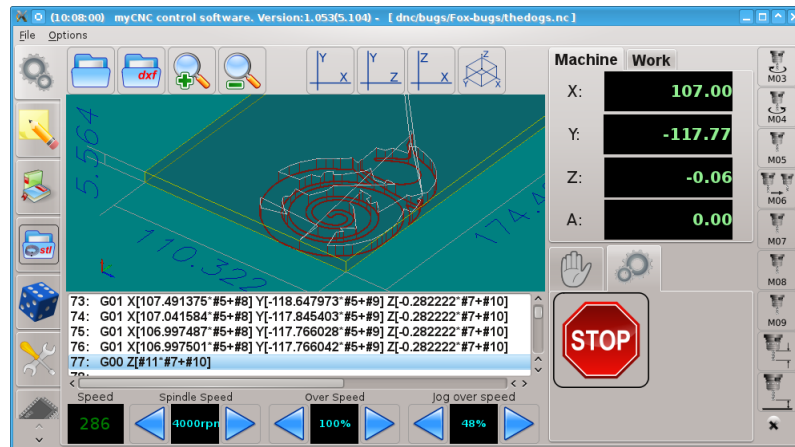


myCNC control & software.

MyCNC control software

User manual (rev 0.05 Preliminary 2011-0601)



myCNC

<http://www.bevelcutting.com>

email: sk@bevelcutting.com

195257, Russia, Saint-Petersburg

Severny st. 73-1-91

Using This Manual

This user manual provides information for proper installation, configuration and operation with myCNC control software.

WARNING: Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machinery. myCNC shall not be liable or responsible for any incidental or consequential damages

Table of content

1 Introduction.....	5
2 MyCNC software files structure.....	5
3 Starting myCNC.....	6
3.1. MyCNCcore running options.....	6
3.2. MyCNC running options.....	6
4 MyCNC software configuration.....	6
4.1. MyCNC profiles.....	6
4.2. myCNC - cnc-config.xml configuration file.....	7
4.2.1 Pulse per motion configuration.....	7
4.2.2 Controller processing time.....	8
4.2.3 Dnc files folder.....	9
4.2.4 Dnc files filter.....	9
4.2.5 global profile settings.....	9
4.2.6 Torch Height control (THC) configuration.....	10
4.2.7 Tablo response period.....	10
4.2.8 Core connection settings.....	11
4.2.9 myCNC to Ethernet controller connection settings.....	11
4.2.10 Touch screen version (Tabbar configuration).....	12
4.2.11 Tab icon size.....	14
4.2.12 Motors configuration.....	15
4.2.13 Homing after position.....	18
4.2.14 Probe sensor configuration.....	18
4.2.15 Tool length sensor configuration.....	20
4.2.16 Surface sensor configuration.....	21
4.3. myCNC – GUI Screen configuration.....	21
4.3.1 GUI definition overview.....	21
4.3.2 myCNC widget definition.....	22
4.3.3 Layout manager.....	23
4.3.4 Layout components.....	26
4.3.5 Layout attributes.....	26
4.3.6 Widget types description.....	27
4.3.7 Widget attributes description.....	28

Table index

Table 1. myCNC subfolder description.....	5
Table 2. Correspondance motor channels and board motor driver connectors for myCNC-UP3, ET1, ET2 boards.....	16

Illustration Index

Figure 1. “cnc-interface-touch-screen=0” - no Touch-screen version (small icons and tabbar on top).....	12
Figure 2. “cnc-interface-touch-screen=1” - Touch screen version (big icons and tabbar on left).....	14

Figure 3. “cnc-interface-tab-icon-size” defines icon size for main widget tab bar.....	15
Figure 4. myCNC-ET1 board outline. STEP/DIR motor interface connectors.....	15
Figure 5. myCNC-UP3 board outline. STEP/DIR motor interface connectors.....	16
Figure 6. the myCNC GUI.....	21
Figure 7. User-defined widget with horizontal orientation of components.....	24
Figure 8. User-defined widget with vertical orientation of components.....	24
Figure 9. User-defined widget with vertical orientation of components and different stretch.....	25
Figure 10. User-defined widget with tree layout.....	26
Figure 11. User-defined widget with tree layout and unused space, taken by stretch.....	26
Figure 12. User-defined widget with tree layout and unused space, taked by stretch.....	27
Figure 13. User-defined widget with tree layout, unused space and “glview” type given for “widget2”.....	28
Figure 14. User-defined widget with tree layout, unused space, “glview” and “mytabwidget”..	29

1 Introduction

The myCNC is software control system for machine tool centers. MyCNC provides:

- Graphic User Interface (GUI) for machine operator including Touch Screen;
- Interpreter G-codes;
- motion planning with look-ahead features;
- realtime operation interface with myCNC Motion controller and PLC controller;
- interface for creating configuration for different type of machines;

myCNC is multi-platform software. There are available versions for Linux, Windows and Mac operating systems.

2 MyCNC software files structure

After installation all files are located in myCNC folder. There is list of subfolders inside the myCNC with short description below.

Table 1. myCNC subfolder description.

Folder name	Folder description
MacOs	binary executable files, compiled for Mac OS
art	Scalable Vector Graphic (SVG) files collection that used as icons and button skins for myCNC
bin	Binary executable myCNC files and dll libraries for proper myCNC software running
dnc	Folder for NC files storage
driver.usb-to-serial	FTDI FT232RL USB-to-serial converter driver for MS Windows (used with USB CNC controller myCNC-UP3)
lib	Shape Library folder
linux	binary executable files, compiled for Linux OS
plc	PLC compiler (binary files for Linux and Windows, additional preprocessor scripts, running scripts).
profiles	Folder for profiles – configuration files, macroses and PLC programs for different machine types.
report	Folder for report files - CVS files with machine work statistics (running and cutting time, nc files, log on time etc)
translations	Multi-lungual interface files

MyCNC software contains two executable files:

- myCNC – GUI program.
- myCNCcore – driver-like program that provides interface between the GUI and myCNC controller

Both programs should be running for proper work of myCNC control system. MyCNC & myCNCcore interconnect with each other via TCP/IP Sockets, so the programs may be started on different computers.

3 Starting myCNC

To run myCNC control two binary files should be started:

- myCNC
- myCNCcore

To start myCNC control with correct parameters and profile are used running scripts (separate for each operating system).

3.1. MyCNCcore running options.

To drive your myCNC controller myCNCcore should be started with the right options. There are available options:

- d start myCNCcore with debug messages (the core tell many things about what happens around)
- e start myCNCcore in LAN/Ethernet mode (to connect with myCNC Ethernet controller myCNC-ET1 or myCNC-ET2)
- u start myCNCcore in USB mode (to connect with myCNC USB controller myCNC-UP3)
- s start myCNCcore in Simulator mode (core doesn't try to connect to the controller)

3.2. MyCNC running options.

To run myCNC software with your machine configuration profile folder should be given for myCNC as -p option:

- p your_profile_name will start myCNC and will load profile form "profiles/your_profile_name" folder

4 MyCNC software configuration

4.1. MyCNC profiles.

MyCNC software can be configured for many type of CNC machinery. For each type of machine can be created profile. Profile includes:

- macroses;
- PLC source programs;
- ROMFS disk image with PLC compiled binary files;
- myCNC configration file "cnc-config.xml" with "read-only" configuration, that is not available for change from myCNC software;
- myCNC variables file "cnc-variables.xml" with parameters and variables, that can be changed from myCNC software dialogs;

4.2. myCNC - cnc-config.xml configuration file.

Cnc-config.xml file contains:

- myCNC GUI configuration (size, position, colors, fonts etc for nc-preview widgets, inputs fields, display fields, buttons, spin-boxes etc) for Main Tab widget, Diagnose Widget, User-defined configuration widgets;
- CNC configuration:
 - mill/lathe/cutting table;
 - controller processing time;
 - PULSE/DIR interface pulse with;
 - electronic gears ratio (pulse per distance ratio);
 - dnc home folder;
 - dnc files filter;
 - Torch height controller parameters;
 - motor connection matrix;
 - Limit switchers configuration;
 - Ready Sensors configuration;

4.2.1 Pulse per motion configuration.

There are 8 parameters available for pulse-per-motion configuration. With this parameters the ratio may be configured separately for each axis:

- "cnc-dimension-linear-step" – based ratio for linear axes (X,Y,Z);
- "cnc-dimension-angular-step" – based ratio for angular axes (A, B, C);
- "cnc-dimension-e-gears-ratio-0" – additional ratio for X axis;
- "cnc-dimension-e-gears-ratio-1" – additional ratio for Y axis;
- "cnc-dimension-e-gears-ratio-2" – additional ratio for Z axis;
- "cnc-dimension-e-gears-ratio-3" – additional ratio for A axis;
- "cnc-dimension-e-gears-ratio-4" – additional ratio for B axis;
- "cnc-dimension-e-gears-ratio-5" – additional ratio for C axis;

Complete pulse-per-motion ratio can be calculated as multiply of 2 coefficients:

- for X: "cnc-dimension-linear-step" * "cnc-dimension-e-gears-ratio-0"
- for Y: "cnc-dimension-linear-step" * "cnc-dimension-e-gears-ratio-1"
- for Z: "cnc-dimension-linear-step" * "cnc-dimension-e-gears-ratio-2"
- for A: "cnc-dimension-angular-step" * "cnc-dimension-e-gears-ratio-3"
- for B: "cnc-dimension-angular-step" * "cnc-dimension-e-gears-ratio-4"

- for C: "cnc-dimension-angular-step" * "cnc-dimension-e-gears-ratio-5"

in configuration file the parameters are recorder in xml format:

```
<value name="cnc-dimension-linear-step">0.002256668824</value>
<value name="cnc-dimension-angular-step">0.002</value>
<value name="cnc-dimension-e-gears-ratio-0">1.0</value>
<value name="cnc-dimension-e-gears-ratio-1">1.0</value>
<value name="cnc-dimension-e-gears-ratio-2">1.0</value>
<value name="cnc-dimension-e-gears-ratio-3">1.0</value>
<value name="cnc-dimension-e-gears-ratio-4">1.0</value>
<value name="cnc-dimension-e-gears-ratio-5">1.0</value>
```

Examples:

1. We use step motor driver with 1600 pulses per 1 turn. A machine moves 1,25 mm with one motor shaft turn for X axis and 1,5 mm for Y axis

Let's set as based X axis. Then for X:

1600 pulses = 1,25 mm , then

1 pulse = 1,25 mm / 1600 = 0.00078125

So this value should be set as cnc-dimension-linear-step:

```
<value name="cnc-dimension-linear-step">0.00078125</value>
```

and e-gears ratio for X should be «1»

```
<value name="cnc-dimension-e-gears-ratio-0">1.0</value>
```

for Y e-gears ratio should be 1,5/1,25=1,2

```
<value name="cnc-dimension-e-gears-ratio-0">1.2</value>
```

2. We use servo motor driver with 10000 pulses per 1 turn for A axis with 1/30 gearbox.

Then for A:

10000 pulses = 1/30 turn = 360/30 degree = 12 degree, then

1 pulse = 12 degree / 10000 = 0.0012

So this value should be set as cnc-dimension-angular-step:

```
<value name="cnc-dimension-angular-step">0.0012</value>
```

and e-gears ratio for A should be «1»

```
<value name="cnc-dimension-e-gears-ratio-3">1.0</value>
```

4.2.2 Controller processing time.

Controller processing time is internal parameter, that should be set depends on myCNC controller:

- 0.000256 – for myCNC-UP3 controller;
- 0.000081 – for myCNC-ET1/myCNC-ET2 controllers;

If wrong value given for this parameter, actual motion speed will be incorrect.

Examples:

for myCNC-UP3

```
<value name="controller-processing-time">0.000256</value>
```

for myCNC-ET1, myCNC-ET2

```
<value name="controller-processing-time">0.000081</value>
```

4.2.3 Dnc files folder.

Folder where myCNC software finds dnc files could be set with parameter “cnc-dnc-folder”.

Examples:

```
<text-value name="cnc-dnc-folder">dnc</value>
```

myCNC software will read folder “dnc” relatively from myCNC home folder.

4.2.4 Dnc files filter.

MyCNC software File Open dialog will try to parse files with given extensions as dnc files (G-codes, ESSI codes)

Example:

```
<text-value name="cnc-dnc-filter">*.nc;*.NC;*.dnc;*.iso;</text-value>
```

myCNC software will read and parse all files:

- *.nc
- *.NC
- *.dnc
- *.iso

as dnc codes.

4.2.5 global profile settings.

myCNC software can be configured for cutting table, mill, turning application by set parameter: “cnc-profile”. Possible values are listed below:

- cutting-table
- mill
- lathe

Example:

```
<text-value name="cnc-profile">cutting-table</text-value>
```

myCNC software is configured as cutting table.

4.2.6 Torch Height control (THC) configuration.

If Torch height control is used, configuration parameters for myTHC can be programmed in “cnc-config.xml” and “cnc-variables.xml”.

It's possible to connect PC/Laptop to myTHC-RU01 USB-slave connector and get real-time diagnostic information (working mode, measured voltage value, given voltage value, PID regulator control value, control speed). This information could be used while tuning of myTHC PID controller. Diagnostic stream can be turned on/off with parameter “thc-diagnostic”

```
<value name="thc-diagnostic">0</value>
```

0 – diagnostic is turned off;

1 – diagnostic is turned on;

Depends on plasma power source and plasma torch type, Plasma voltage measure amplifier ratio can be different. MyCNC receives from myTHC controller 12-bit digital value of measured plasma voltage. To show this value correctly in Volts is used ratio parameter “thc-arc-voltage-Kv”. By default this value equal 0.0732, so maximum of 12-bit value (4095) match to $(4095 * 0.0732) = 300V$

```
<value name="thc-arc-voltage-Kv">0.0732</value>
```

myTHC keeps constant torch-sheet distance by driving Z-axis. MyTHC can be configured either to direct drive DC motor, or to drive a machine Z-axis through myCNC control by sending motion commands to it. Configuration is programmed with parameter “thc-lift-unit-type”:

```
<value name="thc-lift-unit-type">0</value>
```

0 — myTHC drives DC motor with integrated DC motor driver/amplifier;

1 — myTHC controls machine Z-axis by sensing commands to myCNC controller.

4.2.7 Tablo response period.

myCNC controller and other system units (like myTHC, myServo etc) sends information to myCNC software. Usually there are a lot of tabloes on the GUI screen that display this information (current position, speed, plasma voltage, sensors state etc). Controllers may send the new data quite quickly. On slow Host PC system constant repainting on-screen tabloes may take most of CPU resources and may cause to unstable work. For slow systems parameter “tablo-response-period” should be programmed. After repaint is finished each tablo will be insensitive to data changes during this period (in miliseconds). For example, if the parameter value is “1000” on-screen tablo information will be changed each 1 second (not more frequently).

Example:

```
<value name="tablo-response-period">10</text-value>
```

delay for data updates will be not more than 10 ms.

4.2.8 Core connection settings.

Interface between the myCNC GUI and the myCNCcore is implemented via TCP/IP Sockets. Connection parameters can be configured with parameters "cnc-core-ip-address" and "cnc-core-listen-port".

"cnc-core-ip-address" is IP address of PC host where myCNCcore is running. By default myCNC & myCNCcore are running on the same PC, so address is "127.0.0.1".

"cnc-core-listen-port" is TCP/IP port, that listen myCNCcore application. By default the port is "4210".

Example:

```
<text-value name="cnc-core-ip-address">127.0.0.1</text-value>
<value name="cnc-core-listen-port">4210</value>
```

4.2.9 myCNC to Ethernet controller connection settings.

Interface between the myCNC control software and myCNC LAN/Ethernet controllers is implemented via TCP/IP Sockets as well as myCNC to myCNCcore interconnection.

Connection parameters can be configured with parameters "cnc-controller-ip-address" and "cnc-controller-listen-port".

"cnc-controller-ip-address" is physical IP address of myCNC control board. The address is stored in flash memory of the controller and can be changed with terminal commands through USB-slave serial connection. By default IP address of the board is "192.168.4.60".

myCNC control board listens TCP/IP port to establish connection. Value of this port should be given to myCNC software in "cnc-controller-listen-port" parameter. Default port value is "4220".

Example:

```
<text-value name="cnc-controller-ip-address">192.168.4.78</text-value>
<value name="cnc-controller-listen-port">4220</value>
```

4.2.10 Touch screen version (Tabbar configuration).

Parameter "cnc-interface-touch-screen" defines view, size and location of the GUI Main widget tabbar.

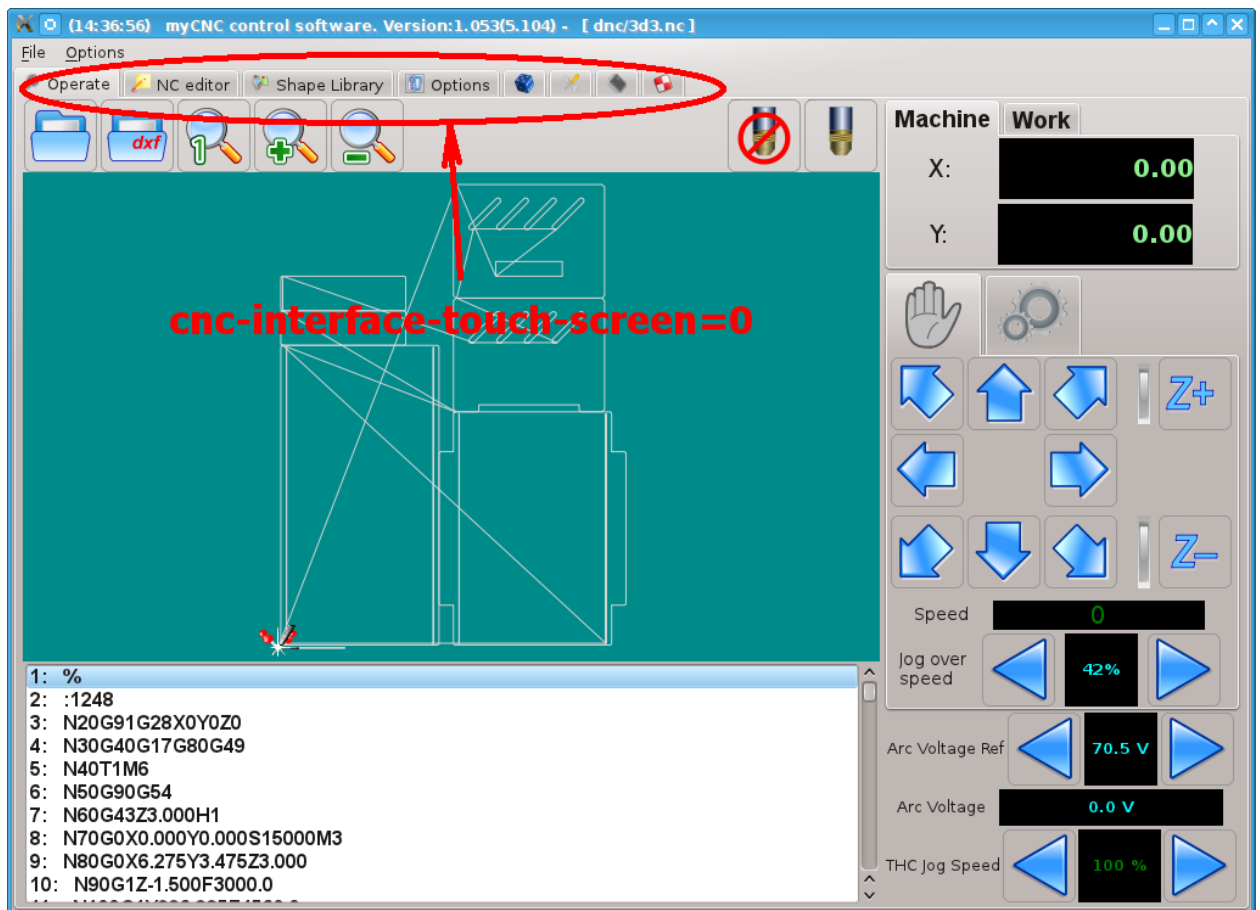


Figure 1. "cnc-interface-touch-screen=0" - no Touch-screen version (small icons and tabbar on top).

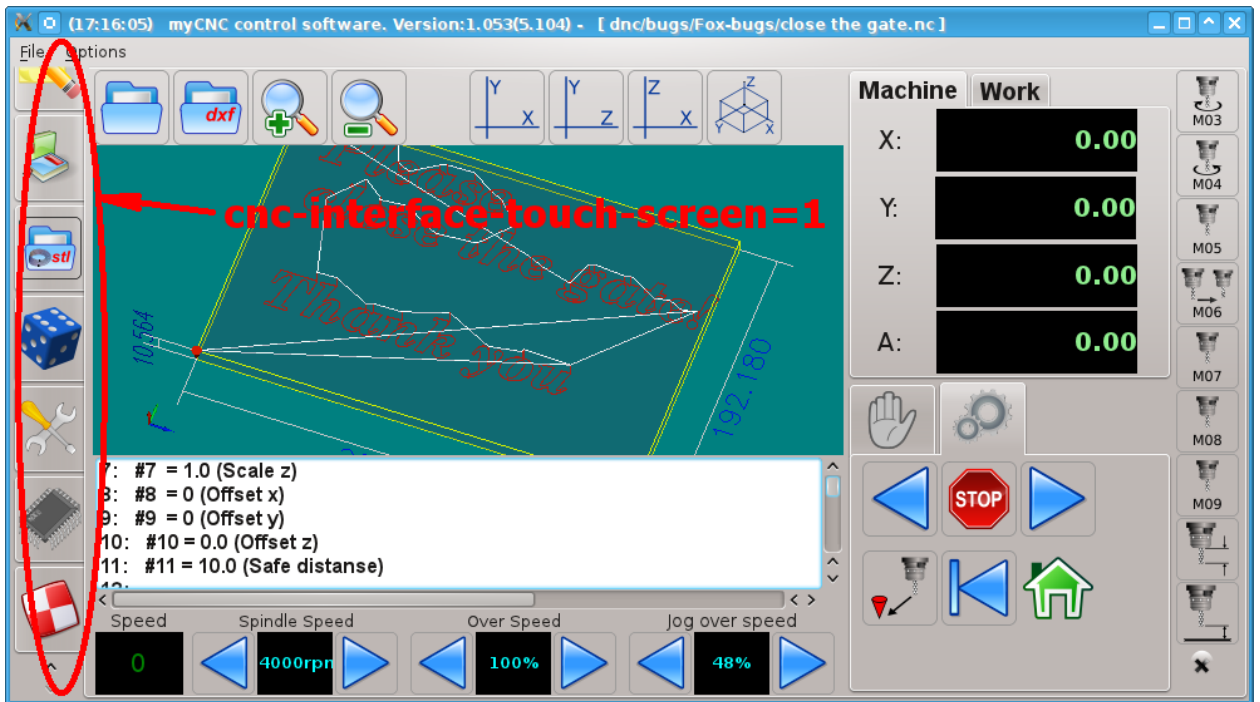


Figure 2. "cnc-interface-touch-screen=1" - Touch screen version (big icons and tabbar on left).

Example:

```
<value name="cnc-interface-touch-screen">0</value>
```

4.2.11 Tab icon size.

Parameter "cnc-interface-tab-icon-size" defines icon size (in pixels) for GUI main widget tab bar (for cnc-interface-touch-screen=1 only).

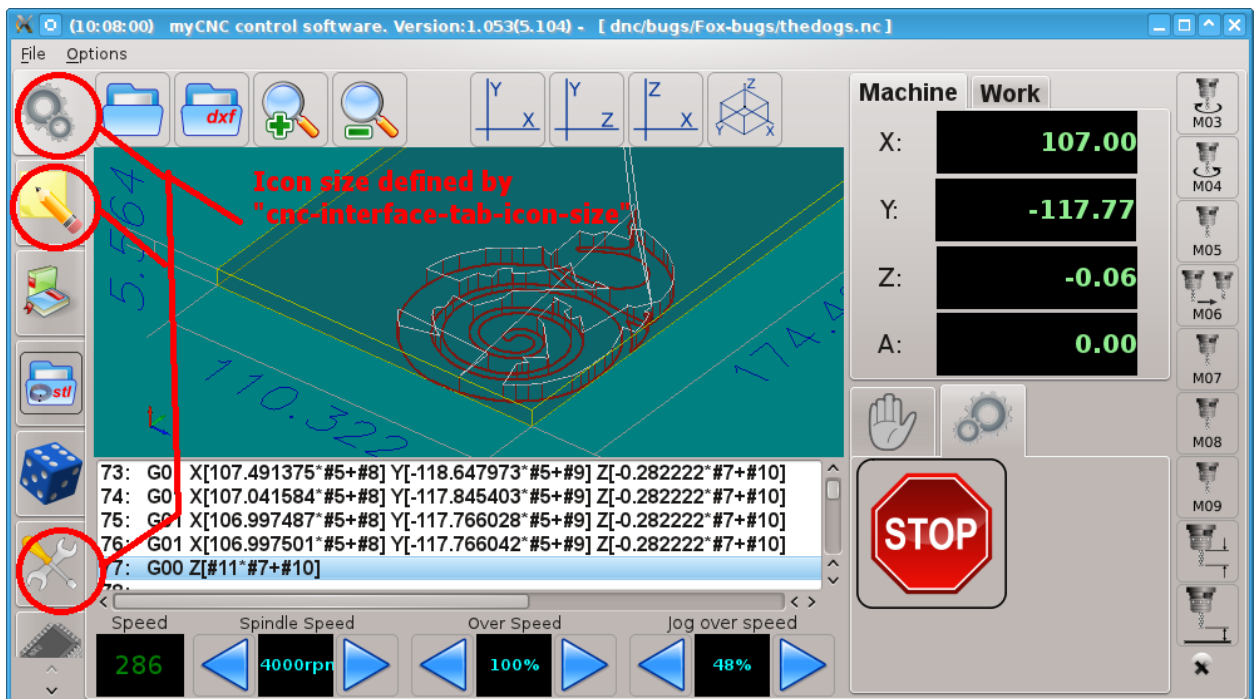


Figure 3. “cnc-interface-tab-icon-size” defines icon size for main widget tab bar.

Example:

```
<value name="cnc-interface-tab-icon-size">50</value>
```

4.2.12 Motors configuration.

MyCNC controllers contains true 6 axes motion controller and may operate with 6 axes simultaneously (XYZABC). Depends on model myCNC control board contains 3-8 STEP/DIR motor driver output channels. Any of motor output channel may be configured flexibly to work as any of 6 axes.

On a figures below are displayed motor driver outputs for myCNC-UP3 & myCNC-ET1 board.

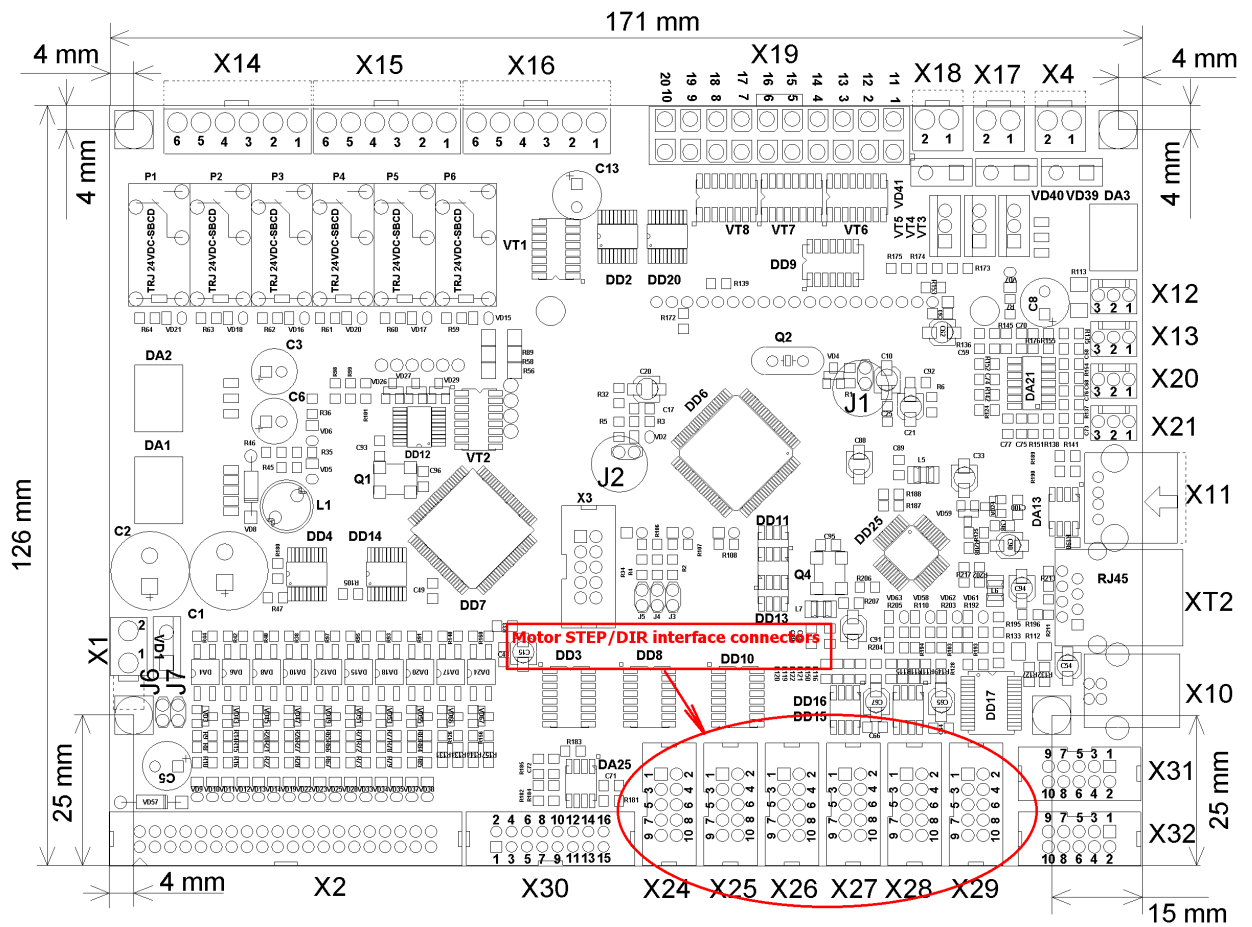


Figure 4. myCNC-ET1 board outline. STEP/DIR motor interface connectors.

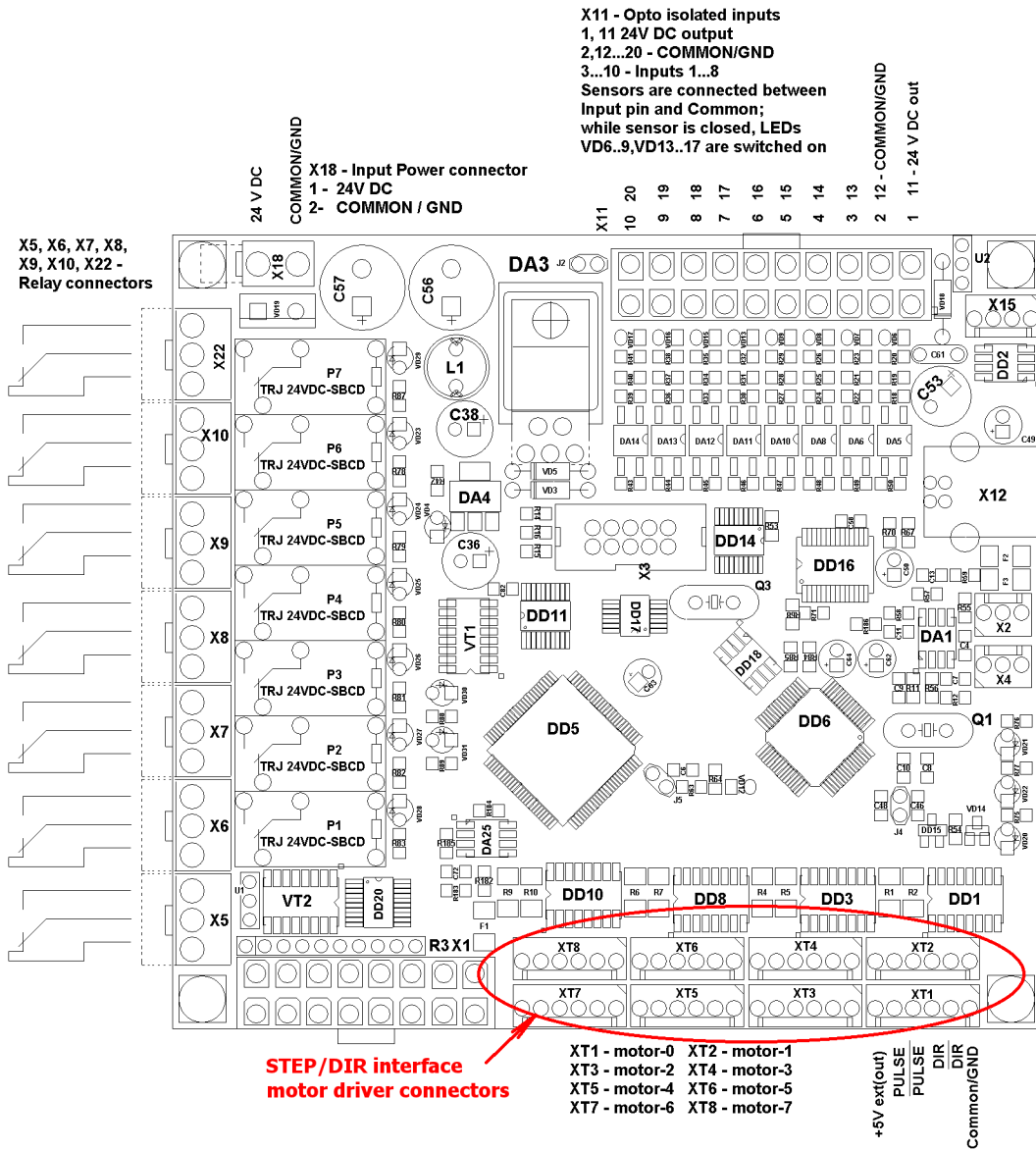


Figure 5. myCNC-UP3 board outline. STEP/DIR motor interface connectors.

Table 2. Correspondance motor channels and board motor driver connectors for myCNC-UP3, ET1, ET2 boards.

Motor channel name	MyCNC-UP3 connectors	MyCNC-ET1 connectors	MyCNC-ET2 connectors
motor-0	XT1	X24	
motor-1	XT2	X25	
motor-2	XT3	X26	
motor-3	XT4	X27	
motor-4	XT5	X28	
motor-5	XT6	X29	
motor-6	XT7	—	
motor-7	XT8	—	

To program motor configuration there are 6 parameters in the “cnc-config.xml” file:

- "cnc-matrix-motor-0"
- "cnc-matrix-motor-1"
- "cnc-matrix-motor-2"
- "cnc-matrix-motor-3"
- "cnc-matrix-motor-4"
- "cnc-matrix-motor-5"
- "cnc-matrix-motor-6"
- "cnc-matrix-motor-7"

Parameter value contains:

- a letter with axis name (“X”, ”Y”, ”Z”, ”A”, ”B” or “C”);
- a sign (“+” or “-”);

For example -

“X+” means, that selected channel (connector) will represent X axis with positive direction (by default, with positive X motion motor shaft will turn in “positive” direction (CCW))

and

“X-” means, that selected channel (connector) will represent X axis with negative direction (by default, with positive X motion motor shaft will turn in “negative” direction (CW))

Example:

We need to configure 2D machine with dual motors on X axis.

Connector 1 (motor-0) is used for Y axis;

Connectors 3 and 4 (motor-2& motor-3) are used for dual X axis; both X motors should turn synchronous but in opposite direction; Configuration for this machine is listed below:

```
<text-value name="cnc-matrix-motor-0">Y+</text-value>
<text-value name="cnc-matrix-motor-2">X+</text-value>
<text-value name="cnc-matrix-motor-3">X-</text-value>
```

Motor configuration is also available via CNC variables layer.

Variables CNC_VAR_MOTOR0_MATRIX (0x70) ... CNC_VAR_MOTOR7_MATRIX (0x77) are responsible for motor configuration.

By changing CNC variable value we change commutation matrix for any of motor channel.

The CNC variable first 4 bits represent the axis:

- 0 – X
- 1 – Y
- 2 – Z
- 3 – A
- 4 – B
- 5 – C
- 6 – axis 7 (reserved)
- 7 – axis 8 (reserved)
- 9...15 – not used (PULSE/DIR switched off)

The CNC variable fifth bit represents inversion (opposite turning direction) of PULSE/DIR interface.

CNC variables are available for change/set from:

- “cnc-variable.xml” file with direct action “direct-set-cnc-var:”;
- G-code programming through G10L80 command;

4.2.13 Homing after position.

To find Home position is used homing procedures, that in fact are macro procedures M131-M136 that are stored in macro folder of current profile. After homing procedure is finished for selected axis, the machine coordinate is set to position given in parameters:

- “homing-after-position-x”
-
- “homing-after-position-c”

Syntax for parameters is shown below. Zero position is programmed in current linear/angular units (mm/inches; degree) in double values and re-calculated into step units by myCNC software.

Example:

```
<value name="homing-after-position-x">200.0</value>
<value name="homing-after-position-y">150.0</value>
<value name="homing-after-position-z">100.0</value>

<value name="homing-after-position-a">0</value>
<value name="homing-after-position-b">0</value>
<value name="homing-after-position-c">0</value>
```

myCNC software loads this values and stores it into Global Variable Array. So G-code programs may access to the Homing position through GVAR array variables:

- #5451 – axis X homing position;
- #5452 – axis Y homing position;
- #5453 – axis Z homing position;
- #5454 – axis A homing position;
- #5455 – axis B homing position;
- #5456 – axis C homing position;
- #5457 – axis 7 (reserved) homing position;
- #5458 – axis 8 (reserved) homing position;

4.2.14 Probe sensor configuration.

Any of binary inputs of myCNC controller can be used as probe sensor. To configure probe sensor are used parameters:

- "cnc-probe-sensor-number" – set number of myCNC input, that is used as probe sensor;

- "cnc-probe-sensor-inversion" – configure active state of probe sensor.
0 – normally opened sensor;
1 – normally closed sensor;

Example:

```
<value name="cnc-probe-sensor-number">5</value>  
<value name="cnc-probe-sensor-inversion">0</value>
```

G-codes G38.2 – G38.5 are used to operate with probe sensor.

4.2.15 Tool length sensor configuration.

Any of binary inputs of myCNC controller can be used as tool length sensor. To configure tool length sensor are used parameters:

- "cnc-tool-length-sensor-number" – set number of myCNC binary input, that is used as tool length sensor;
- "cnc-tool-length-sensor-inversion" – configure active state of tool length sensor.
0 – normally opened sensor;
1 – normally closed sensor;

Example:

```
<value name="cnc-tool-length-sensor-number">5</value>  
<value name="cnc-tool-length-sensor-inversion">0</value>
```

G-code G38.9 is used to operate with tool length sensor.

Parameters "cnc-tool-length-sensor-position-x" ... "cnc-tool-length-sensor-position-c" are used to program actual location of the Tool Length sensor (in machine coordinates).

Example:

```
<value name="cnc-tool-length-sensor-number">5.0</value>  
<value name="cnc-tool-length-sensor-inversion">0.0</value>  
<value name="cnc-tool-length-sensor-position-x">100.0</value>  
<value name="cnc-tool-length-sensor-position-y">110.0</value>  
<value name="cnc-tool-length-sensor-position-z">-5.0</value>  
<value name="cnc-tool-length-sensor-position-a">0.0</value>  
<value name="cnc-tool-length-sensor-position-b">0.0</value>  
<value name="cnc-tool-length-sensor-position-c">0.0</value>
```

Tool length measure procedure is in fact macros “M121”, that is placed in macros folder. It may be redefined by users.

By default, procedure moves the tool toward position, where tool length sensor is located and placed the tool above this position. Then the tool moves down toward the sensor and stops when tool length sensor is pressed. Difference between actual sensor position and position, when the sensor was pressed is the tool length. This value is stored in the controller memory and used as tool length offset.

4.2.16 Surface sensor configuration.

Any of binary inputs of myCNC controller can be used as surface sensor. To configure surface sensor are used parameters:

- "cnc-surface-sensor-number" – set number of myCNC input, that is used as surface sensor;
- "cnc-surface-sensor-inversion" – configure active state of surface sensor.
0 – normally opened sensor;
1 – normally closed sensor;
- "cnc-surface-sensor-width" – actual width of surface sensor. This value is used as additional offset for surface measurement.

Example:

```
<value name="cnc-surface-sensor-number">0</value>  
<value name="cnc-surface-sensor-inversion">0</value>  
<value name="cnc-surface-sensor-width">50.0</value>
```

There is no special code for using surface measure sensor. Probe sensor is used in Surface measure procedure (M120 macros). The parameters for number and inversion of surface sensor are reserved for further development. Sensor width parameter is active.

4.3. myCNC – GUI Screen configuration.

4.3.1 GUI definition overview.

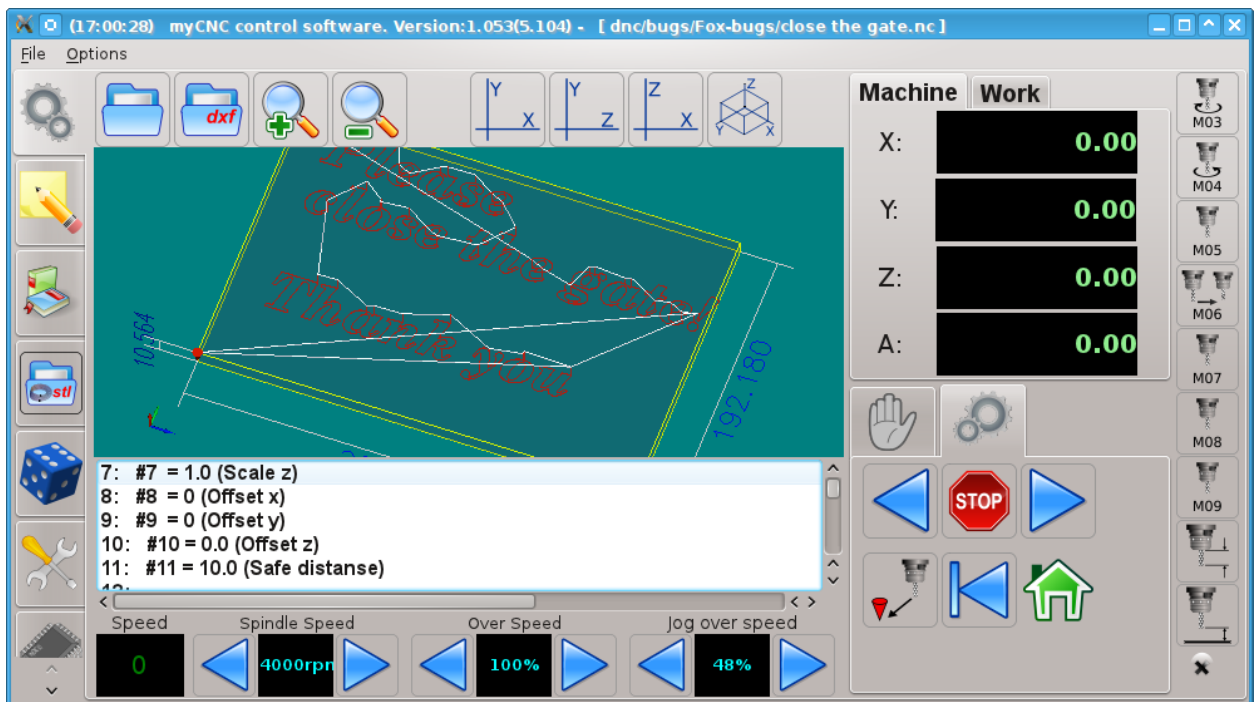


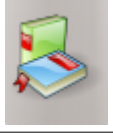

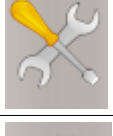
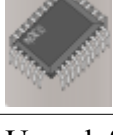


Figure 6. the myCNC GUI.

MyCNC main widget is based on a TabWidget interface.
 Some of tabs are fixed, others are completely flexible and can be redefined by user through “cnc-config.xml”
 Tabs are listed below:

	Operate widget. Completely flexible through «cnc-config.xml» «work-layouts»
	Edit widget. Completely flexible and defined in «cnc-config.xml» «edit-layouts»
	The Shape Library widget. Fixed definition.
	STL import, export to G-code toolpath. Fixed definition. Under developing
	Configuration widget. Fixed definition of the widget. Content of the widget may be modified by edit «cnc-variables.xml».
	Diagnose widget. Completely flexible through «cnc-config.xml» «diagnose-layouts»
User-defined icon	User defined widgets. Completely flexible through «cnc-config.xml» «quick-widget-layout»

«cnc-config.xml» contains section «screen» for GUI widgets definition. Section «screen» contains sub-sections:

- work-layouts — Operate widget definition;
- edit-layouts — Edit widget definition;
- diagnose-layouts — Diagnose widget definition;
- quick-widget-layout — user-defined widget definition; User may add user-defined widgets by adding new «quick-widget-layout» section;

4.3.2 myCNC widget definition.

Each section «work-layouts», «edit-layouts», «diagnose-layouts» contains one or several sections “layout” with widget definition and section “current”. Each section “layout” has attribute “name”. Section with attribute “name” is equal to section “current” value is used as the widget definition. Other “layout” sections are ignored.

This way in the “cnc-config.xml” file may be created several widget configurations and quick switch from one widget view to another is possible by changing “current” section value.

Example:

```
<diagnose-layouts>
  <current>diagnose0</current>
  <layout name="diagnose0" orientation="horizontal">
    ...
  </layout>
  <layout name="diagnose1" orientation="horizontal">
    ...
  </layout>
</diagnose-layouts>
```

Two widget configurations defined (diagnose0, diagnose1). Diagnose0 configuration is active, diagnose1 is ignored.

4.3.3 Layout manager.

Widget configuration definition is based on layout manager — stack of components, that may be placed:

- horizontally from left to right;
- vertically from top to bottom;

«layout» section attribute «orientation» defines type of components placement.

Examples:

```
<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="system-run" orientation="horizontal">
    <widget stretch="0" name="red-widget" orientation="vertical"
bcolor="red">myitems</widget>
    <widget stretch="0" name="blue-widget" orientation="horizntal"
bcolor="blue">myitems</widget>
  </layout>
</quick-widget-layout>
```

User-defined widget — layout with name «quick-02» contains components with horizontal orientation. There are two widgets named «red-widget» and «blue-widget». Widgets attribute «bcolor» defines background color for each widget. Widget attribute orientaton defines orientation of components inside the widget. Attribute «stretch» defines stretch of each widget on the layout. Screenshot of this widget is on a illustration below:

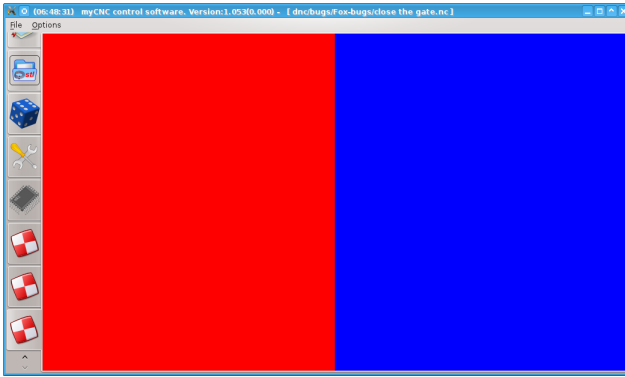


Figure 7. User-defined widget with horizontal orientation of components.

```
<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="system-run" orientation="vertical">
    <widget stretch="0" name="widget0"
bcolor="yellow">myitems</widget>
    <widget stretch="0" name="widget1"
bcolor="magenta">myitems</widget>
  </layout>
</quick-widget-layout>
```

User-defined widget — layout with name «quick-02» contains components with vertical orientation. There are two widgets named «widget0» and «widget2». Attribute «bcolor» defines background color for each widget. Attribute «stretch» defines stretch of each widget on the layout. Screenshot of this widget is on a illustration below:



Figure 8. User-defined widget with vertical orientation of components.

```
<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="system-run" orientation="vertical">
    <widget stretch="1" name="widget0"
bcolor="yellow">myitems</widget>
    <widget stretch="4" name="widget1"
bcolor="magenta">myitems</widget>
  </layout>
</quick-widget-layout>
```

User-defined widget — layout with name «quick-02» contains components with vertical orientation. There are two widgets named «widget0» and «widget2». Attribute «bcolor» defines background color for each widget. Attribute «stretch» defines stretch of each widget on the layout. In this example the widgets has different stretch attribute values. Screenshot of this widget is on a illustration below:

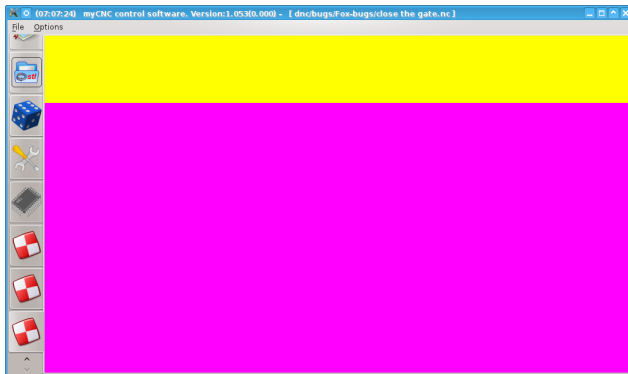


Figure 9. User-defined widget with vertical orientation of components and different stretch.

Tree-like structure is available, so another layout may be used as componet.
Example:

```

<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="system-run" orientation="vertical">
    <widget stretch="1" name="widget0" bcolor="yellow">myitems</widget>
    <layout stretch="4" orientation="horizontal">
      <widget stretch="1" name="widget1"
bcolor="magenta">myitems</widget>
      <widget stretch="2" name="widget2"
bcolor="blue">myitems</widget>
    </layout>
  </layout>
</quick-widget-layout>
  
```

User-defined widget — layout with name «quick-02» contains components with vertical orientation. Components are:

- a widget named «widget0»;
- layout with horizontal orientation of its components (with widget1 & widget2 widgets inside)

Screenshot of this widget is on a illustration below:

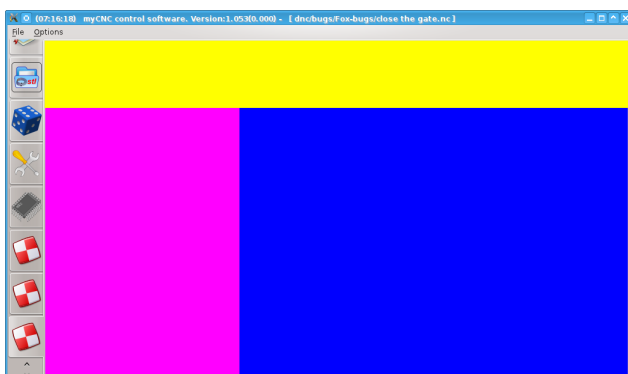


Figure 10. User-defined widget with tree layout.

4.3.4 Layout components.

Inside of each layout there can be component types:

- Layout. Tree-like structure available and child layout may be as a component of parent layout;
- Widget. One of many widgets may be placed inside of layout.
- Stretch. Components in layout take all available space by default. If components should be packet on the screen of some free space should be left, “stretch” component may be used. Stretch section of the “cnc-config.xml” file should contain the-same-name attribute “stretch” with stretch value.

Example:

If compare with the last example — there is stretch section added to child layout between «magenta» and «blue» widgets.

```
<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="system-run" orientation="vertical">
    <widget stretch="1" name="widget0" bcolor="yellow">myitems</widget>
    <layout stretch="4" orientation="horizontal">
      <widget stretch="1" name="widget1"
bcolor="magenta">myitems</widget>
      <stretch stretch="2"/>
      <widget stretch="2" name="widget2"
bcolor="blue">myitems</widget>
    </layout>
  </layout>
</quick-widget-layout>
```

Screenshot of this widget is on a illustration below:

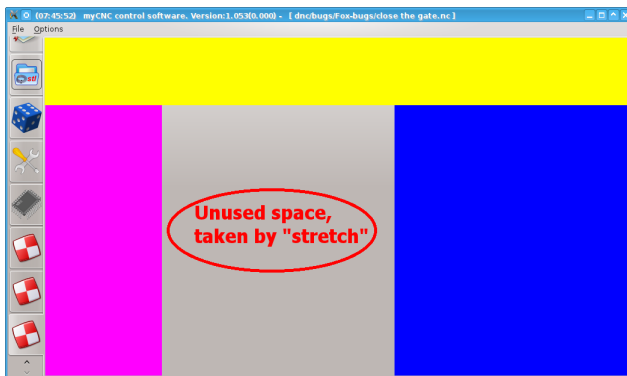


Figure 11. User-defined widget with tree layout and unused space, taken by stretch.

4.3.5 Layout attributes.

Each layout may contain attribute, that describe its behavior.

- “name”. Name of layout used for root layout and defines the layout active or not (with “current” section);
- “orientation” defines horizontal or vertical orientation of layout components;

- “stretch” defines stretch of the layout;
- “image”. Value of “image” attribute for root layout is image filename (in “icons” folder) that is used as Tabbar icon for the widget.

Example:

```

<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="dices-r" orientation="vertical">
  ...
  ...
  </layout>
</quick-widget-layout>

<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="dnc-program" orientation="vertical">
  ...
  ...
  </layout>
</quick-widget-layout>

```

The GUI screenshot is on a illustration below:

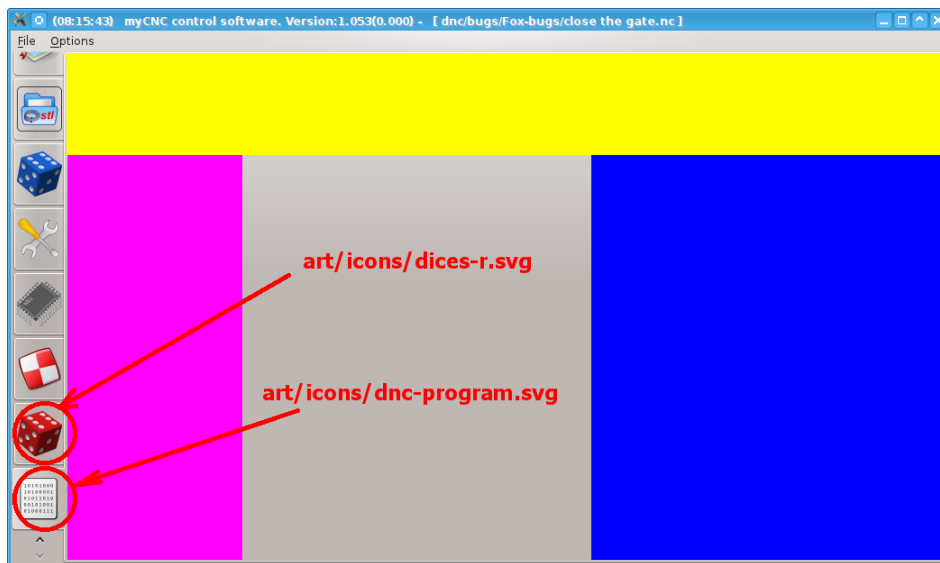


Figure 12. User-defined widget with tree layout and unused space, taked by stretch.

4.3.6 Widget types description.

Section “widget” contains type of the widget and its description.

Type of widget is defined by section value:

```
<SectionName attribute1="value1" ... attributeN="valueN">SectionValue</SectionName>
```

Type of widget defines basic widget behavior. There are types available:

- “ncview” - 2D visualisation of main DNC code;
- “glview” - 3D visualisation of main DNC code (with using opengl extensions);
- “nclist” - ListBox with DNC code listing and automatic highlight of currently running NC line.

- “myitems” - widget that is actually stack of registered graphic items (buttons, leds, tabloes, spin-boxes, text and graphic labels etc).
- “mytabwidget” - Tab widget interface.
- “mytabitems” - widget that is actually stack of registered graphic items (buttons, leds, tabloes, spin-boxes, text and graphic labels etc) and “mylayoutitems” widgets.
- “mylayoutitems” - box layout interface with registered graphic items inside, owned by “mytabwidget”.

Example:

```

<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="dnc-program" orientation="vertical">
    <widget stretch="1" name="widget0" bcolor="yellow">myitems</widget>
    <layout stretch="4" orientation="horizontal">
      <widget stretch="1" name="widget1" bcolor="magenta">myitems</widget>
      <stretch stretch="2"/>
      <widget stretch="2" name="widget2" bcolor="blue">glview</widget>
    </layout>
  </layout>
</quick-widget-layout>

```

If compare with the last example, we've changed type of widget2 to “glview”.
A result is shown on a figure below:

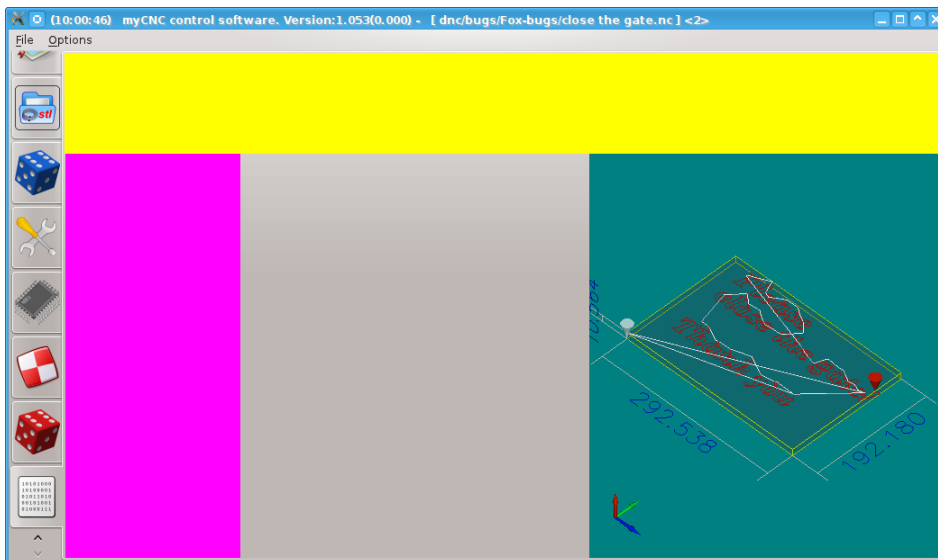


Figure 13. User-defined widget with tree layout, unused space and “glview” type given for “widget2”.

4.3.7 Widget attributes description.

Widget section may contains attributes:

- “name”. Name of the widget. Graphic items will be appended to the stack if “name” of the widget equal to the graphic item “where” attribute value;
- “height” – height of widget in pixels;
- “width” – width of the widget in pixels;
- “iconsize” – Tabbar icon size in pixels for “mytabitems” (default value is 80);
- “stretch” – stretch of widget;
- “orientation” – orientation for widgets components;
- “where” – name of parent widget, where the widget should be placed;

- “children” – flag for “mylayoutitems”, that should be set to “yes” if the widget contains children widgets. If the widget contains only graphic items (“gitem”), the flag should be set to “no”;
- “hidden” – hidden flag may be set to “1” or “0”. If the widget should be hidden after myCNC software started, the flag should be set to “1”. The widget may be shown later by external tools (for example – by pressing some on-screen button).
- “tabname” – for mytabitems defines Tabbar name for this widget;
- “image” – for mytabitems defines icon filename (in a folder “art/icons/”), that will be shown on Tabbar for this widget;

Example:

Change type of “widget1” to mytabwidg, make it wider and add two tabs to the widget with single and double dices icons (files “dices-r.svg” and “dices2.svg” in “art/icons” folder).

A result is shown on a figure below:

```

<quick-widget-layout>
  <current>quick-02</current>
  <layout name="quick-02" image="dnc-program" orientation="vertical">
    <widget stretch="1" name="widget0" bcolor="yellow">myitems</widget>
    <layout stretch="4" orientation="horizontal">
      <widget stretch="3" name="widget1" bcolor="magenta">mytabwidget
</widget>
      <widget iconsize="100" where="widget1" image="dices-r"
name="widget1-1" bcolor="blue">mytabitems</widget>
      <widget iconsize="100" where="widget1" image="dices2"
name="widget1-2" bcolor="red">mytabitems</widget>
      <stretch stretch="1"/>
      <widget stretch="3" name="widget2" bcolor="blue">glview</widget>
    </layout>
  </layout>
</quick-widget-layout>

```

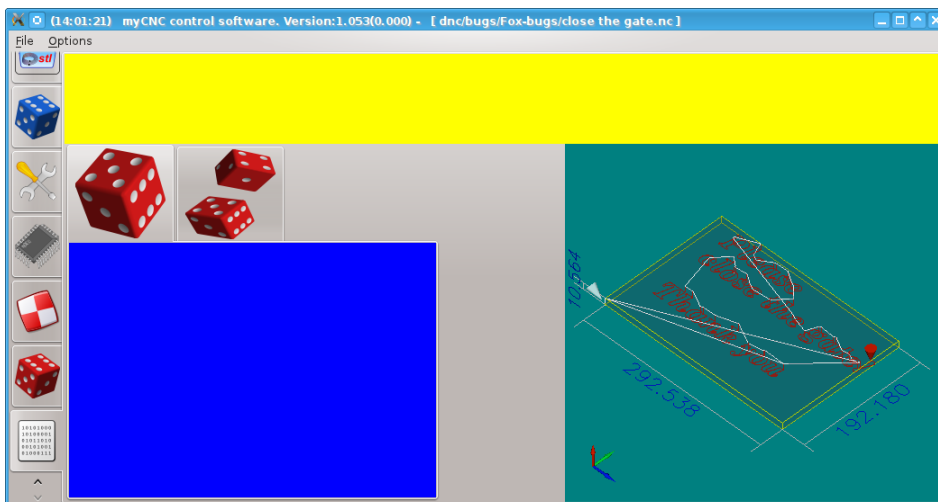


Figure 14. User-defined widget with tree layout, unused space, “glview” and “mytabwidget”.

4.3.8 Myitems & mytabitems component types.

“Myitems” and “mytabitems” widgets provide easy-to-use interface to build user-defined GUI. Many types of graphic components with rich configuration options (graphic items – ”gitems”) may be added to the widgets.

There are next types gitems:

- “button” – Push button with user-defined skin, size, press or release event option and action type;
- “button-group” – several buttons, grouped in one line, vertical or horizontal orientation;
- “stretch” – Stretch component;
- “led” – Display component, that may show state of some variable selected bit. PLC inputs, outputs or other parameters may be used to display;
- “tablo” – Display element, that connected with some CNC or PLC variable and display its value in given format. May be used to display – Current machine or work position, current speed, Spindle speed, THC arc voltage or any THC working state, current value of Programmable Logic Controller (PLC) ADC, DAC, PWM or any variable etc.
- “kspinbox” – Mixed component, that includes tablo-like and two buttons for increment and decrement of the variable. Variable for tablo and actions for the buttons may be assigned flexible;
- “label” – Static text label;
- “info” – turn on real-time alarm watching fot PLC input selected bit and stop nc-code running, if selected event happens;

4.3.9 Gitem: “button” configuration.

Button item may be flexibly configured with button attributes. Possible attributes for butteon are:

- “where” – name of widget, where the button will be placed;
- “image” – image filename (in a folder “art/buttons”) that will be used as the button skin; there should be three image files for normal, hovered and pressed states of the button; if image=abcdef, there should be files:
 - “adcddef.svg”;
 - “adcddef-hovered.svg”;
 - “adcddef-pressed.svg”
- “height” – height and width of the button in pixels;
- “event” – event, that activate the button action. It's “pressed” (Button Pressed) by default. May be redefined to “released”;
- “action” – action that will be activated if selected event happens;

Possible actions for button components are collected in the next table

Table 3. Possible actions for button (or kspinbox button) components.

Action	Action description
One axis Jogging	
jog-0-plus	Jogging in X axis positive direction

jog-0-minus	Jogging in X axis negative direction
jog-1-plus	Jogging in Y axis positive direction
jog-1-minus	Jogging in Y axis negative direction
jog-2-plus	Jogging in Z axis positive direction
jog-2-minus	Jogging in Z axis negative direction
jog-3-plus	Jogging in A axis positive direction
jog-3-minus	Jogging in A axis negative direction
jog-4-plus	Jogging in B axis positive direction
jog-4-minus	Jogging in B axis negative direction
jog-5-plus	Jogging in C axis positive direction
jog-5-minus	Jogging in C axis negative direction
Two axis jogging	
jog-0-plus-1-plus	Simultaneous jogging in X axis positive direction and Y axis positive direction.
jog-0-plus-1-minus	Simultaneous jogging in X axis positive direction and Y axis negative direction.
jog-0-minus-1-minus	Simultaneous jogging in X axis negative direction and Y axis negative direction.
jog-0-minus-1-plus	Simultaneous jogging in X axis negative direction and Y axis positive direction.
Motion settings	
motion-cuttingspeed-dec	Decrease Cutting speed for 1%
motion-cuttingspeed-inc	Increase Cutting speed for 1%
motion-rapidspeed-dec	Decrease Rapid speed for 1%
motion-rapidspeed-inc	Increase Rapid speed for 1%
motion-linear-jog-speed-dec	Decrease Linear (for XYZ axes) jog speed for 1%
motion-linear-jog-speed-inc	Increase Linear (for XYZ axes) jog speed for 1%
NC Player actions	
player-play	Start running nc-program from current nc-position (tool should be situated on the current nc-position – current nc-position=current tool position).
player-stop	Stop running nc-program. Current tool position will be saved as «current nc-position».
player-play-back	Start running nc-program backward from current nc-position.
player-nc-reset	Reset current program position (nc-position and pointer to nc-line) to the start of the NC program
player-nc-tie	«Tie» current nc-position to current tool position.
player-back-to-path	Start motion of tool to current nc-position
Motion overspeed	

motion-overspeed-inc	Increase motion speed on 1%. Will take effect immediately for all motions (rapid, cutting, jogging etc)
motion-overspeed-dec	Decrease motion speed on 1%. Will take effect immediately for all motions (rapid, cutting, jogging etc)
PLC manipulations	
plc-run	Run given PLC procedure with given parameter.
hw-pwm-inc	Increase on 1% value of PLC PWM
hw-pwm-dec	Decrease on 1% value of PLC PWM
hw-dac-inc	Increase on 1% value of PLC DAC
hw-dac-dec	Decrease on 1% value of PLC DAC
THC manipulations	
thc-arc-voltage-ref-dec	Decrease on 0,1V value of THC Reference Arc Voltage
thc-arc-voltage-ref-inc	Increase on 0,1V value of THC Reference Arc Voltage
thc-jog-speed-dec	Decrease on 5% THC Jogging speed (if THC controls Z-motion via built-in DC motor driver/amplifier)
thc-jog-speed-inc	Increase on 5% THC Jogging speed (if THC controls Z-motion via built-in DC motor driver/amplifier)
thc-jog-pos	Turn On/Off Z-axis jogging through THC controller in positive direction
thc-jog-neg	Turn On/Off Z-axis jogging through THC controller in negative direction
Fake	
fake	Fake action (where no needs to action)
CNC controller variables and Devices variables manipulation	
mydev-var	Set selected variable of selected device to given value.
direct-set-cnc-var	Setting CNC variable to given value
Direct Run	
direct-run	Direct running g-code lines, given as parameter of the action
Widgets manipulations	

To be continued... (2011-0601 8:40 pm msk)