

Katedra Metrologii i Systemów Elektronicznych
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

**Projektowanie i Organizacja Systemów
Elektronicznych**

Protokół Modbus

Marek Niedostatkiewicz

Grzegorz Rojek

Gdańsk 2004

ZADANIA DO PRZYGOTOWANIA PRZED LABORATORIUM.

1. Zapoznać się z opisem stanowiska laboratoryjnego (rozdział 1) i protokołu Modbus (rozdział 2) . Jeżeli był już wykład z Modbus w ramach PiOSE, lektura będzie łatwiejsza.
2. Przypomnieć sobie podstawowe wiadomości dotyczące RS485 z wykładu z Interfejsów Systemów Elektronicznych. Ewentualnie pomocny może być (rozdział 3).
3. Znając strukturę systemu i podstawy protokołu (pkt.1) oraz właściwości interfejsu RS485 pracującego w dwudrucie, half-duplex), zapoznać się z rozwiązaniami zastosowanymi w prezentowanym systemie, szczególnie uwzględniając konwertery RS485-RS232 (rozdział 4).
4. Zapoznać się z możliwościami oprogramowania PC (rozdział 6) – implementującego odpowiednio sterownik i urządzenie wykonawcze Modbus.
5. Ewentualną ciekawość – przeznaczenie złącz i przełączników modułów zaspokoi (rozdział 7), zaś chęć poznania metod obliczania sum kontrolnych CRC16 zaspokoi Dodatek A.
6. Zastanowić się:
 - Dlaczego niezbędne jest przełączanie kierunku transmisji w konwerterach?
 - W którym z trybów Modbus (ASCII/RTU) uzyskuje się większą efektywną prędkość transmisji danych przy tej samej prędkości transmisji RS232?
 - Dlaczego tryb RTU nie może mieć ustalonego znaku końca transmisji jak tryb ASCII?
 - Jak można rozwiązać wykrywanie końca transmisji Slave – mikrokontrolerze?
 - Jakie mamy (cztery) typy danych dla funkcji Modbus i co mogą reprezentować?

ZADANIA DO WYKONANIA W LABORATORIUM.

1. Sprawdzić kompletację stanowiska i zgodność ustawień przełączników modułów z opisem. Podłączyć zasilanie
2. Uruchomić oprogramowanie Master na jednym i Slave na drugim z komputerów.
3. Sprawdzić działanie funkcji zaimplementowanych w sterowniku dotyczących wyjść modułów Slave (2 typy danych). Zaobserwować zmiany w modułach Slave – sprzętowym i programowym.
4. Sprawdzić działanie funkcji zaimplementowanych w sterowniku dotyczących wejść modułów Slave (2 typy danych). Korzystając z trybu wielokrotnego odczytu – zmieniać nastawy sygnałów wejściowych modułu Slave na PC lub ustawienie przełącznika zadającego sygnały dla modułu Slave na uC. Obserwować wizualizację odczytanych parametrów na panelu Modbus master.
5. Dla 4 wybranych funkcji, rozpisać ramkę Modbus (widoczną na ekranie komputera z Modbus Master) korzystając z dołączonej do ćwiczenia specyfikacji protokołu – wykorzystać strony z opisem funkcji Modbus.
6. Dla wybranej funkcji symulować stany awaryjne (przekroczenie adresu początkowego cewki/wejścia/rejestru, przekroczenie dopuszczalnej ilości wejść, wyłączenie oprogramowania Slave (brak odpowiedzi). Rozszyfrować otrzymywane kody błędów Modbus – zwrócić uwagę na mechanizm sygnalizacji błędów w Modbus.
7. Ustalić z prowadzącym (i wykonać:) zadanie dodatkowe.

1. Wstęp

Protokół Modbus został opracowany w firmie Modicon w 1980r [2]. Mimo upływu dość znacznego czasu od chwili wprowadzenia jest on nadal szeroko stosowany w aplikacjach automatyki przemysłowej o niskich wymaganiach dotyczących szybkości i częstotliwości transmisji danych, w szczególności w systemach z wydzielonym centrum, do którego przesyłane są dane z urządzeń peryferyjnych. Jest standardem przyjętym przez większość producentów sterowników przemysłowych dla asynchronicznej komunikacji pomiędzy urządzeniami wyposażonymi w interfejs zgodny z RS-232 takich jak: RS-422, RS-485, modem i innych. W procedury komunikacyjne realizujące ten protokół są wyposażone niemal wszystkie dostępne na rynku pakiety SCADA (systemy nadzoru i akwizycji danych).

Modbus swą popularność zyskał dzięki prostocie zastosowanych w nim rozwiązań, jawności specyfikacji protokołu, a ponadto takim cechem jak: dostęp do łącza na zasadzie „Master - Slave” (lub inaczej Query-Response), zabezpieczenie komunikatów przed przekłamaniami, potwierdzenie wykonania rozkazów i sygnalizacja błędów oraz mechanizmy unikające zawieszania się systemu. Pozwala to na łatwą implementację w dowolnym urządzeniu posiadającym mikrokontroler i w znacznym stopniu wpływa na obniżenie kosztów.

W modelu ISO/OSI protokół Modbus zajmuje trzy warstwy: 1 - fizyczną (physical), 2 - łącza danych (data link) oraz 7-aplikacji (application).

Warstwa pierwsza zawiera ustalenia zawarte w standardzie określonego interfejsu szeregowego jako platformy dla protokołu Modbus, dotyczące techniki transmisji jak również samego medium transmisyjnego definiowanego przez ten standard. Warstwa druga (poziom ramki) realizuje metodę („Query-Response”) sterowania dostępem do medium transmisyjnego.

Warstwa siódma określa sposób komunikowania się programów użytkowych z urządzeniami systemu za pośrednictwem protokołu Modbus.

Może pracować w dwóch trybach transmisji: RTU i ASCII. Istnieje również implementacja na stosie TCP/IP w sieci Ethernet.

Przykładowymi aplikacjami Modbus są: zdalna akwizycja danych, kontrola procesów przemysłowych, systemy nadzoru, systemy ochrony, procesy monitorowania, zarządzanie energią, laboratoria automatyki itp.

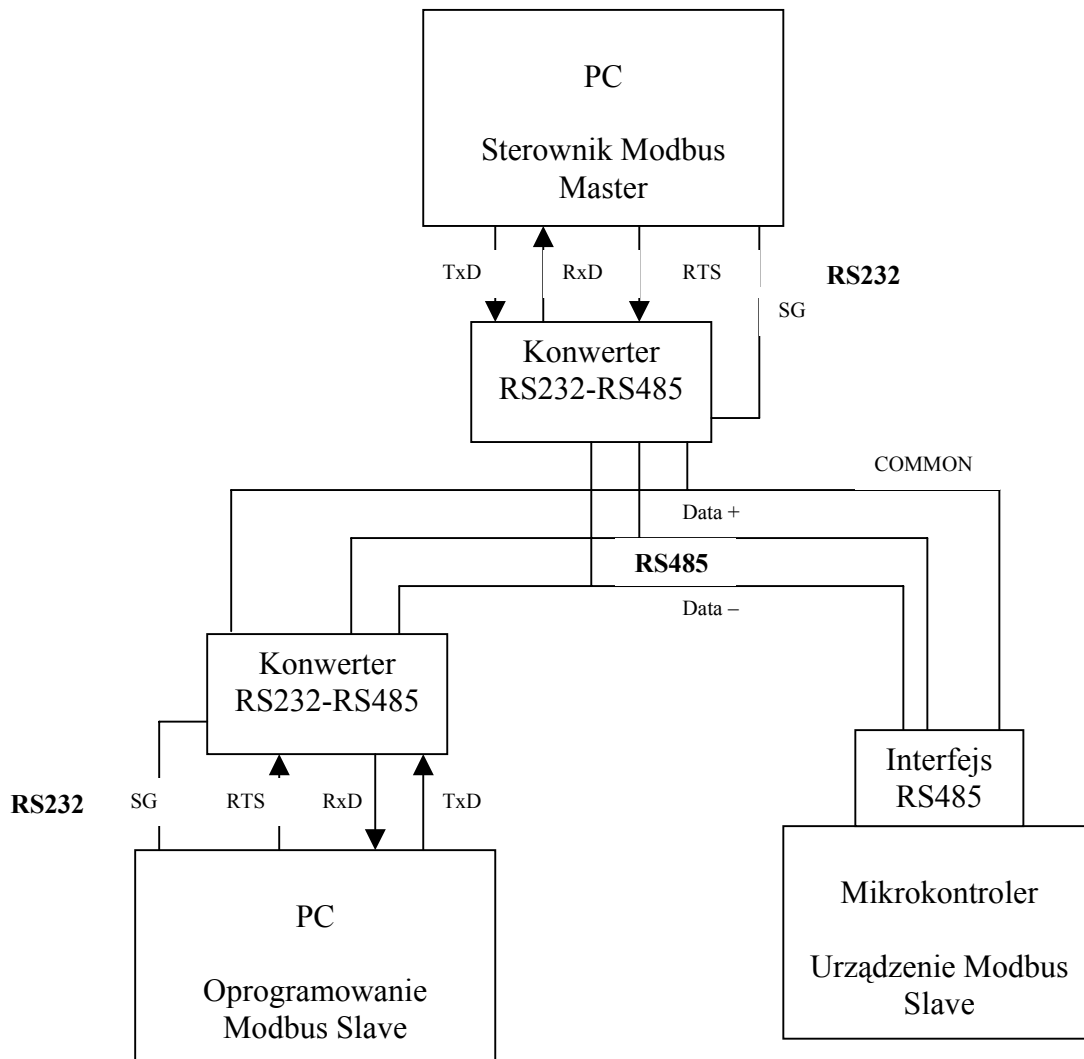
Najczęściej stosowaną i używaną konfiguracją w automatyce przemysłowej jest protokół Modbus współpracujący z interfejsem RS485, gdzie występuje jedno urządzenie nadrzędne (Master) inicjalizujące transakcje (wysyłające polecenie), natomiast pozostałe urządzenia są podrzędne (Slaves), wykonują polecenia Master-a i odsyłają odpowiedź. W danej chwili tylko jeden Slave może odpowiadać na zdalne zapytanie Master-a, natomiast nie ma możliwości komunikacji pomiędzy urządzeniami podrzędnymi. Typowym Master-em jest urządzenie z procesorem głównym (host procesor), zawierające programowalny panel na przykład komputer PC lub nadrzędny sterownik logiczny, a typowy Slave to programowalny sterownik logiczny. Węzły podrzędne (Slaves) są wykorzystywane do sterowania oraz zbierania danych z urządzeń peryferyjnych takich jak: mierników, liczników, przetworników A/C i C/A, czujników, przełączników, sygnalizatorów itp.

Jako interfejs komunikacyjny dla protokołu Modbus zastosowano magistralę RS485, pozwalającą pracować w warunkach silnych zakłóceń (np. w przemyśle) oraz na uzyskanie znacznych zasięgów transmisji i również chętnie stosowaną w układach automatyki i sterowania.

W skład stanowiska laboratoryjnego wchodzi: jedno urządzenie nadrzędne (Master), dwa urządzenia podrzędne (Slaves), oraz dwa konwertery RS232-RS485 niezbędne dla komunikacji urządzeń w standardzie RS485, ponieważ elementy Master i jeden z elementów podrzędnych zostały zrealizowane na komputerach PC, wyposażonych w interfejs RS232C.

- Sterownik Modbus Master - konfigurowany przez użytkownika, umożliwiający sterowanie jednostkami podrzędnymi oraz sygnalizujący potwierdzenie wykonania polecenia, a w przypadku braku wykonania polecenia wskazujący błędy;
- Sterownik Modbus Slave wykonany na komputerze PC - konfigurowany przez użytkownika, sterowany zdalnie przy pomocy poleceń z jednostki Modbus Master, operujący na własnych zasobach systemowych;
- Urządzenie Modbus Slave wykonany na mikrokontrolerze - konfigurowany przez użytkownika, wykonujący zdalne polecenia jednostki Master poprzez zarządzanie cyfrowymi i analogowymi liniami we/wy oraz wewnętrznymi modułami mikrokontrolera takimi jak: pamięć, układy czasowo-licznikowe, układy przerwań przetwornik A/C itp;

- Konwertery RS232-RS485 powinny zapewniać pracę z przełączaniem nadawanie/odbiór sterowanym przez komputer PC oraz pracę z automatycznym załączaniem nadajnika przez konwerter.



Rys.1 Schemat blokowy stanowiska do badania protokołu Modbus.

2. Opis protokołu Modbus

Modbus jest protokołem komunikacyjnym pracującym na bazie interfejsów szeregowych wykorzystujących asynchroniczną transmisję znakową o dostępie do łącza typu „Master – Slave”. Tylko jedno urządzenie może być jednostką nadrzędną – Master, inicjującym transakcję - wysyłającym zapytanie (Query), pozostałe (Slaves) odpowiadają jedynie na jego zdalne zapytania wysyłając odpowiedź (Response). Istnieje również odmiana protokołu Modbus z więcej niż jednym Master-em - Modbus Multi-Master, w którym jednostki nadrzędne przekazują sobie wzajemnie prawo do nadawania.

Protokół pracuje z niewielkimi prędkościami transmisji danych (typowe: 9.6 Kb/s, 19.2 Kb/s) na ograniczonym dystansie wynikającym z typu zastosowanego typu łącza komunikacyjnego (RS-232, RS-422, RS-485, Modem).

Kompletną specyfikację protokołu zawiera dokument „Modicon Modbus Protocol Reference Guide”, ostatnia wersja tego dokumentu jest oznaczona literą J [**Błąd! Nie można odnaleźć źródła odsyłacza.**].

2.1. Cechy protokołu

Najważniejsze cechy protokołu Modbus to:

- zasada dostępu do łącza „Query – Response” („Master-Slave”) gwarantuje bezkonfliktowe współdzielenie magistrali przez wiele węzłów;
- węzeł nadrzędny (Master) - komputer np. klasy PC steruje pracą sieci;
- węzły podrzędne (Slaves) - pozostałe węzły nie podejmują samodzielnie transmisji, odpowiadają na zdalne polecenia od węzła nadrzędnego;
- każdy z węzłów podrzędnych posiada przypisany unikalny adres;
- dwa różne tryby transmisji ASCII (znakowy) lub RTU (binarny),
- komunikaty zawierające polecenia i odpowiedzi mają identyczną strukturę;
- maksymalna długość komunikatów wynosi 256 bajtów;
- znaki są przesyłane szeregowo od najmłodszego do najstarszego bitu.

2.2. Zalety protokołu

Główne zalety protokołu Modbus to:

- prostota zastosowanych w nim rozwiązań;
- jawność specyfikacji protokołu;
- zabezpieczenie przesyłanych komunikatów przed błędami;
- potwierdzanie wykonania rozkazów zdalnych i sygnalizacja błędów;
- stały format ramki i zestaw standardowych funkcji służących wymianie danych;
- mechanizmy zabezpieczające przed zawieszeniem systemu.

2.3. Ramka transmisji

Ramka protokołu Modbus określa format przesyłanych wiadomości i zawiera: adres odbiorcy, kod funkcji reprezentujący żądane polecenie, dane dotyczące funkcji oraz słowo kontrolne zabezpieczające przesyłaną wiadomość.

Postać ramki zapytania wysyłanego przez jednostkę Master i ramki odpowiedzi jednostki Slave jest podobna. Różnica polega na tym, że w polu danych ramki odpowiedzi występują dane, których dostarczenia żądała stacja Master.



Rys. 2 Ramka protokołu Modbus.

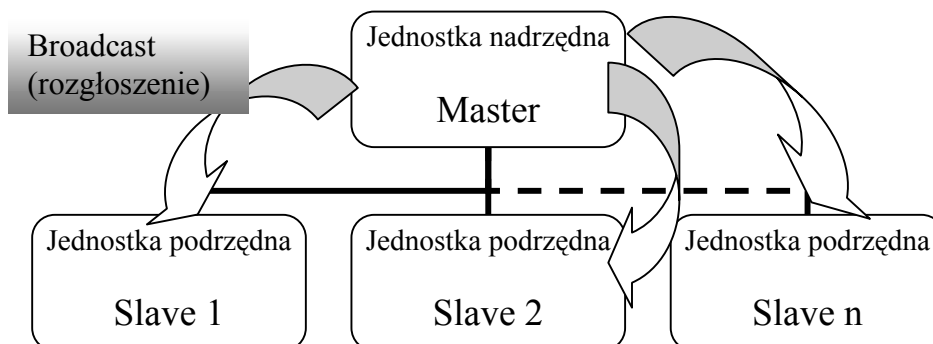
Opis poszczególnych pól ramki:

- *adres SLAVE:* liczba z zakresu 1 – 247, 0 – adres rozgłoszeniowy;
- *funkcja:* jest liczbą z zakresu 1...127;
- *pole danych:* jego długość zależy od rodzaju wiadomości i może zawierać:
 - w przypadku zapytania – argumenty funkcji;
 - w przypadku pozytywnej odpowiedzi - argumenty funkcji;
 - w przypadku szczególnej odpowiedzi - kod błędu;
 - w niektórych przypadkach może być równa 0;
- *suma kontrolna:* wyznaczana z zawartości przesyłanego komunikatu.

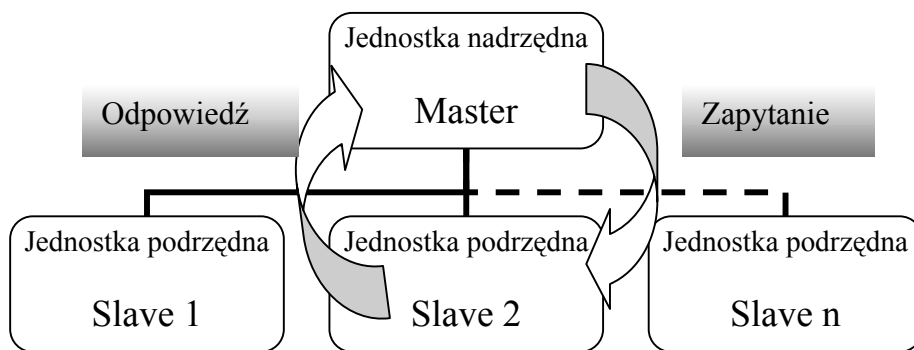
2.4. Typy transakcji

Możliwe są dwa typy transakcji:

- „rozgłoszenie” (*Broadcast*) – jednostka Master wysyła jedynie ramkę rozgłoszenia o adresie 0, który nie jest przypisany do konkretnego urządzenia Slave, ale wszystkie urządzenia ją odbierają i wykonują zawartą w niej funkcję nie odsyłając odpowiedzi.



- „zapytanie – odpowiedź” (*Query-Response*) – jednostka Master wysyła zapytanie i oczekuje na odpowiedź od wybranego urządzenia Slave;

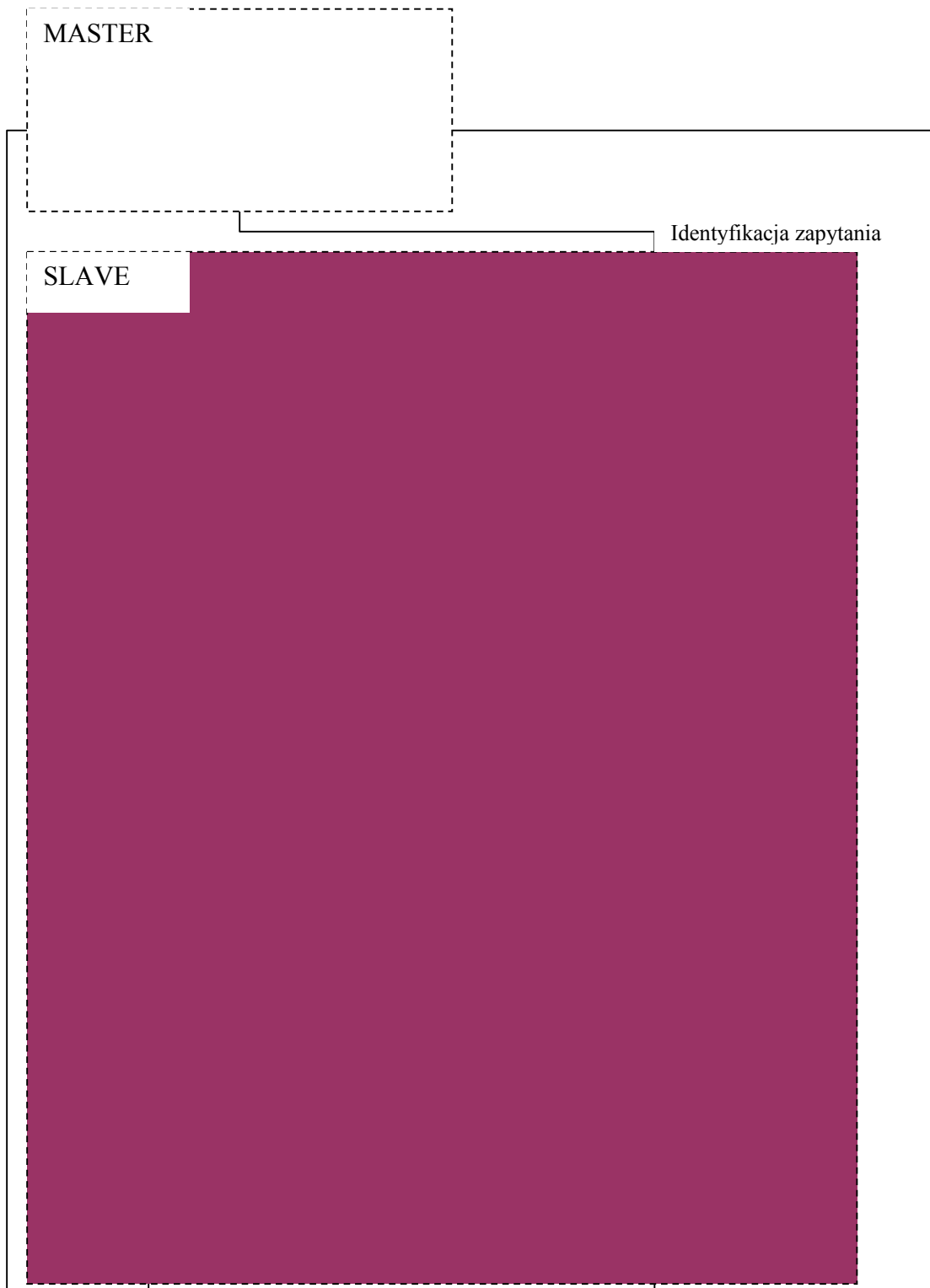


Typowe zachowanie urządzenia Slave po odbiorze zapytania i zweryfikowaniu adresu może wyglądać następująco:

- w przypadku poprawnego odbioru i bezbłędnej interpretacji ramki zapytania wysyłanej przez Master odpowiada również zgodnie z formatem zdefiniowanym w protokole Modbus;
- gdy wystąpi błąd przy odbiorze wiadomości lub Slave nie jest w stanie wykonać polecenia wysyła tzw. szczególną odpowiedź (*Exception Response*), w ramce odpowiedzi wysyłanej przez Slave w polu kodu funkcji ustawiany jest najstarszy bit, natomiast w polu danych umieszczany jest kod błędu, umożliwiający węzłowi nadrzdnemu określenie przyczyny jego wystąpienia;
- w przypadku błędu sumy kontrolnej lub innych błędów w ramce zapytania nie zdefiniowanych w kodach błędów, Slave nie wysyła odpowiedzi.

Standardowe kody błędów odpowiedzi wyjątkowej (szczegółnej):

- 01 – niedozwolona funkcja
- 02 – niedozwolony zakres danych(adres)
- 03 – niedozwolona wartość danej
- 04 – błąd urządzenia Slave
- 05 – potwierdzenie pozytywne
- 06 – brak gotowości urządzenia Slave
- 07 – potwierdzenie negatywne
- 08 – błąd parzystości pamięci



Rys. 3 Diagram transakcji w protokole Modbus.

2.5. Tryby transmisji

Protokół Modbus definiuje dwa tryby transmisji: ASCII (znakowy), RTU (binarny).

Tryb ASCII:

- każdy bajt w wiadomości przesyłany jest w postaci dwóch znaków ASCII,

ZNACZNIK POCZĄTKU	ADRES	FUNKCJA	DANE	KONTROLA LRC	ZNACZNIK KOŃCA
1 ZNAK ';'	2 ZNAKI	2 ZNAKI	n ZNAKÓW	2 ZNAKI	2 ZNAKI 'CR' i 'LF'

Tryb RTU :

- wiadomości rozpoczynają się odstępem czasowym trwającym minimum $3,5 * T_z$, gdzie T_z - czas trwania pojedynczego znaku;
- cała ramka musi zostać przesłana w postaci ciągłej tzn. odstęp pomiędzy kolejnymi znakami nie może być większy od $1,5 * T_z$.

ZNACZNIK POCZĄTKU	ADRES	FUNKCJA	DANE	KONTROLA CRC	ZNACZNIK KOŃCA
ODSTĘP $3,5 x$	8 BITÓW	8 BITÓW	n x 8 BITÓW	16 BITÓW	PRZERWA $3,5 x$

W węzle nadrzędnym ustalany jest pewien maksymalny czas oczekiwania na odbiór ramki zawierającej komunikat z odpowiedzią (Timeout). Jego wartość musi być tak dobrana, aby nawet najwolniejszy z węzłów podrzędnych zdążył odesłać odpowiedź. Przekroczenie tego czasu powoduje przerwanie transakcji. Przyczyn braku odpowiedzi może być kilka.

Najczęstszymi przyczynami braku odpowiedzi są:

- wystąpienie błędu transmisji ramki polecenia,
- zaadresowanie nieistniejącego węzła podrzędnego.

2.6. Format znaku przy transmisji szeregowej

W standardzie Modbus znaki są przesyłane szeregowo od najmłodszego do najstarszego bitu.

Organizacja jednostki informacyjnej w trybie ASCII:

- 1 bit startu,
- 7 bitów pola danych, jako pierwszy wysyłany jest najmniej znaczący bit,
- 1 bit kontroli parzystości (nieparzystości) lub brak bitu kontroli parzystości,
- 1 bit stopu przy kontroli parzystości lub 2 bity stopu przy braku kontroli parzystości.

START	1	2	3	4	5	6	7	PAR		STOP
START	1	2	3	4	5	6	7	STOP		STOP

Organizacja jednostki informacyjnej w trybie RTU:

- 1 bit startu,
- 8 bitów pola danych, jako pierwszy wysyłany jest najmniej znaczący bit,
- 1 bit kontroli parzystości (nieparzystości) lub brak bitu kontroli parzystości,
- 1 bit stopu przy kontroli parzystości lub 2 bity stopu przy braku kontroli parzystości.

START	1	2	3	4	5	6	7	8		PAR	STOP
START	1	2	3	4	5	6	7	8		STOP	STOP

Zabezpieczenie znaku jest generowane przez urządzenie nadające i dołączane do wiadomości przed transmisją. Urządzenie odbierające sprawdza bit parzystości znaku (o ile występuje).

2.7. Standardowe funkcje Modbus:

Funkcje protokołu Modbus dzielą się na funkcje publiczne (standardowe) i funkcje definiowane przez użytkownika.

Są to rozkazy wysyłane przez jednostki nadrzędne oraz interpretowane i wykonywane przez jednostki podrzędne (Slaves).

Większość opisanych w specyfikacji protokołu poleceń służy do realizacji operacji na sprzętowych zasobach sterowników przemysłowych – cyfrowych wejściach (INPUTS) i wyjściach (COIL) oraz rejestrach wejściowych (INPUT REGISTERS) oraz rejestrach wyjściowych (HOLDING REGISTERS). Rejestry są 16-bitowe i mogą mapować np. układy liczników lub przetworników A/C i C/A modułu wykonawczego.

Producenci sterowników przemysłowych wyposażonych w kontrolery przeznaczone dla standardu Modbus opracowują dla wykonywanych przez siebie urządzeń wiele własnych funkcji.

Typowe funkcje zdefiniowane w protokole Modbus:

- Read Coil Status - odczyt bieżącego stanu grupy wyjść cyfrowych,
- Read Input Status - odczyt stanu grupy wejść cyfrowych,
- Read Holding Register - odczyt zawartości grupy rejestrów wyjściowych,
- Read Input Register - odczyt zawartości grupy rejestrów wejściowych,
- Force Single Coil - ustawienie stanu jednego wyjścia cyfrowego,
- Preset Single Register - zapis do pojedynczego rejestru wyjściowego,
- Read Exception Status - odczyt statusu urządzenia Slave,
- Diagnostics - test diagnostyczny,
- Force Multiple Coils - ustawienie stanu grupy wyjść cyfrowych,
- Preset Multiple Register - zapis do grupy rejestrów wyjściowych,
- Report Slave ID - odczyt ID jednostki Slave,
- Reset Communication Link - resetowanie połączenia,
- Read General Reference - odczyt rejestrów w pamięci rozszerzonej,
- Write General Reference - zapis do rejestrów w pamięci rozszerzonej,
- Mask Write 4X Register - maskowanie grupy 4 rejestrów,
- Read/Write 4X Register - odczyt/zapis grupy 4 rejestrów,
- Read FIFO Queue - odczyt kolejki FIFO.

2.8 Generacja słów zabezpieczających

W protokole Modbus jako zabezpieczenie ramki wiadomości stosuje się sumę kontrolną. Jej wartość wyznacza urządzenie nadające dla zawartości przesyłanego komunikatu i umieszcza w ramce po części informacyjnej. Węzeł odbiorczy oblicza sumę kontrolną dla odebranego komunikatu i porównuje jej wartość z wartością otrzymaną. Niezgodność sum świadczy o wystąpieniu błędu. Dla trybu ASCII stosuje się sumę kontrolną typu LRC (Longitudinal Redundancy Check) natomiast dla trybu RTU – CRC (Cyclical Redundancy Check).

Generacja LRC

Wartość LRC (8-bitowa) jest dołączana na końcu ramki w postaci dwóch znaków ASCII. Obliczanie LRC polega na sumowaniu kolejnych 8-bitowych bajtów wiadomości, odrzuceniu przeniesień, a na koniec wyznaczeniu uzupełnienia dwójkowego wyniku. Sumowanie obejmuje całą wiadomość za wyjątkiem znaczników początku i końca ramki.

Generacja CRC

Słowo kontrolne CRC to 16-bitowa wartość dołączana do ramki w postaci dwóch 8-bitowych znaków.

Obliczanie CRC realizowane jest według następującego algorytmu:

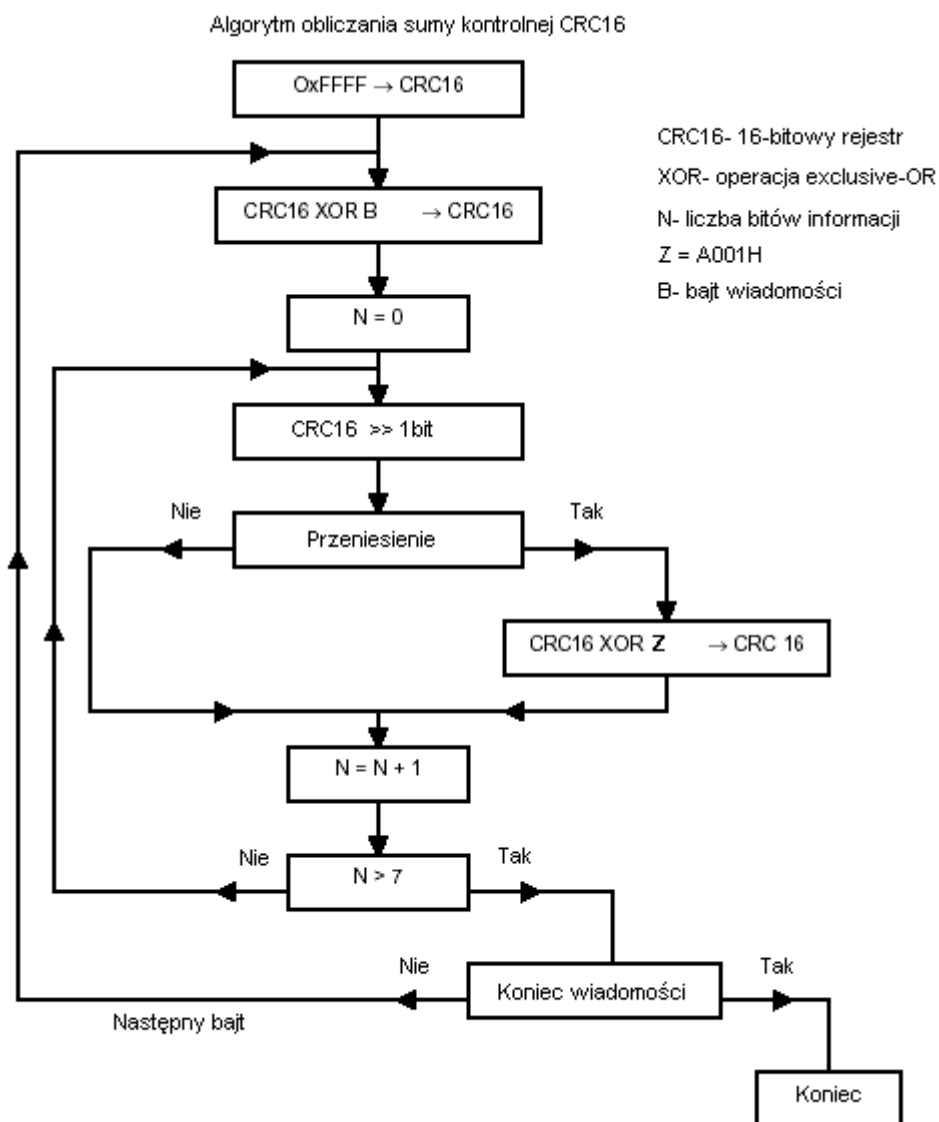
- 1) załadowanie wartości FFFFh do 16-bitowego rejestru;
- 2) pobranie bajtu z bloku danych (zabezpieczana wiadomość) i wykonanie operacji EXOR z młodszym bajtem rejestru, umieszczenie rezultatu w rejestrze;
- 3) przesunięcie zawartości rejestru w prawo o jeden bit połączone z wpisaniem 0 na najbardziej znaczący bit (MSB=0);
- 4) sprawdzenie stanu najmłodszego bitu (LSB) w rejestrze, jeżeli jego stan równa się 0, to następuje powrót do kroku 3 (kolejne przesunięcie), jeżeli 1, to wykonywana jest

operacja EXOR rejestru ze stałą A001h;

5) powtórzenie kroków 3 i 4 osiem razy, co odpowiada przetworzeniu całego bajtu;

6) powtórzenie sekwencji 2, 3, 4, 5 dla kolejnego bajtu wiadomości, kontynuacja tego procesu aż do przetworzenia wszystkich bajtów wiadomości;

7) zawartość rejestru po wykonaniu wymienionych operacji jest poszukiwaną wartością CRC.



Rys. 4 Diagram przedstawiający obliczanie sumy kontrolnej CRC16.

3. Opis interfejsu RS485

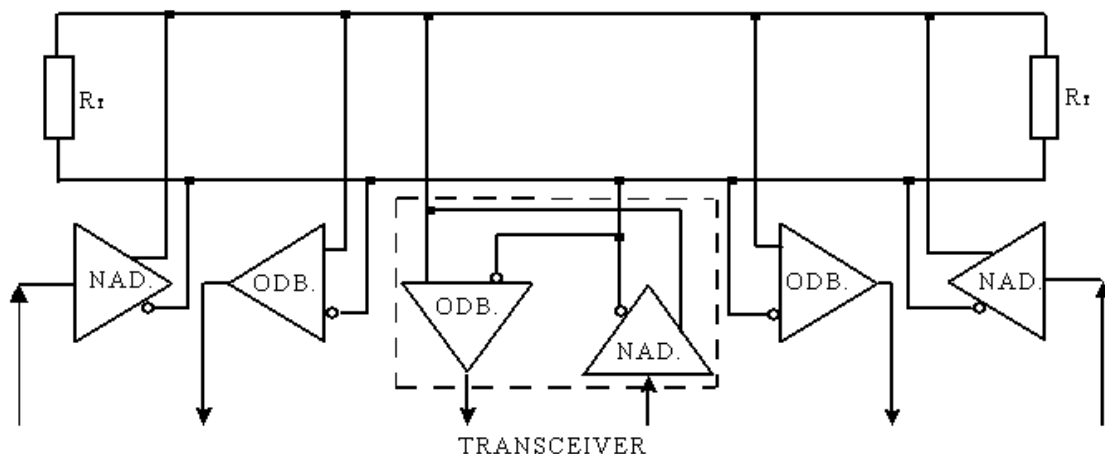
Interfejs RS485 został wprowadzony w 1983 roku jako rozszerzenie standardu RS422 i stał się bazą dla wielu protokołów komunikacyjnych służących do zdalnego programowania, monitorowania, sterowania i akwizycji danych [1]. Jest powszechnie stosowany w systemach pomiarowo-kontrolnych i automatyce przemysłowej, ponieważ umożliwia podłączenie do współdzielonej magistrali RS485 wielu urządzeń wyposażonych w interfejs RS485 takich jak: sterowniki, mierniki itp.

Jest łączem szeregowym asynchronicznym przeznaczonym do realizacji szybkiej transmisji danych na duże odległości w obecności zakłóceń elektromagnetycznych..

System transmisji złożony z różnicowych nadajników, dwuprzewodowego zrównoważonego toru przesyłowego oraz odbiorników o różnicowym obwodzie wejściowym.

Zastosowanie w omawianym standardzie trójstanowych nadajników pozwala na dołączenie do wspólnej linii wielu stacji nadawczo-odbiorczych. Warunkiem poprawnej pracy tak zorganizowanej sieci łączności jest przydzielanie w danym przedziale czasu dostępu do linii wyłącznie jednemu nadajnikowi, pozostałe powinny w tym czasie znajdować się w stanie wysokiej impedancji.

Na rysunku 5 przedstawiono wielopunktowy, zrównoważony interfejs cyfrowy zgodny ze standardem RS-485.



Rys. 5 Standard łącza zgodny z RS-485.

3.1. Cechy interfejsu RS485

Interfejs RS485 posiada następujące cechy:

- interfejs szeregowy;
- asynchroniczna transmisja danych;
- rodzaj transmisji: napięciowa, różnicowa symetryczna (odporna na zakłócenia);
- topologię magistralową (Linear Bus) z terminatorami, umożliwiającą proste dodawanie i usuwanie urządzeń;
- nie definiuje warstw wyższych modelu ISO/OSI;
- baza dla wielu protokołów realizujących wyższe warstwy modelu ISO/OSI;
- medium - najlepiej skrętka dwuprzewodowa, przy większych odległościach i zakłóceniach zaleca się przewód ekranowany;
- zapewnia transfer pół-duplex;
- możliwość “równoległego” dołączenia do linii transmisyjnej wielu urządzeń między którymi informacje mogą być przesyłane w dowolnym kierunku;
- ilość stacji - 32 stacje bez repeterów(wzmacniaków), 127 z repeaterami;
- maksymalna długość linii - 1200m;
- prędkość transmisji do 10Mbps, jednakowa dla wszystkich urządzeń;
- wymaga stosowania na końcach linii odpowiednich terminatorów.

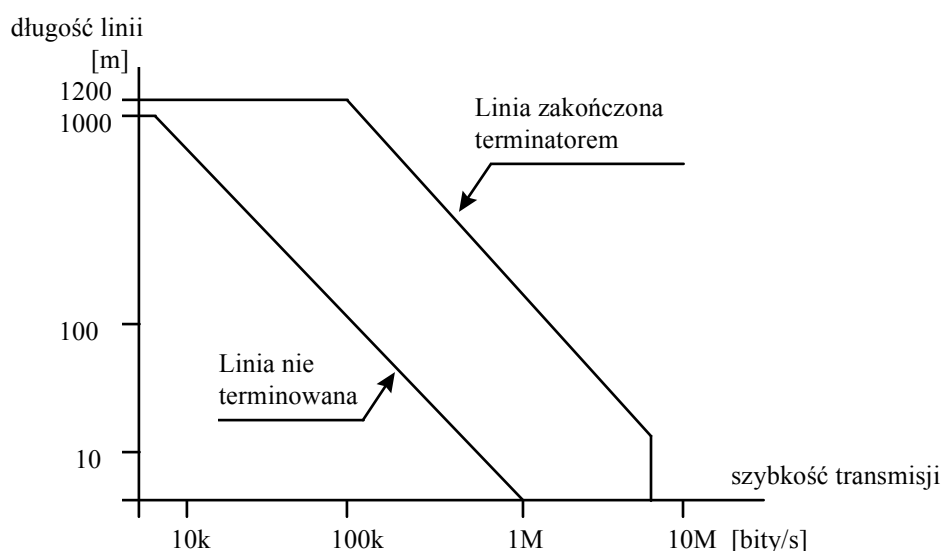
Właściwości nadajników, umożliwiające wielopunktową komunikację w standardzie RS485:

- jeden nadajnik może sterować do 32 jednostkowych obciążeń (obwody wprowadzające obciążenie do 1 mA) oraz zastępczą rezystancją dopasowującą $R_T = 60 \Omega$ lub większą;
- prąd upływu nadajnika w stanie wyłączenia nie może przekraczać 100 μA ;
- rezystancja wyjściowa nadajnika 120k Ω – wysoka impedancja (wyłączone zasilanie);
- nadajnik powinien zapewnić różnicowe napięcie wyjściowe z przedziału [-1.5V - 5V] przy obecności napięcia wspólnego z zakresu [-7V - 12V];
- nadajniki muszą mieć zabezpieczenie przed kolizją (jednoczesne nadawanie przez więcej niż jeden nadajnik nie może uszkodzić nadajnika).

Właściwości odbiorników, umożliwiające wielopunktową komunikację:

- rezystancja wejściowa $\geq 12 \text{ k}\Omega$;
- zakres napięcia wspólnego na wejściu odbiornika [-7V - 12V];
- czułość wejścia różnicowego 200mV w całym zakresie napięcia wspólnego.

Podobnie jak w innych standardach mamy do czynienia z kompromisem pomiędzy szybkością transmisji, a długością linii transmisyjnej. Na rysunku 6 przedstawiono graficzną prezentację tej zależności oraz wpływ dopasowania do linii transmisyjnej na obydwie te parametry.

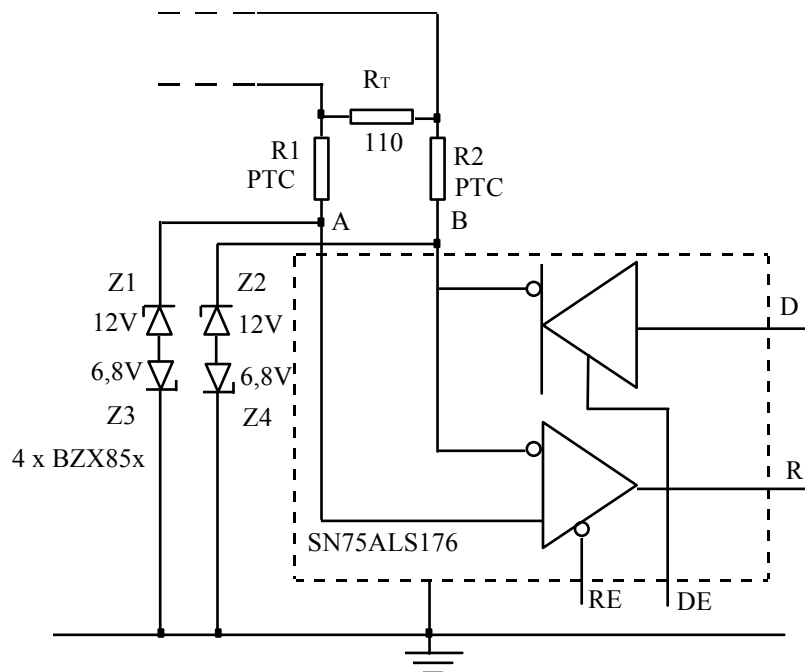


Rys. 6 Zależność długości linii transmisyjnej i szybkości transmisji dla przykładowego odbiornika linii typu SN75ALS194.

Innym ograniczeniem szybkości transmisji i długości linii są transmitery same w sobie. Każdy wnosi ze sobą pewne opóźnienie propagacji i czasu transmisji. Wprowadzono parametr informujący o zniekształceniu sygnału przez układ, definiowany jako stosunek czasu trwania użytecznego sygnału do czasu zmiany sygnału. Dla RS-485 stosunek ten wynosi 1 : 3,3. Standard zakłada użycie dwóch rezystorów dopasowujących zamykających magistrale z obydwu stron. Wartość terminatora dla przykładowego układu została określona

na poziomie $R_T = 110\Omega$ i wynika ona z ograniczeń standardu i parametrów linii transmisyjnej. Idealnym założeniem byłoby gdyby $R_T = Z_o$.

Często w miejscach gdzie dane są wysyłane na długie dystansy we wrogich otoczeniach elektrycznych, np. zautomatyzowanych fabrykach, odporność na zakłócenia linii różnicowych może nie wystarczać i potrzebne jest zewnętrzna pomoc. Na rysunku 7 przedstawiono przykładowe zabezpieczenie układu nadawczo-odbiorczego (transceivera) SN75ALS176 interfejsu RS-485. Diody Z1,Z2,Z3 i Z4 zabezpieczają przeciwko przypadkowym szpilkom napięciowym. Rezystory R1 i R2 typu PCT posiadają minimalną impedancję w normalnych warunkach pracy transceivera. Przy zmianie warunków otoczenia (wzrost temperatury) ich rezystancja rośnie dostarczając limitowanego prądu na wyprowadzeniach A i B za pośrednictwem łańcucha diod zabezpieczających.



Rys. 7 Przykładowe zabezpieczenie linii transmisyjnej standardu RS-485.

Przy konstruowaniu sieci RS485 zalecane jest stosowanie kabli komunikacyjnych kategorii V takich jak np. YNTKSXEK, LIYCY itp., natomiast złącz typu DB9, RJ45 lub RJ11.

3.2 Dedykowane protokoły komunikacyjne standardu RS-485

Na bazie interfejsu RS-485 opracowano kilka przemysłowych protokołów komunikacyjnych, które określają sposób komunikacji pomiędzy urządzeniami tworzącymi system. Do najbardziej popularnych możemy zaliczyć protokoły: Profibus, Modbus, P-Net, Bitbus, Dinbus oraz protokoły mogące współpracować opcjonalnie, takie jak LONTALK, protokół sieci CAN. Ważniejsze parametry wybranych protokołów przedstawia poniższa tabela.

Tabela 1. Parametry protokołów komunikacyjnych dla RS485.

Parametr	Profibus	Modbus	P-Net	Bitbus	Dinbus
Pełna nazwa	Process Field Bus	-	P-Net	Bitbus	
Twórca protokołu	Profibus	Modicon	Proces-Data	Intel	
Zaimplementowane warstwy normy ISO.	1, 2, 7	1, 2, 7	1, 2, 7	1, 2, (5), 7	1, 2
Minimalna szybkość transmisji [kbps]	9,6	-	76,8	62,5	0,11
Maksymalna szybkość transmisji [kbps]	500	19,2	76,8	2400	1000
Typ sieci	Token + Master / Slave	Master/ Slave	Token + Master / Slave	Master/ Slave	Master/ Slave
Ilość jednostek nadrzędnych	127	1	32	1	1
Maksymalna ilość zaadresowanych urządzeń	127 + rozszerzenie	247	125 + rozszerzenie	250	31
Typ kontroli ramki danych	FCS + QP	LCR, (CRC16),	BCS16/8	CRC16	BCC + QP
Dostęp do medium transmisyjnego	Master/Master, Master/Slave, Burst	Master/Slave, Broadcast	Master/Master, Master/Slave, Burst	Master/Slave	Master/Slave, Burst
Zajętość pamięci [kB]	246 (2)	-	56 (2)	128 (2), 18 (5)	128 (2)

4. Warstwa sprzętowa projektu

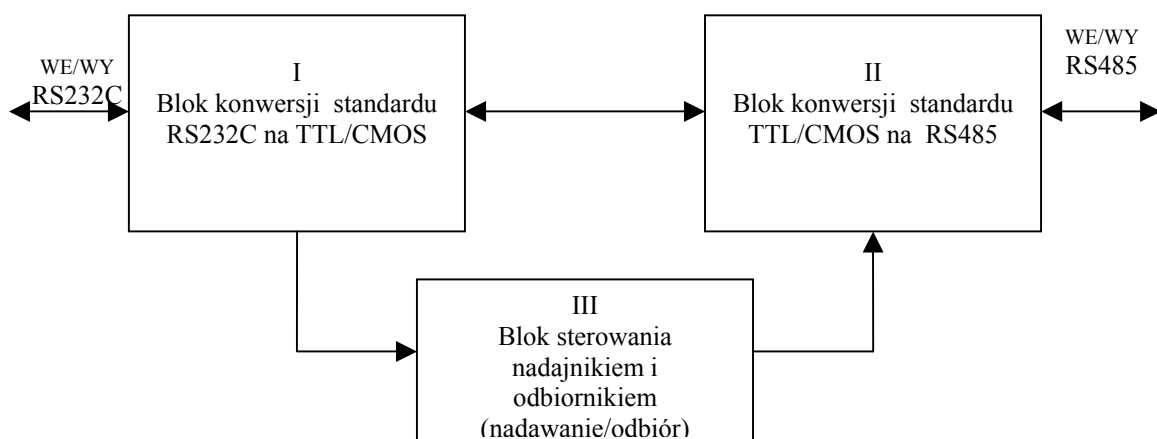
W skład warstwy sprzętowej projektu wchodzi: moduł urządzenia Slave zrealizowany na mikrokontrolerze oraz konwertery RS232-RS485 w wersji z optoizolacją i bez optoizolacji.

4.1 Konwerter RS232-RS485

Konwerter RS232-RS485 jest urządzeniem służącym do zamiany standardu komunikacyjnego RS232 na RS485 i odwrotnie.

Konwertery znajdują zastosowanie w przemyśle, w systemach transmisji i akwizycji danych - umożliwiają podłączenie urządzeń przemysłowych wyposażonych w interfejs RS-232 do sieci (magistrali) interfejsu RS485 według standardu RS-485. Istnieje wielu producentów konwerterów różniących się przede wszystkim jakością i przeznaczeniem. Jako przykład można podać tu firmę Advantech, produkującą zaawansowane konwertery rodziny ADAM. Jeden z modeli tych urządzeń znajduje się w laboratorium Katedry Metrologii i Systemów Elektronicznych na Wydziale ETI Politechniki Gdańskiej.

Poniższy schemat blokowy przedstawia najważniejsze elementy konwertera RS232 - RS485.



Rys. 9 Schemat blokowy konwertera RS232-RS485.

Zasadniczymi elementami konwertera są bloki (I i II) zamiany poziomów logicznych napięć dostosowujące konwerter do interfejsów szeregowych RS232 i RS485. Blok I służy do translacji poziomów logicznych napięć interfejsu RS232(-3...-15V ÷ +3...+15V) na poziomy logiczne układów TTL/CMOS (0÷5V). Natomiast blok II służy do translacji poziomów logicznych układów TTL/CMOS na poziomy logiczne standardu RS485. W interfejsie RS485 występuje transmisja w trybie half-duplex i nie ma fizycznego arbitrażu, dlatego w celu uniknięcia kolizji (jednoczesnego nadawania więcej niż jednego nadajnika) zajmuje się nim protokół. Dlatego konieczne było uwzględnienie bloku (III) sterowania nadajnikiem i odbiornikiem umożliwiającego zastosowanie metody arbitrażu.

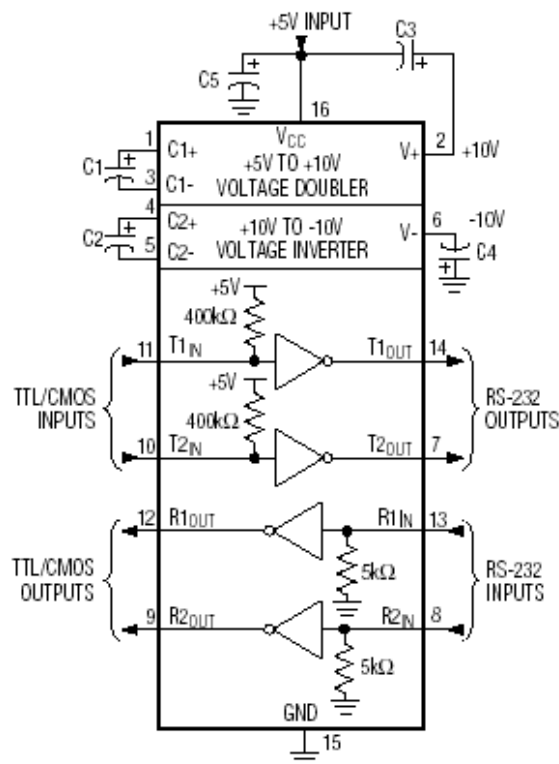
Większość konwerterów umożliwia wybór sposobu sterowania kierunkiem transmisji np:

- za pomocą sygnału RTS portu szeregowego (RTS aktywne = nadawanie);
- automatycznie, przy pomocy detekcji stanu sygnału na linii nadawczej TxD portu szeregowego;
- nietypowo, w zależności od rodzaju konwertera.

Ponadto spora grupa konwerterów pozwala także na dołączenie do linii RS-485 rezystora dopasowującego – terminatora.

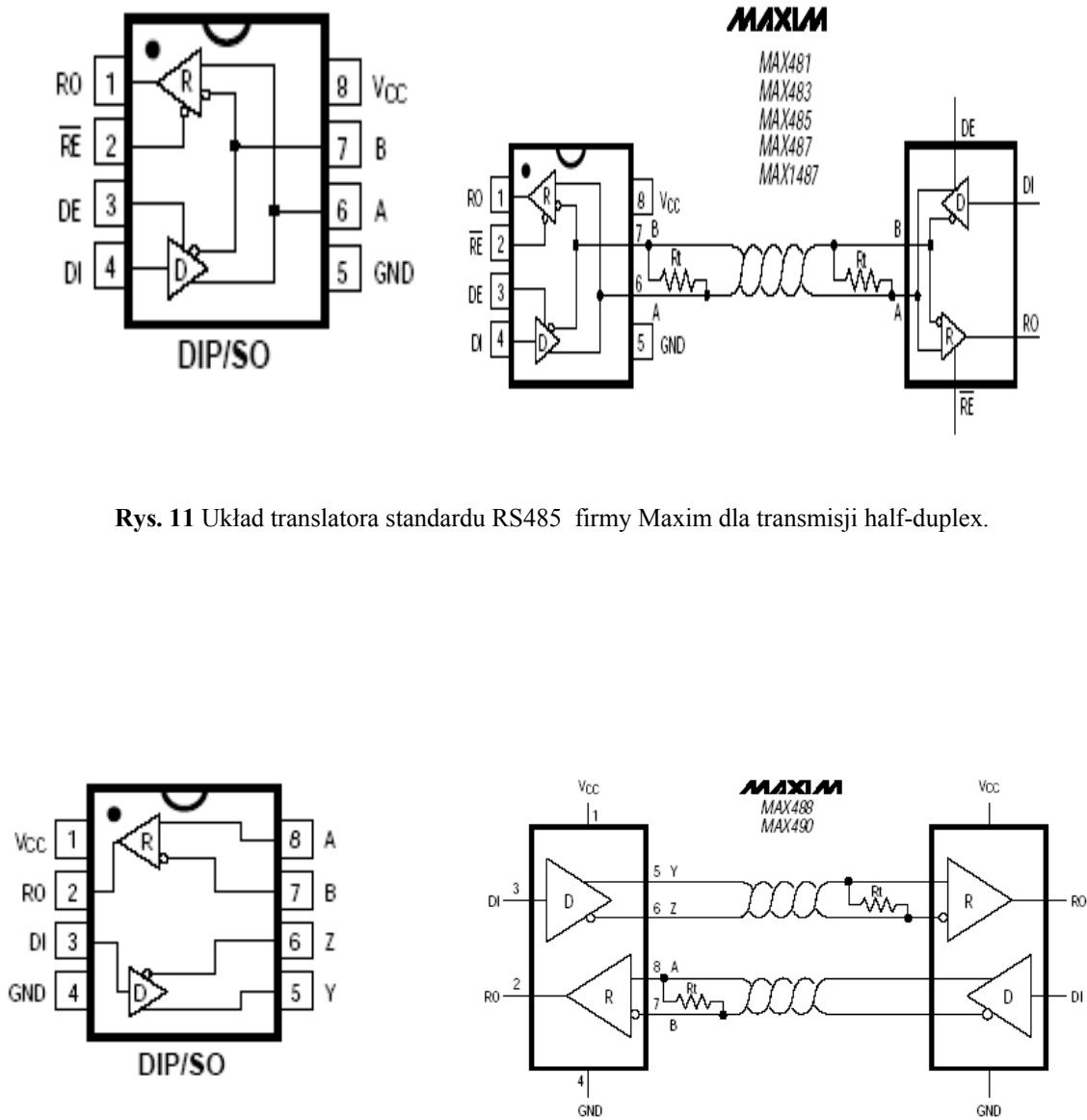
W standardzie RS 485 tylko jeden nadajnik podłączony do linii może w danym momencie nadawać, więc aby nie było kolizji, regulacją dostępu do łącza (arbitrażem) zajmują się protokoły, dlatego do nich należy obowiązek sterowania konwerterami. W trybie half-duplex jedna linia transmisyjna (para przewodów) wykorzystywana jest na przemian do transmisji w obu kierunkach. W czasie, gdy nie ma transmisji w żadnym kierunku konwerter znajduje się w stanie odbioru. W przypadku sterowania konwertera za pomocą linii RTS interfejsu RS232, włączenie linii na moment przed wysłaniem danych i wyłączenie dopiero po wysłaniu ostatniego znaku z odpowiednio dobranym opóźnieniem czasowym jest sterowane za pomocą oprogramowania (protokołu komunikacyjnego). Przełączenie konwertera do stanu nadawania w przypadku automatycznego sterowania konwerterem występuje w momencie pojawienia się znaku na linii TxD (dane nadawane) interfejsu RS232. Po wysłaniu znaku konwerter pozostaje jeszcze pewien czas w stanie nadawania. Czas wydłużenia stanu nadawania po wysłaniu znaku jest określany zależnie od prędkości transmisji i ustawiany przez np. przełącznik lub zworę w konwerterze.

Najważniejszymi elementami konwertera są translatory poziomów logicznych napięć standardu RS232 i RS485. Najbardziej znanym i najczęściej stosowanym układem translatora poziomu napięć dla standardu RS232 jest MAX232 firmy Maxim. Istnieją także inne zamienniki np. ICL232 itp. Podobnie jest z układami dla standardu RS485, tu również produkuje firma Maxim i istnieją również zamienniki innych firm. Jednakże w tym przypadku mamy do czynienia z szeroką gamą układów różniących się budową wewnętrzną, parametrami, trybem transmisji, ilością wyprowadzeń itp. Na przykład układy samej tylko firmy Maxim z których wymienić można: MAX485, MAX490, MAX491, MAX1480A/B, MAX1490A/B itp., różnią się między sobą sposobem nadawania (half-duplex lub full-duplex), możliwością sterowania nadajnikiem i odbiornikiem, wbudowaną optoizolacją itp. Poniżej przedstawione są przykładowe układy translatorów standardu RS232 i RS485 firmy Maxim.



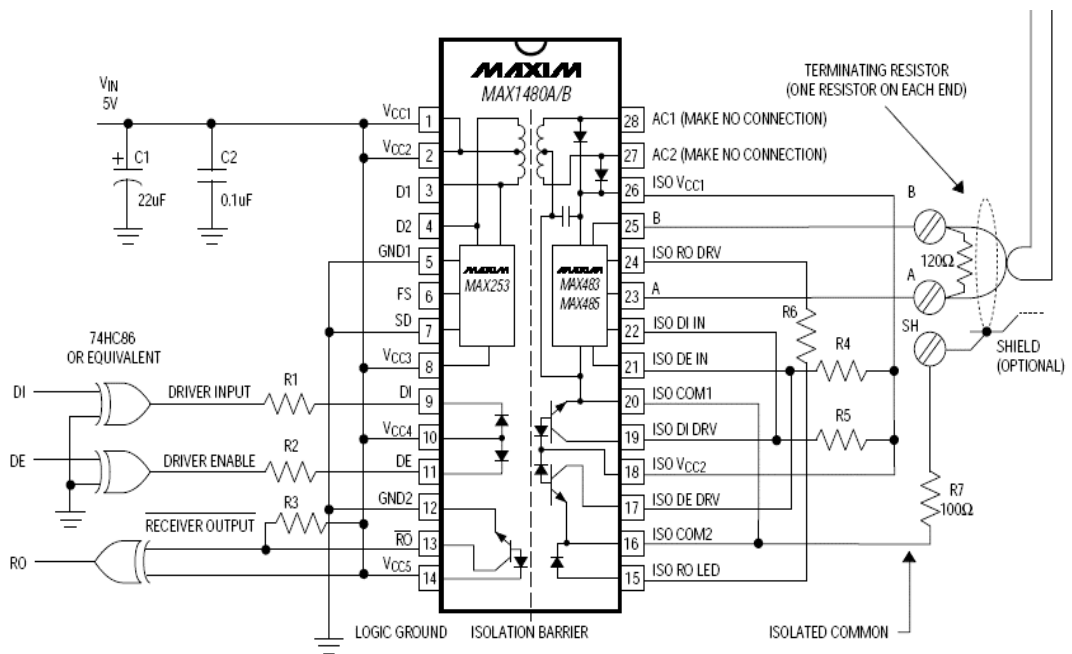
Rys. 10 MAX232 – układ translatora standardu RS232, zamieniający poziom logiczny napięcia interfejsu RS232 na poziom logiczny napięć układów TTL/CMOS.

Układy translacji poziomów logicznych napięć TTL do poziomów logicznych napięć interfejsu RS485

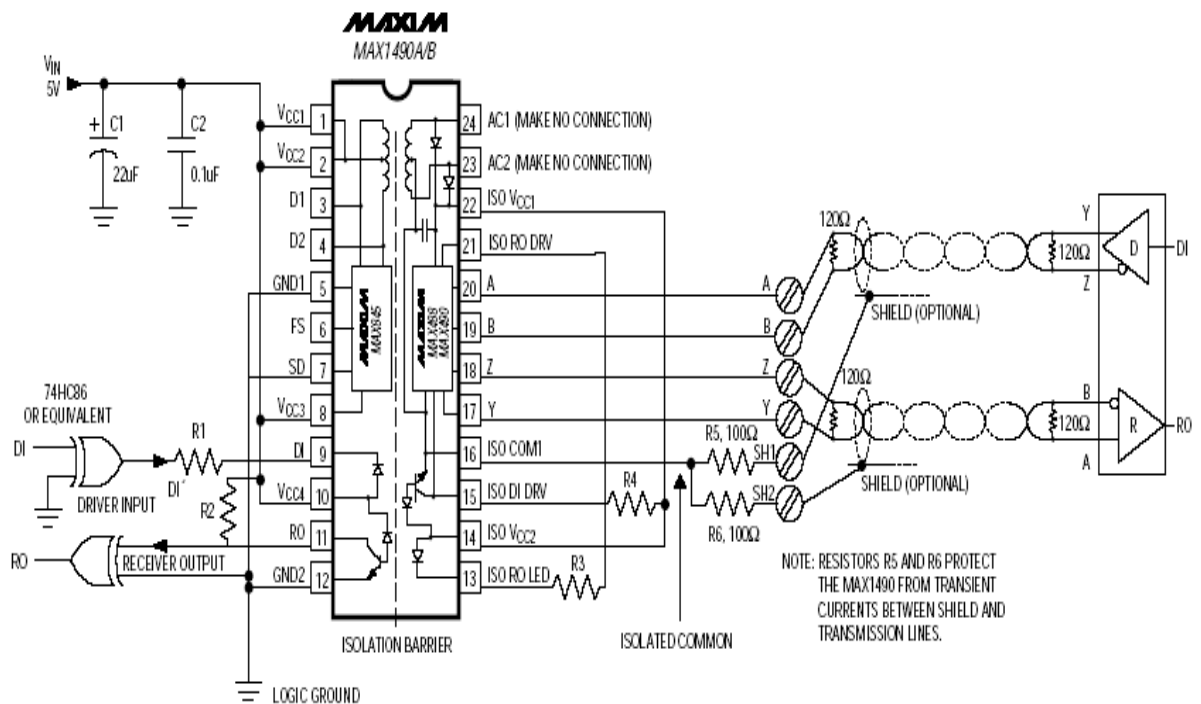


Rys. 11 Układ translatora standardu RS485 firmy Maxim dla transmisji half-duplex.

Rys. 12 Układ translatora standardu RS485 firmy Maxim dla transmisji full-duplex.



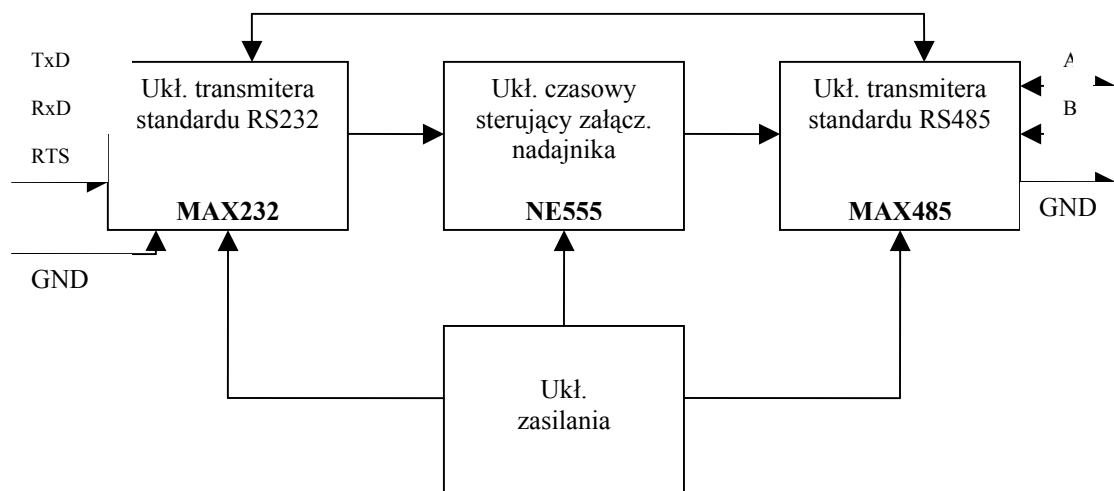
Rys. 13 Układ translatora standardu RS485 z optyzacją firmy Maxim dla transmisji half-duplex.



Rys. 14 Układ translatora standardu RS485 z optyzacją firmy Maxim dla transmisji full-duplex.

4.1.1 Konwerter bez optoizolacji

Konwerter RS232-RS485 w wersji bez optoizolacji jest typowym konwerterem RS232-RS485 nie posiadającym separacji galwanicznej pomiędzy obwodami interfejsów RS232 i RS485. W takim przypadku potencjały „zera” po stronie RS232 i RS485 są jednakowe. Schemat blokowy omawianego konwertera jest przedstawiony na rysunku 15,



Rys. 15 Schemat blokowy konwertera RS232-RS485 w wersji bez optoizolacji.

Jako element nadawczo-odbiorczy(transmitter) po stronie interfejsu RS232 zastosowano układ MAX232, natomiast po stronie RS485 - MAX485. Zaprojektowany konwerter umożliwia pracę w trybie transmisji pół - duplex oraz sterowanie załączaniem nadajnika i wyłączaniem odbiornika. Wybieranie sposobu sterowania konwerterem pomiędzy sterowaniem automatycznym a sterowaniem za pomocą linii RTS oraz dołączanie do linii danych A i B interfejsu RS485 terminatora-rezystora dopasowującego, jest dokonywane za pomocą przełączników.

Przy sterowaniu automatycznym należy, zależnie od prędkości transmisji, regulować opóźnienie, tak by wyłączyć nadajnik dopiero po wysłaniu ramki, co jest uzyskiwane za pomocą układu czasowego, którego opóźnienie jest konfigurowane za pomocą odpowiednio dobranych elementów RC ustawianych za pomocą przełączników (dip switch). Często zamiast przełączników stosuje się zworki (mostki).

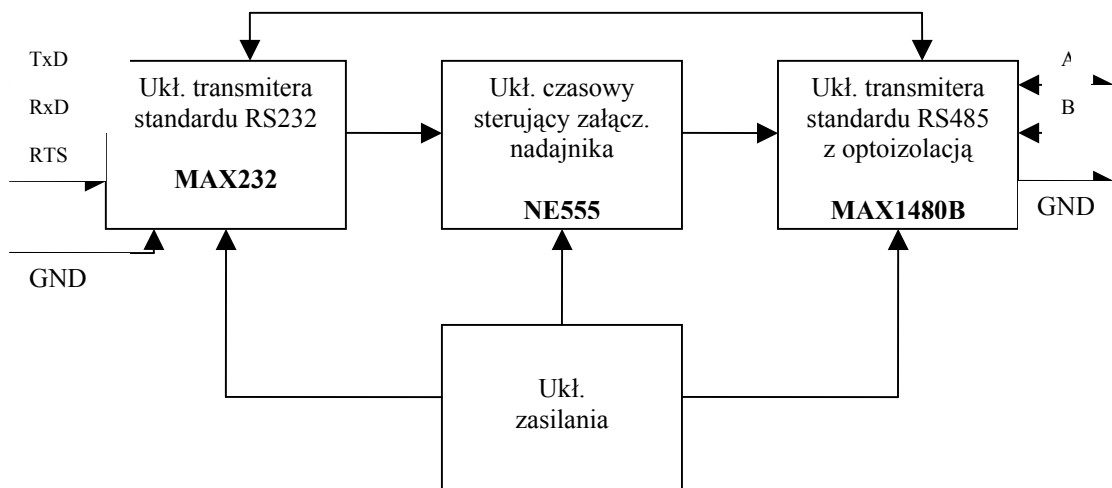
Stany sygnałów na liniach nadawczej i odbiorczej sygnalizowane są diodami elektroluminescencyjnymi. Jako gniazdo interfejsu RS232 zastosowano DB9 przeznaczone dla kabla standardu RS232 typu null-modem(niekrzyżowanego), natomiast dla interfejsu RS485 śrubowe złącze AK/3.

Zasilanie konwertera odbywa się przez podanie napięcia zmiennego lub stałego o wartości 9-12V na zaciski złącza śrubowego (złącze AK/2) lub przez linię DTR interfejsu RS232 dołączaną za pomocą zworki.

4.1.2 Konwerter z optoizolacją

W przypadku konwertera RS232-RS485 w wersji z optoizolacją, jego obwody interfejsu RS485 są elektrycznie odizolowane od obwodów RS232 – jest to bariera galwaniczna od strony urządzenia z interfejsem RS232 i każdego podłączonego urządzenia. Pozwala to uniknąć ryzyka wystąpienia różnic pomiędzy potencjałami „zera” łączonych urządzeń. Różnice takie występują w większości urządzeń np. komputerów zasilanych z różnych faz. W sieci RS485 łączone urządzenia są dość odległe od siebie, więc potencjały zera są zwykle różne.

Schemat blokowy konwertera optoizolowanego przedstawia rysunek 16.



Rys. 16 Schemat blokowy konwertera RS232-RS485 w wersji z optoizolacją.

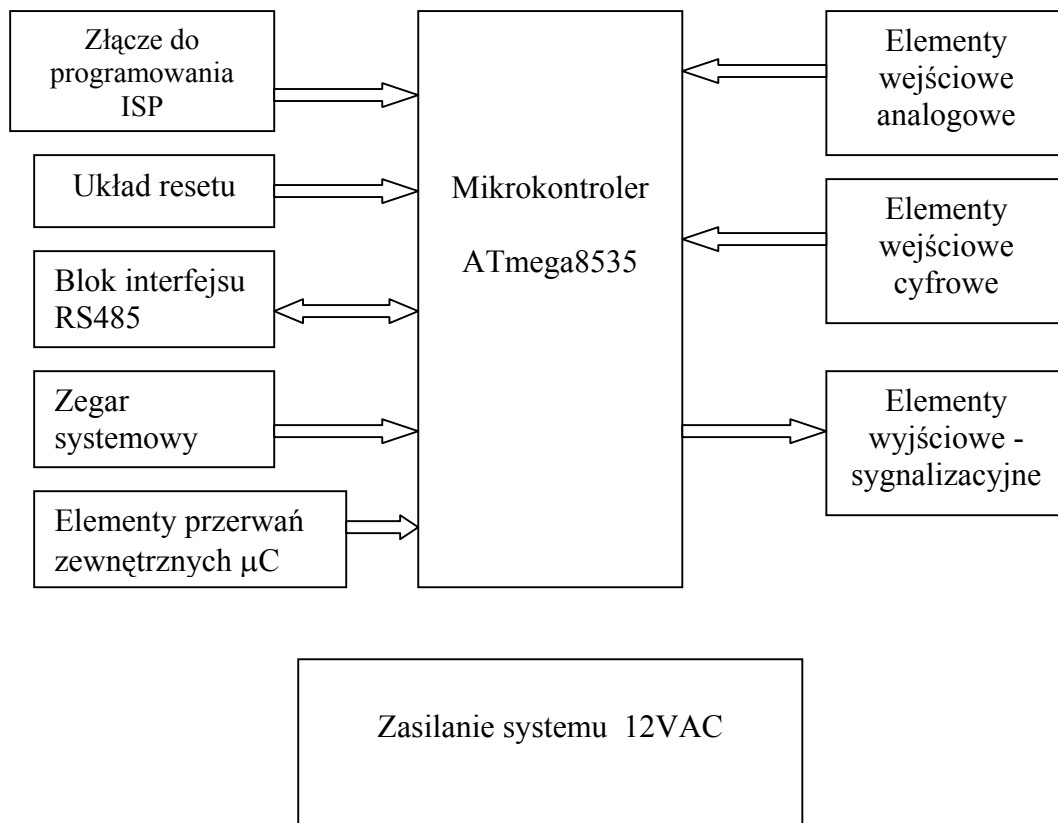
Projekt konwertera RS232-RS485 z optoizolacją, różni się od swojego poprzednika nieoptoizolowanego jedynie układem transmitera interfejsu RS485, wraz z dołączonymi do niego zewnętrznymi elementami niezbędnymi do prawidłowej pracy, takimi jak: bramki Ex-OR, rezystory i kondensatory.

Jest nim układ nadawczo-odbiorczy MAX1480B, który w odróżnieniu od układu MAX485, zastosowanego w konwerterze bez optoizolacji, posiada izolację optyczną, dzięki wbudowanym dwóm transoptorom i przetwornicy separującej zasilanie. Sposób dołączenia elementów zewnętrznych do omawianego układu oraz ich wartości zostały przedstawione w rozdziale 4.1 na rysunku 13, który został pobrany z noty aplikacyjnej tego układu. Układ umożliwia również sterowanie nadajnikiem i jest przeznaczony dla trybu transmisji half-duplex. Metody sterowania nadajnikiem są analogiczne jak w poprzednim konwerterze bez optoizolacji. Wykorzystano również te same elementy elektroniczne, typy złącz, przełączniki, układ zasilania itp.

4.2 Moduł Modbus Slave zrealizowany na mikrokontrolerze

Moduł Modbus Slave został zrealizowany na mikrokontrolerze rodziny AVR - ATmega 8535.

Na rysunku 17 przedstawiono schemat blokowy zrealizowanego modułu Modbus Slave.



Rys. 17 Schemat blokowy modułu Modbus Slave.

4.2.6 Elementy wyjściowe-sygnalizacyjne

W systemie, jako elementy wyjściowe-sygnalizacyjne wykorzystano diody LED, służące do wskazywania stanów logicznych linii wyjść cyfrowych mikrokontrolera.

4.2.7 Elementy wejściowe cyfrowe

Do budowy elementów wejściowych cyfrowych użyto dwóch przełączników dwupozycyjnych (dip switch) konfigurowanych przez użytkownika. Stany logiczne na tych przełącznikach mogą być odczytywane przez kontroler i umieszczane w jego pamięci.

5. Oprogramowanie

5.1 Oprogramowanie Modbus Master na komputerze PC

Modbus Master jest programem (sterownikiem) napisanym w języku C w środowisku LabWindows CVI i łączy w sobie funkcje urządzenia nadrzędnego (Master) sterującego urządzeniami podrzędnymi (Slaves) oraz analizatora protokołu Modbus. Jest wyposażony w podstawowe funkcje i rejestry danych protokołu Modbus. Umożliwia podgląd wysyłanych i odbieranych wiadomości, wizualizację odbieranych danych oraz zapis do pliku. Jako typowy protokół komunikacyjny działający na platformie interfejsu szeregowego pozwala na konfigurowanie parametrów transmisji szeregowej. Pozwala również na ustawienie adresu urządzenia Slave, granicznego czasu (Timeout) odpowiedzi tego urządzenia a także na wybór liczby prób ponownej transmisji, w przypadku braku odpowiedzi. Poniżej zostaną omówione poszczególne elementy tej aplikacji.

Po uruchomieniu aplikacji pojawia się okno powitalne.



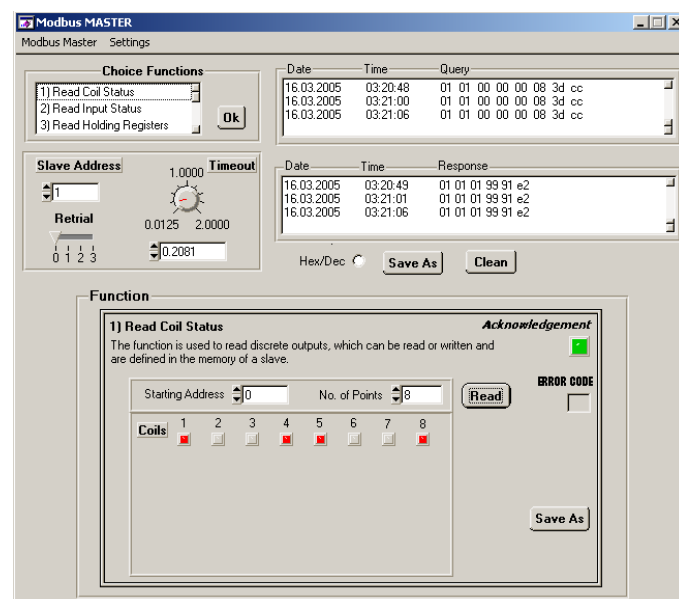
Rys. 23 Okno powitalne programu Modbus Master.

Aby rozpocząć działanie programu należy wybrać **COM** z menu **Settings** i po pojawieniu się okna z możliwymi ustawieniami portu szeregowego dokonać wyboru. Możliwymi konfigurowanymi parametrami portu są: numer portu, prędkość transmisji, kontrola parzystości, czas oczekiwania na odpowiedź (Timeout) oraz możliwość sterowania linią RTS wykorzystywaną do sterowania konwertera RS232-RS485, co przedstawia rysunek 24.



Rys. 24 Okno wyboru parametrów transmisji szeregowej.

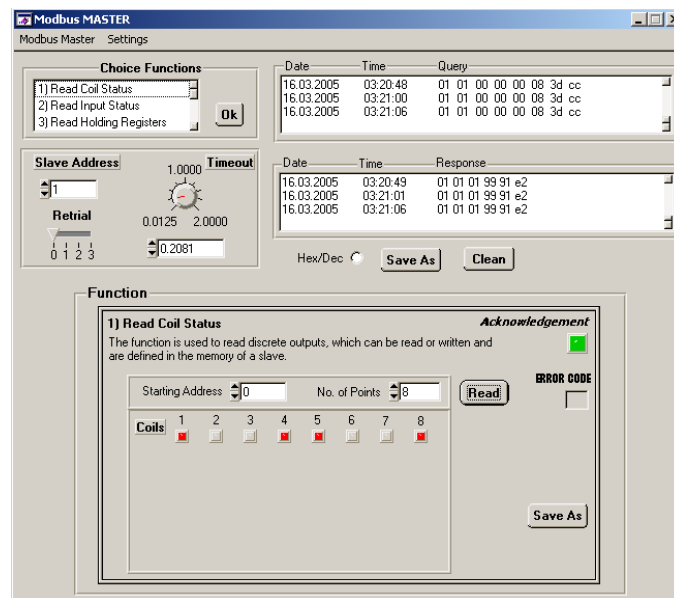
Następnie po zaakceptowaniu wyboru powinien ukazać się główny panel sterowania Modbus Master, o ile nie wystąpił błąd przy otwieraniu portu COM.



Rys. 25 Panel główny Modbus Master.

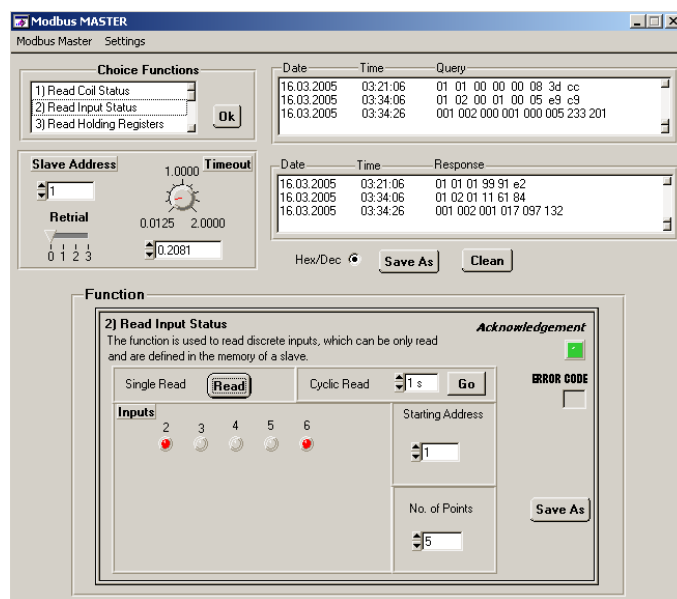
Jest to panel główny aplikacji. Użytkownik ma tu możliwość wybrania adresu wywoływanego urządzenia Slave, ustawienia dla niego granicznego czasu odpowiedzi (Timeout), ilość prób ponownego nawiązania połączenia w przypadku braku odpowiedzi, wyboru jednej z dziewięciu standardowych funkcji protokołu Modbus, które są konfigurowane w części **Function**. Istnieje również możliwość podglądania wysyłanych i odbieranych komunikatów oraz wybór formatu w jakim są wyświetlane komunikaty: dziesiętnego lub szesnastkowego. Można również dokonać zapisu na dysk zarówno komunikatów z okienek zapytania i odpowiedzi jak i wartości kontrolki i danych poszczególnych funkcji przy pomocy przycisku **Save As**.

Poniżej przedstawione są poszczególne funkcje protokołu Modbus oraz ich obsługa.



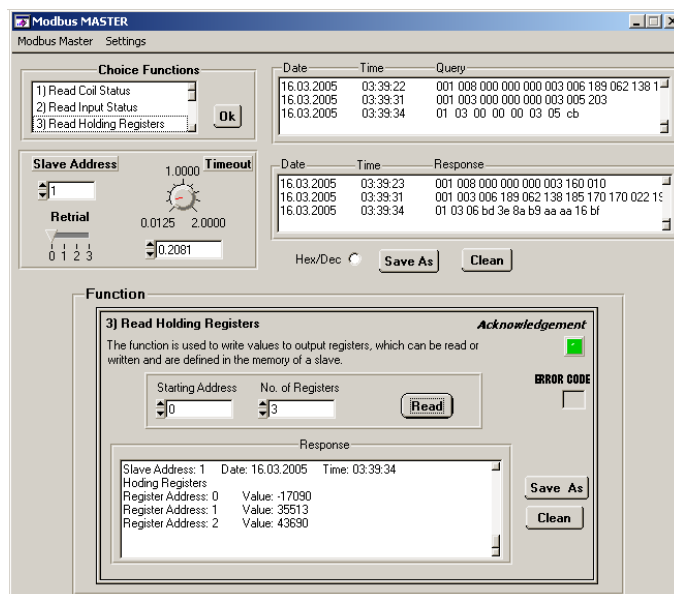
Rys. 26 Funkcja nr 1 Modbus Master.

Funkcja nr 1 pozwala na odczytanie stanu wyjść cyfrowych (max 32 wyjścia) jednostki Slave. W przypadku pozytywnej odpowiedzi zapala się zielona dioda, natomiast przy negatywnej w kontrolce **ERROR CODE** pojawia się kod błędu odpowiedzi szczególnej.



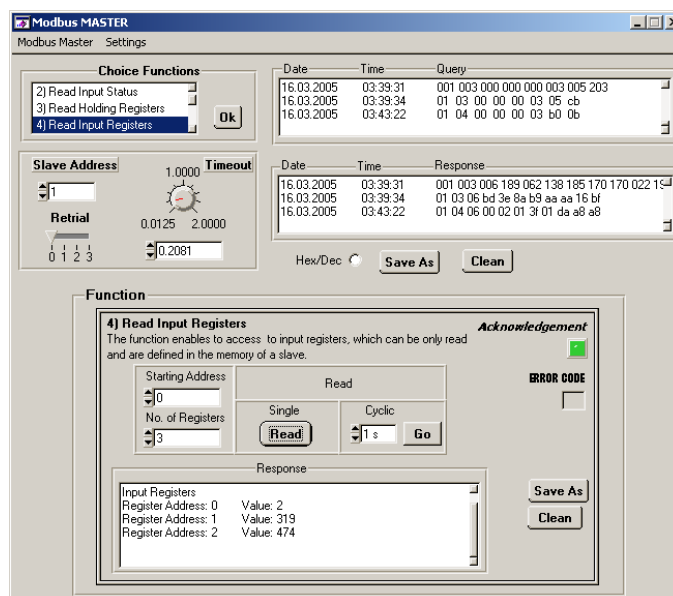
Rys. 27 Funkcja nr 2 Modbus Master.

Funkcja nr 2 pozwala na odczytanie wejść cyfrowych jednostki Slave. Posiada możliwość cyklicznego wysyłania zapytania(przycisk **Go**).



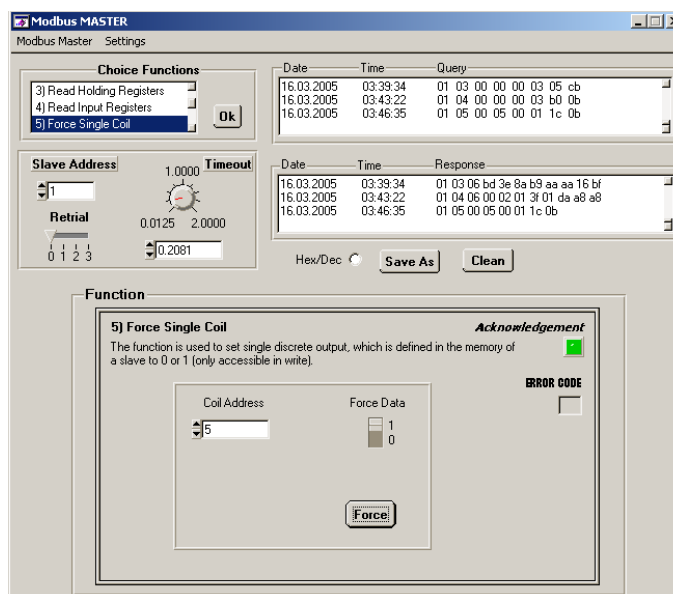
Rys. 28 Funkcja nr 3 Modbus Master.

Funkcja nr 3 pozwala na odczyt rejestrów wyjściowych z jednostki Slave.



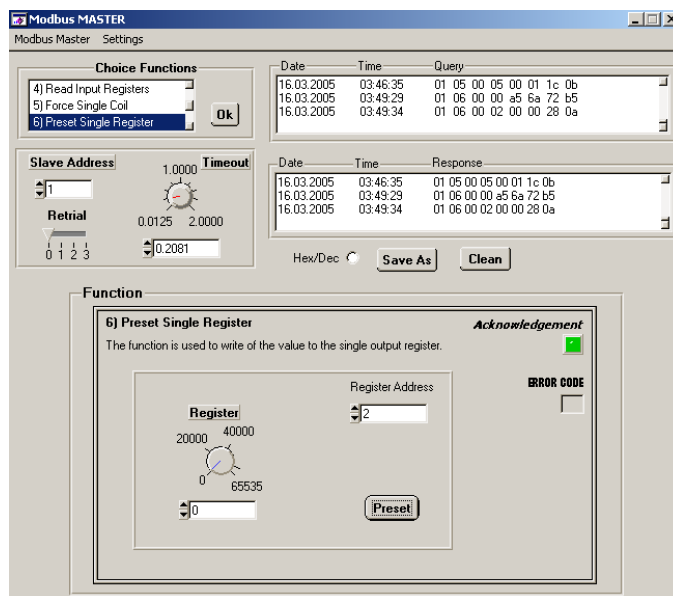
Rys. 29 Funkcja nr 4 Modbus Master.

Funkcja nr 4 pozwala na odczyt rejestrów wejściowych z jednostki Slave i ma możliwość odpytywania cyklicznego.



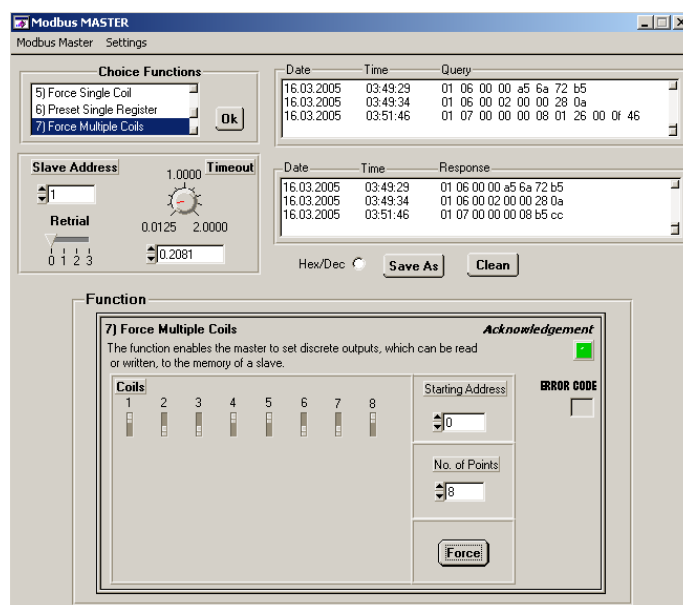
Rys. 30 Funkcja nr 5 Modbus Master.

Funkcja nr 5 pozwala na ustawienie pojedynczego wyjścia cyfrowego jednostki Slave.



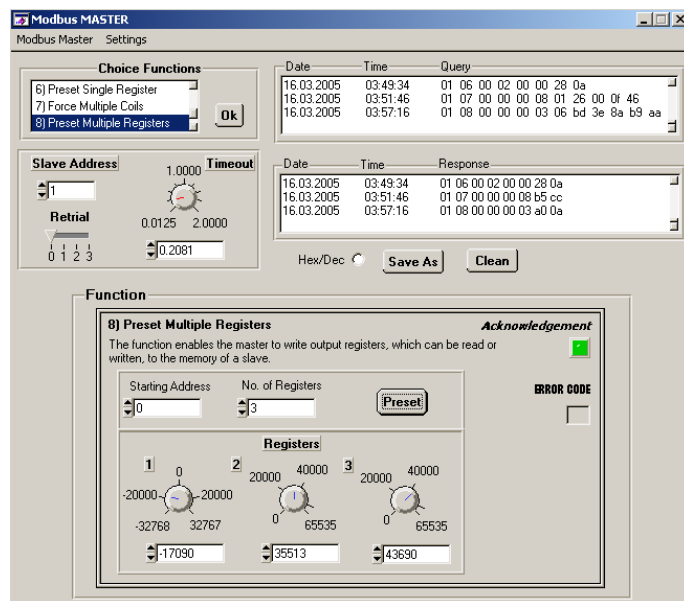
Rys. 31 Funkcja nr 6 Modbus Master.

Funkcja nr 6 pozwala na ustawienie pojedynczego rejestru wyjściowego.



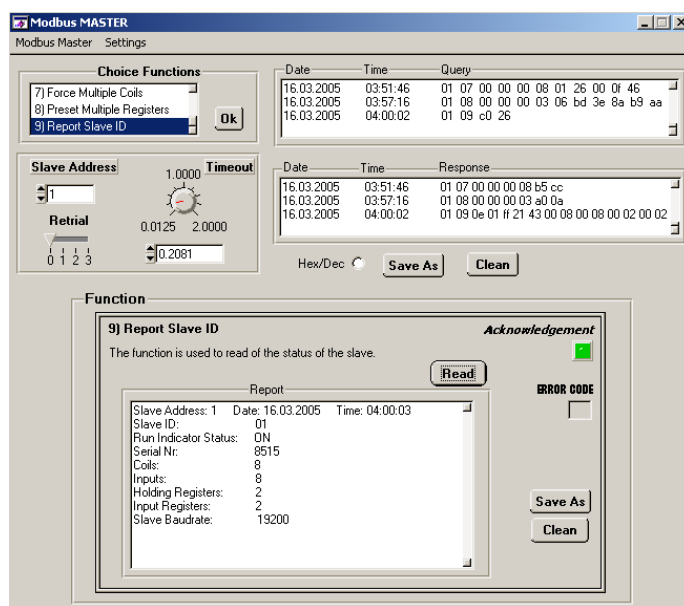
Rys. 32 Funkcja nr 7 Modbus Master.

Funkcja nr 7 pozwala na ustawienie stanu wielu wyjść cyfrowych.



Rys. 33 Funkcja nr 8 Modbus Master.

Funkcja nr 8 pozwala na ustawienie wielu rejestrów wyjściowych jednostki Slave.



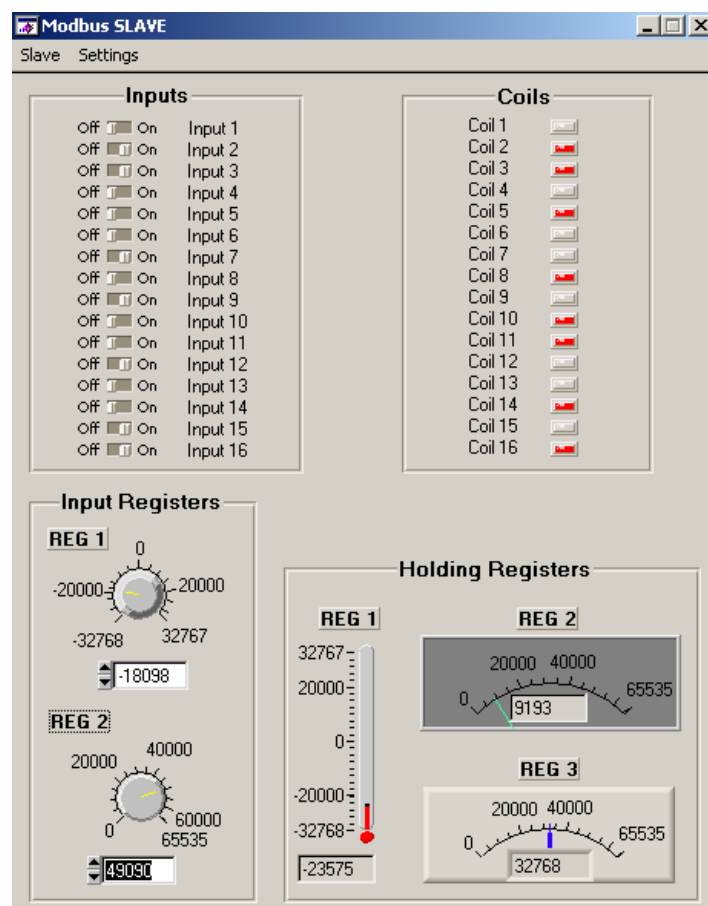
Rys. 34 Funkcja nr 9 Modbus Master.

Funkcja nr 9 umożliwia odczytanie nr ID urządzenia Slave oraz dodatkowych informacji jak np. numer seryjny, liczbę wejść cyfrowych, liczbę wyjść cyfrowych, liczbę rejestrów wejściowych i wyjściowych.

5.3 Oprogramowanie Modbus Slave na komputerze PC

Modbus Slave napisany w Środowisku LabWindows CVI i jest wirtualnym urządzeniem symulującym węzeł podrzędny wykonujący zdalne polecenia urządzenia nadrzędnego (Master). Program pozwala użytkownikowi na konfigurowanie parametrów transmisji szeregowej oraz stanów wejść cyfrowych i rejestrów wejściowych odczytywanych przez Master za pomocą funkcji protokołu Modbus. Stany wyjść cyfrowych i rejestrów wyjściowych, ustawianych przez Master mogą być jedynie odczytywane.

Główny panel wirtualnego urządzenia Modbus Slave wygląda następująco:



Rys. 35 Panel programu Modbus Slave.

SW1 – przycisk resetu mikrokontrolera;

SW2 – przycisk wyzwalania przerwania zewnętrznego INT0 mikrokontrolera, służący do zatwierdzenia wyboru adresu urządzenia Slave;

SW3 – przycisk wyzwalania przerwania zewnętrznego INT0 mikrokontrolera, służący do zatwierdzenia wyboru prędkości transmisji układu UART;

DIP_SW1 – blok 4 przełączników dwupozycyjnych, służący do konfiguracji adresu i prędkości portu szeregowego urządzenia:

Adres - ustawiany w kodzie dwójkowym (zakres adresów: od 0 do 15), np.:

0-wszystkie przełączniki w stanie off,

3-tylko przełączniki 1 i 2 w stanie on,

15-wszystkie przełączniki w stanie on;

Baudrate – do wyboru pięć prędkości transmisji:

2400-wszystkie przełączniki w stanie off,

4800-tylko pierwszy przełącznik w stanie on,

9600-tylko drugi przełącznik w stanie on,

19200-tylko trzeci przełącznik w stanie on,

38400-tylko czwarty przełącznik w stanie on;

DIP_SW2 - blok 8 przełączników dwupozycyjnych, służący do ustawiania stanów na cyfrowych liniach we/wy mikrokontrolera;

Rp – potencjometr regulujący napięcie mierzone przez przetwornik A/C;

J1 – zwora – zwarcie powoduje dołączanie do linii danych interfejsu RS485 rezystora dopasowującego R_t (terminatora);

J2 – zwora - przeznaczona do dołączania zewnętrznego napięcia referencyjnego przetwornika A/C, jakim jest napięcie zasilania po przejściu przez filtr LC.

Elementy sygnalizacyjne:

LED1 – dioda (czerwona) wskazująca obecność napięcia zasilania;

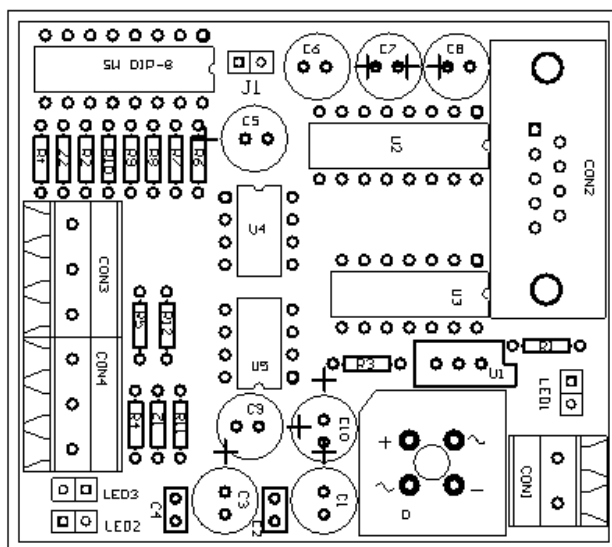
LED2...LED9 – diody (zielone) wskazujące stany na liniach we/wy mikrokontrolera;

LED10 – dioda (zielona) sygnalizująca proces programowania mikrokontrolera przez ISP;

LED11 – dioda (żółta) wskazująca stan wysyłania danych przez mikrokontroler;

LED12 – dioda (zielona) wskazująca pojawienie się danych odbieranych przez kontroler.

6.2 Konwerter RS232-RS485 w wersji bez optoizolacji



Rys. 37 Widok konwertera z góry.

Podłączenie konwertera:

CON1 – złącze typu AK500/2 do podłączania napięcia zasilającego 12V zmiennego lub stałego;

CON2 – złącze typu DB9/F do podłączania konwertera wg standardu RS232;

CON3 i CON4 – złącza typu AK500/3 do podłączania konwertera wg standardu RS485;

Konfiguracja pracy:

SW DIP8 – blok 8 przełączników dwupozycyjnych, za pomocą którego można wybrać sposób sterowania konwerterem, dołączać terminator oraz ustalać opóźnienie czasowe wyłączenia nadajnika RS485:

Opis przełączników:

- 1 - powoduje dołączenie terminatora do linii A i B interfejsu RS485,
- 2 - wybór sterowania konwertera za pomocą linii RTS,
- 3 - wybór automatycznego sterowania konwertera,
- 4...8 - ustalanie opóźnienia zależnego od prędkości transmisji
(38400,19200,9600,4800,2400);

Uwaga!

W przypadku automatycznego sterowania konwertera należy wybrać przełącznik nr 3 oraz jeden z przełączników 4...8 zależnie od prędkości transmisji.

J1 – zwora – zwarcie powoduje dołączenie linii DTR interfejsu RS232, jako drugiego sposobu zasilania konwertera ;

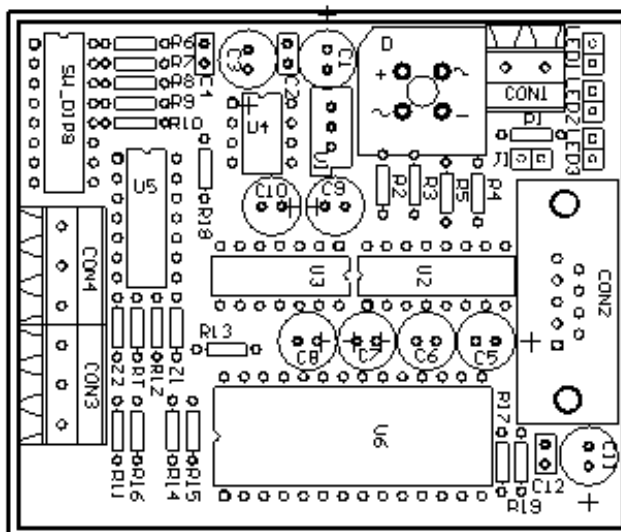
Sygnalizacja optyczna:

LED1 – dioda (czerwona) wskazująca obecność napięcia zasilania;

LED2 – dioda (żółta) wskazująca stan wysyłania danych po stronie łącza RS232;

LED3 – dioda (zielona) wskazująca pojawienie się danych odbieranych po stronie łącza RS232.

6.3 Konwerter RS232-RS485 w wersji z optoizolacją



Rys. 38 Widok konwertera z góry.

Podłączenie konwertera:

CON1 – złącze typu AK500/2 do podłączania napięcia zasilającego 12V zmiennego lub stałego;

CON2 – złącze typu DB9/F do podłączania konwertera wg standardu RS232;

CON3 i CON4 – złącza typu AK500/3 do podłączania konwertera wg standardu RS485;

Konfiguracja pracy:

SW DIP8 – blok 8 przełączników dwupozycyjnych, za pomocą którego można wybrać sposób sterowania konwerterem, dołączać terminator oraz ustalać opóźnienie czasowe wyłączenia nadajnika RS485:

Opis przełączników:

- 8 - powoduje dołączenie terminatora do linii A i B interfejsu RS485,
- 7 - wybór sterowania konwertera za pomocą linii RTS,
- 6 - wybór automatycznego sterowania konwertera,
- 1...5 - ustalanie opóźnienia zależnego od prędkości transmisji
(2400,4800,9600,19200,38400);

Uwaga!

W przypadku automatycznego sterowania konwertera należy wybrać przełącznik nr 6 oraz jeden z przełączników 1...5 zależnie od prędkości transmisji.

J1 – zwora – zwarcie powoduje dołączenie linii DTR interfejsu RS232, jako drugiego sposobu zasilania konwertera ;

Sygnalizacja optyczna:

LED1 – dioda (czerwona) wskazująca obecność napięcia zasilania;

LED2 – dioda (zielona) wskazująca pojawienie się danych odbieranych po stronie łącza RS232;

LED3 – dioda (żółta) wskazująca stan wysyłania danych po stronie łącza RS232.

Dodatek

Przedstawiony został fragment kodu źródłowego: obliczanie sumy kontrolnej CRC.

Obliczanie CRC

```
unsigned char *puchMsg ; /* message to calculate CRC upon */
unsigned short usDataLen ; /* quantity of bytes in message */

unsigned short ObliczCRC16(unsigned char *puchMsg, unsigned short usDataLen)
{
    unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
    unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
    unsigned uIndex ; /* will index into CRC lookup table */
    while (usDataLen --) /* pass through message buffer */
    {
        uIndex = uchCRCHi ^ *puchMsg++ ; /* calculate the CRC */
        uchCRCHi = uchCRCLo ^ uchCRCHi[uIndex] ;
        uchCRCLo = uchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

tablice uchCRCHi oraz uchCRCLo dodaje się do kodu źródłowego.

```
//-----
// tablice do obliczania CRC16
unsigned char uchCRCHi[] = {
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,
0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,
0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x01,0xC0,
0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,
0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,
0x40
} ;

/* Table of CRC values for low-order byte */
unsigned char uchCRCLo[] = {
0x00,0xC0,0xC1,0x01,0xC3,0x03,0x02,0xC2,0xC6,0x06,0x07,0xC7,0x05,0xC5,0xC4,
0x04,0xCC,0x0C,0x0D,0xCD,0x0F,0xCF,0xCE,0x0E,0x0A,0xCA,0xCB,0x0B,0xC9,0x09,
0x08,0xC8,0xD8,0x18,0x19,0xD9,0x1B,0xDB,0xDA,0x1A,0x1E,0xDE,0xDF,0x1F,0xDD,
0x1D,0x1C,0xDC,0x14,0x14,0xD4,0xD5,0x15,0xD7,0x17,0x16,0xD6,0xD2,0x12,0x13,0xD3,
0x11,0xD1,0xD0,0x10,0xF0,0x30,0x31,0xF1,0x33,0xF3,0xF2,0x32,0x36,0xF6,0xF7,
0x37,0xF5,0x35,0x34,0xF4,0x3C,0xFC,0xFD,0x3D,0xFF,0x3F,0x3E,0xFE,0xFA,0x3A,
0x3B,0xFB,0x39,0xF9,0xF8,0x38,0x38,0xE8,0xE9,0x29,0xEB,0x2B,0x2A,0xEA,0xEE,
0x2E,0x2F,0xEF,0x2D,0xED,0xEC,0x2C,0xE4,0x24,0x25,0xE5,0x27,0xE7,0xE6,0x26,
0x22,0xE2,0xE3,0x23,0xE1,0x21,0x20,0xE0,0xA0,0x60,0x61,0xA1,0x63,0xA3,0xA2,
0x62,0x66,0xA6,0xA7,0x67,0xA5,0x65,0x64,0xA4,0x6C,0xAC,0xAD,0x6D,0xAF,0x6F,
0x6E,0xAE,0xAA,0x6A,0x6B,0xAB,0x69,0xA9,0xA8,0x68,0x78,0xB8,0xB9,0x79,0xBB,
0x7B,0x7A,0xBA,0xBE,0x7E,0x7F,0xBF,0x7D,0xBD,0xBC,0x7C,0xB4,0x74,0x75,0xB5,
0x77,0xB7,0xB6,0x76,0x72,0xB2,0xB3,0x73,0xB1,0x71,0x70,0xB0,0x50,0x90,0x91,
0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,0x9C,0x5C,
0x5D,0x9D,0x5F,0x9F,0x9E,0x5E,0x5A,0x9A,0x9B,0x5B,0x99,0x59,0x58,0x98,0x88,
0x48,0x49,0x89,0x4B,0x8B,0x8A,0x4A,0x4E,0x8E,0x8F,0x4F,0x8D,0x4D,0x4C,0x8C,
0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,
0x40
} ;
```