



PikoCNC skrypty

wersja 1.0

2016.06.20

Skrypty

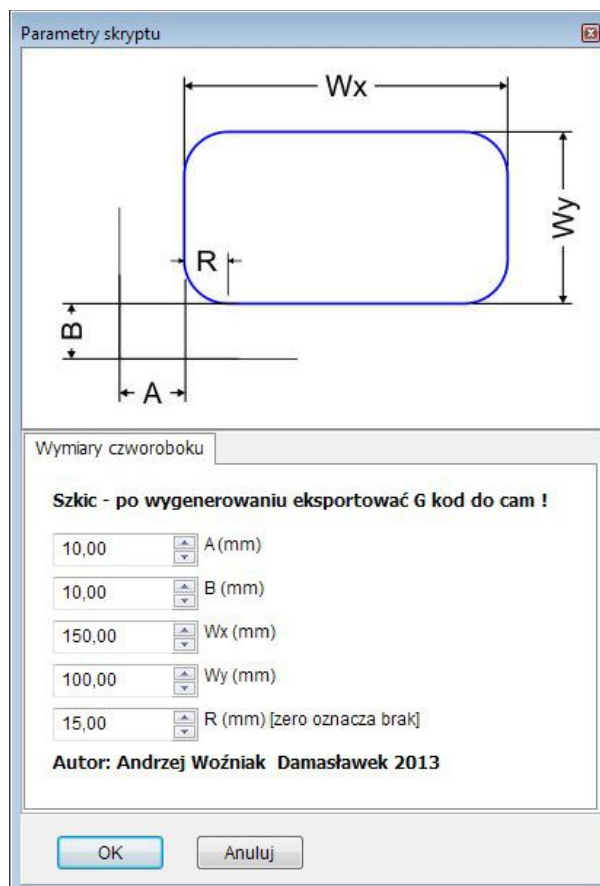
Skrypty umożliwiają generowanie programu g-kod w sposób dynamiczny i parametryczny - co jest wielką zaletą w sytuacjach gdy wykonywane elementy są bardzo różnorodne a jeszcze na tyle proste, że dają się opisać programem w skrypcie.

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

Nazwa	Opis
SetM	<code>procedure SetM(v:byte);</code> Ustawia wartość M-kodu
SetG	<code>procedure SetG(v:byte);</code> Ustawia wartość G-kodu
SetComment	<code>procedure SetComment(txt:string);</code> Ustawia komentarz do aktualnej linii.
Write	<code>procedure Write();</code> Instrukcja zapisu pojedynczej linii g-kody. Musi być wykonana zawsze po ustaleniu argumentów np. <div data-bbox="406 817 1433 952" style="border: 1px solid red; padding: 5px;"><code>SetM(3); Write; SetG(0); SetX(10.0); SetY(20.0); Write; SetZ(1); Write; SetG(1); SetZ(-1.0); SetF(1000); Write;</code></div> Powyższy skrypt wygeneruje następujący g-kod: <div data-bbox="406 1041 1433 1209" style="border: 1px solid red; padding: 5px;"><code>% M3 G0 X10.0 Y20.0 G0 Z1.0 G1 Z-1.0 F1000</code></div> 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	<pre>procedure SetBulge(v:Extended);</pre> <p>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:</p> <pre>SetG(3); SetX(ODSTEP); SetBulge(LUK_WYS); Write;</pre>
SetMove	<pre>procedure SetMove(len,alfa:extended);</pre> <p>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:</p> <pre>SetG(1); SetMove(100.0, 90.0); Write;</pre>
SetMoveR	<pre>procedure SetMoveR(len,alfa:extended);</pre> <p>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.</p>
SetAngle	<pre>procedure SetAngle(alfa:extended);</pre> <p>Ustala kąt początkowy dla SetMoveR. Kąt alfa wyrażony w stopniach.</p>
GetX	<pre>function GetX():Extended;</pre> <p>Zwracają ostatnio zapisaną pozycję X osi. (zapisaną poleceniem „Write”).</p>
GetY	<pre>function GetY():Extended;</pre> <p>j.w. dla osi Y.</p>
GetZ	<pre>function GetZ():Extended;</pre> <p>j.w. dla osi Z.</p>
CirPointX	<pre>function CirPointX(r,alfa:Extended):Extended;</pre> <p>Funkcja zwraca współrzędne X punktu na okręgu o promieniu „r” i kącie położenia „alfa” wyrażonego w stopniach.</p>
CirPointY	<pre>function CirPointY(r,alfa:Extended):Extended;</pre> <p>Funkcja zwraca współrzędne Y punktu na okręgu o promieniu „r” i kącie położenia „alfa” wyrażonego w stopniach.</p>

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;
```

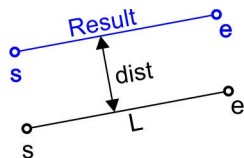
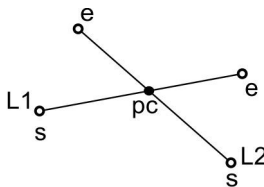
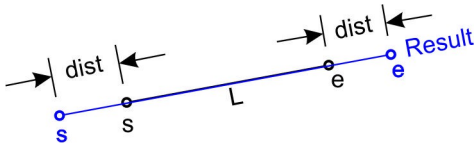
Przykład użycia zmiennych:

```
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	<p><code>procedure SetPT(p:TPoint);</code></p> <p>Funkcja jest funkcjonalnym odpowiednikiem SetX, SetY w odniesieniu do zmiennej typu TPoint. Przykład:</p> <pre style="border: 1px solid red; padding: 2px;">SetPT(p); Write;</pre> <p>to inaczej:</p> <pre style="border: 1px solid red; padding: 2px;">SetX(p.x); SetY(p.y); Write;</pre>
LinePar	<p><code>function LinePar(L:TLine; dist:extended):Tline;</code></p> <p>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.</p> 
LineInt	<p><code>function LineInt(L1,L2:TLine; var pc:TPoint):boolean;</code></p> <p>Funkcja oblicza punkt przecięcia dwóch odcinków. Zwraca TRUE jeżeli podane w argumentach odcinki się przecinają, a wtedy zmienna „pc” zawiera punkt przecięcia się odcinków.</p> 
LineExp	<p><code>function LineExp(L:TLine; dist:extended):TLine;</code></p> <p>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.</p> 

Funkcje wprowadzania argumentów dla skryptu

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

Nazwa	Opis
SetArg	<p><code>procedure SetArg(num:cardinal; txt:string; v:variant);</code></p> <p>Ustala typ i wartość początkową argumentu gdzie: num – numer argumentu (maksymalnie może być 50) txt – tekst objaśniający. v – zmienna która jest wartością wyjściową do wpisywania, jak również jej typ mówi jakiego rodzaju ma być okienko do wpisywania wartości np:</p> <pre>SetArg(0,'Test',120.0); // liczba zmiennoprzecinkowa SetArg(0,'Test',120); // liczba całkowita SetArg(0,'Test',TRUE); // liczba boolean – zostanie utworzony checkbox SetArg(0,'*Test/raz/dwa/trzy',2); // liczba całkowita – zostanie utworzony combobox // z pozycjami raz dwa trzy z wybranym polem „trzy” // (Musi być gwiazdka na pierwszej pozycji !) SetArg(0,'Test','Ala ma kota'); // string – zostanie utworzone pole edycji tekstu</pre>
Arg	<p><code>function Arg(num:cardinal):variant;</code></p> <p>num – numer argumentu.</p> <p>Umożliwia odczyt wprowadzonych argumentów (patrz przykład).</p>
ArgEdit	<p><code>procedure ArgEdit();</code></p> <p>Otwiera okno edycji argumentów.</p> <p>UWAGA ! 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 .</p>
SetArgPic	<p><code>procedure SetArgPic(pname:string);</code></p> <p>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.</p>
ArgShow	<p><code>procedure ArgShow(num:cardinal);</code></p> <p>Organizuje kolejność wyświetlania okienek argumentów na zakładkach np.</p> <pre>ArgPageTitle('Wymiary frontu'); // nazwa pierwszej zakładki ArgShow(0); // dodajemy pierwszy wyświetlany parametr ArgShow(3); // drugi ArgShow(5);</pre>

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

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

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) kopie 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

