BECKHOFF

Zastosowanie biblioteki Serial Communication

Poziom trudności: średniozaawansowany

Wersja dokumentacji: 1.1

Aktualizacja: 15.09.2015

Beckhoff Automation Sp. z o. o.

Spis treści

1. Wstęp

Trzy powszechnie stosowane warstwy fizyczne komunikacji szeregowej to RS232, RS422 oraz RS485. Na każdej z nich można uruchomić dowolną warstwę aplikacyjną – protokół komunikacyjny. Najczęściej spotykanym protokołem komunikacyjnym komunikacji szeregowej jest Modbus RTU. Jednak czasem spotykamy urządzenia, które tego standardu komunikacyjnego nie obsługują i zmuszeni jesteśmy dostosować się do protokołu komunikacyjnego danego producenta. Przykładami takich urządzeń są falowniki, centrale alarmowe, stacje pogodowe. Musimy zgodnie z dokumentacją napisać własną warstwę aplikacyjną. Do tego służy biblioteka Serial Communication.

Biblioteka umożliwia przesyłanie pojedynczych bajtów, ciągów znaków (zmiennych STRING) i dowolnych danych (przekazywanych poprzez wskaźniki).

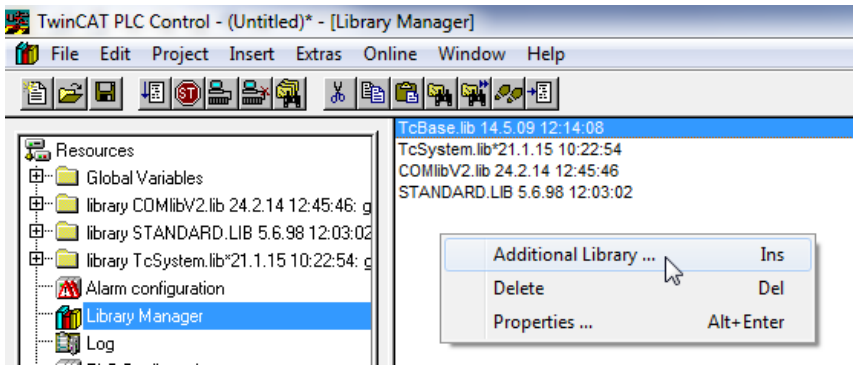
W przykładzie, prezentowanym w niniejszej dokumentacji, będzie modelowana komunikacja typu master - slave. Pokazane będę dwa sposoby wymiany danych – poprzez zmienne typu STRING i poprzez ciągi danych (umieszczone w tablicach zmiennych typu BYTE).

Dokumentacja ta nie zastępuje informacji zawartych w dokumentacji do urządzeń i biblioteki, stanowi tylko krótki opis przykładów. Przykłady i opis zostały stworzone w oparciu o TwinCAT 2.11 B2249. Podobnie konfigurację i program należy wykonać w TwinCAT 3.

2. Wymagania programowe

Potrzebny nam będzie TwinCAT w wersji PLC lub wyższej oraz biblioteka Serial Communication (TS6340-000x). Po zainstalowaniu biblioteki pojawią się w folderze bibliotek pliki dla odpowiednich obiektów docelowych: COMlibV2.lib, COMlibV2.lbx, COMlibV2.lb6.

Bibliotekę dodajemy w TwinCAT PLC Control, zakładka Resources\Library Manager. Tu klikamy PPM i wybieramy opcję Additional Library…

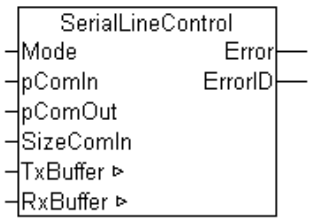


Dla PC lub CX z listy bibliotek wybieramy COMlibV2.lib. Pozostałe biblioteki dołączą się automatycznie.

3. Program PLC

3.1 SerialLineControl

Ten blok funkcyjny jest częścią wspólną każdej aplikacji. SerialLineControl służy przesyłaniu danych z programu PLC do fizycznego buforu portu COM. Dzięki takiemu podejściu bloki wysyłania i odbierania danych są takie same, niezależnie od tego jaki rodzaj portu COM jest fizycznie używany. Rodzaj portu określa zmienna Mode typu ComSerialLineMode_t. Najczęściej wybierane są wartości SERIALLINEMODE_KL6_22B_STANDARD (dla modułów w trybie 22B) lub SERIALLINEMODE_PC_COM_PORT (dla portów COM).



Po wybraniu odpowiedniego typu portu, trzeba podłączyć zmienne zlinkowane z wejściami i wyjściami portu COM w programie TwinCAT System Manager. Zmienne te podpinamy pod 3 wejścia: pComIn, pComOut, SizeComIn. pComIn i pComOut to pointery wskazujące na adres w pamięci, SizeComIn pokazuje jaki jest rozmiar tej zmiennej. Takie rozwiązanie pozwala na wykorzystanie jednego, uniwersalnego bloku funkcyjnego do zmiennych różnych rozmiarów. Zaletą takiego podejścia jest elastyczność. Jeśli w przyszłości będziemy chcieli wykorzystać nasz kod w innej aplikacji, w której jest port COM innego typu, to zmiany ograniczają się do minimum. Przykład przejścia z różnic między trybem 22B a PC COM:

Deklaracja zmiennych		
fbSerialLineControl	SerialLineControl	
COMInData AT %I*	KL6inData22B	PcComInData
COMOutData AT %Q*	KL6outData22B	PcComOutData
Kod programu		
Mode	SERIALLINEMODE_KL6_22B_STANDARD	SERIALLINEMODE_PC_COM_PORT
pComIn	ADR(COMInData)	
pComOut	ADR(COMOutData)	
SizeComIn	SIZEOF(COMInData)	

Zmienne RxBuffer i TxBuffer (obie typu ComBuffer) są to bufory danych, łączone z blokami funkcyjnymi komend wysyłania i odbierania.

Dodatkowe informacje o tym bloku znajdują się w dalszej części dokumentacji.

3.2 Wymiana danych – rola urządzenia Master i Slave

Program szablonowy realizuje komunikację RS typu pytanie – odpowiedź. Master na rozkaz wysyła zapytanie i oczekuje na odpowiedź. Odpowiedź jest analizowana i wracamy do początku. Slave czeka na zapytanie, gdy je otrzyma rozpoczyna analizę i wysyła odpowiednią odpowiedź. Po wszystkim jest powrót do oczekiwania. W przypadku wystąpienia błędu jest on zliczony i zapamiętany. Po określonym czasie następuje restart cyklu i powrót do kroku początkowego.

Kod szablonowych programów napisany jest w języku Structured Text (ST), dla zwiększenia czytelności kroki zostały umieszczone w instrukcji CASE. Blok odbierania danych (ReceiveString lub ReceiveData) znajduje się poza instrukcją CASE, ponieważ odbiór musi być aktywny cały czas. Blok wysyłania danych (SendString lub SendData) wykonywany jest jedynie w kroku, w którym dane są wysyłane. Kody podprogramów wymiany danych i zmiennych typu STRING są do siebie podobne.

Komunikację za pomocą zmiennych typu STRING wykorzystuje się np. do komunikacji z modemami GSM, gdzie do i z urządzenia wysyłane są komendy AT (Hayes AT).

Komunikację za pomocą dowolnego typu danych wykorzystuje się najczęściej do stworzenia komunikacji z falownikiem czy stacją pogodową, gdzie producent opisuje poszczególne bajty ramki komunikacyjnej pytania i odpowiedzi, np.:

W naszym przykładzie komenda pytająca składa się z 5 bajtów:

1	2	3	4	5
Start		Adres	Rozkaz	CRC

Odpowiedź składa się z 10 bajtów:

1	2	3	4	5	6	7	8	9	10
Start		Adres	Rozkaz	Typ	Dane				CRC

3.3 Strona Master

CASE iSate OF

0: Oczekujemy na sygnał do wysłania komendy – zmienna bRead. Gdy go otrzymamy to czyszczona jest zmienna odczytana InData i wstawiana jest wartość do zmiennej wysyłanej OutData. W przykładzie zmienna bRead przyjmuje wartość TRUE co 1s, ponieważ podpięta jest do wyjścia timera fbTimer (linia 15). Można to oczywiście zmienić.

10: Wysłanie ramki – tu nie należy nic modyfikować. Jeśli wszystko zakończy się sukcesem, to idziemy do kroku 20. Gdyby operacja zakończyła się niepowodzeniem, to idziemy do kroku 10000.

20: W tym korku czekamy na odpowiedź urządzenia Slave. Odczytane dane przepisujemy do zmiennej InData. Gdyby operacja zakończyła się niepowodzeniem, to idziemy do kroku 10000.

30: Jest to krok analizy danych – należy zmodyfikować go wedle uznania.

10000: Krok obsługi błędu. Tu również można dokonać modyfikacji. W przykładzie po 5s jest wykonywany reset – zmienne wejścia, wyjścia i kod błędu są zerowane, wracamy do kroku pierwszego.

3.4 Strona Slave

Uwaga! W typowej aplikacji odczytującej dane strona slave jest już w urządzeniu wykonana, do nas należy przygotowanie odpowiedniego zapytania ze strony Master. Przykład nasz pokazuje przykładową implementację obu stron.

CASE iSate OF:

0: Nasłuch – oczekujemy na dane. Gdy go otrzymamy to wpisujemy dane do zmiennej InData, dane wyjściowe są czyszczone i przechodzimy do kroku 10. Gdyby operacja zakończyła się niepowodzeniem, to idziemy do kroku 10000.

10: Dekodowanie danych – tu sprawdzamy czy zapytanie było kierowane do nas, czy zgadza się suma kontrolna CRC, jaki rodzaj zapytania dostaliśmy itp. w zależności od tego jakie zapytanie otrzymaliśmy szykujemy odpowiednią odpowiedź. Jeśli zapytanie nie było kierowane do nas, to wracamy do nasłuchu. Tą część należy zmodyfikować zgodnie z aplikacją.

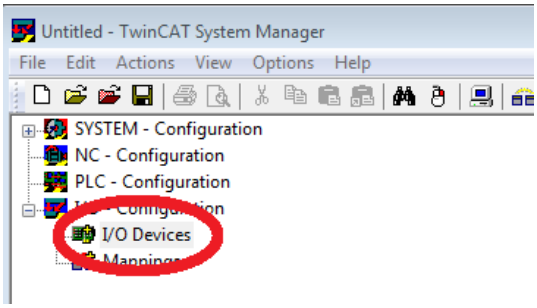
20: Wysłanie wcześniej przygotowanej ramki – tu nie należy nic modyfikować. Jeśli wszystko zakończy się sukcesem, to idziemy do kroku 0 (do nasłuchu). Gdyby operacja zakończyła się niepowodzeniem, to idziemy do kroku 10000.

10000: Krok obsługi błędu. Tu również można dokonać modyfikacji. W przykładzie po 5s jest wykonywany reset – zmienne wejścia, wyjścia i kod błędu są zerowane, wracamy do kroku pierwszego.

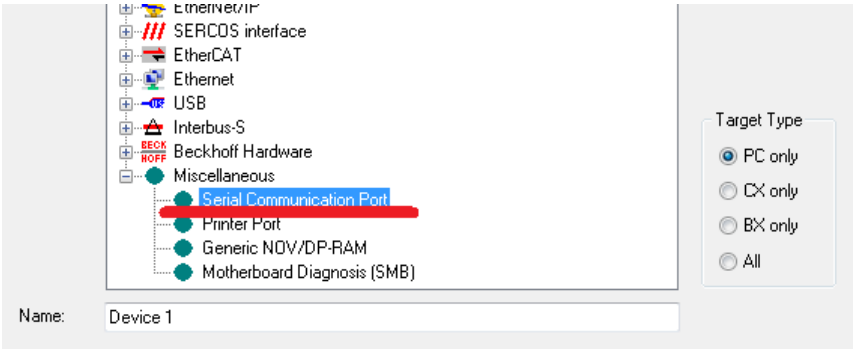
4. Konfiguracja

4.1 Wbudowany port COM

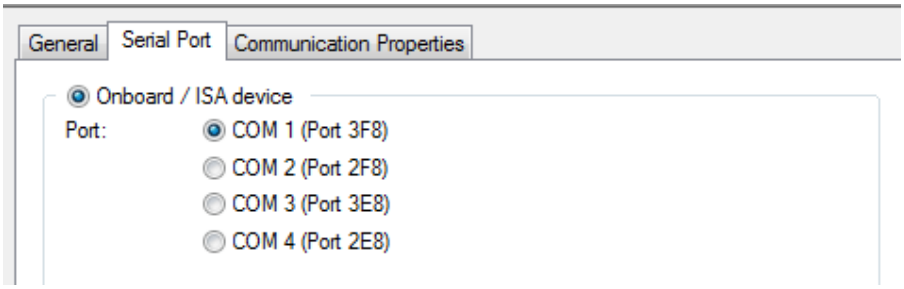
Konfiguracja wbudowanego portu COM odbywa się ręcznie w programie System Manager. W pierwszej kolejności dodajemy port COM. Robimy to klikając w drzewku projektu, prawym przyciskiem myszy, na zakładce **I/O Devices:**



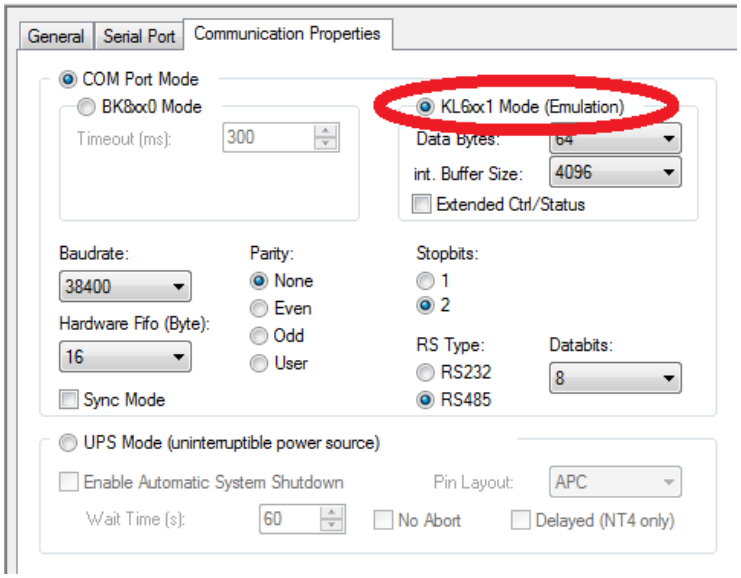
Następnie z listy, która się pojawi, rozwijamy zakładkę **Miscellaneous** i wybieramy **Serial Communication Port**.



Następnie wybieramy z drzewka projektu dodany w ten sposób port COM i przeprowadzamy jego konfigurację. W zakładce **Serial Port** wybieramy, który fizyczny port chcemy używać:

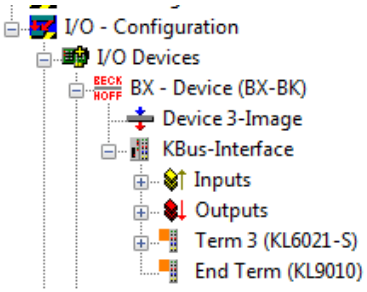


W zakładce **Communication Properties** ustawiamy parametry transmisji. Ważne jest zaznaczenia pola **KL6xx Mode** i ustawienie parametreów transmisji – w obu urządzeniach muszą być takie same:

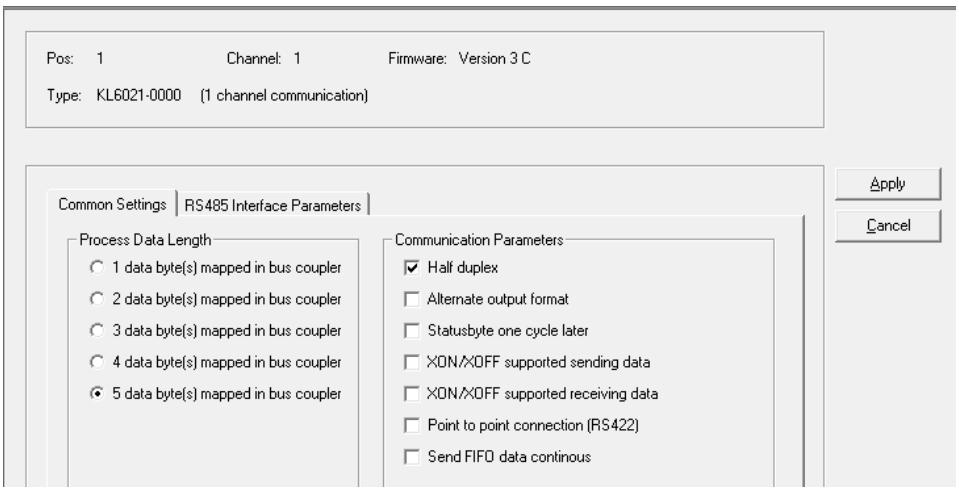


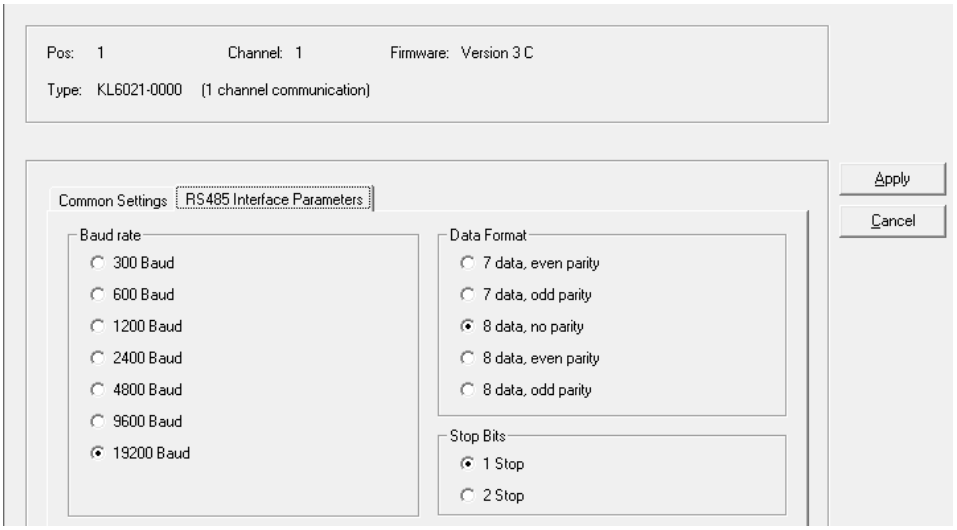
4.2 Moduł magistrali K-Bus

W przypadku komunikacji z wykorzystaniem modułu komunikacji szeregowej magistrali K-Bus (w przykładzie jest to moduł KL6021) najpierw należy wyszukać moduł na magistrali za pomocą programu TwinCAT System Manager.



Następnie najłatwiej parametryzację wykonać w programie KS2000 (jedna zakładka konfiguruje moduł, druga parametry transmisji):



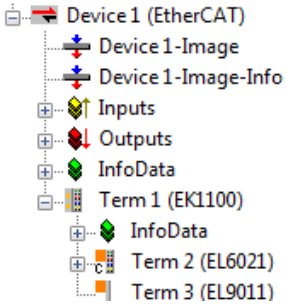


Informacja na temat programu KS2000 można znaleźć w dokumentacji (infosys.beckhoff.com).

Opcjonalnie moduł taki można konfigurować za pomocą bloku funkcyjnego KL6configuration z biblioteki KL6config.

4.3 Moduł magistrali E-Bus

W przypadku komunikacji z wykorzystaniem modułu komunikacji szeregowej magistrali E-Bus (w przykładzie jest to moduł (EL6021) najpierw należy wyszukać moduł na magistrali za pomocą programu TwinCAT System Manager.



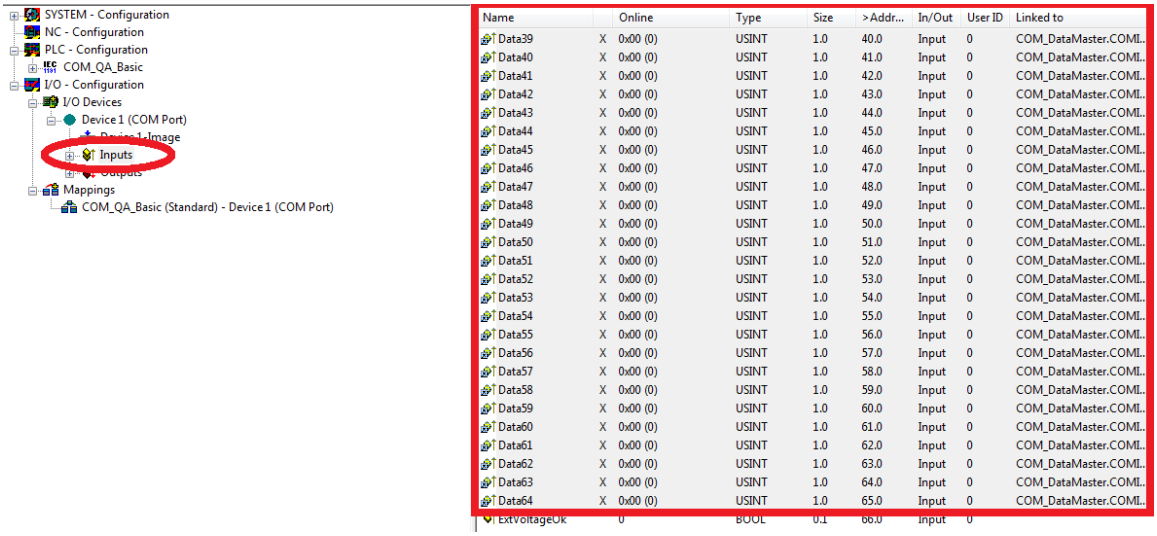
Następnie konfigurujemy go na zakładce CoE – Online:

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	0x00002580 (9600)
8000:1C	Extended data frame	RW	8N1 (3)

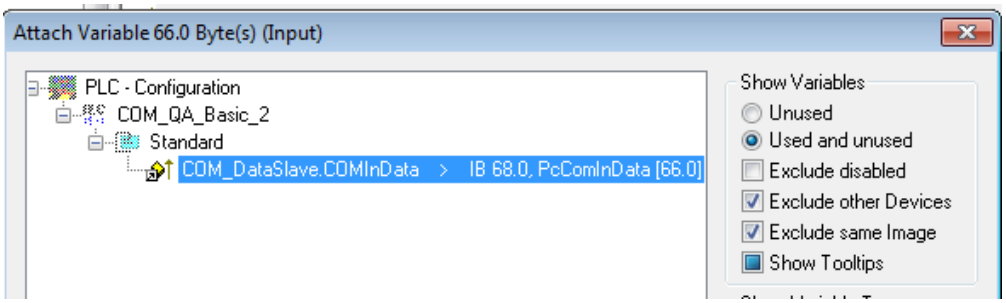
Powyższe parametry są wpisywane do modułu. Można skonfigurować system w ten sposób, że będą wgrywane zawsze przy starcie TwinCATa. Wystarczy odpowiednie parametry dodać na zakładce Startup i aktywować konfigurację. Upraszcza to znacznie procedurę wymiany modułu w przyszłości.

5. Linkowanie zmiennych

W przypadku każdego z czterech omówionych programów (dwa rodzaje master'a i dwa rodzaje slave'a) konieczne jest zlinkowanie zmiennych w konfiguracji sterownika. Robimy to w ten sam sposób dla każdego programu. W drzewku projektu wybieramy port szeregowy, który wcześniej skonfigurowaliśmy. Następnie klikamy zakładkę **Inputs** i zaznaczamy wszystkie zmienne (w przypadku portu PC COM oprócz **ExtVoltageOk**):



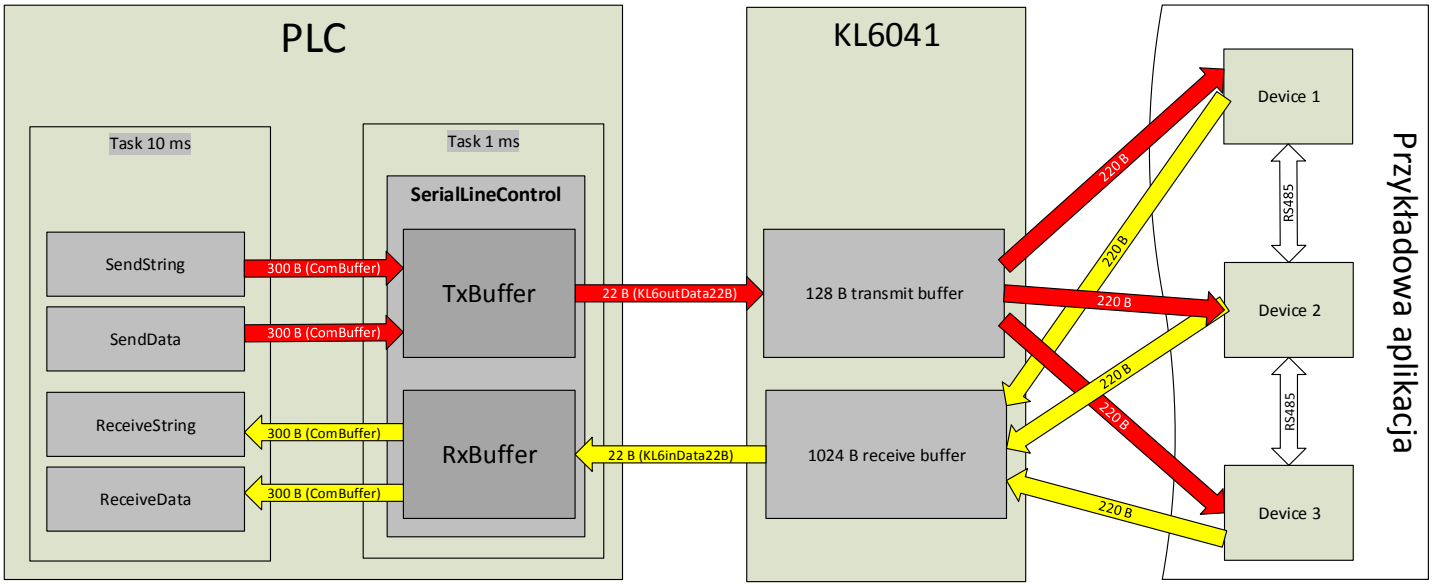
Następnie, klikamy zaznaczenie prawym przyciskiem myszy i wybieramy opcję **Change Multilink**. W oknie, które się pojawi będziemy mieli do wyboru zmienną odpowiedniego typu:



Podobnie robimy w przypadku danych wyjściowych. Klikamy zakładkę **Outputs,** tym razem zaznaczamy jednak wszystkie zmienne. Po kliknięciu Change Multilink pojawi się możliwość zlinkowania zmiennej. Na koniec trzeba aktywować konfigurację.

6. Task

W przypadku modułów pracujących w trybie 5 i 22 bajtowym może być konieczne wywołanie bloku SerialLineControl w szybszym tasku. Jest to wyjaśnione na przykładzie, przedstawionym na poniższym rysunku.



Załóżmy taką sytuację:

1. Task PLC ustawiony na sterowniku wynosi 10 ms.
2. Moduł komunikacyjny to KL6041, wewnętrzny bufor odczytu 1024 bajty, zapisu 128 bajtów.
3. Urządzenia slave wysyłają do modułu co 250 ms jednorazowo ramkę składającą się z 220 bajtów danych.
4. Moduł jest ustawiony w trybie 22 B – oznacza to, że proces image wejść i wyjść wynoszą po 22 bajty – tyle danych jest przesyłanych w jednym cyklu.
5. W jednym cyklu między modułem a programem PLC może być wymienionych 22 B danych – wynika to z proces image modułu.

Prześledźmy proces odczytu danych. Wewnątrz modułu jest bufor odczytu do którego trafiają dane z urządzenia slave. W przypadku KL6041 ma on pojemność 1024 bajtów. Nasze urządzenie slave „wrzuca” do niego 220 bajtów danych. Sterownik odczytuje z tego buforu 22 bajty. Po potwierdzeniu odczytu dane są zdejmowane z buforu (zastosowana jest kolejka FIFO). Proces pobrania danych zajmuje trzy cykle PLC. W naszym przypadku task PLC wynosi 10 ms, więc pobranie z potwierdzeniem 22 bajtów wynosi 30 ms. Zatem pobranie 220 bajtów zajmie 10x30ms = 300 ms. Dane z urządzenia slave są wysyłane co 250 ms, więc nie zdążymy oczyścić buforu przed przyjęciem nowych danych. Na początku niż złego się nie stanie, ponieważ jest miejsce w buforze, a odczyt przez PLC jest realizowany jako FIFO. Ale po pewnym czasie, bufor się przepełni i utracimy część danych. Nie możemy zwiększyć buforu wewnątrz modułu, nie możemy zwiększyć też ilości danych przesyłanych między modułem a programem PLC, jedyne co możemy zmienić to częstotliwość wykonywania programu PLC, czyli task. Jeśli ustawimy go na 2 ms, to czas odczytu danych z bufora zostanie zmniejszony do 60 ms. Szybszy task musi mieć najwyższy priorytet i wystarczy że wywołamy w nim blok SerialLineControl. Blok ten pobierze dane z modułu i umieści w swoim wewnętrznym buforze

(zmienna ComBuffer, rozmiar 300 bajtów, jednakowy dla odbierania i wysyłania). Bufor ten jest współdzielony przez bloki funkcyjne ReceiveString (lub ReceiveData) i SendString (lub SendData), wywoływane w wolniejszym task'u.