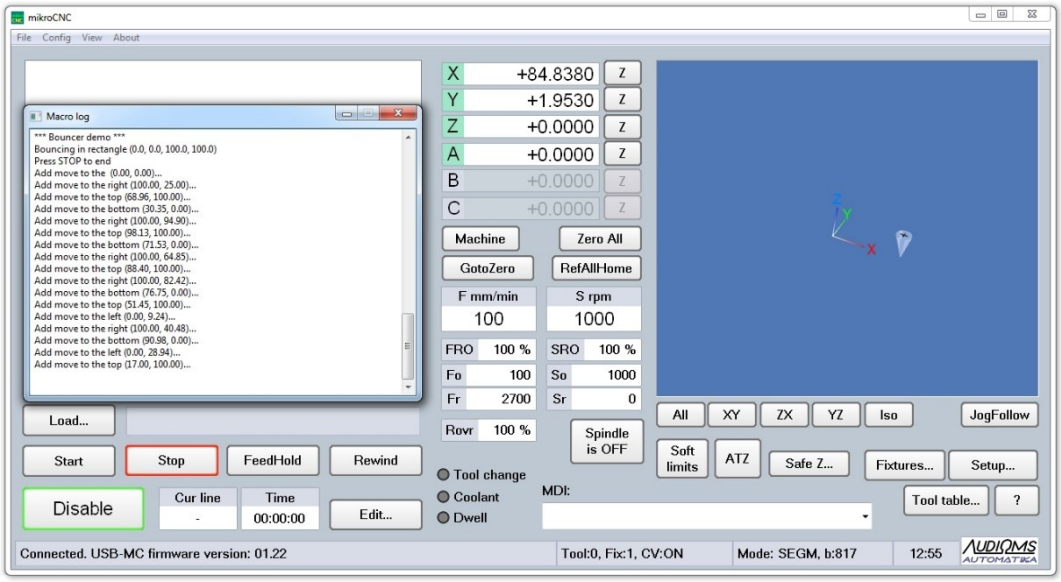
mikroCNC Scripting

Podrška za pisanje korisničkih skripti



AUDIOMS AUTOMATIKA

www.audiohms.com

SADRŽAJ

1 Python interpreter

U program mikroCNC koji se koristi za upravljanje Audioms Automatika kontrolerima kretanja je ugrađen Python 3.x interpreter skripti. Podržan je veliki deo sintakse i funkcija ovog programskog jezika.`

Skripting je podržan od verzije softvera mikroCNC V0.46.

Name	Example	Supported
If Else	`if..else..elif`	✓
Loop	`for/while/break/continue`	✓
Function	`def f(x,*args,y=1):`	✓
Subclass	`class A(B):`	✓
List	`[1, 2, 'a']`	✓
ListComp	`[i for i in range(5)]`	✓
Slice	`a[1:2], a[:2], a[1:]`	✓
Tuple	`(1, 2, 'a')`	✓
Dict	`{'a': 1, 'b': 2}`	✓
F-String	`f'value is {x}'`	✓
Unpacking	`a, b = 1, 2`	✓
Star Unpacking	`a, *b = [1, 2, 3]`	✓
Exception	`raise/try..catch..finally`	✓
Dynamic Code	`eval()/exec()`	✓
Reflection	`hasattr()/getattr()/setattr()`	✓
Import	`import/from..import`	✓
Context Block	`with <expr> as <id>:`	✓
Type Annotation	`def f(a:int, b:float=1)`	✓
Generator	`yield i`	✓
Decorator	`@cache`	✓

1.1 Podržane magic funkcije (magic methods)

Unary operators

- `__repr__`
- `__str__`
- `__hash__`
- `__len__`
- `__iter__`
- `__next__`
- `__neg__`

Logical operators

- `__eq__`
- `__lt__`
- `__le__`
- `__gt__`
- `__ge__`
- `__contains__`

Binary operators

- `__add__`
- `__radd__`
- `__sub__`
- `__rsub__`
- `__mul__`
- `__rmul__`
- `__truediv__`
- `__floordiv__`
- `__mod__`
- `__pow__`
- `__matmul__`
- `__lshift__`
- `__rshift__`
- `__and__`
- `__or__`
- `__xor__`
- `__invert__`

Indexer

- `__getitem__`
- `__setitem__`
- `__delitem__`

Specials

- `__new__`
- `__init__`
- `__call__`
- `__divmod__`
- `__enter__`
- `__exit__`
- `__name__`
- `__all__`

1.2 Builtin funkcije

`repr`	`round`	`ord`
`exit`	`isinstance`	`id`
`len`	`issubclass`	`globals`
`hex`	`callable`	`locals`
`iter`	`getattr`	`exec`
`next`	`setattr`	`eval`
`hash`	`hasattr`	`compile`
`abs`	`delattr`	`quit`
`divmod`	`chr`	

2 Podržani moduli i funkcije mikroCNC programa

Za pristup funkcijama i definisanim konstantama potrebno je uključiti odgovarajući modul putem **import** direktive. Funkcije se pozivaju upotrebom konstrukcije **modul.funkcija().**

Primer koda:

```
import mikrocnc as mcnc
import gcode

print('Loading file...')
ok = gcode.LoadFile('c:\\gcode\\myfile.tap')

if ok: mcnc.DoButton(mcnc.IDC_START)
else: print('Error opening file!')
```

2.1 modul **mikrocnc**

Funkcije:

```
IsBusy() -> bool
WaitIdle()
StatusMsg(text: str)
ErrorMsg(text: str)
Beep()
Sleep(ms: int)
GetSafeZ() -> float
IsSafeZ() -> bool
LinearMove(feed: float, x,y,z,a,b,c:float)  #for rapid, feed = -1
LinearMoveAbs(feed: float, x,y,z,a,b,c:float) #for rapid, feed = -1
LinearMove6(feed: float, p:Point6)
LinearMoveAbs6(feed: float, p:Point6)
IsMoveBusy() ->bool
GetAbsPosition(axis: int) -> float  #axis = 0-5
GetAbsPosition6() -> Point6
GetPosition(axis: int) -> float  #axis = 0-5
GetPosition6() -> Point6
GetSoftMax(axis: int) -> float  #axis = 0-5
GetSoftMin(axis: int) -> float  #axis = 0-5
GetSoftMax6() -> Point6
GetSoftMin6() -> Point6
GetToolChangeStart(axis: int) ->float #axis = 0-5
GetToolChangeStart6() -> Point6
IsHomed(axis: int) ->bool  #axis = 0-5
HomeAxis(axis: int)  #axis = 0-5
HomeCombination(mask: int)
SetCurrentTool(num: int)
ToolTableSet(tool: int, id: int, value: float)
ToolTableGet(tool: int, id: int) -> float
```

```
SpindleStartCW()
SpindleStartCCW()
GetSpindleState() -> int # STOP=0, CW=1, CCW=2
SpindleStop()
CoolantStart(type: int)
CoolantStop()
GetCoolantState() -> int
DoButton(code: int)
MessageBox(type: int, text: str) ->int
CallScript(file_name: str)
GetDRO(id: int) -> float
SetDRO(id: int, value: float)
SetLED(id: int, state: bool)
GetLED(id: int) -> bool
ZeroAxisAbs( axis_mask: int)
SetHomed(axis: int, value: bool)
ActivateESTOP(text: str)
JogOn(axis:int, dir:int, speed:int)
JogOff(axis:int)
SetJogMode(mode:int)
GetJogMode() -> int
GetMacroPath() -> str
GetKeyState(vk_code: int) -> int
SetRetData(value:str|bool|int|float)
```

2.2 modul **gcode**

```
Funkcije:

LoadFile(path_name: str) -> bool
HasFile() ->bool
GetFileName() -> str
Run()
GetRunState() -> int    # STOP=0, RUN=1, PAUSED=2
StopExec()
ExecLine(text: str)
SetFeedrate(feedrate: float)
GetVar(address: int) ->float
SetVar(address: int, val: float)
GetProbeResult() ->int
GetParameter(par: int) -> float | int
GetMin(axis: int) ->float
GetMax(axis: int) ->float
```

2.3 modul **OutputSignal**

```
Funkcije:

SetState( id: int, state: bool)
GetState( id: int) -> bool
```

2.4 modul **InputSignal**

```
Funkcije:

GetState(id: int) -> bool
GetAnalog(num: int) -> int
GetEncPos(num: int) -> int
```

2.5 modul **mcnctypes**

Modul sadrži definiciju klase **Point6**

Ova klasa ima atribute koji predstavljaju 6 koordinata tačke (x,y,z,a,b,c) kao i funkcije za neke osnovne operacije.

2.6 modul **profile**

Modul omogućava korišćenje trenutno aktivnog profila za trajno snimanje i kasnije učitavanje promenljivih tako da su ove promenljive sačuvane i dostupne pri novom pokretanju skripte, kao i pri novom pokretanju programa.

Funkcije:

```
SaveStr(name: str, value: str, glob:bool=False)
LoadStr(name: str, glob:bool=False) -> str
```

2.7 modul **gui**

Modul omogućava implementaciju korisničkog interfejsa u formi dijaloga.

Funkcije:

DialogBox(w: int, h: int, title) -> DialogBox

Klasa **DialogBox** sadrži funkcije:

```
AddButton( x,y,w,h, name, align, offx,offy, label, callback) -> bool
AddCheckBox( x,y,w,h, name, align, offx,offy, label, callback,
default_val) -> bool
```

```
AddEdit( x,y,w,h, name, align,offx,offy,ronly,default_val)
AddText( x,y,w,h, name, align,offx,offy,label) -> bool
AddRadio(self, x,y,w,h, name,align,offx,offy,label,callback, default_val)
-> bool
AddCombo(self, x,y,w,h, name,align,offx,offy,label,callback, default_val)
-> bool
DoModal() ->int
GetValue(name: str) -> item value type (int|bool|str)
SetValue(name: str, value)
EnableItem(name: str, ena: bool)
SetDefaultButton(name: str)
SetMargins(xmarg: int, ymarg: int)
GetRadioValue( name_first: str)
SetRadioValue(name_first:str, value:int)
EndDialog(code: int)
UpdateData()
Delete()
```

3 Pozivanje i čitanje argumenata makroa

Makroi se pozivaju iz MDI linije ili iz G-koda komandom Mxxx, gde je xxx broj makroa.
Pri tome se izvršava Python skript Mxxx.py

.py skripte su obični tekstualni fajlovi koje je moguće editovati nekim tekst editorom (kao notepad.exe u Windows-u ili besplatni Notepad++ koji nudi napredne opcije, syntax highlighting i sl.).

Pri pozivu M makroa podržana su 4 parametra: L, P, Q, R

Parametri koji nisu navedeni program dobija kao prazan string. Prvi argument je ime makroa, a onda slede navedena 4 parametra.

primer poziva: “M123 L1 P2 Q3 R4”

Primer programa:

```
import sys
print(‘args=’, sys.argv)
#================================
# rezultat izvršenja:

args=['M123', '1', '2', '3', '4']
```

NAPOMENA: Pri razvoju skripti iz sigurnosnih razloga preporučljivo je testirati ih prvo bez priključene mašine ili bez alata.

3.1 Napomene i trenutna ograničenja

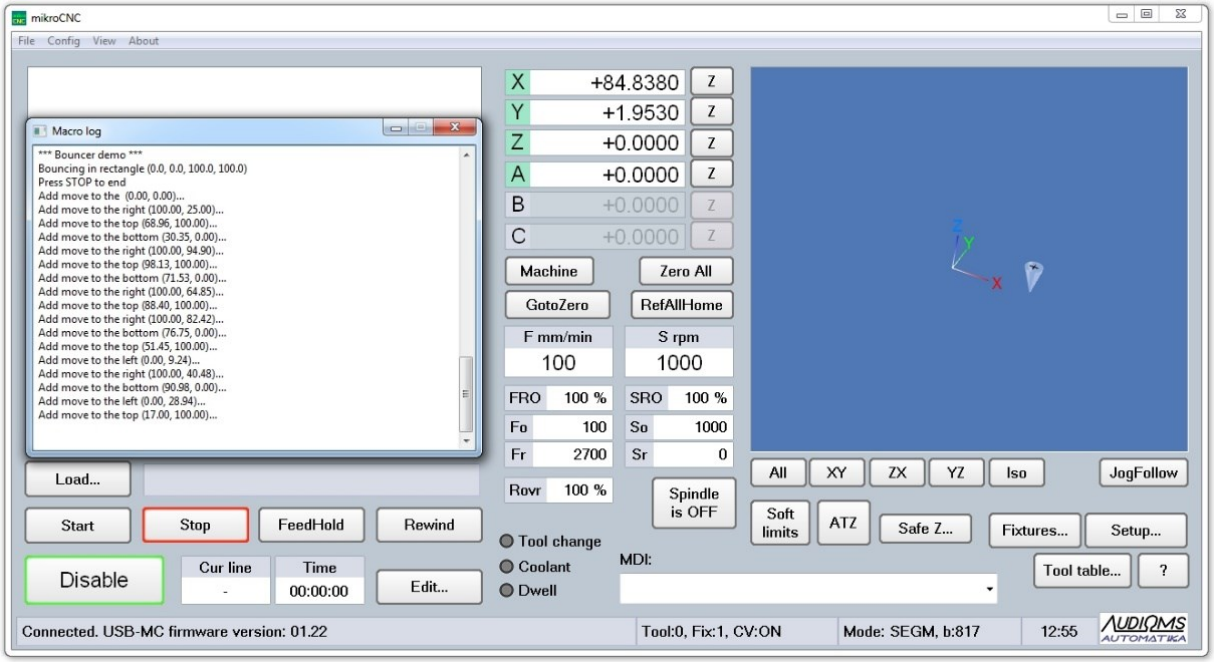
- Trenutno je u jednoj liniji G-koda moguće upotrebiti samo jedan korisnički M makro (ali moguće je u istoj liniji dodatno pozvati i standardne M komande tipa M3, M4 i sl.).

  Moguće je iz jednog skripta koristiti drugi skript tj njegove funkcije i objekte preko **import** direktive. Tada se pozvani kod izvršava u okviru iste virtuelne mašine.

  Takođe je moguće pozvati drugi skript upotrebom funkcije **mikrocnc.CallScript()** i tada se pozvani skript izvršava u zasebnoj virtuelnoj mašini.
- **gcode.ExecLine()** funkcija se izvršava u zasebnom g-kod kontekstu, tj izvršavanje komandi uglavnom ne menja stanje/parametre definisane u glavnom g-kodu.

3.2 Macro log prozor

Macro log prozor (Slika 3.1) se koristi za praćenje izlaza Python skripti kao i za informacije o eventualnim greškama pri izvršavanju skripti.
Ovaj prozor je moguće otvoriti preko opcije iz menija **Menu/View/Macro log window**.



Slika 3.1 Macro Log prozor

Preko konfiguracionog dijaloga u odeljku **UI options** moguće je podesiti opcije za automatsko otvaranje ovog prozora u slučaju greške ili pri pojavi bilo koje nove poruke.

4 Opis funkcija

4.1 Modul mikrocnc

`IsBusy() -> bool`

Funkcija služi za proveru da li je neka operacija u toku. Može se upotrebiti za sinhronizaciju kada je potrebno znati da li je mašina spremna za pokretanja novih operacija.

Povratna vrednost je **bool** tipa: **True** ili **False**.

`WaitIdle()`

Funkcija čeka da se završe sve operacije koje se trenutno izvršavaju. Mnoge funkcije samo iniciraju izvršenje određenih operacija i kontrola se odmah vraća pozivaocu. Često je potrebno da se sačeka da se pokrenute operacije potpuno završe pre zadavanja novih komandi.

Primer je recimo funkcija za izvršavanje linije G-koda **gcode.ExecLine()**.

`StatusMsg(text: str)`

Ispisuje tekstualnu poruku u statusnoj liniji mikroCNC programa.

`ErrorMsg(text: str)`

Ispisuje poruku crvenom bojom u statusnoj liniji mikroCNC programa uz zvučni signal i blinkanje.

`Beep()`

Generiše zvučni signal.

`Sleep(ms: int)`

Funkcija služi za čekanje određeni vremenski interval. Vreme se zadaje u milisekundama. Za vreme čekanja thread skripte ne troši procesorsko vreme pa se ova funkcija pored ostalog može upotrebiti recimo u petlji koja bi se izvršavala neki duži vremenski period, a nije potrebno da bude preterano brza.

`GetSafeZ() -> float`

Funkcija vraća vrednost SafeZ visine. Vrednost koja se vraća je u mašinskim koordinatama i dobijena je preračunavanjem na osnovu opcija koje je korisnik podesio u mikroCNC programu u prozoru za SafeZ podešavanja.

`IsSafeZ() -> bool`

Ovom funkcijom se proverava da li je uključena SafeZ opcija u programu.

```
LinearMove(feed, x,y,z,a,b,c) #default feed = -1 (rapid), x,y,..= None
LinearMoveAbs(feed, x,y,z,a,b,c) #version for absoulte coordinates
```

Od verzije mikroCNC 0.71 pa na dalje, svi parametri su opcioni i imaju odgovarajuće podrazumevane vrednosti. Podržano je navođenje parametara po imenu, na primer: **LinearMove**(x=50, y=100)

feed:float – feedrate brzina kretanja za potez, **podrazumevano je -1 (brzi hod)** ako nije parametar nije naveden,
x,y,z,a,b,c:float – koordinate ciljne tačke. **Podrazumevana vrednost sa svaku koordinatu je None**, i u tom slučaju neće biti promene pozicije date ose.

Funkcije služe za obavljanje kretanja po liniji od trenutne pozicije do ciljane pozicije koja se zadaje koordinatama (x,y,z,a,b,c). Kod funkcije **LnearMove()** koordinate su radne u skladu sa trenutno važećim ofsetima. Kod funkcije **LinearMoveAbs()** ciljana tačka se zadaje u mašinskim (apsolutnim) koordinatama.

Ove funkcije pokreću mehanizam za generisanje kretanja i kontrola se vraća glavnom programu skripte. Generiše se kretanje sa zaletanjem i usporavanjem u skladu sa podešavanjima u programu i potrebni podaci se šalju u bafer kretanja kontroleru za izvršenje.
Pre zadavanja nove **LinearMove()** komande neophodno je proveriti upotrebom funkcije **IsMoveBusy()** da li je program spreman da prihvati novi potez kretanja.
Dakle, ovde nije neophodno sačekati da se sve operacije završe kao kod **gcode.ExecLine()** komande.
Zato je rezultujuće kretanje, kada se sastoji iz više linearnih segmenata, bez pauza između tih segmenata.

```
LinearMove6(feed: float=-1, p:Point6)
LinearMoveAbs6(feed: float=-1, p:Point6)
```

Slično kao prethodno opisane funkcije, ali ove funkcije prihvataju **Point6** kao parametar za koordinate ciljane poziciju. Da bi bilo omogućeno korišćenje Point6 objekta potrebno je da bude uključen preko **import** direktive iz **mcnctypes** moudula na način kao što je prokazano u daljem tekstu u primeru za funkciju **GetAbsPoint6**().
Uobičajeno je koordinate p.x, p.y, p.z, itd budu brojevi, ali podržan je i tip **None** kao vrednost i u tom slučaju pozicija odgovarajuće ose će ostati nepromenjena.

```
IsMoveBusy() ->bool
```

Funkcija služi za proveru da li je moguće zadavanje novog segmenta kretanja putem komande **LinearMove()** ili **LinearMoveAbs().** Kada funkcija vrati **False** to znači da je moguće dodavanje novih podataka u bafer kretanja novim pozivom pomenutih funkcija.

GetAbsPosition(axis: int) -> float #axis = 0-5

Funkcija vraća trenutnu apsolutnu poziciju date ose (u mašinskim koordinatama).

GetAbsPosition6() -> Point6

Funkcija vraća trenutnu apsolutnu poziciju svih 6 osa u obliku class objekta **Point6** koji je definisan u modulu **mcnctypes**.

Point6 ima atribute: x, y, z, a, b, c

Da bi se koristio **Point6** tip objekta neophodno je uključiti ga preko **import** direktive.

Primer koda:

```
import mikrocnc as mcnc
from mcnctypes import Point6
p = mcnc.GetAbsPosition6()
print(p.x, p.y, p.z)
```

GetPosition(axis: int) -> float #axis = 0-5

Funkcija vraća trenutnu poziciju za datu osu u radnim koordinatama.

GetPosition6() -> Point6

Funkcija vraća trenutnu poziciju svih osa u radnim koordinatama u obliku objekta **Point6**.

GetSoftMax(axis: int) -> float #axis = 0-5

Funkcija vraća vrednost konfigurisanog maksimuma SoftLimit-a za datu osu

GetSoftMin(axis: int) -> float #axis = 0-5

Funkcija vraća vrednost konfigurisanog minimuma SoftLimit-a za datu osu

GetSoftMax6() -> Point6

Funkcija vraća vrednost konfigurisanog maksimuma SoftLimit-a za sve ose u obliku objekta **Point6**

GetSoftMin6() -> Point6

Funkcija vraća vrednost konfigurisanog minimuma SoftLimit-a za sve ose u obliku objekta **Point6**

GetToolChangeStart(axis: int) -> float #axis = 0-5

Funkcija vraća poziciju ose u radnim koordinatama za nastavak rada posle promene alata. Koristi se recimo u korisničkoj skripti za M6 komandu za automatsku izmenu alata.

GetToolChangeStart6() -> Point6

Funkcija vraća pozicije svih osa u radnim koordinatama za nastavak rada posle promene alata.

`IsHomed(axis: int) ->bool   #axis = 0-5`

Funkcija služi za proveru da li je osa referencirana.

`HomeAxis(axis: int)   #axis = 0-5`

Funkcija služi za pokretanje referenciranja jedne ose. Referenciranje se dalje odvija samostalno. Moguće je odmah pokrenuti i referenciranje drugih osa bez čekanja da se prva operacija završi (bez poziva **WaitIdle()**) ako je tako poželjno za datu mašinu.

`HomeCombination(mask: int)`

Funkcija služi za istovremeno pokretanje referenciranja više osa. Prvih šest bitova (bitovi 0-5) parametra mask određuje za koje ose se pokreće referenciranje. Bit 0 predstavlja osu X, bit 1 osu Y, itd.

`SetHomed(axis: int, value: bool) #axis = 0-5`

Funkcija se koristi za postavljanje stanja indikatora referenciranosti osa.
Efekat je isti kao kod upotrebe funkcije **SetLED**() kada se koristi identifikator neke ose.

`SetCurrentTool(num: int)`

Funkcija selektuje trenutno važeći alat. Zadaje se redni broj u tabeli alata.
Trenutno aktivni alat kao i novoseletovani alat se mogu dobiti preko funkcije **gcode.GetParameter()**

`ToolTableSet(tool: int, id: int, value: float)`

Funkcija upisuje vrednost u tabelu alata.
Parametri:

tool – redni broj alata,
id – kolona u koju se upisuje (0=Xoffset, 1=Yoffset, 2=Zoffset, 3=Diameter),
value – vrednost u mm koja se upisuje

`ToolTableGet(tool: int, id: int) -> float`

Funkcija vraća vrednost pročitanu iz tabele alata.
Parametri:

tool – redno broj alata,
id – kolona iz koje se čita (0=Xoffset, 1=Yoffset, 2=Zoffset, 3=Diameter)

`SpindleStartCW()`

Funkcija pokreće Spindle u smeru napred (u smeru kazaljki na satu). Ekvivalent komande M3.
Poštuje se zadata pauza kod pokretanja kao kod komande M3.

`SpindleStartCCW()`

Funkcija pokreće Spindle u smeru nazad (u obrnutom smeru od kazaljki na satu). Ekvivalent komande M4.
Poštuje se zadata pauza kod pokretanja kao kod komande M4.

GetSpindleState() -> int # STOP=0, CW=1, CCW=2

Vraća trenutno stanje za Spindle

SpindleStop()

Zaustavlja spindle. Ekvivalent komande M5. Poštuju se podešene pauze za zaustavljanje okretanja Spindle radnog vretena.

CoolantStart(type: int) # FLOOD=1, MIST=2, BOTH=3

Pokreće sisteme za hlađenje. Parametar **type** određuje koje funkcije se pokreću.

CoolantStop()

Zaustavlja oba sistema hlađenja (FLOOD i MIST). Ekvivalent komande M9.

GetCoolantState() -> int # NONE=0, FLOOD=1, MIST=2, BOTH=3

Funkcija vraća trenutno stanje sistema za hlađenje.

DoButton(code: int)

Funkcija služi za pozivanje funkcije datog dugmeta programa (kao da je korisnik kliknuo na dugme).

Button codes

IDC_START	1003
IDC_STOP	1004
IDC_FEEDHOLD	1005
IDC_SPINDLE	1007
IDC_RESET	1015
IDC_ZEROX	1058
IDC_ZEROY	1061
IDC_ZEROZ	1062
IDC_ZEROA	1063
IDC_ZEROB	1064
IDC_ZEROC	1065
IDC_ZEROALL	1067
IDC_REFALL	1022
IDC_ATZ	1110
IDC_SOFTLIMITS	1066
IDC_THCON	1035
IDC_THCMINSPEEDON	1042

GetDRO(id: int) -> float

Funkcija vraća vrednost pročitanu iz navedenog DRO polja.

SetDRO(id: int, value: float)

Funkcija služi za upisivanje određene vrednosti u dato DRO polje.

Definisane konstante za id DRO polja

IDC_FEEDRATE	1001
IDC_SPINDLESPEED	1002
IDC_FRO	1033
IDC_RRO	1034
IDC_SRO	1032
IDC_X	1011
IDC_Y	1012
IDC_Z	1013
IDC_A	1020
IDC_B	1026
IDC_C	1028
IDC_THCSPEED	1036
IDC_THCVNOM	1037
IDC_THCV	1039
IDC_THCMAX	1040
IDC_THCMIN	1038
IDC_THCCUR	1041
IDC_THCMINSPEED	1043
IDC_CURLINE	1018

NAPOMENA: Čitanje vrednosti DRO polja X, Y, Z, itd. uvek vraća radne koordinate čak i onda kada je uključen mod za prikaz mašinskih koordinata.
Takođe, upisivanje vrednosti u jedno od ovih DRO polja će postaviti radne koordinate za trenutnu poziciju, drugim rečima, radni ofset za datu osu će biti izmenjen.

SetLED(id: int, state: bool)

Postavlja stanje LED indikatora ili osvetljenja dugmeta sa indikatorom.
Ova funkcija takođe aktivira/deaktivira pripadajuću funkciju dugmeta, na primer:

```
SetLED(mikrocnc.IDC_SOFTLIMITS, True)
```

uključuje Soft limite.

GetLED(id: int) -> bool

Vraća trenutno stanje LED indikatora ili osvetljenja dugmeta sa indikatorom.

Definisane konstante za LED indikatore

IDC_X	1011
IDC_Y	1012
IDC_Z	1013
IDC_A	1020
IDC_B	1026
IDC_C	1028
IDC_SOFTLIMITS	1066
IDC_THCON	1035
IDC_THCMINSPEEDON	1042

NAPOMENA: IDC_X, IDC_Y, IDC_Z, itd. se mogu upotrebiti za aktiviranje/deaktiviranje indikatora referenciranosti osa uz pomoć funkcije **SetLED**().

```
ZeroAxesAbs( axis_mask: int)
```

Ova funkcija se koristi za resetovanje mašinskih koordinata za date ose. Funkcija je namenjena za upotrebu u skriptama za referenciranje da se omoguće naprednije procedure. Izmena mašinskih koordinata u toku normalnog rada se **ne preporučuje**.
NAPOMENA: Koordinate ose se resetuju na vrednosti homing ofseta koji su zadati u podešavanjima za homing.

axis_mask – mapa bitova koja definiše sve ose za koje će mašinske koordinate biti resetovane. Bit 0 treba da bude setovan za X osu, bit 1 za Y, itd.

```
ActivateESTOP(text: str)
```

Aktivira sigurnosni **ESTOP** mod uz poruku koja se prikazuje u statusnoj liniji programa. Najčešće se koristi u slučaju greške. Aktiviranje ESTOP moda povlači zaustavljanje svih operacija uključujući i sve skripte koje se trenutno izvršavaju. To znači da posle poziva ove funkcije i pozivajuća skripta završava sa radom.

```
CallScript(file_name: str)
```

Poziva izvršenje druge skripte sa zadatim imenom. Pozvana skripta se izvršava u nezavisnoj virtuelnoj mašini. Funkcija se vraća tek kada pozvana skripta završi sa radom.

```
JogOn(axis:int, dir:int, speed:int)
```

axis: osa = 0-5
dir: smer kretanja, 0=napred, 1=nazad
speed: brzina kretanja data u procentima od maksimalne brzine ose
U slučaju da je aktivan kontinualni Jog mod, ova funkcija pokreće Jog kretanje jedne ose. Kretanje je sa ubrzanjem i usporenjem u skladu sa trenutno podešenim profilom u

mikroCNC konfiguraciji. Moguće je pokrenuti simultano kretanje više različitih osa korišćenjem više poziva funkcije **JogOn()** i tako pokrenute ose se kreću nezavisno jedna od druge.
Takođe, moguće je promeniti brzinu Jog kretanja ose zadavanjem nove **JogOn()** komande za tu osu sa novom brzinom. Prelaz sa stare na novu brzinu je sa ubrzanjem/usporenjem po zadatom profilu.
U slučaju da je aktivan step jog mod (korak po korak) poziv ove funkcije izvršava jedan korak. Vrednost koraka je zadata preko mikroCNC Jog&Mpg dijaloga (Jog step [mm]).

```
JogOff(axis:int)
```

Funkcija zaustavlja Jog date ose (axis=0-5). Zaustavljanje se vrši sa usporenjem u skladu sa trenutno podešenim profilom. Funkcija se koristi samo kod kontinualnog Jog moda.

```
SetJogMode(mode:int)
```

Podešava trenutno aktivni mod rada za Jog.
mode=0 : kontinualni Jog, **JogOn()** pokreće kontinualno kretanje,
mode=1 : korak po korak, svaki poziv **JogOn()** izvršava jedan korak,
mode=2 : MPG mod.

```
GetJogMode() -> int
```

Funkcija vraća trenutno aktivni Jog mod. Za značenje povratne vrednosti pogledati prethodnu funkciju.

```
GetMacroPath() -> str
```

Funkcija vraća kompletnu putanju na disku za makro folder trenutno aktivnog profila.

```
MessageBox(type: int, text: str) ->int
```

Ova funkcija omogućava prikazivanje malog prozora sa porukom (koristi se Windows sistemska MessageBox funkcija) a povratna vrednost funkcije ukazuje na to koje je dugme korisnik pritisnuo. Moguće je odabrati razne tipove MessagBox-a, tip ikone kao i tip upita tj. dugmiće koje korisnik može da odabere.
Parametri:

type se formira upotrebom bitwise OR operacije (|) definisanih konstanti za tri parametra: tip MessageBox-a, vrsta ikone, i za default dugme:
type = MessageBox_type | MessageBox_Icon_type | Default_button
text – je tekst koji se prikazuje kao poruka korisniku
Povratna vrednost je **id** dugmeta koje je korisnik pritisnuo za izlazak.

MessageBox types (hex values)

MB_OK	0x00000000
MB_OKCANCEL	0x00000001
MB_ABORTRETRYIGNORE	0x00000002
MB_YESNOCANCEL	0x00000003
MB_YESNO	0x00000004
MB_RETRYCANCEL	0x00000005

MessageBox Icon type (hex values)

MB_ICONHAND	0x00000010
MB_ICONQUESTION	0x00000020
MB_ICONEXCLAMATION	0x00000030
MB_ICONASTERISK	0x00000040
MB_ICONWARNING	MB_ICONEXCLAMATION
MB_ICONERROR	MB_ICONHAND
MB_ICONINFORMATION	MB_ICONASTERISK
MB_ICONSTOP	MB_ICONHAND

MessageBox Default button (hex values)

MB_DEFBUTTON1	0x00000000
MB_DEFBUTTON2	0x00000100
MB_DEFBUTTON3	0x00000200
MB_DEFBUTTON4	0x00000300

Return values

IDOK	1
IDCANCEL	2
IDABORT	3
IDRETRY	4
IDIGNORE	5
IDYES	6
IDNO	7

GetKeyState(vk_code: int) -> int

Funkcija vraća status tipke na tastaturi. Implementacija interno koristi sistemsku Windows funkciju sa istim imenom. Provera statusa tipke je moguća pod uslovom da je mikroCNC trenutno aktivan program (nije minimizovan). U suprotnom funkcija vraća 0.

vk_code – sistemski kod tastera koji se proverava

Neki od kodova tastera su:
VK_F1 = 0x70
VK_F2 = 0x71
VK_F3 = 0x72
VK_F4 = 0x73
VK_F5 = 0x74
VK_F6 = 0x75
VK_F7 = 0x76
VK_F8 = 0x77
VK_F9 = 0x78
VK_F10 = 0x79
VK_F11 = 0x7A
VK_F12 = 0x7B
VK_CTRL = 0x11
VK_SHIFT = 0x10

Povratna vrednost funkcije:

- Ako je bit 15 setovan (1), tipka je pritisnuta; u suprotnom nije pritisnuta.
- Ako je bit 0 setovan (1), tipka je uključena (toggled). Tipke kao CAPS LOCK, mogu biti uključene i isključene. Tipka je isključena ako je bit 0 jednak 0. Indikator uključenosti (ako postoji) će biti upaljen ako je tipka uključena i ugašen ako je tipka isključena.

```
#primer funkcije za proveru da li je tipka pritisnuta
def IsKeyDown(vkey):
    return 0!=(mcnc.GetKeyState(vkey) & 0x8000)
```

SetRetData(value:str|bool|int|float)

Funkcija se koristi za postavljanje proširene povratne vrednosti skripte. Vrednost može biti tipa str, bool int ili float.
Kod pojedinhi funkcija (kao što je wizard za kalibraciju ose ili OnOK skripta) mikroCNC program koristi ovu povratnu vrednost skripte za dalje procesiranje.

4.2 Modul gcode

Funkcije:

`LoadFile(path_name: str) -> bool`
Funkcija učitava G-kod fajl sa datim imenom. Ako je učitavanje uspešno povratna vrednost je **True**, u suprotnom je **False**.

`HasFile() ->bool`
Funkcija vraća **True** ako postoji učitan G-kod fajl, u suprotnom vraća **False**.

`GetRunState() -> int # STOP=0, RUN=1, PAUSED=2`
Funkcija vraća trenutno stanje izvršavanja učitanog G-kod programa.

`GetFileName() -> str`
Vraća ime i potpunu stazu trenutno učitanog G-kod fajla ako postoji. U suprotnom vraća prazan string.

`Run()`
Funkcija pokreće izvršavanje učitanog G-kod programa kao kod pritiska na dugme START.

`StopExec()`
Funkcija zaustavlja izvršavanje G-kod programa. Dalje izvršavanje G-koda će biti zaustavljeno kada linija koja se trenutno izvršava potpuno završi zadate operacije.

`ExecLine(text: str)`
Funkcija izvršava liniju G-koda zadatu u obliku stringa **text**. String **text** može da sadrži sve podržane G i M komande.

Ova funkcija samo pokreće mehanizam izvršavanja operacija i kontrola se vraća pozivaocu odnosno glavnom programu skripte onda kada su sve komande poslate na izvršenje. To ne mora nužno da znači da su sve operacije potpuno i obavljene.
Pre zadavanja nove komande **ExecLine()**, da bi se obezbedilo da su prethodno zadate operacije sve i završene, potrebno je upotrebiti funkciju **mikrocnc.WaitIdle()**
Ako data linija G-koda sadrži poziv neke druge korisničke skripte, pri njenom izvršenju čeka se da ona završi sa radom pre procesiranja ostalih komandi u liniji.

`SetFeedrate(feedrate: float)`
Funkcija se koristi za podešavanje feedrate parametra.

`GetVar(address: int) ->float`
Funkcija se koristi za čitanje vrednost date G-kod varijable. G-kod varijable se u G-kod programu označavaju sa #adresa_varijable

```
SetVar(address: int, val: float)
```
Funkcija se koristi za dodeljivanje vrednosti navedenoj G-kod varijabli. Vrednost je tipa **float**

```
GetProbeResult() ->int     #see defined constants for return value
```
Funkcija se koristi za očitavanje rezultata/stanja prethodno pokrenute Probe operacije. U slučaju da je Probing uspešno završen (povratna vrednost PROBE_HIT) G-kod promenljive **#2000-#2005** sadrže koordinate tačke kontakta. Pri tome su promenljive:

- 2000 -> vrednost X koordinate,
- 2001 -> vrednost Y koordinate,
- 2002 -> vrednost Z koordinate,
- 2003 -> vrednost A koordinate i
- 2004 i 2005 nije podržano.

Povratne vrednosti za GetProbrResult()

PROBE_IDLE = 0	Nije bilo Probing operacije
PROBE_ACTIVE = 1	Probing je u toku
PROBE_NOHIT = 2	Probing je neuspešno završen bez kontakta
PROBE_HIT = 3	Probing je uspešno završen

```
GetParameter(par: int) -> float | int      #see defined constants
```

Definisane konstante za GetParameter()

SPINDLE_SPEED	0
FEEDRATE	1
CIRC_MODE	2
WORK_PLANE	3
CV_MODE	4
IJK_ABS	5
TOOL_OFFSET_NDX	6
TOOL_TABLE_NDX	7
METRIC_UNITS	8
RELATIVE_MODE	9
CURRENT_TOOL	10
SELECTED_TOOL	11

```
GetMin(ax: int) ->float
GetMax(ax: int) ->float
```
Funkcije se koriste za dobijanje gabarita učitanog G-koda. Funkcija **GetMin(ax)** vraća najmanju koordinatu učitane mreže po osi zadatoj preko parametra **ax**.

Slično tome, funkcija **GetMax(ax)** vraća maksimalnu koordinatu učitane mreže za datu osu.

ax – broj ose: 0 = x, 1 = y, 2 = z

4.3 Modul OutputSignal

Funkcije:

```
SetState( id: int, bool state)
```

Funkcija služi za postavljanje stanja izlaznog signala. Zadaje se identifikator signala (videti tabelu) i željeno novo stanje, aktivno ili ne (True/False).

Napomena: Da bi se stanje signala prenosilo na neki izlazni pin kontrolera kretanja potrebno je u konfiguraciji (Config/Setup/Output signals) uključiti dati signal i podesiti željeni izlazni pin, a po potrebi i druge opcije.

```
GetState( id: int) -> bool
```

Funkcija služi za dobijanje trenutnog stanja izlaznog signala. Parametar **id** je identifikator signala (videti tabelu).

Definisane konstante za id izlaznih signala

EXT1	0
EXT2	1
EXT3	2
EXT4	3
EXT5	4
EXT6	5
ENABLE	6
OUTPUT1	9
OUTPUT2	10
...	
OUTPUT10	19

4.4 Modul InputSignal

Funkcije:

```
GetState(id: int) -> bool
```

Funkcija vraća trenutno stanje datog ulaznog signala. Parametar **id** je identifikator signala (videti tabelu). Povratna vrednost je True ili False (aktivan ili neaktivan).

Napomena: Da bi stanje signala odražavalo stanje nekog ulaznog pina kontrolera kretanja potrebno je u konfiguraciji (Config/Setup/Input signals) uključiti dati signal i podesiti željeni ulazni pin, a po potrebi i druge opcije.

GetAnalog(num: int) -> int

Funkcija vraća trenutnu vrednost napona na datom analognom ulazu dobijenu preko A/D konvertora. Vrednost koja se dobija je 16-bitna nepredznačena, što znači da 65535 odgovara maksimalno mogućoj vrednosti napona (5V).

Definisane konstante za id ulaznih signala:

HOMEX	0
LIMITXP	1
LIMITXM	2
HOMEY	3
LIMITYP	4
LIMITYM	5
HOMEZ	6
LIMITZP	7
LIMITZM	8
HOMEA	9
LIMITAP	10
LIMITAM	11
HOMEB	12
LIMITBP	13
LIMITBM	14

HOMEC	15
LIMITCP	16
LIMITCM	17
ESTOP	18
INDEX	19
PROBE	20
THC_ARCOK	21
THC_UP	22
THC_DOWN	23
INPUT1	24
INPUT2	25
...	...
INPUT10	34

GetEncPos(num: int) -> int

Funkcija vraća trenutnu vrednost brojača datog enkodera. Vrednost je 32-bitni integer. Parametar **num** je pozicija enkodera/MPG-a u tabeli podešavanja u mikroCNC programu, počevši od 0.

Napomena: **Period osvežavanja stanja svih ulaza uključujući i analogne ulaze i enkodere je 50ms.**

4.5 Modul mcnctypes

Modul sadrži definiciju klase **Point6** koja ima atribute **x, y, z, a, b, c** koji se mogu smatrati koordinatama tačke (za svih 6 osa) ili 6 komponenti vektora.
Klasa takođe sadrži sledeće funkcije:

__init__(self, x=0,y=0,z=0,a=0,b=0,c=0)

Konstruktor klase za inicijalizaciju koordinata.

__add__(self, p: Point6) ->Point6
Operator sabiranja omogućava vektorsko sabiranje dva **Point6** objekta.

__sub__(self, p: Point6) ->Point6
Operator oduzimanja omogućava vektorsko oduzimanje **Point6** objekata.

__str__(self) -> str
Operator pretvaranja u string daje sve koordinate u obliku printabilnog stringa.

__mul__(self, a) -> Point6
Operator množenja omogućava množenje vektora **Point6** skalarom **a**.

__div__(self, a) -> Point6
Operator deljenja omogućava deljenje komponenti vektora **Point6** skalarom **a**.

Res(self) -> float
Vraća rezultantu vektora **Point6**.

Distance(self, p:Point6) -> float
Vraća rastojanje između dve **Point6** tačke.

4.6 Modul profile

SaveStr(name: str, value: str, glob:bool=False)

Funkcija snima vrednost string promenljive u trenutno aktivni .INI profil. Snimanje u fajl profila se vrši u obliku **name = value.** Ako parametar **glob** nije naveden ili ima vrednost **False** snimanje se vrši u posebnu grupu podataka koja je kreirana za aktivnu skriptu. Druge skripte nemaju pristup ovim podacima već svaka skripta može da manipuliše samo sa svojim snimljenim podacima.
Ako **glob** parametar ima vrednost **True** onda se snimanje promenljive vrši u globalnu sekciju tako da i ostale skripte imaju pristup za čitanje i modifikovanje podataka.

name – ime promenljive pod kojim će biti snimljena vrednost. Ime je string koji može da sadrži slova, brojeve i donju crtu '_'
value – vrednost koja se snima (string)
glob – opcioni paramtear koji određuje da li će se podaci snimiti u privatnu sekciju za skriptu (glob=False) ili u globalnu sekciju tako da i ostale skripte imaju pristup ovim podacima.

```
LoadStr(name: str, glob:bool=False) -> str
```

Funkcija vraća vrednost promenljive iz trenutno aktivnog profila.

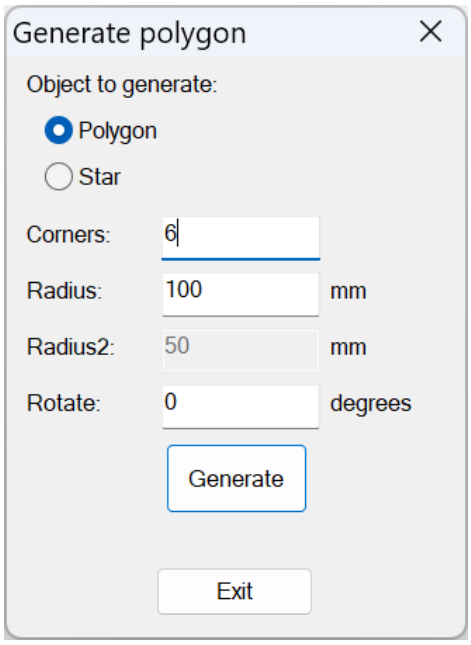
name – ime pod kojim je vrednost snimljena
glob – opcioni fleg koji pokazuje da li se vrednost učitava iz privatne sekcije podataka za ovu skriptu (glob=False) ili iz globalne sekcije profila (glob=True)

4.7 Modul gui

Modul **gui** omogućava implementaciju korisničkog interfejsa u formi dijaloga sa osnovnim tipovima elemenata za interakciju sa korisnikom – windows kontrola (Slika 4.1).

Trenutno podržani elementi koji mogu biti prikazani u dijalogu su:

- edit polje za unos
- dugme
- radio dugme
- check box
- combo box
- tekst



Slika 4.1 Primer dijaloga kreiranog iz skripte (primer M116)

Realizacija **gui** modula je koncipirana tako da se omogući što jednostavnija primena i sa što manje koda. U tu svrhu je obezbeđeno da većina funkcija ima definisane podrazumevane vrednosti parametara tako da se mogu navoditi samo potrebni parametri (navođenje argumenata po imenu, a ne po redosledu).

Takođe, najčešće nije neophodno navoditi apsolutne koordinate i dimenzije elemenata jer je obezbeđeno automatsko pozicioniranje elementa na dijalogu uz upotrebu dostupnih metoda za poravnanje (align), pomeranja (ofsete), kao i podešavanje margina dijaloga.

Objekat (klasa) **DialogBox** se kreira na sledeći način:

```
d = gui.DialogBox(w:int,h:int,title:str)
#promenljiva d sada predstavlja DialogBox objekat
```

Parametri (svi su opcioni):

- **w** – širina radne oblast dijaloga (ne obuhvata okvir i naslov),
- **h** – visina radne oblasti dijaloga (ne obuhvata okvir i naslov),
- **title** – naslov za dijalog

Ovde se Širina i visina (kao i koordinate i sve dimenzije u daljem tesku) navode u **DLU** jedinicama (Dialog Logical Units) kao što je uobičajeno kod Windows Dialog resursa (nisu pikseli već se ove jedinice izvode iz veličine fonta za dijalog).
DLU jedinice su usvojene da bi kreirani dijalog izgledao ispravno i približno isto na svakom sistemu.
Takođe, bitno je napomenuti da su kod ove funkcije w i h, širina i visina **radne oblasti (client area)** dijaloga. To olakšava planiranje pozicija elemenata jer se ne mora uzimati u obzir širina okvira i naslova koji mogu varirati od sistema do sistema.

Funkcija **DialogBox**() vraća objekat preko koga se vrši dalje kreiranje sadržaja dijaloga. Dijalog u ovom trenutku još uvek nije vidljiv na ekranu i postaće vidljiv tek kada se završi sva inicijalizacija i pozove funkcija **DoModal**().

class DialogBox objekat sadrži sledeće funkcije:

Funkcije za kreiranje elemenata (Add*) očekuju koordinate i dimenzije u Windows DLU jedinicama (Dialog Logical Unit).
Svi tipovi elemenata imaju **name** (string) identifikator za kasnije referenciranje (ako je potrebno), **align** (int) vrednost koja određuje način poravnavanja, **offx** i **offy** (int) ofsete za eventualno dodatno relativno pomeranje u odnosu na prethodni element. Većina elemenata ima i **label** string kojim se navodi tekst u tom elementu.

```
AddButton( x:int, y:int, w:int, h:int, name:str, align:int, offx:int, offy:int, label:str, callback:str) -> bool
```

Funkcija kreira dugme. **Svi parametri su opcioni**, a parametri su:

- **x,y** – koordinate gornjeg levog ugla
- **w,h** – širina i visina dugmeta
- **name** – identifikator, neophodan ako je potrebno kasnije referenciranje
  Postoje specijalne predefinisane vrednosti identifikatora za **OK** i **Cancel** dugme, a to su ‘IDOK’ i ‘IDCANCEL’ respektivno. Ako se koriste ovi identifikatori postoji

automatski mehanizam zatvaranja dijaloga uz povratni kod tako da nije potrebno obezbediti callback funkciju koja bi ovo obavljala.

- **align** – vrsta poravnavanja:
  - **gui.LEFT** – horiontalno poravnanje uz levu ivicu prozora (podrazumevano)
  - **gui.CENTER** – horizontalo centriranje po širini prozora
  - **gui.RIGHT** – poravnanje uz desnu ivicu prozora
  - **gui.LEFTPREV** – poravnanje u odnosu na prethodno kreirani element

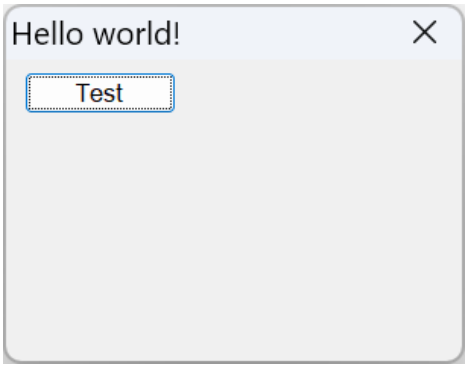
  Prve tri vrste poravnanja impliciraju da se element kreira u novom redu (ispod prethodnog elementa). Jedino poslednja vrednost (gui.LEFTPREV) locira element u produžetku istog reda tj pored prethodnog elementa sa njegove desne strane.
  Napomena: poravnanje tipa **gui**.**LEFTPREV** povezuje dati element sa prethodnim tako da ti elementi (dva ili više uzastopnih) čine grupu. Ova cela grupa elemenata se na dijalogu poravnava onako kako je navedeno za prvi element grupe (za koji poravnanje nije tipa LEFTPREV).
  Na ovaj način je moguće recimo centrirati horizontalno više dugmića (čest primer su OK i Cancel koji su centrirani u dnu dijaloga).
- **offx, offy** – ofseti relativnog pomeranja u odnosu na prethodni element. Ako je data apsolutna koordinata x ili y, ofset za tu osu se ignoriše. Podrazumevane vrednosti su 0.
- **label** – tekst koji će pisati na dugmetu
- **callback** – ime funkcije koja će biti pozvana kada se klikne na dugme (ako je to potrebno).

Primer kreiranja dijaloga i jednog dugmeta sa callback funkcijom (Slika 4.2):

```
import gui

def on_click():
    print('Button clicked!')

d = gui.DialogBox(150,100,'Hello world!') #create new DialogBox
d.AddButton(name='button1', label='Test', w=50, callback='on_click')
#pošto nisu navedene koordinate, dugme će biti kreirano u gornjem levom uglu
d.DoModal() #prikaži dijalog
```



Slika 4.2

Pritisak na dugme će izazvati pozivanje funkcije **on_click()** i ispisivanje poruke u macro log prozoru.

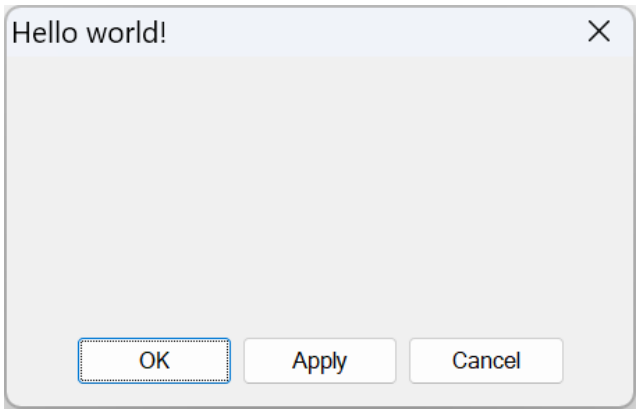
Kod dodavanja svakog sledećeg elementa dijaloga, ako bi se izostavile koordinate *x* i *y* i način poravnavanja (align), novi element bi bio kreiran u sledećem redu, (ovde ispod dugmeta) i uz levu ivicu prozora.

Znači podrazumevano pozicioniranje je: sledeći red i levo poravnanje.

Primer kreiranja grupe centriranih elemenata (u ovom slučaju tri dugmeta) na način kako je objašnjeno u napomeni uz **align** parametar (Slika 4.3).

```
import gui

d = gui.DialogBox(200,100,'Hello world!') #kreiraj novi DialogBox
#prvi element grupe određuje način poravnanja grupe u prozoru
d.AddButton(y=80, name='IDOK', label='OK', w=50, align=gui.CENTER)
#slede ostali elementi grupe (poravnanje tipa LEFTPREV)
d.AddButton(name='apply', label='Apply', w=50, align=gui.LEFTPREV)
d.AddButton(name='IDCANCEL', label='Cancel', w=50, align=gui.LEFTPREV)
d.DoModal() #prikaži dijalog
```



Slika 4.3 Primer centrirane glupe elemenata

AddCheckBox(x:int, y:int, w:int, h:int, name:str, align:int, offx:int, offy:int, label:str, callback:str, default_val:bool) -> bool

Funkcija kreira ckeck box. Svi parametri su opcioni.
Pored do sada opisanih parametara (videti funkciju **AddButton**) moguće je naversti i default vrednost:

- **label** – tekst koji će pisati pored kvardatića
- **default_val** – određuje da li će opcija biti uključena (True) ili isključena (False) pri pojavljivanju dijaloga (podrazumevano je False).

AddEdit(x:int, y:int, w:int, h:int, name:str, align:int, offx:int, offy:int, ronly:bool, default_val:str) -> bool

Funkcija kreira polje za editovanje. Svi parametri su opcioni. Pored opisanih parametara dustipni su i:

- **ronly** – ako je **True** kreira se edit polje read only tipa (podrazumevano je False)
- **default_val** – inicijalna vrednost (string) koja se prikazuje u polju

Edit element podržava samo string vrednosti. Drugi tipovi promenljivih (int, float) se moraju konvertovati u string recimo preko Python funkcije **str**(vrednost).

AddText(x:int, y:int, w:int, h:int, name:str, align:int, offx:int, offy:int, label:str) -> bool

Funkcija kreira tekst.

- **label** - tekst

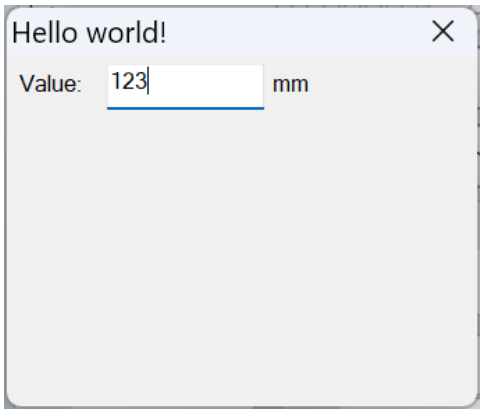
Primer kreiranja edit polja sa tekstom za opis i za jedinicu (Slika 4.4):

```
import gui

#create new DialogBox
d = gui.DialogBox(150,100,'Hello world!')
#kreiraj tekst za opis
d.AddText(label='Value:')
#kreiraj edit polje u produžetku teksta
d.AddEdit(name='edit1', default_val='123', align = gui.LEFTPREV)
#kreiraj tekst za jedinicu u produžetku
d.AddText(label='mm', align=gui.LEFTPREV)

ret=d.DoModal() #prikaži dijalog

if(ret==1): #OK/Enter pritisnuto
 print('Vrednost polja je', d.GetValue('edit1'))
```



Slika 4.4

```
AddRadio( x:int, y:int, w:int, h:int, name:str, align:int, offx:int,
offy:int, label:str, callback:str, default_val:bool) -> bool
```

Funkcija kreira radio dugme.

- **default_val** – inicijalno stanje, selektovano ili ne (podrazumevano je False)

Više uzastopno kreiranih radio elemenata čine grupu tako da kada korisnik selektuje jedno dugme ostala se deselektuju.
Provera stanja pojedinačnih elemenata je moguća preko funkcije **GetValue**(). Takođe je moguće uz pomoć funkcije **GetRadioValue**() dobiti indeks selektovanog elementa radio grupe.

```
AddCombo( x:int, y:int, w:int, h:int, name:str, align:int, offx:int,
offy:int, label:str, callback:str, default_val:int) -> bool
```

Funkcija kreira combo box koji je tipa drop-down liste.

- **label** – parametar je string koji sadrži elemente drop-down liste odvojene sa '\n' kao delimiterom. Na primer: string '**jedan**\n**dva**\n**tri**' navodi tri elementa: 'jedan', 'dva' i 'tri'
- **callback** – opciono ime funkcije koja će biti pozvana kada se promeni selektovani element liste
- **default_va**l – opciono inicijalno selektovani element liste počevši od 0

```
DoModal() ->int
```

Posle kreiranja svih elemenata dijaloga i eventualno pozivanja ostalih funkcija za formatiranje i sl., na kraju je potrebno pozvati funkciju **DoModal**() kako bi pripremljeni dijalog prozor bio prokazan na ekranu.
Ova funkcija obavlja i tzv **modal loop** tj vrši servisiranje svih kontrola dijaloga i ne vraća se dok korisnik ne zatvori dijalog. To recimo može biti klikom na **OK** ili **Cancel** dugme ili na drugi način predvideti da se pozove funkcija **EndDialog**()

Funkcija po zatvaranju dijaloga vraća povratni kod koji ukazuje na koji način je dijalog zatvoren.

-1- greška pri kreiranju dijaloga
0 - function failed
1 - OK dugme
2 - Cancel dugme
3 - došlo je do greške

Povratni kod može biti i neka druga vrednost koja je prosleđena preko funkcije **EndDialog**()

`GetValue(name:str) -> bool/int/str`

Funkcija vraća vrednost povezanu sa datim elementom dijaloga.
name – identifikator elementa

Tip povratne vrednosti zavisi od tipa elementa:

- **Edit polje** -> str, string upisan u polju. Ovaj string je moguće pretvoriti po potrebi u odgovarajući tip (**int**, **float**) upotrebom Python funkcija **int**(str), **float**(str)
- **Check box** -> bool, True ili False u zavisnosti da li je polje čekirano ili ne
- **Radio dugme** -> bool, True ili False u zavisnosti da li je selektovano ili ne
- **Combo box** -> int, indeks odabrane opcije počevši od 0

`SetValue(name:str,value:bool/int/str)`

Funkcija podešava vrednost za dati element.
name – identifikator elementa
value – tip vrednosti zavisi od tipa elementa (pogledati prethodnu funkciju)

`EnableItem(name:str,ena:bool)`

Funkcija podešava stanje datog elementa u dozvoljeno ili zabranjeno (enable/disable)
name – identifikator elementa
ena – True ili False

`SetDefaultButton(name:str)`

Funkcija podešava koje je podrazumevano (default) dugme dijaloga. Kada se pritisne ENTER ovo dugme se aktivira.

name – identifikator dugmeta

`SetMargins( xmarg:int, ymarg:int)`

Funkcija podešava margine radne površine dijaloga za x osu (levo i desno) kao i za y osu (gore i dole). Oba parametra su opciona i navode se u DLU jedinicama.

Ove margine se koriste pri automatskom pozicioniranju elemenata na radnoj površini dijaloga.

Kada je data apsolutna koordinata elementa (x ili y) margine za tu osu se ne primenjuju.

`GetRadioValue(name_first:str)`

Funkcija vraća indeks selektovane opcije u grupi radio elemenata.

name_first – identifikator prvog elementa u grupi radio elemenata

`SetRadioValue(name_first:str, value:int)`

Funkcija se koristi sa selektovanje jedne opcije iz grupe radio elemenata. Ostali elementi grupe se deselektuju.

name_first – identifikator prvog elementa grupe radio elemenata
`value` – redni broj opcije za selektovanje počevši od 0

`EndDialog(code:int)`

Funkcija izaziva zatvaranje dijaloga uz dati povratni kod. Skripta dobija ovaj kod kao povratnu vrednost funkcije **DoModal**()
Tipično je da se **EndDialog**() funkcija poziva recimo iz callback funkcije kada se klikne na neko dugme.
Kao što je već napomenuto, dugme **OK** i **Cancel** ne zahtevaju callback funkciju i pozivanje **EndDialog**() ako se kao identifikatori koriste '**IDOK**' i '**IDCANCEL**'.

`UpdateData()`

Funkcija služi da se pročitaju vrednosti svih elementa windows dijaloga i zapamte u internoj memoriji objekata pre nego što dođe do zatvaranja tog prozora. Ovo mora da se uradi pre poziva funkcije **EndDialog**(). Kada je u pitanju OK dugme, ova funkcija se automatski poziva.

`Delete()`

Funkcija služi za eksplicitno oslobađanje memorije dijaloga kada se završi rad sa dijalogom.
Svi resursi se oslobađaju kada skripta završi sa radom, tako da ako skripta otvara jedan (ili manji broj) dijaloga, pozivanje **Delete**() funkcije je opciono.
Posle pozivanja **Delete**() funkcije taj dijalog objekat više nije moguće koristiti.

5 Skripte specijalne namene

U programu **mikroCNC** je za više funkcija moguće podesiti da se koriste korisničke skripte koje su prilagođene za konkretnu namenu. Moguće je predefinisati funkcije nekoliko dugmića tako da se umesto ugrađene funkcionalnosti programa vrši pozivanje i izvršavanja korisničke skripte.

Pored toga, za još nekoliko funkcija je moguće podesiti da se pozivaju korisničke skripte.

Specijalne skripte koje su trenutno dostupne za prilagođavanje:

- **M6** skripta se koristi za automatsku izmenu alata i poziva se iz G-koda kada je u podešavanjima selektovana treća opcija za Automatic Tool Changer,
- **AutoToolZero** skripta se može pozivati pritiskom na dugme **ATZ** za automatsko umeravanja alata,
- **RefAllHome** skripta se može pozivati pritiskom na dugme **RefAllHome** za referenciranje mašine,
- **M1030** skripta se može pozivati na kraju izvršenja G-code programa posle izvršenja komande **M30** i
- **BackgroundTask** skripta se može postaviti za kontinualno izvršavanje u pozadini.

6 Primeri skripti

Uz program **mikroCNC** stižu i primeri skripti koji demonstriraju različite funkcije. Ovi primeri se nalaze u folderu **macro_examples**

M100

Primer upotrebe funkcije **gcode.ExecLine()** za spiralno kretanje po kvadratu čija se osnovica smanjuje.

M101

Primer upotrebe funkcije **mikrocnc.LinearMove()** za kretanje napred-nazad uz promenu brzine kretanja (feedrate)

M102

Primer upotrebe funkcije **mikrocnc.LinearMove()** za kretanje unutar datog kvadratnog okvira uz odbijanje od zidova (ivica) kvadrata.

M103

Primer upotrebe **mikrocnc.MessageBox()** funkcije za prikaz informacija i dobijanje povratne informacije od korisnika.

M104

Primer upotrebe **fileio** modula za pisanje i čitanje datoteka uz pomoć funkcija **File** klase **fileio.open()**, **File.writeline()**, **File.readline()**

M105

Primer kontrole Spindle radnog vretena uz upotrebu funkcija **mikrocnc.SpindleStartCW()**, **mikrocnc.SpindleStartCCW(), mikrocnc.SpindleStop()**

M106

Primer upotrebe funkcije **gcode.LoadFile()** za učitavanje g-koda i strartovanje preko **mikrocnc.DoButton()**

M107

Test rekurzivnog pozivanja skripte. Pored upotrebe funkcije **ExecLine()** za pokretanje nove instance skripte, demonstrira se i prenošenje argumenata skripti i čitanje preko **sys** modula

M108

Primer provere da li je obavljeno referenciranje osa mašine, prikaz statusa putem **mikrocnc.MessageBox()** i generisanje greške **RuntimeError**() po potrebi.

M109

Programabilno pokretanje Jog kretanja osa uz upotrebu funkcija **mikrocnc.JogOn()**, zaustavljanje preko **mikrocnc.JogOff()** i promenu brzine kretanja u toku kretanja.

M110

Primer manuelne kontrole Jog kretanja preko eksternih signala koji se čitaju preko funkcije **InputSignal.GetState()**

M111

Test čitanja enkodera upotrebom funkcije **InputSignal.GetEncPos()**

M112

Primer pokretanja referenciranja mašine, prvo samostalno ose Z uz upotrebu funkcije **mikrocnc.HomeAxis()**, a zatim i ostalih osa simultano uz upotrebu funkcije **mikrocnc.HomeCombination()**

M113

Skripta generiše g-kod fajl sa mrežom oblika 3d talasa, a zatim se ovaj g-kod automatski učitava putem funkcije **mikrocnc.LoadFile()**

M114

Skripta demonstrira upotrebu modula **profile** i snimanje i učitavanje promenljivih u trenutni .INI profil upotrebom funkcija **profile.SaveStr()** i **profileLoadStr()**

M115

Primer koji demonstrira upotrebu funkcije **os.PlaySound()** za puštanje zvučnog signala .WAV fajla

M116

Primer upotrebe modula **gui** za kreiranje korisničkog interfejsa u obliku dijaloga **gui.DialogBox()**. Skripta generiše poligon ili zvezdu sa datim dimenzijama i datim brojem temena, a onda ovaj g-kod učitava u mikroCNC.

Za kreiranje elemenata dijaloga koriste se funkcije objekta **DialogBox** kao što su: **AddText()**, **AddRadio()**, **AddEdit()** itd.

M117

Jednostavniji primer upotrebe gui modula za otvaranje dijaloga za unos podataka. Pored funkcija za kreiranje elemenata dijaloga demonstrirana je funkcija **GetValue()** za čitanje unetih vrednosti promenljivih.

7 G-Kod promenljive

7.1 Definisane promenljive

G-kod promenljive se navode uz upotrebu # simbola u G-kod programu.

Trenutno definisane G-kod promenljive koje se koriste i podržane su od strane mikroCNC programa:

2000 - 2003 koordinate kontakta vrha probe senzora (x,y,z,a) posle uspešne operacije probinga (umeravanja) G31

5211 - 5216 privremeni ofseti (x,y,z,a,b,c) koji se postavljaju komandama G92/G52

5220 – redni broj trenutno aktivnog seta ofseta (fixture)

5221 - 5226 G54 set, vrednosti radnih ofseta (x,y,z,a,b,c)
5241 - 5246 G55 set, vrednosti radnih ofseta
5261 - 5266 G56 set, vrednosti radnih ofseta
...
5601 - 5606 20th set, vrednosti radnih ofseta

7.2 Trajne promenljive

Korisničke promenljive u opsegu 500-599 su trajne u smislu da su njihove vrednosti sačuvane i posle izlaza iz programa i ponovnog startovanja.

IZMENE DOKUMENTA:

- Ver. 1.0, Decembar 2024., Polazna verzija uputstva za skripting
- Ver. 1.1, Januar 2025., Ispravljene uočene greška
- Ver. 1.5, Februar 2025., Dodate nove funkcije koje Python interpreter podržava
- Ver. 1.5, Maj 2025., Dodat opis modula ’**gui**’ (poglavlje 4.7)
- Ver. 1.51, Septembar 2025., Dopunjen opis funkcija koje su podržane u verziji softvera mikroCNC V0.71
- Ver. 1.73, Decembar 2025., Dopunjen opis funkcija koje su podržane u verziji softvera mikroCNC V0.73