Praktyczne informacjie o protokole MODBUS RTU

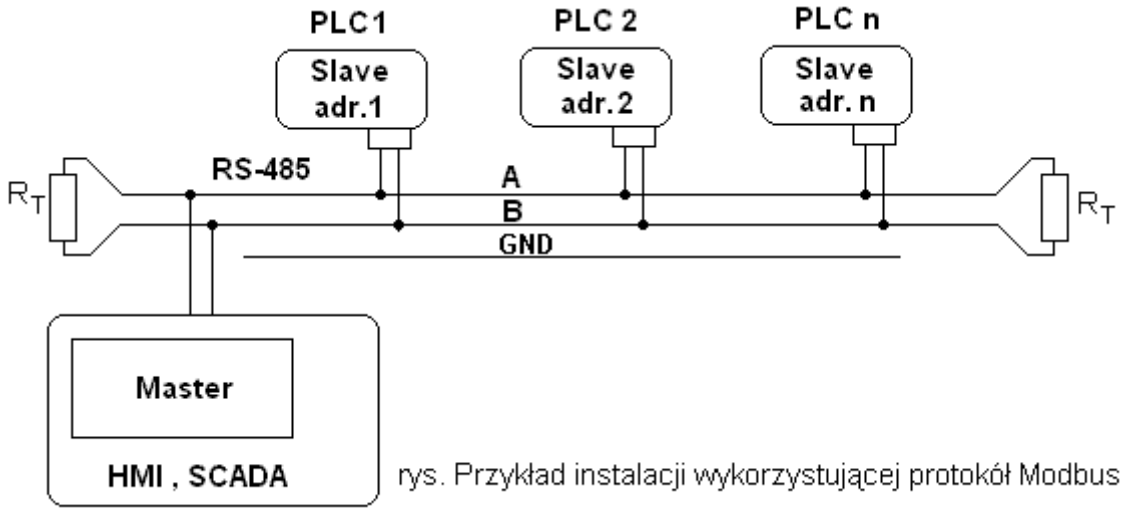
I Wprowadzenie

www.telmatik.pl

Protokół komunikacyjny MODBUS został stworzony przez firmę Modicon, jednak ze względu na liczne zalety, stał się standardem zaakceptowanym przez wielu producentów urządzeń automatyki . Protokół określa zasady wymiany informacji pomiędzy dwoma lub wieloma urządzeniami. MODBUS zapewnia możliwie szybkie przesłanie danych (np. przez grupowanie informacji o zawartości rejestrów i przesyłanie tylko jednego adresu początkowego), przy jednoczesnej kontroli, czy nie zostały one przekłamane.
Organizacja łączności miedzy urządzeniami oparta jest na prostej zasadzie, że nadrzędne jest jedno urządzenie typu Master (Pan) nazywane również klientem a reszta jest podrzędna typu Slave (sługa) inaczej nazywana serwerami..
Taka organizacja porządkuje zasady zajmowania łącza, a w połączeniu z sygnalizacją błędów, chroni przed zawieszeniem komunikacji.
Protokół MODBUS występuje w dwóch odmianach – wcześniejszej ASCII i późniejszej RTU. Ze względów praktycznych poniższe informacje dotyczą RTU i RTU Extended.

II Sprzęt

Inną, niż sam protokół, sprawą jest elektryczny (sprzętowy) sposób łączenia urządzeń. Komunikacja musi zapewniać wymianę informacji między Master a każdym Slave. W przypadku wielu urządzeń Slave konieczne jest inne rozwiązanie, niż proste, dwustronne złącze RS-232. Przykład połączenia sieciowego, zgodnego z dwuprzewodowym standardem RS-485, przedstawia rysunek.



rys. Przykład instalacji wykorzystującej protokół Modbus

W każdej chwili może nadawać tylko jedno urządzenie, pozostałe muszą „słuchać” tj. być w stanie dużej rezystancji. Jednak porty są tak budowane, że nawet przypadkowy konflikt tj. dwa urządzenia nadają jednocześnie, nie uszkadza urządzeń. Sygnałem informacyjnym, jest różnica potencjałów między żyłami A i B . Oczywiście dla takiej sieci muszą być ustalone parametry transmisji takie jak szybkość transmisji, ilość bitów na znak, bit parzystości, stopu np. 9600, 8, N, 1 a każde urządzenie Slave musi mieć swój indywidualny adres (numer)
Prosta komunikacja wygląda następująco: Master zajmuje linię i wysyła polecenie (odczytu lub zapisu rejestru) do urządzenia Slave o określonym adresie . Pytane urządzenie odpowiada informacją albo kodem błędu, o ile polecenie nie może być wykonane z rozpoznanej przyczyny.

III Podstawowe funkcje protokołu MODBUS RTU

Podstawowe polecenia protokołu MODBUS to: odczyt ewentualnie zapis (ustawienie) zmiennej dwustanowej (bit) albo zmiennej liczbowej (word). Zależnie od rodzaju zmiennej i możliwości jej zapisu wprowadzono podział – tabela:

Typ	Rodzaj zmiennej	Możliwość zapisu	Wykorzystanie
Discretes Input Wejścia dwustanowe	**Single bit** Jedno-bitowy	**Read** Tylko odczyt	dwustanowe wejścia albo wyjścia urządzeń
Coils Cewka , przekaźnik	**Single bit** Jedno-bitowy	**Read /Write** Odczyt / zapis	przekaźniki wewnętrzne, wyjścia urządzeń
Input Registers Rejestry wejściowe	**16-bit Word** Słowo 16-bitowe	**Read** Tylko odczyt	Liczbowe wartości na wejściach lub wyjściach
Holding Registers Rejestry pamietajace	**16-bit Word** Słowo 16-bitowe	**Read /Write** Odczyt / zapis	Odczyty i zapisy do rejestrów wewnętrznych

Przyjętym poleceniom zostały nadane kody (numery) funkcji uwzględniające powyższy podział rodzajów zmiennych . Poza podstawowymi funkcjami (poleceniami) wprowadzono jeszcze inne, np. zapisy wielu rejestrów jednym poleceniem (16), zgłaszanie błędów (08).

Zestawienie podstawowych funkcji protokołu MODBUS

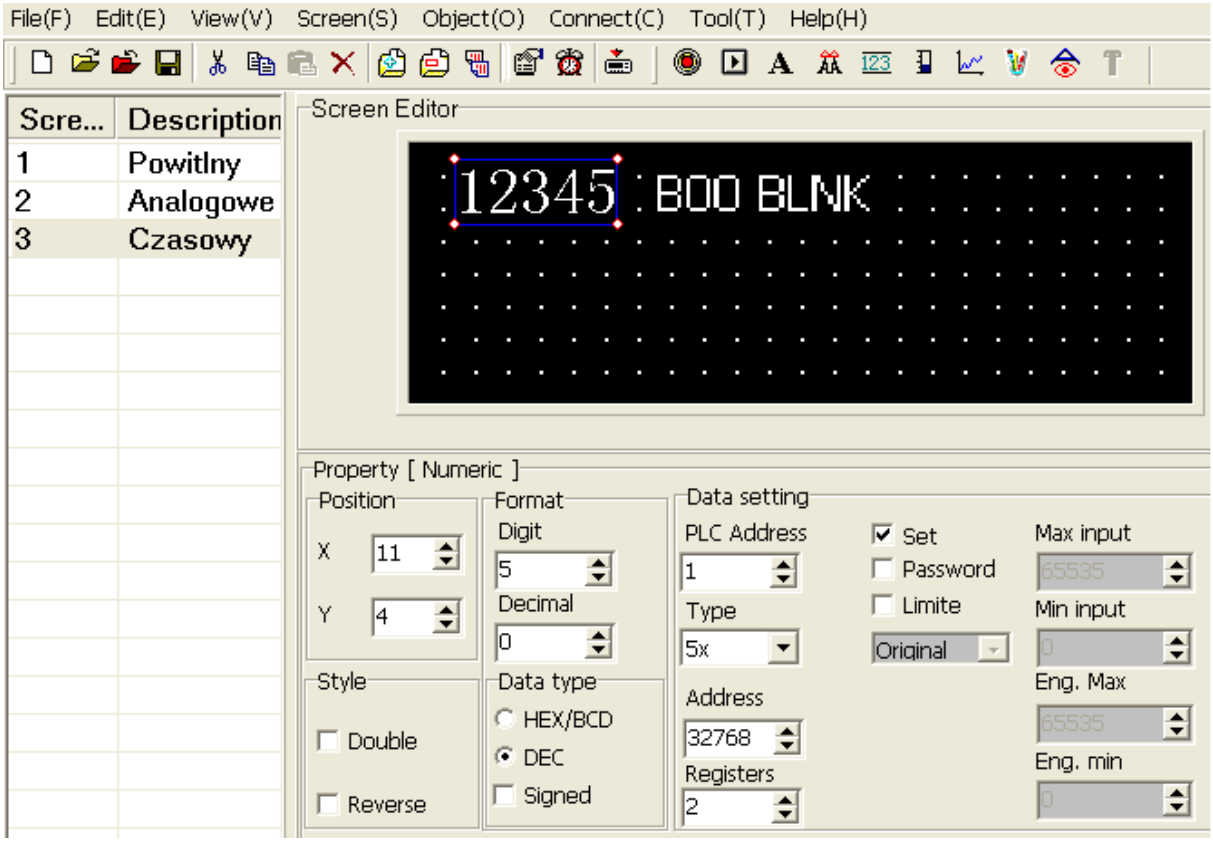
Opis funkcji	Kod funkcji (dziesiętnie)	Kod funkcji (szesnastkowo)
Read Coils Odczyt stanów wyjść binarnych. Np. wyjść PLC	**01**	**0x 01**
Read Discrete Inputs Odczyt stanów wejść binarnych. Np. wejść PLC	**02**	**0x 02**
Read Holding Register Odczyt rejestrów pamiętających	**03**	**0x 03**
Read Input Register Odczyt rejestrów wejściowych np. wartości z wejść analogowych PLC	**04**	**0x 04**
Write SingleCoils Zapis jednego wyjścia binarnego, ustawianie przekaźnika	**05**	**0x 05**
Write single Register Zapis do jednego rejestru pamiętającego	**06**	**0x 06**
Write Multiple Coils Zapis wielu wyjść binarnych, ustawianie przekaźników	**15**	**0x0F**
Write Multiple Registers Zapis do wielu rejestrów	**16**	**0x 10**
Diagnostyka, zgłaszanie błędów	**03**	**0x08**

IV Praktyczne wykorzystanie protokołu MODBUS RTU

Jeśli bezbłędnie znamy parametry urządzeń współpracujących, korzystanie z protokołu MODBUS nie wymaga szczegółowego śledzenia przesyłanych informacji. Wystarczy jedynie wypełnić właściwe pola w programie konfigurującym urządzenie Master (np. panel operatorski , HMI, Scada …) Komunikacja zakłada nadrzędną rolę urządzenia typu Master (Klient). To ono wysyła polecenia do urządzeń Slave (Serwer) i czeka na odpowiedź w postaci danych albo potwierdzenia wykonanego polecenia. Przy wysyłaniu poleceń najczęściej stosowane jest adresowanie indywidualne, ale może być ogólne (rozsiewcze).

O czym warto pamiętać:
- Adresy rejestrów liczbowych i zmiennych dwustanowych podaje producent urządzenia , podobnie, określa możliwość ich zapisu, bądź tylko odczytu.
- Adresy dla potrzeb protokołu MODBUS np. indywidualny dla każdego dwustanowego wejścia / wyjścia sterownika , nie są odzwierciedleniem organizacji pamięci urządzenia.
- Urządzenie typu Master musi w określonym czasie otrzymać informację o którą pytało, ewentualnie potwierdzenie wykonanego polecenia zapisu.
Brak dobrej odpowiedzi skutkuje wyświetleniem komunikatu o jej braku np. „No response", często bez podania przyczyny. W ustaleniu przyczyny bardzo pomocne są programy testujące, opis dalej
- często, panel operatorski, HMI wysyłają pytania związane tylko z elementami aktualnie wyświetlanymi na LCD. Jednak brak poprawnej odpowiedzi może dotyczyć pytań wysyłanych „w tle" np. o stany alarmowe.
- **Programy do konfiguracji urządzeń Master** (pozwalające na przypisanie obiektom adresów i poleceń zgodnie z adresami Slave) **często używają symboli funkcji, które nie są ścisłymi odpowiednikami funkcji MODBUS**, **co może mylić**. Np. pod jednym symbolem programu „4x" (obsługa rejestrów liczbowych), faktycznie obejmuje możliwość korzystania z kilku poleceń MODBUS 0x03, 0x06, 0x10.

Przykład wykorzystania protokołu MODBUS RTU do komunikacji panelu SH-300 ze sterownikiem APB
Panel SH-300 to urządzenie typu interfejs użytkownika (HMI). Wyposażony jest w ekran LCD, na którym można wyświetlać wartości liczbowe, dwustanowe - lampki, komunikaty stałe itd. Istnieją też przyciski powalające użytkownikowi na wprowadzane informacji . Przygotowanie do pracy polega na stworzeniu ekranów z elementami (obiektami) powiązanymi ze sterownikiem adresami dla MODBUS RTU podanymi przez producenta (można sprawdzić testerem), następnie przesłaniu projektu do SH-300.
a) Wyświetlanie i ewentualna zmiana wartości rejestru sterownika APB



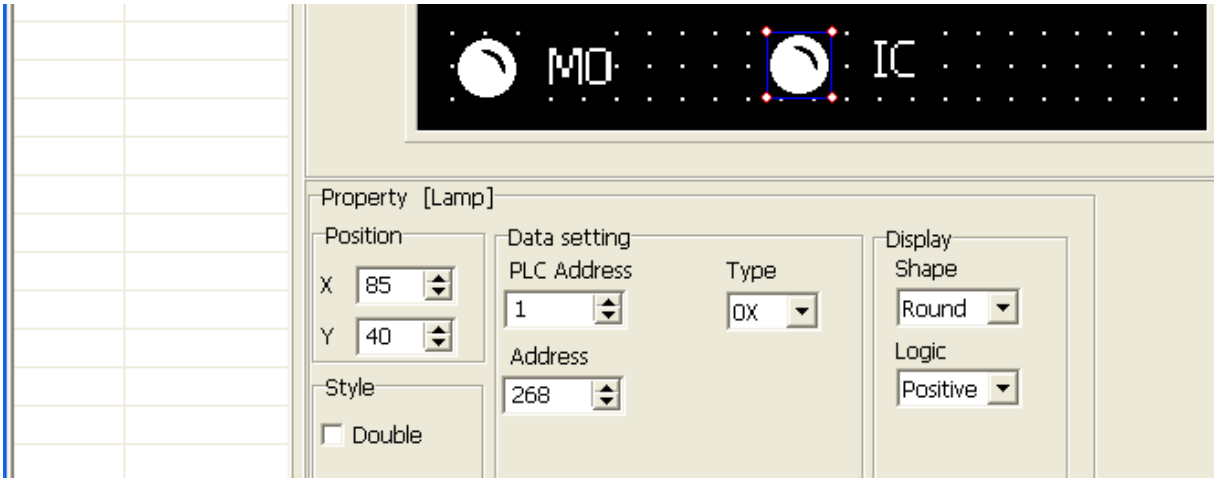
Czarne pole (z kropkami) obrazuje zawartość ekranu, docelowo wyświetlanego na LCD.

W niebieskiej ramce jest element (obiekt) wyświetlania zawartości rejestru, tu nastawy bloku czasowego sterownika. Wybrany typ 5x (opis w instrukcji) obejmuje funkcja MODBUS 0x03, 0x06, 0x10. Odczyty wykonywane będą funkcją 0x03, ale jeśli użytkownik zechce zmienić rejestr (dopuszczalne dzięki zaznaczeniu pola Set), nowa wartość przesłana będzie funkcją MODBUS 0x10. Zastosowanie 0x10 (a nie 0x06) wynika z faktu, że zapisywany rejestr jest 32. bitowy, czyli jakby dwa rejestry po 16 bitów (standard w MODBUS). Uwagę należy zwrócić na format adresu, tu zapis jest DEC (dziesiętny), w instrukcji może być Hex.. Kalkulator Windows potrafi przeliczyć formaty.

b) Przesyłanie licz ujemnych

Sam mechanizm fizycznego przesyłania liczby ujemnej, nie może różnić się od przesyłania liczby dodatniej , bo to jest ciąg 0 i 1 . Dlatego, aby wprowadzić rozróżnienie liczby ujemnej , stosuje się kodowanie zapisu. Można umówić się, że jedynie najstarszy bit, to znak liczby ale w praktyce bardziej praktyczne jest kodowanie U2 (uzupełnienie do dwóch) . Tu, najstarszy bit nie tylko wskazuje na znak ale ma też swoją wagę (wartość zależną od ilości bitów w słowie), potrzebną do obliczania ujemnej wartości przesłanej liczby. Aby urządzenie „ wiedziało", że stosowane jest kodowanie (niezbędne dla liczb ujemnych) w polu „Data type" zaznaczamy „Signet", czyli liczba ze znakiem. Wówczas, jeśli urządzenie stwierdzi 1 na najstarszym bicie liczby, wylicza wartość liczby ujemnej i ją wyświetli z symbolem „-". Przykładowe wyliczenie dla systemu 8 bitowego liczby -2. Ujemna waga najstarszego bitu, to 2 do potęgi (7) czyli -128. Pozostałe 7 bitów to uzupełnienie liczby 2 , czyli 126. Jeśli urządzenie odbierze 254 (11111110) wyświetli -2. Dla 16 bitów (MODBUS) waga MSB to -32768.
Sposób kodowania U2 pozwala interpretować liczby dodatnie wprost jako dwójkowe ale o ograniczonym zakresie (najbardziej znaczący bit musi być 0).
Szczęśliwie, to algorytmy urządzeń zajmują się kodowaniem liczb ujemnych., nam wystarcza zaznaczenie.
UWAGA . MODBUS zakłada, że słowo to 16 bitów . Przy przesyłaniu 32 bitów (DW) mniej znaczące słowo wygląda jak dla 16 bitów , górne to FFFF .

c) Odczyt wartości dwustanowej (bitowej, dyskretnej) np. stanu wejścia , wyjścia PLC



Pokazane w czarnym polu lampki, są obiektami dwustanowymi. Mogą pokazywać stan przekaźnika wewnętrznego, np. M0 , wejścia IC albo wyjścia sterownika.
Zaznaczony niebieską ramka element powiązany jest przez adres 268 (10C hex) z wejściem IC sterownika APB. Odczyt wartości może być funkcją MODBUS 0x01 albo 0x02 co jest „ukryte" w programie pod typem 0x (instrukcja SH-300)

Przytoczone przykłady pokazują, że często projektant nie musi wnikać, jak faktycznie realizowany jest protokół MODBUS. Jednak chcąc zapewnić sobie możliwość wykrywania niesprawności, czy zrozumienia zachodzących mechanizmów, warto poznać go nieco lepiej.

V Działanie protokołu MODBUS RTU

Przesyłane pytania i odpowiedzi maja określony format, opisany tzw. ramką.

Znacznik początku	Adres urządzenia	Kod funkcji	Dane	Kontrola CRC	Znacznik końca
T1-T2-T3-T4	8 bitów	8 bitów	n x 8 bitów	16 bitów	T1-T2-T3-T4

rys. Ramka w trybie RTU

Skrót RTU (Real Time Unit) w nazwie, oznacza, że komunikacja odbywa się z kontrolą czasu rzeczywistego.
T1do T4 to czas przerwy między przesyłanymi ramkami. Ramka musi być przesłana w całości tj, dopuszczalna przerwa między znakami nie może przekraczać 1,5 znaku. Dłuższa może być potraktowana jako przerwa, kończąca ramkę.
Część informacyjna ramki zaczyna się od **adresu urządzenia,** do którego kierowane jest polecenie. Osiem bitów pozwala na adresowanie 255. urządzeń Slave, przyczym, 0 jest wspomnianym adresowaniem rozsiewczym (brodcast), do wszystkich.
Kolejne przesyłane 8 bitów to **kod funkcji** np. jednej z wymienionych w tabeli wyżej. Oczywiście, tylko dla wygody człowieka, adresy, kody i inne informacje wyświetlane (wpisywane) są szesnastkowo albo dziesiętnie a nie binarnie.

UWAGA: należy zwracać uwagę na używane przy adresach systemy liczbowe. Dziesiętny (brak symbolu albo Dec), szesnastkowy (symbol Hex, albo 0x).

Pole **Dane n x 8 bitów** nie jest już tak jednoznaczne . Ilość przesyłanych danych wynika z rodzaju polecenia, ewentualnie odpowiedzi. Należy zwrócić uwagę, że przyjęcie jednostki informacyjnej jako 8 bitów, wymaga **przesyłania niektórych danych i adresów w dwóch częściach (Hi i Lo),** mimo że dotyczy to jednego rejestru . **Standard MODBUS przyjmuje, że rejestry są 16 bitowe** a przesyłanie odbywa się w **część Hi i Lo**. Warto o tym pamiętać, gdy śledzi się komunikację (programem testującym). Analiza trochę komplikuje się, gdy w odpowiedzi otrzymujemy ciąg zawartość kilku, kolejnych rejestrów (począwszy od wskazanego adresem), albo, gdy sterownik ma **rejestry 32 bitowe**. W polu „Dane" mogą znajdować się zarówno adresy rejestrów jak ich zawartości, wartości zmiennych bitowych oraz informacje o ilości przesyłanych danych, ilości przesyłanych bajtów. Szczególną uwagę należy zwrócić na fakt, że podawane adresy dla potrzeb MODBUS, nie są odpowiednikiem metody adresowania pamięci procesora urządzenia Slave. Np. każde dwustanowe wejście czy wyjście sterownika w MODBUS ma swój adres, choć wiadomo, że w procesorach nie ma takiego indywidualnego adresowania każdej zmiennej jednobitowej. Z drugiej strony, mimo, że protokołem czasami pytamy tylko o stan jednej zmiennej jednobitowej (jedno wejście czy wyjście PLC) minimalną jednostką przesyłanej informacji jest bajt (osiem bitów). Wyjaśnienie dalej, przy funkcji 0x01.

VI Kontrola transmisji MODBUS RTU programami testowymi, szczegóły protokołu

Korzystanie z programów sprawdzających działanie protokołu pozwala na lepsze zrozumienie protokołu MODBUS, jak też zapewnia możliwość wykrycia przyczyn ewentualnych kłopotów. Znajomość szczegółów i obserwacja transmisji dostarcza znacznie więcej

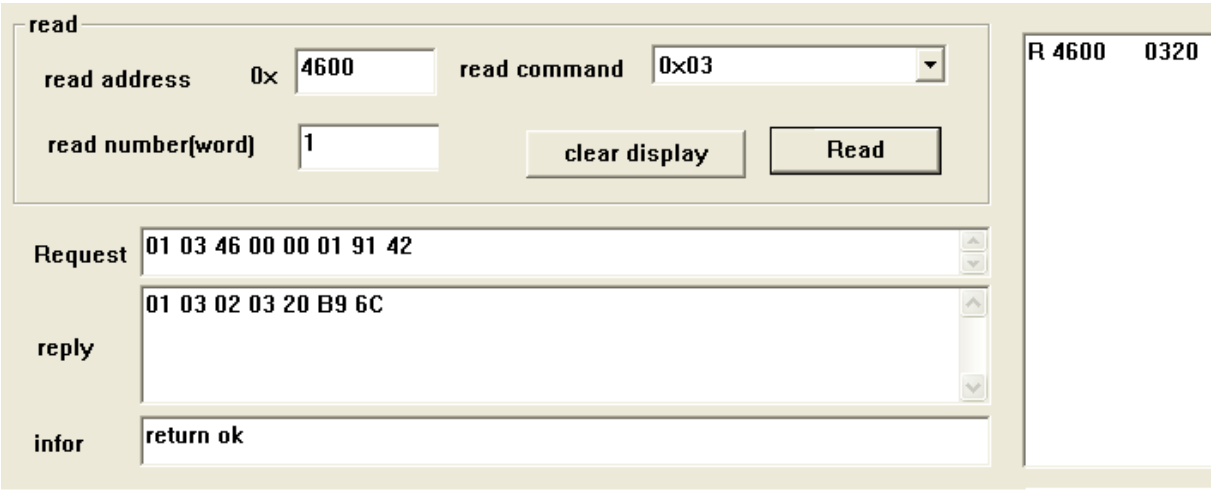
informacji, niż sam efekt końcowy typu działa / nie działa. Podobnie, jak fizyczne urządzenia, programy testujące mogą być stroną Master albo Slave, na co należy zwracać uwagę .

Poniżej przedstawiono widoki ekranów programu komputerowego symulującego stronę Master, który pozwala sprawdzać, jak odpowiada Slave np. sterownik APB. Master (program testujacy) z zadeklarowaną funkcją i wpisanymi adresami, wysyła żądanie (wyświetlone w polu Request) i otrzymuje odpowiedź (wyświetloną w polu Reply).

Oczywiście komunikacja odbywa się binarnie, ale dla ułatwienia obserwacji, program przesyłane wartości wyświetla w systemie szesnastkowym (hex) np. zamiast 1001010 pokaże 4A. Kolejność numerów wyjaśnianych funkcji, ma ułatwić rozumienie mechanizmów.

03 (0x03) Odczyt zawartości rejestrów pamiętających (Read Holding Register)

W prawym górnym rogu, na niżej pokazanym ekranie, widnieje wynik 0320 . Jest to efekt odczytu (Read) zawartości rejestru o adresie 4600. Otrzymana odpowiedź pokazana jest w okienku „reply'. Wysłane pytanie w okienku „Request" .



rys. Odczyt wartości analogowej z wejścia I0 sterownika APB. Funkcja 0x03.
Odczytana wartość 0320 Hex to 800 Dec

Liczby z okienka Request (żądanie mastera) mają ogólne znaczenie jak wcześniej pokazano na rys.„Ramka w trybie RTU" Dokładne wyjaśnienie ciągu liczb pokazano w poniższej tabeli. (nie widać T1- T4).

01	**03**	**46**	**00**	**00**	**01**	**91**	**42**
Adres PLC	Kod funkcji	Adres początkowy część Hi	Adres początkowy część Lo	Ilości rejestrów część Hi	Ilość rejestrów część Lo	CRC	CRC

Każda liczba to 8 bitów (od 00 do FF hex). Jak już wcześniej wspomniano, liczby 16 bitowe wysyłane są w dwóch częściach; Hi i Lo. Ponadto, aby zwiększyć szybkość transmisji, w pytaniu wysyłany jest tylko adres początkowego (pierwszego czytanego) rejestru (4600hex) i łączna liczba wszystkich rejestrów (tu 0001), które chcemy odczytać. Liczba odczytywanych rejestrów również przesyłana jest w dwóch częściach Hi i Lo .

CRC (tu 9142) jest tzw. sumą kontrolną, tj. liczbą obliczoną z zawartości ramki i po to przesyłana , aby strona odbiorcza mogła sprawdzić (obliczyć), czy „po drodze” nie nastąpiły przekłamania.

W odpowiedzi (okienko reply) master otrzymał

01	**03**	**02**	**03**	**20**	**91**	**42**
Adres PLC	Kod funkcji	Ilość przesyłanych bajtów	części Hi pierwszego rejestru	Część Lo pierwszego Rejestru	CRC	CRC

czyli: adres PLC=01, kod funkcji =03, ilość przesłanych bajtów =02 (ilość rejestrów N x 2) oraz zawartość jednego rejestru slave równą 0320 hex (dwa bajty). Jeśli zapytamy o większą (niż jeden) ilość N rejestrów w odpowiedzi uzyskamy Nx2 bajtów. W przypadku odczytu rejestrów 32 bitowych, w programie testującym w polu „read number (word)”, należy wpisać 2 , bo w MODBUS przyjęto za podstawę rejestry 16 bitowe (8bit Hi i 8bit Lo).W takiej sytuacji przesłane będą 04 bajty.

06 (0x06) Zapis do jednego rejestru pamiętającego (Write single Register)

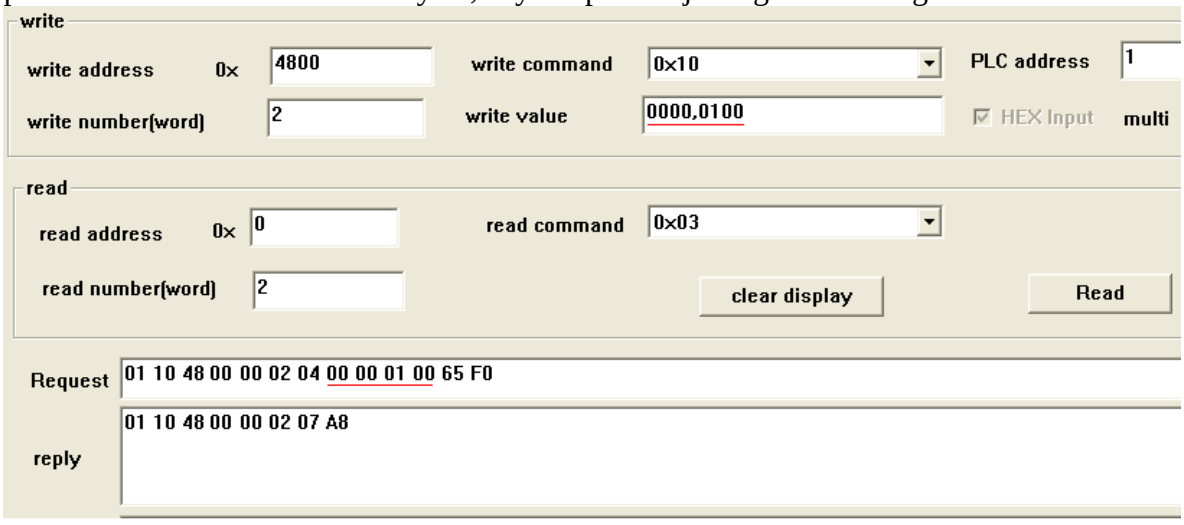
Podobnie jak odczyt pojedynczego rejestru przesyłane jest polecenie zapisu jednego rejestru. Przykład:

01	**06**	**00**	**01**	**00**	**03**		
Adres PLC	Kod funkcji	Część Hi adresu rejestru	Część Lo adresu rejestru	Część Hi zawartości rejestru	Część Lo zawartości rejestru	CRC	CRC

Do rejestru o adresie 0001 zostanie zapisana wartość 0003

16 (0x10) Zapis do wielu rejestrów (Write Multiple Registers)

Funkcja nr16 (0x10) pozwala jednym poleceniem zapisać więcej niż jeden, standardowy rejestr 16 bitowy , ewentualnie jest konieczna przy zapisach do rejestrów 32bitowych. Zapis jednego rejestru 32 bitowego funkcją 06 spowodowałby przesłanie tylko dwóch bajtów, tj Hi i Lo a więc aktualizację tylko 16 bitów. Funkcja 16(0x10) umożliwi jednoczesne przesłanie dwóch słów 16 bitowych , czyli zapisanie jednego 32bitowego.

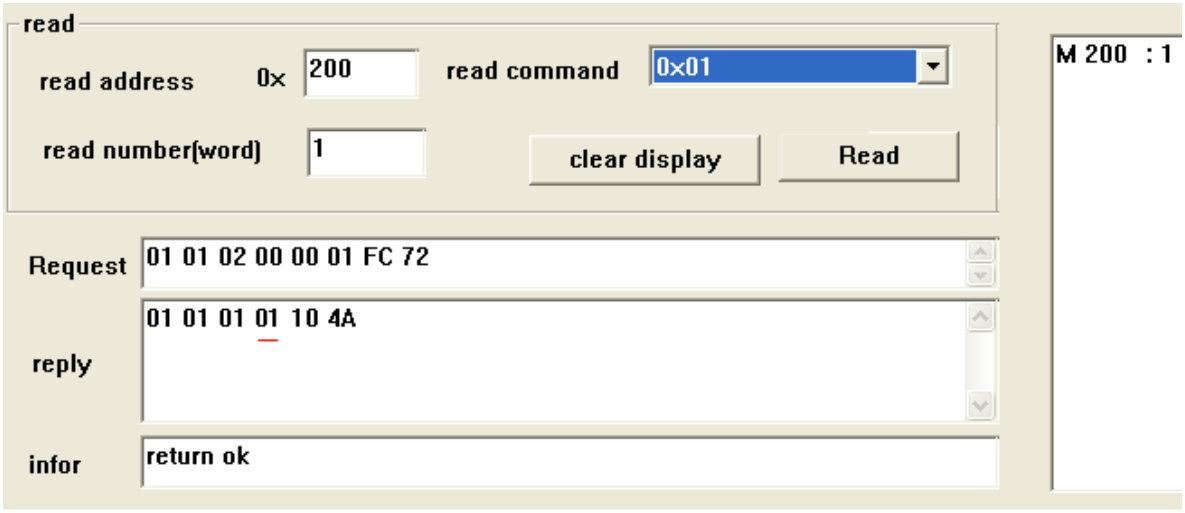


rys. Zapis rejestru 32 bitowego funkcją (0x10) wielokrotnego zapisu standardowych rejestrów 16 bitowych

Znaczenie **01**- adres PLC, **10** - kod funkcji , **4800**-adres początkowy, **0002**-liczba rejestrów, **04**-liczba przesyłanych bajtów, **00 00 01 00** – przesyłana zawartość dwóch rejestrów (dziesiętnie to jest 256) 16 bitowych albo jednego 32 bitowego.

UWAGA Wpisywana w testerze wartość „write value „ jest pełną liczbą (np. 12345678) zamieniona na hex (np. bc,614e). Przecinek jest tylko pomocniczy, oddziela słowa 16bitowe. Przy urządzeniach 32 bitowych numery kolejnych rejestrów zmieniają się co 2 , ponieważ słowo w Modus RTU jest 16 bitowe (2 bajty) . W APB, 32 bitowe rejestry DW (Double Word) są 4800, 4802, 4804 itd.

01 (0x01) Odczyt wyjść dwustanowych, stanów przekaźników (Read Coils)



rys. Pytanie o stan wyjścia Q0 sterownika i uzyskana odpowiedź. Funkcja 0x01

Nieco zamieszania jest przy odczycie / zapisie zmiennych dwustanowych (input , coils) Rysunek przedstawia przebieg przesłania pytania (okienko Request) o wartość tylko jednej zmiennej dwustanowej (funkcja 0x01 Read Coils , sterownika o adresie 01). Zgodnie z instrukcją sterownika, ta informacja występuje pod adresem 0x200 (zapisany w okienku read address). W odpowiedzi uzyskano informację aż ośmiobitową (podkreślona), bo jednostką przesyłanej informacji jest bajt. Tak naprawdę w tym przypadku interesuje nas tylko stan bitu wysłanego jako pierwszy, który tu ma wartość 1 (wyświetlony w prawym górnym rogu programu) .

Znaczenia poszczególnych liczb w wyżej wysłanym poleceniu - pytaniu (Pole Request):

01	**01**	**02**	**00**	**00**	**01**	**FC**	**72**
Adres PLC	Kod funkcji	Adres początkowy część Hi	Adres początkowy część Lo	Ilości cewek, część Hi	Ilość cewek, część Lo	CRC	CRC

Ilość wyjść (stanów cewek), o które możemy pytać, zawiera się między 1 a 2000 (0x7D0)

Opis znaczenia liczb uzyskanej od Slave odpowiedzi (Pole Reply)

01	**01**	**01**	**01**	**10**	**4A**
adres PLC	kod funkcji	ilość bajtów	wartość bajtu -ów	CRC	CRC

Wyjaśnienia wymaga jedynie pole odpowiedzi „wartość bajtu” (osiem bitów). Mimo, że pytanie dotyczyło stanu jednego wyjścia, sterownik odpowiedział całym bajtem. Wyświetlona liczba (tu w hex) to stan interesującego bitu uzupełniony do ośmiu samymi 0 albo rzeczywistymi stanami kolejnych wyjść (z wagą dwójkową …32, 16, 8, 4, 2, 1). Ten drugi przypadek następuje, gdy pytamy o więcej niż jedno wyjście (read number) ale mniej

nż osiem. Przesyłanie nadmiarowej informacji (zamiast 0) o faktycznym stanie kolejnych wyjść nie przeszkadza, bo jest ignorowane. Np. przy stanie wyjść Qn=0 … Q1=1, Q0=1 i pytaniu tylko o stan Q0, w polu wartość bajtu, odpowiedź może być 00000001 (01hex) albo 00000011 (03hex), zależnie od urządzenia, ale ważny będzie tylko stan wyróżniony kolorem. **Uwaga na kolejność numeracji –** w sterowniku wyjście zerowe często jest z lewej, a w zapisie dwójkowym najmniej znacząca pozycja LSB (zerowa) jest z prawej .

Jeśli pytanie będzie o więcej niż 8 wyjść, ilość bajtów odpowiedzi będzie liczbą całkowitą powstałą przez podzielenie ilości wyjść przez 8 z zaokrągleniem w górę .
Przy wielu bajtach odpowiedzi należy uważać na ich kolejność i powiązanie z numerami wyjść. Najpierw wysyłany jest bajt z informacją o niższych numerach wyjść, później o wyższych , ale w ramach bajtu kolejność jest odwrotna (MSB … LSB).
Przykład:
Pytamy o 16 wyjść począwszy od adresu 200hex tj wyjścia Q0 sterownika (instrukcja PLC)

Pytanie (bez CRC na końcu)

01	**01**	**02**	**00**	**00**	**10**
Adr PLC	Funkcja	Część Hi adresu początkowego	Część Lo adresu początkowego	Część Hi ilości wyjść	Część Hi ilości wyjść

Odpowiedzi powinna zawierać dwa bajty z czego pierwszy zawiera informację o stanie wyjść o adresach 207 do 200 czyli wyjść sterownika Q7 do Q0 a drugi Q15 do Q8 (w podanej kolejności)

Przykład odpowiedzi (bez CRC na końcu)

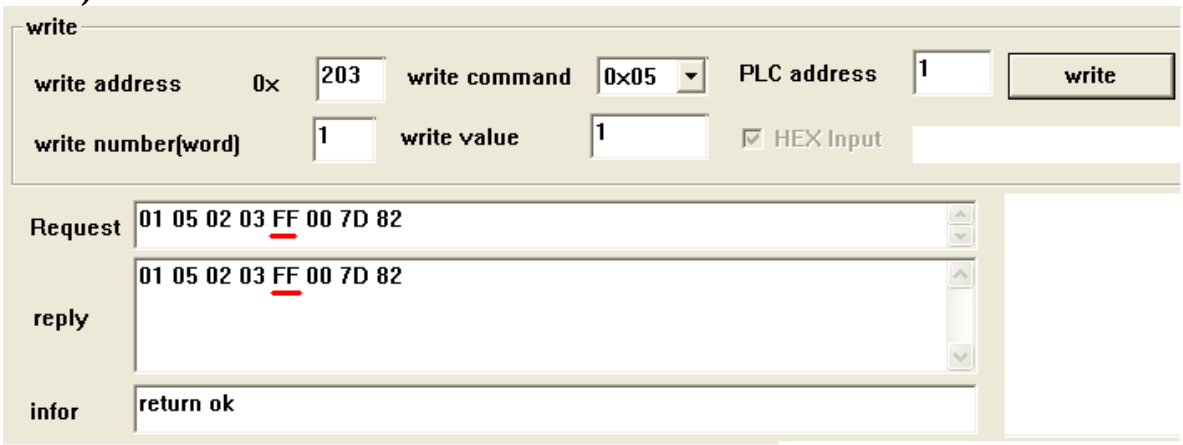
01	**01**	**02**	**CD**	**6B**
Adr PLC	Funkcja	Ilość bajtów	Stan wyjść 207 -200 (Q7 – Q0)	Stan wyjść 20F -200 (Q15-Q8)

Czyli wyjścia sterownika Q7 … Q0 mają stan 11001101 (CD hex)
a wyjścia Q15…Q8 mają stan 01101011 (6B hex)

02 (0x02) Odczyt stanów wejść binarnych. Np. wejść PLC (Read Discrete Inputs)

Przebieg transmisji przy odczycie stanu wejść (0x02) jest analogiczny jak przy odczycie wyjść (0x01). Producenci urządzeń niekiedy stosują wyłącznie funkcję (0x01)

05 (0x05) Zapis pojedynczego wyjścia dwustanowego, ustawianie przekaźnika (Write Coils)



rys. Ustawienie stanu "1" na wyjściu Q3 sterownika APB. Funkcja 0x05.

01 – adres sterownika, **05** – kod funkcji , **0203 -** adres ustawianego wyjścia. **FF00-** polecenie ustawienia w stan 1 (ON), **7D82** - CRC **.**
Jeśli wyjście (cewka) ma być zerowane (OFF), zamiast **FF00** wysłane zostanie **0000**
Występuje więc zasada jak przy odczycie 0x01, że pierwszy przesyłany bajt np. FF dotyczy wyjść z niższymi adresami (numerami) ale tak naprawdę chodzi tylko o jeden bit w stanie 1.
Uwaga: Spotykane, nazywanie tych bajtów jako Hi i Lo jest mylące !

15 (0x0F) Zapis wielu wyjść binarnych, ustawianie przekaźników (**Write Multiple Coils)**

Zapis wielu wyjść binarnych odbywa się na zasadach podobnych jak odczyt. 0x01, z tym, że jako polecenie przesyłana jest treść zapisu (dwa bajty), a nie pytanie o stan wyjść (cewek) .

Polecenie zapisu 10 wyjść począwszy od adresu 200hex (bez CRC)

01	**0F**	**02**	**00**	**00**	**0A**	**02**	**CD**	**01**
Adres PLC	funkcja	cz. Hi adresu początkowego	cz. Lo adresu początkowego	Ilość wyjść Hi	Ilość wyjść Lo	Ilość bajtów	Bajt 1	Bajt 2

Takie polecenie oznacza, że wyjścia o adresach od 200hex do 209hex (10 wyjść sterownika tj. od Q0 do Q9) będą zapisane jak niżej :

207 hex … 200hex (Q7 …Q0) wartościami 11001101 (bajt 1 = CD hex)
209 hex , 208 hex (Q9, Q8) wartościami - - - - - - 01 (bajt 2 = 01 hex)

Odpowiedź potwierdzająca wykonanie polecenie ustawienia wyjść zawiera kod funkcji, adres początkowy, ilość zapisanych wyjść (cewek)

Odpowiedź (bez CRC na końcu)

01	**0F**	**02**	**00**	**00**	**0A**
Adres PLC	funkcja	cz. Hi adresu początkowego	cz. Lo adresu początkowego	Ilość wyjść część Hi	Ilość wyjść część Lo

08 (0x08) Diagnostyka, zgłaszanie błędów

O ile zachodzi komunikacja między urządzeniami, uzyskiwane odpowiedzi choć niezgodne z oczekiwaniem, mogą zawierać ważne informacje o przyczynach błędów.

Informacja o błędach składa się z: bajtu adresu urządzenia , bajtu informującego o błędzie, bajtu o rodzaju błędu i dwóch CRC.
Przykład pytania

01	01	02	00	00	01	**FC**	**72**

Odpowiedź informująca o błędzie

01	81	02	91	C1

Pierwszy bajt (po adresie urządzenia) informuje o błędzie w wykonaniu polecenia
wg/ zasady nr funkcji = nr błedu; 0x01 = 81, 0x02 = 82, 0x3=83 …0x0F=8F, 0x10 = 0x90

Drugi bajt zawiera informację o rozpoznanej przyczynie. Podstawowe kody błędów :

01 - niedopuszczalna funkcja
02 - niedopuszczalny adres danych

03 - niedopuszczalna wartość danych
04 - błąd urządzenia slave (niezidentyfikowana przyczyna odmowy wykonania polecenia)

Numery wyższe to błędy specjalistyczne np.
05 - potwierdzenie prawidłowego odbioru (stosowane gdy właściwa odpowiedź jest długa i może powodować przekroczenie czasu - timeout)