ЦЕЛЬ РАБОТЫ

Углубление теоретических знаний в области помехоустойчивого кодирования данных в информационных системах, исследование способов построения кода, исправляющего одиночные ошибки и схемной реализации кодера и декодера кода Хемминга, а также приобретение практических навыков исследования процессов помехоустойчивого кодирования информационных сообщений на программно-аппаратном уровне.

ПРОГРАММА ЛАБОРАТОРНОЙ РАБОТЫ

1. Изучить по рекомендуемой литературе теоретический материал по теме динамического кодирования строк переменной длины равномерными кодами и разобрать примеры построения кодов LZW. Выполняется в процессе домашней подготовки.

2. Рассчитать по выражениям значения проверочных бит для всех возможных 16-ти значений информационных комбинаций.

3. Составить в рабочем поле эмулятора схему кодера.

4. Набрать с помощью ключей SW-1…SW-4 16 кодовых комбинаций и записать значения проверочных битов.

5. Сравнить полученные экспериментально проверочные комбинации с комбинациями, вычисленными теоретически.

6. Составить в рабочем поле эмулятора схему декодера Хемминга.

7. Подавать последовательно с помощью ключей SW-1…SW-7 все закодированные кодом Хемминга комбинации, полученные в п.4.4 и проследить отображаемую информацию на светодиодных индикаторах.

8. Изменяя поочередно один из битов входной информации, имитируя одиночную ошибку, проследить процесс коррекции ошибки декодером.

9. Сформулировать выводы по работе и оформить отчет.

ХОД РАБОТЫ

1. Создадим схему в среде Proteus рисунке 1.

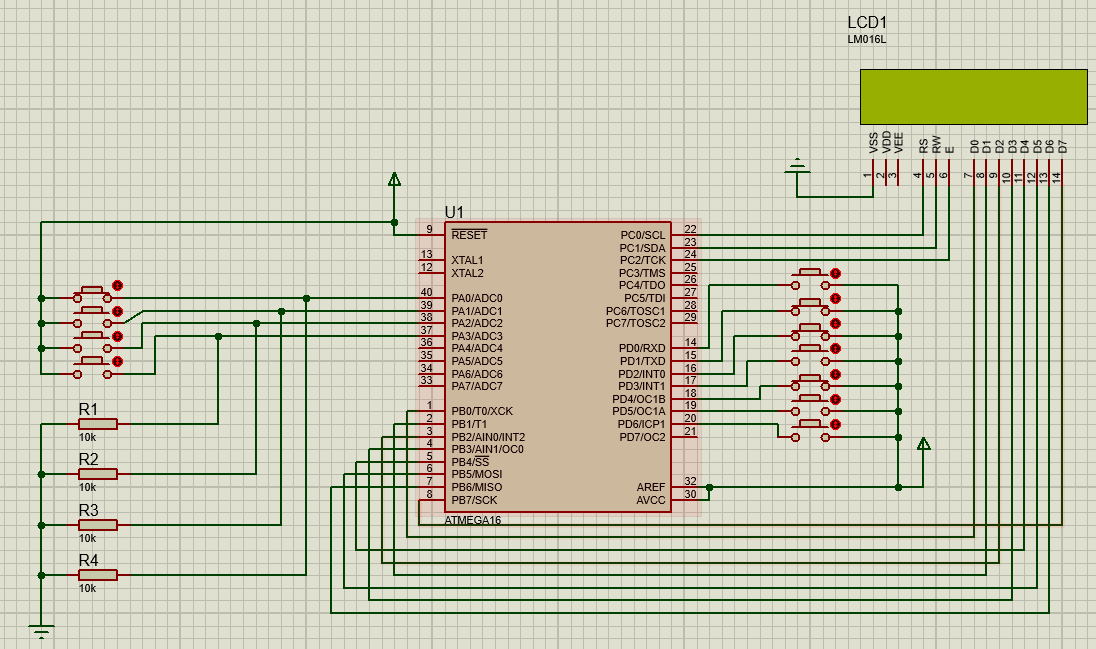


Рисунок 1 — Построение схемы в среде Proteus

2. Напишем код, который будет кодировать и декодировать введенный код.

#define F\_CPU 8000000UL /\* Define CPU Frequency e.g. here 8MHz \*/

#include <avr/io.h> /\* Include AVR std. library file \*/

#include <math.h>

#include <util/delay.h> /\* Include inbuilt defined Delay header file \*/

#define LCD\_Data\_Dir DDRB /\* Define LCD data port direction \*/

#define LCD\_Command\_Dir DDRC /\* Define LCD command port direction register \*/

#define LCD\_Data\_Port PORTB /\* Define LCD data port \*/

#define LCD\_Command\_Port PORTC /\* Define LCD data port \*/

#define RS PC0 /\* Define Register Select (data/command reg.)pin \*/

#define RW PC1 /\* Define Read/Write signal pin \*/

#define EN PC2 /\* Define Enable signal pin \*/

// Store input bits

int input[7];

// Store hamming code

int code[7];

int k, c, c1, c2, c3;

char pos[2];

void LCD\_Command(unsigned char cmnd)

{

LCD\_Data\_Port= cmnd;

LCD\_Command\_Port &= ~(1<<RS); /\* RS=0 command reg. \*/

LCD\_Command\_Port &= ~(1<<RW); /\* RW=0 Write operation \*/

LCD\_Command\_Port |= (1<<EN); /\* Enable pulse \*/

\_delay\_us(1);

LCD\_Command\_Port &= ~(1<<EN);

\_delay\_ms(3);

}

void LCD\_Char (unsigned char char\_data) /\* LCD data write function \*/

{

LCD\_Data\_Port= char\_data;

LCD\_Command\_Port |= (1<<RS); /\* RS=1 Data reg. \*/

LCD\_Command\_Port &= ~(1<<RW); /\* RW=0 write operation \*/

LCD\_Command\_Port |= (1<<EN); /\* Enable Pulse \*/

\_delay\_us(1);

LCD\_Command\_Port &= ~(1<<EN);

\_delay\_ms(1);

}

void LCD\_Init (void) /\* LCD Initialize function \*/

{

LCD\_Command\_Dir = 0xFF; /\* Make LCD command port direction as o/p \*/

LCD\_Data\_Dir = 0xFF; /\* Make LCD data port direction as o/p \*/

\_delay\_ms(20); /\* LCD Power ON delay always >15ms \*/

LCD\_Command (0x38); /\* Initialization of 16X2 LCD in 8bit mode \*/

LCD\_Command (0x0C); /\* Display ON Cursor OFF \*/

LCD\_Command (0x06); /\* Auto Increment cursor \*/

LCD\_Command (0x01); /\* Clear display \*/

LCD\_Command (0x80); /\* Cursor at home position \*/

}

void LCD\_String (char \*str) /\* Send string to LCD function \*/

{

int i;

for(i=0;str[i]!=0;i++) /\* Send each char of string till the NULL \*/

{

LCD\_Char (str[i]);

}

}

void LCD\_String\_xy (char row, char pos, char \*str)/\* Send string to LCD with xy position \*/

{

if (row == 0 && pos<16)

LCD\_Command((pos & 0x0F)|0x80); /\* Command of first row and required position<16 \*/

else if (row == 1 && pos<16)

LCD\_Command((pos & 0x0F)|0xC0); /\* Command of first row and required position<16 \*/

LCD\_String(str); /\* Call LCD string function \*/

}

void LCD\_Clear()

{

LCD\_Command (0x01); /\* clear display \*/

LCD\_Command (0x80); /\* cursor at home position \*/

}

int main()

{

DDRA = 0x00;

PORTA = 0x00;

DDRD = 0x00;

PORTD = 0x00;

LCD\_Init();

while(1) {

/\* ENCODE \*/

for(k = 0; k<4; k++ ){

if ((PINA & (1<<k)))

input[k] = 1;

else

input[k] = 0;

}

code[0] = input[0];

code[1] = input[1];

code[2] = input[2];

code[4] = input[3];

code[6]=code[0]^code[2]^code[4];

code[5]=code[0]^code[1]^code[4];

code[3]=code[0]^code[1]^code[2];

LCD\_String("Generated Code");

LCD\_Command(0xC0);

for (k=0;k<7;k++) {

if (code[k])

LCD\_String\_xy(2,k+1,"1");

else

LCD\_String\_xy(2,k+1,"0");

}

/\* DECODE \*/

/\*

for(k = 0; k<7; k++ ){

if ((PIND & (1<<k)))

input[k] = 1;

else

input[k] = 0;

}

c1=input[6]^input[4]^input[2]^input[0];

c2=input[5]^input[4]^input[1]^input[0];

c3=input[3]^input[2]^input[1]^input[0];

c=c3\*4+c2\*2+c1;

if(c == 0) {

LCD\_String("No errors: ");

LCD\_Command(0xC0);

for (k=0;k<7;k++) {

if (input[k])

LCD\_String\_xy(2,k+1,"1");

else

LCD\_String\_xy(2,k+1,"0");

}

} else {

LCD\_String("Erron on: ");

pos[0] = c + '0';

LCD\_String(pos);

LCD\_Command(0xC0);

LCD\_String("Correct:");

if(input[7-c]==0)

input[7-c]=1;

else

input[7-c]=0;

for (k=0;k<7;k++) {

if (input[k])

LCD\_String\_xy(2,k+10,"1");

else

LCD\_String\_xy(2,k+10,"0");

}

}

\*/

\_delay\_ms(750);

LCD\_Clear();

}

return 0;

}

3. Вычислим значения кода Хэмминга для всех 16 комбинаций.

|  |  |
| --- | --- |
| Кодируемое сообщение | Выходное сообщение |
| 0000 | 0000000 |
| 0001 | 1001011 |
| 0010 | 0101010 |
| 0011 | 1100001 |
| 0100 | 0011001 |
| 0101 | 1010010 |
| 0110 | 0110011 |
| 0111 | 0110100 |
| 1000 | 0000111 |
| 1001 | 1001100 |
| 1010 | 0101101 |
| 1011 | 1100110 |
| 1100 | 0011110 |
| 1101 | 1010101 |
| 1110 | 0110100 |
| 1111 | 1111111 |

4. Изменим один бит в передаваемом сообщении и посмотрим, как система реагирует на ошибку.

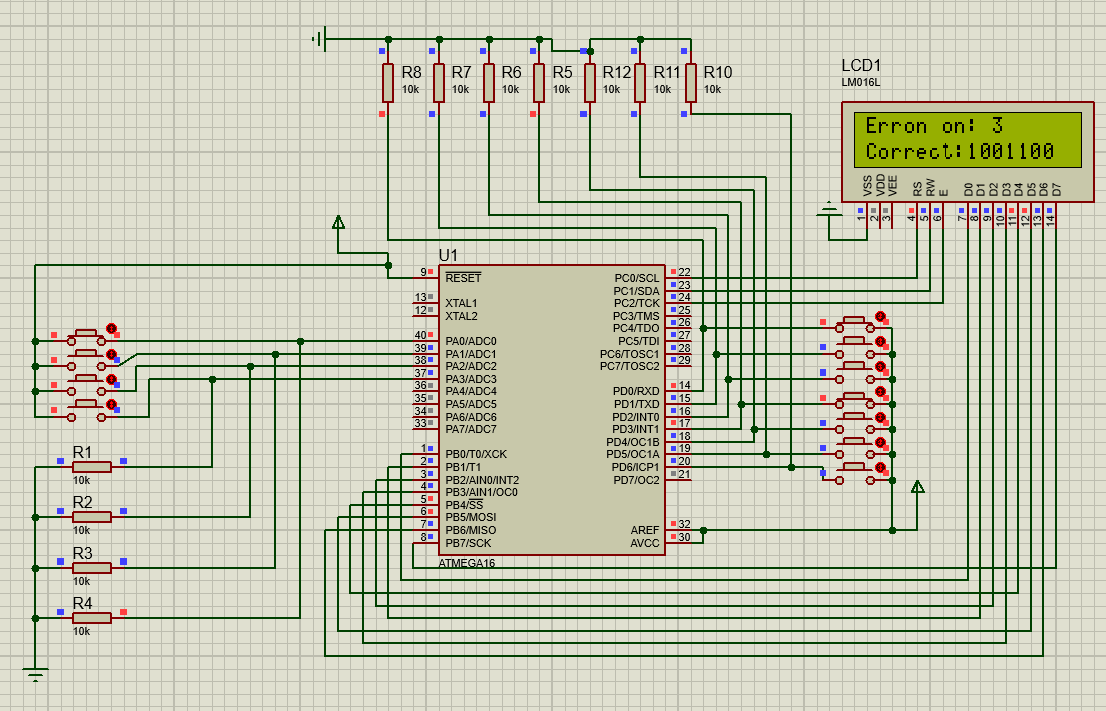


Рисунок 2 — Передача сообщения с ошибкой

ВЫВОДЫ

В ходе лабораторной работы были исследованы методы помехоустойчивого кодирования с помощью кода Хемминга, была построена схема, которая может кодировать и раскодировать сообщения.