**2.1.4 Wyjścia PWM**

dodaj linijkę kodu do opisu:

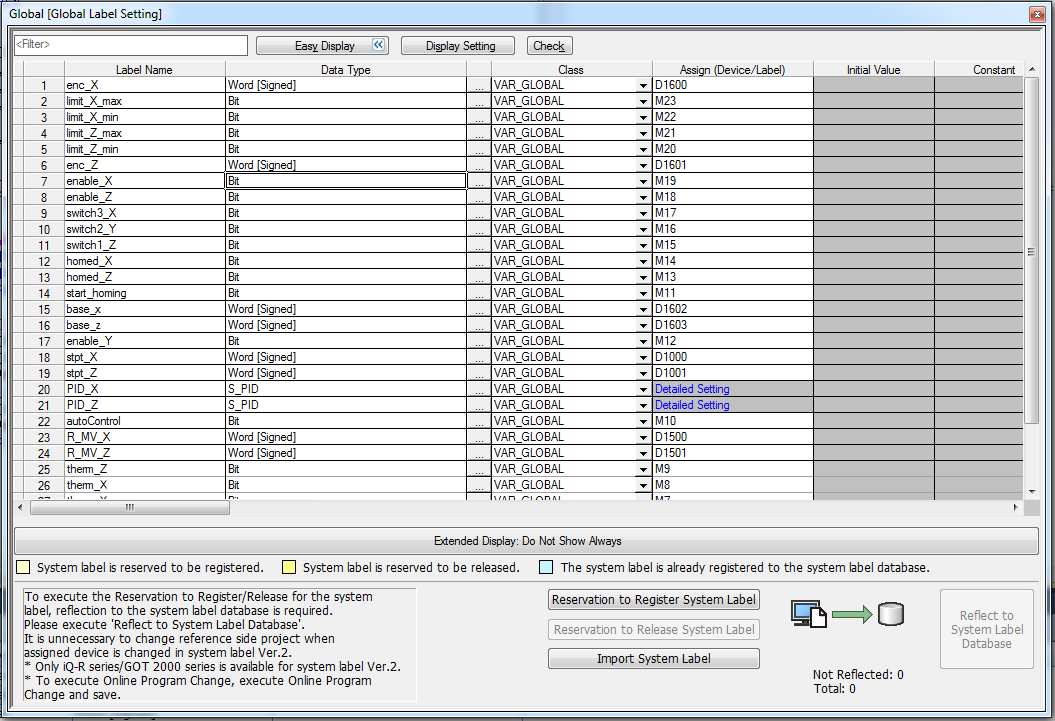
HIOEN(TRUE, K0, K17, K0);

“aktywacja wyjść PWM na kanałach CH1 i CH5”

ustawione bity pierwszy i piąty dają dziesiętnie 17 obecne jako argument HIOEN

**2.2. Mechanizm labeli**

Zarejestrowaliśmy szereg labeli globalnych zastępujących m.in. wejścia X, ponieważ własne nazwy zwiększają uniwersalność i znacznie zwiększają czytelność kodu.



global\_label.png

rys. Ustawienia labeli globalnych

**2.3. Skalowanie i bazowanie**

2.3.1 Skalowanie osi X

Odczytując wartości enkodera osi X w dwóch skrajnych pozycjach i dzieląc ich różnicę(9421) przez maksymalną założoną wartość pozycji zadanej(100) otrzymaliśmy zaokrąglony współczynnik skalowania(94)

Efektem jest operacja:

MOV(TRUE, stpt\_X\*K94, PID\_X.SV);

2.3.2 Skalowanie osi Z

Odczytując wartości enkodera osi Z w dwóch skrajnych pozycjach i dzieląc ich różnicę(2300) przez maksymalną założoną wartość pozycji zadanej(100) otrzymaliśmy zaokrąglony współczynnik skalowania(23)

Efektem jest operacja:

MOV(TRUE, stpt\_Z\*K23, PID\_Z.SV);

2.3.3 Bazowanie osi

Bazowanie wszystkich osi inicjowane jest stanem wysokim zmiennej *start\_homing*. Po spełnieniu tego warunku rozpoczynany jest ruch osi w kierunku krańcówek. Osie poruszają się do chwili gdy osiągnął pozycje swoich krańcówek bazujących. W następnym kroku osie są zatrzymywane, ustawiane są bity informujące o zakończeniu bazowania konkretnej osi i wykonywane jest zerowanie liczników enkoderów osi.

Zbazowanie każdej z osi kończy proces bazowania dzwigu. Możliwe staje się sterowanie za pomocą pozycji zadanych poprzez regulatory PID.

IF start\_homing = TRUE THEN

IF homed\_X = FALSE THEN

IF switch3\_X = FALSE THEN

Y7:=TRUE; // DEAKTYWACJA HAMULCA

PWM(TRUE, K30, K100, Y0); // PWM OSI X - KARETKA

ELSE

PWM(FALSE, K1, K100, Y0);

Y7:=FALSE; // AKTYWACJA HAMULCA

DHCMOVP(TRUE, 0, 0, SD4500); // WYZEROWANIE ENKODERA OSI X

homed\_X:=TRUE;

END\_IF;

END\_IF;

IF homed\_Z = FALSE THEN

IF switch1\_Z = FALSE THEN

Y5:=TRUE; // DEAKTYWACJA HAMULCA

PWM(TRUE, K30, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

ELSE

PWM(FALSE, K1, K100, Y2);

DHCMOVP(TRUE, 0, 0, SD4620); // WYZEROWANIE ENKODERA OSI Z

Y5:=FALSE; // AKTYWACJA HAMULCA

homed\_Z:=TRUE;

END\_IF;

END\_IF;

IF homed\_X AND homed\_Z THEN

start\_homing := FALSE;

END\_IF;

END\_IF;

**2.4. Obsługa I/O cyfrowych**

2.4.1 Odczyt wejść cyfrowych

Stany wejść cyfrowych w celu zwiększenia czytelności kodu są przepisywane do uprzednio zdefiniowanych labeli, które następnie są wykorzystywane w programie.

Przykładowy odczyt stanu krańcówek osi:

MOVB(TRUE, X13, switch3\_X);

MOVB(TRUE, X14, switch2\_Y);

MOVB(TRUE, X12, switch1\_Z);

Dalsze przykładowe wykorzystanie labeli do sterowania:

IF switch3\_X = FALSE THEN

Y7:=TRUE; // DEAKTYWACJA HAMULCA

PWM(TRUE, K30, K100, Y0); // PWM OSI X - KARETKA

ELSE

PWM(FALSE, K1, K100, Y0);

Y7:=FALSE; // AKTYWACJA HAMULCA

DHCMOVP(TRUE, 0, 0, SD4500); // WYZEROWANIE ENKODERA OSI X

homed\_X:=TRUE;

END\_IF;

2.4.2 Zapis wyjść cyfrowych

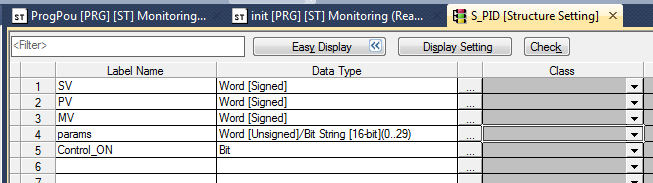
Zapis wyjść cyfrowych odbywa się poprzez bezpośrednie przypisanie stanu wyjścia. Nie zdecydowaliśmy się na zastosowanie dedykowanych labeli, ponieważ wyjść było stosunkowo niewiele.

Przykładowy zapis wyjścia cyfrowego:

Y5:=TRUE; // DEAKTYWACJA HAMULCA OSI Z

**2.5. PID**

Zastosowaliśmy wbudowaną strukturę regulatora PID. Do jej wykorzystania potrzebne było stworzenie struktury *S\_PID* zawierającej SV, PV, MV, parametry regulatora oraz bit aktywujący regulator.



s\_pid\_struct.png

struktura S\_PID

W programie inicjującym znajduje się inicjalizacja parametrów regulatorów PID

//Parametry regulatora wbudowanego PID osi X

PID\_X.params[0] := K100; //okres regulacji w milisekundach

PID\_X.params[3] := K3; //wzmocnienie regulatora P

PID\_X.params[4] := K5; //TI = 0 oznacza nieskonczony czas calkowania - inaczej

//mowiac calkowanie wylaczone

PID\_X.params[5] := K0; //KD = 0 oznacza zerowe wzmocnienie rozniczkowania

PID\_X.params[6] := K0; //TD = 0 oznacza wylaczone rozniczkowanie

PID\_X.params[22] := K100; //gorny limit wartosci wyjsciowej z regulatora - zapobiega

//rowniez efektowi wind-up

PID\_X.params[23] := K1; //dolny limit wartosci wyjsciowej z regulatora - -||-

SET(TRUE, PID\_X.params[1].5); //aktywacja limitow na wyjsciu regulatora

SET(TRUE, PID\_X.params[1].0); //trzeba odwrocic kierunek dzialania PID

//Parametry regulatora wbudowanego PID osi Z

PID\_Z.params[0] := K100; //okres regulacji w milisekundach

PID\_Z.params[3] := K1; //wzmocnienie regulatora P

PID\_Z.params[4] := K2; //TI = 0 oznacza nieskonczony czas calkowania - inaczej

//mowiac calkowanie wylaczone

PID\_Z.params[5] := K0; //KD = 0 oznacza zerowe wzmocnienie rozniczkowania

PID\_Z.params[6] := K0; //TD = 0 oznacza wylaczone rozniczkowanie

PID\_Z.params[22] := K100; //gorny limit wartosci wyjsciowej z regulatora - zapobiega

//rowniez efektowi wind-up

PID\_Z.params[23] := K0; //dolny limit wartosci wyjsciowej z regulatora - -||-

SET(TRUE, PID\_Z.params[1].5); //aktywacja limitow na wyjsciu regulatora

SET(TRUE, PID\_Z.params[1].0); //trzeba odwrocic kierunek dzialania PID

regulatory PID pracują w głównym programie. Kolejne wartości sterowania MV wyznaczane są poprzez wywołania funkcji PID(). Jeśli wyznaczone MV jest większe od 50 wykonywany jest ruch w przód z zadanym wypełnieniem PWM, a w przeciwnym wypadku wykonywany jest ruch w tył z zadanym wypełnieniem PWM.

Sekcja regulatora PID osi X:

MOVB(TRUE, TRUE, PID\_X.Control\_ON);

MOV(TRUE, enc\_X, PID\_X.PV);

MOV(TRUE, stpt\_X\*K94, PID\_X.SV);

PID(PID\_X.Control\_ON, PID\_X.SV , PID\_X.PV , PID\_X.params[0] , PID\_X.MV);

PWM(TRUE, PID\_X.MV, K100, Y0); // PWM OSI X - KARETKA

IF PID\_X.MV < K50 THEN

MOVB(TRUE, FALSE, Y10);

MOVB(TRUE, TRUE, Y7);

PWM(TRUE, K51-PID\_X.MV, K100, Y0); // PWM OSI X - KARETKA

ELSE

IF PID\_X.MV > K50 THEN

MOVB(TRUE, TRUE, Y10);

MOVB(TRUE, TRUE, Y7);

PWM(TRUE, PID\_X.MV-K49, K100, Y0); // PWM OSI X - KARETKA

ELSE

MOVB(TRUE, FALSE, Y7);

PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA

END\_IF;

END\_IF;

Sekcja regulatora PID osi Z:

MOVB(TRUE, TRUE, PID\_Z.Control\_ON);

MOV(TRUE, enc\_Z, PID\_Z.PV);

MOV(TRUE, stpt\_Z\*K23, PID\_Z.SV);

PID(PID\_Z.Control\_ON, PID\_Z.SV , PID\_Z.PV , PID\_Z.params[0] , PID\_Z.MV);

PWM(TRUE, PID\_Z.MV, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

IF PID\_Z.MV < K50 THEN

MOVB(TRUE, FALSE, Y6);

MOVB(TRUE, TRUE, Y5);

PWM(TRUE, K51-PID\_Z.MV, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

ELSE

IF PID\_Z.MV > K50 THEN

MOVB(TRUE, TRUE, Y6);

MOVB(TRUE, TRUE, Y5);

PWM(TRUE, PID\_Z.MV-K49, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

ELSE

MOVB(TRUE, FALSE, Y5);

PWM(FALSE, K1, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

END\_IF;

END\_IF;

**2.6. Tryb sterowania ręcznego**

Sterowanie ręczne możliwe jest po zresetowaniu flagi AUTO\_CONTROL(flaga domyślnie ustawiona). Jeśli flaga jest zresetowana program przechodzi do sekcji kodu w której odbywa się bezpośrednie wysterowanie wyjść PWM wartościami zmiennych R\_MV\_Z dla osi Z i R\_MV\_X dla osi X

sekcja kodu sterowania ręcznego osią Z:

IF enc\_Z > K2350 OR enc\_Z < -K50 THEN

// PRZEKROCZENIE ZAKRESÓW OSI Z

MOVB(TRUE, FALSE, Y5);

PWM(FALSE, K1, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

MOVB(TRUE, FALSE, homed\_Z);

ELSE

//STEROWANIE RĘCZNE OSI Z

PWM(TRUE, R\_MV\_Z, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

IF R\_MV\_Z < K50 THEN

MOVB(TRUE, FALSE, Y6);

MOVB(TRUE, TRUE, Y5);

PWM(TRUE, K51-R\_MV\_Z, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

ELSE

IF R\_MV\_Z > K50 THEN

MOVB(TRUE, TRUE, Y6);

MOVB(TRUE, TRUE, Y5);

PWM(TRUE, R\_MV\_Z-K49, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

ELSE

MOVB(TRUE, FALSE, Y5);

PWM(FALSE, K1, K100, Y2);

END\_IF;

END\_IF;

END\_IF;

sekcja kodu sterowania ręcznego osią X:

IF enc\_X > K9450 OR enc\_X < -K50 THEN

// PRZEKROCZENIE ZAKRESÓW OSI X

MOVB(TRUE, FALSE, Y7);

PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA

MOVB(TRUE, FALSE, homed\_X);

ELSE

// STEROWANIE RĘCZNE OSI X

PWM(TRUE, R\_MV\_X, K100, Y0); // PWM OSI X - KARETKA

IF R\_MV\_X < K50 THEN

MOVB(TRUE, FALSE, Y10);

MOVB(TRUE, TRUE, Y7);

PWM(TRUE, K51-R\_MV\_X, K100, Y0); // PWM OSI X - KARETKA

ELSE

IF R\_MV\_X > K50 THEN

MOVB(TRUE, TRUE, Y10);

MOVB(TRUE, TRUE, Y7);

PWM(TRUE, R\_MV\_X-K49, K100, Y0); // PWM OSI X - KARETKA

ELSE

MOVB(TRUE, FALSE, Y7);

PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA

END\_IF;

END\_IF;

END\_IF;

**2.7. Zabezpieczenia ruchów krańcowych**

W projekcie założyliśmy, że sytuacja w której któraś z osi wyjdzie poza założony dopuszczalny zakres ruchów jest ona natychmiast zatrzymywana i gaszona jest flaga zbazowania danej osi. Działaniem naprawczym w takiej sytuacji jest przejście na tryb sterowania automatycznego i ponowne zainicjowanie bazowania osi ustawiając flagę start\_homing

sekcja kodu zabezpieczająca ruchy krańcowe osi X:

IF homed\_X = TRUE THEN

IF enc\_X > K9450 OR enc\_X < -K50 THEN

MOVB(TRUE, FALSE, Y7);

PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA

MOVB(TRUE, FALSE, homed\_X);

ELSE

//automatyczne lub ręczne sterowanie osi X

…

END\_IF;

END\_IF;

sekcja kodu zabezpieczająca ruchy krańcowe osi Z:

IF homed\_Z = TRUE THEN

IF enc\_Z > K2350 OR enc\_Z < -K50 THEN

MOVB(TRUE, FALSE, Y5);

PWM(FALSE, K1, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU

MOVB(TRUE, FALSE, homed\_Z);

ELSE

//automatyczne lub ręczne sterowanie osi Z

…

END\_IF;

END\_IF;

**2.8. Język ST**

Listingi kodu zawarte powyżej jednoznacznie wskazują na pomyślne wykorzystanie języka ST.