04/01/24

Experiment-1

Introduction to C-Programming

Task 1:

Write a C program to get familiarized with the data types & its memory allocation & address.

```
#include <stdio.h>
void main(){
    int s = 10;
    float p = 5.6;
    double q = 7.3;
    long r = 100;
    printf("the value of s = %d\n", s);
    printf("the allocated space for s = %d\n", sizeof(s));
    printf("the memory address of s = %d\n", &s);
    printf("the value of p = %f\n", p);
    printf("the allocated space for s = %d\n", sizeof(p));
    printf("the memory address of s = %d\n", &p);
    printf("the value of q = %lf\n", q);
    printf("the allocated space for q = %d\n", sizeof(q));
    printf("the memory address of q = %d\n", &q);
    printf("the value of r = %ld\n", r);
    printf("the allocated space for r = %d\n", sizeof(r));
    printf("the memory address of r = %d\n", &r);
}
```

Task 2:

Find the average of 5 numbers;

```
#include <stdio.h>
void main(){
    int a, b, c, d, e; a = 7; b = 2; c = 3; d = 4; e = 5;
    float avg; sum; sum = a+b+c+d+e;
    avg = (float) sum/5;
    printf("the average is found to be %f", avg);
}
```

Task - 3:

Write a C-program to convert celcius to farenheit & vice versa.

```
#include <stdio.h>
void main(){
    float c, f, c1;
    c = 58;
    f = c x 1.8 + 32;
    c1 = (f - 32) x 0.55;
    printf("Temperature in farenheit = %f\n", f);
    printf("Temperature in celcius = %f", c1);
```

Output:

Task-1:

The value of s = 10
The allocated space for s = 4 bytes
The memory address of s = 6487580
The value of p = 5.600000
The allocated space for p = 4 bytes
The memory address of p = 6487576
The value of q = 7.300000
The allocated space for q = 8 bytes
The memory address of q = 6487568
The value of r = 100
The allocated space for r = 4 bytes
The memory address of r = 6487564

Task-2:

The average is found to be 4.200000

Task-3:

Temperature in farenheit = 136.399994
Temperature in celcius = 57.419998

18/01/24

Experiment - 2

Understanding Control Statements

Task1:

Find the average of four numbers using for loop & by getting four numbers from the user in C-language.

```
#include <stdio.h>
void main(){
    float w, avg, avg1, sum;
    int i;
    for (i=0; i<4; i++){
        scanf("%f", &w);
        sum += w;
        avg1 += (w/4);
    }
    avg = sum/4;
    printf("the avg is %f\n", avg);
    printf("the avg1 is %f\n", avg1);
```

Task 2:

Assign grades according to marks obtained in C-lang.

```
#include <stdio.h>
void main(){
    int mark = 87;
    char grade;
    if (mark >= 90){
        grade = 'S';
    }else if (mark >= 80){
        grade = 'A';
    }else if (mark >= 70){
        grade = 'B';
    }else if (mark >= 60){
```

```
    grade = 'C';
} else if (mark >= 50){
    grade = 'D';
} else {
    grade = 'F';
}
printf("The grade is %c", grade);
}
```

Task: 3

Check whether a given no is a prime or composite.

```
#include <std.io.h>
void main(){
    int num = 10;
    int s, i;
    for (i = 2; i < num; i++){
        if (num % i == 0){
            printf("The no %d is composite", num);
            break;
        } else {
            s = 1
        }
    }
    if (s == ){
        printf("The no %d is prime", num);
    }
}
```

Results:

Task1: The avg is 6.400000
The avg1 is 6.400000

Task2: The Grade is A

Task3: The no 10 is composite

Task4:

Assign the grades according to the marks obtain by using switch case

```
#include <stdio.h>
void main(){
    int mark = 87/10;
    char grade;
    switch(mark){
    case 10
    case 9:
        printf("S
        grade = 'S';
        break;
    case 8:
        grade = 'A';
        break;
    case 7:
        grade = 'B';
        break;
    case 6:
        grade = 'C';
        break;
    case 5:
        grade = 'D';
        break;
    default:
        grade = 'F';
        break;
    }
    printf("Grade = %c", grade);
}
```

Result: Grade = A

20/01/24

Experiment - 3
Arrays & Pointers.

Task1:

Write a C-programm to find the average of 6 nos using 1D array. Take input from user. Greet the user with his name entered in an array.

```
#include <stdio.h>
void main(){
    int a,i;
    float avg;
    int arr[6];
    char name[] = {'p','r','n','r'};
    for(i=0; i<6 ; i++){
        scanf("%d",&a);
        arr[i] =a;
    }
    for(i=0; i<6 ; i++){
        avg = avg + (float) arr[i]/6 ;
    }
    printf(" the avg is %.6f", avg);
    printf("hello ");
    for(i=0; i<6 ;i++){
        printf("%c", name[i]);
    }
}
```

Result: (Ip nos = 2,1,1,0,3,3)

The avg is 1.666667

hello prnv ✓

Task2:

Write a C-program to find the transpose of a matrix. Key in the values of matrix from register no (21BEC1033). 1033 are the four values of the first row of matrix, subsequent columns do a circular shift.

```
#include <stdio.h>
void main(){
    int mat[4][4] = {{1,0,3,3},{3,1,0,3},{3,3,1,0},{0,3,3,1}};
    int i, int j;
    int tran[4][4];
    for(i=0; i<4; i++){
        for(j=0; j<4; j++){
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
    for(i=0; i<4; i++){
        for(j=0; j<4; j++){
            tran[j][i] = mat[i][j];
        }
    }
    printf("\n");
    for(i=0; i<4; i++){
        for(j=0; j<4; j++){
            printf("%d ", tran[i][j]);
        }
        printf("\n");
    }
}
```

Result:

```
1033
3103
3310
0331

1330
0133
3013
3301
```

Task-3:

Write a C program to swap two values using pointers without a temporary variable

```
#include <stdio.h>
void main(){
    int *ptr1;
    int *ptr2;
    //int a = &ptr1;
    //int a =
    int i = 5;
    int j = 10;
    ptr1 = &i;
    ptr2 = &j;
    printf("i = %d  &i = %d\n", i, &i);
    printf("j = %d  &j = %d\n", j, &j);
    *ptr1 = *ptr1 + *ptr2;
    *ptr2 = *ptr1 - *ptr2;
    *ptr1 = *ptr1 - *ptr2;
    printf("After Swapping\n");
    printf("i = %d  &i = %d\n", i, &i);
    printf("j = %d  &j = %d\n", j, &j);
```

Result:

```
i = 5   &i = 6487564
j = 10  &j = 6487560
After Swapping
i = 10  &i = 6487564
j = 5   &j = 6487560.
```

01/02/24

Experiment - 4

Functions

1Q: Analyze the scope of the variables in the given program.

(a)

```
#include <stdio.h>
static int j;
void test (int i);
void main (){
    int k = 10;
    int i;
    for(i=0; i<10; i++){
        test (k);
    }
    printf ("local var to main k = %d, Global var
            j = %d", k, j);
}
void test (int x){
    j++;
    ++x;
    printf ("j = %d, x = %d", j, x);
}
```

Results: j = 1, x = 11; j = 2, x = 11; j = 3, x = 11; j = 4, x = 11;
local var to main k = 10, Global var j = 10
j = 5, x = 11; j = 6, x = 11; j = 7, x = 11; j = 8, x = 11; j = 9, x = 11;
j = 10, x = 11;

(b)

```
#include <stdio.h>
static int j;
void test (int i);
void main (){
    int k = 10;
    int i;
    for (i=0; i<10; i++){
        test (k);
        printf ("loop %d: local var to main
                k = %d, Global var j = %d", i, k, j);
    }
```

```
}
void test (int x){
        j++;
        ++x;
        printf("j=%d, x=%d\n", j, x);
}
```

c.)

```
#include <stdio.h>
static int j;
void scopetest(int *x);
void main(){
    int k=10;
    int i;
    for(i=0; i<10; i++){
        scope test(&k);
        printf("for loop k=%d, for loop j=%d\n", k, j);
    }
}
void scope test (int *x){
    j++;
    ++*x
    printf("j=%d, *x=%d", j, *x);
}
```

Results:

j=1, *x=11
for loop k=11, for loop j=1
j=2, *x=12
for loop k=12, for loop j=2
j=3, *x=13
for loop k=13, for loop j=3
j=4, *x=14
for loop k=14, for loop j=4
j=5, *x=15
for loop k=15, for loop j=5
j=6, *x=16.
for loop k=16, for loop j=6
j=7, *x=17
for loop k=17, for loop j=7
j=8, *x=18
for loop k=18, for loop j=8
j=9, *x=19
for loop k=19 for loop j=9
j=10, *x=20
for loop k=20, for loop j=10.

d.)

```
#include <stdio.h>
static int j;
int k=10;
void test(int i);
void main(){
    int i;
    for(i=0; i<10; i++){
        test(k);
        printf("loop %d: Global var k=%d,
                Global var j=%d\n", i, k, j);
    }
}
void test(int x){
    j++;
    ++x;
    printf("j=%d, x=%d\n", j, x);
}
```

Results:

j=1, x=11
loop 0; Global var k=10; Global var j=1
j=2, x=11

loop1: Global var k = 10, Global var j = 2
j=3, x=11
loop2: Global var k=10, Global var j=3
j=4, x=11
loop3: Global var k=10, Global var j=4
j=5, x=11
loop4: Global var k=10, Global var j=5
j=6, x=11
loop5: Global var k=10, Global var j=6
j=7, x=11
loop6: Global var k=10, Global var j=7
j=8, x=11
loop7: Global var k=10, Global var j=8
j=9, x=11
loop8: Global var k=10, Global var j=9
j=10, x=11
loop9: Global var k=10, Global var j=10

2Q. Write a C-program to print string using own user defined function called printstr(char *a) Print your name in the console.

```
#include <stdio.h>
#include <string.h>
void printstr(char *a);
void main(){
    char name[] = "pranav";
    printstr(name);
}
void printstr(char *a){
    while(*a != '\0'){
        printf("%c", *a);
        a++;
    }
}
```

Result:
pranav.

3Q. Swap two numbers using functions, pointers without a temporary variable.

```
#include <stdio.h>
void swap(int x, int y);
void main(){
    int a = 5;
    int b = 3;
    swap(a, b);
}
void swap(int x, int y){
    int a = x;
    int b = y;
    int *ptr1;
    int *ptr2;
    ptr1 = &i;
    ptr2 = &j;
    printf("Before swapping\n");
    printf("a = %d  &a = %d\n", a, &a);
    printf("b = %d  &b = %d\n", b, &b);
    *ptr1 = *ptr1 + *ptr2;
    *ptr2 = *ptr1 - *ptr2;
    *ptr1 = *ptr1 - *ptr2;
    printf("After swapping\n");
    printf("a = %d  &a = %d\n", a, &a);
    printf("b = %d  &b = %d\n", b, &b);
}
```

Result:
Before swapping
a = 5 &a = 6487500
b = 3 &b = 6487496
After swapping
a = 3 &a = 6487500
b = 5 &b = 6487496

10. b.) Result:

j = 1, x = 11
loop 0: local var to main k = 10, Global var j = 1
j = 2, x = 11
loop 1: local var to main k = 10, Global var j = 2
j = 3, x = 11
loop 2: local var to main k = 10, Global var j = 3
j = 4, x = 11
loop 3: local var to main k = 10, Global var j = 4
j = 5, x = 11
loop 4: local var to main k = 10, Global var j = 5
j = 6, x = 11
loop 5: local var to main k = 10, Global var j = 6
j = 7, x = 11
loop 6: local var to main k = 10, Global var j = 7
j = 8, x = 11
loop 7: local var to main k = 10, Global var j = 8
j = 9, x = 11
loop 8: local var to main k = 10 Global var j = 9
j = 10, x = 11
loop 9: local var to main k = 10, Global var j = 10

22/02/24

Experiment - 5
Introduction to Keil

Basic Inc Program:

```
#include <reg51.h>
void main(){
    unsigned int x;
    for(x=0; x<255; x++){
        P1 = x;
    }
}
```

Results: P2 : 0xFF = 1111 1110

Task 1 : (Visualizing the complementation in P2)

```
#include <reg51.h>
void delay();
void main(){
    for(;;){
        P2 = 0x55;
        delay();
        P2 = 0xAA;
    }
}
void delay(){
    unsigned int x;
    for(x=0; x<255; x++);
}
```

Result:
P2 : 0x55 delay P2 : 0xAA delay P2 : 0x55 delay ....
This continues infinite times.

Task 2:

Monitor $P0.2$ pin & if it is one send AF to P_2 else send 00 to P_2.

```
#include <reg51.h>
sbit mybit = P0^2;
void main (){
    unsigned int x;
    mybit = 1;
    for (x=0; x<3; x++){
        if (mybit == 1){
            P2 = 0xAF;
        } else {
            P2 = 0x00;
        }
    }
}
```

Results:

loop 0: $P0.2 = 1$ $\Rightarrow$ $P_2 = 0xAF = 10101111$
loop 1: $P0.2 = 0$ $\Rightarrow$ $P_2 = 0x00 = 00000000$
loop 2: $P0.2 = 1$ $\Rightarrow$ $P_2 = 0xAF = 10101111$

Task 3:

Monitor P_0 & send P_0 to P_1, Complement P_0 & give it to P_2.

```
#include <reg51.h>
void main (){
    unsigned int x;
    P0 = 0x55;
    for (x=0; x<3; x++){
        P1 = P0;
        P2 = ~P1;
    }
}
```

Results:

loop 0: $P0 = 0x55 \Rightarrow P_1 = 0x55$, $P_2 = 0xAA$.
loop 1: $P0 = 0x5F \Rightarrow P_1 = 0x5F$, $P_2 = 0xA0$.
loop 2:
loop 3:

(10)

B
25/4/24

14/03/24

Experiment - 6
Blinking LED

Task 1:
Blinking LED using simulation

Task 2:
Blinking LED using 8051 kit

Output Verified
14/03/24

Task 1:

```
#include <reg51.h>
sbit LED = P1^0;
void main(){
    unsigned int k;
    while(1){
        LED = 1;
        for(k=0; k<6000; k++);
        LED = 0;
        for(k=0; k<6000; k++);
    }
}
```

Time period = 96.68 - 74.6 = 22.08 ms.

```
#include <reg51.h>
sbit LED = P1^0;
~~void main(){~~
~~    unsigned int k=250;~~
void delay(unsigned int time);
void main(){
    unsigned int k=250;
    while(1){
        LED = 1;
        delay(k);
        LED = 0;
        delay(k);
    }
}

void delay(unsigned int time){
    unsigned int i,j;
    for(i=0; i<time; i++)
        for(j=0; j<1275; j++);
}
```

Output: Time Period = 249.996 ms.

Task 2:

500 ms
71.1 ms

```
#include <reg81.h>
sbit LED = P1^0;
void delay();
void main(){
    LED = 0;
    unsigned int k = 7;
    while(k>0){
        delay();
        k--;
    }
}
void delay(){
    TMOD = 0x00;
    TH0 = 0;
    TL0 = 0;
    TR0 = 1;
    while(TF0==1){
        TR0 = 0;
        TF0 = 0;
    }
}
```

Time Period = 499.78 ms. → Output

```
#include <reg81.h>
sbit led = P1^0;
void delay();
void main(){
    LED = 0;
    unsigned int k = 14;
    while (k > 0){
        delay();
        k--;
    }
}
void delay(){
    TMOD = 0x00;
    TH0 = 0;
    TL0 = 0;
    TR0 = 1;
    while (TF0 == 1){
        TR0 = 0;
        TF0 = 0;
    }
}
```

Output: Time period = 9.99.89 ms.

Inference?

(9)

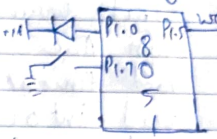
25/11/24

21/10/24

Experiment-7

Interrupt Programming.

TASK1: Connect the led to P1.0 & a switch to P1.7 when switch is turned ON, Make the LED to glow. Generate a wave at P1.5.

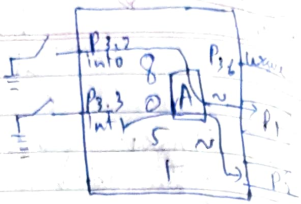



```
#include <reg51.h>
sbit sw = P1^7;
sbit ind = P1^0;
sbit wave = P1^5;
void timer0(void) interrupt 1 {};
void main(){
    SW = 1;
    TMOD = 0x01;
    TL0 = 0xFF;
    TH0 = 0x1A;
    IE = 0x82 //time 0 enabled.
    TR0 = 1;
    while (1){
        ind = sw;
    }
}

void timer0(void) interrupt 1 {
    wave = ~wave;
}   TF0 = 0;
```

Task2: Generate a waveform at P3.6 when the timer 0 interrupt is raising. Whenever interrupt 0 is enabled complement Acc reg & load it to P1 likewise for when int 1 is enabled complement Acc reg & load it in P2.

Task 2:




```
#include <reg51.h>
sbit int0 = P3^2;
sbit int1 = P3^3;
sbit wave = P3^6;
sfr Acc = 0xE0;
void timer0(void) interrupt 1 {};
void int0(void) interrupt 0 {};
void int1(void) interrupt 2 {};
void main(){
    TMOD = 0x01
    int0 = 1;
    int1 = 1;
    IP = 0x02 // for T0 priority.
    //IP = 0x04 for ext_int1 priority.
    TL0 = 0XFF;
    TH0 = 0X1A;
    IE = 0x87;
    TR0 = 1;
    Acc = 0xAA;
    while (1) {
        P0 = Acc;
    }.
}.

void int0(void) interrupt 0{ Acc = ~Acc;
    ~~wave = ~wave~~; P1 = Acc;
}
void int1(void) interrupt 1{ Acc = ~Acc;
    ~~wave = ~wave~~; P2 = Acc;
}.
void timer0(void) interrupt 2{
    wave = ~wave
}.
```

Task 1, 2 o/p verified 28/03/24

(a) 25/4/24 Sequence

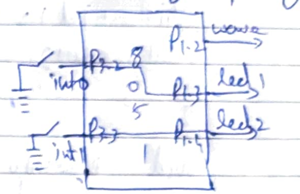
Experiment - 8
Interrupt Programming using Hardware.

1.) Write a C program to toggle P1.2 with delay & toggle P1.3 & P1.4 using INT0 & 1 using default priority [IP = 0x01;]

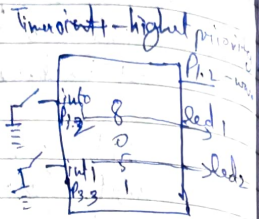
```
# include <reg51.h>

sbit wave = P1^2;
sbit led1 = P1^3;
sbit led2 = P1^4;
sbit int0 = P3^2;
sbit int1 = P3^3;
void main(){
    TMOD = 0x01;
    int0 = 1;
    int1 = 1;
    TL0 = 0xFF;
    TH0 = 0x1A;
    IE = 0x87;
    while (1){
        TR0 = 1;
    }.
}.
void timer0(void) interrupt 1{
    wave = ~wave;
}
void ext_int0(void) interrupt 0{
    led1 = ~led1;
}
void ext_int1(void) interrupt 2{
    led2 = ~led2;
}.
```

int0 - highest priority

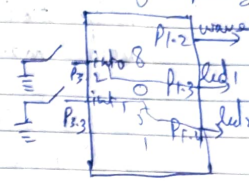


2.) Write a C program to toggle $P_{1.2}$ with delay & toggle $P_{1.3}$ & $P_{1.4}$ using INT0 & INT1 with timer0 given highest priority.




```
#include <reg51.h>
sbit wave = P1^2;
sbit led1 = P1^3;
sbit led2 = P1^4;
sbit int0 = P3^2;
sbit int1 = P3^3;
void timer0(void) interrupt 1 {
        wave = ~wave;
}
void ext_int0(void) interrupt 0 {
        led1 = ~led1;
}
void ext_int1(void) interrupt 2 {
        led2 = ~led2;
}
void main(){
    TMOD = 0x01;
    int0 = 1;
    int1 = 1;
    TL0 = 0xFF;
    TH0 = 0x1A;
    IE = 0x87;
    IP = 0x02;
    while(1){
        TR0 = 1;
    }
}
```

3.) Write a C program to toggle $P_{1.2}$ with delay & toggle on $P_{1^\wedge 3}$ & $P_{1^\wedge 4}$ using INT0 & INT1 with ~~timer0~~ given INT1 given highest priority.




```
#include <reg51.h>
sbit wave = P1^2;
sbit led1 = P1^3;
sbit led2 = P1^4;
sbit int0 = P3^2;
sbit int1 = P3^3;
void timer0(void) interrupt 1 {
        wave = ~wave;
}
void ext_int0(void) interrupt 0 {
        led1 = ~led1;
}
void ext_int1(void) interrupt 2 {
        led2 = ~led2;
}
void main(){
    TMOD = 0x01;
    int0 = 1;
    int1 = 1;
    TL0 = 0xFF;
    TH0 = 0x1A;
    IE = 0x87;
    IP = 0x04;
    while(1){
        TR0 = 1;
    }
}
```

(9) 25/4/24 Inference?

Experiment - 9

Serial Communication & LCD Interfacing.

→ Serially Transmitting first letter of the name with 9600 baud rate.

```
#include <reg51.h>
void tx(unsigned char x);
void main (void){
    TMOD = 0x20
    TH1 = 0xFA
    SCON = 0x50;
    TR1 = 1;
    while (1){
        tx('P').
    }
}

void tx(unsigned char x){
    SBUF = x;
    P2 = SBUF;
    while (TI == 0);
    TI = 0;
}
```

T1, T2 O/P verified 18/4/24

Output: PPPP...

→ Serially Receiving the char at 9600 baud rate.

```
#include <reg51.h>
void main{
    unsigned char mybyte;
    TMOD = 0x20;
    TH1 = 0xFA;
    SCON = 0x50;
    TR1 = 1;
    while (1){
        while (RI == 0);
        mybyte = SBUF;
        P1 = mybyte;
        RI = 0;
    }
}
```

(10) 25/4/24

Output: → Sin='a' → P1 = 61.

→ To display first name on LCD.

```
#include <reg51.h>

void LCD_CMD (unsigned char CMD);
void LCD_DATA (unsigned char DATA);
void Delay_ms (unsigned char j);

sbit RS = P3^7;
sbit RW = P3^6;
sbit EN = P3^5;

void main(){
    P2 = 0x00;
    LCD_CMD (0x01);
    Delay_ms (5);
    LCD_CMD (0x0E);
    Delay_ms (5);
    LCD_CMD (0x38);
    Delay_ms (5)
    LCD_CMD (0x80);
    Delay_ms (5);
    LCD_DATA ('P');
    LCD_DATA ('R');
    LCD_DATA ('A');
    LCD_DATA ('N');
    LCD_DATA ('A');
    LCD_DATA ('V');
    while (1);
}
```

O/P verified 18/4/24

(10) 25/4/24

LCD

Output: PRANAV

25/04/24

Experiment - 11
Stepper Motor.

Make the stepper motor to rotate it in clockwise & anti clockwise direction.

```
#include <reg52.h>
#define ut P0.

void delay (unsigned int count) {
int i, j.
 for (i=0; j< count; i++) {
  for (j=0; j<112; j++);
}
}

int main (void) {
    int i, period;
    period = 100.
    while (1) {
        for (i=0; i<12; i++) {
            ut = 0x09;
            delay (period);
            ut = 0x08;
            delay (period);
            ut = 0x0C;
            delay (period);
            ut = 0x04;
            delay (period);
            ut = 0x06;
            delay (period);
            ut = 0x02;
            delay (period);
            ut = 0x03;
            delay (period);
            ut = 0x01;
            delay (period);
        }
        ut = 0x09;
        delay (period);
        delay (1000);
        for (i=0; i<12; i++) {
            ut = 0x09;
            delay (period);
            ut = 0x03
            delay (period);
            ut = 0x06
            delay (period);
            ut = 0x0C
            delay (period);
        }
        ut = 0x09;
        delay (period);
        delay (1000);
    }
}
```

o/p verified

Result:

The motor rotated clockwise with half step sequence & it rotated in anti clockwise direction with full step sequence.

(10)

B

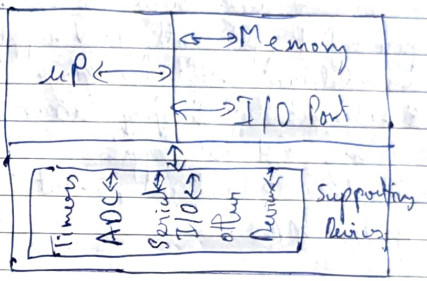
study

* Power Consumption is high	* Power Consumption is less.
* Complex Ckt	* Less complex Ckt.
* Used in personal Computer	* Used in Washing machine, AC.
* Von Neumann Architecture	* Harvard Archi
* Eg: 8086, 8085	Eg: 8051, PIC

→ **μP: (CPU)**

* Brain of μComp.
* Fetch, Decode & Execute.
* 8086 - 16bit μP
* Register, ALU, Timing & control ckt - Intel
* I/O

→ **μC:**



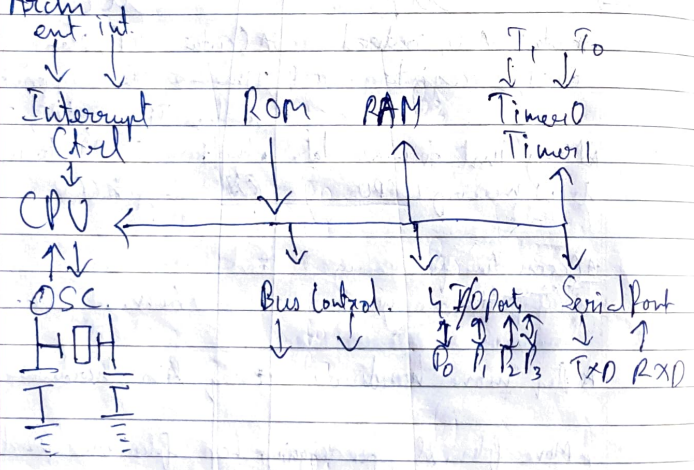
→

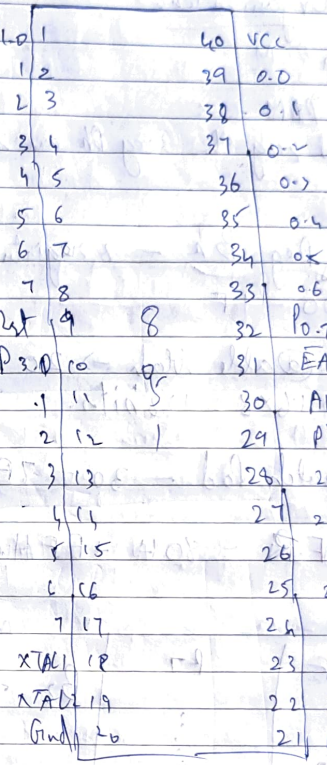
μP	μC
* Contain ALU, control unit & register.	* Contain μP, Memory, I/O port Timer & other supporting chips such as Timers, ADC, on Single Chip.
* Many inst to move data b/w memory & CPU	* Only two inst to move data from memory & CPU.
* Access time for memory & I/O ports are more.	* Less.
* Req more hardware.	* req less hardware.
* More flexible for designing	* Less flexible for design.
* Expensive	* Cheap.
* Data & Prog code is stored in single memory	* seperate memory.

-> Features:

* 8 bit ALU * Data bus.
* 16 bit @ bus
* RAM (Data Memory) - 128 bytes.
* ROM (Prog ") - 4kB - Permanent
* four 8 bit bidirectional I/O port.
* 2 - 16 bit timers/counters.
* 40 pin - DIP IC with +5 V supply.
* 128 user defined software flags.
* 12MHz crystal clock freq for most instructions
* 2 priority level interrupts
* 5 interrupts - 2 ext int, 3 int int.
* 4 register banks.

Archi





1kB = 1024 bytes.

→ Memory Organization:

8051 → ROM
→ RAM - 128

★ Using PSW reg the reg Bk are selected. (RS1, RS0)

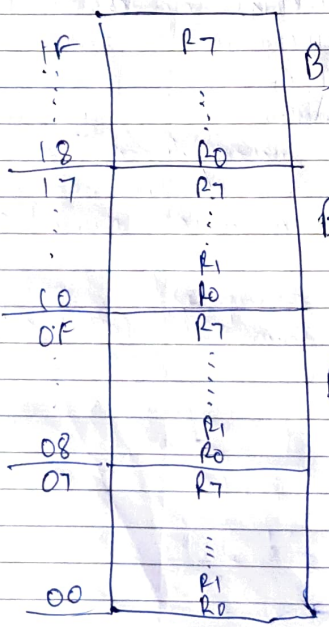
→ RAM:

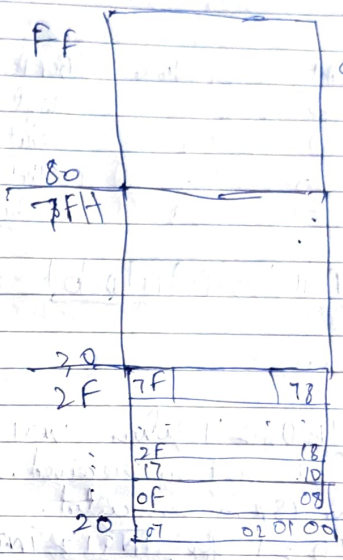
★ 4 Reg Bk – 8 registers R0–R7 in each
00 – 1F

★ Bit addressable block → 20 – 2FH.
↳ 16×8 = 128 bits.

★ Scratch Pad – 30 – 7FH.

★ SFR – 80H – FFH.





→ SFR:

ACC - 8bit
B - 8bit
PSW - [C|AC|F0|RS1|RS0|OV|-|P]
SP - 8bit
DPTR - 16bit (stores the @)
P0|P1|P2|P3 - 80|90|A0|B0
IE - [EA|Res|Res|ES|ET1|EX1|ET0|EX0]
IP - [-|-|PT2|PS|PT1|PX1|PT0|PX0]
TH0/TL0 - T0/T1 - 16bit [Gate|C/T|M1|M0|Gate|C/T|M1|M0] ← 8 bit T1 T0.
TMOD - HH/TL
TCON - [TF1|TR1|TF0|TR0|IE1|IT1|IE0|IT0]
SBUF -
SCON - [SM0|SM1|SM2|REN|TB8|RB8|TI|RI]
PCON - [SM00|-|-|-|GF1|GF0|PD|IDL]
PC - 16bit

SCON:

SM0	SM1	Serial Mode	SBUF Description	Baud rate
0	0	0	Shift Reg	fosc/12
0	1	1	8bit UART	Var
1	0	2	9bit UART	fosc/32 / fosc/64
1	1	3	9bit UART	Var.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

TCON:

* TF1 / TF0 = 1 when INT1 & INT0 is low i.e what int is received. else becomes
* 0 after ISR is executed.
* IT1 / IT0 = 1 for INT1 & INT0 to be -ve edge triggered.
  IT1 / IT0 = 0 for INT1 & INT0 to be level triggered.

PCON:

* SMOD → to modify (then) baud rate is doubled.
* PD → Power down mode when 1
  ↳ 0 " " " is disabled.
* IDL → 0 → idle power mode is disabled
  ↳ 1 → idle power " is enabled.

Anode → To turn on LED → 0

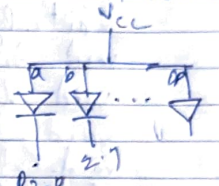
P2 = C0

1100 0000
7 6 5 4 3 2 1 0

0111 0000

1111 1001

P2.0	a	
2.1	b	
2.2	c	
2.3	d	
2.4	e	
2.5	f	
2.6	g	
2.7	DP	



db

→ LCD:

RS = 0 , Cmd mode
RS = 1 → data "

EN = 1 to 0 to send the data to LCD

RW = 0 → write
(1 → read.)

Cmd = 01 → clear.
80 → 1st line
C0 → 2nd line
0C → disp on, cursor off
0E → " " " on
0F → " on, " blink
10 → cursor position shift left
14 → " " → " Right
38 → 8bit mode → 2 line 5x7 matrix
28 → 4bit mode → " "

```
#include <reg51.h>
void lcdn sfr lcd=P2;
sbit EN = P3.5;
sbit RW = P3.6;
sbit RS = P3.7;
void delay (unsigned int d);
void lcd_init();
void lcd_cmd (unsigned char c);
void lcd_data (unsigned char d);
void lcd_string (unsigned char *s);
void main (){

    lcd_init();
    lcd_string("pmv");
    lcd_cmd(0x80);
    lcd_string("tow");

}

lcd_init(){
    lcd_cmd(0x0C|0x0E|0x0F);
    delay(100);
    lcd_cmd(0x38);
    lcddelay();
    lcd_cmd(0x01);
    delay();
    lcd_cmd(0x80);
}

void delay(unsigned int d){
    unsigned int i,j;
    for(i=0;i<d;i++)
        for(j=0;j<922;j++);
}
```

```
void lcd_cmd(unsigned char c){
    lcd = c; //(lcd & 0x0F)|(cmd & 0xF0)
    RS = 0;
    RW = 0;          //(cmd & 0x0F)|(cmd<<4)
    EN = 1;
    delay(10);
    EN = 0;
}

void lcd_data(unsigned char da){
    lcd = da;
    RS = 1;
    RW = 0;
    EN(1);
    delay(10);
    EN(0);
}

void lcdstr(unsigned char *s){
    while(*s){
        lcddata(*s++);
    }
}
```

op port → 0
ip port → 1

→ Keypad:

```
#include <regs.h>

bit P0^4 = R1, P0^5 = R2, P0^6 = R3, P0^7 = R4;
sbit P0^1 = C1, P0^2 = C2, P0^3 = C3, P0^0 = C4;
void keypad();

void main() {
    lcd_init();
    lcd_str("pressing key");
    lcd_cmd(0x80);
    while(1){
        keypad();
    }

void keypad() {
    C1 = C2 = C3 = C4 = 1;
    R1 = 0; R2 = 1; R3 = 1; R4 = 1;
    if(C1 == 0) {
        while(C1 == 0);
        lcd_data('1');
    } else if (C2 == 0){

        lcd_data('2')
```

```
    R1 = 0; R2 = 1; R3 = 1; R4 = 1;
    R3 = 0; R1 = 1; R2 = 1; R3 = 1;
    if(C1 == 0){
        lcd_cmd(0x01);
        lcd_str("");
        lcd_d(0x00);
```

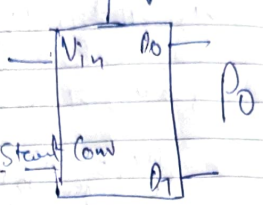
→ Serial Comm:

fosc ÷12 → MC freq ÷12 → UART ÷ TH1 → baud.

baud	TH1
9600	-3 | FD
4800	-6 | FA
2400	-12 | F4
1200	-24 | E8.

```
TMOD = 0x20
SCON = 40|50 → TX|RX
TH1 = -3 / -6 / -12 / -24
TR1 = 1;
while(1)
    SBUF = 'A';
    while(TI == 0);
    TI = 0;
```

→ ADC: (0804)



* Start conv → 0 to 1
* Resolution = $\frac{V_{ref}}{2^n} = \frac{V_{ref}}{2^8}$
* When $\frac{V_{ref}}{2^n}$ is small then it is said to have high resolution.

* P0 req pull up resistors.

Q. T0, 16 bit mod mode: 1
1 : 16
2 : 8bit.

```
#include <regs51.h>
void main(){
    while(1){
        P1 = 0x00;
        delay();
        P1 = 0xFF;
    }
}

void delay(){
    TMOD = 0x01;
    TH0 = 0x00;
    TL0 = 0x00;
    TR0 = 1;
    while(TF0 == 0);
    TR0 = 0;
    TF0 = 0;
}
```

Q. f = 11.0592 MHz.
MC = 921.6 kHz.
T = 1.085 μs
50ms / 1.085μs = 46083
65536 - 46083 = 19453.
TH0 = 0x4B; TL0 = 0xFD;

```
#incl
unit(){
    while(1){
        P2 = 0;
        delay(); for(i=0; i<10; i++){delay();}
        P2 = 1;
    }
}
```

```
delay(){
    TMOD = 0x01;
    TH0 = 4B;
    TL0 = FD;
    TR0 = 1;
    while(TF0 == 0);
    TR0 = 0;
    TF0 = 0;
}
```

sbit sw = P3.2; (into)
after led = P0;

```
void main(){
    sw = 0;
    led = 0xFF; IE = 0x81;
    while(1){
    }led = 0;
}

void into (void) interrupt 0{
    led = 0xFF
}
```

→ PSW - C | AC | F0 | RS1 | RS0 | OV | - | P |

→ IE = | EN | - | - | ET2 | ES | ET1 | EX1 | ET0 | EX0 |

IP = - | - | - | PS | PT2 | PT1 | PX1 | PT0 | PX0 |

TMOD - Gate | C/T | M1 | M0 | Gate | C/T | M1 | M0 |
(T1) (T0)

TCON - TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

SCON - SM0 SM1 SM2 REN TB8 RB8 TI RI

PCON - SMOD - - - GF1 GF0 PD IDL

order 1

WR → ⌐ IC0 → start conv
RD → ⌐ → to read the data
CS → 0
INT → 0 → make a int sig to 8051 stating that A conv is done

8:51

Max $65536 \times 1.085\mu s$ = 71.076ms.

$\frac{65536}{12}$ = 5461.33

$\frac{5ms}{1.085\mu s}$ = 4608.)

$\frac{1.0592}{12} = 921.6kHz$

$6553 6 - 4608$

$1MC = 1.085\mu s$

TH

TL

$0 \rightarrow 1$

$1.085\mu s$.

50ms.

50ms.

$255_n = n/10$; -25

$d_3 = 5 = n/10$

$d_2 = 5 = n/10$

$d_1 = 2 = n/10$

Data (d_1 + 0x30).

SCON = 50 -> Tx & Rx

SCON = 40 -> Tx.

TMOD = 0x20

SBUF

$T_5 = 32$ $1 = -3n - 25$.

FD -3

FA -6

+2

-24

$BR = \frac{2^{SMOD}}{32} \times \frac{f_{OSC}}{12(255 - TH1)}$

```
void cm(unsigned char x){
    rs=0;
    rw=0;
    P2=x;
    en=1;
    delay
    en=0;
}
void data(unsigned char y){
    rs=1;
    rw=0;
    en=1; P2=y;
    delay;
    en=0;
}
void lcd init(){
    cmd(0x38);
    delay
    cm(0x0C);
    delay
    cmd(0x01);
    cmd(0x80);
    cm(0xC0);
}
void str(unsigned char *s){
    while(*s!='\0'){
        data(*s);
        s++
    }
}
```

Keypad:

Col → ip port. → Pull up
↳ connect to Vcc
Row → op port. → zero.

Duty Cyc = 30% → ON - 30%, OFF - 70%

Stepper Motor. → PWM signal.

TH0, TL0.

LCD

RW = 0. 0x01
0x80 → first line
0xC0 → 2nd line.
0x38 → 8 bit 5x7 2 line
0x28 → 4 bit 5x7 - 2 line

LCD initial:
0x38 → 8 bit data.
0x0C → disp on, cursor off.
0x01 → clear
0x80 → 1st
0xC0 → 2nd

VSS - gnd.
VDD - supply

```
    lcd cmd (0x80);
    lcd dat ('A');
    delay (200);
    lcd dat ('C');
}.

void delay (int n){
    int j=0;
    for(j=0;j++
```

LCD, Serial Comm.

→ ADC:

8 bit → 256 levels.

$\frac{5}{256} = 0.019V = 19mV$

$\frac{25}{255}$ = conv

$\frac{V_{ref}}{2}$ → has to be given as ip.

(S) → Minimum.

★ ISR → no need to do config. only in main.

ISR → can start the timer.
↳ TF gets cleared.

LCD Interfacing:

14 pins → V_{cc}, gnd, 8 - data pins.
R/W, EN, RS.

R/W → 0 as LCD is a op device.
en → high to low has to be given to latch the data into the device.
RS →
reserve select.

7 segment / LCD. not same.

fixed: $P_{3.5}$ → RS P_1 → Data DB0 to 7 of LCD
$P_{3.6}$ → en
$P_{3.7}$ → R/W
bit

1×16
2×16
4×16.

R/W → always grded.

```
#include <regs51.h>

sbit Rs = P3^5
sbit en = P3^6
sbit RW = P3^7.
sfr dat = P1;
void lcd_cmd (unsigned char );
void lcddat (unsigned char );
void lcdini ();
void delay (unsigned int n);
void main (){ RS=1, en=1, RW=0;
    lcdini ();
    lcd cmd (0x01);
```

PCON = PCON | 0x80;

RX:

```
TMOD = 0x20;
TH1 = -3, -6, -12, -24;
SCON = 50;
TR1 = 1;
while (1){

    while (RI==0);
→   var = SBUF;
    P1 = var;
    RI = 0
}
```

TH1	SMOD=0	SMOD=1
FD (-3)	9600	19200
FA (-6)	4800	9600
F4 (-12)	2400	4800
E8 (-24)	1200	2400

Timer:

```
TMOD = 0x01; //16 bit timer.
TH0 =
TL0 =
TR0 = 1;
while (TF0==0);
TR0 = 0;
TF0 = 0;
```

Calculat:

$$\frac{11.0592\ MHz}{12} = 921kHz$$

$$\frac{1}{921kHz} = 1.085\ \mu s$$

$$\frac{delay}{1.085} = x$$

↓ convert to hexa.

Serial Comm:

→ TX:

```
TMOD = 0x20;                    // 11.0592 MHz / 12 = 921.6 kHz
TH1 = FD/FA/F4/E8;              // 9600, 4800, 2400, 1200
SCON = 50/40; → TX & RX / RX.
TR1 = 1;
while (1){
    SBUF = 'Y';
    while (TI==0);
    TI = 0;
    SBUF = 'E';
    while (TI==0);
    TI = 0;
    SBUF = 'S';
    while (TI==0);
    TI = 0;
}
```

256 - 185 = 71; → TH0.

Q. Write a prog to Tx your first name which is stored at 8500h in ROM location at 24 bit baud rate using int mode.
str, chr, bit →
print statement ×
No generic lib. only reg51.h
Schematic.
void fun only.
Timer prog → cal
8051 archi;

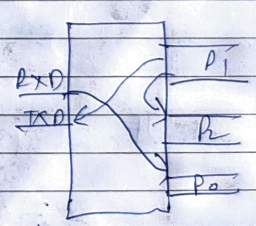
Void main (void){
SE = 0x90;
SCON = 0x50;
TMOD = 0x20;
TH1 = -3;
TR1 = 1;

}.

Void serint (void) Interrupt 4{
if (RI == 0){
RI = 0;
ACC = SBUF;
P1 = ACC;
}.

Q. Write c prog
a.) Rx data serially & send it to P0.
b.) Read P1, Tx serially & give a copy to P1.
c.) Make a T0 to gen a sq wave of 5 Hz freq on P2.6
4800 bps, XTAL = 11.0592 MHz.

void main(void){
TMOD = 0x22;
SCON = 0x50;
TH1 = -6;
TH0 = -71;



Timer 0 → Mode 2
Timer 1 → Mode 2
1/5h = 200m
200m / 1.085u ≈ 184.3

```
void main(){  A='A';
  SCON = 0x50;
  TMOD = 0x20;
  TH1 = -6;
  SBUF = A;
  while (TI==0);
  TI = 0;
  while (1){
    SBUF = A;
    while (TI==0);
    TI = 0;
  }
```

TMOD: | G | C/T | 1 | 0 |

sfr A = 0xE0;

TI => set when data is sent out

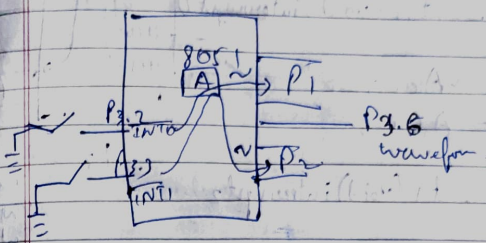
→ Rx data & send it through P1.

```
SCON = 0x50;
TMOD = 0x20;
TH1 = -3;
PCON = 1
ACC = PCON
ACC^7 = 1;
PCON = ACC;
TH1 = -3;
TR1 = 1;
while(1){ while (RI==0);
  ACC = SBUF;
  P1 = ACC;
  RI = 0;
}
```

IE = | GB | - | - | Serial 0 | Timer 1 | Int 1 | T0 | INT0 |

IE = 0x90;

```
TR0 = 1;
while (1){
  IN0 = SW
}
```



→ Priority to INT1 using IP Reg.

IP = 02 → Priority to T0.
IP = 04 → " " int 1.
IP = 0x03; → int 1

→ Timer, Serial Comm, Int :- Cat-2.

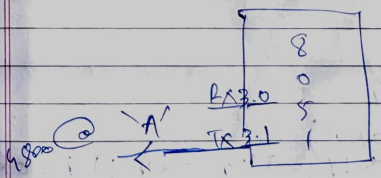
Timer 1, mode 2 → serial comm.

SCON - 40 for Tx only
SCON - 50 for " & Rx.

T ÷12 → T ÷32 → Serial Com.

SMOD = 1 → then ÷16 → 57600 Hz
SMOD = 0 → ÷32 → 28800 Hz

Q: Send 'A' serially with 4800 baudrate.



ISR → void.

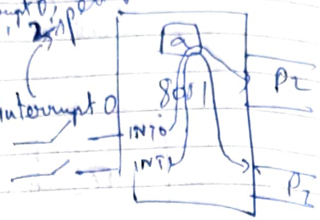
RETI → IE0,1 will be zero ... the ISR

sbit IF... ACC: sbit Int0=P3.2 Int1=P3.3 ... interrupt 0; ... specifies

void int_ent0(void) interrupt 0

Q.

```
#include <reg51.h>
void int_ent0 (void) interrupt 0
{
    Acc = ~Acc;
    P2 = Acc;
}
void int_ent1 (void) interrupt 2
{
    Acc = ~Acc;
    P1 = Acc;
}
void main (void) {
    Int0 = 1;
    Int1 = 1;     } → assign as ip port.
    IE = 0x85;
    ACC = 0xAA;
}.
```

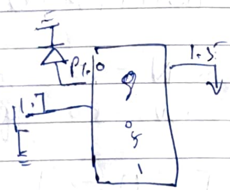


P3 → RxD P3.0
TxD P3.1
INT0
Int1 3
T0 3
T1 3
RD
WR 3

Counter

Q.

```
#include <reg51.h>
~~void int~~
sbit sw = P1.7;
 "   ind = P1.0;
     wave = P1.5;
void timer0(void) interrupt 1 {
     Wave = ~Wave.
}.
void main (){
     SW=1;
     //ind = 0;
     //wave = 0;
     TMOD = 0x01;
     TL0 = 0xFF; TH0 = 0x1A; IE = 0x82
```



4800 baud rate, Tx "YES"

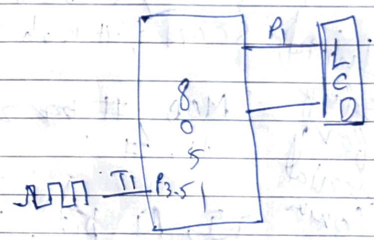
```
MOV SCON, #40h          → Mov R0, #03h.
MOV TMOD, #20h
MOV TH1, -6.  (Mov TH1, #FAh)
MOV DPTR, #200h         → SetB TR1
loop1: MOVC A, @a+DPTR
       MOV SBUF, a      → INC DPTR → clr A, DJNZ R0, loop1
loop:  JNB TI, loop
       clr TI
Org 0200h.
DB ~~DPTR~~, "YES"
```

→ Rx 2 data & store in 60h, 61h.

```
SCON, #50h.
MOV R0, #2.
```

→ Interrupt:

```
sbit T1 = P3^5;
void main
   T1=1;
   TMOD = 0x60
   TH1 = 0;
   while (1)
   {
   do{ TR1 = 1;
       P1 = TL1;
   }
     while (TF1==0){
       TR1 = 0;
       T = 0;
   }.
```



interrupt	
0	ext int
1	T0
2	Ext int
3	T1
4	

MAX 32 pin hex to there for serial comm.

~~92~~ 11.6 MHz → Timer clk

÷32 → 29900 Hz → UART clk.

Config TMOD ÷12 → 57600 Hz

Mode 2

* Timers will be used in UART/serial.

28800 Hz → 34.7 µs.

57600 Hz → 104 µs.

TH1, -3 = TH1, FDH → 9600 bd

→ Serial Control Register (SCON).

Ren → rec enable. for Rxing.

Initial setup for serial comm

SCON, #50h

SCON, #40h → for Txing.

TMOD, #20

TH1, # → baud rate.

TI, RI bit in SCON.

Serial Buffer (SBUF Register)

Mov SBUF, #05h } Txing.

loop: JNB TI, loop;

Mov SBUF, #03h } Rxing.

loop: JNB RI, loop

LAB 6

Logic Analyzer → setup → add → P1 → ok

i < 250

j < 1275

Pin 13 - P1.0

14 - P1.1

26 - Gnd

25 - Vcc

COM11

Target → freq = 11.0592

10 chip Rom

↑ XRAM

↑ DPTR

Delay (250):

75.66 ms - 64.92 ms.

10.74 ms

for loop ();

96.68 - 74.6

22.08 ms.

386.6096

386.6203

0.0107

Serial Communication.

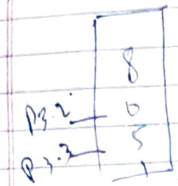
UART → 12V → (TS, RTS

RS 232 protocol RXD, TXD pin

RS 422 → " → differential protocol.

RXD, TXD → digital pins. 5V.

Signal gen using IE0, IE1



on click of button 1 generate a waveform 1kHz of square pulse

on click of button 2 gen a ~~wavefor~~ sawtooth pulse.

```
void Sawtooth(){
   while(1){
      for(i=0; i<256; i++)
         P2=1
      for(i=255; i>0; i--)
         P2=i
   }
}
```

```
#in
sbit   button1   P3^2;
 "     button2   P3^3;
 "     sect_wf   P1^r;
void delay(int);
void click on button1 (void) { interrupt 0 {

}

void click on button2 (void interrupt 2 {
   delay()
   sect_wf = ~ sect_wf;
   delay();
}
void main(){
   IE = 0x85;
   TMOD =        ;
```

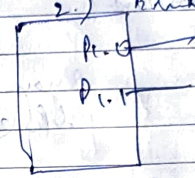
P3.2 -> INT0.

main prg must start @ 30h.
IP, #04h -> setting priority of IE1
0 1 2 3 4
IE0, T0, IE1, T1, Serial com

```
void click on button click (void){
   interrupt 0 -> IE0  (int 1 -> T0, int 2 -> IE1)
   upto p2-1
```

Lab:

1) Simulation of Timer prog in C.
Blink an LED connected at P1.0 using delay.
2.) Blink a LED at P1.0 " "




```
#include <reg8051.h>
sbit led P1^0;
void main(){
   ~~led = 1~~  int i;
   while (1){
      led=1;
   }
```

To delay() ← for(i=0; i<4000; i++){
   led = 0;
}.

1sec / 500ms → delay.
33MHz → 11.0592 MHz
change the freq. (stimulation & hardware freq same)

load TMOD, TH0, TL0

```
void T0 delay () {
   TH0 =
   TL0 =
   TR0 = 1
   while (TF0==0){
      TR0 = 0;
      TF0 = 0;
   }
```

clearing of TF0 will be done automatically.

→ Interrupt enable register:

Global interrupt:

MOV 2E, #81h

* b/w 00 to 30 we must not write a code it is specially preserved for ISR coding.

favours embedded C prog errors.?

ext int 0	3h	8 mem locations
Tim 0 ovrflw	8h	bfr @
ext int 1	13h	
Tim 1 ovrflw	18h	
Serial	23h	
Tim 2 ovrflw	2bh.	

→ Default priority (high to low)

ext int 0
Tim int 0
ext int 1
Tim int 1
Serial Comm.

→ ISR → 8 bit byte. Int source Routine.
return on int → reti

Stack → LIFO. in 8051.

* 64kB - program memory.
* 128 bytes - data "
* 4 Register Banks.
* 128 user define software flags.
* 8051 uC - 12MHz crystal → this produces clk cycles.
* 32 bidir I/O pins - 4 (8bit) ports.
* 16 bit Timers/Counters - 2
* Dual in line package
* +5V power supply.

→ Port 0: AD0 - AD7 | Port 1: $P_{1.0}$ to $P_{1.7}$

Port 2: AD8 - AD15 | Port 3: Tx & Rx, WR, RD, INT0 & 1, T0 & T1

RST → Flags will become zero. (or Register)
↳ shd be high for 2 machine cycle

* PSEN & EA must be low. for accessing ext memory

ALE → when 0 → Data is valid.
1 → Address is enabled

12/03/24. Int prog:
→ INT0 → ext 0
ex 1 → ex 7

* Polling:

Tim 0, 1; ext int 0, 1; Serial int - 5 interrupts.

2 level priority.
IP registers
IE "

Acall → PC address will be pushed to stack & PC will be loaded with branch @.

* Mode 1 → FFFF = 65536.

10
65536 → FFFF

* Mode - 0 → 3F FF
  ↓
  3FFF → 8912

* Mode 2 → FF
  256 count

Mode - 1 → FFFF

* IT1 & IT0 are 1 → edge triggered
  IT1 & IT0 are 0 → level

→ TCON - instruction.
select mode, load value to reg, start timer, poll check the flag.

Q: $\frac{1}{100} = 0.2$ ms → waveform time period.

clk freq → 11.0592 MHz → $\frac{1}{11.0592M} = 0.0904$ µs.

1 M C = 12 × 0.0904 = 1.085 µs.

$\frac{2 m}{1.085 \mu s} = 1843.2$ cycles.

$\frac{1m}{1.085 ms} = 921.6 \simeq 922$ cycles.

TMOD - #01H
TH0 , #0FCH
TL0 , #66

Mode 0, 1 → we have to reload the TL, TH reg after reaching the max value (FFFF, 3FFF).

Liked - 2 - 6 classmate
Fri → 2 - 6 Date Page

* Reviews - 50%. Simulation & Hardware.
  ↳ last week of march. (24th 28th March last date).
* March 20th guest lecture. extra curricular hour
  ↳ Vishnu Employee (TOT ABI) 11:40 am

→ Timer: 2 tim - 16 bits — TH0 TH1
TCON → TR0 bit — TL0 TL1

Mode 0 → 13 bit
1 → 16 bit
2 → 8 bit auto reload mode.
3 →

TMOD - Register:

MSB					LSB			
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Timer 1				Timer 0			

C/T = 1 → counter
Gate → for starting the timer. → Hardware start (or)
TCON → TR0 is set the timer will start → software start.

* 12 clock cycle = 1 Machine cycle.

MOV TMOD, #20H → timer 1 2nd mode.
" " , #12H → timer 1 1st mode, timer 0 mode 2.

→ TCON - Reg. (interrupt)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

every 12 clock cycles the nth TL0 increments by 1.

TH0	TL0
FF	FD
	FE
	FF

When it reaches FF FF → TF0 = 1
TF0 will be 1 (rollover)

Q: If door is open store the data in bit @ location.
"

bit mybit;
"
:
{
mybit = 1
buz = 1;
}
"

Q: Send a seq of no from 1 to 10 to port 1
Q: Send ur name to port 2

```
#include <reg51.h>
void main (){
    char x;
    for (x=1; x<11; x++){
        P1 = x;
    }.
}.
```

sbit → input reg bit. (reg / port / @).

Logic Analyzer

>, |, ^, ~

code z = 0x10; → z will be stored in 4k memory code mei else it will be stored in 128 byte RAM.

* LSB occur front → Little indian
* MSB " " → Big indian.

MC

char ⟶ 8 bit signed. -128 to 127. / 0 to 255 unsigned
int z; ⟶ 16 bit unsigned - 0 to 65k
signed - -32k to 32k

* can't use C lib generic, bcz it consume huge space

```
#include <reg51.h>
void main (){
unsigned int x; unsigned char temp;
    for (x=0; x<50000; x++){
        temp = P1;
    }  P2 = temp;
    }
}
```

P_1 to P_2 50k times

```
#include <reg51.h>
void main (){
    char i; char temp;
    for (i=0; i<50; i++){
        temp = P1;
        P2 = temp;
    }
}
```

Q: Write C prog to monitor door sensor if door opens send buzzer signal.

```
#include <reg51.h>.
sbit door = P0^1; buz = P2^1;
void main (){
    if (door == 1){
        buz = 1;
    } else {
        buz = 0;
    }
}
```

```
door = 1; // ip port
if (door == 1) {
    buz = 1;
} else {
    buz = 0;
}
```

Not needed

LAB5.

8051

```
#include <reg51.h>
```

data types: bit, sbit, sfr, sfr16.
1 1 8 16 - bits.

* sbit LED P0^1 → telling the port.
  sfr → used for spl reg. → Reg51.h
* SFT TMOD.

PC, DPTR - 16 bit.

* unsigned char - 1 byte 0 to 255.

P2 = 0x55; → P2 is there in reg51.h library.

1. Keil → new proj → select device for target → 8051 EDU
2. Copy startup → No
3. Create a new file save with '.c' ext.
4. type the code save it.
5. Rig click source group1 → Add existing files to Group Source Grp1 → add the saved 'c' file
6. Project → Build target (F7) check build op window such that there are no errors.
7. Project → options for target → Debug → Stimulation
8. Click debug session. → debug environment
9. Step by step on.
   Peripherals → I/O ports → P1 → to view the inc in P1
   x = R7.

1. Monitor P0.2 if it is one send FF to P2.
2. Monitor P0. & send the P0 to P1, complement P0 & give it to P2.

```
while(*st != '\0'){
    printf("%c", *st);
    st++;
}
printf("%s", st);
```

* <math.h>
  2, 5, 9, 13, 17.

Q4: Q.

```
#include <stdio.h>
void main(){
    int no = 7;
    if (no & 1){
        printf("%d is odd", no);
    } else {
        printf("%d is even", no);
    }
```

~~if (no >= 1)~~

// 170 lb → 4.71/pound.

1Q. h − 6 ; c − 3.25 ; 6×h + 3.25 × c = 170×4.71 ; 6×(170 − c) + 3.25×c =

2Q. $w = 5 + h$. $\sqrt{4^2 + w^2} = 12$

3Q. (10,10) (11,16) (11,20) → find the area using matrix eqn.
t, w

4Q.

Eg: int *ptr = k;

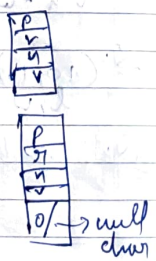
```
#include <stdio.h>
void test (int *x);
void main(){
    int k=10;
    int i;
    for(i=0; i<10; i++){
        test(&k);
        printf("%d, %d", k, j);
    }
}
void test (int *x){
    j++;
    ++*x;
    printf("inside fn %d, %d\n", j, *x);
}
```

Eg:

```
static int j;
int k=10;
void test();
void main(){
    int i;
    for(){
        test();
        printf( );
    }
```

→ ~~Parsing~~ String:

```
#include <string.h>
char ch[] = "pnv";
string st[] = "pnv";
```



fns: gets, puts, strlen()

* strlen(name of st) = j → j=5.

① header file include.
② prototype of fns / Signature
int area (int, int); → passing parameters
↳ must not be a keyword. = fn name
↳ Return type.

```
③ void main(){
    int w, L, a;
    scanf("ent L & W %d, %d", &W, &L);
    a = area(w, L);
    printf("area = %d", a);
}
int area(int a, int b){
    int ar;  → ar is a local var
    ar = a*b;   can't be accessed in
    return ar;   main() fn. so that
}   only we are returning
    ar at last.
```

Eg:

```
#include <stdio.h>
static int j;  → (global var → heap)
void test(int i);
void main(){  → stack.
int k=10;  → loc var to main fn
for(i=0; i<10; i++)
    test(k);
printf(k, j);
printf(k, j);
}
void test(int x){
    j++;
    ++x;
}
```

key words

pnv 10, 10

* static int j; → j=0, not a garbage value.
* for(; ;) { task } = for( ) task.
* for(;;){ } This loop runs ∞
* while(true){ }

*a	2	a[0]
*(a+1)	4	a[1]
*(a+2)	5	a[2]
*(a+3)	-8	a[3]
*(a+4)	6	a[4]

```
int a[] = {2, 4, 5, -8, 6};
printf("%d, %d, %d", a[0], *(a+1), (*a)+1);
```

a[1] ≠ a[4] = 6

a is holding a address → 2 → 4 → 2+1 = 3

```
int b[5] = {1, 3, 4, 5, -3};
int i;
for (i=0; i<5; i++){
    printf("%d, %d, %d, %d", b[i], *(b+i), (*b)+1, *(3+i));
}
```

Op: i=0, 1, 1, 2, 6 | i=1 3, 3, 2, 6

```
int a[5] = {1,3,4,5,-3};
int i;
for (i=4; i>0; i--){
    printf(a[i], *(a-2), a[i--], a[--i])
}
```

i=4 → 5, garbage, 3, -3, 4 | j=1 3, garbage, 3, -3

★ Feb 8 → Quiz
↳ Proj Title

→ Functions:
↳ we are used to reuse, there by reducing the no. of lines.

library function → printf, scanf } → are there
→ gets, puts } → string.lib in 'stdio.h' lib.
→ stdlib.h → malloc, calloc, free
↳ conio.h.

```
int mk[6], sum;
for (i=0; i<4; i++){
    Scanf("%d", &mark[i]);
    sum = sum + mark[i];
}
```

★ Array → uniform data types can only be stored.

```
char a[] = {'A', 'B'};
```

String s = "pnrv"; for this complier stores '\0' at last / end

s: p r n v \0 | Array: p r n v

→ 2D-Arrays.

```
int mat[2][3];   2×3 matrix.
for (int i=0; i<2; i++){
    for (j=0; j<3; j++){
        mat[i][j] = Scanf("%d", &mat[i][j]);
    }
}
```

mat = {1,0,2,4; 4,1,0,2;

→ Pointers: → is a memory address which holds integers

```
int i;
int *ptr;   → pointer initialization.
// i = *ptr;   -> i = value stored @ ptr address.
printf("%d", *ptr);   -> prints the value stored in ptr add
printf("%d", i);
i = 5;
ptr = &i;
printf("%d, %d, %d", i, ptr, *ptr);
```

i: 5 (8000, 16 bit add) | ptr: 8000

op = 5, 8000, 5

(ptr+1) → = *(8004) ≠ *(8001)

printf("%d → only for int data type.

Server → Services → email → domain Name:
(@ cn.(com)
→ user & password, now press '+'

Now go to PC0 & goto desktop → mail → Give the email id

Incoming mail server = IP of server.
Out " " " = " " "
user & pass → save.

Compose

Note IP..

C & Embedded C

→ Arrays: Typecasting:

* int num, num;
float avg;
if $\frac{Sum}{num} = 43.5$
avg = $\frac{sum}{num}$; → only int value will be stored. → 43
avg = (float) avg. → stores decimal value to avg. → 43.5
avg = (float) sum →
char c = 'A';
int i = (int) c++; → ASCII value will be incre
int j = 60;
printf("%d", j++) → 60
int k = 60;
printf("%d", ++k) → 61

→ Array: → int mk[4]; /int mark[] = {1, 2, 3};
→ 16 bytes. = 4 × 4 int.

array size can't be changed.