THE MICROPROCESSORS & MICROCONTROLLERS

Instructor : The Tung Than
Student: Bach Hoang Phan Thanh
ID: 21520599

PRATICE EXERCISE #2:
COMMUNICATION WITH 7-SEGMENT LED AND TIMER

TABLE OF CONTENTS

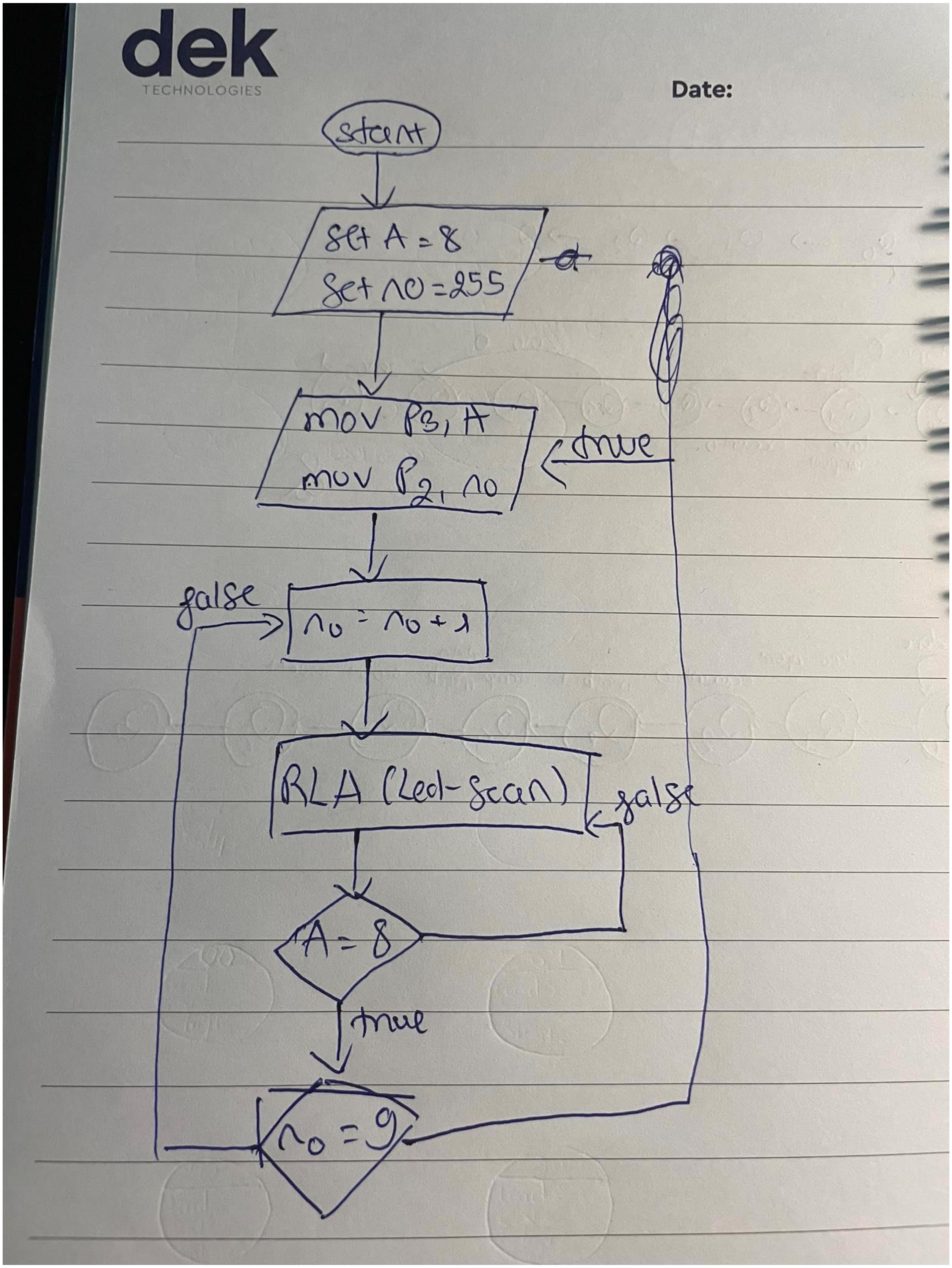
1. Design Result

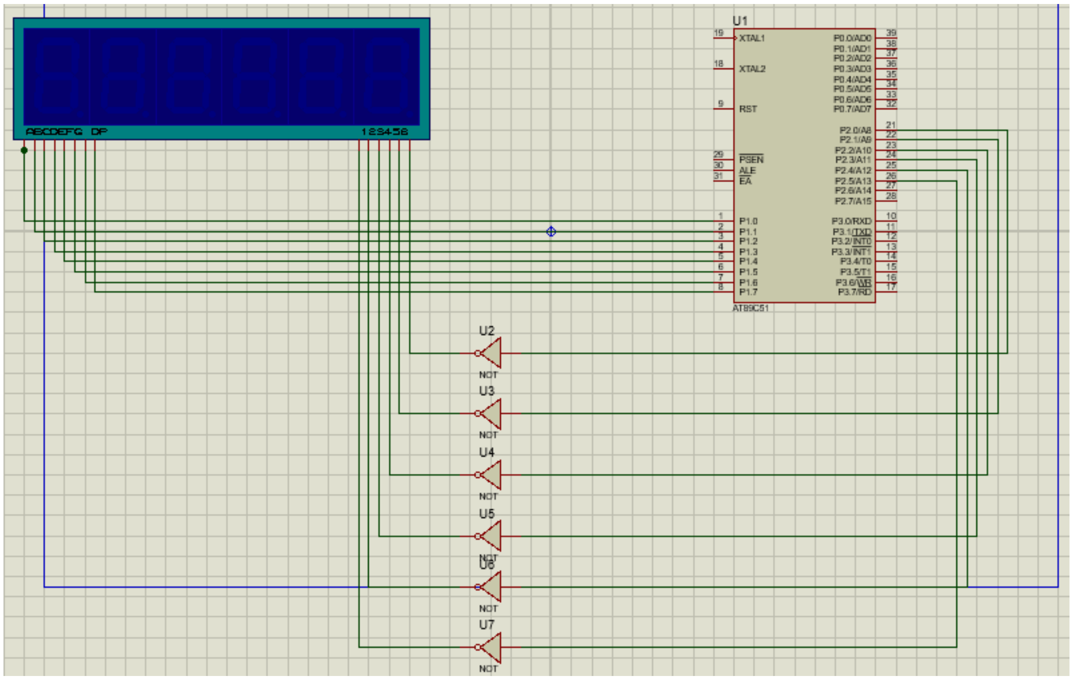
Task:

+ Present and draw a flowchart of the LED scanning algorithm applied to display 7-segment led.

+ Using the 8051 microcontroller's Timer, design a clock circuit with 24h format with the initial time set in the source code

Picture of Result:


dek TECHNOLOGIES
Date:
start
set A = 8
set r0 = 255
mov P3, A
mov P2, r0
true
false
r0 = r0 + 1
RLA (Led-scan)
false
A = 8
true
r0 = 9


ABCDEFG DP
123456
U1
19
XTAL1
18
XTAL2
9
RST
29
PSEN
30
ALE
31
EA
1
P1.0
2
P1.1
3
P1.2
4
P1.3
5
P1.4
6
P1.5
7
P1.6
8
P1.7
P0.0/AD0
39
P0.1/AD1
38
P0.2/AD2
37
P0.3/AD3
36
P0.4/AD4
35
P0.5/AD5
34
P0.6/AD6
33
P0.7/AD7
32
P2.0/A8
21
P2.1/A9
22
P2.2/A10
23
P2.3/A11
24
P2.4/A12
25
P2.5/A13
26
P2.6/A14
27
P2.7/A15
28
P3.0/RXD
10
P3.1/TXD
11
P3.2/INT0
12
P3.3/INT1
13
P3.4/T0
14
P3.5/T1
15
P3.6/WR
16
P3.7/RD
17
AT89C51
U2
NOT
U3
NOT
U4
NOT
U5
NOT
U6
NOT
U7
NOT

2. Explain the operating principle of the effects

- Google Drive link: tinyurl.com/4arp2kb7
- Source code, include English Explanation:
- **Note:** When copy from Proteus to Word, the color for each instruction, initial number has been changed to black. So I try to change the color like which it display on Proteus.

Source Code (Include English Explanation)

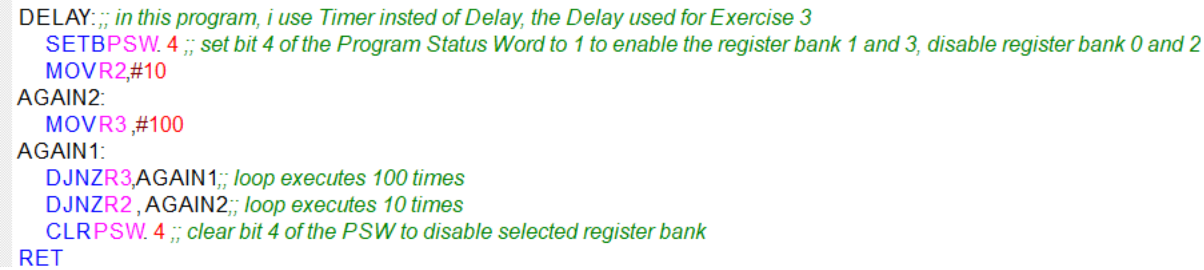
```
ORG 00H
    LJMP MAIN
ORG 30H  ;;set the starting address
MAIN:
    MOV DPTR , #MYDATA  ;; move the address of the data stored in MYDATA to the DPTR  - data pointer register
START:
    MOV R0,#00H ;; move the value 00h into R0 register, the same to line 8 - 12 code
    MOV R1,#00H ;; R0, R1, R2 , R3, R4, R5  is the led in  7seg - mpx6 - cc - blue
    MOV R2,#00H
    MOV R3,#00H
    MOV R4,#00H
    MOV R5,#00H
    ;; the purpose of line 14 code is to set up the initial time of the clock is 23h59m59s
    CJNE R7 ,#0D , DISPLAY  ;; if R7 are not equal to the value 0D, it jumps to DISPLAY, otherwise it continues with the line 15 code
    MOV R0,#9H
    MOV R1,#5H
    MOV R2,#9H
    MOV R3,#5H
    MOV R4,#3H
    MOV R5,#2H
   ;; the line 15 - 20 code is for setting up the initial time value of the clock is 23h59m59s
    MOV R7,#00H;; move the value 00h into register R7, so when it turn to line 14 code, it will jump to DISPLAY instead of continues with the line 15 code
 DISPLAY:
    INC R7 ;; increase the value of register R7
    ACALL SHOW  ;; call the SHOW subroutine
BACK:
    ;; first, to increase the seconds of the clock, i use R0 as the units digit and R1 for the tens unit
    INC R0
    CJNE R0, #10D, DISPLAY;
    MOV R0, #00H
    ;; when R0 reach 10, it will reset back to 0, and increse the R1 register
    INC R1
    CJNE R1, #6D ,DISPLAY
    MOV R1, #00H
    ;; when the second reach 60, it will reset to 0 and increase the minute, i continually use R2 as the units digit and R3 for the tens unit
    INC R2
    CJNE R2,#10D,DISPLAY
    MOV R2,#00H
    ;; when R2 reach 10, it will reset back to 0 and increase the R3 register
    INC R3
    CJNE R3,#6D,DISPLAY
    MOV R3,#00H
    MOV A,R5 ;; move the contents of R5 register into A register
    XRL  A, #2D ;; XOR between A and 2D, it will stored the result in register
    ;; in this program, the result of XOR will be 0, 2 or 3
    JZ  ZERO  ;; if the previous instruction, mean the result of the line 45 code is 0, it will jump to ZERO
    JNZ  NOTZERO ;; otherwise, it jumps to NOTZERO
    ;; to increase the hour, i use R4 as the units digit and R5 for the tens unit
    ;; NOTZERO use for when  the time is around 0h to 19h
NOTZERO:
    INC R4
    CJNE R4,#10D,DISPLAY ;; when R4 reach 10, it will reset back to 0
    MOV R4,#00H
    SJMP NOTZERONEXT
    ;;ZERO  is used for when the time is around 20h to 23h
ZERO:
    INC R4
    CJNE R4,#4D,DISPLAY;; when R4 reach 4, it will reset back to 0
    MOV R4 ,#00H
    MOV R5,#02D;; move the value 02D into the R5 register
    ;; NOTZERONEXT is used to increase the R5 register when the time is around 0h to 19h
NOTZERONEXT:
    INC R5
    CJNE R5,#3D,DISPLAY;; when R5 - the tens digit to perform hour in clock reach 3, it will reset back to 0
    SJMP  START ;; jump to START

SHOW:
    MOV R6,#33d ;; move the value 33 decimal value into r6 register, r6 register is used like a delay for REPEAT
REPEAT:
    MOV A,R0
    MOVC A,@A+DPTR ;; copy the value of the byte pointed to DPTR plus the value in A into the A register
    SETB P2.0 ;;sets bit 0 of the port 2 to high
    MOV P1,A ;; move the contents of A register into the P1 port which controls the lod
    ACALL LOOP ;; call LOOP for delay
    CLR P2.0 ;; clear bit 0 of the port 2, mean turn off the LED
;; the same is for code line 78 to 111
    MOV A,R1
    MOVC A,@A+DPTR
    SETB P2.1
    MOV P1,A
    ACALL LOOP
    CLR P2.1

    MOV A,R2
    MOVC A,@A+DPTR
    SETB P2.2
    MOV P1,A
    ACALL LOOP
    CLR P2.2

    MOV A,R3
    MOVC A,@A+DPTR
    SETB P2.3
    MOV P1 ,A
    ACALL LOOP
    CLR P2.3

    MOV A,R4
    MOVC A,@A+DPTR
    SETB P2.4
    MOV P1,A
    ACALL LOOP
    CLR P2.4

    MOV A,R5
    MOVC A,@A+DPTR
    SETB P2.5
    MOV P1,A
    ACALL LOOP
    CLR P2.5

    DJNZ  R6, REPEAT ;; decrease the value of R6 register and jump back
to REPEAT if the result is not zero
RET

DELAY: ;; in this program, i use Timer insted of Delay, the Delay used for
Exercise 3
    SETB PSW. 4 ;; set bit 4 of the Program Status Word to 1 to enable the
register bank 1 and 3, disable register bank 0 and 2
    MOV R2,#10
AGAIN2:
    MOV R3 ,#100
AGAIN1:
    DJNZ R3,AGAIN1 ;; loop executes 100 times
    DJNZ R2 , AGAIN2 ;; loop executes 10 times
    CLR PSW. 4 ;; clear bit 4 of the PSW to disable selected register bank
RET

LOOP:
    MOV TMOD, #01H ;; set the timer mode to Mode 1, which is a 16-bit
timer with auto - reload
    MOV TH0, #HIGH(-5000) ;;loads the high byte of the initial value for
Timer 0 to delay for 5000us
    MOV TL0, #LOW(-5000) ;; loads the low byte of the initial value for
Timer 0 to delay for 5000us
```

```
     SETB TR0 ;;sets the TR0 bit to statrts Timer0
     JNB TF0, $ ;;check if the Timer0 overflow flag is 0
;; If 0, the program conitinues excecuting from the next line
;; If 1, it will causes and infinite loop
    CLR TR0 ;; clear the TR0 bit, mean stop Timer0
    CLR TF0 ;; clear the TF0 bit, mean resets Timer0 overflow flag
RET
ORG 300H
MYDATA:
  ;; MYDATA defines a block of 10 consecutive bytyes
  ;; each byte represents a specific pattern of seven segments which can be
used to display decimal digits in seven-segment display
   DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
END
```

3. Exercise Report

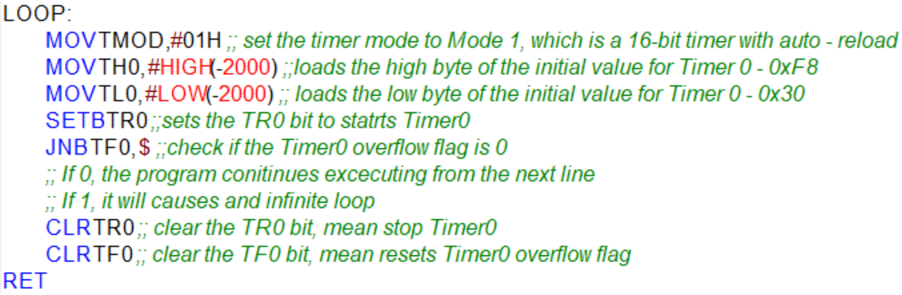
	Advantages	Disadvantages
Using delays	The function is simple to implement, requires minimal code. The delay function does not require any special hardware, easily implemented using software. The delay time is easily controlled and can be adjusted to any desired value.	The delay function can only perform simple time delays and is not suitable for complex timing requirements. The delay function consumes CPU cycles, which can slow down other operations in the program. Delay functions may not always provide accurate timing because the delay time is dependent on the CPU clock frequency and other factors.
Using timers	Timers can be used for a variety of timing functions and can be programmed to perform a range of tasks. Timers are hardware-based and do not consume CPU cycles, which leaves more processing power available for other tasks. Timers provide accurate timing because they are not affected by CPU clock speed or other factors.	Timer functions can be more complex to implement than delay functions and may require more code. Timers require special hardware in the microcontroller, which may limit their availability in some applications. Timers have a finite resolution, which may limit their precision for certain applications.

Picture of Delays:



```
 DELAY:;; in this program, i use Timer insted of Delay, the Delay used for Exercise 3
    SETBPSW. 4 ;; set bit 4 of the Program Status Word to 1 to enable the register bank 1 and 3, disable register bank 0 and 2
    MOVR2,#10
 AGAIN2:
    MOVR3 ,#100
 AGAIN1:
    DJNZR3,AGAIN1;; loop executes 100 times
    DJNZR2 , AGAIN2;; loop executes 10 times
    CLRPSW. 4 ;; clear bit 4 of the PSW to disable selected register bank
 RET
```

Picture of Timers:



```
LOOP:
    MOVTMOD,#01H ;; set the timer mode to Mode 1, which is a 16-bit timer with auto - reload
    MOVTH0,#HIGH(-2000) ;;loads the high byte of the initial value for Timer 0 - 0xF8
    MOVTL0,#LOW(-2000) ;; loads the low byte of the initial value for Timer 0 - 0x30
    SETBTR0;;sets the TR0 bit to statrts Timer0
    JNBTF0,$ ;;check if the Timer0 overflow flag is 0
    ;; If 0, the program conitinues excecuting from the next line
    ;; If 1, it will causes and infinite loop
    CLRTR0;; clear the TR0 bit, mean stop Timer0
    CLRTF0;; clear the TF0 bit, mean resets Timer0 overflow flag
RET
```