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

使用手冊

NY8 系列範例程式

MCU Assembly Programmer

Version 1.0
May 26, 2015

改 版 記 錄

版本	日期	內 容 描 述	修正頁
1.0	2015/5/26	新發佈。	-

目　錄

1 簡介

隨著科技的進步及電子產品不斷更新。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 NY8 系列 8 位元單晶片，並提供優質的服務。為使客戶能夠更方便快捷的使用九齊 NY8 系列 IC，九齊科技針對 NY8 系列 IC 開發一系列的範例程式— NY8 Example Code。透過本篇介紹的 NY8 系列 IC 範例程式，初學者只需要進行簡單的培訓，即可在短時間內瞭解 NY8 系列 IC 的程式撰寫技巧、功能以及應用方式，以致完成 NY8 系列 IC 的產品專案開發。

使用者可以在九齊科技網站來獲得 *NYIDE* 的安裝程式檔。安裝 *NYIDE* 後，開啟 *NYIDE* 程式並在建立新專案時就可選擇所需的範例程式。

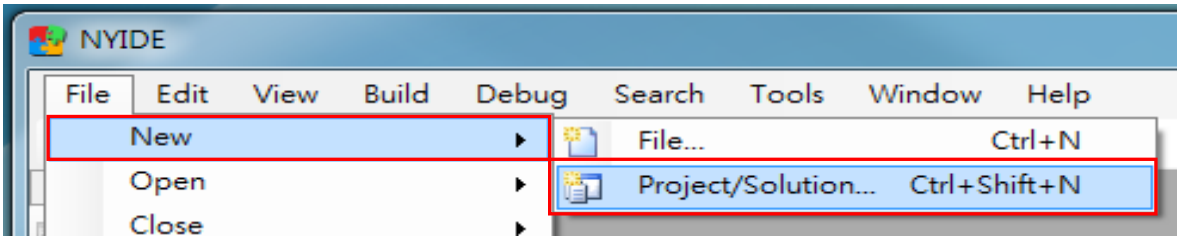
1.1 內容

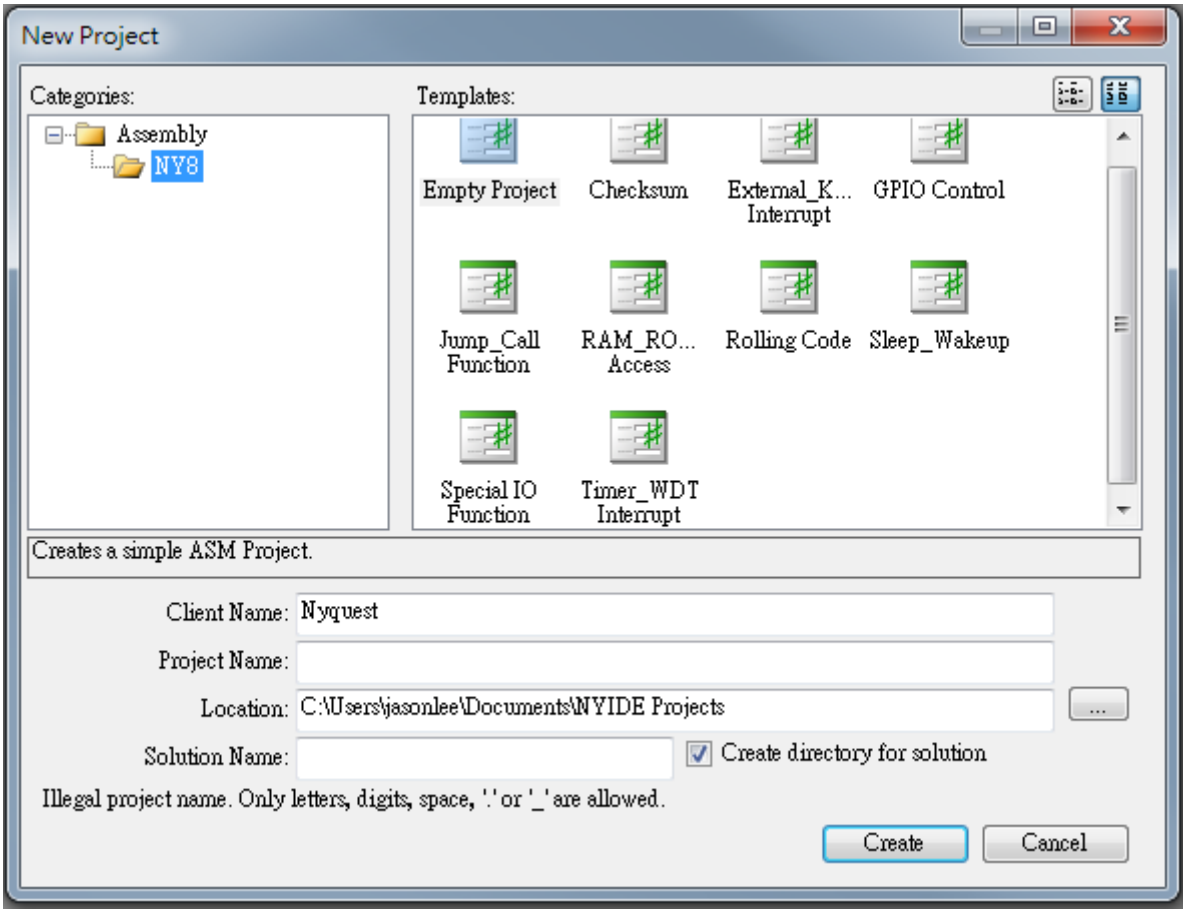
1.2 NY8 範例程式項目

- Empty Project……..(空白專案)
- External_Key Change Interrupt……(外部中斷與 PB input change 中斷)
- Timer_WDT Interrupt……..(時鐘中斷與看門狗時鐘中斷)
- GPIO Control……..(通用輸入/輸出口控制)
- Special_IO Function……(IR carrier、PWM、Buzzer 輸出)
- Jump_Call Function……(分支跳躍指令與副程式呼叫指令)
- RAM_ROM_REG Access……..(RAM、ROM、SFR 存取)
- Sleep_Wakeup……(切換至 Halt / Standby mode 與喚醒)
- Rolling Code……(讀取滾碼)
- Checksum……...(計算 ROM 校驗和)

1.3 安裝軟體工具(*NYIDE*)

請先安裝九齊科的*NYIDE*。然後執行*NYIDE*程式後接著選擇開啟新專案，就可選擇所需的NY8範例程式(NY8 Example Code)。操作步驟如下圖所示：





注意：在安裝NYIDE 時，需要同時安裝 NYASM。

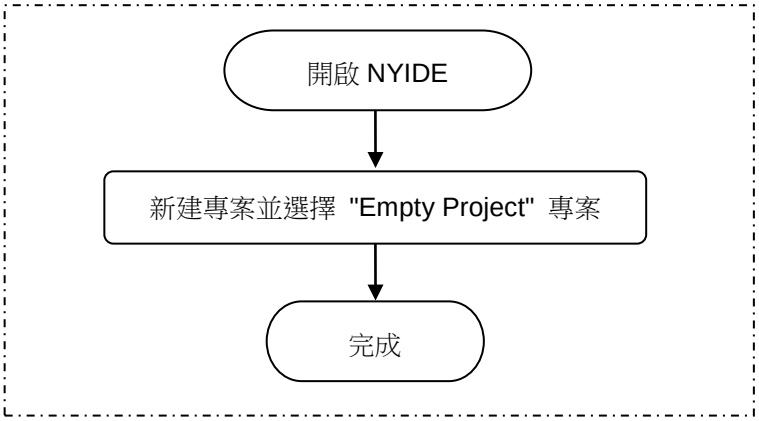
2 NY8 範例程式說明

2.1 Empty Project (空白專案)

2.1.1 功能介紹

功能說明：建立新專案並載入Header file "NY8.H"。	
輸入	功能及輸出說明
NA	建立新專案並載入Header file "NY8.H"。

2.1.2 設計流程圖



2.1.3 程式說明

```
;---------------------------
; 表頭註解
; Project:          ; 專案名稱
; File:             ; 檔案名稱
; Description:      ; 程式敘述
; Author:           ; 程式開發者
; Version:          ; 版本別
; Date:             ; 日期
;---------------------------
; 添加 NY8A051A / 053A Series Header File
#include    NY8.H
;---------------------------
; 定義變數
;---------------------------
; 定義常數
```

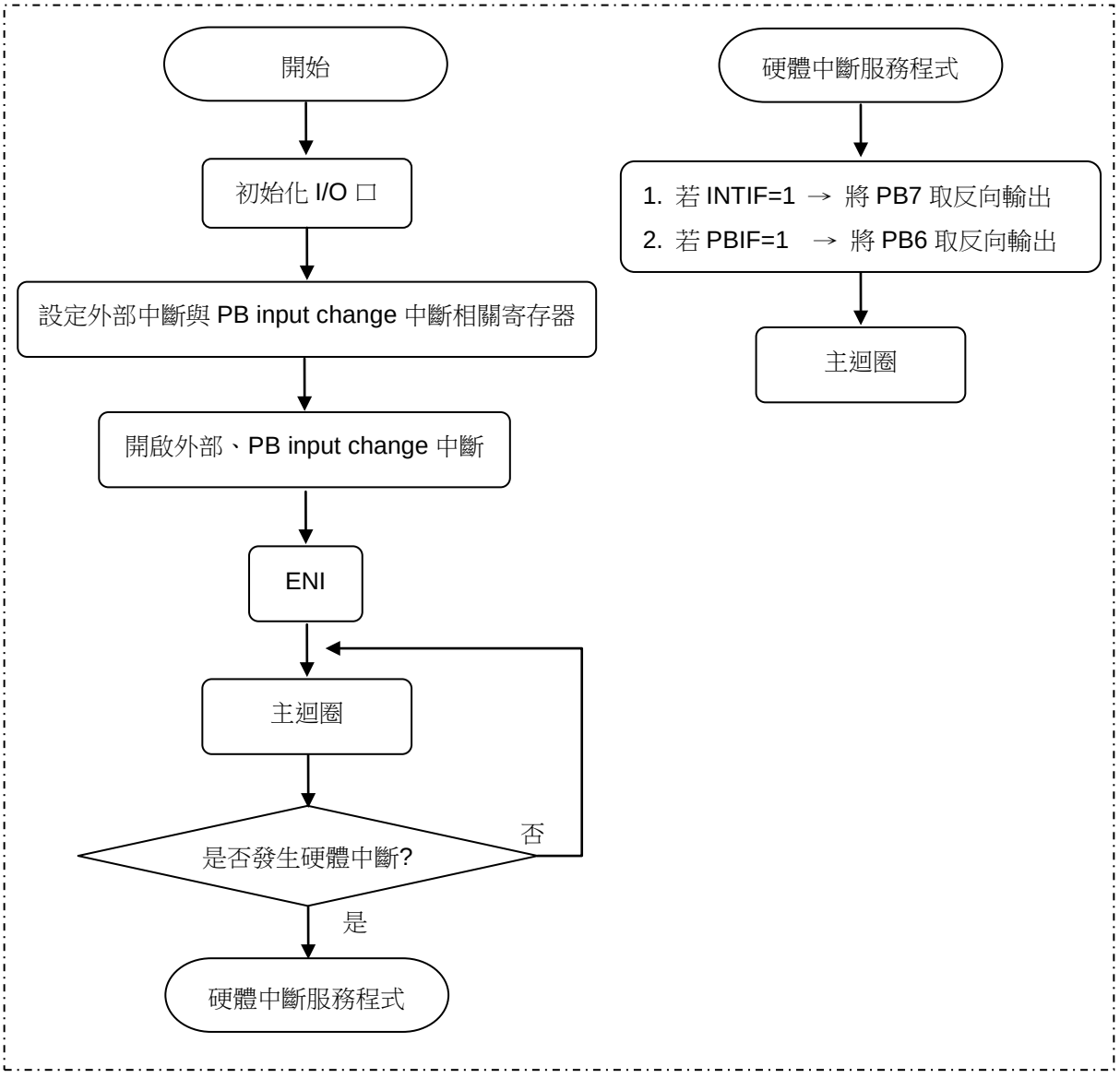
```
;---------------------------
; 定義向量
ORG     0x000                   ; 電源重置向量位址
goto    V_Main
ORG     0x008                   ; 硬體中斷向量位址
goto    V_INT
;---------------------------
; 程式開頭處
ORG     0x010                   ; ROM位址 0x010
V_Main:

L_MainLoop:                     ; 程式主迴圈
    clrwdt                      ; 清除看門狗時鐘
    lgoto   L_MainLoop
;---------------------------
; 硬體中斷服務程式
V_INT:                          ; 硬體中斷服務程式開頭處
    retie                       ; 從硬體中斷服務程式返回
;---------------------------
; 結束程式
end
```

2.2 External_Key Change Interrupt (外部中斷與 PB input change 中斷)

2.2.1 功能介紹

功能說明：1. 使用外部中斷與PB input change中斷。	
輸入	功能及輸出說明
INT	輸入INT信號上升緣時發生中斷 → PB7 取反向輸出
PB1	輸入PB1信號上升/下降緣時發生中斷 → PB6 取反向輸出

2.2.2 設計流程圖



2.2.3 程式說明

```
V_Main:
; 關閉所有中斷
    disi
; 開啟PB1與PB0上拉電阻
    movia   ~(C_PB1_PHB | C_PB0_PHB)
    movar   Pr_PB_PH_Ctrl
; 開啟PB1 input change wakeup
    movia   C_PB1_Wakeup
    movar   Pr_PB_WakeUp_Ctrl
; 設定PB1與PB0為輸入口
    movia   C_PB1_Input | C_PB0_Input
    iost    Pf_PB_Dir_Ctrl
    movia   0x00
    movar   Pr_PB_Data
; 設置外部中斷相關寄存器
    movia   C_EXINT_Edge
    t0md
    movia   C_ExtINT_En
    movar   Pr_PWR_Ctrl
; 開啟外部中斷與PB input change中斷
    movia   C_INT_EXT | C_INT_PBKey
    movar   Pr_INT_Ctrl
    movia   0x00                        ; 清除所有的中斷旗標
    movar   Pr_INT_Flag
    eni
;--------------------------
; 主迴圈
L_MainLoop:
    clrwdt
    goto    L_MainLoop
;---------------------------
; 硬體中斷服務程式開始處
V_INT:
    movar   R_AccBuf                    ; 保存ACC值到變數R_AccBuf
    swapr   R_AccBuf,C_SaveToReg
    movr    Pr_Status,C_SaveToAcc
    movar   R_StatusBuf                 ; 保存STATUS值到變數R_StatusBuf
```

```
;----------------------------
; 外部中斷服務程式
L_EX_INT:
    btrss   Pr_INT_Flag,C_INT_EXT_Bit
    goto    L_PBX_INT
    movia   0x80
    xorar   Pr_PB_Data,C_SaveToReg
    movia   ~C_INT_EXT                  ; 清除外部中斷旗標
    movar   Pr_INT_Flag
    goto    L_RET2Main
;----------------------------
; PB input change中斷服務程式
L_PBX_INT:
    btrss   Pr_INT_Flag,C_INT_PBKey_Bit
    goto    L_RET2Main
    movia   0x40
    xorar   Pr_PB_Data,C_SaveToReg
    movia   ~C_INT_PBKey                ; 清除PB input change中斷旗標
    movar   Pr_INT_Flag

L_RET2Main:
    movr    R_StatusBuf,C_SaveToAcc
    movar   Pr_Status                   ; 從R_StatusBuf回存STATUS值
    swapr   R_AccBuf,C_SaveToAcc        ; 從R_AccBuf回存ACC值
    retie                               ; 從硬體中斷服務程式返回
```

Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.3 Timer_WDT Interrupt (時鐘中斷與看門狗時鐘中斷)

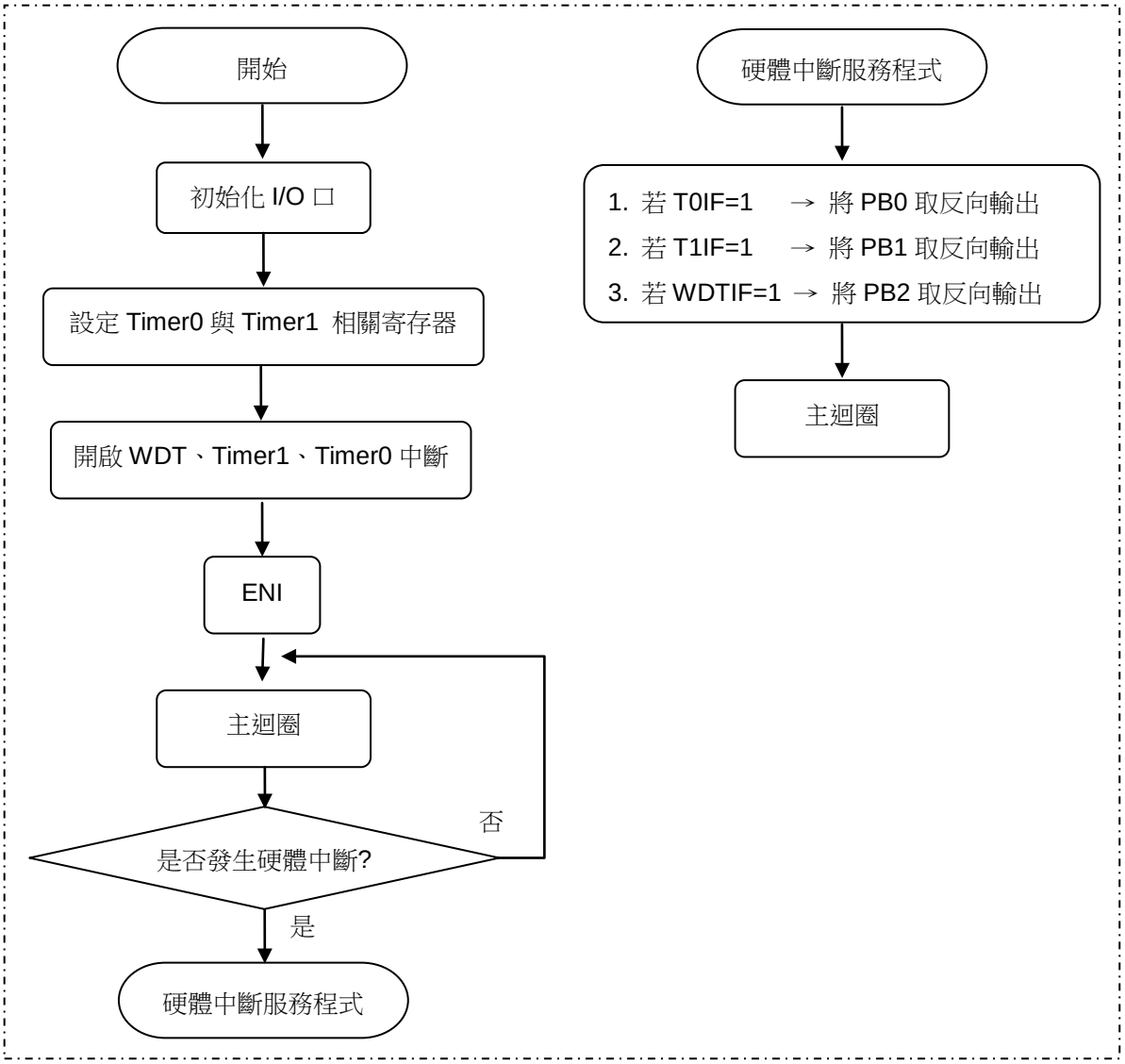
2.3.1 功能介紹

功能說明：

1. F_{INST}= 4MHz/4T (I_HRC) = 1MHz
2. 請將*NYIDE* 的 Project Options中 WDT time base設為 3.5ms
3. 設置Timer0中斷每 2048個指令週期發生一次，並將PB0取反向輸出
4. 設置Timer1中斷每 1024個指令週期發生一次，並將PB1取反向輸出
5. 設置WDT中斷每 3.5ms發生一次，並將PB2取反向輸出

輸入	功能及輸出說明
NA	1. Timer0中斷每 2048個指令週期發生一次，並將PB0取反向輸出
NA	2. Timer1中斷每 1024個指令週期發生一次，並將PB1取反向輸出
NA	3. WDT中斷每 3.5ms發生一次，並將PB2取反向輸出

2.3.2 設計流程圖



2.3.3 程式說明

```
; 設置 Timer0 相關寄存器
    movia       C_TMR0_Dis
    iost        Pf_PWR_Ctrl1                    ; 關閉 Timer0
    movia       0x00
    movar       Pr_TMR0_Data
    movia       C_PS0_TMR0 | C_PS0_Div8
    t0md
;-------------------------------------------------------------------------
   ;若 WDT 需要 Prescaler0 分頻 ( Prescaler0 分頻只能擇一讓 Timer0 或 WDT 使用)
    ;movia      (0x00 | C_PS0WDT_Sel)
    ;t0md
;-------------------------------------------------------------------------
; 設置 Timer0 相關寄存器
    movia       0xFF
    sfun        Ps_TMR1_Data
    movia       C_TMR1_Reload | C_TMR1_En
    sfun        Ps_TMR1_Ctrl1
    movia       C_TMR1_ClkSrc_Inst | C_PS1_Div4
    sfun        Ps_TMR1_Ctrl2
; 開啟 WDT 中斷、Timer1 中斷、Timer0 中斷
    movia       C_INT_WDT | C_INT_TMR1 | C_INT_TMR0
    movar       Pr_INT_Ctrl
; 開啟 WDT
    movia       C_WDT_En | C_LVR_En
    movar       Pr_PWR_Ctrl
; 開啟 Timer0
    movia       C_TMR0_En
    iost        Pf_PWR_Ctrl1
    eni                                         ; 開啟 all unmasked 中斷
;-------------------------------------------------------------
; 主迴圈
L_MainLoop:
    nop
    goto        L_MainLoop


;--------------------------
; 硬體中斷服務程式開始處
```

```
V_INT:
    movar   R_AccBuf                ; 保存ACC值到變數R_AccBuf
    swapr   R_AccBuf,C_SaveToReg
    movr    Pr_Status,C_SaveToAcc
    movar   R_StatusBuf             ; 保存 STATUS 值到變數 R_StatusBuf
;---------------------------
; Timer1中斷服務程式
L_TIME1_INT:
    btrss   Pr_INT_Flag,C_INT_TMR1_Bit
    goto    L_TIME0_INT
    btrss   R_shift_regl,1
    goto    L_TURNOFF_PORTB1
    bcr     R_shift_regl,1
    movr    R_shift_regl,C_SaveToAcc
    movar   Pr_PB_Data
    goto    L_clr_flag_Timer1
L_TURNOFF_PORTB1:
    bsr     R_shift_regl,1
    movr    R_shift_regl,C_SaveToAcc
    movar   Pr_PB_Data
L_clr_flag_Timer1:
    movia   ~C_INT_TMR1             ; 清除 Timer1 中斷旗標
    movar   Pr_INT_Flag
;---------------------------
; Timer0中斷服務程式
L_TIME0_INT:
    btrss   Pr_INT_Flag,C_INT_TMR0_Bit
    goto    L_WDT_INT
    movia   0x00
    movar   Pr_TMR0_Data            ; 重新載入 0x00 到 TMR0
    btrss   R_shift_regl,0
    goto    L_TURNOFF_PORTB0
    bcr     R_shift_regl,0
    movr    R_shift_regl,C_SaveToAcc
    movar   Pr_PB_Data
    goto    L_clr_flag_Timer0
L_TURNOFF_PORTB0:
    bsr     R_shift_regl,0
```

```
        movr    R_shift_regl,C_SaveToAcc
        movar   Pr_PB_Data

L_clr_flag_Timer0:
        movia   ~C_INT_TMR0                 ; 清除 Timer0 中斷旗標
        movar   Pr_INT_Flag
;----------------------------
; WDT  中斷服務程式
L_WDT_INT:
        btrss   Pr_INT_Flag,C_INT_WDT_Bit
        goto    L_RET2Main
        btrss   R_shift_regl,2
        goto    L_TURNOFF_PORTB2
        bcr     R_shift_regl,2
        movr    R_shift_regl,C_SaveToAcc
        movar   Pr_PB_Data
        goto    L_clr_flag_WDT
L_TURNOFF_PORTB2:
        bsr     R_shift_regl,2
        movr    R_shift_regl,C_SaveToAcc
        movar   Pr_PB_Data
        goto    L_clr_flag_WDT
L_clr_flag_WDT:
        movia   ~C_INT_WDT                  ; 清除 WDT 中斷旗標
        movar   Pr_INT_Flag
L_RET2Main:
        movr    R_StatusBuf,C_SaveToAcc
        movar   Pr_Status                   ; 從R_StatusBuf回存STATUS值
        swapr   R_AccBuf,C_SaveToAcc        ; 從R_AccBuf回存ACC值
        retie                               ; 從硬體中斷服務程式返回
```

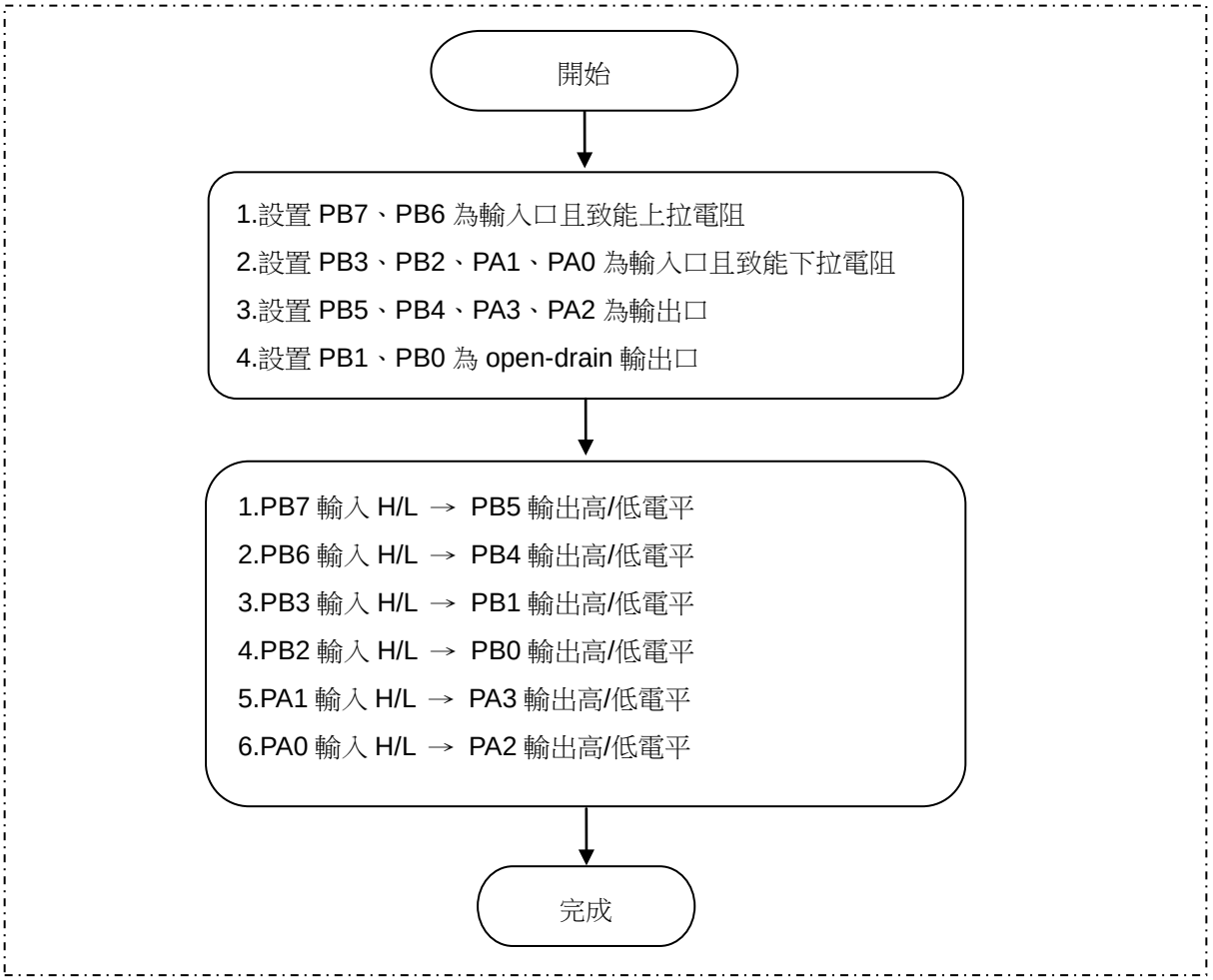
Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.4 GPIO Control (通用輸入/輸出口控制)

2.4.1 功能介紹

功能說明：1. 設置 GPIO 相關寄存器。	
輸入	功能及輸出說明
PB7	PB7 輸入 H/L → PB5 輸出 高/低電平
PB6	PB6 輸入 H/L → PB4 輸出 高/低電平
PB3	PB3 輸入 H/L → PB1 輸出 高/低電平
PB2	PB2 輸入 H/L → PB0 輸出 高/低電平
PA1	PA1 輸入 H/L → PA3 輸出 高/低電平
PA0	PA0 輸入 H/L → PA2 輸出 高/低電平

2.4.2 設計流程圖



2.4.3 程式說明

```
; 開啟 PB1、PB0為 open-drain 輸出
    movia     C_PB1_OD | C_PB0_OD
    iost      Pf_PB_OD_Ctrl
; 開啟 PB3、PB2、PA1、PA0 下拉電阻
    movia     ~( C_PB3_PLB |  C_PB2_PLB |  C_PA1_PLB |  C_PA0_PLB)
    movar     Pr_PAB_PL_Ctrl
; 開啟 PB7、PB6 上拉電阻
    movia     ~( C_PB7_PHB | C_PB6_PHB)
    movar     Pr_PB_PH_Ctrl
; 設置 PB7、PB6、PB3、PB2為輸入口
; 設置 PB5、PB4、PB1、PB0為輸出口
    movia     C_PB7_Input | C_PB6_Input | C_PB3_Input | C_PB2_Input
    iost      Pf_PB_Dir_Ctrl
    movia     0x03
    movar     Pr_PB_Data
; 設置 PA1、PA0 為輸入口
; 設置 PA3、PA2 為輸出口
    movia     C_PA1_Input | C_PA0_Input
    iost      Pf_PA_Dir_Ctrl
    movia     0x0C
    movar     Pr_PA_Data
```

Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.5 Special IO Function (IR carrier、PWM、Buzzer 輸出)

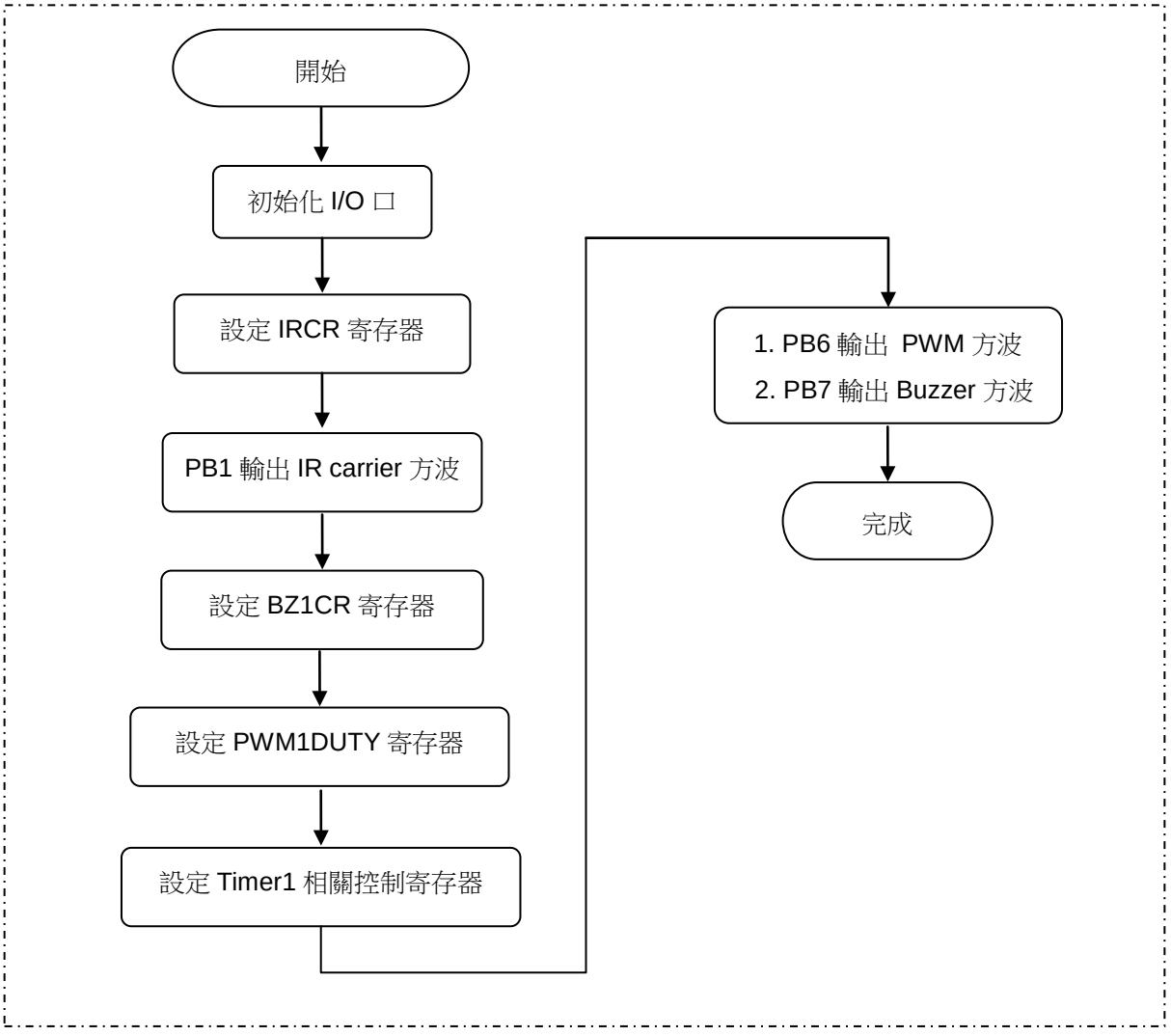
2.5.1 功能介紹

功能說明：

1. 本範例程式使用外接晶振 3.58MHz (PB4 / PB5)
2. 由PB1腳位輸出約38KHz的 IR carrier方波
3. 由PB6腳位輸出頻率約為1.75KHz ， duty cycle為25%的PWM方波
4. 由PB7腳位輸出頻率約為55.94KHz的Buzzer方波

輸入	功能及輸出說明
NA	1. 由PB1腳位輸出約38KHz的 IR carrier方波
NA	2. 由PB6腳位輸出頻率約為1.75KHz ， duty cycle為25%的PWM方波
NA	3. 由PB7腳位輸出頻率約為55.94KHz的Buzzer方波

2.5.2 設計流程圖



2.5.3 程式說明

```
; 設定 IR carrier 寄存器
    bsr     Pr_PB_Data,1                    ; 設定 PB1 data buffer = 1
    movia   C_IR_ClkSrc_358M | C_IR_En
    sfun    Ps_IR_Ctrl
; 設定 Buzzer1 寄存器
    movia   C_BZ1_En | C_BZ1_TMR1B2
    sfun    Ps_BZ1_Ctrl
INIT_TIME1:
    movia   0xFF
    sfun    Ps_TMR1_Data
; 設定 PWM1DUTY 寄存器
    movia   C_PWM_DUTY_25
    sfun    Ps_PWM1_Duty                    ; PWM1 Duty = 64/256 = 25%
; 設定 Timer1 相關控制寄存器
    movia   C_PWM1_En | C_TMR1_Reload | C_TMR1_En
    sfun    Ps_TMR1_Ctrl1
    movia   0x00
    sfun    Ps_TMR1_Ctrl2
```

Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.6 Jump_Call Function (分支跳躍指令與副程式呼叫指令)

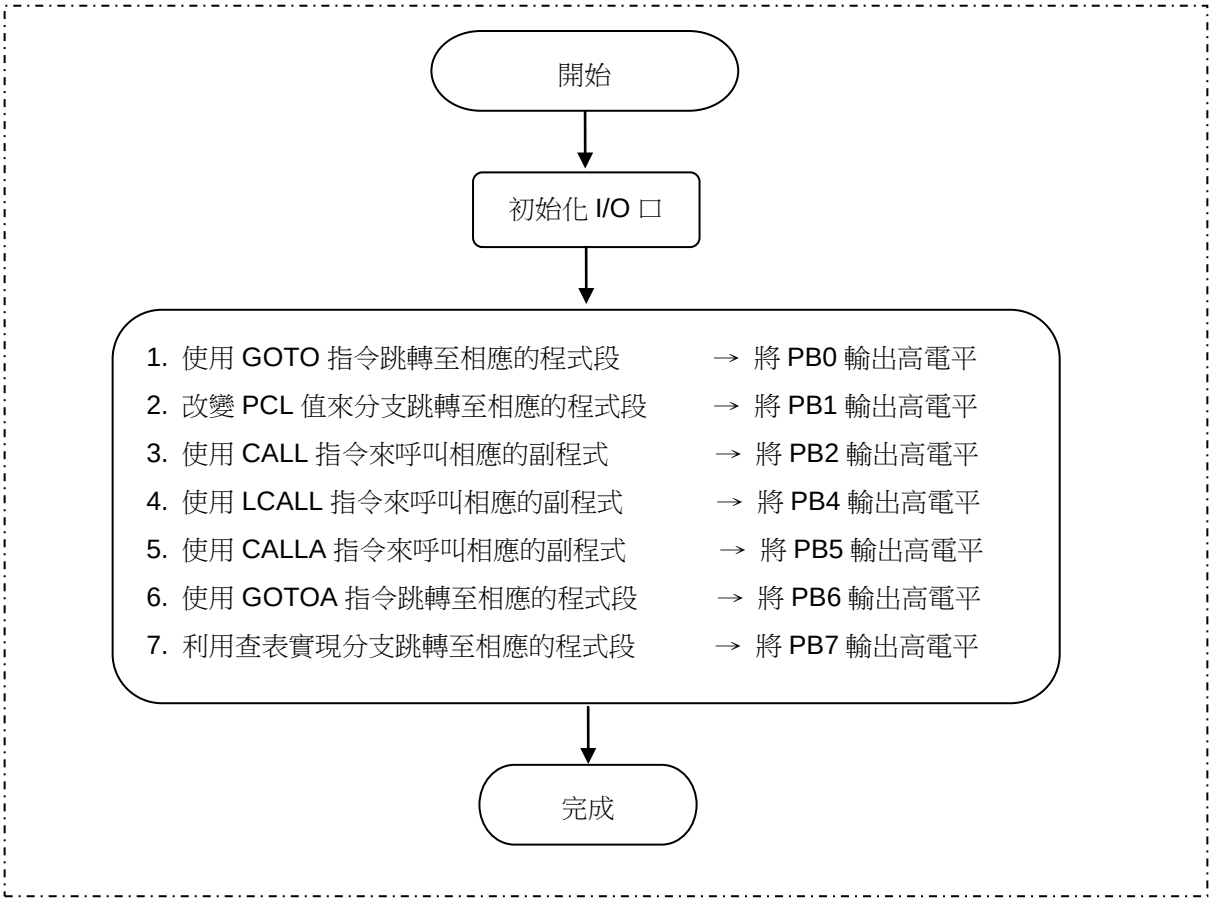
2.6.1 功能介紹

功能說明：

1. 使用分支跳轉指令(GOTO、GOTOA)。
2. 使用呼叫副程式指令(CALL、LCALL、CALLA)。
3. 改變PCL值來實現分支跳轉
4. 利用查表實現分支跳轉

輸入	功能及輸出說明
NA	使用GOTO指令跳轉至相應的程式段 → 將PB0輸出高電平
NA	改變PCL值來分支跳轉至相應的程式段 → 將PB1輸出高電平
NA	使用CALL指令來呼叫相應的副程式 → 將PB2輸出高電平
NA	使用LCALL指令來呼叫相應的副程式 → 將PB4輸出高電平
NA	使用CALLA指令來呼叫相應的副程式 → 將PB5輸出高電平
NA	使用GOTOA指令跳轉至相應的程式段 → 將PB6輸出高電平
NA	利用查表實現分支跳轉至相應的程式段 → 將PB7輸出高電平

2.6.2 設計流程圖



2.6.3 程式說明

```
;---------------------------
; 使用GOTO指令跳轉至相應的程式段
    movia       Mid L_Goto_Label
    movar       Pr_PCHigh_Data
    goto        L_Goto_Label
    lgoto       L_FailLoop

    ORG         0x020
L_Goto_Label:
    bsr         Pr_PB_Data,0
;---------------------------
; 改變PCL值來實現分支跳轉至相應的程式段
    movia       0x00
    movar       Pr_PCHigh_Data
    movia       0x40
    movar       Pr_PCLow_Data
    lgoto       L_FailLoop

    ORG         0x040
    bsr         Pr_PB_Data,1
;---------------------------
; 使用CALL指令來呼叫相應的副程式
    call        F_sub1
;---------------------------
; 使用LCALL指令來呼叫相應的副程式
    lcall       F_sub2
;---------------------------
; 使用CALLA指令來呼叫相應的副程式
    movia       0x02
    sfun        Ps_TbHigh_Addr
    movia       0x20
    calla
```

```
;----------------------------
; 使用GOTOA指令跳轉至相應的程式段
    movia   0x03
    sfun    Ps_TbHigh_Addr
    movia   0x10
    gotoa                           ; 分支跳轉至ROM位址"0x310"
;----------------------------
利用查表實現分支跳轉
    movia   0x03
    sfun    Ps_TbHigh_Addr
    movia   0x50
    tablea
    movar   R_TableLowByte
    sfunr   Ps_TbHigh_Data
    sfun    Ps_TbHigh_Addr
    movr    R_TableLowByte,C_SaveToAcc
    gotoa                           ; 分支跳轉至ROM位址"0x330"
;----------------------------
; 定義表格
    ORG     0x350                   ; ROM位址"0x350"
T_DataTbl:
    DW      0x0330                  ; 表格內容
```

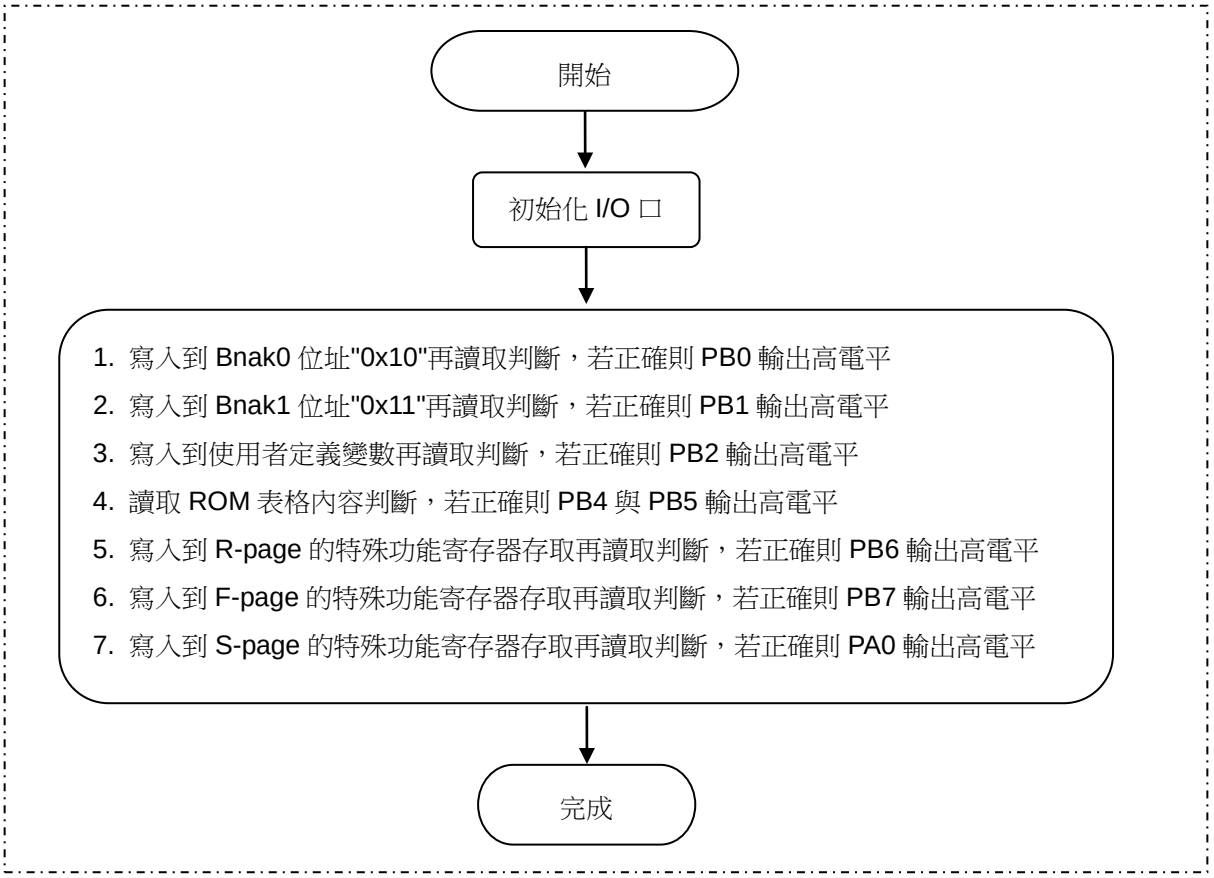
Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.7 RAM_ROM_REG Access (RAM、ROM、SFR 存取)

2.7.1 功能介紹

功能說明： 1. 一般用途寄存器存取 (General Purpose Register) 2. 讀取ROM資料 (Program Memory) 3. 特殊功能寄存器存取 (Special-Function Register)	
輸入	功能及輸出說明
NA	寫入到Bnak0位址"0x10"再讀取判斷，若正確則PB0輸出高電平
NA	寫入到Bnak1位址"0x11"再讀取判斷，若正確則PB1輸出高電平
NA	寫入到使用者定義變數再讀取判斷，若正確則PB2輸出高電平
NA	讀取ROM表格內容判斷，若正確則PB4與PB5輸出高電平
NA	寫入到R-page的特殊功能寄存器存取再讀取判斷，若正確則PB6輸出高電平
NA	寫入到F-page的特殊功能寄存器存取再讀取判斷，若正確則PB7輸出高電平
NA	寫入到S-page的特殊功能寄存器存取再讀取判斷，若正確則PA0輸出高電平

2.7.2 設計流程圖



2.7.3 程式說明

```
; 寫入 "0x55" 到 RAM Bank0 位址 "0x10"
    movia     C_SFR_Bank0 | 0x10
    movar     Pr_File_Sel
    movia     0x55
    movar     Pr_Indir_Addr
    clra
    clrr      Pr_File_Sel
; 讀取 RAM Bank0 位址 "0x10" 內容
    movia     C_SFR_Bank0 | 0x10
    movar     Pr_File_Sel
    movr      Pr_Indir_Addr,C_SaveToAcc
; 寫入 "0xAA" 到 RAM Bank1 位址 "0x11"
    movia     C_SFR_Bank1 | 0x11
    movar     Pr_File_Sel
    movia     0xAA
    movar     Pr_Indir_Addr
    clra
    clrr      Pr_File_Sel
; 讀取 RAM Bank1 位址 "0x11" 內容
    movia     C_SFR_Bank1 | 0x11
    movar     Pr_File_Sel
    movr      Pr_Indir_Addr,C_SaveToAcc
; 寫入 "0x5A" 到使用者定義變數 "R_RAM_0x20"
    movia     0x5A
    movar     R_RAM_0x20
    clra
; 讀取使用者定義變數 "R_RAM_0x20" 內容
    movr      R_RAM_0x20,C_SaveToAcc
```

```
; 讀取 ROM 位址 "0x150" 內容
    movia       0x01
    sfun        Ps_TbHigh_Addr
    movia       0x50
    tablea
; 寫入 "0xAA" 到 R-page 特殊功能寄存器 "TMR0"
    movia       0xCC
    movar       Pr_TMR0_Data
    clra
; 讀取 R-page 特殊功能寄存器 "TMR0" 內容
    movr        Pr_TMR0_Data,C_SaveToAcc
; 寫入 "0x08" 到 F-page 特殊功能寄存器 "IOSTB"
    movia       0x08
    iost        Pf_PB_Dir_Ctrl
    clra
; 讀取 F-page 特殊功能寄存器 "IOSTB" 內容
    iostr       Pf_PB_Dir_Ctrl
; 寫入 "0xBB" 到 S-page 特殊功能寄存器 "TMR1"
    movia       0xBB
    sfun        Ps_TMR1_Data
    clra
; 讀取 S-page 特殊功能寄存器 "TMR1" 內容
    sfunr       Ps_TMR1_Data
```

Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.8 Sleep_Wakeup (切換至 Halt / Standby mode 與喚醒)

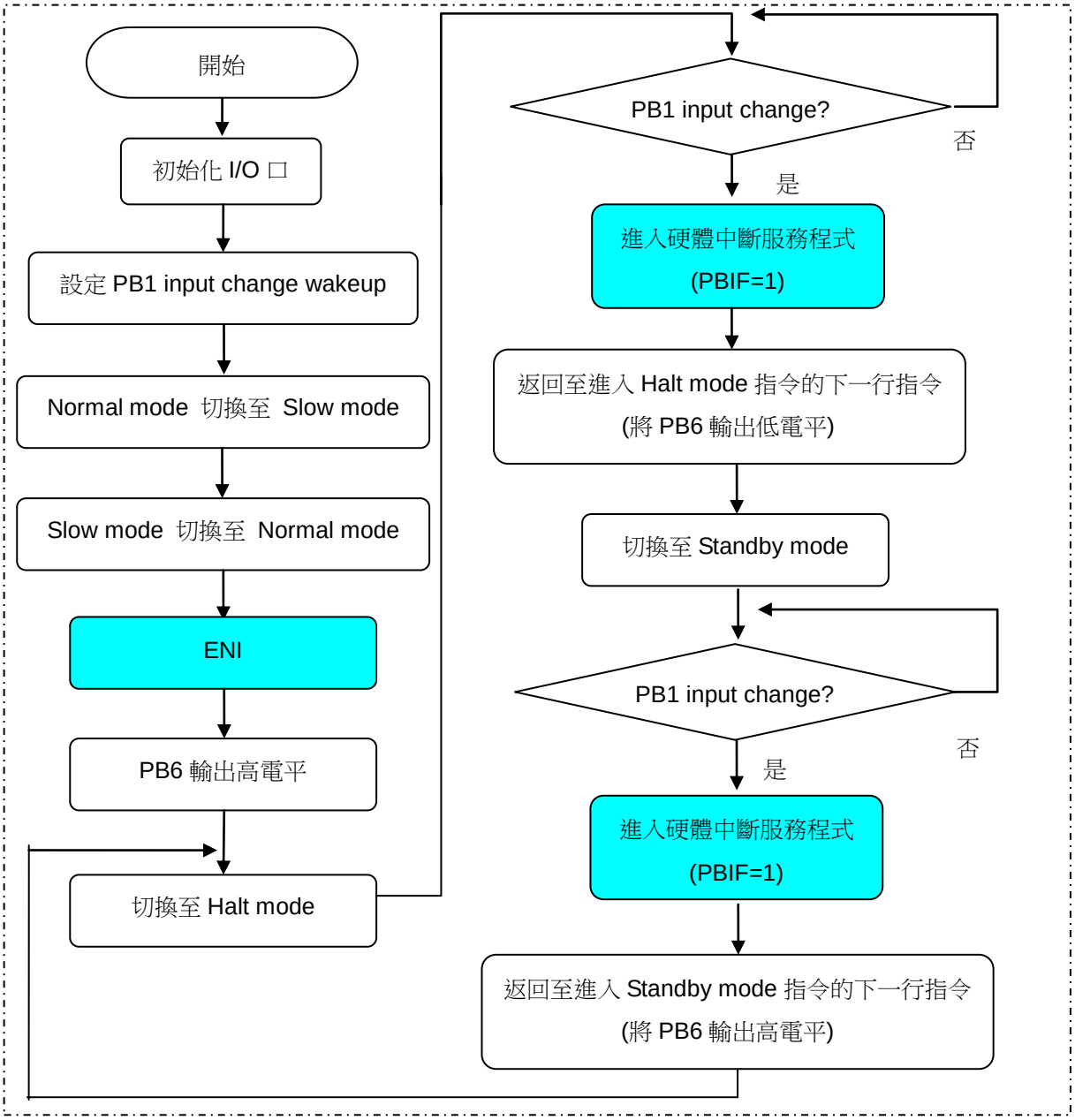
2.8.1 功能介紹

功能說明：

1. Normal mode 切換至 Slow mode 再切換回 Normal mode
2. 切換至 Halt mode 並以 PB1 input change 喚醒
3. 切換至 Standby mode 並以 PB1 input change 喚醒

輸入	功能及輸出說明
PB1	2. 將PB6輸出高電平再切換至Halt mode，從Halt mode喚醒後將PB6輸出低電平
PB1	3. 接著切換至Standby mode，從Standby mode喚醒後將PB6輸出高電平

2.8.2 設計流程圖



2.8.3 程式說明

```
; 設置 PB1 input change wakeup
    movia       C_PB1_Wakeup
    movar       Pr_PB_WakeUp_Ctrl
; 開啟 PortB input change wakeup 中斷
    movia       C_INT_PBKey
    movar       Pr_INT_Ctrl
    movia       0x00                        ; 清除所有的中斷旗標
    movar       Pr_INT_Flag
; Normal mode 切換至 Slow mode
    movia       C_FLOSC_Sel
    sfun        Ps_SYS_Ctrl
; Slow mode 切換至 Normal mode
    movia       C_FHOSC_Sel
    sfun        Ps_SYS_Ctrl
;-------------------------------------------------------------------------------------------------
; 選擇 “eni” 或 “disi” 指令,來決定從 Halt mode 或 Standby mode 喚醒後是否進入硬體中斷服務程式
    eni         ; 從 Halt mode 或 Standby mode 喚醒後進入硬體中斷服務程式

    ;disi       ; 從 Halt mode 或 Standby mode 喚醒後不進入硬體中斷服務程式
    ;-----------------------------------------------------------------------------------
    movr        Pr_PB_Data,C_SaveToReg
    ; 下列有兩種指令方式進入 Halt mode,請擇一使用
    ; 指令一
    sleep
    ; 指令二
    ;movia      C_Halt_Mode | C_FHOSC_Sel   ; 從 Normal mode 進入 Halt mode
    ;sfun       Pr_SYS_Ctrl
    ;-----------------------------------------------------------------------------------
    nop                                     ; for sleep wakeup interrupt latency time
    bcr         Pr_PB_Data,6                ; 從 Halt mode 喚醒後將 PB6 輸出低電平
    movia       ~C_INT_PBKey                ; 清除 PB input change 中斷旗標
    movar       Pr_INT_Flag
; -------------------------------------------------------------------------------------------------
    ; 從 Normal mode 進入 Standby mode
    movr        Pr_PB_Data,C_SaveToReg
    movia       C_Standby_Mode | C_FHOSC_Sel
```

```
    sfun    Ps_SYS_Ctrl
    bsr     Pr_PB_Data,6                ; 從 Standby mode 喚醒後將 PB6 輸出高電平
    movia   ~C_INT_PBKey                ; 清除 PB input change 中斷旗標
    movar   Pr_INT_Flag
;--------------------------
; 硬體中斷服務程式
V_INT:
    movar   R_AccBuf                    ; 保存ACC值到變數R_AccBuf
    swapr   R_AccBuf,C_SaveToReg
    movr    Pr_Status,C_SaveToAcc
    movar   R_StatusBuf                 ; 保存 STATUS 值到變數 R_StatusBuf
; PBX Interrupt
L_PBX_INT:
    btrss   INTF,C_INT_PBKey
    goto    L_RET2Main
L_clr_flag_PBX:
    movia   ~C_INT_PBKey                ; 清除 PB input change 中斷旗標
    movar   Pr_INT_Flag
L_RET2Main:
    movr    R_StatusBuf,C_SaveToAcc
    movar   Pr_Status                   ; 從R_StatusBuf回存STATUS值
    swapr   R_AccBuf,C_SaveToAcc        ; 從R_AccBuf回存ACC值
    retie                               ; 從硬體中斷服務程式返回
```

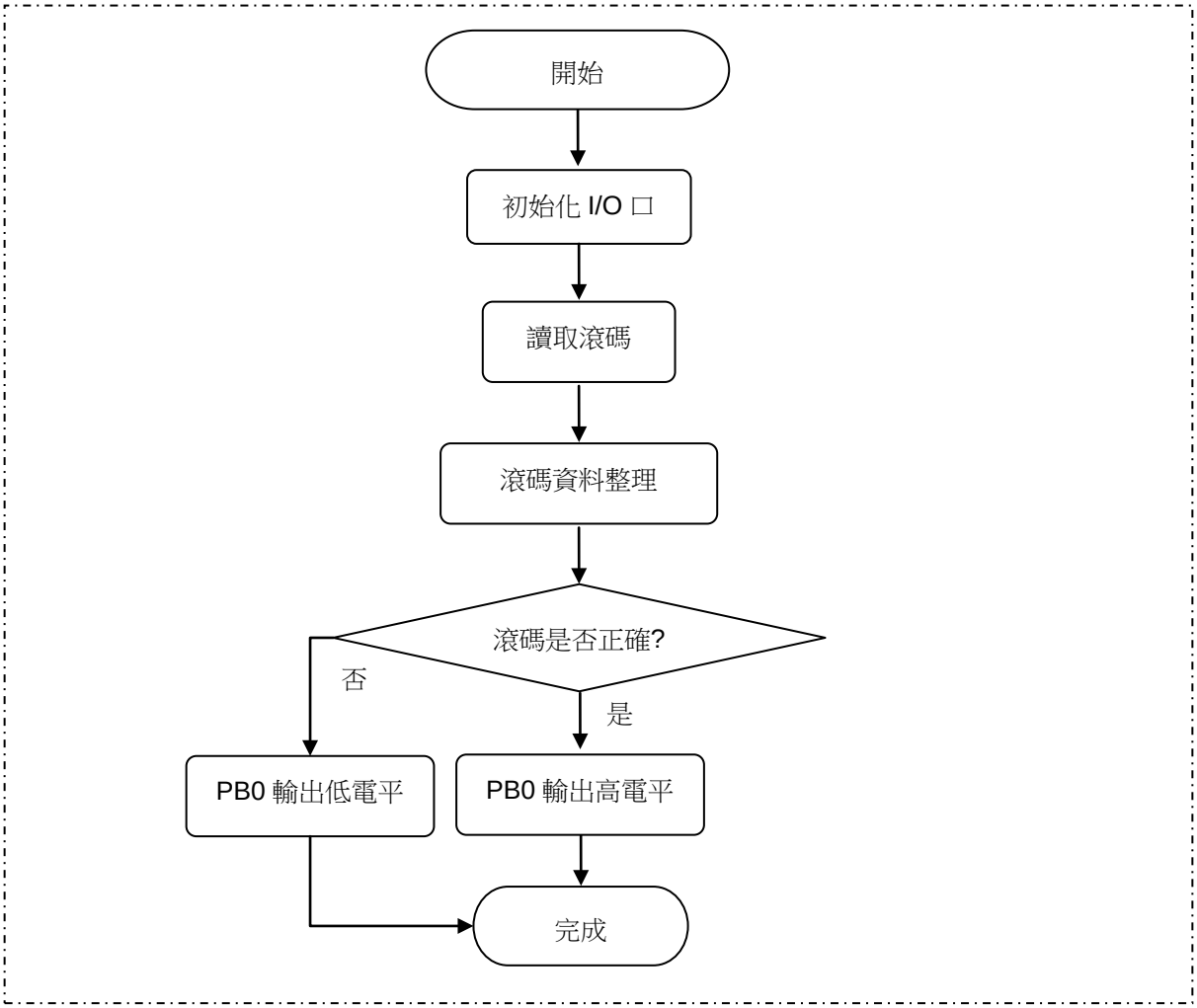
Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.9 Rolling Code (讀取滾碼)

2.9.1 功能介紹

功能說明：1. 讀取滾碼	
輸入	功能及輸出說明
NA	讀取滾碼並判斷，若正確則PB0輸出高電平

2.9.2 設計流程圖



2.9.3 程式說明

```
;---------------------------
; 定義變數
 R_RollingCode_byte0     EQU     10H         ; 存放滾碼計算值 位元 7 ~ 位元 0
 R_RollingCode_byte1     EQU     11H
 R_RollingCode_byte2     EQU     12H         ; 存放滾碼計算值 位元 15 ~ 位元 8
 R_RollingCode_byte3     EQU     13H         ; 存放滾碼計算值 位元 19 ~ 位元 16
```

```
;---------------------------
; 定義常數
; 假設使用者從 Q-Writer 燒入的滾碼為 961109(十進位) = 0xEAA55
 C_RC_B0                EQU     55H             ; 滾碼預設值 位元 7 ~ 位元 0
 C_RC_B1                EQU     AAH             ; 滾碼預設值 位元 15 ~ 位元 8
 C_RC_B2                EQU     0EH             ; 滾碼預設值 位元 19 ~ 位元 16


; 讀取程式記憶體(ROM)位址 0x0E 和 0x0F
    movia       0x00
    sfun        Ps_TbHigh_Addr
    movia       0x0E
    tablea
    movar       R_RollingCode_byte0
    sfunr       Ps_TbHigh_Data
    andia       0x03
    movar       R_RollingCode_byte1
    movia       0x00
    sfun        Ps_TbHigh_Addr
    movia       0x0F
    tablea
    movar       R_RollingCode_byte2
    sfunr       Ps_TbHigh_Data
    andia       0x03
    movar       R_RollingCode_byte3
; 滾碼資料整理
    bcr         Pr_Status,C_Status_C_Bit
    rlr         R_RollingCode_byte3,C_SaveToReg
    rlr         R_RollingCode_byte3,C_SaveToReg
    btrsc       R_RollingCode_byte2,7
    bsr         R_RollingCode_byte3,1
    btrsc       R_RollingCode_byte2,6
    bsr         R_RollingCode_byte3,0

    bcr         Pr_Status,C_Status_C_Bit
    rlr         R_RollingCode_byte2,C_SaveToReg
    bcr         Pr_Status,C_Status_C_Bit
    rlr         R_RollingCode_byte2,C_SaveToReg
    btrsc       R_RollingCode_byte1,1
```

```
        bsr     R_RollingCode_byte2,1
        btrss   R_RollingCode_byte1,1
        bcr     R_RollingCode_byte2,1

        btrsc   R_RollingCode_byte1,0
        bsr     R_RollingCode_byte2,0
        btrss   R_RollingCode_byte1,0
        bcr     R_RollingCode_byte2,0
;判斷滾碼是否正確
        movia   C_RC_B0
        bcr     Pr_Status,C_Status_Z_Bit
        xorar   R_RollingCode_byte0,C_SaveToAcc
        btrss   Pr_Status,C_Status_Z_Bit
        lgoto   L_MainLoop

        movia   C_RC_B1
        bcr     Pr_Status,C_Status_Z_Bit
        xorar   R_RollingCode_byte2,C_SaveToAcc
        btrss   Pr_Status,C_Status_Z_Bit
        lgoto   L_MainLoop

        movia   C_RC_B2
        bcr     Pr_Status,C_Status_Z_Bit
        xorar   R_RollingCode_byte3,C_SaveToAcc
        btrss   Pr_Status,C_Status_Z_Bit
        lgoto   L_MainLoop

        movia   C_PB0_Data
        movar   Pr_PB_Data                      ;滾碼正確,將 PB0 輸出高電平
```

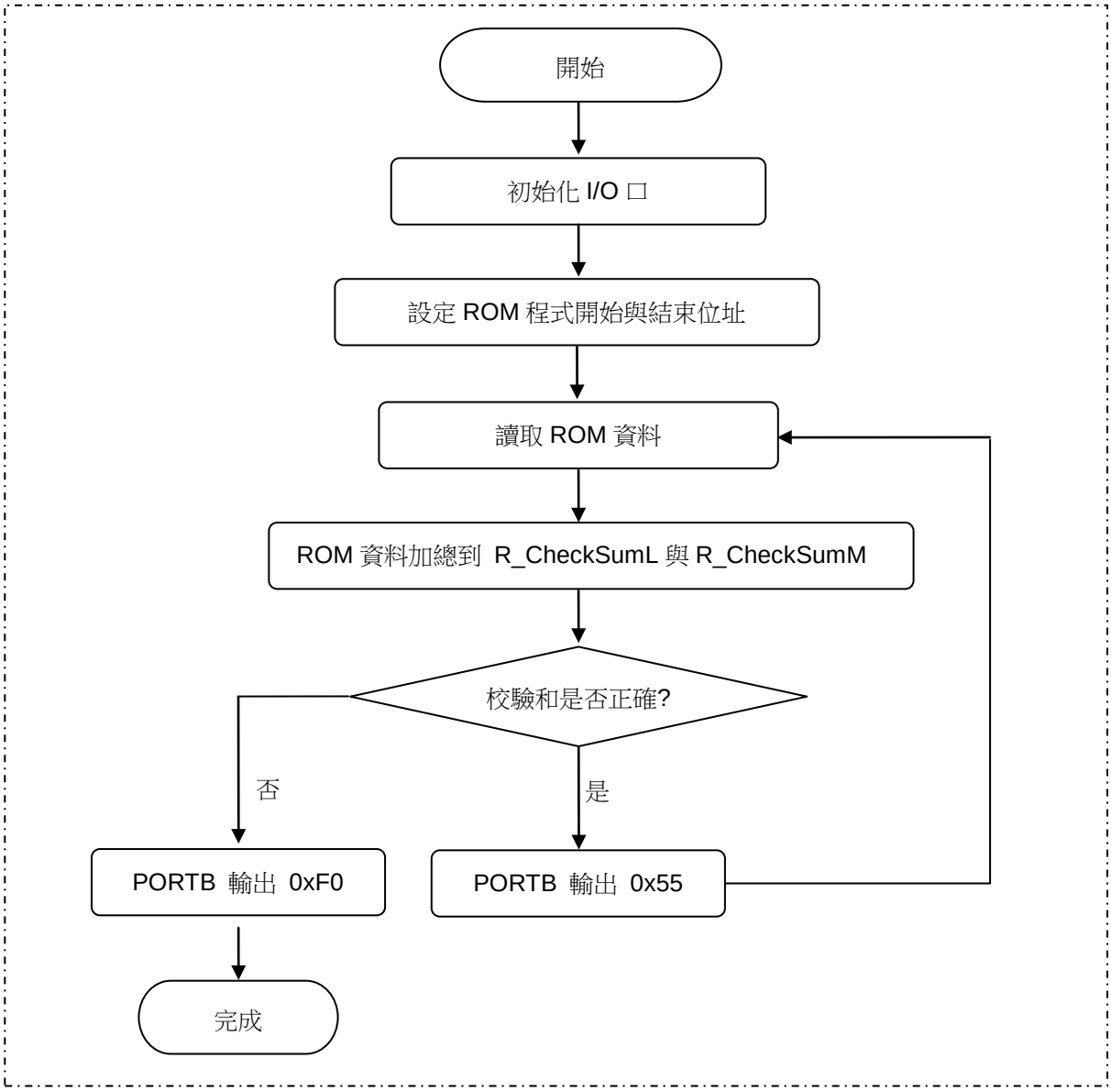
Note: 僅貼出重點程式段，完整範例程式請參考NYIDE中的Example Code。

2.10 Checksum (計算 ROM 校驗和)

2.10.1 功能介紹

功能說明：1. ROM校驗和計算。	
輸入	功能及輸出說明
NA	1. 校驗和正確 → PORTB 輸出 0x55 2. 校驗和錯誤 → PORTB 輸出 0xF0

2.10.2 設計流程圖



2.10.3 程式說明

```
;--------------------------
; 定義變數
R_Tab_IndexL    EQU     0X10            ; 存放 ROM 位址索引值(低位元組)
R_Tab_IndexH    EQU     0X11            ; 存放 ROM 位址索引值(高位元組)
R_CheckSumL     EQU     0X12            ; 存放計算後的校驗和(低位元組)
R_CheckSumM     EQU     0X13            ; 存放計算後的校驗和(高位元組)
;--------------------------
; 定義常數
#Define        C_StartAddr     0x0000          ; 定義計算 ROM 起始位址

L_ReadROM_Start:
; 初始化變數
    movia   0x00
    movar   R_CheckSumL
    movar   R_CheckSumM
    movia   Low C_StartAddr
    movar   R_Tab_IndexL
    movia   Mid C_StartAddr
    movar   R_Tab_IndexH
    sfun    Ps_TbHigh_Addr
; 讀取ROM資料
L_ReadROM_Loop:
    clrwdt
    movr    R_Tab_IndexL,C_SaveToAcc
    tablea
    addar   R_CheckSumL,C_SaveToReg
    sfunr   Ps_TbHigh_Data
    adcar   R_CheckSumM,C_SaveToReg
    movia   Low L_CheckSum_Addr-1
    cmpar   R_Tab_IndexL
    btrss   Pr_Status,C_Status_Z_Bit
    goto    L_ReadROM_IndexInc
    movia   Mid L_CheckSum_Addr
    cmpar   R_Tab_IndexH
    btrss   Pr_Status,C_Status_Z_Bit
    goto    L_ReadROM_IndexInc
    clrr    R_Tab_IndexL
```

```
        clrr    R_Tab_IndexH
        lgoto   L_CheckValue

L_ReadROM_IndexInc:
        incr    R_Tab_IndexL,C_SaveToReg
        btrss   Pr_Status,C_Status_Z_Bit
        goto    L_ReadROM_Loop
        incr    R_Tab_IndexH,C_SaveToReg
        movr    R_Tab_IndexH,C_SaveToAcc
        sfun    Ps_TbHigh_Addr
        lgoto   L_ReadROM_Loop

; 判斷校驗和是否正確
L_CheckValue:
        movia   Mid L_CheckSum_Addr
        sfun    Ps_TbHigh_Addr
        movia   Low L_CheckSum_Addr
        tablea
        cmpar   R_CheckSumL
        btrss   Pr_Status,C_Status_Z_Bit
        lgoto   L_FailLoop

        movia   Low L_CheckSum_Addr
        addia   0x1
        btrss   Pr_Status,C_Status_C_Bit
        goto    L_Check_HighByte
        movia   Mid L_CheckSum_Addr
        addia   0x1
        sfun    Ps_TbHigh_Addr

L_Check_HighByte:
        movia   Low L_CheckSum_Addr+1
        tablea
        cmpar   R_CheckSumM
        btrss   Pr_Status,C_Status_Z_Bit
        lgoto   L_FailLoop
        clrr    R_CheckSumL
        clrr    R_CheckSumM
```

```
        movia   0x55                    ; 校驗和正確
        movar   Pr_PB_Data              ; PORTB輸出0x55
        movr    Pr_PA_Data,C_SaveToAcc
        xoria   0x01
        movar   Pr_PA_Data
        lgoto   L_ReadROM_Start
; 校驗和錯誤
L_FailLoop:
        movia   0xF0                    ; PORTB輸出0xF0
        movar   Pr_PB_Data
        clrwdt
        lgoto   L_FailLoop

L_CheckSum_Addr:                        ; ROM程式資料最尾處
;---------------------------
; 結束程式
        end
```

Note: 僅貼出重點程式段，完整範例程式請參考 NYIDE 中的 Example Code。