



Modbus RTU w TwinCAT 3

Pełna instrukcja obsługi Modbus RTU

Wersja dokumentacji 1.0

Aktualizacja: 22.10.2021

Kontakt: *support@beckhoff.pl*

Beckhoff Automation Sp. z o. o.

Spis treści

1	Wstęp.....	5
2	Biblioteka TC2_Modbus_RTU	6
2.1	Wybór bloku funkcyjnego do obsługi Modbus RTU	6
2.1.1	Jak sprawdzić rozmiar bufora danych ?	7
2.2	Opis bloku Modbus RTU Master	8
2.3	Opis bloku Modbus RTU Slave	10
2.4	Obszary Pamięci.....	11
2.4.1	Adresacja obszaru wejść.....	11
2.4.2	Adresacja obszaru wyjść	11
2.4.3	Adresacja obszaru pamięci	12
3	Konfiguracja sprzętowa	13
3.1	COM Port	13
3.2	Konfiguracja KL60XX na przykładzie KL6021 – 22 bajt	14
3.2.1	Konfiguracja za pomocą programu KS2000.....	14
3.2.2	Konfiguracja poprzez blok funkcyjny KL6Configugation.....	15
3.3	EL60xx	16
3.3.1	Konfiguracja poprzez zakładkę CoE – Online.....	16
3.3.2	Konfiguracja poprzez zakładkę CoE – Startup	18
3.4	Przykładowe linkowanie zmiennych procesu komunikacyjnego.....	21
4	Quick Start - Przykładowy projekt	23
4.1	Utworzenie nowego projektu.....	23
4.2	Konfiguracja przykładowego obiektu	25
4.3	Utworzenie obiektu typu Modbus Slave	28
4.4	Utworzenie obiektu typu Modbus Master + przykład komunikacji	30
5	Tips & Tricks.....	34
5.1	Wyprowadzenia portu DB9 i przykład połączenia	34
5.1.1	Realizacja połączenia Full-Duplex	35
5.1.2	Realizacja połączenia Half-Duplex	35

Uwaga! Poniższy dokument zawiera przykładowe zastosowanie produktu oraz zbiór zaleceń i dobrych praktyk. Służy on wyłącznie celom szkoleniowym i wymaga szeregu dalszych modyfikacji przed zastosowaniem w rzeczywistej aplikacji. Autor dokumentu nie ponosi żadnej odpowiedzialności za niewłaściwe wykorzystanie produktu. Dany dokument w żadnym stopniu nie zastępuje dokumentacji technicznej dostępnej online na stronie <https://infosys.beckhoff.com> . Instrukcja jednocześnie zakłada znajomość podstaw programowania w środowisku TwinCAT 3.

© Beckhoff Automation Sp. z o.o.

Wszystkie obrazy są chronione prawem autorskim. Wykorzystywanie i przekazywanie osobom trzecim jest niedozwolone.

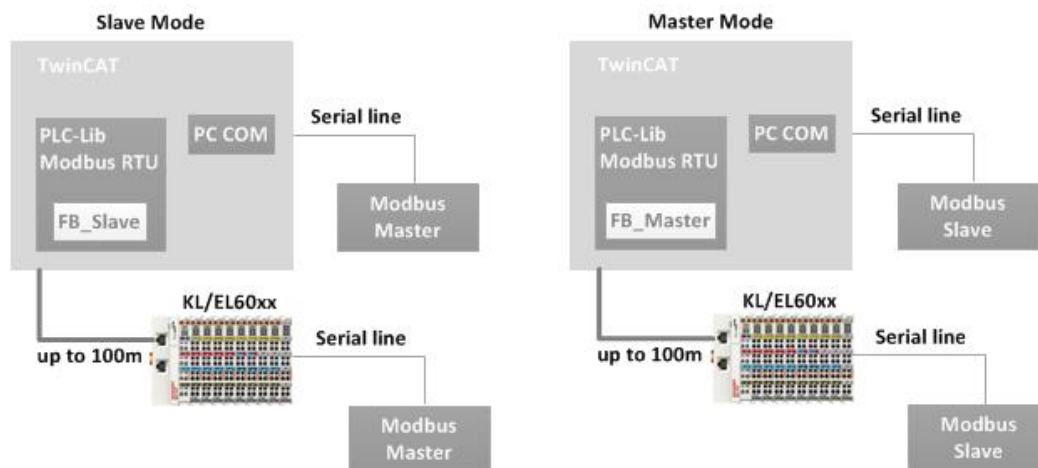
Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® i XTS® są zastrzeżonymi znakami towarowymi i licencjonowanymi przez Beckhoff Automation GmbH. Inne oznaczenia użyte w niniejszym dokumencie mogą być znakami towarowymi, których użycie przez osoby trzecie do własnych celów może naruszać prawa właścicieli.

Informacje przedstawione w tym dokumencie zawierają jedynie ogólne opisy lub cechy wydajności, które w przypadku rzeczywistego zastosowania nie zawsze mają zastosowanie zgodnie z opisem, lub które mogą ulec zmianie w wyniku dalszego rozwoju produktów. Obowiązek przedstawienia odpowiednich cech istnieje tylko wtedy, gdy zostanie to wyraźnie uzgodnione w warunkach umowy.

1 Wstęp

Rozszerzenie TF6255 wbudowane w środowisko TC3 XAE pozwala na komunikację poprzez protokół Modbus RTU. Każde urządzenie wykorzystujące system TC3 XAR oraz posiadające wyprowadzenie do magistrali RS232, RS422 lub RS485 może zostać Masterem bądź też Slavem w sieci Modbusa RTU. Programista ma możliwość dołączenia urządzenia z TC3 do magistrali komunikacyjnej poprzez następujące komponenty hardware'owe:

- Poprzez port szeregowy na maszynie PC, IPC lub też na sterowniku CX
- Terminale magistrali K-BUS z serii KL60xx
- Terminale magistrali E-BUS z serii EL60xx

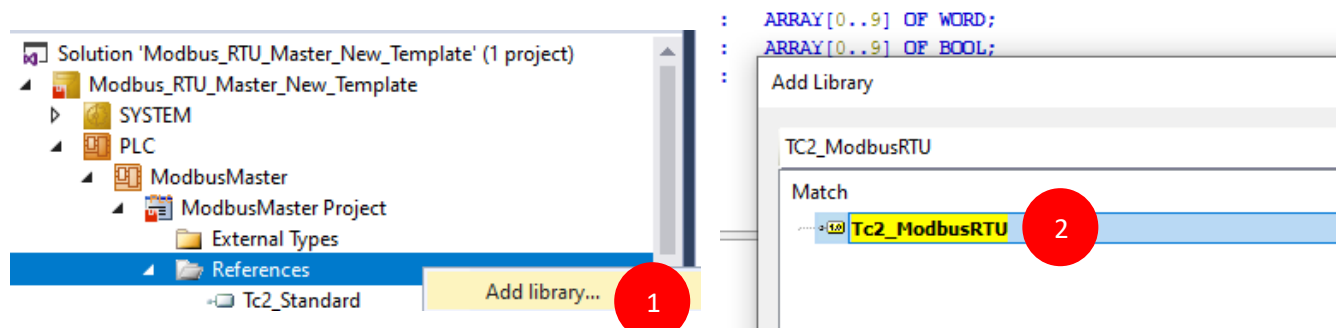


Niezależnie od wybranej konfiguracji sprzętowej do komunikacji po protokole Modbus RTU, funkcjonalność pozostaje identyczna.

W zależności od danego wyprowadzenia do komunikacji po protokole Modbus RTU możemy posiadać różne rozmiary buforów danych (domyślnie mogą to być 5 bajt, 22 bajt lub też 64 bajt), maksymalną szybkość transmisji jak i inne dodatkowe cechy charakterystyczne dla magistrali RS232, RS422 lub też RS485.

2 Biblioteka TC2_Modbus_RTU

Biblioteka TC2_Modbus_RTU jest kluczowym elementem konfiguracji komunikacji poprzez protokół Modbus RTU. Samą bibliotekę można dodać do projektu PLC poprzez zakładkę References:



2.1 Wybór bloku funkcyjnego do obsługi Modbus RTU

Przeglądając zawartość biblioteki można zauważyć następujące bloki funkcyjne :

Nazwa bloku	Tryb komunikacji	Rozmiar pakietu danych	Proponowane przeznaczenie
ModbusRtuMaster_PcCom	Master	64 Bajt	Moduły/wyprowadzenia z Port COM
ModbusRtuSlave_PcCom	Slave	64 Bajt	Moduły/wyprowadzenia z Port COM
ModbusRtuMaster_KL6x5B	Master	5 Bajt	Moduły KL z rozwinięciem 6001,6011,6021
ModbusRtuSlave_KL6x5B	Slave	5 Bajt	Moduły KL z rozwinięciem 6001,6011,6021
ModbusRtuMaster_KL6x22B	Master	22 Bajt	Moduły KL6031, KL6041 oraz niektóre moduły EL60XX
ModbusRtuSlave_KL6x22B	Slave	22 Bajt	Moduły KL6031, KL6041 oraz niektóre moduły EL60XX

W zależności od wymaganego bufora danych oraz posiadanej konfiguracji sprzętowej należy wybrać odpowiedni blok funkcyjny. Zasada wyboru boku zawsze przebiega w dowolnej kolejności przez następujące 2 etapy:

1. Czy wyprowadzenie do magistrali protokołu Modbus RTU zawiera 5 bajtów, 22 bajty czy 64 bajty bufora danych?
2. Czy dany port ma pracować w trybie Master czy Slave ?

2.1.1 Jak sprawdzić rozmiar bufora danych ?

Wchodząc na stronę beckhoff.com oraz wyszukując dany moduł do magistrali komunikacyjnej RS232/422/485 możemy odczytać liczbę bajtów które możemy wykorzystać w TwinCAT 3 do buforowania danych. Na jej podstawie będziemy wybierali blok funkcyjny który zostanie wykorzystany w programie PLC.

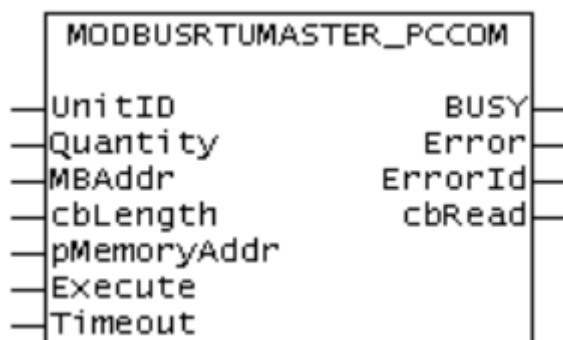
Electrical isolation	500 V (K-bus/signal voltage)
Data buffer	1024 bytes receive buffer, 128 bytes transmit buffer
Bit width in the process image	input/output: 22 x 8 bit user data, 2 x 8 bit control/status (up to 22 byte user data are possible)
Configuration	no address setting, configuration via Bus Coupler or controller
Special features	high interference immunity, electrically isolated signals

Rozmiar proces data możemy również odczytać bezpośrednio z modułu w konfiguracji I/O projektu PLC.

Domyślnie wbudowane porty COM posiadają 64 bajty do komunikacji oraz wszystkie moduły z serii EL60xx posiadają 22 Bajty.

2.2 Opis bloku Modbus RTU Master

Niezależnie od wybranego bloku funkcyjnego funkcjonalność oraz wyprowadzenia pozostają identyczne.



Sama funkcjonalność bloków funkcyjnych ModbusRTUMaster opiera się na wywoływaniu sparametryzowanych akcji pozwalających na nadawanie komend oraz analizowaniu informacji zwrotnej w zależności od rodzaju komendy. Elementami wymagającymi parametryzacji są :

Wartości wejściowe

UnitID	Adres z zakresu 0 – 247 określający ID urządzenia docelowego danej komendy. Adres 0 jest przeznaczony na komendy typu broadcast
Quantity	Liczba wartości jaka ma zostać odczytana z adresu docelowego. Przykładowo liczba bitów lub cewek
MAddr	Adres danych modbusa z którego mają być odczytane dane
cbLength	Rozmiar danych jaki zostanie odczytany. Zalecane jest zastosowanie rozmiaru $\geq 2 * \text{Quantity}$
pMemoryAddr	Wskaźnik na adres z pamięci. W przypadku wysyłania danych, wartości są ładowane z tego adresu. W przypadku odczytu danych, wartości zostaną zapisane na tym adresie
Execute	Sygnał startu wyzwalany poprzez wykrycie zbocza narastającego na tym wejściu
Timeout	Wartość czasu po jakim komenda zostanie unieważniona w przypadku braku informacji zwrotnej

Wartości wyjściowe

Busy	Wyprowadzenie będzie posiadać stan wysoki w przypadku jeżeli obecnie blok funkcyjny wykonuje daną akcję. W przypadku powodzenia lub też niepowodzenia wartość spada do stanu False. W trakcie obecności wartości tego wyprowadzenia w stanie wysokim nie można używać innych akcji !
Error	Wyprowadzenie przechodzi w stan wysoki jeżeli w trakcie ostatniej akcji komunikacji wystąpiło niepowodzenie
ErrorId	Numer błędu z listy MODBUS_ERRORS
cbRead	Informuje o odczytanej liczbie bajtów po zakończeniu ostatniej akcji

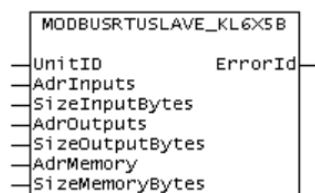
Niezależnie od rodzaju zastosowanego bloku funkcyjnego ModbusRTUMaster każdy blok posiada identycznie wyzwalane akcje pozwalające na komunikację. Zostały one utworzone zgodnie z ideą podziału funkcji w dokumentacji protokołu Modbus RTU. Podział Akcji wygląda następująco :

Nr funkcji Modbus	Nazwa Akcji	Opis
1	.ReadCoils	Odczytuje aktualny stan wyjść. Dane przychodzą w skompresowanej formie 8 wyjść na bajt(8 bitów na bajt).
2	.ReadInputStatus	Odczytuje aktualny stan wejść. Dane przychodzą w skompresowanej formie 8 wejść na bajt(8 bitów na bajt).
3	.ReadRegs	Odczytuje dane z urządzenia docelowego. Dane są odczytywane w formacie WORD
4	.ReadInputRegs	Odczytuje wartości rejestrów wejść . Dane są odczytywane w formacie WORD.
5	.WriteSingleCoil	Zapisz nową wartość na pojedynczym wyjściu. Dane mówiące o stanie muszą zostać wysłane w skompresowanej formie 8 wyjść na bajt (8 bit na bajt). Zostaną one odczytane z adresu pMemoryAddr
6	.WriteSingleRegister	Zapisz pojedynczy rejestr pamięci. Wartość zostanie odczytana z adresu pMemoryAddr
15	.WriteMultipleCoils	Zapisz nowe wartości na wyjściach. Dane mówiące o stanie muszą zostać wysłane w skompresowanej formie 8 wyjść na bajt (8 bit na bajt). Zostaną one odczytane z adresu pMemoryAddr.
16	.WriteRegs	Zapisz nowe wartości w pamięci. Dane zostaną odczytane z adresu pMemoryAddr.
8	.Diagnostics	

Przykładowe wywołanie bloku odczytu wejść

```
(*Funkcja 4 (Read Input Registers) - Odczyt rejestrów z pamięci Input*)
MB.ReadInputRegs(
    UnitID:=1 ,                               (*Adres Slave*)
    Quantity:=10 ,                            (*Ilość czytanych rejestrów w słowach (ilość rejestrów 16bit)*)
    MAddr:=16#0 ,                             (*Adres modbusowy - Input 16#0*)
    cbLength:= 20,                            (*Ilość rejestrów 8bit (Quantity*2)*)
    pMemoryAddr:=ADR(arrInputs) ,             (*Adres zmiennej do której zwracany jest wynik zapytania *)
    Execute:= TRUE,
    timeout := T#1s,
    Error=>bError
);
```

2.3 Opis bloku Modbus RTU Slave



Bloki Funkcyjne ModbusRTUSlave_xxx pozwalają na utworzenie slave'a Modbusowego poprzez zmienne wskazujące obszary wejść, wyjść jak i obszary pamięci. Jednocześnie na jednym wyprowadzeniu do magistrali RS – owej może istnieć jeden slave Modbusowy. Blok Funkcyjny ModbusRTUSlave_xxx w celu prawidłowego funkcjonowania musi być wywoływany w każdym cyklu. Sam blok funkcyjny jest pasywny i nie zużywa czasu procesora aż do momentu przyjścia komendy zewnętrznej. Blok funkcyjny posiada następujące wyprowadzenia:

Parametry wejściowe

UnitID	Adres ID slave'a z zakresu 1-247
AdrInputs	Adres początkowy tablicy wejść. W celu jego określenia może zostać użyty blok ADR();
SizeInputBytes	Rozmiar tablicy wejść. W celu jego wyliczenia może zostać użyta komenda SIZEOF()
AdrOutputs	Adres początkowy tablicy wyjść. W celu jego określenia może zostać użyty blok ADR();
SizeOutputBytes	Rozmiar tablicy wyjść. W celu jego wyliczenia może zostać użyta komenda SIZEOF()
AdrMemory	Adres początkowy tablicy pamięci wewnętrznej. W celu jego określenia może zostać użyty blok ADR();
SizeMemoryBytes	Rozmiar tablicy pamięci wewnętrznej. W celu jego wyliczenia może zostać użyta komenda SIZEOF()

Parametry wyjściowe

ErrorId	Numer kodu błędu z listy Modbus_ERRORS
---------	--

Przykładowa implementacja

```

PROGRAM ModbusSlave
VAR
    arrInput      : ARRAY[0..20] OF WORD;
    arrOutput     : ARRAY[0..20] OF WORD;
    arrMemory     : ARRAY[0..20] OF WORD;
    MB: ModbusRtuSlave_PcCOM;
END_VAR

```

```

MB(
    UnitID:=1 ,
    AdrInputs:= ADR(arrInput),
    SizeInputBytes:=SIZEOF(arrInput) ,
    AdrOutputs:=ADR(arrOutput) ,
    SizeOutputBytes:=SIZEOF(arrOutput) ,
    AdrMemory:=ADR(arrMemory) ,
    SizeMemoryBytes:=SIZEOF(arrMemory) ,
    ErrorId=> );

```

2.4 Obszary Pamięci

W komunikacji po protokole Modbus RTU istotne jest właściwe adresowanie wejścia komunikacyjnego *MBAAdd* w celu poprawnego określenia adresu przeznaczenia używanego przez daną akcję odczytu/zapisu. **Uwaga:** niezależnie od wybranego sposobu deklaracji zajmowanego obszaru pamięci, nie będzie to miało wpływu na funkcjonowanie komunikacji przy wykorzystaniu Modbusa RTU.

2.4.1 Adresacja obszaru wejść

Definicja wejść w TwinCAT 3 nieco różni się od standardowej definicji wejść. Programista ma możliwość dokładnego określenia pozycji zadeklarowanej zmiennej w obszarze pamięci wejść:

```
Inputs AT%IW0 : ARRAY[0..255] OF WORD;
```

Jak i również ma możliwość utworzenia zmiennej w pamięci stosując automatyczne przypisanie adresu w obszarze pamięci:

```
Inputs : ARRAY[0..255] OF WORD;
```

Dostęp do odczytu wejść cyfrowych możemy użyć komendy *.ReadInputStatus* w bločku mastera Modbusa RTU. W przypadku chęci odczytu wartości wejściowych analogowych można zastosować komendę *.ReadInputRegs*.

Adresacja

Zmienna PLC	Typ zmiennej	Adres nadany w telegramie Modbus'a RTU (MBAAddr)	Adres rzeczywisty zależny od urządzenia
Inputs[0]	Word	16#0	30001
Inputs[1]	Word	16#1	30002
Inputs[0], Bit 0	Bit	16#0	10001
Inputs[1], Bit 14	Bit	16#1E	1001F

Przykładowo w celu odczytu danego rejestru przy użyciu komendy *.ReadInputRegs* użyjemy adresowania po zmiennych typu WORD używając odpowiednio dla każdego adresu 16#0, 16#1, 16#2 itd.

W przypadku chęci odczytu pojedynczych wejść cyfrowych będziemy używali adresowania bitowego w przestrzeni wordów adresowanego odpowiednio: 16#00, 16#01, 16#02 ... 16#0F, 16#10, 16#11 itd.

2.4.2 Adresacja obszaru wyjść

Adekwatnie do deklaracji obszaru wejść, deklarowanie obszaru wyjść posiada identyczne możliwości co do określenia docelowego obszaru pamięci wyjść. Pozwala na bezpośrednie przypisanie adresu wyjść do danej zmiennej na lokalnym sterowniku :

```
Outputs AT%QW0 : ARRAY[0..255] OF WORD;
```

Jak i programista ma możliwość użycia automatycznego przypisania obszaru pamięci

```
Outputs : ARRAY[0..255] OF WORD;
```

W przypadku chęci odczytu wyjść cyfrowych programista ma możliwość użycia akcji *.ReadCoils* lub też *.ReadHoldingRegisters*. Programista ma również możliwość wysterowania wyjść poprzez akcje *.WriteSingleCoil*, *.WriteSingleRegister*, *.WriteMultipleCoils* lub też *.WriteMultipleRegisters*. Szczegółowa funkcjonalność została opisana w rozdziale 2.2.

Adresacja

Obszar wyjść posiada stałe przesunięcie o wartości 16#800.

Zmienna PLC	Typ adresowania	Adres w telegramie Modbus'a RTU	Adres rzeczywisty na urządzeniu docelowym (zależnie od urządzenia)
Outputs[0]	Word	16#800	40801
Outputs[1]	Word	16#801	40802
Outputs[0], Bit 0	Bit	16#800	00801
Outputs[1], Bit 14	Bit	16#81E	0081F

2.4.3 Adresacja obszaru pamięci

Również w przypadku deklarowania obszaru pamięci programista ma możliwość wybrania docelowego obszaru pamięci wewnętrznej poprzez użycie następującej deklaracji :

```
Memory AT%MW0 : ARRAY[0..255] OF WORD;
```

Lub też użycie automatycznego przypisania obszaru pamięci:

```
Memory : ARRAY[0..255] OF WORD;
```

W celu odczytu danego obszaru pamięci można użyć akcji .ReadRegs. W celu zapisania obszarów pamięci można użyć z kolei akcji .WriteSingleRegister lub .WriteRegs.

Adresacja

PLC variable	Access type	Address in the Modbus telegram	Address in the end device (device-dependent)
Memory[0]	Word	16#4000	44001
Memory[1]	Word	16#4001	44002

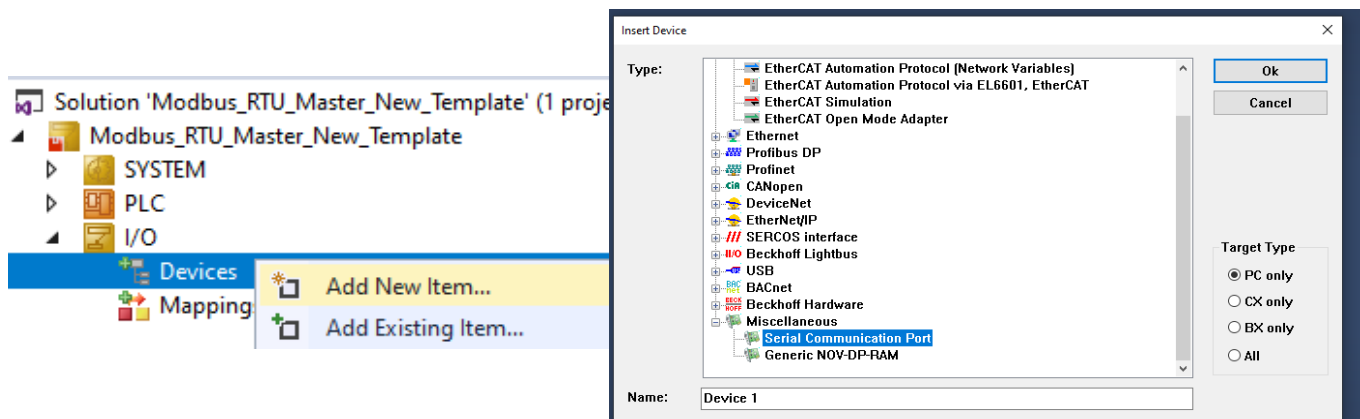
Obszar pamięci posiada stałe przesunięcie o wartości 16#4000.

Wszystkie komendy możliwe do zastosowanie na powyżej wymienionych obszarach pamięci zostały opisane w rozdziale 2.2.

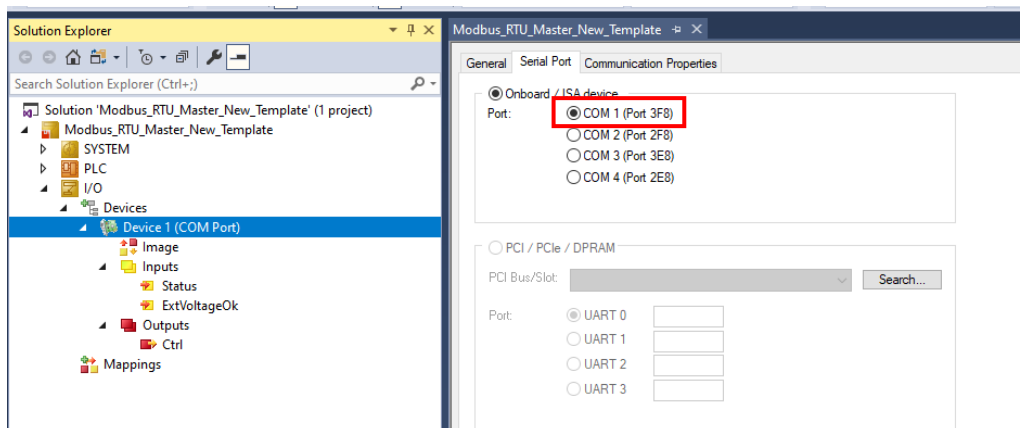
3 Konfiguracja sprzętowa

3.1 COM Port

W przypadku jeżeli urządzenie posiada port COM który będziemy wykorzystywać (np. CX9020 z opcją N031) mamy możliwość użycia tego portu w programie PLC jako interfejs magistrali np. RS485. W celu konfiguracji danego portu w pierwszej kolejności należy wyprać PPM na elemencie I/O / Devices i dodać element Serial Communication Port.



Pierwszym elementem, na który warto zwrócić uwagę jest ustawienie portu komunikacyjnego jako COM 1. Jest to domyślny adres pierwszego portu szeregowego dla każdego urządzenia z serii CX. Ustawienie można sprawdzić poprzez zakładkę Serial Port w danym obiekcie COM Port.



Następnie główną konfigurację danego portu można przeprowadzić w zakładce Communication Properties. Ze względu na użycie portu jako interfejsu komunikacyjnego do Modbusa należy zmienić tryb COM Port Mode na KL6xx1 Mode. Poniżej została przedstawiona przykładowa konfiguracja na sterowniku CX9020 z portem RS485 dla ramki Modbusa RTU o parametrach:

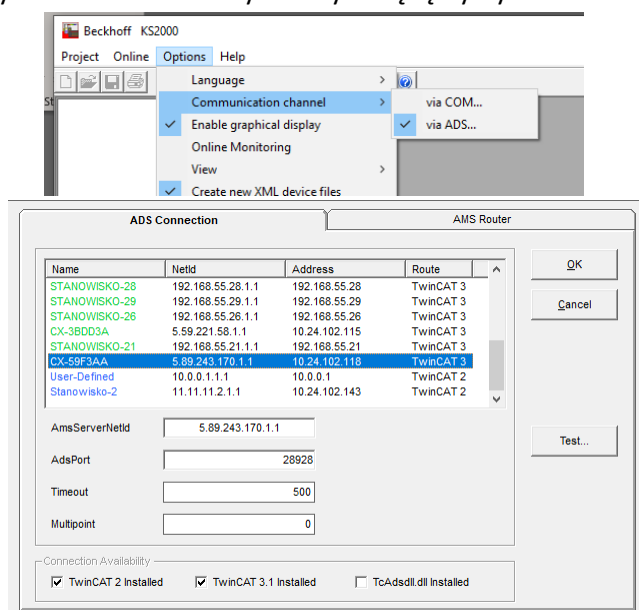
Baudrate	Parity	Stopbits	RS Type	Databits	Hardware Fifo
9600	None	1	RS485	8	16

3.2 Konfiguracja KL60XX na przykładzie KL6021 – 22 bajt

3.2.1 Konfiguracja za pomocą programu KS2000

Przed przystąpieniem do konfiguracji modułu KL60xx należy mieć zainstalowany dodatkowo płatny program KS2000. Pozwala on na konfigurację modułów K-Busowych. Następnie należy wykonać skanowanie **istniejącego** modułu w następującej kolejności :

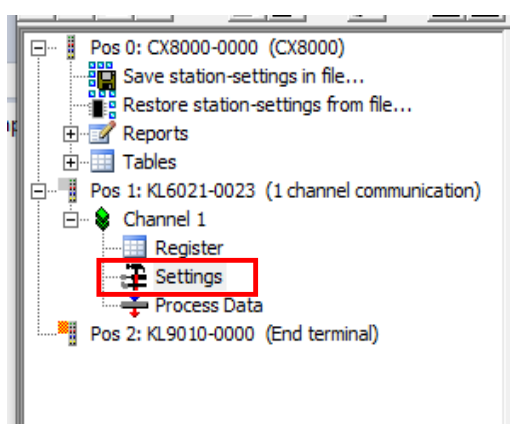
1. Zrestartować sterownik docelowy w tryb Config
2. Zeskanować konfigurację magistrali K-BUS **bez włączania trybu Toggle Free Run State**
3. Przejść do programu KS2000
4. Wybieramy sterownik docelowy z którym się łączymy



5. Logujemy się za pomocą programu KS2000 do sterownika



6. Wchodzimy w zakładkę **Settings** wybranego modułu komunikacyjnego

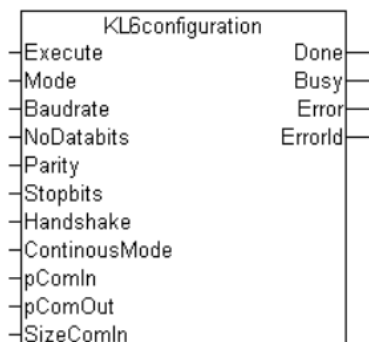


7. Konfigurujemy parametry komunikacji. Przykładowo:

8. Klikamy przycisk **Apply**

9. Zlinkować zmienne z I/O karty KL6021 adekwatnie według instrukcji opisanej w rozdziale 3.4

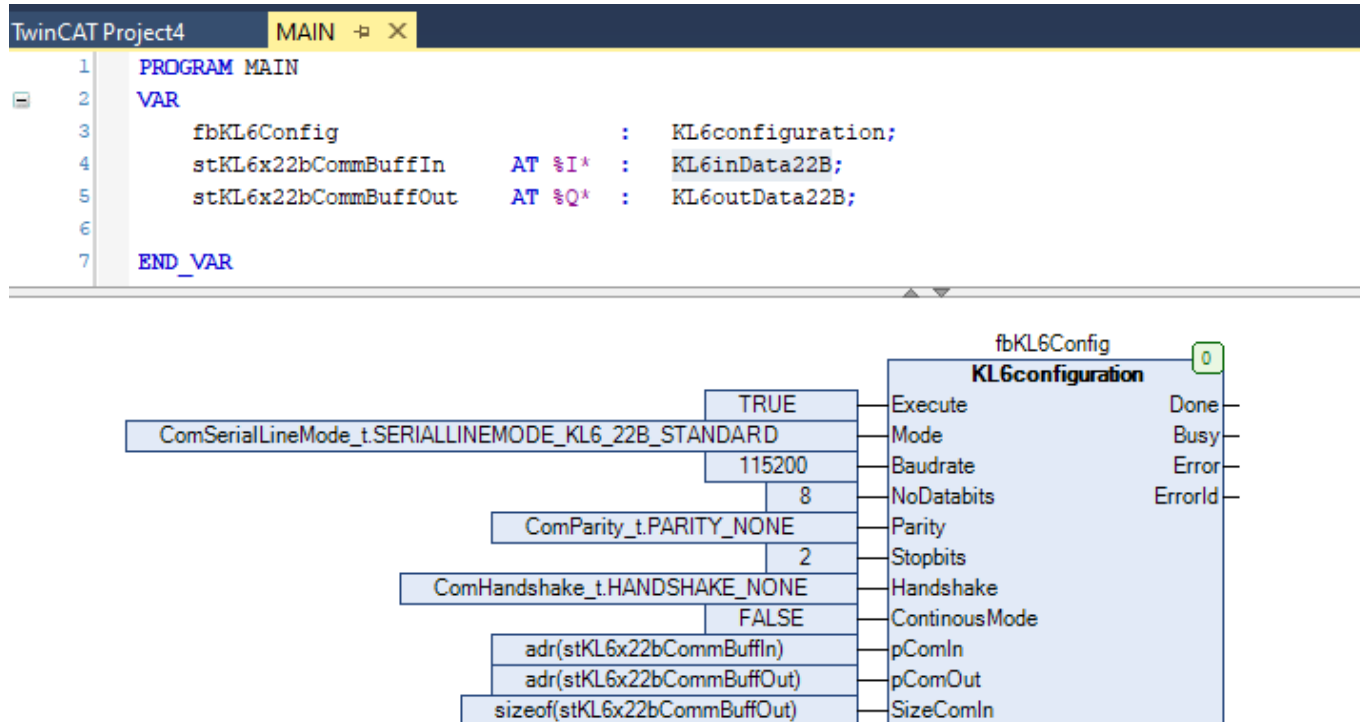
3.2.2 Konfiguracja poprzez blok funkcyjny KL6Configuation



Drugą możliwością konfiguracji karty jest użycie bloku funkcyjnego [KL6Configuration](#). Jest to blok pozwalający na określenie parametrów nastaw komunikacji wykorzystujących magistralę szeregową. W celu określenia który port należy utworzyć zmienne będące buforem komunikacyjnym posiadający możliwość podlinkowana do zmiennych komunikacyjnych karty KL6xxx. Zmiennymi buforowymi pozwalającymi na linkowanie są:

Data In	KL6inData, KL6inData5b, KL6inData22b
Data Out	KL6outData, KL6outData5b, KL6outData22b

Poniżej zaprezentowano przykładową parametryzację:



Zmienne stKL6x22bCommBuff zostały odpowiednio zlinkowane zgodnie z rozdziałem 3.4.

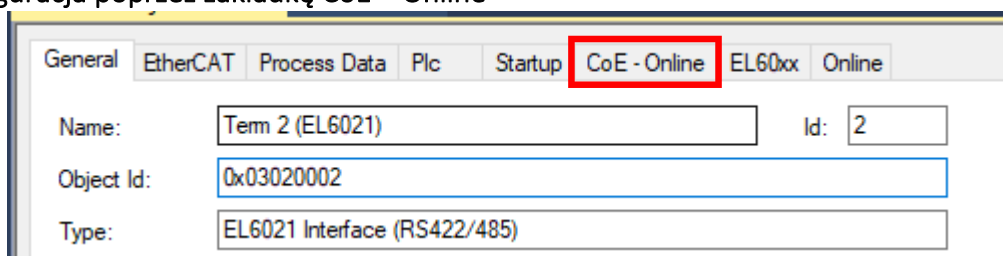
3.3 EL60xx

W przypadku kart z serii EL60xx mamy do dyspozycji rozmiar bufora danych w wielkości 22 bajtów niezależnie od wybranej karty jak i możliwość konfiguracji parametrów komunikacyjnych bez użycia oprogramowania KS2000. Samą konfigurację karty możemy przeprowadzić poprzez konfigurację po protokole CAN over Ethercat.

Komunikacja CAN over Ethercat pozwala na wprowadzenie parametrów komunikacyjnych bezpośrednio w przestrzeni rejestrów karty docelowej. Niezależnie od rozwiązania możemy konfigurować parametry na przykładowe 2 sposoby:

- konfiguracja poprzez zakładkę CoE – Online
- konfiguracja poprzez zakładkę CoE – Startup

3.3.1 Konfiguracja poprzez zakładkę CoE – Online



Część kart wymieniających informację poprzez magistralę E-BUS posiada możliwość konfiguracji parametrów nastaw bezpośrednio na samej karcie, niezależnie od konfiguracji sprzętowej. Parametryzację samej karty w trybie Online można wykonać poprzez zakładkę CoE-Online.

Unikatowe parametry konfiguracyjne charakterystyczne dla danej karty w większości przypadków znajdują się pod adresem 8000.

Index	Name	Flags	Value
7000:0	COM Outputs	RO	> 114 <
7001:0	COM ext. outputs	RO	> 66 <
8000:0	COM Settings	RW	> 28 <
8000:02	Enable XON/XOFF supported tx data	RW	FALSE
8000:03	Enable XON/XOFF supported rx data	RW	FALSE
8000:04	Enable send FIFO data continuous	RW	FALSE
8000:05	Enable transfer rate optimization	RW	TRUE
8000:06	Enable half duplex	RW	FALSE
8000:07	Enable point to point connection (RS...	RW	FALSE
8000:11	Baudrate	RW	9600 Baud (6)
8000:15	Data frame	RW	8N1 (3)
8000:1A	Rx buffer full notification	RW	0x0360 (864)
8000:1B	Explicit baudrate	RW	0x0002580 (9600)
8000:1C	Extended data frame	RW	8N1 (3)

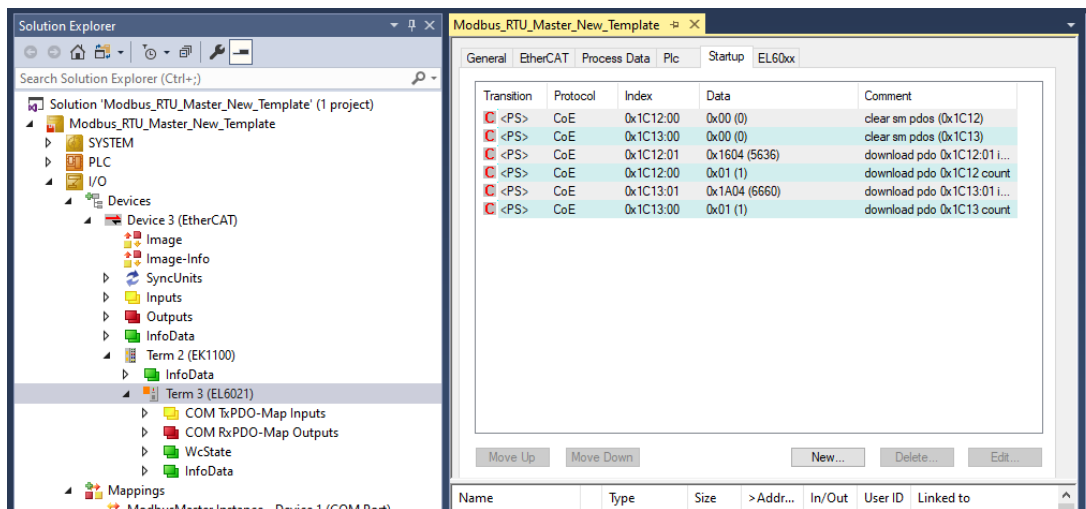
Zmienne z flagą RW – Read/Write programista ma możliwość nadpisania i zmiany parametrów. Zmianę danego parametru z listy można wykonać poprzez dwukrotne kliknięcie danego parametru LPM. W przypadku zmiennych przyjmujących wartości nieokreślone żadnym standardem (np. Enable Half Duplex) otworzy się nam okno posiadający standardowy szablon wprowadzenia nowej wartości. Nową wartość można wprowadzić w dowolnym formacie numerycznych i zapisać poprzez kliknięcie przycisku OK.

W przypadku zmiennych posiadających jakieś bliżej określone standardowe wartości (np. Baudrate) użytkownik ma możliwość wyboru danej wartości z rozwijanego paska wyboru, ułatwiając wprowadzanie.

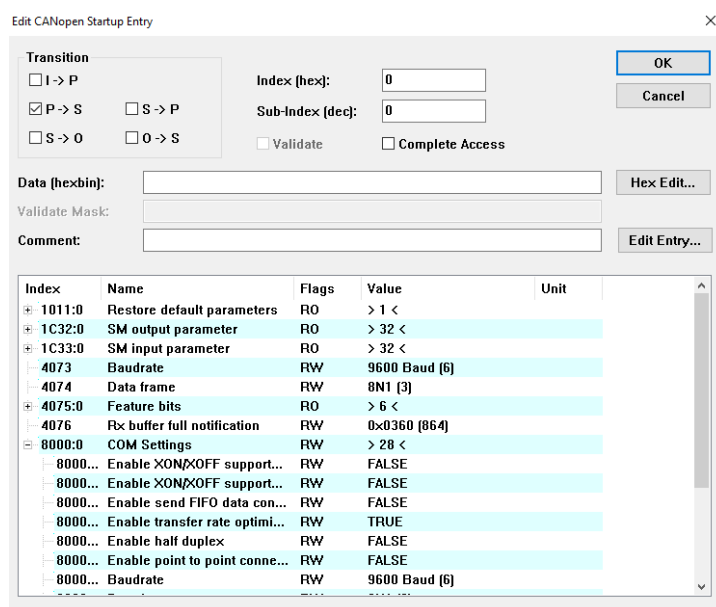
3.3.2 Konfiguracja poprzez zakładkę CoE – Startup

W przypadku konfiguracji poprzez zakładkę CoE – Online edytujemy bezpośrednio wartości na karcie docelowej w trybie online. Wadą tego rozwiązania jest możliwość utracenia konfiguracji w przypadku uszkodzenia danej karty podczas działania. W momencie wymiany karty poprzednia konfiguracja zostaje utracona.

Dlatego też w tym przykładzie zostanie pokazana konfiguracja poprzez zakładkę CoE – Startup . W przypadku konfiguracji poprzez zakładkę CoE – Startup parametry konfiguracyjne zostają przepisane z pamięci sterownika i wprowadzone na karcie docelowej w trakcie przejścia w tryb Run.



W celu wprowadzenia nowej konfiguracji danego parametru należy w zakładce *Startup* kliknąć przycisk *New...*



W nowo otwartym oknie rozwijamy zakładkę *COM Settings* z indeksem 8000. Zawarte są w niej wszystkie ustawienia parametrów komunikacyjnych.

Index	Name	Flags	Value	Unit
8000:0	COM Settings	RW	> 28 <	
8000...	Enable XON/XOFF support...	RW	FALSE	
8000...	Enable XON/XOFF support...	RW	FALSE	
8000...	Enable send FIFO data con...	RW	FALSE	
8000...	Enable transfer rate optimi...	RW	TRUE	
8000...	Enable half duplex	RW	FALSE	
8000...	Enable point to point conne...	RW	FALSE	
8000...	Baudrate	RW	115.2 kBaud (10)	
8000...	Data frame	RW	8N1 (3)	
8000...	Rx buffer full notification	RW	0x0360 (864)	
8000...	Explicit baudrate	RW	0x00002580 (9600)	
8000...	Extended data frame	RW	8N1 (3)	

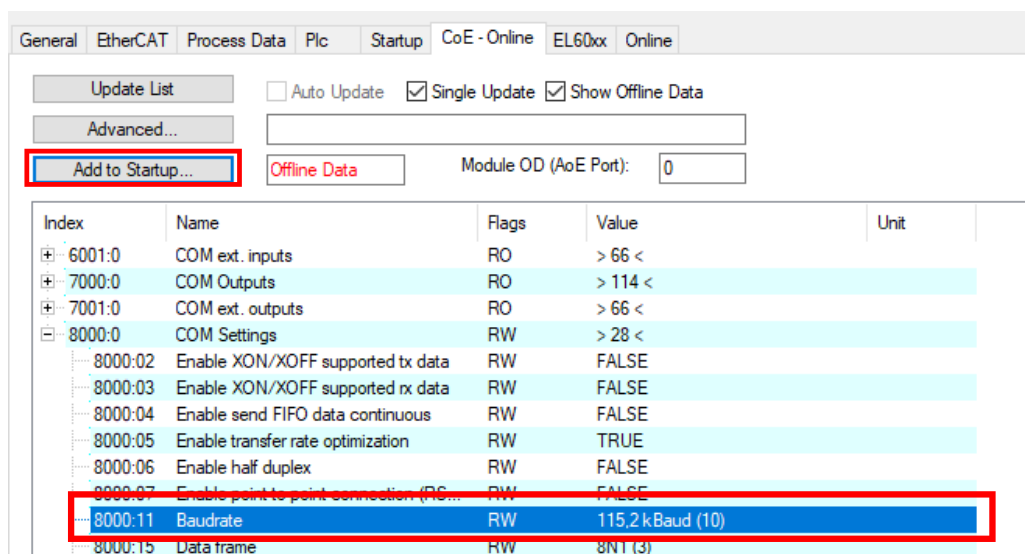
Przykładowo klikając dwukrotnie LPM na parametrze *Baudrate* jesteśmy w stanie edytować wartość prędkości komunikacji poprzez rozsuwany pasek z enumeracyjnymi wartościami prędkości komunikacji. Następnie nowo wybraną wartość prędkości komunikacji zatwierdzamy przyciskiem OK.

Po wprowadzeniu nowej wartości należy, zaznaczając zmieniony przez nas parametr, kliknąć przycisk OK.

Zmiana danego parametru zostanie wówczas zapisana w oknie Startup i zostanie wczytana na kartę w momencie przejścia karty ze stanu Pre-OP do Safe-OP.

Transition	Protocol	Index	Data	Comment
<PS>	CoE	0x1C12:00	0x00 (0)	clear sm pdos (0x1C12)
<PS>	CoE	0x1C13:00	0x00 (0)	clear sm pdos (0x1C13)
<PS>	CoE	0x1C12:01	0x1604 (5636)	download pdo 0x1C12:01 i...
<PS>	CoE	0x1C12:00	0x01 (1)	download pdo 0x1C12 count
<PS>	CoE	0x1C13:01	0x1A04 (6660)	download pdo 0x1C13:01 i...
<PS>	CoE	0x1C13:00	0x01 (1)	download pdo 0x1C13 count
PS	CoE	0x8000:04	115,2 kBaud (10)	Enable send FIFO data co...

Programista ma również możliwość szybkiego przepisania wartości zmiennych z zakładki CoE – Online do zakładki Startup poprzez wybranie zmienionego parametru i kliknięcie przycisku *Add to startup...*



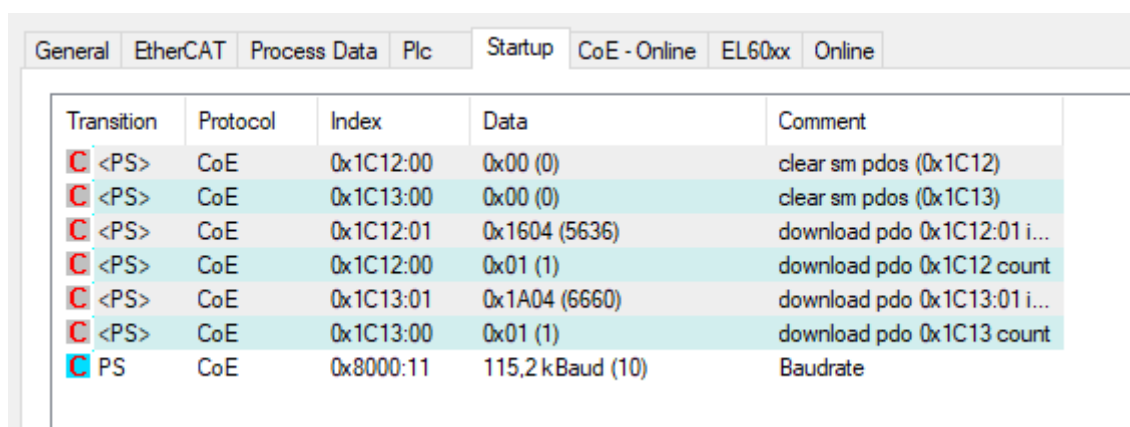
Update List ☐ Auto Update ☒ Single Update ☒ Show Offline Data

Advanced...

Add to Startup... Offline Data Module OD (AoE Port):

Index	Name	Flags	Value	Unit
6001:0	COM ext. inputs	RO	> 66 <	
7000:0	COM Outputs	RO	> 114 <	
7001:0	COM ext. outputs	RO	> 66 <	
8000:0	COM Settings	RW	> 28 <	
8000:02	Enable XON/XOFF supported tx data	RW	FALSE	
8000:03	Enable XON/XOFF supported rx data	RW	FALSE	
8000:04	Enable send FIFO data continuous	RW	FALSE	
8000:05	Enable transfer rate optimization	RW	TRUE	
8000:06	Enable half duplex	RW	FALSE	
8000:07	Enable point to point connection (RS...	RW	FALSE	
8000:11	Baudrate	RW	115.2 kBaud (10)	
8000:15	Data frame	RW	8N1 (3)	

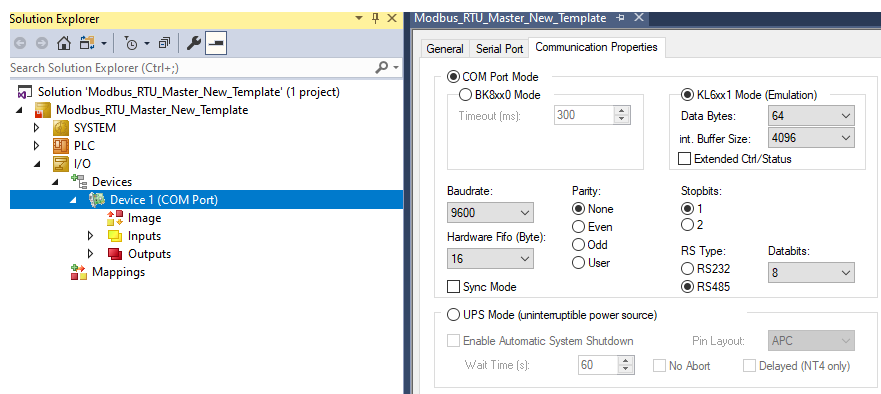
Wybrana konfiguracja danej zmiennej, wraz z określoną jej wartością zostanie przeniesiona na koniec Startup listy.



Transition	Protocol	Index	Data	Comment
<PS>	CoE	0x1C12:00	0x00 (0)	clear sm pdos (0x1C12)
<PS>	CoE	0x1C13:00	0x00 (0)	clear sm pdos (0x1C13)
<PS>	CoE	0x1C12:01	0x1604 (5636)	download pdo 0x1C12:01 i...
<PS>	CoE	0x1C12:00	0x01 (1)	download pdo 0x1C12 count
<PS>	CoE	0x1C13:01	0x1A04 (6660)	download pdo 0x1C13:01 i...
<PS>	CoE	0x1C13:00	0x01 (1)	download pdo 0x1C13 count
PS	CoE	0x8000:11	115,2 kBaud (10)	Baudrate

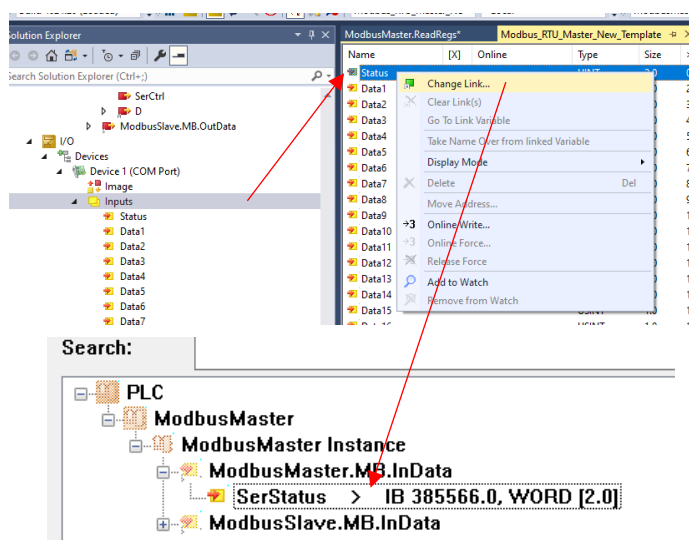
Po takim skonfigurowaniu parametrów komunikacyjnych karty należy już tylko utworzyć połączenie pomiędzy programem PLC a wyprowadzeniami utworzonymi w karcie EL60xx zgodnie z rozdziałem 3.4

3.4 Przykładowe linkowanie zmiennych procesu komunikacyjnego

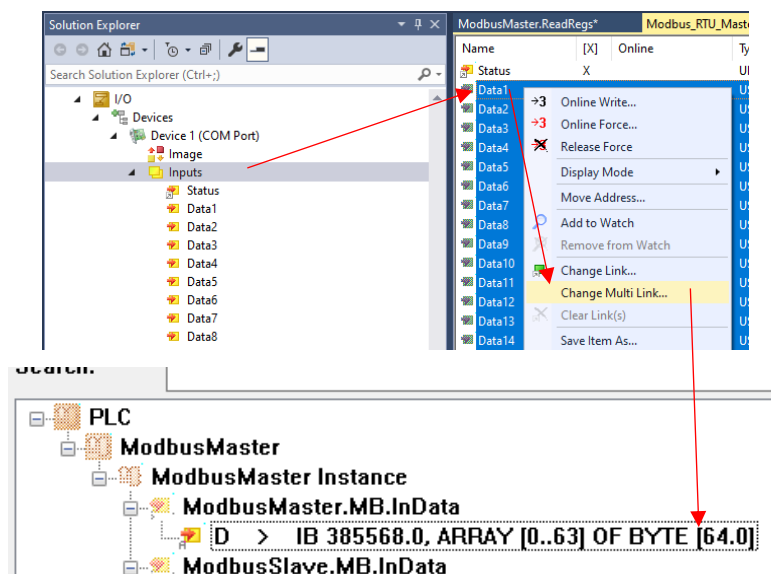


Posiadając utworzony w programie blok ModbusRtuMaster_xxx lub ModbusRtuSlave_xxx mamy możliwość podlinkowania Process Image z programu PLC do COM Port'u. Połączenie należy utworzyć w sposób następujący :

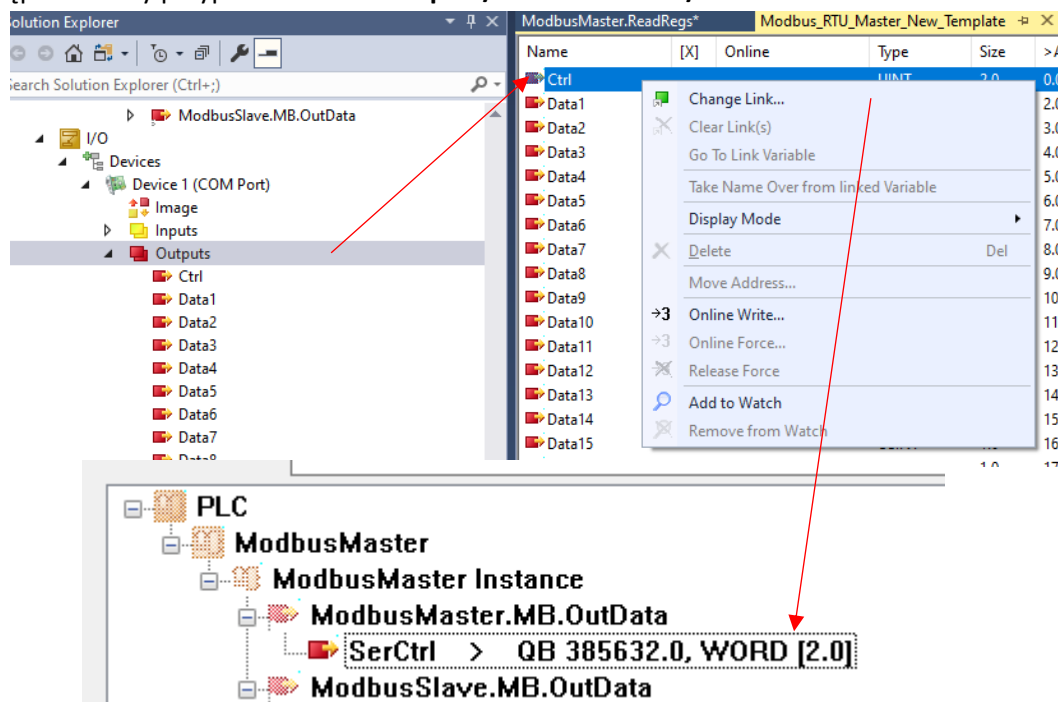
1. Połączenie elementu Input/Status do elementu InData/SerStatus



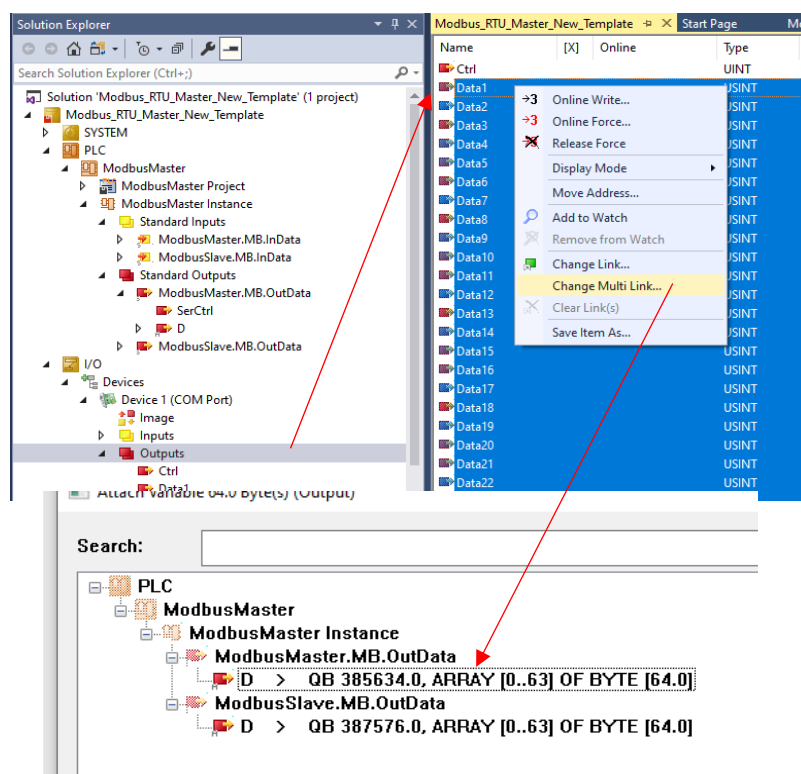
2. Połączenie elementów od Input/Data1 do Input/Data64 za pomocą skrótu *Change multi-link* do elementu InData/D



3. Następnie należy przypisać element **Outputs/Ctrl** do **OutData/SerCtrl**



4. Połączenie zmiennych od **Outputs/Data1** do **Outputs/Data64** poprzez *Change multi-link* do **OutData/D**

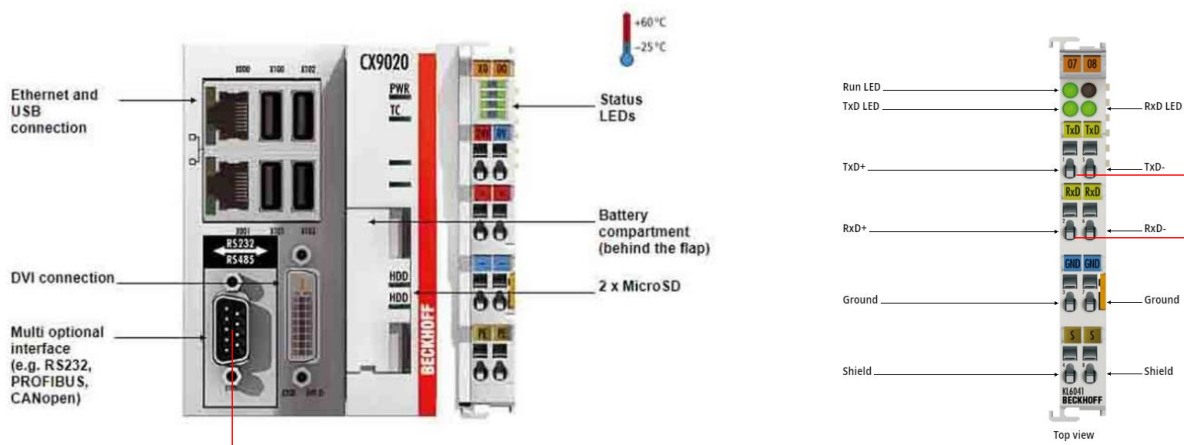


Po utworzeniu połączeń, a następnie po przeaktywaniu nowej konfiguracji strona hardware'owa jest przygotowana do działania. W przypadku jeżeli w projekcie PLC jest utworzona zmienna typu ModbusRtuMaster_PcCom lub ModbusRtuSlave_PcCom, lecz nie widać jej w elemencie do podlinkowania należy przebudować program PLC. Niezależnie czy wykorzystujemy karty z serii KL6xxx czy karty EL6xxx linkowanie zmiennych komunikacyjnych wygląda identycznie.

4 Quick Start - Przykładowy projekt

Stanowisko testowe:

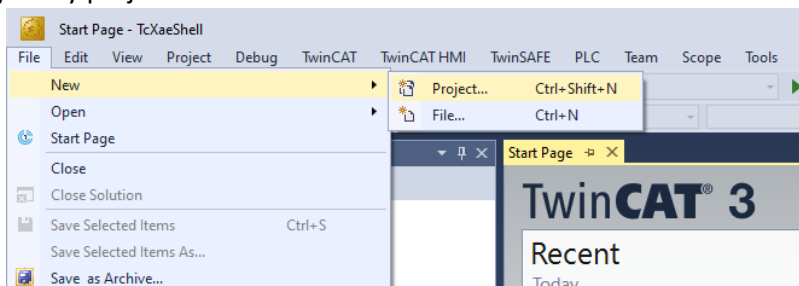
- CX9020 – N031 – Basic CPU with RS485 interface, D-sub socket, 9-pin
- KL6041 - Bus Terminal, 1-channel communication interface, serial, RS422/RS485



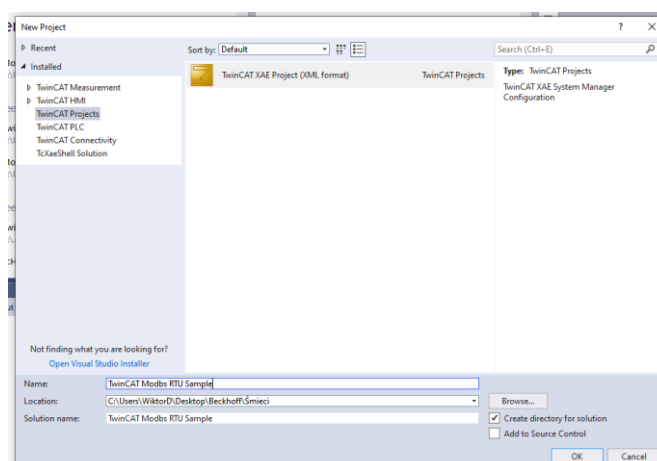
Zakładamy

4.1 Utworzenie nowego projektu

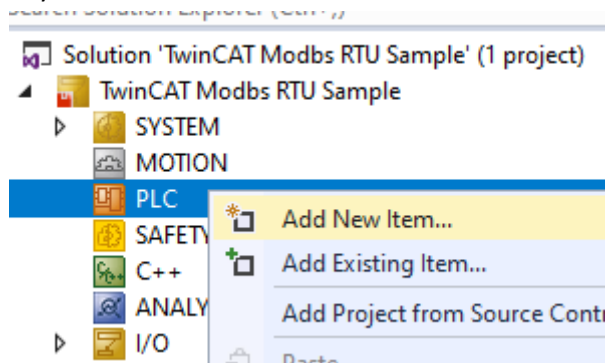
1. Tworzymy nowy projekt



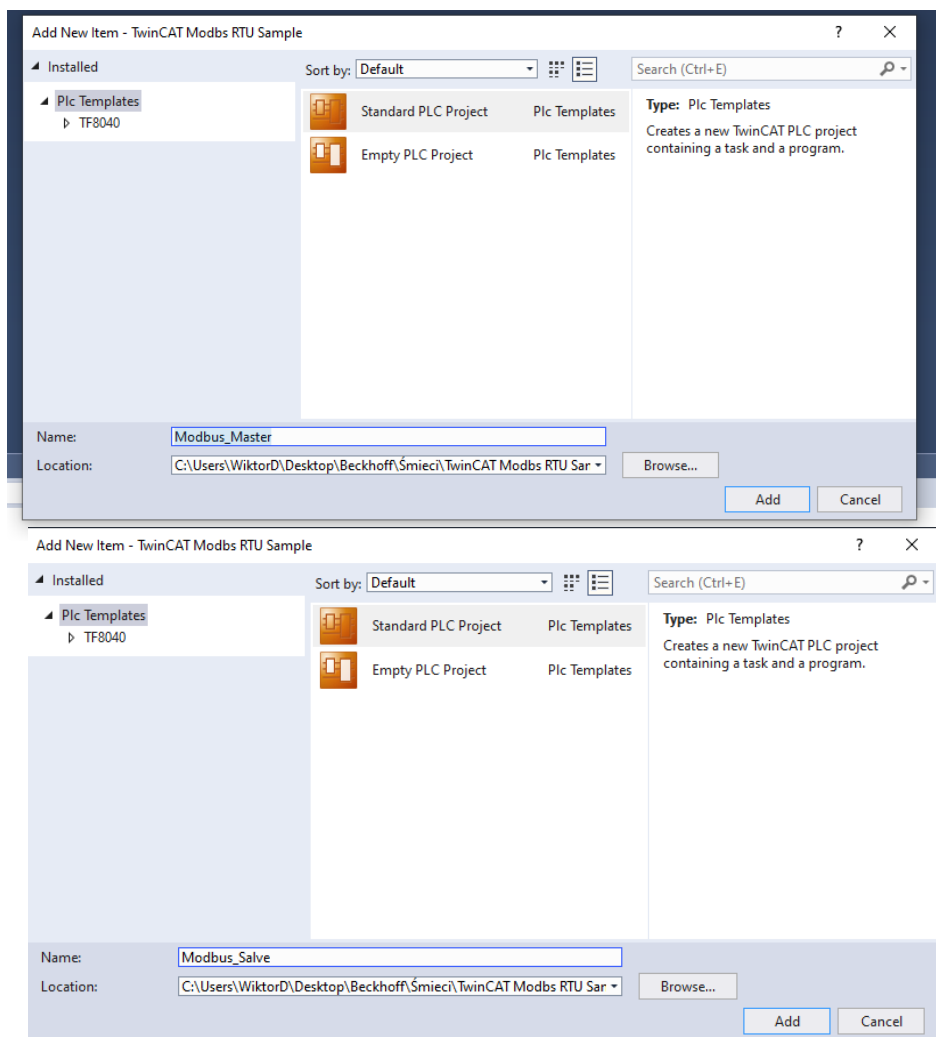
2. Wprowadzamy nazwę projektu PLC



3. Tworzymy 2 nowe projekty PLC



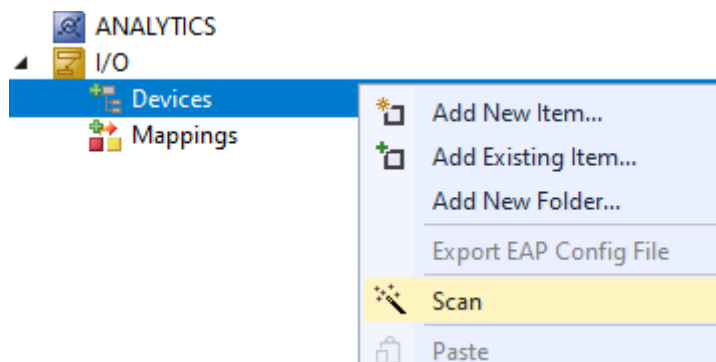
4. Wprowadzamy nazwę Projektu Mastera oraz Projektu Slave'a Modbusowego



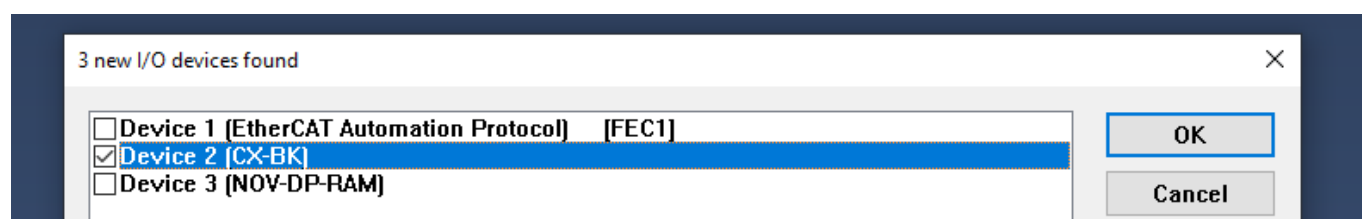
5. Łączymy się ze sterownikiem – instrukcja połączenia ze sterownikiem pod adresem ftp://Poland@transfer.beckhoff.com/Pomoc/TC3/TC3_Podstawy.pdf

4.2 Konfiguracja przykładowego obiektu

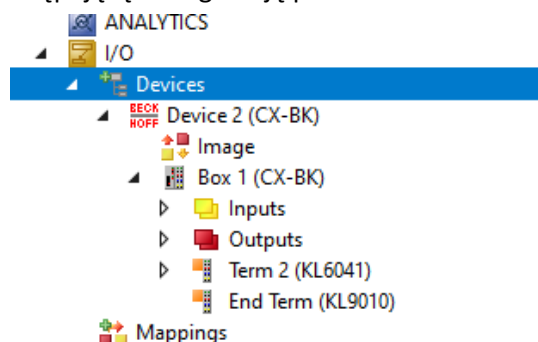
1. Skanujemy konfigurację sprzętową



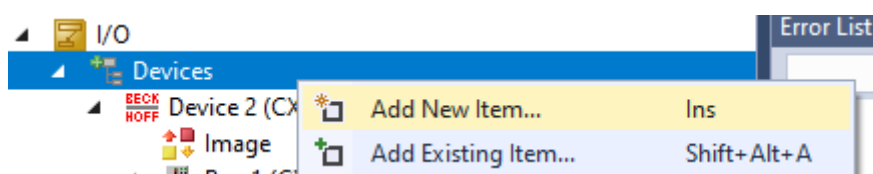
2. W przypadku karty na magistrali K-BUS wybieramy skanowanie portu CX-BK



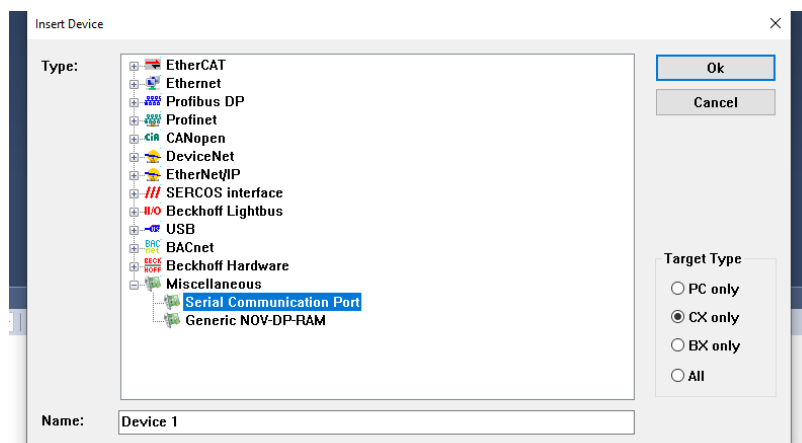
3. W efekcie otrzymujemy następującą konfigurację po skanowaniu



4. W celu dodania portu COM w konfiguracji I/O klikamy PPM na obiekt Devices i wybieramy Add new item...

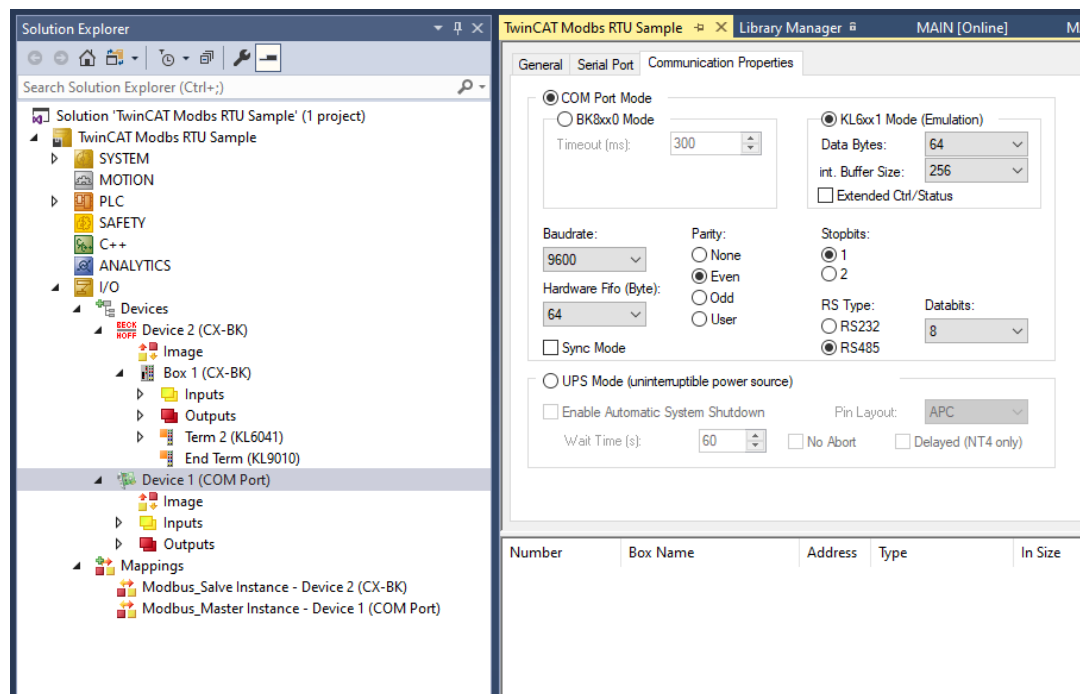


5. Dodajemy element Serial Communication Port



6. Ustalamy parametry komunikacji

Baudrate	Parity	Stopbits	RSType	Databits
9600	Even	1	RS485	8

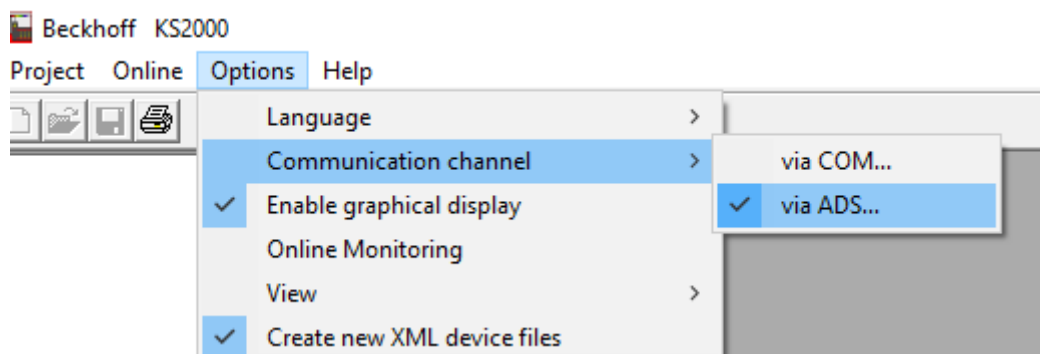


7. Wprowadzamy wybraną konfigurację na porcie COM w zakładce *Communication Properties*
8. W celu konfiguracji karty KL6021 uruchamiamy środowisko KS2000 (więcej informacji na temat programu na stronie pod [tym](#) adresem)

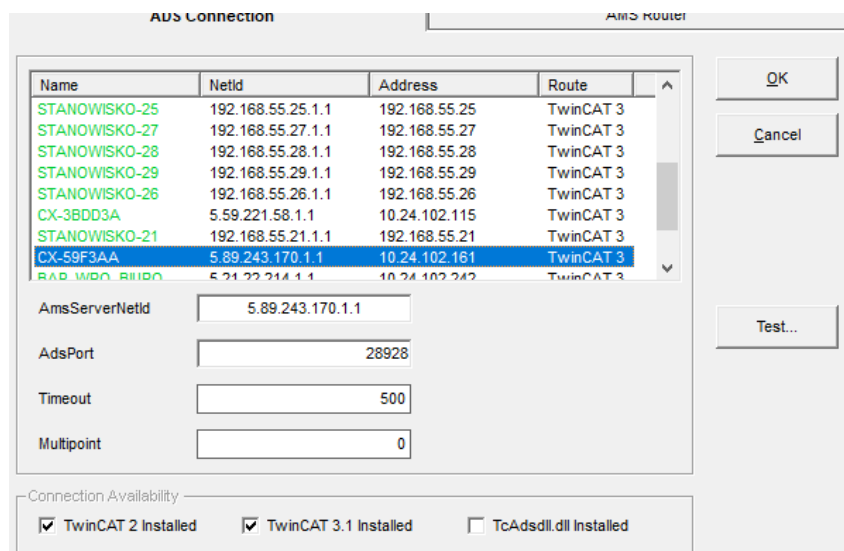


KS2000 v5
Aplikacja

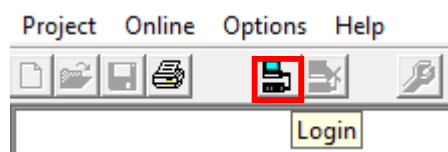
9. Wchodzimy w ustawienia konfiguracji ADS



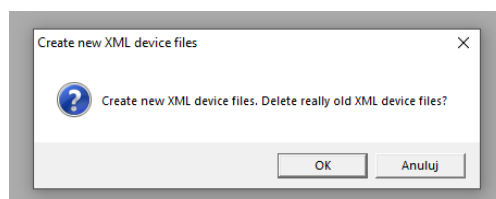
10. Wybieramy docelowy sterownik i zatwierdzamy przyciskiem OK



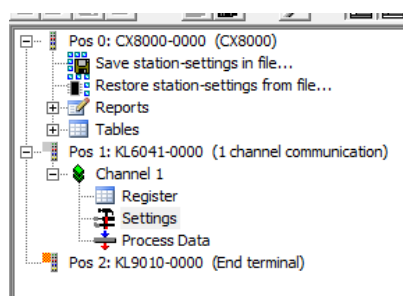
11. Łączymy się ze sterownikiem



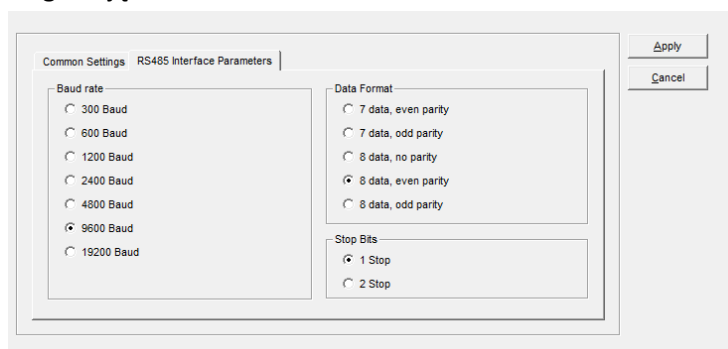
12. Tworzymy nowe zestawienie konfiguracji XML klikając przycisk OK



13. Wchodzimy w zakładkę *Settings* na karcie docelowej



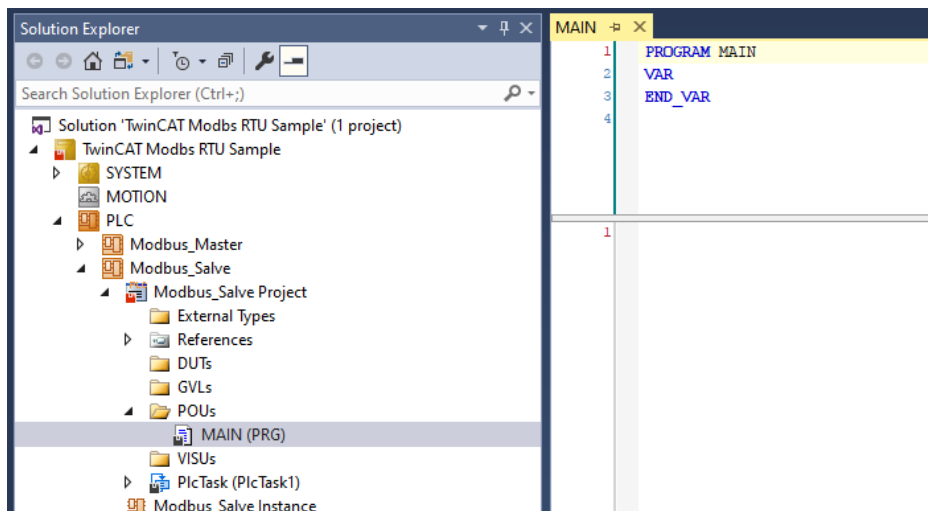
14. Wprowadzamy konfigurację w zakładce RS485 Interface Parameters i zatwierdzamy przyciskiem Apply



15. Wstępna konfiguracja sprzętowa zakończona

4.3 Utworzenie obiektu typu Modbus Slave

1. Wchodzimy we wcześniej utworzony program Main w projekcie PLC Modbus_Slave



2. Tworzymy zmienne obszaru wejść, wyjść i pamięci wewnętrznej

```
PROGRAM MAIN
VAR
    arrInput      : ARRAY[0..20] OF WORD;
    arrOutput     : ARRAY[0..20] OF WORD;
    arrMemory     : ARRAY[0..20] OF WORD;
END_VAR
```

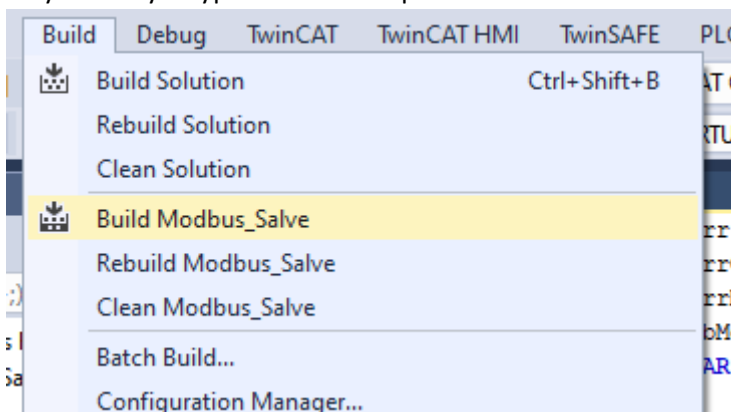
3. Jako że chcemy postawić obiekt typu Modbus RTU Slave na karcie KL6041 tworzymy blok funkcyjny z buforem 22 bajt;

```
PROGRAM MAIN
VAR
    arrInput      : ARRAY[0..20] OF WORD;
    arrOutput     : ARRAY[0..20] OF WORD;
    arrMemory     : ARRAY[0..20] OF WORD;
    fbModbusSlave : ModbusRtuSlave_KL6x22B;
END_VAR
```

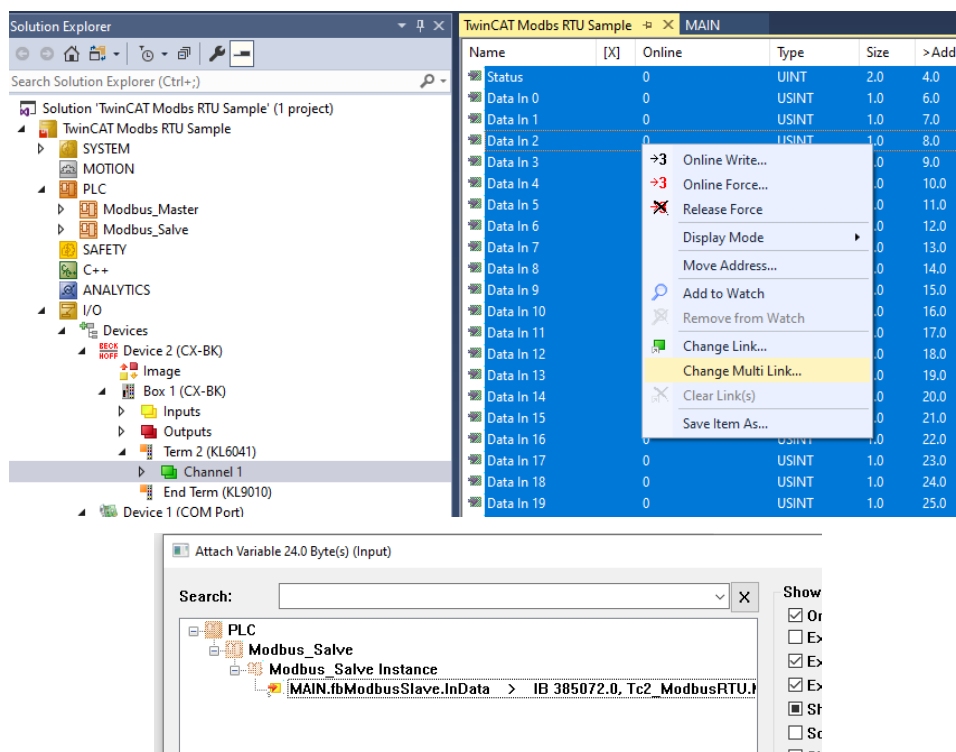
4. Konfigurujemy wyprowadzenia i uruchamiamy blok fbModbusSlave

```
fbModbusSlave(UnitID := 1,
               AdrInputs := ADR(arrInput),
               SizeInputBytes := SIZEOF(arrInput),
               AdrOutputs := ADR(arrOutput),
               SizeOutputBytes := SIZEOF(arrOutput),
               AdrMemory := ADR(arrMemory),
               SizeMemoryBytes := SIZEOF(arrMemory));
```

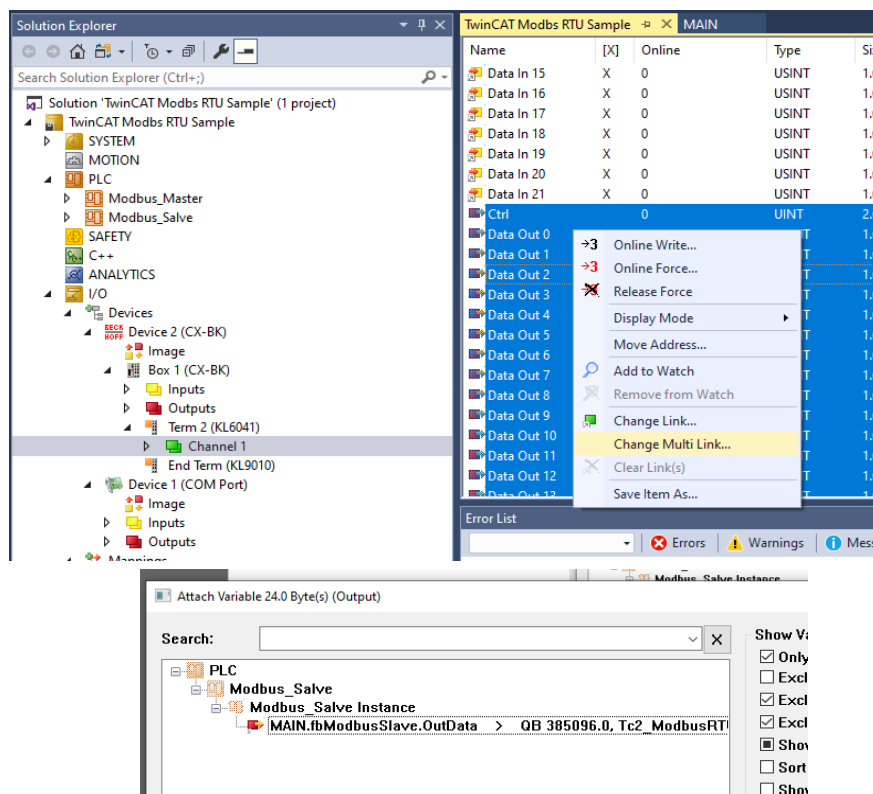
5. Budujemy projekt aby utworzyć wyprowadzenia w pliku TMC



6. Tworzymy połączenia pomiędzy wyprowadzeniami karty KL6041 a programem Modbus_Slave
- a. Tworzymy połączenie zmiennych wejściowych w konfiguracji karty KL6041 za pomocą Change Multilink



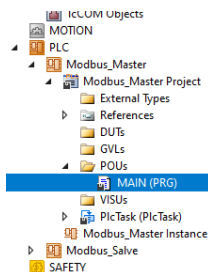
- b. Tworzymy połączenie zmiennych wyjściowych w konfiguracji karty KL6041 za pomocą Change Multilink. Niezależnie czy będziemy jednocześnie linkowali parametr CTRL + Data Out czy osobno zgodnie z rozdziałem 3.4 działanie programu będzie identyczne.



7. Aktywujemy konfigurację i uruchamiamy program
8. Obiekt Slave Modbusa RTU z identyfikatorem 1 został tym samym utworzony.

4.4 Utworzenie obiektu typu Modbus Master + przykład komunikacji

1. Otwieramy program MAIN w projekcie PLC Mastera Modbus'a RTU



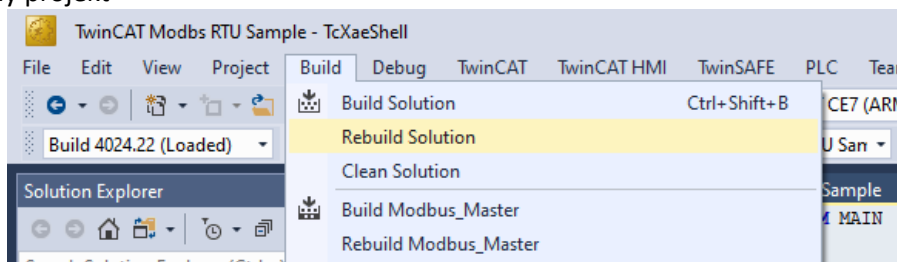
2. Piszemy przykładowy program

```
PROGRAM MAIN
VAR
    fbModbusMaster      :      ModbusRtuMaster_PcCOM;
    bReadRegs           :      BOOL;;
    bError              :      BOOL;
    arrInputs            :      ARRAY[0..9] OF WORD;
END_VAR
```

```
IF bReadRegs THEN
    fbModbusMaster.ReadInputRegs(
        UnitID:=1 ,                                (*Adres Slave*)
        Quantity:=5 ,                               (*Ilość czytanych rejestrów w słowach (ilość rejestrów 16bit)*)
        MBAddr:=16#0 ,                               (*Adres modbusowy - Input 16#0*)
        cbLength:= 10,                               (*Ilość rejestrów 8bit (Quantity*2)*)
        pMemoryAddr:=ADR(arrInputs) ,                (*Adres zmiennej do której zwracany jest wynik zapytania *)
        Execute:= TRUE,
        timeout := T#1s,
        Error=>bError );

    IF NOT fbModbusMaster.BUSY AND NOT bError THEN
        bReadRegs := FALSE;
        fbModbusMaster.ReadInputRegs(Execute:= FALSE);
    ELSIF bError THEN
        bReadRegs := FALSE;
        fbModbusMaster.ReadInputRegs(Execute:= FALSE);
    END_IF
END_IF
```

3. Przebudowujemy projekt



4. Linkujemy wejścia oraz wyjścia

Solution Explorer

- Solution 'TwinCAT Modbus RTU Sample' (1 project)
 - TwinCAT Modbus RTU Sample
 - SYSTEM
 - MOTION
 - PLC
 - SAFETY
 - C++
 - ANALYTICS
 - I/O
 - Devices
 - Device 2 (CX-BK)
 - Image
 - Box 1 (CX-BK)
 - Inputs
 - Outputs
 - Term 2 (KL6041)
 - End Term (KL9010)
 - Device 1 (COM Port)
 - Image
 - Inputs
 - Outputs
 - Mappings
 - Modbus_Slave Instance - Device 2 (CX-BK)
 - Modbus_Master Instance - Device 1 (COM Port)

Library Manager

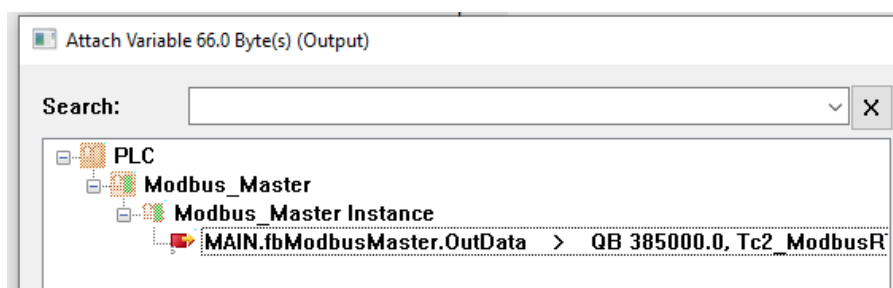
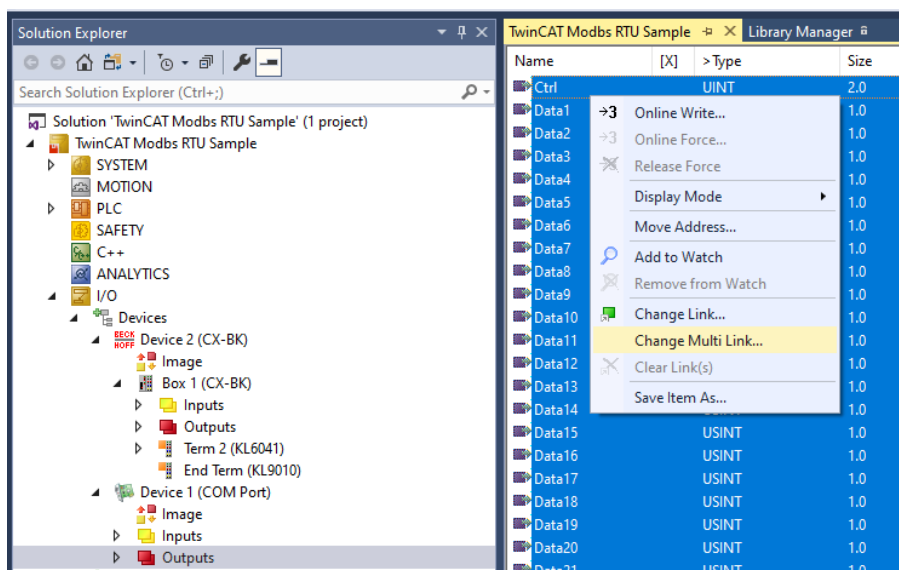
Name	[X]	Type	Size	>Ad...	In/Out
Data32	X	USINT	1.0	33.0	Input
Data33	X	USINT	1.0	34.0	Input
Data34	X	USINT	1.0	35.0	Input
Data35	X	USINT	1.0	36.0	Input
Data36	X	USINT	1.0	37.0	Input
Data37	X	USINT	1.0	38.0	Input
Data38	X	USINT	1.0	39.0	Input
Data39	X	USINT	1.0	40.0	Input
Data40	X	USINT	1.0	41.0	Input
Data41	X	USINT	1.0	42.0	Input
Data42	X	USINT	1.0	43.0	Input
Data43	X	USINT	1.0	44.0	Input
Data44	X	USINT	1.0	45.0	Input
Data45	X	USINT	1.0	46.0	Input
Data46	X	USINT	1.0	47.0	Input
Data47	X	USINT	1.0	48.0	Input
Data48	X	USINT	1.0	49.0	Input
Data49	X	USINT	1.0	50.0	Input
Data50	X	USINT	1.0	51.0	Input
Data51	X	USINT	1.0	52.0	Input
Data52	X	USINT	1.0	53.0	Input
Data53	X	USINT	1.0	54.0	Input
Data54	X	USINT	1.0	55.0	Input
Data55	X	USINT	1.0	56.0	Input
Data56	X	USINT	1.0	57.0	Input
Data57	X	USINT	1.0	58.0	Input
Data58	X	USINT	1.0	59.0	Input
Data59	X	USINT	1.0	60.0	Input
Data60	X	USINT	1.0	61.0	Input
Data61	X	USINT	1.0	62.0	Input
Data62	X	USINT	1.0	63.0	Input
Data63	X	USINT	1.0	64.0	Input
Data64	X	USINT	1.0	65.0	Input
ExtVoltageOk		BIT	0.1	66.0	Input

Attach Variable 66.0 Byte(s) (Input)

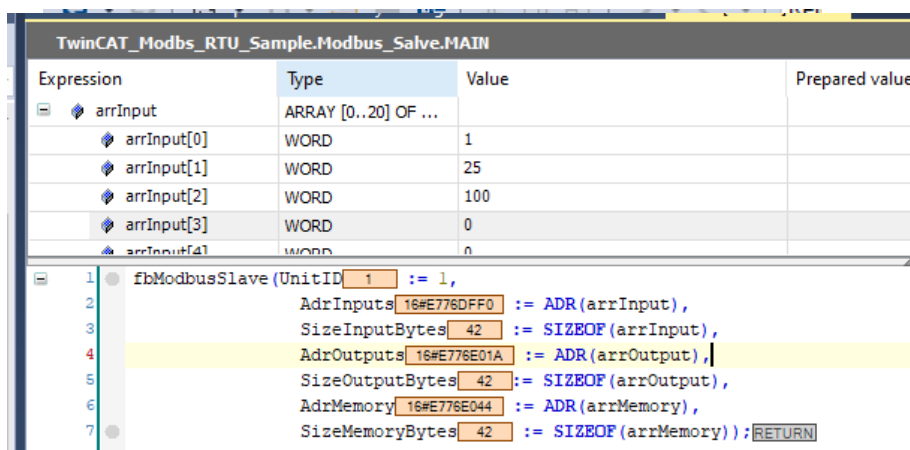
Search:

PLC

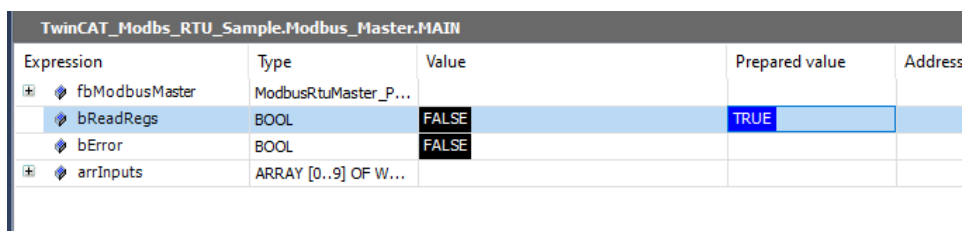
- Modbus_Master
 - Modbus_Master Instance
 - MAIN.fbModbusMaster.InData > IB 384934.0, Tc2 ModbusRTU



5. Aktywujemy konfigurację
6. Logujemy się jednym i drugim programem PLC i je uruchamiamy
7. Zmieniamy wartości w trybie online w programie PLC Slave'a



8. W programie mastera zmieniamy stan wartości bReadRegs w trybie online do stanu true



9. Sprawdzamy pobrane dane

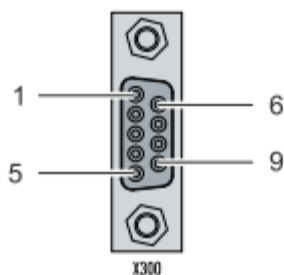
Expression	Type	Value	Pre
fbModbusMaster	ModbusRtuMaster_P...		
bReadRegs	BOOL	FALSE	
bError	BOOL	FALSE	
arrInputs	ARRAY [0..9] OF W...		
arrInputs[0]	WORD	1	
arrInputs[1]	WORD	25	
arrInputs[2]	WORD	100	
arrInputs[3]	WORD	0	

W bazie danych posiadamy również przykładowy projekt zawierający konfigurację Mastera oraz Slave'a dla wszystkich funkcji Modbus'a RTU. W celu otrzymania projektu prosimy o kontakt pod adresem support@beckhoff.pl.

5 Tips & Tricks

5.1 Wyprowadzenia portu DB9 i przykład połączenia

W zależności od sterownika możemy posiadać różne konfiguracje wyprowadzeń na porcie DB9. W przypadku sterowników serii CX9020 z rozszerzeniem na port COM wyprowadzenie na RS485 prezentuje się następująco

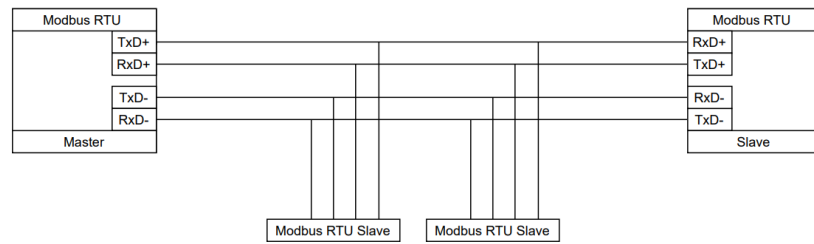


PIN	Signal	Type	Description
2	TxD+	Data-Out +	Transmit 422
3	RxD+	Data-In +	Receive 422
5	GND	Ground	Ground
6	VCC	VCC	+5 V
7	TxD-	Data-Out -	Transmit 422
8	RxD-	Data-In -	Receive 422

W przypadku sterowników CX8180 oraz CX7080 wyprowadzenie DB9 jest przygotowane do konfiguracji magistrali RS232 lub RS485 w Half-Duplex'ie :

PIN	Meaning	Description	Signal
1	RS485	(+)	A
2	RxD (RS232)	Signal in	Receive Data
3	TxD (RS232)	Signal out	Transmit Data
4	+ 5 V	+	Vcc
5	GND	Ground	Ground
6	RS485	(-)	B
7	RTS (RS232)	Signal out	Request to Send
8	CTS (RS232)	Signal in	Clear to Send
9	GND	Ground	Ground

5.1.1 Realizacja połączenia Full-Duplex



5.1.2 Realizacja połączenia Half-Duplex

