**Data wykonania ćwiczenia: 18.11.2014 r.**

 **Data oddania sprawozdania: 28.11.2014 r.**

Prowadzący: dr inż. Jan Nikodem Wydział Elektroniki, **Informatyka**

Grupa:

Termin: Wt TN 13:15

**Sprawozdanie z laboratorium**

**Urządzenia Peryferyjne**

*Ćwiczenie 4*

*Współpraca komputera z modemem*

**Cel ćwiczenia**

Naszym zadaniem było nawiązanie połączenia między komputerem i modemem korzystając z portu szeregowego, a następnie uzyskanie połączenia między dwoma komputerami wykorzystując modem oraz wewnętrzną sieć telefoniczną (w drugim wariancie należało podłączyć modemy bezpośrednio między sobą). Kolejnym zadaniem była wymiana danych tekstowych między komputerami, a także przesłanie pliku. W celu realizacji ćwiczeń zapoznaliśmy się ze specyfiką obsługi modemu oraz służącymi do tego komendami.

**Wstęp teoretyczny**

Do wykonania zadań udostępnione zostały nam dwa modemy Shadow 56k firmy Pentagram. Modem ten do komunikacji z komputerem wykorzystuje port szeregowy, a do linii telefonicznej można podłączyć go używając kabla RJ-11. Oprócz tego modem posiada wyłącznik zasilania, wbudowany głośnik oraz zestaw diod sygnalizacyjnych, których znaczenie jest następujące:

* MR (modem ready) – modem gotowy do współpracy z komputerem
* HS (high speed) – transmisja danych odbywa się z maksymalną prędkością
* AA (auto answer) – praca w trybie automatycznego odbierania połączeń
* OH (on hook) – sygnalizacja podniesionej słuchawki
* RD (recived) – sygnalizuje odbieranie danych
* TD (transmit) – sygnalizuje wysyłanie danych
* CD (carrier detected) – obecność nośnej, gdy modem jest podłączony, a na linii znajduje się nośna

Sterowanie modemem możliwe jest dzięki wykorzystaniu komend Hayes’a. Cechą dla nich charakterystyczną jest to, że każda rozpoczyna się od znaków AT (skrót od attention). W trakcie zajęć korzystaliśmy z następujących komend:

* AT – sprawdzenie łączności między komputerem i modemem – modem odpowiada OK.
* ATA – odebranie połączenia przychodzącego (podniesienie słuchawki)
* ATH – zakończenie połączenia (odłożenie słuchawki)
* ATDT *numer* – wybranie tonowe podanego numeru telefonu
* ATE n – włączenie (dla n=1) lub wyłączenie (dla n=0) funkcji echa
* ATS0=n – włączenie (n=1) lub wyłączenie (n=0) funkcji automatycznego odbierania połączeń
* ATO – powrót z trybu komend (wywołanego sekwencją +++) do trybu danych.

Ponadto modem może wysyłać do komputera komunikaty informacyjne, np.:

* OK – potwierdzenie prawidłowego wykonania komendy
* RING – informuje o połączeniu przychodzącym
* CONNECT – informuje o pomyślnym nawiązaniu łączności między modemami i przejściu do trybu danych
* NO CARRIER – informuje o braku sygnału nośnej (np. linia telefoniczna nie jest podłączona)
* BUSY – informuje, że wybierany numer jest obecnie zajęty

Modem może pracować w jednym z dwóch trybów: komend bądź danych. Domyślnym trybem jest tryb komend, który umożliwia wysyłanie poleceń do modemu. Kiedy uzyskamy już połączenie, modem automatycznie przechodzi do trybu transmisji danych. W trybie danych sterowanie modemem za pomocą komend nie jest możliwe, bowiem przesyła on dalej wszystkie dane otrzymywane z komputera. W celu przełączenia modemu ponownie na tryb komend należy wysłać sekwencję +++. Aby ponownie przełączyć się z trybu komend na tryb danych, trzeba użyć polecenia ATO.

**Komunikacja między komputerami przy wykorzystaniu modemów**

Do wykonania zadań związanych z nawiązywaniem połączenia i wymianą danych między komputerami użyliśmy windowsowego programu Hyper Terminal. Przed przystąpieniem do wykonywania zadań upewniliśmy się, czy modemy są włączone oraz prawidłowo podłączone do linii telefonicznej, a także jakie numery telefonów są przypisane tym liniom. Ponadto sprawdziliśmy do którego portu COM podłączone są modemy – okazał się być to COM1. W Hyper Terminalu utworzyliśmy nowe połączenie (port COM1, szybkość: 9600, bity danych: 8, brak parzystości, bity stopu: 1). Następnie na jednym z komputerów wydaliśmy polecenie ATD 3964. Na drugim komputerze w terminalu pojawił się komunikat RING. Po wydaniu polecenia ATA rozpoczął się proces nawiązywania połączenia. Kiedy w obu terminalach otrzymaliśmy komunikat CONNECTED mogliśmy wymieniać dane między komputerami – znaki wpisywane na jednym komputerze pojawiały się na ekranie drugiego i na odwrót. Dla modemów połączonych bezpośrednio (bez wykorzystania sieci telefonicznej) proces łączenia wyglądał podobnie, z tym, że nie musieliśmy podawać numeru telefonu, na który ma zostać wykonane połączenie. Połączenie zakończyliśmy poprzez przełączenie się ponownie na tryb komend (sekwencją +++) i wysłaniem polecenia ATH.

Program Hyper Terminal umożliwia także przesyłanie plików, co wykorzystaliśmy do realizacji drugiego zadania. Na komputerze wysyłającym należało wybrać Transfer -> Wyślij plik i określić lokalizację pliku na dysku. Drugi komputer należało ustawić w tryb odbierania poprzez wybranie Transfer -> Odbierz plik i ustawić lokalizację dla odbieranego pliku. Ze względu na niską szybkość transferu danych, funkcja ta może być przydatna jedynie dla małych plików (poniżej 0,5 MB). Transfer wykonaliśmy zarówno z komputera 1 na 2, jak i z 2 na 1, w obu przypadkach przesyłany plik (obrazek 100 kB) został przesłany poprawnie.

Ponadto napisaliśmy terminal, który służy do przesyłania danych między komputerami. Na terminal składają się 2 aplikacje: Terminal nadawczy oraz Terminal odbiorczy. Terminal nadawczy służy do nawiązywania połączenia oraz wysyłania wpisywanych z klawiatury danych, natomiast Terminal odbiorczy po uruchomieniu ustawia modem w tryb automatycznego odbioru połączeń i czeka na nawiązanie połączenia. Kiedy połączenie zostanie nawiązanie, Terminal odbiorczy będzie wyświetlał odbierane znaki. W obu aplikacjach rozłączenie następuje po naciśnięciu klawisza ESC. Terminal działa tylko przy bezpośrednim połączeniu modemów (bez udziału sieci telefonicznej). Do komunikacji szeregowej wykorzystaliśmy wcześniejsze rozwiązanie, z którego korzystaliśmy w czasie laboratoriów z GPS.   
Działanie programu zostało przetestowane dzięki wykorzystaniu emulatora modemu Virtual Modem firmy Fabulatech. Poniżej przedstawiamy najważniejsze części kodu źródłowego programu.

**Terminal odbiorczy:**

#include <iostream>

#include <windows.h>

#include <string>

#include <stdlib.h>

#include <sstream>

#include "Serial.h"

#include "stdafx.h"

using namespace std;

#define RX\_BUFFSIZE 512

int i;

bool wyjscie = true;

int \_tmain(int argc, \_TCHAR\* argv[])

{

try

{

cout << "Otwieram port..." << endl;

Serial serial(9600);

cout << "Polaczono" << endl << "Aby sie rozlaczyc wcisnij klaiwsz ESC" << "\n\n";

char tablica[RX\_BUFFSIZE];

serial.write("A"); //ustawienie modemu w tryb automatycznego odbierania po pierwszym sygnale

serial.write("T");

serial.write("S");

serial.write("0");

serial.write("=");

serial.write("1");

serial.write("\r");

do {

if (GetAsyncKeyState(VK\_ESCAPE)) //sprawdzanie stanu klawisza ESC

{

wyjscie = false;

}

else

{

int charsRead = serial.read(tablica, RX\_BUFFSIZE); //wyswietlanie zawatrosci tablicy w petli

cout << tablica;

}

}

while (wyjscie);

serial.write("A"); //zakonczenie polaczenia

serial.write("T");

serial.write("H");

serial.write("\r");

serial.close();

cout << "\nRozlaczono";

}

catch(const char \*msg)

{

cout << msg << endl;

}

cout << "\nNacisnij ENTER..." << endl;

cin.get();

return 0;

}

**Terminal nadawczy:**

#include <iostream>

#include <windows.h>

#include <string>

#include <stdlib.h>

#include <sstream>

#include "Serial.h"

#include "stdafx.h"

#include <conio.h>

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

try

{

cout << "Otwieram port..." << endl;

Serial serial(9600);

cout << "Polaczono" << endl;

serial.write("A"); // komenda ATD - nawiazanie polaczenia

serial.write("T");

serial.write("D");

serial.write("\r");

Sleep(3000);

cout << "Nadawanie rozpoczete, naciscnij ESC zeby przerwac\n";

serial.write("Polaczenie rozpoczete\n\r");

char litera[2];

litera[1] = 0;

while (true)

{

litera[0] = getch(); //pobranie znaku z klawiatury

if (litera[0] == 27) //jesli znak to escape to przerwij wczytywanie

break;

serial.write(litera); //wyslij znak

}

serial.write("+"); //sekwencja escape'owa- wyjscie z trybu danych, przejscie do

Sleep(50); //trybu komend

serial.write("+");

Sleep(50);

serial.write("+");

Sleep(1000);

serial.write("A"); // komenda ATH- zakonczenie polaczenia

serial.write("T");

serial.write("H");

serial.write("\r");

cout << "\n\nPolaczenie zakonczone\n\n";

serial.close(); //zamkniecie polaczenia

}

catch(const char \*msg)

{

cout << msg << endl;

}

cout << "Nacisnij ENTER..." << endl;

cin.get();

return 0;

}

**Wnioski**

Dzięki tym zajęciom mieliśmy okazję zapoznać się ze specyfiką pracy modemu telefonicznego – jak komunikuje się on z komputerem oraz jak wygląda komunikacja, zarówno poprzez sieć telekomunikacyjną, jak i przy połączeniu bezpośrednim. Ponadto poznaliśmy budowę i zastosowanie komend Hayes’a. Zdobyta wiedza pozwoliła nam stworzyć programy wykorzystujące połączenie modemowe do komunikacji. Program nadawczy można ponadto rozszerzyć o funkcję wybierania numeru, jednak ze względu na to, że nie mieliśmy do dyspozycji „fizycznych” modemów, a modem emulowany wykonywał tylko połączenia na drugi emulowany modem, nie mieliśmy możliwości przetestowania tej funkcji, wobec tego z niej zrezygnowaliśmy. Poza tym warto zauważyć, że modemy telefoniczne, ze względu na bardzo niską prędkość transmisji danych oraz rezygnowanie przez ludzi ze stacjonarnych linii telefonicznych praktycznie wyszły już z użycia.