# 微

# 机

# 原

# 理

# 实

# 验

# 报

# 告

# 学院：电子信息与通信学院

# 姓名：周睿科 赵展

# 学号：U202113957 U202117282

# 实验时间：2023年7月14日

1. **实验名称**

按键实验

1. **实验任务**

（基础部分）

* 1. 编写程序检测按键，并将检测结果通过数码管反应
  2. 用可重定位汇编语言实现
  3. 应考虑反应按键触发的次数

（拓展部分）

1. 考虑实现单击、双击或长按、短按的检测
2. **实验条件**
   1. MPLAB X IDE v6.10（代码编写与仿真）
   2. Xc8 编译器
   3. PIC16F18854 单片机及一些拓展模块
   4. 面包板、0.28 英寸极性共阴四位数码管、十按键电路面板、电阻等元器件
3. **实验设计方案——硬件**
   1. **电路设计：**

电路连接采用PORTA0~6位连接A~G，PORTC0~3位对应位码控制数码管1~4的亮灭，与此同时，PORTB0~3位连接控制检测按键的K0~K3

1. **实验设计方案——软件**
   1. **扫描逻辑**
      1. 通过时钟中断，在每次中断程序中进行一轮轮询实现全扫描
      2. Kn 均为 1，检查 K3 K2 K1 K0：若命中，则进入 btn 7 8 9 a 处理程序
      3. K3 置为 0，检查 K2 K1 K0，若命中，则进入 btn 4 5 6 处理程序
      4. K2 置为 0，检查 K1 K0，若命中，则进入 btn 2 3 处理程序
      5. K1 置为 0，检查 K0，若命中，则进入 btn 1 处理程序
      6. Kn 置为 1，重新配置端口全输入
      7. 逻辑映射：

* 一个 query 对应一行 ，一个 index 对应一列
* Query 内部进行查表，Table\_n [index] 可以唯一确定一个按钮的输出
* 每次扫描检查时，index += 1
  1. **事件处理思路（两种）**

**采用状态机+计数变量相结合**

1. 状态图

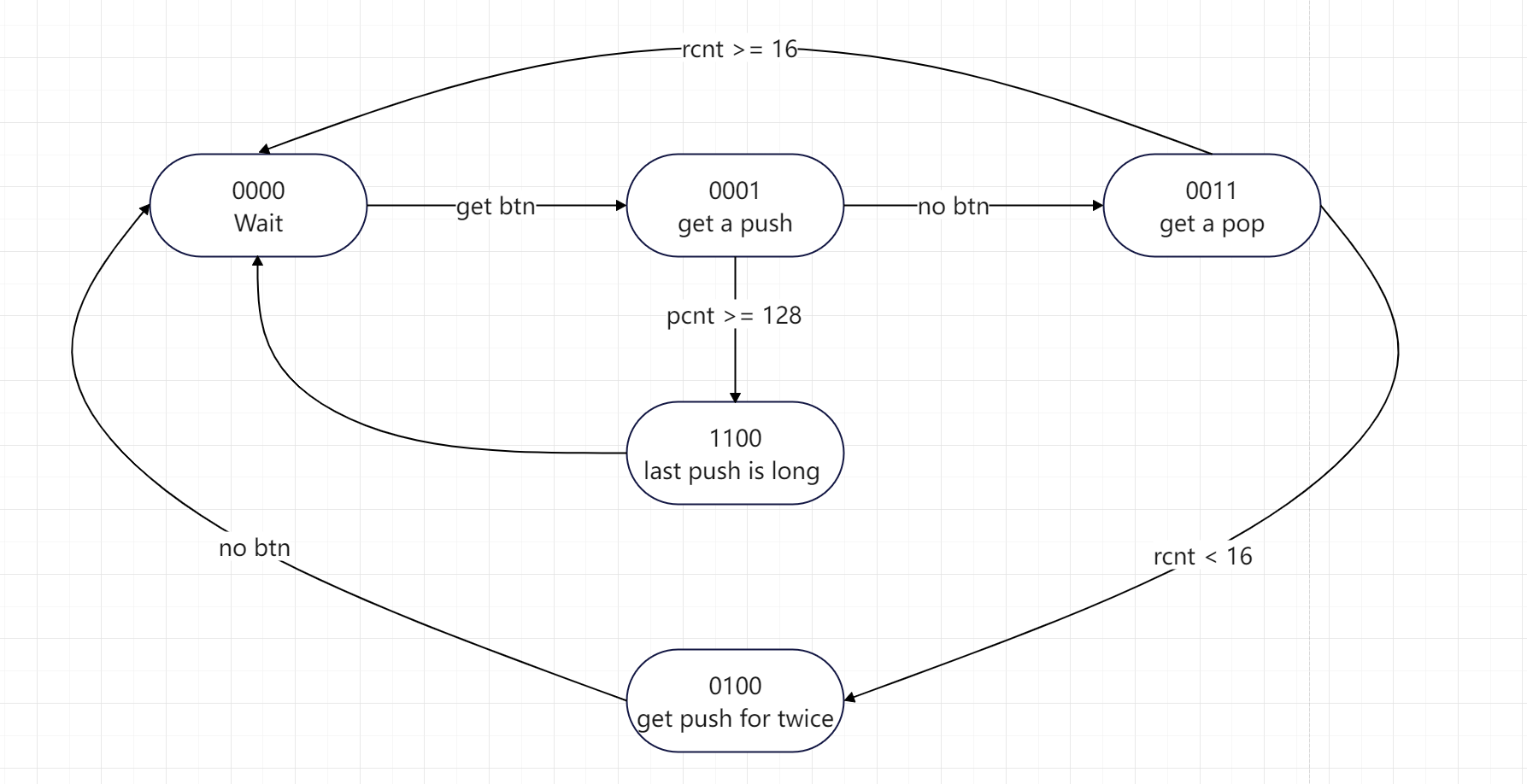


图 1 主程序流程图

1. 模式转换

* 需要在轮询中翻转 mode
* Mode的作用仅仅是：
  + 控制在未检测到按钮时，程序的行为
  + 直接返回则可以保存状态，相当于no\_btn时不显示
  + 由于 no\_btn 中会对输出变量清零，因此若进入 label exit 统一进行输出，就可以实现实时显示

1. 单击逻辑

* 总体的思路是采用 状态机 + 计数变量结合的方式
* 若检查到存在按钮被按下：
  + 进入 get\_btn 处理程序
  + set Flag[0] = 1，代表成功按下一次
* 若检查到不存在按钮被按下：
  + 进入 not\_btn 处理程序
  + 若 Flag[0] = 1，代表此前有按钮被按下
  + clear Flag[0] - Set Flag[1] = 1
  + 此时状态代表按钮成功弹起

1. 长按逻辑

* 如何判定长按
  + 按一次，一直按下 -> 长按
  + 双击后，一直按下 -> 非长按
  + 按一次，不松开，切换按键 -> 长按，并显示切换后的按键
* 若检查到存在按钮被按下
  + Pcnt += 1
  + Pcnt[7] == 1，set long[0] = 1
  + long 在进入 not\_btn 时才有可能被清零，因此一直长按，该变量值会保持不变 （详见 not\_btn）
* long 变量：
  + 在 output 中控制 posCode
  + 同时也是状态机的一环
  + 通过test bit 控制状态转移

1. 双击逻辑

* 如何判定双击
  + 短按 + 短按 -> 双击
  + 短按 + 长按 -> 双击
  + 长按 + 短按 -> 非双击
  + B1 短按 + B2 短按 -> 非双击
* 状态转换条件：
  + 上一次存在按钮被按下
  + 时间间隔在 160ms 内
  + 上一次不是长按
  + 上一次不是另一个按钮

**采用pcnt和rcnt两个变量**

1. 变量设置

* Count：记录一次轮询之后的按键是否被按下
* Pcnt：对按键被按下时进行计数
* Rcnt：对按键弹开时进行计数
* Poscode：位码控制数码管的亮灭
* Segcode：段码控制数码管显示的数字
* Index：对按键的列数进行记录方便查表
* Cur：表示当前PORTB的值

1. 流程图

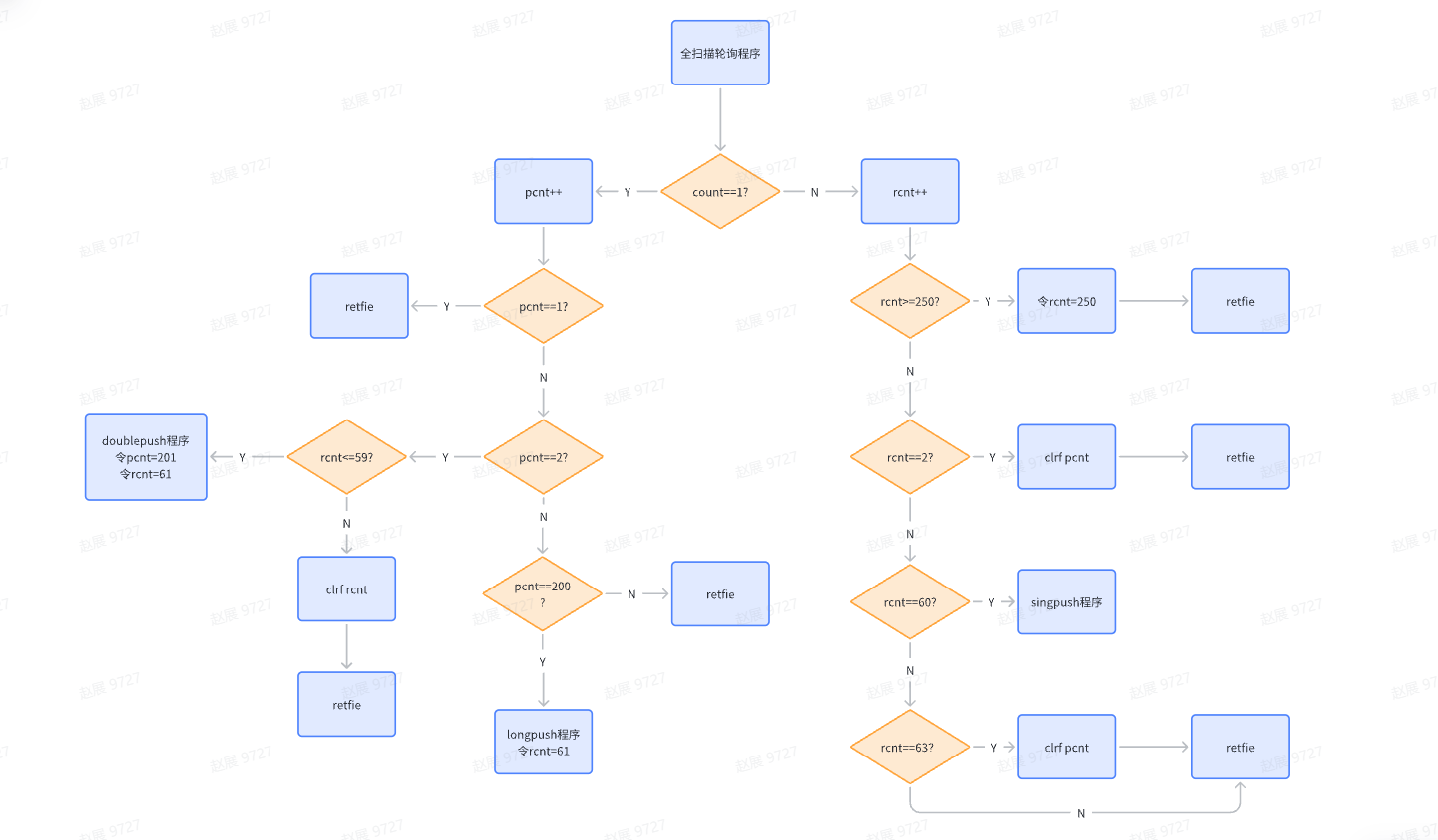


图 2 ISR程序流程图

如图表示的是进入时钟中断时，进行轮询全扫描之后的逻辑处理，主要是对pcnt以及rcnt值的处理和判断

1. **实验源代码及注释**
   1. **思路一：**
   2. **思路二：**

#include <xc.inc>

psect init, class=CODE, delta=2

psect end\_init, class=CODE, delta=2

psect powerup, class=CODE, delta=2

psect cinit,class=CODE,delta=2

psect functab,class=ENTRY,delta=2

psect idloc,class=IDLOC,delta=2,noexec

psect eeprom\_data,class=EEDATA,delta=2,space=3,noexec

psect intentry, class=CODE, delta=2

psect reset\_vec, class=CODE, delta=2

global \_main, reset\_vec, start\_initialization

psect config, class=CONFIG, delta=2

dw 0xDFEC

dw 0xF7FF

dw 0xFFBF

dw 0xEFFE

dw 0xFFFF

psect reset\_vec

reset\_vec:

ljmp \_main

psect cinit

start\_initialization:

psect CommonVar, class=COMMON, space=1, delta=1

charcase: ds 1h

GLOBAL pcnt,rcnt,posCode,segCode,index,cur,count

pcnt:ds 1h

rcnt:ds 1h

posCode:ds 1h

segCode:ds 1h

index:ds 1h

cur:ds 1h

count:ds 1h

psect intentry

intentry:

movlw 0

movwf count

; Interrupt service routine

; clear overflow flag

banksel PIR0

bcf PIR0, 5

; reset

banksel TRISB

movlw 11111111B

movwf TRISB

banksel WPUB

movlw 0xff

movwf WPUB

movlw 0x0

movwf index

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PORTB: 1111 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

banksel PORTB

movf PORTB, w

movwf cur

; query table

btfss cur, 3

goto query\_3

btfss cur, 2

goto query\_2

btfss cur, 1

goto query\_1

btfss cur, 0

goto query\_0

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PORTB: 0111 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

banksel TRISB

bcf TRISB, 3

banksel PORTB

bcf PORTB, 3

; PORTB to cur, index += 1

movf PORTB, w

movwf cur

incf index

; query table

btfss cur, 2

goto query\_2

btfss cur, 1

goto query\_1

btfss cur, 0

goto query\_0

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PORTB: 0011 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

banksel TRISB

bcf TRISB, 2

banksel PORTB

bcf PORTB, 2

; PORTB to cur, index += 1

movf PORTB, w

movwf cur

incf index

; query table

btfss cur, 1

goto query\_1

btfss cur, 0

goto query\_0

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PORTB: 0001 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

banksel TRISB

bcf TRISB, 1

banksel PORTB

bcf PORTB, 1

; PORTB to cur, index += 1

movf PORTB, w

movwf cur

incf index

; query table

btfss cur, 0

goto query\_0

begin:

;check if there is a btn was pushed

check:

;if count==1,then STATUS[2]==1,go to Pbtn

;else go to Rbtn

movlw 1

xorwf count,w

btfss STATUS,2

goto Rbtn

goto Pbtn

;release the btn

Rbtn:

incf rcnt

;if rcnt>=250,then STATUS[0]==1,let rcnt = 250

movlw 250

subwf rcnt,w

btfsc STATUS,0

goto resetrcnt

;if rcnt>=2,then STATUS[2]==1,clrf pcnt

movlw 2

subwf rcnt,w

btfsc STATUS,0

clrf pcnt

;if rcnt==60,then STATUS[2]==1,go to singlepush

movlw 60

xorwf rcnt,w

btfsc STATUS,2

goto singlepush

movlw 63

xorwf rcnt,w

btfsc STATUS,2

clrf pcnt

retfie

;push the btn

Pbtn:

;if pcnt==1,then STATUS[2]==1,retfie

incf pcnt

movlw 1

xorwf pcnt,w

btfsc STATUS,2

retfie

;if pcnt==2,then STATUS[0]==1,go to checkdoublepush

movlw 2

xorwf pcnt,w

btfsc STATUS,2

goto checkdoublepush

continue:

;if pcnt==200 ,then STATUS[2]==1,go to longpush

;else retfie

movlw 200

xorwf pcnt,w

btfss STATUS,2

retfie

goto longpush

checkdoublepush:

;if rcnt<=59 ,then STATUS[0]==0,go to doublepush,

;else goto continue

movlw 59

subwf rcnt,w

btfss STATUS,0

goto doublepush

clrf rcnt

retfie

resetrcnt:

movlw 250

movwf rcnt

retfie

singlepush:

; position

banksel PORTC

movlw 11110111B

movwf PORTC

banksel PORTA

movf segCode, w

movwf PORTA

retfie

doublepush:

movlw 201

movwf pcnt

movlw 61

movwf rcnt

; position

banksel PORTC

movlw 11111011B

movwf PORTC

banksel PORTA

movf segCode, w

movwf PORTA

retfie

longpush:

movlw 61

movwf rcnt

decf pcnt

; position

banksel PORTC

movlw 11111101B

movwf PORTC

banksel PORTA

movf segCode, w

movwf PORTA

retfie

query\_3:

incf count

movf index, w

call Table\_3

movwf segCode

goto begin

query\_2:

incf count

movf index, w

call Table\_2

movwf segCode

goto begin

query\_1:

incf count

movf index, w

call Table\_1

movwf segCode

goto begin

query\_0:

incf count

movf index, w

call Table\_0

movwf segCode

goto begin

Table\_3:

addwf PCL, f

retlw 11110111B ;10

Table\_2:

addwf PCL, f

retlw 0xef ;9

retlw 0xfd ;6

Table\_1:

addwf PCL, f

retlw 0xff ;8

retlw 11101101B ;5

retlw 11001111B ;3

Table\_0:

addwf PCL, f

retlw 0x27 ;7

retlw 11100110B ;4

retlw 11011011B ;2

retlw 00000110B ;1

psect main,class=CODE,delta=2 ; PIC10/12/16

global \_main

\_main:

;TODO

;PORTA config

banksel PORTA

clrf PORTA

banksel LATA

clrf LATA

banksel ANSELA

clrf ANSELA

banksel TRISA

movlw 00000000B

movwf TRISA

;PORTC config

banksel PORTC

clrf PORTC

banksel LATC

clrf LATC

banksel ANSELC

clrf ANSELC

banksel TRISC

movlw 00000000B

movwf TRISC

;PORTB config

banksel PORTB

clrf PORTB

banksel LATB

clrf LATB

banksel ANSELB

clrf ANSELB

banksel TRISB

movlw 11111111B

movwf TRISB

banksel WPUB

movlw 0xff

movwf WPUB

;TIMER0 config

banksel T0CON0

movlw 10000000B

movwf T0CON0

banksel T0CON1

movlw 01010001B

movwf T0CON1

banksel PIR0

bcf PIR0, 5

banksel TMR0H

movlw 0xfa

movwf TMR0H

; INTERRUPT config

banksel PIR0

bcf PIR0, 5

; GIE, PIE enable

banksel INTCON

bsf INTCON, 7

bsf INTCON, 6

; TMR0IE enable

banksel PIE0

bsf PIE0, 5

movlw 0

movwf pcnt

movlw 250

movwf rcnt

; init LED

; test

loop:

nop

goto loop

;END

end reset\_vec