



## **Projektowanie Układów Sterowania**

### **Laboratorium 5**

#### **Instrukcja – sterownik PLC, panel operatora Część I**

## SPIS TREŚCI

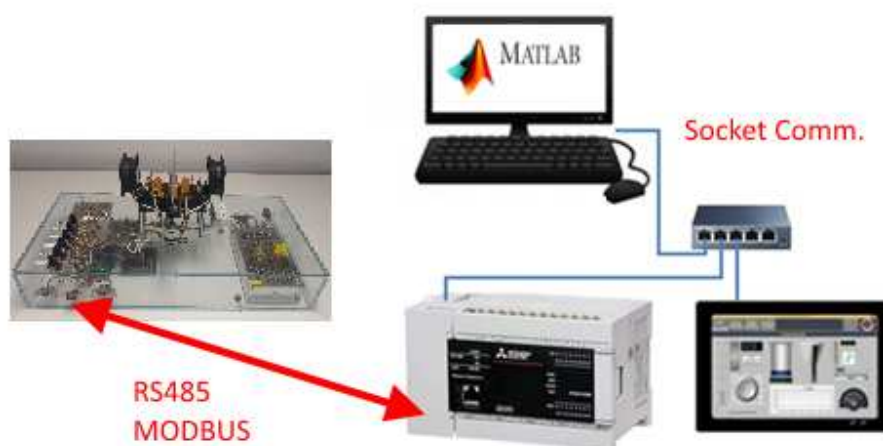
<b>1</b>	<b>CEL ĆWICZENIA .....</b>	<b>3</b>
<b>2</b>	<b>WYKAZ SPRZĘTU ORAZ OPROGRAMOWANIA.....</b>	<b>3</b>
<b>3</b>	<b>TWORZENIE KODU STERUJĄCEGO STEROWNIKA PLC W ŚRODOWISKU GXWORKS3.....</b>	<b>3</b>
3.1	TWORZENIE NOWEGO PROJEKTU.....	4
3.2	ELEMENTY JĘZYKA FBD/LD .....	5
3.3	DEFINICJA ZMIENNYCH .....	9
3.4	TWORZENIE KODU STERUJĄCEGO.....	10
3.5	KONFIGURACJA STEROWNIKA .....	12
3.6	PROGRAMOWANIE STEROWNIKA.....	14
3.7	DIAGNOSTYKA, MONITOROWANIE DZIAŁANIA PROGRAMU .....	15
3.8	PIERWSZY PROGRAM PLC.....	16
3.9	PIERWSZY PROGRAM PLC – REGULACJA CIĄGŁA.....	17
<b>4</b>	<b>DEFINICJA TABLICY TYPU FLOAT .....</b>	<b>21</b>
<b>5</b>	<b>OPIS KOMUNIKACJI RS485 MODBUS, SOCKET COMMUNICATION.....</b>	<b>22</b>
<b>6</b>	<b>TWORZENIE GRAFIK OPERATORSKICH W ŚRODOWISKU GT DESIGNER 3 .....</b>	<b>26</b>
6.1	PROJEKT DEMO .....	26
6.2	PANEL MENU – 1.....	26
6.3	PANEL WEWY – 2.....	28
6.4	PANEL PAMIEC – 3.....	28
6.5	PANEL MANUAL – 4 .....	29
6.6	PANEL PROCES – 5.....	30
6.7	PANEL WYKRES – 6.....	30
6.8	PANEL AUTOMAT – 7.....	31
6.9	PANEL AUTOMAT P – 8.....	32
6.10	PANEL AUTOMAT P – 10.....	33
6.11	WGRYWANIE PROJEKTU DO PANELA OPERATORA.....	34
<b>7</b>	<b>DOKUMENTACJA.....</b>	<b>35</b>

## 1 Cel ćwiczenia

Zapoznanie się ze sposobem tworzenia oprogramowania dla sterownika programowalnego FX5U firmy Mitsubishi oraz wizualizacji procesu na panelu operatora typu GOT Simple. W ramach ćwiczenia studenci tworzą projekt sterownika w środowisku GxWorks3 oraz wizualizację na panel operatora w środowisku GT Designer 3. Obiektem sterowania będzie stanowisko laboratoryjne grzejąco-chłodzące w konfiguracji wielowymiarowej. Komunikacja ze stanowiskiem będzie się odbywać przy pomocy RS485 protokołem MODBUS. Do odbierania i archiwizacji danych posłuży skrypt napisany w MATLAB na komputerze PC. Komunikacja z MATLAB będzie wykonana przy pomocy Socket Communication.

## 2 Wykaz sprzętu oraz oprogramowania

Stanowiska laboratoryjne składają się z zestawów dydaktycznych wyposażonych w sterownik PLC MELSEC FX5U firmy Mitsubishi, panel operatorski GOT, komputer stacjonarny oraz stanowisko badawcze.



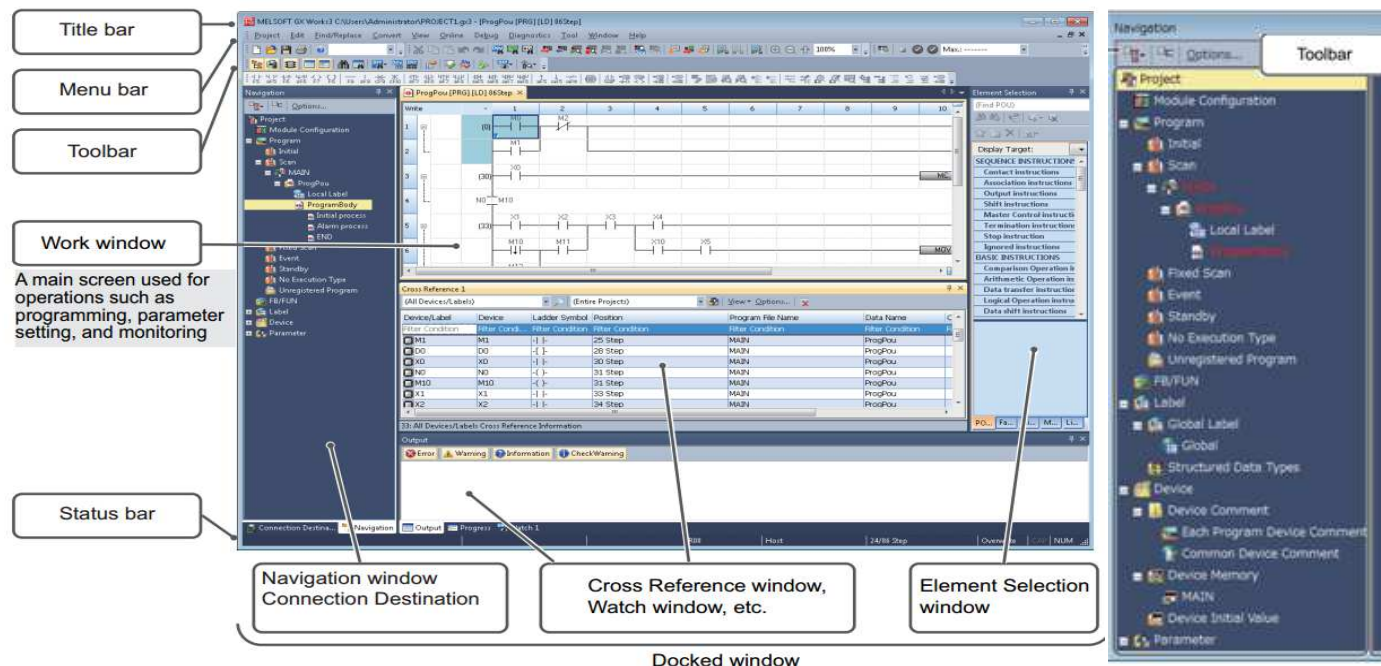
*Rysunek 1 Wyposażenie stanowisk laboratoryjnych*

W trakcie realizacji ćwiczenia wykorzystane zostanie następujące oprogramowanie:

- **GxWorks3** – oprogramowanie służące do tworzenia aplikacji dla sterownika PLC (*kompilacja, komunikacja ze sterownikiem, debug, alarmowanie*),
- **GT Designer 3** – oprogramowanie służące do tworzenia aplikacji dla panela operatora GOT Simple
- **MATLAB** – odbieranie danych ze sterownika PLC, rysowanie wykresów na komputerze.

## 3 Tworzenie kodu sterującego sterownika PLC w środowisku GxWorks3

### *Środowisko GxWorks3*



*Rysunek 2 Okno główne programu GxWorks3 (z lewej); Pasek nawigacji projektu (z prawej)*

### 3.1 Tworzenie nowego projektu

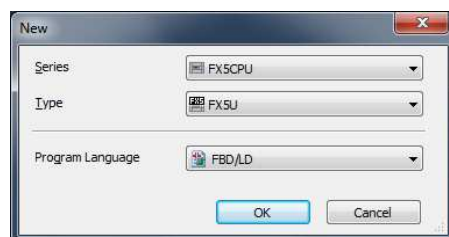
Proszę utworzyć nowy projekt ([Project] → [New]) (  )

Proszę zidentyfikować model oraz serię sterownika PLC znajdującego się na stanowisku. Na poniższym rysunku oznaczono czerwonym prostokątem miejsce, w którym znajduje się wymagana informacja.



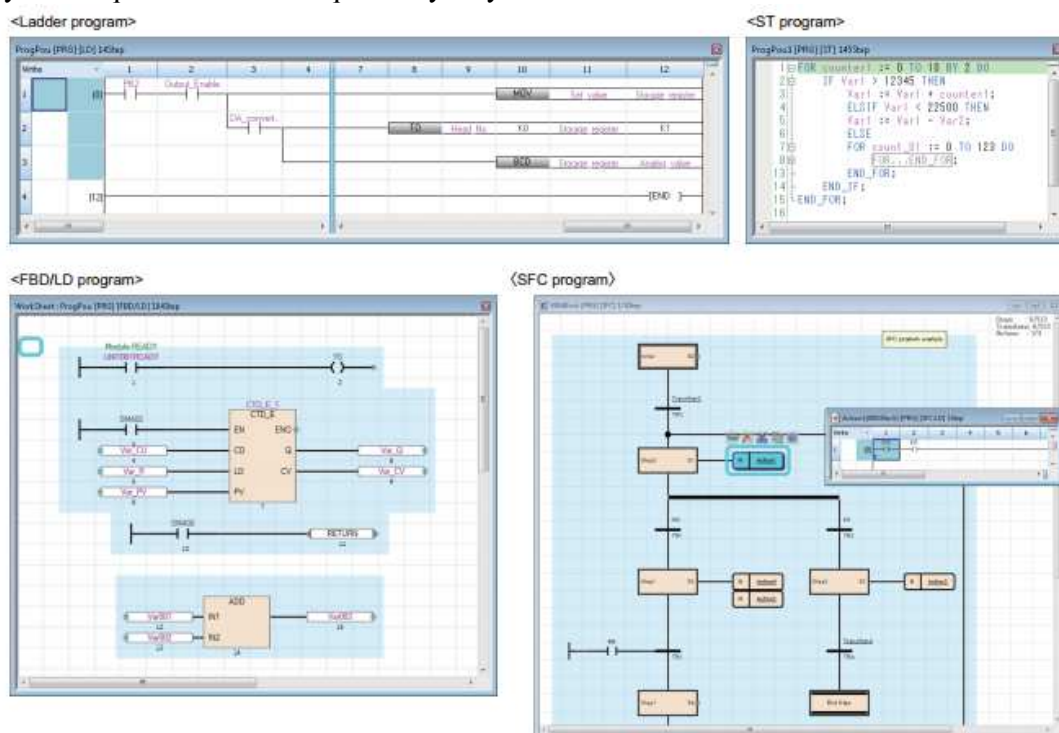
*Rysunek 3 Oznaczenie modelu i serii sterownika.*

Następnie w programie GxWorks3 proszę wprowadzić dane sterownika oraz wybrać język programowa **FBD/LD** po czym zatwierdzić ustawienia klikając przycisk OK.



**Rysunek 4** Parametry wstępne projektu – NIE POMYLIĆ SERII STEROWNIKA

Środowisko GxWorks3 wspiera programowanie zgodnie z normą IEC61131-3 (wsparcie dla: FBD/LD, Ladder Diagram, ST i SFC). Przykłady programów we wspomnianych językach zaprezentowano na poniższym rysunku.



**Rysunek 5** Języki programowania wspierane przez aplikację GxWorks3..

---

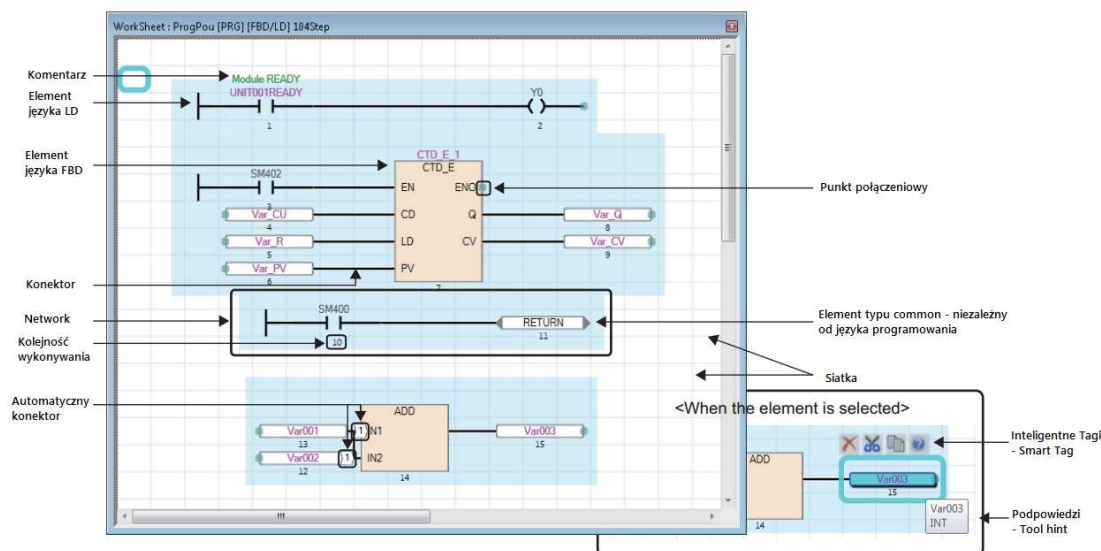
**Uwaga:** Proszę upewnić się, że projekt został zapisany [Project] → [Save] (💾)

---

### 3.2 Elementy języka FBD/LD

Pasek narzędziowy zawiera wszystkie elementy strukturalne języka FBD/LD:





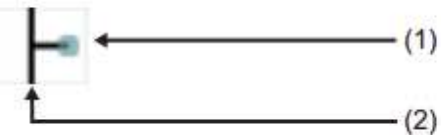
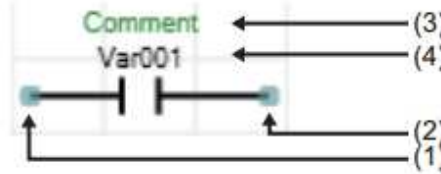
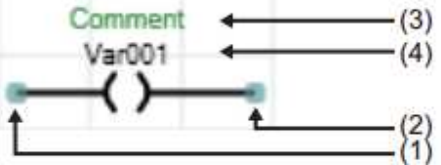
Rysunek 6 Edytor języka FBD/LD.

Element	Opis
Komentarz	Komentarz etykiety lub bloku funkcjonalnego. Nie podlega kompilacji.
Element języka LD	Element pochodzący z języka programowania Ladder Diagram
Element języka FBD	Element pochodzący z języka Function Block Diagram (FBD)
Element typu common	Element wbudowany, niezależny od języka programowania
Konektor	Linia łącząca punkty pomiędzy elementami programu. Możliwe jest automatyczne łączenie punktów poprzez zbliżenie bloków.
Network	Pojedyncza sieć zbudowana ze wszystkich elementów podłączonych razem. Program może zawierać maksymalnie 4096 networków.
Kolejność wykonywania (execution order)	Liczba określająca kolejność wykonania danego elementu programu.
Automatyczny konektor	Jeśli konektor nie może być wyświetlony w danym miejscu, wtedy zostaje zastąpiony liczbą.
Punkt podłączeniowy	Terminal (punkt) pozwalający na połączenie bloków/elementów programu poprzez konektor. Punkty powinny być łączone z uwzględnieniem typów danych.
Siatka	Linie siatki arkusza na którym umieszczane są elementy programu
Smart tag	Przyciski wyświetlane nad wybranym elementem, pozwalające na wykonanie operacji t.j np. usuwanie lub kopiowanie

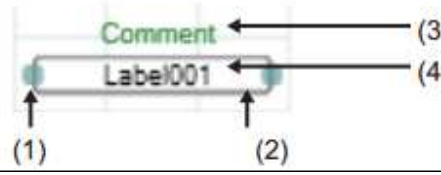
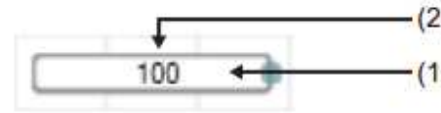


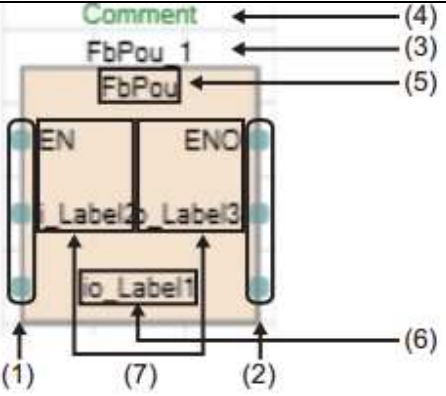
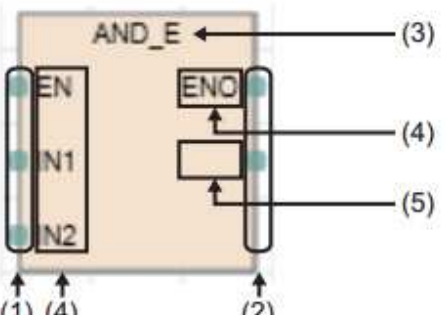
	elementu.
Tool hint	Informacja o elementach programu wyświetlana po najechaniu kursorem myszki

### Elementy języka LD



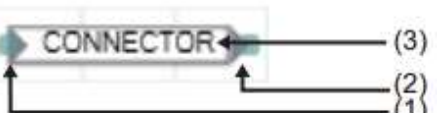
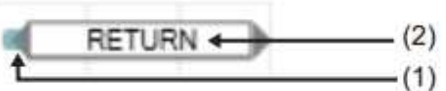
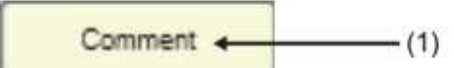
Element	Opis
<p>Lewa szyna zasilająca</p> 	<p>(1) Wyjściowy punkt połączeniowy (2) Lewa szyna zasilająca</p>
<p>Element stykowy</p>  <p>Cewka</p> 	<p>(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Komentarz urządzenia/etykiety (4) Urządzenie/etykieta</p>

### Elementy języka FBD

Element	Opis
<p>Zmienna (lokalna/globalna)</p> 	<p>(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Komentarz urządzenia/etykiety (4) Urządzenie/etykieta</p>
<p>Stała</p> 	<p>(1) Wyjściowy punkt połączeniowy (2) Stała wartość</p>
<p>Blok funkcyjny</p>	<p>(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Nazwa instancji FB (4) Komentarz etykiety (5) Typ danych</p>

	<p>(6,7) Etykieta wejścia/wyjścia</p> <p><b>Uwaga:</b> wyjścia z bloku mogą być umieszczone również z lewej strony symbolu bloku funkcyjnego</p>
<p>Funkcja</p> 	<p>(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Typ danych (4) Etykieta wejścia/wyjścia</p> <p><b>Uwaga:</b> wyjścia z bloku mogą być umieszczone również z lewej strony symbolu bloku funkcyjnego</p>

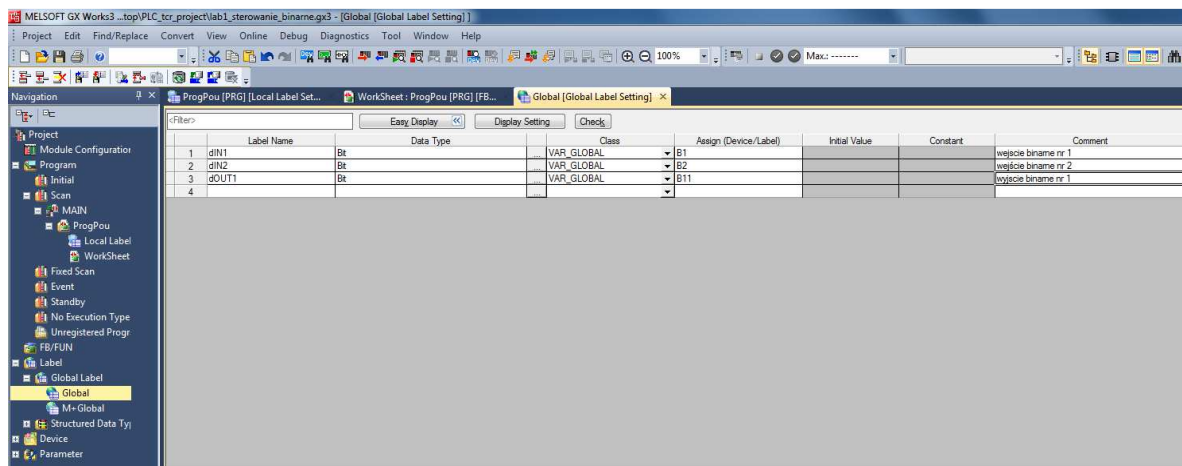
### Elementy wspólne (common element)

Element	Opis
<p>Instrukcja skoku (Jump element)</p> 	<p>(1) Wejściowy punkt połączeniowy (2) Etykieta</p>
<p>Etykieta instukcji skoku</p> 	<p>(1) Wyjściowy punkt połączeniowy</p>
<p>Konektor</p> 	<p>(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Komentarz etykiety</p>
<p>Instrukcja return</p> 	<p>(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy</p>
<p>Blok komentarza</p> 	<p>(1) Powierzchnia na której wyświetlona zostanie treść komentarza</p>



### 3.3 Definicja zmiennych

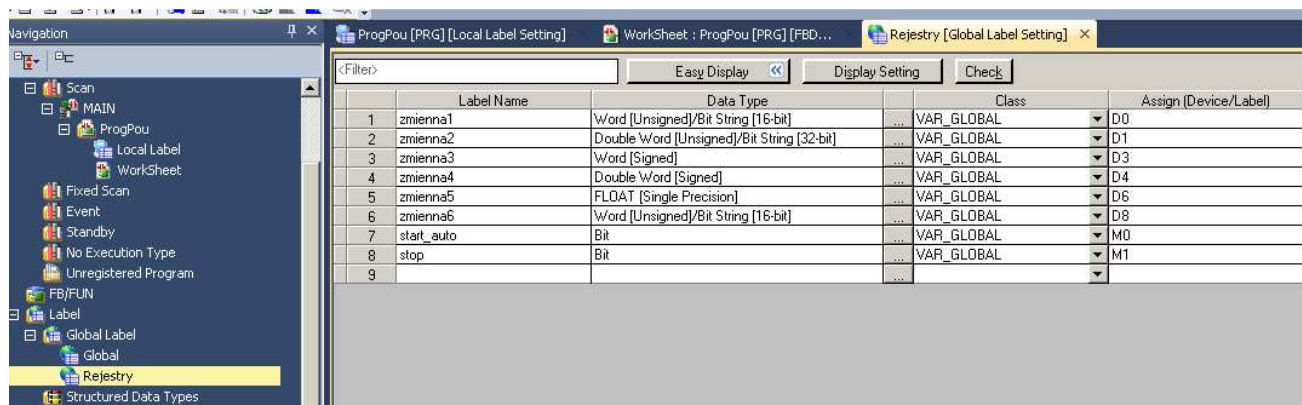
W celu tworzenia czytelnego kodu zalecane jest wykorzystanie zmiennych symbolicznych zamiast adresowania bezpośredniego. W tym celu należy wykorzystać „Labeli”, które mogą mieć zakres lokalny (widoczne tylko w danym komponencie) lub globalny (widoczne w całym systemie i propagowane po sieci – to rozwiązanie jest zalecane z uwagi na możliwość przypisania fizycznych urządzeń, które później będą używane w panelu operatora). Definicja „Labeli” możliwa jest poprzez wypełnienie tabeli (patrz rysunek poniżej) lub poprzez bezpośrednie definiowanie w trakcie tworzenia kodu sterującego - wpisanie nazwy nowej zmiennej symbolicznej w miejscu jej użycia powoduje otwarcie okna dialogowego, gdzie możliwa jest konfiguracja zmiennej.



*Rysunek 7 Deklaracja lokalnych/globalnych „Labeli”.*

Możliwe jest również tworzenie grup zmiennych tworząc pomocnicze kontenery (np. Wejścia, Wyjścia, Sygnały\_analogowe, Sygnały\_dyskretne itp.). Aby to zrobić należy w oknie Navigation przejść do zakładki Label->Global Label, następnie kliknąć prawym przyciskiem myszy i wybrać opcję Add New Data.

W trakcie laboratorium przydatne będą urządzenia typu Bit, Rejestr. Zakres urządzeń typu Bit zawiera się w zakresie od M0 do M7680 numerowane co jeden. Zakres urządzeń typu rejestr (16 bit) zawiera się w zakresie D0 do D7999 numerowane co jeden. Proszę zwrócić szczególną uwagę, że zmienne typu Double lub Float zajmują dwa rejestry D. Rysunek 7a przedstawia przykładową konfigurację. W programie PLC można używać „Labeli”, ale do elementów w panelu operatora należy używać bezpośrednich adresów pamięci zadeklarowanych w kolumnie Assign.

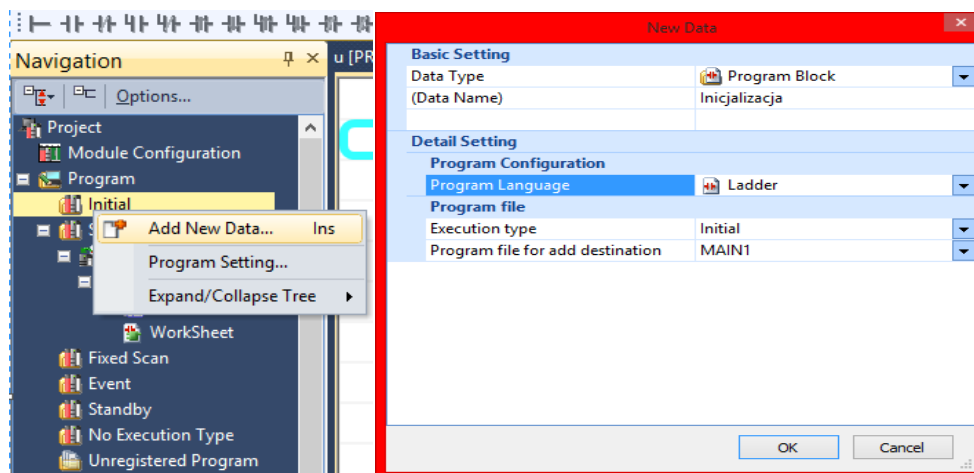


*Rysunek 8a Deklaracja globalnych „Labeli”.***3.4 Tworzenie kodu sterującego**

W zależności od sposobu wykonywania programu sterującego należy określić jego lokalizację w drzewie projektu. Wspierane są następujące sekcje:

- Initial - instrukcje wykonywane tylko w pierwszym cyklu sterownika,
- Scan - główny skan procesora, czas cyklu zależny obciążenia,
- Fixed scan - skan z narzuconym okresem wykonania (czas konfigurowalny),
- Event - obsługa zdarzeń,
- No execution Type - magazyn kodu, który nie jest wykonywany.

W każdej z sekcji można stworzyć kilka podprogramów klikając w sekcję prawym przyciskiem myszy a następnie wybierając z menu opcję „Add New Data” (Patrz poniżej).



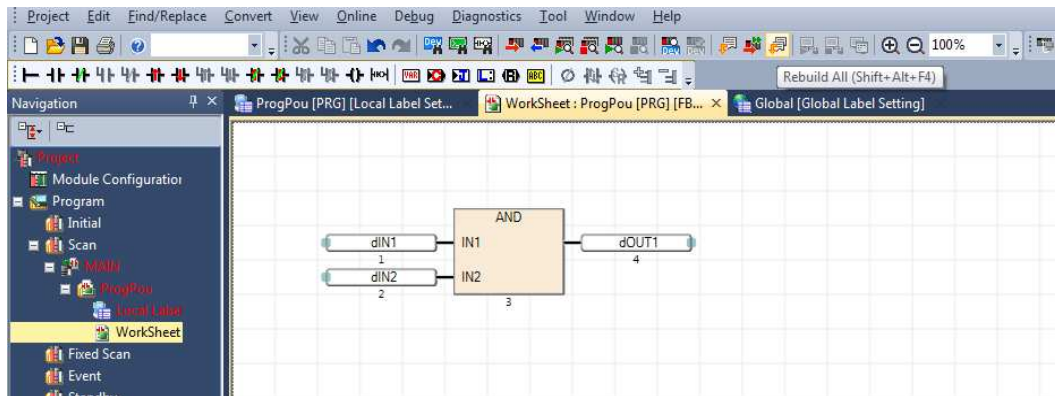
*Rysunek 9 Dodawanie nowych podprogramów*

Wstawianie elementów języka FBD na arkusz roboczy może odbywać się dzięki technice „drag and drop” z biblioteki, lub poprzez bezpośrednie wpisywanie z klawiatury nazwy bloku funkcyjnego. W trakcie wpisywania kolejnych znaków podpowiedzi o dostępnych blokach są wyświetlane pod tworzoną blokiem.

---

**Uwaga:** Należy zwrócić szczególną uwagę na kolejność wykonywania algorytmów.

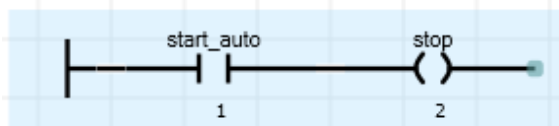
---



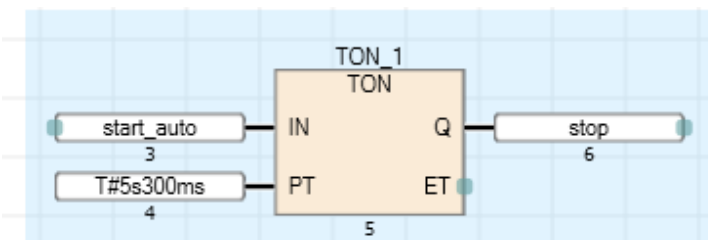
Rysunek 10 Przykład tworzenia kodu sterującego.

W czasie laboratorium najbardziej użyteczne będą następujące instrukcje:

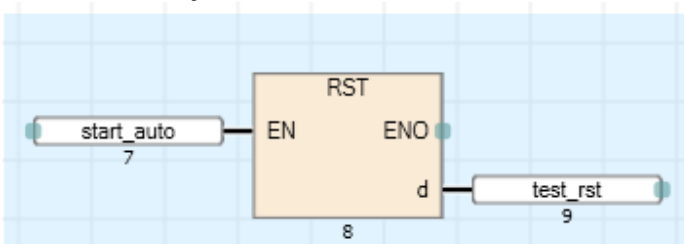
1. Styk normalnie otwarty
2. Cewka wyjściowa (należy pamiętać, że w programie powinna być tylko jeden raz do jednej zmiennej)



3. Opóźnienie załączenia TON, opóźnienie wyłączenia TOF, impuls o zadanym czasie TP



4. Instrukcja ustawienia SET, kasowania RST



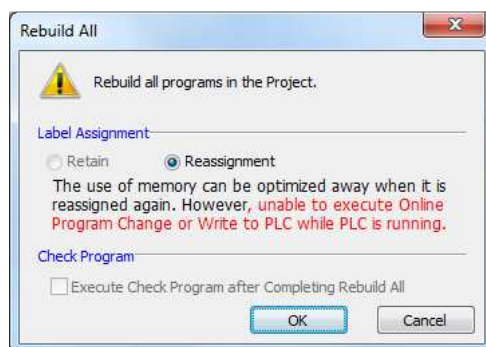
Należy unikać stosowania nazw zmiennych, które mogą być nazwami własnymi zastrzeżonymi w programie np. STOP, ST. Mogą one oznaczać nazwy instrukcji lub nazwy urządzeń fizycznych.

## Kompilacja kodu



- 1 – Kompilacja (po małych modyfikacjach)
- 2 – Rekompilacja (po zmianach konfiguracyjnych)

Rekompilacja wymaga potwierdzenia komunikatu:

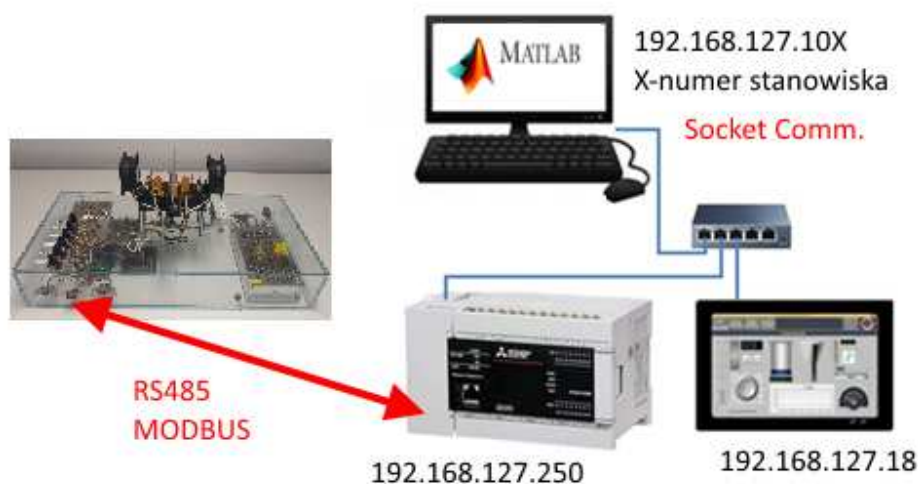


*Rysunek 11 Rekompilacja – okno dialogowe*

Po rekompilacji projektu nie możliwe będzie ładowanie sterownika w trybie Online. Z tego powodu drobne zmiany w programie należy zatwierdzać bezpośrednio zapisując projekt i wywołując komendę „Online Program Change”.

### 3.5 Konfiguracja sterownika

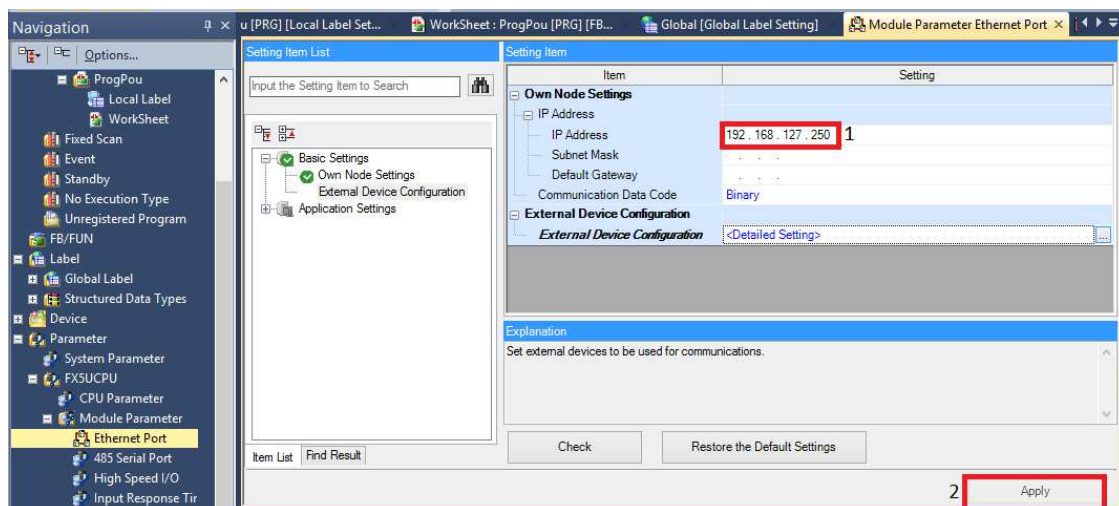
W celu umożliwienia komunikacji sterownika z panelem GOT oraz komputerem (MATLAB) należy skonfigurować jego ustawienia sieciowe. Poniższe instrukcję przeprowadzają przez wymagane operacje.



*Rysunek 12 Adresacja urządzeń w sieci lokalnej stanowiska*

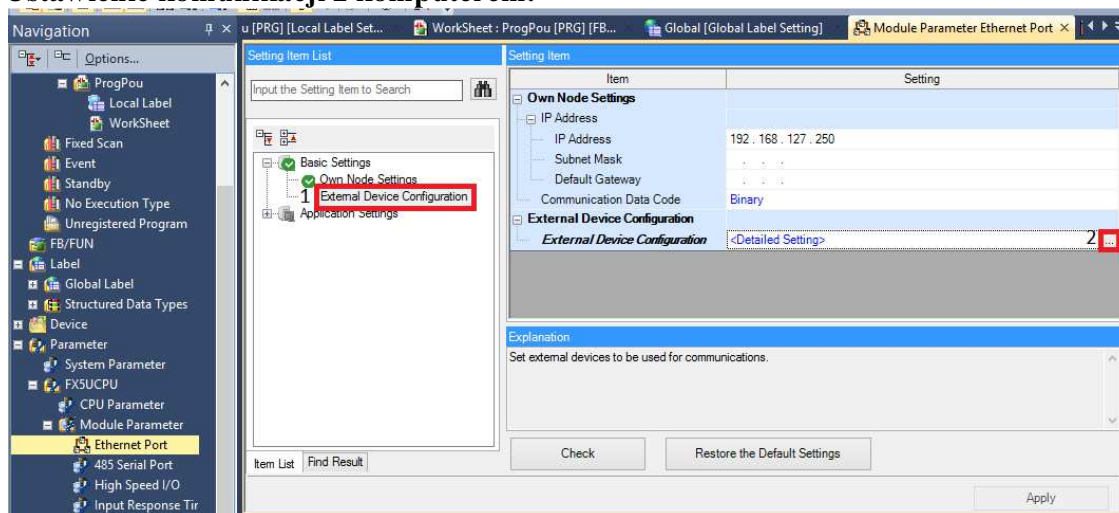
Ustawienie adresu IP sterownika:

## PUST – Projektowanie Układów Sterowania

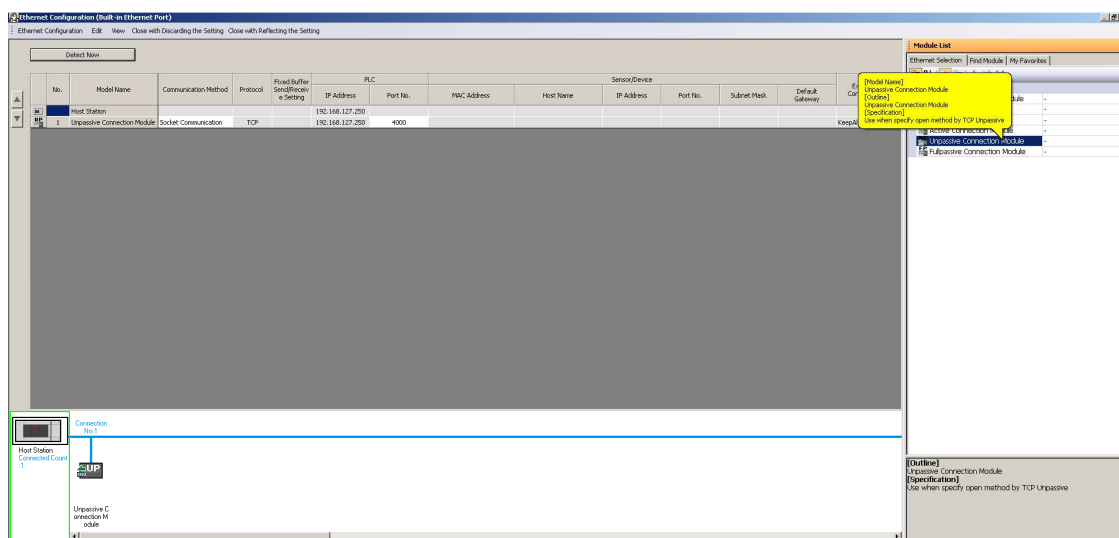


*Rysunek 13 Ustawienie adresu IP portu Ethernet*

### Ustawienie komunikacji z komputerem:



*Rysunek 14 Wywołania okna konfiguracji zewnętrznej komunikacji*

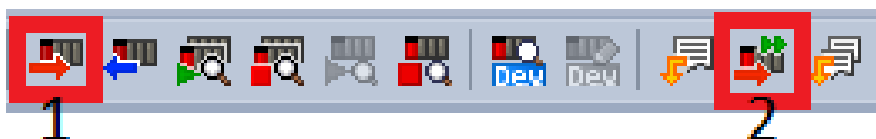




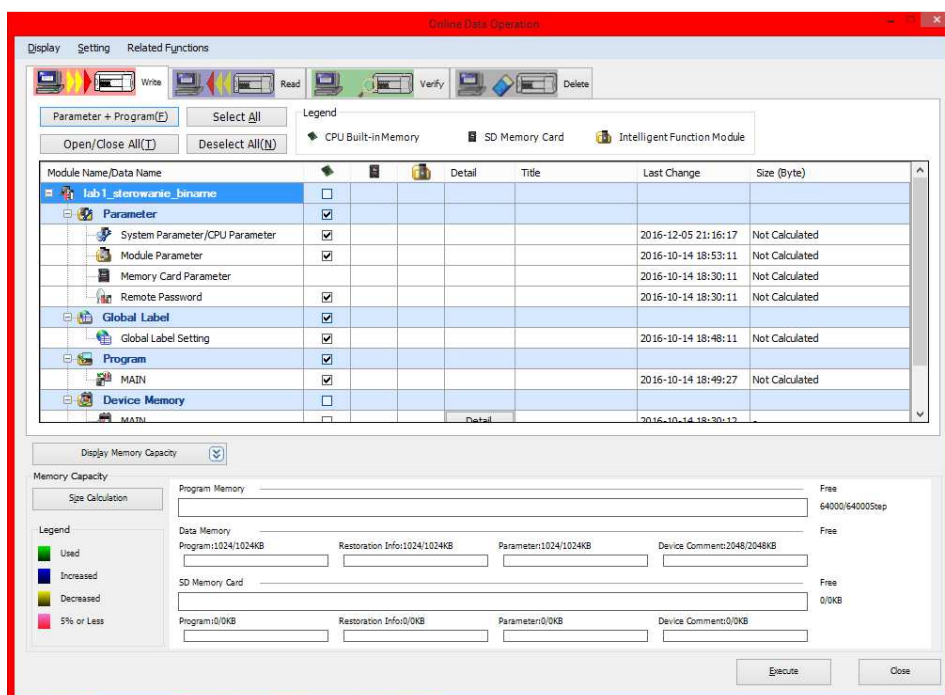
**Rysunek 15** Dodanie komunikacji Unpassive (port 4000)

### 3.6 Programowanie sterownika

Zmiany konfiguracyjne wymagają pełnego ładowania sterownika z ręcznym restartem. W tym celu należy skompilować projekt i wybrać opcję „Write to PLC” (opcja 1 z poniższego rysunku). Do wprowadzenia szybkich zmian na sterowniku (np. modyfikacja logiki kontrolera) bez restartu kontrolera należy wykorzystać opcję „Online Program Change” (opcja 2 z poniższego rysunku). Operacja ta nie może być poprzedzona kompilacją, gdyż ta odbywa się automatycznie przed aktualizacją programu sterującego.

**Rysunek 16** Operacja ładowania sterownika.

Wybór opcji „Write to PLC” przekierowuje do okna „Online Data Operation”, gdzie można przeprowadzić operacje: zapisu, odczytu, weryfikacji oraz czyszczenia pamięci kontrolera.

**Rysunek 17** Okno Online Data Operation – Zapis/Odczyt/Verifyfikacja/Czyszczenie sterownika.

**Uwaga 1:** Przed operacją ładowania kontrolera należy upewnić się że projekt nie zawiera błędów



**Uwaga 2:** Jeżeli przy próbie wgrywania programu do sterownika otrzymamy komunikat błędu „Inconsistency..... ” należy wówczas przejść do zakładki Delete, wybrać wszystkie elementy przez Select All i wcisnąć Execute (nastąpi usunięcie starych parametrów i programów ze sterownika). Następnie należy powrócić do zakładki Write i przez Select All a następnie Execute wgrać nowy program i parametry do sterownika.

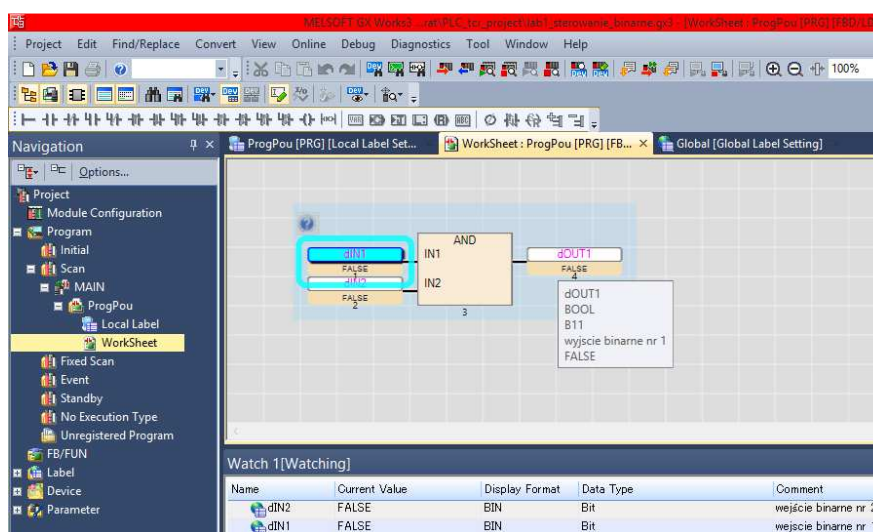
**Uwaga 3:** Po wykonaniu operacji wgrania parametrów i programu należy wykonać sprzętową RESET sterownika PLC. Wykonuje się to przez otwarcie pokrywki po lewej stronie sterownika, przełączenie dźwigi z pozycji RUN do RESET, przytrzymanie dźwigi do momentu pojawienia się diody ERR na sterowniku a następnie powrót do pozycji RUN. W tym momencie sterownik został zresetowany i można kontynuować pracę.

### 3.7 Diagnostyka, monitorowanie działania programu

Po załadowaniu kontrolera możliwy jest podgląd wykonywania programu za pomocą opcji „Start Monitoring”.

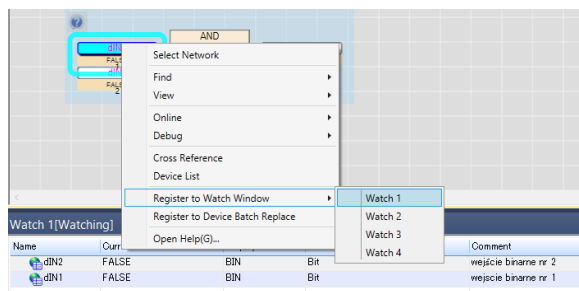


Rysunek 18 Uruchamianie Monitora.



Rysunek 19 Podgląd wykonywania programu

W celu zmiany/wyświetlania wartości zmiennych należy dodać je do podglądu za pomocą mechanizmu Watch'a. Należy najechać kursorem na nazwę zmiennej a następnie prawym przyciskiem myszy wybrać otworzyć menu i wybrać Register to Watch Window -> Watch 1. Z poziomu okienka Watch można zmieniać wartości zmiennych w celu testowania działania programu.



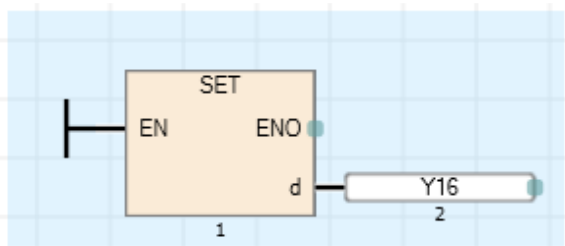
Rysunek 20 Uruchomienie Watch'a.

### 3.8 Pierwszy program PLC

Pierwszy program wgrywany do sterownika powinien obejmować:

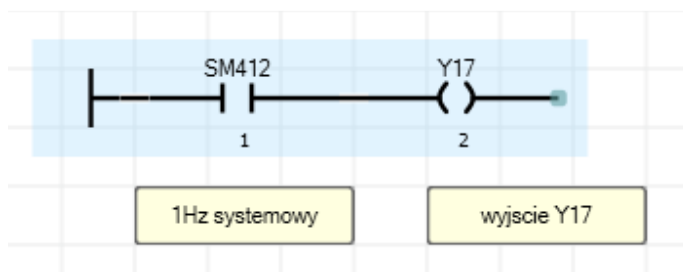
1. ustawienie adresu IP sterownika PLC na 192.168.127.250
2. konfigurację komunikacji dla MATLAB poprzez dodanie Unpassive TCP connection i ustawienie portu 4000
3. dodanie programu w sekcji INIT – inicjalizacja zmiennych dla symulacji procesów i regulatorów
4. dodanie programu w sekcji SCAN – operacje wykonywane cyklicznie ze skanem procesora
5. dodanie programu FIXED SCAN – operacje wykonywane cyklicznie ze skanem 1000ms (czas dyskretyzacji procesów regulacji i regulatorów)

Program przykładowy w sekcji INIT:



Po uruchomieniu sterownika lub jego resecie po wgraniu programu powinno aktywować się wyjście Y16, co można zaobserwować na zielonych diodach na sterowniku.

Program przykładowy w sekcji SCAN:



Program powinien mrugać wyjściem Y17 zgodnie z zegarem wewnętrznym 1Hz.

Później ten program posłuży do stworzenia pierwszego powiązania panela operatora ze sterownikiem PLC.

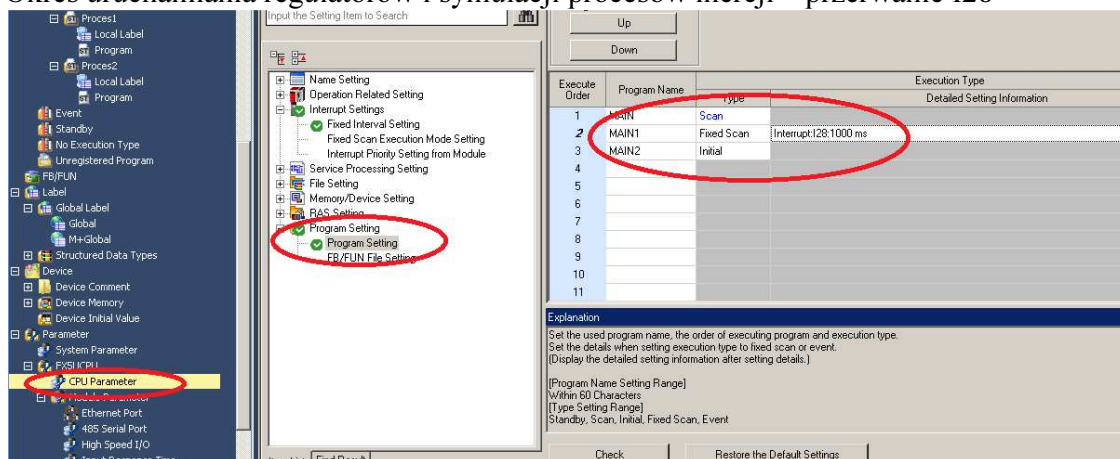
Po utworzeniu programu wstępnego należy go skompilować opcją Rebuild All a następnie wgrać do sterownika. Po wykonaniu restartu sterownika wyjście Y16 powinno się zapalić a wyjście Y17 powinno cyklicznie się zmieniać.

### 3.9 Pierwszy program PLC – regulacja ciągła

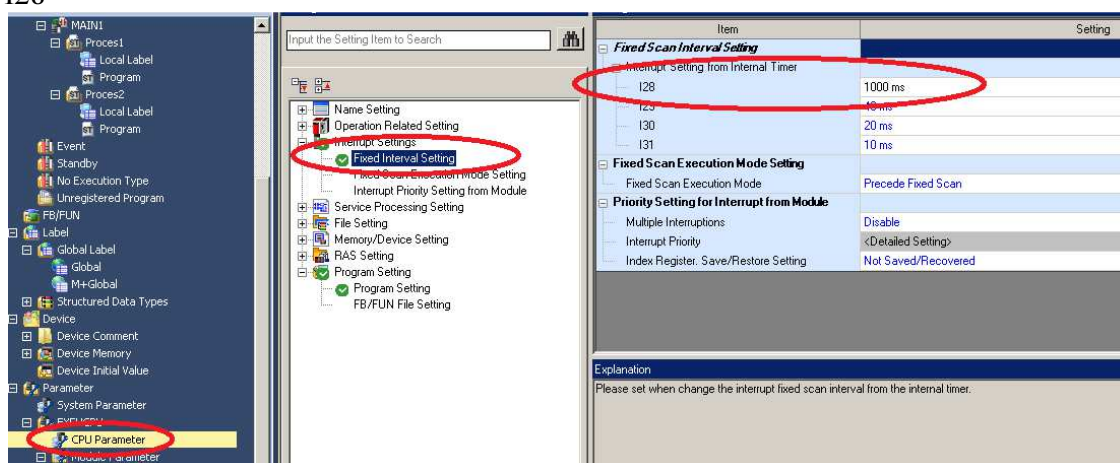
W tej części zostanie dokładnie omówiona część programowa sterownika PLC, która pozwoli na realizację regulatora (PID wbudowany, PID z równania różnicowego). Na laboratorium należy uzupełnić podany przykładowy program.

Poniżej przedstawiono najważniejsze punkty programów.

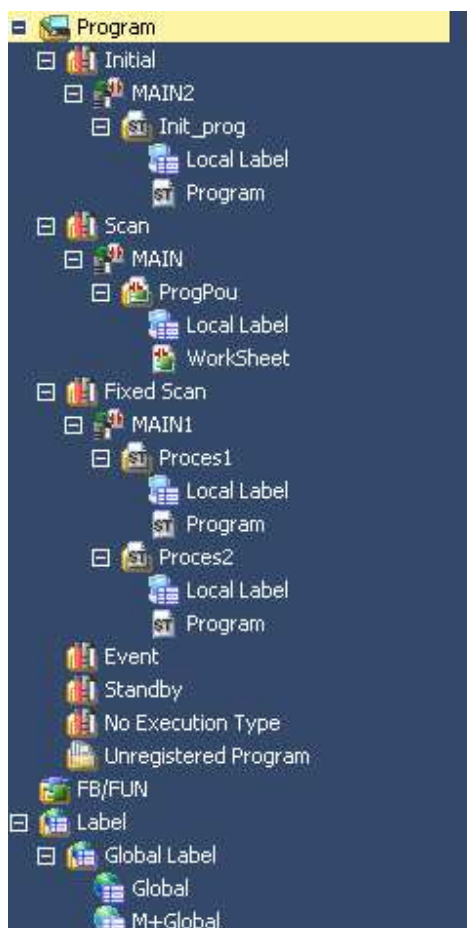
Okres uruchamiania regulatorów i symulacji procesów inercji – przerwanie I28



Okres uruchamiania regulatorów i symulacji procesów inercji – edycja okresu przerwania I28



Struktura przygotowanej części programów



Program Init\_prog – kod źródłowy w języku ST

//Okres probkowania 1s=1000ms - parametr w programie FIXED SCAN  
EMOV(TRUE, 1.0, okres\_probkowania);

// Ustawienie wartosci początkowych procesu 1  
EMOV(TRUE, 5.0, stala\_czasowa1);  
EMOV(TRUE, 10.0, K\_p\_proces1);

// Ustawienie wartosci początkowych procesu 2  
EMOV(TRUE, 9.0, stala\_czasowa2);  
EMOV(TRUE, 3.0, K\_p\_proces2);

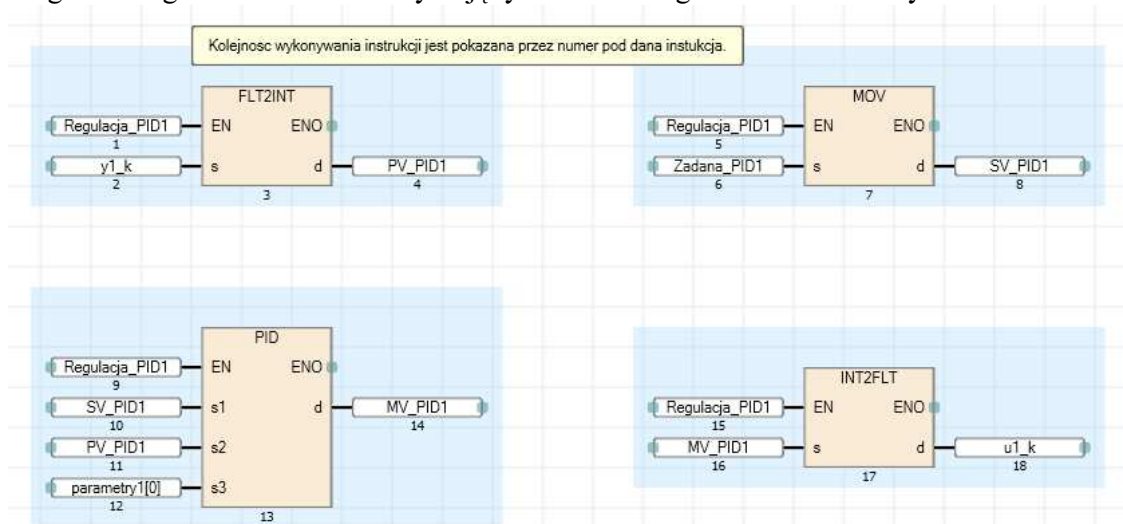
//Parametry regulatora wbudowanego PID  
parametryl[0] := K1000; //okres regulacji w milisekundach  
parametryl[3] := K1; //wzmocnienie regulatora P  
parametryl[4] := K0; //TI = 0 oznacza nieskonczony czas całkowania - inaczej mowiac całkowanie wylaczone  
parametryl[5] := K0; //KD = 0 oznacza zerowe wzmocnienie rozniczkiowania  
parametryl[6] := K0; //TD = 0 oznacza wylaczone rozniczkiowanie  
parametryl[22] := 100; //gorny limit wartosci wyjsciowej z regulatora - zapobiega rowniez efektowi wind-up

```
parametry1[23] := 0; //dolny limit wartosci wyjsciowej z regulatora - -||-
SET(TRUE, parametry1[1].5); //aktywacja limitow na wyjsciu regulatora
```

```
//Parametry regulatora z dyskretnego rownania roznicowego
```

```
K_PID3 := 1.0;
TI_PID3 := 99999.0;
TD_PID3 := 0.00001;
E0_PID3 := 0.0;
E1_PID3 := 0.0;
E2_PID3 := 0.0;
R0_PID3 := 0.0;
R1_PID3 := 0.0;
R2_PID3 := 0.0;
U_PID3 := 0.0;
Zadana_PID3 := 0;
```

Program ProgPou – kod źródłowy w języku FBD – regulator wbudowany PID



Definicje zmiennych globalnych – deklaracje parametrów regulatorów, procesów, zmiennych pomocniczych – należy pamiętać o prawidłowym przydzielaniu fizycznych rejestrów, aby później było możliwe odczytywanie zmiennych w systemie SCADA.

	Label Name	Data Type		Class	Assign (Device/Label)
1	SV_PID1	Word [Signed]	...	VAR_GLOBAL	▼ D2000
2	PV_PID1	Word [Signed]	...	VAR_GLOBAL	▼ D2001
3	MV_PID1	Word [Signed]	...	VAR_GLOBAL	▼ D2002
4	parametry1	Word [Unsigned]/Bit String [16-bit][0..29]	...	VAR_GLOBAL	▼ D2010
5	Regulacja_PID1	Bit	...	VAR_GLOBAL	▼ M0
6	Zadana_PID1	Word [Signed]	...	VAR_GLOBAL	▼ D2050
7			...		▼
8			...		▼
9	okres_probkowania	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1000
10			...		▼
11	stala_czasowa1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1002
12	Alfa1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1004
13	K_p_proces1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1006
14	u1_k	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1008
15	u1_k_1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1010
16	y1_k	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1012
17	y1_k_1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1014
18	A_p1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1016
19	stala_czasowa2	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1022
20	Alfa2	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1024
21	K_p_proces2	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1026
22	u2_k	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1028
23	u2_k_1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1030
24	y2_k	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1032
25	y2_k_1	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1034
26	A_p2	FLOAT [Single Precision]	...	VAR_GLOBAL	▼ D1036
27			...		▼

Program PID3 – kod źródłowy w języku ST – regulator PID opisany równaniem różnicowym – **do dokończenia**

//Regulator PID na podstawie rownania roznicowego

SV\_PID3 := INT\_TO\_REAL(Zadana\_PID3);  
PV\_PID3 := y3\_k;

//Wylczenie parametrow

R0\_PID3 := 1.0; //r0 = K\*( 1+(Tp/(2\*Ti))+Td/Tp );  
R1\_PID3 := 1.0; //r1 = K\*( (Tp/(2\*Ti))-(2\*Td/Tp)-1 );  
R2\_PID3 := 1.0; //K\*Td/Tp;

//Wylczenie uchybu regulacji i przesuniecie historii

E2\_PID3 := E1\_PID3;  
E1\_PID3 := E0\_PID3;  
E0\_PID3 := SV\_PID3 - PV\_PID3;

//Obliczenie sterowania

U\_PID3 := R2\_PID3\*E2\_PID3 + R1\_PID3\*E1\_PID3 + R0\_PID3\*E0\_PID3 + U\_PID3;  
//u = R2\*E2 + R1\*E1 + R0\*E0 + u;

IF (U\_PID3 > 100.0) THEN  
    U\_PID3 := 100.0;



```

END_IF;

IF (U_PID3 < 0.0) THEN
    U_PID3 := 0.0;
END_IF;

```

Alternatywnie łatwiejsza implementacja regulatora PID w języku ST:

Program w grupie SCAN:

```

PID(Reguluj, SV_PID, PV_PID, parametry, MV_PID);

MOV(Reguluj, MV_PID, D114); //wysterowanie grzałki 1
D110 := 400;                //wysterowanie wentylatora 1

MOV(Reguluj, D100, PV_PID);

```

Definicje zmiennych globalnych:

	Label Name	Data Type	Class	Assign (Device/Label)
1	SV_PID	Word [Signed]	VAR_GLOBAL	D4500
2	PV_PID	Word [Signed]	VAR_GLOBAL	D4501
3	parametry	Word [Unsigned]/Bit String [16-bit]	VAR_GLOBAL	D4000
4	MV_PID	Word [Signed]	VAR_GLOBAL	D4502
5	Reguluj	Bit	VAR_GLOBAL	
6				

Inicjalizacja regulatora:

```

//Ustawienia PID
D4000 := 100; //okres regulacji 100ms
D4022 := 1000; //limit gorny wartosci wyjsciowej
D4023 := 0;    //limit dolny wartosci wyjsciowej
SET(TRUE, D4001.5); //aktywacja limitow wyjsciowych – od razu antiwindup
D4003 := 10; //wzmocnienie regulatora
D4004 := 20; //stala czasowa calkowania regulatora
SET(TRUE, D4001.0); //aktywacja trybu grzania – znak petli sprzezenia
zwrotnego

```

#### 4 Definicja tablicy typu float

W pierwszej kolejności definiujemy strukturę, w której dodajemy jeden element typu wektor float o zadanej długości. Następnie dodajemy zmienną w wybranej grupie Labeli. Zmienna będzie typu stworzonej przed chwilą struktury. Zmienna powinna być zadeklarowana, jako wektor, dzięki czemu uzyskamy wektor wektorów – czyli tablicę dwuwymiarową. Przykład zapisu stałej wartości do tablicy:

```

tablica_MP[0].wiersz[0] := 1.321;

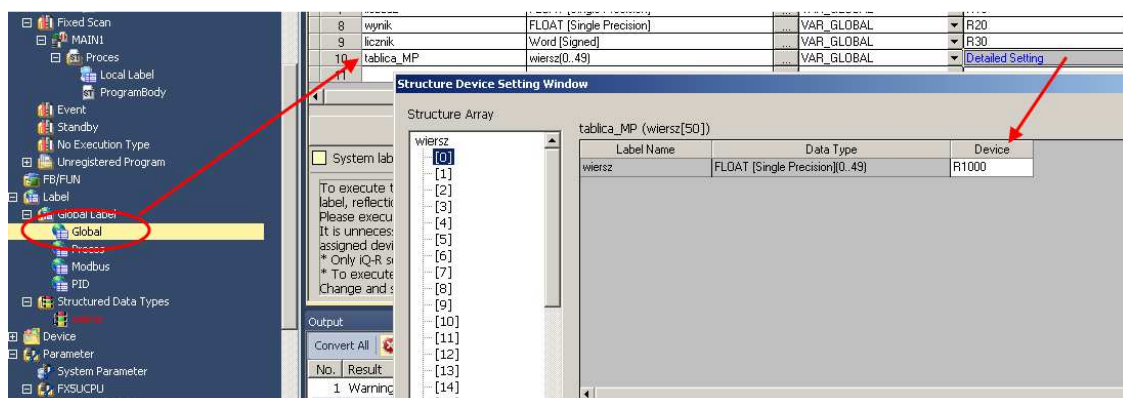
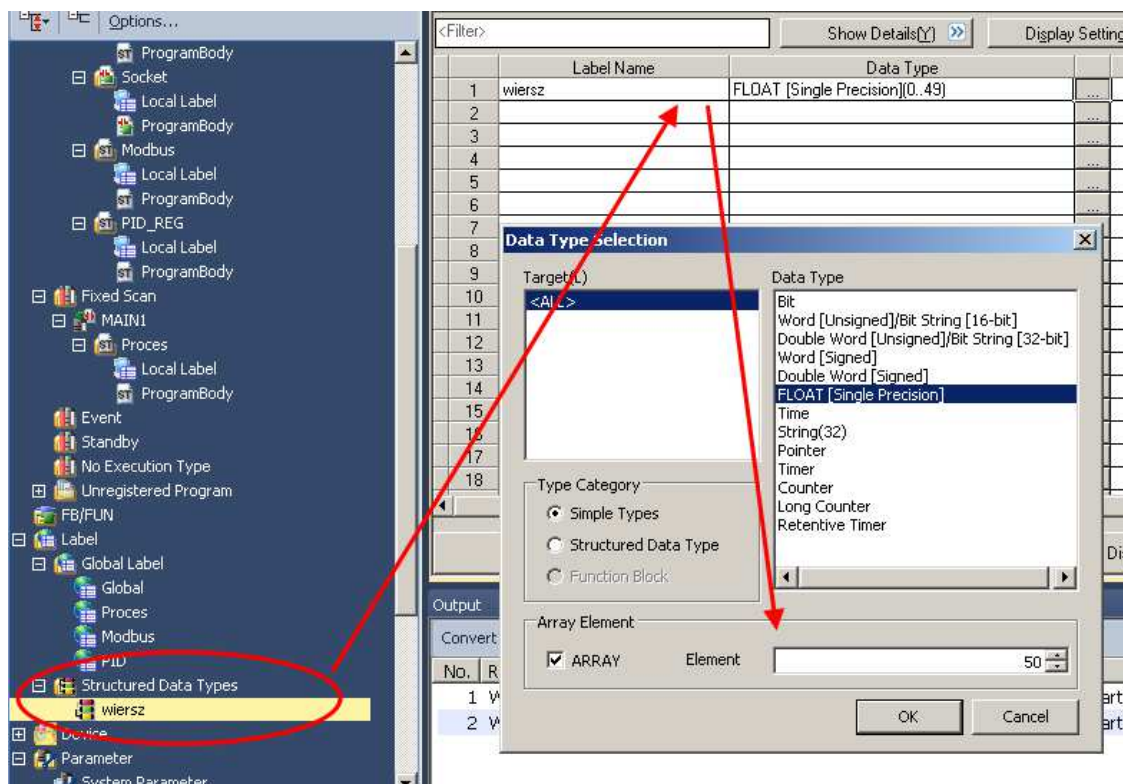
```

Użycie tablicy w obliczeniach:

```

FOR licznik := 0 TO 49 BY 1 DO
    wynik := liczba1 * licznik;
    wynik := tablica_MP[licznik].wiersz[licznik] * INT_TO_REAL(licznik);
END_FOR;

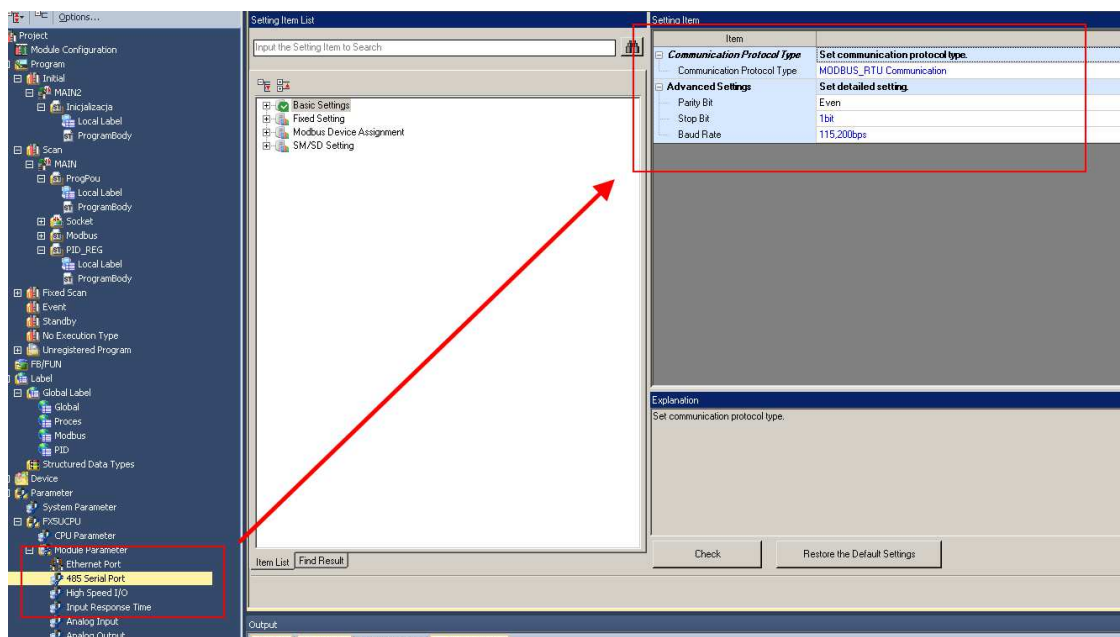
```



## 5 Opis komunikacji RS485 MODBUS, Socket Communication

### MODBUS:

Parametry komunikacji



## Deklaracja zmiennych

<Filter>		Easy Display		Display Setting		Check	
	Label Name	Data Type		Class		Assign (Dev	
1	Slave_adres	Word [Signed]	...	VAR_GLOBAL	▼		
2	Function_code	Word [Signed]	...	VAR_GLOBAL	▼		
3	Modbus_adres	Word [Signed]	...	VAR_GLOBAL	▼		
4	Device_count	Word [Signed]	...	VAR_GLOBAL	▼		
5	Pomiar_MODBUS	Bit	...	VAR_GLOBAL	▼		
6	Zapis_MODBUS	Bit	...	VAR_GLOBAL	▼		
7	Pozwolenie_pomiar_MODBUS	Bit	...	VAR_GLOBAL	▼		
8	Pozwolenie_zapis_MODBUS	Bit	...	VAR_GLOBAL	▼		
9					▼		

## Inicjalizacja

```
//Inicjalizacja MODBUS
```

```
Pomiar_MODBUS := 0;
```

```
Zapis_MODBUS := 0;
```

```
MOV(TRUE, K11, Slave_adres);
```

```
MOV(TRUE, K4, Function_code); //4-pomiar, 3-sterowanie
```

```
MOV(TRUE, K0, Modbus_adres); //zaczynamy liczyc od 0
```

```
MOV(TRUE, K7, Device_count); //7 pomiarow, 6 sterowan
```

```
//Ustawienie poczatkowe wyjsc procesu
```

```
ZRST(TRUE, D110, D120);
```

## Komunikacja

```
SET(Pozwolenie_pomiar_MODBUS AND LDP(TRUE, SM413), Pomiar_MODBUS);
```

```

IF (Pomiar_MODBUS) THEN
    Function_code := 4;
    Device_count := 7;
    ADPRW( Pomiar_MODBUS AND NOT Zapis_MODBUS, Slave_adres , Function_code ,
Modbus_adres, Device_count , D100, M100);
    IF (M101) THEN
        RST(TRUE, Pomiar_MODBUS);
        RST(TRUE, M101);
        RST(TRUE, M100);
    END_IF;
END_IF;

SET(Pozwolenie_zapis_MODBUS AND LDF(TRUE, SM413), Zapis_MODBUS);
IF (Zapis_MODBUS) THEN
    Function_code := 16;
    Device_count := 6;
    ADPRW( Zapis_MODBUS AND NOT Pomiar_MODBUS, Slave_adres , Function_code ,
Modbus_adres, Device_count , D110, M110);
    IF (M111) THEN
        RST(TRUE, Zapis_MODBUS);
        RST(TRUE, M111);
        RST(TRUE, M110);
    END_IF;
END_IF;

IF Zapis_MODBUS AND Pomiar_MODBUS THEN
    RST(TRUE, Zapis_MODBUS);
    RST(TRUE, Pomiar_MODBUS);
END_IF;

```

### Socket Communication:

Parametry komunikacji -> patrz rozdział 3.5

Deklaracja zmiennych

Lokalne

		Label Name	Data Type	
1		wyslalem	Bit	...
2		nie_wyslalem	Bit	...
3		zmienna_int	Word [Signed]	...
4		auto_send	Bit	...
5				...

Globalne

5/10/2017

Strona 24 z 35

## PUST – Projektowanie Układów Sterowania

<Filter> Easy Display << Display Setting Check				
	Label Name	Data Type	Class	Assign (Device)
1	send_string	String(32)	VAR_GLOBAL	D301
2	temp_string	String(32)	VAR_GLOBAL	D2020
3	tekst_temp	String(32)	VAR_GLOBAL	
4	dlugosc_tekstu	Word (Signed)	VAR_GLOBAL	
5				

Przygotowanie ramki

```
//Generacja tekstu do wysłania przez socket communication
```

```
tekst_temp := 'U=';
```

```
//tekst_temp := CONCAT(tekst_temp, REAL_TO_STRING(u_k));
```

```
tekst_temp := CONCAT(tekst_temp, INT_TO_STRING(D114));
```

```
tekst_temp := CONCAT(tekst_temp, ';Y=');
```

```
tekst_temp := CONCAT(tekst_temp, INT_TO_STRING(D100));
```

```
//tekst_temp := CONCAT(tekst_temp, REAL_TO_STRING(y_k));
```

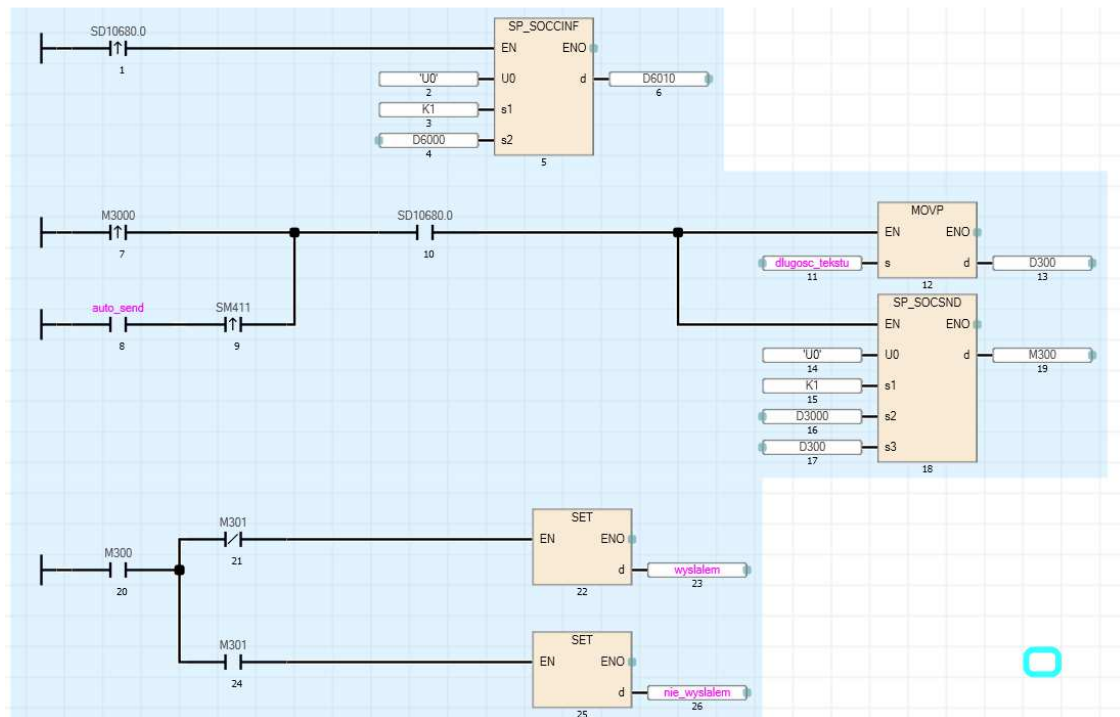
```
tekst_temp := CONCAT(tekst_temp, '$L');
```

```
send_string := tekst_temp;
```

```
//Dlugosc tekstu
```

```
dlugosc_tekstu := LEN(send_string);
```

Komunikacja



Skrypt MATLAB 2017

```
delete(instrfindall)
```

5/10/2017

Strona 25 z 35

```

pause(2);

close all;
clear all;

t = tcpip('192.168.127.250',4000, 'NetworkRole', 'client');

t.OutputBufferSize = 3000;
t.InputBufferSize = 3000;

fopen(t);
fprintf('Fopen zadzialal');
iterator = 1;
data = zeros(2,2);
figure(1);
while(1)
    if (t.BytesAvailable ~= 0)
        temp = fscanf(t);
        temp
        eval(temp);
        data(1,iterator) = U;
        data(2,iterator) = Y;
        fprintf('Fscanf zadzialal');
        iterator=iterator + 1;
        plot(1:length(data(2,:)), data(2,:));
        hold on;
        grid on;
        plot(1:length(data(1,:)), data(1,:));
        hold off;
    end
    pause(0.05);
end

fclose(t);
delete(t);
clear t;

```

## 6 Tworzenie grafik operatorskich w środowisku GT Designer 3

### 6.1 Projekt demo

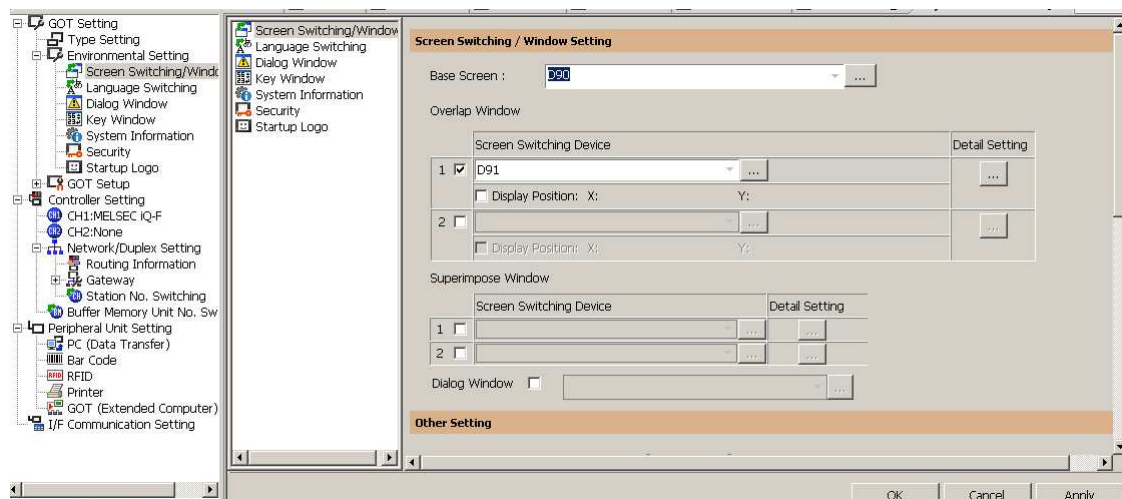
W ramach tego ćwiczenia studenci otrzymują przykładowy projekt na panel operatorski GOT Simple. Na bazie tego projektu należy przygotować aplikację zgodnie z opisem podanym w instrukcji do ćwiczenia. Projekt zawiera przykładowe ekrany z podstawowymi operacjami dostępnymi na panelach operatora. Należy zapoznać się z funkcjami i wykorzystać je w dalszej pracy do realizacji finalnej aplikacji. W następnych rozdziałach opisane zostaną poszczególne panele. Finalny projekt powinien opierać się na zaproponowanej strukturze. Jednak ocena końcowa będzie silnie zależała od wprowadzonych modyfikacji w panelu i użyciu nowatorskich pomysłów.

### 6.2 Panel MENU – 1



Po uruchomieniu panelu operatora lub po wgraniu projektu pierwszy ekran, który się zgłosi przedstawiono na poniższym rysunku. Po prawej stronie znajdują się przyciski do przechodzenia między innymi ekranami. W górnej części mamy pola godziny i daty pobieranej ze sterownika PLC. Po lewej stronie znajduje się lampka podłączona do bitu PLC SM412 (1Hz). Zgodnie z komentarzem na zielono wymagany jest adres sterownika PLC 192.168.127.250. W środkowej części można zobaczyć pole wyświetlające stały ciąg znaków w zależności od rejestru panela operatora GD1000 – rejestr ten jest ustawiany w polu wejściowym numerycznym znajdującym się na prawo od wyświetlanego tekstu. Na samym dole znajduje się przycisk, który pozwala na wyświetlenie okienka typu „popup”. Wyświetlenie tego okienka odbywa się przez wpisanie odpowiedniej wartości – numeru ekranu – do rejestru sterownika PLC. Połączenie między rejestrem sterownika a opisywaną funkcją znajduje się na kolejnym rysunku. Zgodnie z rysunkiem rejestr ekranów głównych to D90, rejestr ekranów typu „popup” to D91. W rejestrze D90 zawsze znajduje się numer aktualnie wyświetlanego ekranu głównego. Możliwa jest też jego zmiana z poziomu PLC. Natomiast wpisana wartość do rejestru D91 powoduje wyświetlenie odpowiedniego numeru ekranu „popup”. Jeżeli w rejestrze D91 znajduje się wartość 0 to żaden ekran „popup” nie jest wyświetlany.





### 6.3 Panel WEWY – 2

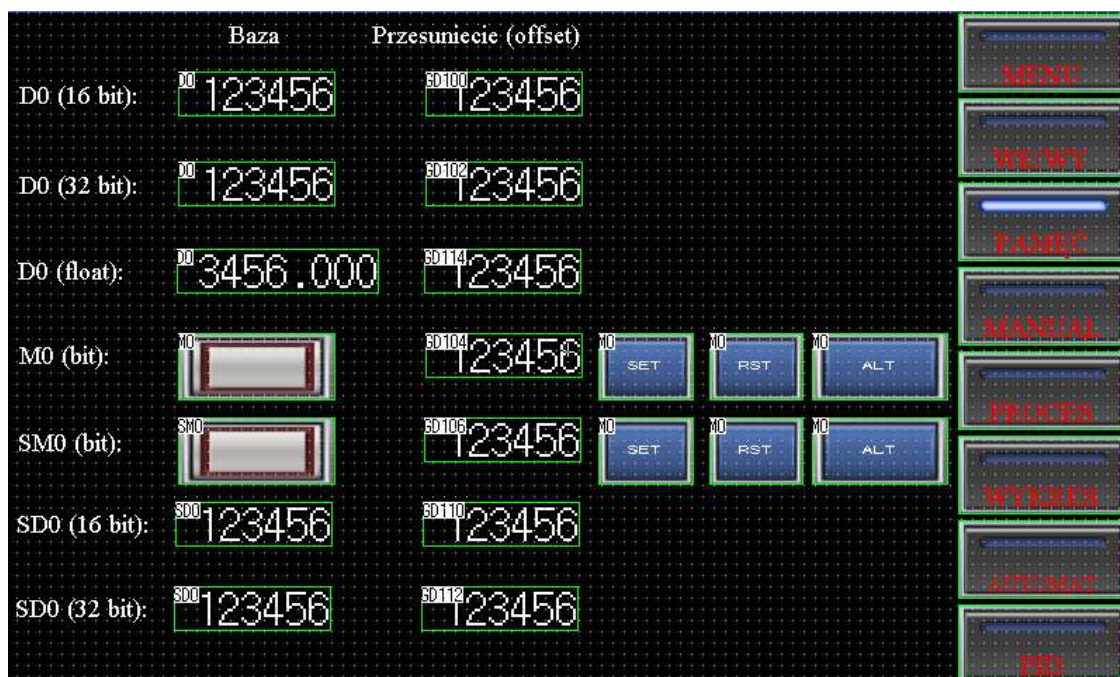
Na panelu drugim można obserwować aktualny stan wejść i wyjść sterownika. W przypadku wejść są to tylko lampki sygnalizujące stan. W przypadku wyjść są to przyciski, które można wciskać i zmieniać tym samym stan wyjścia w trybie „alternate” – zawsze zmiana na stan przeciwny.



### 6.4 Panel PAMIEC – 3

W panelu trzecim można obserwować i modyfikować pamięć sterownika PLC. W kolumnie baza widoczna jest wartość rejestru. Obserwowany rejestr jest zawsze

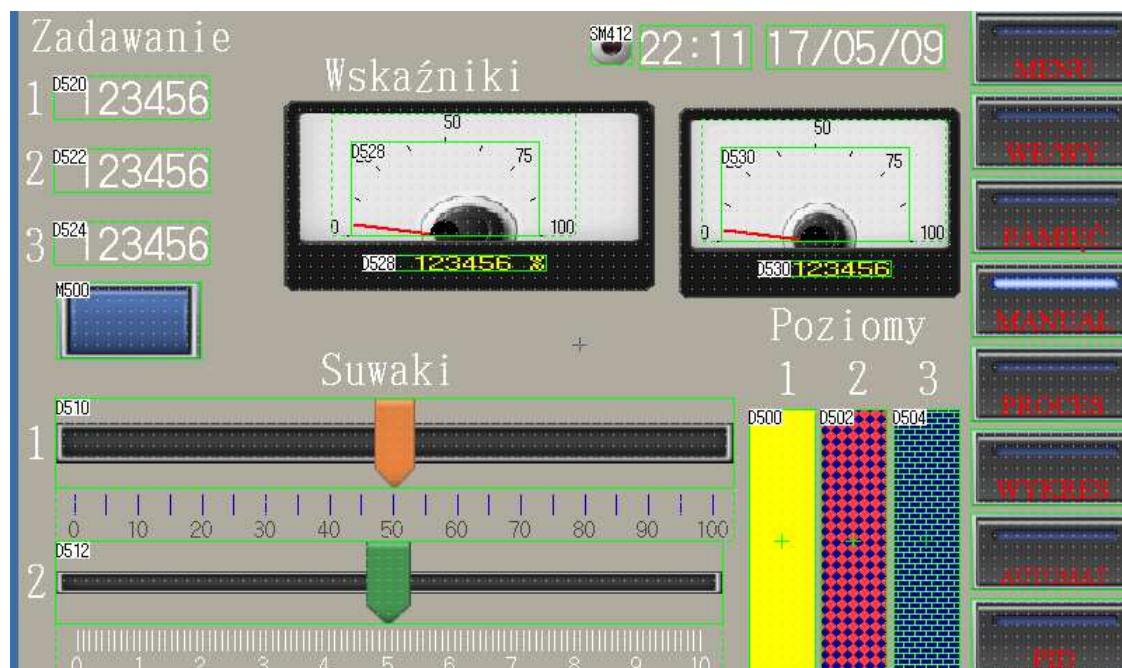
określony jako np. D(0 + offset). Offset jest rodzajem przesunięcia numeru rejestru – wskaźnik w tablicy – w rozważanym przypadku do offsetu użyto rejestrów panela operatora zaczynając od GD100. W przypadku bitów pamięci jak np. M0 możliwe operacje do wykonania to SET, RESET i ALTERNATE.



## 6.5 Panel MANUAL – 4

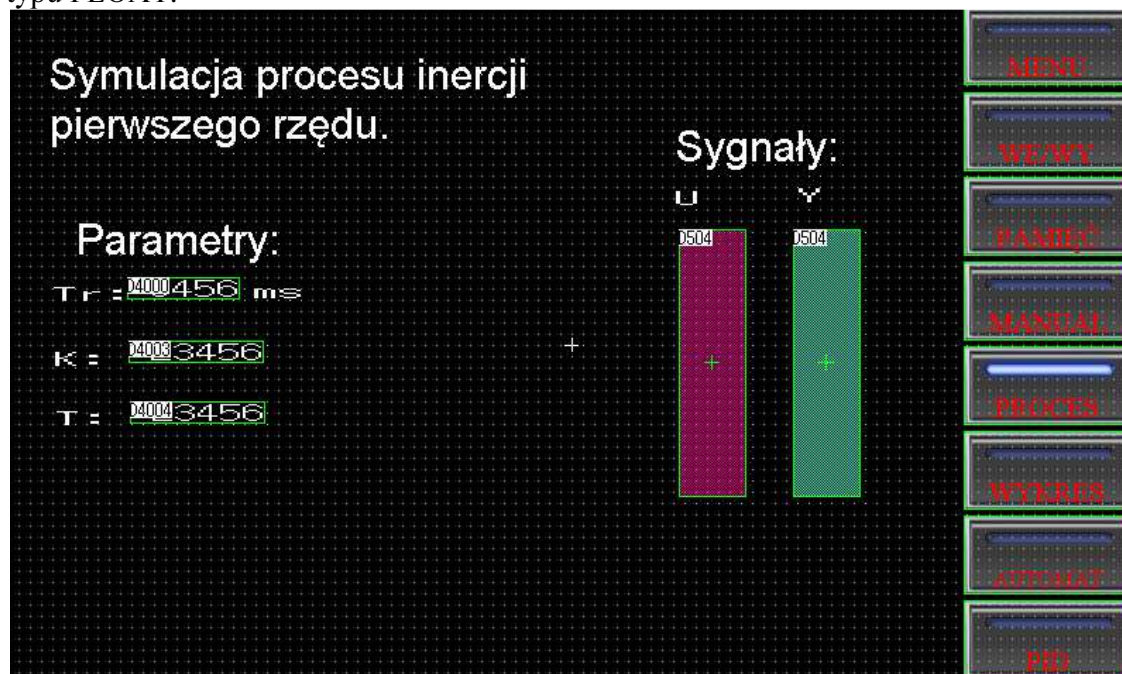
Na czwartym panelu znajdują się pola wpisu wartości, wskaźniki wychyłowe, poziomy oraz suwaki do zadawania wartości. Wszystkie elementy pobierają lub zapisują dane bezpośrednio w rejestrach typu D w PLC. Dla przypomnienia rejestry te są domyślnie 16 bitowe. Należy zwrócić szczególną uwagę przy ustawieniu parametrów danego elementu na panelu odnośnie liczby bitów zmiennej. Jeżeli oczywiście w PLC używamy zmiennej 32 bitowej należy tak samo to ustawić w parametrach elementu na panelu.





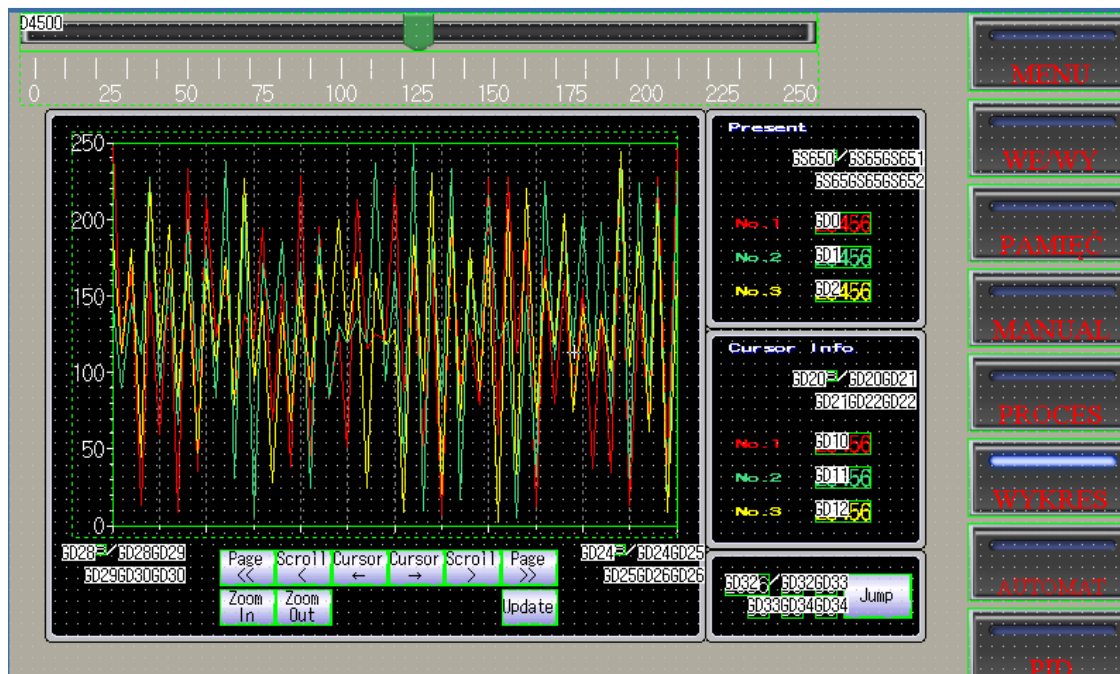
## 6.6 Panel PROCES – 5

Na panelu piątym przedstawiono parametry symulowanego procesu inercji pierwszego rzędu oraz aktualne wartości sygnału wejściowego U i wyjściowego Y. Symulacja procesu może zostać zrealizowana w sterowniku PLC przy użyciu języka ST i zmiennych typu FLOAT.



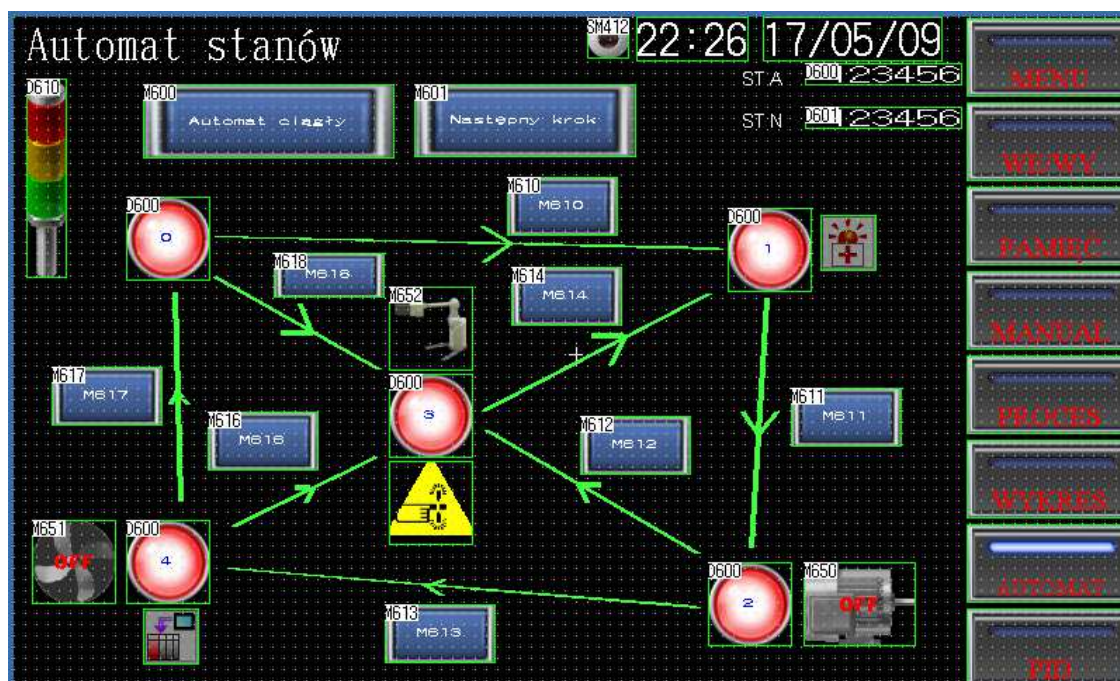
## 6.7 Panel WYKRES – 6

Na szóstym panelu przedstawiono możliwości rysowania wykresów. Rozwiązanie przedstawia rysowanie trzech pisaków o różnych kolorach. Możliwe jest obserwowanie aktualnej wartości, przejścia do historii, wstrzymanie i wznowienie rysowania. Na samej górze zamieszczono suwak do zmiany na przykład wartości zadanej.



## 6.8 Panel AUTOMAT – 7

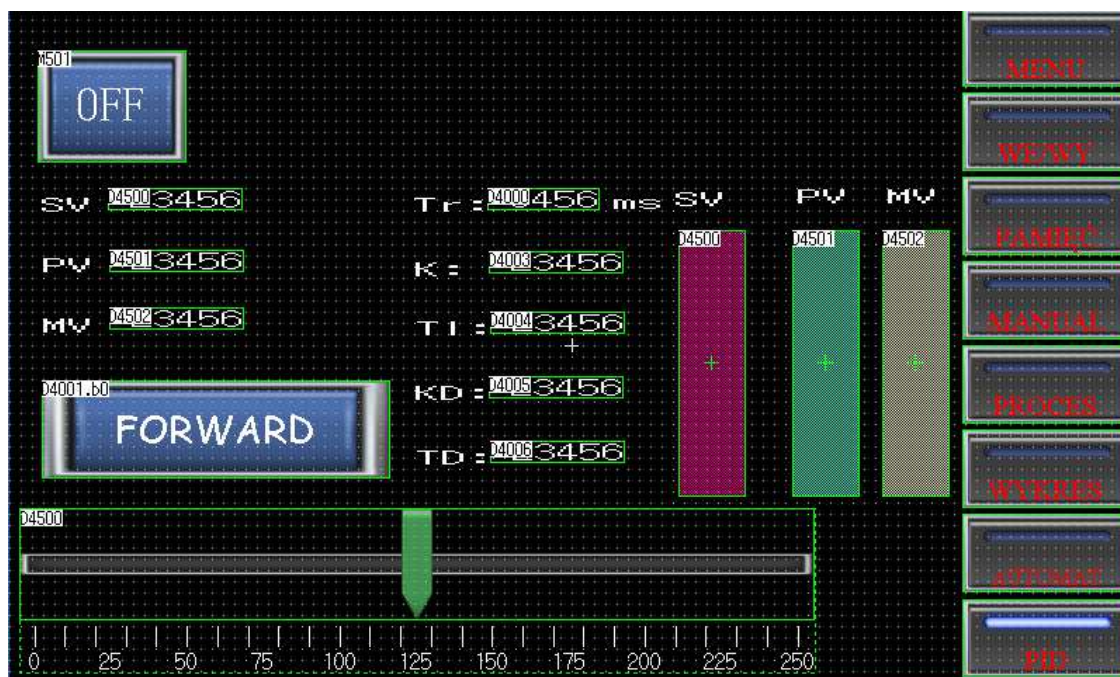
Na panelu siódmym przedstawiono graficzną reprezentację automatu stanów zrealizowane w języku ST w sterowniku PLC. Automat ten może zostać użyty do opracowania głównego automatu stanów, który będzie między innymi mógł wybierać odpowiedni algorytm pracy, przechodzić między pracą auto i ręczną.



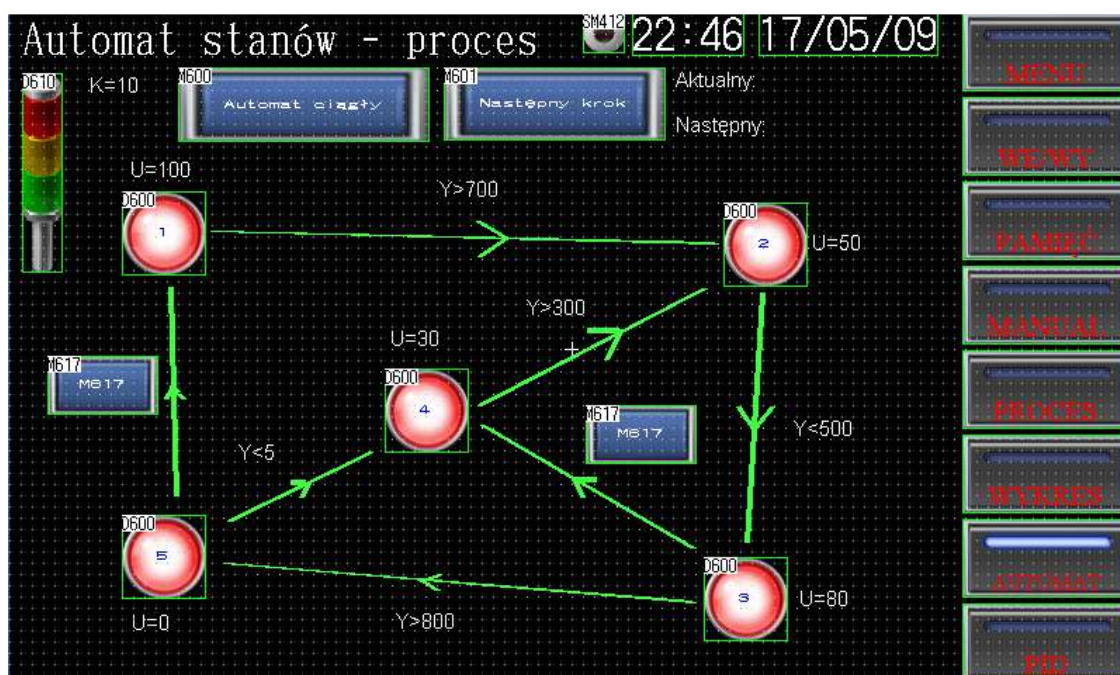
## 6.9 Panel AUTOMAT P – 8

Na panelu ósmym przedstawiono parametry oraz odpowiednie wartości wejściowe/wyjściowe dla regulatora PID. Znajdują się tutaj wartości zadana SV, mierzona procesa PV, sterowanie MV. Wartości wyświetlają się w postaci numerycznej oraz w postaci wykresów słupkowych. Wartość zadana SV można zmieniać przy pomocy suwaka. W środkowej części ekranu znajdują się podstawowe parametry regulatora. Aby załączyć regulator należy wcisnąć przycisk OFF na samej górze, jest on połączony z bitem w PLC M501. Ten z kolei powinien być odpowiednia zaprogramowany, aby uruchomić blok PID w sterowniku PLC.





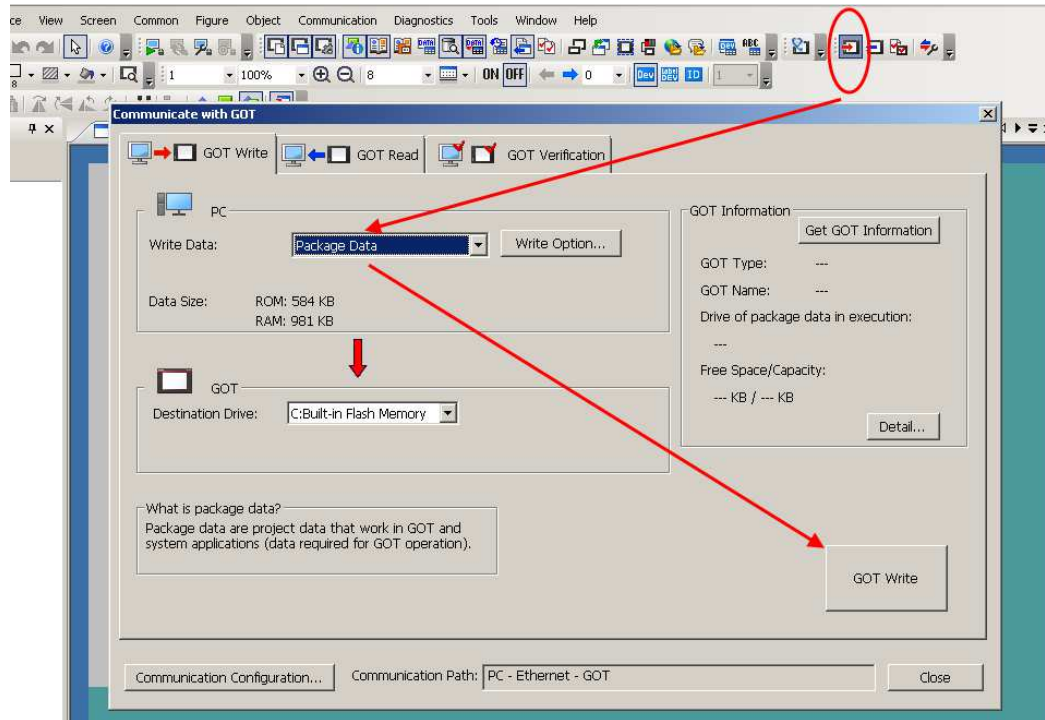
## 6.10 Panel AUTOMAT P – 10



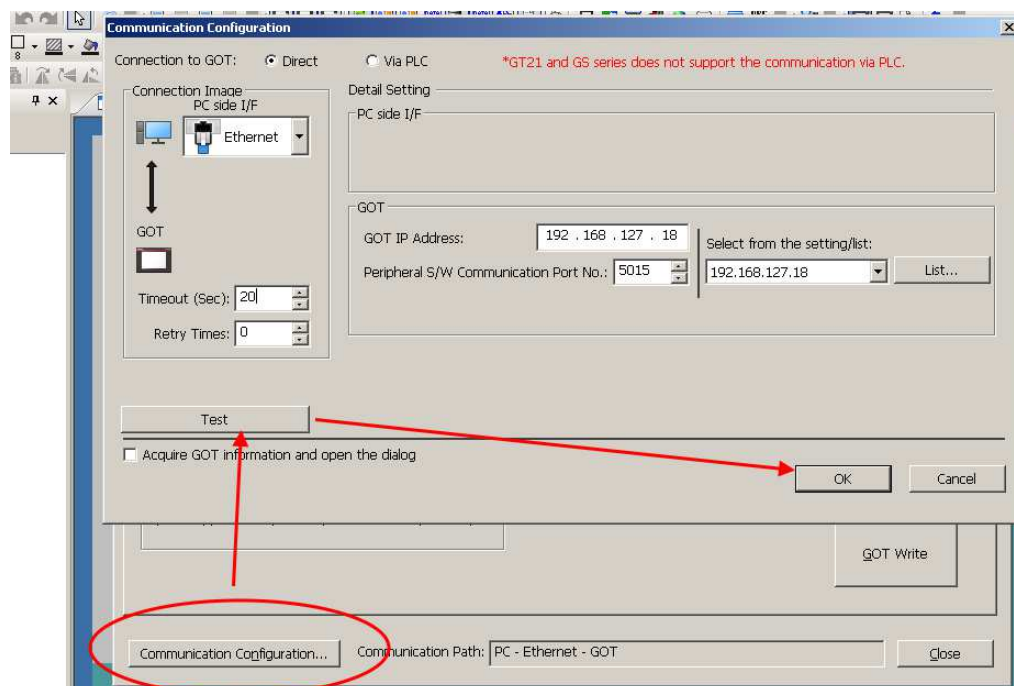
Na panelu dziesiątym przedstawiono graficzną reprezentację automatu stanów zrealizowane w języku ST w sterowniku PLC. Automat ten powinien zostać użyty do zmiany wartości zadanych w algorytmach regulacji w celu całkowitej automatyzacji pomiarów. Automat ten powinien rozpoczynać działanie tylko wtedy, gdy wybrany jest tryb pracy automatyczny.

## 6.11 Wgrywanie projektu do panela operatora

Przy edycji projektu na panel operatora nie jest wymagana żadna kompilacja. Po wprowadzeniu zmian w projekcie można od razu wgrać zmiany na panel. Operację wgrywania wykonuje się jak przedstawiono na rysunku poniżej.



Gdy procedura wgrywania jest niemożliwa z uwagi na brak komunikacji należy sprawdzić to zgodnie ze schematem z poniższego rysunku.



Gdy komunikacja jest niemożliwa i test nie wykonuje się poprawnie należy spróbować wykonać ping w kierunku adresu panela z poziomu konsoli cmd w Windows.

## **7 Dokumentacja**

Dokumentacja do pobrania z internetu

<http://app.mitsubishielectric.com/app/fa/download/search.do?kisyu=/plcf&mode=manual>

GT Designer 3: Help >> GT Designer 3 Help >> E-Manual Viewer

GX Works 3: Help >> GX Works 3 Help >> E-Manual Viewer