Politechnika Poznańska
Wydział Budowy Maszyn
i Zarządzania

Automatyzacja i Nadzorowanie Maszyn
Zajęcia laboratoryjne

Ćwiczenie 9
Podstawy programowania PLC w języku ST

24.04.2018

Poznań 2017

OGÓLNE ZASADY BEZPIECZEŃSTWA
PODCZAS WYKONYWANIA ĆWICZEŃ LABORATORYJNYCH

! Przed przystąpieniem do ćwiczenia należy zapoznać się z instrukcją dydaktyczną.

! Dokonać oględzin urządzeń, przyrządów i przewodów używanych podczas ćwiczenia. W przypadku zauważenia nieprawidłowości lub uszkodzeń bezzwłocznie powiadomić prowadzącego.

! Zabrania się samodzielnego załączania stanowiska bez sprawdzenia połączeń i wydaniu zgody przez prowadzącego.

! Zmian parametrów lub konfiguracji stanowiska przy użyciu dostępnych przełączników i potencjometrów można dokonywać po uprzednim przeanalizowaniu skutków takich działań.

! Zmian w konfiguracji obwodów elektrycznych polegających na zmianie połączeń przewodów lub wymianie przyrządów, należy dokonywać po uprzednim wyłączeniu zasilania stanowiska.

! Zabrania się wykonywania przełączeń (przewodów, urządzeń) w układzie znajdującym się pod napięciem.

! Przy obsłudze stanowisk, które zawierają elementy zasilane napięciem elektrycznym wyższym niż napięcie bezpieczne, należy zachować szczególną ostrożność w celu uniknięcia porażenia prądem elektrycznym.

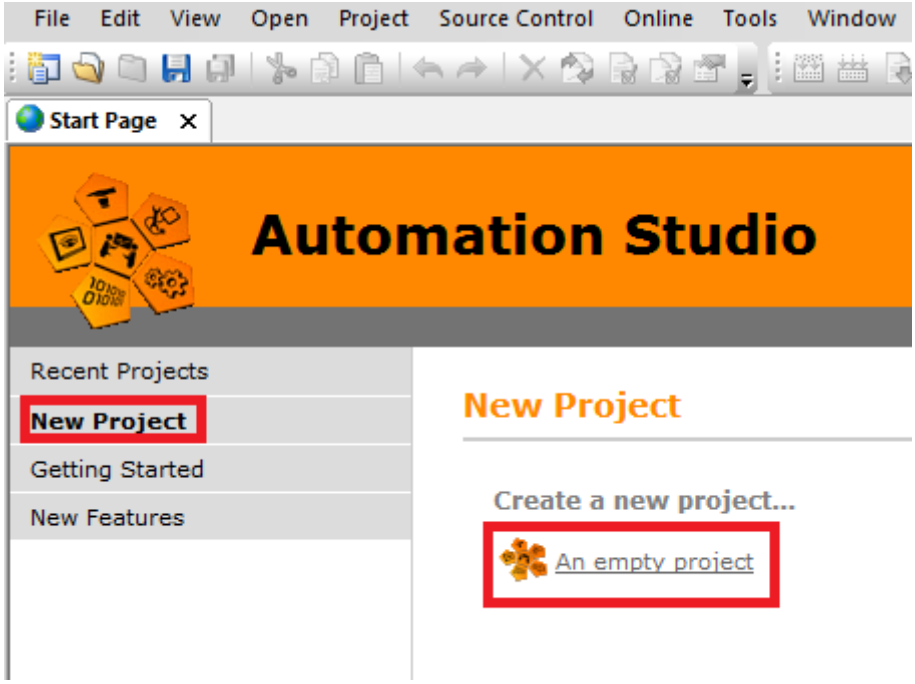
! Stosowanie ustawień i procedur innych niż opisane w instrukcji lub zalecone przez prowadzącego może spowodować nieprzewidziane działanie, a nawet uszkodzenie stanowiska.

! Przekroczenie dopuszczalnych parametrów (napięć, prądów) może doprowadzić do uszkodzenia elementów stanowiska, pożaru lub porażenia prądem.

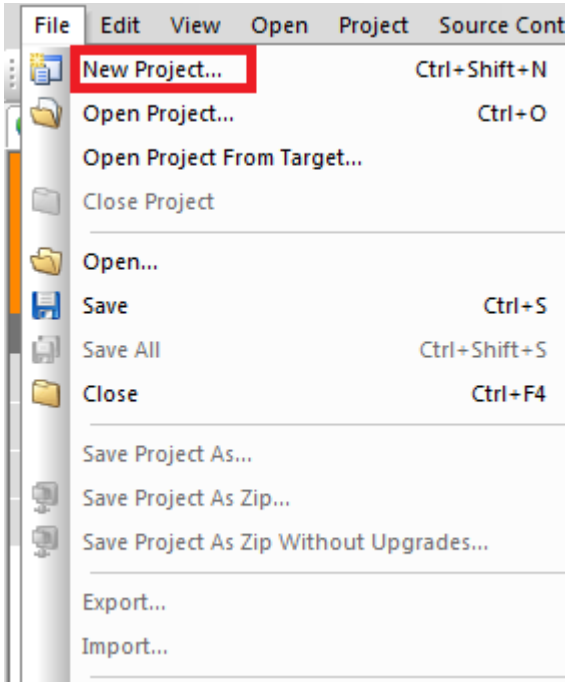
! W przypadku nieprawidłowego działania urządzeń lub wystąpienia objawów uszkodzeń (np. iskrzenie, zapach spalenizny) należy natychmiast wyłączyć stanowisko i powiadomić prowadzącego.

1. Tworzenie nowego projektu w Automation Studio

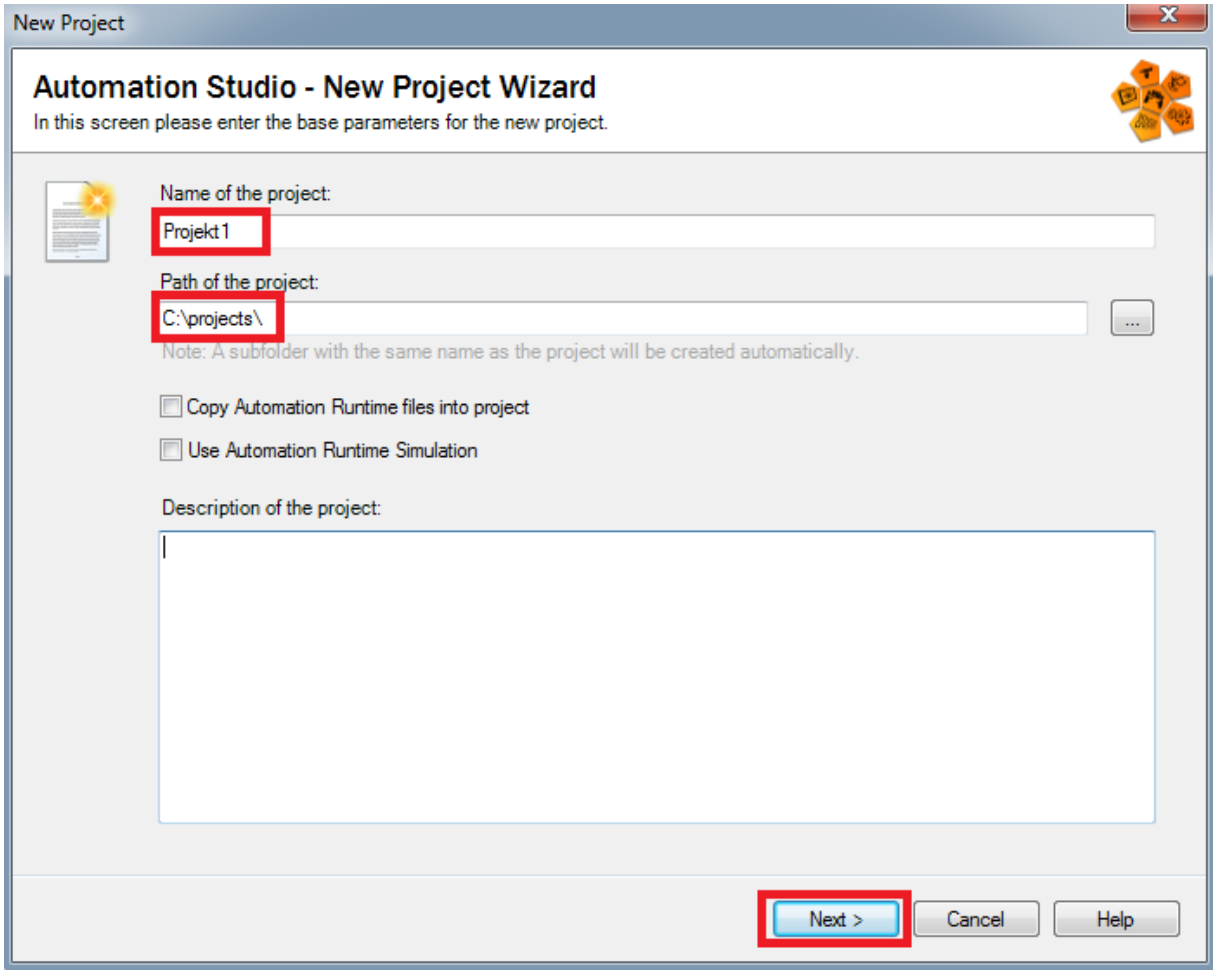
Po uruchomieniu programu Automation Studio wybieramy New Project, a następnie An empty project.



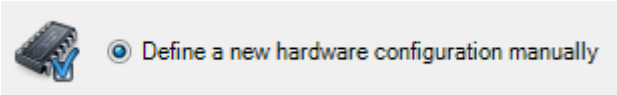
Alternatywną metodą jest wybór z menu File -> New Project



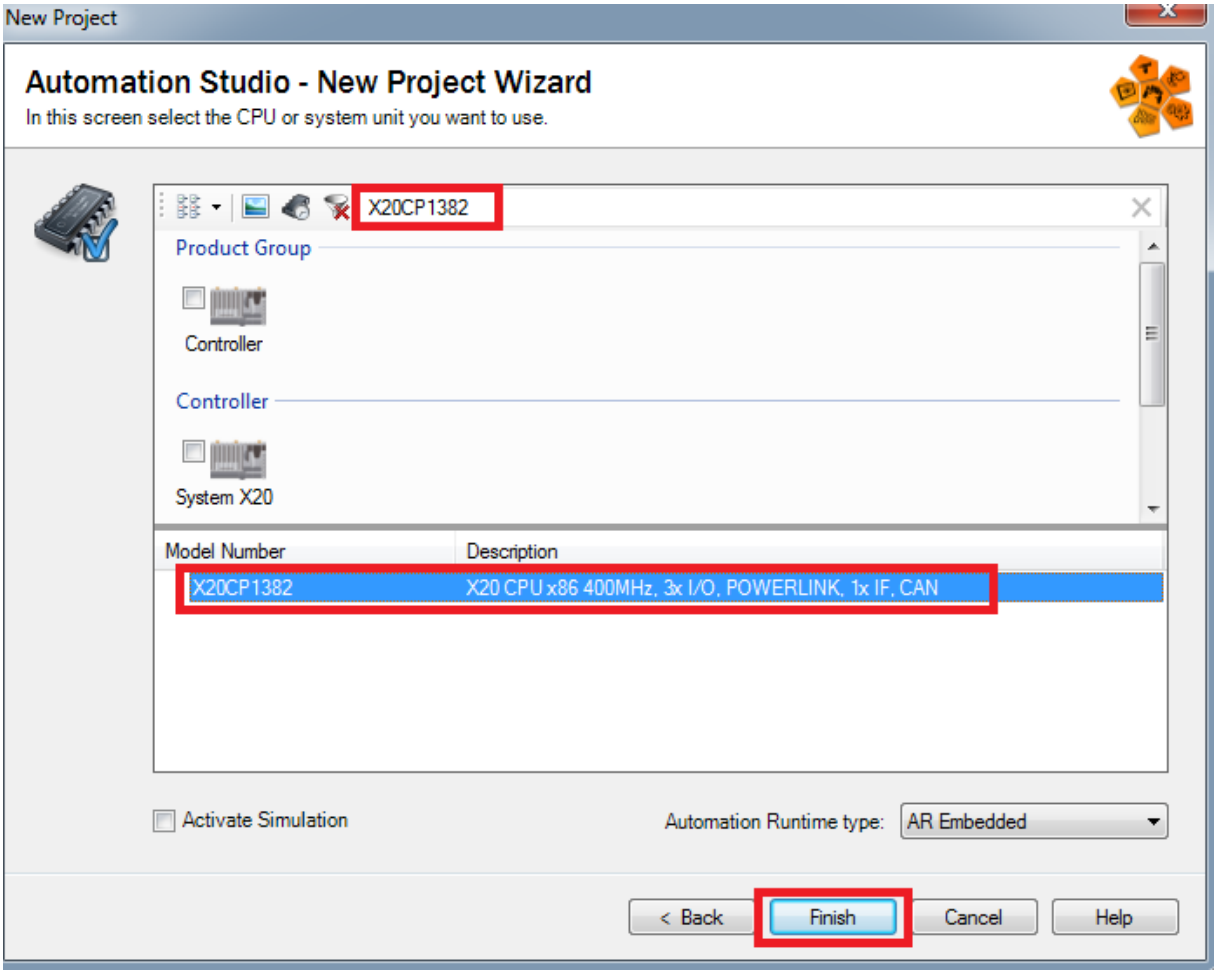
Po wybraniu nowego projektu ukaże się kreator konfiguracji. W pierwszym oknie wpisujemy nazwę projektu oraz ścieżkę, pod którą ma zostać on zapisany.



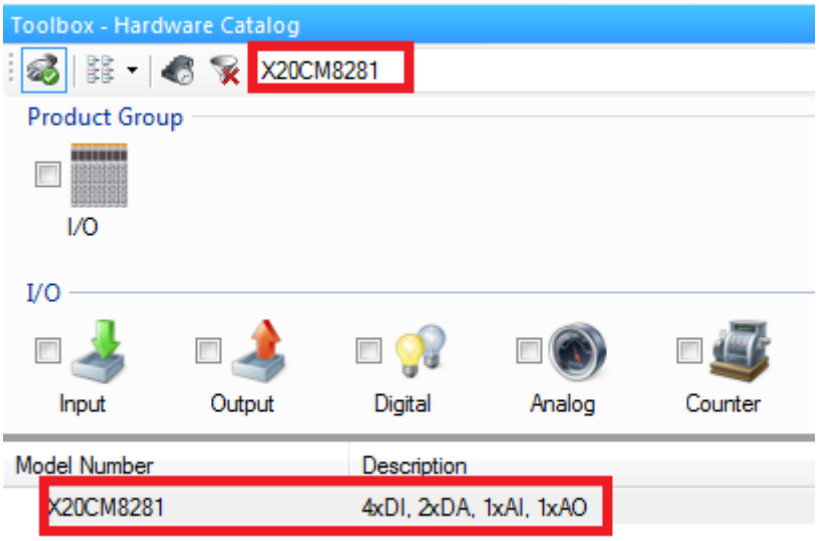
Po kliknięciu na Next > należy zdecydować jak chcemy zdefiniować sprzęt. W naszym wypadku zrobimy to ręcznie dlatego wybieramy opcję



I klikamy Next >. Kolejnym oknem jest wybór sterownika. W polu Search wpisujemy **X20CP1382**.



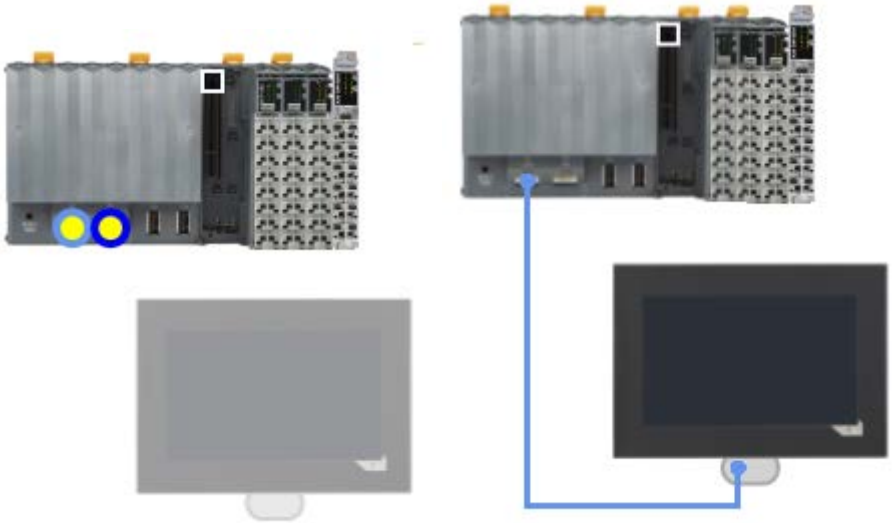
Kreator kończymy przyciskiem [Finish]. Po jego kliknięciu zobaczymy okno System Designer i obrazek naszego sterownika. Teraz należy dodać resztę potrzebnych elementów. W pierwszej kolejności dodamy moduł wejść i wyjść **X20CM8281.** Po prawej stronie okna znajduje się Toolbox.



W polu Search wpisujemy nazwę modułu, a następnie klikając na nazwę z listy przeciągamy moduł na zielone pole.

1.1. Stanowisko z panelem

Osoby posiadające stanowisko z panelem w Toolboxie wpisują **6PPT30.0702-20B**, a następnie przeciągają go do podświetlonego portu (jaśniejszy niebieski).



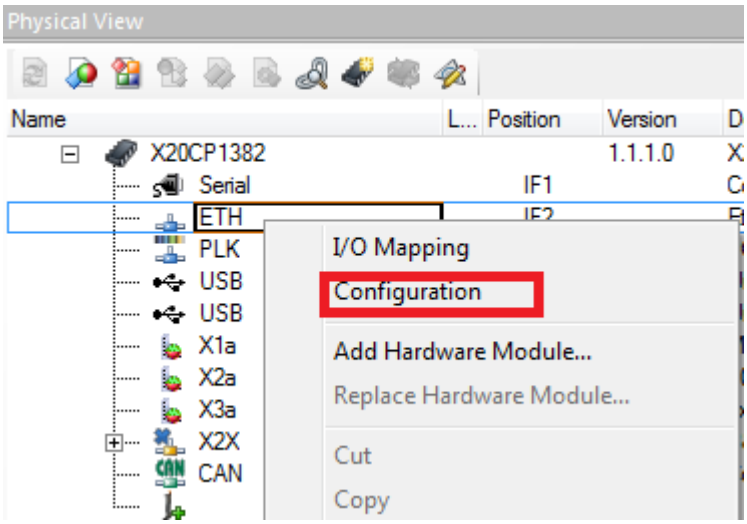
1.2. Stanowisko z silnikiem

Osoby posiadające stanowisko z silnikiem w Toolboxie wpisują **X20SM1436** i podobnie jak poprzednio przeciągają na zielone pole. Należy pamiętać, że kolejność modułów ma znaczenie i musi być identyczna jak w rzeczywistości.

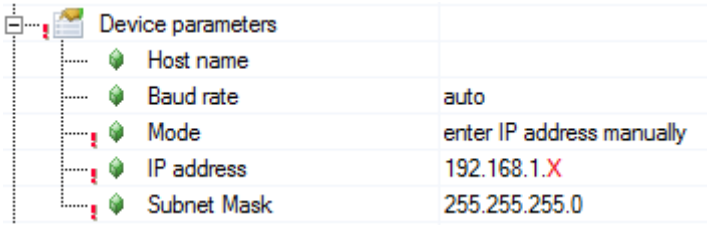
Po dodaniu modułu otworzy się konfigurator silnika. W pierwszym oknie nic nie zmieniamy i klikamy Next >. Język wybieramy wg predyspozycji i klikamy Next >. Kolejne okno również pozostawiamy bez zmian i również w tym wypadku klikamy przycisk Next >. Kreator kończymy przyciskiem Finish. Wszystkie informacje dotyczące pominiętych ustawień zostaną przedstawione w późniejszym terminie na zajęciach dotyczących silników.

2. Ustawienie adresu IP sterownika w projekcie

W pierwszej kolejności klikamy Physical View , a następnie prawym przyciskiem myszy na ETH i wybieramy Configuration.

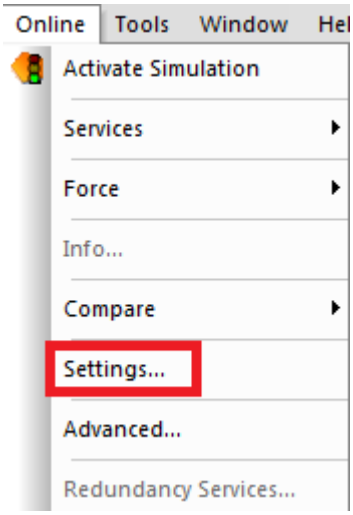


W kolejnym kroku zmieniamy ustawienia jak na obrazku poniżej, gdzie X jest numerem stanowiska .

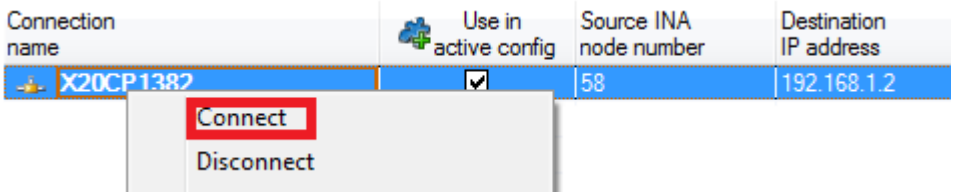


3. Połączenie się ze sterownikiem

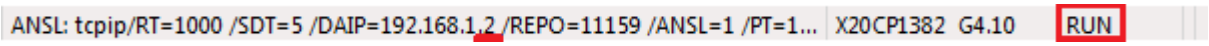
W pierwszym kroku wybieramy z paska menu Online opcję Settings.



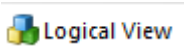
Teraz wystarczy zaznaczyć „Use in active config”, a następnie kliknąć prawym klawiszem myszy na połączenie i wybrać

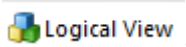


Po połączeniu się ze sterownikiem na dolnej belce statusu pojawi się RUN i adres, z którym się połączyliśmy. Proszę upewnić się, że ostatni numer w adresie IP pokrywa się z numerem sterownika.

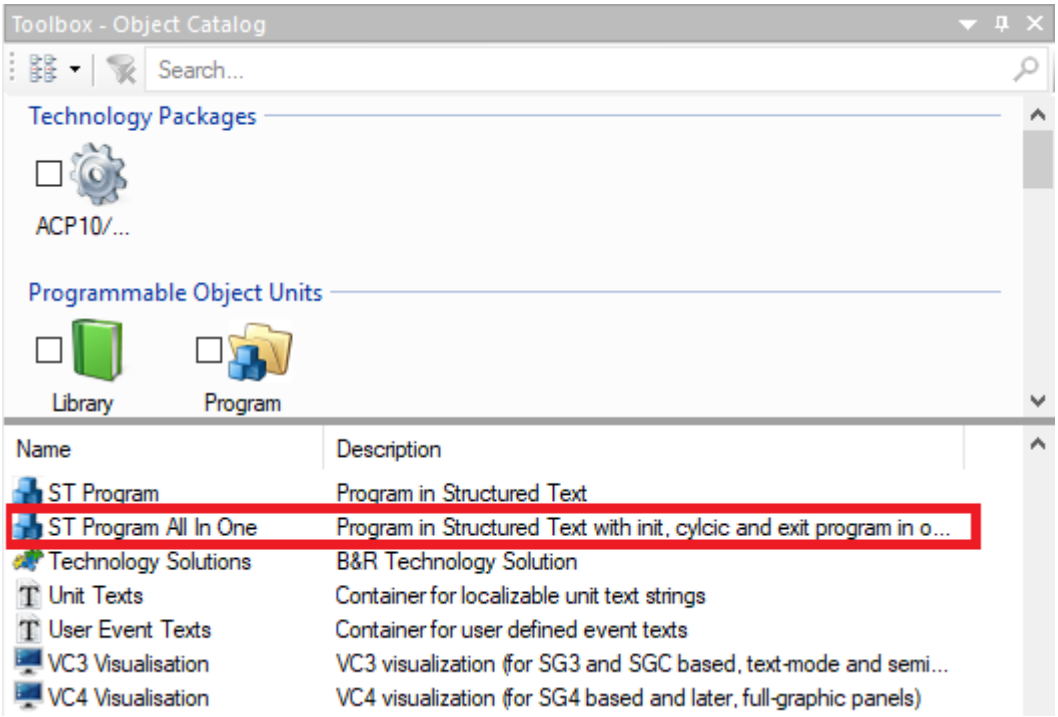


4. Dodawanie programu w języku ST

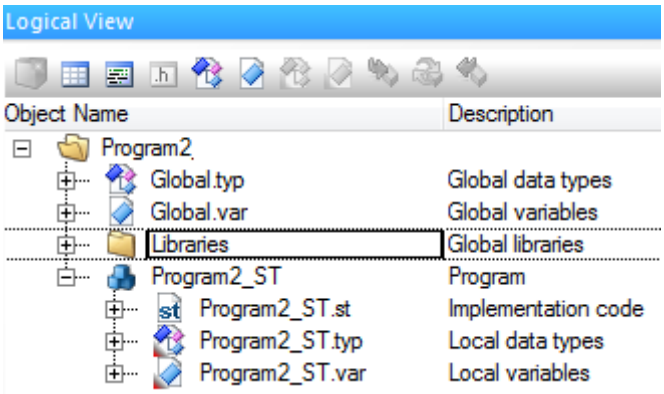
W pierwszym kroku kursorem wybieramy przycisk 

, a następnie klikamy lewym przyciskiem myszy na nazwę projektu i w Toolboxie wybrać **ST Program All In One** i przeciągamy na nazwę swojego projektu w 

.



W oknie projektu dodał się program o nazwie jakiej wcześniej wybraliśmy. Pojawiły się też pliki .var i .typ, w których znajdować się będą zmienne lokalne.



4.1. Informacje ogólne na temat języka ST

Tekst strukturalny (Structured Text, ST) jest językiem programowania wysokiego poziomu, którego podstawowa koncepcja opiera się na elementach języka BASIC, Pascal i ANSI C. Najważniejszą cechą tekstu strukturalnego są zrozumiałe konstrukty standardowe, które umożliwiają szybkie i wydajne programowanie aplikacji dla automatyki.

4.1.1. Przypisanie zmiennej

Przypisanie składa się ze zmiennej, przedstawionej po lewej stronie i do której przydzielony jest wynik obliczenia lub wyrażenia po prawej stronie za pomocą operatora ":=". Każde przypisanie musi kończyć się znakiem średnika, ";".

```
Result := ProcessValue * 2;
```

4.1.2. Operacje na funkcjach Boole'a

Argumenty operacji nie muszą zawsze mieć typu danych "BOOL".

Symbol	Operacja logiczna	Przykład
NOT	Negacja binarna	a := NOT b;
AND	Logiczne AND	a := b AND c;
OR	Logiczne OR	a := b OR c;
XOR	Wyłączne OR	a := b XOR c;

4.1.3.Operacje arytmetyczne

Symbol	Operacja arytmetyczna	Przykład
:=	Przypisanie	a := b;
+	Dodawanie	a := b + c;
-	Odejmowanie	a := b - c;
*****	Mnożenie	a:= b *c;
/	Dzielenie	a := b / c;
MOD	Modulo, reszta dzielenia całkowitoliczbowego	a := b MOD c;

Typ danych zmiennych i wartości zawsze ma krytyczne znaczenie dla obliczeń. Wynik obliczany jest po prawej stronie wyrażenia i dopiero następnie przydzielany do odpowiadającej mu zmiennej. Wynik zależy od przyjętych typów danych i składni (notacji).

4.1.4.Funkcja decyzyjna IF

Instrukcja "IF" jest jednym ze sposobów podejmowania decyzji w programie.

Słowa kluczowe	Składnia	Opis
IF .. THEN	IF a > b THEN	1. porównanie
	Result := 1;	Instrukcja "jeżeli 1. porównanie" jest "TRUE"
ELSIF .. THEN	ELSIF a > c THEN	2. porównanie
	Result := 2;	Instrukcja "jeżeli 2. porównanie" jest "TRUE"
ELSE	ELSE	Gałąź alternatywna, bez porównań "TRUE"
	Result := 3;	Instrukcja gałęzi alternatywnej
END_IF	END_IF	Koniec decyzji

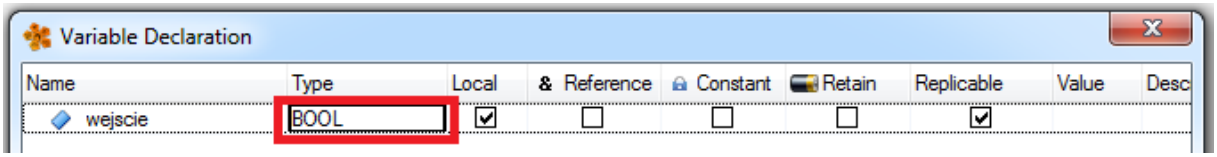
5. Prosty program w języku ST

Klikamy dwukrotnie na program lewym przyciskiem myszy. Po otwarciu okna do pisania programu w języku ST dodamy prosty program zapalający diodę po przełączeniu przycisku. W oknie programu wpisujemy IF i klikamy przycisk TAB. Po tej czynności powinna uzupełnić się składnia funkcji IF.

```
IF expression THEN

END_IF;
```

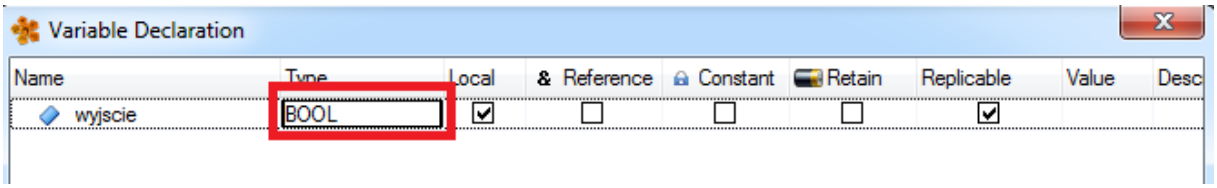
W miejscu **expression** wpisujemy **wejscie**, przechodzimy na koniec linii i klikamy ENTER. Program chce dodać zmienną. Ustawiamy jej typ jako BOOL.



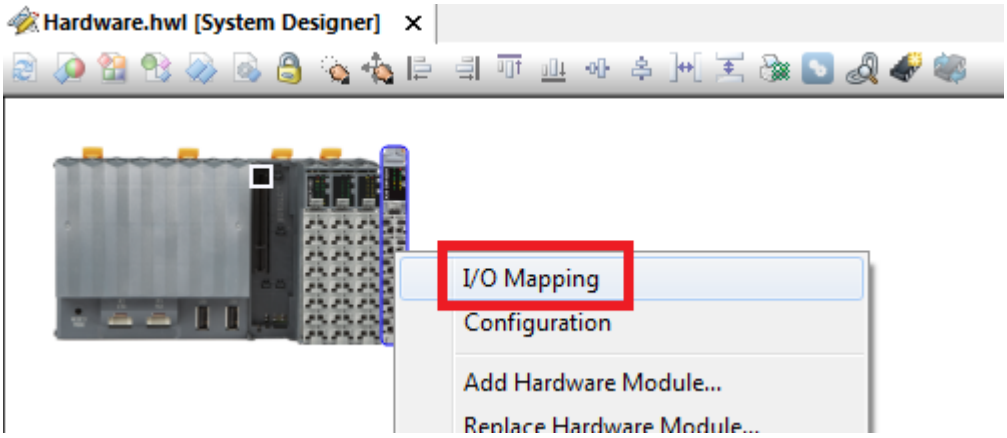
Po słowie THEN wpisujemy **wyjscie:=true;**

```
IF wejscie THEN
    wyjscie:=true;
END_IF;
```

Program chce dodać zmienną. Ustawiamy jej typ jako BOOL.



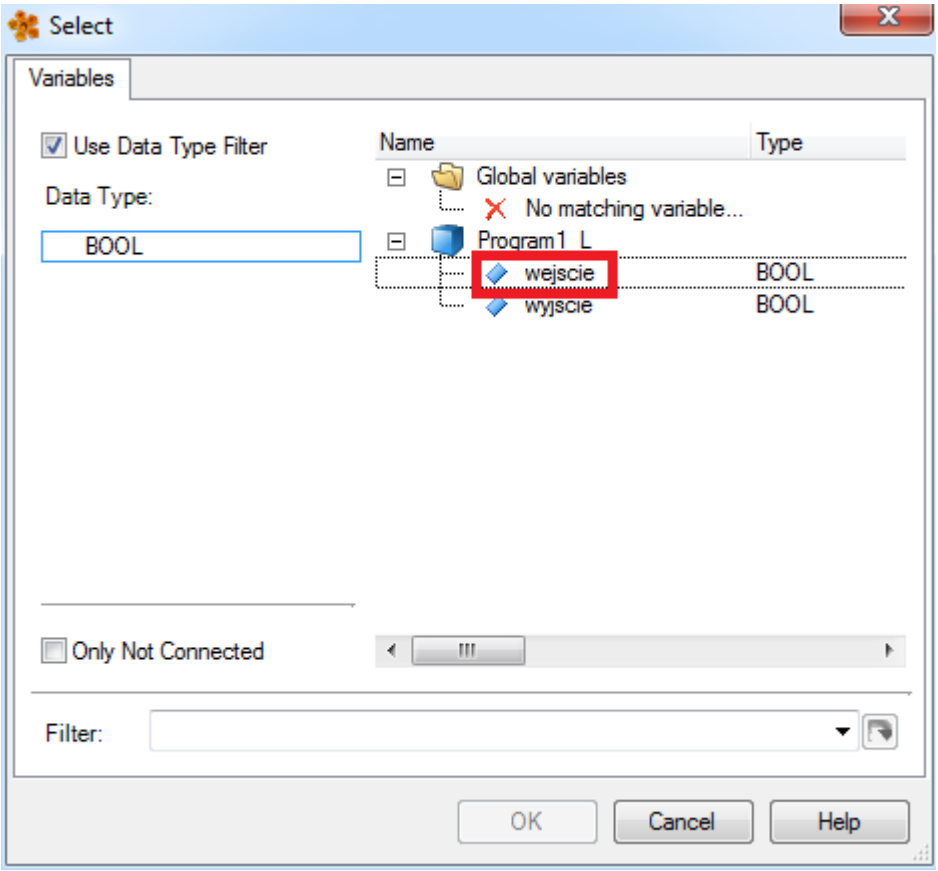
Przechodzimy zakładki Physical View do System Designera klikając ikonę , a następnie prawym przyciskiem myszy na moduł **CM8281** i wybieramy opcję I/O Mapping.



Następnie w oknie przypisujemy nasze zmienne do wejść i wyjść. Klikamy dwukrotnie na kolumnę Process Variable przy wybranym wejściu, w tym wypadku DigitalInput01.

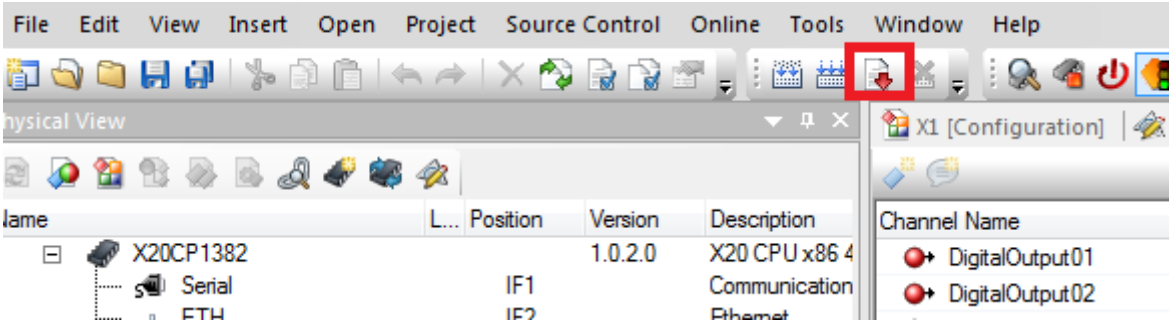
Channel Name	Process Variable	Description [1]
ModuleOk		Module status (1 = module present)
StaleData		Data not from latest cycle
SerialNumber		Serial number
ModuleID		Module ID
HardwareVariant		Hardware variant
FirmwareVersion		Firmware version
DigitalInput01		24 VDC, 0.01 to 25 ms switching delay, sink
DigitalInput02		24 VDC, 0.01 to 25 ms switching delay, sink
DigitalInput03		24 VDC, 0.01 to 25 ms switching delay, sink
DigitalInput04		24 VDC, 0.01 to 25 ms switching delay, sink

Klikamy na symbol kropek [...] , wybieramy program z którego chcemy podłączyć zmienną i samą zmienną.

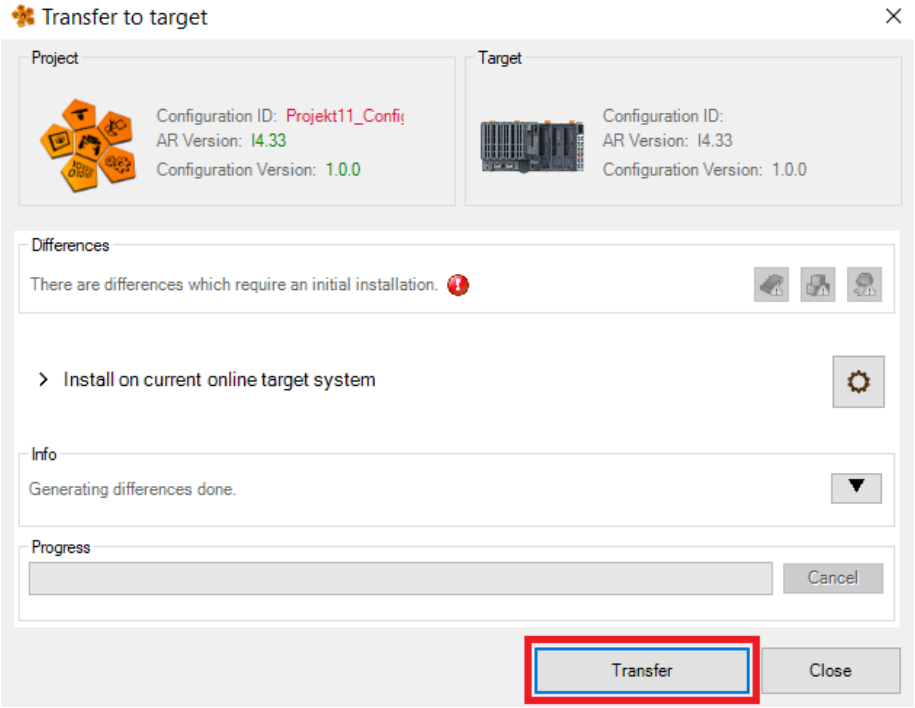


Po zatwierdzeniu przyciskiem [OK] w kolumnie Process Variable pojawi się ścieżka do zmiennej. Te same czynności powtarzamy dla wyjścia **DigitalOutput01** i zmiennej **wyjscie**.

Ostatnim krokiem jest wgranie programu do sterownika. Do tego użyjemy przycisku Transfer



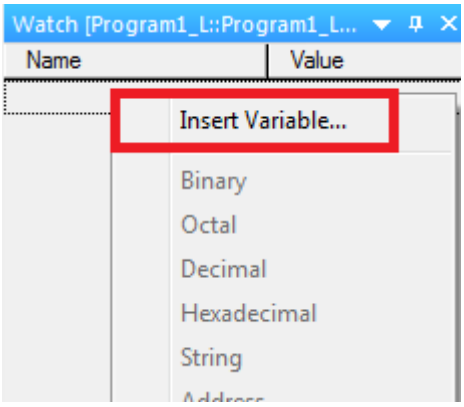
W oknie Output wyświetlany jest przebieg kompilacji. Ostrzeżenia wyświetlane są na zielono, a błędy na czerwono. Dalej wystarczy wysłać program na sterownik.



Zmiany potwierdzamy klikając Tak.

5.1.1. Podgląd zmiennych i działania programu

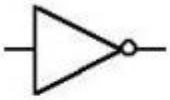
Po przejściu do **okna programu** i wybraniu przycisku monitora możemy podglądać pracę programu online. Po prawej stronie otworzy nam się okienko Watch, do którego możemy dodawać zmienne albo przeciągając je do okienka albo klikając prawym przyciskiem myszy na pole Watcha i wybierając **Insert Variable…**



5.1.2.Zadanie dla studentów nr 1

Student ma za zadanie napisać program, który zasymulują działanie bramek logicznych na sterowniku. Wszystkie bramki mają znajdować się w jednym programie. Wszystkie wejścia i wyjścia z bramek mają zostać podłączone do fizycznych wejść i wyjść sterownika. Poniżej podano wymagane bramki wraz z tabelami.

INWENTOR - NOT

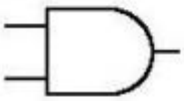


TABLICA PRAWDY	
we	wyj
0	1
1	0

NAND

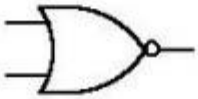
A	B	Wyj
0	1	1
0	0	1
1	1	0
1	0	1

AND



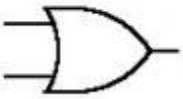
A	B	Wyj
0	1	0
0	0	0
1	1	1
1	0	0

NOR



A	B	Wyj
0	0	1
0	1	0
1	0	0
1	1	0

OR



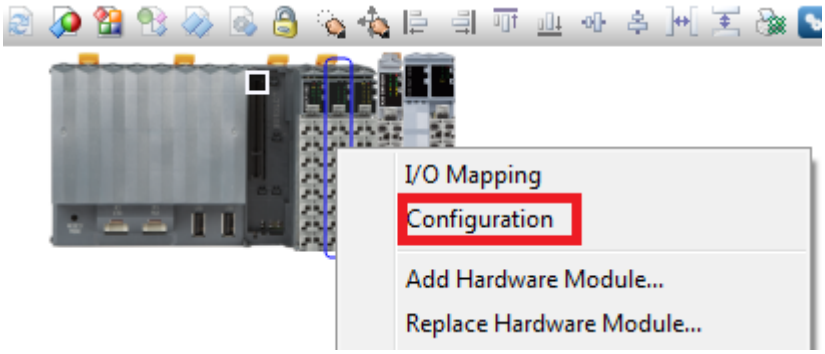
A	B	Wyj
0	0	0
0	1	1
1	0	1
1	1	1

EXOR

A	B	Wyj
0	0	0
0	1	1
1	0	1
1	1	0

6. Konfiguracja modułu jako wejście enkoderowe

Przechodzimy do System Designera klikając ikonę , a następnie prawym przyciskiem myszy na moduł **X2** i wybieramy opcję Configuration.



W sekcji Counter configuration wybieramy dwa enkodery AB tak jak pokazano poniżej.

Counter configuration		
Counter mode	AB incremental encoder 2x	Select counter mode
Time information	off	Display time information

Po zapisaniu zmian w I/O Mapping modułu **X2** pojawią się dwa wejścia enkoderowe i dwa wyjścia służące do zerowania enkodera.

Counter01Reset	Clear counter	BOOL
Counter01Value	Current counter value AB-01	DINT
Counter02Reset	Clear counter	BOOL
Counter02Value	Current counter value AB-02	DINT

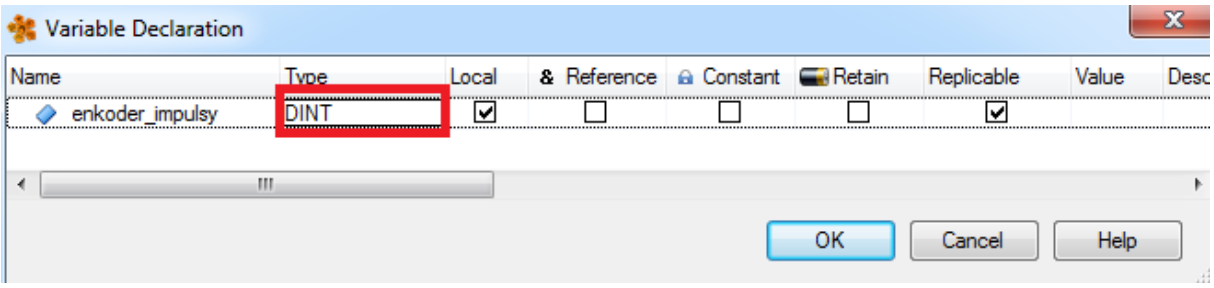
6.1. Przykładowy program do obsługi enkodera

Program będzie miał za zadanie zliczać impulsy do zmiennej **enkoder_impulsy**, a po przekroczeniu wartości 1000 będzie zerowany zmienną **enkoder_zeruj**. Zerowanie będzie następowało po wystawieniu wartości 1 w zmiennej. W oknie programu wpisujemy IF i klikamy przycisk TAB. Po tej czynności powinna uzupełnić się składnia funkcji IF.

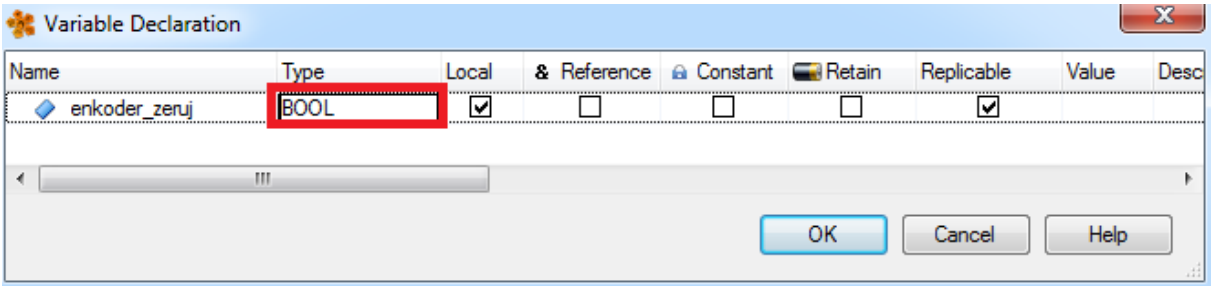
```
IF expression THEN

END_IF;
```

W miejscu **expression** wpisujemy **enkoder_impulsy>=1000**, przechodzimy na koniec linii i klikamy ENTER. Program chce dodać zmienną. Jej rodzaj zależy od informacji w mapowaniu wejść modułu. Jak pokazano w punkcie 2. chcemy dodać zmienną do **Counter01Value**. Po prawej stronie na różowy kolor podany jest typ zmiennej jaki możemy do tego wejścia podłączyć, dlatego wybieramy typ **DINT**.



W kolejnej linii programu piszemy **enkoder_zeruj:=1;**. Oznacza to tyle, że jeśli impulsów będzie więcej niż 1000 ma ustawić zmienną na 1 i wyzerować wartość na enkoderze. Po wpisaniu linii klikamy ENTER i dodajemy zmienną. Tym razem zgodnie z mapowaniem będzie to zmienna typu **BOOL**.



W tym momencie dopisujemy dwie linijki kodu, które ustawią zmienną **enkoder_zeruj** na 0 w innych wypadkach niż po przekroczeniu żądanej wartości. Cały kod podany poniżej.

```
IF enkoder_impulsy>=1000 THEN
    enkoder_zeruj:=1;
ELSE
    enkoder_zeruj:=0;
END_IF;
```

Ostatnim krokiem jest przypisanie zmiennych do modułu tak jak pokazano poniżej.

Counter01Reset	::Program2_S:enkoder_zeruj	Clear counter	BOOL
Counter01Value	::Program2_S:enkoder_impulsy	Current counter value AB-01	DINT

Teraz wysyłamy program na sterownik i w Watchu podglądamy czy jego działanie jest poprawne.

6.2. Zadanie dla studentów nr 2

Student ma za zadanie bazując na poprzednim przykładzie napisać program, który będzie zapalał diody w zależności od ilości obrotów. Kręcąc w przeciwną stronę diody mają gasnąć. Należy zablokować maksymalną ilość odczytanych obrotów do ilości diod.

7. Używanie wejść i wyjść analogowych

W zestawach są 3 wejścia analogowe i 1 wyjście analogowe: 2 wejścia w module **X1** oraz w module **X20CM8281** 1 wejście i 1 wyjście analogowe. Aby ich użyć nie trzeba jak w poprzednim wypadku przeprowadzać żadnej konfiguracji. Wystarczy zmapować zmienną do modułu. Wszystkie zmienne powinny mieć typ INT.

AnalogInput01	±10 V, resolution 12 bit	INT
AnalogInput02	±10 V, resolution 12 bit	INT

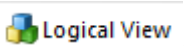
AnalogInput01	±10 V / 0 to 20 mA / 4 to 20 mA, resolution 12 bit	INT

AnalogOutput01	±10 V / 0 to 20 mA, resolution 12 bit	INT

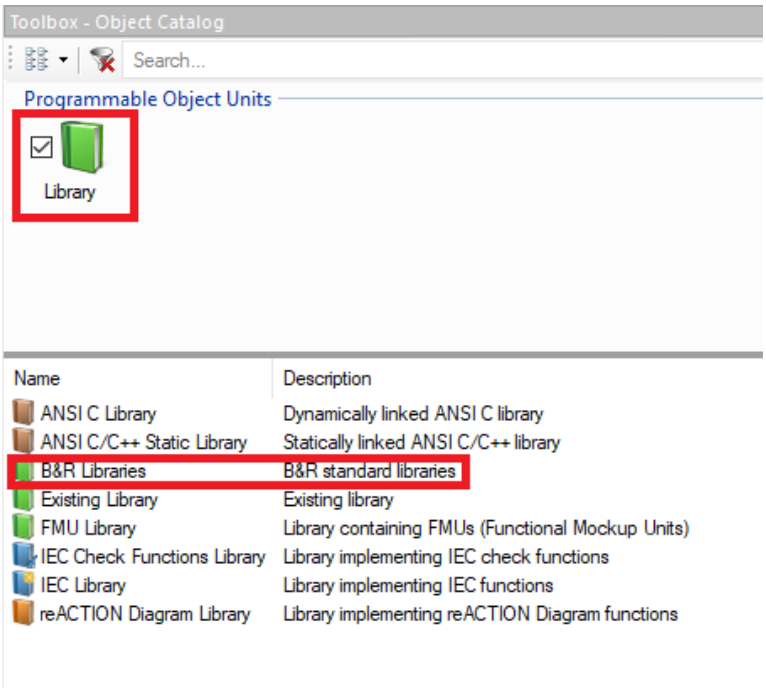
7.1. Zadanie dla studentów nr 3

Student ma za zadanie bazując na poprzednim przykładzie napisać program, który będzie zmieniał prędkość obrotów silniczka w zależności od obrotów enkodera. Próg impulsów ma być ustawiany za pomocą pokrętła analogowego, a nie jak w poprzednim wypadku ustawiony na stałą wartość 1000 impulsów.

8. Używanie Timera

Przed użyciem Timera należy dodać bibliotekę standard. Klikamy w 

. Następnie wybieramy lewym klawiszem myszy na folderze Libraries. W Toolboxie zaznaczyć należy **Library** i kliknać 2 razy w **B&R Libraries**.



Dalej z listy wybieramy bibliotekę standard i potwierdzamy Finish.

Timer jest blokiem funkcyjnym odliczającym podany czas. W pierwszym kroku należy wypisać parametry bloku takie jak odmierzany czas i enable bloku.

```
(*-- Parametry *)
TON_1.IN := TON_wejscie;
TON_1.PT := T#1s;
```

Zmienne **TON_1** deklarujemy jako typ **TON**, a zmienną **TON_wejscie** jako **BOOL**. W kolejnym kroku przypisujemy wyjście z bloku do zmiennej **TON_wyjscie**, którą również deklarujemy jako **BOOL**.

```
(*-- Czytanie wyników *)
TON_wyjscie := TON_1.Q;
```

Ostatnim i jednocześnie najważniejszym elementem jest wywołanie funkcji timera. Robimy to na końcu programu nie zamykając wywołania w żadnej funkcji warunkowej.

```
(*-- Wywołanie bloku funkcyjnego*)
TON_1();
```

Po wgraniu programu do sterowanika, przechodzimy do Watcha, przeciągamy tam wszystkie zmienne od timera. Zmieniamy **TON_wejscie** na 1 i sprawdzamy czy timer działa poprawnie. Wyzerowanie wyjścia następuje po zmianie **TON_wejscie** na 0.

diButton	TRUE
doBell	FALSE
TON_bell	
IN	TRUE
PT	T#01s
Q	FALSE
ET	T#800ms
StartTime	T#10m_26s_340ms
M	TRUE
Restart	0

8.1. Zadanie dla studentów nr 4

Student ma za zadanie napisać program, który będzie mrugał dowolną diodą z okresem 1s. Mruganie odbywa się tylko w momencie kiedy wybrany przycisk jest załączony.

8.2. Zadanie dla studentów nr 5*

Student ma za zadanie napisać program, który będzie mrugał dowolną diodą z okresem zależnym od ustawienia potencjometru.