ISLAMIC UNIVERSITY OF TECHNOLOGY

PROJECT REPORT

EEE-4706

PULSE WIDTH AND FREQUENCY MEASUREMENT

GROUP-4(A1)

TEAM MEMBERS:

190021105

190021115

190021111

190021147

Objectives

In our project we measured pulse width and frequency measurement using an 8051 microcontroller. This project integrates fundamental functions including monitoring frequency and duty cycle, producing warnings based on thresholds, and synthesizing a new signal by combining two input signals.

Project Goals: The project's major goal is to design a system that accurately measures input signal frequency and duty cycle. An 8051 microcontroller will capture and analyze the signal and show the measured results on an LCD. The device will also notify and display the frequency deviation on the LCD screen if it deviates from 50Hz.

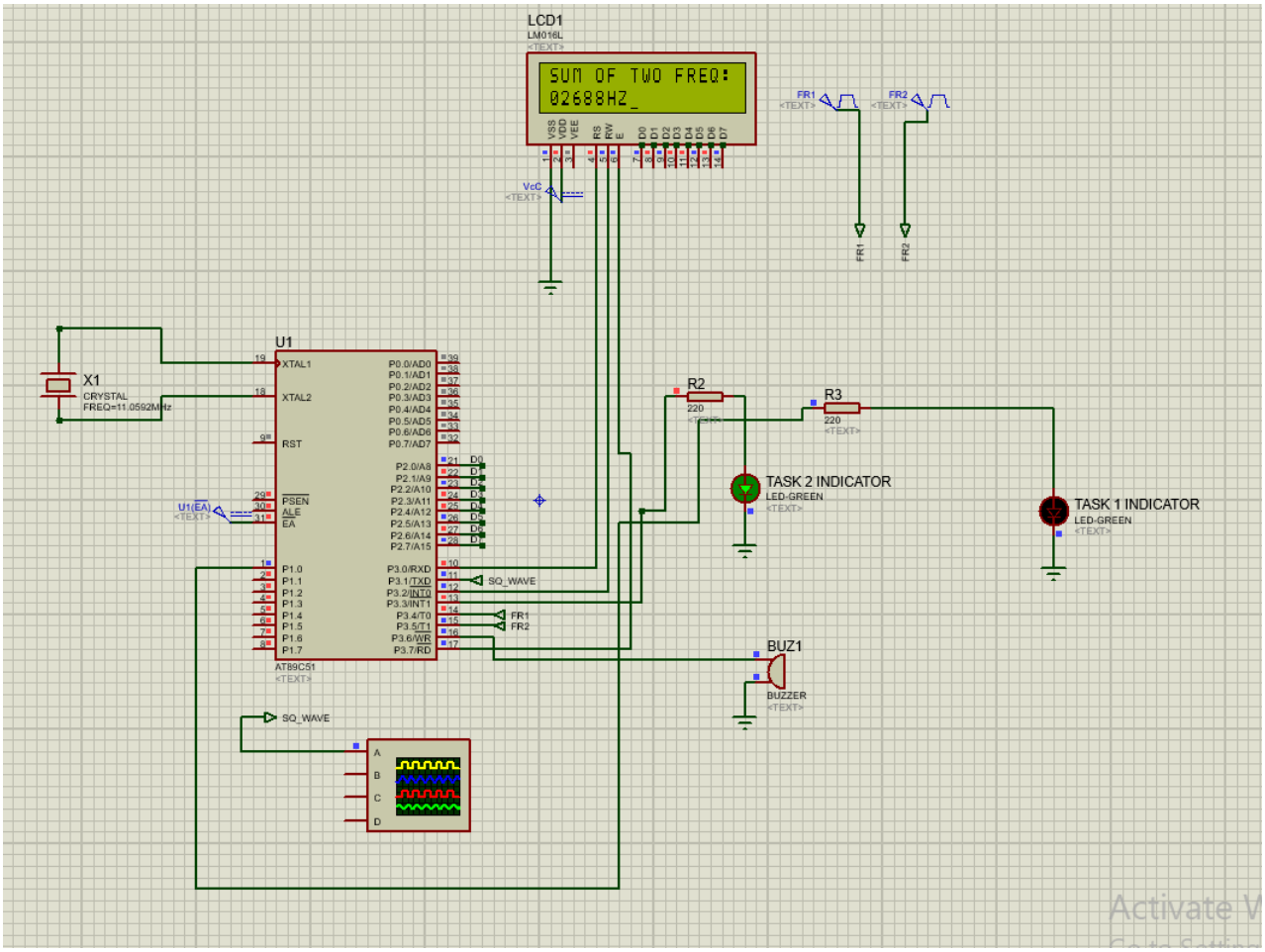
Signal Synthesis: Another objective is to accept two input signals, calculate their frequencies, and synthesize a new signal using the sum of their frequencies. The 8051 microcontroller accurately measures each input signal's frequency and facilitates the creation of a composite signal that accurately represents the sum of its frequencies.The project uses task-specific indicators to improve the user interface and provide rapid feedback on ongoing operations. Indicators use LEDs. LED 1 will light up when the system monitors signal frequency and duty cycle. LED 2 activity shows that the microcontroller is mixing two input frequencies to form a new signal.

Uninterrupted Operation: The project aims for seamless, uninterrupted operation so the system can repeat its tasks. This ensures uninterrupted frequency, duty cycle, and signal synthesis. The system's continuous operation makes it beneficial in many applications.

Required Components

Components	Cost
AT89C52 Development Board	7000
Signal Generator	0
Jumper Wire	25
16x2 LCD Display	170
Total	**7200 BDT**

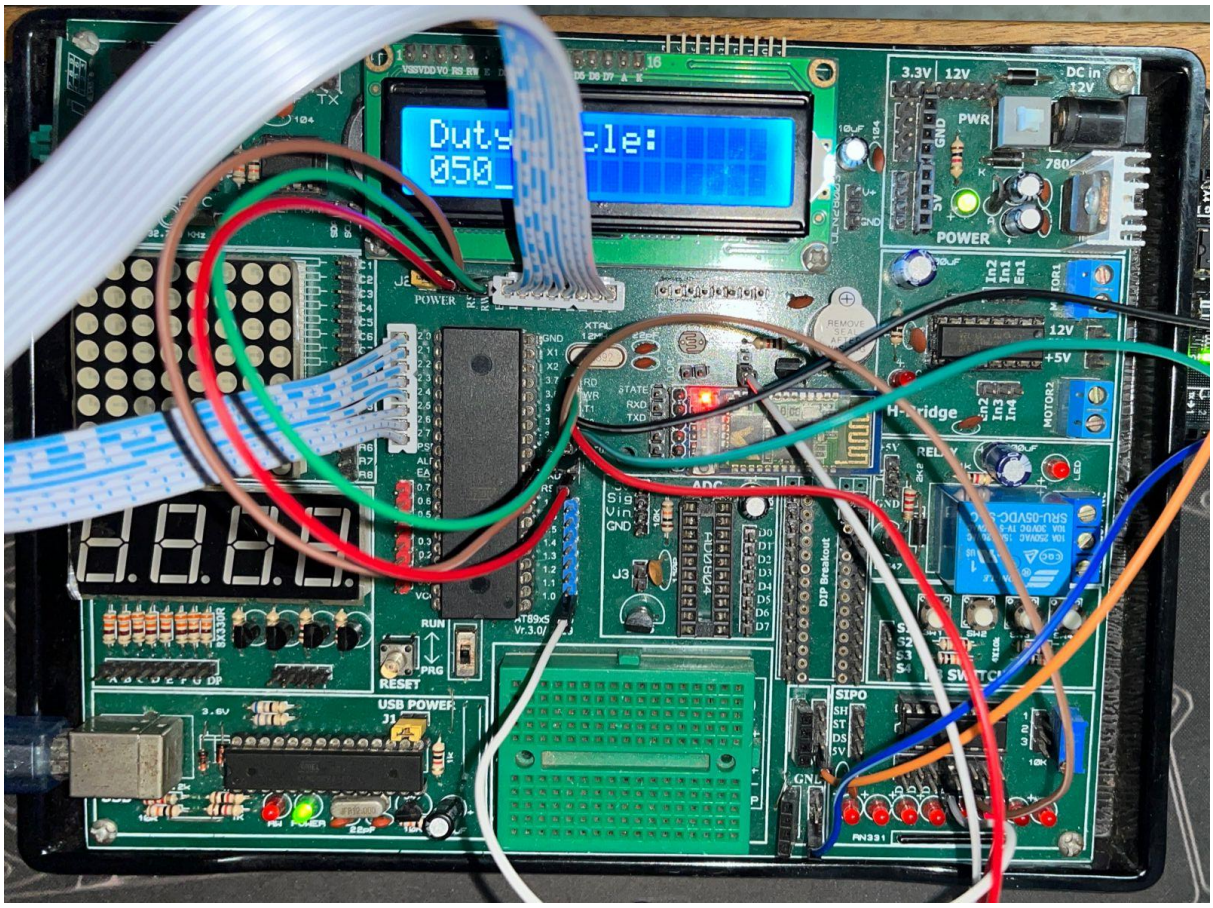
Circuit Diagram



In the circuit we implemented all 3 tasks accordingly. For task 1 the pulse generator is connected with pin 3.4 which is timer 0 and using both timer 0 & 1 we are counting the frequency and the output is shown in the LCD display. Also when the frequency is not 50hz we send a high signal to the buzzer. When task 1 is being carried out we turn our LED on for task 1 indication.

After Showing the required values of task 1 our task 1 indicator is automatically turned off and for task 2 another LED is turned on. In task 2 for getting the 2nd signal's frequency we gave input at pin 3.5. Now the timer and counter which is used to measure the frequency is totally opposite of the first one.

Hardware Implementation



Features

Mandatory Features:

i) The microcontroller has to take input from the signal generator on pin 3.4, display Measure the frequency and duty cycle of a signal and display it on the LCD. If it is less/more than 50Hz sound an alarm and specify the difference on the LCD.

ii)Microcontroller takes another signal from pin 3.5 , measures the frequencies and generate a new signal equal to the sum of those frequencies.

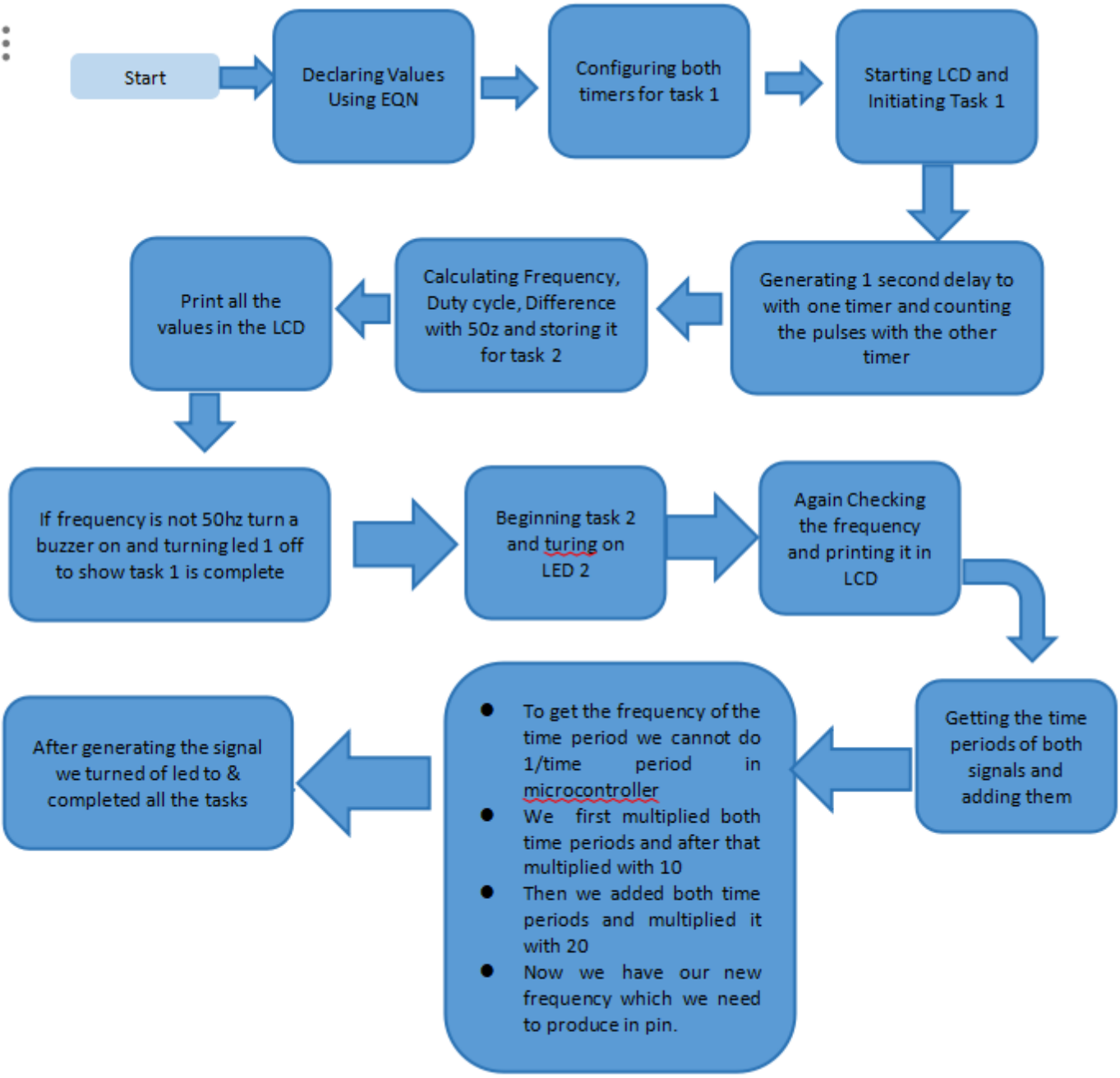
iii) It continuously carry out the above tasks and LED 1 should be turned on when the first task is being carried out, and LED 2 should be turned on when the second task is being carried on.

Additional Features:

i) Microcontroller is not bounded to 255hz range. It can measure upto 65535 using 2 16 bit arithmatic sub routines.

ii) The output is given in decimal rather than hexadecimal.

WORKING PROCEDURE:



Program Code:

```
ORG 00H


RS EQU P3.0
RW EQU P3.2
ENBL EQU P3.7

NEW_SIGNAL EQU P3.1
BUZZER EQU P3.6
LED1 EQU P1.0
LED2 EQU P3.3

MS1 EQU 41H     ; MSD STRG
MS2 EQU 42H
MS3 EQU 43H
MS4 EQU 44H
MS5 EQU 45H     ; LSD STRG

CARRI EQU 46H
CARRI1 EQU 47H
TOFF1 EQU 48H
TOFF2 EQU 49H
TONE1 EQU 50H
TONE2 EQU 51H
MUL1 EQU 52H
MUL2 EQU 53H
MUL3 EQU 54H
MUL4 EQU 55H

FREQ2 EQU 6AH   ; HIGHER BYTE OF FIRST FREQENCY
FREQ1 EQU 6BH   ; LOWER  BYTE OF FIRST FREQUENCY
FREQ2_H EQU 6CH ; AND SO ON
FREQ2_L EQU 6DH
FREQ3_H EQU 6EH
FREQ3_L EQU 6FH

TIME1_H EQU 60H ; HIGH BYTE OF TIME PERIOD OF SIGNAL 1
TIME1_L EQU 61H ; LOW  BYTE OF TIME PERIOD OF SIGNAL 1
TIME2_H EQU 62H ; HIGH BYTE OF TIME PERIOD OF SIGNAL 2
TIME2_L EQU 63H ; LOW  BYTE OF TIME PERIOD OF SIGNAL 2

DUTY EQU 64H
QDIVH EQU 65H
QDIVL EQU 66H

T1_10H EQU 67H
T1_10L EQU 68H
T1_T2H EQU 69H
T1_T2L EQU 70H
```

```
        TN_H EQU 75H
        TN_L EQU 76H


        ;Defining O/P Pins
        CLR BUZZER
        CLR NEW_SIGNAL
        CLR LED1
        CLR LED2


START:

        ACALL INITLCD
        ACALL DELAY
        SETB LED1
;-------------------FREQUENCY 1 MEASUREMENT----------------------------------------
        LCALL FREQUENCY1D
;--------------------DISPLAY FREQUENCY 1-----------------------------------

        ACALL INITLCD
        MOV DPTR, #MSG_3
        ACALL DISPLAY_LINE1

        MOV A, #0C0H    ; display line 2
        ACALL COMMAND
        ACALL DELAY

        CLR A
        MOV R1,A ;Clear
        MOV R0,A ;Clear

        MOV R1, FREQ2   ; TH0 CONTAINS HIGHER BYTE OF FREQ
        MOV R0, FREQ1   ; TL0 CONTAINS LOWER  BYTE OF FREQ
        LCALL PRINT_DIGIT_16

        MOV DPTR, #HZ
        ACALL DISPLAY_LINE1
        ;LJMP START
```

```
;-----------------------DIFFERENCE CALCULATION-----------------------------

        CLR A
        MOV A, FREQ2
        CJNE A,#00H, GREATER_50

        MOV A, FREQ1
        CJNE A, #32H, ERPOR

ERPOR:  JNC GREATER_50

        MOV B, FREQ1
        MOV A, #32H
        SUBB A,B

        MOV R1,#00H     ; HIGHER BYTE
        MOV R0,A        ; LOWER  BYTE
        LCALL DIFF_DISP


GREATER_50:
        SETB BUZZER



        MOV R2,#00H
        MOV R3,#00H

        MOV R6,FREQ2
        MOV R7,FREQ1


        MOV R4,#00H
        MOV R5,#32H

        LCALL SUBB16_16

        CLR A

        MOV A,R2
        MOV R1,A

        CLR A
        MOV A,R3
        MOV R0,A
```

```
        LCALL DIFF_DISP

        DIFF_DISP:

        CLR A
        LCALL INITLCD
        MOV DPTR, #MSG_4
        LCALL DISPLAY_LINE1

        MOV A, #0C0H
        LCALL COMMAND
        LCALL DELAY

        LCALL PRINT_DIGIT_16


        LCALL DELAY

        CLR BUZZER

---------------------------------DUTY CYCLE CALCULATION-------------------------

        LCALL DUTY_CALCULATION
        LCALL DELAY

-------------------------------- PRINT DUTY CYCLE--------------------------------

        LCALL INITLCD
        MOV DPTR, #MSG_7
        LCALL DISPLAY_LINE1

        MOV A, #0C0H
        LCALL COMMAND


        CLR A
        MOV A,DUTY
        LCALL PRINT_DIGIT
        LCALL DELAY_1

        CLR LED1
        SETB LED2

        ACALL INITLCD


;---------------------------------- TASK2---------------------------------------

;--------------------FREQUENCY MEASUREMENT-------------------------------------

        MOV TL1,#00H
        MOV TH1,#00H
        MOV TL0,#00H
        MOV TH0,#00H
        CLR TR1
        CLR TR0

        MOV TMOD, #51H ; T1 as COUNTER and T0 as TIMER
        MOV TCON, #00H
        SETB P3.5
        MOV TL1,#00H
        MOV TH1,#00H
        SETB TR1

        CLR A
        MOV R0,A
        MOV R0,#14
AGN1:   MOV TL0, #00H
        MOV TH0, #00H
        SETB TR0
BACK1:  JNB TF0, BACK1
        CLR TF0
        CLR TR0
        DJNZ R0, AGN1
        CLR TR1

        MOV FREQ2_H, TH1
        MOV FREQ2_L, TL1
```

```
;--------------------DISPLAY----------------------------------------

        ACALL INITLCD
        MOV DPTR, #MSG_5
        ACALL DISPLAY_LINE1

        MOV A, #0C0H
        ACALL COMMAND
        ACALL DELAY

        CLR A
        MOV R1,A
        MOV R0,A

        MOV R1, FREQ2_H
        MOV R0, FREQ2_L
        LCALL PRINT_DIGIT_16

        MOV DPTR, #HZ
        MOV DPTR, #HZ
        ACALL DISPLAY_LINE1
        LCALL DELAY_2

        ;reset
        MOV TL1,#00H
        MOV TH1,#00H
        MOV TL0,#00H
        MOV TH0,#00H
        CLR TR1
        CLR TR0

;-----------------TIME PERIOD-----------------------------------------
        LCALL TIME_PERIOD_CALC_2

;-------------ADDITION FREQ1+FREQ2-----------------------;


        MOV R6,FREQ2
        MOV R7,FREQ1

        MOV R4,FREQ2_H
        MOV R5,FREQ2_L

        LCALL ADD16_16

        MOV FREQ3_H, R2
        MOV FREQ3_L, R3
```

```
;---------------DISPLAY ADDITION------------

        ACALL INITLCD
        MOV DPTR, #MSG_6
        ACALL DISPLAY_LINE1

        MOV A, #0C0H
        ACALL COMMAND
        ACALL DELAY


        CLR A
        MOV R1,A
        MOV R0,A

        MOV R1, FREQ3_H ; TH1 CONTAINS HIGHER BYTE OF FREQ
        MOV R0, FREQ3_L ; TL1 CONTAINS LOWER  BYTE OF FREQ
        LCALL PRINT_DIGIT_16

        MOV DPTR, #HZ
        MOV DPTR, #HZ
        ACALL DISPLAY_LINE1
        LCALL DELAY_2


;------------NEW SIGNAL FORMATION--------------

;NEW TIME PERIOD CALCULATION

;-------------------------------------T1*10-----------------------------------

        ;Load the first value into R6 and R7
        MOV R6,TIME1_H
        MOV R7,TIME1_L

        ;Load the first value into R4 and R5
        MOV R4,#00h
        MOV R5,#0Ah

        ;Call the 16-bit subtraction routine
        LCALL MUL16_16

        MOV T1_10H, R2
        MOV T1_10L, R3
```

```
;------------------------------------ (T1*10)/(T1+T2)-----------------------------


;-------------------------------------T1+T2-----------------------------------------

        ;Load the first value into R6(H) and R7
        MOV R6,TIME1_H  ; high byte
        MOV R7,TIME1_L

        ;Load the second value into R4(H) and R5
        MOV R4,TIME2_H  ; high byte
        MOV R5,TIME2_L

        ;Call the 16-bit addition routine
        LCALL ADD16_16  ; T1+T2

        MOV T1_T2H,R2
        MOV T1_T2L,R3

;-------------------------------------- DIVIDE ------------------------------------

        MOV R1, T1_10H
        MOV R0, T1_10L

        MOV R3,T1_T2H
        MOV R2,T1_T2L   ; divide by

        LCALL DIV_16BIT ; r1 r0/ r3 r2= result R3 R2 ,33H 34H reminder
;R3 R2

;----------------------------------------MULTIPLY BY T2---------------------------------------

        ;Load the first value into R6 and R7
        MOV A, R3
        MOV R6, A
        MOV A, R2
        MOV R7, A

        ;Load the first value into R4 and R5
        MOV R4,TIME2_H
        MOV R5,TIME2_L

        ;Call the 16-bit subtraction routine
        LCALL MUL16_16
        ;R2 R3
```

```
;---------------------------------------DIVIDE BY 20-----------------------------------

        ; for division, taking it as dividend
        MOV A, R2       ; R2 HIGH BYTE
        MOV R1,A        ; R1 HIGH BYTE

        MOV A, R3       ; R3 LOW  BYTE
        MOV R0,A        ; R0 LOW  BYTE


        MOV R3,#00H
        MOV R2,#20      ; divide by 20
        LCALL DIV_16BIT ; r1 r0/ r3 r2= result R3 R2 ,33H 34H reminder
        ;R3 R2
        MOV TN_H, R3    ; HI BYTE
        MOV TN_L, R2    ; LW BYTE

;-------------------------------------------------SUBTRACTION. 65536-n--------------------------


        MOV R6,#0FFh
        MOV R7,#0FFh
        MOV R4,TN_H
        MOV R5,TN_L
        LCALL SUBB16_16

        MOV A, R2       ;R2 TO R6
        MOV R6,A                ;high byte
        MOV A, R3       ;R3 TO R7
        MOV R7,A


        MOV R4,#00H     ; high byte
        MOV R5,#01H

        LCALL ADD16_16  ; FFFF-N+1

        MOV TN_H, R2    ;HIGH BYTE
        MOV TN_L, R3    ;LOW BYTE

;---------------------------------------GENERATE THE REQQUENCY FOR 200 CYCLES--------------------
        MOV TMOD, #01H
        MOV R7,#200
HEREU: MOV TL0, TN_L
        MOV TH0, TN_H
        CPL NEW_SIGNAL
        ACALL RUN
        SJMP HEREU
```

```
RUN:
        SETB TR0
AGAINU: JNB TF0,AGAINU
        CLR TR0
        CLR TF0
        DJNZ R7,HEREU
        ;RET

;--------------------RESET-----------------------------------;


        MOV TL1,#00H
        MOV TH1,#00H
        MOV TL0,#00H
        MOV TH0,#00H
        CLR LED2
        CLR TR1
        CLR TR0

; RETURN TO THE ORIGIN
        LJMP START

FREQUENCY1:

;--------------------FREQUENCY MEASUREMENT FOR SIGNAL 1--------------------------

        MOV TMOD, #15H
        MOV TCON, #00H
        SETB P3.4
        MOV TL0,#00H
        MOV TH0,#00H
        SETB TR0
        MOV R0,#14
AGN:    MOV TL1, #00H
        MOV TH1, #00H
        SETB TR1
BACK:   JNB TF1, BACK
        CLR TF1
        CLR TR1
        DJNZ R0, AGN
        CLR TR0
        MOV FREQ2, TH0  ; HIGH BYTE OF FREQ 1
        MOV FREQ1, TL0  ; LOW BYTE OF FREQ2
RET
```

```
;--------------------DUTY CYCLE CALCULATIION-------------------------------------
DUTY_CALCULATION:

;--------------------TOFF CALCULATION--------------------------------------------

T_OFF:
        MOV TL1, #00H
        MOV TH1, #00H
;----------------------------SKIP 2 CYCLES-------------------------------------
        CYCLE1: JNB P3.4,CYCLE1
        CYCLE2: JB P3.4,CYCLE2
        CYCLE3: JNB P3.4,CYCLE3
        CYCLE4: JB P3.4,CYCLE4
        CYCLE_ON: SETB TR1
CYCLE5: JNB P3.4 , CYCLE6 ;
        SJMP CYCLE_END
CYCLE6: JNB TF1  , CYCLE5 ;
        INC CARRI1
        CLR TF1
        SJMP CYCLE5
CYCLE_END: CLR TR1
        MOV TOFF1, TH1
        MOV TOFF2, TL1



;--------------------TON CALCULATION---------------------------;

T_ONE:
        MOV TL1, #00H
        MOV TH1, #00H
CYCLE1S:  JB P3.4 ,CYCLE1S
CYCLE2S: JNB P3.4,CYCLE2S
CYCLE3S:  JB P3.4,CYCLE3S
CYCLE4S: JNB P3.4,CYCLE4S
;----------------------------SKIP 2 CYCLES-------------------------------------
First_task1: SETB TR1
CYCLE5S: JB P3.4 , CYCLE6S ;
        SJMP CYCLE_ENDS
CYCLE6S: JNB TF1  , CYCLE5S ;
        INC CARRI
        CLR TF1         ;
        SJMP CYCLE5S
CYCLE_ENDS: CLR TR1
        MOV TONE1, TH1
        MOV TONE2, TL1
```

```
;-------------------------------------DUTY CYCLE MEASUREMENT-----------------------------------------

;---------------------------------(TON*10)/(TOTAL/10)---------------------------

        MOV R6,TONE1 ;HIGH
        MOV R7,TONE2 ;LOW

        MOV R4,#00h
        MOV R5,#0Ah

        LCALL MUL16_16

        MOV MUL1, R0 ;HIGH
        MOV MUL2, R1
        MOV MUL3, R2
        MOV MUL4, R3 ;LOW

;-------------------------------(TOTAL/10)-------------------------------------

        MOV R6,TONE1 ;HIGH
        MOV R7,TONE2 ;LOW

        MOV R4,TOFF1 ;HIGH
        MOV R5,TOFF2 ;LOW

        LCALL ADD16_16

        MOV TIME1_H, R2 ; TIME PERIOD HIGH
        MOV TIME1_L, R3 ; TIME PERIOD LOW

;---------------------------------- TOTAL/10-------------------------------------

        MOV A, R2       ;HIGH
        MOV R1,A        ;HIGH

        MOV A, R3       ;LOW
        MOV R0,A        ;LOW

        MOV R3,#00H
        MOV R2,#0AH
        LCALL DIV_16BIT ; r1 r0/ r3 r2= QIOTIENT R3 R2 ,33H 34H reminder

        MOV QDIVH, R3   ; TOTAL/10 HIGH
        MOV QDIVL, R2   ; TOTAL/10 LOW
```

```
;--------------------(TON*10)/(TOTAL/10)--------------------------
        MOV R1, MUL3
        MOV R0, MUL4

        MOV R3, QDIVH
        MOV R2, QDIVL

        LCALL DIV_16BIT

        MOV DUTY, R2

RET



TIME_PERIOD_CALC_2:

;--------------------TOFF CALCULATION

T_OFF_2:
        MOV TL0, #00H
        MOV TH0, #00H
        KYCLE1: JNB P3.5,KYCLE1
        KYCLE2: JB P3.5,KYCLE2
        KYCLE3: JNB P3.5,KYCLE3
        KYCLE4: JB P3.5,KYCLE4
        KYCLE_ON: SETB TR0
KYCLE5: JNB P3.5 , KYCLE6 ;
        SJMP KYCLE_END
KYCLE6: JNB TF0  , KYCLE5 ;
        INC CARRI1
        CLR TF0
        SJMP KYCLE5
KYCLE_END: CLR TR0
        MOV TOFF1, TH0
        MOV TOFF2, TL0


;--------------------TON CALCULATION----------------------------;

T_ONE_2:
        MOV TL0, #00H
        MOV TH0, #00H
                                SKIP 2 CYCLES
```

```
;--------------------------SKIP 2 CYCLES---------------------
        KYCLE1S: JNB P3.5,KYCLE1S
        KYCLE2S: JB P3.5,KYCLE2S
        KYCLE3S: JNB P3.5,KYCLE3S
        KYCLE4S: JB P3.5,KYCLE4S
        KYCLE_ONS: SETB TR0
KYCLE5S: JNB P3.5 , KYCLE6S ;
        SJMP KYCLE_ENDS
KYCLE6S: JNB TF0  , KYCLE5S ;
        INC CARRI
        CLR TF0
        SJMP KYCLE5S
KYCLE_ENDS: CLR TR0
        MOV TOFF1, TH0
        MOV TOFF2, TL0


;-----------------------------------TOTAL TIME PERIOD---------------

   MOV R6,TONE1
   MOV R7,TONE2

   MOV R4,TOFF1
   MOV R5,TOFF2

   LCALL ADD16_16

        MOV TIME2_H, R2 ; TIME PERIOD HIGH
        MOV TIME2_L, R3 ; TIME PERIOD LOW

RET



;--------------------------------PRINTING 1 BIT NUMBER-------------------------

PRINT_DIGIT_16:

        MOV MS1, #0H    ;MSD
        MOV MS2, #0H
        MOV MS3, #0H
        MOV MS4, #0H
        MOV MS5, #0H    ;LSD

        ; Taking out each digit
        ;1
        ;DIVISOR = 10
        MOV R3, #00H
        MOV R2, #0AH
```

```
        LCALL DIV_16BIT
        MOV MS5, 34H    ; LOW BYTE OF REMAINDER -- LSD


        ;2
        MOV R1,23H      ; QOUTIENT HIGH TO DIVIDEND
        MOV R0,24H      ; QOUTIENT LOW  TO DIVIDEND
        ;DIVISOR = 10
        MOV R3, #00H
        MOV R2, #0AH

        LCALL DIV_16BIT
        MOV MS4, 34H    ; LOW BYTE OF REMAINDER



        ;3
        MOV R1,23H      ; QOUTIENT HIGH TO DIVIDEND
        MOV R0,24H      ; QOUTIENT LOW  TO DIVIDEND
        ;DIVISOR = 10
        MOV R3, #00H
        MOV R2, #0AH

        LCALL DIV_16BIT
        MOV MS3, 34H    ; LOW BYTE OF REMAINDER



        ;4
        MOV R1,23H      ; QOUTIENT HIGH TO DIVIDEND
        MOV R0,24H      ; QOUTIENT LOW  TO DIVIDEND
        ;DIVISOR = 10
        MOV R3, #00H
        MOV R2, #0AH

        LCALL DIV_16BIT
        MOV MS2, 34H    ; LOW BYTE OF REMAINDER



        ;5
        MOV MS1, 24H    ; LOW BYTE OF QUOTIENT -- MSD


;---------------------------ASCII CONVERSION--------------------
         CLR A
         MOV A, MS1
         ORL A, #30H
         LCALL DISPLAY


         CLR A
         MOV A, MS2
         ORL A, #30H
         LCALL DISPLAY


         CLR A
         MOV A, MS3
         ORL A, #30H
         LCALL DISPLAY


         CLR A
         MOV A, MS4
         ORL A, #30H
         LCALL DISPLAY


         CLR A
         MOV A, MS5
         ORL A, #30H
         LCALL DISPLAY

         RET
```

```
PRINT_DIGIT:
        MOV MS1, #0H    ;MSD
        MOV MS2, #0H
        MOV MS3, #0H    ;LSD

        MOV B,#10
        DIV AB
        MOV MS3, B      ; LSD
        MOV B,#10
        DIV AB
        MOV MS2, B
        MOV MS1, A      ; MSD

        MOV A, MS1
        ORL A, #30H
        LCALL DISPLAY

        MOV A, MS2
        ORL A, #30H
        LCALL DISPLAY

        MOV A, MS3
        ORL A, #30H
        LCALL DISPLAY
        RET

DISPLAY_LINE1:
        CLR A
        MOVC A,@A+DPTR
        JZ FINISH_1
        LCALL DISPLAY
        LCALL DELAY
        INC DPTR
        LJMP DISPLAY_LINE1
FINISH_1: RET

STR_DISP_LINE2:
        ;MOV A, #0C0H
        ;ACALL COMMAND
        ;ACALL DELAY

LOOP_1: CLR A
        MOVC A, @A+DPTR
        JZ FINISH_2
        LCALL DISPLAY
        LCALL DELAY
        INC DPTR
        LJMP LOOP_1
FINISH_2: RET
```

```
INITLCD:

        ; DISPLAY COMMANDS
        MOV     A, #38H ;init. LCD 2 lines, 5x7 matrix
        LCALL   COMMAND ;call command subroutine
        LCALL   DELAY   ;give LCD some time
        MOV     A, #0EH ;dispplay on, cursor on
        LCALL   COMMAND ;call command subroutine
        LCALL   DELAY   ;give LCD some time
        MOV     A, #01  ;clear LCD
        LCALL   COMMAND ;call command subroutine
        LCALL   DELAY   ;give LCD some time
        MOV     A, #06H ;shift cursor right
        LCALL   COMMAND ;call command subroutine
        LCALL   DELAY   ;give LCD some time
        MOV     A, #80H ;cursor at line 1 postion 1
        LCALL   COMMAND ;call command subroutine
        LCALL   DELAY   ;give LCD some time
        RET

COMMAND:LCALL READY
        MOV P2, A
        CLR RS
        CLR RW
        SETB ENBL
        LCALL DELAY
        CLR ENBL
        RET

DISPLAY: LCALL READY
        MOV P2, A
        SETB RS
        CLR RW
        SETB ENBL
        LCALL DELAY
        CLR ENBL
        RET

READY:  SETB P2.7
        CLR RS
        SETB RW
WAIT:   CLR ENBL
        ACALL DELAY
        SETB ENBL
        JB P2.7, WAIT
        RET
```

```
;------------------------------DELAY 25ms-------------
DELAY:  MOV R3, #50
AGAIN_2:MOV R4, #255
AGAIN:  DJNZ R4, AGAIN
        DJNZ R3, AGAIN_2
        RET

;------------------------------DELAY 1S----------------------
DELAY_1:MOV R5, #45
AGAIN_3N:ACALL DELAY
        DJNZ R5, AGAIN_3N
        RET

;------------------------------DELAY 2.5S-------------------------
DELAY_2:MOV R5, #100
AGAIN_3:ACALL DELAY
        DJNZ R5, AGAIN_3
        RET

;------------------------------DELAY5S----------------------------
DELAY_5S:MOV R5, #180
AGAIN_3G:ACALL DELAY
        DJNZ R5, AGAIN_3G
        RET
;-------------------------ADDITION SUBROUTINE--------------------

 ADD16_16:
        CLR C
    ;Step 1 of the process
    MOV A,R7     ;Move the low-byte into the accumulator
    ADD A,R5     ;Add the second low-byte to the accumulator
    MOV R3,A     ;Move the answer to the low-byte of the result

    ;Step 2 of the process
    MOV A,R6     ;Move the high-byte into the accumulator
    ADDC A,R4    ;Add the second high-byte to the accumulator, plus carry.
    MOV R2,A     ;Move the answer to the high-byte of the result

    ;Step 3 of the process
    MOV A,#00h   ;By default, the highest byte will be zero.
    ADDC A,#00h  ;Add zero, plus carry from step 2.
    MOV R1,A ;Move the answer to the highest byte of  the result

    ;Return - answer now resides in R1, R2, and R3.
    RET
```

```
;--------------------------SUBTRACTION SUBROUTINE---------------------
SUBB16_16:
  ;Step 1 of the process
  MOV A,R7  ;Move the low-byte into the accumulator
  CLR C     ;Always clear carry before first subtraction
  SUBB A,R5 ;Subtract the second low-byte from the accumulator
  MOV R3,A  ;Move the answer to the low-byte of the result

  ;Step 2 of the process
  MOV A,R6  ;Move the high-byte into the accumulator
  SUBB A,R4 ;Subtract the second high-byte from the accumulator
  MOV R2,A  ;Move the answer to the low-byte of the result

  ;Return - answer now resides in R2, and R3.
  RET

;--------------------------DIVISION SUBROUTINE--------------------

DIV_16BIT:
;======== 16 BIT DIVISION SUBROUTINE.

; R3 -- HIGH BIT OF THE DIVISOR
; R2 -- L0W  BIT OF THE DIVISOR

; R1 -- HIGH BIT OF THE DIVIDEND
; R0 -- LOW  BIT OF THE DIVIDEND

; 23h,R3  -- high-byte  quotient
; 24h,R2  -- low-byte   quotient
; 33h,R1  -- high-byte  remainder
; 34h,R0  -- low-byte   remainder

  CLR C       ;Clear carry initially
  MOV R4,#00h ;Clear R4 working variable initially
  MOV R5,#00h ;CLear R5 working variable initially
  MOV R6,#00h
  ;MOV R3,#00h
  ;MOV R2,#10D
  MOV 20h,R1 ;Save high-bit of the dividend
  MOV 21h,R0 ;Save low-bit of the dividend
  MOV 30h,R3 ;;Save high-bit of the divisor
  MOV 31h,R2 ;Save low-bit of the divisor
  MOV B,#00h  ;Clear B since B will count the number of left-shifted bits
```

```
div1:
  INC B      ;Increment counter for each left shift
  MOV A,R2   ;Move the current divisor low byte into the accumulator
  RLC A      ;Shift low-byte left, rotate through carry to apply highest bit to high-byte
  MOV R2,A   ;Save the updated divisor low-byte
  MOV A,R3   ;Move the current divisor high byte into the accumulator
  RLC A      ;Shift high-byte left high, rotating in carry from low-byte
  MOV R3,A   ;Save the updated divisor high-byte
  JNC div1   ;Repeat until carry flag is set from high-byte
div2:        ;Shift right the divisor
  MOV A,R3   ;Move high-byte of divisor into accumulator
  RRC A      ;Rotate high-byte of divisor right and into carry
  MOV R3,A   ;Save updated value of high-byte of divisor
  MOV A,R2   ;Move low-byte of divisor into accumulator
  RRC A      ;Rotate low-byte of divisor right, with carry from high-byte
  MOV R2,A   ;Save updated value of low-byte of divisor
  CLR C      ;Clear carry, we don't need it anymore
  MOV 07h,R1 ;Make a safe copy of the dividend high-byte
  MOV 06h,R0 ;Make a safe copy of the dividend low-byte
  MOV A,R0   ;Move low-byte of dividend into accumulator
  SUBB A,R2  ;Dividend - shifted divisor = result bit (no factor, only 0 or 1)
  MOV R0,A   ;Save updated dividend
  MOV A,R1   ;Move high-byte of dividend into accumulator
  SUBB A,R3  ;Subtract high-byte of divisor (all together 16-bit substraction)
  MOV R1,A   ;Save updated high-byte back in high-byte of divisor
  JNC div3   ;If carry flag is NOT set, result is 1
  MOV R1,07h ;Otherwise result is 0, save copy of divisor to undo subtraction
  MOV R0,06h
div3:
  CPL C      ;Invert carry, so it can be directly copied into result
  MOV A,R4
  RLC A      ;Shift carry flag into temporary result
  MOV R4,A
  MOV A,R5
  RLC A
  MOV R5,A
  DJNZ B,div2 ;Now count backwards and repeat until "B" is zero
  MOV R3,05h  ;Move result to R3/R2
  MOV R2,04h  ;Move result to R3/R2
  MOV R6,#00H
  MOV 23h,R3  ;send high-byte of the quotient in the data memory
  MOV 24h,R2  ;send low-byte of the quotient in the data memory
  MOV 33h,R1  ;send high-byte of the remainder in the data memory
  MOV 34h,R0  ;send low-byte of the remainder in the data memory
  RET
```

```
 ;---------------------------MULTIPLY SUBROUTINE---------------------

MUL16_16:
; First number in R6(H) and R7
; Second number in R4(H) and R5.
; Result oin R0(MSB), R1, R2 and R3(LSB).


;Multiply R5 by R7
MOV A,R5 ;Move the R5 into the Accumulator
MOV B,R7 ;Move R7 into B
MUL AB   ;Multiply the two values
MOV R2,B ;Move B (the high-byte) into R2
MOV R3,A ;Move A (the low-byte) into R3

;Multiply R5 by R6
MOV A,R5    ;Move R5 back into the Accumulator
MOV B,R6    ;Move R6 into B
MUL AB      ;Multiply the two values
ADD A,R2    ;Add the low-byte into the value already in R2
MOV R2,A    ;Move the resulting value back into R2
MOV A,B     ;Move the high-byte into the accumulator
ADDC A,#00h ;Add zero (plus the carry, if any)
MOV R1,A    ;Move the resulting answer into R1
MOV A,#00h  ;Load the accumulator with  zero
ADDC A,#00h ;Add zero (plus the carry, if any)
MOV R0,A    ;Move the resulting answer to R0.

;Multiply R4 by R7
MOV A,R4   ;Move R4 into the Accumulator
MOV B,R7   ;Move R7 into B
MUL AB     ;Multiply the two values
ADD A,R2   ;Add the low-byte into the value already in R2
MOV R2,A   ;Move the resulting value back into R2
MOV A,B    ;Move the high-byte into the accumulator
ADDC A,R1  ;Add the current value of R1 (plus any carry)
MOV R1,A   ;Move the resulting answer into R1.
MOV A,#00h ;Load the accumulator with zero
ADDC A,R0  ;Add the current value of R0 (plus any carry)
MOV R0,A   ;Move the resulting answer to R1.

;Multiply R4 by R6
MOV A,R4  ;Move R4 back into the Accumulator
MOV B,R6  ;Move R6 into B
MUL AB    ;Multiply the two values
ADD A,R1  ;Add the low-byte into the value already in R1
MOV R1,A  ;Move the resulting value back into R1
MOV A,B   ;Move the high-byte into the accumulator
ADDC A,R0 ;Add it to the value already in R0 (plus any carry)
MOV R0,A  ;Move the resulting answer back to R0
```

```
 ;Return - answer is now in R0, R1, R2, and R3
 RET

; Message promts
INTRO:  DB "GROUP 2", 0

HZ:     DB "HZ",0
MSG_1:  DB "FREQUENCY",0
MSG_2:  DB "MEASUREMENT",0
MSG_3:  DB "FREQ 1: ",0
MSG_4:  DB "DIFFERENCE:",0
MSG_5:  DB "FREQ 2: ",0
MSG_6:  DB "ADDITION FREQ: ",0
MSG_7:  DB "DUTY: ",0
MSG_8:  DB "TIME PERIOD 2",0


TIMEOFF: DB "TOFF: ",0
TIMEONE: DB "TONE: ",0



        END
```

Problems Faced & Limitations

- A standard microcontroller typically consists of registers that are 8 bits in size. When employing 8-bit registers, the microcontroller's resolution will be limited to a range of 0Hz to 256Hz, rendering it impractical for most real-world applications. Consequently, we have to increase the frequency range in order to get a more feasible frequency resolution (0-65536Hz).

- Due to the fact that the values were encoded as 16-bit, they were split into two separate 8-bit registers. As a result, all mathematical calculations, including addition, multiplication, and division, had to be carried out using 16-bit manipulation. This is far more complex than the conventional 8-bit calculations that are taught to us.

- DUTY cycle cannot be calculated for every frequency from 1hz to 65536 hz. This is because when the frequency is too low the time period increases as a result. In our project we have taken the on off time period in micro second. As a result, when on/off time exceeds above 65536 micro second the program cannot calculate the duty cycle.

- Similarly, in task2 we have use time period to calculate the new frequency as a result similar problem exists for TASK2.

- While merging task1 and task2 sometime adding a subroutine from task2 to task1 created problems in the whole code. Though that subroutine wasn't used/called by the main file.

- As we have worked with 16bit frequency measurement. For every calculation we had to use multiple registers.

Conclusion

This project enhanced our comprehension of timer and counter operations, as well as our understanding of the process of frequency measurement in the 8051 microcontrollers. We have employed our fundamental theoretical knowledge received from our classes and the book in the successful implementation of the necessary aspects of our project. Specifically, we have successfully quantified the frequency of a series of pulses when fed into our microcontroller and accurately measured the durations of both the ON and OFF states. At last, we have successfully included the necessary additional functionalities. Nevertheless, the implementation of this project proved to be challenging due to significant disparities between the hardware and software components. Notably, the hardware aspect presented a high level of complexity. In the end, we could overcome the problems via the united effort of all our team members. All in all, this was a hard assignment which supported us in converting our academic knowledge into actual scenarios and so reaching a fruitful conclusion.