# B.M.S COLLEGE OF ENGINEERING

**Department of Electronics and Communication Engineering**

Bull Temple Road, Basavanagudi, Bangalore - 560 019

Project Report

on

# “ANALYSIS OF ADVANCED LINE CODING TECHNIQUES”

Submitted by

| **Name of the Student** | **USN** |
|---|---|
| CHINMAYI B TEGGI | 1BM14EC022 |
| DHEERAJ D KAMATH | 1BM14EC027 |
| DIGANTH P | 1BM14EC028 |
| JAHNAVI REDDY | 1BM14EC042 |
| NEHA B S | 1BM14EC060 |

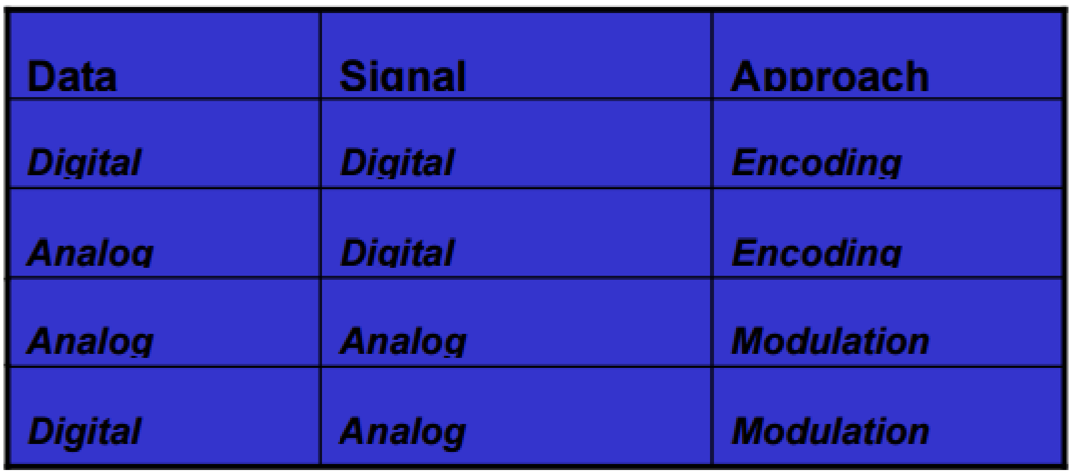
**Name of the Faculty In-charge: NARAYAN T DESPANDEY**

**Designation: Assistant professor, ECE Department**

# CONTENTS

## INTRODUCTION:

A computer network is used for communication of data from one station to another station in the network. We have seen that analog or digital data traverses through a communication media in the form of a signal from the source to the destination. The channel bridging the transmitter and the receiver may be a guided transmission medium such as a wire or a wave-guide or it can be an unguided atmospheric or space channel. But, irrespective of the medium, the signal traversing the channel becomes attenuated and distorted with increasing distance. Hence a process is adopted to match the properties of the transmitted signal to the channel characteristics so as to efficiently communicate over the transmission media. There are two alternatives; the data can be either converted to digital or analog signal. Both the approaches have pros and cons. What to be used depends on the situation and the available bandwidth. Now, either form of data can be encoded into either form of signal. For digital signalling, the data source can be either analog or digital, which is encoded into digital signal, using different encoding techniques. The basis of analog signalling is a constant frequency signal known as a carrier signal, which is chosen to be compatible with the transmission media being used, so that it can traverse a long distance with minimum of attenuation and distortion. Data can be transmitted using these carrier signals by a process called modulation, where one or more fundamental parameters of the carrier wave, i.e. amplitude, frequency and phase are being modulated by the source data. The resulting signal, called modulated signal traverses the media, which is demodulated at the receiving end and the original signal is extracted. All the four possibilities are shown in figure below:



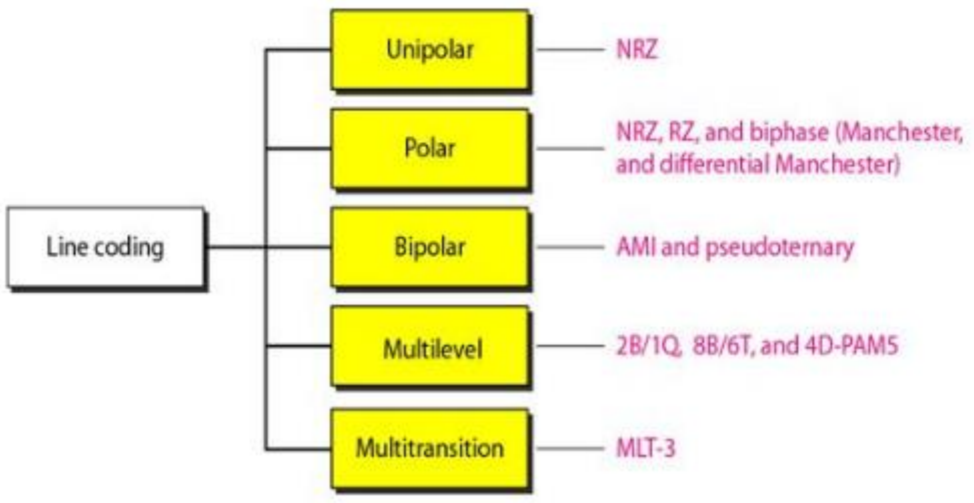
| Data | Signal | Approach |
|---|---|---|
| *Digital* | *Digital* | *Encoding* |
| *Analog* | *Digital* | *Encoding* |
| *Analog* | *Analog* | *Modulation* |
| *Digital* | *Analog* | *Modulation* |

Various approaches for conversion of data to signal This lesson will be concerned with various techniques for conversion digital and analog data to digital signal, commonly referred to as **encoding techniques**.

## LITERATURE SURVEY:

Digital Line Coding is a special coding system chosen to allow transmission to take place in a communications system. The chosen code or pattern of voltage used to represent binary digits on a transmission medium is called line encoding. In telecommunication, a line code is a code chosen for use within a communications system for transmitting a digital signal down a line. Line coding is often used for digital data transport. Line coding consists of representing the digital signal to be transported, by a waveform that is appropriate for the specific properties of the physical channel (and of the receiving equipment). The common types of line encoding are unipolar, polar, bipolar, and Manchester encoding.

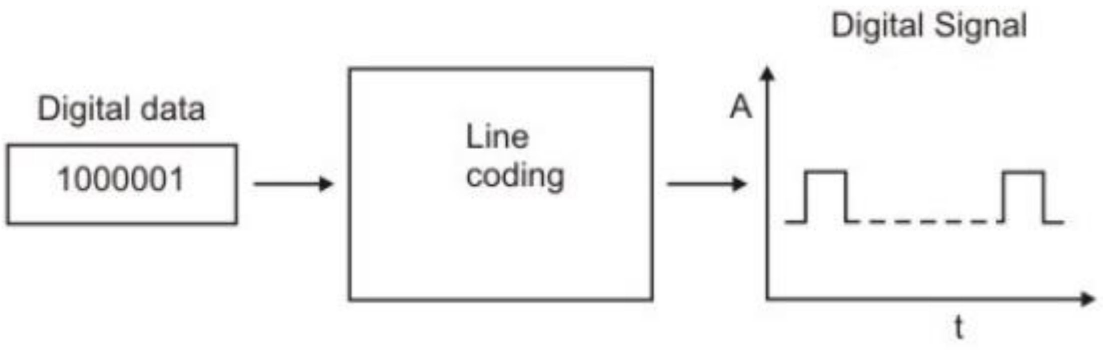
However there are many advanced Line Coding schemes and are categorized as shown in the following figure:


Line coding
Unipolar
NRZ
Polar
NRZ, RZ, and biphase (Manchester, and differential Manchester)
Bipolar
AMI and pseudoternary
Multilevel
2B/1Q, 8B/6T, and 4D-PAM5
Multitransition
MLT-3

Many advancements thus have been done in the field of line encoding and this project will try to compare and evaluate each of these techniques, the advantages and disadvantages and aim to formulate the same, with simulation results using MATLAB.

## LINE ENCODING

The first approach converts digital data to digital signal, known as line coding, as shown in figure below.



Important parameters those characteristics line coding techniques are mentioned below.

**No of signal levels:** This refers to the number values allowed in a signal, known as signal levels, to represent data.

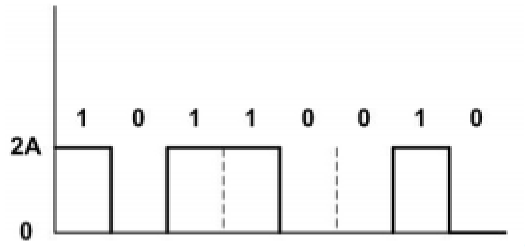
**Bit rate versus Baud rate:** The bit rate represents the number of bits sent per second, whereas the baud rate defines the number of signal elements per second in the signal. Depending on the encoding technique used, baud rate may be more than or less than the data rate.

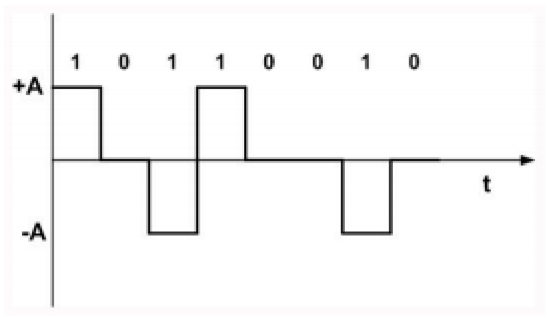
**DC components:** After line coding, the signal may have zero frequency component in the spectrum of the signal, which is known as the direct-current (DC) component. DC component in a signal is not desirable because the DC component does not pass through some components of a communication system such as a transformer. This leads to distortion of the signal and may create error at the output. The DC component also results in unwanted energy loss on the line.

**Signal Spectrum:** Different encoding of data leads to different spectrum of the signal. It is necessary to use suitable encoding technique to match with the medium so that the signal suffers minimum attenuation and distortion as it is transmitted through a medium.

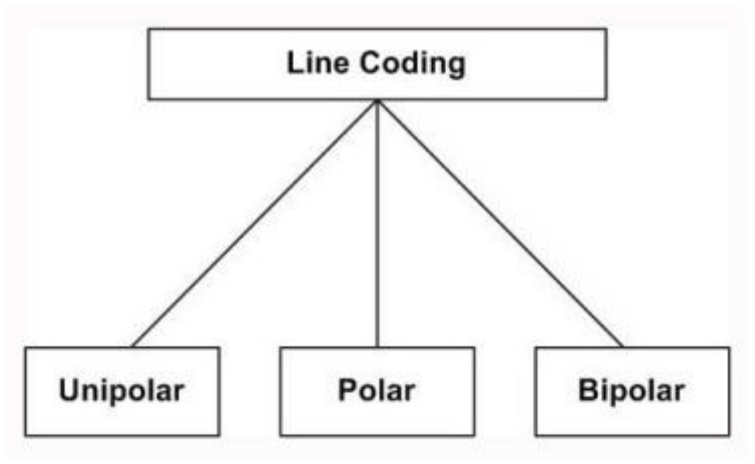
**Synchronization:** To interpret the received signal correctly, the bit interval of the receiver should be exactly same or within certain limit of that of the transmitter. Any mismatch between the two may lead wrong interpretation of the received signal. Usually, clock is generated and synchronized from the received signal with the help of a special hardware known as Phase Lock Loop (PLL). However, this can be achieved if the received signal is self-synchronizing having frequent transitions (preferably, a minimum of one transition per bit interval) in the signal.

**Cost of Implementation:** It is desirable to keep the encoding technique simple enough such that it does not incur high cost of implementation.



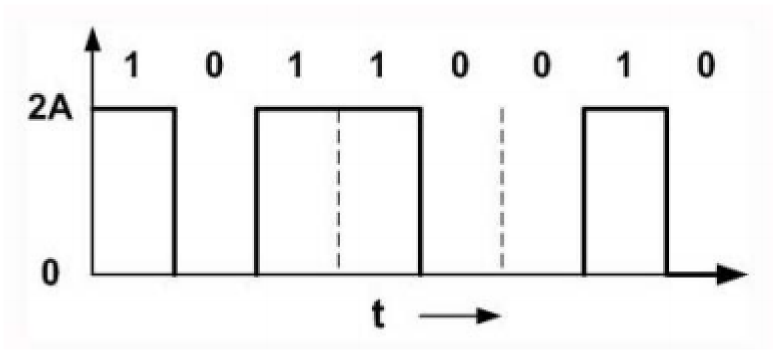


Line coding techniques can be broadly divided into three broad categories: Unipolar, Polar and Bipolar, as shown in Figure below:



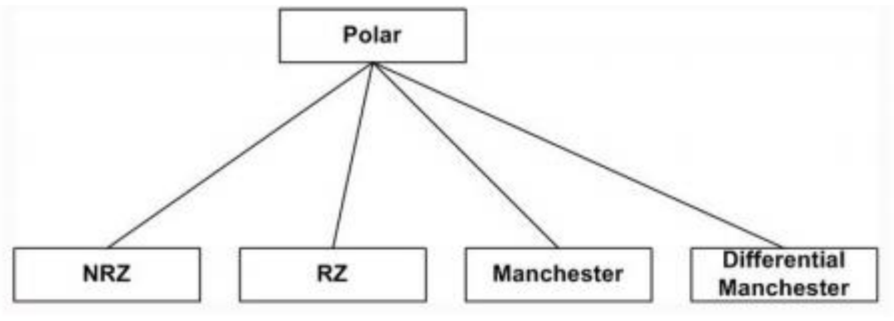
## Unipolar:

In unipolar encoding technique, only two voltage levels are used. It uses only one polarity of voltage level as shown in Fig. 2.4.5. In this encoding approach, the bit rate same as data rate. Unfortunately, DC component present in the encoded signal and there is loss of synchronization for long sequences of 0's and 1's. It is simple but obsolete.

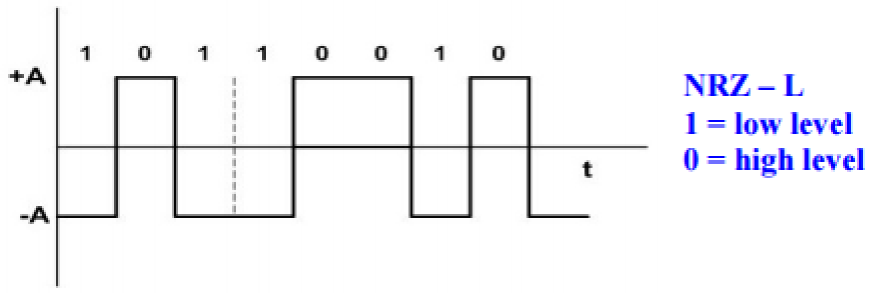


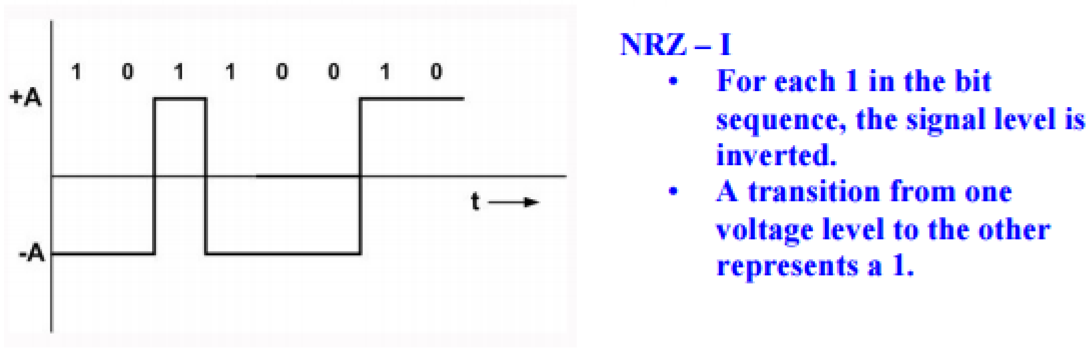
## Polar:

Polar encoding technique uses two voltage levels – one positive and the other one negative. Four different encoding schemes shown in figure below:



Non Return to zero (NRZ): The most common and easiest way to transmit digital signals is to use two different voltage levels for the two binary digits. Usually a negative voltage is used to represent one binary value and a positive voltage to represent the other. The data is encoded as the presence or absence of a signal transition at the beginning of the bit time. As shown in the figure below, in NRZ encoding, the signal level remains same throughout the bit-period. There are two encoding schemes in NRZ: NRZ-L and NRZ-I, as shown in figure:



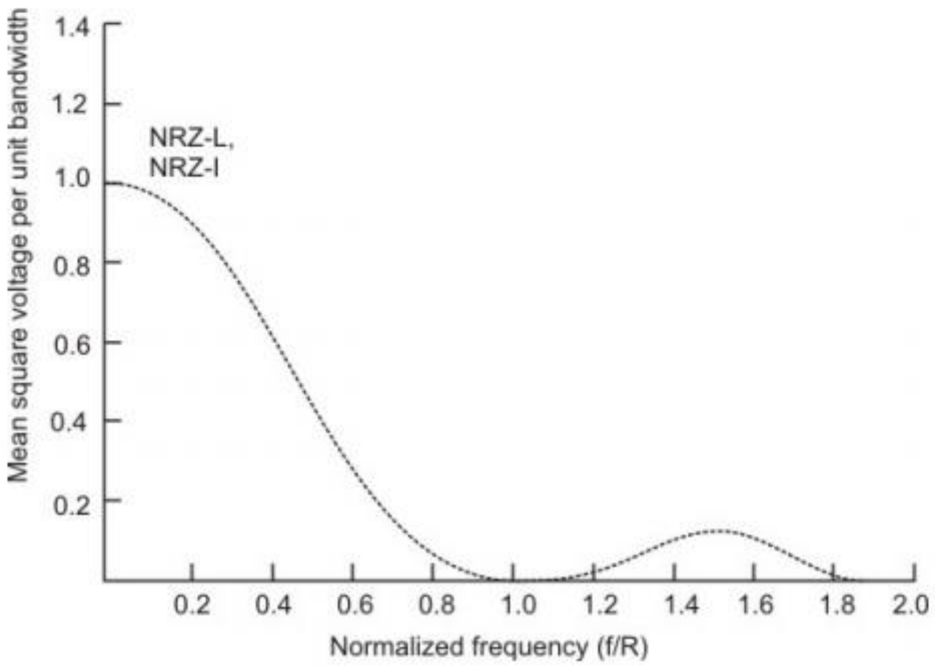

1 0 1 1 0 0 1 0
+A
-A
t
NRZ – I
For each 1 in the bit sequence, the signal level is inverted.
A transition from one voltage level to the other represents a 1.

The advantages of NRZ coding are:

• Detecting a transition in presence of noise is more reliable than to compare a value to a threshold.

• NRZ codes are easy to engineer and it makes efficient use of bandwidth.

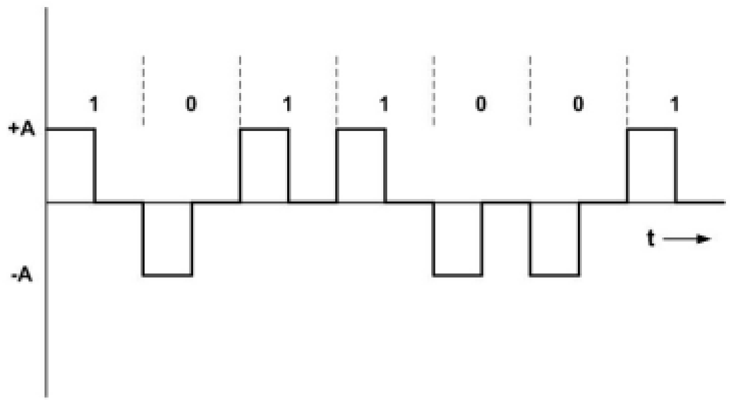
The spectrum of the NRZ-L and NRZ-I signals are shown in Fig. 2.4.8. It may be noted that most of the energy is concentrated between 0 and half the bit rate. The main limitations are the presence of a dc component and the lack of synchronization capability. When there is long sequence of 0's or 1's, the receiving side will fail to regenerate the clock and synchronization between the transmitter and receiver clocks will fail.



Return to Zero RZ: To ensure synchronization, there must be a signal transition in each bit as shown in figure below:

Key characteristics of the RZ coding are:

- Three levels
- Bit rate is double than that of data rate
- No dc component
- Good synchronization
- Main limitation is the increase in bandwidth



Biphase: To overcome the limitations of NRZ encoding, biphase encoding techniques can be adopted. Manchester and differential Manchester Coding are the two common Biphase techniques in use, as shown in Figure below. In Manchester coding the mid-bit transition serves as a clocking mechanism and also as data.

In the standard Manchester coding there is a transition at the middle of each bit period. A binary 1 corresponds to a low-to-high transition and a binary 0 to a high-to-low transition in the middle.

In Differential Manchester, inversion in the middle of each bit is used for synchronization. The encoding of a 0 is represented by the presence of a transition both at the beginning and at the middle and 1 is represented by a transition only in the middle of the bit period.

Key characteristics are:

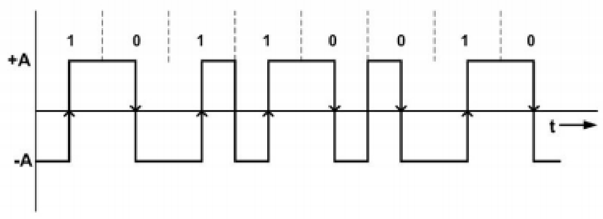
• Two levels

• No DC component

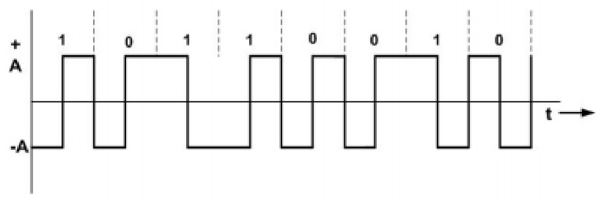
• Good synchronization

• Higher bandwidth due to doubling of bit rate with respect to data rate

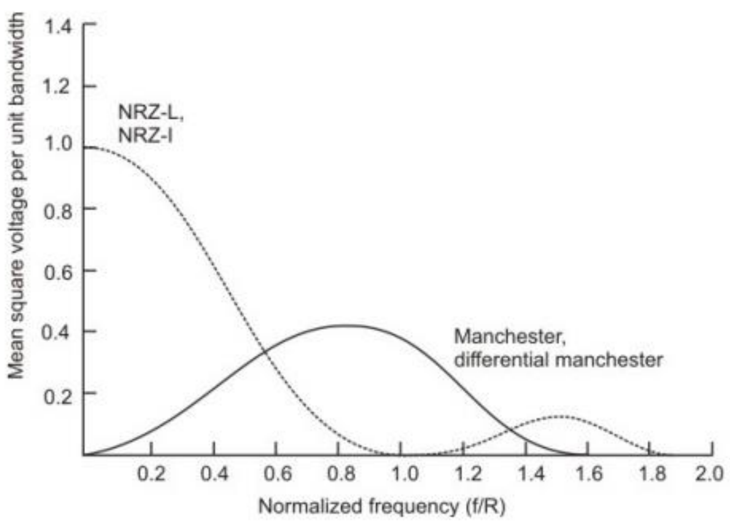
The bandwidth required for biphase techniques are greater than that of NRZ techniques, but due to the predictable transition during each bit time, the receiver can synchronize properly on that transition. Biphase encoded signals have no DC components as shown below. A Manchester code is now very popular and has been specified for the IEEE 802.3 standard for base band coaxial cables and twisted pair CSMA/CD bus LANs.



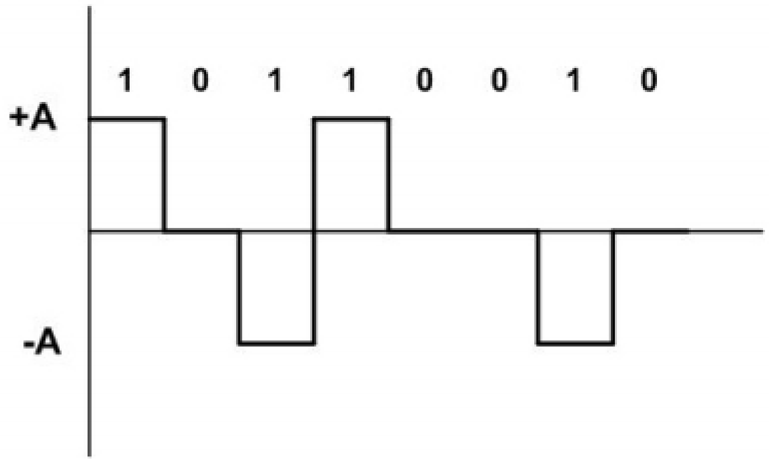
Manchester Encoding



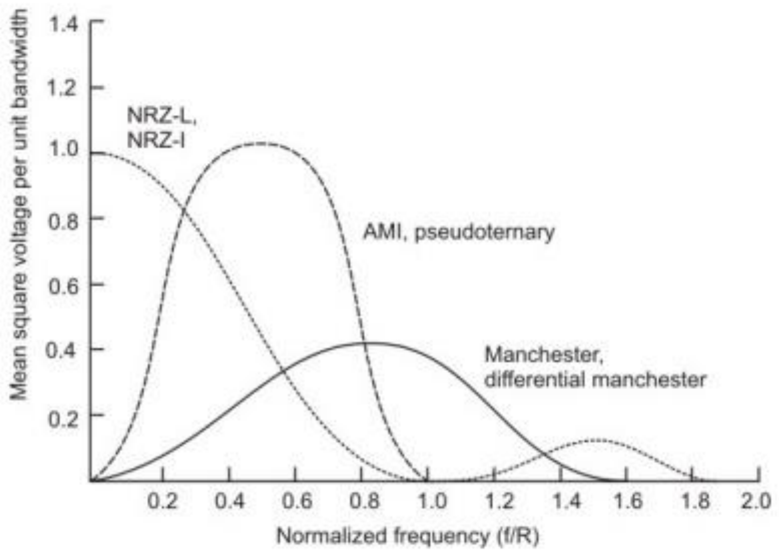
Differential Manchester Encoding



**Bipolar Encoding:** Bipolar AMI uses three voltage levels. Unlike RZ, the zero level is used to represent a 0 and a binary 1's are represented by alternating positive and negative voltages, as shown in Fig below:



Frequency spectrum of different encoding schemes have been compared in Fig below:



## 2B1Q Encoding

Linear codes such as Bi phase and AMI, allow the system to determine the next output baud without resorting to conversion look-up tables. Based on the current bit stream, the next output baud can be determined. **2B1Q** falls into this category.

2B1Q is a four-level pulse amplitude modulation (PAM-4) scheme without redundancy, mapping two bits (2B) into one quaternary symbol (1Q).

It takes two 2-level bits and converts then into one 4-level baud (quat) as indicated in Table 1.

The first bit of the dibit is called the “sign-bit”. If it is 0, the output quat will have a negative sign. If the first bit is 1, then the output quat will have a positive sign. The second bit of the dibit is called the amplitude bit, and it determines the magnitude of the output quat If it is 0, then the output level has an amplitude of 3. If the second bit of the dibit is 1, then the output amplitude is 1. This provides for a very simple means of encoding a binary bit stream into a 4-level code.

| DIBIT | OUTPUT QUAT |
|---|---|
| 10 | +3 |
| 11 | +1 |
| 01 | -1 |
| 00 | -3 |

Table 1. 2B1Q Coding Rules

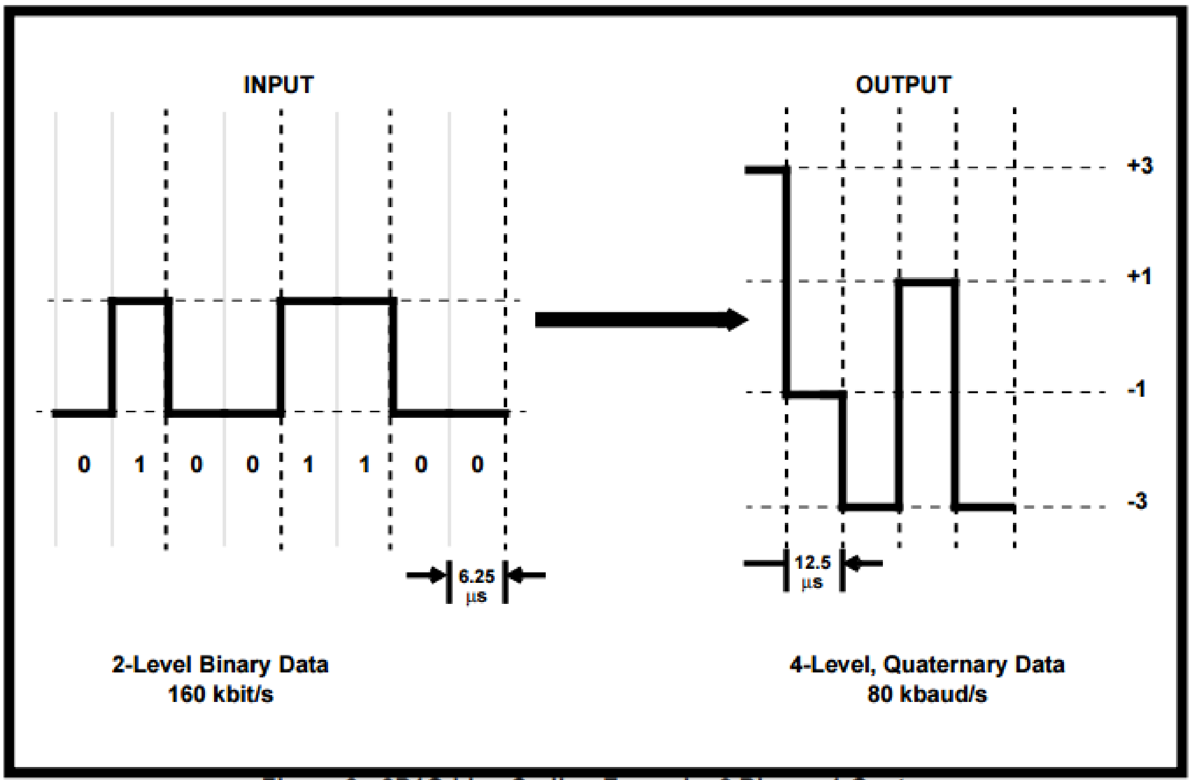


Figure 2 - 2B1Q Line Coding Example, 2 Binary, 1 Quaternary

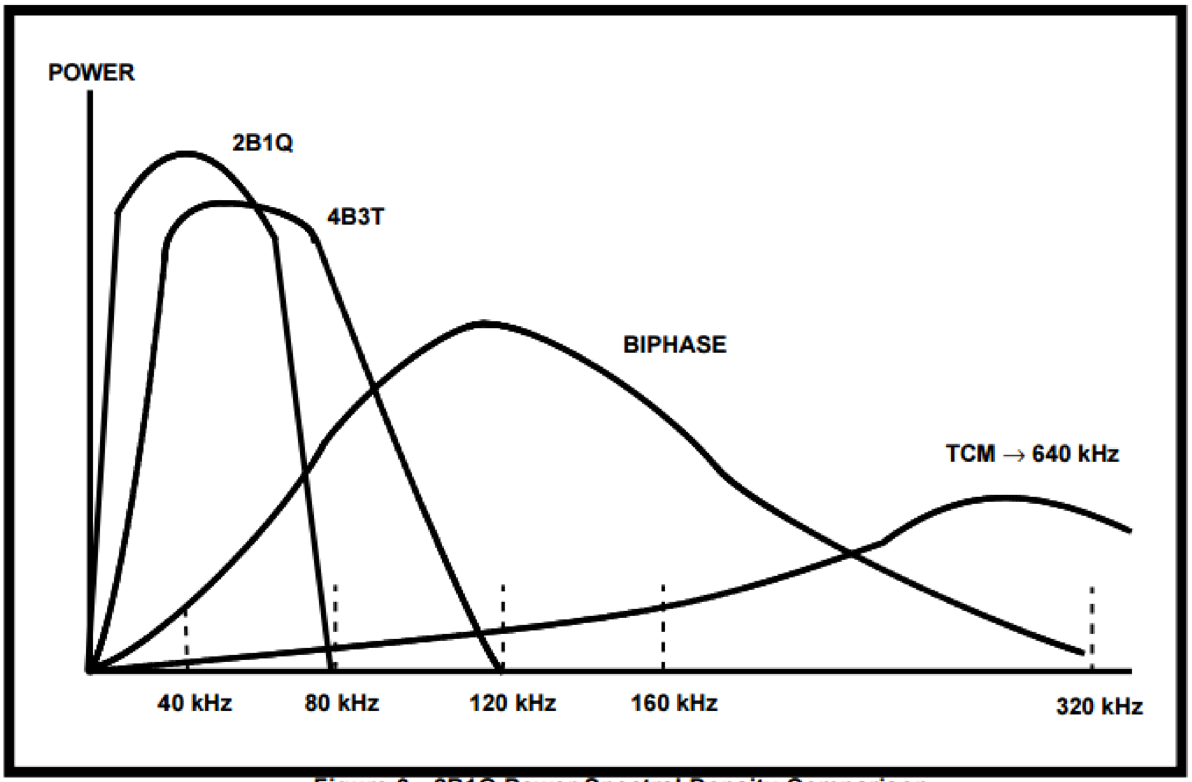


**Figure 3 - 2B1Q Power Spectral Density Comparison**

## 4B3T Encoding

4B3T, which stands for 4 (four) Binary 3 (three) Ternary, is a line encoding scheme used for ISDN PRI interface. 4B3T represents four binary bits using three pulses.

1. + (positive pulse),
2. 0 (no pulse), and
3. − (negative pulse)

This means we have 24 = 16 input combinations to represent, using 33 = 27 output combinations. 000 is not used to avoid long periods without a transition. 4B3T uses a paired disparity code to achieve an overall zero DC bias: six triplets are used which have no DC component (0+−, 0−+, +0−, −0+, +−0, −+0), and the remaining 20 are grouped into 10 pairs with differing disparity (e.g. ++− and −−+). When transmitting, the DC bias is tracked and a combination chosen that has a DC component of the opposite sign to the running total.

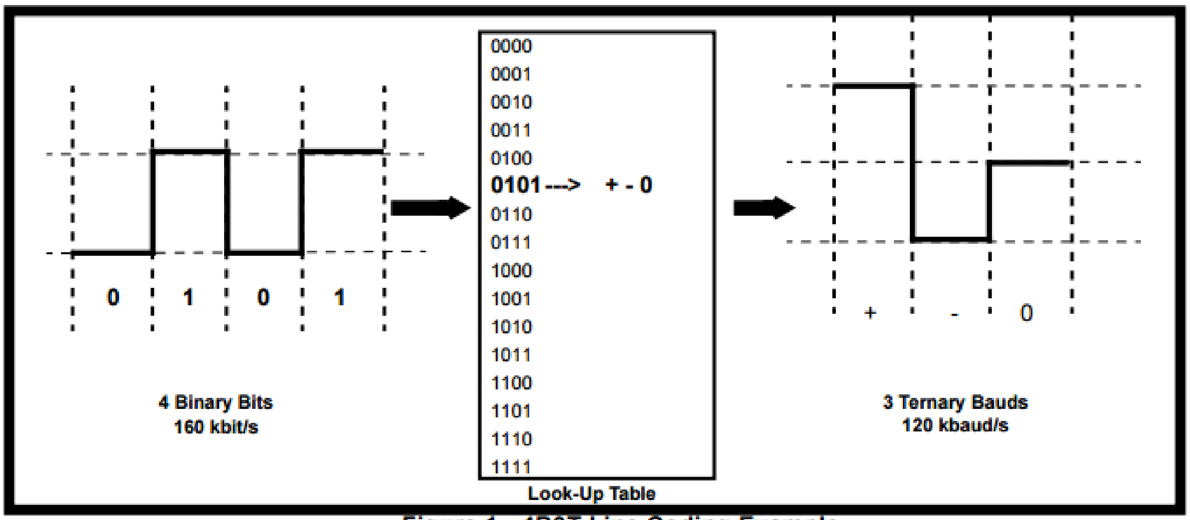


**Figure 1 - 4B3T Line Coding Example**

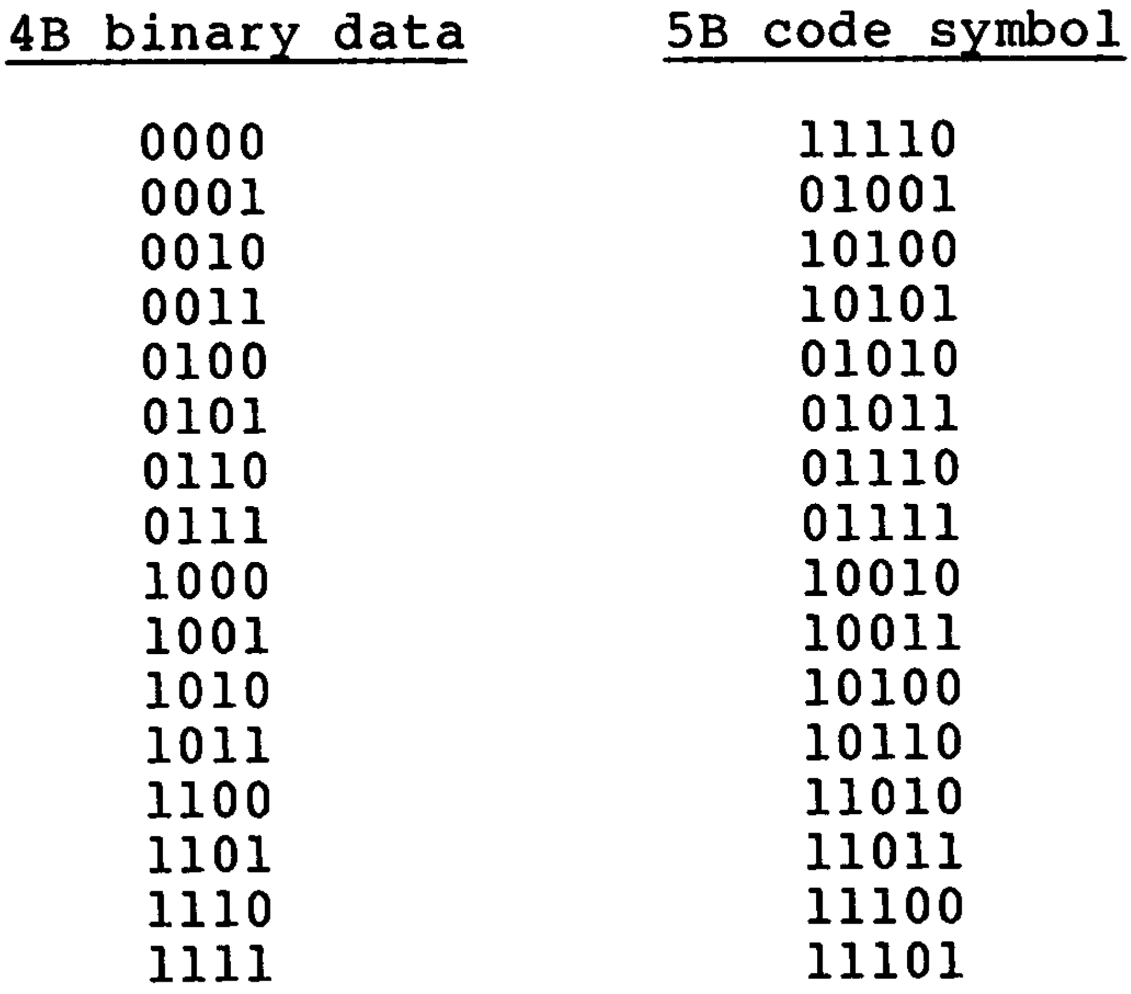
In four bits, we can have 16 possible numbers. In 3 ternary digits, we can have 27 possible numbers. We can represent each 4-bit number by a 3-ternary digit number, and have some 3-ternary digit numbers left over. Therefore, we allocate alternative codes to some of the binary numbers and use these (i) to keep the average signal level zero, and (ii) to ensure significant carrier content for receiver synchronization.

| BINARY | (a) | TERNARY | (b) |
|---|---|---|---|
| 0000 | – – – | | + + + |
| 0001 | – – 0 | | + + 0 |
| 0010 | – 0 – | | + 0 + |
| 0011 | 0 – – | | 0 + + |
| 0100 | – – + | | + + – |
| 0101 | – + – | | + – + |
| 0110 | + – – | | – + + |
| 0111 | – 0 0 | | + 0 0 |
| 1000 | 0 – 0 | | 0 + 0 |
| 1001 | 0 0 – | | 0 0 + |
| 1010 | | 0 + – | |
| 1011 | | 0 – + | |
| 1100 | | + 0 – | |
| 1101 | | – 0 + | |
| 1110 | | + – 0 | |
| 1111 | | – + 0 | |

Represent “+V” by "+", “–V” by "–", & “0 volts” by “0”. Choose either column (a) or column (b) where there is a choice. When decoded they give the same sequence of 4-bits. If the "accumulated disparity" is "+", i.e. if we have previously sent more "+" pulses than "–" pulses, choose column (a) to redress the balance. Otherwise choose column (b).

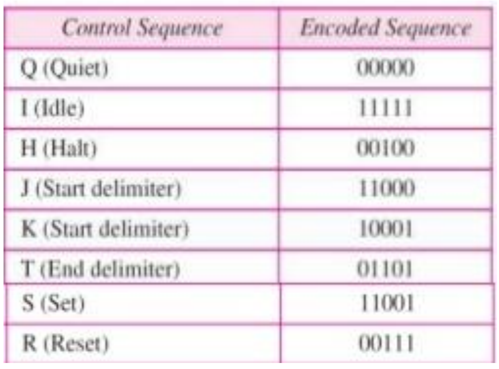
## 4B5B Encoding

4B/5B encoding is a type of 'Block coding'. This processes groups of bits rather than outputting a signal for each individual bit (as in Manchester encoding). A group of 4 bits is encoded so that an extra 5th bit is added. Since the input data is taken 4-bits at a time, there are 2^4, or 16 different bit patterns. The encoded bits use 5-bit, and hence have 2^5 or 32 different bit patterns.



| 4B binary data | 5B code symbol |
|---|---|
| 0000 | 11110 |
| 0001 | 01001 |
| 0010 | 10100 |
| 0011 | 10101 |
| 0100 | 01010 |
| 0101 | 01011 |
| 0110 | 01110 |
| 0111 | 01111 |
| 1000 | 10010 |
| 1001 | 10011 |
| 1010 | 10100 |
| 1011 | 10110 |
| 1100 | 11010 |
| 1101 | 11011 |
| 1110 | 11100 |
| 1111 | 11101 |

## Control Sequence



| *Control Sequence* | *Encoded Sequence* |
|---|---|
| Q (Quiet) | 00000 |
| I (Idle) | 11111 |
| H (Halt) | 00100 |
| J (Start delimiter) | 11000 |
| K (Start delimiter) | 10001 |
| T (End delimiter) | 01101 |
| S (Set) | 11001 |
| R (Reset) | 00111 |

## 6b/8b encoding

In telecommunications, **6b/8b** is a line code that expands 6-bit codes to 8-bit symbols for the purposes of maintaining DC-balance in a communications system.

Each 8-bit output symbol contains 4 zero bits and 4 one bits, so the code can, like a parity bit, detect all single-bit errors.

The number of binomial coefficient 8-bit patterns with 4 bits set is = 70. Further excluding the patterns 11110000 and 00001111, this allows 68 coded patterns: 64 data codes, plus 4 additional control codes.

### Coding rules

The 64 possible 6-bit input codes can be classified according to their disparity, the number of 1 bits minus the number of 0 bits:

| Ones | Zeros | Disparity | Number |
|---|---|---|---|
| 0 | 6 | −6 | 1 |
| 1 | 5 | −4 | 6 |
| 2 | 4 | −2 | 15 |
| 3 | 3 | 0 | 20 |
| 4 | 2 | +2 | 15 |
| 5 | 1 | +4 | 6 |
| 6 | 0 | +6 | 1 |

The 6-bit input codes are mapped to 8-bit output symbols as follows:

- The 20 6-bit codes with disparity 0 are prefixed with 10
  *Example: 000111 → **10**000111*
  *Example: 101010 → **10**101010*
- The 14 6-bit codes with disparity +2, other than 001111, are prefixed with 00
  *Example: 010111 → **00**010111*
- The 14 6-bit codes with disparity −2, other than 110000, are prefixed with 11
  *Example: 101000 → **11**101000*
- The remaining 20 codes: 12 with disparity ±4, 2 with disparity ±6, 001111, 110000, and the 4 control codes, are assigned to codes beginning with 01 as follows:

| Type | Input | Output | Type | Input | Output | Complement |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| −6 | 000000 | 010**11**00**1** | +6 | 111111 | 011**00**11**0** | 01_xx__x |
| −4 | 000001 | 01**11**0001 | +4 | 111110 | 01**00**1110 | 01xx____ |
| | 000010 | 01**11**0010 | | 111101 | 01**00**1101 | |
| | 000100 | 01**1**0010**1** | | 111011 | 01**0**1101**0** | 01x____x |
| | 001000 | 01**1**0100**1** | | 110111 | 01**0**1011**0** | |
| | 010000 | 010100**11** | | 101111 | 011011**00** | 01_____xx |
| | 100000 | 011000**11** | | 011111 | 010111**00** | |
| −2 | 110000 | 01110**1**00 | +2 | 001111 | 01001**0**11 | 01___x__ |
| Control | K 000111 | 01000111 | Control | K 111000 | 01111000 | |
| | K 010101 | 01010101 | | K 101010 | 01101010 | |

Obviously, no data symbol contains more than four consecutive matching bits, and because the patterns 11110000 and 00001111 are excluded, no data symbol begins or ends with more than three identical bits. Thus, the longest run of identical bits that will be produced is 6. (I.e. this is a (0,5) RLL code, with a worst-case running disparity of +3 to −3.)

## 8b/10b encoding

In telecommunications, **8b/10b** is a line code that maps 8-bit words to 10-bit symbols to achieve DC-balance and bounded disparity, and yet provide enough state changes to allow reasonable clock recovery. This means that the difference between the counts of ones and zeros in a string of *at least* 20 bits is no more than two, and that there are not more than five ones or zeros in a row. This helps to reduce the demand for the lower bandwidth limit of the channel necessary to transfer the signal.

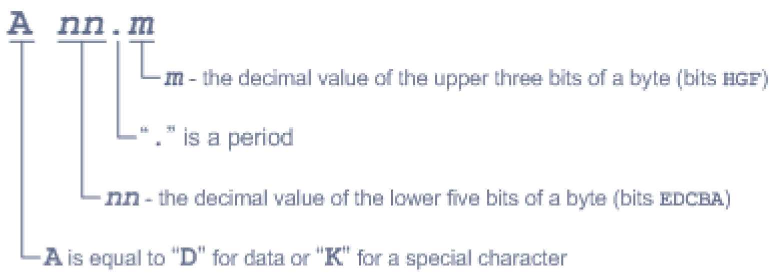
An 8b/10b code can be implemented in various ways, where the design may focus on specific parameters such as hardware requirements, DC-balance, etc.

As the scheme name suggests, eight bits of data are transmitted as a 10-bit entity called a *symbol*, or *character*. The low five bits of data are encoded into a 6-bit group (the 5b/6b portion) and the top three bits are encoded into a 4-bit group (the 3b/4b portion). These code groups are concatenated together to form the 10-bit symbol that is transmitted on the wire.

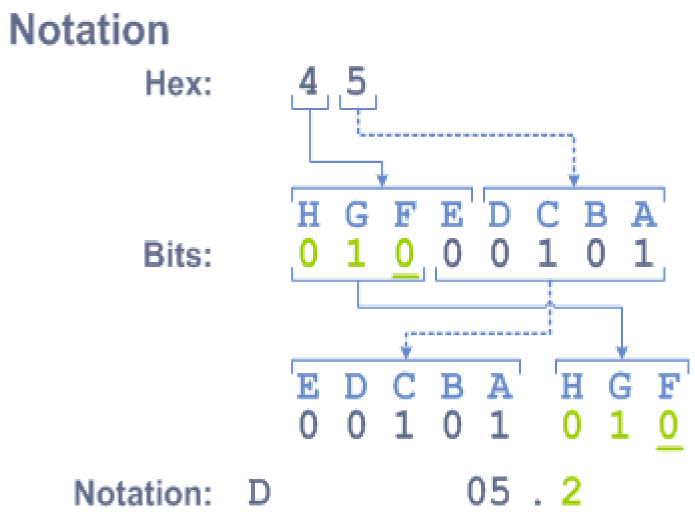
## Details of the Encoding Process

The 8b/10b encoding process also provides a number of 10-bit characters conforming to the coding rules which are not required for representing data characters (there are more available code points in the 10-bit space than the corresponding 8-bit data space). These 10-bit characters are available for use as special characters to identify control or management functions.

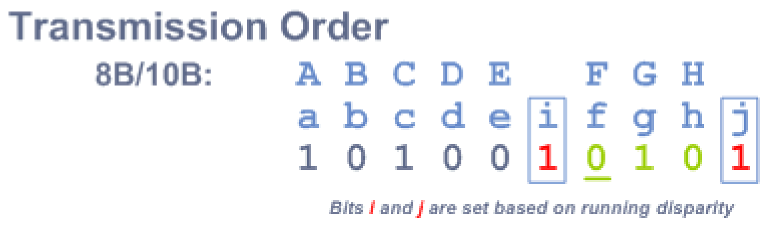
It is helpful to recognize the notation structure because you might see it in an error messages. The notation structure is:


A nn.m
m - the decimal value of the upper three bits of a byte (bits HGF)
"." is a period
nn - the decimal value of the lower five bits of a byte (bits EDCBA)
A is equal to "D" for data or "K" for a special character

As you can see, there are two types of characters, special characters and data characters. What follows is the process of encoding. It begins with a character to be transmitted. All that follows is in silicon.



A character enters and lives as a binary representation. The message is transmitted in this bit sequence:



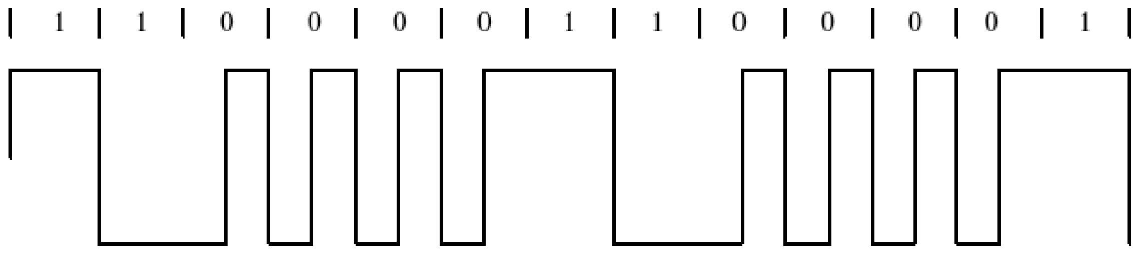
## Hamming Code

Hamming code is a set of error-correction code s that can be used to detect and correct bit errors that can occur when computer data is moved or stored. Hamming code makes use of the concept of parity and parity bit s, which are bits that are added to data so that the validity of the data can be checked when it is read or after it has been received in a data transmission.

## Coded Mark Inversion(CMI)

In telecommunication, coded mark inversion (CMI) is a non-return-to-zero (NRZ) line code. It encodes zero bits as a half bit time of zero followed by a half bit time of one, and while one bits are encoded as a full bit time of a constant level. The level used for one bits alternates each time one is coded.

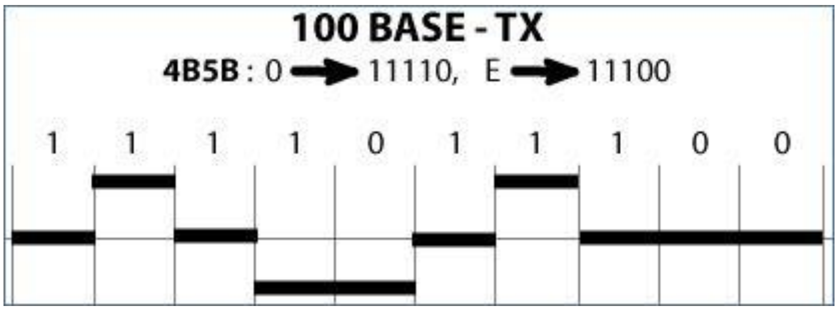
This code, designated as Bi ϕ-M, is similar to a Miller code in that a binary 1 is represented by a mid bit transition, and a binary 0 has no mid bit transition. However, this code always has a transition at the beginning of a bit period [10]. Thus, the code is easy to synchronize and has no dc. A Biphase(space) or Bi ϕ-S is similar to Bi ϕ-M, except the role of the binary data is reversed. Here a binary 0(space) produces a mid bit transition, and a binary 1 does not have a mid bit transition.  Both Bi ϕ-S and Bi ϕ-M are transition codes with memory.



## MLT 3 Encoding

MLT-3 encoding (Multi-Level Transmit) is a line code (a signaling method used in a telecommunication system for transmission purposes) that uses three voltage levels. An MLT-3 interface emits less electromagnetic interference and requires less bandwidth than most other binary or ternary interfaces that operate at the same bit rate (see PCM for discussion on bandwidth / quantization tradeoffs), such as Manchester code or Alternate Mark Inversion.

MLT-3 encodes a bit as presence or lack of transition, exactly as in NRZI. What makes MLT-3 different is that the base waveform is a 3-state alternating wave. Rather than alternating between 0 and 1 as in Manchester encoding and NRZI, MLT-3 alternates from -1 to 0 to +1, back to 0, then back to -1, repeating indefinitely. A zero is encoded as a halt in the back-and-forth progression. It may be useful to think of MLT-3 as a stop-and-go sine wave, encoding 0 as stop and 1 as go. Using MLT-3, it is therefore possible to represent four or more bits with every complete waveform, at 0, +1, 0, and -1.



The data being transmitted in this example is the hexadecimal byte "0E". First, the byte is broken down into four bit "nibbles", giving a nibble of "0" and a nibble of "E". Then, each nibble is looked up in a table to find the byte code associated with that hexadecimal number. The code for "0" hex is "11110" and the code for "E" hex is "11100". Finally, the byte codes are transmitted onto the wire using MLT-3 encoding.

## Conditioned diphase coding

Differential Manchester encoding Scheme is a line code in which data and clock signals are combined to form a single 2-level self-synchronizing data stream. It is a differential encoding, using the presence or absence of transitions to indicate logical value. It is not necessary to know the polarity of the sent signal since the information is not kept in the actual values of the voltage but in their change: in other words it does not matter whether a positive or negative voltage is received, but only whether the polarity is the same or different from the previous one; this makes synchronization easier. This encoding scheme is frequently called Differential Manchester since one way to describe it is applying Manchester encoding to differentiated data. Other names are biphase mark code (BMC) and Conditioned diphase.

## Miller Encoding/ Delay Encoding

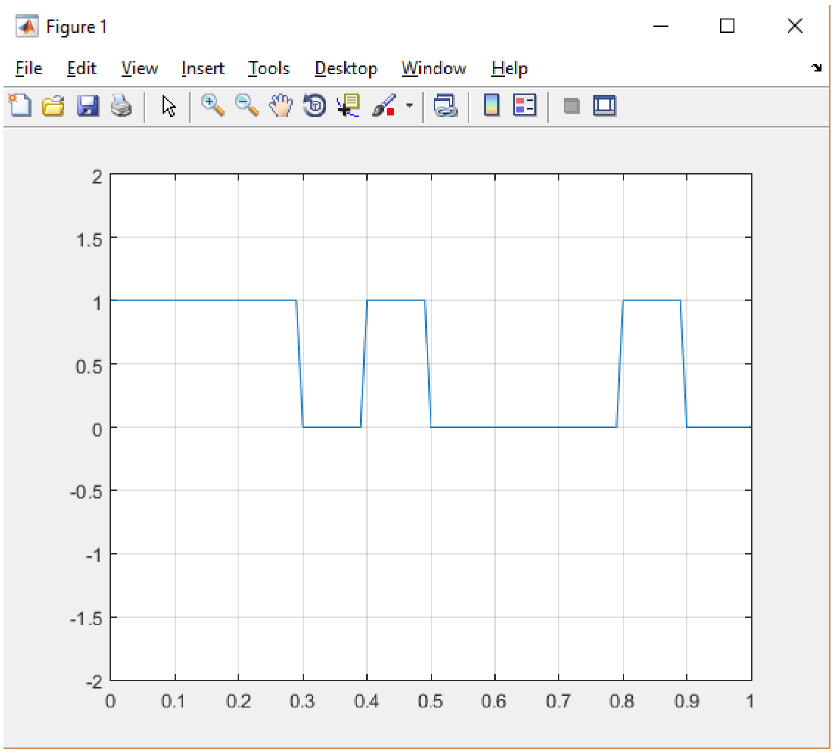
In telecommunications, **delay encoding** is the encoding of binary data to form a two-level signal where (a) a "0" causes no change of signal level unless it is followed by another "0" in which case a transition to the other level takes place at the end of the first bit period; and (b) a "1" causes a transition from one level to the other in the middle of the bit period.

# MATLAB Codes for each of the encoding

## Unipolar NRZ

```
function upnrz = unipolarnrz(n)    //n - no of samples
x=randint(1,n)
t=0:1/n^2:1;                       // t – time period
signal=zeros(1,length(t));
bit1=ones(1,n);
for i=1:length(x)
 if (x(i)==1)
   signal(1,i*n-(n-1):i*n)=bit1;
 else
   signal(1,i*n-(n-1):i*n)=0*bit1;
 end
 plot(t,signal),axis([0,1,-2,2]), grid on;
end

end
```

Input Sequence: 1 1 1 0 1 0 0 0 1 0

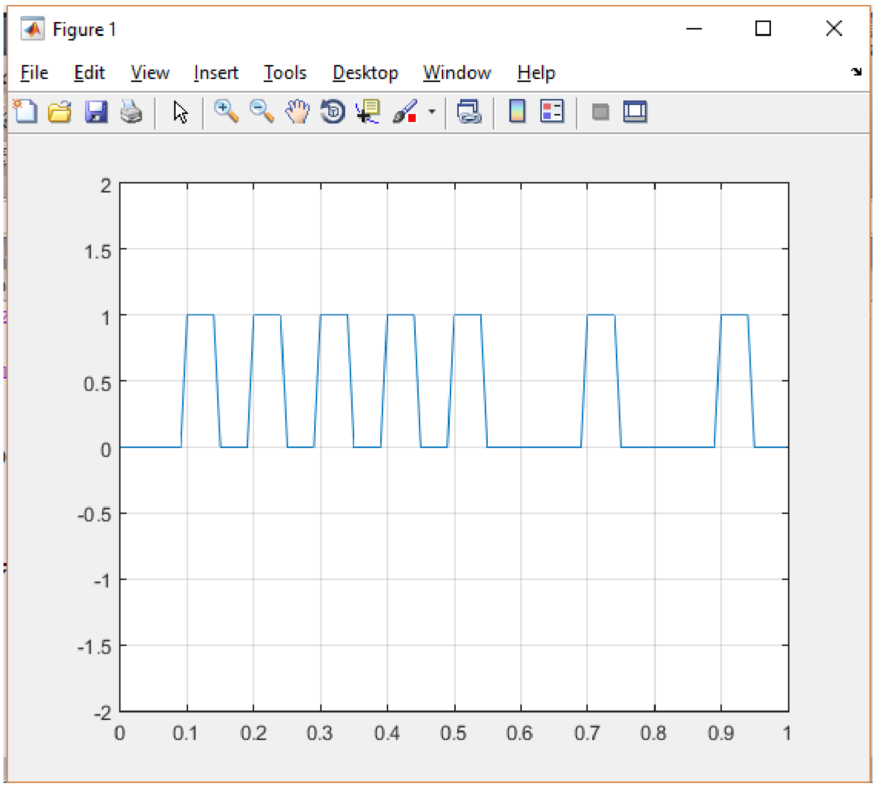


## Unipolar RZ

```
function uprz = unipolarrz(n)
x=randint(1,n)
t=0:1/n^2:1;
signal=zeros(1,length(t));
bit1=ones(1,floor(n/2));
bit2=ones(1,n);
for i=1:length(x)
 if (x(i)==1)
   signal(1,i*n-(n-1):i*n-(abs(n/2)))=bit1;
```

```
    signal(1,i*n-(abs(n/2)-1):i*n)=0*bit1;
  else
    signal(1,i*n-(n-1):i*n-(abs(n/2)))= 0 * bit1;
    signal(1,i*n-(abs(n/2)-1):i*n)=0*bit1;
  end
  plot(t,signal),axis([0,1,-2,2]), grid on;
end

end
```

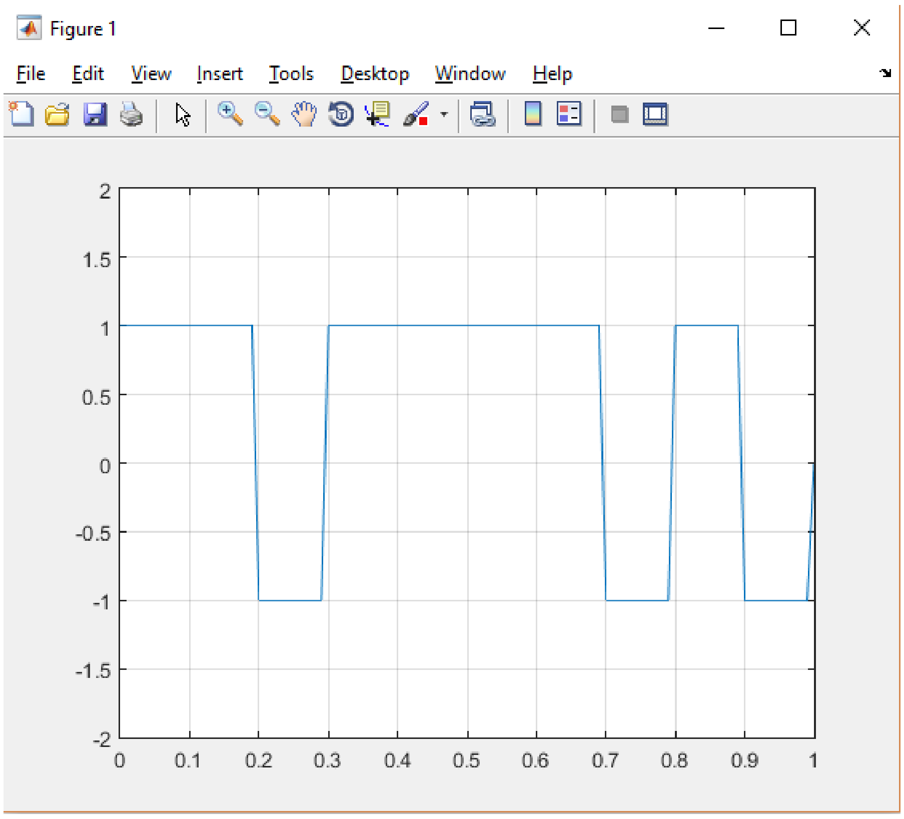
Input Sequence: 0 1 1 1 1 1 0 1 0 1



## Polar NRZ

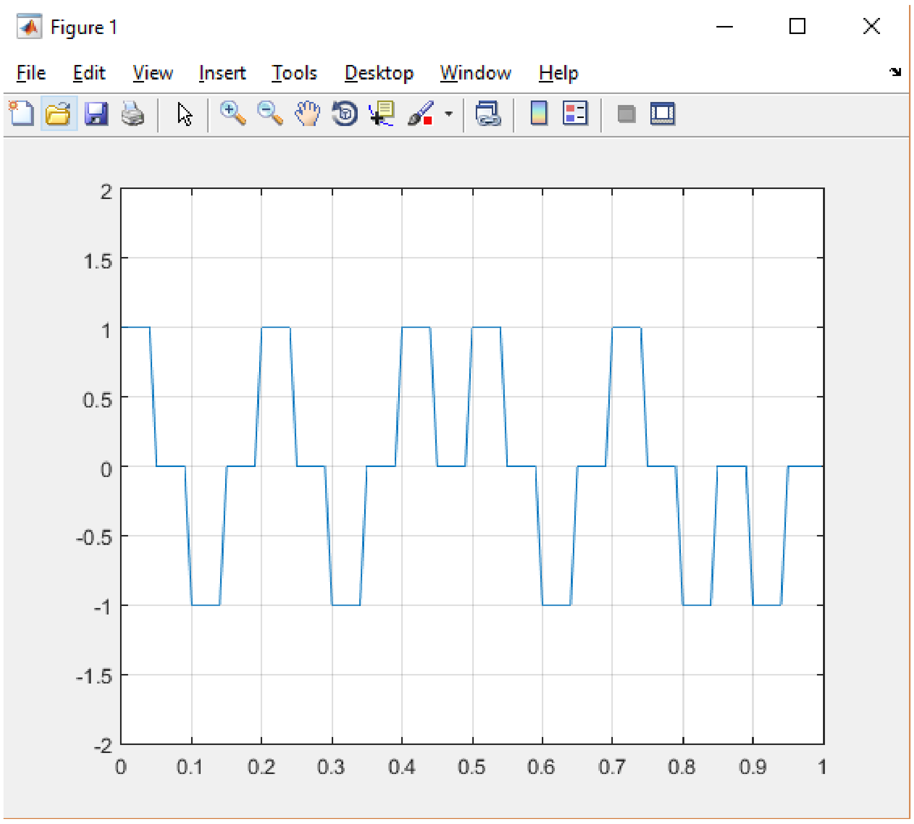
```
function pnrz = polarnrz(n)
x=randint(1,n)
t=0:1/n^2:1;
signal=zeros(1,length(t));
bit1=ones(1,n);
for i=1:length(x)
  if (x(i)==1)
    signal(1,i*n-(n-1):i*n)=bit1;
  else
    signal(1,i*n-(n-1):i*n)=-1*bit1;
  end
  plot(t,signal),axis([0,1,-2,2]), grid on;
end

end
```

Input sequence: 1 1 0 1 1 1 1 0 1 0



## Polar RZ

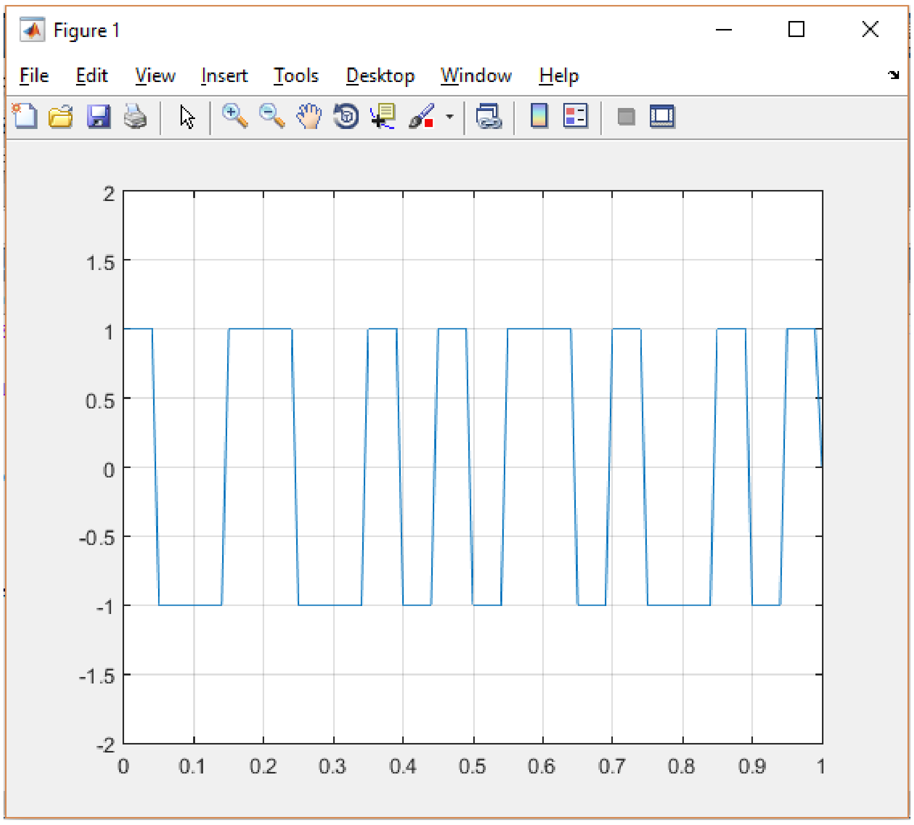
```
function prz = polarrz(n)
x=randint(1,n)
t=0:1/n^2:1;
signal=zeros(1,length(t));
bit1=ones(1,floor(n/2));
bit2=ones(1,n);
for i=1:length(x)
 if (x(i)==1)
   signal(1,i*n-(n-1):i*n-(abs(n/2)))= bit1;
   signal(1,i*n-(abs(n/2)-1):i*n)= 0*bit1;
 else
   signal(1,i*n-(n-1):i*n-(abs(n/2)))= -1*bit1;
   signal(1,i*n-(abs(n/2)-1):i*n)= 0 * bit1;

 end
 plot(t,signal),axis([0,1,-2,2]), grid on;
end

end
```



## Manchester Encoding

```
function mnch = manchester(n)
x=randint(1,n)
t=0:1/n^2:1;
signal=zeros(1,length(t));
bit1=ones(1,floor(n/2));
for i=1:length(x)
 if (x(i)==1)
   signal(1,i*n-(n-1):i*n-(abs(n/2)))=bit1;
   signal(1,i*n-(abs(n/2)-1):i*n)=-1*bit1;
 else
   signal(1,i*n-(n-1):i*n-(abs(n/2)))=-1*bit1;
   signal(1,i*n-(abs(n/2)-1):i*n)=bit1;
 end
 plot(t,signal),axis([0,1,-2,2]), grid on;
end

end
```

```
Input Sequence: 1      0      1      0      0      0      1      1      0      0
```



## 2b1q Encoding

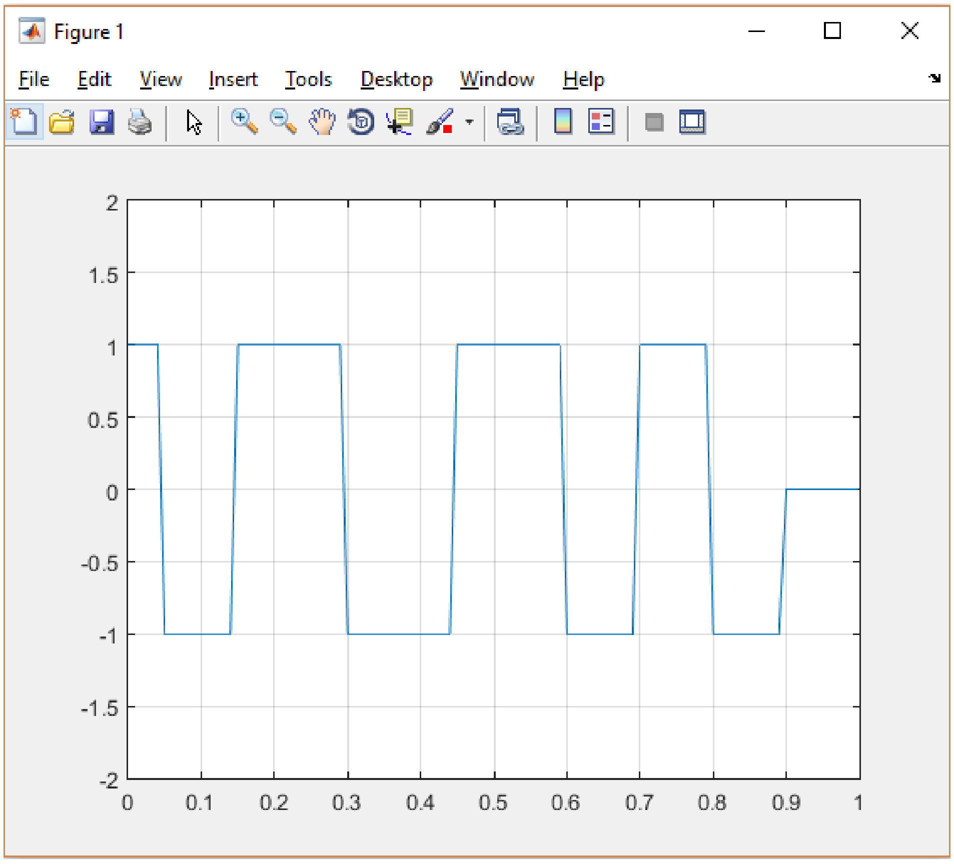
```
function c2b = c2b1q(n)
x=randint(1,n)
t=0:1/n^2:1;
signal=zeros(1,length(t));
bits1=ones(1,n);
for i=1:2:(length(x)-1)
    a=x(i); b=x(i+1);
    m=strcat(num2str(a),num2str(b));
    m=str2double(m);

    switch m
        case 00
            signal(1,i*n-(n-1):i*n)=-2;
        case 01
            signal(1,i*n-(n-1):i*n)=-1;
        case 10
            signal(1,i*n-(n-1):i*n)=2;
        case 11
            signal(1,i*n-(n-1):i*n)=1;
    end
end
plot(t,signal), axis([0,1,-3,3]),grid on;
```

## Delay Encoding/Miller Encoding

```
function mll = miller(n)
x=randint(1,n)
t=0:1/n^2:1;
```

```
signal=zeros(1,length(t));
bit1=ones(1,floor(n/2));
previous=bit1;
for i=1:(length(x)-1)
if x(i)==1
signal(1,i*n-(n-1):i*n-(abs(n/2)))=previous;
signal(1,i*n-(abs(n/2)-1):i*n)=-1*previous;
previous=-1*previous;
end
if x(i)==0 && x(i+1)~=0
signal(1,i*n-(n-1):i*n-(abs(n/2)))=previous;
signal(1,i*n-(abs(n/2)-1):i*n)=previous;
end
if x(i)==0 && x(i+1)==0
signal(1,i*n-(n-1):i*n-(abs(n/2)))=previous;
signal(1,i*n-(abs(n/2)-1):i*n)=previous;
previous=-1*previous;
end
end
plot(t,signal),axis([0 1 -2 2]),grid on;
```

Input Sequence:



## Conclusion

Thus the advancements in line coding have been very prosperous but each of the line codes have their own drawbacks and are used based on application. So a general ideal line code which can be used for all purposes is yet to be developed and if determined could pave a whole new path for digital encoding.

Further areas of research can be in how line coding can be replaced with something more dynamic and tuneable according to application.

## References

1) http://nptel.ac.in/courses/106105080/pdf/M2L4.pdf
2) http://www.myreadingroom.co.in/notes-and-studymaterial/68-dcn/719-different-line-coding-techniques.html
3) matlab-line-codes-digital-communication-systems-lab-report (1).pdf
4) http://www.frontiernet.net/~prof_tcarr/4B-5B/4B-5B.html
5) https://www.cse.iitk.ac.in/users/dheeraj/cs425/lec03.html
6) http://media.digikey.com/pdf/Data%20Sheets/Infineon%20PDFs/PEF%2080902.pdf
7) https://www.savvius.com/resources/compendium/fast_ethernet/signal_4b5b