PikoCNC Macro manual

wersja 1.1

Wstęp

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

Język programowanie 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 katalogu Macros/ naszego katalogu z profilem (najczęściej to katalog ProfilDef). Plik jest najpierw szukany w katalogu Macros/ a jeśli go tam nie ma, to przeszukiwany jest katalog Macros/Elcosimo/ . Jednak własne makra zawsze trzeba zapisywać w Macros/ gdyż katalog Elcosimo/ jest nadpisywany przy każdej instalacji programu.

W katalogu Macros/Elcosimo/ można znaleźć szereg przykładów i otwierając za pomocą wbudowanego edytora makr 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”.

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.
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.
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.: ...kod... :main_loop // etykieta o nazwie main_loop ...kod...
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.

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.

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.
	
	np.;
	SetZ(20.0); ExeMove(0); // Przejazd G0 osi Z na pozycję 20.0
	SetX(10.0); SetY(20.0); ExeMove(1000); // Przejazd G1 F1000 na pozycję x10.0 y20.0

WriteMove	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.
ExeSetPos	procedure ExeSetPos() Ustawia pozycję osi na podstawie zawartości rejestru pozycji. np.; SetZ(0.0); ExeSetPos; // Zerowanie pozycji osi Z

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.
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 np.; SetX(PosBase(AXIS_X)); // Ustawia rejestr pozycji osi X położeniem jego krańcówki HOME
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. Dla osi narzędziowej zwraca wysokość czujnika.
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.
TAMoveRef	procedure TAMoveRef() Inicjuje cykl jazdy referencyjnej osi technicznej. Kolejna linia programu będzie wykonana gdy jazda referencyjna się zakończy.
RefNeed	function RefNeed():boolean Pierwsze wywołanie po resecie tej funkcji zwraca TRUE, kolejne FALSE. Może informować o konieczności jazdy referencyjnej.

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(posZ:extended) Ustala pozycję końca narzędzia.
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.
ValidTool	function ValidTool(num:cardinal):boolean Sprawdza czy narzędzie o danym numerze jest już zmierzone.

Nazwa	Opis
GetActToolLen	function GetActToolLen():extended Funkcja zwraca długość aktualnego narzędzia (czyli aktualną korekcje dla osi narzędziowej).

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.)

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.

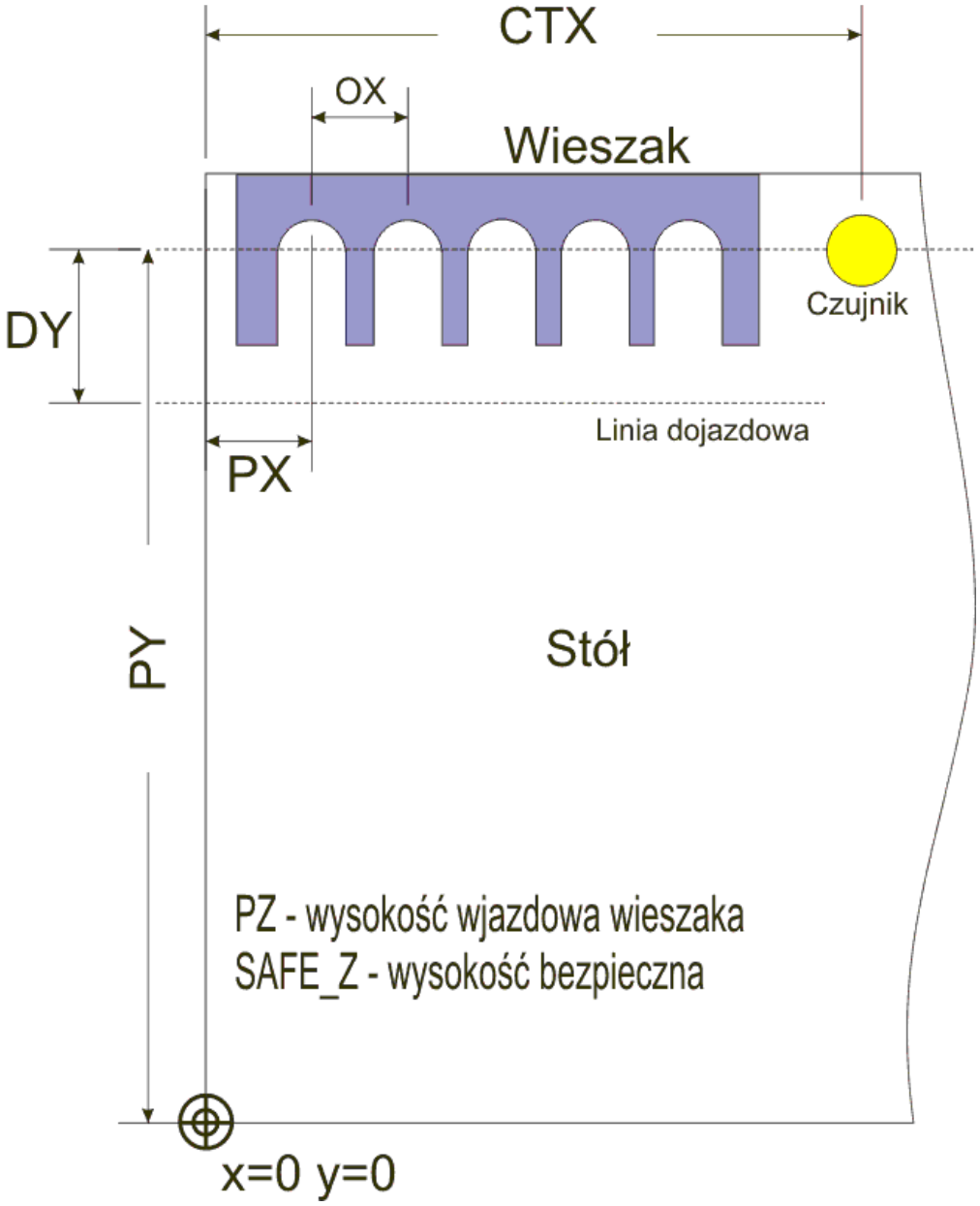
Przykłady

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