九齐科技股份有限公司

Nyquest Technology Co., Ltd.

用户手册

NY8 系列范例程序

MCU Assembly Programmer

Version 1.0

May 26, 2015

改 版 记 录

版本	日期	内 容 描 述	修正页
1.0	2015/5/26	新发布。	-

目 录

1 简介

随着科技的进步及电子产品不断更新。九齐科技始终秉持着诚信、精确、效率的经营理念，为客户提供高质量及高附加价值的 NY8 系列 8 位单芯片，并提供优质的服务。为使客户能够更方便快捷的使用九齐 NY8 系列 IC，九齐科技针对 NY8 系列 IC 开发一系列的范例程序— NY8 Example Code。通过本篇介绍的 NY8 系列 IC 范例程序，初学者只需要进行简单的培训，即可在短时间内了解 NY8 系列 IC 的程序撰写技巧、功能以及应用方式，以致完成 NY8 系列 IC 的产品项目开发。

用户可以在九齐科技网站来获得 *NYIDE* 的安装程序文件。安装 *NYIDE* 后，打开 *NYIDE* 程序并在建立新项目时就可选择所需的范例程序。

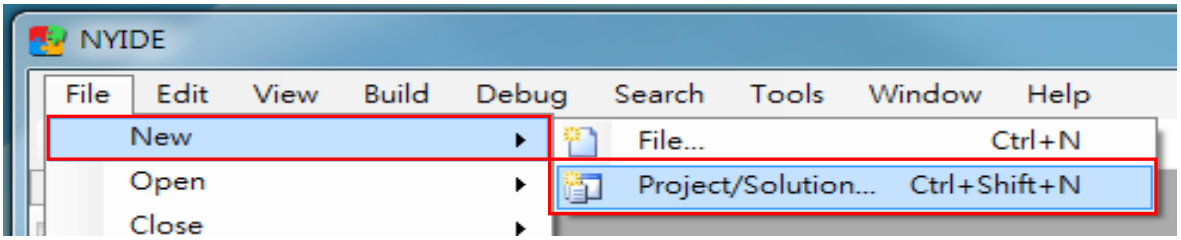
1.1 内容

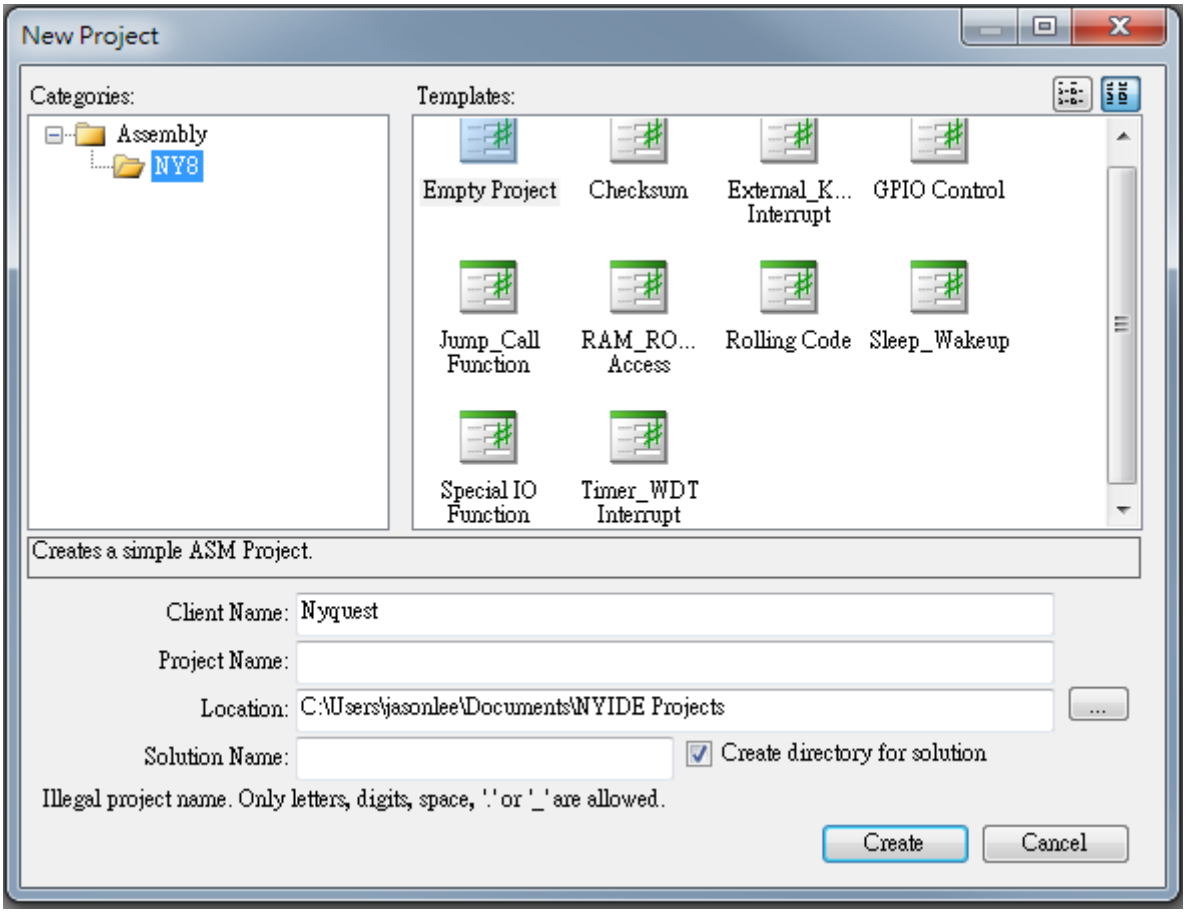
1.2 NY8 范例程序项目

- Empty Project……..(空白专案)
- External_Key Change Interrupt……(外部中断与 PB input change 中断)
- Timer_WDT Interrupt……..(时钟中断与看门狗时钟中断)
- GPIO Control……..(通用输入/输出口控制)
- Special_IO Function……(IR carrier、PWM、Buzzer 输出)
- Jump_Call Function……(分支跳跃指令与子过程调用指令)
- RAM_ROM_REG Access……..(RAM、ROM、SFR 存取)
- Sleep_Wakeup……(切换至 Halt / Standby mode 与唤醒)
- Rolling Code……(读取滚动码)
- Checksum……...(计算 ROM 校验和)

1.3 安装软件工具(*NYIDE*)

请先安装九齐科的*NYIDE*。然后执行*NYIDE*程序后接着选择打开新项目，就可选择所需的NY8范例程序(NY8 Example Code)。操作步骤如下图所示：





注意：在安装 NYIDE 时，需要同时安装 NYASM。

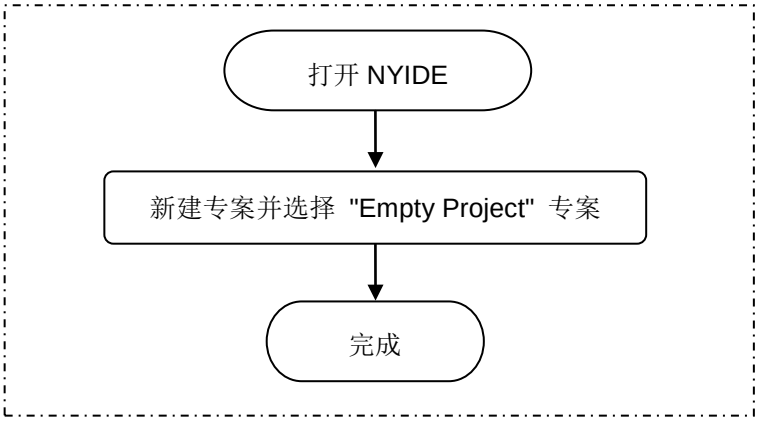
2 NY8 范例程序说明

2.1 Empty Project (空白专案)

2.1.1 功能介绍

功能说明：建立新项目并加载Header file "NY8.H"。	
输入	功能及输出说明
NA	建立新项目并加载Header file "NY8.H"。

2.1.2 设计流程图



2.1.3 程序说明

```
;---------------------------
; 表头批注
; Project:          ; 专案名称
; File:             ; 文件名称
; Description:      ; 程序叙述
; Author:           ; 程序开发者
; Version:          ; 版本别
; Date:             ; 日期
;---------------------------
; 添加 NY8A051A / 053A Series Header File
#include    NY8.H
;---------------------------
; 定义变量
;---------------------------
; 定义常数
```

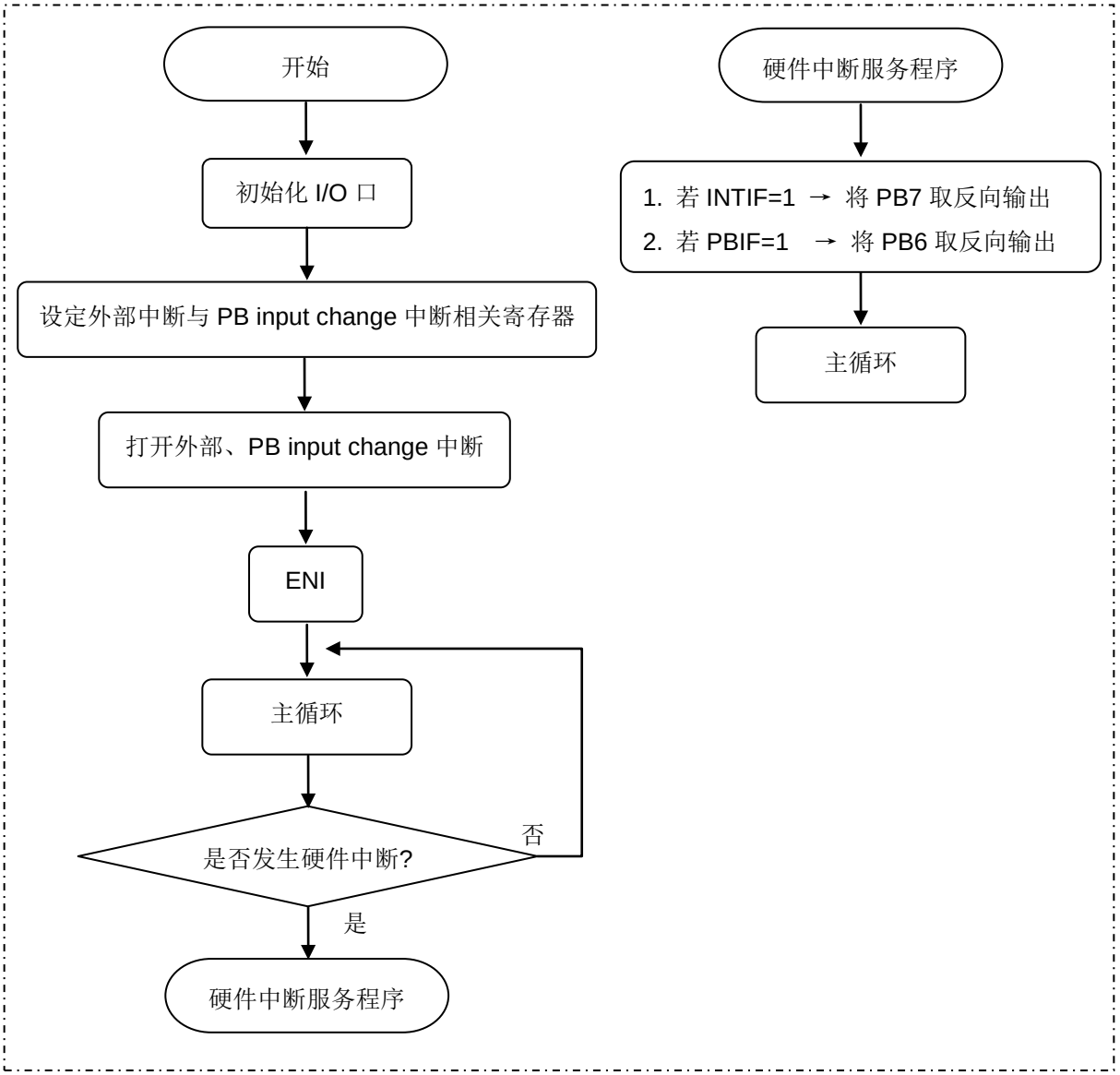
```
;--------------------------
; 定义向量
ORG     0x000                   ; 电源重置向量地址
goto    V_Main
ORG     0x008                   ; 硬件中断向量地址
goto    V_INT
;--------------------------
; 程序开头处
ORG     0x010                   ; ROM地址  0x010
V_Main:

L_MainLoop:                     ; 程序主循环
    clrwdt                      ; 清除看门狗时钟
    lgoto           L_MainLoop
;--------------------------
; 硬件中断服务程序
V_INT:                          ; 硬件中断服务程序开头处
    retie                       ; 从硬件中断服务程序返回
;--------------------------
; 结束程序
end
```

2.2 External_Key Change Interrupt (外部中断与 PB input change 中断)

2.2.1 功能介绍

功能说明：1. 使用外部中断与PB input change中断。	
输入	功能及输出说明
INT	输入INT信号上升缘时发生中断 → PB7 取反向输出
PB1	输入PB1信号上升/下降缘时发生中断 → PB6 取反向输出

2.2.2 设计流程图



2.2.3 程序说明

```
V_Main:
; 关闭所有中断
    disi
; 打开PB1与PB0上拉电阻
    movia   ~(C_PB1_PHB | C_PB0_PHB)
    movar   Pr_PB_PH_Ctrl
; 打开PB1 input change wakeup
    movia   C_PB1_Wakeup
    movar   Pr_PB_WakeUp_Ctrl
; 设定PB1与PB0为输入口
    movia   C_PB1_Input | C_PB0_Input
    iost    Pf_PB_Dir_Ctrl
    movia   0x00
    movar   Pr_PB_Data
; 设置外部中断相关寄存器
    movia   C_EXINT_Edge
    t0md
    movia   C_ExtINT_En
    movar   Pr_PWR_Ctrl
; 打开外部中断与PB input change中断
    movia   C_INT_EXT | C_INT_PBKey
    movar   Pr_INT_Ctrl
    Movia   0x00                        ; 清除所有的中断旗标
    Movar   Pr_INT_Flag
    eni
;---------------------------
; 主循环
L_MainLoop:
    clrwdt
    goto    L_MainLoop
;---------------------------
; 硬件中断服务程序开始处
V_INT:
    movar   R_AccBuf                    ; 保存ACC值到变量R_AccBuf
    swapr   R_AccBuf,C_SaveToReg
    movr    Pr_Status,C_SaveToAcc
```

```
    movar     R_StatusBuf                    ; 保存STATUS值到变量R_StatusBuf
;----------------------------
; 外部中断服务程序
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                     ; 清除外部中断旗标
    movar     Pr_INT_Flag
    goto      L_RET2Main
;----------------------------
; PB input change中断服务程序
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中断旗标
    movar     Pr_INT_Flag

L_RET2Main:
    movr      R_StatusBuf,C_SaveToAcc
    movar     Pr_Status                      ; 从R_StatusBuf回存STATUS值
    swapr     R_AccBuf,C_SaveToAcc           ; 从R_AccBuf回存ACC值
    retie                                    ; 从硬件中断服务程序返回
```

注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

2.3 Timer_WDT Interrupt (时钟中断与看门狗时钟中断)

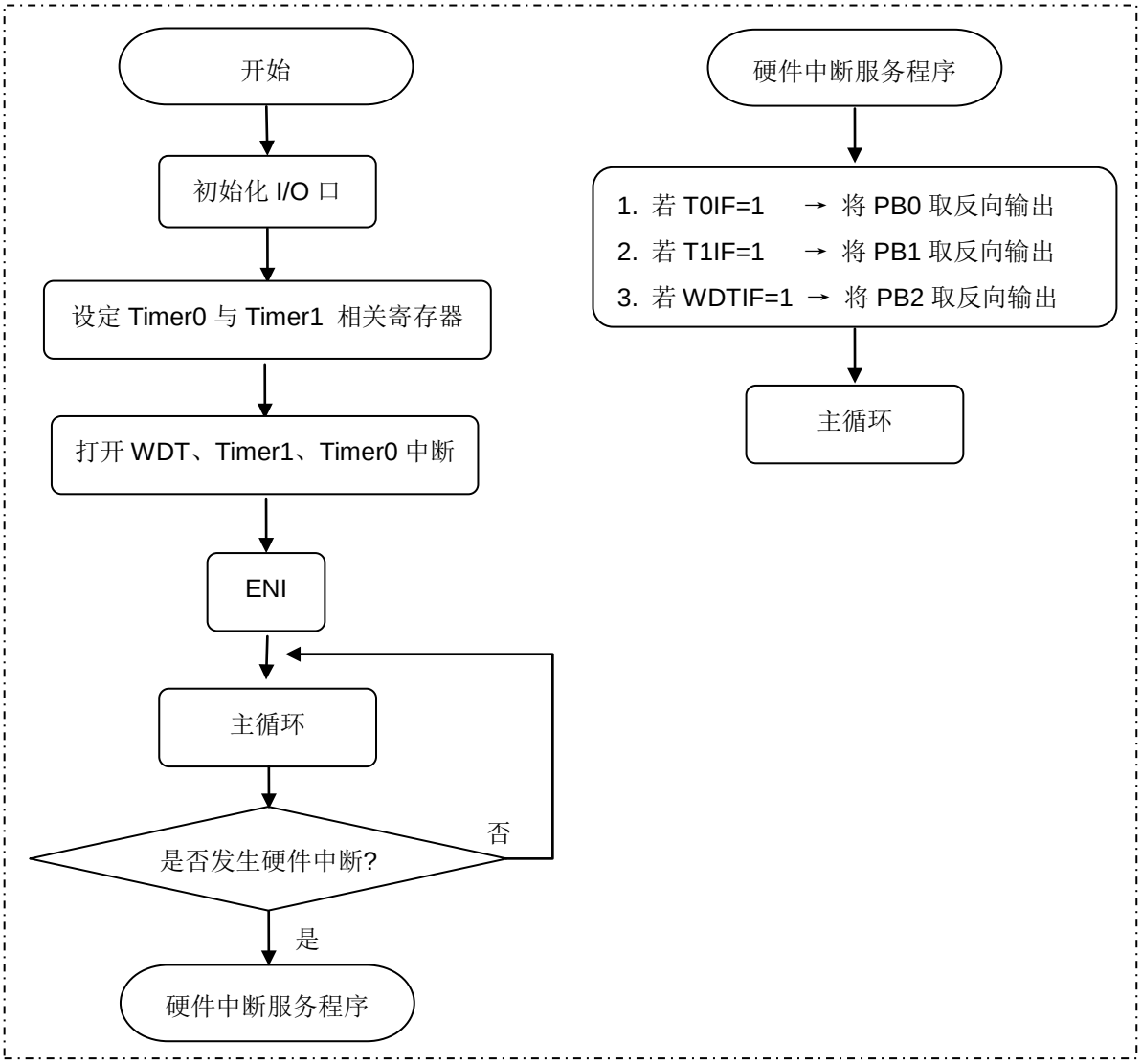
2.3.1 功能介绍

功能说明：

1. F_{INST}= 4MHz/4T (I_HRC) = 1MHz
2. 请将*NYIDE* 的 Project Options中 WDT time base设为 3.5ms
3. 设置Timer0中断每 2048个指令周期发生一次， 并将PB0取反向输出
4. 设置Timer1中断每 1024个指令周期发生一次， 并将PB1取反向输出
5. 设置WDT中断每 3.5ms发生一次， 并将PB2取反向输出

输入	功能及输出说明
NA	1. Timer0中断每 2048个指令周期发生一次， 并将PB0取反向输出
NA	2. Timer1中断每 1024个指令周期发生一次， 并将PB1取反向输出
NA	3. WDT中断每 3.5ms发生一次， 并将PB2取反向输出

2.3.2 设计流程图



2.3.3 程序说明

```
; 设置 Timer0 相关寄存器
        movia   C_TMR0_Dis
        iost    Pf_PWR_Ctrl1                    ; 关闭 Timer0
        movia   0x00
        movar   Pr_TMR0_Data
        movia   C_PS0_TMR0 | C_PS0_Div8
        t0md
;-------------------------------------------------------------------
   ;若 WDT 需要 Prescaler0 分频 ( Prescaler0 分频只能择一让 Timer0 或 WDT 使用)
   ;    movia   (0x00 | C_PS0WDT_Sel)
   ;    t0md
;-------------------------------------------------------------------
; 设置 Timer0 相关寄存器
        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
; 打开 WDT 中断、Timer1 中断、Timer0 中断
        movia   C_INT_WDT | C_INT_TMR1 | C_INT_TMR0
        movar   Pr_INT_Ctrl
; 打开 WDT
        movia   C_WDT_En | C_LVR_En
        movar   Pr_PWR_Ctrl
; 打开 Timer0
        movia   C_TMR0_En
        iost    Pf_PWR_Ctrl1
        eni                                     ; 打开 all unmasked 中断
;-------------------------------------------------------------
; 主循环
L_MainLoop:
        nop
        goto    L_MainLoop

;-------------------------
; 硬件中断服务程序开始处
```

```
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中断服务程序
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 中断旗标
    movar   Pr_INT_Flag
;---------------------------
; Timer0中断服务程序
L_TIME0_INT:
    btrss   Pr_INT_Flag,C_INT_TMR0_Bit
    goto    L_WDT_INT
    movia   0x00
    movar   Pr_TMR0_Data            ; 重载 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 中断旗标
        movar   Pr_INT_Flag
;--------------------------
; WDT 中断服务程序
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 中断旗标
        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
```

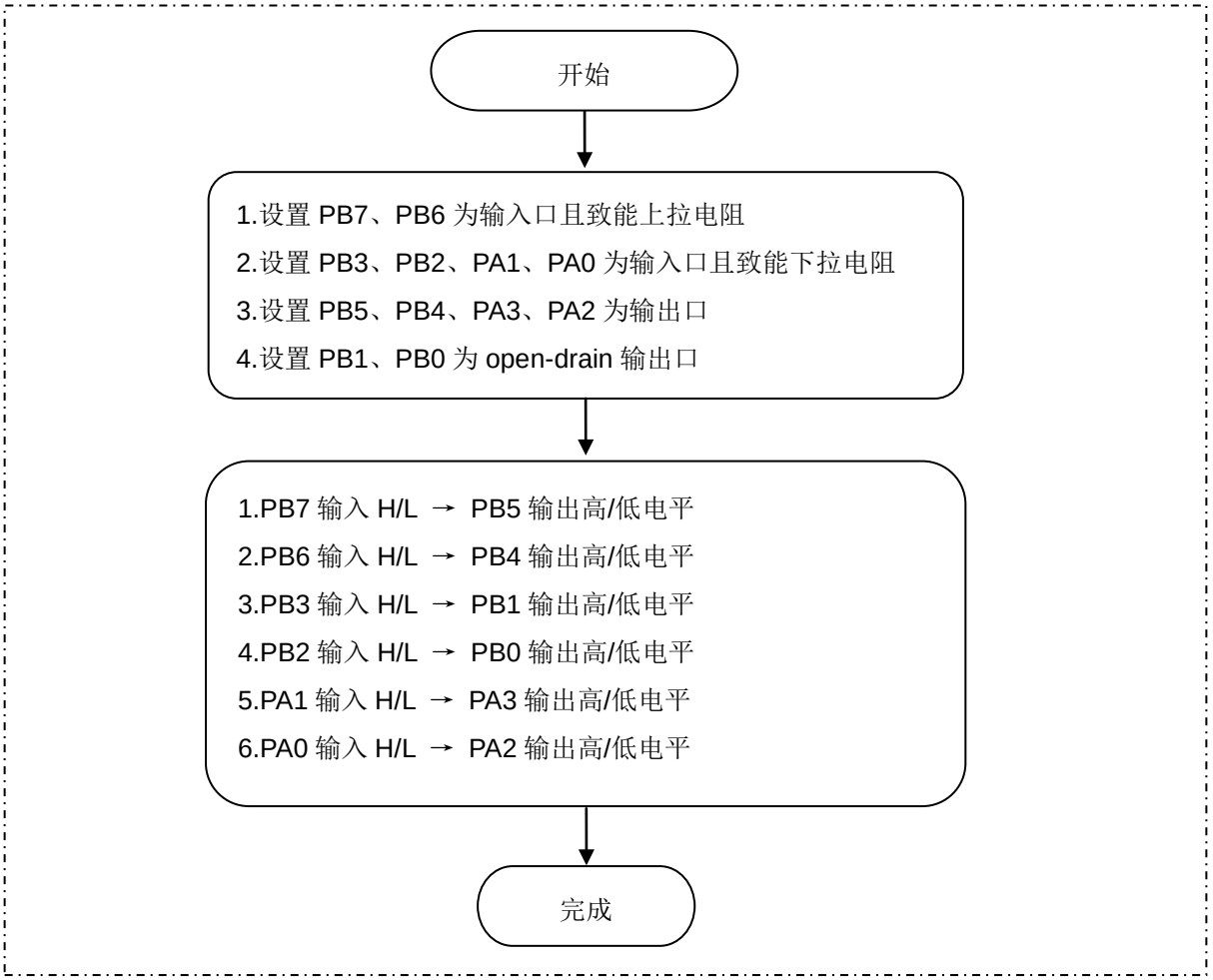
注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

2.4 GPIO Control (通用输入/输出口控制)

2.4.1 功能介绍

功能说明：1. 设置 GPIO 相关寄存器。	
输入	功能及输出说明
PB7	PB7 输入 H/L → PB5 输出 高/低电平
PB6	PB6 输入 H/L → PB4 输出 高/低电平
PB3	PB3 输入 H/L → PB1 输出 高/低电平
PB2	PB2 输入 H/L → PB0 输出 高/低电平
PA1	PA1 输入 H/L → PA3 输出 高/低电平
PA0	PA0 输入 H/L → PA2 输出 高/低电平

2.4.2 设计流程图



2.4.3 程序说明

```
; 打开 PB1、PB0为 open-drain 输出
    movia     C_PB1_OD | C_PB0_OD
    iost      Pf_PB_OD_Ctrl
; 打开 PB3、PB2、PA1、PA0 下拉电阻
    movia     ~( C_PB3_PLB |   C_PB2_PLB |   C_PA1_PLB |   C_PA0_PLB)
    movar     Pr_PAB_PL_Ctrl
; 打开 PB7、PB6 上拉电阻
    movia     ~( C_PB7_PHB | C_PB6_PHB)
    movar     Pr_PB_PH_Ctrl
; 设置 PB7、PB6、PB3、PB2为输入口
; 设置 PB5、PB4、PB1、PB0为输出口
    movia     C_PB7_Input | C_PB6_Input | C_PB3_Input | C_PB2_Input
    iost      Pf_PB_Dir_Ctrl
    movia     0x03
    movar     Pr_PB_Data
; 设置 PA1、PA0 为输入口
; 设置 PA3、PA2 为输出口
    movia     C_PA1_Input | C_PA0_Input
    iost      Pf_PA_Dir_Ctrl
    movia     0x0C
    movar     Pr_PA_Data
```

注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

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

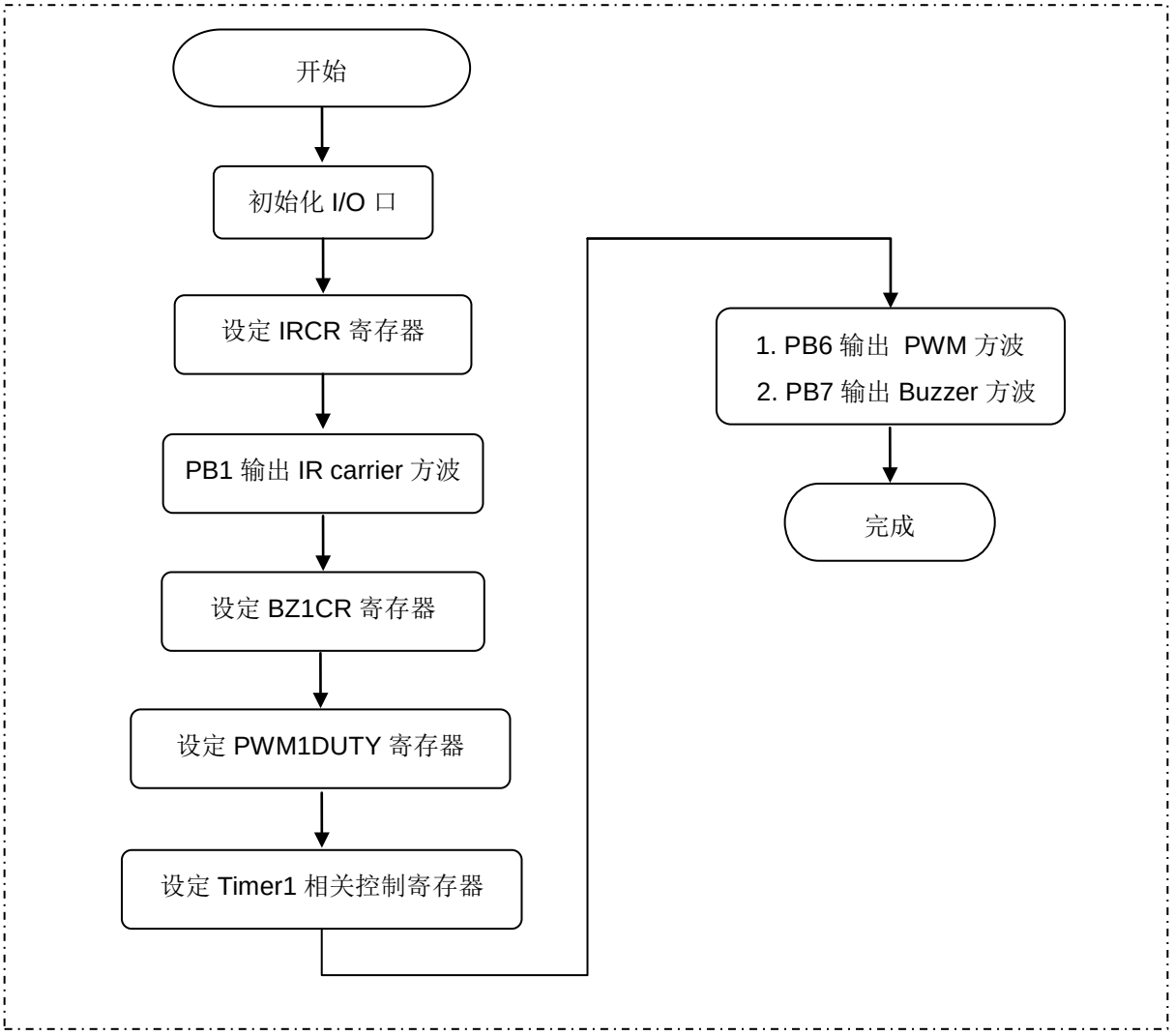
2.5.1 功能介绍

功能说明：

1. 本范例程序使用外接晶振 3.58MHz (PB4 / PB5)
2. 由PB1脚位输出约38KHz的 IR carrier方波
3. 由PB6脚位输出频率约为1.75KHz， duty cycle为25%的PWM方波
4. 由PB7脚位输出频率约为55.94KHz的Buzzer方波

输入	功能及输出说明
NA	1. 由PB1脚位输出约38KHz的 IR carrier方波
NA	2. 由PB6脚位输出频率约为1.75KHz， duty cycle为25%的PWM方波
NA	3. 由PB7脚位输出频率约为55.94KHz的Buzzer方波

2.5.2 设计流程图



2.5.3 程序说明

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

注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

2.6 Jump_Call Function (分支跳跃指令与子过程调用指令)

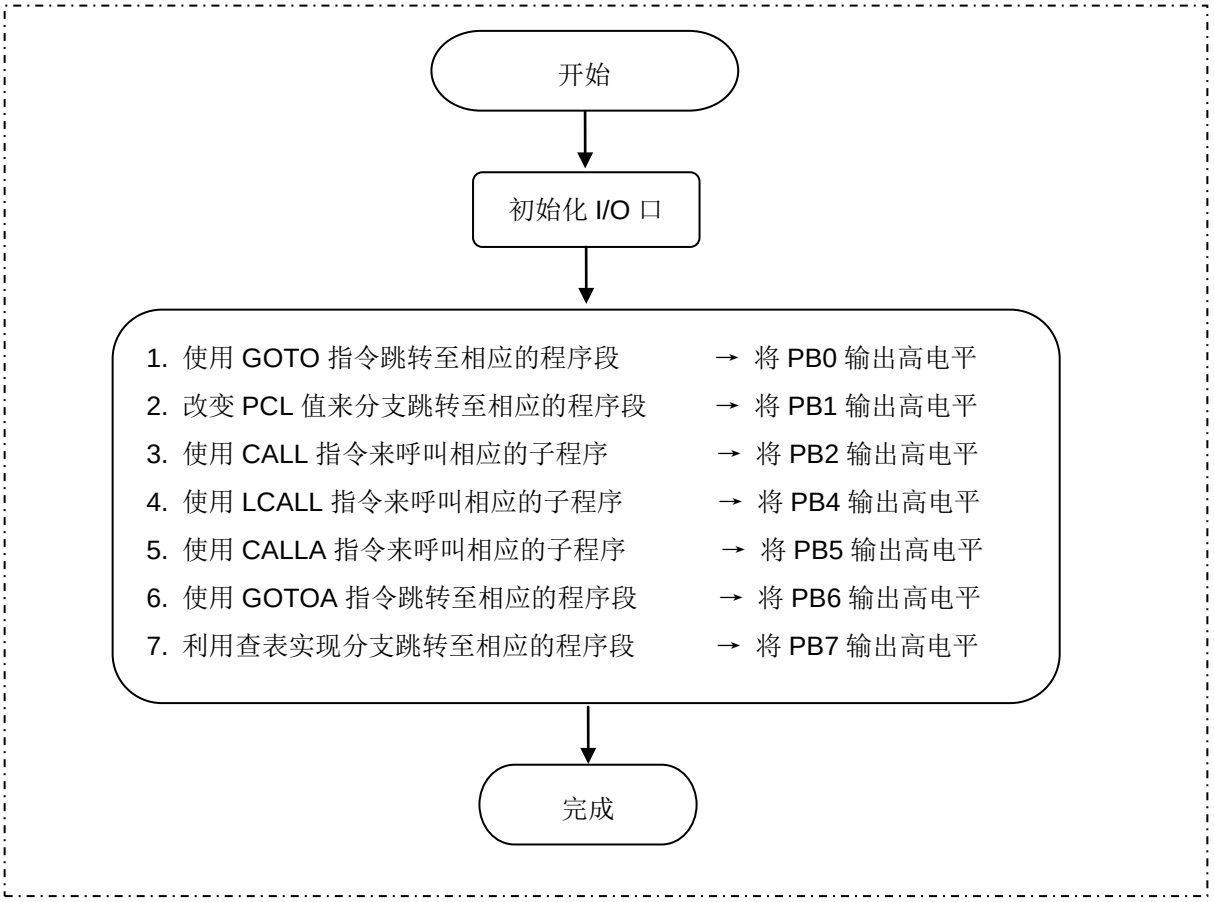
2.6.1 功能介绍

功能说明：

1. 使用分支跳转指令(GOTO、GOTOA)
2. 使用呼叫子程序指令(CALL、LCALL、CALLA)
3. 改变PCL值来实现分支跳转
4. 利用查表实现分支跳转

输入	功能及输出说明	
NA	使用GOTO指令跳转至相应的程序段	→ 将PB0输出高电平
NA	改变PCL值来分支跳转至相应的程序段	→ 将PB1输出高电平
NA	使用CALL指令来呼叫相应的子程序	→ 将PB2输出高电平
NA	使用LCALL指令来呼叫相应的子程序	→ 将PB4输出高电平
NA	使用CALLA指令来呼叫相应的子程序	→ 将PB5输出高电平
NA	使用GOTOA指令跳转至相应的程序段	→ 将PB6输出高电平
NA	利用查表实现分支跳转至相应的程序段	→ 将PB7输出高电平

2.6.2 设计流程图



2.6.3 程序说明

```
;---------------------------
; 使用GOTO指令跳转至相应的程序段
        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
;---------------------------
; 改变PCL值来实现分支跳转至相应的程序段
        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指令来呼叫相应的子程序
        call    F_sub1
;---------------------------
; 使用LCALL指令来呼叫相应的子程序
        lcall   F_sub2
;---------------------------
; 使用CALLA指令来呼叫相应的子程序
        movia   0x02
        sfun    Ps_TbHigh_Addr
        movia   0x20
        calla
```

```
;--------------------------
; 使用GOTOA指令跳转至相应的程序段
        movia   0x03
        sfun    Ps_TbHigh_Addr
        movia   0x10
        gotoa                           ; 分支跳转至ROM地址"0x310"
;--------------------------
利用查表实现分支跳转
        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                           ; 分支跳转至ROM地址"0x330"
;--------------------------
; 定义表格
        ORG     0x350                   ; ROM地址"0x350"
T_DataTbl:
        DW      0x0330                  ; 表格内容
```

注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

2.7 RAM_ROM_REG Access (RAM、ROM、SFR 存取)

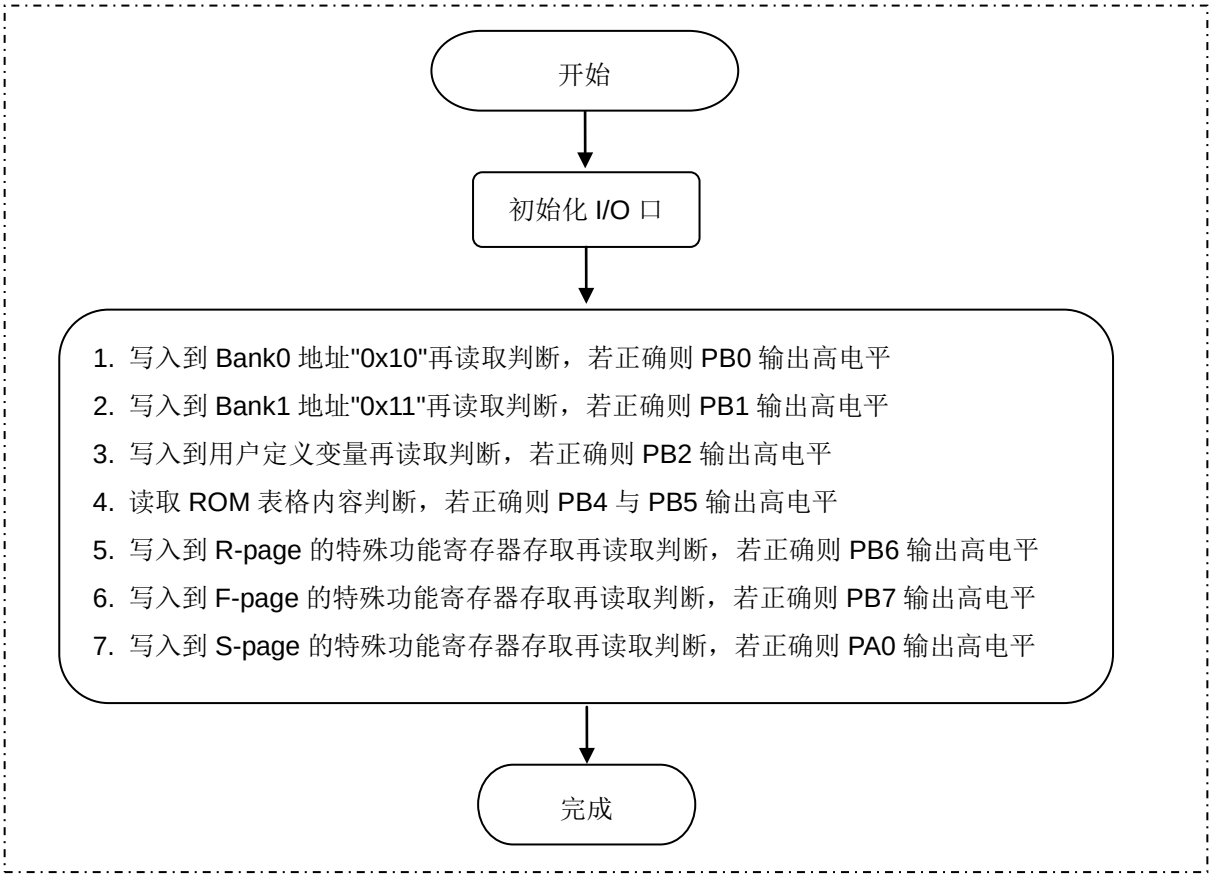
2.7.1 功能介绍

功能说明：

1. 一般用途寄存器存取 (General Purpose Register)
2. 读取ROM数据 (Program Memory)
3. 特殊功能寄存器存取 (Special-Function Register)

输入	功能及输出说明
NA	写入到Bank0地址"0x10"再读取判断， 若正确则PB0输出高电平
NA	写入到Bank1地址"0x11"再读取判断， 若正确则PB1输出高电平
NA	写入到用户定义变量再读取判断， 若正确则PB2输出高电平
NA	读取ROM表格内容判断， 若正确则PB4与PB5输出高电平
NA	写入到R-page的特殊功能寄存器存取再读取判断， 若正确则PB6输出高电平
NA	写入到F-page的特殊功能寄存器存取再读取判断， 若正确则PB7输出高电平
NA	写入到S-page的特殊功能寄存器存取再读取判断， 若正确则PA0输出高电平

2.7.2 设计流程图



2.7.3 程序说明

```
; 写入 "0x55" 到 RAM Bank0 地址 "0x10"
        movia   C_SFR_Bank0 | 0x10
        movar   Pr_File_Sel
        movia   0x55
        movar   Pr_Indir_Addr
        clra
        clrr    Pr_File_Sel
; 读取 RAM Bank0 地址 "0x10" 内容
        movia   C_SFR_Bank0 | 0x10
        movar   Pr_File_Sel
        movr    Pr_Indir_Addr,C_SaveToAcc
; 写入 "0xAA" 到 RAM Bank1 地址 "0x11"
        movia   C_SFR_Bank1 | 0x11
        movar   Pr_File_Sel
        movia   0xAA
        movar   Pr_Indir_Addr
        clra
        clrr    Pr_File_Sel
; 读取 RAM Bank1 地址 "0x11" 内容
        movia   C_SFR_Bank1 | 0x11
        movar   Pr_File_Sel
        movr    Pr_Indir_Addr,C_SaveToAcc
; 写入 "0x5A" 到用户定义变量 "R_RAM_0x20"
        movia   0x5A
        movar   R_RAM_0x20
        clra
; 读取用户定义变量 "R_RAM_0x20" 内容
        movr    R_RAM_0x20,C_SaveToAcc
```

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

注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

2.8 Sleep_Wakeup (切换至 Halt / Standby mode 与唤醒)

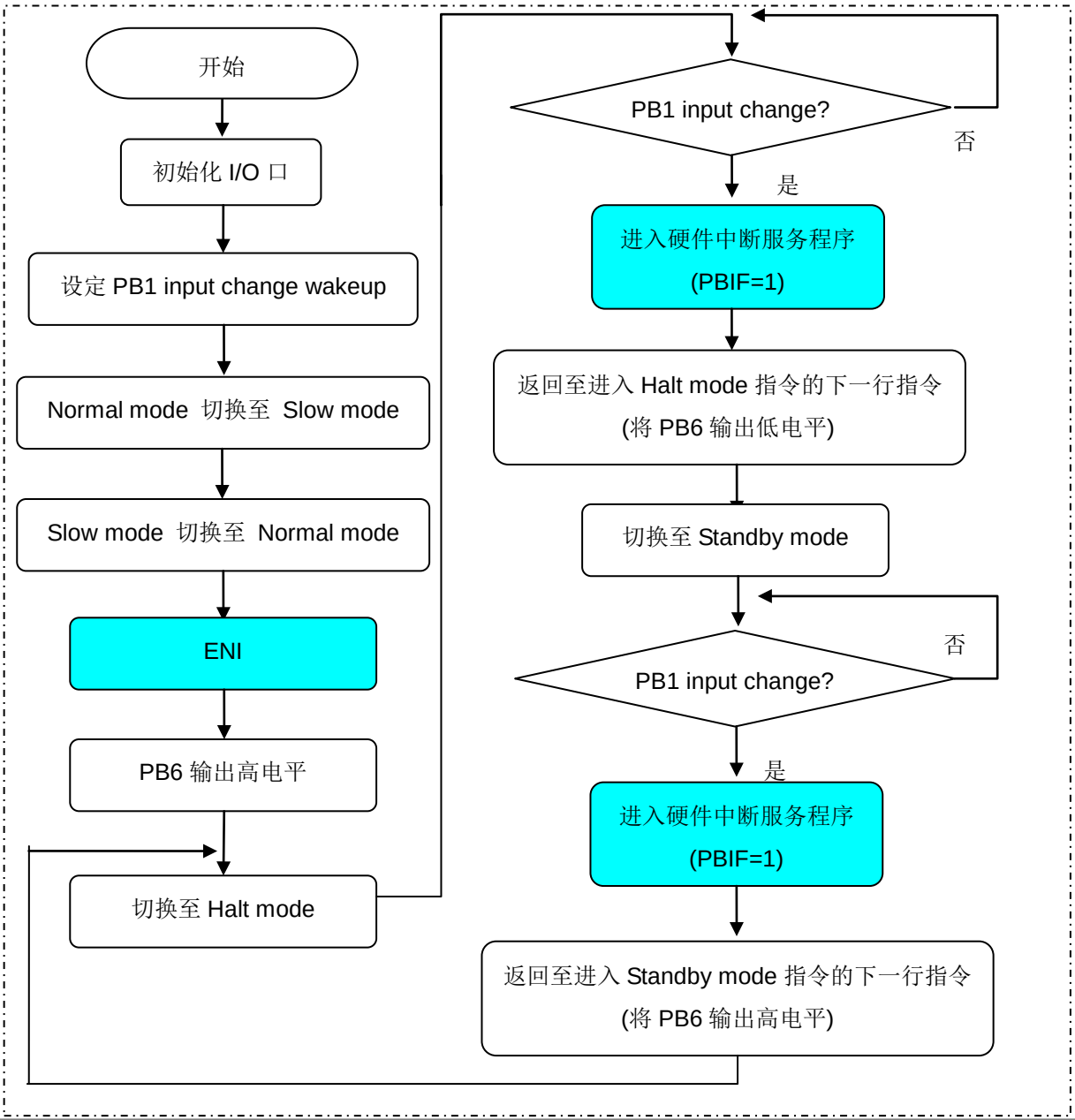
2.8.1 功能介绍

功能说明：

1. Normal mode 切换至 Slow mode 再切换回 Normal mode
2. 切换至 Halt mode 并以 PB1 input change 唤醒
3. 切换至 Standby mode 并以 PB1 input change 唤醒

输入	功能及输出说明
PB1	2. 将PB6输出高电平再切换至Halt mode， 从Halt mode唤醒后将PB6输出低电平
PB1	3. 接着切换至Standby mode， 从Standby mode唤醒后将PB6输出高电平

2.8.2 设计流程图



2.8.3 程序说明

```
; 设置 PB1 input change wakeup
        movia   C_PB1_Wakeup
        movar   Pr_PB_WakeUp_Ctrl
 ; 打开 PortB input change wakeup 中断
        movia   C_INT_PBKey
        movar   Pr_INT_Ctrl
        movia   0x00                            ; 清除所有的中断旗标
        movar   Pr_INT_Flag
 ; Normal mode 切换至 Slow mode
        movia   C_FLOSC_Sel
        sfun    Ps_SYS_Ctrl
 ; Slow mode 切换至 Normal mode
        movia   C_FHOSC_Sel
        sfun    Ps_SYS_Ctrl
;-------------------------------------------------------------------------------------------
; 选择 “eni” 或 “disi” 指令,来决定从 Halt mode 或 Standby mode 唤醒后是否进入硬件中断服务程序
        eni             ; 从 Halt mode 或 Standby mode 唤醒后进入硬件中断服务程序

        ;disi           ; 从 Halt mode 或 Standby mode 唤醒后不进入硬件中断服务程序
        ;-----------------------------------------------------------------------------------
        movr    Pr_PB_Data,C_SaveToReg
        ; 下列有两种指令方式进入 Halt mode,请择一使用
        ; 指令一
        sleep
        ; 指令二
        ;movia  C_Halt_Mode | C_FHOSC_Sel       ; 从 Normal mode 进入 Halt mode
        ;sfun   Pr_SYS_Ctrl
        ;-----------------------------------------------------------------------------------
        nop                                     ; for sleep wakeup interrupt latency time
        bcr     Pr_PB_Data,6                    ; 从 Halt mode 唤醒后将 PB6 输出低电平
        movia   ~C_INT_PBKey                    ; 清除 PB input change 中断旗标
        movar   Pr_INT_Flag
; -------------------------------------------------------------------------------------------
        ; 从 Normal mode 进入 Standby mode
        movr    Pr_PB_Data,C_SaveToReg
        movia   C_Standby_Mode | C_FHOSC_Sel
```

```
        sfun    Ps_SYS_Ctrl
        bsr     Pr_PB_Data,6                ; 从Standby mode唤醒后将PB6输出高电平
        movia   ~C_INT_PBKey                ; 清除 PB input change 中断旗标
        movar   Pr_INT_Flag
;---------------------------
; 硬件中断服务程序
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 中断旗标
        movar   Pr_INT_Flag
L_RET2Main:
        retie                               ; 从硬件中断服务程序返回
```

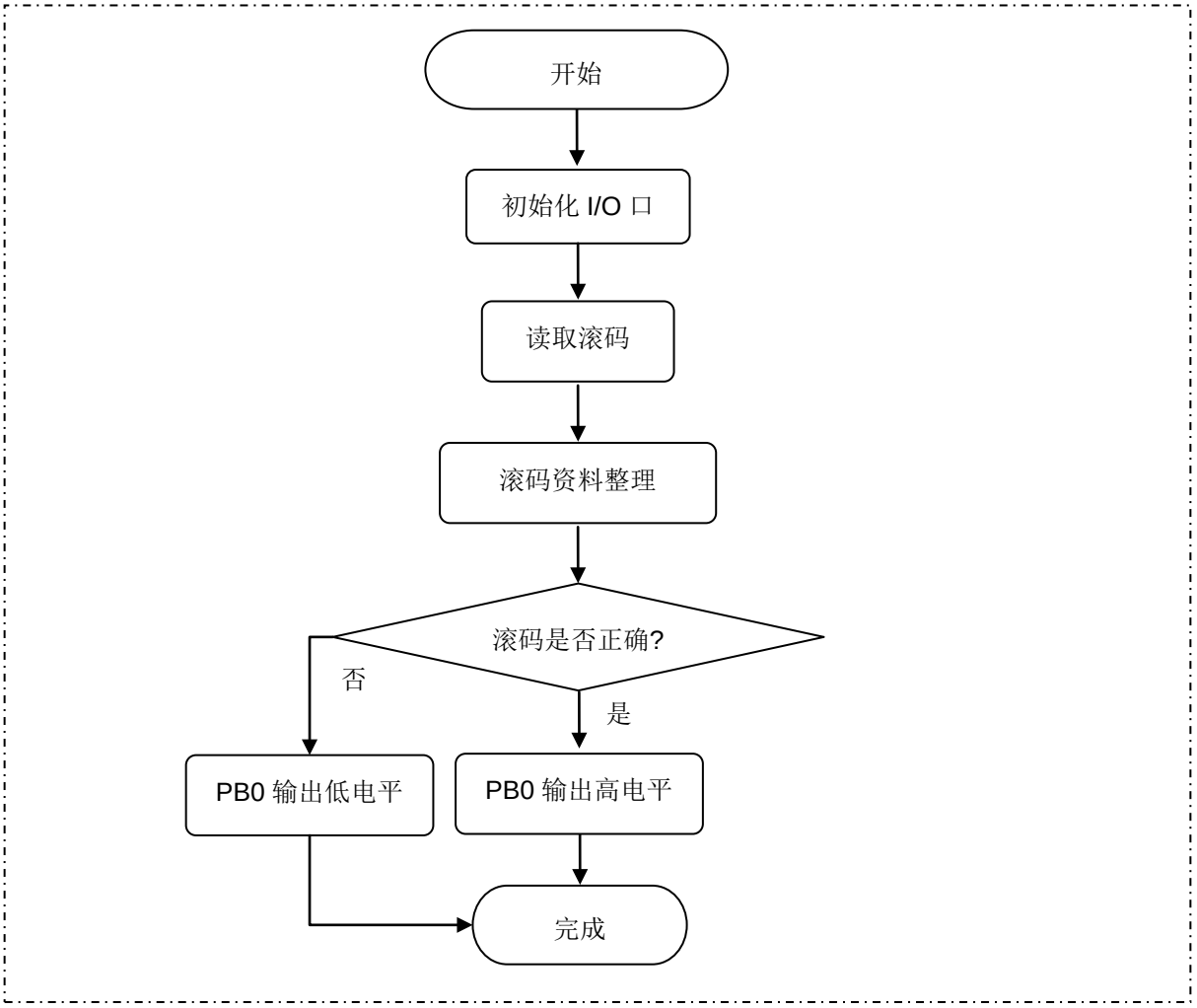
注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

2.9 Rolling Code (读取滚动码)

2.9.1 功能介绍

功能说明：1. 读取滚动码	
输入	功能及输出说明
NA	读取滚动码并判断， 若正确则PB0输出高电平

2.9.2 设计流程图



2.9.3 程序说明

```
;---------------------------
; 定义变量
 R_RollingCode_byte0     EQU     10H         ; 存放滚动码计算值 位 7 ~ 位 0
 R_RollingCode_byte1     EQU     11H
 R_RollingCode_byte2     EQU     12H         ; 存放滚动码计算值 位 15 ~ 位 8
 R_RollingCode_byte3     EQU     13H         ; 存放滚动码计算值 位 19 ~ 位 16
```

```
;--------------------------
; 定义常数
; 假设用户从 Q-Writer 烧入的滚动码为 961109(十进制) = 0xEAA55
 C_RC_B0                EQU     55H             ; 滚动码默认值 位 7 ~ 位 0
 C_RC_B1                EQU     AAH             ; 滚动码默认值 位 15 ~ 位 8
 C_RC_B2                EQU     0EH             ; 滚动码默认值 位 19 ~ 位 16

; 读取程序内存(ROM)地址 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
; 滚动码资料整理
        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
;判断滚动码是否正确
        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                  ;滚动码正确,将 PB0 输出高电平
```

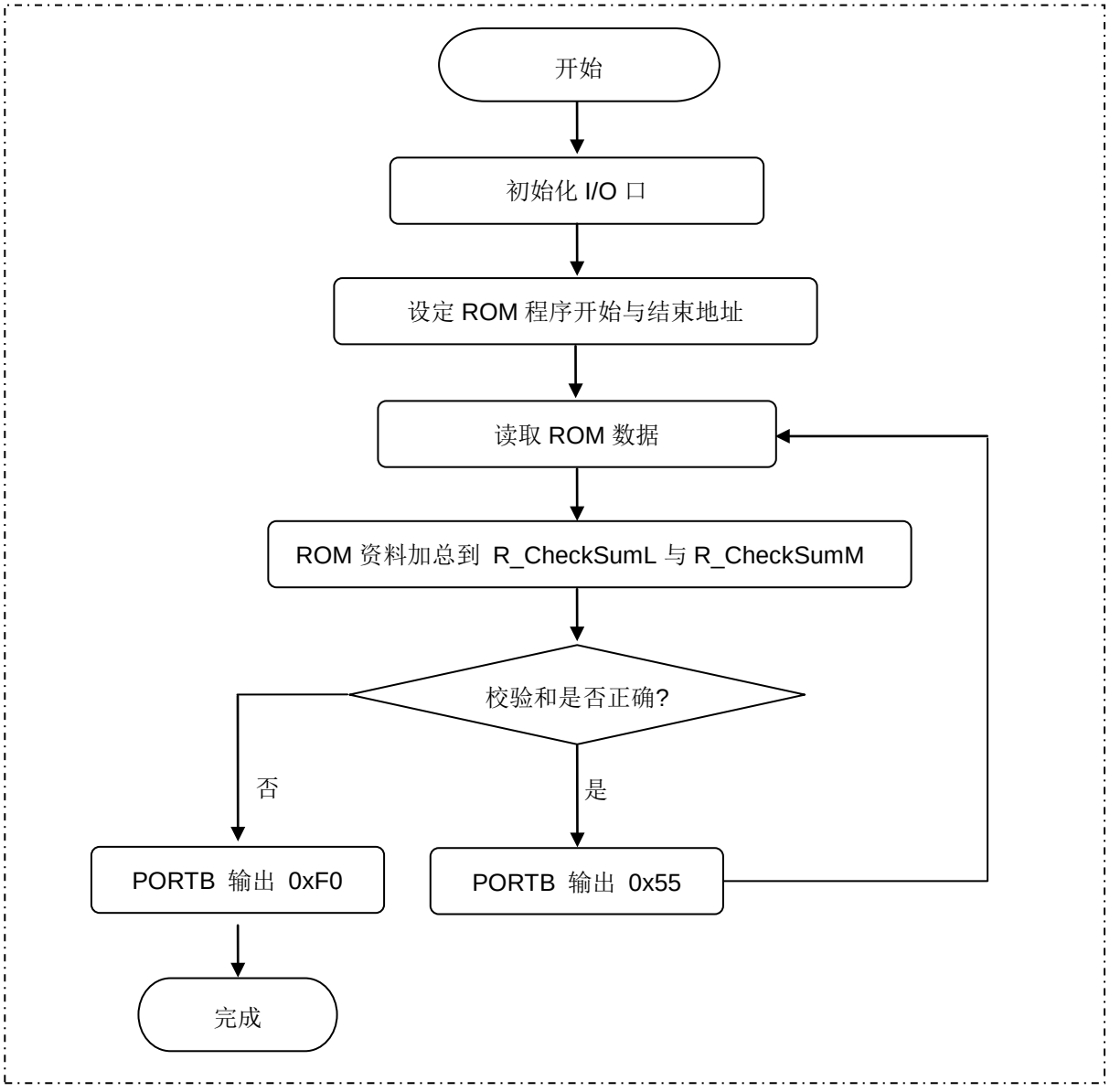
注意：仅列出重点程序段，完整范例程序请参考NYIDE中的Example Code。

2.10 Checksum (计算 ROM 校验和)

2.10.1 功能介绍

功能说明：1. ROM校验和计算。	
输入	功能及输出说明
NA	1. 校验和正确 → PORTB 输出 0x55 2. 校验和错误 → PORTB 输出 0xF0

2.10.2 设计流程图



2.10.3 程序说明

```
;--------------------------
; 定义变量
R_Tab_IndexL      EQU     0X10            ; 存放 ROM 地址索引值(低字节)
R_Tab_IndexH      EQU     0X11            ; 存放 ROM 地址索引值(高字节)
R_CheckSumL       EQU     0X12            ; 存放计算后的校验和(低字节)
R_CheckSumM       EQU     0X13            ; 存放计算后的校验和(高字节)
;--------------------------
; 定义常数
#Define          C_StartAddr     0x0000          ; 定义计算 ROM 起始地址

L_ReadROM_Start:
; 初始化变数
    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
; 读取ROM数据
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

; 判断校验和是否正确
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                    ; 校验和正确
        movar   Pr_PB_Data              ; PORTB输出0x55
        movr    Pr_PA_Data,C_SaveToAcc
        xoria   0x01
        movar   Pr_PA_Data
        lgoto   L_ReadROM_Start
; 校验和错误
L_FailLoop:
        movia   0xF0                    ; PORTB输出0xF0
        movar   Pr_PB_Data
        clrwdt
        lgoto   L_FailLoop

L_CheckSum_Addr:                        ; ROM程序数据最尾处
;---------------------------
; 结束程序
        end
```

注意：仅列出重点程序段，完整范例程序请参考 NYIDE 中的 Example Code。