Scilab Manual for
Digital Communication
by Prof Kalawati Patil
Others
Thakur College of Engineering & Technology[1]

Solutions provided by
Mr Sanjay Rawat
Others
Mumbai University/Thakur College of Engg. & Tech.

July 16, 2024

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

Contents

List of Experiments

Experiment: 1

Study and generate different Line Codes - 1 (Unipolar and Polar RZ and NRZ)

Scilab code Solution 1.1 Study and generate different Line Codes 1 Unipolar NRZ

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Study and generate different Line Codes - 1(Unipolar NRZ)

clear;
close;
clc;
clf;
x=[1 0 1 0 0 1 1 0]//Data Stream

//NRZ
```

```
z=0;//Starting value on x axis
for i=1:length(x)

    t=[z:1:z+1]//Set of x cordinates for current bit
        duration
    subplot(2,1,1)
    a=gca();
    a.data_bounds=[0,-1.5;length(x),1.5]
    a.grid=[1,-1]
    title('Data')
    plot(t,x(i))//Plot current data bit

    subplot(2,1,2)
    a=gca();
    a.data_bounds=[0,-1.5;length(x),1.5]
    a.grid=[1,-1]
    title('NRZ')
    if(x(i)==0)
        plot(t,0)//Plot 0 for current bit duration
    else
        plot(t,1)//Plot 1 for current bit duration
    end

    z=z+1//Increament starting value on x axis by
       one bit period
end
```

Scilab code Solution 1.2 Study and generate different Line Codes 1 Polar NRZ

```
//Note: Details of scilab software version and OS
    version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
     SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
```

```
64 bit versions)
//Program Title: Study and generate different Line Codes - 1(Polar NRZ)

clear;
close;
clc;
x=[1 0 1 0 0 1 1 0]//Data Stream

//Polar NRZ
z=0;//Starting value on x axis
for i=1:length(x)
    t=[z:1:z+1]//Set of x cordinates for current bit duration
        subplot(2,1,1)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Data')
        plot(t,x(i))//Plot current data bit

        subplot(2,1,2)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Polar NRZ')
        if(x(i)==0)
            plot(t,-1)//Plot -1 for current bit duration
        else
            plot(t,1)//Plot 1 for current bit duration
        end

    z=z+1//Increament starting value on x axis by one bit period
end
```

Scilab code Solution 1.3 Study and generate different Line Codes 1 Unipolar RZ

```
//Note: Details of scilab software version and OS
    version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
     SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
    64 bit versions)
//Program Title: Study and generate different Line
    Codes – 1(Unipolar RZ)

clear;
close;
clc;
x=[1 0 1 0 0 1 1 0]//Data Stream

//RZ
z=0;//Starting value on x axis
for i=1:length(x)
    t=[z:z+1]//Set of x cordinates for current bit
        duration
        subplot(2,1,1)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Data')
        plot(t,x(i))//Plot current data bit

    t=[z:0.5:z+0.5]//Set of x cordinates for first
        half bit duration
        subplot(2,1,2)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
```

```
        a.grid=[1,-1]
        title('Polar RZ')
        if(x(i)==0)
            plot(t,0)//Plot 0 for first half bit
               duration
        else
            plot(t,1)//Plot 1 for first half bit
               duration
        end
    t=[z+0.5:0.5:z+1]//Set of x cordinates for
       second half bit duration
    plot(t,0)//Plot 0 for second half bit duration

    z=z+1;//Increament starting value on x axis by
       one bit period
end
```

Scilab code Solution 1.4 Study and generate different Line Codes 1 Polar RZ

```
//Note: Details of scilab software version and OS
   version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
    SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
   64 bit versions)
//Program Title: Study and generate different Line
   Codes – 1(Polar RZ)

clear;
close;
clc;
x=[1 0 1 0 0 1 1 0]//Data Stream

//Polar RZ
```

```
z=0;//Starting value on x axis
for i=1:length(x)
    t=[z:z+1]//Set of x cordinates for current bit duration
        subplot(2,1,1)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Data')
        plot(t,x(i))//Plot current data bit

    t=[z:0.5:z+0.5]//Set of x cordinates for first half bit duration
        subplot(2,1,2)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Polar RZ')
        if(x(i)==0)
            plot(t,-1)//Plot -1 for first half bit duration
        else
            plot(t,1)//Plot 1 for first half bit duration
        end

    t=[z+0.5:0.5:z+1]//Set of x cordinates for second half bit duration
    plot(t,0)//Plot 0 for second half bit duration

    z=z+1;//Increament starting value on x axis by one bit period
end
```

Experiment: 2

Study and generate different Line Codes - 2(Bipolar, Manchestre and Quaternary)

Scilab code Solution 2.1 Study and generate different Line Codes 2 Bipolar NRZ

```
//Note: Details of scilab software version and OS
    version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
     SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
    64 bit versions)
//Program Title: Study and generate different Line
    Codes - 2(Bipolar NRZ)

clear;
close;
clc;
x=[1 0 1 0 0 1 1 0]//Data stream

//Bipolar NRZ

```

```
z=0;//Starting point of plot on x-axis
ob=-1;//Initial o/p bit value

for i=1:1:length(x)

    subplot(2,1,1)//Data Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Data')

        t=[z:1:z+1]//Plot range on x-axis (One bit period)
        plot(t,x(i))

    subplot(2,1,2)//Bipolar Bipolar NRZ
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Bipolar NRZ')

        if(x(i)==0)
            t=[z:1:z+1]//Plot range on x-axis (One bit period)
            plot(t,0)//Plot zero
        else
            t=[z:1:z+1]//Plot range on x-axis (One bit period)
            ob=-ob//Invert previous o/p bit value
            plot(t,ob)// Plot o/p bit
        end

    z=z+1//Move starting point of plot on x-axis by one bit period
end
```

Scilab code Solution 2.2 Study and generate different Line Codes 2 Bipolar RZ

```
//Note: Details of scilab software version and OS
    version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
     SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
    64 bit versions)
//Program Title: Study and generate different Line
    Codes - 2(Bipolar RZ)

clear;
close;
clc;
x=[1 0 1 0 0 1 1 0]//Data stream

//Bipolar RZ

z=0;//Starting point of plot on x-axis
ob=-1;//Initial o/p bit value

for i=1:1:length(x)

    subplot(2,1,1)//Data Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Data')

        t=[z:1:z+1]//Plot range on x-axis (One bit
           period)
        plot(t,x(i))

```

```
    subplot(2,1,2)//Bipolar Bipolar RZ
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Bipolar RZ')

        if(x(i)==0)
            t=[z:1:z+1]//Plot range on x-axis (One
               bit period)
            plot(t,0)//Plot zero
        else
            t=[z:0.5:z+0.5]//Plot range on x-axis (
               first half bit period)
            ob=-ob//Invert previous o/p bit value
            plot(t,ob)// Plot o/p bit
            t=[z+0.5:0.5:z+1]//Plot range on x-axis
               (second half bit period)
            plot(t,0)
        end

     z=z+1//Move starting point of plot on x-axis by
        one bit period
end
```

Scilab code Solution 2.3 Study and generate different Line Codes 2 Manchestre

```
//Note: Details of scilab software version and OS
   version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
    SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
   64 bit versions)
//Program Title: Study and generate different Line
   Codes - 2(Manchestre)

```

```
clear;
close;
clc;
x=[1 0 1 0 0 1 1 0]//Data Stream

//Manchester
z=0;//Starting value on x axis
for i=1:length(x)
    t=[z:1:z+1]//Plot range on x-axis (One bit
       period)
        subplot(2,1,1)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Data')
        plot(t,x(i))//Plot current data bit

    t=[z:0.5:z+0.5]//Plot range on x-axis (first
       half bit period)
        subplot(2,1,2)
        a=gca();
        a.data_bounds=[0,-1.5;length(x),1.5]
        a.grid=[1,-1]
        title('Manchester')
        if(x(i)==0)
            plot(t,1)//Plot 1 for first half bit
               duration
            t=[z+0.5:0.5:z+1]//Plot range on x-axis
               (second half bit period)
            plot(t,-1)//Plot 1 for second half bit
               duration
        else
            plot(t,-1)//Plot -1 for first half bit
               duration
            t=[z+0.5:0.5:z+1]//Plot range on x-axis
               (second half bit period)
            plot(t,1)//Plot 1 for second half bit
               duration
```

```
        end
    z=z+1;//Increament starting value on x axis by one bit period
end
```

Scilab code Solution 2.4 Study and generate different Line Codes 2 Quaternary

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Study and generate different Line Codes - 2(Quaternary)

clear;
close;
clc;
x=[1 0 0 1 1 1 0 0]//Data Stream
//x=[0 0 0 1 1 0 1 1]//Data Stream
a=1;
//Polar NRZ
z=0;//Starting value on x axis
for i=1:2:length(x)
    subplot(2,1,1)
    g=gca();
    g.data_bounds=[0,-1.5;length(x),1.5]
    g.grid=[1,-1]
    title('Data')
        t=[z:1:z+1]//Plot range on x-axis (One bit period for current bit)
        plot(t,x(i))//Plot curent bit
        t=[z+1:1:z+2]//Plot range on x-axis (One bit
```

```
            period for next bit)
        plot(t,x(i+1))//Plot next bit


    subplot(2,1,2)
    g=gca();
    g.data_bounds=[0,-2;length(x),2]
    g.grid=[1,-1]
    title('2B1Q (Quaternary)')
        t=[z:2:z+2]//Plot range on x-axis (two bit periods for current and next bit)
        if((x(i)==0)&(x(i+1)==0))//Check current and next bit combination
            plot(t,-3/2*a)//if 00 then plot -3/2*a
        elseif((x(i)==0)&(x(i+1)==1))//Check current and next bit combination
            plot(t,-1/2*a)//if 01 then plot -1/2*a
        elseif((x(i)==1)&(x(i+1)==0))//Check current and next bit combination
            plot(t,1/2*a)//if 10 then plot 1/2*a
        elseif((x(i)==1)&(x(i+1)==1))//Check current and next bit combination
            plot(t,3/2*a)//if 11 then plot 3/2*a
        end
    z=z+2//Increament starting value on x axis by two bits period
end
```

Experiment: 3

Study Carrier Modulation Techniques using BASK, BPSK and BFSK

Scilab code Solution 3.1 Study Carrier Modulation Techniques using BASK

```
//Note: Details of scilab software version and OS
   version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
    SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
   64 bit versions)
//Program Title: Study Carrier Modulation Techniques
    using BASK

clear;
clc;
close;
t = 0:0.01:1; // One symbol period
f=2; // Carrier cycles per symbol period
I=[0,0,1,1,0,1,0,1]; //data stream

//Generation of ASK Waveform
```

```
z=0;
for n=1:length(I)
    subplot(3,1,1) //Carrier Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('Carrier')
            plot((t+z),sin(2*%pi*f*t));

    subplot(3,1,2) //Data Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('Data')
            plot((t+z),I(n));

    subplot(3,1,3) //ASK Waveform Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('ASK Waveform')
            plot((t+z),(sin(2*%pi*f*t))*(I(n)));
    z=z+1;
end
```

Scilab code Solution 3.2 Study Carrier Modulation Techniques using BFSK

```
//Note: Details of scilab software version and OS
    version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
    SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
    64 bit versions)
//Program Title: Study Carrier Modulation Techniques
    using BFSK

clear;
clc;
close;
t = 0:0.01:1; // One symbol period
f1=2; // Carrier cycles per symbol period
f2=4; // Carrier cycles per symbol period
I=[0,0,1,1,0,1,0,1]; //data stream

//Generation of FSK Waveform

z=0;
for n=1:length(I)
    subplot(4,1,1) //Carrier 1 Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set
           the boundary values for the x-y
           coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('Carrier 1')
            plot((t+z),sin(2*%pi*f1*t));

    subplot(4,1,2) //Carrier 2 Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set
           the boundary values for the x-y
           coordinates.
```

```
        a.x_location="bottom";
        a.grid=[1,-1];
        title('Carrier 2')
            plot((t+z),sin(2*%pi*f2*t));

    subplot(4,1,3) //Data Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('Data')
            plot((t+z),I(n));

    subplot(4,1,4) //FSK Waveform Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('FSK Waveform')
            if (I(n)==0)
                plot((t+z),sin(2*%pi*f1*t));
            elseif (I(n)==1)
                plot((t+z),sin(2*%pi*f2*t));
            end
    z=z+1;
end
```

Scilab code Solution 3.3 Study Carrier Modulation Techniques using BPSK

```
//Note: Details of scilab software version and OS version used:
```

```
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Study Carrier Modulation Techniques using BPSK

clear;
clc;
close;
t = 0:0.01:1; // One symbol period
f=2; // Carrier cycles per symbol period
I=[0,0,1,1,0,1,0,1]; //data stream

//Generation of PSK Waveform

z=0;
for n=1:length(I)
    subplot(3,1,1) //Carrier Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('Carrier')
            plot((t+z),sin(2*%pi*f*t));

    subplot(3,1,2) //Data Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('Data')
            plot((t+z),I(n));

```

```
    subplot(3,1,3) //PSK Waveform Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="bottom";
        a.grid=[1,-1];
        title('PSK Waveform')
            if (I(n)==1)
                plot((t+z),sin(2*%pi*f*t));
            elseif (I(n)==0)
                plot((t+z),sin((2*%pi*f*t)+%pi));
            end
    z=z+1;
end
```

Experiment: 4

Study and generate OQPSK waveforms

Scilab code Solution 4.1 Study and generate offset QPSK waveforms

```
//Note: Details of scilab software version and OS
   version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
    SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
   64 bit versions)
//Program Title: Study and generate Offset QPSK (
   OQPSK) waveforms

clear;
clc;
close;
T=2;// One symbol period
t = 0:0.01:T/2;//Sampling Matrix for half symbol
   period
f=1; // Carrier frequency (cycles per bit period)
//I=[0 0 1 1 0 0 1 1]; //data stream
I=[0 0 0 1 1 0 1 1]; //data stream giving dibit
   equivalent to 0,1,2,3
```

```
//I=[1 1 0 0 0 1 1 1]; //data stream

//Polar NRZ Converter
I_PNRZ = [] //empty matrix for Polar NRZ data
       for n = 1:length(I)
           if I(n)== 0 then
               I_PNRZ = [I_PNRZ, -1]
           else
               I_PNRZ = [I_PNRZ, 1]
           end
       end

I_Carrier = sqrt(2/T)*cos(2*%pi*f*t); //In phase carrier
Q_Carrier = sqrt(2/T)*sin(2*%pi*f*t); //Quadrature phase carrier

//Generation of OQPSK Waveform
z=0;//Starting point of plot on x-axis
    subplot(3,1,1) //I-PSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('I-Data and I-PSK')
            plot((t+z),I_Carrier*I_PNRZ(1));//I_Carrier * First bit (I Balance Modulator)
            plot((t+z),I_PNRZ(1),'r');//First bit Data for reference
//xpause(2000000);//Delay for observation
z=z+1;//Move starting point of plot on x-axis by 1 bit (half symbol) period
for n=2:1:length(I_PNRZ)
    if modulo(n,2)==0 then//Check for odd-even bit
        I_Bit=I_PNRZ(n-1)//set I bit as previous bit
        Q_Bit=I_PNRZ(n)//set Q bit as current bit
    else
```

```
        I_Bit=I_PNRZ(n)//set I bit as current bit
        Q_Bit=I_PNRZ(n-1)//set Q bit as previous bit
    end

    subplot(3,1,1) //I-PSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('I-Data and I-PSK')
            plot((t+z),I_Carrier*I_Bit);//I_Carrier * Even bit (I Balance Modulator)
            plot((t+z),I_Bit,'r');//I Data for reference

    subplot(3,1,2) //Q-PSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('Q-Data and Q-PSK')
            plot((t+z),Q_Carrier*Q_Bit);//Q_Carrier * Odd bit (Q Balance Modulator)
            plot((t+z),Q_Bit,'r');//Q Data for reference

    subplot(3,1,3) //QPSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="Origin";
        a.grid=[1,1];
        title('OQPSK and I-Carrier')
            plot((t+z),(I_Carrier*I_Bit)+(Q_Carrier*Q_Bit));//I-PSK + Q-PSK (Adder)
            plot((t+z),I_Carrier,'r');//I Carrier for reference
z=z+1;//Move starting point of plot on x-axis by 1 bit (half symbol) period
```

```
//xpause(2000000);//Delay for observation
end
```

Experiment: 5

Study and generate NON OQPSK waveforms

Scilab code Solution 5.1 Study and generate Non offset QPSK waveforms

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Study and generate Non-Offset QPSK waveforms

clear;
clc;
close;
T=2; //One Symbol period
t = 0:0.01:T; // Sampling Matrix for one symbol period
f=1; // Carrier frequency (cycles per bit period)
I=[0 0 0 1 1 0 1 1]; //data stream giving dibit equivalent to 0,1,2,3
//I=[0 1 1 0 1 0 0 0]; //data stream Simon Hykin Ex
```

```
    6.1

//Polar NRZ Converter
I_PNRZ = [] //empty matrix for Polar NRZ data
       for n = 1:length(I)
           if I(n)== 0 then
               I_PNRZ = [I_PNRZ, -1]
           else
               I_PNRZ = [I_PNRZ, 1]
           end
       end

I_Carrier = sqrt(2/T)*cos(2*%pi*f*t); // In phase
    carrier
Q_Carrier = sqrt(2/T)*sin(2*%pi*f*t); // Quadrature
    phase carrier

//Generation of QPSK Waveform

z=0;//Starting point of plot on x-axis
for n=1:2:length(I_PNRZ)
    I_Bit=I_PNRZ(n)
    Q_Bit=I_PNRZ(n+1)
    subplot(3,1,1) //I-PSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('I-Data and I-PSK')
            plot((t+z),I_Carrier*I_Bit);//I_Carrier
               * Even bit (I Balance Modulator)
            plot((t+z),I_Bit,'r');//I Data for
               reference

    subplot(3,1,2) //Q-PSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
```

```
        a.grid=[1,1];
        title('Q-Data and Q-PSK')
            plot((t+z),Q_Carrier*Q_Bit);//Q_Carrier * Odd bit (Q Balance Modulator)
            plot((t+z),Q_Bit,'r');//Q Data for reference

    subplot(3,1,3) //QPSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('QPSK and I-Carrier')
            plot((t+z),(I_Carrier*I_Bit)+(Q_Carrier*Q_Bit));//I-PSK + Q-PSK (Adder)
            plot((t+z),I_Carrier,'r');//I Carrier for reference
z=z+2;//Move starting point of plot on x-axis by 2 bits (1 symbol) period
end
```

Experiment: 6

Study and generate 8-QAM waveforms

Scilab code Solution 6.1 Study and generate 8QAM waveforms

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Study and generate 8-QAM waveforms

clear;
clc;
close;
T=3; //One Symbol period
t = 0:0.01:T; // Sampling Matrix for one symbol period
f=1/T; // Carrier frequency (cycles per bit period)
I=[0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1]; //data stream giving tribits equivalent to 0,1,2,3,4,5,6,7

```

```
//Polar NRZ Converter
I_PNRZ = [] //empty matrix for Polar NRZ data
      for n = 1:length(I)
          if I(n)== 0 then
              I_PNRZ = [I_PNRZ, -1]
          else
              I_PNRZ = [I_PNRZ, 1]
          end
      end

I_Carrier = sqrt(2/T)*cos(2*%pi*f*t); // In phase carrier
Q_Carrier = sqrt(2/T)*sin(2*%pi*f*t); // Quadrature phase carrier

//Generation of 8-QAM Waveform

z=0;//Starting point of plot on x-axis
for n=1:3:length(I_PNRZ)
    Q_Bit=I_PNRZ(n)//Set Q Bit Value
    I_Bit=I_PNRZ(n+1)//Set I Bit Value
    C_Bit=I_PNRZ(n+2)//Set C Bit Value
        if C_Bit==-1 then //Set PAM, Product of C with I or Q
            QC=0.5*Q_Bit//Set half amplitude
            IC=0.5*I_Bit//Set half amplitude
        else
            QC=Q_Bit//Set full amplitude
            IC=I_Bit//Set full amplitude
        end

    subplot(3,1,1) //QC Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('Q-PAM')
            plot((t+z),Q_Carrier*QC);//Q_Carrier * Q
```

```
-PAM (Q Balance Modulator)
            plot((t+z),QC,'r');//Q-PAM Output

    subplot(3,1,2) //IC Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('I-PAM')
            plot((t+z),I_Carrier*IC);//I_Carrier * I-PAM (I Balance Modulator)
            plot((t+z),IC,'r');//I-PAM Output

    subplot(3,1,3) //8-QAM Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];
        a.x_location="origin";
        a.grid=[1,1];
        title('8-QAM')
            plot((t+z),(I_Carrier*IC)+(Q_Carrier*QC));//I-PAM + Q-PAM (Adder)
            plot((t+z),I_Carrier,'r');//I Carrier for reference
            plot(((t/3)+z),Q_Bit,'c');//Q Bit for reference
            plot(((t/3)+1+z),I_Bit,'b');//I Bit for reference
            plot(((t/3)+2+z),C_Bit,'m');//C Bit for reference

z=z+3;//Move starting point of plot on x-axis by 3 bits (1 symbol) period
end
```

Experiment: 7

Study and generate MSK waveforms (PSK Approach)

Scilab code Solution 7.1 Study and generate MSK waveforms with PSK Approach

```
//Note: Details of scilab software version and OS
    version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
     SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
    64 bit versions)
//Program Title: Study and generate MSK waveforms (
    PSK Approach)

clear;
clc;
close;
h=1/2;
T=1;// One symbol period
t = 0:0.01:T; // One symbol period vector
f=1; // Carrier cycles per symbol period "t"
//I=[0 1 0 1 0 1 0 1]; //data stream in binary to
    test worst case
```

```
I=[0 0 0 1 1 0 1 1]; //data stream giving dibits equivalent to 0,1,2,3

//PNRZ Converter, converts data to PNRZ (Bi-Polar Signal)
    I_PNRZ = [] //empty matrix for PNRZ data
        for n = 1:length(I)
            if I(n)== 0 then
                I_PNRZ = [I_PNRZ, -1]
            else
                I_PNRZ = [I_PNRZ, 1]
            end
        end

//Generation of MSK Waveform using PSK approach

theta=0;//Initial phase in radians

z=0;//Starting point of plot on x-axis

for n=1:1:length(I_PNRZ)
    subplot(3,1,1) //Data Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="origin";
        a.grid=[1,-1];
        title('Data')
            plot((t+z),I_PNRZ(n));

    subplot(3,1,2) //MSK Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="origin";
        a.grid=[1,-1];
```

```
        title('MSK')
            theta_change = theta + ((I_PNRZ(n))*((%pi*h*t)/T));//Phase variation over a bit period
            plot((t+z),sqrt(2/T)*cos(2*%pi*f*t + theta_change)); // MSK Plot
            plot((t+z),sqrt(2/T)*cos(2*%pi*f*t),'r'); // Carrier for reference

    subplot(3,1,3) // Plot for MSK Phase variation wrt Carrier
        a=gca();
        a.x_location="bottom";
        a.grid=[1,1];
        title('MSK Phase variation wrt Carrier')
        theta_degrees = theta_change*(180/%pi);//converts radians to degrees
            plot((t+z),theta_degrees);// plote phase variation for a bit period

theta=theta_change(length(theta_change));//Stores last value of phase to theta
z=z+1;//Move starting point of plot on x-axis by 1 bit period
//xpause(2000000);    //Delay for observation
end
```

Experiment: 8

Study and generate MSK waveforms (FSK Approach)

Scilab code Solution 8.1 Study and generate MSK waveforms with FSK Approach

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Study and generate MSK waveforms (FSK Approach)

clear;
clc;
close;
h=1/2;
T=1;// One symbol period
t = 0:0.01:T; // One symbol period vector
f=1; // Carrier cycles per symbol period "t"
//I=[0 1 0 1 0 1 0 1]; //data stream in binary to test worst case
```

```
I=[0 0 0 1 1 0 1 1]; //data stream giving dibits equivalent to 0,1,2,3

//PNRZ Converter, converts data to PNRZ (Bi-Polar Signal)
    I_PNRZ = [] //empty matrix for PNRZ data
        for n = 1:length(I)
            if I(n)== 0 then
                I_PNRZ = [I_PNRZ, -1]
            else
                I_PNRZ = [I_PNRZ, 1]
            end
        end

//Generation of MSK Waveform using FSK approach

bitchange=0;//Initial bit state (before first bit of sequence)
theta=0;//Initial phase state in radians (before first bit of sequence)
theta_degrees=[0,0];//Initial phase state in degrees (first element = start value, second element = last value)

z=0;//Starting point of plot on x-axis
for n=1:1:length(I_PNRZ)
    subplot(3,1,1) //Data Plot
        a=gca();
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5]; //set the boundary values for the x-y coordinates.
        a.x_location="origin";
        a.grid=[1,-1];
        title('Data')
            plot((t+z),I_PNRZ(n));

    subplot(3,1,2) //MSK Plot
        a=gca();
```

```
        a.data_bounds=[0,-1.5;length(I_PNRZ),1.5];//set the boundary values for the x-y coordinates.
        a.x_location="origin";
        a.grid=[1,-1];
        title('MSK')
        fm = f + (I_PNRZ(n)*(h/(2*T)));//Generating two frequencies corresponding to
                                      //binary 0 (-1 in PNRZ)and binary 1 (1 in PNRZ)
                                      //(0 --> fc - h/2T)
                                      //(1 --> fc + h/2T)
          plot((t+z),sqrt(2/T)*cos(2*%pi*fm*t + theta)); // MSK Plot
          plot((t+z),sqrt(2/T)*cos(2*%pi*f*t),'r'); // Carrier for reference

    subplot(3,1,3) // Plot for MSK Phase variation wrt Carrier
        a=gca();
        a.x_location="bottom";
        a.grid=[1,1];
        title('MSK Phase variation wrt Carrier')
        bitchange=bitchange+I_PNRZ(n);//Bit State value (cumulative)
        theta = bitchange*((%pi*h)/T);//Phase state at the end of bit period, in radians
        theta_degrees(2)=theta*180/%pi;//Phase state at the end of bit period, in degrees
          plot([z n],theta_degrees);// plote phase variation for a bit period

theta_degrees(1)=theta_degrees(2);//Copy end phase value to start phase value for next cycle
```

```
z=z+1;//Move starting point of plot on x-axis by 1
    bit period
//xpause(2000000); //Delay for observation
end
```

Experiment: 9

To calculate all Codewords, error detection and correction capability of given LBC

Scilab code Solution 9.1 Linear Block Codes

```
//Note: Details of scilab software version and OS
   version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
    SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
   64 bit versions)
//Program Title: Linear Block Codes (7,4)

clc;
clear;

k = 4; //Information message length
n = 7; //Coded word length

P = [1 1 0;0 1 1 ;1 1 1;1 0 1]//Parity Matrix
disp(P,'Parity Matrix P')

```

```
G = [P eye(k,k)]//Generator Matrix to create code word in P1P2P3D1D2D3D4 format
disp(G,'Generator Matrix G')

H=[eye(n-k,n-k);P]'//Parity Check Matrix
disp(H,'Parity Check Matrix H')

//All_M = All 16 posibilities for Information Message Matrix
All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]

//Calculate all 16 posibile codewords
CodedMat=All_M*G;
CodedMat = modulo(CodedMat,2);//Convert generated code into binary
disp(CodedMat,'Codewords Matrix')

//Calculate Hamming Distances
HamDist=sum(CodedMat,'c')//Sum over the rows of CodedMat(column of values)
disp(HamDist,'Hamming Distances');

//Find Minimum non-zero Hamming Distance
[row,col]=find(HamDist==0);//find elements that are zero
HamDist(row,:)=[];//Remove all rows that are zero (replace by null)
MinHamDist=min(HamDist)//Find Minimum non-zero Hamming Distance
disp(MinHamDist,'Minimum Non-Zero Hamming Distance')

//Calculate Error Detection Capability
ErrDetCap=MinHamDist-1;
disp(ErrDetCap,'Error Detection Capability');
```

```
//Calculate Error Correction Capability
ErrCorCap=(MinHamDist-1)/2;
disp(ErrCorCap,'Error Correction Capability');

//Generate random message
RandMessage=modulo(round(16*rand()),16)+1//Get random number between 1 to 16

M=All_M(RandMessage,:)//Select a random row from Message Matrix All_M as Information Message
disp(M,'Information Message M')

C = CodedMat(RandMessage,:)//Select a random row from Coded Matrix CodedMat as Coded Message
disp(C,'Coded Message C')

//Transmit random message
R=C//Create recieved code word

//Generate error at random bit position

ErrPos=modulo(round(8*rand()),8)//Get random number between 0 to 7

if ErrPos==0 then
    //Do nothing, as '0' means no error
else
    if R(ErrPos)==0 then
        R(ErrPos)=1//Invert bit at Erroneous Bit Position
    else
        R(ErrPos)=0//Invert bit at Erroneous Bit Position
    end
end

disp(R,'Recieved Code word R')

```

```
//Error Correction

S=R*H'//Find Syndrome Matrix
S = modulo(S,2);//Convert Syndrome Matrix into binary
disp(S,'Syndrome Matrix R*H(transpose)')

if S==[0 0 0]  then //[0 0 0] indicates no error
    disp(R,'Recieved Code without error')
    disp(R(4:7),'Recieved Information Message')//Extract and display  Message from code word
else
    //Find erroneous bit position
    //Here we find colomn within H matrix with pattern simmilar to Syndrome Matrix
    //The position number of that colomn is equivalent to erroneous bit position

    ErrPos=1//Initiallize erroneous bit position
    d=[H(:,ErrPos)]'//Transpose of first coloumn of H matrix
                    //(Transpose is used to convert colomn to row as syndrome is in row format)

        while ((d(1)<>S(1))|(d(2)<>S(2))|(d(3)<>S(3))) do //Check element wise inequallity for any element (OR condition)
            ErrPos=ErrPos+1//Increament erroneous bit position (Point to next colomn)
            d=[H(:,ErrPos)]'//Transpose of next coloumn of H matrix
        end

        disp(ErrPos,'Erroneous Bit Position')

        //Error correction
        if R(ErrPos)==0 then
```

```
                R(ErrPos)=1//Invert bit at Erroneous Bit Position
                disp(R,'Recieved Code with error corrected')
                disp(R(4:7),'Recieved Information Message')//Extract and display Message from code word
        else
            R(ErrPos)=0//Invert bit at Erroneous Bit Position
            disp(R,'Recieved Code with error corrected')
            disp(R(4:7),'Recieved Information Message')//Extract and display Message from code word
        end
end
```

Experiment: 10

To encode Cyclic code and calculate Syndrome for the given generator polynomial

Scilab code Solution 10.1 Cyclic Codes

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Cyclic Codes (7,4)

clc;
clear;
k = 4; //Information Message Length
n = 7; //Codeword Length

//Generator Polynomial
x=poly(0,'x');
GenPoly=1+x+x^3;
disp(GenPoly,'The Generator Polynomial');
```

```

//Generating Random Message

//All_M = All 16 posibilities for Information Message Matrix
All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]

RandMessage=modulo(round(16*rand()),16)+1//Get random number between 1 to 16

M=All_M(RandMessage,:)//Select a random row from Message Matrix All_M as Information Message
disp(M,'Information Message M')

//Message Polynomial
MesPoly=(M(1)*1) + (M(2)*(x^1)) + (M(3)*(x^2)) + (M(4)*(x^3));
disp(MesPoly,'Message Polynomial u(x)');

//Encoding---------------------------------------------------------------

//Generating Codeword Polynomial
p=(x^(n-k))*(MesPoly);//Step 1 - multiply MesPoly by x^(n-k), [x^(n-k)*u(x)]
[RemPoly,q]=pdiv(p,GenPoly);//Step 2 - divide above product by GenPoly, g(x)(Polynomial Division)
RemPoly=modulo(RemPoly,2);//Convert Remainder Polynomial to binary to get parity check polynomial, b(x)
disp(RemPoly,'Remainder Polynomial b(x)');
CodePoly=RemPoly+(MesPoly*(x^(n-k)));//Step 3 - add (x^(n-k)*u(x)) and b(x) to get Codeword Polynomial
```

```
disp(CodePoly,'Codeword Polynomial c(x)');

//Finding Coefficients of Codeword Polynomial
CodePolyCoeff=coeff(CodePoly);
    //Removal of - signs from Coefficients of
       Codeword Polynomial
    for i=1:length(CodePolyCoeff)
        if (CodePolyCoeff(i)==-1) then
            CodePolyCoeff(i)=1;
        end
    end
disp(CodePolyCoeff,'Coefficients of Codeword
   Polynomial');

//Generating 7 bit Codeword from Coefficients of
   Codeword Polynomial
C=CodePolyCoeff;
if length(C)<7 then
    C(1,7)=0;//Assigning a value outside arrey
       dimension will automatically
            //pad additional zeros to resize the
               arrey / vector
end
disp(C,'Generated Codeword');

//Transmition
   ------------------------------------------------------------------

R=C//Create recieved code word

    //Generate error at random bit position
    ErrPos=modulo(round(8*rand()),8)//Get random
       number between 0 to 7
    //ErrPos=0 //for testing
    if ErrPos==0 then
        //Do nothing, as '0' means no error
    else
        if R(ErrPos)==0 then
```

```
            R(ErrPos)=1//Invert bit at Erroneous Bit Position
        else
            R(ErrPos)=0//Invert bit at Erroneous Bit Position
        end
    end

//Reception and Decoding ------------------------------------------------------------------------

disp(R,'Recieved Code word R')

//Received Polynomial
RecPoly=(R(1)*1) + (R(2)*(x^1)) + (R(3)*(x^2)) + (R(4)*(x^3)) + (R(5)*(x^4)) + (R(6)*(x^5)) + (R(7)*(x^6));
disp(RecPoly,'Received Polynomial u(x)');

//Syndrome Polynomial
[SynPoly,q]=pdiv(RecPoly,GenPoly);
SynPoly=modulo(SynPoly,2)
disp(SynPoly,'Syndrome Polynomial')

//Finding Coefficients of Syndrome Polynomial
SynPolyCoeff=coeff(SynPoly);
    //Removal of - signs from Coefficients of Syndrome Polynomial
    for i=1:length(SynPolyCoeff)
        if (SynPolyCoeff(i)==-1) then
            SynPolyCoeff(i)=1;
        end
    end
disp(SynPolyCoeff,'Coefficients of Syndrome Polynomial');

//Generating 3 bit Syndrome from Coefficients of
```

```
Syndrome Polynomial
if length(SynPolyCoeff)<3 then
    SynPolyCoeff(1,3)=0;//Assigning a value outside arrey dimension will automatically
                //pad additional zeros to resize the arrey / vector
end
disp(SynPolyCoeff,'Syndrome');



        //Create H (Parity check matrix) as error lookup table
        P = [1 1 0;0 1 1 ;1 1 1;1 0 1]//Parity Matrix
        H=[eye(n-k,n-k);P]'//Parity Check Matrix
        //disp(H,'Parity Check Matrix H') //for testing

if SynPolyCoeff==[0 0 0]  then //[0 0 0] indicates no error
    disp(R,'Recieved Code without error')
    disp(R(4:7),'Recieved Information Message')//Extract and display  Message from code word
else
    //Find erroneous bit position
    //Here we find colomn within H matrix with pattern simmilar to Syndrome Matrix
    //The position number of that colomn is equivalent to erroneous bit position

    ErrPos=1//Initiallize erroneous bit position
    d=[H(:,ErrPos)]'//Transpose of first coloumn of H matrix
                        //(Transpose is used to convert colomn to row as syndrome is in row format)

```

```
        while ((d(1)<>SynPolyCoeff(1))|(d(2)<>
           SynPolyCoeff(2))|(d(3)<>SynPolyCoeff(3)))
            do //Check element wise inequallity for
           any element (OR condition)
            ErrPos=ErrPos+1//Increament erroneous
               bit position (Point to next colomn)
            d=[H(:,ErrPos)]'//Transpose of next
               coloumn of H matrix
        end

        disp(ErrPos,'Erroneous Bit Position')

        //Error correction
        if R(ErrPos)==0 then
            R(ErrPos)=1//Invert bit at Erroneous Bit
                Position
        else
            R(ErrPos)=0//Invert bit at Erroneous Bit
                Position
        end
        disp(R,'Recieved Code with error corrected')
        disp(R(4:7),'Recieved Information Message')
           //Extract and display  Message from code
           word
end
disp(M,'Information Message M that was sent...')
```

Experiment: 11

To encode and decode Hamming code

Scilab code Solution 11.1 Hamming Codes

```
//Note: Details of scilab software version and OS
    version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP
     SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and
    64 bit versions)
//Program Title: Hamming Codes (7,4)

clc;
clear;

k = 4; //Information message matrix length
n = 7; //Coded word length

P = [1 1 0;0 1 1 ;1 1 1;1 0 1]//Parity Matrix
disp(P,'Parity Matrix P')

G = [P eye(k,k)]//Generator Matrix to create code
    word in P1P2P3D1D2D3D4 format
```

```
G(:,[3 4])=G(:,[4 3])//Swap colomn 3 and 4 of G to
    create code word in P1P2D1P3D2D3D4 format
disp(G,'Generator Matrix G')

H=[eye(n-k,n-k);P]'//Parity Check Matrix
H(:,[3 4])=H(:,[4 3])//Swap colomn 3 and 4 of H to
    satisfy GH'=0
disp(H,'Parity Check Matrix H')

//disp(modulo(G*H',2),'GH')//Check if the condition
    GH'=0 satisfy (for testing)

//M = [1 1 0 1]//Information Message Matrix for
    testing

//Generate random message
//All_M = All 16 posibilities for Information
    Message Matrix
All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]
RandMessage=modulo(round(16*rand()),16)+1//Get
    random number between 1 to 16
M=All_M(RandMessage,:)//Select a random row from 1
    to 16 as Information Message

disp(M,'Information Message M')

C = M*G;//Generate code word
C = modulo(C,2);//Convert generated code into binary
disp(C,'Code word of (7,4) Hamming code M*G')

R=C//Create recieved code word

//Generate error at random bit position

ErrPos=modulo(round(8*rand()),8)//Get random number
```

```
   between 0 to 7

if ErrPos==0 then
    //Do nothing , as '0' means no error
else
    if R(ErrPos)==0 then
        R(ErrPos)=1//Invert bit at Erroneous Bit Position
    else
        R(ErrPos)=0//Invert bit at Erroneous Bit Position
    end
end

disp(R,'Recieved Code word R')

//Error Correction

S=R*H'//Find Syndrome Matrix
S = modulo(S,2);//Convert Syndrome Matrix into binary
disp(S,'Syndrome Matrix R*H(transpose)')

if S==[0 0 0]  then //[0 0 0] indicates no error
    disp(R,'Recieved Code without error')
    disp([R(3) R(5:7)],'Recieved Information Message ')//Extract and display  Message from code word
else
    //Find erroneous bit position
    //Here we find colomn within H matrix with pattern simmilar to Syndrome Matrix
    //The position number of that colomn is equivalent to erroneous bit position

    ErrPos=1//Initiallize erroneous bit position
    d=[H(:,ErrPos)]'//Transpose of first coloumn of H matrix
```

```
                    //(Transpose is used to convert colomn to row as syndrome is in row format)

        while ((d(1)<>S(1))|(d(2)<>S(2))|(d(3)<>S(3))) do //Check element wise inequallity for any element (OR condition)
            ErrPos=ErrPos+1//Increament erroneous bit position (Point to next colomn)
            d=[H(:,ErrPos)]'//Transpose of next coloumn of H matrix
        end

        disp(ErrPos,'Erroneous Bit Position')

        //Error correction
        if R(ErrPos)==0 then
                R(ErrPos)=1//Invert bit at Erroneous Bit Position
                disp(R,'Recieved Code with error corrected')
                disp([R(3) R(5:7)],'Recieved Information Message')//Extract and display  Message from code word
        else
            R(ErrPos)=0//Invert bit at Erroneous Bit Position
            disp(R,'Recieved Code with error corrected')
            disp([R(3) R(5:7)],'Recieved Information Message')//Extract and display Message from code word
        end
end
```