DOKUMENTACJA PROJEKTU

Kierunek: INFORMATYKA STOSOWANA

Przedmiot: PROGRAMOWANIE W ŚRODOWISKU WINDOWS

Forma zajęć: Projekt

Rok akademicki: 2018/2019

Temat: Interfejs komunikacji host-modem ST7580 wraz z GUI

Student: Krzysztof Mantych IS/ST Grupa 3, Semestr V.

WSTĘP

Założenia projektu:

Celem projektu jest zaimplementowanie komunikacji host-modem ST7580 wraz z GUI. Aplikacja umożliwia resetowanie modemu oraz wysłanie i odebranie ramki zdefiniowanej przez użytkownika.

Narzędzia:

* Język programowania: C#
* Microsoft Visual Studio Enterprise
* Modem ST7580

SPIS TREŚCI

* Opis realizacji – strona 3
* Opis elementów projektu – strona 4 – 8
* Działanie – strona 9 – 10
* Podsumowanie – strona 10

Opis realizacji projektu

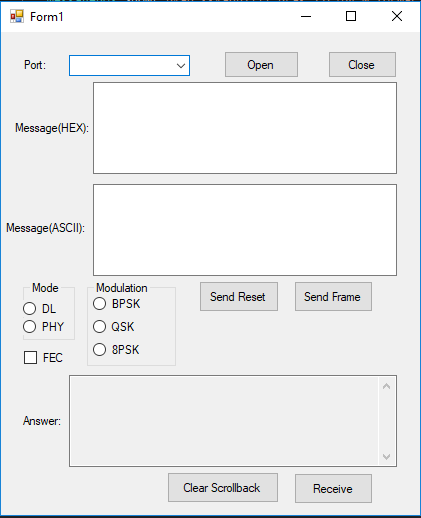
1. Wymagana dokumentacja ST7580-FSK, PSK - chapter 3.
2. GUI zawiera elementy kontrolne pozwalające na realizację następujących funkcji:

* reset modemu (przycisk Send Reset)
* elementy do wyboru numeru portu szeregowego
* otwieranie i zamykanie połączenia
* elementy do edycji komunikatu w formacie ASCII i HEX
* element do wyświetlania odbieranych komunikatów

1. Monitor komunikacji szeregowej z wykorzystaniem elementu GUI.
2. Auto detekcja fizycznych (dostępnych) portów szeregowych.
3. Odbiór ramki na stanach.

Opis poszczególnych elementów projektu

Okienkowy interfejs użytkownika (GUI):



Elementy:

* Wybór portu
* Przyciski otwarcia / zamknięcia portu

|  |
| --- |
| private void btnOpen\_Click(object sender, EventArgs e)  {  string[] ports = SerialPort.GetPortNames();  cboPorts.Items.AddRange(ports);  cboPorts.SelectedIndex = 0;  btnClose.Enabled = false;  btnOpen.Enabled = false;  btnClose.Enabled = true;  try  {  serialPort1.PortName = cboPorts.Text;  serialPort1.BaudRate = 57600;  serialPort1.Parity = Parity.None;  serialPort1.DataBits = 8;  serialPort1.StopBits = StopBits.One;  serialPort1.Open();  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  private void btnClose\_Click(object sender, EventArgs e)  {  btnOpen.Enabled = true;  btnClose.Enabled = false;  try  {  serialPort1.Close();  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  } |

Objaśnienie:

Przy otwarciu portu wybierany jest pierwszy wolny port, a w trakcie działania mamy możliwość dynamicznego przepięcia portu.

* Pola tekstowe do wprowadzania ramki w formacie heksadecymalnym i translacji ramki na kod ASCII
* Przyciski wysyłania resetu i ramki zdefiniowanej przez użytkownika

|  |
| --- |
| private void sendReset\_Click(object sender, EventArgs e)  {  string frame0 = "02";  string frame1 = "00";  string frame2 = "3C";  string frame3 = "3C";  string frame4 = "00";  string resetFrame = "";  frame0 = ConvertHexToASCII(frame0);  frame1 = ConvertHexToASCII(frame1);  frame2 = ConvertHexToASCII(frame2);  frame3 = ConvertHexToASCII(frame3);  frame4 = ConvertHexToASCII(frame4);  resetFrame = frame0 + frame1 + frame2 + frame3 + frame4;  try  {  if (serialPort1.IsOpen)  {  serialPort1.RtsEnable = true;  serialPort1.WriteLine(resetFrame);  serialPort1.BaseStream.Flush();  serialPort1.RtsEnable = false;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  private void btnSend\_Click(object sender, EventArgs e)  {  string message = txtMessage.Text;  if (isValid(message) == false)  {  MessageBox.Show("Błąd składni!!! Brak liczby w formacie heksadecymalnym!!! (Przykład: FF, A0)");  return;  }  message = message.Replace(" ", String.Empty);  if (message.Length % 2 != 0)  {  MessageBox.Show("Podana ramka jest błędna! Sprawdz wysyłane wartości!");  return;  }  int length;  length = message.Length / 2;  string[] frame = new string[length];  for (int i = 0; i < length; i++)  {  frame[i] = message.Substring(i \* 2, 2);  }  string finalFrame = "";  for (int i = 0; i < length; i++)  {  finalFrame += ConvertHexToASCII(frame[i]);  }  if (serialPort1.IsOpen)  {  serialPort1.RtsEnable = true;  serialPort1.Write(finalFrame);  serialPort1.RtsEnable = false;  }  txtMessageASCII.Text += finalFrame;  } |

Objaśnienie:

Wysłanie resetu odbywa się poprzez wysłanie spreparowanych łańcuchów, które są najpierw konwertowane z zapisu heksadecymalnego na kod ASCII, a następnie są scalane w gotową ramkę i wysyłane na wybrany port. Przed wysłaniem linia RTS jest podnoszona, a po wysłaniu jest dezaktywowana.

W przypadku wysyłania ramki zdefiniowanej przez użytkownika za pomocą wyrażenia regularnego wybierane są wartości heksadecymalne, których ilość jest obliczana. Następnie tworzona jest tablica, która przetrzymuje wartości pobrane z interfejsu, które następnie są konwertowane na wartości kodów ASCII, scalane i wysyłane jak w przypadku wyżej. Analogicznie linia RTS jest podnoszona przed wysłaniem i dezaktywowana po wysłaniu.

* Tryby i modulacje (nieoprogramowane)
* Pole tekstowe służące do wydruku odpowiedzi modemu.
* Przycisk wczytania odebranej ramki

|  |
| --- |
| Private void btnReceive\_Click(object sender, EventArgs e)  {  int bytes = serialPort1.BytesToRead;  byte[] respBuffer = new byte[bytes];  serialPort1.Read(respBuffer, 0, bytes);  States[] checklist = new States[6];  checklist[0] = States.BEGINING;  checklist[1] = States.LENGTH;  checklist[2] = States.CONTROL;  checklist[3] = States.GET\_FRAME;  checklist[4] = States.CHECK\_BAIT;  checklist[5] = States.CHECK\_CRC;  byte[] correctFrame = new byte[6];  try  {  if (serialPort1.IsOpen)  {  if (checklist[0] == States.BEGINING)  {  if (respBuffer[0].ToString() == "2")  {  correctFrame[0] = respBuffer[0];  }  else  {  MessageBox.Show("Nie odnaleziono początku ramki!!!");  return;  }  }  if (checklist[1] == States.LENGTH)  {  if (ASCIITOHex(respBuffer[1].ToString()) != null)  {  correctFrame[1] = respBuffer[1];  }  else  {  MessageBox.Show("Nieznana długość ramki!!!");  return;  }  }  if (checklist[2] == States.CONTROL)  {  if (ASCIITOHex(respBuffer[2].ToString()) != null)  {  correctFrame[2] = respBuffer[2];  }  else  {  MessageBox.Show("Nie rozpoznano kontroli!!!");  return;  }  }  if (checklist[3] == States.GET\_FRAME)  {  if (ASCIITOHex(respBuffer[3].ToString()) != null)  {  correctFrame[3] = respBuffer[3];  }  else  {  MessageBox.Show("Nie rozpoznano ramki!!!");  return;  }  }  if (checklist[4] == States.CHECK\_BAIT)  {  if (ASCIITOHex(respBuffer[4].ToString()) != null)  {  correctFrame[4] = respBuffer[4];  }  else  {  MessageBox.Show("Nie rozpoznano instrukcji!!!");  return;  }  }  if (checklist[5] == States.CHECK\_CRC)  {  if (ASCIITOHex(respBuffer[5].ToString()) != null)  {  correctFrame[5] = respBuffer[5];  string ACK = ConvertHexToASCII("06");  serialPort1.Write(ACK);  }  }  else  {  MessageBox.Show("Nie znaleziono końca ramki!!!");  return;  }  }  string hex = BitConverter.ToString(correctFrame).Replace("-", " ");  txtReceive.Text += hex;  txtReceive.Text += " \n";  Array.Clear(respBuffer, 0, respBuffer.Length);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  } |

Objaśnienie:

Przy odbiorze danych wczytujemy bajty przesłane na port szeregowy do bufora, który pełni funkcję kolejki. Następnie tworzona jest tablica z poprawną ramką oraz tablica z wartościami enuma służąca do sprawdzenia poprawności wysyłanych danych. Jeśli ramka nie spełnia wymogów jest odrzucana. Walidowany jest początek ramki oraz zawartość poszczególnych komórek tablicy, w której są przechowywane sformatowane dane, które zostały poprawnie zwalidowane. Po sprawdzeniu ramki odsyłana jest wartość ACK (06). Na samym końcu ramka jest wyświetlana na ekranie.

* Przycisk czyszczący pola odpowiedzi i translacji na ASCII

|  |
| --- |
| private void clearScrollback\_Click(object sender, EventArgs e)  {  txtReceive.Text = String.Empty;  txtMessageASCII.Text = String.Empty;  } |

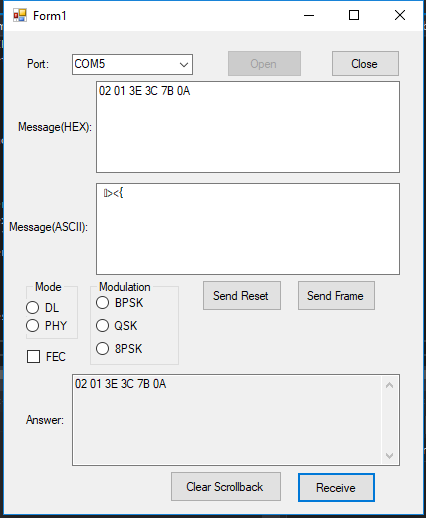
Enum:

|  |
| --- |
| public enum States { BEGINING, LENGTH, CONTROL, GET\_FRAME, CHECK\_BAIT, CHECK\_CRC } |

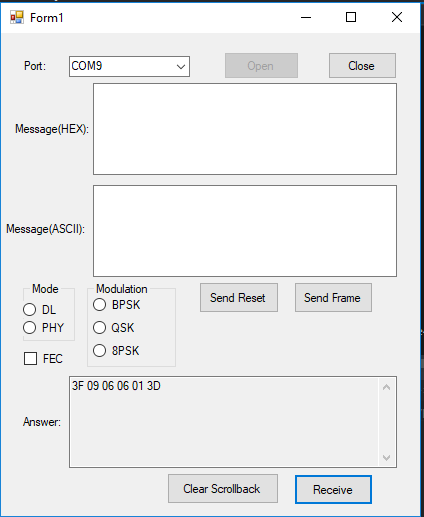
Dodatkowe metody:

|  |
| --- |
| private static bool isValid(String str)  {  bool valid = false;  Regex chars = new Regex((@"^([a-fA-F0-9]{2}\s+)+"));  if (chars.Match(str).Success)  valid = true;  else  valid = false;  return valid;  }  public string ConvertHexToASCII(String hexString)  {  StringBuilder sb = new StringBuilder();  for (int i = 0; i < hexString.Length; i += 2)  {  string hs = hexString.Substring(i, 2);  sb.Append(Convert.ToChar(Convert.ToUInt32(hs, 16)));  }  String ascii = sb.ToString();  return ascii;  }  public string ASCIITOHex(string ascii)  {  StringBuilder sb = new StringBuilder();  byte[] inputBytes = Encoding.UTF8.GetBytes(ascii);  foreach (byte b in inputBytes)  {  sb.Append(string.Format("{0:x2}", b));  }  string hex = sb.ToString();  return hex.ToUpper();  } |

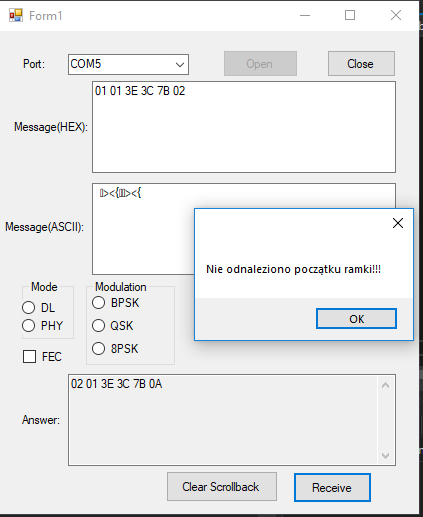
Działanie:



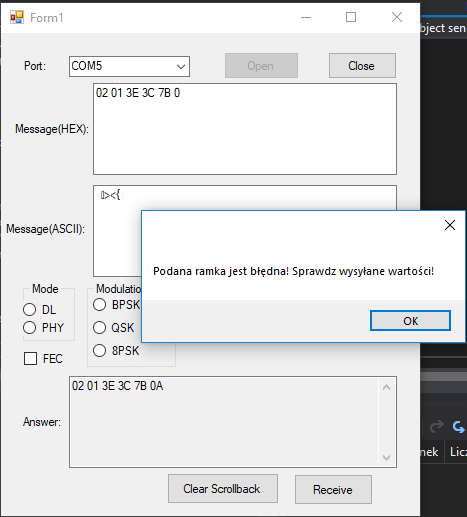
Przykład ukazujący walidację ramki wraz z zamianą na kod ASCII.



Odpowiedź modemu na wysłanie resetu (bez odsyłania ACK)



Błędna ramka – zły początek ramki



Błędna wartość liczby w formacie heksadecymalnym

Podsumowanie

Projekt był pewnego rodzaju wyzwaniem. Kilka jego elementów stanowiło o trudności wykonania zadania. Konwersje danych i odpowiednia prezentacja danych były dość uciążliwe, jednak ostatecznie udało się rozwiązać większość problemów dotyczących implementacji rozwiązania ukazanego powyżej.