Министерство науки и высшего образования Российской Федерации

Новосибирский Государственный технический университет

Кафедра автоматизированных систем управления



ОТЧЁТ

по лабораторной работе №3

по дисциплине «Защита информации»

«Вычисление циклического контрольного кода»

Выполнили: Проверил:

Студенты гр. «АВТ-812», «АВТФ» к.т.н., доцент

Березин Дмитрий Качальский В.Г.

Бородина Алина

Глинин Евгений

«\_\_\_» \_\_\_\_\_\_ 20\_\_г. «\_\_\_» \_\_\_\_\_\_ 20\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

г. Новосибирск

2021 г.

Содержание

[1 Цель работы 3](#_Toc85048237)

[2 Постановка задачи 3](#_Toc85048238)

[4 Преобразования исходного текста в криптограмму и обратно 5](#_Toc85048239)

[5 Примеры шифрования-дешифрования текста 6](#_Toc85048240)

[6 Выводы 8](#_Toc85048241)

[7 Листинг 9](#_Toc85048242)

# 1 Цель работы

Изучить методы вычисления циклического контрольного кода.

# 2 Постановка задачи

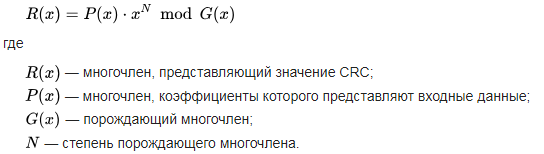
* На основе Примера 1. Привести 2-3 своих примера получения остатка в двоичной и полиномиальной арифметике.
* Выполнить компьютерную реализацию прямого алгоритма вычисления CRC на основе Примера 2.
* Изучить самостоятельно табличные алгоритмы вычисления CRC и алгоритмы вычисления CRC для любой длины информационного кода /обратный алгоритм/.
* Реализовать алгоритмы CRC8 и CRC16 для табличного метода и привести примеры получения контрольной суммы произвольного файла, приписать CRC к концу файла и поверить его программно на контрольную сумму.

**3 Теоретическая часть работы**

Алгоритм CRC базируется на свойствах деления с остатком двоичных многочленов, то есть многочленов над конечным полем GF(2). Значение CRC является по сути остатком от деления многочлена, соответствующего входным данным, на некий фиксированный порождающий многочлен. Каждой конечной последовательности битов a0 a1 … an-1 взаимно однозначно сопоставляется двоичный полином, последовательность коэффициентов которого представляет собой исходную последовательность. Например, последовательность битов 1011010 соответствует многочлену:



Количество различных многочленов степени, меньшей N, равно 2N, что совпадает с числом всех двоичных последовательностей длины N. Значение контрольной суммы в алгоритме с порождающим многочленом G(x) степени N определяется как битовая последовательность длины N, представляющая многочлен R(x), получившийся в остатке при делении многочлена P(x), представляющего входной поток бит, на многочлен G(x):



**Примеры:**

**Пример №1**

Двоичная арифметика:

A(x) = 10110111

G(x) = 1101

A(x) \* xr = 10110111000

Полиномиальная арифметика:

A(x) \* xr = x10 + x8 + x7 + x5 + x4 + x3

G(x) = x3 + x2 + x0

После сложения по модулю два, получаем остаток R(x) = 111 или R(x) = x2 + x1 + x0

**Пример №2**

Двоичная арифметика:

A(x) = 111100011

G(x) = 101

A(x) \* xr = 11110001100

Полиномиальная арифметика:

A(x) \* xr = x10 + x9 + x8 + x7 + x3 + x2

G(x) = x2 + x0

После сложения по модулю два, получаем остаток R(x) = 11 или R(x) = x1 + x0.

# 4 Вычисление CRC кода

Описанный ниже алгоритм используется для вычисления двух разных CRC-кодов. Последовательность вычислений, следующая:

1. Нахождение логического исключающего ИЛИ между младшим битом начального значения CRC (00 или FF) и младшим битом данных.

2. Если результат равен 0, то: a. Сдвиг вправо CRC. 3. Если результат равен 1:

a. Поиск нового значения CRC путем вычисления логического исключающего ИЛИ между CRC(полученным остатком) и полиномом CRC.

b. Сдвиг вправо CRC

c. Установка старшего бита CRC в 1.

4. Сдвиг вправо данных.

5. Повтор данной последовательности 8 раз для CRC8.

# 5 Программная реализация с примерами

Для проверки данных пользователю необходимо ввести необходимый текст в поле «Текст для прямого метода CRC», затем пользователю необходимо нажать кнопку «Прямой метод CRC8» или «Прямой метод CRC16» (Рисунок 1):

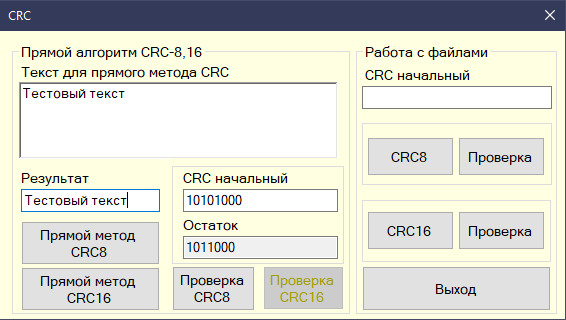


Рисунок 1 – ввод данных

После нажатия кнопки «Прямой метод CRC8» или «Прямой метод CRC16» в поле «CRC начальный» появится полином с помощью которого будет найден остаток деления и выведен в поле «Остаток» (Рисунок 1). Если изменить данные в поле «Результат» и нажать кнопку «Проверка CRC8» то появится ошибка, так как данные были изменены (Рисунок 2). Вернём строку и исходное состояние и проверим (Рисунок 3).

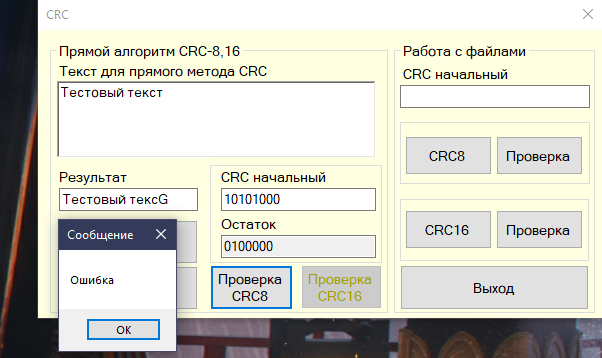


Рисунок 2 – результат проверки с ошибкой

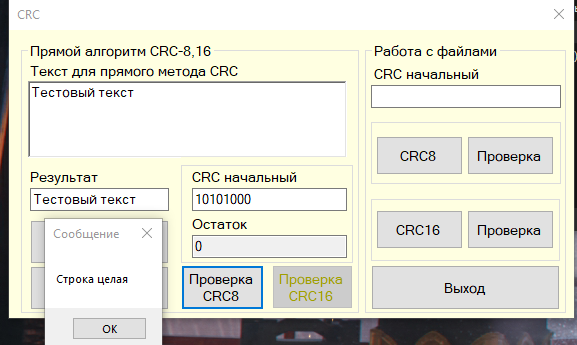


Рисунок 3 – результат проверки

Работа с файлами. Для проверки файла пользователю необходимо выбрать CRC и затем выбрать файл для добавления остатка (Рисунок 4-8):

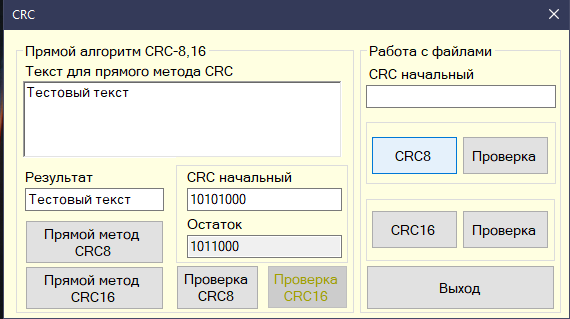


Рисунок 4 – Выбор CRC8 для применения к файлу

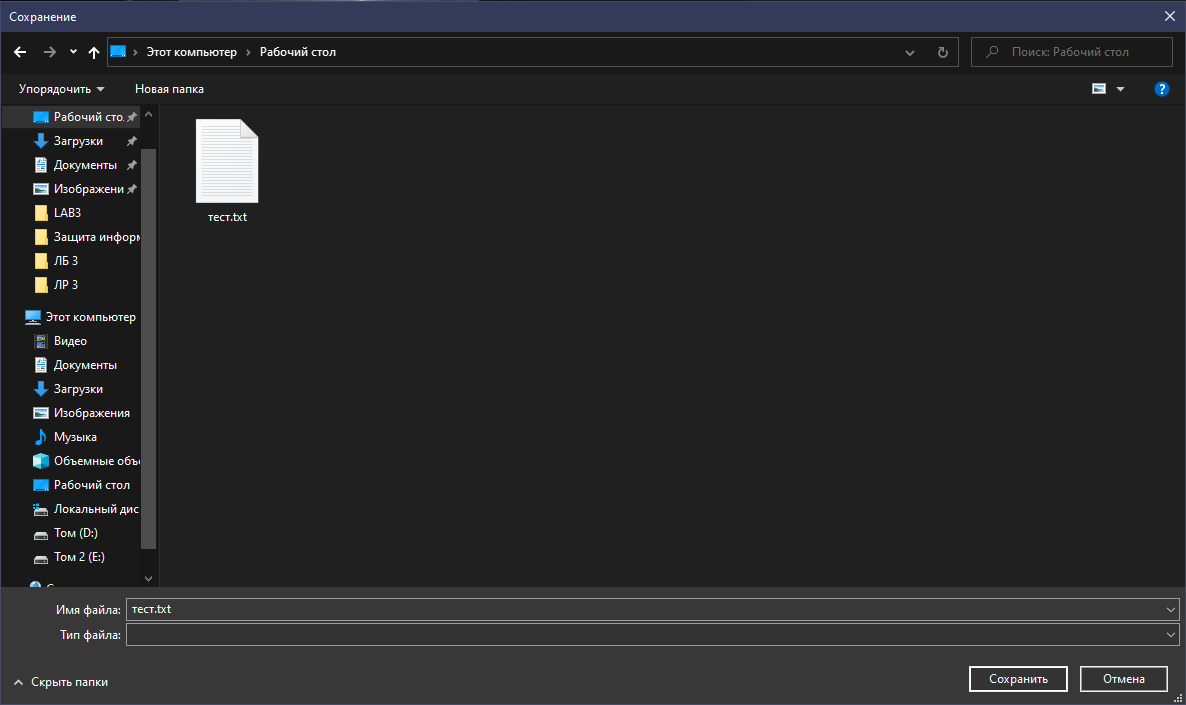


Рисунок 5 – Выбор файла

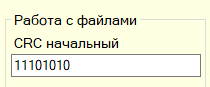


Рисунок 6 – Начальный CRC для файла с определёнными данными

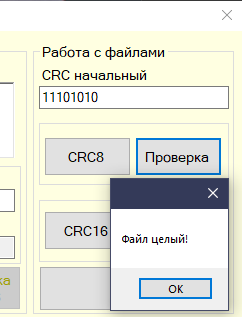


Рисунок 7 – Результат проверки файла с целыми данными

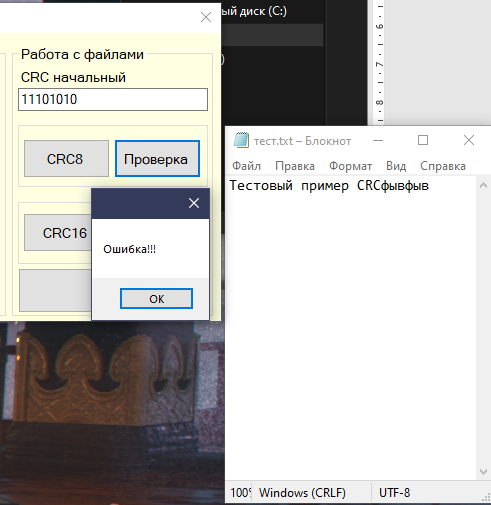


Рисунок 8 – Результат проверки файла с изменёнными данными

# 6 Выводы

В ходе лабораторной работы был рассмотрен алгоритм проверки целостности информации набранного с клавиатуры произвольного текста и бинарных файлов, а также выполнена программная реализация прямого и табличного алгоритма CRC.

# 7 Листинг

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button7\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

if (richTextBox1.Text.Trim() != "")

{

string data = richTextBox1.Text;

string dataBIN = CRC.getByte(data);

byte crc = CRC.CRC8code(data);

string crc\_bin = CRC.byteToStr(crc);

string remains = CRC.CRC\_get(dataBIN, crc\_bin);

textBox3.Text = crc\_bin;

textBox5.Text = remains;

textBox4.Text = data;

textBox2.Text = "";

textBox2.Text += Convert.ToChar(CRC.StrToByte8(remains));

button2.Enabled = true;

button9.Enabled = false;

}

else

{

MessageBox.Show("Введите текст!");

button2.Enabled = false;

button9.Enabled = true;

textBox3.Text = "";

textBox5.Text = "";

}

}

private void button8\_Click(object sender, EventArgs e)

{

if (richTextBox1.Text.Trim() != "")

{

string data = richTextBox1.Text;

string dataBIN = CRC.getByte(data);

UInt16 crc = CRC.CRC16\_table(data);

string crc\_bin = CRC.byteToStr(crc);

string remains = CRC.CRC\_get(dataBIN, crc\_bin);

textBox3.Text = crc\_bin;

textBox5.Text = remains;

textBox4.Text = data;

textBox2.Text = CRC.StrTo2Char(remains);

button9.Enabled = true;

button2.Enabled = false;

}

else

{

MessageBox.Show("Введите текст!");

button9.Enabled = false;

button2.Enabled = true;

textBox3.Text = "";

textBox5.Text = "";

}

}

private void button3\_Click(object sender, EventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

if (ofd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string data = File.ReadAllText(ofd.FileName);

string dataBIN = CRC.getByte(data);

byte crc = CRC.CRC8code(data);

string crc\_bin = CRC.byteToStr(crc);

string remains = CRC.CRC\_get(dataBIN, crc\_bin);

textBox1.Text = crc\_bin;

textBox2.Text = "";

textBox2.Text += Convert.ToChar(CRC.StrToByte8(remains));

SaveFileDialog sfdDialog = new SaveFileDialog();

if (sfdDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

File.WriteAllText(sfdDialog.FileName, data);

}

}

}

private void button5\_Click(object sender, EventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

if (ofd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string data = File.ReadAllText(ofd.FileName);

string dataBIN = CRC.getByte(data);

UInt16 crc = CRC.CRC16\_table(data);

string crc\_bin = CRC.byteToStr(crc);

string remains = CRC.CRC\_get(dataBIN, crc\_bin);

textBox1.Text = crc\_bin;

textBox2.Text = CRC.StrTo2Char(remains);

SaveFileDialog sfdDialog = new SaveFileDialog();

if (sfdDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

File.WriteAllText(sfdDialog.FileName, data);

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

if (ofd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string data = File.ReadAllText(ofd.FileName) + textBox2.Text;

string crc\_check = textBox1.Text;

string start = CRC.getByteInCheck8(data, crc\_check.Length);

string remains = CRC.CRC\_remainder(start, crc\_check);

if (CRC.StrToByte8(remains) == 0)

MessageBox.Show("Файл целый!");

else

MessageBox.Show("Ошибка!!!");

}

}

private void button6\_Click(object sender, EventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

if (ofd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string data = File.ReadAllText(ofd.FileName) + textBox2.Text;

string crc\_check = textBox1.Text;

string start = CRC.getByteInCheck16(data, crc\_check.Length);

string remains = CRC.CRC\_remainder(start, crc\_check);

if (CRC.StrToByte16(remains) == 0)

MessageBox.Show("Файл целый!");

else

MessageBox.Show("Ошибка!!!");

}

}

private void button2\_Click(object sender, EventArgs e)

{

string data = textBox4.Text + textBox2.Text;

string crc = textBox3.Text;

string start = CRC.getByteInCheck8(data, crc.Length);

string remains = CRC.CRC\_remainder(start, crc);

if (CRC.StrToByte8(remains) == 0)

{

textBox5.Text = "0";

MessageBox.Show("Строка целая", "Сообщение");

}

else

{

textBox5.Text = remains;

MessageBox.Show("Ошибка", "Сообщение");

}

}

private void button9\_Click(object sender, EventArgs e)

{

string data = textBox4.Text + textBox2.Text;

string crc = textBox3.Text;

string start = CRC.getByteInCheck16(data, crc.Length);

string remains = CRC.CRC\_remainder(start, crc);

if (CRC.StrToByte16(remains) == 0)

{

textBox5.Text = "0";

MessageBox.Show("Строка целая", "Сообщение");

}

else

{

textBox5.Text = remains;

MessageBox.Show("Ошибка", "Сообщение");

}

}

}

}

CRC.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WindowsFormsApp1

{

public class CRC

{

public static byte[] crc8x\_table = {

0x00, 0x31, 0x62, 0x53, 0xc4, 0xf5, 0xa6, 0x97, 0xb9, 0x88, 0xdb, 0xea, 0x7d,

0x4c, 0x1f, 0x2e, 0x43, 0x72, 0x21, 0x10, 0x87, 0xb6, 0xe5, 0xd4, 0xfa, 0xcb,

0x98, 0xa9, 0x3e, 0x0f, 0x5c, 0x6d, 0x86, 0xb7, 0xe4, 0xd5, 0x42, 0x73, 0x20,

0x11, 0x3f, 0x0e, 0x5d, 0x6c, 0xfb, 0xca, 0x99, 0xa8, 0xc5, 0xf4, 0xa7, 0x96,

0x01, 0x30, 0x63, 0x52, 0x7c, 0x4d, 0x1e, 0x2f, 0xb8, 0x89, 0xda, 0xeb, 0x3d,

0x0c, 0x5f, 0x6e, 0xf9, 0xc8, 0x9b, 0xaa, 0x84, 0xb5, 0xe6, 0xd7, 0x40, 0x71,

0x22, 0x13, 0x7e, 0x4f, 0x1c, 0x2d, 0xba, 0x8b, 0xd8, 0xe9, 0xc7, 0xf6, 0xa5,

0x94, 0x03, 0x32, 0x61, 0x50, 0xbb, 0x8a, 0xd9, 0xe8, 0x7f, 0x4e, 0x1d, 0x2c,

0x02, 0x33, 0x60, 0x51, 0xc6, 0xf7, 0xa4, 0x95, 0xf8, 0xc9, 0x9a, 0xab, 0x3c,

0x0d, 0x5e, 0x6f, 0x41, 0x70, 0x23, 0x12, 0x85, 0xb4, 0xe7, 0xd6, 0x7a, 0x4b,

0x18, 0x29, 0xbe, 0x8f, 0xdc, 0xed, 0xc3, 0xf2, 0xa1, 0x90, 0x07, 0x36, 0x65,

0x54, 0x39, 0x08, 0x5b, 0x6a, 0xfd, 0xcc, 0x9f, 0xae, 0x80, 0xb1, 0xe2, 0xd3,

0x44, 0x75, 0x26, 0x17, 0xfc, 0xcd, 0x9e, 0xaf, 0x38, 0x09, 0x5a, 0x6b, 0x45,

0x74, 0x27, 0x16, 0x81, 0xb0, 0xe3, 0xd2, 0xbf, 0x8e, 0xdd, 0xec, 0x7b, 0x4a,

0x19, 0x28, 0x06, 0x37, 0x64, 0x55, 0xc2, 0xf3, 0xa0, 0x91, 0x47, 0x76, 0x25,

0x14, 0x83, 0xb2, 0xe1, 0xd0, 0xfe, 0xcf, 0x9c, 0xad, 0x3a, 0x0b, 0x58, 0x69,

0x04, 0x35, 0x66, 0x57, 0xc0, 0xf1, 0xa2, 0x93, 0xbd, 0x8c, 0xdf, 0xee, 0x79,

0x48, 0x1b, 0x2a, 0xc1, 0xf0, 0xa3, 0x92, 0x05, 0x34, 0x67, 0x56, 0x78, 0x49,

0x1a, 0x2b, 0xbc, 0x8d, 0xde, 0xef, 0x82, 0xb3, 0xe0, 0xd1, 0x46, 0x77, 0x24,

0x15, 0x3b, 0x0a, 0x59, 0x68, 0xff, 0xce, 0x9d, 0xac };

public static UInt16[] crc16x\_table = {

0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,

0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,

0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,

0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,

0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,

0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,

0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,

0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,

0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,

0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,

0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,

0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,

0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,

0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,

0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,

0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,

0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,

0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,

0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,

0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,

0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,

0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,

0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,

0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,

0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,

0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,

0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,

0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,

0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,

0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,

0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,

0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0 };

static public byte CRC8code(string data)

{

byte crc = 0xff;

byte[] dataNew = Encoding.GetEncoding("windows-1251").GetBytes(data);

for (int i = 0; i < data.Length; i++)

{

crc ^= dataNew[i];

for (int j = 0; j < 8; j++)

{

if ((crc & 0x80) != 0)

crc = (byte)((crc << 1) ^ 0x31); //CRC8 0x31

else

crc <<= 1;

}

}

return crc;

}

static public byte CRC8\_table(string data)

{

byte crc = 0xff;

byte[] dataNew = Encoding.GetEncoding("windows-1251").GetBytes(data);

for (int i = 0; i < dataNew.Length; ++i)

crc = Convert.ToByte(crc8x\_table[crc ^ dataNew[i]]);

return crc;

}

static public UInt16 CRC16\_table(string data)

{

UInt16 crc = 0xff;

byte[] dataNew = Encoding.GetEncoding("windows-1251").GetBytes(data);

for (int i = 0; i < dataNew.Length; ++i)

{

crc = crc16x\_table[(crc >> 8) ^ dataNew[i]];

}

return crc;

}

static public string CRC\_get(string data, string code)

{

for (int i = 0; i < code.Length - 1; ++i)

data += "0";

return CRC\_remainder(data, code);

}

static public string CRC\_remainder(string data, string code)

{

string codeNew = code;

StringBuilder dataNew = new StringBuilder(data);

for (int i = dataNew.Length; i >= codeNew.Length; --i)

{

if (dataNew[dataNew.Length - i] == '0')

continue;

for (int j = 0; j < codeNew.Length; ++j)

dataNew[dataNew.Length - i + j] = stringXOR(dataNew[dataNew.Length - i + j], code[j]);

}

return Convert.ToString(dataNew).Substring(dataNew.Length - codeNew.Length + 1, codeNew.Length - 1);

}

static public char stringXOR(char a, char b)

{

return (a == b) ? '0' : '1';

}

static public string byteToStr(byte data)

{

string res = "";

while (true)

{

if (data == 0)

return res;

if ((data & 0x1) == 1)

{

res = "1" + res;

}

else

{

res = "0" + res;

}

data >>= 1;

}

}

static public string byteToStr(UInt16 data)

{

string res = "";

while (true)

{

if (data == 0)

return res;

if ((data & 0x1) == 1)

{

res = "1" + res;

}

else

{

res = "0" + res;

}

data >>= 1;

}

}

static public string getByte(string str)

{

string res = "";

for (int i = 0; i < str.Length; ++i)

{

string temp = "";

temp += byteToStr((byte)str[i]);

for (int j = temp.Length; j < 8; ++j)

{

temp = "0" + temp;

}

res += temp;

}

return res;

}

static public byte StrToByte8(string str)

{

byte res = 0;

for (int i = 0; i < str.Length; ++i)

{

res \*= 2;

if (str[i] == '1')

res += 1;

}

return res;

}

static public UInt16 StrToByte16(string str)

{

UInt16 res = 0;

for (int i = 0; i < str.Length; ++i)

{

res \*= 2;

if (str[i] == '1')

res += 1;

}

return res;

}

static public string StrTo2Char(string str)

{

byte res1 = 0;

byte res2 = 0;

string res = "";

for (int i = str.Length; i < 16; ++i)

str = "0" + str;

for (int i = 0; i < str.Length / 2; ++i)

{

res1 \*= 2;

res2 \*= 2;

if (str[i] == '1')

res1 += 1;

if (str[i + 8] == '1')

res2 += 1;

}

res += Convert.ToChar(res1);

res += Convert.ToChar(res2);

return res;

}

static public string getByteInCheck8(string str, int sizeCRC)

{

int i;

string res = "", crc;

for (i = 0; i < (str.Length - 1); ++i)

{

string temp = "";

temp += byteToStr((byte)str[i]);

for (int j = temp.Length; j < 8; ++j)

{

temp = "0" + temp;

}

res += temp;

}

crc = byteToStr((byte)str[i]);

for (int k = crc.Length; k < sizeCRC - 1; ++k)

{

crc = "0" + crc;

}

return res + crc;

}

static public string getByteInCheck16(string str, int sizeCRC)

{

int i;

string res = "", crc1, crc2;

for (i = 0; i < (str.Length - 2); ++i)

{

string temp = "";

temp += byteToStr((byte)str[i]);

for (int j = temp.Length; j < 8; ++j)

{

temp = "0" + temp;

}

res += temp;

}

crc1 = byteToStr((byte)str[i]);

crc2 = byteToStr((byte)str[i + 1]);

for (int k = crc1.Length; k < sizeCRC - 9; ++k)

crc1 = "0" + crc1;

for (int l = crc2.Length; l < 8; ++l)

crc2 = "0" + crc2;

return res + crc1 + crc2;

}

}

}