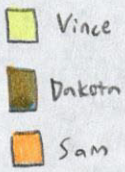


```
; @file: project1.asm
; @authors: Vincent Allen, Sam Itschner, Dakota Ewigman
; @date: 3/31/2014
;
; @registers:
; R0 - Bits 0 through 4 track previous button states each iteration.
;    - Bit 5 tracks state of Vince's button (toggles functionality)
;    - Bit 6 determines which set of LEDs will change with input.
;    - Bit 7 tracks whether or not the 4 bit value R3 turned over.
; R3 - Count stored here (4 bit)
; R1 - Least significant counter for delay
; R2 - Most significant counter for delay
; R4 - Oscillator counter
; R5,R6,R7 - Delay for Sam's button.

;===============================================================================
;                                                                      MAIN LOOP
;===============================================================================
                ORG 0

                MOV 0xA4, #0    ; Set Port 2 to bi-directional
                MOV 0x91, #0    ; Set Port 1 to bi-directional
                MOV 0x84, #0    ; Set Port 0 to bi-directional

                MOV R0, #0      ; R0.7: no turn over, R0.6: use corner LEDs, R0.5:
                start with 1's comp, R0.0-R0.4: not pressed
                MOV R1, #0
                MOV R2, #0
                MOV R3, #0      ; Count set to 0
                MOV R4, #0

                CLR P1.7        ; Default alarm off

MainLoop:       LCALL CheckIncBtn
                LCALL CheckDecBtn
                LCALL CheckVincesBtn
                LCALL CheckSamsBtn
                LCALL CheckDakotasBtn
                LCALL PostProcessing
                SJMP MainLoop

;===============================================================================
;                                                      POST, and BTN CHECKS
;===============================================================================

;-------------------------------------------
; @name: CheckIncBtn
; @brief: If button is being held down, set the
;         state to 1. Else check if was set to 1
;         last iteration. If so, it was just
;         released and we need to increment,
;         update LEDs, the duration of the alarm
```

```
;         and set the state to 0 (for this btn).
;-----------------------------------------------
CheckIncBtn:        MOV A, R0
                    JB  P0.1, IncNotPressed     ; If pressed:
                        SETB ACC.0              ;     Change state to pressed
                        MOV R0, A
                        SJMP CheckIncBtnRET
IncNotPressed:      NOP                         ; Delay - consistent alarm frequency
                    NOP
                    JNB ACC.0, CheckIncBtnRET   ; Else, if pressed last check:
                        CLR ACC.0               ;     Change state to released
                        MOV R0, A
                        LCALL IncMethod         ;     Increment counter
                        LCALL CheckForTurnOver  ;     If turned over, R0.7 set, else clr
                        LCALL SetLEDs           ;     Update LEDs
                        LCALL SetAlarm          ;     If turned over, set R1, R2, R4 -
                        sound alarm
CheckIncBtnRET:     RET

;-----------------------------------------------
; @name: CheckDecBtn
;-----------------------------------------------
CheckDecBtn:        MOV A, R0
                    JB  P0.3, DecNotPressed
                        SETB ACC.1
                        MOV R0, A
                        SJMP CheckDecBtnRET
DecNotPressed:      NOP
                    NOP
                    JNB ACC.1, CheckDecBtnRET
                        CLR ACC.1
                        MOV R0, A
                        LCALL DecMethod
                        LCALL CheckForTurnOver
                        LCALL SetLEDs
                        LCALL SetAlarm
CheckDecBtnRET:     RET

;-----------------------------------------------
; @name: CheckVincesBtn
;-----------------------------------------------
CheckVincesBtn:     MOV A, R0
                    JB  P0.2, VincesNotPressed
                        SETB ACC.2
                        MOV R0, A
                        SJMP CheckVincesBtnRET
VincesNotPressed:   NOP
                    NOP
                    JNB ACC.2, CheckVincesBtnRET
                        CLR ACC.2
                        MOV R0, A
                        LCALL VincesMethod
```

```
                        LCALL CheckForTurnOver
                        LCALL SetLEDs
                        LCALL SetAlarm
CheckVincesBtnRET:  RET

;-----------------------------------------------
; @name: CheckSamsBtn
;-----------------------------------------------
CheckSamsBtn:       MOV A, R0
                    JB  P1.4, SamsNotPressed
                        SETB ACC.3
                        MOV R0, A
                        SJMP CheckSamsBtnRET
SamsNotPressed:     NOP
                    NOP
                    JNB ACC.3, CheckSamsBtnRET
                        CLR ACC.3
                        MOV R0, A
                        LCALL SamsMethod
                        LCALL CheckForTurnOver
                        LCALL SetLEDs
                        LCALL SetAlarm
CheckSamsBtnRET:    RET

;-----------------------------------------------
; @name: CheckDakotasBtn
;-----------------------------------------------
CheckDakotasBtn:    MOV A, R0
                    JB  P0.0, DakotasNotPressed
                        SETB ACC.4
                        MOV R0, A
                        SJMP CheckDakotasBtnRET
DakotasNotPressed:  NOP
                    NOP
                    JNB ACC.4, CheckDakotasBtnRET
                        CLR ACC.4
                        MOV R0, A
                        LCALL DakotasMethod
                        LCALL CheckForTurnOver
                        LCALL SetLEDs
                        LCALL SetAlarm
CheckDakotasBtnRET: RET

;-----------------------------------------------
; @name: PostProcessing
;-----------------------------------------------
PostProcessing:     LCALL Delay
                    RET

;===============================================================================
;                                                                  OTHER METHODS
;===============================================================================
```

```
;-----------------------------------------------
; @name: SetLEDs
; @brief: Sets the LEDs to represent Register 3
;-----------------------------------------------
SetLEDs:

                MOV A,R0
                JB ACC.6, Sides  ; Jump to side lights if 1

                MOV A,R3         ; Turn on corner LEDs
                CPL A            ; Account for active low

                MOV C,ACC.3
                MOV P2.7,C
                MOV C,ACC.2
                MOV P2.4,C
                MOV C,ACC.1
                MOV P2.5,C
                MOV C,ACC.0
                MOV P2.6,C
                SJMP SetLEDsRET

Sides:          MOV A,R3         ; Turn on Side LEDs
                CPL A            ; Account for active low

                MOV C,ACC.3
                MOV P0.4,C
                MOV C,ACC.2
                MOV P0.5,C
                MOV C,ACC.1
                MOV P0.6,C
                MOV C,ACC.0
                MOV P0.7,C

SetLEDsRET:     RET

;-----------------------------------------------
; @name: IncMethod
; @brief: Increments Register 3
;-----------------------------------------------
IncMethod:      INC R3
                RET

;-----------------------------------------------
; @name: DecMethod
; @brief: Decrements Register 3
;-----------------------------------------------
DecMethod:      DEC R3
                RET

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

```
; @name: VincesMethod
; @brief: Toggle what this button does after
;         each time it is pressed:
;         1)Flip the bits of the current counter
;         value (R3)
;         2)Rotate the counter bits (left).
;-----------------------------------------------
VincesMethod:       MOV A, R0
                    JB ACC.5, RotateR3Bits

                    MOV A, R3           ; Flip the bits
                    CPL ACC.0
                    CPL ACC.1
                    CPL ACC.2
                    CPL ACC.3
                    MOV R3, A
                    SJMP ToggleBtnFunct

RotateR3Bits:       MOV A, R3           ; Rotate the bits
                    MOV B, #2
                    MUL AB              ; First double to shift bits (left)
                    MOV C, ACC.4        ; Take 4th bit and rotate back to front
                    MOV ACC.0, C
                    CLR ACC.4
                    MOV R3, A

ToggleBtnFunct:     MOV A, R0
                    CPL ACC.5           ; Toggle flip/rotate
                    MOV R0, A
                    RET

;-----------------------------------------------
; @name: DakotasMethod
; @brief: 'Rotate' which LED's are used
;-----------------------------------------------
DakotasMethod:
                    MOV A,R0
                    CPL ACC.6       ; change LED outputs
                    MOV R0,A

                    SETB P2.4       ; clear corner lights
                    SETB P2.7
                    SETB P2.5
                    SETB P2.6

                    SETB P0.5       ; clear side lights
                    SETB P0.4
                    SETB P0.7
                    SETB P0.6

                    RET             ; Return
```

```
;-----------------------------------------------
; @name: CheckForTurnOver
; @brief: If operation caused turn over, set
;         R0.7 to 1. Clear R3.4 thru R3.7
;-----------------------------------------------
CheckForTurnOver:
                    MOV A,R3
                    MOV B,R0
                    JNB ACC.4,Over  ; if R3.4 is 0, jump to Over
                    SETB B.7
                    CLR ACC.4       ; clear excess bits
                    CLR ACC.5
                    CLR ACC.6
                    CLR ACC.7
                    MOV R3,A

Over:
                    JNB ACC.7,Under ; if R3.7 is 0, jump to Under
                    SETB B.7
                    CLR ACC.4
                    CLR ACC.5
                    CLR ACC.6
                    CLR ACC.7
                    MOV R3,A

Under:
                    MOV R0,B
                    RET

;-----------------------------------------------
; @name: SetAlarm
; @brief: Sound the alarm for .1 sec with a
;         frequency of 1kHz.
;-----------------------------------------------
SetAlarm:           MOV A, R0
                    JNB ACC.7, SetAlarmRET  ; Skipping this if didn't turn over
                    MOV R4, #25             ; Iterations per half-period
                    MOV R2, #50             ; Most significant duration count
                    MOV R1, #40             ; Least significant duration count
                    CLR ACC.7
                    MOV R0,A
SetAlarmRET:        RET

;-----------------------------------------------
; @name: Delay
; @brief: Oscillate the speaker between on and
;              off for a brief period of time
;-----------------------------------------------
Delay:
                    MOV A,R1
                    JZ ChkR2
```

```
                    MOV A,R4
                    JNZ OscDec

                    DEC R1          ; Dec R1 (R1 is not 0, R4 is 0)
                    CPL P1.7        ; Toggle sound
                    MOV R4, #26     ; R4 = 25+1 to account for dec
                    DEC R4          ; Dec R4 (R1 is not 0)
                    SJMP DelayRet ; Finished

OscDec:             NOP
                    NOP
                    NOP
                    DEC R4          ; Dec R4 (R1 is not 0)
                    SJMP DelayRet ; Finished

ChkR2:              NOP
                    MOV A,R2
                    JZ DelayEnd

                    DEC R4          ; Dec R4  (R1 is 0, R2 is not 0)
                    DEC R2          ; Dec R2
                    MOV R1, #40     ; R1 = 40
                    SJMP DelayRet ; Finished
DelayEnd:
                    CLR P1.7        ; Sound off (R1 is 0, R2 is 0)
                    MOV R1, #0      ; Clear registers
                    MOV R2, #0
                    MOV R4, #0
DelayRet:
                    RET


;-----------------------------------------------
; @name: SamsDelay
; @brief: Time Delay used by Sam
;-----------------------------------------------
SamsDelay:
                    MOV R5,#10
ACH:
                    MOV R6,#250
HERE:               MOV R7,#250
AGAIN:              DJNZ R7,AGAIN
                    DJNZ R6,HERE
                    DJNZ R5,ACH
                    RET
;-----------------------------------------------
; @name: SamsMethod
; @brief: Clears Register R3 (the count)
; Also flashes all LEDs twice for input feedback
;-----------------------------------------------
SamsMethod:
                    MOV R3,#0
```

```
                    lcall TurnOffLEDS
                    lcall SamsDelay
                    lcall TurnOnLEDS
                    lcall SamsDelay
                    lcall TurnOffLEDS
                    lcall SamsDelay
                    lcall TurnOnLEDS
                    lcall SamsDelay
                    lcall TurnOffLEDS
                    RET

;-----------------------------------------------
; @name: TurnOnLEDs
; @brief: Turns on all LEDs
;-----------------------------------------------
TurnOnLEDS:         CLR P2.4
                    CLR P0.5
                    CLR P2.7
                    CLR P0.6
                    CLR P1.6
                    CLR P0.4
                    CLR P2.5
                    CLR P0.7
                    CLR P2.6
                    RET
;-----------------------------------------------
; @name: TurnOffLEDs
; @brief: Turns off all LEDs
;-----------------------------------------------
TurnOffLEDS:        SETB P2.4
                    SETB P0.5
                    SETB P2.7
                    SETB P0.6
                    SETB P1.6
                    SETB P0.4
                    SETB P2.5
                    SETB P0.7
                    SETB P2.6
                    RET


end
```