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

使用手冊

NY9T 系列 *Q-Code* 範例

PowerSpeech Format Programmer

Version 1.0
Aug. 31, 2015

改 版 記 錄

版本	*日期*	*內 容 描 述*	*修正頁*
1.0	2015/8/31	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，為客戶提供高品質及高附加價值的 Touch IC，並提供優質的服務。為使客戶能夠更方便快捷的使用九齊 IC，九齊科技針對 NY9T 系列 IC 開發了一套軟體工具 — *Q-Code_NY9T*。通過 *Q-Code_NY9T*，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，讓初學者能夠快速瞭解到 *Q-Code_NY9T* 的基本功能和應用。

1.1 內容

1.2 何謂 *Q-Code_NY9T*？

Q-Code_NY9T 為九齊科技股份有限公司所提供，它是針對九齊科技的 NY9T 系列而開發的軟體工具。它提供簡易的圖形處理界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_NY9T* 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_NY9T* 的安裝程式檔案，按兩下執行檔案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_NY9T* 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_NY9T* 時同步安裝。

在使用 *Q-Code_NY9T* 時還可能用到以下一些相關工具：

Q-Touch：此軟體為 NY9T 系列觸控 IC 的開發輔助工具。提供簡明的圖形化界面，快速的掃描並取得各按鍵預設的一組靈敏度數值。使用者可通過模擬功能，查看並微調所需的靈敏度，最後保存為 T9X 檔，即可於 *Q-Code_NY9T* 中使用。

Q-Visio： 此軟體是信號編輯軟體，可以產生 VIO 格式的檔，其中 VIO 檔可以添加到 *Q-Code_NY9T* 的 Action 區段中，*Q-Code_NY9T* 自動將信號編碼轉化成資料表供自己使用。

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

注意：在安裝 Q-Code_NY9T 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

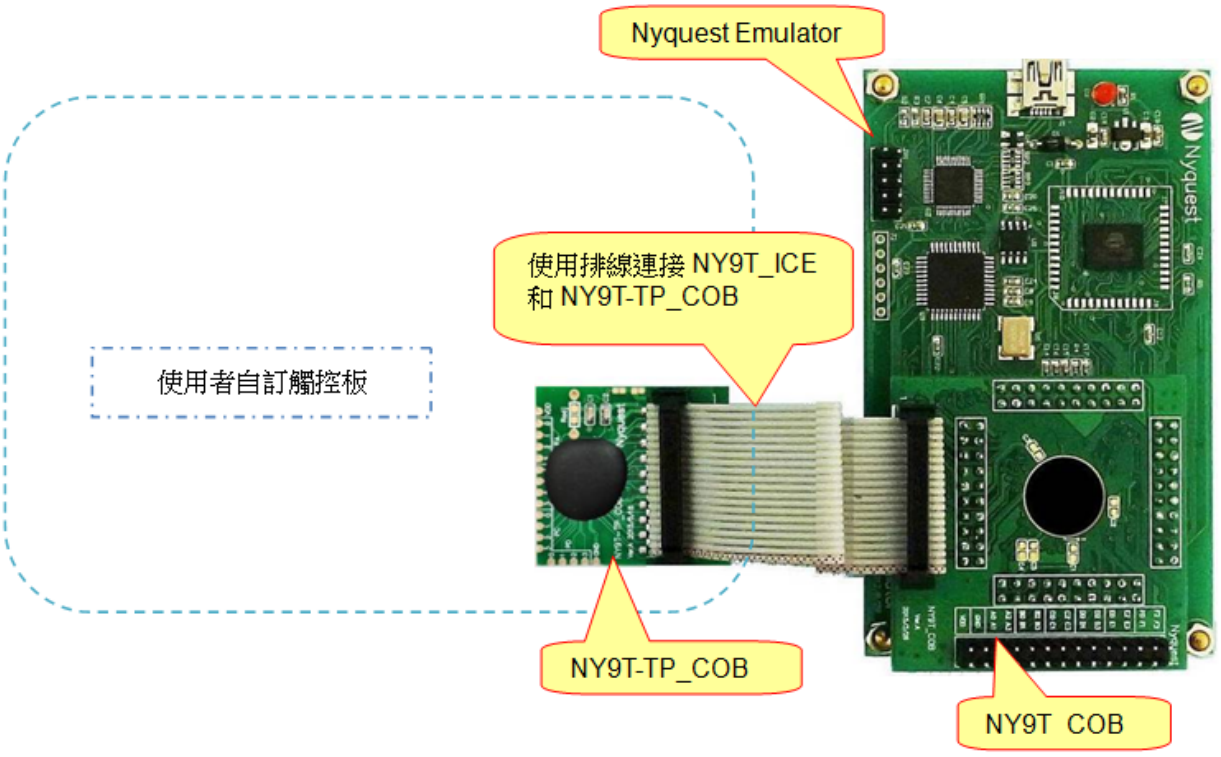
當完成程式編寫並編譯後，使用者可以將程式下載至 ICE 上進行模擬，或者透過燒錄器 FDB_Writer 或 Q-FDB_Writer 燒錄至 Demo 板上進行演示。

何謂ICE? ICE是In Circuit Emulator的縮寫，中文稱之為實體模擬器或模擬器。只需要將ICE通過USB連接至個人電腦，便可以將編譯好的程式下載至ICE進行驗證。(有關ICE硬體安裝請參考*NYIDE* 使用者手冊。)
當使用者使用*Q-Touch*之前，需先架設好ICE環境，方能進行*Q-Touch*軟體的使用。

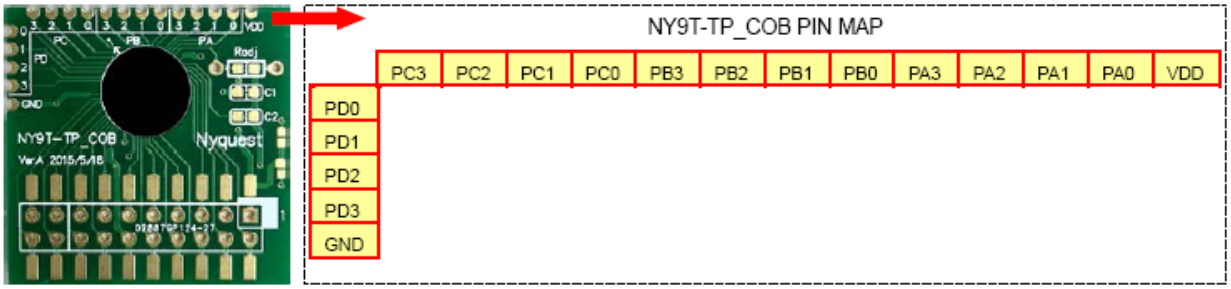
NY9T_ICE 分為3個部分：

1. Nyquest Emulator (與Nyquest 其它系列IC共用)
2. NY9T_COB (EV Chip – MCU部分)
3. NY9T-TP_COB (EV Chip – Touch Pad 部分)

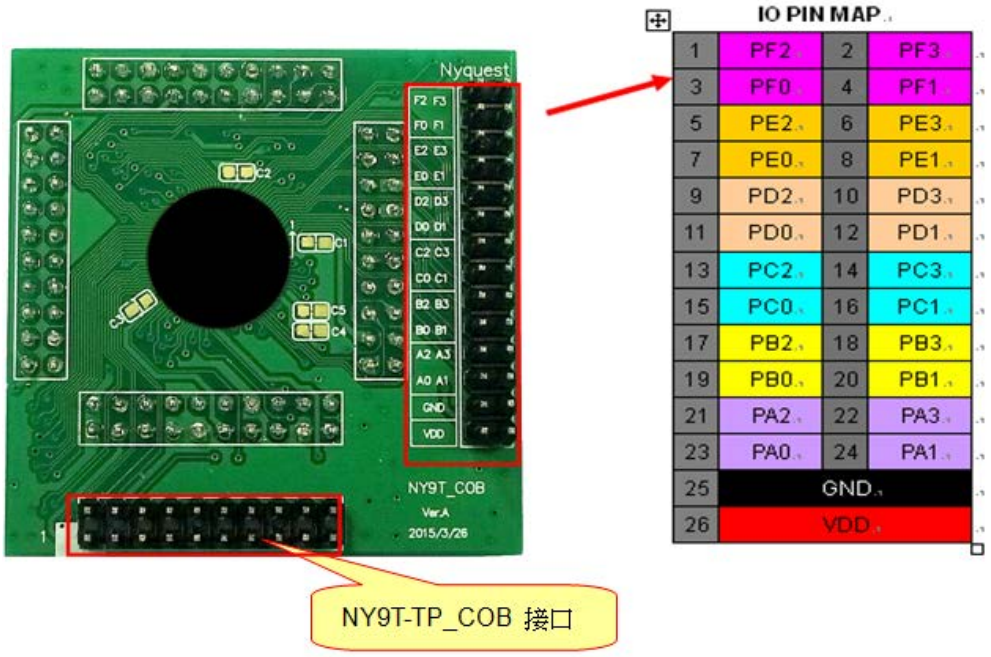
NY9T 開發環境：



NY9T-TP_COB：



NY9T_COB：



何謂*Q-Writer*？*Q-Writer* 是一個圖形界面的燒錄系統，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。

2 基本功能

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

IC Body：根據實際應用選擇適當的 IC Body。

例. IC Body = NY9T004A

Client：必須填入客戶名稱，否則編譯將不能通過。（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，一般為3.0V或是4.5V。(頻率參考電壓)

例. Voltage = 3.0V

例. Voltage = 4.5V

Time Base：設置時間基準為1ms或4ms。

例. TimeBase = 1ms

Touch Key：觸摸按鍵相關option設定，請參考Q-Code_NY9T_中觸摸按鍵(Touch Key)_設定。

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

例. Reset = P:W

Action：設定由一般I/O輸出VIO波形的Frame Rate和Compression。

Frame Rate：Action 訊號的更新率（默認為 80）。

Compression：開啟 Action 訊號壓縮功能（默認為 Enable）。

例. Action_FrameRate = 80

例. ActionCompression = Enable

Random：Random 表示產生出來的亂數範圍。亂數的值可選 0~255 之間。(預設值=disable)

例. Random = 122

Debounce：使用者可以選擇普通按鍵的 debounce 時間，時間範圍：1 ~ 1000ms。（預設值= 16ms）

例. Debounce = 16ms

Power On Trigger：可以选择是否要有 Power On Trigger 功能。（預設值=Enable）

例. PowerOnTrigger = Disable **；關閉Power On Trigger 功能。**

Short Debounce：使用者可以選擇哪一個獨立普通按鍵的 Short debounce，時間範圍：0 ~ 1ms。

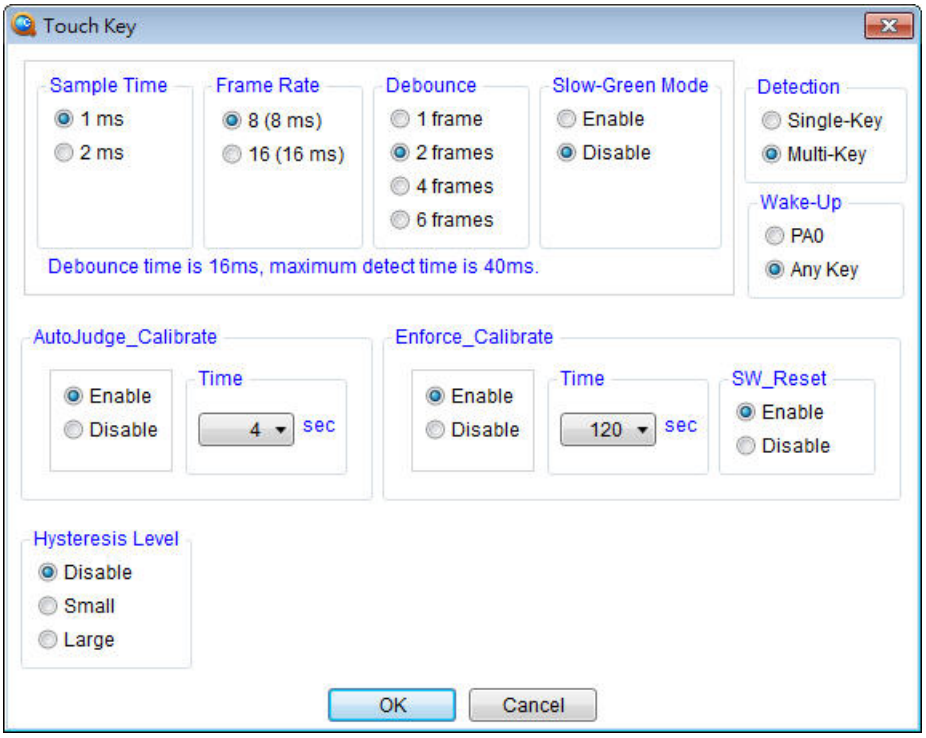
例. ShortDebounce = PA0, PA1, PA2 **；PA0, PA1, PA2為Short Debounce功能。**

Serial Control：可連接一般微控制器，將NY9T當成微控制器的按鍵。提供3種不同的傳輸協議，分別是SPI_Like、NY3 Serial_Trigger、IR_Trigger。請參考*Q-Code_NY9T* 中**Serial Control 設定**。

Interrupt Service：*Q-Code_NY9T*提供時間中斷服務。使用者若是對於精確度有較高的要求，可以將該功能改由中斷來處理。

例. InterruptService = Timer **；使用時間中斷。**

➢ ***Q-Code_NY9T* 中觸摸按鍵(Touch Key) 設定：**



Sample Time：掃描一個觸摸鍵的時間。較長的取樣時間觸摸更精確，但偵測到 Touch Key 被觸摸的時間會較久。有兩個選項：1ms/2ms。(預設值=1ms)

Frame Rate：在一個 Frame 要取樣的數量，即完成所有觸摸鍵的掃描時間。取樣的數量越多則較不耗電，但偵測到 Touch Key 被觸摸的時間會較久。有兩個選項：8/16 samples。

Debounce：連續幾個 Frame 被偵測到，按鍵才會成立。越多 Frame 觸摸更精確，但偵測到 Touch-Key 被觸摸的時間會越久，對於較容易有外部干擾的環境建議選擇多些 Frame。次數選項：1/2/4/6 frames。(預設值=2 frames)

Slow-Green Mode：只有在 Touch Key 設定使用超過 8 個鍵時(含 8 個)，才能打開或關閉此項功能。若使用者有下達"TouchKey_Scan_Slow"指令：則 Slow-Green Mode 打開時表示在進入睡眠後會維持 Slow-Green Mode，反之關閉 Slow-Green Mode 時表示進睡眠後會維持 Slow Mode。Slow-Green Mode 比 Slow Mode 更省電，但偵測到 Touch Key 被觸摸的時間會較久。(預設值=Disable)

TouchKey_DebounceTime 為一般模式下偵測到 Touch Key 被觸摸的時間；而 maximum Detect Time 為進入睡眠後的最長偵測時間，實際上可能少於最長偵測時間。公式如下：

Normal Mode = SampleTime x FrameRate x Debounce

Slow Mode = SampleTime x FrameRate x (3 + Debounce)

Slow-Green Mode = SampleTime x FrameRate x (4 + Debounce)

注意：

1. 選用 NY9T016A 時，Frame Rate 無法設定成 8ms。

2. NY9T001A/NY9T004A 無 Slow-Green Mode 設定。

3. Slow-Green Mode 只有在進入睡眠後，且切換到 TouchKey_Scan_Slow 才有效。

4. 在正常工作模式下，都維持 Normal Mode Scan，不管有沒有下達"TouchKey_Scan_Slow"指令。

例.

TouchKey_DebounceTime=64ms{SampleTime=2ms,FrameRate=32ms, Debounce=2, Slow-Green_Mode=Enable} ；一般掃描模式下觸摸鍵的偵測時間為 64ms。

TouchKey_DetectTime = 192ms ；進入睡眠後觸摸鍵的最長偵測時間為 192ms。

注意：Sample Time、Frame Rate、Debounce 和 Slow-Green Mode 皆會影響觸摸鍵的反應時間及耗電流。

Detection：使用者可以選擇觸摸鍵的偵測方式，共有 2 種偵測方式。Single-Key 為同一時間只能接受單一觸摸鍵被按下，Multi-Key 為可同一時間多個觸摸鍵被按下。(預設值=Multi-Key)

例. **TouchKey_Detection = Multi-Key** ；可同時接受多個觸摸鍵被按下。

Wakeup：使用者可以選擇只用 PA0 來喚醒或是全部的觸摸鍵皆可喚醒。(預設值=AnyKey)

例. **TouchKey_Wakeup = PA0** ；觸摸鍵只有 PA0 可以喚醒 IC。

例. **TouchKey_Wakeup = AnyKey** ；所有的觸摸鍵皆可喚醒 IC。

注意：

1. NY9T001A 無此設定。

2. 使用單鍵（PA0）喚醒可以獲得更佳的省電效能。

AutoJudge_Calibrate：觸摸鍵自動環境校正。使用觸摸鍵時，建議經過一段時間後要進行觸摸鍵校正。系統提供自動校正，並可設定多久作一次自動校正(時間=4~60 秒，預設值=4 秒)。使用者也可將自動校正功能關閉，並自行利用 4 秒路徑來計數更長的時間。

例. **AutoJudge_Calibrate = 4s** ；觸摸鍵每 4 秒自動校正一次。

例. **AutoJudge_Calibrate = Disable** ；關閉觸摸鍵自動校正功能。

注意：觸摸鍵功能被關閉後，則自動校正功能也會被關閉。

Enforce_Calibrate：強制進行觸摸鍵自動環境校正。為了避免觸摸鍵非人為按下，而導致 IC 非預期性的工作，建議經過一段時間後要進行強制觸摸鍵校正，若觸摸鍵被誤觸，則因強制校正將此誤觸視為環境參數，可避免 IC 無法進入睡眠而持續耗電。若不使用系統提供設定的時間可自行將強制校正功能關閉，由使用者利用 4 秒路徑來計數時間，以進行校正。(時間=4~120 秒，預設值=120 秒)

例. **Enforce_Calibrate = 60s** ；觸摸鍵每 60 秒自動校正一次。

例. **Enforce_Calibrate = Disable** ；關閉觸摸鍵自動校正功能。

注意：當觸摸鍵被按住時，Enforce_Calibrate 計數器才會開始計數，按鍵放開則會復位數目器。

SW_Reset：在執行觸摸鍵的強制校正後，是否進行復位 RAM 及 IO 狀態。(預設值=Enable)

例. **SW_Reset = Enable** ；觸摸鍵強制校正後，進行復位。

例. **SW_Reset = Disable** ；觸摸鍵強制校正後，不進行復位。

注意：使用者如需要記憶設定或操作模式，可將 SW_Reset 功能關閉，改成在 Enforce_Calibrate 路徑中來進行手動復位。

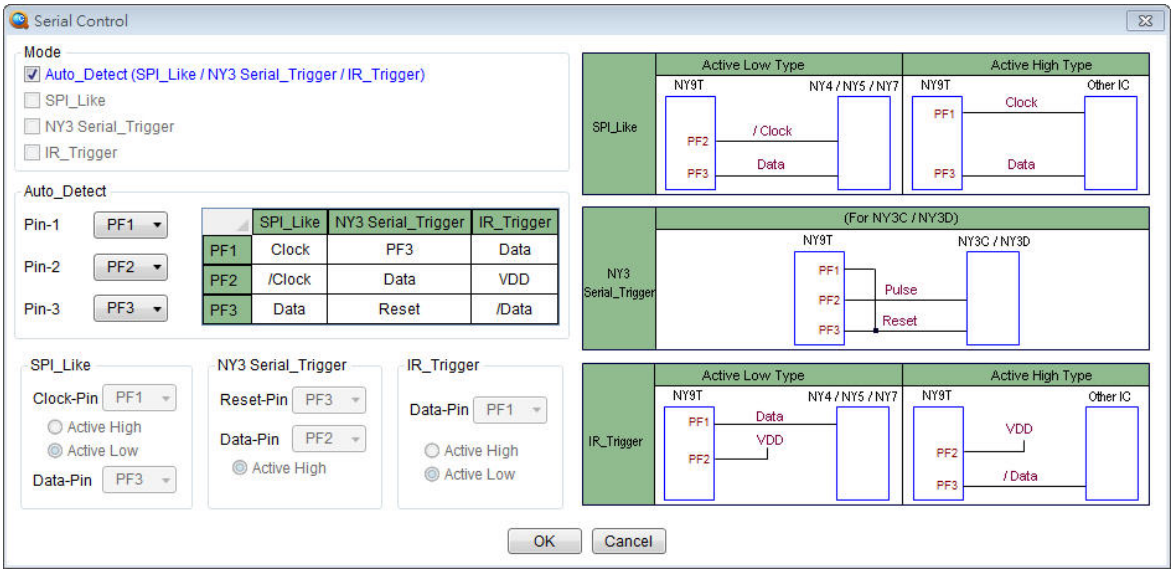
Hysteresis Level：觸摸鍵遲滯功能。可以調整觸摸鍵按下後的靈敏度，避免摸到觸摸點邊緣部分時，可能會造成觸摸鍵處於不穩定狀態。（預設值=Disable）。

例. **TouchKey_Hysteresis_Level = Disable** ；觸摸鍵觸發後，靈敏度不自動調整。

例. **TouchKey_Hysteresis_Level = Small** ；觸摸鍵觸發後，該觸摸鍵靈敏度提高一階。

例. **TouchKey_Hysteresis_Level = Large** ；觸摸鍵觸發後，該觸摸鍵靈敏度增強 50%。

➢ **Q-Code_NY9T 中Serial Control 設定：**



自動偵測：可由 3 根腳位的設定方式來決定，NY9T 要使用何種傳輸協定。自動偵測的腳位設定方式如下表所示：

自動模式偵測			
Mode	**Pin-3**	**Pin-2**	**Pin-1**
IR_Trigger	X	VDD	X
NY3 Serial_Trigger	Pin-3 connected to Pin-1	X	Pin-3 connected to Pin-1
SPI_Like	No Connected	No Connected	No Connected

例. **SerialControl = Auto**

SerialControl_Pin1 = PF.1

SerialControl_Pin2 = PF.2

SerialControl_Pin3 = PF.3 ；自動偵測傳輸協定。

SPI_Like：使用 2 根腳位，達成類似 SPI 傳輸協議。

例. **SerialControl = SPI_Like**

SerialControl_SPI_Clock = PF.2

SerialControl_SPI_Data = PF.3

SerialControl_SPI_ClockTrigger = High or Low **；傳輸協議為正緣觸發或是負緣觸發。**

NY3 Serial_Trigger：使用者可以透過串列訊號與 NY3 連線，將 NY9T 當成 NY3 的觸發按鍵。

例. **SerialControl = NY3**

SerialControl_NY3_Reset = PF.3

SerialControl_NY3_Data = PF.2

IR_Trigger：使用 1 個腳位來模擬 IR 的接收訊號，可直接使用目前的 NY4/5/7 系列的 IR 通訊。

例. **SerialControl = IR_Trigger**

SerialControl_IR_Data = PF.3

SerialControl_IR_DataBusy = High or Low **；傳輸協議為正緣觸發或是負緣觸發。**

注意：

1. *同一時間只能有一種傳輸協議*
2. *開啟 Auto-Detect 功能時，PIN-1 會被設成 Input Floating，PIN-2/PIN-3 會被設成 Input Weak-Pull-Low。*
3. *打開 Auto-Detect 功能時，接收端的 IC，必須將連接腳設定成 Input floating。*

2.1.2 I/O Pin 設定

設定I/O Pin的組態，分為 Normal-IO、Touch-Key、PWM-IO三種選擇，其中PA0~PD3為Touch-Key與Normal-IO共用、PE0~PF3為PWM-IO與Normal-IO共用，具體設定選擇如下：

Name	Mode	Direction	Input Type	Output Type	Sink Type	Sink Current	Initial	Busy
PA0~PD3	Touch-Key	Input	-	-	-	-	-	-
	Normal-IO	Input	Weak-Pull-Low	CMOS	Normal	-	-	Low
			Strong-Pull-Low	Open-Drain				High
			Floating					
		Output	Weak-Pull-Low	CMOS	Normal	-	High	Low
			Strong-Pull-Low	Open-Drain			Low	High
			Floating					
PE0~PF3	Normal-IO	Input	Weak-Pull-Low	CMOS	Normal	-	-	Low
			Strong-Pull-Low	Open-Drain				High
			Floating					
		Output	Weak-Pull-Low	CMOS	Constant	100%	High	Low
						83%		High
						50%	Low	Low
						33%		High
			Strong-Pull-Low		Large	100%	High	Low
						83%		High
				Open-Drain		50%	Low	Low
						33%		High
			Floating		Normal	-	High	Low
								High
							Low	Low
								High
	PWM-IO	Output	-	Sink	Constant	100%	High	-
						83%		
					Large	50%		
						33%		
					Normal	-	High	-
				Drive	-	-	Low	-

Mode：設定 I/O 模式。

- Touch-Key：觸摸鍵模式。僅支持 PA0~PD3。
- Normal-IO：目前該腳位為一般 I/O 功能。
- PWM-IO：目前該腳位為 PWM 輸出功能。僅支持 PE0~PF3。

Direction：設定初始時 I/O 的輸入/出方向。

- Input：初始時為輸入模式。
- Output：初始時為輸出模式。

Input Type：設定一般 I/O 的輸入模式中的下拉電阻。

- Weak-Pull-Low：有弱下拉電阻的輸入模式。設定成下拉電阻時，為高電平觸發。
- Strong-Pull-Low：有強下拉電阻的輸入模式。設定成下拉電阻時，為高電平觸發。
- Floating：無下拉電阻的輸入模式。

Output Type：設定輸出模式的輸出狀態。

- CMOS：為一般 CMOS 的 I/O 功能，僅設定成 Normal-IO 模式時提供。
- Open-Drain：無上拉電阻的開漏極 I/O 模式，僅設定成 Normal-IO 模式時提供。
- Sink：設定成 PWM-IO 模式時，LED 的輸出方式為 Active Low。
- Drive：設定成 PWM-IO 模式時，LED 的輸出方式為 Active High。

Sink Type：

- Normal：一般電流輸出。
- Constant：恒流輸出，僅支援 PE0~PF3。
- Large：大電流輸出，僅支援 PE0~PF3。

Sink-Current：設定電流輸出能力，提供四種不同的電流輸出模式，僅在 Constant 與 Large 模式時有檔案。

- 33%：提供 33%的電流輸出。
- 50%：提供 50%的電流輸出。
- 83%：提供 83%的電流輸出。
- 100%：提供 100%的電流輸出。

Initial：初始輸出電壓設定。

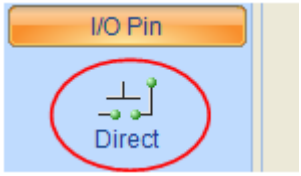
- High：初始輸出狀態為 Output High。
- Low：初始輸出狀態為 Output Low。

Busy：Action 信號的輸出方式。

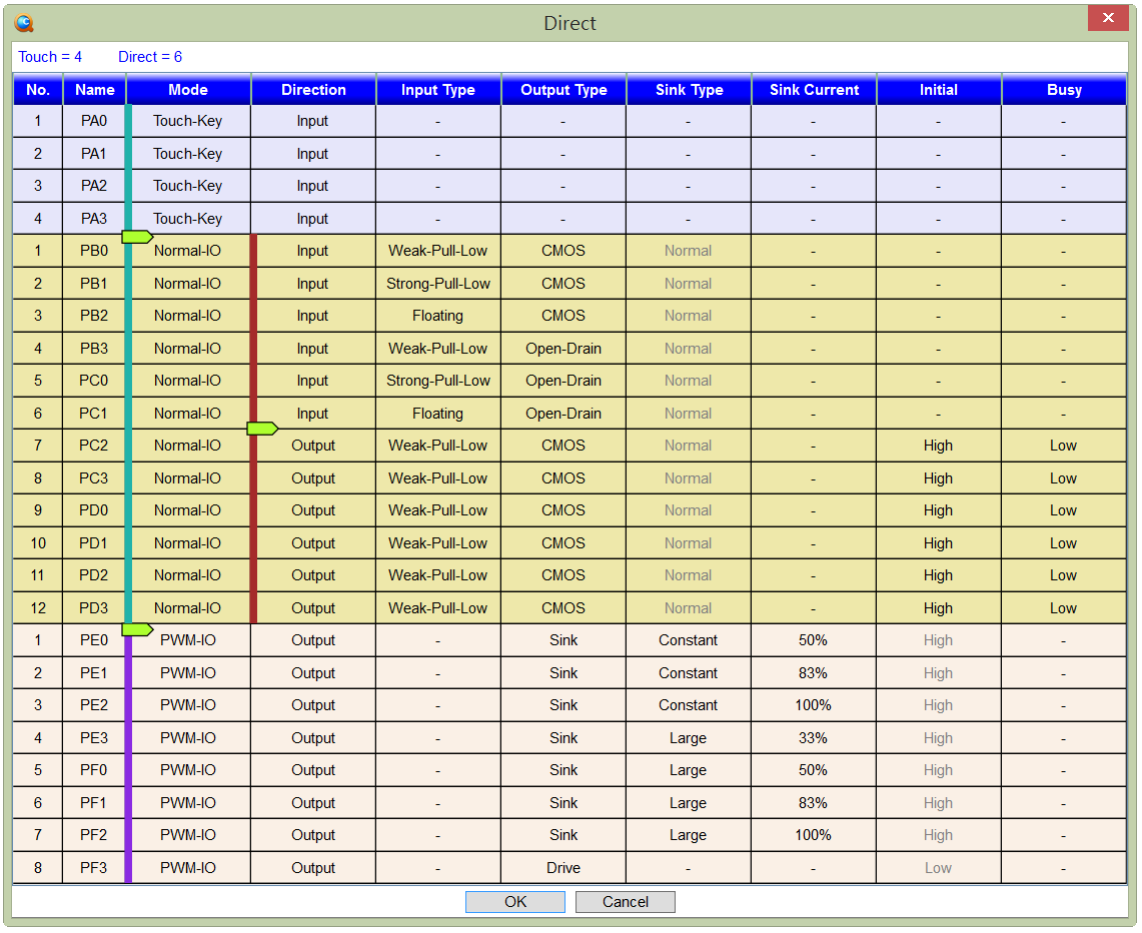
- Low：Action 為 Active Low（Sink Output）。
- High：Action 為 Active High（Drive Output）。

➢ ***Q-Code_NY9T* 中 I/O Pin 設定：**

按兩下 I/O Pin 區段 Direct ：



彈出 I/O Pin 設定界面：



No.	Name	Mode	Direction	Input Type	Output Type	Sink Type	Sink Current	Initial	Busy
1	PA0	Touch-Key	Input	-	-	-	-	-	-
2	PA1	Touch-Key	Input	-	-	-	-	-	-
3	PA2	Touch-Key	Input	-	-	-	-	-	-
4	PA3	Touch-Key	Input	-	-	-	-	-	-
1	PB0	Normal-IO	Input	Weak-Pull-Low	CMOS	Normal	-	-	-
2	PB1	Normal-IO	Input	Strong-Pull-Low	CMOS	Normal	-	-	-
3	PB2	Normal-IO	Input	Floating	CMOS	Normal	-	-	-
4	PB3	Normal-IO	Input	Weak-Pull-Low	Open-Drain	Normal	-	-	-
5	PC0	Normal-IO	Input	Strong-Pull-Low	Open-Drain	Normal	-	-	-
6	PC1	Normal-IO	Input	Floating	Open-Drain	Normal	-	-	-
7	PC2	Normal-IO	Output	Weak-Pull-Low	CMOS	Normal	-	High	Low
8	PC3	Normal-IO	Output	Weak-Pull-Low	CMOS	Normal	-	High	Low
9	PD0	Normal-IO	Output	Weak-Pull-Low	CMOS	Normal	-	High	Low
10	PD1	Normal-IO	Output	Weak-Pull-Low	CMOS	Normal	-	High	Low
11	PD2	Normal-IO	Output	Weak-Pull-Low	CMOS	Normal	-	High	Low
12	PD3	Normal-IO	Output	Weak-Pull-Low	CMOS	Normal	-	High	Low
1	PE0	PWM-IO	Output	-	Sink	Constant	50%	High	-
2	PE1	PWM-IO	Output	-	Sink	Constant	83%	High	-
3	PE2	PWM-IO	Output	-	Sink	Constant	100%	High	-
4	PE3	PWM-IO	Output	-	Sink	Large	33%	High	-
5	PF0	PWM-IO	Output	-	Sink	Large	50%	High	-
6	PF1	PWM-IO	Output	-	Sink	Large	83%	High	-
7	PF2	PWM-IO	Output	-	Sink	Large	100%	High	-
8	PF3	PWM-IO	Output	-	Drive	-	-	Low	-

由左而右，從 Mode 到 Current 依序設定。Mode 和 Direction 可通過拖曳綠色指標設定，其它選項通過滑鼠點擊該欄位來切換設定。點擊 OK，I/O Pin 設置完成。

2.1.3 Input State

格式	按鍵觸發型態	注　釋
X	忽略	無動作。
Path1	按下	當按鍵按下時，執行Path1路徑。
/Path2	放開	當按鍵放開時，執行Path2路徑。
Path1/Path2	按下&放開	當按鍵按下時，執行Path1路徑。 當按鍵放開時，執行Path2路徑。

2.1.4 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。
An	表示該腳位所對應到的ActionLabel為哪一個（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_NY9T* 中，使用者可以對觸摸按鍵或一般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。

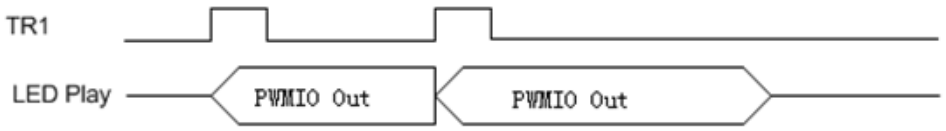
觸發模式縮寫	觸發模式	說明
E/U/R	Edge mode，Unhold，Retrigger	邊沿觸發，非保持，可重複觸發。
E/U/I	Edge mode，Unhold，Irretrigger	邊沿觸發，非保持，不可重複觸發。
L/U/R	Level mode，Unhold，Retrigger	電位觸發，非保持，可重複觸發。
L/U/I	Level mode，Unhold，Irretrigger	電位觸發，非保持，不可重複觸發。
E/H/X	Edge mode，Hold，X	邊沿觸發，保持。
L/H/X	Level mode，Hold，X	電位觸發，保持。
Toggle On/Off	Toggle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

注意：當為 Touch-Key 時，Touch 觸發表示' 1 '高電位，未觸發表示' 0 '低電位。

2.2.1 Edge mode, Unhold, Retrigger

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

功能：壓下或觸摸按鍵後開始播放 LED 閃法，使用者可以在 LED 閃爍時再 trigger，每一次 trigger 只會播放一種閃法。



例. PA0 (E/U/R) = PWM-IO

[Input State]

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

[Path]

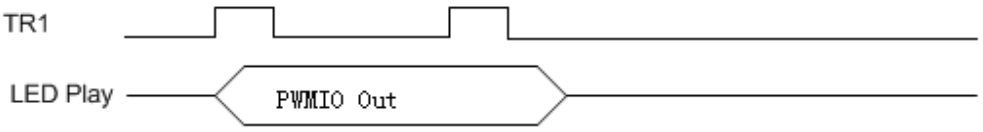
PowerOn: **TR1_Enable**, **IC_Init**

TR1: **PWMout**(@1100,40%,5s),**IC_Sleep**

2.2.2 Edge mode, Unhold, Irretrigger

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

功能：壓下或觸摸按鍵後開始播放 LED 閃法，使用者不可以在第一次閃法播放中再 trigger，必須等到第一次播放的閃法播放完畢後才能夠再 trigger，每一次 trigger 只會播放一次閃法。



例. PA0 (E/U/I) = PWM-IO

[Input State]

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

[Path]

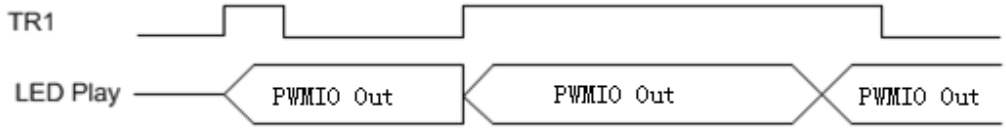
PowerOn: **TR1_Enable**, **IC_Init**

TR1: **TR1_Disable**, **PWMout**(@1100,40%,**15s**),**TR1_Enable**, **IC_Sleep**

2.2.3 Level mode, Unhold, Retrigger

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

功能：壓下或觸摸按鍵後開始播放 LED 閃法，使用者可以在 LED 閃法播放中再 trigger，如果按鍵一直被按著則 LED 閃法會一直重複播放。



例. PA0 (L/U/R) = PWM-IO

[Input State]

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

[Path]

PowerOn: **TR1_Enable, IC_Init**

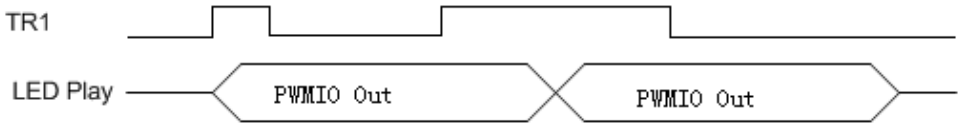
TR1F: **R0**=1, **PWMout**(20%,40%,80%,100%,**15s**),**R0**=1?TR1F, **IC_Sleep**

TR1R: **R0**=0

2.2.4 Level mode, Unhold, Irretrigger

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

功能：壓下或觸摸按鍵後開始播放 LED 閃法，使用者不可以在 LED 閃法播放中再 trigger，如果按鍵一直被按著則 LED 閃法會一直重複播放，按鍵放開後必須等到 LED 閃法播放完畢後才能夠 trigger。



例. PA0 (L/U/I) = PWM-IO

[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, **PWMout**(20%,40%,80%,100%,**15s**), **R0**=1?TR1F, **TR1_Enable**, **IC_Sleep**

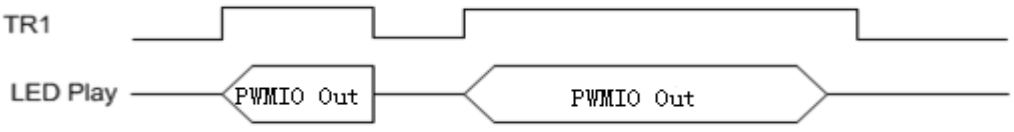
TR1R: **R0**=0

TR1F1: **R0**=1

2.2.5 Edge mode, Hold, X

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

功能：壓下或觸摸按鍵後開始播放 LED 閃法，放開按鍵後 LED 閃法將會停止播放，每一次的 trigger 只會播放一次 LED 閃法。



例. PA0 (E/H/X) = PWM-IO

[Input State]

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

[Path]

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

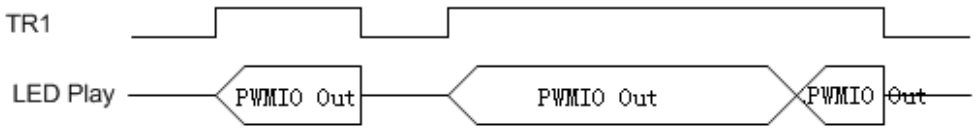
TR1F: **PWMout**(@1100,40%,**15s**)

TR1R: **IC_Sleep**

2.2.6 Level mode, Hold, X

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

功能：壓下或觸摸按鍵後開始播放 LED 閃法，當按鍵被按著時 LED 閃法會一直重複播放，放開按鍵後 LED 閃法將會停止播放。



例. PA0 (L/H/X) = PWM-IO

[Input State]

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

[Path]

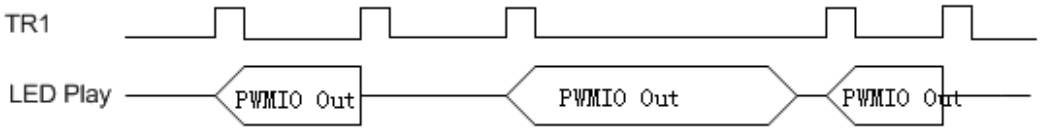
PowerOn: **TR1_Enable**, **IC_Init**

TR1F: **R0**=1, **PWMout**(@1100,80%,**15s**), **R0**=1?TR1F, **IC_Sleep**

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

2.2.7 Toggle On/Off

功能：按下或觸摸按鍵後開始播放 LED 閃法，當按鍵再次按下或觸摸時，LED 閃法停止，如果再次按下或觸摸按鍵時，LED 閃法又開始播放，如此重複。



例. PA0 (E/U/R, On/Off) = PWM-IO

[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**, **PWMout**(@1100,10%,**15s**), **TR1_Enable**, **IC_Sleep**

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

2.3 添加 Action

2.3.1 製作 PWM-IO 輸出的 Action 波形

當需要使用 PWM-IO 輸出 Action 波形時，先開啟 *Q-Visio* 並畫好每個 PWM-IO 需要輸出的 Action 波形。(具體操作請參考 *Q-Visio* UM)

2.3.2 添加 Action File

在 Action File 區段按兩下 Add File，添加.Vio 檔案。

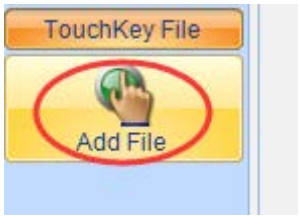
2.4 添加觸摸鍵 Touch-Key File

2.4.1 調試 Touch-Key 靈敏度

使用者可根據功能需要，設置不同的靈敏度。觸摸鍵靈敏度檔案由 *Q-Touch* 來編寫，調試好的靈敏度檔案保存副檔名為的.t9x，*Q-Touch* 使用請參考 *Q-Touch* 操作說明。

2.4.2 添加 TouchKey File

在 TouchKey File 區段按兩下 Add File，添加.t9x 檔案。



2.5 PWM-IO 輸出

2.5.1 使用 PlayPWM 指令播放.Vio 檔案

PlayA 指令為軟體輸出 Action 信號到一般 IO，而硬體 PWM-IO 播放 Action 信號使用的指令為 **PlayPWM / PlayPWMS(PWMEn, VIOLabel, Extension)** 。

PWMEn：腳位及檔案對應的方式是依照 **[I/O Pin]** 段落中所設定的 PWM 輸出腳位順序及 Action 檔案中的 Label "A1~A8" 的順序，最多對應至 PE0~PF3，共 8 個 pin。PWMEn=1 or 0，1=該腳位在播放 Action 檔案時，開啟 PWM 輸出功能；0=不啟動。(未填的部分會自動補 0)

a) 直接輸出到設定腳位 — *例.* **PlayPWM(@1111, $VIO0, 4)**

b) Ri (支持到 2 個 Ri) — *例.* **PlayPWM(R1:R0, $VIO0, 4)**

c) Xi (支持到 1 個 Xi) — *例.* **PlayPWM(X0, $VIO0, 4)**

d) 填 X 的話，會維持上一次的設定 — *例.* **PlayPWM(X, $VIO0, 4)**

注意：

1. ***PWMEn 參數的最左邊對應至設定成 PWM-IO 輸出腳位的最高位；Action 檔案則是 A1 對應到最低位。例如: @11111111 腳位從左至右對應到 PF3, PF2, PF1, PF0, PE3, PE2, PE1, PE0; 而腳位對應 Action 為: PF3 對應到 A1，PF2 對應到 A2，依此類推。***
2. ***Action 訊號會依照 PWM-IO 開啟的數量來對應，以 NY9T004A 為例, A1 會以 PE0 為優先對應腳位，若 PE0 未被設定成 PWM-IO，則對應至 PE1，依此類推。***

VIOLabel：加入多個 VIO 檔案時，須指定所要引用的 Action File 中的哪一組。

a) 直接設定要輸出的檔案 — *例.* **PlayPWM(@1111, $VIO0)**

b) Ri（支持到 4 個 Ri） — *例.* **PlayPWM(@1111, R3:R2:R1:R0)**

c) Xi（支持到 2 個 Xi） — *例.* **PlayPWM(@1111, X1:X0)**

注意：若是播放 VIO 檔案，且同時輸出多個訊號時，須注意各訊號的輸出時間長度必須相等，若時間長度不相等時，則會以輸出時間最長的訊號為基準，並且將其餘訊號輸出時間不足的部份強制輸出 0%。

Extension：可將該訊號整個依照倍數來延長，已節省 ROM Size。 Extension=1/2/4/8，1 倍可省略不寫。

a) 直接賦值。 — *例.* **PlayPWM(@1111, $VIO0, 4)**

b) Ri（支持到 1 個 Ri） — *例.* **PlayPWM(@1111, $VIO0, R0)**

c) 填寫 X 的話，會維持上一次的設定 — *例.* **PlayPWM(@1111, $VIO0, X)**

注意：Ri 的內容值必須為 1/2/4/8，非這四個數值時，皆會變成 1 倍。

例.

```
[Action File]
VIO0 = Action1.vio          ; Action 檔案中，含有 ActionLabel "A1~A6"的訊號。
VIO1 = Action2.vio          ; Action 檔案中，含有 ActionLabel "A1~A6"的訊號。
[Path]
Path1: PlayPWM(@1100, $VIO0)    ; 播放"VIO0"，輸出到 PE2, PE3。
```

```
Path2: PlayPWM(@0011, $VIO0)          ; 播放"VIO0"，輸出到 PE0, PE1。
Path3: PlayPWM(@00001111, $VIO1)      ; 播放"VIO1"，輸出到 PF0, PF1, PF2, PF3。
Path4: PlayPWM(@1111, $VIO1, 4)       ; VIO1 訊號延長 4 倍來播放，輸出到 PE0, PE1, PE2, PE3。
Path5: PlayPWMS(@1111, $VIO1), PlayA..    ; 播放"VIO1"後，立即執行 PlayA 指令。
Path6: PlayPWM(R1:R0,$VIO.A1,4)           ; 播放"VIO1"到 R1 和 R0 對應位置。
Path7: PlayPWM(X0,$VIO.A1,4)          ; 播放"VIO1"到 X0 數值對應位置。
```

2.5.2 使用 PWMOUT 指令直接輸出百分比或占空比階數

PWMOut(PWMEn, Percentage, Time): *Q-Code_NY9T* 提供使用者簡易的指令來達成 PWM-IO 的輸出方式。

PWMEn：腳位及檔案對應的方式是依照 **[I/O Pin]** 段落中所設定的 PWM 輸出腳位順序及 Action 檔案中的 Label "A1~A8" 的順序，最多對應至 PE0~PF3，共 8 個 pin。PWMEn=1 or 0，1=該腳位在播放 Action 檔案時，開啟 PWM 輸出功能；0=不啟動。（未填的部分會自動補 0）

a) 直接設定輸出腳位：**PWMOut(@1111,30%)**
b) Ri(支持到兩個 Ri)：**PWMOut(R1:R0,30%)**
c) Xi(支持到一個 Xi)：**PWMOut(X0,30%)**
d) 填寫 X 的話會維持上一次的資料：**PWMOut(X,30%)**

Percentage：設定輸出的百分比或階數。Ri=High-Byte，Rj=Low-Byte；暫存器可接受的範圍為 0~255。

a) 直接設定輸出的百分比（0%~100%）：**PWMOut(@1111,30%)**
b) 直接設定輸出的階數(0~255)：**PWMOut(@1111,127)**
c) Ri (支持到兩個 Ri)：**PWMOut(@1111,R1:R0)**
d) Xi (支持到一個 Xi)：**PWMOut(@1111,X1)**

Time：可以設定 PWM-IO 輸出的時間，不填寫的話預設為 16ms。(時間=16ms~30sec)

例.

```
[Path]
TR1: PWMOut(@1111, 30%)                    ; PE0~PE3 輸出 30%的亮度。
TR2: PWMOut(@1111, 30)                     ; PE0~PE3 輸出 30/256 的亮度。
TR3: R1=0x1, R0=0xE, PWMOut(@1111, R1:R0); PE0~PE3 輸出 30/256 的亮度。
TR4: X0=0x1E, PWMOut(@1111,X0)             ; PE0~PE3 輸出 30/256 的亮度。
TR5: PWMOut(@1111, 30%, 16ms)              ; PE0~PE3 輸出 30%亮度及播放 16 毫秒的時間。
TR6: PWMOut(@1111, 30, 2s)                 ; PE0~PE3 輸出 30/256 的亮度及 2 秒的時間。
```

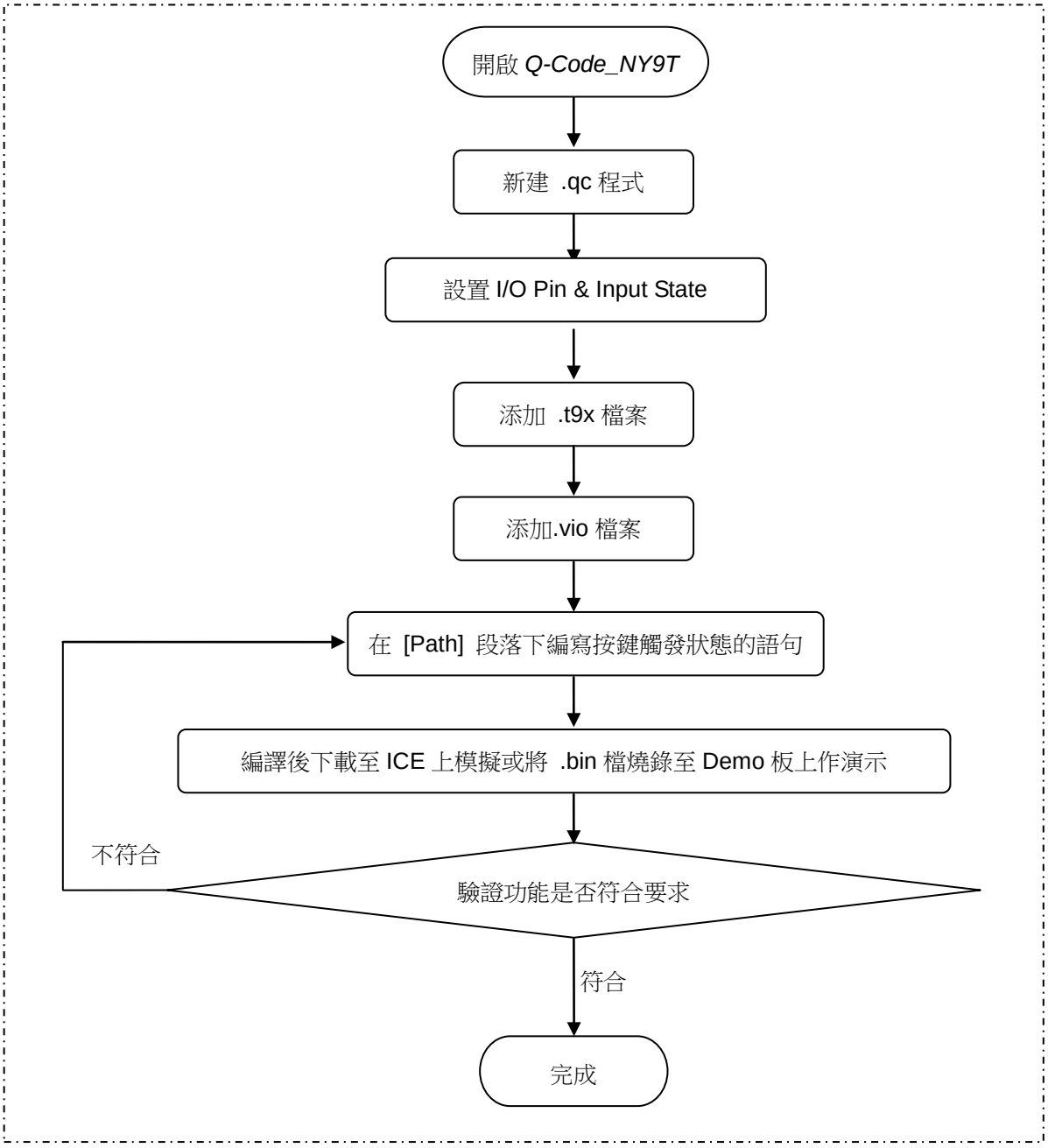
PWMCtrl(PWMEn,Extension): 可以開關在[I/O:Pin]段落中已被設定的 PWM-IO 腳位。PWMEn 的使用方式同 PlayPWM 指令。

HoldPwm: 保持 PWMOut 的最後狀態，並且 IC 不會進入睡眠。

PausePwm(n)?Path: n 為 0,1,2。若 HoldPwm 在該通道處於打開狀態，則跳轉至 Path 路徑。

3 應用實例

3.1 設計流程圖



3.2 功能介紹

<table>
<tr><td colspan="2">功能說明：演示Touch-Key不同觸發組合模式和 PWM-IO不同輸出方式。</td></tr>
<tr><td colspan="2">PA0：Touch-Key
PA1/PA2/PA3：Option 選擇
PE0/PE1：Normal I/O (PE0 Busy High,PE1 Busy Low)
PE2/PE3：PWM-IO (PE2 Drive輸出，PE3 Constant Sink 輸出)</td></tr>
<tr><td rowspan="7">TR1 鍵 (PA0)</td><td>E/U/R [PA3: PA2: PA1] = [0: 0: 0]，則觸摸PA.0後開始播放EUR_Visio.vio，在LED Action播放中可以再次觸發該按鍵。</td></tr>
<tr><td>E/U/I [PA3: PA2: PA1] = [0: 0: 1]，則觸發PA.0後開始播放PWMout，在LED播放中不可以再次觸發該按鍵，必須等到LED播放完畢後才能夠再次觸發。</td></tr>
<tr><td>L/U/R [PA3: PA2: PA1] = [0: 1: 0]，則觸發PA.0後開始播放PWMout，在LED Action播放中可以再次觸發該按鍵，如果按鍵一直被按著則LED Action會一直重複播放。</td></tr>
<tr><td>L/U/I [PA3: PA2: PA1] = [0: 1: 1]，則觸發PA.0後播放LUI_Visio.vio，在LED Action播放中不可以再次觸發該按鍵，如果按鍵一直被按著則LED Action會一直重複播放。</td></tr>
<tr><td>E/H/X [PA3: PA2: PA1] = [1: 0: 0]，則觸發PA.0後開始播放LHX_Visio.vio，放開按鍵後LED 將會立即停止播放，每一次的觸發只會播放一次 LED。</td></tr>
<tr><td>L/H/X [PA3: PA2: PA1] = [1: 0: 1]，則觸發PA.0後開始播放LHX_Visio.vio，如果按鍵一直被按著則LED Action會一直重複播放，放開按鍵後LED Action將會停止播放。</td></tr>
<tr><td>ON/OFF [PA3: PA2: PA1] =[1: 1: 0]，按鍵一次會播放，再按一次會停止。</td></tr>
</table>

3.3 *Q-Code_NY9T* 程式

```
; ------------------------------------------------------------------------------
; 基本的 Option 選項設定
[Option]
ICBody = NY9T004A
Client = Nyquest
Voltage = 3.0V
TouchKey_DebounceTime = 32ms{SampleTime=1ms, FrameRate=16ms, Debounce=2}
TouchKey_DetectTime = 80ms
TouchKey_Detection = Multi-Key
TouchKey_Wakeup = AnyKey
AutoJudge_Calibrate = 4s
Enforce_Calibrate = 40s
SW_Reset = Enable
; ------------------------------------------------------------------------------
; 添加 Visio 檔案
[Action File]
VIO0 = EUR_Visio.vio
VIO1 = LHX_Visio.vio
VIO2 = LUI_Visio.vio
; ------------------------------------------------------------------------------
; 添加觸摸按鍵 TouchKey File
[TouchKey File]
; Touch-Key Number = 2
; Scan Mode          = Preset
QTouch_001.t9x
; ------------------------------------------------------------------------------
; I/O 設定
[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
;       Name   PE0    PE1    PE2    PE3
```

```
Touch = 1
Direct = 3
Input_Mode = [P:W:CMOS:N P:W:CMOS:N P:W:CMOS:N ]
Output_Mode = [P:W:CMOS:N:L:BL P:W:CMOS:CS_100%:H:BL PWM:D:L PWM:CS_100%:H ]
;-------------------------------------------------------------------------
; 設置 Input 狀態
[Input State]
Input_X:     X       X     X     X
Input_EUR: EUR     X     X     X
Input_EUI:  EUI     X     X     X
Input_LUR: LUR_F/LUR_R     X     X     X
Input_LUI:  LUI_F/LUI_R     X     X     X
Input_LUI_2: LUI_F_2/LUI_R     X     X     X
Input_EHX: EHX_F/EHX_R     X     X     X
Input_LHX: LHX_F/LHX_R     X     X     X
Input_On:   TR1ON     X     X     X
Input_Off:   TR1OFF     X     X     X
;-------------------------------------------------------------------------
; 設置 Output 狀態
[Output State]
Output_Init:     0     1                    ; 初始輸出狀態
Output_Busy:    1     0                    ; Busy 輸出狀態
;-------------------------------------------------------------------------
[Symbol]
Key_Flag = R0
; Key_Flag.0 = R0.0          LUR  是否按下標誌位
; Key_Flag.1 = R0.1          LUI  是否按下標誌位
; KEY_Flag.2 = R0.2          LHX  是否按下標誌位
; -------------------------------------------------------------------------
; Main 主程序
[Path]
PowerOn:
  Output_Init,                          ; PE0=Low,PE1=High
  delay(20ms),
  ReadRadj(R1),                         ; 檢查外部 Radj 電阻
  R1=0XF?L_Sensitivity_Typical,         ; Radj 無外部電阻,程式指定一階 Sensitivity
  TouchKey_Sensitivity(Radj),           ; Radj 外部有接電阻，根據電阻設定 Sensitivity
  L_Option
```

```
L_Sensitivity_Typical:
  Touchkey_Sensitivity(4),                    ; 程式指定 Sensitivity 為 typical 值(4)
  L_Option
L_Option:
  R1=0,
  R1=PA,
  PAM=0x1,                                    ; PA1/PA2/PA3  切換為輸出模式
  PA=R1,                                      ; 將 Option 狀態重新賦值回 PA1/PA2/PA3，防止 I/O 漏電
  R1=R1&0XE,
  R1=R1>>1,                                   ; 上電選擇輸出模式
  Switch(R1)=[Mode_EUR,Mode_EUI,Mode_LUR,Mode_LUI,Mode_EHX,Mode_LHX,Mode_ONOFF]
; PA3 PA2 PA1
;   0    0    0      E/U/R Mode
;   0    0    1      E/U/I Mode
;   0    1    0      L/U/R Mode
;   0    1    1      L/U/I Mode
;   1    0    0      E/H/x Mode
;   1    0    1      L/H/x Mode
;   1    1    0      ON/OFF Mode
; ------------------------------------------------------------------------
; E/U/R 模式：按鍵可重複觸發
Mode_EUR:
  Input_EUR,
  L_End
;-------
EUR:
  Output_Busy,                                ; PE0 Busy High,PE1 Busy Low
  PlayPWM(@11,$VIO0),                         ; PE2/PE3  根據 Action 漸變
  L_End


; ------------------------------------------------------------------------
; E/U/R 模式：按鍵不可重複觸發
Mode_EUI:
  Input_EUI,
  L_End
;-------
EUI:
  Input_X,                                    ; 設定按鍵無效
```

```
  Output_Busy,                    ; PE0 Busy High,PE1 Busy Low
  X1=0,                           ; PWMOUT 直接改變階數，PE2/PE3 輸出漸變效果
  L_Ascend
L_Ascend:
  X1++,
  PWMOUT(@11,X1),                 ; 每階停留時間為 2*Frame=15.6ms
  X1=255?L_Descend,               ; 最大亮度開始漸滅
  L_Ascend                        ; 漸亮
L_Descend:
  X1--,
  PWMOUT(@11,X1),
  X1=0?L_EUI_End,                 ; 漸變結束
  L_Descend                       ; 漸滅
L_EUI_End:
  Input_EUR,                      ; 設定按鍵有效
  L_End

; ---------------------------------------------------------------------------
; L/U/R 模式：按鍵可重複觸發(按鍵保持壓下會一直迴圈)
Mode_LUR:
  Input_LUR,
  L_End
;-----------
LUR_F:
  Output_Busy,                    ; PE0 Busy High,PE1 Busy Low
  Key_Flag.0=1,                   ; LUR 按鍵壓下標誌
  PWMOUT(20%,80%,5s),             ; 使用 PWMOut 的方式輸出 PE2 20%亮度，PE3 80%亮度
  Key_Flag.0=1?LUR_F,             ; 判斷按鍵標誌位，標誌位為 1(按鍵按下)，則迴圈
  L_End

LUR_R:
  Key_Flag.0=0                    ; LUR 按鍵鬆開標誌。

; ---------------------------------------------------------------------------
; L/U/I 模式：按鍵不可重複觸發(按鍵保持壓下會一直迴圈)
Mode_LUI:
  Input_LUI,
  L_end
```

```
;-----------
LUI_F:
  Input_LUI_2,                    ; 設定按鍵按下不重複觸發，但重新設置按鍵壓下標誌
  Output_Busy,                    ; PE0 Busy High,PE1 Busy Low
  Key_Flag.1=1,                   ; LUI 按鍵壓下標誌位
  PlayPWM(@11,$VIO2),             ; Action 輸出 PE2 20%亮度，PE380%亮度
  Key_Flag.1=1?LUI_F,             ; 判斷 LUI 標誌位，為 1 則迴圈播放
  Input_LUI,                      ; 設定按鍵按下有效
  L_End
LUI_R:
  Key_Flag.1=0                    ; LUI 按鍵鬆開標誌位
LUI_F_2:
  Key_Flag.1=1                    ; LUI 按鍵壓下標誌位

; ------------------------------------------------------------------------
; E/H/x  模式：按鍵按住播放一次，按鍵鬆開結束
Mode_EHX:
  Input_EHX,
  L_End
;-----------
EHX_F:
  Output_Busy,                    ; PE0 Busy High,PE1 Busy Low
  PlayPWM(@11,$VIO1),             ; Action 輸出 3HZ 閃爍,波形長度為五秒
  L_End
EHX_R:
  L_end                           ; 鬆開按鍵，停止所有動作，IC 睡眠

; ------------------------------------------------------------------------
; L/H/x  模式：按鍵保持按住狀態一直迴圈播放，按鍵鬆開結束
Mode_LHX:
  Input_LHX,
  L_End
;-----------
LHX_F:
  Output_Busy,                    ; PE0 Busy High,PE1 Busy Low
  Key_Flag.2=1,                   ; LHX 按鍵標誌位
  PlayPWM(@11,$VIO1),             ; Action 輸出 3HZ 閃爍,波形長度為五秒
  Key_Flag.2=1?LHX_F,             ; 若 Key_Flag 標誌位為 1（即按下狀態），則迴圈
```

```
  L_End                                  ; 否則，跳轉至 Sleep，停止所有動作，IC 睡眠
LHX_R:
  Key_Flag.2=0,                          ; LHX 按鍵釋放標誌位
  L_End

; ---------------------------------------------------------------------------
;ON/OFF 模式：
Mode_ONOFF:
  Input_On,
  L_End
;-------------
TR1ON:
  Input_Off,                             ; 設定按鍵 OFF 有效
  Output_Busy,                           ; PE0 Busy High,PE1 Busy Low
  L_3Hz                                  ; PE2/PE3 3Hz 閃爍 5 秒
L_3Hz:
  PWMOUT(@11,100%,167ms),
  PWMOUT(@11,0%,167ms),
  X1++,
  X1>=15?TR1OFF,L_3Hz
TR1OFF:
  Input_On,                              ; 設定按鍵 ON 有效
  X1=0,
  L_End

; ---------------------------------------------------------------------------
; 程式結束路徑
L_End:
  Output_Init,                           ; PE0=Low,PE1=High
  Stop                                   ; 所有標誌位清零，停止 IC 所有動作。

; ---------------------------------------------------------------------------
; Q-Code_NY9T 預設路徑，睡眠前自動執行該路徑一次
Sleep:
  TouchKey_Scan_Slow                     ; IC Sleep 前 Touch 進入 Slow Mode
                                         ; IC 喚醒後，Touch 自動進入 Normal
```

4 Q-Code_NY9T 指令

Q-Code_NY9T 指令表及說明如下。

4.1 算數邏輯指令 (Arithmetic Command)

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 位為 1，R0 的第 0 位為 0

Ex3: R0.2=1, R1=0x8, R1.1=R0.2 ; 賦值操作，R1 最終的值為 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 位為 1, X0 的第 0 位為 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 為 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 條件跳轉指令 (Condition Jump Command)

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	Delay(n)?Path	Action(Ch)?Path
PWMIO(n)?Path	PauseD(n)?Path	PauseA(Ch)?Path	Pause(n)?Path	CheckSum?Path
Calibrate?Path	PausePWM(n)?Path	HoldPWM(n)?Path	RandomL = data?Path	
RandomH = data?Path	Switch(Px[d x d x]) = [Path0, Path1, Path2….Path15]			
Switch(Ri) = [Path0, Path1, Path2,..Path15]	Switch(Px) = [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: 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

Ex25: 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

Ex26: PWMIO(0)?Path1, Path2 ; 前景 PlayPWM/PlayPWMS 執行中則跳至 Path1，反之，程式跳至 Path2
; 0 代表前景，1 代表背景 1，2 代表背景 2

Ex27: PauseD(0)?Path1, Path2 ; 前景 Delay 暫停執行中，則跳至 Path1，反之，程式跳至 Path2
; 0 代表前景，1 代表背景 1，2 代表背景 2

Ex28: PauseA(1)?ResumeAction ; Action 通道 1 正在暫停播放中，則跳至“ResumeAction”Path
; 0~7 共 8 個通道

Ex29: Pause(0)?Path1, Path2 ; 前景暫停執行中，則跳至 Path1，反之，程式跳至 Path2
; 0 代表前景，1 代表背景 1，2 代表背景 2

Ex30: 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

Ex31: Calibrate?Path1, Path2 ; 自動觸摸功能 AutoJudge_Calibrate 或強制觸摸功能 Enforce_Calibrate，
; 執行中則跳轉至 Path1，否則跳轉至 Path2.

Ex32: PausePWM(0)?Path1 ; 前景 PlayPWM/PlayPWMS 暫停中，則跳至 Path1

Ex33: HoldPWM(0)?Path1 ; 前景 PlayPWM/PlayPWMS 暫停中，則跳至 Path1

Ex34: RandomL=0x6?Path1,Path2 ; 判斷亂數的低四位是否等於 0x6，等於則跳至 Path1,
; 反之, 跳至 Path2

Ex35: RandomH=0xB?Path1,Path2 ; 判斷亂數的高四位是否等於 0xB，等於則跳至 Path1,
; 反之, 跳至 Path2

Ex36: Switch(R0)=[Path0, Path1, Path2,..Path15] ; R0=2，此時程式執行 Path2

Ex37: Switch(PA)=[Path0, Path1, Path2,..Path15] ; 如果 PA=2，此時程式執行 Path2

Ex38: Switch(PA[d d d d])=[Path0, Path1, Path2,..Path15] ; 如果 PA=0xF，此時程式執行 Path15

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

Ex40: X0!=X1?Path1, Path2 ; X0 不等於 X1 程式跳至 Path1，否則程式跳至 Path2

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

Ex42: X0<=X1?Path1, Path2 ; X0 小於或等於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex43: X0>X1?Path1, Path2 ; X0 大於 X1 程式跳至 Path1，否則程式跳至 Path2

Ex44: X0<X1?Path1, Path2 ; X0 小於 X1 程式跳至 Path1，否則程式跳至 Path2

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

Ex46: X0!=0x54?Path1, Path2 ; X0 不等於 0x54 程式跳至 Path1，否則程式跳至 Path2

Ex47: X0>=0xC?Path1, Path2 ; X0 大於或等於 0xC 程式跳至 Path1，否則程式跳至 Path2

Ex48: X0<=0x91?Path1, Path2 ; X0 小於或等於 0x91 程式跳至 Path1，否則程式跳至 Path2

Ex49: X0>0x3D?Path1, Path2 ; X0 大於 0x3D 程式跳至 Path1，否則程式跳至 Path2

Ex50: X0<0xF3?Path1, Path2 ; X0 小於 0xF3 程式跳至 Path1，否則程式跳至 Path2

Ex51: X0.1=0?Path1, Path2 ; X0.1 等於 0 則程式跳至 Path1，否則程式跳至 Path2

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

Ex53: X0.1!=0?Path1, Path2 ; X0.1 不等於 0 則程式跳至 Path1，否則程式跳至 Path2

Ex54: X0.1!=1?Path1, Path2 ; X0.1 不等於 1 則程式跳至 Path1，否則程式跳至 Path2

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

Ex56: Random!=250?Path1, Path2 ; 如果 Random 不等於 250，則程式執行 Path1，否則執行 Path2

Ex57: Switch(X0)=[Path0, Path1, Path2,..Path255] ; X0=220，此時程式執行 Path220

Ex58: Switch(Random)=[Path0, Path1, Path2,..Path255] ; Random=220，此時程式執行 Path220

4.3 I/O 指令 (I/O Command)

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	XiL.n = Px.n	XiH.n = Px.n	Xi.n = Px.n
Xi = [Px, Py]	Px = XiL	Px = XiH	Px.n=XiL.n	Px.n = XiH.n
Px.n = Xi.n	[Px, Py] = Xi	-	-	-

以下指令都會適用的範圍為：NY9T001A/004A series：x = A/E，NY9T008A series：x = A/B/E/F，NY9T016A series：x = A~F。

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 的位為輸出，等於 1 的位為輸入）

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 1 x]                       ; 將 PB.1 輸出 1，將 PB.2 輸出 0，將 PB0 和 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: PB=0x5, X0=0x0, X0L.2 = PB.0       ; 將 PB.0 的值賦給 X0L.2 後，X0L = 0x4
Ex26: PB=0x5, X0=0x0, X0H.2 = PB.0       ; 將 PB.0 的值賦給 X0H.2 後，XiH = 0x4
Ex27: PB=0x5, X0=0x0, X0.2 = PB.0        ; 將 PB.0 的值賦給 X0.2 後，X0 = 0x4
Ex28: X0=[PA, PB]                        ; 將 PA、PB 的值讀入 X0，X0 低位讀入 PB 的值，X0 高位讀入 PA 的值
Ex29: X0L=0x3, PB=X0L                    ; 將 X0L 的值寫入到 PB 後，PB = 0x3
Ex30: X0H=0x3, PB=X0H                    ; 將 X0H 的值寫入到 PB 後，PB = 0x3
Ex31: PB=0x0, X0=0x5, PB.2=X0L.0         ; 將 X0L.0 的值賦給 PB.2 後，PB = 0x4
Ex32: PB=0x0, X0=0x50, PB.2=X0H.0        ; 將 X0H.0 的值賦給 PB.2 後，PB = 0x4
Ex33: PB=0x0, X0=0x5, PB.2=X0.0          ; 將 X0.0 的值賦給 PB.2 後，PB = 0x4
Ex34: X0H=0x3, X0L=0x1, [PA, PB]=X0      ; 將 X0 的值寫入 PA、PB，X0 低位的值寫入 PB，X0 高位的
                                         ; 值寫入 PA 後，PA = 0x3，PB = 0x1
```

4.4 路徑指令 (Path Command)

Path Command				
ASM	BG(BG1,BG2)	Break	StopFG	StopBG
StopBG1	StopBG2	Macro	Subroutine	Label(Pathname)
1ms_RET	4ms_RET	-	-	-

Ex1: ASM, Macro, Subroutine

; 1) ASM 是插入彙編的界面

; 2) Macro 是巨集的界面

; 3) Subroutine 是副程式的界面

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: BG1：Break ; 結束當前前景運行的 PlayA、PlayPWM 或 Delay 指令

Ex6: StopFG ; 停止所有前景運行的動作

Ex7: StopBG ; 停止所有背景運行的動作

Ex8: StopBG1 ; 停止所有背景 1 運行的動作

Ex9: StopBG2 ; 停止所有背景 2 運行的動作

Ex10: PlayA(Ch1,$VIO0), Label(Loop_Path), PlayA(Ch1,$VIO1), Loop_Path

; 播放完$VIO0 後，不停播放$VIO1

Ex11：1ms: !PA, 1MS_RET ; 執行完!PA 後，執行 1MS_RET 時，會返回主程序

Ex12：4ms: !PA, 4MS_RET ; 執行完!PA 後，執行 4MS_RET 時，會返回主程序

4.5 查表指令 (Table Command)

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)

TableL(TableName, Rx, Ry, Ri) 指令為間接定址指令，可用來讀取在 [Table] 段落中定義數值的 Low-Byte。

TableH、TableM、Table 功能類似。

TableName：是一個在 [Table] 段落中以字元定義的 table 名稱。

Rx：Rx 內容值代表 X 軸的位址。

Ry：Ry 內容值代表 Y 軸的位址。

Ri：將 Table 取到的數據放入 Ri 內。 X 軸和 Y 軸的數值範圍是（0~15）。

```
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 ]
    }
```

4.6 時間延遲指令 (Delay Command)

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), OUT ; 等背景 1 的延時結束以後再執行 OUT 路徑

; (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，該條指令為閉環計時，會影響按鍵掃描等

4.7 動作指令 (Action Command)

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), OUT1 ; 播放 Action 馬上執行 OUT1

Ex3: WaitAN(1), OUT2 ; 等背景 1 的 Action 結束以後再播放 OUT2，

; (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

4.8 脈衝調變 IO 指令 (PWM-IO Command)

PWM-IO Command				
PlayPWM	PlayPWMS	WaitPN	PausePWM	HoldPWM
ResumePWM	StopPWM	PWMOut	PWMCtrl	-

Ex1: PlayPWM(@1111, $VIO1, 4) ; VIO1 信號延長 4 倍來播放，輸出到 PE0, PE1, PE2, PE3

Ex2: PlayPWMS(@1111, $VIO1), PlayA… ; 播放"VIO1"後，立即執行 PlayA 指令

Ex3: WaitPN, BG1 ; 若 PWM-IO 正在執行，則 PWM-IO 執行完後，執行背景

Ex4: PausePWM ; 暫停執行 PWM-IO 輸出，IC 會進入睡眠

Ex5: ResumePWM ; 恢復 PWM-IO 輸出

Ex6: HoldPWM ; 暫停執行 PWM-IO 輸出，PWM-IO 維持目前輸出狀態

Ex7: StopPWM ; 停止執行 PWM-IO 輸出

Ex8: PWMOut(@1111, 30, 2s) ; PE0~PE3 輸出 30/256 的亮度及 2 秒的時間

Ex9: PWMCtrl(X, 4) ; 改變 Extension 為 4 倍

4.9 觸摸鍵指令 (Touch-Key Command)

Touch-Key Command			
TouchKey_ON	TouchKey_OFF	TouchKey_CLR	TouchKey_Scan_Slow
TouchKey_Scan_Normal	TouchKey_Sensitivity	Calibrate_ON	Calibrate_OFF
AutoJudge_Calibrate	Enforce_Calibrate	-	-

Ex1: TouchKey_ON ; 開啟觸摸鍵掃描功能，打開則持續掃描

Ex2: TouchKey_OFF ; 關閉觸摸鍵掃描功能，關閉則不掃描

Ex3: TouchKey_CLR ; 清除當前觸摸鍵狀態

Ex4: TouchKey_Scan_Slow ; 將觸摸鍵掃描切換成慢速掃描模式，可以省電

Ex5: TouchKey_Scan_Normal ; 將觸摸鍵掃描切換成一般掃描模式

Ex6: TouchKey_Sensitivity(2) ; 切換觸摸鍵靈敏度為第 2 階，共有 8 段(0~7)可調整

Ex7: Calibrate_ON ; 開啟觸摸鍵的靈敏度校正功能（默認為打開）

Ex8: Calibrate_OFF ; 關閉觸摸鍵的靈敏度校正功能

Ex9: 4Sec: KeyPress=1?Check40sec, AutoJudge_Calibrate

Check40sec: Count++, Count=10?Timeout_40sec

Timeout_40sec: Count=0, Enforce_Calibrate

；每 4 秒進行一次自動校正，當按鍵持續被按下 40 秒後，會強迫進行校正一次

4.10 串列控制指令（Serial Control Command）

Serial Control Command		
SC_TX(Mode, data)	SC_TX(Mode, Ri:Rj:Rl:Rk)	SC_TX(Mode, Xi:Xj)

Ex1: SC_TX(Auto, 0xFFFF) ; 自動偵測使用的傳輸協定，輸出立即值為 0xFFFF 的訊號

Ex2: SC_TX(SPI_Like, 0xFFFF) ; 使用 SPI_Like 傳輸協定，輸出立即值為 0 的訊號

Ex3: R3=0x0, R2=0, R1=0, R0=0, SC_TX(NY3, R3:R2:R1:R0)

; 使用 NY3 Serial_Trigger 傳輸協定，輸出[R3:R2:R1:R0] 暫存器內容值的訊號

Ex4: R3=0x0, R2=0, R1=0, R0=0, SC_TX(IR_Trigger, X1:X0)

; 使用 IR_Trigger 傳輸協定，輸出[X1:X0] 暫存器內容值的訊號

4.11 一般指令 (MISC Command)

MISC Command				
Input State	Key_ON	Key_OFF	Key_CLR	Debounce(Time)
ReadRadj(Ri)	ReadRollingCode	Stop	Pause(n)	Resume(n)
WDT_CLR	SW_Reset	Repeat	END	-

Ex1: KEY1: TR1 TR2 X /TR4 ; 在[Input State]設置 KEY1 狀態

Ex2: KEY_ON ; 開啟按鍵掃描功能

Ex3: KEY_OFF ; 關閉按鍵掃描功能

Ex4: KEY_CLR ; 清除當前按鍵狀態，重新掃描按鍵

Ex5: Debounce(16ms) or Debounce(0.016) ; 設定目前的 Debounce 時間為 16ms

Ex6: ReadRadj(Ri) ; 讀取外部電阻值，指令會回傳 0x0~0x7 數值，若沒有接外部電阻值時,
; 則回傳 0xF

Ex7: Stop ; 結束所有的動作

Ex8: 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

Ex9: 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

Ex10: WDT_CLR ; 清看門狗

Ex11: {PlayPWM(1100, $VIO0), Delay(2) }*3 ; 重複播放，“{ }” 內的程式會執行 3 次

Ex12: END ; 整潔代碼，無實在意義，表示程式結束。

Ex13: Enforce_Calibrate: SW_Reset ; 強制校正以後，SW_Reset（Software Reset）可恢復
; 成上電時的初始狀態

5 *Q-Touch* 操作說明

NY9T 使用電容式觸控。基本原理為按鍵帶有電容，而人體也接地帶電，觸碰按鍵後會造成電容值改變，因此可通過電容的變化來判斷按鍵是否被按下。按鍵前後的電容變化量會因以下條件改變而不同：

- 按鍵的大小：按鍵越小，則電容變化量會越小，反之按鍵越大，電容變化量會越大。但若按鍵大小超過與手指的接觸面積，則按鍵越大，所得到的電容變化量會越小。
- 介質的厚度：厚度越厚，則電容變化量越小，反之厚度越薄，電容變化量會越大。
- 介質的材質：不同的材質會有不同的介電係數，將會影響所得到的電容變化量。

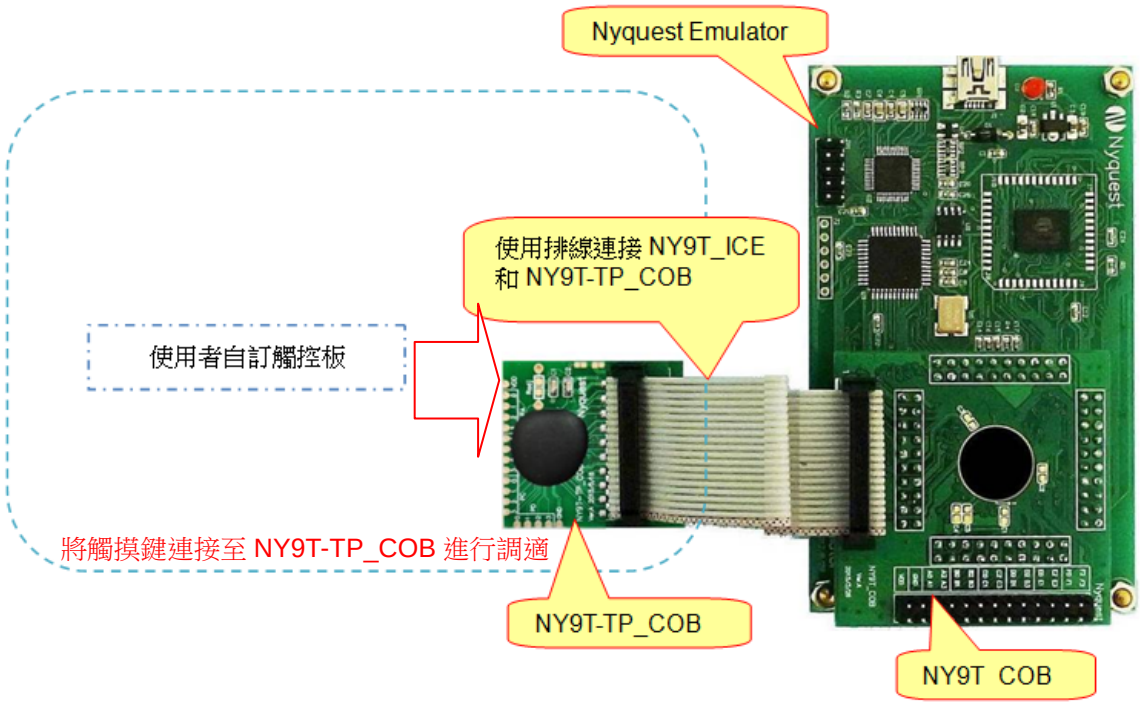
Q-Touch 中的 Preset 模式，會去計算各按鍵被按下前後的電容變化量，以算出的平均電容變化量，如此可不受按鍵電容改變產生的影響（例如走線長度、走線方式等的變化），能不受生產流程中可能產生的變化影響，因此建議使用此選項。若應用為各按鍵之間的介質厚度不同或按鍵大小不同，仍可通過調整其它條件來滿足電容變化量固定的條件（例如若某些按鍵的介質較厚，就可放大按鍵部分，達到電容變化量相同的效果）。

Q-Touch 中的 Custom 模式則是根據目前的環境，去調整每個按鍵的靈敏度。優點在於環境變化不大的情況下，能夠設置出精準的靈敏度條件。但是若在生產流程中，對於環境有較大的變更，都可能使得此組靈敏度無法正確運作，必須重新使用 *Q-Touch* 進行設置。

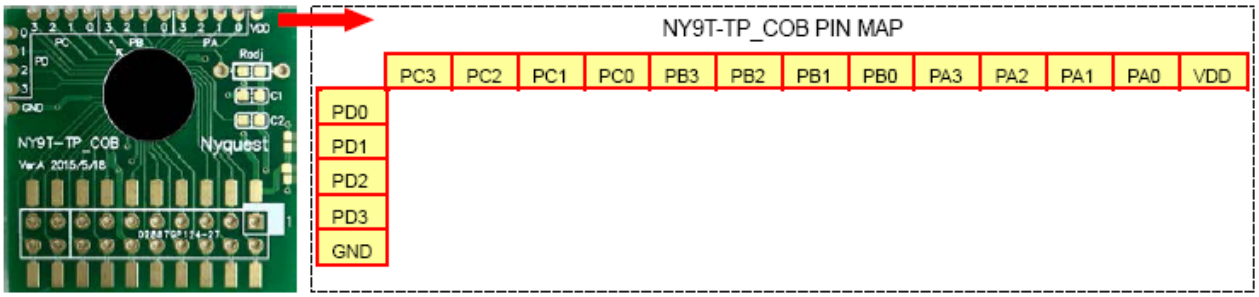
5.1 硬體設備

NY9T_ICE 和 NY9T-TP_COB。

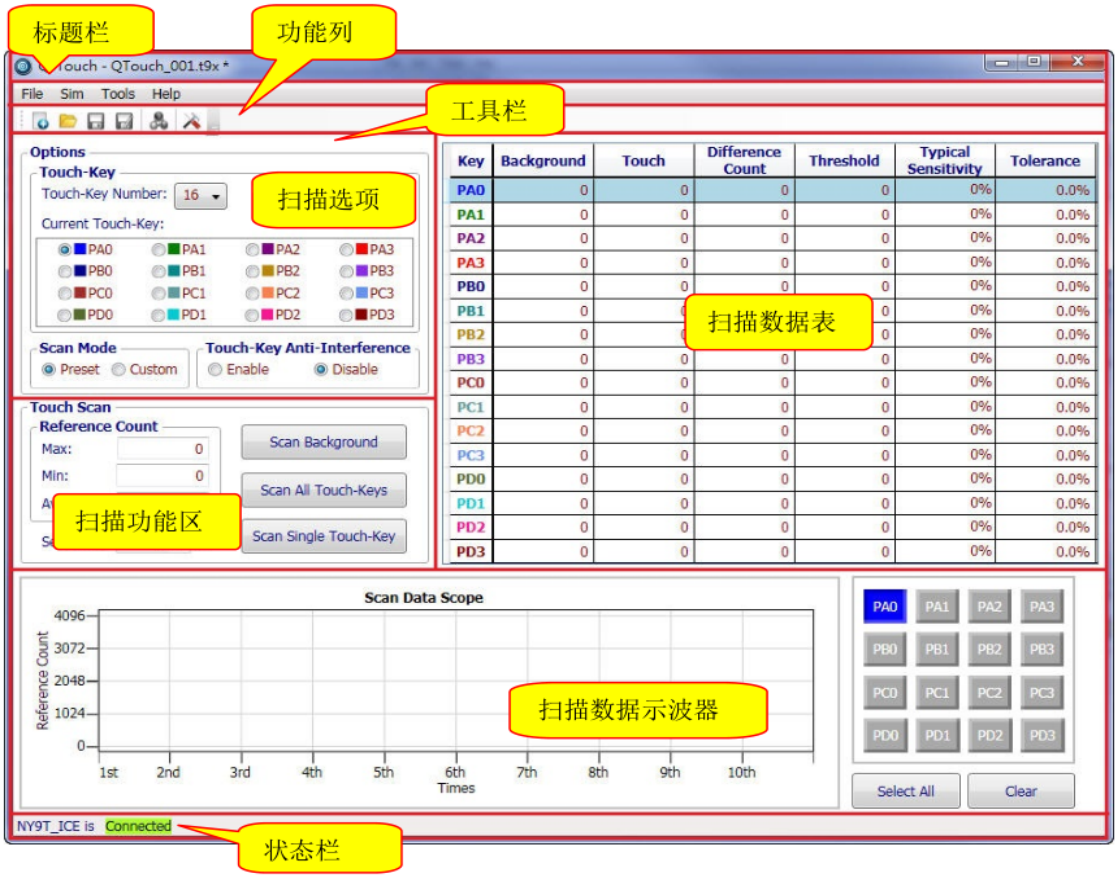
NY9T 開發環境：



根據實際應用的Touch-Key個數，依次從NY9T-TP_COB的PA0連線到Touch PAD。



5.2 軟體 *Q-Touch* 介紹



Q-Touch 主畫面如上圖，相關功能說明，請參考下列介紹。

標題列：程式名(*Q-Touch*) - 現在打開的檔案名稱。

功能列：所有功能列表。

工具列：常用到的功能按鈕。

掃描選項：與掃描操作相關的選項。

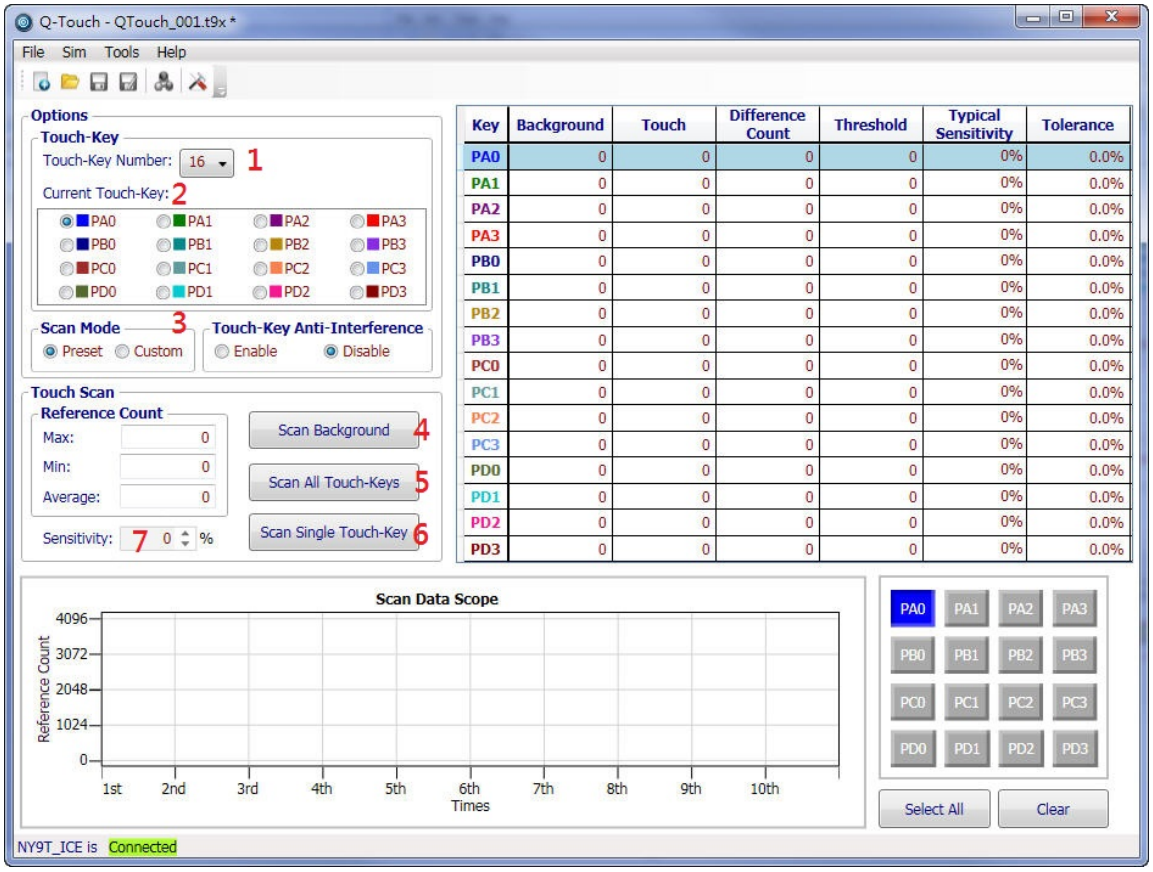
掃描功能區：進行掃描操作與微調。

掃描資料表：顯示掃描後得到的資料。

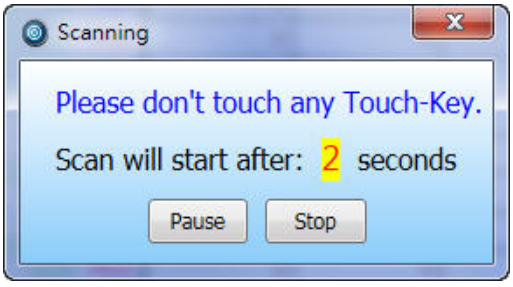
掃描資料示波器：以圖表方式顯示掃描得到的資料。

狀態列：顯示 ICE 連接狀態。

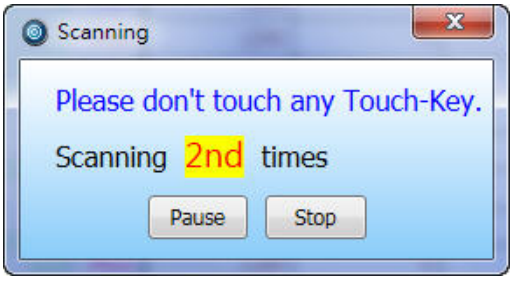
5.3 操作流程



1. 設置所需使用到的 Touch-Key 數量。
2. 設定 Current Touch-Key，預設為 PA0，若無特殊需求不須調整。
3. 設定 Scan Mode，預設為 Preset 模式。設定 Touch-Key Anti-Interference，預設為 Disable。
4. 點擊 [Scan Background] 按鈕，會彈出以下倒數提示畫面。

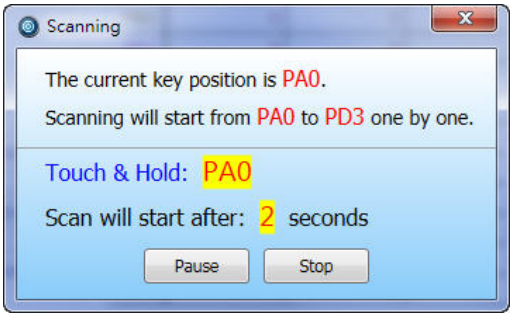


請依照提示不要按壓任何按鍵，倒數結束後，會彈出以下掃描提示畫面。（若想暫停倒數可點擊 [Pause] 按鈕。取消掃描背景動作可點擊 [Stop] 按鈕。）

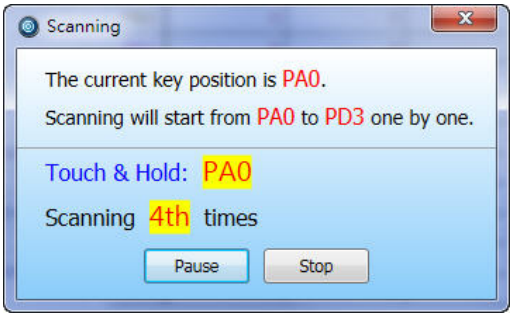


掃描時請依照提示不要按壓任何按鍵，避免影響掃描結果。掃描完成後，會將掃描得到的數值填入掃描資料表中的 Background 欄位。(若想暫停掃描可以點擊 [Pause] 按鈕。取消 Scan Background 動作可點擊 [Stop] 按鈕。)

5. 點擊 [Scan All Touch-Keys] 按鈕，會彈出以下倒數提示畫面。



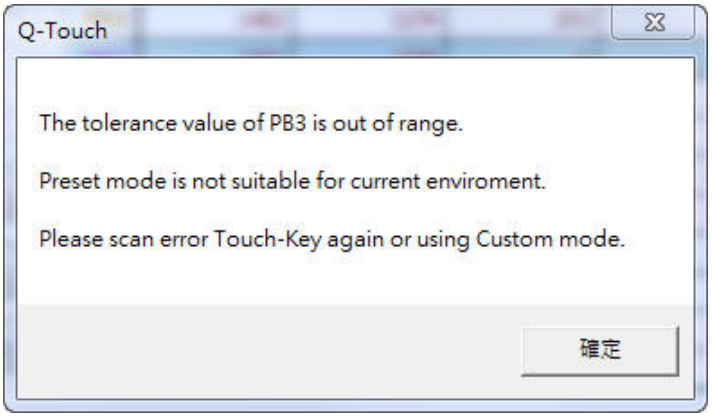
請依照提示按住指定按鍵，倒數結束後，會彈出以下掃描提示畫面。



請依照提示按住不放指定按鍵，直到掃描結束。掃描結束會將結果填至掃描資料表對應按鍵的 Touch 欄位中，並跳出下一個按鍵的倒數提示，如此迴圈直到所有按鍵掃描完畢。

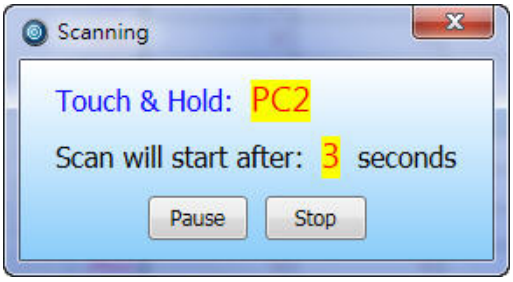
注意：

1. ***掃描過程中可通過 [Pause] 按鍵進行暫停，[Stop] 按鍵結束掃描。若想從特定按鍵開始掃描，僅需將 Current Touch-Key 設置為指定按鍵，再點擊 Scan All Touch-Keys 按鈕即可。***
2. ***全部按鍵掃描完成後，工具會自動計算預設的 Threshold 與 Tolerance 填入掃描資料表中。若有按鍵的 Tolerance 超過 ±30% 的範圍，會彈出以下對話方塊。***

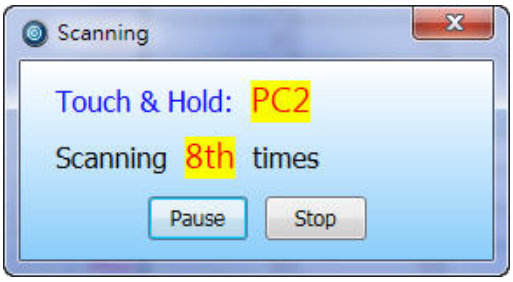


此對話方塊說明目前的環境不適合使用 Preset 模式，請重新掃描 Tolerance 超過 ±30% 的按鍵或是切換至 Custom 模式來設置個別的靈敏度數值。

6. 若某些按鍵的 Tolerance 數值超出限制範圍，可以通過點擊 [Scan Single Touch-Key] 按鈕來重新掃描指定按鍵。(請先於 Current Touch-Key 設定選擇指定按鍵後，再點擊此按鈕。）此動作會彈出以下倒數提示畫面。



請依照提示按住指定按鍵，倒數結束後，會彈出以下掃描提示畫面。



請依照提示按住不放指定按鍵直到掃描結束，掃描結果會自動填入掃描資料表，並依據新資料更新 Threshold 與 Tolerance 數值。

7. 全部按鍵掃描完畢後，可依據需求調整靈敏度，即完成 *Q-Touch* 基本操作流程，模擬測試調節實際應用需要的 sensitivity，然後保存得到.t9x 檔案供 *Q-Code_NY9T* 調用。

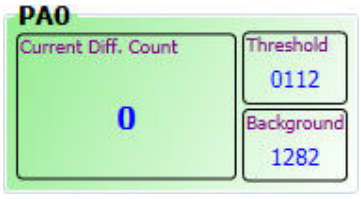
5.4 模擬 (Simulation)

靈敏度設置完成後，可依照下列流程來進行模擬動作，檢查目前設定的靈敏度是否與期望效果接近。

1. 可直接於工具列上點擊 [Simulation] 功能，或是通過功能列中的 [Sim] -> [Simulation] 打開模擬畫面。

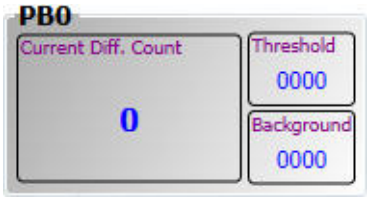
2. 打開後的模擬畫面如下所示：

上方按鍵狀態面板中，每個按鍵會顯示 Current Difference Count、Threshold 與 Background 資料，方便使用者進行觀察。按鍵狀態圖示會有三種狀態，說明如下：



此圖示為初始狀態或按鍵被判定為未被按下狀態。可觀察此圖示中 Current Difference Count 欄位中的數值，必定小於右邊的 Threshold 數值。

此圖示為按鍵被判定為按下狀態。可觀察此圖示中 Current Difference Count 欄位中的數值，必定大於右邊的 Threshold 數值。

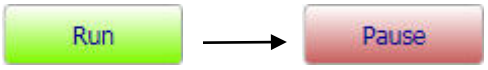


此圖示為按鍵未啟用狀態。在主畫面 Touch-Key Number 設定之外的按鍵，都將以未啟用的狀態顯示，模擬過程中也不會有任何的數值變化。

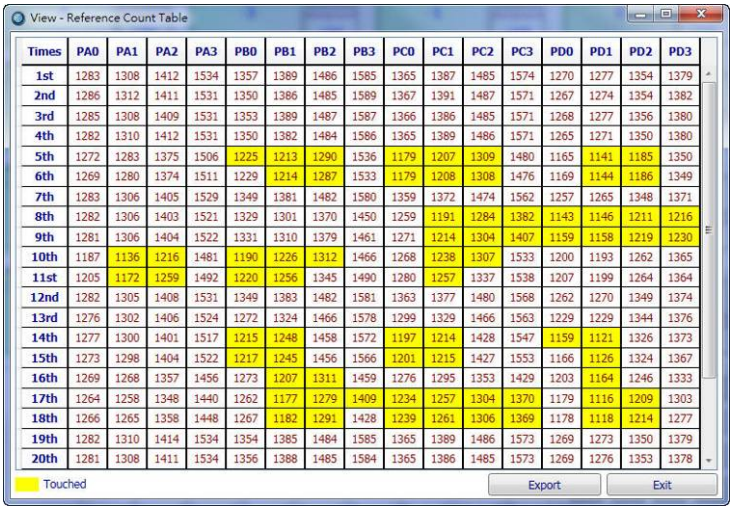
此圖示為靈敏度等級下拉式功能表。可調整欲模擬的靈敏度等級（0~7），預設值為等級 4。

3. 點擊 [Run] 按鈕，啟動 NY9T_ICE 開始進行模擬動作。此時可按壓 Touch-Key Number 範圍中的任意按鍵，觀察按鍵狀態面板是否將對應按鍵判定為按下狀態，來判斷靈敏度是否符合需求。也可操作下方示波器的按鍵切換面板，觀察不同按鍵的數值變化。

4. [Run] 按鈕被點擊開始進行模擬後，會轉變為 [Pause] 按鈕，可使用 [Pause] 按鈕進行暫停模擬動作。此按鈕會在這兩種狀態中進行切換。



5. 模擬過程中可對得到的資料進行相關分析，點擊 [View…] 按鈕打開 Reference Count Table 視窗。此視窗中會包含此次模擬過程中的所有 Reference Count 資料，並將被判定為按下狀態的資料以黃色背景色顯示。



Times	PA0	PA1	PA2	PA3	PB0	PB1	PB2	PB3	PC0	PC1	PC2	PC3	PD0	PD1	PD2	PD3
1st	1283	1308	1412	1534	1357	1389	1486	1585	1365	1387	1485	1574	1270	1277	1354	1379
2nd	1286	1312	1411	1531	1350	1386	1485	1589	1367	1391	1487	1571	1267	1274	1354	1382
3rd	1285	1308	1409	1531	1353	1389	1487	1587	1366	1386	1485	1571	1268	1277	1356	1380
4th	1282	1310	1412	1531	1350	1382	1484	1586	1365	1389	1486	1571	1265	1271	1350	1380
5th	1272	1283	1375	1506	1225	1213	1290	1536	1179	1207	1309	1480	1165	1141	1185	1350
6th	1269	1280	1374	1511	1229	1214	1287	1533	1179	1208	1308	1476	1169	1144	1186	1349
7th	1283	1306	1405	1529	1349	1381	1482	1580	1359	1372	1474	1562	1257	1265	1348	1371
8th	1282	1306	1403	1521	1329	1301	1370	1450	1259	1191	1284	1382	1143	1146	1211	1216
9th	1281	1306	1404	1522	1331	1310	1379	1461	1271	1214	1304	1407	1159	1158	1219	1230
10th	1187	1136	1216	1481	1190	1226	1312	1466	1268	1238	1307	1533	1200	1193	1262	1365
11st	1205	1172	1259	1492	1220	1256	1345	1490	1280	1257	1337	1538	1207	1199	1264	1364
12nd	1282	1305	1408	1531	1349	1383	1482	1581	1363	1377	1480	1568	1262	1270	1349	1374
13rd	1276	1302	1406	1524	1272	1324	1466	1578	1299	1329	1466	1563	1229	1229	1344	1376
14th	1277	1300	1401	1517	1215	1248	1458	1572	1197	1214	1428	1547	1159	1121	1326	1373
15th	1273	1298	1404	1522	1217	1245	1456	1566	1201	1215	1427	1553	1166	1126	1324	1367
16th	1269	1268	1357	1456	1273	1207	1311	1459	1276	1295	1353	1429	1203	1164	1246	1333
17th	1264	1258	1348	1440	1262	1177	1279	1409	1234	1257	1304	1370	1179	1116	1209	1303
18th	1266	1265	1358	1448	1267	1182	1291	1428	1239	1261	1306	1369	1178	1118	1214	1277
19th	1282	1310	1414	1534	1354	1385	1484	1585	1365	1389	1486	1573	1269	1273	1350	1379
20th	1281	1308	1411	1534	1356	1388	1485	1584	1365	1386	1485	1573	1269	1276	1353	1378

若需要將資料匯出分析，可使用 [Export] 按鈕。會將 Reference Count Table 中的數據匯出至指定的 HTML 檔案中。

6. 使用 Preset mode 時，模擬視窗可針對不同的靈敏度等級進行模擬，提供使用者於 *Q-Code_NY9T* 中調整靈敏度等級的參考。可通過靈敏度等級（Sensitivity Level）下拉功能表選擇想要模擬的等級，再重複上述動作進行模擬。

7. 若想重定模擬的資料，可點擊 [Reset] 按鈕。點擊 [Exit] 按鈕則可離開模擬畫面。

注意：

1. ***模擬功能需完成所有資料掃描，並在 NY9T_ICE 有連接的狀態下才能打開。***
2. ***若需要調整靈敏度再進行模擬，不需關掉模擬畫面，可直接於主畫面調整靈敏度，修改會直接反應到模擬畫面中。***
3. ***此模擬功能為軟體模擬，主要用來測試所設置的靈敏度條件。因受限於軟體、USB 與硬體之間的溝通，每次獲取資料時間約為 0.5 秒，故無法模擬反應時間等相關應用。***
4. ***靈敏度等級調整僅供模擬使用，此調整並不影響 Q-Touch 產生的靈敏度表格。若需調整應用上的靈敏度等級，請於 Q-Code_NY9T 中調整。***