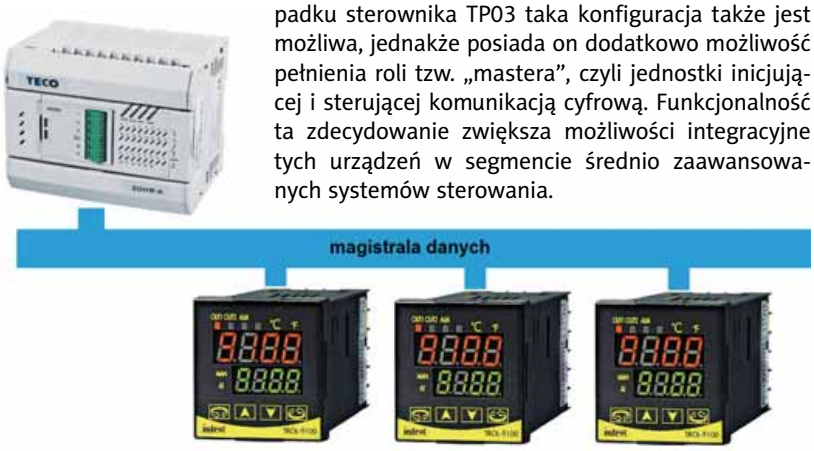
# Szkolenie PLC cz. 12
# Komunikacja szeregowa w sterowniku TECO TP03

***Sterownik PLC z powodzeniem może zostać wykorzystany jako autonomiczne urządzenie sterujące procesem, jednakże w ogromnej ilości aplikacji konieczna jest jego integracja z innym urządzeniem lub urządzeniami. Połączenie może zostać zrealizowane dzięki wykorzystaniu sygnałów binarnych lub cyfrowej komunikacji szeregowej, która to będzie tematem głównym poniższego artykułu.***

## KOMUNIKACJA SZEREGOWA

Cyfrowa komunikacja szeregowa jest najpopularniejszą metodą komunikacji pomiędzy urządzeniami elektronicznymi. W zależności od implementacji, parametrów elektrycznych oraz wykorzystywanego protokołu komunikacyjnego, można wyróżnić szereg standardów stosowanych w aplikacjach pomiarowych i sterujących, zarówno przemysłowych jak i laboratoryjnych. Ogólna charakterystyka najpopularniejszego w naszej ocenie protokołu, przedstawiona została szerzej w jednym z poprzednich artykułów. Na przykładzie przekaźnika programowalnego SG2 oraz panelu operatorskiego WEINTEK, omówiona została implementacja protokołu MODBUS RTU z wykorzystaniem interfejsu RS485. W wymienionym przykładzie, sterownik SG2 pełnił rolę urządzenia podrzędnego, odpytywanego i parametryzowanego przez urządzenie nadrzędne, którym był panel dotykowy. W przypadku sterownika TP03 taka konfiguracja także jest możliwa, jednakże posiada on dodatkowo możliwość pełnienia roli tzw. „mastera", czyli jednostki inicjującej i sterującej komunikacją cyfrową. Funkcjonalność ta zdecydowanie zwiększa możliwości integracyjne tych urządzeń w segmencie średnio zaawansowanych systemów sterowania.

▼ *Rysunek 1* Struktura przykładowego systemu




## KOMUNIKACJA MASTER-SLAVE NA STEROWNIKU TP03

Aby przybliżyć fizyczny charakter sieci wymiany danych pomiędzy urządzeniami, komunikacja zostanie omówiona w oparciu o sterownik TP03 pełniący rolę jednostki centralnej systemu i regulatory TROL 9100 jako elementy podrzędne – tzw. „slave'y". Na rysunku nr 1 przedstawiona została struktura przykładowej aplikacji typu single master – multi slave (pojedyncze urządzenie nadrzędne oraz kilka urządzeń klienckich).

Każde z urządzeń klienckich musi posiadać unikalny adres z zakresu 0-255. W innym przypadku, zapytania kierowane do urządzenia ze zwielokrotnionym adresem, będą prawdopodobnie zakłócane i/lub tracone ze względu na możliwą, wzajemną interferencję odpowiedzi pochodzących od różnych slave'ów. Dlatego też istotne jest odpowiednie skonfigurowanie każdego z urządzeń przed rozpoczęciem prac programistycznych.

Z punktu widzenia połączenia elektrycznego, implementacja interfejsu RS485 realizowana jest poprzez wykorzystanie dwu-przewodowego kabla. Transmisja ta jest transmisją różnicową, co oznacza że zawsze sygnał na jednym z przewodów jest zanegowanym, chwilowym sygnałem z przewodu drugiego. W modułach odbierających dane, sygnały te są odpowiednio analizowane i przekazywane dalej. Wykorzystanie transmisji różnicowej zapobiega indukowaniu się zakłóceń z otoczenia, gdyż potencjalne zakłócenie ingeruje w obie linie danych. W odbiorniku sygnały są odejmowane, dlatego też ewentualne szumy z obu linii znoszą się. Możliwe jest także podłączenie sygnału masy (GND) zarówno w nadajniku i odbiorniku, jednakże nie musi być to stosowane na krótkich odległościach pomiędzy urządzeniami.

Sterownik TP03 wyposażony jest w co najmniej jeden port danych interfejsu RS485 oraz jeden port komunikacyjny interfejsu RS232. Dodatkowo można sterownik doposażyć w moduł komunikacyjny rozszerzający możliwości łączeniowe o drugi port RS485. Ma to szczególne zastosowanie w systemach, w których sterownik pracuje jako master dla części urządzeń, oraz jako slave dla nadrzędnego systemu danych, takich jak panel operatorski.

## PARAMETRYZACJA PORTU SZEREGOWEGO

Podobnie jak we wspomnianym artykule, zajmiemy się protokołem MODBUS RTU, jednakże tym razem postaramy się zaimplementować urządzenie typu master na sterowniku PLC. Pierwszym krokiem jest parametryzacja sprzętowego portu szeregowego znajdującego się na każdej jednostce centralnej. Konfiguracja odbywa się poprzez wpisanie odpowiednio spreparowanej liczby szesnastobitowej do rejestru przyporząd-

▶▶ Tabela 1 Konfiguracja portu szegerowego

| Numer bitu | Parametr | Zawartość |
|---|---|---|
| b0 | ilość bitów danych | 0: 7 bitów<br>1: 8 bitów |
| b2, b1 | parzystość | (0,0): brak<br>(0,1): odd<br>(1,0): even |
| b3 | bit stopu | 0: 1 bit<br>1: 2 bity |
| b7,b6,b5,b4 | prędkość transmisji | (0,1,1,1): 9,600<br>(1,0,0,0): 19,200<br>(1,0,0,1): 38,400<br>(1,0,1,0): 57,600<br>(1,0,1,1): 76,800<br>(1,1,0,0): 128,000<br>(1,1,0,1): 153,600<br>(1,1,1,0): 307,200 |
| b8 | znak startowy | 0: brak<br>1: STX |
| b9 | znak stopu | 0: brak<br>1: ETX |
| b10 – b15 | - | niedostępna |

*Szkolenie prowadzi:*
**Dominik Szewczyk**

*tel.: 32 789 00 13*

kowanego dla każdego z portów. W przypadku portu domyślnego jest to rejestr **D8120**. W tabeli nr 1 przedstawione zostały możliwe do ustawienia parametry oraz ich pozycje w rejestrze konfiguracyjnym.

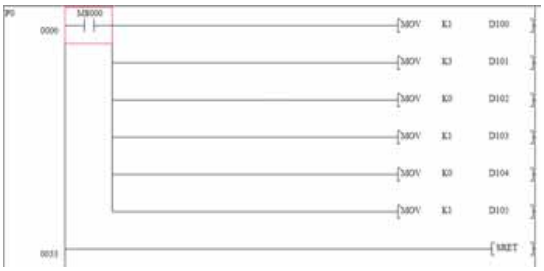
W przypadku konfiguracji naszego systemu, liczbą wynikową jest **81** szesnastkowo, czyli **129** w systemie dziesiętnym i **0000 0000 1000 0001** binarnie. Konfiguracja jest więc następująca:

- 8 bitów danych,
- brak kontroli parzystości,
- jeden bit stopu,
- prędkość transmisji 19,200 bps,
- brak znaku startowego,
- brak znaku stopu.

## IMPLEMENTACJA PROTOKOŁU MODBUS RTU

Podstawy programowania w języku drabinkowym były omówione wcześniej, dlatego też od razu przejdziemy do analizy istotnych fragmentów kodu źródłowego. Pierwszym zadaniem jest wspomniana już parametryzacja interfejsu RS485. Na rysunku nr 2 przedstawione zostały odpowiednie instrukcje. Wszystkie cztery równoległe linie kody poprzedzone są znacznikiem **M8002**, który jest aktywny tylko podczas pierwszego cyklu skanowania kodu. Instrukcje te są więc wykonywane tylko raz, podczas startu sterownika. Pierwsza linia to wspomniana już parametryzacja portu, druga linia definiuje czas oczekiwania na odpowiedź od urządzenia klienckiego (wpisując liczbę 5, uzyskujemy czas t=5*10ms, czyli 50 ms). W przypadku przekroczenia tego czasu, zwracany jest błąd transmisji. Znacznik **M8161** dotyczy wyboru pomiędzy trybem komunikacji ośmiobitowym i szesnastobitowym (w naszym przypadku wykorzystujemy tylko osiem bitów rejestru, stąd też ustawienie **M8161**). Ostania instrukcja pokazanego fragmentu realizuje skok do podprogramu **P0**. Jak już zostało wspomniane w poprzednich szkoleniach, skoki są pozbawione parametrów, a zmienne są globalne i widoczne w każdym podprogramie.

Skok nawiguje do fragmentu kodu odpowiedzialnego za konstrukcje ramki danych, gdyż w sterowniku TP03 użytkownik sam musi nadać ramce kształt. Jest to zarówno udogodnienie jak i problem związany z większym nakładem pracy. Jednakże wraz z większym skomplikowaniem kodu, zwiększają się możliwości funkcjonalne urządzenia nadrzędnego, pracującego w trybie master. Na rysunku nr 3 przedstawiony został podprogram tworzący ramkę danych.



Dzięki wykorzystaniu instrukcji inicjacji danych **Mov**, wpisujemy odpowiednie liczby do rejestrów **D100 – D105**, które będą wykorzystane jako źródło danych dla zapytania. Pierwszy rejestr to adres urządzenia klienckiego, drugi to typ rozkazu (liczba 3 oznacza rozkaz odczytu rejestrów wewnętrznych urządzenia), kolejne dwa bity określają adres danych, a ostatnie ilość danych. Powyższa ramka odpytuje urządzenie podrzędne o wartość rejestru **0×01**. Ostatnia linia jest informacją dla sterownika, iż w tym miejscu kończy się skanowanie podprogramu i należy wrócić do miejsca w kodzie, skąd wykonywany był skok.

Do tej pory użytkownik dokonał parametryzacji portu szeregowego oraz stworzył ramkę danych dla zapytania modbus'owego. Kolejny fragment dotyczy wysłania ramki poprzez interfejs. Na rysunku nr 4 widzimy znacznik zbocza narastającego na rejestrze binarnym **M8013**. Jest to nic innego jak impulsator, aktywny dokładnie co 1 sekundę. Szeregowo z impulsatorem, umieszczona jest zanegowana instrukcja wejściowa markera **M0**. Można to odczytać w ten sposób, iż instrukcje wyjściowe będą realizowane co sekundę, pod warunkiem, że marker **M0** jest nieaktywny (sens tej konstrukcji będzie opisany w dalszej części szkolenia). Pierwsza linia realizuje instrukcję parametryzacji bufora wyjściowego. Marker **M8122** ustawiany jest przed każdorazowym wysłaniem ramki na port i kasowany automatycznie po udanej transmisji. Ostatecznie w drugiej linii, ustawiamy marker **M0**, który kontroluje fizyczne wysłanie danych do sieci.

Ostatni fragment kodu (rysunek nr 5) realizuje najistotniejsze instrukcje z punktu widzenia transmisji, gdyż dotyczy wysłania ramki danych z zapytania na port szeregowy oraz odbioru ewentualnych danych zwrotnych, pochodzących z urządzenia podrzędnego. Pierwsza linia wysyła ośmiobitowe dane rozpoczynające się od rejestru **D100** (wysyłanych jest 6 kolejnych rejestrów, o czym świadczy drugi parametr instrukcji **MBUS**). Dane zwrotne maja być zapisane począwszy od rejestru **D200** w sześciu kolejnych jednostkach pamięci. Ostatnim parametrem instrukcji jest numer portu – **K0** oznacza wbudowany port RS485.

Odbiór odpowiedzi przez bufor wejściowy wskazywany jest przez aktywację markera **M8123**. W tym momencie resetowany jest wskaźnik **M0**, blokujący wysyłanie danych, więc kolejne zapytanie może zostać zrealizowane (pozwolenie na transmisję). Kasowany jest także sam wskaźnik odbioru danych. Dane zwrotne znajdują się w rejestrach **D200 – D205** i są odświeżane każdorazowo po odebraniu ramki zwrotnej. W sytuacji gdy programista chce zachować te dane, należy je skopiować do innych rejestrów.

Powyższy przykład jest działającą implementacją urządzenia typu master na sterowniku TP03. Należy jednak pamiętać, iż żeby transmisja taka miała sens, dane muszą być wykorzystane do innych operacji arytmetycznych lub logicznych. Obróbka surowych danych pochodzących z transmisji i ich zastosowanie przy sterowaniu procesami będzie tematem kolejnych szkoleń.

◀ *Rysunek 4* Wysłanie ramki danych

◀◀ *Rysunek 2* Parametryzacja transmisji

◀ *Rysunek 5* *Wysłanie i odbiór ramki*

◀◀ *Rysunek 3* Konstrukcja ramki danych