

# A REPORT ON SPIRIT LEVEL TESTER

---

## PROBLEM STATEMENT

System to be designed: Spirit Level Tester System Description:

Used for testing the sobriety of a person. This system tests the sobriety of the user by giving a sequence of hexadecimal characters of varying length(6 to 12 characters) by showing it on the LCD. It records the reaction time of the user in ms and displays the same on the LCD on a successful patch. The sequence should disappear after 2 seconds. The user is given a new pattern if he/she enters a wrong pattern. The user is given a total of three chances. After three mismatches of patterns an alarm is sounded. The pattern is generated randomly by the systems.

## ASSUMPTIONS

---

- ☐ Pseudo random numbers are generated with a cycle of 19683 and hence, in this condition, supposed to be fairly random
- ☐ Software delays are producing the same amount of delay as hardware delay.
- ☐ Simulation time is not very slow as compared to real time.
- ☐ All the port driven i/o devices are working with any voltage output from the 8255A pins
- ☐ Reaction time is being shown in microseconds irrespective of the sobriety of the person.

## LIST OF COMPONENTS USED

---

CHIP NUMBER QTY. CHIP PURPOSE

```
8086 1 Microprocessor Central Processing Unit
74HC373 4 8 bit Octal Latches Latching the Address Bus and
the buzzer
74 HC245 4 8 bit Octal Latches Latching the Data Bus and the
LCD Panel
74LS138 1 3:8 Decoder Chip Select for ROM and Ram
in memory interfacing
74154 1 4:16 Decoder Chip Select for 8255 A to
connect the I/O devices
2732 2 ROM (4KB each) Read Only Memory
To store the code
6116 2 RAM (2KB each) Random Access Memory
To store and retrieve temporary
memory
8255A 2 Programmable
Peripheral
Interface
```

Connects I/O devices to the memory
555 1 555 Timer Generates Clock pulse to measure reaction time LM020L 1 Liquid Crystal Display
Displays the random patterns
BUZZER 1 Buzzer Beeps as affirmative

OTHER HARDWARE USED

- ☐ Logic Gates - Used for building decoding logic for memory interfacing
- ☐ A Hex Keypad - Used for entering patterns

MEMORY MAPPING

The System uses 4KB RAM (2KB banks of each odd and even) and 8KB ROM (4KB banks for each odd and even) thus enabling copy of 16-bit of data in one cycle.

ROM : 8KB

Even bank starting Address — 00000h Even bank ending Address — 01FFEh Odd bank starting address — 00001h Odd bank ending address — 01FFFh

RAM : 4KB

Even bank starting address - 02000h Even bank ending address - 02FFEh Odd bank starting address - 02001h Odd bank ending address - 02FFFh
--

MEMORY AND ADDRESS MAP TABLE

A0 and BHE' signal is used for separating even and odd banks.

## I/O INTERFACING

---

## FLOW CHART

---

## CODE

---

```
.model tiny .data
```

```
; _____ ; assigning address for counters ; ----- count1 db 3
count2 dw 0 count3 dw 0 flag db 0 seed dw 0deafh,0abbah,9876h,0fafah,8e34h,3847h,9218h,0fadeh
```

```
; ----- ; Actual and entered word ; -----
```

```
randomdata db 12 dup(0) inputs db 12 dup(0) random dw 0
```

```
; ----- ; assigning port addresses 8255- 1 ; ----- porta equ 10h
portb equ 12h portc equ 14h creg equ 16h
```

```
; _____ ; assigning port addresses for 8255- 2 ; -----
```

```
porta2 equ 20h portb2 equ 22h portc2 equ 24h creg2 equ 26h
```

```
table_keys db 0eeh,0edh,0ebh,0e7h db 0deh,0ddh,0dbh,0d7h db 0beh,0bdh,0bbh,0b7h db
07eh,07dh,07bh,077h
```

```
; ----- ; CODE
```

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

```
.code .startup call port_initialization call port2_initialization call lcd_initialization
```

```
try: ;Attempts Left=Count dec count
```

```
lea si,seed add si,count mov dx,[si] add count2,
```

```
;generating random string length call random_integer
```

```
mov cx,[random]
```

```
lea si,seed add si,count mov dx,[si] add count2,
```

```
call random_string call write_pattern call delay call delay_ call cls
```

```
call keypad_interface
```

```
call comparison cmp flag, jz reactiontime
```

```
cmp count1, jnz try
```

```

; if reaches here then he/she is drunk.

; call buzzer mov al,00000001b out porta2,al

mov al,00000001b out portb2,al call cls call write_d call delay_ call cls jmp eop

reactiontime:

mov al,00000001b out portb2,al

call write_nd call cls

; display reaction time on lcd call delay_ call delay_ call delay_ call delay_ call delay_

eop:

.exit

;----- ;Initialising port 1 and port 2 ;-----

; port 1

port_initialization proc near

mov al,10001000b out creg,al

ret

port_initialization endp

; port 2

port2_initialization proc near

mov al,10000000b out creg2,al

mov al,00000000b out porta2,al

mov al,00000000b out portb2,al

ret port2_initialization endp

; ----- ; Delay Functions ; -----

; delay in the circuit here the delay of 20 millisecond is produced

delay proc mov cx, 1325 ; 1325*15.085 usec = 20 msec

w1: nop nop nop nop nop

loop w

ret

delay endp

```

;delay in the circuit here the delay of 2000 millisecond is

delay\_2000 proc

mov cx, t1: loop t

ret

delay\_2000 endp

;-----

;Input from Keyboard ;-----

keypad\_interface proc near

mov al,00000001b out portb2,al

; counter 3 used for randomized counter can hold any initial value

mov cx, lea di,inputs mov al,00h

cl0: stosb loop cl

mov al, lea di,count stosb

mov al,88h out creg,al

mov al,0ffh out portb,al

; control word for counter0-mode ; control word for counter 1- mode ; control word for counter2 - mode ;  
loading words to counters ; ----- ; cascaded counter 0 and 1 count till 2 seconds ; input  
frequency to clock = 5 MHZ ; output frequency = 0.5HZ ; therefore counter 0 loaded with 10000 = 2710h and  
counter1 with 1000 = 03E ; to give final N factor of  $10^7$  thus reducing frequency to 0.5 hz ; -----  
-----

x0: mov al,00h out portc,al

x1: in al, portc

and al,0f0h cmp al,0f0h jnz x

call d20ms

mov al,00h out portc ,al

x2: in al, portc and al,0f0h cmp al,0f0h jz x

call d20ms

mov al,00h out portc ,al

in al, portc

and al,0f0h cmp al,0f0h jz x

```
mov al, 0eh mov bl,al out portc,al

in al,portc

and al,0f0h cmp al,0f0h jnz x

mov al, 0dh mov bl,al out portc ,al

in al,portc

and al,0f0h cmp al,0f0h jnz x

mov al, 0bh

mov bl,al out portc,al

in al,portc

and al,0f0h cmp al,0f0h jnz x

mov al, 07h mov bl,al out portc,al

in al,portc

and al,0f0h cmp al,0f0h jz x

x3: or al,bl mov cx,0fh mov di,00h

x4: cmp al,table_keys[di] jz x

inc di loop x

x5: mov ax,di and ax,000ffh

cmp al,0ah jae atofkey

add al, jmp display

atofkey: add al, jmp display

display: lea di,inputs add di,count stosb

mov al,[di-1]

call datwrit

inc count

mov bx,count cmp bx,random jz donewithinput

jmp x

d20ms: ;delay generated will be approx 0.45 secs mov cx,

xn: loop xn
```

```
donewithinput: mov al,00000000b out portb2,al
```

```
ret
```

```
keypad_interface endp
```

```
comparison proc near
```

```
lea si,randomdata lea di,inputs mov cx,random
```

```
c1: mov al,[si] mov bl,[di] cmp al,bl jnz eoproc
```

```
inc si inc di loop c
```

```
mov flag,
```

```
eoproc: ret
```

```
comparison endp
```

```
;----- ;Random Generatoor ;-----
```

```
;random no generator, value of 'n' is in cx, dx value should also be initialized with a different seed everytime
```

```
random_integer proc
```

```
lea di,random mov ax,dx mov bx, mul bx
```

```
add ax, mov bx, div bx mov ax,dx
```

```
mov cl,07h and ax,00ffh div cl
```

```
mov al,ah add al,
```

```
stosb ret
```

```
random_integer endp
```

```
random_string proc
```

```
mov cx,random lea di,randomdata
```

```
r1: mov ax,dx mov bx, mul bx add ax,
```

```
mov bx, div bx
```

```
mov bx,dx and bx,000fh mov al,bl
```

```
cmp al,0ah jae atof
```

```
add al, jmp store
```

```
atof: add al,
```

```
store: stosb loop r
```

ret

random\_string endp

;-----;LCD Functions;-----

;Initialise

lcd\_initialization proc near

mov al, 38h ;initialize lcd for 2 lines & 5\*7 matrix

call comndwrt ;write the command to lcd call delay ;wait before issuing the next command call delay ;this command needs lots of delay call delay

mov al, 0eh ;send command for lcd on, cursor on

call comndwrt call delay

mov al, 01 ;clear lcd

call comndwrt call delay

mov al, 06 ;command for shifting cursor right

call comndwrt call delay

ret

lcd\_initialization endp

; Clear

cls proc mov al, 01

call comndwrt call delay call delay

ret

cls endp

; Writing a command

comndwrt proc

mov dx, porta out dx, al ;send the code to port a

mov dx, portb mov al, 00000100b ;rs=0,r/w=0,e=1 for h-to-l pulse out dx, al

nop nop

mov al, 00000000b ;rs=0,r/w=0,e=0 for h-to-l pulse out dx, al

ret

comndwrt endp



; Write drunk

write\_d proc near

call cls

mov al, 'd' call datwrit call delay call delay

mov al, 'r' call datwrit call delay call delay

mov al, 'u' call datwrit call delay call delay

mov al, 'n' call datwrit call delay call delay

mov al, 'k' call datwrit call delay call delay

ret

write\_d endp

; Write not drunk

write\_nd proc near call cls mov al, 'n' call datwrit call delay call delay

mov al, 'o' call datwrit call delay call delay

mov al, 't' call datwrit call delay call delay

mov al, 'd' call datwrit call delay call delay

mov al, 'r' call datwrit call delay call delay

mov al, 'u' call datwrit call delay call delay

mov al, 'n' call datwrit call delay call delay

mov al, 'k' call datwrit call delay call delay

ret

write\_nd endp

; ----- ; Other Functions ; -----

write\_pattern proc near

lea di,randomdata call cls

mov cx,random mov si,cx

x10: mov al, [di]

call datwrit call delay call delay

inc di dec si

jnz x10

ret

write\_pattern endp

datwrit proc

push dx ;save dx

mov dx,porta ;dx=port a address out dx, al ;issue the char to lcd

mov al, 00000101b ;rs=1, r/w=0, e=1 for h-to-l pulse mov dx, portb ;port b address out dx, al ;make enable high

mov al, 00000001b ;rs=1,r/w=0 and e=0 for h-to-l pulse out dx, al

pop dx

ret

datwrit endp ;writing on the lcd ends

end

## CIRCUIT DIAGRAM

---