PikoCNC Macro manual

wersja 6.0 04.04.2024

Wstęp

Makra pełnią rolę pomocniczą, umożliwiają tworzenie własnych prostych sekwencji ruchów, czynności, obsługę magazynków narzędzi itp.

Język programowania makr składniowo jest podobny z zastosowanym w skryptach pascalem. Różnice są następujące:

- Blok z komendami musi być poprzedzony znakiem „%”
- Nie można wprost korzystać ze zmiennych globalnych.

Pliki makr muszą znajdować się w folderze `Macros` naszego folderu z profilem - najczęściej będzie to folder o nazwie `ProfilDef`. Plik makra jest najpierw szukany w folderze `ProfilDef/Macros`, a jeśli go tam nie ma, to przeszukiwany jest folder `Elcosimo/Macros`.

> Własne makra zawsze należy zapisywać w `ProfilDef/Macros` gdyż folder `Elcosimo/Macros` jest nadpisywany przy każdej instalacji programu. Także w przypadku eksportu profilu, makra nie znajdujące się w `ProfilDef/Macros` nie zostaną zapisane w pliku profilu!.

W folderze `Elcosimo\Macros\Examples` można znaleźć szereg przykładów i otwierając je za pomocą wbudowanego edytora makr można prześledzić kod.

Podstawowe typy zmiennych na jakich operują funkcje opisano w temacie funkcji **`SetVar`**.

Lista funkcji

Funkcje związane z obsługą argumentów

Wykonując makra z okienka MDI możemy podawać argumenty np. G0 X1.0 Y12 gdzie G0 to nazwa makra a X1.0 i Y12 to argumenty. Argumenty muszą być rozdzielone znakiem spacji. Nazwa argumentu to pierwszy znak, pozostałe muszą być liczbą i tak: „X” i „Y” to nazwy argumentów a występujące za nimi liczby to ich wartość. Wyjątkową nazwą argumentu jest znak „?” za którym nie musi być wartości liczbowej. Może być wykorzystany do trybu „help”. Argumentami można także przekazywać wartości binarne (klucze) wtedy przed argumentem dodajemy znak „-” np. „M6 T2 -M” Klucz -M wymusza w tym wypadku pomiar narzędzia. Komendy, jako argument mogą mieć także liczbę po kropce, którą możemy odczytać z poziomu makra np. „M6.1”.

Nazwa	Opis
IsArg	function IsArg(name:string):boolean Sprawdza czy argument o danej nazwie został podany.
GetArg	function GetArg(name:string):variant Zwraca wartość argumentu o danej nazwie. Jeśli argument o takiej nazwie nie został podany to zostanie zgłoszony błąd i makro zostanie zatrzymane.
GetArgDef	function GetArgDef(name:string; deflaut:variant):variant Zwraca wartość argumentu o danej nazwie. Jeśli argument o takiej nazwie nie został podany to zwrócona zostanie wartość deflaut.
DotVal	function DotVal():cardinal Zwraca liczbę po kropce w podanej w komendzie, dla np. ”G43.1” funkcja zwróci wartość 1. Należy tutaj podkreślić, że makro nazywa się „G43”, a „.1” dodajemy w MDI. Jeżeli w komendzie nie ma kropki zwrócone zostanie zero. Wartość musi być dodatnia.
UserExe	function UserExe():boolean' Zwraca TRUE jeśli makro uruchomione jest ręcznie z okienka MDI lub przycisku z zakładce „Macro”. Jeżeli makro uruchomił program otrzymamy FALSE.

Funkcje zgłaszania stanów wyjątkowych

Nazwa	Opis
UserError	procedure UserError(info:string) Otwiera okienko z komunikatem „info”. Makro zostaje zatrzymane.

Funkcje obsługi zmiennych

Nazwa	Opis
SetVar	procedure SetVar(num:cardinal; v:variant) Ustawia zmienną o danym numerze wartością „v”. Funkcje SetVar/GetVar pracują na zmiennych typu „variant” czyli akceptują takie typy zmiennych jak: Extended – liczba zmiennoprzecinkowa np. 56.123 / -0.45 / 3.14 Cardinal – liczba całkowita bez znaku np. 10 / 123 / 19203 Integer – liczba całkowita ze znakiem np. -200 / 150 / -4534 String – ciąg znaków (tekst) np. 'ala ma kota' Bool – TRUE / FALSE. Zapis zmiennej za pomocą SetVar równocześnie nadaje typ tej zmiennej. Dlatego przed odczytem zmiennej wcześniej trzeba ją zapisać aby miała ustalony typ. Nie należy też mieszać typów przy odczycie i np. próbować uzyskać zmienną typu string z indeksu gdzie zapisaliśmy cardinal. Tablica zmiennych nie jest kasowana przed wykonaniem makra dlatego może służyć do wymiany informacji między kolejnymi makrami. Przy standardowej długości tablicy zmiennych parametr „num” musi być w zakresie 0-249. Od wersji programu 5.8.0 Zmienne zapisane w indeksach 200-249 zapisywane są na dysk i pamiętane także po wyłączeniu zasilania.
GetVar	function GetVar(num:cardinal):variant Zwraca wartość zmiennej o danym numerze.
SetVarSize	procedure SetVarSize(num:cardinal) Ustawia rozmiar tablicy zmiennych, gdzie „num” – nowy rozmiar tablicy zmiennych. Domyślnie jest to 250. Jeśli potrzebna jest większa ilość można tą funkcją to zmienić. Wywołanie powyższych funkcji z indeksem przekraczającym rozmiar tablicy spowoduje błąd i zatrzymanie makra.

Funkcje związane ze skokami

Nazwa	Opis

GotoLabel	procedure GotoLabel(name:string)
	
	Wykonuje skok do etykiety o podanej nazwie. Etykieta to nazwa występująca jako jedyna w linii poprzedzona znakiem „:” np.:
	
	`GotoLabel('main_loop');`
	`...`
	`...`
	`:main_loop  // etykieta o nazwie main_loop`
	`...`

ProcedureLabel	procedure ProcedureLabel(name:string) Wykonuje skok do etykiety o podanej nazwie, ale w odróżnieniu od GotoLabel po napotkaniu instrukcji Return wróci do miejsca z którego skok został wykonany (tzn. do kolejnej po nim instrukcji).
Return	procedure Return() Jeśli wystąpi w głównym wątku programu spowoduje jego zakończenie. Jeśli w wątku wywołanym przez ProcedureLabel to nastąpi powrót do miejsca wywołania.

Funkcje związane z odczytem aktualnej pozycji maszyny

Nazwa	Opis
PosX	function PosX():Extended Zwraca aktualną pozycję maszynową osi X.
PosY	function PosY():Extended j.w dla osi Y.
PosZ	function PosZ():Extended j.w dla osi Z.
PosA	function PosA():Extended j.w dla osi A.
PosB	function PosB():Extended j.w dla osi B.

Funkcje związane z wykonywaniem ruchu oraz zmianą pozycji osi

Nazwa	Opis
SetX	procedure SetX(pos:extended) Ustawia rejestr pozycji dla osi X. Jest to zawsze pozycja maszynowa !
SetY	procedure SetY(pos:extended) j.w. dla osi Y.
SetZ	procedure SetZ(pos:extended) j.w. dla osi Z.
SetA	procedure SetA(pos:extended) j.w. dla osi A.
SetB	procedure SetB(pos:extended) j.w. dla osi B.

ExeMove	procedure ExeMove(speed:integer)
	
	Wykonanie ruch na podstawie zawartości rejestru pozycji. Argument „speed” może przyjmować następujące wartości:
	0 - Wykonany ruch będzie G0.
	>0 - Wykonany ruch będzie G1 z podaną prędkością.
	-1 - Ruch nie będzie wykonany natomiast wyczyszczony zostanie rejestr pozycji.
	
	Po wywołaniu ExeMove kolejna linia programu będzie wykonana gdy zadany ruch się zakończy. Dlatego nie wolno w jednej linii programu umieszczać więcej niż jedną taką komendę. Funkcje ExeMove, WriteMove, ExeSetPos po wywołaniu czyszczą automatycznie rejestr pozycji osi.
	
	Przykład:
	
	`SetZ(20.0); ExeMove(0);//Przejazd G0 na pozycję Z20.0`
	
	`SetX(10.0); SetY(20.0);`
	`ExeMove(1000); // Przejazd G1 F1000 na pozycję X10 Y20`

ExeMoveExt	procedure ExeMoveExt(speed:integer; rel:boolean; translation_mode:cardinal) Rozbudowana wersja funkcji **`ExeMove`** speed - j.w. rel – ruch będzie przesunięciem względem aktualnej pozycji jeśli TRUE. translation_mode – określa jaka będzie translacja ruchu: TR_NONE – surowa pozycja maszynowa TR_MACHINE – pozycja maszynowa uwzględniająca mapę korekcji geometrii. TR_WORK – pozycja materiałowa

<table>
<tr><th>Nazwa</th><th>Opis</th></tr>
<tr><td>WriteMove</td><td>procedure WriteMove(speed:integer)

Podobnie jak ExeMove, ale nie czeka na zakończenie zadanego ruchu, tylko od razu przechodzi do kolejnej linii programu – dlatego trzeba ostrożnie ją stosować, ze świadomością konsekwencji jakie to może nieść.</td></tr>
<tr><td>ExeSetPos</td><td>procedure ExeSetPos()

Ustawia pozycję osi na podstawie zawartości rejestru pozycji.

Przykład:

<pre>SetZ(0.0); ExeSetPos(); // Zerowanie pozycji osi Z</pre></td></tr>
<tr><td>ExeSetCorrection</td><td>procedure ExeSetCorrection()

Ustawia korekcję osi na podstawie zawartości rejestru pozycji.</td></tr>
<tr><td>ExeSetCorrectionExt</td><td>procedure ExeSetCorrectionExt(mode:cardinal; tool_H:integer)

Na podstawie numeru narzędzia „tool_H” ustawia aktualną korekcję w trybie „mode”.

Jeżeli „tool_H” jest większe lub równe zero, to oznacza numer narzędzia w magazynku i przeładowane będą korekcje wszystkich osi korekcjami tego narzędzia. Jeżeli mniejsze od zera, to oznacza rejestr pozycji (SetX itd.) i przeładowane będą tylko wybrane osie.

Tryb pracy zależne od „mode”

<table><tr><td>0</td><td>Aktualna wartość korekcji zostanie zastąpiona</td></tr><tr><td>1</td><td>Do aktualnej wartości zostanie dodana</td></tr><tr><td>2</td><td>Od aktualnej wartości zostanie odjęta</td></tr></table>
Przykład:

<pre>// Przeładownie korekcji dla aktualnego narzędzia
ExeSetCorrectionExt(0, GetActTool());

// Zerowanie korekcji
ExeSetCorrectionExt(0, 0);</pre></td></tr>
<tr><td>ExeCancelCorrection</td><td>procedure ExeCancelCorrection()

Zeruje korekcję wszystkich osi.</td></tr>
</table>

Funkcje związane z jazdą referencyjną

Nazwa	Opis
RefX	procedure RefX(dir:integer) Ustala kierunek jazdy referencyjnej dla osi X. Parametr „dir” określa kierunek: (dir = -1) Kierunek w lewo. (dir = 1) Kierunek w prawo.
RefY	procedure RefY(dir:integer) j.w. dla osi Y.
RefZ	procedure RefZ(dir:integer) j.w. dla osi Z.
RefA	procedure RefA(dir:integer) j.w. dla osi A.
RefB	procedure RefB(dir:integer) j.w. dla osi B.
ExeRef	procedure ExeRef(mode:cardinal; speed:integer) Wykonanie jazdy referencyjnej na podstawie zawartości rejestru pozycji. Argument „mode” może przyjmować następujące wartości: HOME_ON_SOFT Jazda do momentu aktywacji wejścia HOME osi z łagodnym hamowaniem. HOME_ON - Jazda do momentu aktywacji wejścia HOME osi. HOME_OFF - Jazda do momentu dezaktywacji wejścia HOME osi. INDEX_ON - Jazda do momentu aktywacji wejścia INDEX osi. PROBE_ON - Jazda do momentu aktywacji wejścia PROBE. PROBE_OFF - Jazda do momentu dezaktywacji wejścia PROBE. PROBE_ON_SOFT - Jazda do momentu aktywacji wejścia PROBE z łagodnym hamowaniem. Parametr „speed” określa prędkość jazdy. Jeśli speed=0 to prędkość będzie jak do G0. Kolejna linia programu będzie wykonana gdy jazda referencyjna się zakończy.
SoftLimit	procedure SoftLimit(st:boolean) Za pomocą funkcji można kontrolować wykrywanie kolizji „soft limit”. Niekiedy na czas jazdy referencyjnej zachodzi potrzeba wyłączenia SL np. w sytuacji gdy czujnik długości narzędzia ma ujemną wysokość. Po operacji zawsze trzeba załączyć SL.

Funkcje związane z pozycją elementów maszyny

Nazwa	Opis
PosSafe	function PosSafe():extended Zwraca pozycję bezpieczną dla osi narzędziowej. Jest to pozycja krańcówki HOME minus wpisany w ustawieniach zjazd. Zazwyczaj jest to pozycja HOME osi „Z”.

PosBase	function PosBase(axis:cardinal):extended
	
	Zwraca pozycję krańcówki HOME danej osi. Wszędzie gdzie trzeba podać numer osi można stosować zadeklarowane stałe: AXIS_X, AXIS_Y, AXIS_Z, AXIS_A
	
	Przykład:
	`// Ustawia rejestr pozycji osi X położeniem jego krańcówki`
	`// HOME`
	
	`SetX(PosBase(AXIS_X)); ExeSetPos();`

PosPark	function PosPark(axis,num:cardinal):extended Zwraca pozycję park gdzie: axis – numer osi, num – numer pozycji PARK (0-7).
SetPosPark	procedure SetPosPark(axis,num:cardinal; pos:extended) Ustawia pozycję park gdzie: axis – numer osi, num – numer pozycji PARK (0-7). pos – nowa pozycja.
PosMaterial	function PosMaterial(axis:cardinal):extended Zwraca pozycję materiału w danej osi.
SetPosMaterial	procedure SetPosMaterial(axis:cardinal; pos:extended) Ustawia pozycję materiału gdzie: axis – numer osi, pos – nowa pozycja.
MaterialSize	function MaterialSize(axis:cardinal):extended Zwraca rozmiar materiału w danej osi.
SetMaterialSize	procedure SetMaterialSize(axis:cardinal; size:extended) Ustawia rozmiar materiału gdzie: axis – numer osi, size – nowy rozmiar.
PosToolSensor	function PosToolSensor(axis:cardinal):extended Zwraca pozycję czujnika długości narzędzia w danej osi.
SetSpindle	SetSpindle(val:cardinal)' Ustawia parametr „S”

Funkcje związane z osią techniczną

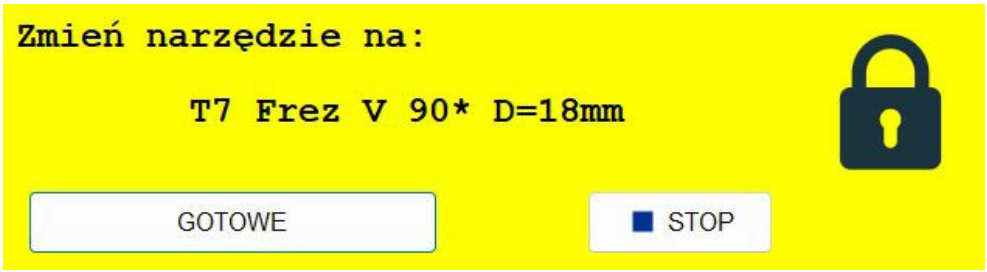
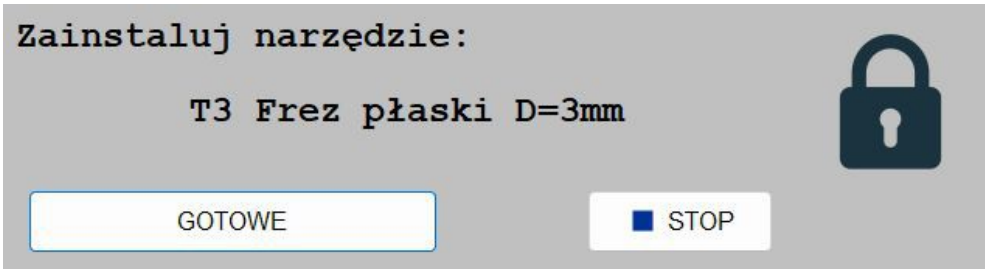
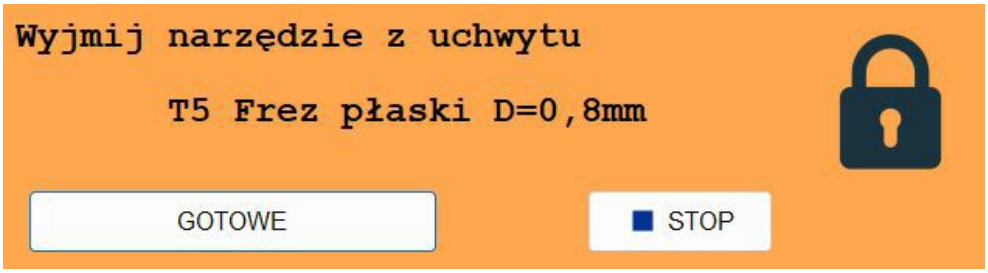
Nazwa	Opis
TAMove	procedure TAMove(pos:cardinal) Inicjuje przejazd osi technicznej do danej pozycji. Pozycja wyrażona jest w impulsach. Kolejna linia programu będzie wykonana gdy zadany ruch się zakończy.
TAMoveInc	procedure TAMoveInc(pos:integer) jw. Inicjuje przejazd osi technicznej o podaną liczbę impulsów – czyli relatywnie względem aktualnej pozycji. Kolejna linia programu będzie wykonana gdy zadany ruch się zakończy.
TAMoveRef	procedure TAMoveRef() Inicjuje cykl jazdy referencyjnej osi technicznej. Kolejna linia programu będzie wykonana gdy jazda referencyjna się zakończy. Funkcja ta powoduje jazdę w kierunku ujemnym, z twardym hamowaniem w momencie aktywacji wejścia TA_PROBE. Po zakończeniu cyklu licznik pozycji osi jest zerowany.
TAMoveRefExt	procedure TAMoveRefExt(mode:cardinal; dir:integer; speed:cardinal) jw. Inicjuje cykl jazdy referencyjnej osi technicznej. Kolejna linia programu będzie wykonana gdy jazda referencyjna się zakończy. UWAGA - po zakończeniu, licznik pozycji osi nie jest ustawiany żadną wartością. Argument „mode” może przyjmować następujące wartości: TA_REF_ON - Jazda do momentu aktywacji wejścia TA_PROBE TA_REF_ON_SOFT - Jazda do momentu aktywacji wejścia TA_PROBE osi z łagodnym hamowaniem. TA_REF_OFF - Jazda do momentu dezaktywacji wejścia TA_PROBE osi. TA_REF_OFF_SOFT - Jazda do momentu dezaktywacji wejścia TA_PROBE osi z łagodnym hamowaniem. Parametr „dir” określa kierunek: (dir = -1) Kierunek w lewo. (dir = 1) Kierunek w prawo. Parametr „speed” określa prędkość wyrażoną w procentach prędkości maksymalnej ustawionej w parametrach osi technicznej np. jeśli wartość maksymalna STEP ustawiona jest na 5KHz to speed = 50 spowoduje jazdę z krokiem o częstotliwości 2.5KHz.
TASpeed	procedure TASpeed(speed:integer) Ustala bieżącą częstotliwość STEP dla osi technicznej. Tak samo jak wyżej, wyrażoną w procentach częstotliwości maksymalnej. Od tego momentu wszystkie ruchy osi będą odbywały się z zadaną częstotliwością STEP.
TASetPosition	procedure TASetPosition(pos:integer) Ustawia pozycję osi wyrażoną w impulsach.
RefNeed	function RefNeed():boolean Pierwsze wywołanie po resecie tej funkcji zwraca TRUE, kolejne FALSE. Może informować o konieczności jazdy referencyjnej.

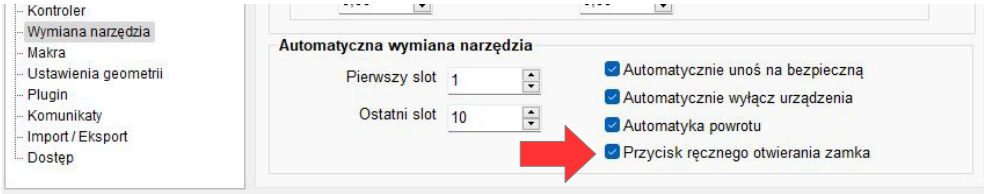
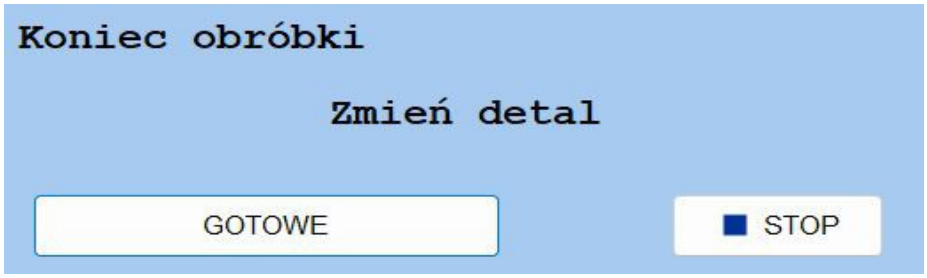
Nazwa	Opis
	Aktualnie, nie zaleca się używania tej funkcji, zamiast tego **`GetRefNeed`** oraz **`SetRefNeed`**.
GetRefNeed	function GetRefNeed():boolean Podobnie jw. funkcja zwraca wartość TRUE jeśli oś techniczna wymaga bazowania, jednak jej wywołanie nie gasi automatycznie flagi. Po rzeczywistym ukończeniu bazowania musimy wywołać funkcję **`SetRefNeed(FALSE)`**.
SetRefNeed	procedure SetRefNeed(st:boolean) Umożliwia kontrolę na flagą konieczności bazowania (patrz wyżej).

Funkcje związane z obsługą magazynka narzędzi

Nazwa	Opis
GetActTool	*function GetActTool():cardinal* Zwraca numer aktualnego narzędzia.
SetActTool	procedure SetActTool(num:cardinal) Funkcja ustawia narzędzie o podanym numerze jako aktualne. Jeżeli jako numer narzędzia podamy zero oznacza to brak narzędzia w uchwycie, a przede wszystkim wyzerowanie korekcji dla osi narzędziowej (najczęściej „Z”). Ważne jest to szczególnie w makrach wymiany narzędzia gdzie istotne jest aktualne położenie oprawki narzędzia a nie jego końca.
GetReqTool	function GetReqTool():cardinal Funkcja zwraca numer narzędzia wymaganego przez program. (wykorzystywana w makrach automatycznej wymiany narzędzia).
SetToolPos	procedure SetToolPos(pos:extended) Ustala pozycję końca aktualnego narzędzia w osi narzędziowej. Ustalona korekcja narzędzia zapisywana jest w magazynku.
SetToolPosExt	procedure SetToolPosExt(axis:cardinal; pos:extended) Ustala pozycję aktualnego narzędzia w podanej osi. Ustalona korekcja narzędzia zapisywana jest w magazynku.
GetToolSlot	function GetToolSlot(num:cardinal):integer Zwraca numer slotu narzędzia o danym numerze. Zwrócona wartość = -1 mówi o tym, że narzędzie nie ma przypisanego slotu.
SetToolSlot	procedure SetToolSlot(num:cardinal; slot:integer) Ustawia numer slotu dla narzędzia o numerze „num”. Jeżeli podany numer slotu

Nazwa	Opis
	przypisany był już do innego narzędzia, to temu narzędziu przypisany jest slot = -1 (brak przypisanego slotu).
GetToolMultiSlot	function GetToolMultiSlot(num:cardinal):cardinal Zwraca 32-bitową liczbę, której każdy bit oznacza wartość odhaczoną w magazynku narzędzi w kolumnie MultiSlot.
ValidTool	function ValidTool(num:cardinal):boolean Sprawdza czy narzędzie o danym numerze jest już zmierzone.
SetValidTool	procedure SetValidTool(num:cardinal; state:boolean) Ustawia stan walidacji narzędzia o numerze „num”. State = TRUE oznacza narzędzie zmierzone.
GetActToolLen	function GetActToolLen():extended Funkcja zwraca długość aktualnego narzędzia (czyli aktualną korekcje dla osi narzędziowej).
GetToolTag	function GetToolTag(num:cardinal):integer Funkcja zwraca wartość z pola Tag narzędzia o danym numerze.
GetToolTyp	function GetToolTyp(num:cardinal):cardinal Funkcja zwraca numer oznaczający typ narzędzia. Zwracane wartości: 0 = Brak narzędzia 1 = Frez płaski 2 = Frez kulowy 3 = Frez V 4 = Wiertło 5 = Frez z promieniem naroża 6 = Nóż tokarski 7 = Sonda pomiarowa
GetToolControlPoint	function GetToolControlPoint(num:cardinal):cardinal Zwraca wartość ustawienia ControlPoint dla narzędzi tokarskich. Num – numer narzędzia.

Nazwa	Opis
ShowToolConfirm	procedure ShowToolConfirm() Wyświetla systemowe okienko żądania zmiany narzędzia na podane. Umożliwia to napisanie ręcznej zmiany narzędzia w makrze. Jest to to samo okno, które pojawia się gdy mamy ustawioną ręczną zmianę narzędzia. Kliknięcie w przycisk „Gotowe” automatycznie ustawia w systemie żądane narzędzie jako aktualne, ale tylko w zakresie numeru. Aby załadować korekcję kolejną instrukcją powinno być **`SetActTool()`**. Aktualnie zalecane jest korzystać wyłącznie z funkcji **`ShowToolConfirmExt()`**
ShowToolConfirmExt	procedure ShowToolConfirmExt(act_tool:cardinal; req_tool:cardinal) Procedura podobna jw. lecz umożliwia podanie własnych numerów aktualnego i żądanego narzędzia. Zależnie od tego dobierany jest komunikat ekranowy: act_tool > 0 oraz req_tool > 0  act_tool = 0 oraz req_tool > 0  act_tool > 0 oraz req_tool = 0  Kliknięcie „Gotowe” nie zmienia nic w systemie. Kolejną instrukcją powinno być **`SetActTool()`**. Ikona kłódki służy do ręcznego otwierania zamka wrzeciona. Jej widoczność można ustawić w *Ustawieniach kontrolera / Wymiana narzędzia.*

Nazwa	Opis
	 Kliknięcie w kłódkę aktywuje w MEMO bit o nazwie C_MTR. Kiedy bit jest aktywny przycisk „Gotowe” jest nieaktywny.
ShowUserConfirm	procedure ShowUserConfirm(caption:string; text:string) Podobnie jak wyżej, ale umożliwia wyświetlenie własnych komunikatów np. `ShowUserConfirm('Koniec obróbki', 'Zmień detal');` 

IsConfirm	function IsConfirm():boolean
	
	Oczekiwanie na potwierdzenie. Starsze wersje programu <5.8 wymagały tej funkcji po wywołaniu **`ShowToolConfirm()`** np.:
	`ShowToolConfirm(); // Wyświetlenie komunikatu`
	`if not IsConfirm() then Wait; // Oczekiwanie na potwierdzenie`
	
	Aktualnie nie ma potrzeby używania tej funkcji!

Funkcje sterowania pętlą G25..G125

Nazwa	Opis
GetLoopCounter	function GetLoopCounter():cardinal; Funkcja zwraca liczbę, która mówi ile razy pętla będzie jeszcze powtórzona. Jeśli zwróci zero oznacza, że nie będzie więcej powtórzeń.
SetLoopCounter	procedure SetLoopCounter(cnt:cardinal); Możemy wymusić daną liczbę powtórzeń lub zakończyć pętlę podając zero jako argument. Obie powyższe procedury powinny być stosowane w makrach, które są uruchamiane wewnątrz pętli G25 G125.

Funkcje czasowych opóźnień

Nazwa	Opis
WaitTime	procedure WaitTime(time:extended) Funkcja zatrzymuje wykonywanie makra na czas „time” wyrażony w sekundach.

Funkcje odczytu przetwornika ADC

Nazwa	Opis
GetADC	function GetADC(chanel:cardinal):extended Funkcja zwraca wartość podanego kanału przetwornika ADC. Zwracana wartość to liczba z zakresu 0.0 – 1.0

Funkcje zapisu/odczytu rejestrów PLC

num – numer bitu (0-31).
devname – nazwa bitu zdefiniowana w PLC.
st – docelowy stan bitu.
W – zapis do rejestru, R – Odczyt rejestru.

Rejestr	R/W	Funkcja / procedura
MEMO	W	procedure SetMemo(num:cardinal; st:boolean)
	W	procedure SetMemoN(devname:string; st:boolean)
	R	function Memo(num:cardinal):boolean
	R	function MemoN(devname:string):boolean
IN	R	function Input(num:cardinal):boolean
	R	function InputN(devname:string):boolean
OUT	R	function Output(num:cardinal):boolean
	R	function OutputN(devname:string):boolean
TIMER	W	procedure SetTimer(num:cardinal; time:extended) Procedura ustala czas timera gdzie: num – numer timera (0-7), time – czas timera w sekundach (0.001 – 65 sek.)
R0 – R21 [PLC v2]	W	procedure SetRegN(devname:string; val:integer) devname – zadeklarowana nazwa rejestru. Jeśli rejestr o takiej nazwie nie istnieje, to zgłoszony zostanie błąd, a makro przerwane. val – zapisywana wartość.
	W	procedure SetReg(num:cardinal; val:integer) num – numer rejestru (0 – 21) val – zapisywana wartość.
	R	function GetRegN(devname:string):integer j.w.
	R	function GetReg(num:cardinal):integer j.w.
Kontrolki wirtualne [PLC v2]	R	function GetVirtual(devname:string):boolean Funkcja odczytuje stan kontrolki wirtualnej o podanej nazwie. Jeżeli kontrolka o takiej nazwie nie jest zadeklarowana w PLC, to funkcja zwróci wartość FALSE;

Kinematyka

<table>
<tr><th>Nazwa</th><th>Opis</th></tr>
<tr><td>KinematicLoad</td><td>procedure KinematicLoad(slot:integer; KinName:string)

Procedura ładuje kinetykę o nazwie podanej w parametrze „KinName” do slotu o numerze podanym w parametrze „slot”. Parametr slot może przyjmować wartości od 0 do 9, zatem możemy mieć jednocześnie zdefiniowanych 10 kinematyk. Slot nr. 0 jest slotem domyślnym i zazwyczaj tylko z niego będziemy korzystać.

W parametrze „KinName” podajemy nazwę kinematyki, aktualnie dostępne są trzy rodzaje:

<table><tr><td>„xyzab_srt”</td><td>obrotowo - uchylne wrzeciono</td></tr><tr><td>„xyzab_trtx”</td><td>obrotowo – uchylny w osi X stolik</td></tr><tr><td>„xyzab_trty”</td><td>obrotowo – uchylny w osi Y stolik</td></tr></table>
Przykład:
<pre>KinematicLoad(0,'xyzab_srt');</pre></td></tr>
<tr><td>KinematicSetParam</td><td>procedure KinematicSetParam(slot:integer; indx:cardinal; val:extended)

Każda kinematyka posiada szereg parametrów, które trzeba ustawić przed jej użyciem. Slot – slot kinematyki, indx – numer parametru, val – wartość parametru.

Przykład, ustawienie środka obrotu (pivot):
<pre>KinematicLoad(0,'xyzab_trtx');
KinematicSetParam(0, 3, 200.0); // X pivot
KinematicSetParam(0, 4, 150.0); // Y pivot
KinematicSetParam(0, 5, 80.0); // Z pivot</pre>
Pełną listę parametrów można znaleźć niżej.

Nie należy ustawiać parametrów, które nie są udokumentowane.</td></tr>
<tr><td>KinematicSet</td><td>procedure KinematicSet(slot:integer)

Załączenie kinematyki z danego slotu. Jeżeli chcemy wyłączyć podajmy wartość -1 w slot. Jednocześnie aktywna jest tylko jedna kinematyka.

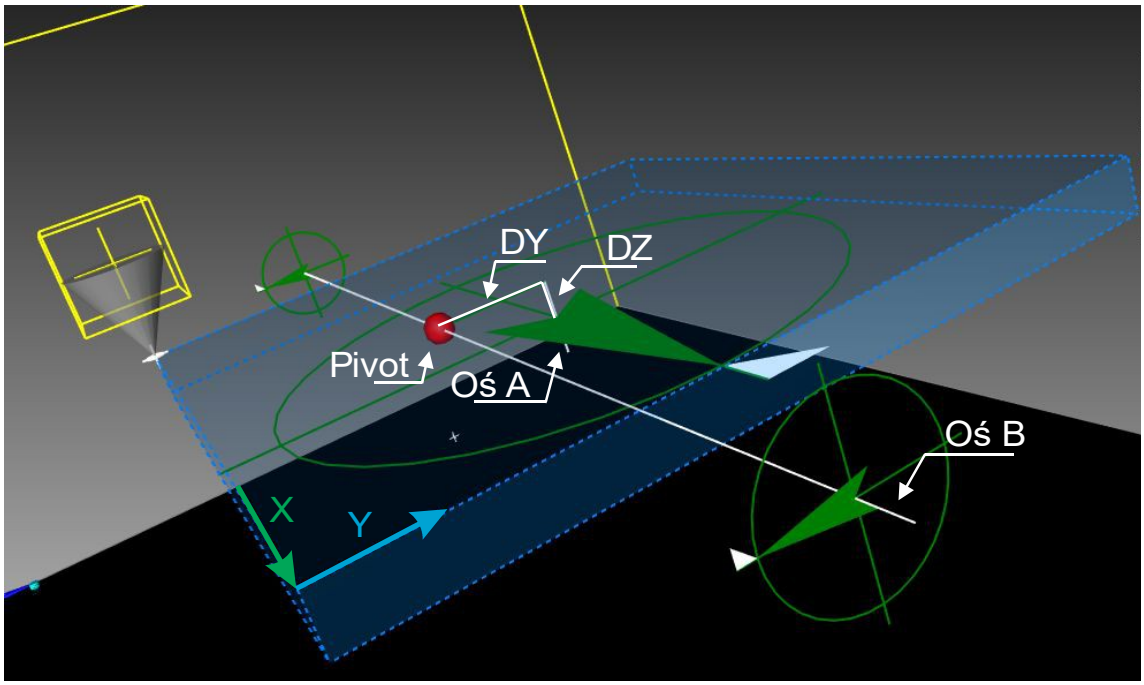
Przykład:
<pre>KinematicSet(0) // załączenie kinematyki ze slotu 0
KinematicSet(-1) // wyłącznie kinematyki</pre></td></tr>
<tr><td>KinematicGetCurrent</td><td>function KinematicGetCurrent():integer

Funkcja zwraca slot aktywnej kinematyki. Wartość -1 oznacza, że żadna nie jest aktywna.</td></tr>
</table>

Parametry kinematyki

Obrotowo – uchylny w osi X stolik „xyzab_trtx”

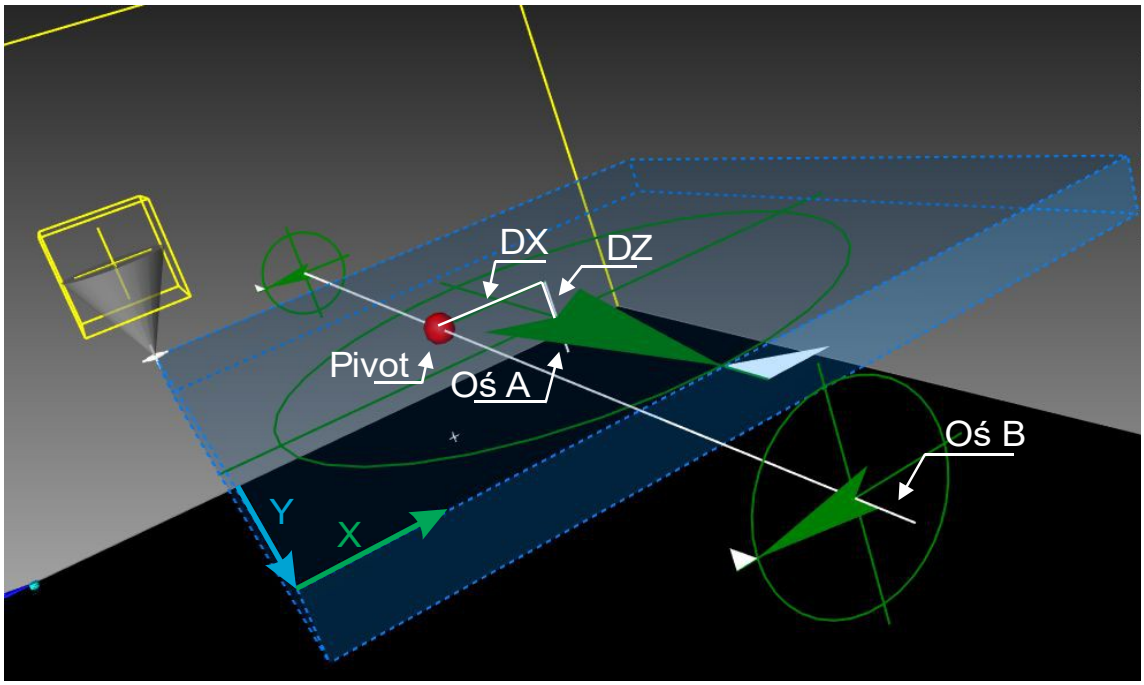
Nr.	Parametr	Wartość domyślna
0	--------	
1	DY Offset Pivot Y do Y środka obrotu stolika	0.0
2	DZ Offset Pivot Z do Z powierzchni stolika	0.0
3	Pozycja X Pivot	0.0
4	Pozycja Y Pivot	0.0
5	Pozycja Z Pivot	0.0
6	Średnica stolika (tylko do wizualizacji)	100.0
7	Sposób wyświetlania kinetyki osi B: 0 - oba końce, 1 - lewy, 2 - prawy	0
10	Krok interpolacji osi obrotowych (stopnie)	1



Aby załączyć podgląd kinematyki jak na powyższym obrazku, należy w ustawieniach widoku kliknąć w „Szczegóły”.

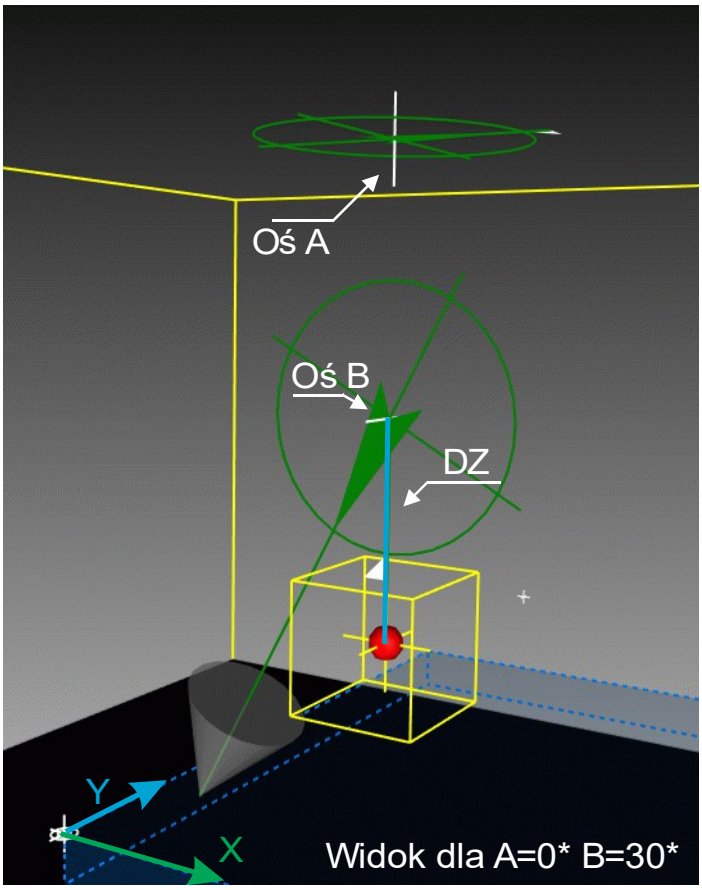
Obrotowo – uchylny w osi Y stolik „xyzab_trty”

Nr.	Parametr	Wartość domyślna
0	DX Offset Pivot X do X środka obrotu stolika	0.0
1	--------	
2	DZ Offset Pivot Z do Z powierzchni stolika	0.0
3	Pozycja X Pivot	0.0
4	Pozycja Y Pivot	0.0
5	Pozycja Z Pivot	0.0
6	Średnica stolika (tylko do wizualizacji)	100.0
7	Sposób wyświetlania kinetyki osi B: 0 - oba końce, 1 - lewy, 2 - prawy	0
10	Krok interpolacji osi obrotowych (stopnie)	1



Obrotowo - uchylne wrzeciono „xyzab_srt”

Nr.	Parametr	Wartość domyślna
0	DZ Odległość punktu referencyjnego wrzeciona do środka obrotu osi B	50.0
6	Średnica kół osi (tylko do wizualizacji)	60.0
10	Krok interpolacji osi obrotowych (stopnie)	1



Kinematyka, przykłady konfiguracji

Parametry kinematyki powinny być ustawiane w makrze OnRun

```
// OnRun
%
KinematicLoad(0, 'xyzab_trtx');  // musi być pierwsze!
KinematicSetParam(0, 2, -10.0);  // Dz
KinematicSetParam(0, 1, 30.0);   // Dy
KinematicSetParam(0, 3, 200.0);  // X pivot
KinematicSetParam(0, 4, 150.0);  // Y pivot
KinematicSetParam(0, 5, 80.0);   // Z pivot
KinematicSetParam(0, 10, 0.5);
KinematicSetParam(0, 7, 0);
KinematicSetParam(0, 6, 60);
```

Zauważmy, że część parametrów będzie zawsze miała stałą wartość, a część może się okresowo zmieniać np. pozycja punktu Pivot, dlatego przypisywanie tam stałych jako parametry (jak w przykładzie wyżej), jest mało praktyczne. W tych przypadkach lepiej stosować funkcje **`GetVar`**. Funkcja ta, dla indeksów z zakresu 200-249 pamięta wartości także po wyłączeniu zasilania, dlatego z tego zakresu należy korzystać.

```
...
KinematicSetParam(0, 3, GetVar(200));  // X pivot
KinematicSetParam(0, 4, GetVar(201));  // Y pivot
KinematicSetParam(0, 5, GetVar(202));  // Z pivot
...
```

Aby ułatwić ustawianie punktu Pivot dodano makro systemowe o tej nazwie o następującej składni:

```
//=============================================================================
// X Y Z - wybór osi, przypisanie aktualnej pozycji narzędzia powiększonej o
//         podaną wartość.
// I J K - wybór osi, wartość bezpośrednia pozycji dla danej osi (X,Y,Z)
//=============================================================================
```

Przykład:

PIVOT X0 Y0 Z0	Przypisanie do pivot aktualnej pozycji narzędzia XYZ
PIVOT X0 Y-30 Z10	Przypisanie do pivot aktualnej pozycji XYZ narzędzia. Narzędzie w pozycji środka obrotowego stolika
PIVOT I100	Przypisanie X pivot wartości 100.

Makro PIVOT wykorzystuje indeksy 200-202 dla przechowywania wartości XYZ.

Kinematyka, załączanie

Z poziomu MDI (G-kodu) możemy kontrolować kinematykę kodami:

G234	Załączenie. Po kropce można ewentualnie podać, o który slot chodzi np. G234.1. Domyślnie slot zero.
G235	Wyłączenie.

Jeżeli chcemy załączyć kinetykę z poziomu makra, to musimy pamiętać, aby wcześniej wyzerować korekcję narzędzia ponieważ kinematyka musi pracować na zerowych offsetach.

Systemowe makro G234 załączające kinematykę.

```
// G234
%
ExeSetCorrectionExt(0, 0); // Zerowanie offsetów
KinematicSet(DotVal());
```

Natomiast, po wyłączeniu kinetyki należy przywrócić offsety aktualnego narzędzia.

Systemowe makro G235 wyłączające kinematykę.

```
// G235
%
KinematicSet(-1);
ExeSetCorrectionExt(0, GetActTool()); //Przeładownie
```

MODBUS

Z poziomu makr możemy komunikować się poprzez protokół Modbus RTU z zewnętrznymi urządzeniami takimi jak zdalne wyjścia/wejścia, PLC, itd. Konfigurację połączenia należy dokonać w oknie ustawień kontrolera pozycja „MODBUS”. Szczegółowy opis samego protokołu można znaleźć pod linkiem: MODBUS

Nazwa	Opis

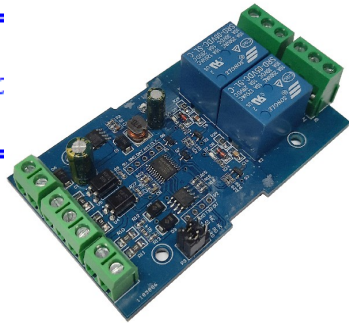
MB_SendStr	function MB_SendStr(data:string):boolean;
	
	Wysyła ramkę danych zapisaną jako ciąg liczb szesnastkowych. W ramce pomijamy ostatnie dwa bajty sumy kontrolnej – jest ona wyliczana automatycznie.
	
	Przykład:
	`MB_SendStr('FF 05 00 00 FF 00');`

MB_Send	function MB_Send():boolean;
	
	Wysyła ramkę danych utworzoną za pomocą procedur: MB_InitFrame, MB_wr_BYTE, MB_wr_WORD. Tak, jak wyżej w ramce pomijamy ostatnie dwa bajty sumy kontrolnej.
	
	Przykład wysłania ramki identycznej jak w powyższym przykładzie:
	`MB_InitFrame($FF, 5);`
	`MB_ wr_WORD(0);`
	`MB_ wr_WORD($FF00);`
	`MB_Send():`

MB_InitFrame	procedure MB_InitFrame(addr:byte; command:byte); Inicjuje tworzenie nowej ramki. addr – adres urządzenia slave, command – komenda Modbus.
MB_wr_BYTE	procedure MB_wr_BYTE(data:byte); Zapisuje kolejny bajt do ramki danych do wysłania.
MB_wr_WORD	procedure MB_wr_WORD(data:word); Zapisuje kolejne słowo (2 bajty) do ramki danych do wysłania.
MB_rd_BYTE	function MB_rd_BYTE(indx:cardinal):byte; Odczyt bajtu danych z ramki, którą wysłało urządzenie. indx – indeks bajtu (numeracja od zera).
MB_rd_WORD	function MB_rd_WORD(indx:cardinal):word; Odczyt słowa (2 bajty) danych z ramki, którą wysłało urządzenie. indx – indeks pierwszego bajtu (numeracja od zera).

Nazwa	Opis
MB_rd_STR	function MB_rd_STR(indx:cardinal):string; Funkcja zwraca ramkę powrotną w postaci tekstowej (bez bajtów sumy kontrolnej) – jako ciąg bajtów zapisanych szesnastkowo. np. 'FF 05 00 00 FF 00'.
MB_RxCount	function MB_RxCount():cardinal; Funkcja zwraca ilość bajtów w ramce powrotnej – bez dwóch bajtów sumy kontrolnej.

Niżej przykład użycia popularnego modułu z dwoma wejściami i dwoma wyjściami. Program w zamkniętej pętli załącza i wyłącza naprzemiennie przekaźniki, natomiast stan wejść przepisuje na bity 30,31 rejestru MEMO. Program znajduje się w przykładach dołączonych do programu. Wyjścia sterowane są pojedynczo za pomocą komendy „5” Modbus, natomiast wejścia odczytywane komendą „2” i odczytany stan bitów kopiowany do 10 i 11-tego rejestru zmiennych. Moduł ma przypisany adres 255 ($FF).



```
//=============================================
//
//                          Przykład użycia MC
//
//=============================================

procedure MB_OUT(num:byte; state:boolean);
begin
  MB_InitFrame($FF,5);
  MB_wr_word(num);
  if state then MB_wr_word($FF00) else MB_wr_word($0000);
  MB_Send();
end;

procedure MB_READ_IN();
begin
   MB_SendStr('FF 02 00 00 00 08');
   SetVar(10, boolean(MB_rd_byte(3) and $01));
   SetVar(11, boolean(MB_rd_byte(3) and $02));
end;

%
:LOOP
  MB_READ_IN();
  SetMemo(30, GetVar(10));
  SetMemo(31, GetVar(11));
  MB_OUT(0, TRUE);
  MB_OUT(0, FALSE);
  MB_OUT(1, TRUE);
  MB_OUT(1, FALSE);
  Gotolabel('LOOP');
```

Funkcje związane z drukowaniem komunikatów w okienku MDI

Nazwa	Opis
Log	procedure Log(txt:string) Dodaje w oknie komunikatów nową linię z podanym tekstem.
LogClr	procedure LogClr() Czyści okno komunikatów.
LogShow	procedure LogShow() Otwiera okienko MDI jeśli nie jest widoczne.
FloatToStr	function FloatToStr(v:Extended):string Konwertuje zmienną typu Extended na tekst.
IntToStr	function IntToStr(v:integer):string Konwertuje zmienną typu integer lub cardinal na tekst.
VarToStr	function VarToStr(num:cardinal):string Konwertuje zmienną o danym numerze na tekst. Typ zmiennej rozpoznawany jest automatycznie.
Time	function Time():string Zwraca aktualny czas w postaci tekstu.

Komentarze

Możemy stosować dwa rodzaje komentarzy dla pojedynczej linii „//” lub dla większej ilości linii nawiasy klamrowe: „{„ oraz „}”. Znaki klamry muszą być jednak jako pierwsze znaki w linii nie licząc spacji i tabulacji. Natomiast komentarz typu „//” może być w dowolnym miejscu linii tekstu.

Pozostałe zadeklarowane stałe

Nazwa	Opis
SMT_HEIGHT	Wysokość czujnika długości narzędzia zadeklarowana w ustawieniach.
SMM_HEIGHT	Wysokość czujnika wysokości materiału zadeklarowana w ustawieniach.

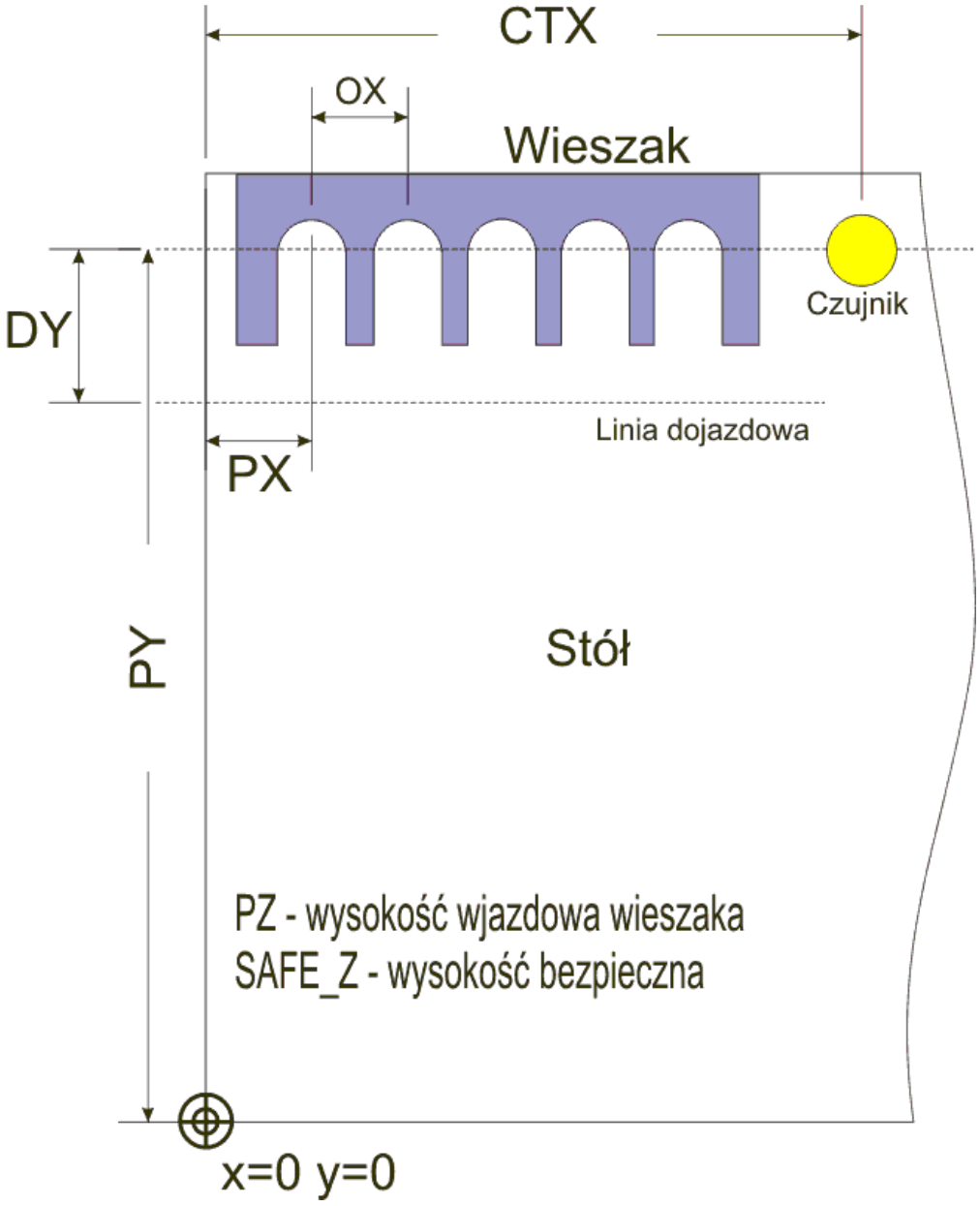
Przykłady

Makro wymiany narzędzia M6

Załączenie automatycznej wymiany narzędzia dokonuje się w ustawieniach zakładka „G kod / Zmiana narzędzia (M6)”. Należy tam zaznaczyć „Makro M6”. Makro można także wywoływać z okienka MDI np. wpisując „M6 T6” zmienimy narzędzie na T6.

Przedstawione niżej makro można znaleźć w katalogu macros/exaples. Aby poeksperymentować z nim należy je przenieś do katalogu macros/.

Rozmieszczenie elementów: wieszak na oprawki narzędzi zamocowany u góry stołu, obok czujnik do pomiaru długości narzędzia.



```
Const                          // Deklaracja stałych
  F_NAJAZD=400;
  F_ZJAZD=800;
  F_MESS=600;
  PX=20.0;                     // deklaracja wymiarów do powyższego rysunku
  PY=270.0;
  PZ=80.0;
  DY=30.0;
  Z_UP=40.0;
  OX=40.0;
  CTX=250;
  CT_HEIGHT=29.0;        // wysokość czujnika długości narzędzia
  ZAMEK=6;               // Numer bitu dla zamka
  TOOL_IN_HANDLE=6;  // Wejście wykrywające narzędzie w uchwycie

  USER_POS_X=0.0;        // Pozycja dojazdowa narzędzia w sytuacji uruchomienia makra ręcznie
  USER_POS_Y=0.0;        // ….

  ACT_TOOL=0;            // Indeks zmiennej pamiętającej numer aktualnego narzędzia
  REQ_TOOL=1;            // Indeks zmiennej pamiętającej numer żądanego narzędzia

%
  if OutputN('SPINDLE') then UserError('Wrzeciono jest załączone !');
  SetZ(PosSafe); ExeMove(0);             // Uniesienie „Z” na wysokość bezpieczną
  SetVar(ACT_TOOL,GetActTool);           // Zapamiętanie aktulnego narzędzia
  if UserExe then SetVar(REQ_TOOL, GetArgDef('T',0)) else SetVar(REQ_TOOL,GetReqTool);
  if (GetToolSlot(GetVar(REQ_TOOL))<0) then UserError('Narzędzie T'+IntToStr(GetVar(REQ_TOOL))+' nie
        ma przypisanego slotu !');
  if (GetToolSlot(GetVar(ACT_TOOL))<0) then UserError('Narzędzie T'+IntToStr(GetVar(ACT_TOOL))+' nie
        ma przypisanego slotu !');
  if (GetVar(ACT_TOOL)=0) and (GetVar(REQ_TOOL)=0) then return;
  if (GetVar(ACT_TOOL)=0) then GotoLabel('M6_GET');  // Skok do M6_GET jeśli w uchwycie nie ma
        narzędzia
  if (GetVar(ACT_TOOL)=GetVar(REQ_TOOL)) then GotoLabel('M6_MESS');

:M6_PUT
  SetX(PX+(OX*(GetToolSlot(GetVar(ACT_TOOL)) - 1))); SetY(PY-DY); ExeMove(0);
  SetActTool(0);                 // Odwołanie korekcji „Z” -- BARDZO WAŻNE !!!
  SetZ(PZ); ExeMove(0);          // Uchwyt na pozycję wjazdową „Z”
  SetY(PY); ExeMove(0);          // Wjazd w uchwyt
  SetMemo(ZAMEK, TRUE);                // Zwolnienie zamka
```

```
  WaitTime(0.2);                    // Oczekiwanie 0.2 sek.
  SetZ(PZ+Z_UP); ExeMove(0);        // Wyjazd w górę
  SetMemo(ZAMEK, FALSE);            // Załączenie zamka

:M6_GET
  if (Input(TOOL_IN_HANDLE)) then UserError('Narzędzie nadal w uchwycie !');
  if (GetVar(REQ_TOOL)=0) then GotoLabel('M6_ENDPROCES');
  SetX(PX+((GetToolSlot(GetVar(REQ_TOOL))-1)*OX)); SetY(PY); ExeMove(0); // Dojazd nad odpowiedni
        slot
  SetMemo(ZAMEK, TRUE);             // Zwolnienie zamka
  SetZ(PZ); ExeMove(0);             // „Z” na pozycję wyjazdową (najazd na narzędzie)
  SetMemo(ZAMEK, FALSE);            // Załączenie zamka
  WaitTime(0.2);
  SetY(PY-DY); ExeMove(0);          // Wyjazd z uchwytu
  SetActTool(GetVar(REQ_TOOL));     // Ustalenie narzędzia jako aktualnego
  SetZ(PosSafe); ExeMove(0);        // Uniesienie na wysokość bezpieczną

:M6_MESS
  if (not Input(TOOL_IN_HANDLE)) then UserError('Brak narzędzia w uchwycie !');
  if ValidTool(GetActTool) or UserExe then GotoLabel('M6_ENDPROCES');
  SetX(CTX); SetY(PY);  ExeMove(0);
  RefZ(-1); ExeRef(PROBE_ON, F_MESS);
  SetToolPos(CT_HEIGHT);

:M6_ENDPROCES
  SetZ(PosSafe); ExeMove(0);
  if UserExe then begin SetX(USER_POS_X); SetY(USER_POS_Y); ExeMove(0); end;
  return;
```

Makro G0

Makro można znaleźć w katalogu Macros/Elcosimo/. Aby przetestować w okienku MDI można wpisać np. „G0 X0 Y0” +ENTER .

```
//=============================================================================
//  Makro symulujące kod G0. Parametry:
//  X Y Z A - Docelowa pozycja osi.
//=============================================================================
%
  if IsArg('X') then SetX(GetArg('X')+PosMaterial(AXIS_X));
  if IsArg('Y') then SetY(GetArg('Y')+PosMaterial(AXIS_Y));
  if IsArg('Z') then SetZ(GetArg('Z')+PosMaterial(AXIS_Z));
  if IsArg('A') then SetA(GetArg('A')+PosMaterial(AXIS_A));
  ExeMove(0);
  Return;
```

Makro do testowania położenia maszyny

Makro można znaleźć w katalogu Macros/Elcosimo/

```
//============================================================================
// Makro do sprawdzania położenia maszyny
// ( odchyłek od poprzedniej jazdy referencyjnej )
//============================================================================
const
F_NAJAZDU = 0;    // Prędkość najazdu na czujnik  ( 0 oznacza max )
F_REF = 600;
DIR_X = 1;        // Kierunki szukania krańcówek HOME 1 w prawo, -1 w lewo
DIR_Y = 1;        // ...
TOOL_MESS = TRUE; // Czy pomiar narzędzia...
FAST_DOWN = 0.0;     // Zakres szybkiego ruchu w dół do czujnika

%
  RefZ(1); ExeRef(HOME_ON_SOFT, F_NAJAZDU);
  RefX(DIR_X); RefY(DIR_Y); ExeRef(HOME_ON_SOFT, F_NAJAZDU);
  RefX(-1*DIR_X); RefY(-1*DIR_Y); RefZ(-1); ExeRef(HOME_OFF, F_REF);

  LogShow;
  Log('=========================================');
  Log('['+Time+'] Test położenia osi...')
  Log('=========================================');
  Log('dX = '+FloatToStr(PosX - PosBase(AXIS_X)));
  Log('dY = '+FloatToStr(PosY - PosBase(AXIS_Y)));
  Log('dZ = '+FloatToStr(PosZ - PosBase(AXIS_Z)));

  SetX(PosToolSensor(AXIS_X)); SetY(PosToolSensor(AXIS_Y));  ExeMove(0);
  if not TOOL_MESS then GotoLabel('END_PROCES');

  SetZ(PosZ - FAST_DOWN); ExeMove(0);
  Softlimit(FALSE);
  RefZ(-1); ExeRef(PROBE_ON, F_REF);
  Log('dT = '+FloatToStr(PosZ - PosToolSensor(AXIS_Z)));
  SetZ(PosSafe); ExeMove(0);
  Softlimit(TRUE);

:END_PROCES
  Log('=========================================');
```

PPHU ELCOSIMO
Andrzej Woźniak
ul. Zielona 1B
62-110 Damasławek