RAMAIAH Institute of Technology

nology



**Certificate**

*This report is submitted for the evaluation of Practical Assignment component for the subject "DATA COMMUNICATION" with the subject code CS44 during the term January to May 2019.*

Submitted by

**1MS17CS081 POOJA KUMARI**

**1MS18CS401 ASHWINI T**

**1MS18CS417 SAHANA LM**

Signature of Faculty

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BENGALURU-560054**

www.msrit.edu

# January 2019 – May 2019

**Table of contents:**

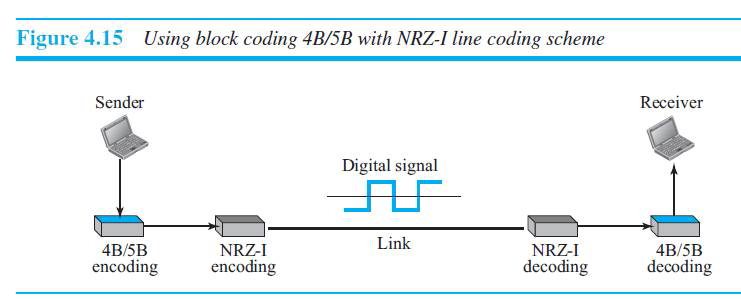
|  |  |  |
| --- | --- | --- |
| **SL NO** | **Topic** | **Page no** |
| 1 | Introduction | 03 |
| 2 | Implementation | 04 |
| 3 | Code | 06 |
| 4 | Results | 14 |
| 5 | References | 16 |

**4B/5B ENCODING**

**INTRODUCTION**

Write a C/C++/Java program to implement 4B/5B block coding. The program should read 4 bit data word from user and convert it into a 5 bit code word at the sender. After transmission of code word, the receiver should convert the 4 bit code word back to data word if the received code word is valid. Otherwise it should detect error and discard the packet.

4B/5B is a form of data communications line code. 4B/5B maps groups of 4 bits of data onto groups of 5 bits for transmission. These 5 bit words are pre-determined in a dictionary and they are chosen to ensure that there will be sufficient transitions in the line state to produce a self-clocking signal. A collateral effect of the code is that 25% more bits are needed to send the same information.



Depending on the standard or specification of interest, there may be several 5-bit output codes left unused. The presence of any of the unused codes in the data stream can be used as an indication that there is a fault somewhere in the link. Therefore, the unused codes can be used to detect errors in the data stream.

4B/5B was popularized by fibre distributed data interface (FDDI) in the mid-1980s, and was later adopted by Fast Ethernet standard defined by IEEE 802.3u in 1995 and AES10-2003 Multichannel Audio Digital Interface (MADI).[[1]](https://en.wikipedia.org/wiki/4B5B#cite_note-1)

The name *4B/5B* is generally taken to mean the FDDI version. Other 4-to-5-bit codes have been used for magnetic recording and are known as group coded recording(GCR), but those are (0,2) run-length limited codes, with at most two consecutive zeros. 4B5B allows up to three consecutive zeros (a (0,3) RLL code), providing a greater variety of control codes.

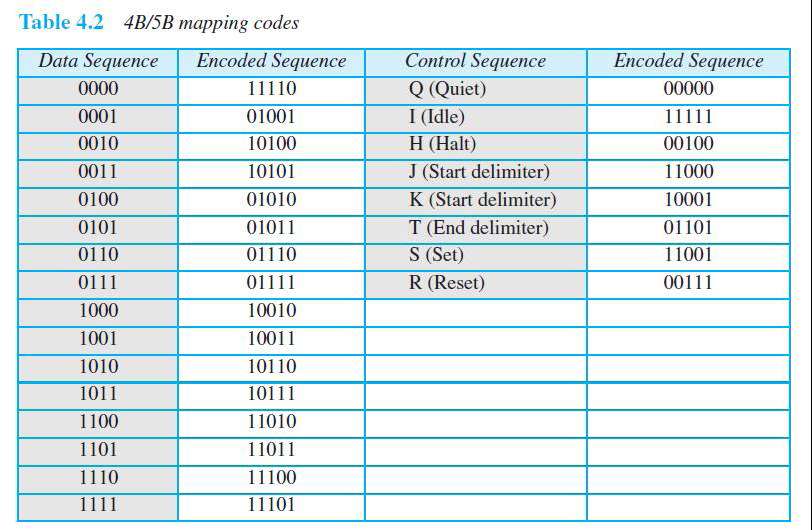
On optical fibre, the 4B/5B output is NRZI-encoded. FDDI over copper (CDDI) uses MLT-3 encoding instead, as does 100BASE-TX.

The 4B/5B encoding is also used for USB Power Delivery communication on CC pin, over BMC protocol.

**IMPLEMENTATION**

The sender sends a 4 bit data word which is in combination with NRZ-I and in turn converted to a 5 bit code word and transmits to the receiver end.

The following table encoding table is used for its implementation.



The first two columns pair a 4 –bit group with a 5-bit group. A group of bits can have only 16 different combinations while a group of 5-bits can have 32 different combinations. this means that there are 16 different groups that are not used for 4B/5B encoding.

Some of these unused groups are used for control purposes. The others are not used at all. The latter provides a kind of error detection.

At the receiver, the NRZ-I encoded digital signal is first decoded into a stream of bits and decoded to remove the redundancy. It maps the 5 bit code word to the corresponding data word. using the lookup table which is shown above.

Case 1:No error is detected at receiver end as it is a valid code word.

The data word sent: 0011

Corresponding codeword:10101

Code word received:10101

As it is a valid code word it is converted to dataword:0011 and the packet is extracted at receiver end.

Case 2:Error is detection at receiver end.

The data word sent: 1101

Corresponding codeword:11011

Code word received:11000

The 5B output i.e. the code word at the receiver end can have no more than 2 trailing zeros. So the is detected and packet is discarded.

**CODE**

#include<iostream>

#include<cstring>

#include<map>

#include<string>

#include<windows.h>

#include<iomanip>

#include<time.h>

using namespace std;

map <string,string> bit4to5;//to store lookup table

void line()//creates a line across the console

{

cout << setw(15);

for(int i=1;i<33;i++)

{

cout<<"--";

}

cout<<"\n";

}

void create\_table()//initialization of all values in the lookup table

{

bit4to5.insert(pair<string,string>("0000","11110"));

bit4to5.insert(pair<string,string>("0001","01001"));

bit4to5.insert(pair<string,string>("0010","10100"));

bit4to5.insert(pair<string,string>("0011","10101"));

bit4to5.insert(pair<string,string>("0100","01010"));

bit4to5.insert(pair<string,string>("0101","01011"));

bit4to5.insert(pair<string,string>("0110","01110"));

bit4to5.insert(pair<string,string>("0111","01111"));

bit4to5.insert(pair<string,string>("1000","10010"));

bit4to5.insert(pair<string,string>("1001","10011"));

bit4to5.insert(pair<string,string>("1010","10110"));

bit4to5.insert(pair<string,string>("1011","10111"));

bit4to5.insert(pair<string,string>("1100","11010"));

bit4to5.insert(pair<string,string>("1101","11011"));

bit4to5.insert(pair<string,string>("1110","11100"));

bit4to5.insert(pair<string,string>("1111","11101"));

}

class Sender//sender end

{

string dataword;//stores the actual data

string codeword;//stores the converted data

public:

Sender()

{

codeword = "-1";

}

void read()//read the 4 bit dataword from the user

{

cout << "Enter the dataword: ";

cin >> dataword;

cout << endl;

}

string get\_codeword()//to return the converted dataword

{

return codeword;

}

void convert()//to convert the dataword to codeword

{

map <string,string>::iterator itr;

itr = bit4to5.find(dataword);//searching for the match for the 4 bit dataword

if (itr!=bit4to5.end())//if codeword to corresponding dataword found

{

cout << "Converting dataword to codeword" << endl;

Sleep(3000);

codeword = itr->second;//conversion

cout << "Codeword: " << codeword << endl << endl;//printing codeword

}

else// implies dataword entered by user is invalid

{

cout << "Invalid Dataword" << endl;

exit(0);

}

}

void data\_change()//to introduce error into the data as it is transmitted through channel

{

if (codeword[0]=='0')//the codeword should not have more than two leading zeros(left)

{

codeword[1]='0';

}

else//the codeword should not have more than three trailing zeros(right)

{

codeword[2]='0';

codeword[3]='0';

codeword[4]='0';

}

}

};

class Receiver//receiver end

{

string codeword;//stores the codeword

string dataword;//stores dataword(useful information)

bool data\_check;//checks if the received codeword is valid

public:

Receiver()

{

dataword="-1";

}

void set\_code(Sender a)//receive codeword from sender as it is transmitted through the channel

{

codeword = a.get\_codeword();

}

void print\_codeword()//prints codeword

{

cout << "Codeword: " << codeword << endl;

}

void convert()//convert codeword back to dataword

{

cout << "Codeword received:" << codeword << endl;

map<string, string>::iterator itr;

for (itr=bit4to5.begin();itr!=bit4to5.end();itr++)//iterates through the map values to find the corresponding key

{

if (itr->second==codeword)

{

dataword=itr->first;//conversion to dataword

data\_check=true;//to show that codeword is valid and has been converted to dataword

break;

}

}

if (dataword=="-1")//implies that the corresponding dataword could not be found

{

data\_check=false;

}

}

void status\_check()

{

if (data\_check)//implies codeword is valid and has successfully been converted

{

cout << "Codeword valid" << endl;

cout << "Converting to dataword " << endl;

Sleep(3000);

cout << "Dataword: " << dataword << endl << endl;

}

else//implies codeword is invalid and has error

{

cout << "Codeword Invalid " << endl;

cout << "Discarding packet " << endl << endl;

}

}

};

void transmit(Sender& a, Receiver& b)//transmission across channel

{

int isecret;

srand(time(NULL));

isecret=rand()%10+1;//introducing random errors

if (isecret%2==0) {

a.data\_change();//introduces errors in the sent codeword

}

cout << "Sending codeword" << endl ;

cout << ".. " << endl;

Sleep(3000);

cout << ". " << endl;

Sleep(4000);

b.set\_code(a);//codeword sent to receiver's end

cout << "Transmission successful" << endl;

}

void display\_table()//to display the lookup table

{

cout << setw(52) << "LOOKUP TABLE" << endl;

line();

cout << setw(30) << "Dataword" << setw(35) << "Codeword" << endl;

line();

map<string, string>::iterator itr;

for (itr=bit4to5.begin();itr!=bit4to5.end();itr++)

{

cout << setw(30) << itr->first << setw(35) << itr->second << endl;//printing the dataword and corresponding codeword

}

line();

cout << endl << endl;

}

int main()

{

create\_table();//setting values in lookup table

Sender a;

Receiver b;

while(1)

{ int i;

cout << "Enter 1 to input dataword and convert\nEnter 2 to transmit and check at receiver end\nEnter 3 to display table\n";

cout << "Enter 0 to exit " << endl;

cin >> i;

switch(i){

case 1: cout << endl;

a.read();//dataword inputted by user

cout << endl;

a.convert();//converting dataword to codeword

break;

case 2: cout << endl;

transmit(a,b);//transmit codeword

b.convert();//check for errors at receiving end

b.status\_check();//print if codeword is valid or invalid

break;

case 3: cout << endl;

display\_table();//display lookup table

break;

case 0: exit(0);

}

}

return 0; }

**RESULTS**

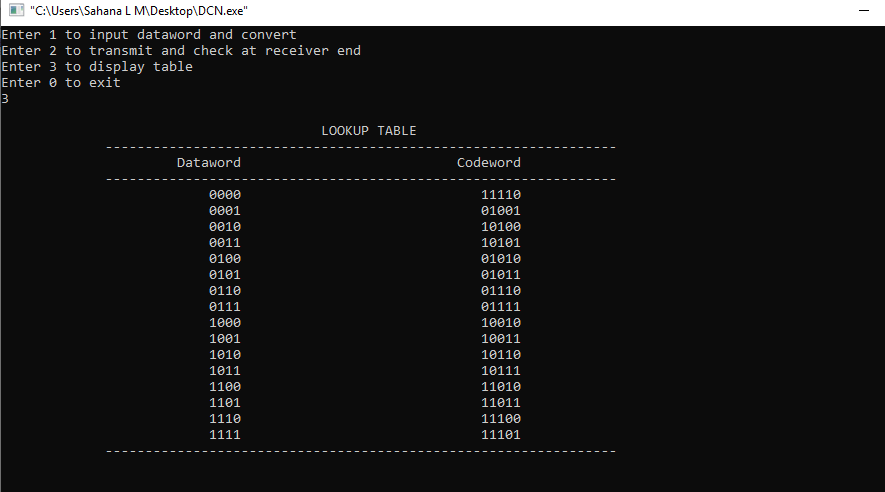


Fig: lookup table

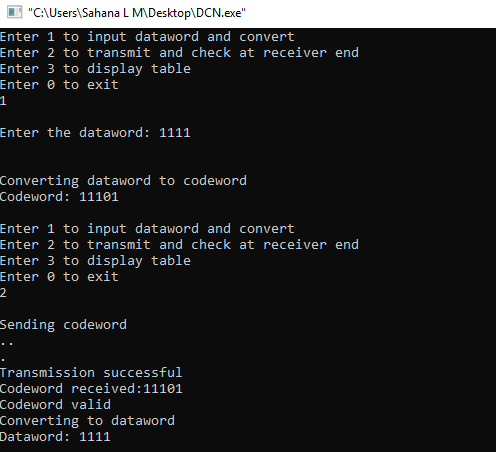


Fig: valid dataword received

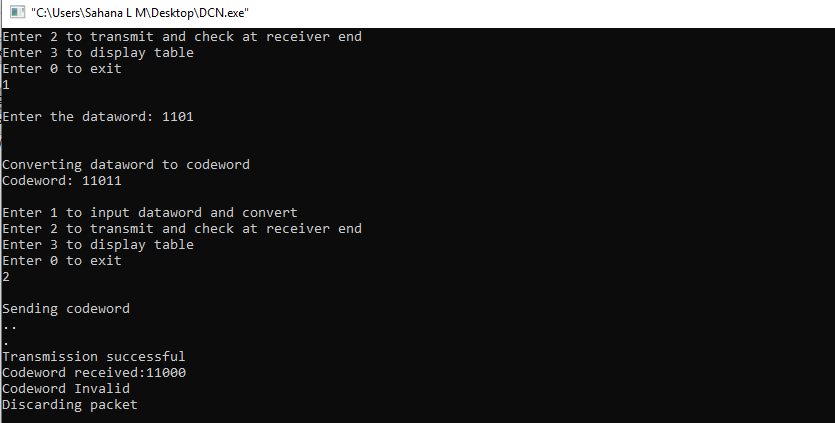


Fig: Invalid code word received

**REFERENCES:**

* Data communication and networking, Behrouz A .Forouzan, McGraw Hill, 5th edition, 2008
* <https://en.wikipedia.org/wiki/4B5B>
* <https://erg.abdn.ac.uk/users/Gorry/course/phy-pages/4b5b.html>
* <https://www.savvius.com/networking-glossary/fast_ethernet/signal_encoding/signal_4b5b/>
* <https://www.geeksforgeeks.org/digital-electronics-block-coding/>