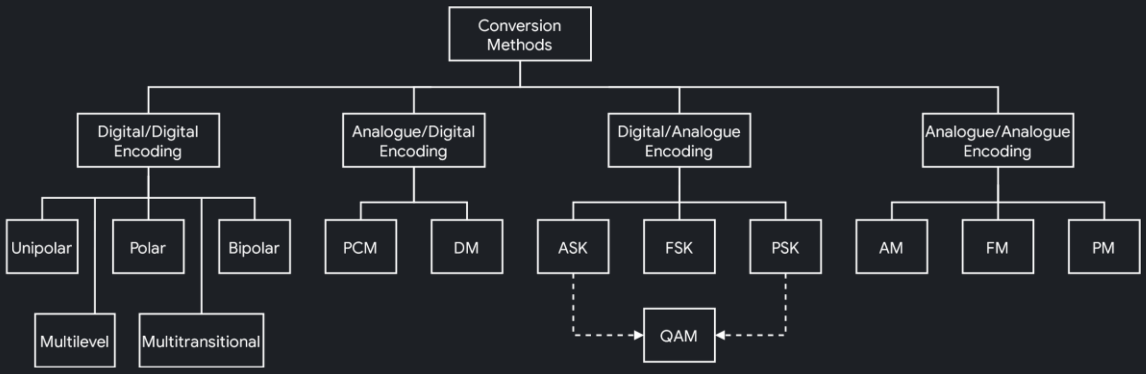
Chapter 4: Digital Transmission

Table of Contents

To be able to transmit data, we first have to convert it to either a digital or an analogue signal for transmission. Such conversions are handled by many different methods depending on which type of data is being converted to which type of signal.

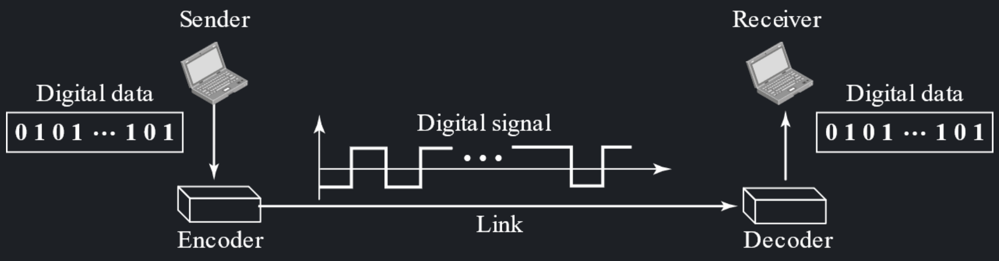


4.1 Digital to Digital Conversion

The conversion methods used to convert digital data to a digital signal involves three techniques: line coding, block coding and scrambling. Block coding and scrambling are not always required, but line coding is always required. As such, digital to digital conversion is also known as the line coding scheme, after the schemes used for line coding.

Line Coding

In line coding, a sequence of bits is converted to a digital signal via an encoder on the sender's end and a decoder does the opposite at the receiver's end.

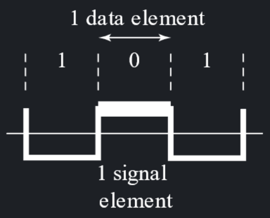


Before we can get into the details of how this works, we need to discuss a few topics.

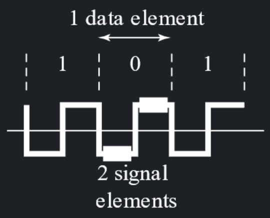
Signal Elements and Data Elements

A signal element carries data elements. It is the shortest unit of a digital signal.

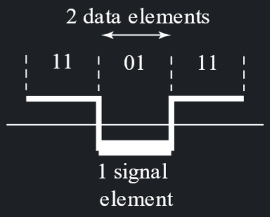
A data element is the smallest entity that can represent a piece of information, i.e. a bit.



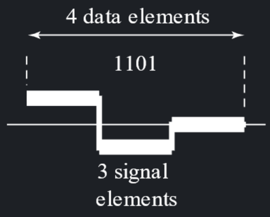
a. One data element per one signal element ($r = 1$)



b. One data element per two signal elements $\left(r = \frac{1}{2}\right)$



c. Two data elements per one signal element ($r = 2$)



d. Four data elements per three signa elements $\left(r = \frac{4}{3}\right)$

In the first figure, each data element is represented by a single signal element, thus, $r = 1$. In the second figure, 2 signal elements are used to represent a single data element. The advantage here is that this has a good DC balance, but we will come back to that issue later on. However, since the value of r is $\frac{1}{2}$ here, the required bandwidth is very high. In the third figure, a single signal element represents two data elements, meaning $r = 2$, and in the fourth figure, 3 signal elements represent 4 data elements, thus $r = \frac{4}{3}$.

The idea being presented here is that there is a ratio that we need to consider. We shall get into details about which ratios to use later on.

The number of data elements sent per second is called the data rate. It is measured in bits per second (bps).

The number of signal elements sent per second is called the signal rate. It is also called the pulse rate, the modulation rate or the baud rate. It is measured in bauds. It represents the required bandwidth to transmit the digital signal.

$$S = \frac{cN}{r}$$

where S is the signal rate, N is the data rate, r is the number of data elements carried by each signal element, and c is the case factor.

The case factor is an odd concept. It represents the best- or worst-case scenario. In the best case, when we are transmitting a series of 1s and 0s, we need the lowest frequency, 0. Thus $c = 0$. In the worst case, we are transmitting an alternating series of bits, such as 101010. Here, we need the highest frequency, and thus $c = 1$. Generally, an average case value is used, where $c = \frac{1}{2}$.

We know that the bandwidth of a digital signal is theoretically infinite, but of course, practically it is finite. The signal rate determines what this bandwidth will be.

The minimum bandwidth is given by: $B_{\text{minimum}} = \frac{cN}{r}$

The maximum data rate is given by: $N_{\text{maximum}} = \frac{Br}{c}$

DC Components

When the voltage level in a digital signal is constant for a while, i.e. a sequence of 1s or 0s is being transmitted, the spectrum creates very low frequencies. These frequencies, which are nearly 0, are called DC components.

DC components are a problem since some systems, like a transformer, will not allow them to pass. In other cases, the low frequencies will be too low to carry. For example, telephone lines cannot pass frequencies below 200Hz.

The line coding schemes we will study will show us how to avoid DC components.

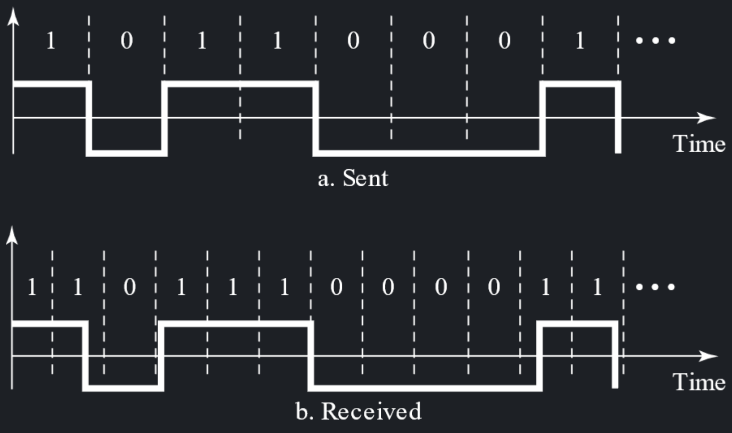
Baseline Wandering

When a digital signal is being received, the decoder compares the signal's voltage value to a baseline value at regular intervals. For example, if the voltage is $+5\text{V}$, it represents a 1.

However, the baseline is not some fixed value. Instead, the running average of the signal is used as the baseline. This is a problem, because due to the nature of digital signals, this baseline can sometimes shift. For example, if a series of 1s or 0s is continuously being received, the running average will begin to shift, and thus, so will the baseline. This is called baseline wandering.

Self-Synchronization

When a digital signal is being transmitted, the sender's clock and the receiver's clock must be synchronized. The receiver's bit intervals and the sender's bit intervals must be the same. If this is not done, the receiver will misinterpret the data being sent.



In these situations, as shown in the diagram, the receiver will be unable to tell when a data element starts and when it ends. This problem is magnified for a series of 1s and 0s.

To prevent this from happening, the signal itself should carry some synchronization information.

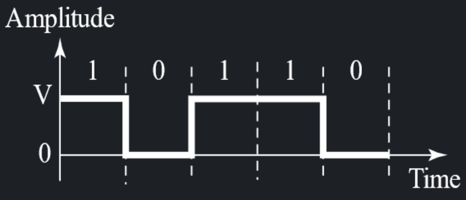
Line Coding Schemes

We can now begin our discussion on actual line coding schemes. The different line coding schemes we will be covering are:

- Unipolar – Here, only positive voltage is acknowledged. Anything else is considered to be 0. The only unipolar scheme we will be studying is non-return to zero (NRZ).
- Polar – Here, both positive and negative voltages are acknowledged. There are four polar schemes, non-return to zero (NRZ), return to zero (RZ), Manchester and differential Manchester. The last two are said to be bi-phase.
- Bipolar – These identify 3 levels, positive, negative and 0. In this category, we will learn alternate mark inversion (AMI) and pseudo-ternary. These are opposites of each other.
- Multi-Level – Instead of two levels in the signal, there may be multiple. There are three techniques in this category, 2-binary-1-quarternary (2B/1Q), 8-binary-6-ternary (8B/6T) and 4-dimnesional pulse amplitude modulation with level 5 (4D-PAM5).
- Multi-Transition – The only scheme in this category is MLT-3.

Unipolar Schemes

In unipolar NRZ, only one pole exists. 1s are represented using positive voltages and 0s are represented using 0 voltages.



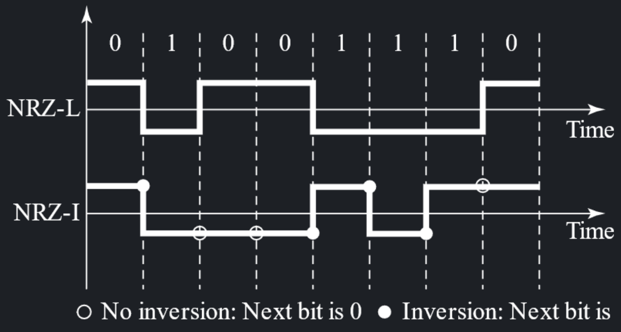
It is called non-return to zero (NRZ) since the signal elements themselves never come back to 0.

This scheme has lots of problems. It produces DC components, it has the synchronization problem and it has a very high normalized power. Normalized power is the powered required to send 1 bit per unit line resistance. For unipolar NRZ, this is given by $\frac{1}{2}V^2 + \frac{1}{2}(0)^2 = \frac{1}{2}V^2$.

Polar Schemes

Polar schemes use two voltage levels, a positive one and a negative one.

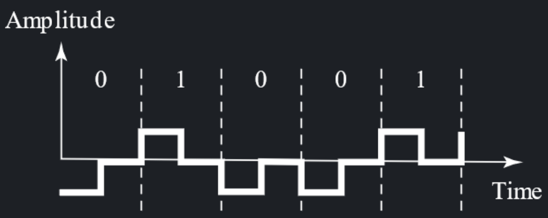
The NRZ polar scheme has two variations, non-return to zero-level (NRZ-L) and non-return to zero-invert (NRZ-I). In NRZ-L the level of the signal is dependant on the state of the bit. In NRZ-I the signal is inverted if a 1 is encountered.



NRZ-L is better than unipolar NRZ, but it is not free of problems. For example, DC components can still arise. Between NRZ-L and NRZ-I though, NRZ-I is better. If we have a stream of 0s, a DC component can occur, but for a stream of 1s, it will not. Thus, half the problem is solved.

The ratio r is 1 for polar NRZ, and the average signal rate is $\frac{N}{2}$, which is considered a good signal rate.

In the polar RZ scheme, every data element has a signal element that returns to 0 in the middle of the level. Thus, every data element spends half of its time at 0.

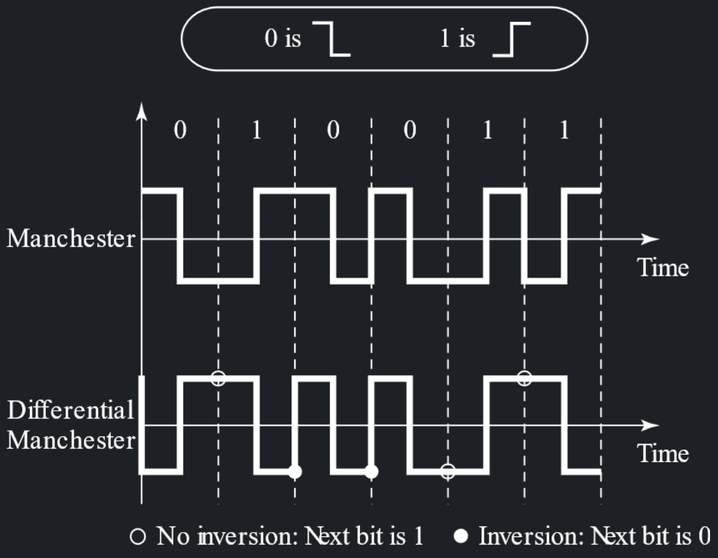


The advantage to doing this is that there is no DC component. Even if there is a series of 0 and 1s, the fact that the signal element returns to 0 in the middle of transmitting one bit means there is some change, thus avoiding the creation of DC components. The same fact also solves the synchronization problem, since the receiver can easily detect exactly when each individual bit starts and ends.

The required bandwidth of polar RZ is much higher though, since we need two signal elements to transmit a single data element. It has a ratio, r, of $\frac{1}{2}$ and an average signal rate of N.

Polar RZ is not really used, in lieu of better schemes.

Polar bi-phase schemes include Manchester and differential Manchester. Manchester is a better version of NRZ-L and differential Manchester is a better version of NRZ-I. They remove synchronization problems, but they have a bandwidth requirement of N.



In both Manchester and differential Manchester, the polarity of the signal is flipped in the middle of the bit.

In the Manchester scheme, for a 0 bit, we always start from a positive polarity and flip to a negative polarity in the middle of the bit. For a 1 bit, we do the opposite, starting from a negative polarity and flipping to a positive polarity in the middle of the bit.

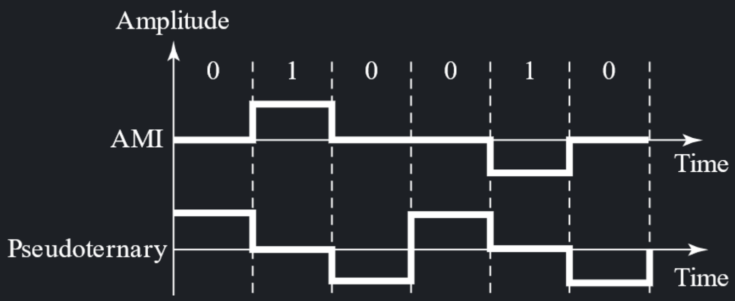
In differential Manchester, regardless of the current polarity, it is flipped if the next bit is a 0 and it is kept as it is if the next bit is 1. Keep in mind that the polarity is flipped in the middle of every bit as well, regardless of whether it is a 0 or a 1.

Also remember that which bit is represented by a positive polarity and which by a negative polarity and which bit will cause an inversion is always decided by us, even for other schemes. As long as the overall scheme is applied correctly, we can use whatever representation we want. The general convention has been used here, but it is not a given that this convention has to be followed.

Notice that the concept used in Manchester and differential Manchester schemes is similar to that used in polar RZ, yet these are considered far better than polar RZ. This is because polar RZ used the transition point to return to a 0 level. This means polar RZ consists of three different levels between which devices need to differentiate. This makes the process more complicated than Manchester and differential Manchester, which both use just two levels.

Bipolar Schemes

Bipolar encoding uses three levels – positive, negative and zero. There are two schemes under this category, Alternate Mark Inversion (AMI) and pseudoternary.



In telegraphy, the word ‘mark’ means 1. Thus, in Alternate Mark Inversion, every alternate 1 bit is inverted. This gives a good DC balance. With a long sequence of 1s, there is some self-synchronization as well. This self-synchronization is not available with a long sequence of 0s. However, the problem of DC components is completely avoided, since the only flat parts are caused by the 0s, but a zero level is not a DC component.

Pseudoternary is just the opposite of AMI, where a 0 bit is given a positive and negative voltage alternately, and a 1 bit is represented by a zero level.

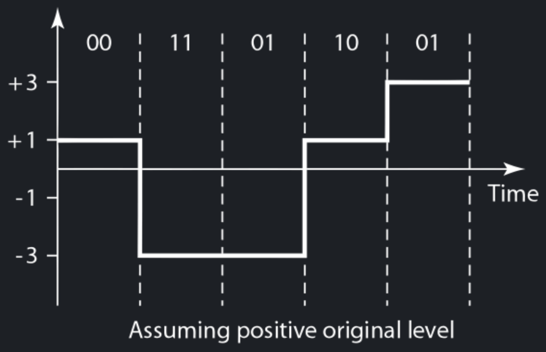
Both of these schemes have a bandwidth of $\frac{1}{2}\mathrm{N}$.

AMI can be useful for long distance communication due to its low bandwidth, but the problem with self-synchronization due to a long sequence of 0 bits still needs to be handled. We will come back to how this is done when we study scrambling.

Multilevel Schemes

With every scheme, our goal is to increase the data rate while decreasing the signal rate. This has resulted the creation of multilevel schemes. Here, m data elements are encoded into a pattern of n signal elements. Thus, these schemes will have names of the format mBnL in which $2^{\mathrm{m}} \leq \mathrm{L}^{\mathrm{n}}$. In this inequality, we are essentially saying that there should be more signal levels than data elements, e.g. for 3 bits of data, we need at least $2^3 = 8$ levels, given that $\mathrm{n} = 1$, meaning 3 bits are represented by 1 signal element. The B stands for binary by the way, because, well, binary data.

The first scheme we will study is $2\mathrm{B}1\mathrm{Q}$. This means 2 bits of data will be represented by 1 signal element, and there will be 'Quaternary', or 4 levels.



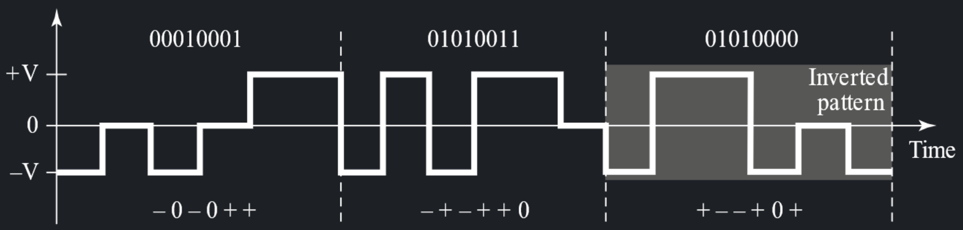
There are some transition rules associated with this diagram.

Previous Level:	Positive	Negative
Next Bits	Next Level	Next Level
00	$+1$	-1
01	$+3$	-3
10	-1	$+1$
11	-3	$+3$

The table is actually generated in a specific way, but for now, we can assume that the table is given and we generate the signal using it.

For 2B1Q, the ratio $\mathrm{r} = 2$. This causes the signal rate to be $\frac{\mathrm{N}}{4}$, which is great. However, this is still not a very good scheme since there is no redundancy. We have just enough levels required for 2 bits of data, $2^2 = 4$, which limits our capabilities a little, as we will see when we explore those capabilities in the later schemes.

The next scheme we will explore is 8B6T, where 8 bits of data are represented by 6 signal elements in 'Ternary', or 3 levels.



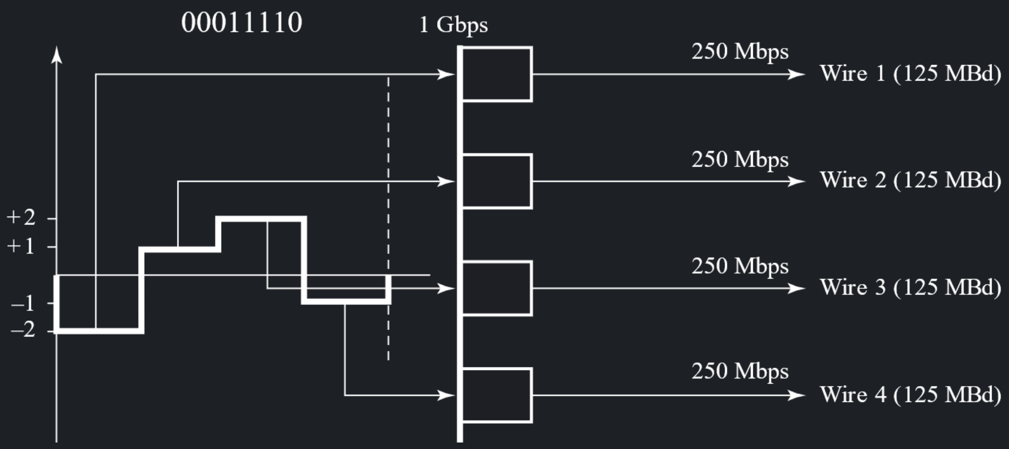
8 bits of data is $2^8 = 256$ different data elements. 6 signal elements spread over 3 levels give $3^6 = 729$ signal elements. We need to map 256 data elements to 729 signal elements. There is a table for this mapping, which we do not need to worry about. This mapping is done in such a way so as to give a good DC balance, so that the overall DC balance of each section is either positive or 0.

If two consecutive sections have a positive DC balance, the latter signal is inverted. In the example above, the first section had a zero DC balance, while the second and third both had positive DC balances. Thus, the third section is an inverted version of the actual encoding. This is done because the inverted portion will then have a negative DC balance, which will cause the overall DC balance of the entire signal to be neutralized.

Here, 8 bits are converted to 6 signal elements, so the signal rate is $\frac{1}{\frac{8}{6}\times 2}\mathrm{N} = \frac{3}{8}N$, which is still better than Manchester and differential Manchester, where the signal rate was N.

The next scheme is 4D-PAM5. The 4D means that data is sent over four wires at the same time. 5 voltage levels are used, but the zero level is only used for error detection. Thus, if we assume that the code is just one-dimensional, the result is something like $8B4Q$. Since 8 bits are translated to 4 signal elements, the signal rate is $\frac{1}{\frac{8}{4}\times 2}\mathrm{N} = \frac{1}{4}N$.

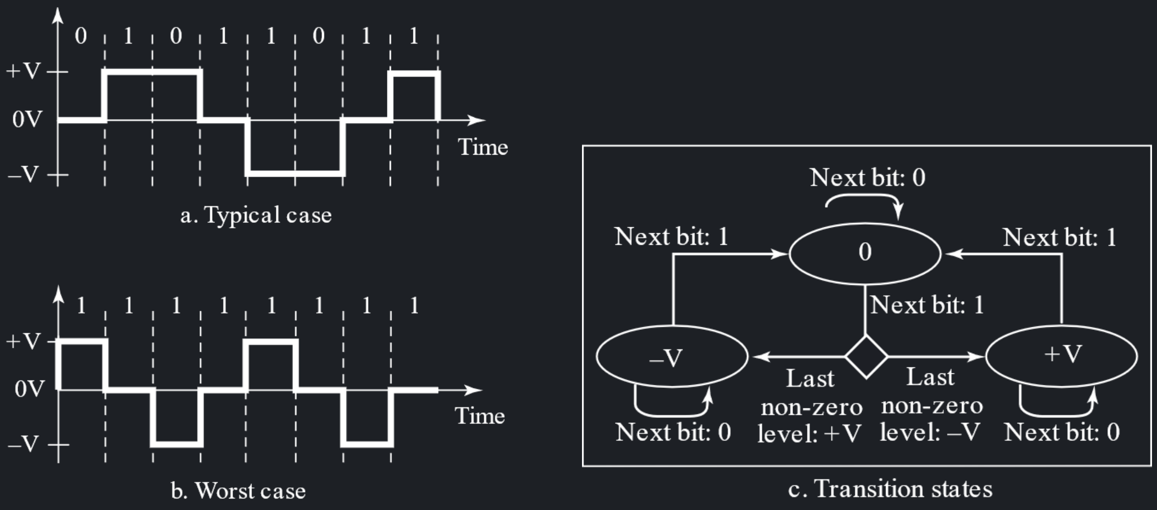
However, in reality, all the four levels of a single signal element are sent at the same time using the four wires. Thus, the actual signal rate is $\frac{1}{4} \times \frac{\mathrm{N}}{4}$.



This method is used by Gigabit LAN. There is a lot of redundancy (which is good) since 2^8 data patterns are mapped to $4^4 = 256$ signal patterns. The extra signal patterns can be used for error detection and other purposes.

Multiline Transmission

The final scheme we will learn is a multiline transmission scheme, MLT3. It is similar to NRZ-I and Manchester and differential Manchester, except that we have three levels here. The rules being used for this diagram are defined using a state transition diagram.



Essentially, a 0 bit causes no change. Whatever level we are in, we stay there. For a 1 bit, if we are in a positive or negative level, we will go to the 0 level. If instead we are in the 0 level when we get a 1 bit, we will go to the opposite of the last non-zero level.

If we consider the worst-case scenario, we will see that an aperiodic signal has been converted to a periodic one. The signal rate is also $\frac{\mathrm{N}}{2}$, which is great.

Using this scheme, we are able to send data at $100\mathrm{Mbps}$ on a copper wire that cannot support more than $32\mathrm{MHz}$. However, the synchronization problem for a sequence of 0 bits is still present here.

Summary of Line Coding Schemes

Category	Scheme	Bandwidth (average)	Characteristics
Unipolar	NRZ	$B = \frac{N}{2}$	Cost, no self-synchronization if long 0s or 1s, DC
Polar	NRZ-L	$B = \frac{N}{2}$	No self-synchronization if long 0s or 1s, DC
	NRZ-I	$B = \frac{N}{2}$	No self-synchronization for long 0s, DC
	Biphase	$B = N$	Self-synchronization, no DC, high bandwidth
Bipolar	AMI	$B = \frac{N}{2}$	No self-synchronization for long 0s, DC
Multilevel	2B1Q	$B = \frac{N}{4}$	No self-synchronization for long same double bits
	8B6T	$B = \frac{3N}{4}$	Self-synchronization, no DC
	4D-PAM5	$B = \frac{N}{8}$	Self-synchronization, no DC
Multitransitional	MLT-3	$B = \frac{N}{3}$	No self-synchronization for long 0s

Block Coding

The main technique of digital-to-digital conversion is line coding. The techniques we will learn now, block coding and, in the next section, scrambling, exist to assist line coding, not to replace it.

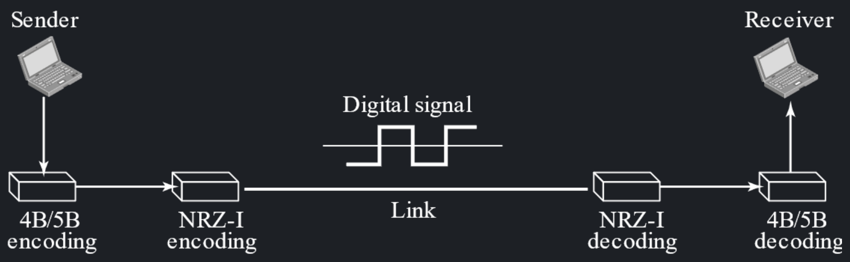
The main problems we saw with line coding was with long sequences of 0s and 1s, DC components and self-synchronization problems occurred. The main idea behind block coding is that the digital data should be changed in such a way so that such sequences do not exist.

Block coding is also sometimes called $\mathrm{mB/nB}$ coding, since m-bit groups are replaced with n-bit groups. We will be substituting one bit pattern with another, but to do this, we need some redundancy. The existence of this redundancy can also help us add some error-detecting capabilities.

The steps of transmission using block coding are:

1. Divide the sequence of bits into groups of m bits
2. Substitute each m-bit group with an n-bit one using a provided mapping table
3. Use line coding to create the signal

One of the most important block coding schemes is $4\mathrm{B}/5\mathrm{B}$, meaning 4-bit groups with 16 possible patterns are replaced by 5-bit groups with 32 different patterns. The extra patterns available is what we mean by redundancy.



Notice that, in the above diagram, NRZ-I is being used. In the last section, we learnt that this is a good line coding scheme, with a low signal rate of $\frac{\mathrm{N}}{2}$. However, it did face problems with long sequences of 0s, problems that we can now solve using block coding.

Data Sequence	Encoded Sequence	Control Sequence	Encoded Sequence
`0000`	`11110`	`Q` (Quiet)	`00000`
`0001`	`01001`	`I` (Idle)	`11111`

0010	10100	H (Halt)	00100
0011	10101	J (Start delimiter)	11000
0100	01010	K (Start delimiter)	10001
0101	01011	T (End delimiter)	01101
0110	01110	S (Set)	11001
0111	01111	R (Reset)	00111
1000	10010		
1001	10011		
1010	10110		
1011	10111		
1100	11010		
1101	11011		
1110	11100		
1111	11101		

From the mapping table, it can be seen that there are 15 sequences that are used directly with the 15 possible 4-bit data patterns and another 8 that are used for control purposes. A special feature of the encoded sequences for the data sequences is that none of the 5-bit codes have more than one leading 0 or more than two trailing 0s. This ensures that there is at most three consecutive 0s in the output. As a result, long sequences of 0s are avoided.

Since we are adding more bits, we will end up needing a higher bandwidth to transmit the data. This is a problem, which is why we use a line coding scheme like NRZ-I, which has a good signal rate.

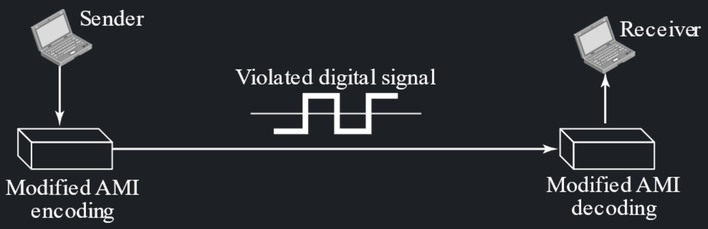
Note that block coding does not convert digital data to a digital signal. That is still the job of line coding. Block coding takes digital data and gives us digital data back.

There is another block coding scheme called $8\text{B}/10\text{B}$, which takes groups of 8-bit data and converts each group into a 10-bit code. This is actually made up of two smaller block coders, $5\text{B}/6\text{B}$, which deals with the leftmost 5 bits from the original sequence, and $3\text{B}/4\text{B}$, which deals with the rightmost 3 bits from the original sequence. The extra bits generated by each part can also cause problems, with both producing an extra 1 or 0, which would again lead to a long sequence of 0s or 1s. Such problems are handled by a disparity controller.

Scrambling

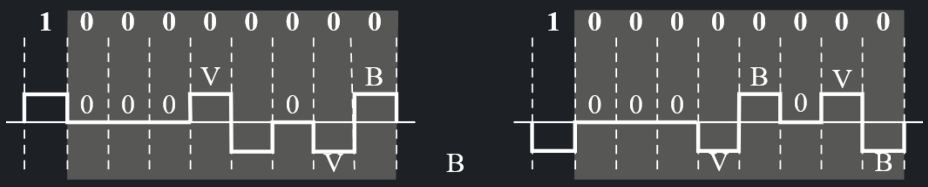
With block coding, there is a lot of overhead, since we have to do a lot of extra work. Scrambling on the other hand, is far simpler. It is done during line coding.

There are two common scrambling techniques, Bipolar 8 Zero Substitution ($\text{B8Z}S$) and High-Density Bipolar 3 Zero Substitution HDB3. If we have 8 consecutive 0s and we are using a bipolar line coding scheme, then we can use $\text{B8Z}S$ to substitute those 0s. HDB3 essentially does the same thing, but with 4 consecutive 0s. HDB3 is more aggressive than $\text{B8Z}S$.



If we consider the AMI scheme, it had no problems with DC components, just with self-synchronization due to the long sequence of 0s. This, together with the low signal rate of $\frac{\text{N}}{2}$, made it a good candidate for long distance communication. AMI, used along with scrambling, would make it perfect.

$\mathrm{B8ZS}$



a. Previous level is positive. b. Previous level is negative.

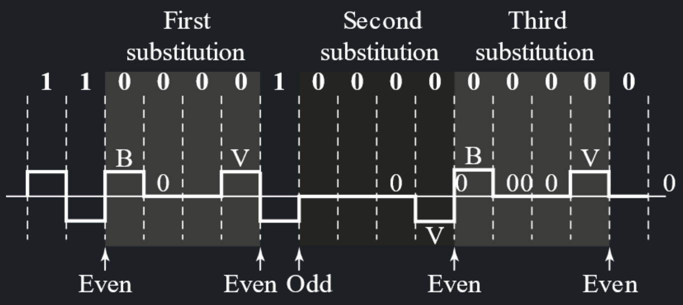
$\mathrm{B8ZS}$ substitutes a 8 consecutive 0s with $000VB0VB$, where V stands for violation and B stands for bipolar. Violation means we take the last non-zero value and violate the AMI rule. Bipolar means we take the last non-zero value and follow the AMI rule.

Consider the diagram on the left. According to the AMI rule, if we had a 1 in place of the first V in $000VB0VB$, we should have a negative signal element. However, since this is a violation, we have a positive one. Next, with the first bipolar, we follow the AMI rule, giving a negative signal element since the last non-zero value (the violation) was positive. The diagram on the right works similarly.

Using this formula, the data will not have more than three consecutive 0s.

HDB3

HDB3 is used in Europe, as opposed to B8ZS, which is used in North America. It is more aggressive than its counterpart.



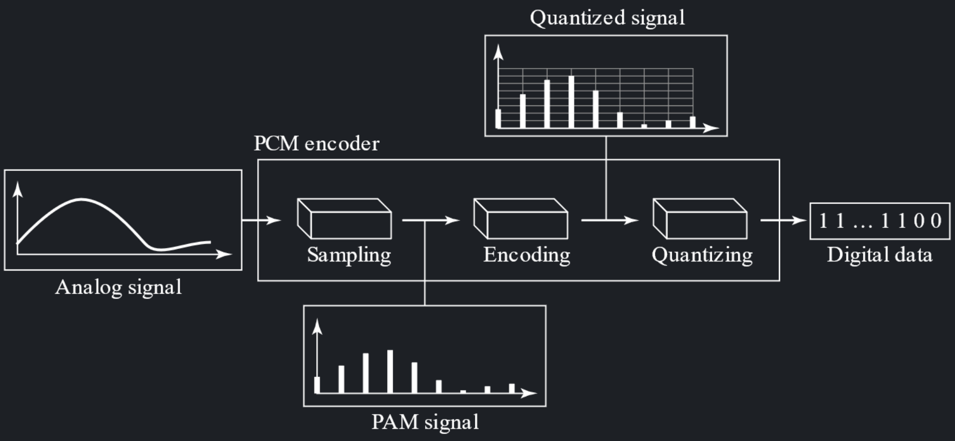
Here, the rule is, if we have an even number of non-zero signal elements after the last substitution, the 4 consecutive 0s will be replaced with B00V. If there are an odd number of consecutive non-zero elements after the last substitution, the 4 consecutive 0s will be replaced by 000V.

4.2 Analogue-to-Digital Conversion

We essentially have two approaches to converting analogue data to digital signals, pulse code modulation (PCM) and delta modulation. PCM is the more common one.

Pulse Code Modulation

The process of changing an analogue signal to digital data is called digitization. The process of digitization through PCM looks like this.

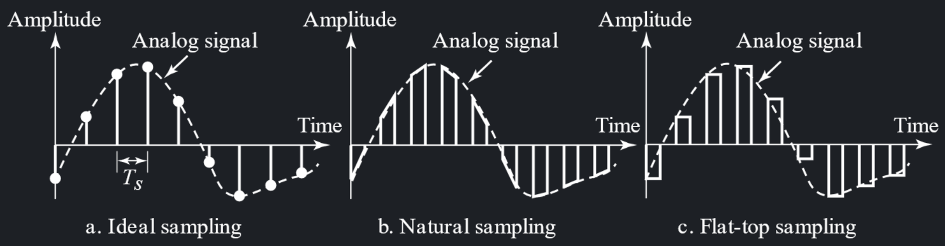


Analogue signals contain infinitely many levels. It is not possible for us to actually take all of those levels, so we have to approximate. Essentially, we are taking pulses and encoding them. This is where Pulse Amplitude Modulation (PAM), the sampling part of the PCM encoder, comes in. We use it take samples of the analogue signal. The more frequently we take samples, the better we will be able to represent the signal. Of course, this will also increase the bandwidth requirement. We need to at least take enough samples to allow us to create a good approximation of the actual analogue signal.

After we have taken samples, we need to map them. This is done through quantization. Essentially, we need to decide to what degree of accuracy we want to represent the samples we took. There is an error here, which will decrease if we take a higher level of accuracy. This error is called the quantization error.

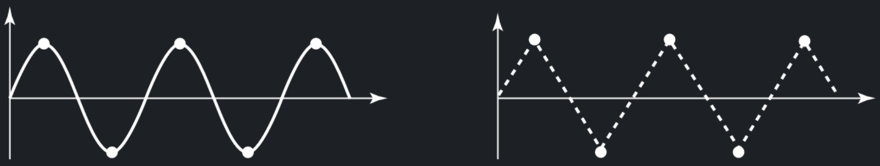
Finally, we simply convert the quantized levels of the samples to binary data via encoding.

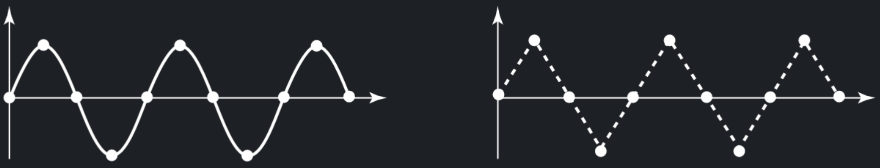
Sampling

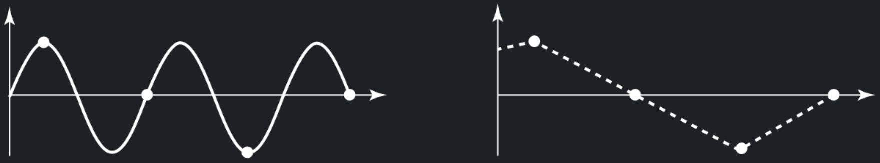


There are essentially three types of sampling, Ideal sampling, where a specific value at a specific moment is recorded, natural sampling, where a range of values are recorded, and flat-top sampling, where a sample is taken at a moment and held for some time. For our purposes, we will assume that ideal sampling is used, even though it is more difficult to achieve than the other methods.

The Nyquist theorem gives us a good way to tell how frequently we need to take samples. It says that the sampling rate should be at least twice the highest frequency of the contained signal (not the bandwidth, the maximum frequency).

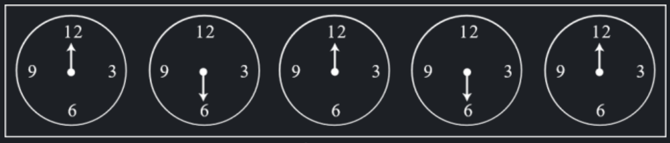

a. Nyquist rate sampling: $f_s = 2f$

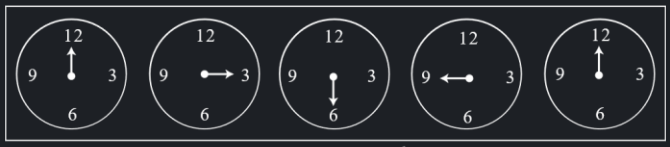

b. Oversampling: $f_s = 4f$

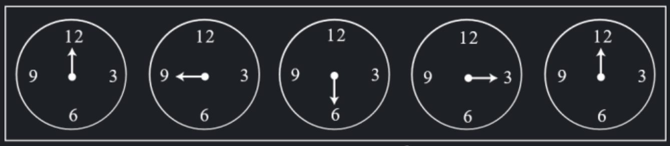

c. Undersampling: $f_s = f$

From the above diagrams, we can tell that the Nyquist theorem is right. Sampling more often does not give us an improvement since oversampling occurs, while sampling less often does not accurately capture the signal, since under-sampling occurs.

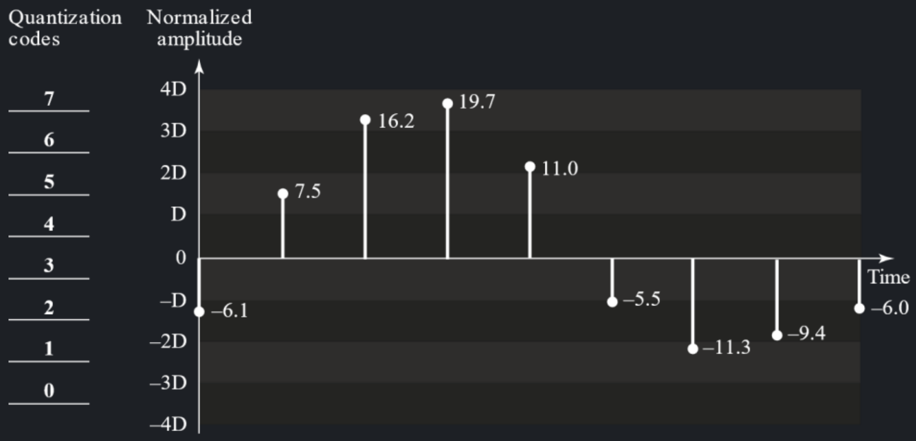
If we consider a video of a clock, at Nyquist sampling rate, we will not be able to tell if the clock is moving forwards or backwards. If oversampling is done, it will be clear that the clock is moving forwards. However, with under-sampling, it will appear that the clock is moving backwards.


a. Sampling at Nyquist rate: $T_s = T\frac{1}{2}$


b. Oversampling (above Nyquist rate): $T_s = T\frac{1}{4}$


c. Undersampling (below Nyquist rate): $T_s = T\frac{3}{4}$

Quantization



The first step is to normalize the PAM values. Here, the range of values are from -20V to $+20\text{V}$ (assumed) and it is decided that there should by 8 levels. For human voice, we would need 256 levels. The number of levels is our decision. Thus, each level has a height, Δ, of $\frac{20+20}{8} = 5$. The normalized PAM values are given by dividing each PAM value by 5.

Next, we quantize the values, essentially rounding them. This is done using the mid values between two levels. For example, the mid value between D and 2D is $\frac{7.5}{5} = 1.5$, so any values between D and 2D will be quantized to 1.5.

Of course, there is some error here. The closer the value is to the edge of a level, the higher the error. However, this error will not be greater than half of the level. For this example, the maximum quantization error is 0.5V. The quantization error in dB is given by $\text{SNR}_{\text{dB}} = 6.02\text{n}_\text{b} + 1.76\,\text{dB}$, where n_b is the number of bits per sample. For this example, $\text{n}_\text{b} = 3$ since there are $2^3 = 8$ levels.

Finally, the values are encoded to digital data. This is done by using the corresponding quantization codes for the respective levels.

The entire process results in the following table:

Normalized PAM Values	-1.22	1.50	3.24	3.94	2.20	-1.10	-2.26	-1.88	-1.20

Normalized Quantized Values	-1.50	1.50	3.50	3.50	2.50	-1.50	-2.50	-1.50	-1.50
Normalized Error	-0.38	0	$+0.26$	-0.44	$+0.30$	-0.40	-0.24	$+0.38$	-0.30
Quantization Error	2	5	7	7	6	2	1	2	2
Encoded Words	010	101	111	111	110	010	001	010	010

Encoding

We have already seen how encoding is done. It is a simple enough process. Instead, let’s consider an example. What should the bit rate of the digitization of the human voice be assuming 8 bits per sample?

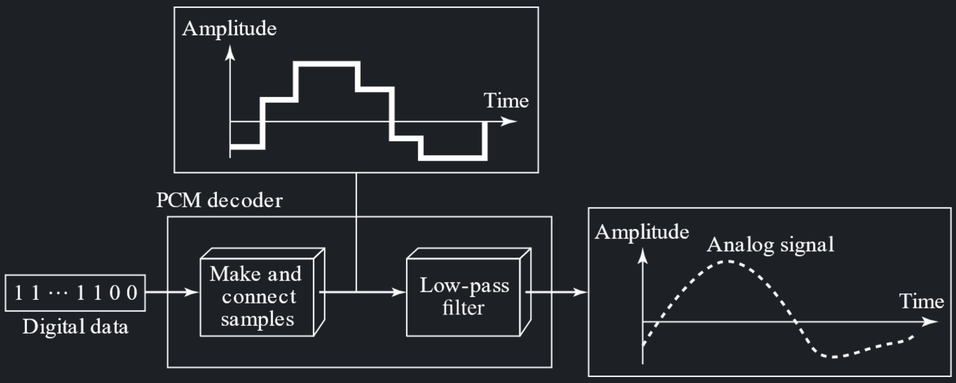
The human voice ranges from 0 to 4000Hz, so the sampling rate should be $4000 \times 2 = 8000\text{Hz}$.

$$\text{bit rate} = \text{sampling rate} \times \text{bits per sample} = \mathrm{f_s} \times \mathrm{n_b}$$

$$= 8000 \times 8 = 64000\text{bps} = 64\text{kbps}$$

Original Signal Recovery

The original signal is recovered on the receiver’s end using a PCM decoder. This makes use of a low-pass filter.



PCM Bandwidth

After digitization, the new bandwidth of the channel is:

$$B_{min} = c \times N \times \frac{1}{r} = c \times n_b \times f_s \times \frac{1}{r} = c \times n_b \times 2 \times B_{analogue} \times \frac{1}{r}$$

The minimum bandwidth, for $\frac{1}{r} = 1$ for NRZ or bipolar schemes, $c = \frac{1}{2}$, is given by

$$B_{min} = n_b \times B_{analogue}$$

The maximum data rate of a channel is given by

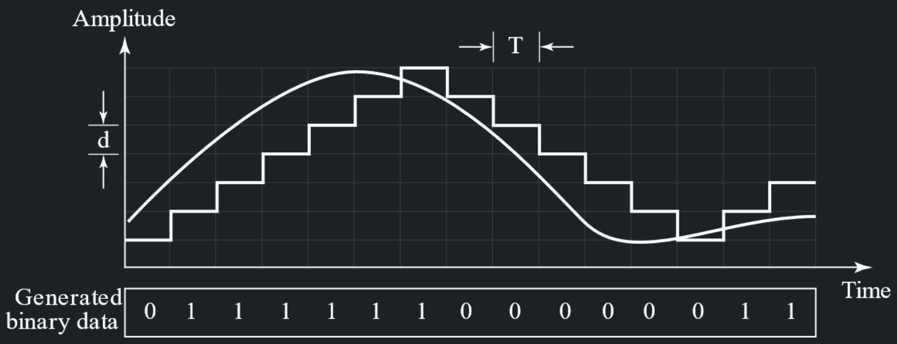
$N_{max} = 2 \times B \times \log_2 L$, in bps

The minimum bandwidth is given by

$B_{min} = \frac{N}{2 \times \log_2 L}$ in Hz

Delta Modulation

PCM is, well, complicated. This is why a simpler modulation technique, Delta Modulation (DM) was developed. Essentially, samples are taken and, if the value of the sample is greater than the value of the previous sample, a 1 bit is generated. If it is less, a 0 bit is generated.



The result is a graph that has a bunch of consecutives 1s and 0s. Sadly, the representation is not all that accurate so DM is only used when less granularity is required.

4.3 Transmission Modes

There are two transmission modes, parallel and serial.

In parallel transmission, there are multiple lines in parallel, meaning multiple bits are sent at each clock tick. Parallel transmission is used in the computer bus system.

In long distance communication, serial transmission is used. This means that using devices like computers, that use parallel transmission, with communication methods that use serial transmission, requires that a parallel-to-serial converter be used.

In serial transmission, data is sent bit by bit. There are three types of serial transmission, asynchronous, synchronous and isochronous.

In asynchronous communication, the clocks on the sender and receiver end do not need to be synchronized. This method is normally used. Some delimiter bits are required to indicate where a frame or a block of data starts and where it ends. Normally, 0 is used as a start bit and 1 is used as a stop bit.

Asynchronous communication refers to the byte level, meaning the bytes of data are not synchronized. However, on the bit level, synchronization still occurs. The bit length of both machines needs to be of the same length. Otherwise, the sender could send say 8 bits but the receiver would try to receive say 10 bits.

In synchronous communication, both clocks on the sender and receiver end need to be synchronized. The data is sent without delimiters and the receiver is given the job of figuring out how to group the bits.

Isochronous transmission is used for real-time communication. When more than one stream is involved, such as video and audio, both streams need to be synchronized.