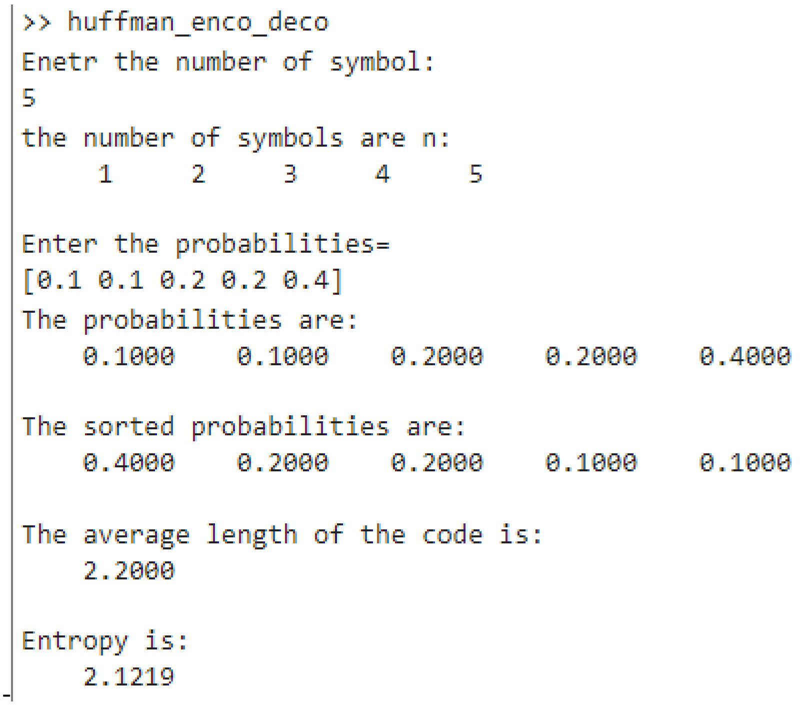
Input:

```
x=input('Enter the number of symbol:');
N=1:x;
disp('the number of symbols are n:');
disp(N);
P=input('Enter the probabilities=');
disp('The probabilities are:');
disp(P);
S=sort(P,'descend');
disp('The sorted probabilities are:');
disp(S);
[dict,avglen]=huffmandict(N,S);
disp('The average length of the code is:');
disp(avglen);
H=0;
for i=1:x
    H=H+(P(i)*log2(1/P(i)));
end
disp('Entropy is:');
disp(H);
disp('bits/msg');
E=(H/avglen)*100;
disp('Efficiency is:');
disp(E);
codeword=huffmanenco(N,dict);
disp('The codewords are:');
disp(codeword);
decode=huffmandeco(codeword,dict);
disp('Decoded output is:');
disp(decode);
```

Output:



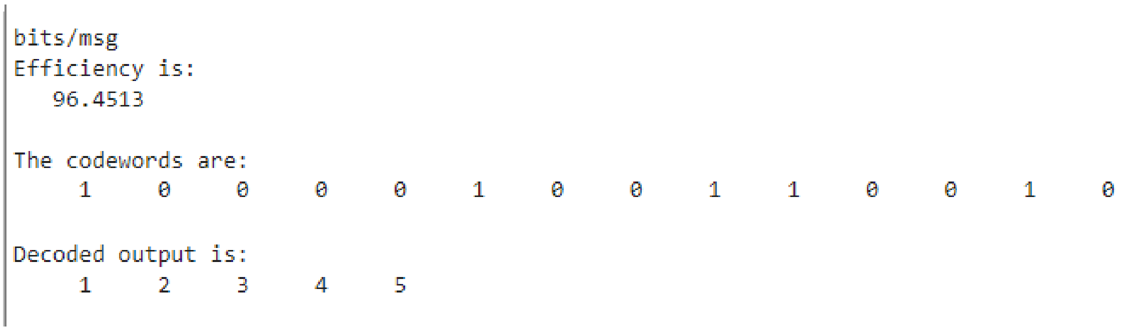
```
>> huffman_enco_deco
Enetr the number of symbol:
5
the number of symbols are n:
     1     2     3     4     5

Enter the probabilities=
[0.1 0.1 0.2 0.2 0.4]
The probabilities are:
    0.1000    0.1000    0.2000    0.2000    0.4000

The sorted probabilities are:
    0.4000    0.2000    0.2000    0.1000    0.1000

The average length of the code is:
    2.2000

Entropy is:
    2.1219
```



```
bits/msg
Efficiency is:
   96.4513

The codewords are:
     1     0     0     0     0     1     0     0     1     1     0     0     1     0

Decoded output is:
     1     2     3     4     5
```

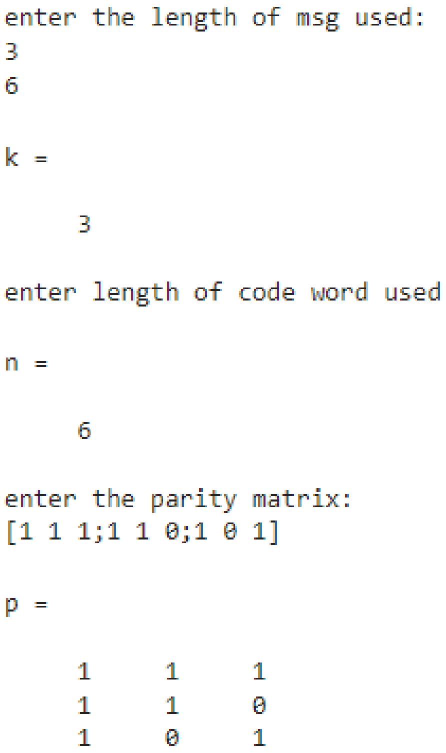
Input:

```
clc
clear all
k = input ('enter the length of msg used: ')
n = input ('enter length of code word used')
p = input ('enter the parity matrix: ')
G = [eye(k) p]
m = input ('enter message word ')
c = encode(m,n,k,'linear',G)

D= decode(c,n,k,'linear',G)
H= [p' eye(n-k)]
R= input ('enter the received code word')

s = rem(R*H',2)
```

Output:



```
enter the length of msg used:
3
6

k =

     3

enter length of code word used

n =

     6

enter the parity matrix:
[1 1 1;1 1 0;1 0 1]

p =

     1     1     1
     1     1     0
     1     0     1
```

```
G =

     1     0     0     1     1     1
     0     1     0     1     1     0
     0     0     1     1     0     1

enter message word
[1 1 1]

m =

     1     1     1


c =

     1     1     1     1     0     0

Single-error patterns loaded in decoding table.  1 rows remaining.
2-error patterns loaded.  0 rows remaining.

D =

     1     1     1


H =

     1     1     1     1     0     0
     1     1     0     0     1     0
     1     0     1     0     0     1

enter the recieved code word
[1 0 1 1 0 0]

R =

     1     0     1     1     0     0


s =

     1     1     0
```

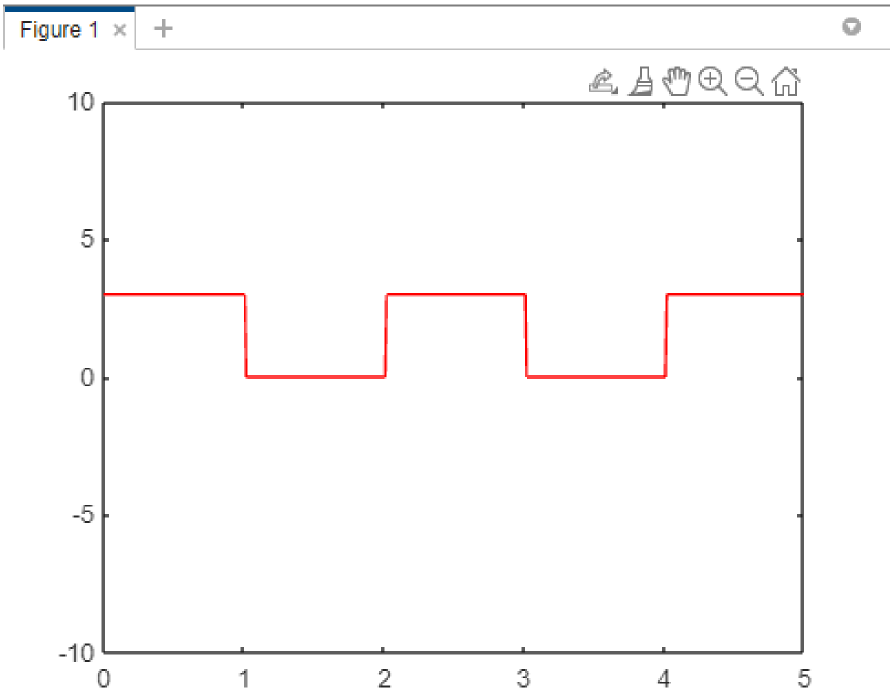
Unipolar NRZ

Input:

```
clc
clear all;
n = input('Enter the input bit:')
n = [1,0,1,0,1];
%mapping 1 = 3V AND 0 = 0V
for ii = 1: length(n)
    if n (ii) == 1;
        nn(ii) = 3;
    else nn(ii) = 0;
    end
end

%pulse shapping
i = 1;
t = 0:0.01 :length(n);
for j = 1 : length(t)
    if t(j) <= i;
        y(j)=nn(i);
    else
        y(j) = nn(i)
        i = i+1;
    end
end
plot(t,y,'-r')
axis([0 length(n) -10 10])
```

Output:

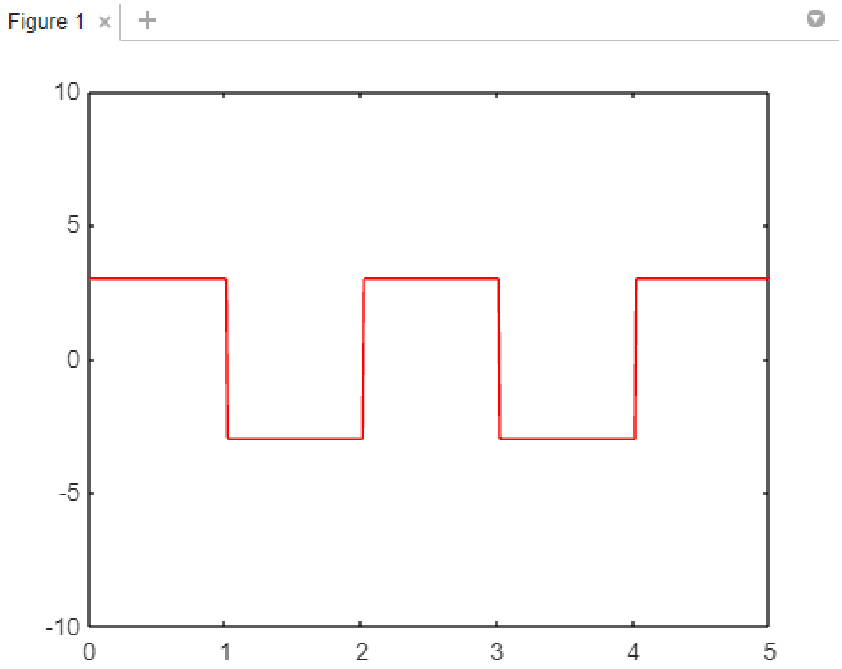


Polar NRZ

Input:

```
clc
clear all;
n = input('Enter the input bit:')
n = [1,0,1,0,1]
%mapping 1 = 3V AND 0 = 0V
for ii = 1:length(n)
    if n (ii) == 1;
        nn(ii) = 3;
    else nn(ii) = -3;
    end
end
%pulse shapping
i = 1;
t = 0:0.01 :length(n);
for j = 1 : length(t)
    if t(j) <= i;
        y(j)=nn(i);
    else
        y(j) = nn(i)
        i = i+1;
    end
end
plot(t,y,'-r')
axis([0 length(n) -10 10])
```

Output



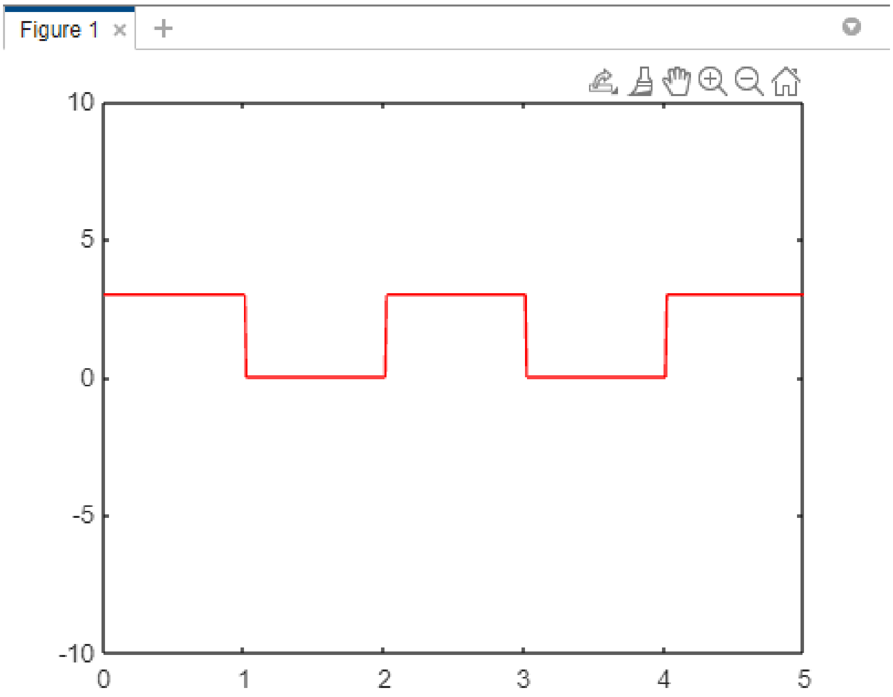
Unipolar NRZ

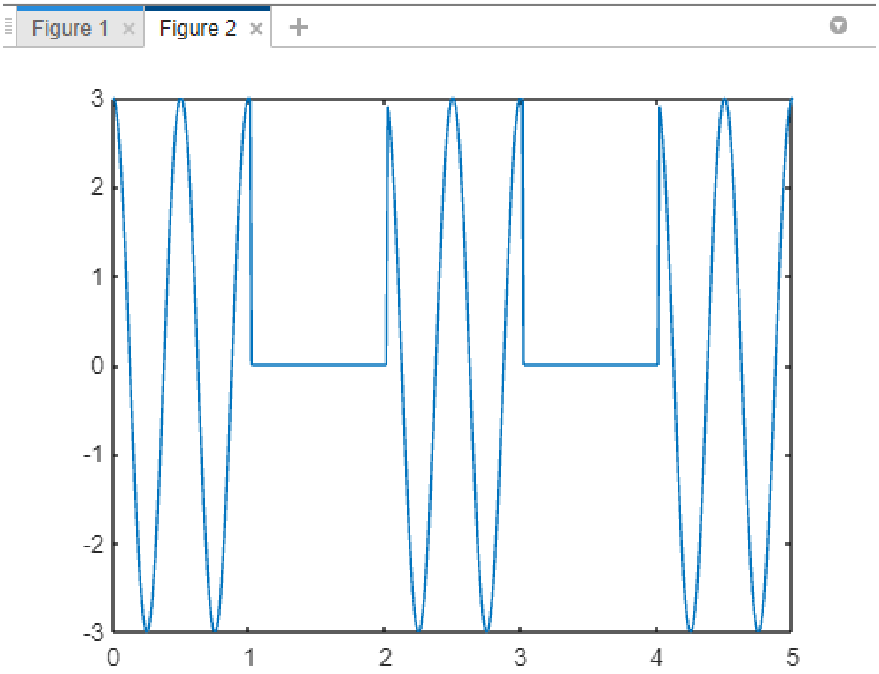
Input:

```
clc
clear all;
n = input('Enter the input bit:')
n = [1,0,1,0,1];
%mapping 1 = 3V AND 0 = 0V
for ii = 1: length(n)
    if n (ii) == 1;
        nn(ii) = 3;
    else nn(ii) = 0;
    end
end

%pulse shapping
i = 1;
t = 0:0.01 :length(n);
for j = 1 : length(t)
    if t(j) <= i;
        y(j)=nn(i);
    else
        y(j) = nn(i)
        i = i+1;
    end
end
plot(t,y,'-r')
axis([0 length(n) -10 10])
c=cos(2*pi*2*t);
x=y.*c;
%ploting
figure(1)
plot(t,y,'r-')
axis([0 length(n) -5 5]);
figure(2)
plot(t,x)
```

Output:



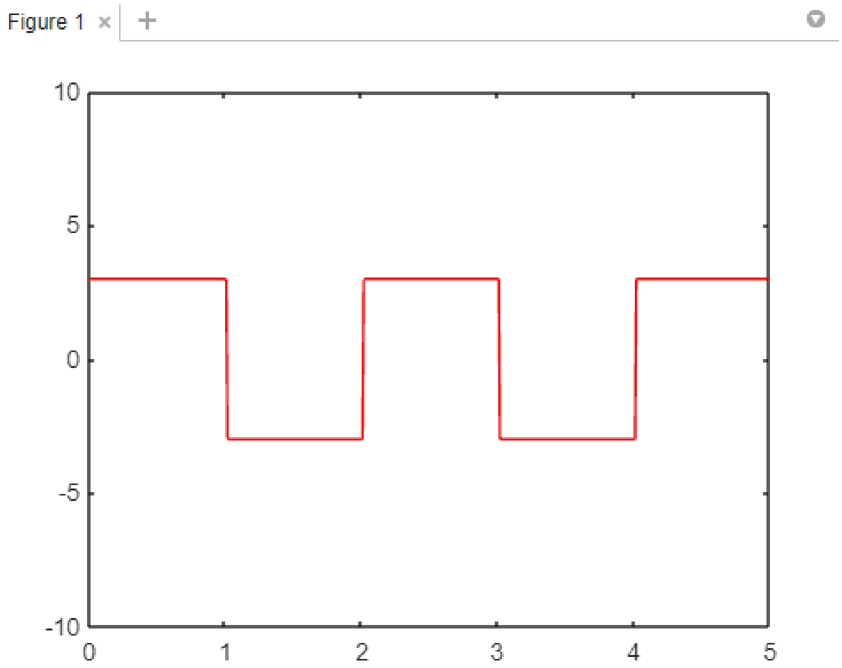


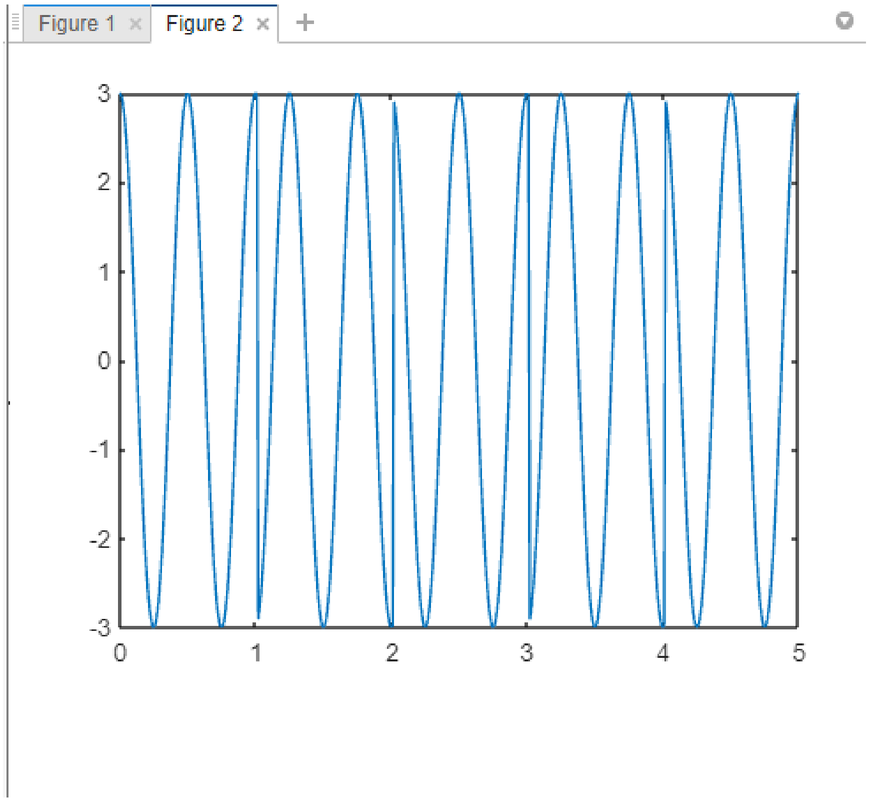
Polar NRZ

Input:

```
clc
clear all;
n = input('Enter the input bit:')
n = [1,0,1,0,1]
%mapping 1 = 3V AND 0 = 0V
for ii = 1:length(n)
    if n (ii) == 1;
        nn(ii) = 3;
    else nn(ii) = -3;
    end
end
%pulse shapping
i = 1;
t = 0:0.01 :length(n);
for j = 1 : length(t)
    if t(j) <= i;
        y(j)=nn(i);
    else
        y(j) = nn(i)
        i = i+1;
    end
end
plot(t,y,'-r')
axis([0 length(n) -10 10])
c=cos(2*pi*2*t);
x=y.*c;
%ploting
figure(1)
plot(t,y,'r-')
axis([0 length(n) -5 5]);
figure(2)
plot(t,x)
```

Output





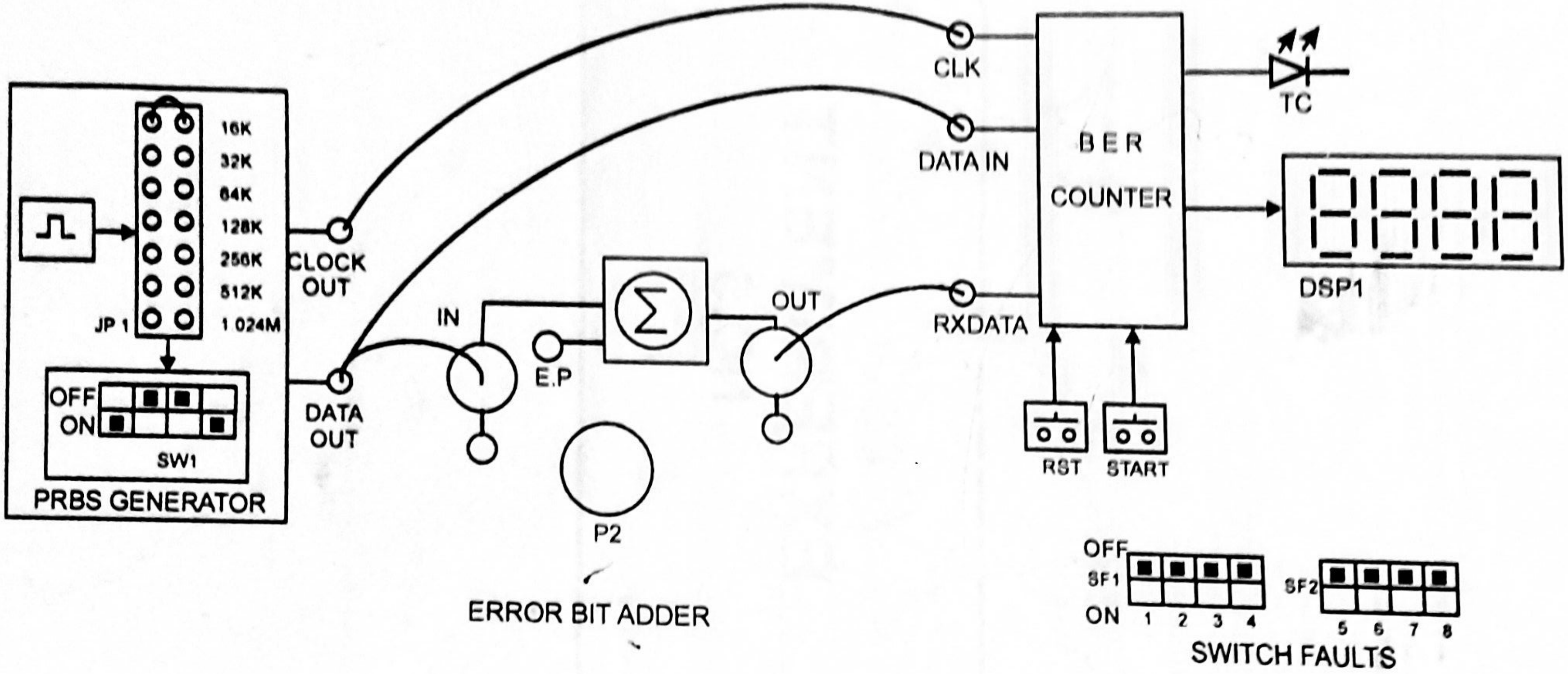


FIG. 5.1 BLOCK DIAGRAM FOR MEASUREMENT OF BIT ERROR RATE USING BINARY DATA

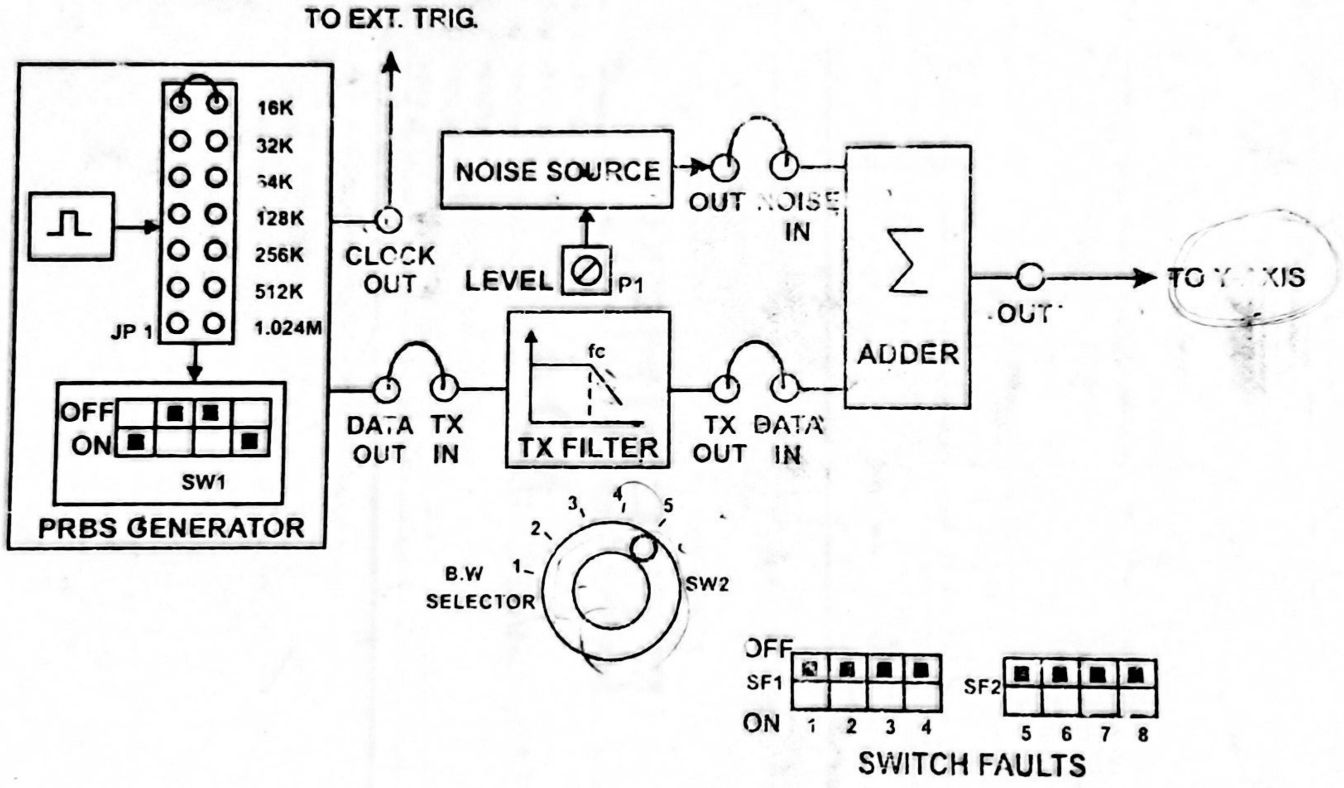


FIG. 4.1 BLOCK DIAGRAM FOR STUDY OF EYE PATTERN