

Wykorzystanie bibliotek PLCOPEN w sterownikach Astraada One

Praca z rzeczywistym sterownikiem i serwonapędem Astraada SRV

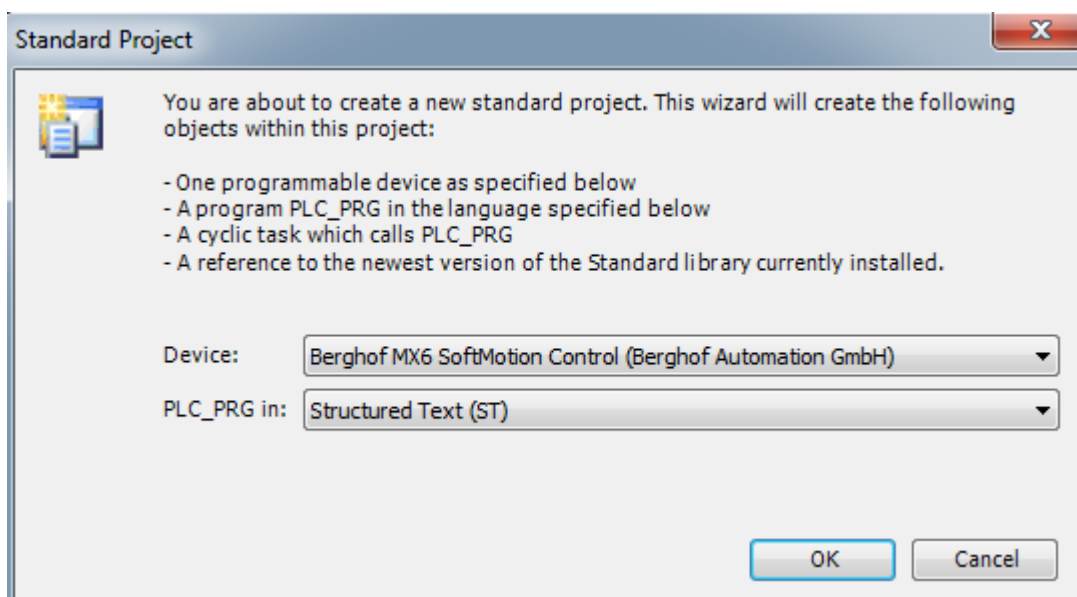
Rozpoczęcie pracy

W pierwszej kolejności należy połączyć się ze sterownikiem z poziomu przeglądarki internetowej i sprawdzić jakie licencje zaimplementowane są w sterowniku. Należy wpisać adres IP sterownika w przeglądarce, standardowo jest to adres 169.254.255.XX, gdzie XX to 2 – ostatnie cyfry numeru seryjnego (wyjątek: 00 -> adres 100). Standardowe dane logowanie admin, admin:

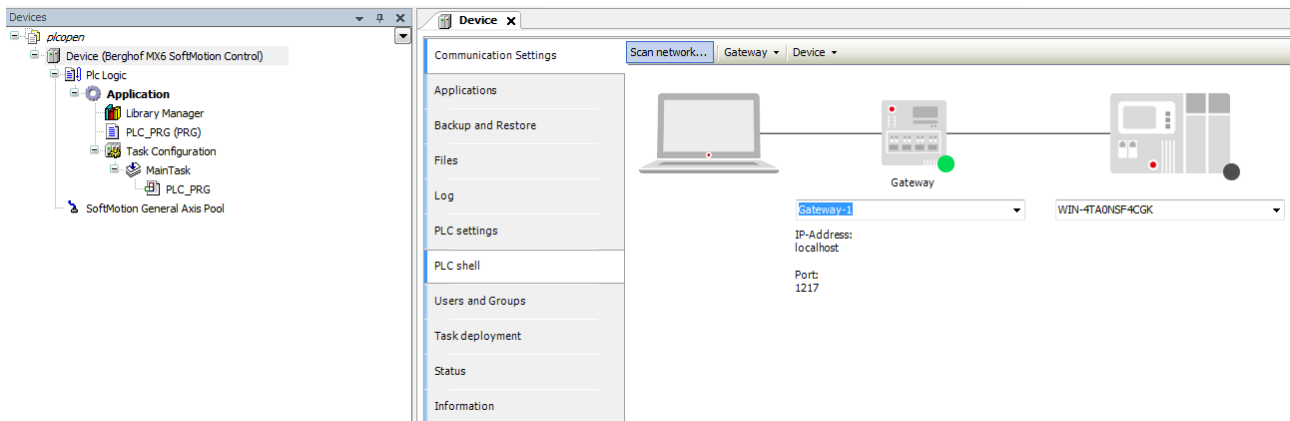
System Info Licenseinfo	Codesys RTS Version:	3.5.8.40
	Licenses:	TARGETVISU WEBVISU SOFTMOTION-DEMO MODBUS-TCP MODBUS-RTU

W zakładce Info mamy podgląd wgranych licencji do sterownika. Do wykorzystywania bloków PLCopen sterownik powinien mieć co najmniej licencję SoftMotion Demo. Ta licencja jest darmowa, można ją uzyskać w Firmie Astor. Jest to 2 godzinna licencja, po upływie 2 godzin należy zrestartować sterownik i można dalej testować urządzenie.

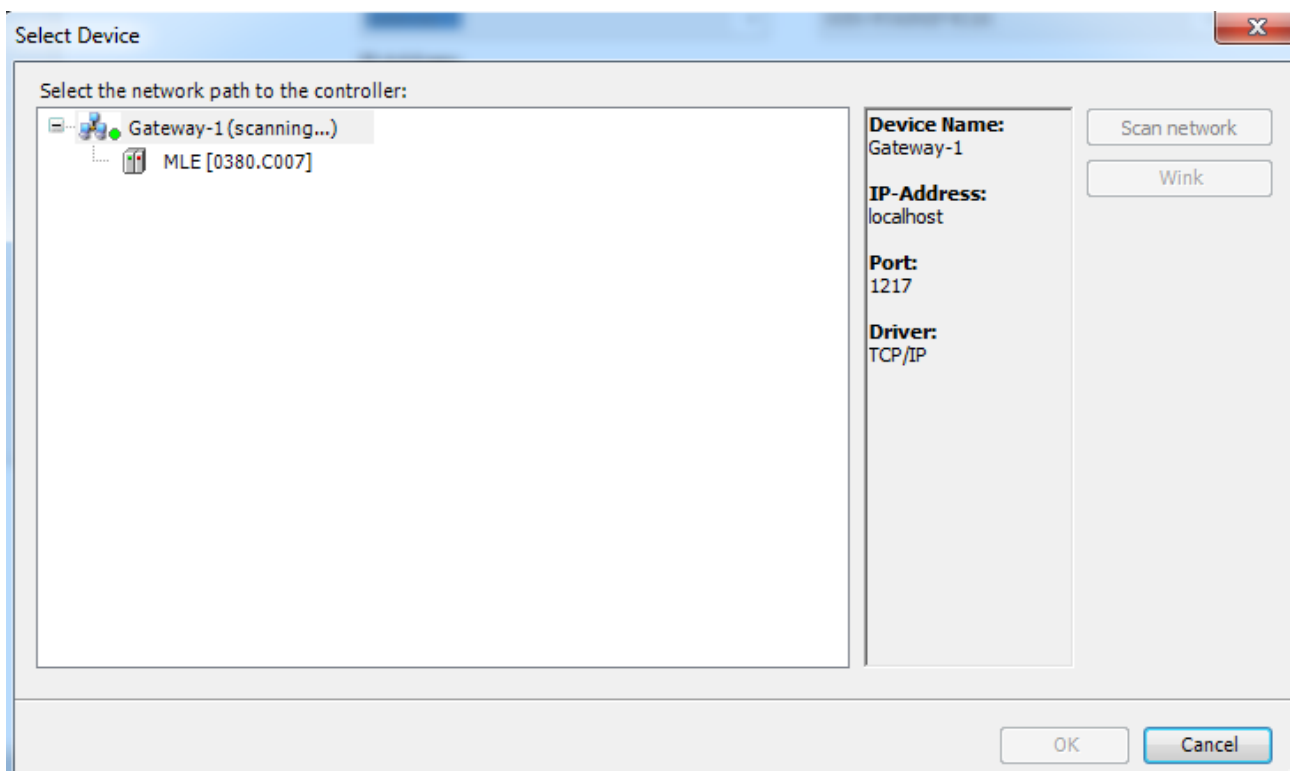
Po uruchomieniu oprogramowania Codesys, należy wybrać urządzenie BERGHOF MX6 SoftMotion Control:

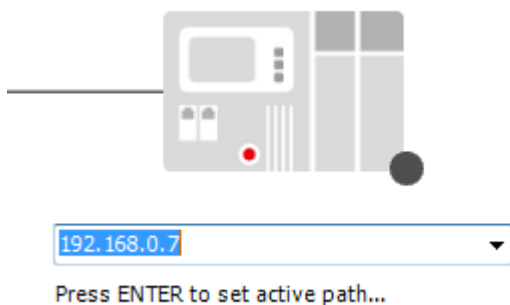


W oprogramowaniu CODESYS należy przejść do zakładki Device w drzewku projektowym:

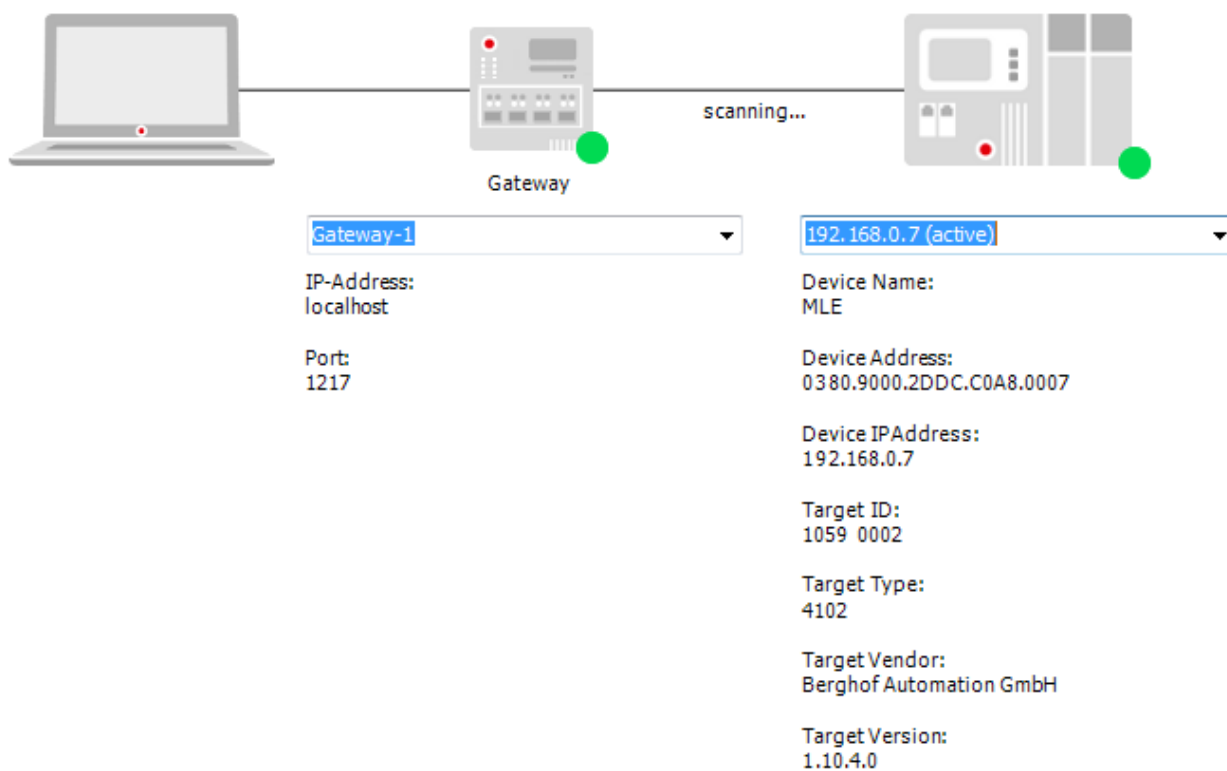


Klikamy w opcję Scan network i wyszukujemy sterownik, który będzie posiadał taką nazwę jak nazwa skonfigurowana z poziomu przeglądarki lub wpisujemy adres IP sterownika:



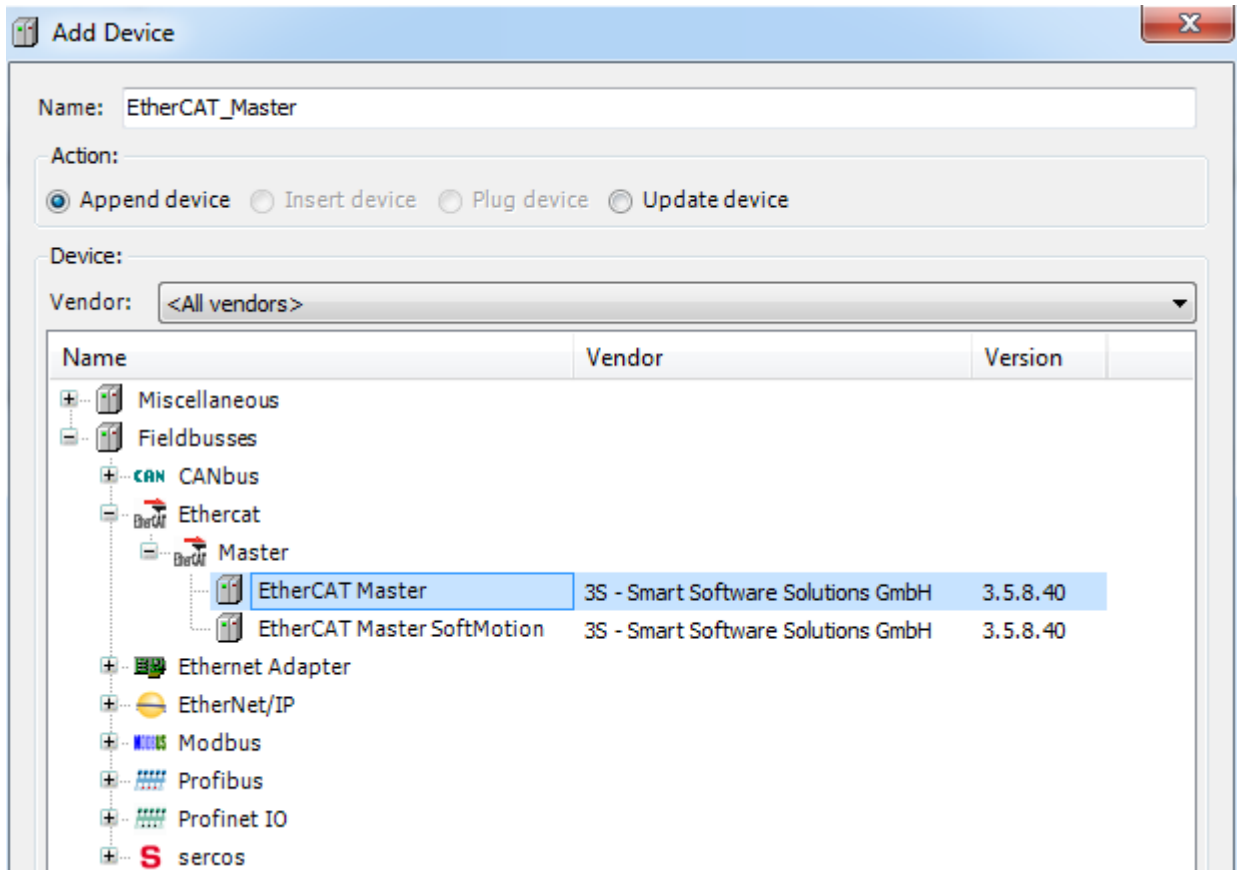


Potwierdzamy wybór i w oknie Device mamy podgląd, że oprogramowanie widzi sterownik:

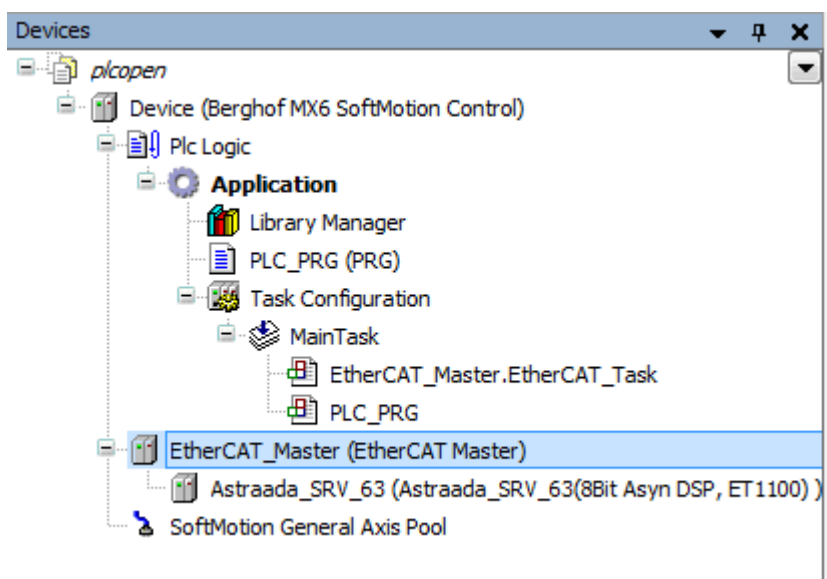
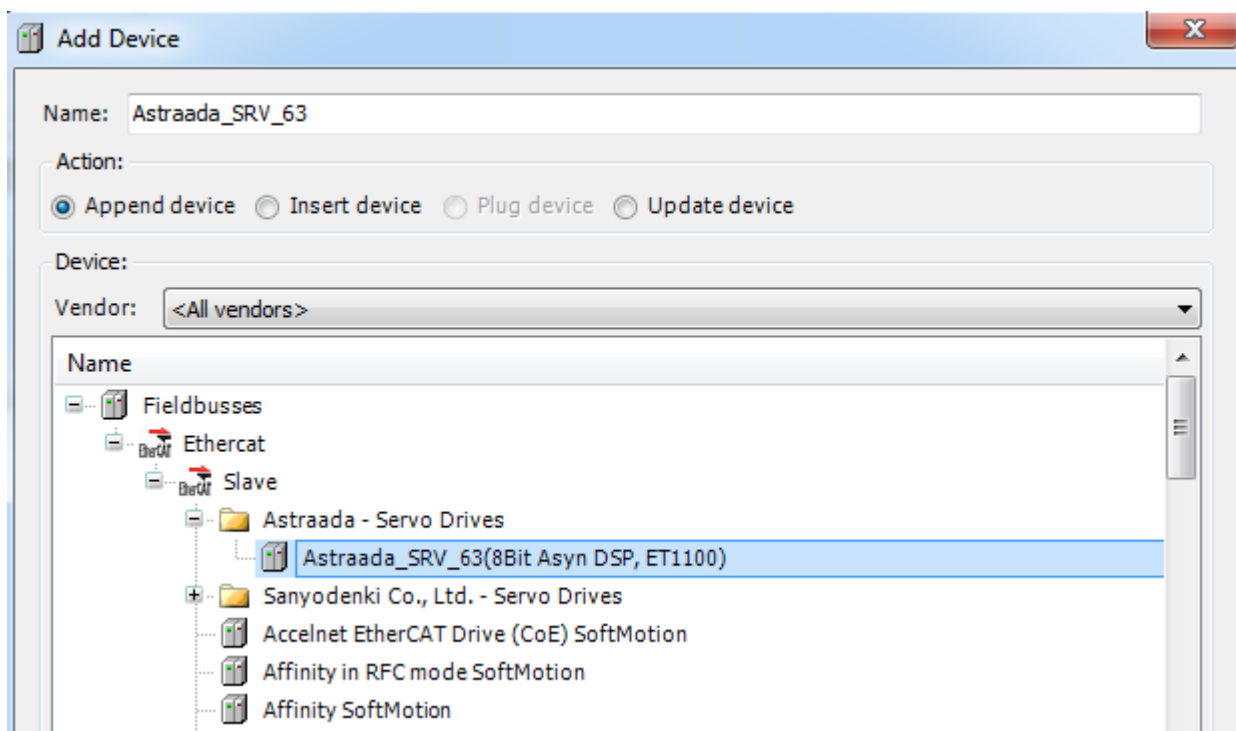


Przygotowanie aplikacji

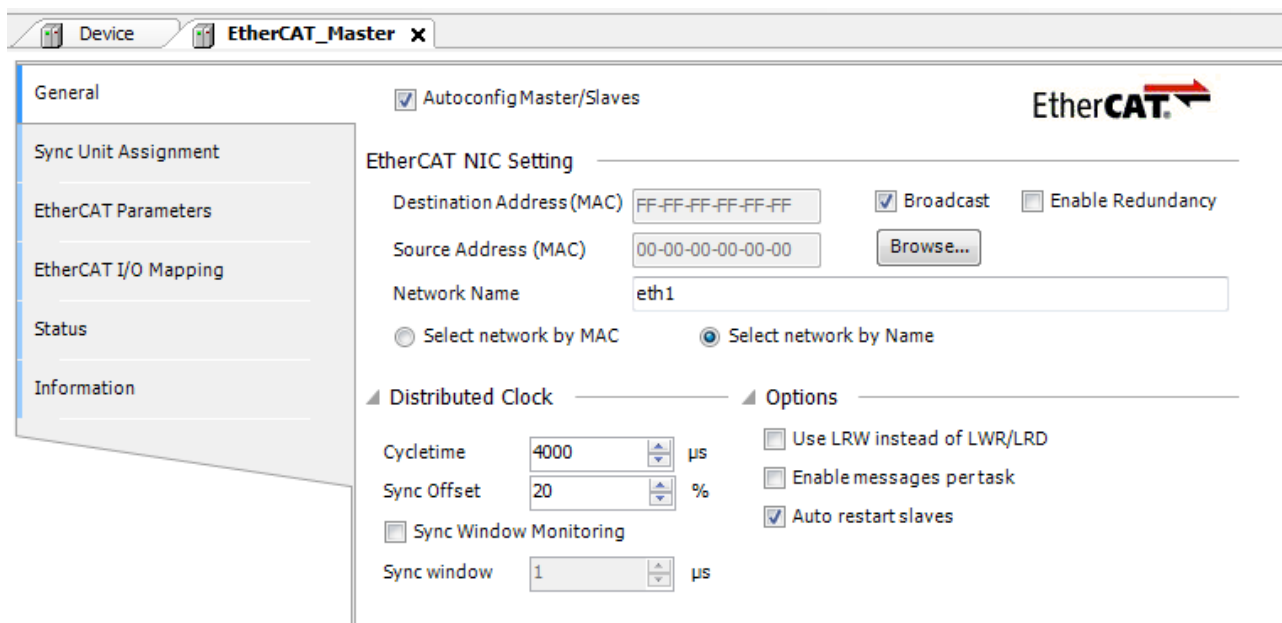
Dodajemy do drzewka projektowego napędy, które ze sterownikiem będą połączone po protokole EtherCAT:



Do EtherCAT Master dołączamy serwonapędy Astraada SRV. Możemy dodać je ręcznie lub jeżeli są fizycznie podłączone automatycznie zeskanować sieć i skopiować do projektu.



Jeżeli chcemy automatycznie zeskanować urządzenia, w pierwszej kolejności należy skonfigurować połączenie EtherCAT. Po dwukrotnym kliknięciu EtherCAT Master w drzewku projektowym wprowadzamy nazwę sieci (karty komunikacyjnej):



Device EtherCAT_Master x

General Autoconfig Master/Slaves **EtherCAT**

EtherCAT NIC Setting

Destination Address (MAC) FF-FF-FF-FF-FF-FF Broadcast Enable Redundancy

Source Address (MAC) 00-00-00-00-00-00

Network Name eth1

Select network by MAC Select network by Name

Distributed Clock **Options**

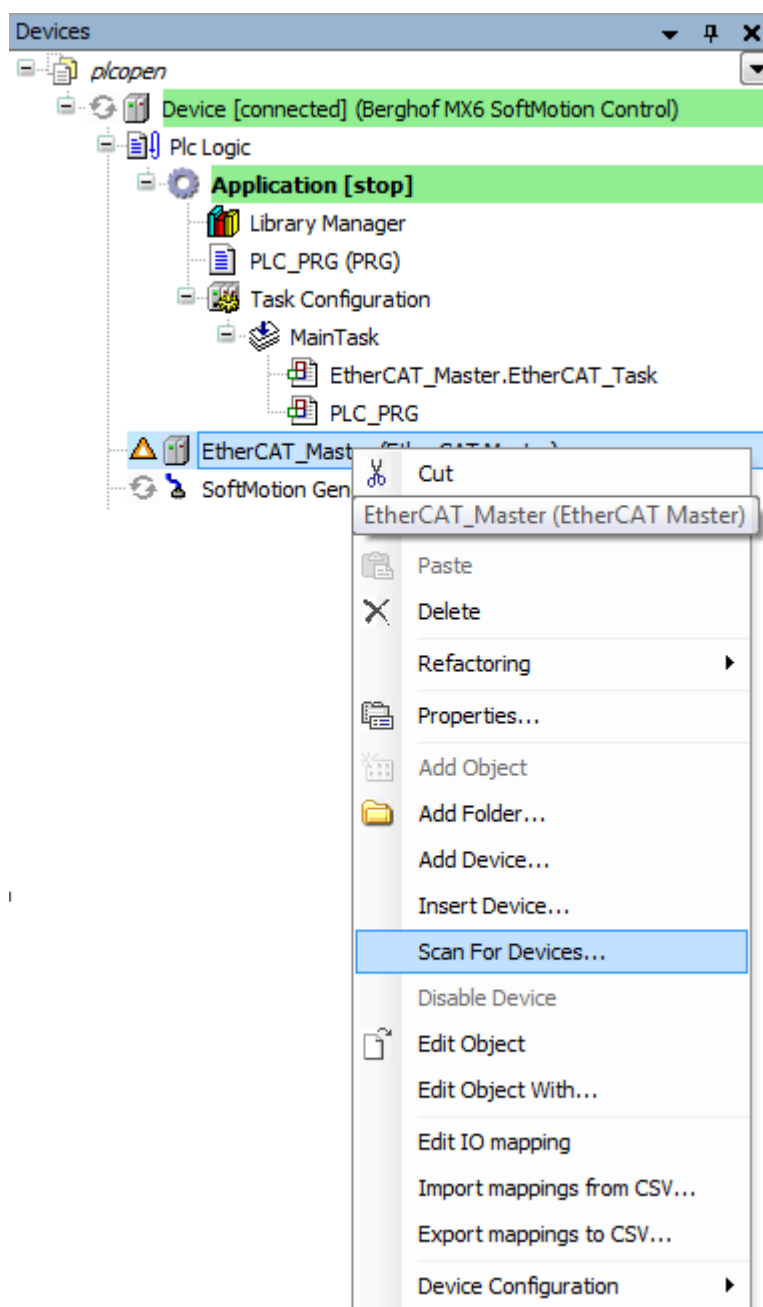
Cycletime 4000 µs Use LRW instead of LWR/LRD

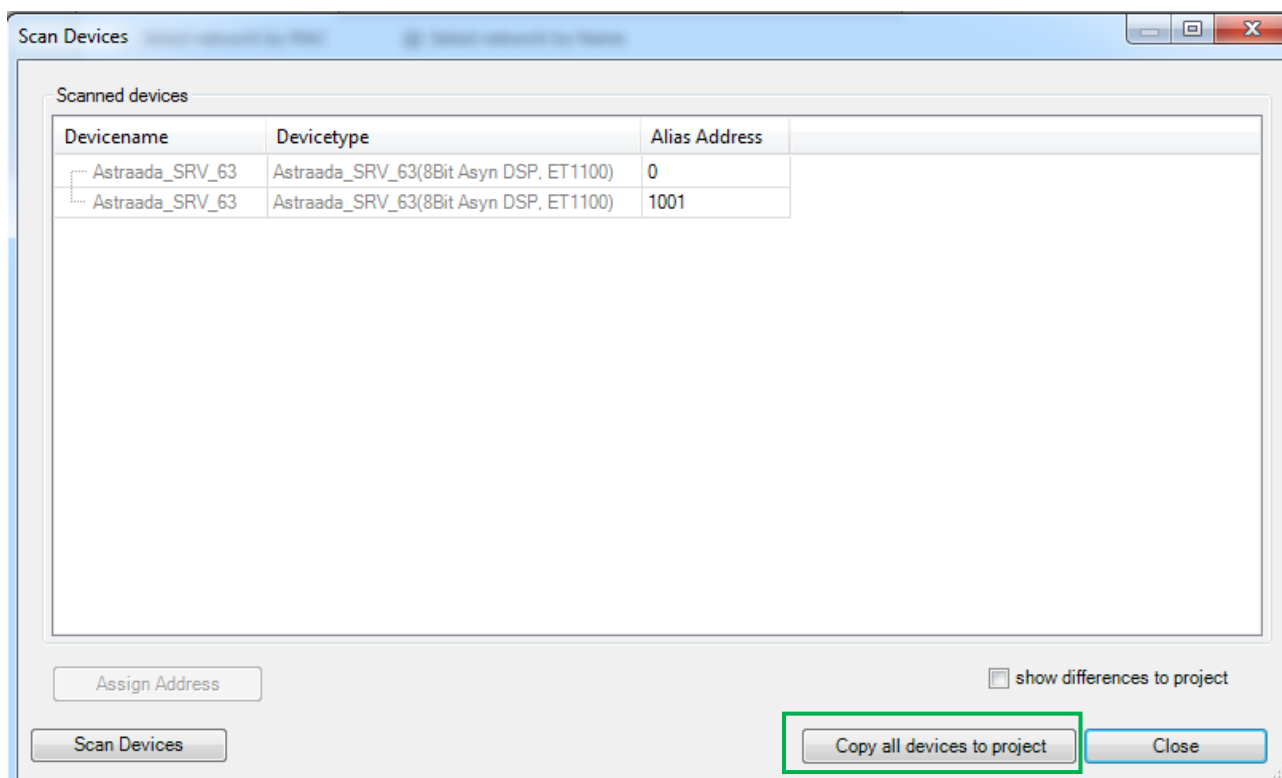
Sync Offset 20 % Enable messages per task

Sync Window Monitoring Auto restart slaves

Sync window 1 µs

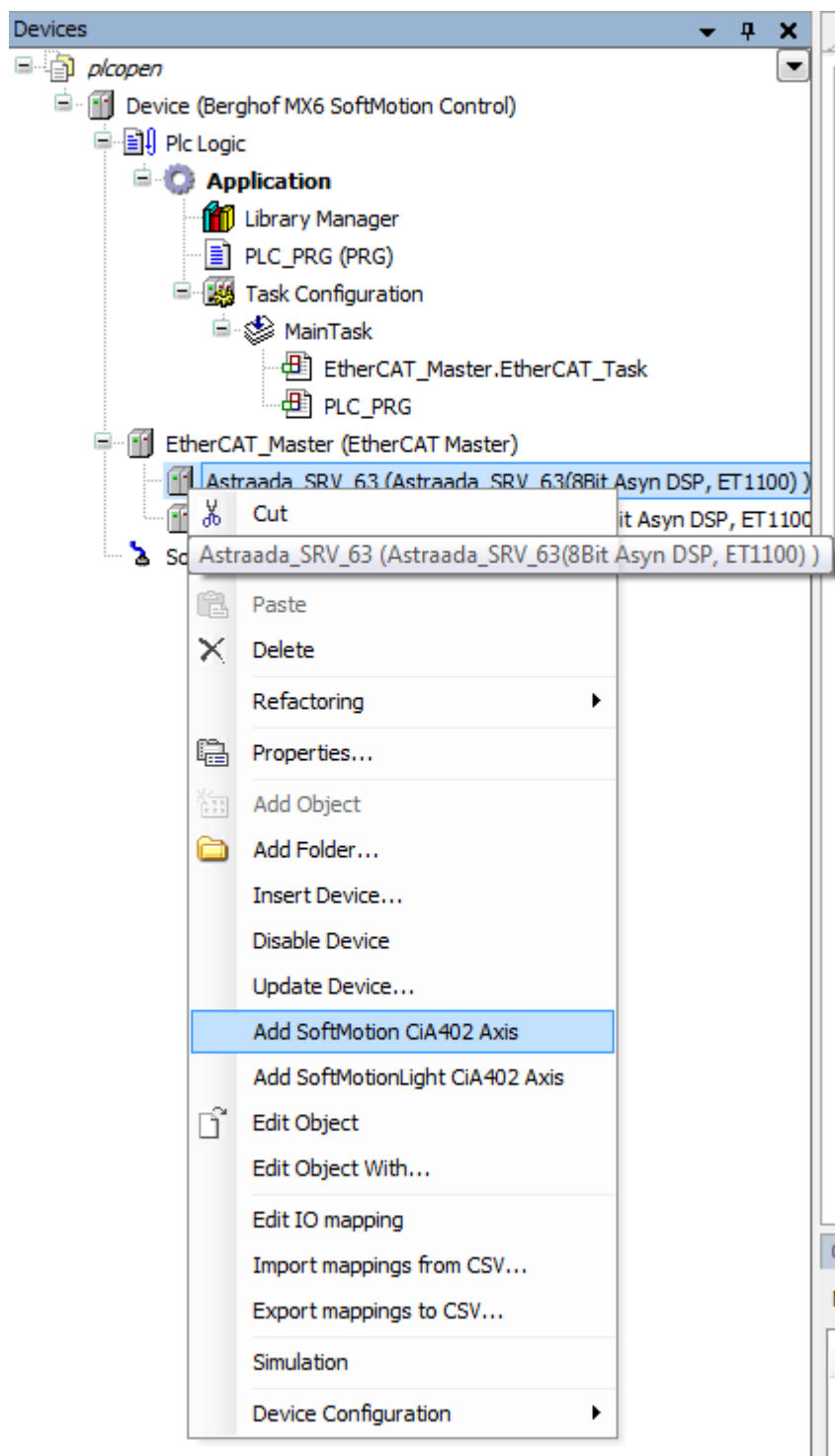
Łączymy się ze sterownikiem, należy kliknąć PPM na EtherCAT Master i wybrać opcję Scan for Devices:



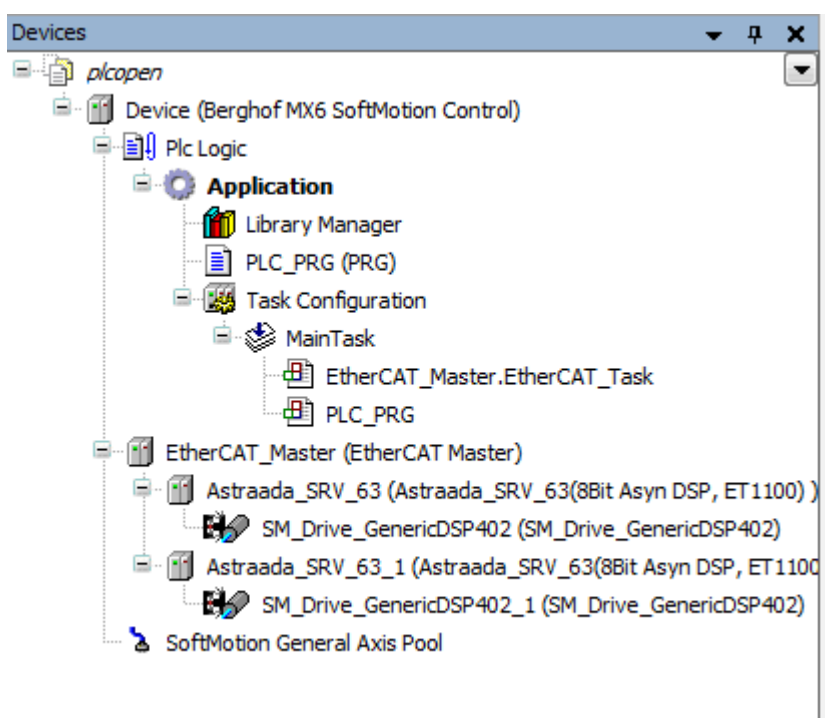
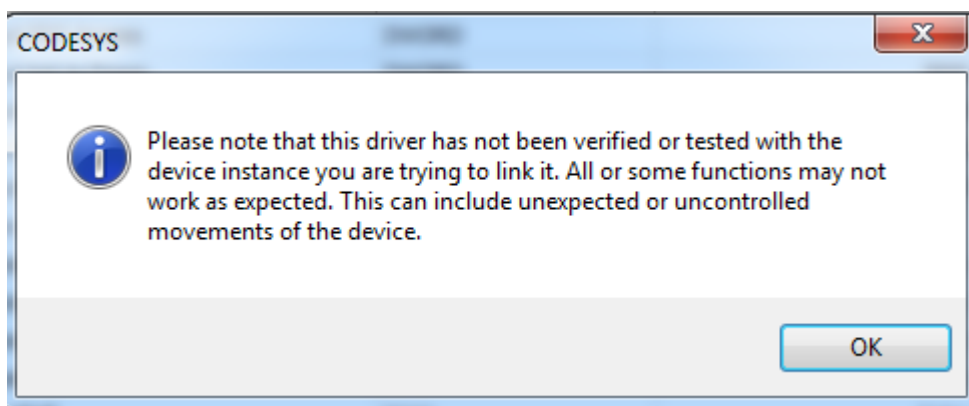


Po skopiowaniu urządzeń do projektu należy się wylogować ze sterownika.

Do urządzeń Astraada SRV należy dodać oś, w tym celu klikamy PPM na urządzenie i wybieramy Add SoftMotion CiA402 Axis:

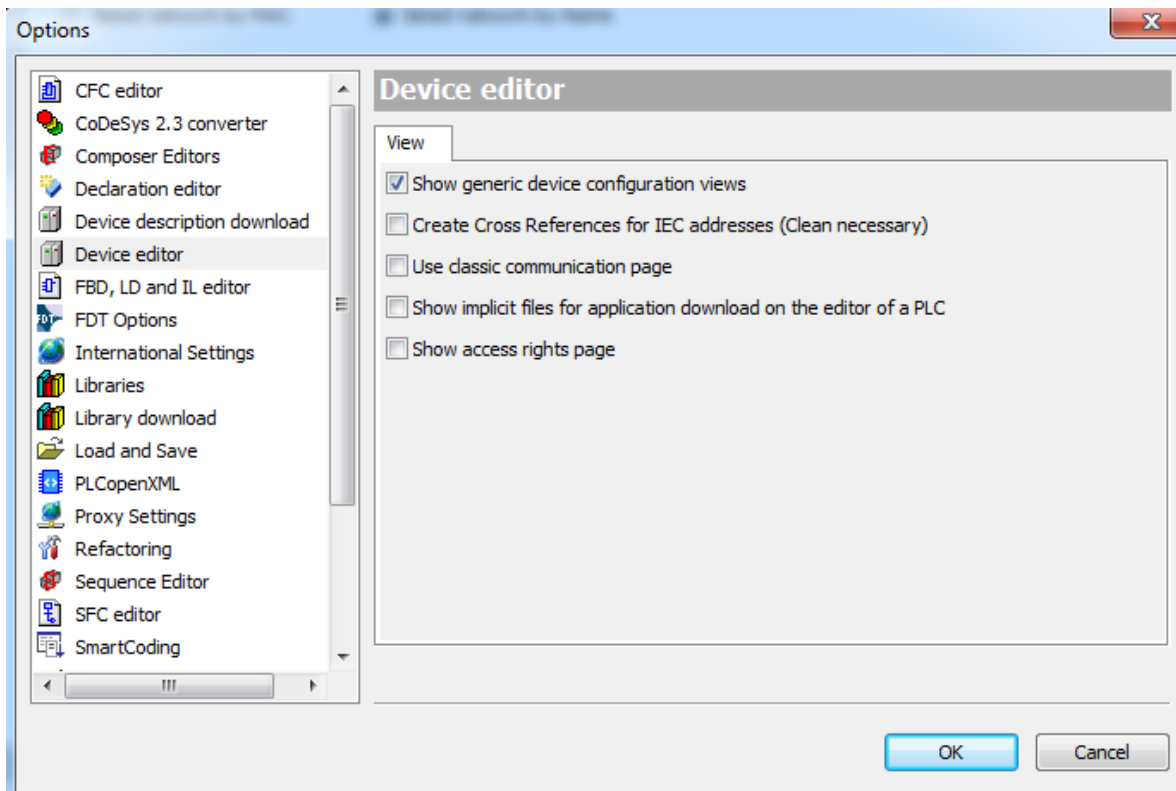


Z racji tego, że urządzenie Astraada SRV nie są certyfikowane w Codesysie pojawia się następujący komunikat:



Aby móc korzystać z bibliotek PLCopen w serwonapędach Astraada SRV należy odznaczyć dwie opcje parametryczne.

Najpierw należy przejść do zakładki Tools -> Options -> Device Editor i zaznaczyć opcję Show generic device configuration views:



Następnie dwukrotnie klikając w dodaną oś w drzewku projektowym należy przejść do zakładki SM_Drive_ETC_GenericDSP402: Parmaters:

Parameter	Type	Value	Default Value	Unit
ScalingGearOutput2	DINT	1	1	
ScalingUnits	DINT	1	1	
InvertDirection	BOOL	FALSE	FALSE	
logical device settings				
logical device number	USINT	0	0	
standard driver settings				
Set60C2	BOOL	TRUE	TRUE	
AXIS_REF: DSP402 configuration				
DSP402._bImmediateDisabling	BOOL	FALSE	FALSE	
DSP402._uiPreHomingWait	UINT	10	10	
DSP402._uiPostHomingWait	UINT	0	0	
DSP402._uiHomingMinCycles	UINT	0	0	
DSP402._uiWaitCyclesForStateSwitch	UINT	1000	1000	
DSP402._bPreHomingWaitBit12Clear	BOOL	FALSE	FALSE	
DSP402._bCheckBit10PostHoming	BOOL	TRUE	TRUE	
DSP402._bCheckOpMode	BOOL	TRUE	TRUE	
DSP402._bForbidReenableDuringDisabling	BOOL	FALSE	FALSE	
DSP402._bCheckBit12InPositionMode	BOOL	TRUE	TRUE	
DSP402._bDoHaltWhenStopInterruptsHome	BOOL	TRUE	TRUE	
DSP402._bCheckBit13InHomingMode	BOOL	TRUE	TRUE	
possible cyclic driver in-/outputs				
NumberOfOutputMappingParams	INT	9	9	
ControlWord (out.wControlWord)	INT	10064	10064	
Object_8010	STRING	16 #6040: 16 #00	16 #6040: 16 #00	
AdrOffset_8010	UDINT	0		
BitOffset_8010	USINT	0		

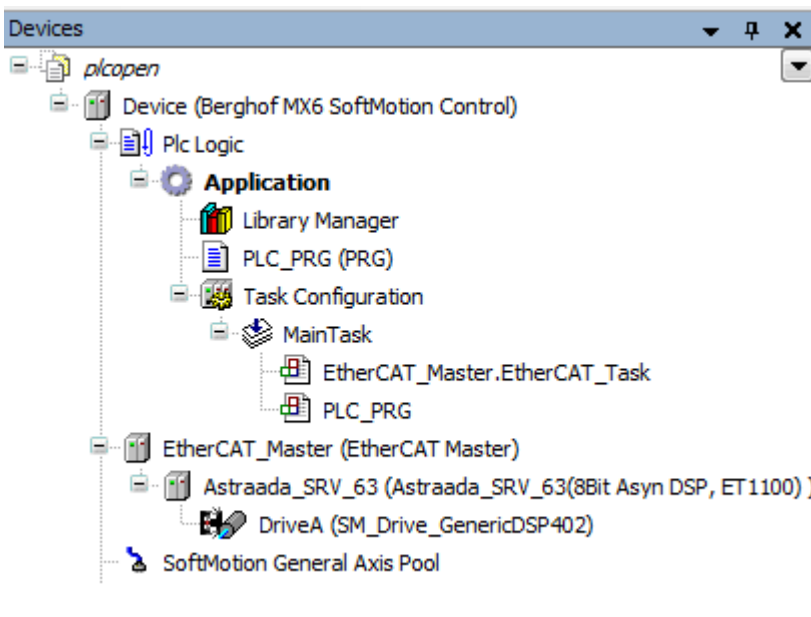
Należy przestawić paramter DSP402._bCheckOpMode na FALSE

General	Parameter	Type	Value	Default Value	Unit
Scaling/Mapping	ScalingGearOutput2	DINT	1	1	
	ScalingUnits	DINT	1	1	
Commissioning	InvertDirection	BOOL	FALSE	FALSE	
SM_Drive_ETC_GenericDSP402: Parameters	logical device settings				
	logical device number	USINT	0	0	
SM_Drive_ETC_GenericDSP402: I/O Mapping	standard driver settings				
	Set60C2	BOOL	TRUE	TRUE	
Status	AXIS_REF: DSP402 configuration				
Information	DSP402._bImmediateDisabling	BOOL	FALSE	FALSE	
	DSP402._uiPreHomingWait	UINT	10	10	
	DSP402._uiPostHomingWait	UINT	0	0	
	DSP402._uiHomingMinCycles	UINT	0	0	
	DSP402._uiWaitCyclesForStateSwitch	UINT	1000	1000	
	DSP402._bPreHomingWaitBit12Clear	BOOL	FALSE	FALSE	
	DSP402._bCheckBit10PostHoming	BOOL	TRUE	TRUE	
	DSP402._bCheckOpMode	BOOL	FALSE	TRUE	
	DSP402._bForbidReenableDuringDisabling	BOOL	FALSE	FALSE	
	DSP402._bCheckBit12InPositionMode	BOOL	FALSE	TRUE	
	DSP402._bDoHaltWhenStopInterruptsHome	BOOL	TRUE	TRUE	
	DSP402._bCheckBit13InHomingMode	BOOL	TRUE	TRUE	
	possible cyclic driver in-/outputs				
	NumberOfOutputMappingParams	INT	9	9	

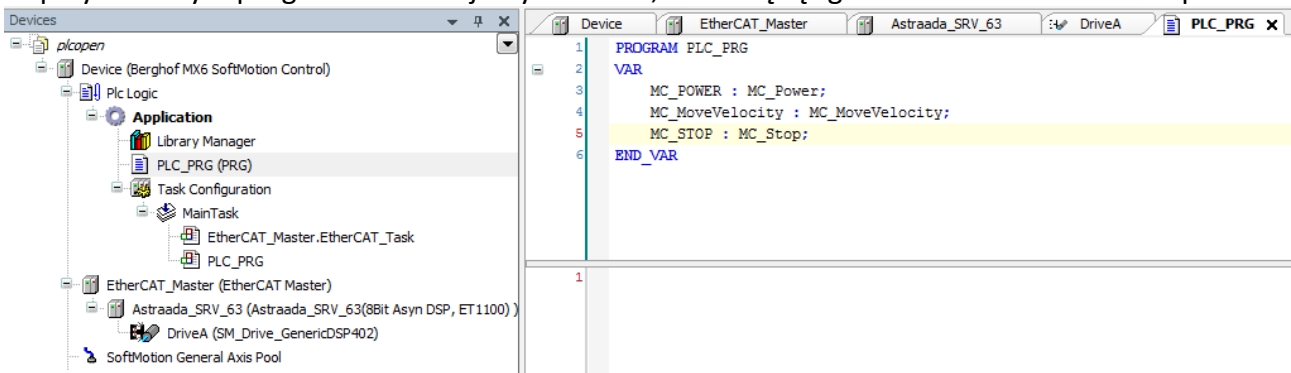
oraz DSP402._bCheckBit12InPositionMode na FALSE:

General	Parameter	Type	Value	Default Value	Unit
Scaling/Mapping	ScalingGearOutput2	DINT	1	1	
	ScalingUnits	DINT	1	1	
Commissioning	InvertDirection	BOOL	FALSE	FALSE	
SM_Drive_ETC_GenericDSP402: Parameters	logical device settings				
	logical device number	USINT	0	0	
SM_Drive_ETC_GenericDSP402: I/O Mapping	standard driver settings				
	Set60C2	BOOL	TRUE	TRUE	
Status	AXIS_REF: DSP402 configuration				
Information	DSP402._bImmediateDisabling	BOOL	FALSE	FALSE	
	DSP402._uiPreHomingWait	UINT	10	10	
	DSP402._uiPostHomingWait	UINT	0	0	
	DSP402._uiHomingMinCycles	UINT	0	0	
	DSP402._uiWaitCyclesForStateSwitch	UINT	1000	1000	
	DSP402._bPreHomingWaitBit12Clear	BOOL	FALSE	FALSE	
	DSP402._bCheckBit10PostHoming	BOOL	TRUE	TRUE	
	DSP402._bCheckOpMode	BOOL	FALSE	TRUE	
	DSP402._bForbidReenableDuringDisabling	BOOL	FALSE	FALSE	
	DSP402._bCheckBit12InPositionMode	BOOL	FALSE	TRUE	
	DSP402._bDoHaltWhenStopInterruptsHome	BOOL	TRUE	TRUE	
	DSP402._bCheckBit13InHomingMode	BOOL	TRUE	TRUE	

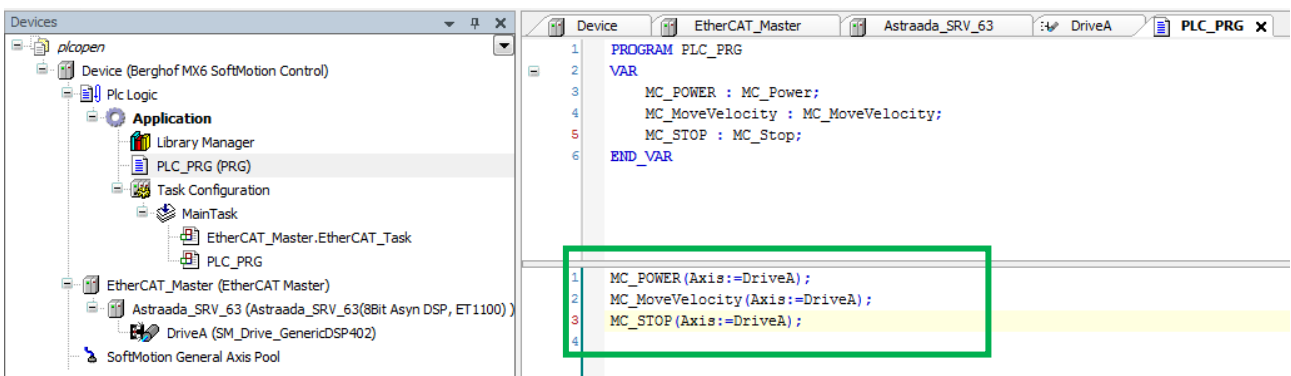
Następnie można według potrzeb zmienić nazwę napędu. W tym przykładzie będzie to nazwa DriveA:



W przykładowym programie definiujemy zmienne, które będą zgodne ze standardem PLCopen:



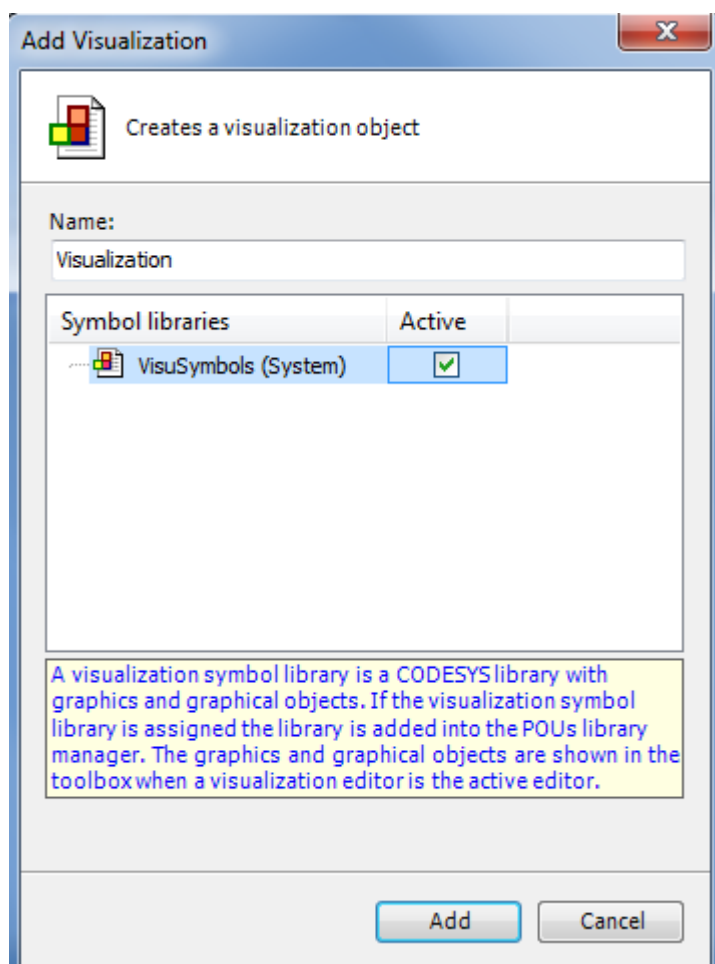
Następnie nadajemy niezbędne wejścia dla poszczególnych bloków:

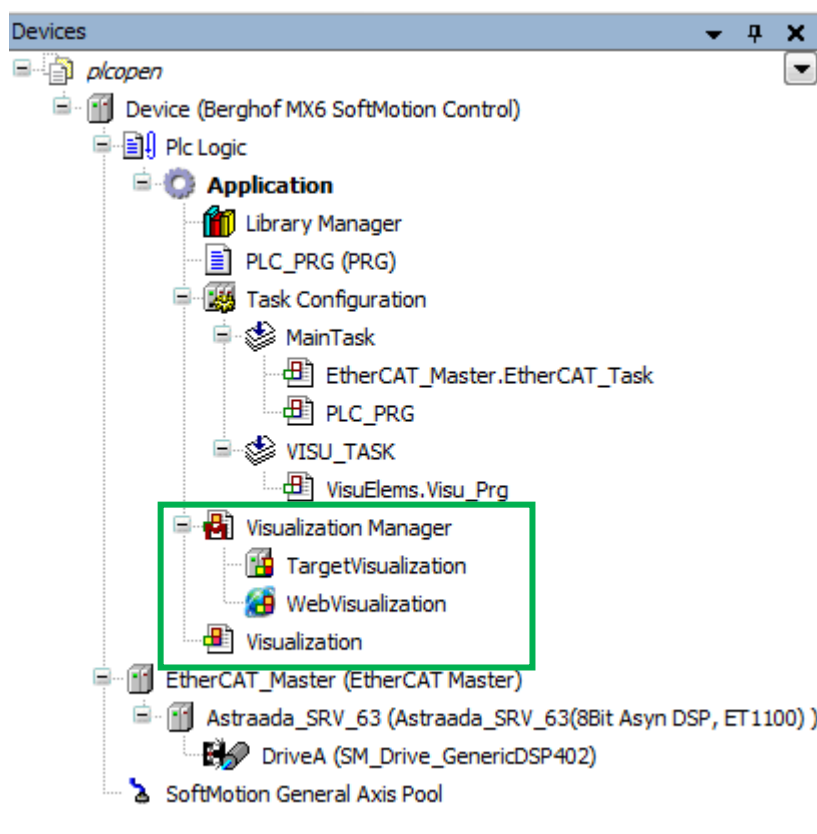


Przygotowanie wizualizacji

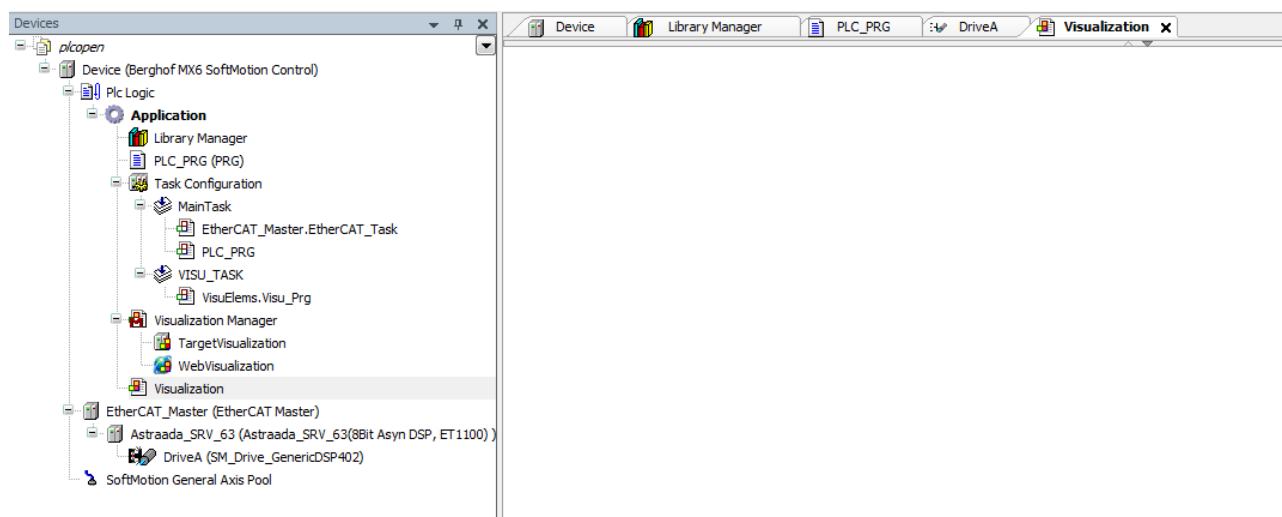
Należy kliknąć PPM na Application i wybrać Visualization:

W nowszych wersjach Codesysa pojawi się okno, pytające użytkownika czy chce aby dodane zostały dodatkowe symbole graficzne:

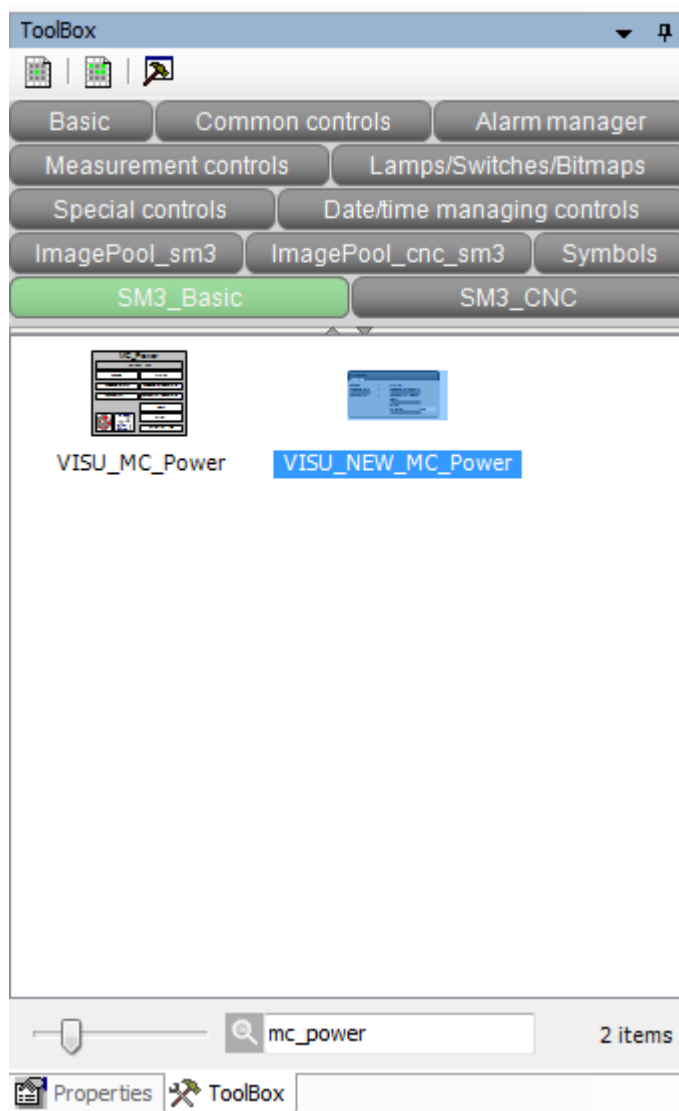




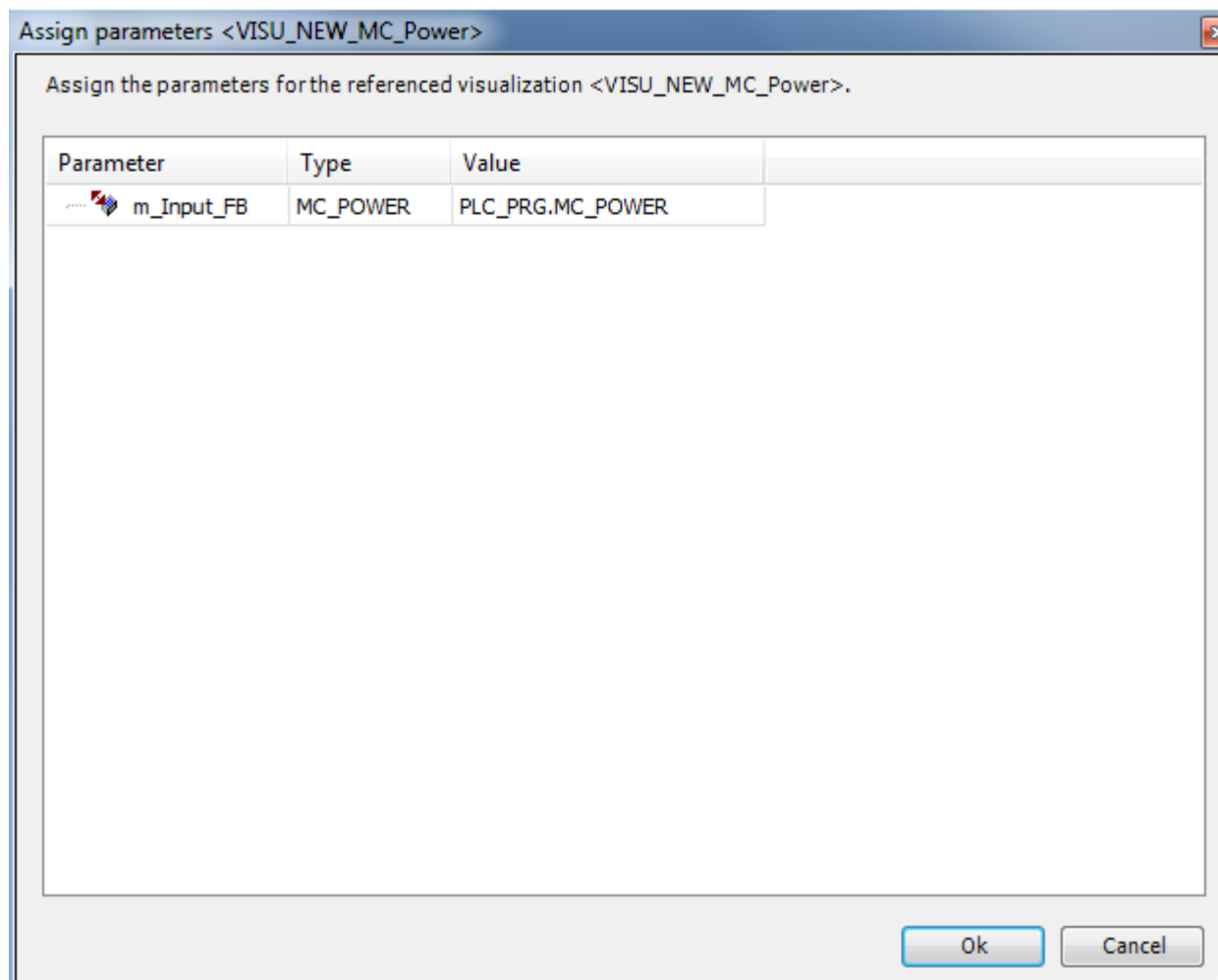
Dwukrotne kliknięcie w Visualization pozwoli na otwarcie pierwszego okna wizualizacji:



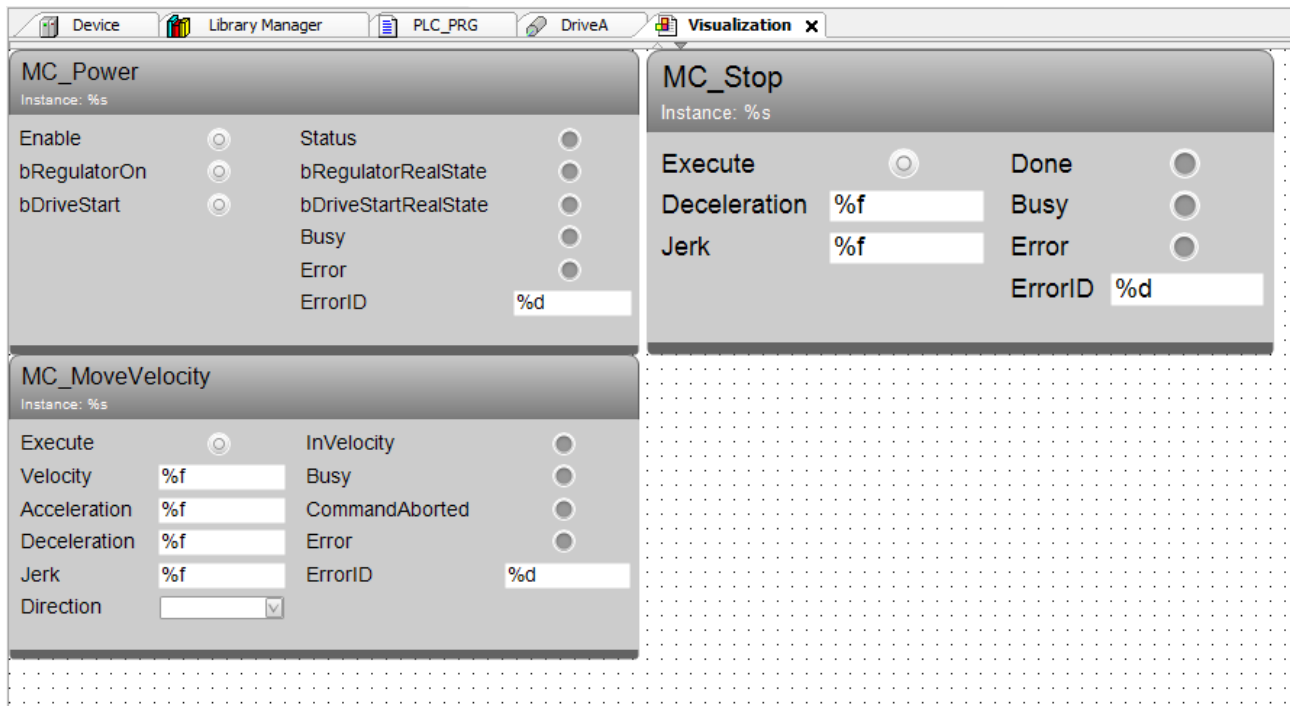
Wybieramy obiekty graficzne zgodnie z napisanym programem. W prawej części okna, z zakładki ToolBox wyszukujemy odpowiednio MC_POWER, MC_MoveVelocity oraz MC_STOP i przeciągamy na okno wizualizacji:



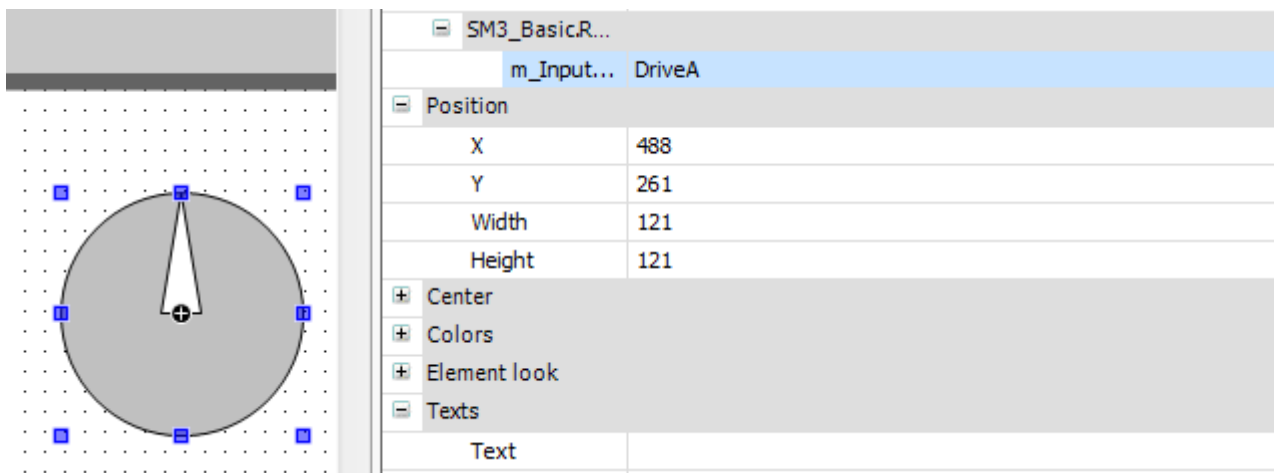
Po przeciągnięciu elementu przypinamy do niego zmienną z programu, odpowiadającą danemu typowi:



Analogicznie postępujemy z kolejnymi bloczkami:

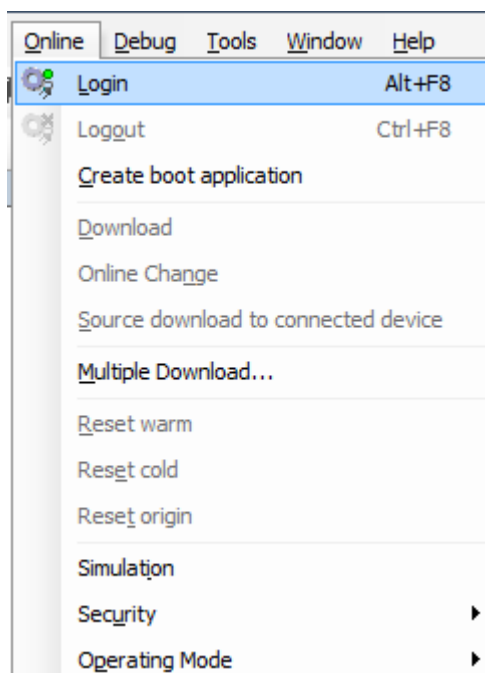


Dodamy jeszcze element odpowiadający za ruch osi, do którego przypinamy "zmienną"/oś DriveA:

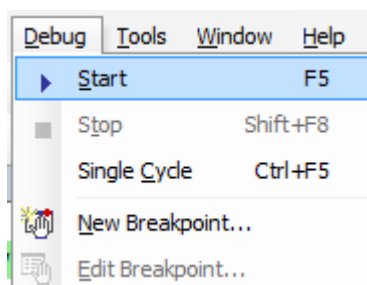


Uruchomienie aplikacji

Wgrywamy aplikację do sterownika, Online -> Login:

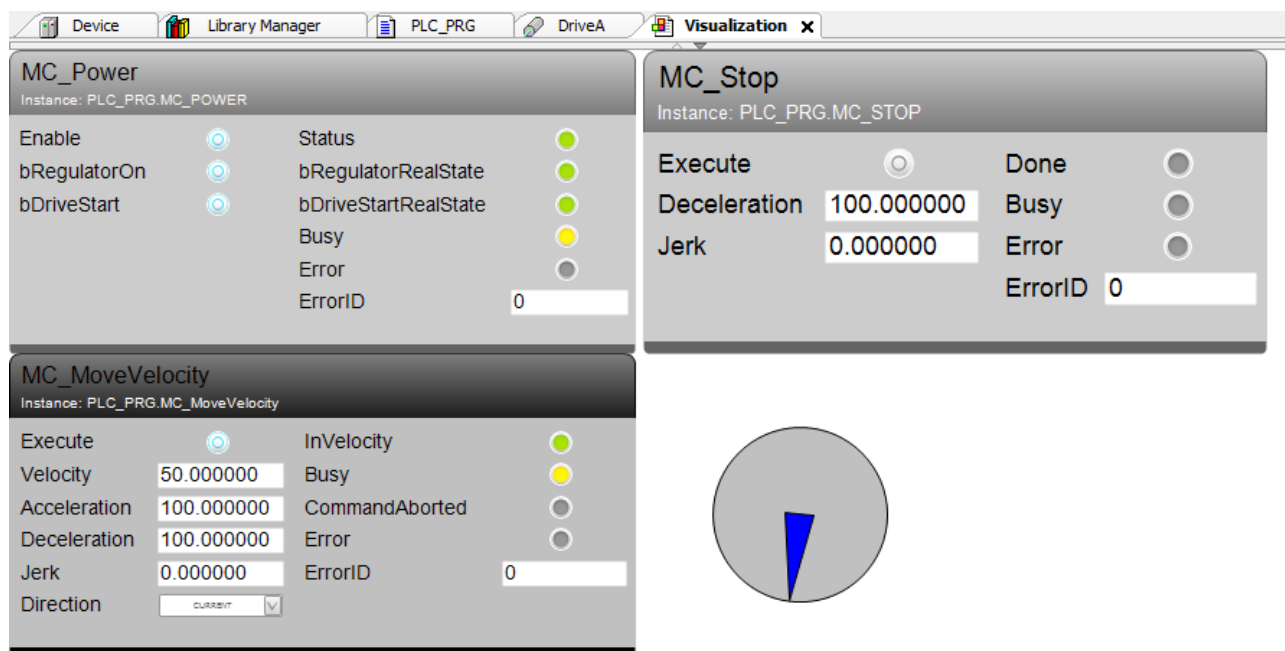


I przechodzimy w tryb RUN sterownika Debug -> Start:



Po wgraniu otworzy się okno gdzie możemy przetestować aplikację, również można to zrobić z oprogramowania Codesys.

Załączamy MC_POWER, zadajemy parametry ruchu: prędkość, przyspieszenie, hamowanie i testujemy działanie:



W tym przykładzie zastosowano tylko 3 bloki oparte na standardzie PLCopen. Natomiast z każdego innego można korzystać, konfigurując je w ten sam sposób.