TableName, Rx, Ry, Ri)	Table(TableName, Rx, Ry, Rh, Rm, Rl)
TableL(TableName, X, Y, Ri)	TableM(TableName, X, Y, Ri)
TableH(TableName, X, Y, Ri)	Table(TableName, X, Y, Rh, Rm, Rl)
TableL(TableName, Xx, Xy, Xi)	TableH(TableName, Xx, Xy, Xi)
Table(TableName, Xx, Xy, Xh, Xl)	TableL(TableName, X, Y, Xi)
TableH(TableName, X, Y, Xi)	Table(TableName, X, Y, Xh, Xl)

Ex1: R0=2, R1=1, TableL(trans, R0, R1, R2) ；R2=0x6

Ex2: R0=2, R1=1, TableM(trans, R0, R1, R2) ；R2=0xF

Ex3: R0=2, R1=1, TableH(trans, R0, R1, R2) ；R2=0x3

Ex4: R0=2, R1=1, Table(trans, R0, R1, R2, R3, R4) ；R2=0x3, R3=0xF, R4=0x6

Ex5: TableL(trans, 2, 1, R2) ；R2=0x6

Ex6: TableM(trans, 2, 1, R2) ；R2=0xF

Ex7: TableH(trans, 2, 1, R2) ; R2=0x3

Ex8: Table(trans, 2, 1, R2, R3, R4) ; R2=0x3, R3=0xF, R4=0x6

Ex9: X0=2, X1=1, TableL(trans, X0, X1, X2) ; X2=0xF6

Ex10: X0=2, X1=1, TableH(trans, X0, X1, X2) ; X2=0x3

Ex11: X0=2, X1=1, Table(trans, X0, X1, X2, X3) ; X2=0x3, X3=0xF6

Ex12: TableL(trans, 2, 1, X2) ; X2=0xF6

Ex13: TableH(trans, 2, 1, X2) ; X2=0x3

Ex14: Table(trans, 2, 1, X2, X3) ; X2=0x3, X3=0xF6

[Table]

```
Trans :
{
[0x1, 0x3, 0x5, 0x7, 0x9],
[0x4, 0x5, 0x3F6, 0x8, 0x10],
[0x0, 0x1, 0x2, 0x3, 0xf4 ]
}
```

4.10 IR Command（紅外線指令）

IR Command				
TX = data	TX(Ri:Rj:Rk:Rl)	TX(Xi:Xj)	RX_ON	RX_OFF
RX = data?Path	RX != data?Path	-	-	-

Ex1: TX=0x01 ; 發射代碼爲 0x01

Ex2: R0=1, R1=2, R2=3, R3=4, TX(R0) ; 發射代碼爲 0x1, 4 位發生

Ex3: R0=1, R1=2, R2=3, R3=4, TX(R1: R0) ; 發射代碼爲 0x21, 8 位發生

Ex4: R0=1, R1=2, R2=3, R3=4, TX(R2: R1: R0) ; 發射代碼爲 0x321, 12 位發生

Ex5: R0=1, R1=2, R2=3, R3=4, TX(R3: R2: R1: R0) ; 發射代碼爲 0x4321, 16 位發生

Ex6: R0=1, R1=2, R2=3, R3=4, TX(X1: X0) ; 發射代碼爲 0x4321, 16 位發生

Ex7: RX_ON ; 開啓紅外接收功能

Ex8: RX_OFF ; 關閉紅外接收功能

Ex9: RX=0x01?Path1, Path2 ; 如果接收到的代碼爲 0x01 時，程式執行 Path1，否則執行 ; Path2

Ex10: RX!=0x01?Path1, Path2 ; 如果接收到的代碼不是 0x01 時，程式執行 Path1，否則執行 ; Path2

4.11 Interrupt Command（中斷指令）

Interrupt Command				
INT_ON	INT_OFF	INT_RET	INT=n	-

Ex1: INT_ON ；開中斷

Ex2: INT_OFF ；關中斷

Ex3: INT_RET ；中斷返回

Ex4: INT=n ；n=0.256、n=0.512、n=1.024

4.12 Delay Command（時間延遲指令）

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

Ex1: Delay(8ms) ；延時 8ms，延時的範圍 4ms~15sec

Ex2: R2=0, R1=0xF, R0=0xA, Delay(R2:R1:R0) ；延時 1s，延時的範圍 4ms~16.38s, 4ms 一個梯度

Ex3: WaitDN(1), PlayV($v0) ；等背景 1 的延時結束以後再播放 PlayV($v0),

；(n=0 前景, n=1 背景 1, n=2 背景 2，不指定 n 表示所有路徑)

Ex4: StopD(1) ；結束背景 1 的延時，不指定 n 表示所有路徑

Ex5: PauseD(1) ；暫停背景 1 的延時，不指定 n 表示所有路徑

Ex6: ResumeD(1) ；恢復背景 1 的延時，不指定 n 表示所有路徑

Ex7: Sdelay(1ms) ；延時 1ms，該條指令爲閉環計時，會影響按鍵掃描等

4.13 Action Command（動作指令）

Action Command				
PlayA	PlayAS	WaitAN(n)	PauseA(n)	ResumeA(n)
StopA(n)	-	-	-	-

Ex1: PlayA(PB.0, ch1, $A1) ；輸出口 PB.0，通道 1，播放 Action A1

Ex2: PlayAS(PB.0, ch1, $A1), PlayV($v0) ；播放 Action 馬上播放 PlayV($v0)

Ex3: WaitAN(1), PlayV($v0) ；等背景 1 的 Action 結束以後再播放 PlayV($v0),

；(n=0 前景, n=1 背景 1, n=2 背景 2，不指定 n 表示所有路徑)

Ex4: PauseA(1) ；暫停背景 1 的 Action，不指定通道表示所有路徑

Ex5: ResumeA(1) ；恢復背景 1 的 Action，不指定通道表示所有路徑

Ex6: StopA(1) ；結束背景 1 的 Action，不指定通道表示所有路徑

4.14 MISC Command（一般指令）

MISC Command				
Input State	Midi Mark State	NoteOn State	KEY_CLR	Key_On
Key_Off	Debounce(Time)	Stop	Pause(n)	Resume(n)
ReadTempo	ReadChannel	Audio_On	Audio_Off	AudioMode
NoiseFilter_On	NoiseFilter_Off	RampUp	RampDown	WDT_CLR
Repeat	END	ReadRollingCodeRi:Rj:Rk:Rl:Rm)		-

Ex1: Input State ；按鍵輸入狀態切換

Ex2: Midi Mark State ；Midi Mark 輸出狀態切換

Ex3: NoteOn State ；NoteOn State 輸出狀態切換

Ex4: KEY_CLR ；清除當前按鍵狀態，重新掃描按鍵

Ex5: KEY_On ；開啓按鍵掃描功能

Ex6: KEY_Off ；關閉按鍵掃描功能

Ex7: Debounce(16ms) or Debounce(0.016)
；設定目前的 Debounce 時間爲 16ms

Ex8: Stop ；結束所有的動作

Ex9: Pause(0) ；暫停前景背景的所有動作, n=0 前景, n=1 背景 1, n=2 背景 2，不指定 n 表示所有路徑

Ex10: Resume(0) ；恢復前景背景的所有動作, n=0 前景, n=1 背景 1, n=2 背景 2，不指定 n 表示所有路徑

Ex11: ReadTempo(X0), [PC,PB]=X0 ；將 Tempo 的設定值輸出到 PB, PC

Ex12: ReadChannel(X0), [PC,PB]=X0 ；將 Channel 的設定值輸出到 PB, PC

Ex13: Audio_On ；開啓 Audio 輸出

Ex14: Audio_Off ；關閉 Audio 輸出

Ex15: AudioMode=DAC ；選擇聲音輸出模式，設定 DAC 輸出
；n=DAC/PWM/PP

Ex16: NoiseFilter_On ；打開雜訊濾波功能

Ex17: NoiseFilter_Off ；關閉雜訊濾波功能

Ex18: RampUp ；RampUp 指令

Ex19: RampDown ；RampDown 指令

Ex20: WDT_CLR ；清看門狗

Ex21: Repeat ；重複播放, {PlayV($V0), Delay(2) }*3, “{ }” 內的程式會執行 3 次

Ex22: END ；整潔代碼，無實在意義

Ex23: ReadRollingCode(R4:R3:R2:R1:R0) ；將讀出的 rolling code 存到 R0~R4