;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*

; This file is a basic code template for assembly code generation \*

; on the PIC16F84A. This file contains the basic code \*

; building blocks to build upon. \*

; \*

; Refer to the MPASM User's Guide for additional information on \*

; features of the assembler (Document DS33014). \*

; \*

; Refer to the respective PIC data sheet for additional \*

; information on the instruction set. \*

; \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*

; Filename: xxx.asm \*

; Date: \*

; File Version: \*

; \*

; Author: Tyler Youk \*

; Company: \*

; \*

; \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*

; Files required: P16F84A.INC \*

; \*

; \*

; \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*

; Notes: \*

; \*

; \*

; \*

; \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

list p=16F84A ; list directive to define processor

#include <p16F84A.inc> ; processor specific variable definitions

\_\_CONFIG \_CP\_OFF & \_WDT\_OFF & \_PWRTE\_ON & \_RC\_OSC

; '\_\_CONFIG' directive is used to embed configuration data within .asm file.

; The lables following the directive are located in the respective .inc file.

; See respective data sheet for additional information on configuration word.

;\*\*\*\*\* VARIABLE DEFINITIONS

w\_temp EQU 0x0C ; variable used for context saving

status\_temp EQU 0x0D ; variable used for context saving

DVAR EQU 0x0F

DVAR2 EQU 0x10

SOK equ B'00000000'

s1 equ B'00000001'

s2 equ B'00000010'

s3 equ B'00000100'

s4 equ B'00001000'

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ORG 0x000 ; processor reset vector

goto Main ; go to beginning of program

ORG 0x004 ; interrupt vector location

movwf w\_temp ; save off current W register contents

movf STATUS,w ; move status register into W register

movwf status\_temp ; save off contents of STATUS register

; isr code can go here or be located as a call subroutine elsewhere

movf status\_temp,w ; retrieve copy of STATUS register

movwf STATUS ; restore pre-isr STATUS register contents

swapf w\_temp,f

swapf w\_temp,w ; restore pre-isr W register contents

retfie ; return from interrupt

; program code goes here

Main

bsf STATUS, RP0 ;go to bank 1

movlw B'00000000' ;configure port b

movwf TRISB

movlw B'01111' ;configure port a

movwf TRISA

bcf STATUS,RP0 ;back to bank 0

clrf PORTB

clrf PORTA

movlw s1 ;otherwise rotate through lights and next state

movwf PORTB

goto states2

states1

call delay ;first call delay

movf PORTA,w ;then check if guess was given during delay

xorlw b'00000000' ;check if empty guess. if empty returns 1 Z=1

btfss STATUS,Z

goto checkGuess ;if guess was given then check it

movlw s1 ;otherwise rotate through lights and next state

movwf PORTB

goto states2

states2

call delay ;first call delay

movf PORTA,w ;then check if guess was given during delay

xorlw b'00000000' ;check if empty guess. if empty returns 1 Z=1

btfss STATUS,Z

goto checkGuess ;if guess was given then check it

movlw s2 ;otherwise rotate through lights and next state

movwf PORTB

goto states3

states3

call delay ;first call delay

movf PORTA,w ;then check if guess was given during delay

xorlw b'00000000' ;check if empty guess. if empty returns 1 Z=1

btfss STATUS,Z

goto checkGuess ;if guess was given then check it

movlw s3 ;otherwise rotate through lights and next state

movwf PORTB

goto states4

states4

call delay ;first call delay

movf PORTA,w ;then check if guess was given during delay

xorlw b'00000000' ;check if empty guess. if empty returns 1 Z=1

btfss STATUS,Z

goto checkGuess ;if guess was given then check it

movlw s4 ;otherwise rotate through lights and next state

movwf PORTB

goto states1

checkGuess

movfw PORTB

xorwf PORTA,W ;if same then Z=1

btfsc STATUS,Z ; (guess = answer) then execute code following macro

goto stateSok ;

goto stateErr

stateSok

movlw SOK

movwf PORTB

goto states1

stateErr

movlw b'10000000'

movwf PORTB

goto states1

delay: ; create a delay of about 1 second

MOVLW d'128'

MOVWF DVAR2 ; initialize outer loop counter to 128

d1: clrf DVAR ; initialize inner loop counter to 256

d2: decfsz DVAR,F ; if (--ictr != 0) loop to d2

goto d2

decfsz DVAR2,F ; if (--octr != 0) loop to d1

goto d1

return

END ; directive 'end of program'