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

User Manual

NY8 Series Example Code

MCU Assembly Programmer

Version 1.0
May 26, 2015

Revision History

Version	Date	Description	Modified Page
1.0	2015/5/26	New release.	-

Table of Contents

1 Introduction

With advancement of technology and updating electronic products, Nyquest always upholds the philosophy of honesty, precision and efficiency to provide customers high quality products, high value-add-on values and high quality service. In order to make customers to use Nyquest NY8 series IC with much convenience and efficiency, Nyquest provides NY8 Example Code user manual for users to edit programs. Through the introductions of the NY8 example code, beginners only need simple training and short-time learning to understand the program writing skills, functions and application modes of NY8 series IC to complete project development.

Users can contact Nyquest Technology to acquire the updated *NYIDE* installation file. After the installation of *NYIDE* is finished, users can open *NYIDE* and select the desired example code when creating a new project.

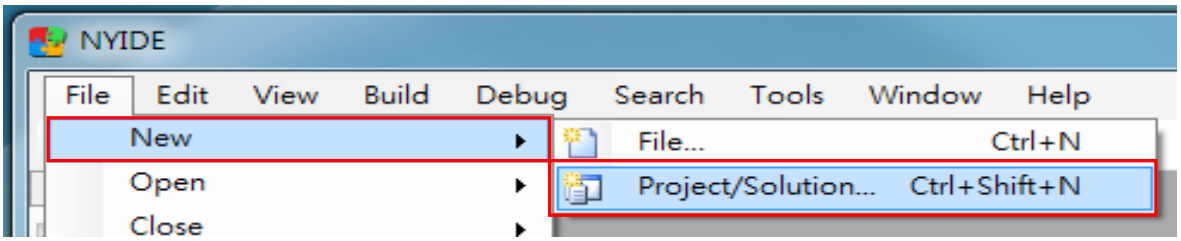
1.1 Contents

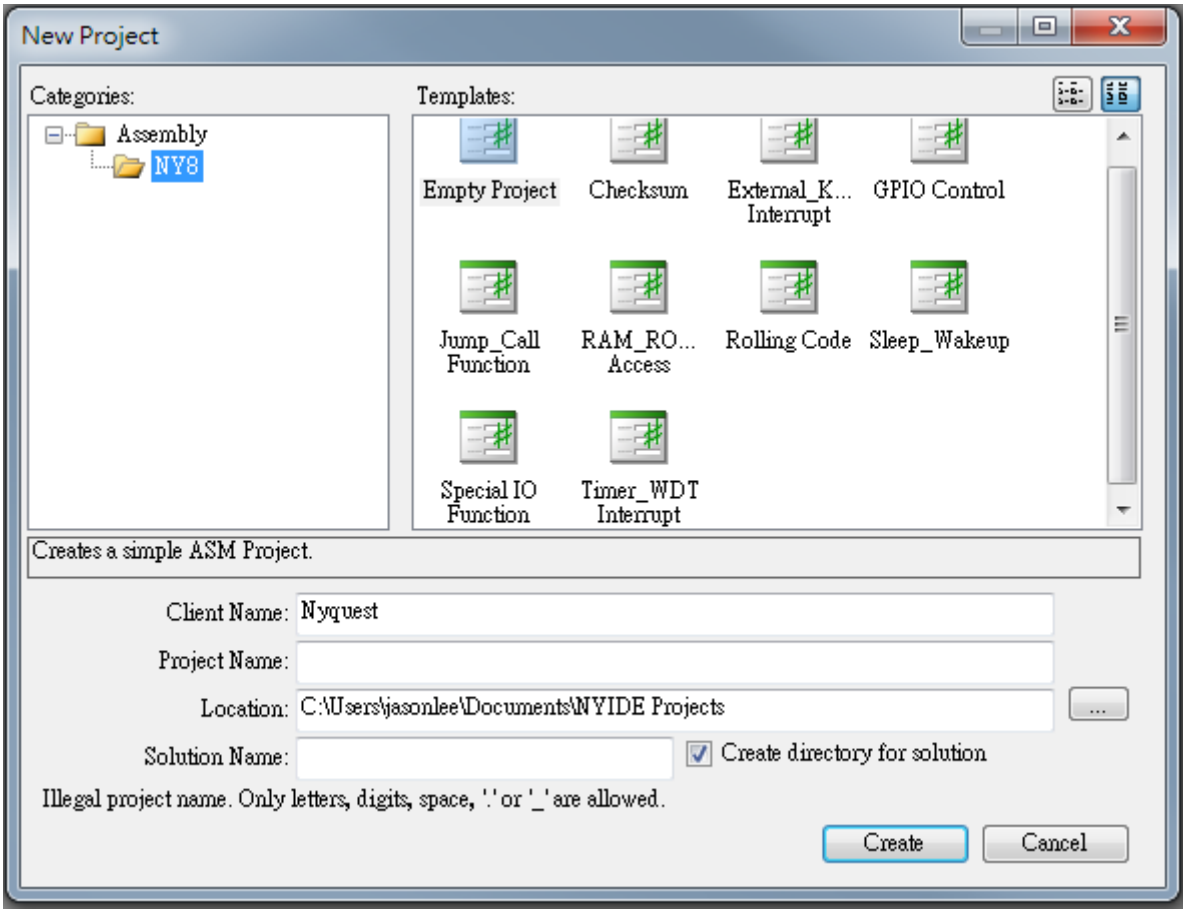
1.2 NY8 Example Code

- Empty Project
- External_Key Change Interrupt
- Timer_WDT Interrupt
- GPIO Control
- Special_IO Function
- Jump_Call Function
- RAM_ROM_REG Access
- Sleep_Wakeup
- Rolling Code
- Checksum

1.3 Installation

Please install the *NYIDE* software first. After *NYIDE* is executed, users can select "Project/Solution" by clicking on "New" of "File". Then, users can select the desired NY8 Example Code. The operating steps are shown below.





Note: When installing NYIDE, user needs to install NYASM.

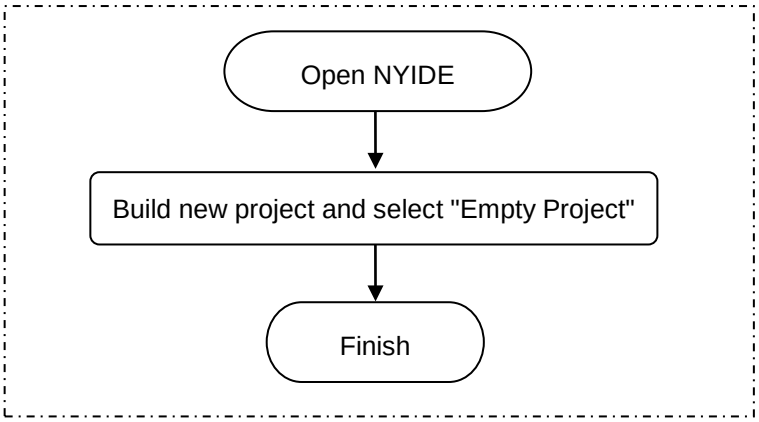
2 Descriptions of NY8 Example Code

2.1 Empty Project

2.1.1 Function Introduction

Function description: Build new project and load Header file "NY8.H".	
Input	Functions
NA	Build new project and load Header file "NY8.H".

2.1.2 Flowchart



2.1.3 Program

```
;----------------------------
; Project:
; File:
; Description:
; Author:
; Version:
; Date:
;----------------------------
; Add   NY8A051A / 053A Series to Header File
#include       NY8.H
;----------------------------
; Define variables
;----------------------------
; Define contants
;----------------------------
; Define vector
ORG      0x000                    ; Power on reset vector address
```

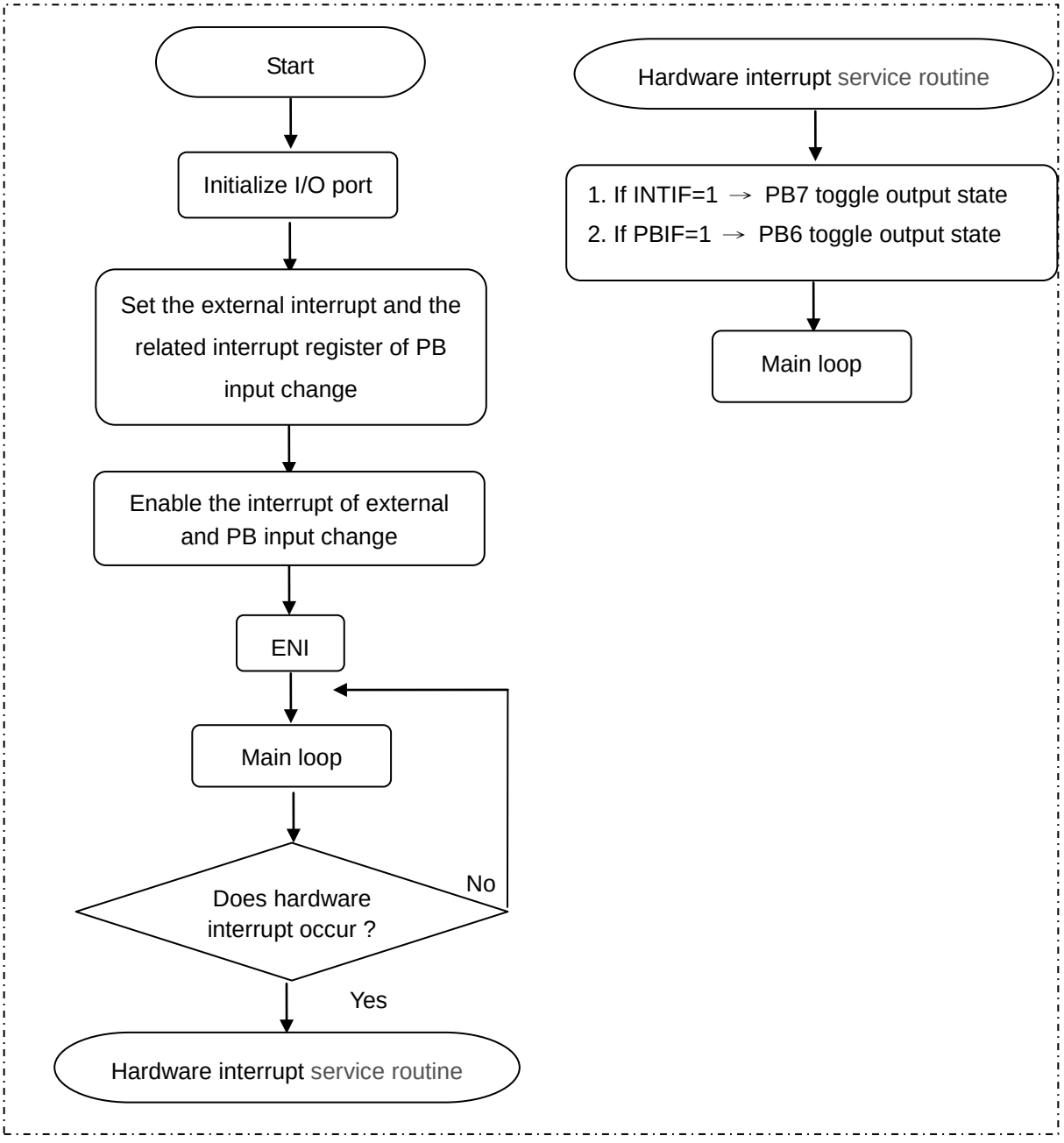
```
    goto    V_Main
ORG     0x008                   ; Hardware interrupt vector address
    goto    V_INT
;--------------------------
; The beginning of program
ORG     0x010                   ; ROM address 0x010
V_Main:

L_MainLoop:                     ; Main loop
    clrwdt                      ; Clear watch-dog timer
    lgoto   L_MainLoop
;--------------------------
; Hardware interrupt service routine
V_INT:                          ; The beginning of hardware interrupts
    retie                       ; Return from interrupts
;--------------------------
; End of program
end
```

2.2 External_Key Change Interrupt

2.2.1 Function Introduction

Function description: Use the external interrupt and PB input change interrupt.	
Input	Functions
INT	The interrupt occurred when pin INT inputted rising edge signal → PB7 toggle output state
PB1	The interrupt occurred when pin PB1 inputted rising/falling edge signal → PB6 toggle output state

2.2.2 Flowchart



2.2.3 Program

```
V_Main:
; Disable all interrupts
    disi
; Enable the pull-high resistors of PB1 and PB0
    movia   ~(C_PB1_PHB | C_PB0_PHB)
    movar   Pr_PB_PH_Ctrl
; Enable PB1 input change wake up
    movia   C_PB1_Wakeup
    movar   Pr_PB_WakeUp_Ctrl
; Set PB1 and PB0 as input pins
    movia   C_PB1_Input | C_PB0_Input
    iost    Pf_PB_Dir_Ctrl
    movia   0x00
    movar   Pr_PB_Data
; Set the related registers of the external interrupt
    movia   C_EXINT_Edge
    t0md
    movia   C_ExtINT_En
    movar   Pr_PWR_Ctrl
; Enable the external interrupt and PB input change interrupt
    movia   C_INT_EXT | C_INT_PBKey
    movar   Pr_INT_Ctrl
    Movia   0x00                        ; Clear all interrupt flags
    Movar   Pr_INT_Flag
    eni
;-------------------------
; Main loop
L_MainLoop:
    clrwdt
    goto    L_MainLoop
;-------------------------
; Hardware interrupt service routine
V_INT:
;-------------------------
; External interrupt service routine
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                  ; Clear external interrupt flag
        movar   Pr_INT_Flag
        goto    L_RET2Main
;----------------------------
; PB input change interrupt service routine
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                ; Clear PB input change interrupt flag
        movar   Pr_INT_Flag

L_RET2Main:
        Retie                               ; Return from the H/W interrupt service routine
```

Note: For the complete program, please refer to the example code of NYIDE.

2.3 Timer_WDT Interrupt

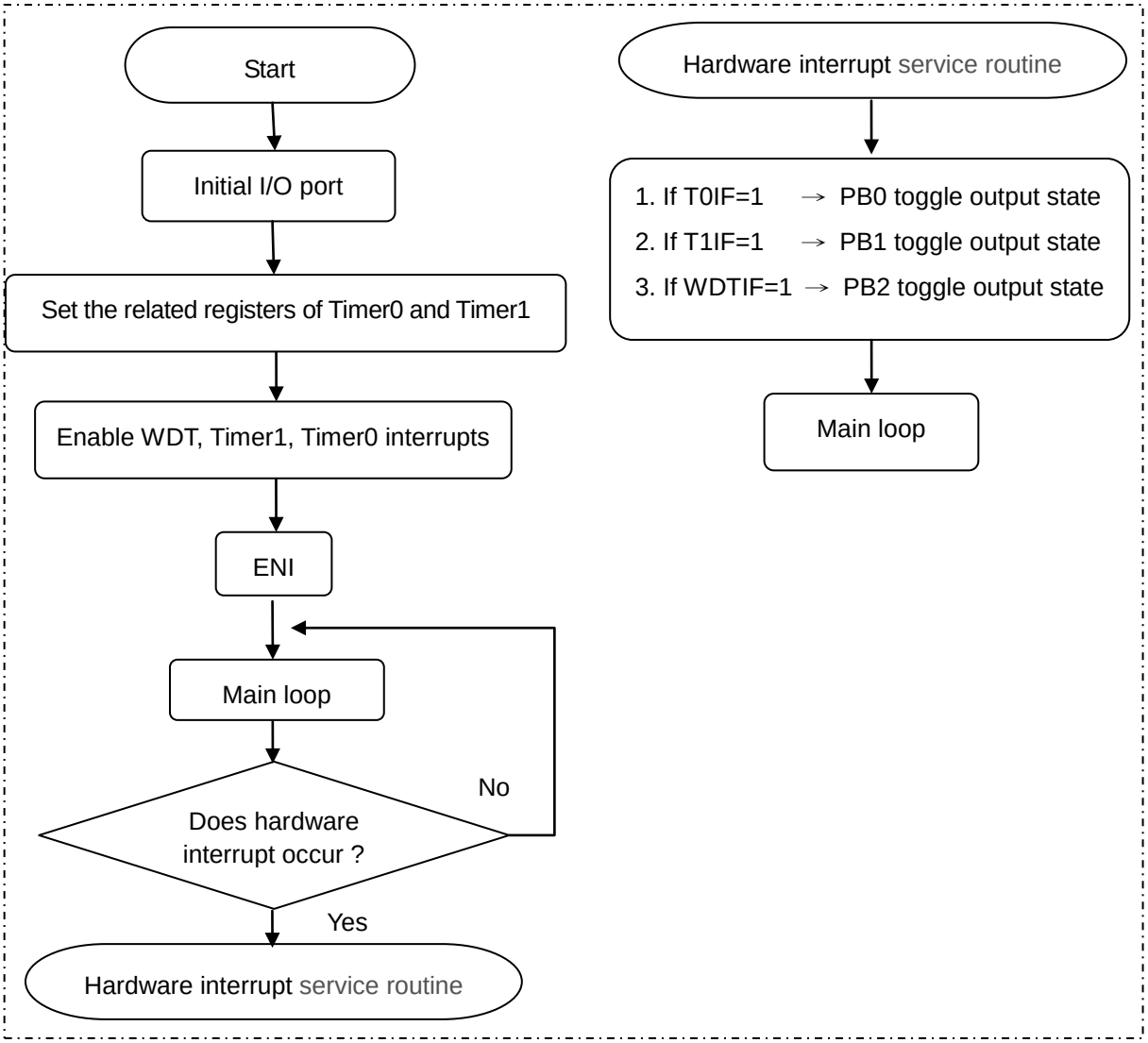
2.3.1 Function Introduction

Function description:

1. F_{INST}= 4MHz/4T (I_HRC) = 1MHz
2. Set the WDT time base as 3.5ms by "*NYIDE* Project Config Block setting "
3. Set Timer0 to be interrupted every 2048 instruction cycle and PB0 toggle output state
4. Set Timer1 to be interrupted every 1024 instruction cycle and PB1 toggle output state
5. Set WDT to be interrupted every 3.5ms and PB2 toggle output state

Input	Functions
NA	1. Timer0 is interrupted every 2048 instruction cycle and PB0 toggle output state
NA	2. Timer1 is interrupted every 1024 instruction cycle and PB1 toggle output state
NA	3. WDT is interrupted every 3.5ms and PB2 toggle output state

2.3.2 Flowchart



2.3.3 Program

```
; Set the related registers of Timer0
    movia     C_TMR0_Dis
    iost      Pf_PWR_Ctrl1                    ; Disable Timer0
    movia     0x00
    movar     Pr_TMR0_Data
    movia     C_PS0_TMR0 | C_PS0_Div8
    t0md
;-------------------------------------------------------------------------
    ;If WDT needs Prescaler0 divider (the Prescaler0 divider is used by selecting Timer0 or WDT)
    ;     movia     (0x00 | C_PS0WDT_Sel)
    ;     t0md
;-------------------------------------------------------------------------
; Set the related registers of Timer0
    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
; Enable the interrupts of WDT, Timer1 and Timer0
    movia     C_INT_WDT | C_INT_TMR1 | C_INT_TMR0
    movar     Pr_INT_Ctrl
; Enable WDT
    movia     C_WDT_En | C_LVR_En
    movar     Pr_PWR_Ctrl
; Enable Timer0
    movia     C_TMR0_En
    iost      Pf_PWR_Ctrl1
    eni                                       ; Enable the all unmasked interrupts
;-------------------------------------------------------------------
; Main loop
L_MainLoop:
    nop
    goto  L_MainLoop


;--------------------------
; Hardware interrupt service routine
```

```
V_INT:
;----------------------------
; Timer1 interrupt service routine
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                 ; Clear Timer1 interrupt flag
    movar   Pr_INT_Flag
;----------------------------
; Timer0 interrupt service routine
L_TIME0_INT:
    btrss   Pr_INT_Flag,C_INT_TMR0_Bit
    goto    L_WDT_INT
    movia   0x00
    movar   Pr_TMR0_Data                ; Reload 0x00 to 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                ; Clear Timer0 interrupt flag
        movar   Pr_INT_Flag
;---------------------------
; WDT interrupt service routine
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               ; Clear WDT interrupt flag
        movar   Pr_INT_Flag
L_RET2Main:
        retie                            ;Return from the H/W interrupt service routine
```

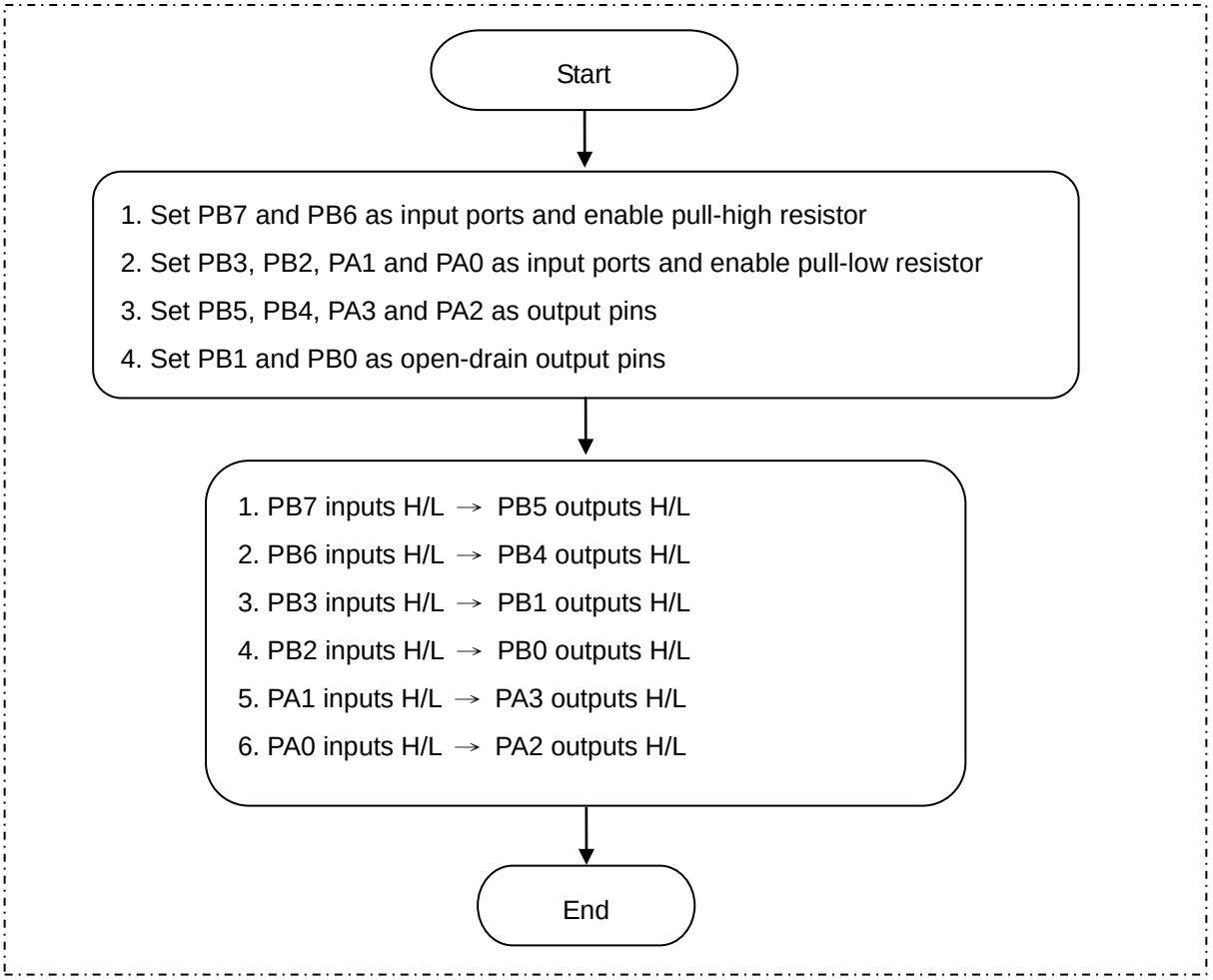
Note: For the complete program, please refer to the example code of NYIDE.

2.4 GPIO Control

2.4.1 Function Introduction

Function description: 1. Set the related registers of GPIO	
Input	Functions
PB7	PB7 inputs H/L → PB5 outputs H/L
PB6	PB6 inputs H/L → PB4 outputs H/L
PB3	PB3 inputs H/L → PB1 outputs H/L
PB2	PB2 inputs H/L → PB0 outputs H/L
PA1	PA1 inputs H/L → PA3 outputs H/L
PA0	PA0 inputs H/L → PA2 outputs H/L

2.4.2 Flowchart



2.4.3 Program

```
; Set PB1 and PB0 as open-drain outputs
    movia       C_PB1_OD | C_PB0_OD
    iost        Pf_PB_OD_Ctrl
; Enable the pull-low resistor of PB3, PB2, PA1 and PA0
    movia       ~( C_PB3_PLB |   C_PB2_PLB |   C_PA1_PLB |   C_PA0_PLB)
    movar       Pr_PAB_PL_Ctrl
; Enable the pull-high resistor of PB7 and PB6
    movia       ~( C_PB7_PHB | C_PB6_PHB)
    movar       Pr_PB_PH_Ctrl
; Set PB7, PB6, PB3 and PB2 as input pins
; Set PB5, PB4, PB1 and PB0 as output pins
    movia       C_PB7_Input | C_PB6_Input | C_PB3_Input | C_PB2_Input
    iost        Pf_PB_Dir_Ctrl
    movia       0x03
    movar       Pr_PB_Data
; Set PA1 and PA0 as input pins
; Set PA3 and PA2 as output pins
    movia       C_PA1_Input | C_PA0_Input
    iost        Pf_PA_Dir_Ctrl
    movia       0x0C
    movar       Pr_PA_Data
```

Note: For the complete program, please refer to the example code of NYIDE.

2.5 Special IO Function (IR carrier, PWM, Buzzer Output)

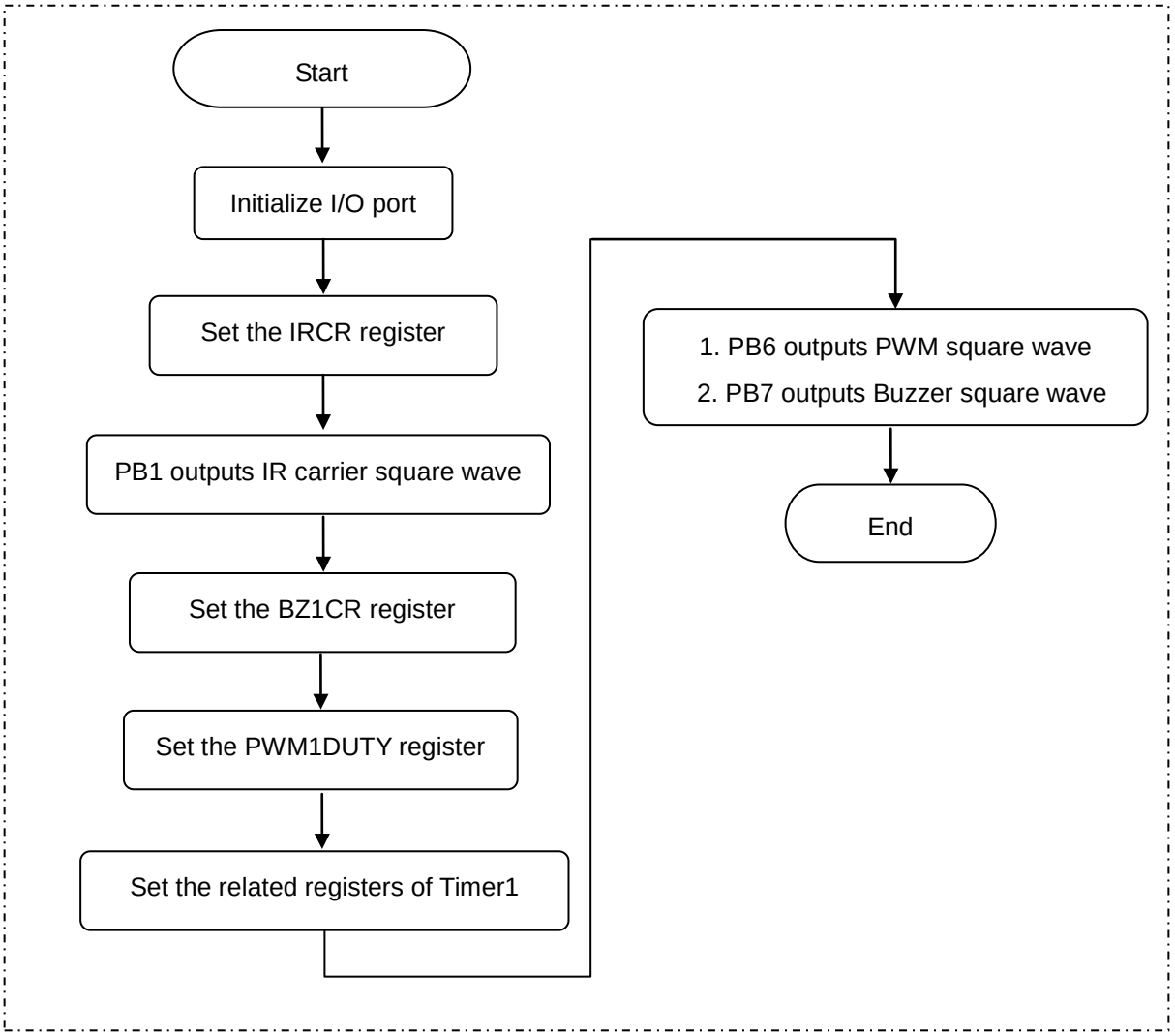
2.5.1 Function Introduction

Function description:

1. The example code uses external crystal 3.58MHz (PB4 / PB5)
2. Output IR carrier square wave of 38KHz by pin PB1
3. Output 1.75KHz of 25% duty cycle of PWM square wave by pin PB6
4. Output Buzzer square wave of 55.94KHz by pin PB7

Input	Functions
NA	1. Output IR carrier square wave of 38KHz by pin PB1
NA	2. Output 1.75KHz of 25% duty cycle of PWM square wave by pin PB6
NA	3. Output Buzzer square wave of 55.94KHz by pin PB7

2.5.2 Flowchart



2.5.3 Program

```
; Set the IR carrier register
    bsr     Pr_PB_Data,1                ; Set PB1 data buffer = 1
    movia   C_IR_ClkSrc_358M | C_IR_En
    sfun    Ps_IR_Ctrl
; Set the Buzzer1 register
    movia   C_BZ1_En | C_BZ1_TMR1B2
    sfun    Ps_BZ1_Ctrl
INIT_TIME1:
    movia   0xFF
    sfun    Ps_TMR1_Data
; Set the PWM1DUTY register
    movia   C_PWM_DUTY_25
    sfun    Ps_PWM1_Duty                ; PWM1 Duty = 64/256 = 25%
; Set the related registers of Timer1
    movia   C_PWM1_En | C_TMR1_Reload | C_TMR1_En
    sfun    Ps_TMR1_Ctrl1
    movia   0x00
    sfun    Ps_TMR1_Ctrl2
```

Note: For the complete program, please refer to the example code of NYIDE.

2.6 Jump_Call Function

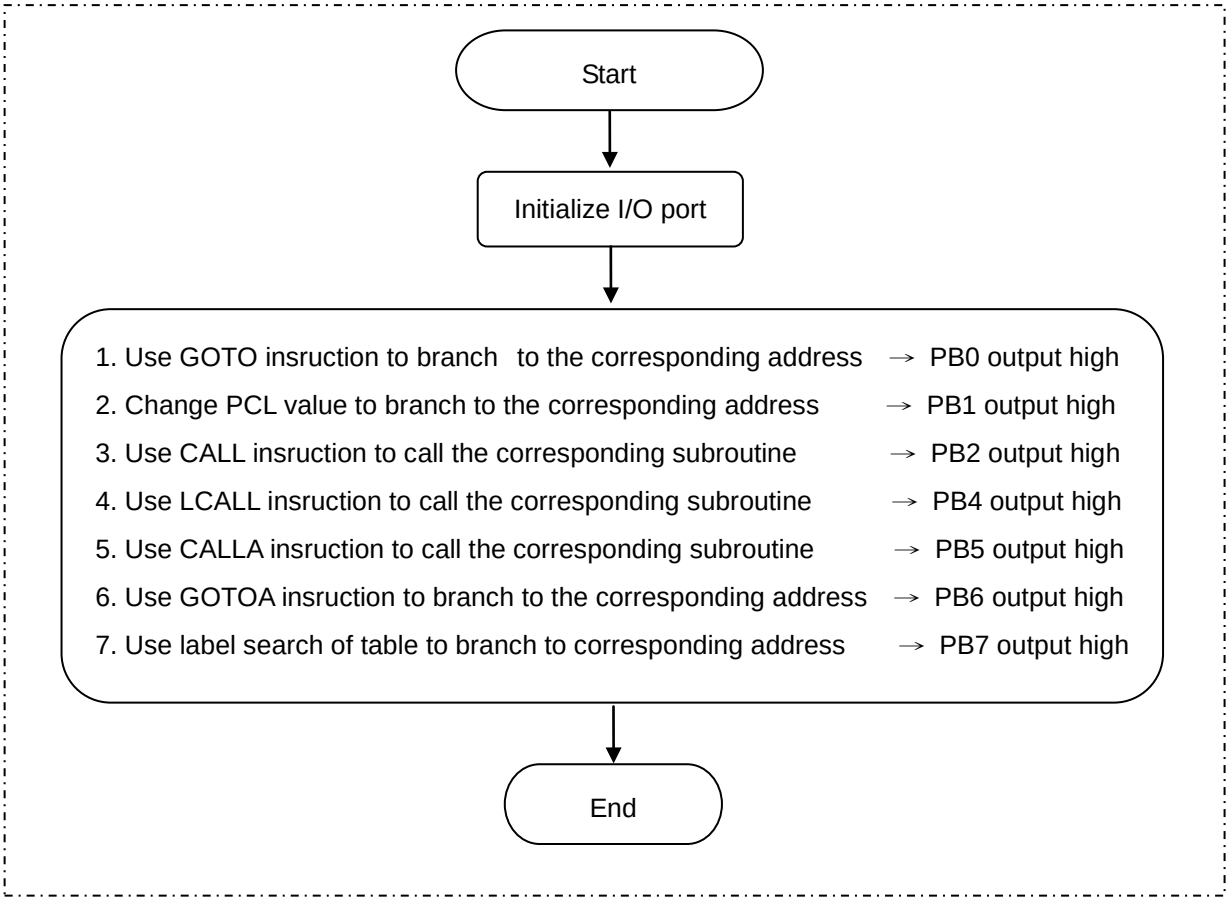
2.6.1 Function Introduction

Function description:

1. Use the Jump insructions (GOTO, GOTOA).
2. Use the Call-Subroutine insructions (CALL, LCALL and CALLA).
3. Change the PCL value to implement the unconditional branch.
4. Use label search of table to implement the unconditional branch.

Input	Functions
NA	Use GOTO insruction to branch to the corresponding address → PB0 output high
NA	Change PCL value to branch to the corresponding address → PB1 output high
NA	Use CALL insruction to call the corresponding subroutine → PB2 output high
NA	Use LCALL insruction to call the corresponding subroutine → PB4 output high
NA	Use CALLA insruction to call the corresponding subroutine → PB5 output high
NA	Use GOTOA insruction to branch to the corresponding address → PB6 output high
NA	Use label search of table to branch to corresponding address → PB7 output high

2.6.2 Flowchart



2.6.3 Program

```
;---------------------------
; Use GOTO insruction to branch   to the corresponding address
    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
;---------------------------
; Change PCL value to branch to the corresponding address
    movia       0x00
    movar       Pr_PCHigh_Data
    movia       0x40
    movar       Pr_PCLow_Data
    lgoto       L_FailLoop

ORG         0x040
    bsr         Pr_PB_Data,1
;---------------------------
; Use CALL insruction to call the corresponding subroutine
    call        F_sub1
;---------------------------
; Use LCALL insruction to call the corresponding subroutine
    lcall       F_sub2
;---------------------------
; Use CALLA insruction to call the corresponding subroutine
    movia       0x02
    sfun        Ps_TbHigh_Addr
    movia       0x20
    calla




;---------------------------
```

```
; Use GOTOA insruction to branch to the corresponding address
    movia     0x03
    sfun      Ps_TbHigh_Addr
    movia     0x10
    gotoa                                   ; Branch to ROM address "0x310"
;---------------------------
; Use label search of table to branch to corresponding address
    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                                   ; Branch to ROM address "0x330"
;---------------------------
; Define ROM table
    ORG       0x350                         ; ROM address "0x350"
T_DataTbl:
    DW        0x0330                        ; The contents of Table
```

Note: For the complete program, please refer to the example code of NYIDE.

2.7 RAM_ROM_REG Access

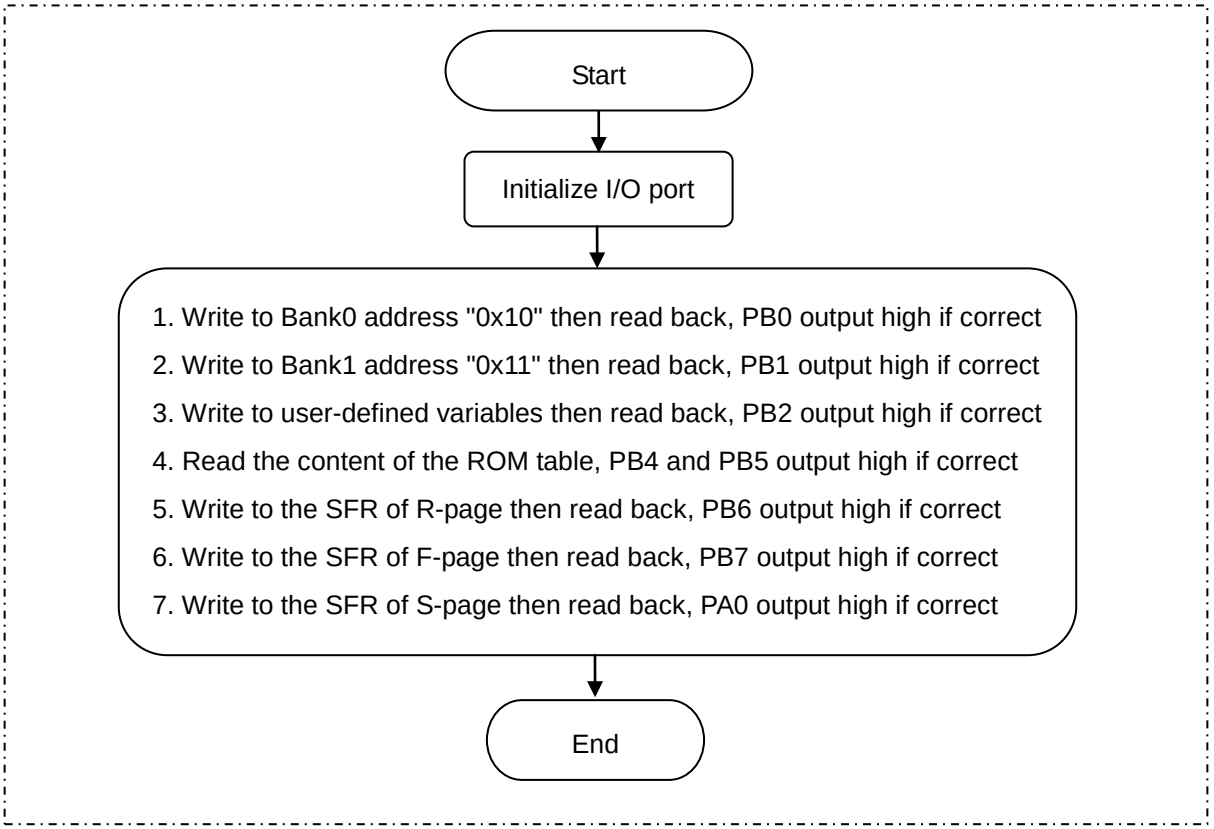
2.7.1 Function Introduction

Function description:

1. General Purpose Register access
2. Read ROM data (Program Memory)
3. Special-Function Register (SFR) access

Input	Functions
NA	Write to Bank0 address "0x10" then read back, PB0 output high if correct
NA	Write to Bank1 address "0x11" then read back, PB1 output high if correct
NA	Write to user-defined variables then read back, PB2 output high if correct
NA	Read the content of the ROM table, PB4 and PB5 output high if correct
NA	Write to the SFR of R-page then read back, PB6 output high if correct
NA	Write to the SFR of F-page then read back, PB7 output high if correct
NA	Write to the SFR of S-page then read back, PA0 output high if correct

2.7.2 Flowchart



2.7.3 Program

```
; Write "0x55" to RAM Bank0 address "0x10"
    movia     C_SFR_Bank0 | 0x10
    movar     Pr_File_Sel
    movia     0x55
    movar     Pr_Indir_Addr
    clra
    clrr      Pr_File_Sel
; Read the content of RAM Bank0 address "0x10"
    movia     C_SFR_Bank0 | 0x10
    movar     Pr_File_Sel
    movr      Pr_Indir_Addr,C_SaveToAcc
; Write "0xAA" to RAM Bank1 address "0x11"
    movia     C_SFR_Bank1 | 0x11
    movar     Pr_File_Sel
    movia     0xAA
    movar     Pr_Indir_Addr
    clra
    clrr      Pr_File_Sel
; Read the content of RAM Bank1 address "0x11"
    movia     C_SFR_Bank1 | 0x11
    movar     Pr_File_Sel
    movr      Pr_Indir_Addr,C_SaveToAcc
; Write "0x5A" to the user-defined variables "R_RAM_0x20"
    movia     0x5A
    movar     R_RAM_0x20
    clra
; Read the content of user-defined variables "R_RAM_0x20"
    movr      R_RAM_0x20,C_SaveToAcc
; Read the content of ROM address "0x150"
    movia     0x01
    sfun      Ps_TbHigh_Addr
    movia     0x50
    tablea
```

```
; Write "0xAA" to R-page special function register "TMR0"
    movia   0xCC
    movar   Pr_TMR0_Data
    clra
; Read the content of R-page special function register "TMR0"
    movr    Pr_TMR0_Data,C_SaveToAcc
; Write "0x08" to F-page special function register "IOSTB"
    movia   0x08
    iost    Pf_PB_Dir_Ctrl
    clra
; Read the content of F-page special function register "IOSTB"
    iostr   Pf_PB_Dir_Ctrl
; Write "0xBB" to S-page special function register "TMR1"
    movia   0xBB
    sfun    Ps_TMR1_Data
    clra
; Read the content of S-page special function register "TMR1"
    sfunr   Ps_TMR1_Data
```

Note: For the complete program, please refer to the example code of NYIDE.

2.8 Sleep_Wakeup (Switch to Halt / Standby mode then Wakeup)

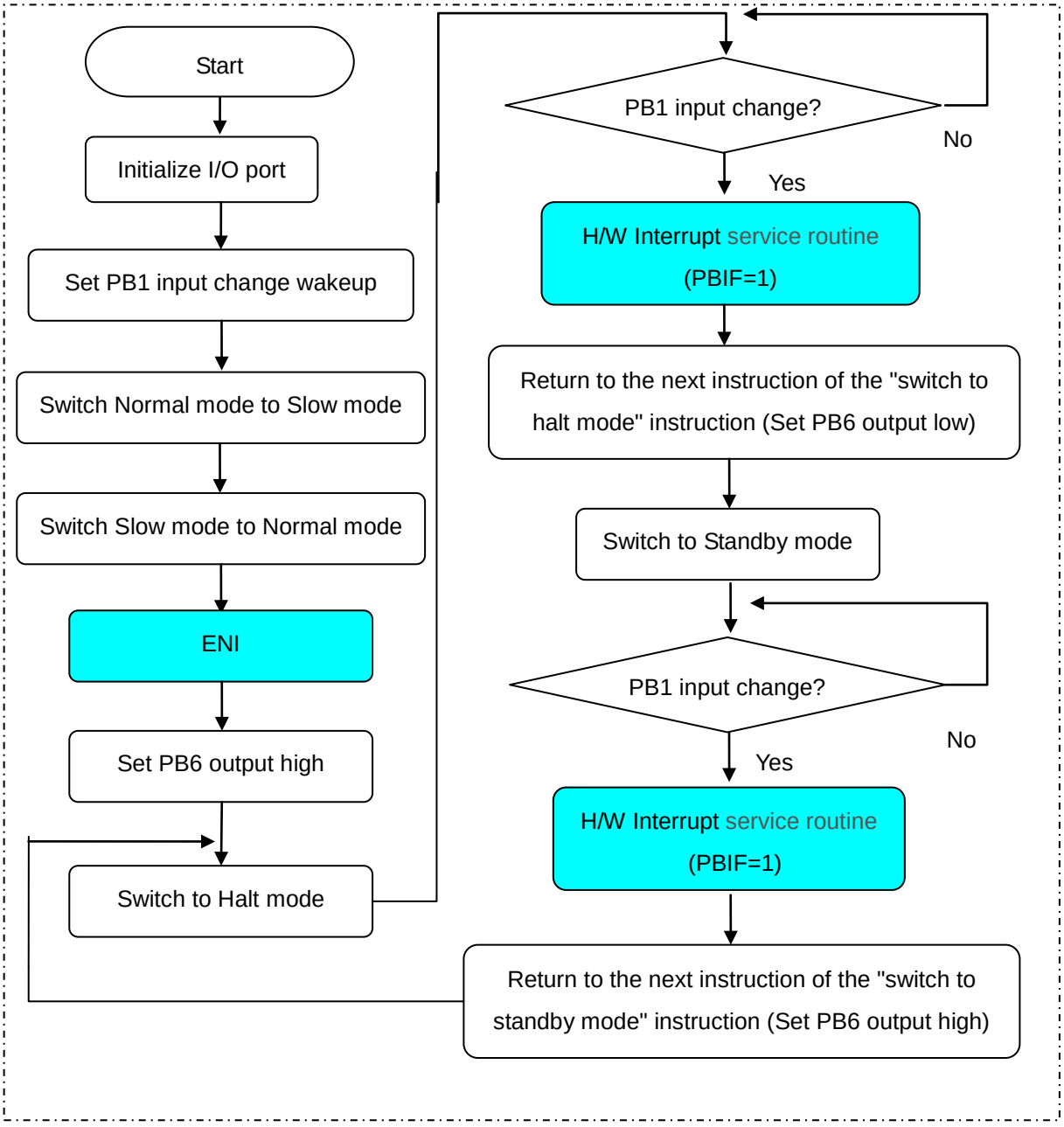
2.8.1 Function Introduction

Function description:

1. Switch Normal mode to Slow mode then return to Normal mode
2. Switch to Halt mode then waking up by PB1 input change
3. Switch to Standby mode then waking up by PB1 input change

Input	Functions
PB1	Set PB6 output high then switch to Halt mode, after waking up setting PB6 output low
PB1	Switch to Standby mode, after waking up setting PB6 output high

2.8.2 Flowchart



2.8.3 Program

```
; Set PB1 input change wakeup
    movia       C_PB1_Wakeup
    movar       Pr_PB_WakeUp_Ctrl
; Enable PortB input change interrupt
    movia       C_INT_PBKey
    movar       Pr_INT_Ctrl
    movia       0x00                ; Clear all interrupt flags
    movar       Pr_INT_Flag
; Switch Normal mode to Slow mode
    movia       C_FLOSC_Sel
    sfun        Ps_SYS_Ctrl
; Switch Slow mode to Normal mode
    movia       C_FHOSC_Sel
    sfun        Ps_SYS_Ctrl
;-------------------------------------------------------------------------------------------
; Select “eni” or “disi” instruction to decide to enter the H/W interrupt service routine or not after waking up from Halt mode or Standby mode
    eni         ; After waking up form Halt mode or Standby mode, entering the H/W interrupt
                ; service routine

    ;disi       ; After waking up form Halt mode or Standby mode, do not entering the H/W interrupt
                ; service routine
    ;-------------------------------------------------------------------------------------
    movr        Pr_PB_Data,C_SaveToReg
    ; Please select desired instruction to enter Halt mode from the following instructions
    ; Instruction 1
    sleep
    ; Instruction 2
    ;movia      C_Halt_Mode | C_FHOSC_Sel   ; Enter Halt mode from Normal mode
    ;sfun       Pr_SYS_Ctrl
    ;-------------------------------------------------------------------------------------
    bcr         Pr_PB_Data,6        ; After waking up from Halt mode, PB6 output low
    movia       ~C_INT_PBKey        ; Clear the interrupt flag of PB input change
    movar       Pr_INT_Flag
; -------------------------------------------------------------------------------------------
    ; Enter Standby mode from Normal mode
    movr        Pr_PB_Data,C_SaveToReg
```

```
    movia     C_Standby_Mode | C_FHOSC_Sel
    sfun      Ps_SYS_Ctrl
    bsr       Pr_PB_Data,6        ; After waking up from Standby mode, PB6 output high
    movia     ~C_INT_PBKey        ; Clear the interrupt flag of PB input change
    movar     Pr_INT_Flag
;----------------------------
; Hardware interrupt service routine
V_INT:
; PB input change Interrupt service routine
L_PBX_INT:
    btrss     INTF,C_INT_PBKey
    goto      L_RET2Main
L_clr_flag_PBX:
    movia     ~C_INT_PBKey        ; Clear the interrupt flag of PB input change
    movar     Pr_INT_Flag
L_RET2Main:
    Retie                         ; Return from the hardware interrupt service routine
```

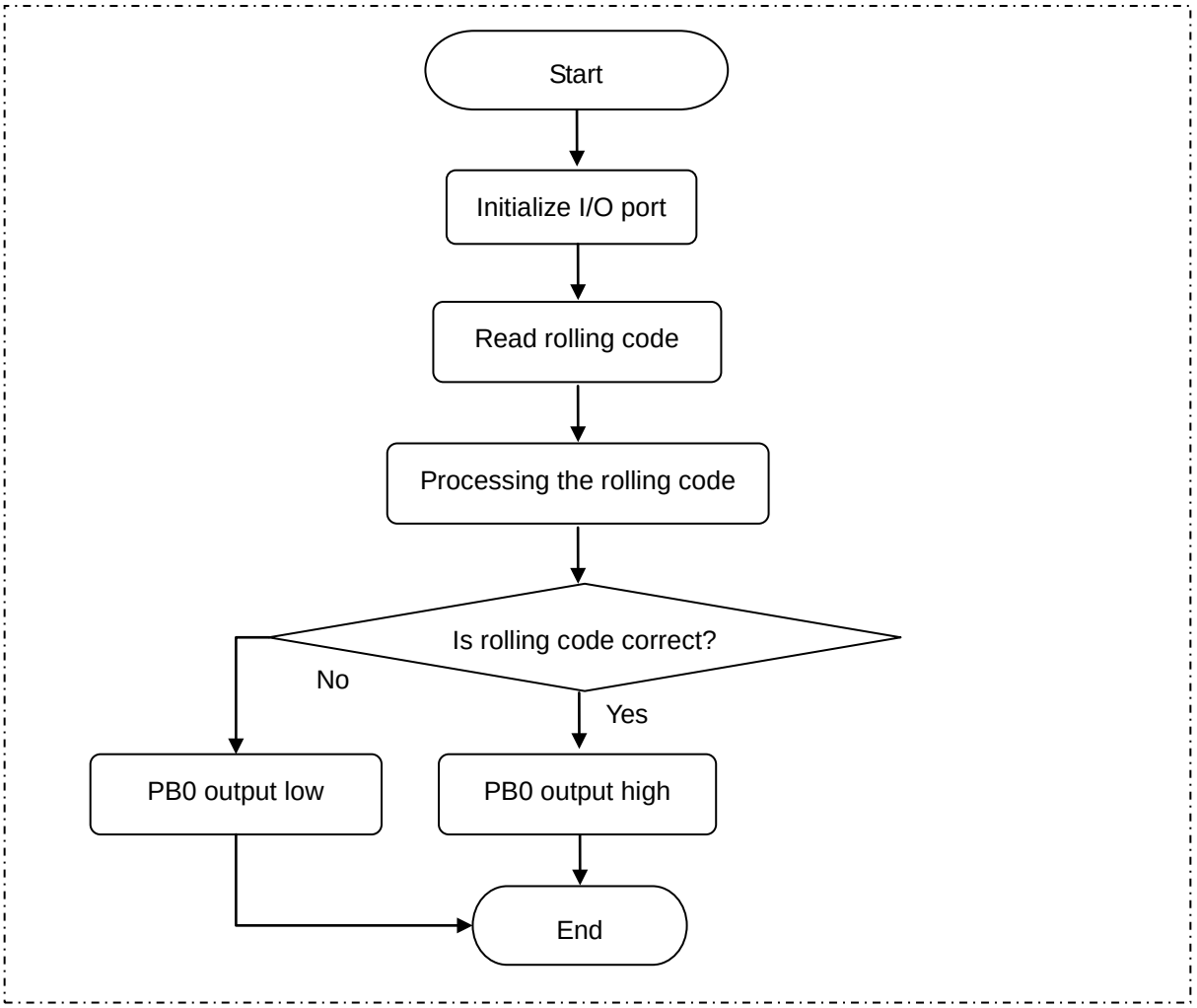
Note: For the complete program, please refer to the example code of NYIDE.

2.9 Rolling Code

2.9.1 Function Introduction

Function description: Read rolling code	
Input	Functions
NA	Read and determine the rolling code, setting PB0 to output high if correct

2.9.2 Flowchart



2.9.3 Program

```
;----------------------------
; Define variables
 R_RollingCode_byte0     EQU     10H          store rolling code value, bit 7 ~ 0
 R_RollingCode_byte1     EQU     11H
 R_RollingCode_byte2     EQU     12H          ; store rolling code value, bit 15 ~ 8
 R_RollingCode_byte3     EQU     13H          ; store rolling code value, bit 19 ~ 16
```

```
;----------------------------
; Define constants
; If user programs the rolling code via Q-Writer is 961109 (decimal) = 0xEAA55
 C_RC_B0                EQU     55H             ; Rolling code default value bit 7 ~ 0
 C_RC_B1                EQU     AAH             ; Rolling code default value bit 15 ~ 8
 C_RC_B2                EQU     0EH             ; Rolling code default value bit 19 ~ 16


; Read the ROM address 0x0E and 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
; Processing the rolling code
    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
; Determine rolling code is correct or not
        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          ; Set PB0 to output high if rolling code is correct
```

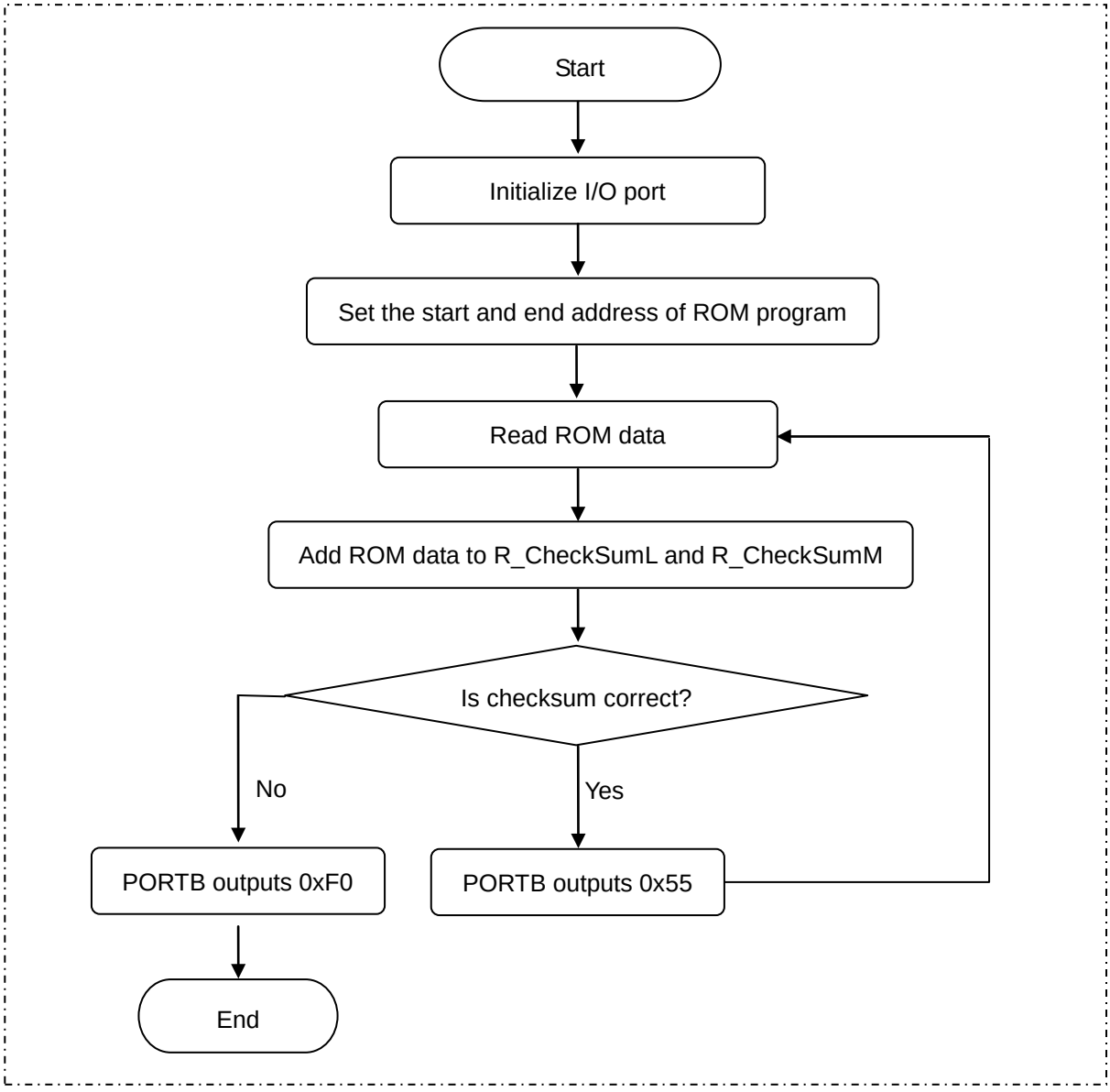
Note: For the complete program, please refer to the example code of NYIDE.

2.10 Checksum

2.10.1 Function Introduction

Function description: ROM checksum calculation.	
Input	Functions
NA	1. The checksum is correct → PORTB outputs 0x55 2. The checksum is error → PORTB outputs 0xF0

2.10.2 Flowchart



2.10.3 Program

```
;----------------------------
; Define variables
R_Tab_IndexL      EQU     0X10        ; Store the index of ROM address (low-byte)
R_Tab_IndexH      EQU     0X11        ; Store the index of ROM address (high-byte)
R_CheckSumL       EQU     0X12        ; Store the calculated checksum (low-byte)
R_CheckSumM       EQU     0X13        ; Store the calculated checksum (high-byte)
;----------------------------
; Define constants
#Define          C_StartAddr     0x0000      ; Define the start address of ROM program

L_ReadROM_Start:
; Initialize variables
    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
; Read ROM data
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

; Determine the checksum is correct or not
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                    ; The checksum is correct
        movar   Pr_PB_Data              ; PORTB outputs 0x55
        movr    Pr_PA_Data,C_SaveToAcc
        xoria   0x01
        movar   Pr_PA_Data
        lgoto   L_ReadROM_Start
; The checksum is error
L_FailLoop:
        movia   0xF0                    ; PORTB outputs 0xF0
        movar   Pr_PB_Data
        clrwdt
        lgoto   L_FailLoop

L_CheckSum_Addr:                        ; The end address of ROM data
;----------------------------
; End of program
        end
```

Note: For the complete program, please refer to the example code of NYIDE.