**U19EC046 | DCOM | LAB 8**

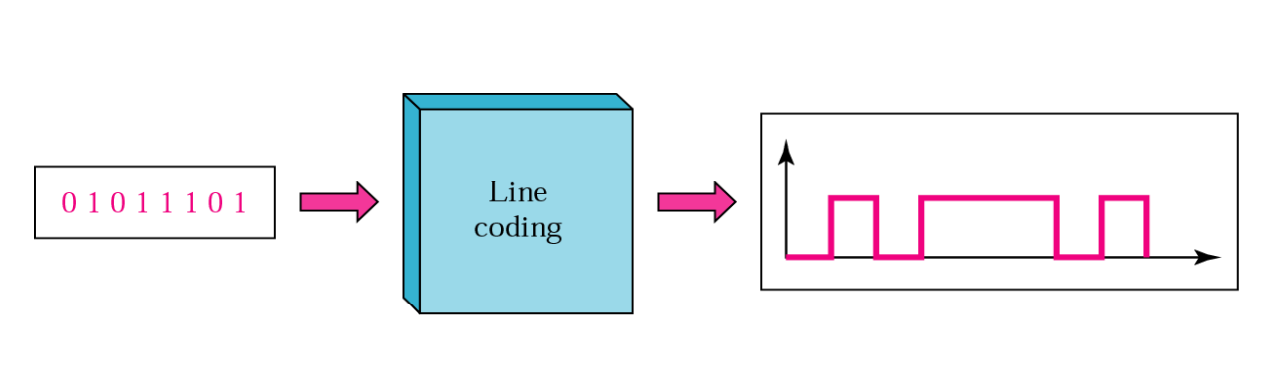
**Date: 18-10-2021**

**AIM**

To Study and simulate various Line Coding Techniques

**THEORY**

A **line code** is the code used for data transmission of a digital signal over a transmission line. This process of coding is chosen so as to avoid overlap and distortion of signal such as inter-symbol interference.



### Types of Line Coding

There are 3 types of Line Coding

* Unipolar
* Polar
* Bi-polar

## Unipolar Signaling

Unipolar signaling is also called as **On-Off Keying** or simply **OOK**.

The presence of pulse represents a **1** and the absence of pulse represents a **0**.

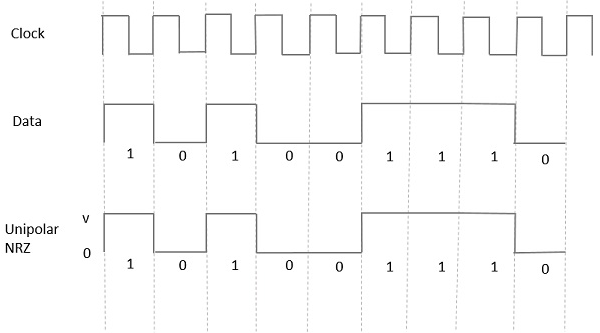
There are two variations in Unipolar signaling −

* Non Return to Zero NRZNRZ
* Return to Zero RZRZ

### Unipolar Non-Return to Zero NRZNRZ

In this type of unipolar signaling, a High in data is represented by a positive pulse called as **Mark**, which has a duration **T0** equal to the symbol bit duration. A Low in data input has no pulse.

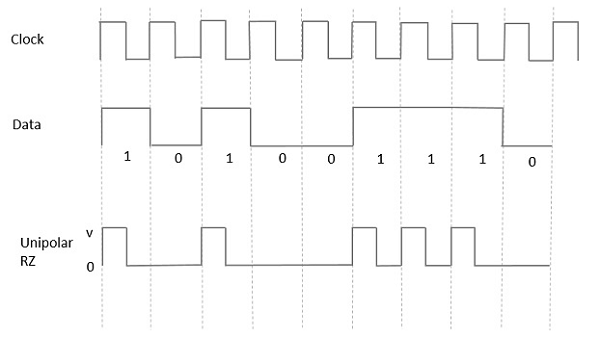
The following figure clearly depicts this.



### Unipolar Return to Zero RZRZ

In this type of unipolar signaling, a High in data, though represented by a **Mark pulse**, its duration **T0** is less than the symbol bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.

It is clearly understood with the help of the following figure.



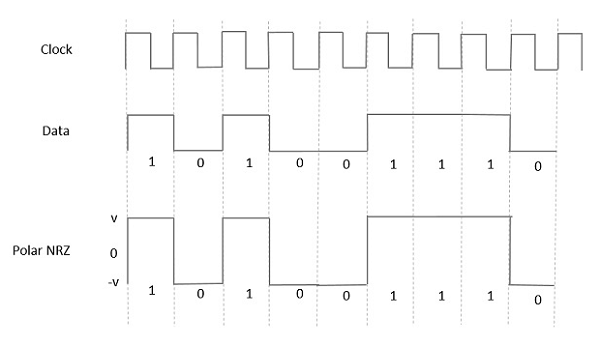
## Polar Signaling

There are two methods of Polar Signaling. They are −

* Polar NRZ
* Polar RZ

### Polar NRZ

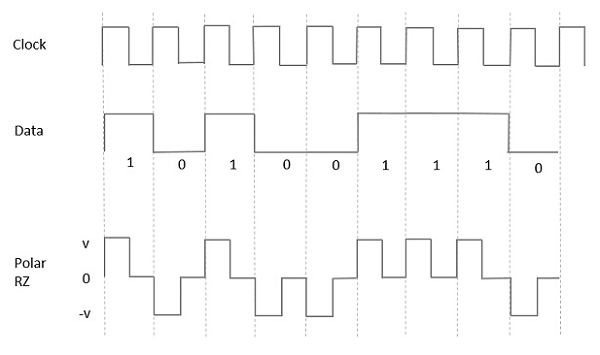
In this type of Polar signaling, a High in data is represented by a positive pulse, while a Low in data is represented by a negative pulse. The following figure depicts this well.



### Polar RZ

In this type of Polar signaling, a High in data, though represented by a **Mark pulse**, its duration **T0** is less than the symbol bit duration. Half of the bit duration remains high but it immediately returns to zero and shows the absence of pulse during the remaining half of the bit duration.

However, for a Low input, a negative pulse represents the data, and the zero level remains same for the other half of the bit duration. The following figure depicts this clearly.



## Bipolar Signaling

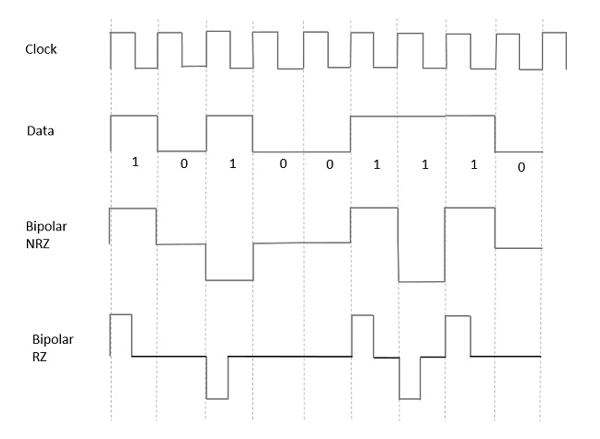
This is an encoding technique which has three voltage levels namely **+, -** and **0**. Such a signal is called as **duo-binary signal**.

An example of this type is **Alternate Mark Inversion**AMIAMI. For a **1**, the voltage level gets a transition from + to – or from – to +, having alternate **1s** to be of equal polarity. A **0** will have a zero voltage level.

Even in this method, we have two types.

* Bipolar NRZ
* Bipolar RZ

From the models so far discussed, we have learnt the difference between NRZ and RZ. It just goes in the same way here too. The following figure clearly depicts this.

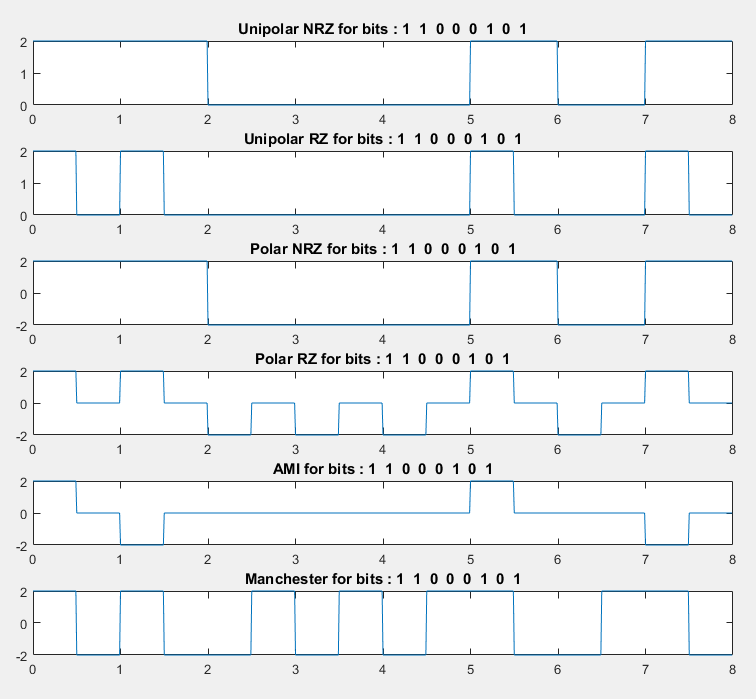


The above figure has both the Bipolar NRZ and RZ waveforms. The pulse duration and symbol bit duration are equal in NRZ type, while the pulse duration is half of the symbol bit duration in RZ type.

**MATLAB CODE**

|  |
| --- |
| ***//plotters.m***  **function op = plotters( bits, amp )**  **dt = 0.01;**  **t = 0 : dt : length(bits) - dt;**    **subplot(6, 1, 1)**  **plot(t, amp.\*unipolarNRZ(bits));**  **title(['Unipolar NRZ for bits : ' num2str(bits)]);**    **subplot(6, 1, 2)**  **plot(t, amp.\*unipolarRZ(bits));**  **title(['Unipolar RZ for bits : ' num2str(bits)]);**    **subplot(6, 1, 3)**  **plot(t, amp.\*polarNRZ(bits));**  **title(['Polar NRZ for bits : ' num2str(bits)]);**    **subplot(6, 1, 4)**  **plot(t, amp.\*polarRZ(bits));**  **title(['Polar RZ for bits : ' num2str(bits)]);**    **subplot(6, 1, 5)**  **plot(t, amp.\*AMI(bits));**  **title(['AMI for bits : ' num2str(bits)]);**    **subplot(6, 1, 6)**  **plot(t, amp.\*manchester(bits));**  **title(['Manchester for bits : ' num2str(bits)]);**  **end**  **function op = unipolarNRZ( bits )**  **one = ones(1, 100);**  **zero = zeros(1, 100);**  **graph = [];**  **for i = 1:length(bits)**  **if(bits(i)==1)**  **graph = [graph one];**  **else**  **graph = [graph zero];**  **end**  **end**  **op = graph;**  **end**  **function op = unipolarRZ( bits )**  **one = [ones(1, 50) zeros(1, 50)];**  **zero = zeros(1, 100);**  **graph = [];**  **for i = 1:length(bits)**  **if(bits(i)==1)**  **graph = [graph one];**  **else**  **graph = [graph zero];**  **end**  **end**  **op = graph;**  **end**  **function op = polarNRZ( bits )**  **one = ones(1, 100);**  **zero = zeros(1, 100) - 1;**  **graph = [];**  **for i = 1:length(bits)**  **if(bits(i)==1)**  **graph = [graph one];**  **else**  **graph = [graph zero];**  **end**  **end**  **op = graph;**  **end**  **function op = polarRZ( bits )**  **one = [ones(1, 50), zeros(1, 50)];**  **zero = [zeros(1, 50) - 1, zeros(1, 50)];**  **graph = [];**  **for i = 1:length(bits)**  **if(bits(i)==1)**  **graph = [graph one];**  **else**  **graph = [graph zero];**  **end**  **end**  **op = graph;**  **end**  **function op = AMI( bits )**  **one = [ones(1, 50), zeros(1, 50)];**  **onebar = [zeros(1, 50)-1 , zeros(1, 50)];**  **zero = zeros(1, 100);**  **graph = [];**  **flag = 1;**  **for i = 1:length(bits)**  **if(bits(i)==1)**  **if(flag == 1)**  **graph = [graph one];**  **flag = 0;**  **else**  **graph = [graph onebar];**  **flag = 1;**  **end**  **else**  **graph = [graph zero];**  **end**  **end**  **op = graph;**  **end**  **function op = manchester( bits )**  **one = [ones(1, 50), zeros(1, 50)-1];**  **zero = [zeros(1, 50) - 1, ones(1, 50)];**  **graph = [];**  **for i = 1:length(bits)**  **if(bits(i)==1)**  **graph = [graph one];**  **else**  **graph = [graph zero];**  **end**  **end**  **op = graph;**  **end**  ***//main.m***  **clc; clear all;**  **bits = input('Enter Bit vector : ');**  **amp = input('Enter Amplitude : ');**  **plotters(bits, amp);** |

**OUTPUT**



**CONCLUSION**

In this practical we studied different line coding techniques and also implemented them using matlab code.