Elcosimo

# *PikoCNC G-kod, Skrypty*

***wersja 1.0***

***2016.06.20***

# Interpretacja G-Kodu

## *Znak „%”*

Każdy program (komendy dla maszyny) powinien rozpoczynać się od znaku „%”. Wszystkie linie, które występują w programie przed tym znakiem nie są interpretowane i są traktowane jako komentarz !!. Tak więc jeśli po wczytaniu jakiegoś programu na ekranie nic się nie pojawi - to znak, że brak „%” i trzeba go dopisać.

*Np:*

```
Krótki program          ; linia bez znaczenia !
Frez 2mm                ; linia bez znaczenia !
G0 X30 Y30              ; linia bez znaczenia !
%                       ; zaczyna się program !!
G0 X10 Y10
G0 Z-1
G1 X20 Y20
```

## *Komentarze*

Znaki występujące w linii po „ ( „ lub „ ' ” (apostrof) lub „ ; ” są traktowane jako komentarz.

*Np:*

```
( Frez 2mm        - komentarz
' Frez 2mm        - komentarz
; Frez 2mm        - komentarz
 G1 X20.6 Y36  ( koniec
```

## *Format G-kodu*

Nie ma znaczenia wielkość liter, spacje, kropka czy przecinek, czy też sposób zapisu numeru G-kodu. Wszystkie linie poniżej są sobie równoważne:

```
G1 X45.5 Y30.2
G01 x45,5 Y 30,2
g1x45.5y30.2
```

## *Numeracja linii*

Numeracja jest całkowicie ignorowana przez program i jest obojętne czy jest czy jej nie ma. Program jest wykonywany według kolejności występowania komend w programie.

## *Obsługiwane G-kody*

| Kod | Argumenty | Opis |
|---|---|---|
| G0 | X,Y,Z,A | Dojazdy. Ruch wykonywany zawsze z prędkością maksymalną. |
| G1 | X,Y,Z,A | Ruch roboczy po prostej. Wykonywany z prędkością ustawioną parametrem F |
| G2 | X,Y,Z,I,J,R | Ruch roboczy po łuku. Kreśli łuk (zgodnie z kierunkiem zegara) od aktualnej pozycji do X,Y o środku o współrzędnych X+I,Y+J lub o promieniu R. |
| G3 | X,Y,Z,I,J,R | Ruch roboczy po łuku. Kreśli łuk (przeciwnie do kierunku zegara) od aktualnej pozycji do X,Y o środku o współrzędnych X+I,Y+J lub o promieniu R. |
| G4 | P | Czasowe zatrzymanie programu. Czas podawany w sekundach. Np. G4 P0.5 zatrzyma wykonywanie programu na 0.5 sek. |
| G20 | | Program w systemie calowym. |
| G21 | | Program w systemie metrycznym. |
| G28 | | Przejazd do pozycji HOME1 (najpierw oś narzędziowa potem pozostałe) |
| G30 | | Przejazd do pozycji HOME2 (najpierw oś narzędziowa potem pozostałe) |
| G40 | | Odwołanie kompensacji narzędzia (patrz temat kompensacja...) |
| G41 | R | Kompensacja lewostronna (patrz temat kompensacja...) |
| G42 | R | Kompensacja prawostronna (patrz temat kompensacja...) |
| G80 | | Odwołanie cyklu wiercenia. |
| G81 | X,Y,Z,R | Podstawowy cykl wiercenia.<br>%<br>G0 Z2.<br>G98 (ustalenie aktualnego Z jako wysokości przejazdowej<br>G81 X40 Y100 Z-3.2 R1.0 F150<br>X50 Y100<br>X60 Y110<br>X70 Y120<br>G80<br>W wyniku działania tego programu zostaną wywiercone cztery otwory o głębokości 3.2mm. W czasie przejazdów między otworami frez będzie się unosił 2mm nad materiałem. Cykl wiercenia pojedynczego otworu wygląda następująco:<br>1. Zejście G0 na wysokość R<br>2. Zejście G1 na głębokość Z-3.2 z prędkością F150<br>3. Wycofanie z prędkością G0 do wysokości R<br>4. Jeżeli załączony jest tryb G98 to dojazd do Z przejazdowego (2mm)<br>5. Przejazd XY do pozycji następnego otworu i kolejny cykl. |
| G83 | X,Y,Z,R,Q | Cykl wiercenia z wycofaniem (dzięcioł)<br>%<br>G0 Z2.<br>G98 (ustalenie aktualnego Z jako wysokości przejazdowej<br>G81 X40 Y100 Z-6 Q2.0 R1.0 F150<br>X50 Y100<br>X60 Y110<br>X70 Y120<br>G80 |

| Kod | Argumenty | Opis |
|---|---|---|
| | | W wyniku działania tego programu zostaną wywiercone cztery otwory o głębokości 3.2mm. W czasie przejazdów między otworami frez będzie się unosił 2mm nad materiałem. Cykl wiercenia pojedynczego otworu wygląda następująco:<br><br>1. n=0; („n” - numer przejścia)<br>2. Zejście G0 na wysokość R<br>3. Zejście G0 na głębokość R-(n*Q)<br>4. Zejście G1 na głębokość R-((n+1)*Q) z prędkością F150<br>5. Wycofanie z prędkością G0 do wysokości R<br>6. n=n+1;<br>7. Skok do punktu nr.3 do momentu osiągnięcia docelowej głębokości.<br>8. Jeżeli załączony jest tryb G98 to dojazd do Z przejazdowego (2mm)<br>9. Przejazd XY do pozycji następnego otworu i kolejny cykl. |
| G84 | X,Y,Z,R,Q | Cykl gwintowania za pomocą głowicy TAPMATIC.<br><br>%<br>G0 Z20.<br>G84 X40 Y100 Z-10.0 Q5.0 R1.0 F150<br>X50 Y100<br>X60 Y110<br>X70 Y120<br>G80<br><br>W wyniku działania tego programu zostaną nagwintowane cztery otwory na głębokość 10mm. W czasie przejazdów między otworami gwintownik będzie się unosił 20mm nad materiałem. Cykl gwintowania pojedynczego otworu wygląda następująco:<br><br>1. Zejście G0 na wysokość R<br>2. Zejście G1 na głębokość Z-10.0 z prędkością F150<br>3. Wycofanie G0 do głębokości Z-Q<br>4. Wycofanie G1 z prędkością F150 do Z20<br>5. Przejazd XY do pozycji następnego otworu i kolejny cykl. |
| G80 | | Odwołanie cyklu wiercenia / gwintowania. |
| G90 | | Załączenie trybu absolutnego zadawania pozycji dla ruchów ustawczych i roboczych. |
| G91 | | Załączenie trybu przyrostowego zadawania pozycji dla ruchów ustawczych i roboczych. |
| G98 | | Ustalenie aktualnego Z jako wysokości przejazdowej dla cykli G81, G83. |
| G99 | | Ustalenie R jako wysokości przejazdowej dla cykli G81, G83. |
| G100.. G110 | P,Q | Wykonanie makra o takiej nazwie. Jako argumenty mogą być przekazane parametry P,Q np.<br><br>G100 P20 Q12.4<br><br>uruchomi makro o nazwie G100 z argumentami „P20 Q12.4” |
| G115 | | Wyłączenie automatyki dojazdu na początku i końcu programu. Wystąpienie tego kodu gdziekolwiek w programie wyłącza automatykę dojazdów – maszyna nie wykona żadnego innego ruchu niż to co zapisano w programie. |
| G116 | | Odblokowanie możliwości naciśnięcia pauzy. |

| Kod | Argumenty | Opis |
|---|---|---|
| G117 | | Zablokowanie możliwości naciśnięcia pauzy. |

**Uwaga !**

- G-kody oraz M-kody nie obsługiwane przez program są ignorowane.
- Argumenty nie obsługiwane przez program lub dany g-kod są ignorowane.

## *Obsługiwane M-kody*

| M kod | Opis |
|---|---|
| M0 | Program stop. Program zostanie zatrzymany w ten sposób jakby naciśnięto pauzę. Po ponownym naciśnięciu pauzy program jest wznawiany. |
| M3 | Załączenie wrzeciona obroty CW.<br>Reakcja programu na ten kod zależy od ustawień programu, które mogą być następujące:<br>• Zapalenie bit C_M3 w rejestrze MEMO PLC.<br>• Uruchomienie makra o nazwie „M3” |
| M4 | Załączenie wrzeciona obroty CCW. Fizycznie kod zapala bit C_M4 w rejestrze MEMO PLC. |
| M5 | Wyłączenie wrzeciona.<br>Reakcja programu na ten kod zależy od ustawień programu, które mogą być następujące:<br>• Gaszenie bity C_M3 oraz C_M4 w rejestrze MEMO.<br>• Uruchomienie makra o nazwie „M5” |
| M6 | Zmiana narzędzia. Reakcja programu na ten kod zależy od ustawień programu, które mogą być następujące:<br>• Brak reakcji.<br>• Wywołanie wbudowanej procedury ręcznej wymiany narzędzia.<br>• Wywołanie procedury automatycznej wymiany narzędzia (makro o nazwie „M6”).<br><br>M6 musi występować zawsze razem z numerem żądanego narzędzia np. M6T5 |
| M7 | Zapala bit C_M7 w rejestrze MEMO. |
| M8 | Załączenie chłodziwa. Zapala bit C_M8 w rejestrze MEMO. |
| M9 | Gasi bity C_M7 oraz C_M8 w rejestrze MEMO. |
| M10 | Zapala bit C_M10 w rejestrze MEMO. |
| M11 | Gasi bit C_M10 w rejestrze MEMO. |
| M30 | Zakończenie wykonywania programu. |
| M99 | Zapętlenie programu. Po napotkaniu kodu program jest wykonywany od początku. Należy zadbać o właściwe przejście między końcem a początkiem programu (np. odpowiednie uniesienie osi „Z”) gdyż nie działa tu automatyka dojazdów. |

**UWAGA !**

Kody M3-M5 M7-M11 wymagają implementacji w PLC. W domyślnym pliku PLC zdefiniowana jest obsługa M3,M4,M5 oraz M8,M9.

## *Pozostałe parametry:*

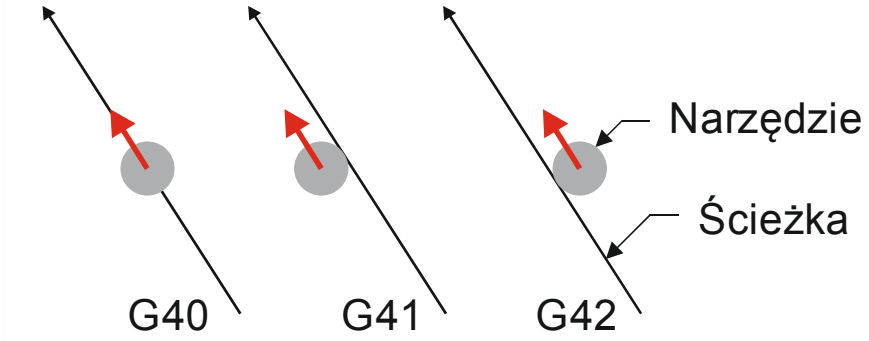
F - określa prędkość dla ruchu roboczego w mm/min

S - określa obroty wrzeciona

T – określa numer aktualnego narzędzia

## *Kompensacja promienia narzędzia G40 G41 G42*

Kompensacja G41 G42 umożliwia odsunięcie ścieżki narzędzia o podany offset (promień narzędzia). G41 włącza kompensacje lewostronną, natomiast G42 włącza kompensacje prawostronną. G40 odwołuje kompensację.

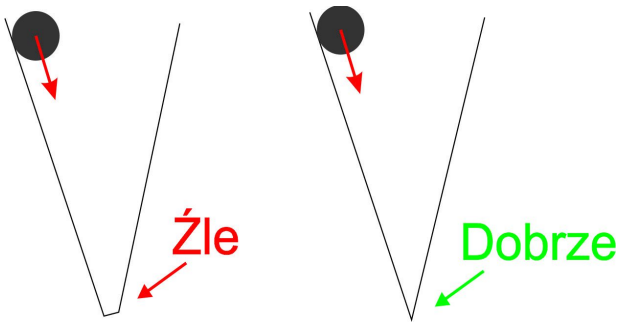



Rozmiar kompensacji pobierany jest w następujący sposób:

- Przez podanie parametru R np.
  G41 R2.0 ; włączy kompensacje z promieniem (offsetem) 2mm.
- Jeśli parametr R nie zostanie podany to rozmiar jest brany z magazynka narzędzi dla aktualnego narzędzia.
- Dla niektórych implementacji programu promień kompensacji pobierany jest z interfejsu programu np. w wersji dla plazmy.

Ścieżka przeznaczona do kompensacji musi spełniać pewne warunki.

- Nie może zawierać innych G kodów niż podstawowe G0 G1 G2 G3.
- Kompensowana ścieżka nie może zawierać gwałtownych zakrętów na wektorach krótszych niż średnica narzędzia. Generalnie kompensacja tego typu nie posiada (prawie) żadnej analizy do przodu (tak, jak jest to w Cam-ie). Jest to kompensacja „od wektora do wektora” Dlatego w wyniku działania kompensacji żaden wektor czy łuk nie powinien (teoretycznie) przestać istnieć – gdyż taki element będzie zakłócał ścieżkę kompensacji.



Kompensację włączamy przed dojazdem G0 do punktu rozpoczęcia obróbki a wyłączamy po zakończeniu wszystkich operacji w osi Z np.:

```
N070 G41 R2.0          ; Włączenie kompensacji
N080 G0 X26 Y46,35     ; Dojazd na miejsce obróbki
N090 G0 Z1
N100 G1 Z-3,2 F150     ; zejście do obróbki i praca
……..                  ; Praca w płaszczyźnie X,Y,Z G1 G2 G3
……..
N160 G0 Z2             ; Uniesienie Z
N150 G40               ; Wyłączenie kompensacji
```

Jeżeli wszystkie elementy programu wykonywane są z tą samą kompensacją wystarczy ją włączyć tylko raz na początku i wyłączyć na końcu programu.

Analiza „Do przodu” w tej kompensacji dotyczy jednego aspektu: Jeżeli wektor roboczy (G1 G2 G3) rozpoczynający obróbkę przecina się wektorem roboczym kończącym obróbkę (ostatni przed kolejnym G0 XY) to są one wzajemnie „docinane” dzięki czemu nie trzeba się martwić o prawidłowe zakończone wewnętrznych konturów.

Na ekranie graficznym normalnie wyświetlana jest ścieżka po kompensacji. Jeśli chcemy zobaczyć „oryginalną” ścieżkę to w zakładce „WIDOK/ELEMENTY” możemy wyłączyć z widoku kompensację „G41/42”. Kliknięcie na „STRAT” zawsze ponownie włącza ten element.

# Skrypty

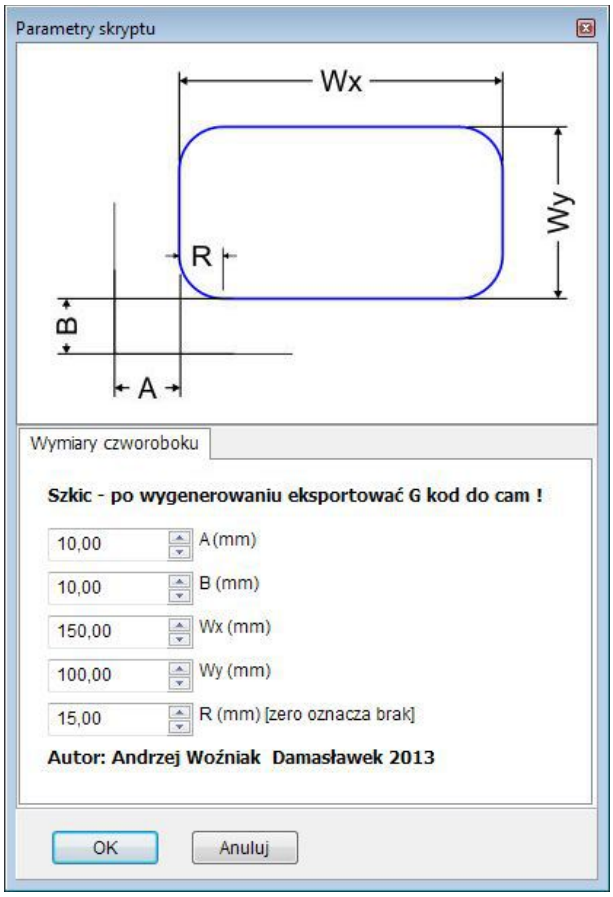
Skrypty umożliwiają generowanie programu g-kod w sposób dynamiczny i parametryczny - co jest wielką zaletą w sytuacjach gdy wykonywane elementy są bardzo różnorodne a jeszcze na tyle proste, że dają się opisać programem w skrypcie.

Skrypty mogą służyć na dwa sposoby – można nimi generować kompletną ścieżkę narzędzia gotową do natychmiastowego wykorzystania lub w drugim przypadku tylko zarys geometrii – wtedy g-kod który powstanie należy wyeksportować do cam gdzie poddaje się go dalszym operacjom. W dołączonych przykładach skrypty tego typu mają nazwę zaczynającą się słowem „szkic_*”. Po wygenerowaniu g-kodu, z menu podręcznego na liście g-kodu wybieramy opcję „Eksport G-code => CAM”. Skrypty tego typu nie muszą zawierać żadnych innych informacji niż na temat trajektorii w osi XY.

Skrypty bazują na składni języka pascal. Opis samego języka wykracza daleko poza zakres tego dokumentu, a szczegółów jego dotyczących należy szukać w sieci pod hasłami „pascal” „Delphi” „Lazarus”. Opisane będą tylko dodatkowe funkcje zaimplementowane dla potrzeb PikoCNC.

Uruchamianie okna parametrów skryptu bez otwierania okna edytora można wykonać za pomocą klawisza F8.

Przykładowy interfejs skryptu:



## *Funkcje budowania linii G-kodu:*

Podstawowe funkcje budowania linii g-kodu.

| Nazwa | Opis |
|---|---|
| **SetX** | procedure SetX(v:Extended);<br>Ustawia wartość argumentu X |
| **SetY** | procedure SetY(v:Extended);<br>Ustawia wartość argumentu Y |
| **SetZ** | procedure SetZ(v:Extended);<br>Ustawia wartość argumentu Z |
| **SetA** | procedure SetA(v:Extended);<br>Ustawia wartość argumentu A |
| **SetI** | procedure SetI(v:Extended);<br>Ustawia wartość argumentu I |
| **SetJ** | procedure SetJ(v:Extended);<br>Ustawia wartość argumentu J |
| **SetR** | procedure SetR(v:Extended);<br>Ustawia wartość argumentu R |
| **SetF** | procedure SetF(v:Extended);<br>Ustawia wartość argumentu F |
| **SetS** | procedure SetS(v:Extended);<br>Ustawia wartość argumentu S |
| **SetT** | procedure SetT(v:byte);<br>Ustawia wartość argumentu T |
| **SetP** | procedure SetP(v:Extended);<br>Ustawia wartość argumentu P |
| **SetQ** | procedure SetQ(v:Extended);<br>Ustawia wartość argumentu Q. |

| Nazwa | Opis |
|---|---|
| **SetM** | procedure SetM(v:byte);<br><br>Ustawia wartość M-kodu |
| **SetG** | procedure SetG(v:byte);<br><br>Ustawia wartość G-kodu |
| **SetComment** | procedure SetComment(txt:string);<br><br>Ustawia komentarz do aktualnej linii. |
| **Write** | procedure Write();<br><br>Instrukcja zapisu pojedynczej linii g-kody. Musi być wykonana zawsze po ustaleniu argumentów np.<br><br>`SetM(3); Write;`<br>`SetG(0); SetX(10.0); SetY(20.0); Write;`<br>`SetZ(1); Write;`<br>`SetG(1); SetZ(-1.0); SetF(1000); Write;`<br><br>Powyższy skrypt wygeneruje następujący g-kod:<br><br>`%`<br>`M3`<br>`G0 X10.0 Y20.0`<br>`G0 Z1.0`<br>`G1 Z-1.0 F1000`<br><br>Należy też pamiętać, że G-kody z zakresu G0-G3 są modalne – to znaczy, że jeśli w jakiejś linii ustawimy np. SetG(1) to G1 będzie obowiązywało do momentu, aż funkcją SetG tego nie zmienimy! Dlatego powtarzanie w każdej kolejnej linii przykładowego SetG(1) nie jest konieczne. |

## *Funkcje ułatwiające prowadzenie ścieżki*

| Nazwa | Opis |
|---|---|
| **SetBulge** | procedure SetBulge(v:Extended);<br><br>Funkcja stosowana dla G2 G3. Tak dobiera parametr „R” aby łuk który powstanie miał wysokość „V”. Przed użyciem SetBulge musi być już ustawione G2 lub G3 oraz docelowy koniec łuku (x,y). Przykład:<br><br>SetG(3); SetX(ODSTEP); SetBulge(LUK_WYS); Write; |
| **SetMove** | procedure SetMove(len,alfa:extended);<br><br>Funkcja tak ustawia X,Y aby od bieżącego punktu powstała ścieżka o długości „len” i pochylona do osi X pod wyrażonym w stopniach kątem „alfa”. Przykład:<br><br>SetG(1); SetMove(100.0, 90.0); Write; |
| **SetMoveR** | procedure SetMoveR(len,alfa:extended);<br><br>Funkcja SetMoveR jest podobna do SetMove z tą różnicą, że kąt jest liczony względem poprzedniego wektora (ale wytyczonego tą funkcją!). Wartość początkową kąta ustala się funkcją SetAngle. Kąt alfa wyrażony w stopniach. |
| **SetAngle** | procedure SetAngle(alfa:extended);<br><br>Ustala kąt początkowy dla SetMoveR. Kąt alfa wyrażony w stopniach. |
| **GetX** | function GetX():Extended;<br><br>Zwracają ostatnio zapisaną pozycję X osi. (zapisaną poleceniem „Write”). |
| **GetY** | function GetY():Extended;<br><br>j.w. dla osi Y. |
| **GetZ** | function GetZ():Extended;<br><br>j.w. dla osi Z. |
| **CirPointX** | function CirPointX(r,alfa:Extended):Extended;<br><br>Funkcja zwraca współrzędne X punktu na okręgu o promieniu „r” i kącie położenia „alfa” wyrażonego w stopniach. |
| **CirPointY** | function CirPointY(r,alfa:Extended):Extended;<br><br>Funkcja zwraca współrzędne Y punktu na okręgu o promieniu „r” i kącie położenia „alfa” wyrażonego w stopniach. |

## *Zmienne typu TPoint, TLine oraz operacje na nich*

Aby ułatwić w skryptach operacje geometryczne wbudowano w interpreter dodatkowe typy danych TPoint zawierający współrzędne punktu, oraz TLine do opisu odcinka.

*Budowa rekordu TPoint:*

```
TPoint= record
  x:extended; // współrzędna x punktu
  y:extended; // współrzędna y punktu
end;
```

*Budowa rekordu TLine:*

```
TLine= record
  s:TPoint;  // współrzędne początku odcinka (start)
  e:TPoint;  // współrzędne końca odcinka (end)
end;
```

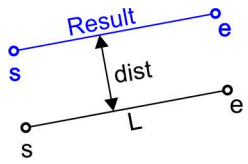
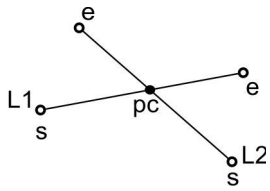
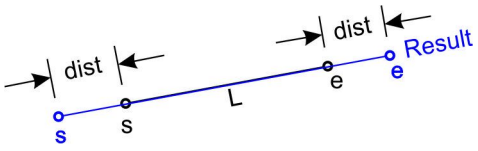
*Przykład użycia zmiennych:*

```
var ln:TLine;  pt:TPoint;              // deklarujemy zmienne
. . . .
ln.s.x:=0.0;   ln.s.y:=1.0;            // ustalamy współrzędne początku
ln.e.x:=10.0; ln.e.y:=15.0;            // ustalamy współrzędne końca
pt:=ln.s;                              // teraz punkt „pt” wskazuje początek odcinka ln

SetG(0); SetX(ln.s.x); SetY(ln.s.y); Write; // Przejazd do początku odcinka
SetG(1); SetX(ln.e.x); SetY(ln.e.y); Write; // Ruch roboczy do końca odcinka

// lub to samo prościej:
SetG(0); SetPT(ln.s); Write;           // Przejazd do początku odcinka
SetG(1); SetPT(ln.e); Write;           // Ruch roboczy do końca odcinka
```

## *Funkcje wykorzystujące zmienne typu TPoint, TLine*

| Nazwa | Opis |
|---|---|
| **SetPT** | procedure SetPT(p:TPoint);<br><br>Funkcja jest funkcjonalnym odpowiednikiem SetX, SetY w odniesieniu do zmiennej typu TPoint. Przykład:<br><br>`SetPT(p); Write;`<br><br>to inaczej:<br><br>`SetX(p.x); SetY(p.y); Write;` |
| **LinePar** | function LinePar(L:TLine; dist:extended):Tline;<br><br>Funkcja zwraca odcinek (zmienną typu TLine) odsuniętą (równoległą) od podanego odcinka „L” o offset „dist”. Jeżeli „dist” jest dodatnie offset jest lewostronny jeżeli ujemny prawostronny.<br><br>  |
| **LineInt** | function LineInt(L1,L2:TLine; var pc:TPoint):boolean;<br><br>Funkcja oblicza punkt przecięcia dwóch odcinków. Zwraca TRUE jeżeli podane w argumentach odcinki się przecinają, a wtedy zmienna „pc” zawiera punkt przecięcia się odcinków.<br><br>  |
| **LineExp** | function LineExp(L:TLine; dist:extended):TLine;<br><br>Funkcja zwraca odcinek „L” z końcami wydłużonymi o długość „dist”. Jeżeli wartość „dist” podamy ujemną to odcinek z każdej strony zostanie skrócony o tą wartość.<br><br>  |

## *Funkcje wprowadzania argumentów dla skryptu*

Jest to zestaw funkcji, dzięki którym możemy stworzyć prosty interfejs do wprowadzania parametrów skryptu.

| Nazwa | Opis |
|---|---|
| **SetArg** | procedure SetArg(num:cardinal; txt:string; v:variant);<br><br>Ustala typ i wartość początkową argumentu gdzie:<br>num – numer argumentu (maksymalnie może być 50)<br>txt – tekst objaśniający.<br>v – zmienna która jest wartością wyjściową do wpisywania, jak również jej typ mówi jakiego rodzaju ma być okienko do wpisywania wartości np:<br><br>SetArg(0,'Test',120.0); // liczba zmiennoprzecinkowa<br>SetArg(0,'Test',120); // liczba całkowita<br>SetArg(0,'Test',TRUE); // liczba boolean – zostanie utworzony checkbox<br>SetArg(0,'*Test/raz/dwa/trzy',2); // liczba całkowita – zostanie utworzony combobox<br>// z pozycjami raz dwa trzy z wybranym polem „trzy”<br>// (Musi być gwiazdka na pierwszej pozycji ! )<br>SetArg(0,'Test','Ala ma kota'); // string – zostanie utworzone pole edycji tekstu |
| **Arg** | function Arg(num:cardinal):variant;<br><br>num – numer argumentu.<br><br>Umożliwia odczyt wprowadzonych argumentów (patrz przykład). |
| **ArgEdit** | procedure ArgEdit();<br><br>Otwiera okno edycji argumentów.<br><br>**UWAGA !**<br>Jeżeli przed wywołaniem „ArgEdit” nie wywołamy żadnej z funkcji formatowania wyświetlanych argumentów (np. ArgShow) to wszystkie zdefiniowane parametry zostaną wyświetlone na jednej zakładce w kolejności nadanych przez funkcje „SetArg” numerów . |
| **SetArgPic** | procedure SetArgPic(pname:string);<br><br>Ustawia nazwę pliku obrazu, który będzie służył jako ilustracja do wprowadzanych parametrów – można go jednak pominąć. Plik musi być w tym samym katalogu co skrypt i wystarczy podać samą nazwę. |
| **ArgShow** | procedure ArgShow(num:cardinal);<br><br>Organizuje kolejność wyświetlania okienek argumentów na zakładkach np.<br><br>ArgPageTitle('Wymiary frontu'); // nazwa pierwszej zakładki<br>ArgShow(0); // dodajemy pierwszy wyświetlany parametr<br>ArgShow(3); // drugi ....<br>ArgShow(5); |

| Nazwa | Opis |
|---|---|
| | Powyższy przykład sprawi, że na pierwszej zakładce licząc od góry argumenty będą wyświetlane w kolejności 0, 3, 5.<br><br>Oczywiście najpierw należy parametry zdefiniować (SetArg) a dopiero później organizować porządek wyświetlania (patrz przykład). |
| **ArgNewPage** | procedure ArgNewPage(txt:string);<br><br>Dodanie nowej zakładki o nazwie „txt” |
| **ArgPageTitle** | procedure ArgPageTitle(txt:string);<br><br>Ustala nazwę pierwszej zakładki okna parametrów |
| **ArgSeparator** | procedure ArgSeparator(txt:string);<br><br>Dodanie na liście argumentów linii separatora o treści „txt”. np.<br><br>ArgSeparator('Autor: Andrzej Woźniak Damasławek 2016'); |

*Przykład:*

```
// nadanie argumentom nazwy, typu i wartości domyślnej
if (RUN_CNT=0) then begin                    // wykonywane tylko przy pierwszym uruchomieniu
    SetArgPic('meblowy_info.jpg');           // obrazek objaśniający
    SetArg(0,'Szerokość frontu (mm)',120.0); // Arg. nr 0 liczba zmiennoprzecinkowa
    SetArg(1,'Wysokość  frontu (mm)',60.0);  // Arg. nr 1 liczba zmiennoprzecinkowa
    SetArg(2,'Wyskość łuku (mm)',20.0);      // Arg. nr 2 liczba zmiennoprzecinkowa
    SetArg(3,'Odległość od krawędzi (mm)',10.0);
    SetArg(4,'*Górna linia/Łuk/Prosta/',0);  // Arg. nr 4 liczba całkowita - combobox
    SetArg(5,'Szybkość obróbki w osi XY (mm/min)',2000);     // Arg. nr 5 liczba całkowita
    SetArg(6,'Zaokrąglone rogi',FALSE);      // Arg. nr 6 liczba typu Boolean
 end;

// Teraz organizowanie sposobu wyświetlania parametrów.
ArgPageTitle('Wymiary frontu');      // nazwa pierwszej zakładki
ArgShow(0);                          // dodajemy pierwszy wyświetlany parametr
ArgShow(1);                          // drugi ....
ArgShow(2);
ArgSeparator('');                    // dodanie separatora – tutaj pusta linia
ArgSeparator('Autor: Andrzej Woźniak          Damasławek 2016');

ArgNewPage('Łuk 3');                 // dodanie nowej zakładki o nazwie „Łuk 3”
ArgShow(3);                          // dodajemy parametry do nowej zakładki
ArgShow(4);
ArgShow(5);

ArgNewPage('Łuk 2');                 // i kolejna zakładka o nazwie „Łuk 2”
ArgShow(6);

ArgEdit;                             // otwiera okno edycji parametrów

// I odczyt wprowadzonych argumentów
szer:=Arg(0); wys:=Arg(1); luk_wys:=Arg(2); odstep:=Arg(3);
Fxy:=Arg(5); rogi:=Arg(6);
```

## *Funkcje komunikacji z użytkownikiem*

| Nazwa | Opis |
|---|---|
| **MsgOk** | procedure MsgOk(txt:string);<br><br>Wyświetla komunikat z jednym przyciskiem „Ok”. np. MsgOk('Ala ma kota'); |
| **MsgYesNo** | function MsgYesNo(txt:string):boolean;<br><br>Wyświetla komunikat z przyciskami „Yes” „No”. Zwraca TRUE jeśli naciśniemy „Yes”. |
| **Dbg** | procedure Dbg(txt:string);<br><br>Wypisuje tekst na konsoli pod oknem edytora. |
| **Dbg_clr** | procedure Dbg_clr();<br><br>Czyści zawartość konsoli pod oknem edytora. |
| **Log** | procedure Log(txt:string);<br><br>Wypisuje tekst w oknie Log (zakładka „START” programu głównego). |
| **FloatToStr** | function FloatToStr(v:Extended):string;<br><br>Zamienia liczbę zmiennoprzecinkową na tekst. |

## *Dodatkowe funkcje matematyczne*

*function Tan(v:Extended):extended;*

*function ArcSin(v:Extended):extended;*

*function ArcCos(v:Extended):extended;*

*function ArcTan(v:Extended):extended;*

*Dodatkowo zaimplementowane funkcje matematyczne (funkcje sin() cos() są zaimplementowane „w standardzie”).*

## *Stałe*

Program pewne stałe deklaruje automatycznie:

**MAT_SIZE_X, MAT_SIZE_Y, MAT_SIZE_Z** (rozmiary materiału) kopia wartości z pól „RozmiarXYZ” zakładki „Materiał”.

**ACT_POS_X, ACT_POS_Y, ACT_POS_Z** (aktualna pozycja osi - względem materiału)

**ACT_POS_X_M, ACT_POS_Y_M, ACT_POS_Z_M** ( aktualna pozycja osi – współrzędne maszynowe)

**ACT_F, ACT_S** – są to kopie wartości wpisanych w okienka F i S programu głównego zakładka „Sterownie”

**ACT_ZRAPID, ACT_F_Z** – kopie wartości z pól „wysokość dojazdowa” i „prędkość schodzenia” z zakładki „Maszyna” programu głównego.

**TAB_SIZE_X, TAB_SIZE_Y** -wymiary obszaru roboczego z ustawień.

**RUN_CNT** - licznik uruchomień skryptu zerowany po wczytaniu i wprowadzeniu zmian w skrypcie

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