Flex GUI

John Thornton

Nov 10, 2025

CONTENTS:

CHAPTER

ONE

FLEX GUI DESCRIPTION

Flex GUI is a flexible GUI that can be customized to suit your needs.

- Uses stock Qt Designer 5 or 6
- Widget names are used to connect controls to the correct code
- Widgets are auto-discovered at startup
- Special widgets only need Dynamic Properties to be discovered and created
- Your GUI can have exactly the controls and labels you want
- You can create and use your own style sheet, changing fonts, colors, etc.
- All Flex GUI configuration is done in the .ini file
- Flex GUI remembers the size and position of your GUI

For more information on Dynamic Properties see *Dynamic Properties*

1.1 Features

- Status Labels
- Menu Actions
- Button Actions
- Button Controls
- Plotter
- MDI Input and Buttons
- Spindle Controls
- Probing with Spindle Safety
- Tool Change Controls
- Coordinate System Controls
- HAL Buttons, Spinboxes, Sliders, Labels and LCDs
- HAL I/O Controls
- Touch Screen Controls and Popups

1.2 Acronyms

The following acronyms will be found in this document

- NC = *Numerical Control* code includes G, M and O codes
- G = *Geometric* code is used for geometric movements
- M = *Miscellaneous* code is used for non-movement functions
- O = *Organizing* code is used in CNC programs to control flow

CHAPTER

TWO

INSTALLING

2.1 Install with Apt

The advantage of using apt to install Flex GUI is when a new version of Flex GUI is released apt will know a new version is available when you run *sudo apt update*. This will allow you to install the new version of Flex GUI along with other Debian software.

Note: on a brand-new Rpi5 using the linuxcnc.iso, use apt *–fix-broken install* to install several qt6 sub-dependencies

The first command will ask for your password. Neither command will print anything in the terminal.

For a PC to create an apt sources file for Flex GUI copy and paste this command in a terminal

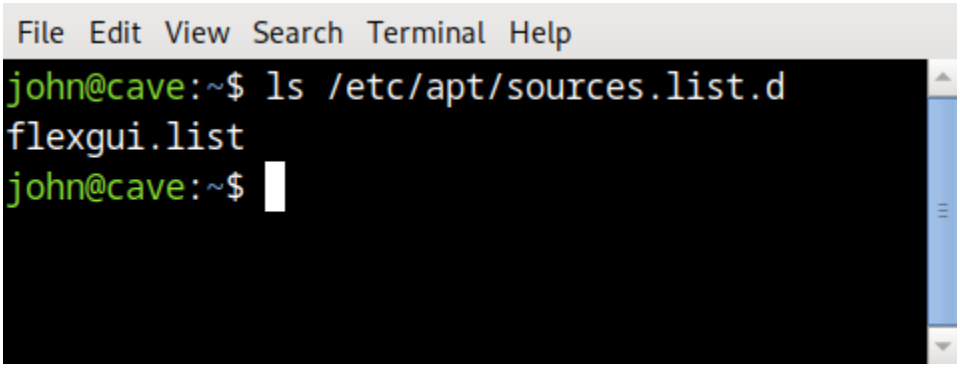
```
echo 'deb [arch=amd64] https://gnipsel.com/flexgui/apt-repo stable main' | sudo tee /etc/
↪apt/sources.list.d/flexgui.list
```

For a Raspberry Pi create an apt sources file for Flex GUI copy and paste this command in a terminal

```
echo 'deb [arch=arm64] https://gnipsel.com/flexgui/apt-repo stable main' | sudo tee /etc/
↪apt/sources.list.d/flexgui.list
```

To check the above command worked you can list the file with this command

```
ls /etc/apt/sources.list.d
```



Next get the public key for Flex GUI and copy it to trusted.gpg.d

```
sudo curl --silent --show-error https://gnipsel.com/flexgui/apt-repo/pgp-key.public -o /
↪etc/apt/trusted.gpg.d/flexgui.asc
```

If curl is not installed you can install it with the following command

```
sudo apt install curl
```

Next update apt

```
sudo apt update
```

If you have Flex GUI installed you can see what packages can be upgraded with the following command

```
apt list --upgradable
```

If Flex GUI is not installed you can install it with the following command

```
sudo apt install flexgui
```

2.2 Install the Deb

You can still download the deb from github and install with gdebi if that works better for you. If you don't have an internet connection this is the best way to install Flex GUI

Note: Between releases the deb will have the latest bug fixes

Installing Flex GUI Tutorial

Download the latest deb file from >HERE<.

If the link is not clickable, copy and paste the following URL into your browser

```
https://github.com/jethornton/flexgui/releases
```

~amd64.deb is for PC's and ~arm64.deb is for Raspberry Pi.

Select the latest release and click on the .deb to start a download.

Right click on the deb file and select *Open with GDebi Package Installer*. If that option is not there then GDebi is not installed, open a terminal and run this command to install it:

```
sudo apt install gdebi
```

An alternative is to install from the terminal outright using *dpkg*. Make sure the version number is correct for the deb you have the following command may be an older version.

```
sudo dpkg -i flexgui_1.1.0_amd64.deb
```

2.3 Build and Install

If you plan on changing code in Flex GUI you can clone the repository and build the deb after making changes. The target directory is optional.

```
git clone https://github.com/jethornton/flexgui.git (target/directory)
```

Before building the deb you will need to install some programs that do the building. Open a terminal and run the following to install devscripts

```
sudo apt install devscripts
```

Open a terminal in the top most flexgui directory and use this command to build a deb file.

```
debuild -us -uc
```

2.4 Copy Example Files

After installing Flex GUI, a menu item *Copy Flex Examples* is added to the *CNC* menu. This will copy the Flex GUI example files to ~/linuxcnc/configs/*flex_examples*.

Note: After updating the Flex GUI some examples may have changed. To get a fresh copy of the examples delete the *linuxcnc/configs/flex_examples* or rename it.

CHAPTER

THREE

QT DESIGNER

3.1 Installing the Qt Designer

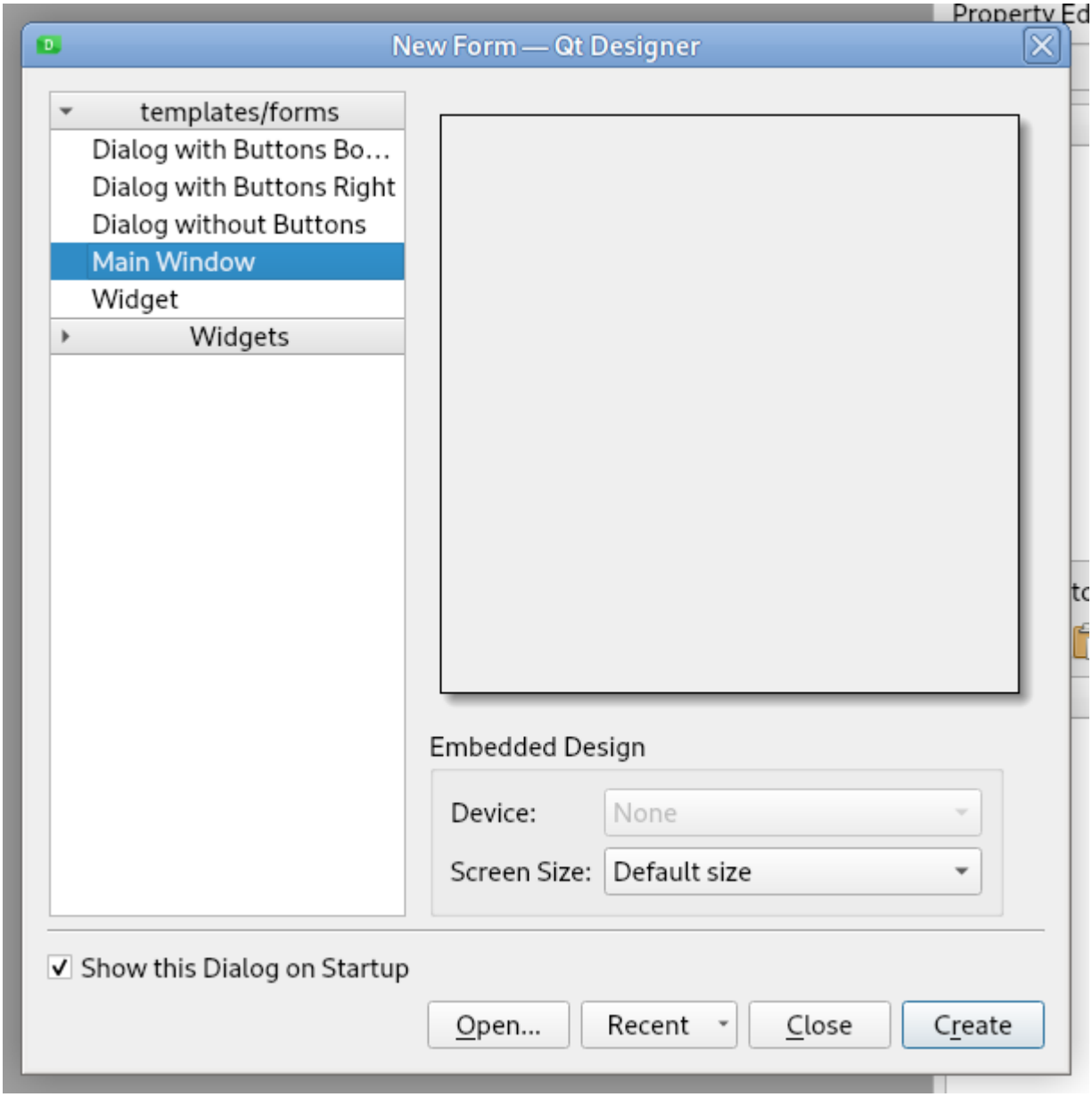
In a terminal, install Qt Designer 5 with

```
sudo apt install qttools5-dev-tools
```

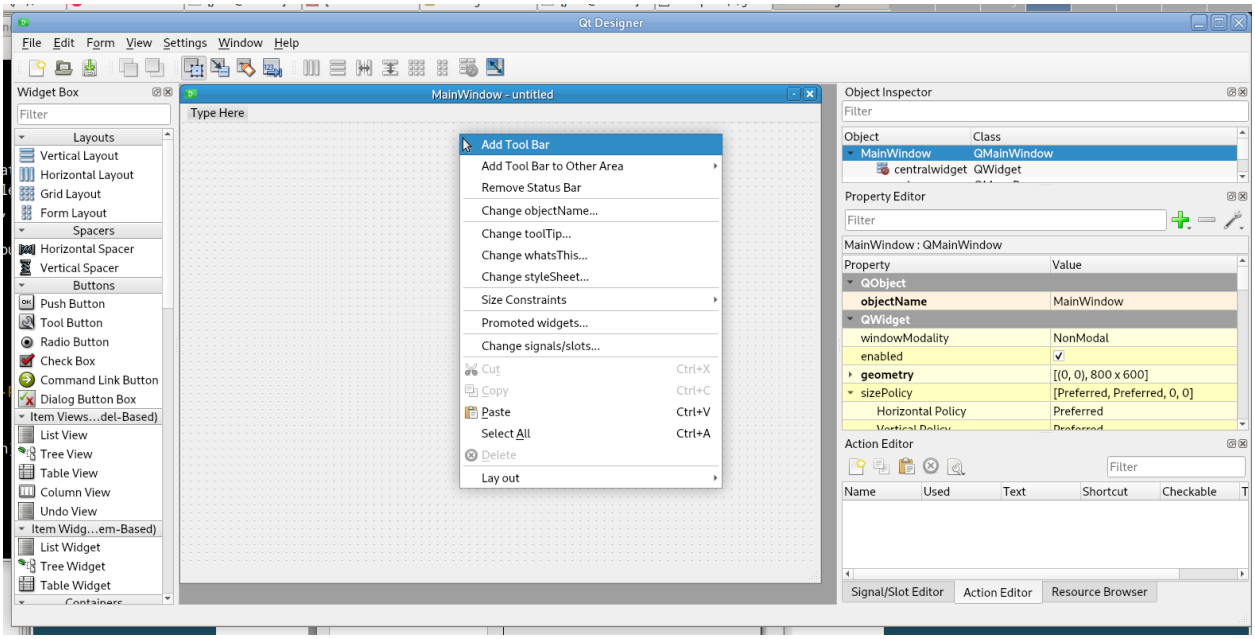
Note: The Qt6 Designer is not required nor better; Qt5 Designer is fine.

3.2 Building a GUI

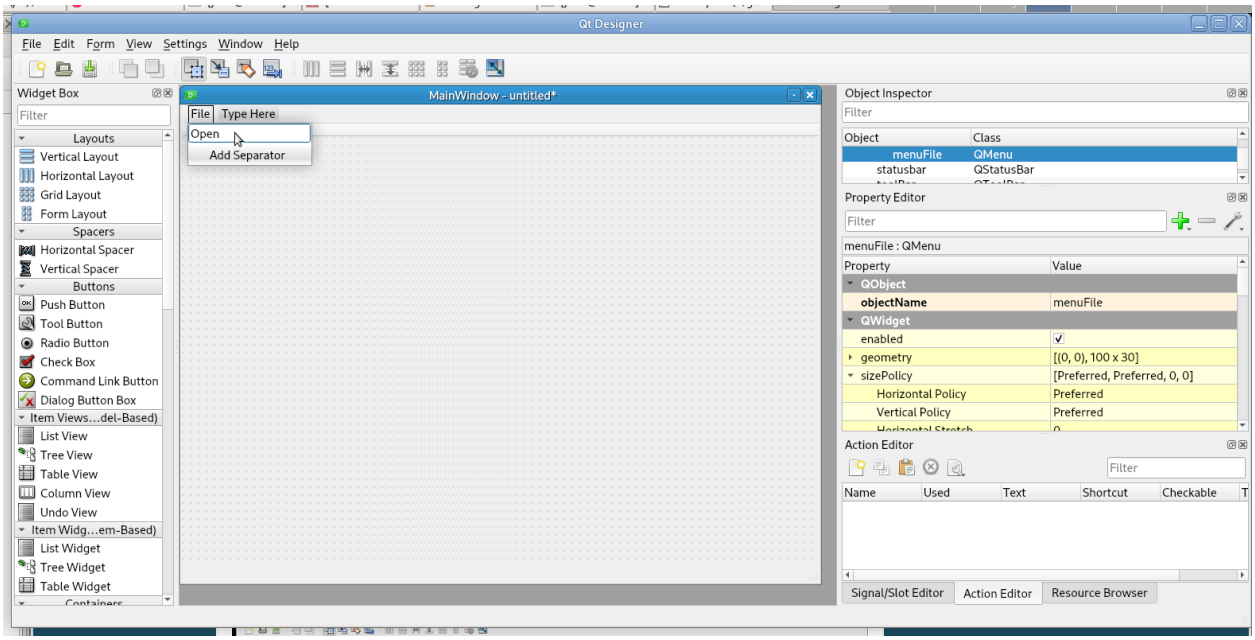
Run the Qt Designer from the Applications > Programming menu and create a new *Main Window*



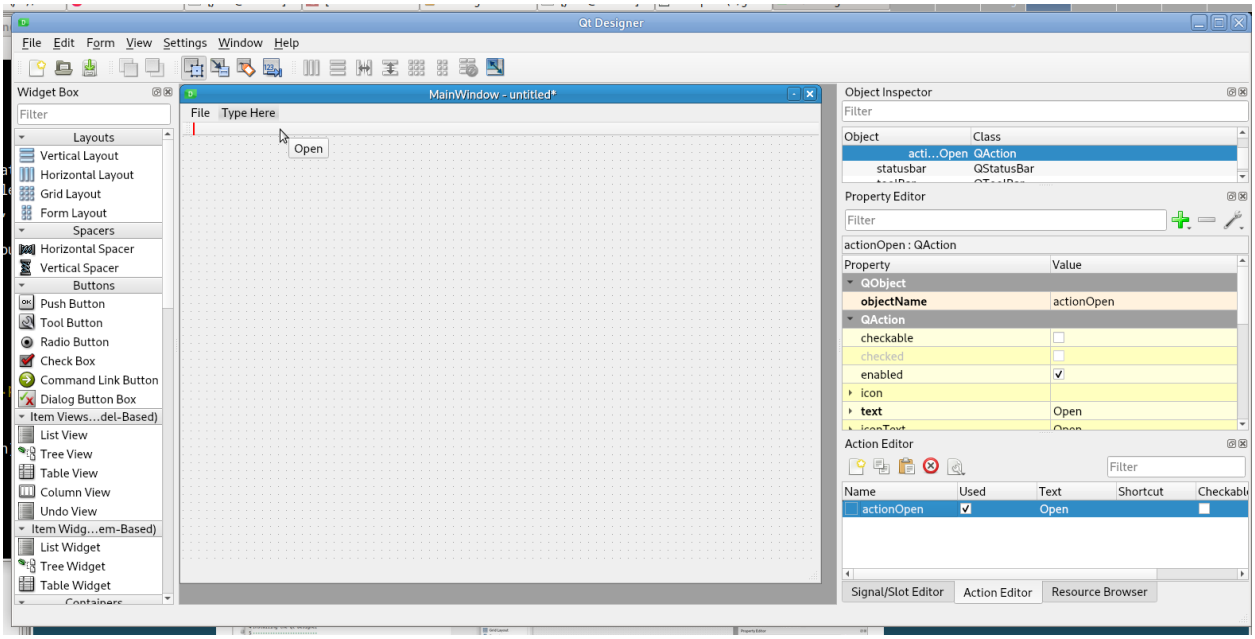
To add a Tool Bar, right click on the main window and select *Add Tool Bar*



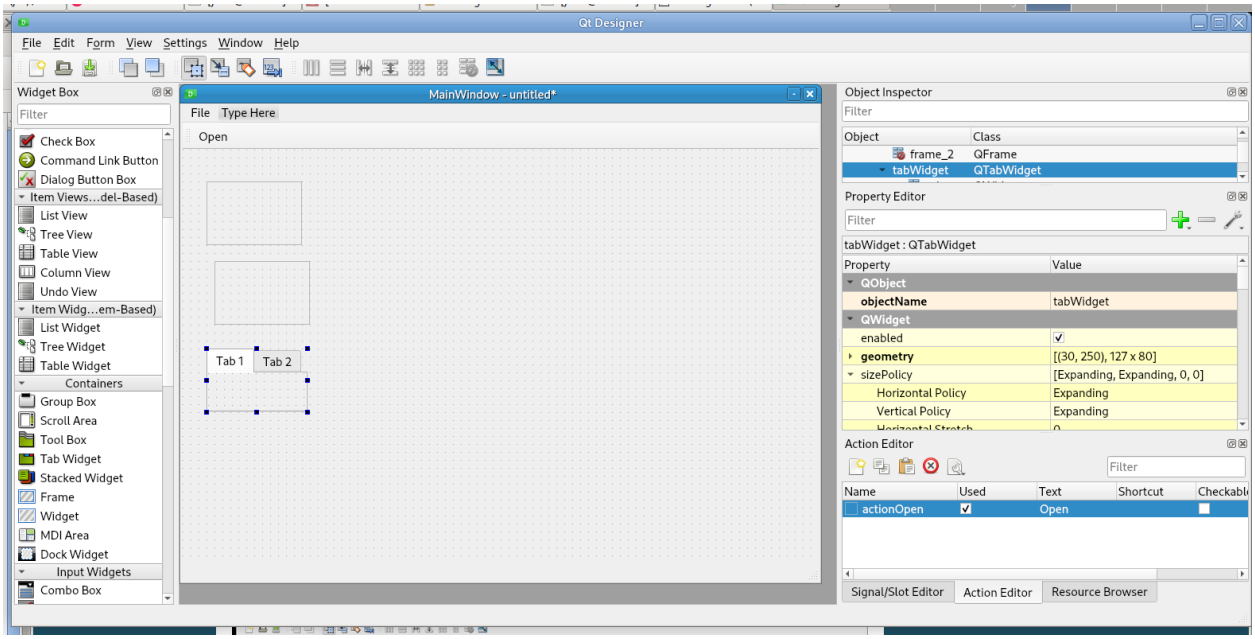
To add a Menu, type in the menu area and press enter



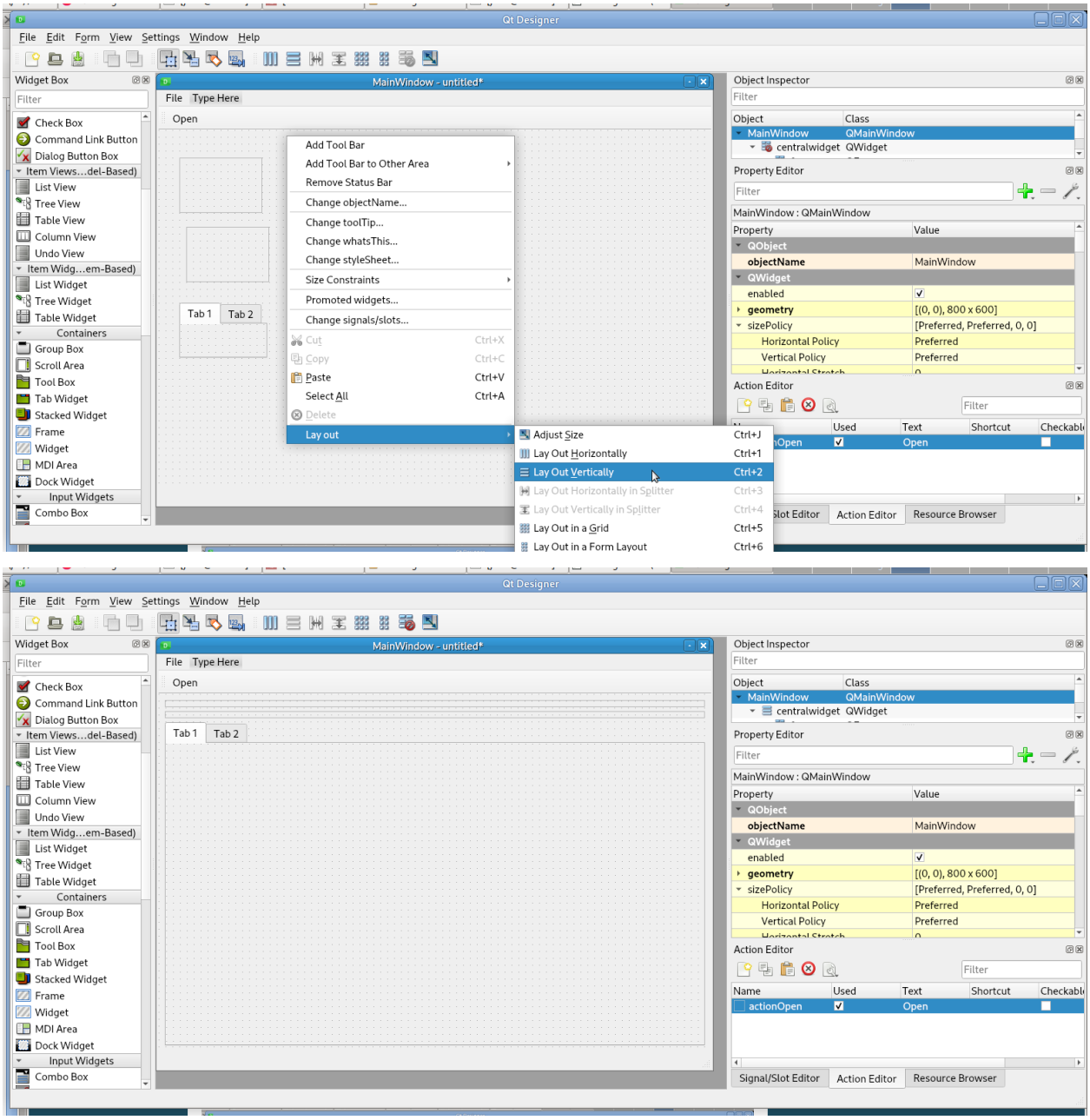
When you create a Menu item it creates an action; this action can be dragged to the Tool Bar to create a tool bar button



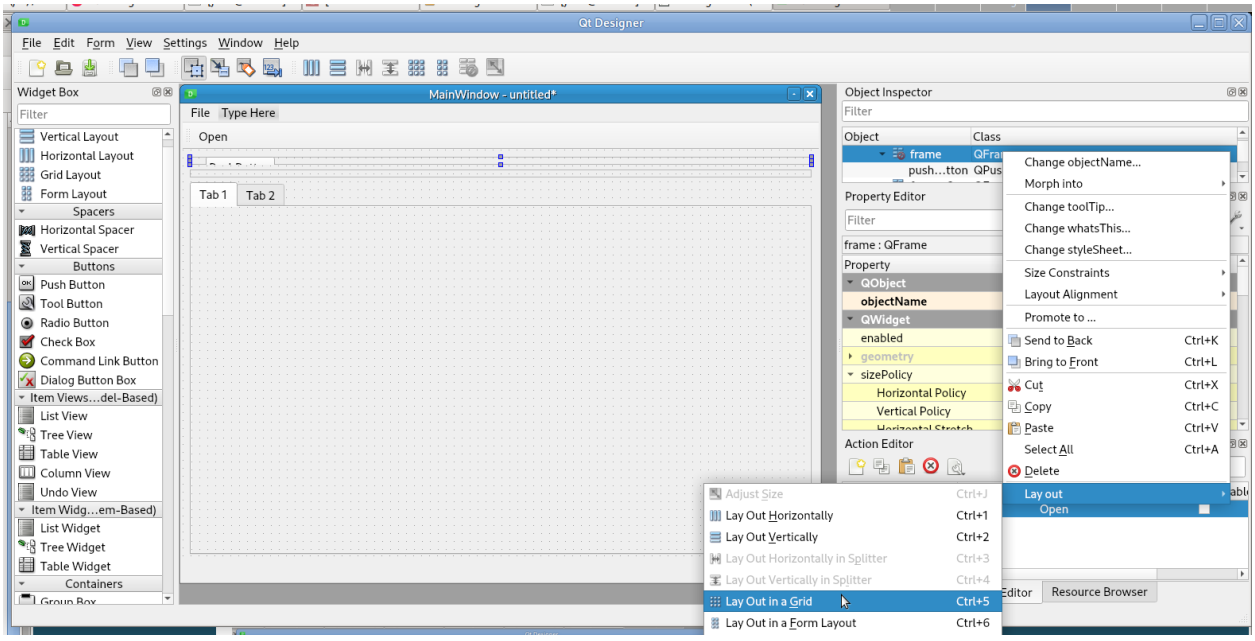
Adding items from the Widget Box is drag-and-drop. To create a basic layout from Containers, add two Frames and a Tab Widget

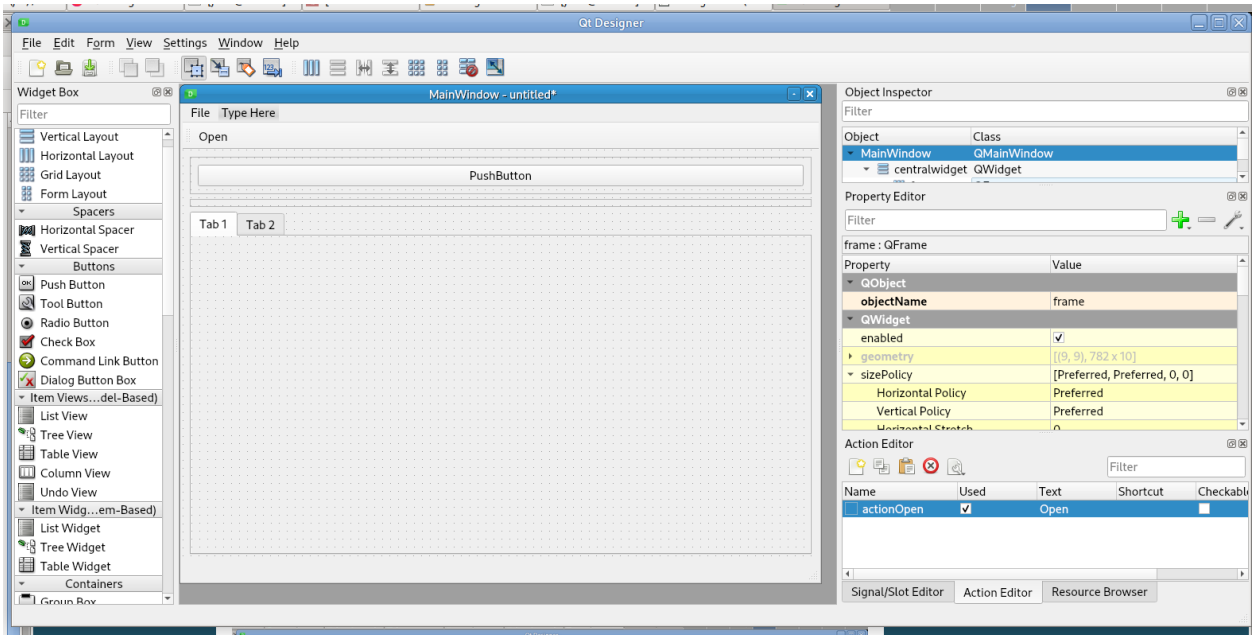


Right click in the QMainWindow and select Lay out –> Lay out Vertically



Add a Push Button to the QFrame, then right click on the frame or the QFrame in the Object Inspector and set the lay out to *grid*





After dragging a widget into the window, make sure you use the correct objectName for that widget. For example the E-Stop button is called estop_pb.

Note: Each object name must be unique; designer will not allow duplicate names.

Save the GUI in the configuration directory alongside your .ini file.

You can start Qt5 Designer from a terminal with *designer &* which spawns a new process (gives you back the terminal prompt.)

Note: There is an documented issue with Qt5 Designer and bold fonts not appearing properly.

3.3 Qt6 Designer

Qt6 Designer can be installed from a terminal with

```
sudo apt install designer-qt6
```

To run Qt6 Designer you must use the full path to the executable

```
/usr/lib/qt6/bin/designer
```

CHAPTER

FOUR

BUILDING A GUI

After installing Flex GUI if you have not copied the examples from the CNC menu select Copy Flex Examples. This will put the Flex examples in the directory linuxcnc/configs/flex_examples.

The starters have all the files needed to run a simulation without any coding needed. The starters have a very simple GUI to start, just enough to show you that they work.

4.1 Copy a Starter

From the linuxcnc/configs/flex_examples/starters copy one of the starter types to the linuxcnc/configs directory.

- Rename the directory to the name of your choice.
- Rename the .ui and .ini files to the name of your choice.
- Edit the .ini file and change the GUI to the name of your .ui file.
- Edit the MIN_LIMIT and MAX_LIMIT for each axis and joint to match your machine
- From the CNC menu select LinuxCNC and pick your configuration, check Create Desktop Shortcut then click OK to run your configuration.

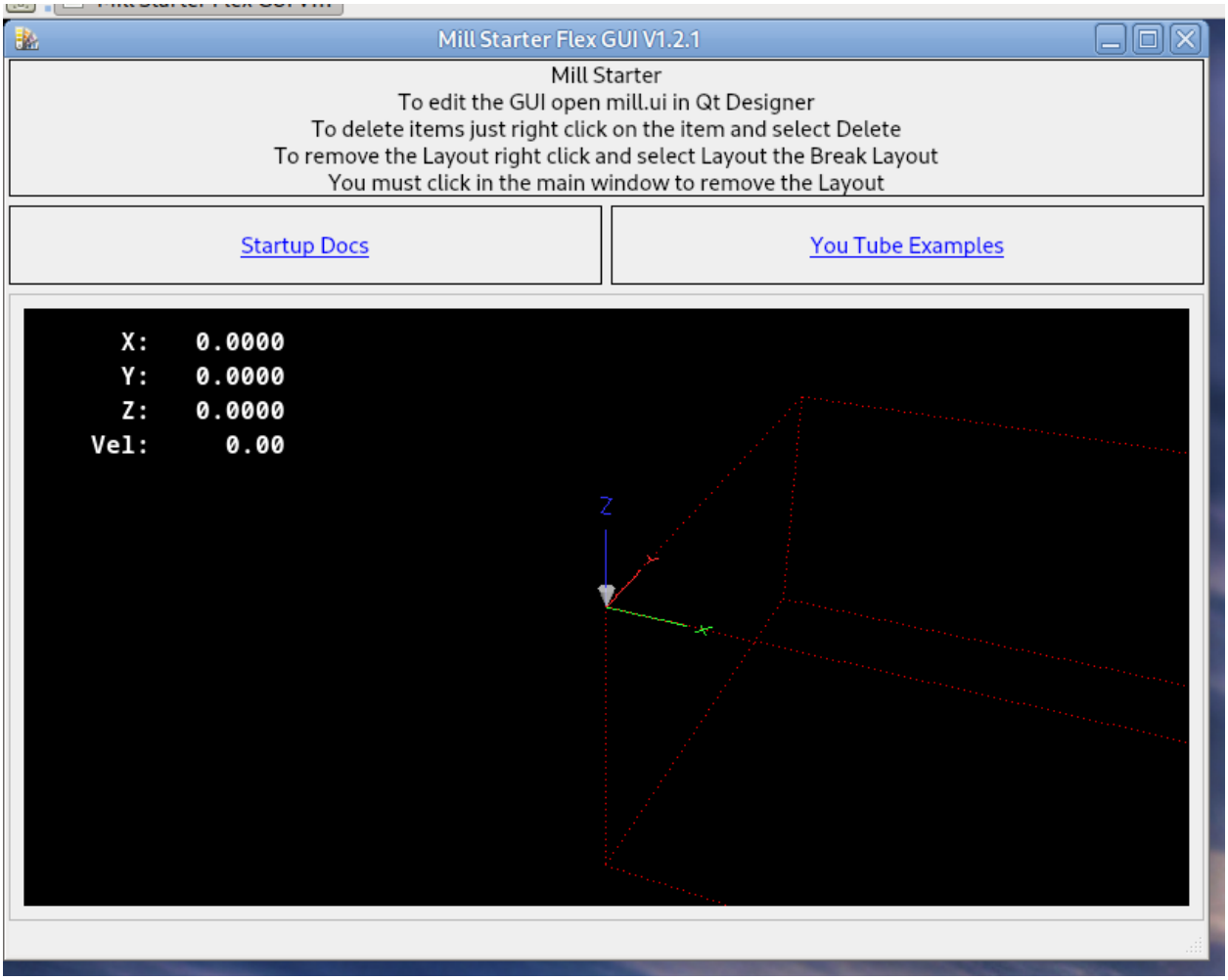


Fig. 1: Mill Starter Example

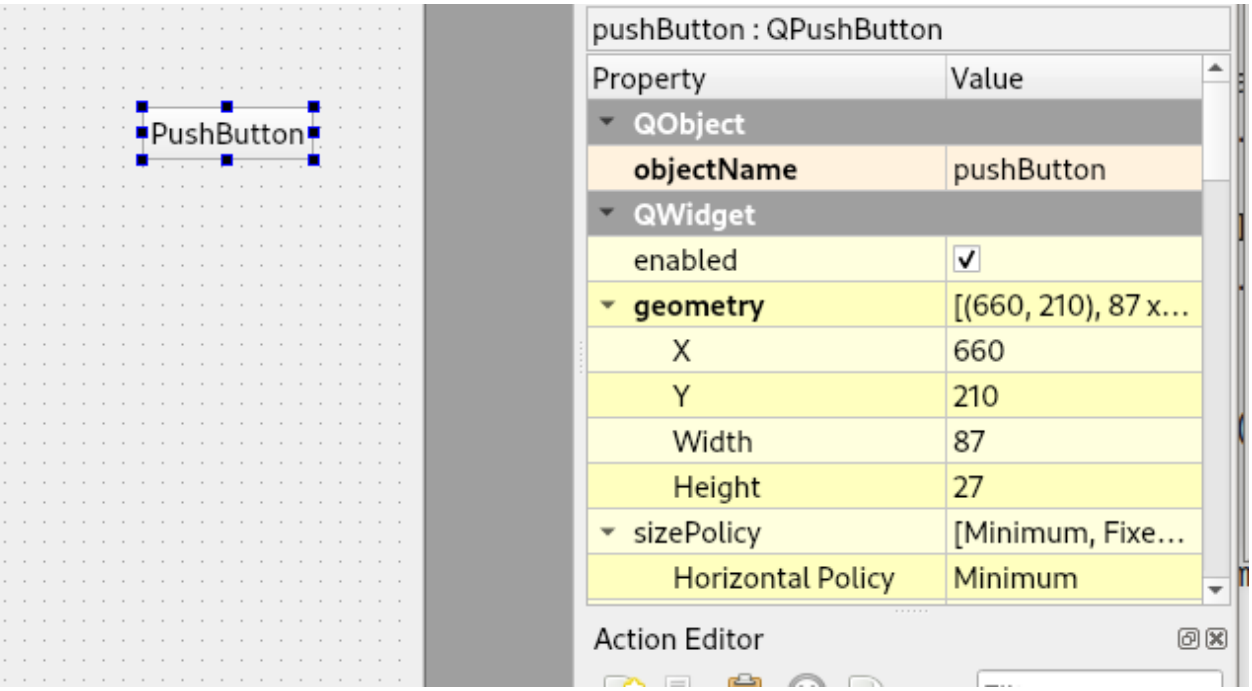
CHAPTER

FIVE

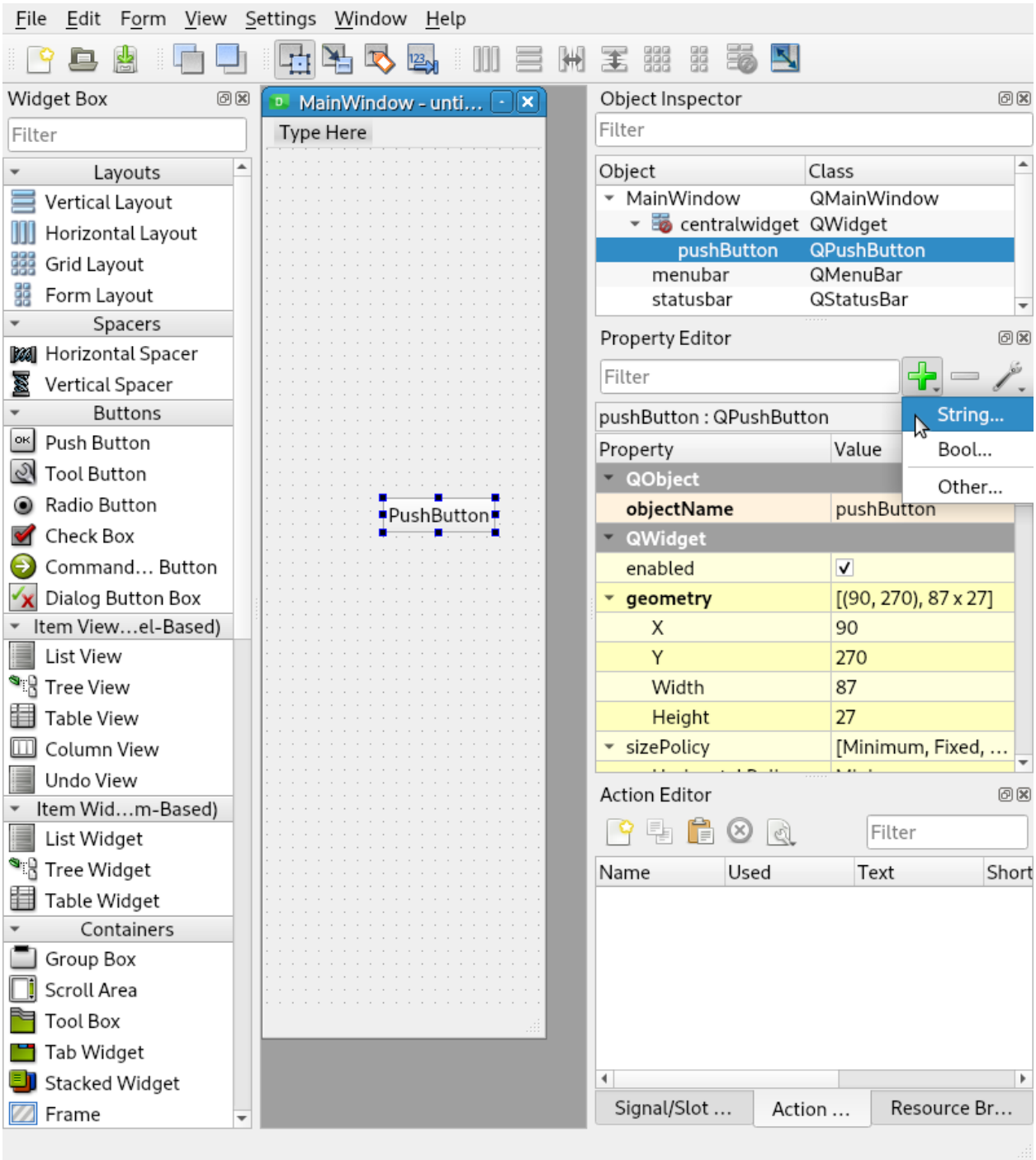
DYNAMIC PROPERTIES

Flex GUI uses a lot Dynamic Properties to customize widget behavior and add more functionality to a widget.

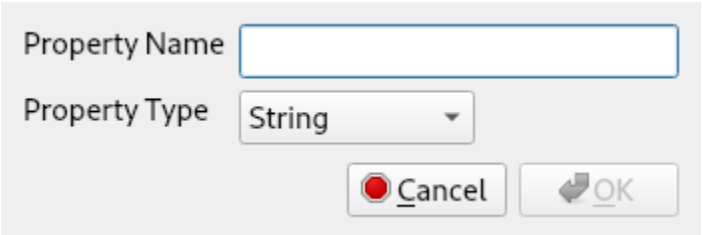
To Create a Dynamic Property the first thing you do is select the widget.



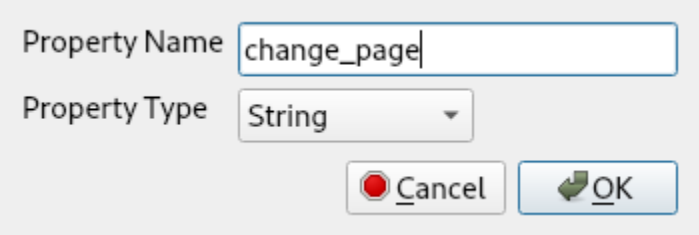
Next left click on the green plus sign in the Property Editor.



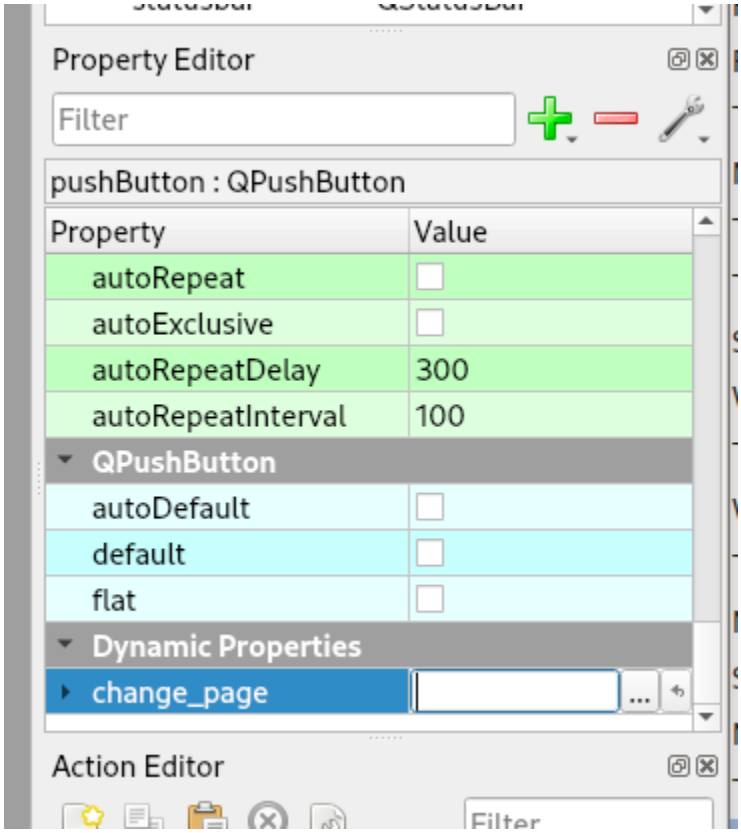
Typically Flex GUI uses a string type Dynamic Property with some exceptions.



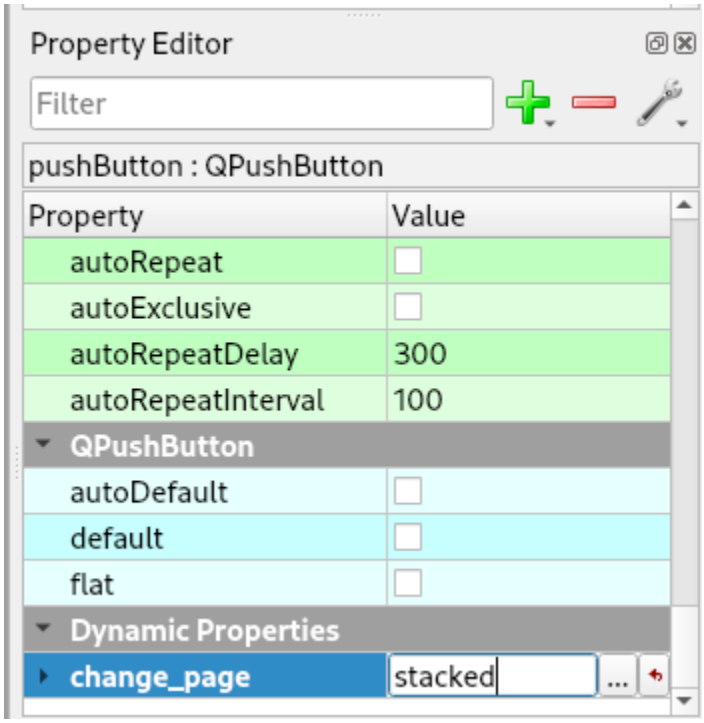
Next enter the Dynamic Property Name, which must be exactly like shown in the documents.



Press OK then the Dynamic Property will show up in the Property Editor.



Now you can enter the Value for that Dynamic Property.



Note: The Dynamic Properties section does not show up in the Property Editor until you create a Dynamic Property.

CHAPTER

SIX

INI SETTINGS

Video Tutorial

Note: The following Flex GUI settings are all located in the [DISPLAY] section of your LinuxCNC .ini file.

6.1 [EMC]

The optional DEBUG key in the [EMC] section can be used to show various levels of debug information when running in a terminal. The DEBUG key is not required.

```
DEBUG = option number
```

Debug Options

```
EMC_DEBUG_CONFIG               2
EMC_DEBUG_VERSIONS             8
EMC_DEBUG_TASK_ISSUE          10
EMC_DEBUG_NML                 40
EMC_DEBUG_MOTION_TIME         80
EMC_DEBUG_INTERP             100
EMC_DEBUG_RCS                200
EMC_DEBUG_INTERP_LIST        800
EMC_DEBUG_IOCONTROL         1000
EMC_DEBUG_OWORD             2000
EMC_DEBUG_REMAP             4000
EMC_DEBUG_PYTHON            8000
EMC_DEBUG_NAMEDPARAM       10000
EMC_DEBUG_GDBONSIGNAL      20000
EMC_DEBUG_STATE_TAGS       80000
EMC_DEBUG_UNCONDITIONAL 40000000
EMC_DEBUG_ALL           7FFFFFFF
```

6.2 [DISPLAY]

6.2.1 Flex GUI

To use the Flex GUI (as opposed to Axis or others), change the DISPLAY value to

```
DISPLAY = flexgui
```

If no GUI is specified then the default GUI will be used.

Note: Any Flex GUI .ui and .qss files must be in the same LinuxCNC configuration directory as the .ini file.

To use your .ui file (created with Qt Designer), add a GUI key to the .ini with its *filename*:

```
GUI = my-file-name.ui
```

6.2.2 Jog Increments

The following settings can be used in the [DISPLAY] section of the ini file to preset jog items

```
INCREMENTS = 0.100, 0.010, 0.001
or
INCREMENTS = 1 inch, 0.5 in, 1 cm, 1 mm
MIN_LINEAR_VELOCITY = 0.1
MAX_LINEAR_VELOCITY = 1.0
DEFAULT_LINEAR_VELOCITY = 0.2
```

Warning: [DISPLAY] INCREMENTS must be a comma seperated list or it will be ignored.

Note: Jog incremnts can have unit labels, the following are valid unit labels cm, mm, um, inch, in or mil. If no unit labels are found the the configuration units are used.

6.2.3 Startup File

To automatically open a NC file on startup, add the OPEN_FILE key with any valid path. Use ~/ as a shortcut to the users home directory. Use ./ to indicate that the file is in the configuration directory

```
Full Path to the file
OPEN_FILE = /home/john/linuxcnc/configs/myconfig/welcome.ngc
or use the ~ for the users home directory
OPEN_FILE = ~/linuxcnc/configs/flex_examples/probe_sim/square.ngc
or use the ./ to use the current configuration directory
OPEN_FILE = ./welcome.ngc
or use the ../ to use the parent directory of the configuration
OPEN_FILE = ../welcome.ngc
```

6.2.4 File Location

Likewise, to specify a default location for NC files, add the PROGRAM_PREFIX item.

```
PROGRAM_PREFIX = /home/john/linuxcnc/configs/myconfig
or
PROGRAM_PREFIX = ~/linuxcnc/configs/flex_examples/probe_sim
or
PROGRAM_PREFIX = ./
or
PROGRAM_PREFIX = ../
```

6.2.5 Tool Table Editor

To specify a different tool table editor add an entry to the [DISPLAY] section. If no entry is found then the default tool editor is used

```
TOOL_EDITOR = tooledit
```

To control the columns displayed by the default tool editor add any of the valid column specifiers separated by a space.

```
TOOL_EDITOR = tooledit x y z a b c u v w diam front back orien
```

If no entry is found then the axes in the configuration and diameter are shown. Tool, Pocket and Comment are always shown.

6.2.6 File Extensions

The keyboard file dialog defaults to **.ngc* and this ignores case. To specify the file extensions you want the file dialog to show, add an EXTENSIONS key with the desired extensions separated by a comma. The extensions must be in the format **.ext* with the asterisk and dot

```
EXTENSIONS = `*.nc`, `*.G-code`, `*.ngc`, `*.txt`
```

6.3 [FLEXGUI]

6.3.1 Themes

Themes are just style sheets that get applied to the widgets. The theme files are in the themes directory of the example files if you want to copy and customize one of the themes.

```
blue.qss
blue-touch.qss
dark.qss
dark-touch.qss
keyboard.qss
touch.qss
```

To use a built-in theme with no color changes add one of the following to the [FLEXGUI] section of the ini file.

```
THEME = touch
THEME = keyboard
```

To use a built in theme with coloring add one of the following to the [FLEXGUI] section of the ini file.

```
THEME = blue
THEME = blue-touch
THEME = dark
THEME = dark-touch
```

Note: Touch themes use tabs set to South for rounding and non touch use tabs set to North.

Note: THEME is checked first then QSS so the first entry found is used.

To use a custom .qss style sheet you created add the name of the stylesheet to the QSS option in [FLEXGUI] section of the ini file.

```
QSS = name_of_stylesheet.qss
```

For more information on style sheets see *StyleSheet*

6.3.2 Resource File

To use a .py resource file (to add images to buttons with your qss stylesheet) place the .py resource file in the configuration directory and add the following line to the .ini file

```
RESOURCES = resources.py
```

See the section on Resources for more info.

6.3.3 Screen Size

To control the initial size of the screen, add of the following values.

```
SIZE = minimized
SIZE = normal
SIZE = maximized
SIZE = full
```

Note: the values are case sensitive

Warning: Full size screen does not have any window controls. Make sure there is a way to close the GUI like an Exit button or you may not be able to close the application. As a last-resort, pressing ALT-F4 will close it.

6.3.4 Plotter

The plotter background color can be set in the [FLEXGUI] section of the ini. The value is the Red,Green,Blue color numbers from 0 to 1 with no space. So an entry of 0.0,0.0,0.0 is black and 1.0,1.0,1.0 is white. Use a RGB 0-1 Color Picker to select the RGB values.

```
[FLEXGUI]
PLOT_BACKGROUND_COLOR = 0.0,0.0,0.0
```

The plotter orientation can be set to one of the following x, x2, y, y2, z, or p.

```
[DISPLAY]
VIEW = x
```

The font size for the plotter can be set in the ini by adding the following to the FLEXGUI section. The font size must be an integer.

```
[FLEXGUI]
DRO_FONT_SIZE = 12
```

6.3.5 Colors

The E-Stop can have a static color for Open and Closed.

The Power Button can have a static color for Off and On.

Create a key in the ini file called FLEXGUI and use the following to control the static color of these items. The value can be any valid color specification in the RGB, RGBA or Hex color format.

```
[FLEXGUI]
ESTOP_OPEN_COLOR = 128, 255, 128
ESTOP_CLOSED_COLOR = 255, 77, 77
POWER_OFF_COLOR = 255, 128, 128
POWER_ON_COLOR = #00FF00
PROBE_ENABLE_ON_COLOR = 255, 0, 0, 255
PROBE_ENABLE_OFF_COLOR = 0, 125, 0, 125
```

Note: Color pairs need to have both colors specified or the color will only toggle once.

Another way to achieve this is is via adding and editing a .qss stylesheet file. See the *StyleSheet* section for more info.

6.3.6 LED Defaults

LED buttons can have defaults set in the ini file. This makes it easier to have consistent LED size, position and colors. These options go in the [FLEXGUI] section.

The color options can be specified using HEX, RGB or RGBA.

Valid RGB(A) Red, Green, Blue (Alpha) values are 0 to 255.

Valid HEX values are #000000 to #ffffff

In PyQt6 the Alpha channel is 0 to 255. 0 represents a fully transparent color, while 255 represents a fully opaque color. If Alpha is ommitted then it's set to fully opaque or 255.

The Diameter and Offset values are whole numbers only.

```
[FLEXGUI]
LED_DIAMETER = 15
LED_RIGHT_OFFSET = 5
LED_TOP_OFFSET = 5
LED_ON_COLOR = 0, 255, 0
LED_OFF_COLOR= 125, 0, 0, 255
```

For more information on LED buttons see *LED Buttons*

6.3.7 Touch Screens

Options for touch screen users.

Set the touch screen file chooser to automatically adjust the width by adding the following to the FLEXGUI section.

```
[FLEXGUI]
TOUCH_FILE_WIDTH = True
```

Add popup keypad to all spin boxes.

```
[FLEXGUI]
TOUCH_SPINBOX = True
```

To add a manual tool change popup to add the following to the ini file in the [FLEXGUI] section.

```
[FLEXGUI]
MANUAL_TOOL_CHANGE = True
```

CHAPTER

SEVEN

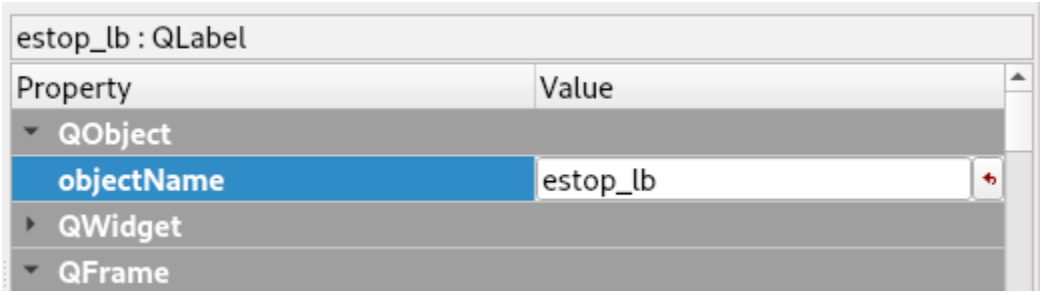
STATUS LABELS

Status Labels Tutorial

Status labels are created using a QLabel and setting the *Object Name*. Status labels come in two forms. A single-status-label like *Machine Status* only contains one piece of information, such as *OFF*, *RUN*, etc.

A multiple-status-label like the *axis* or *joint* dictionaries have multiple items and displays for all joints. Multiple-status-labels use a number identifier to select the axis, joint, or spindle information wanted.

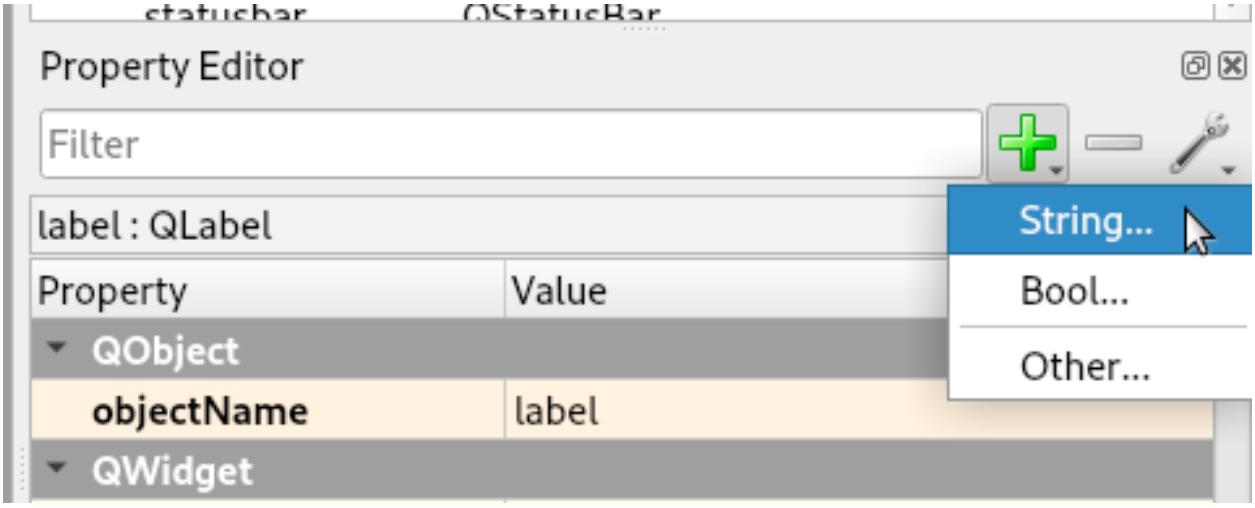
When creating a status label, set the *objectName* to the status you want to display



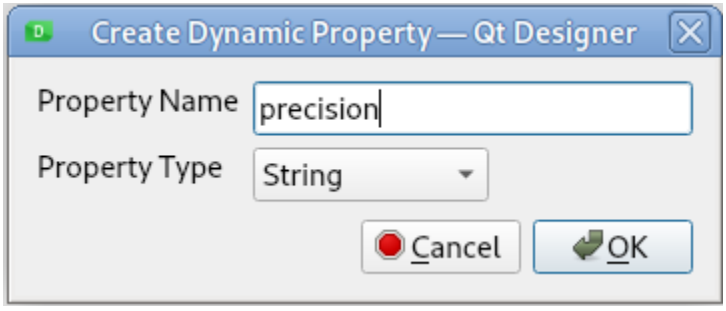
7.1 Precision

Labels that return float values default to 3 decimal places for metric and 4 for inch.

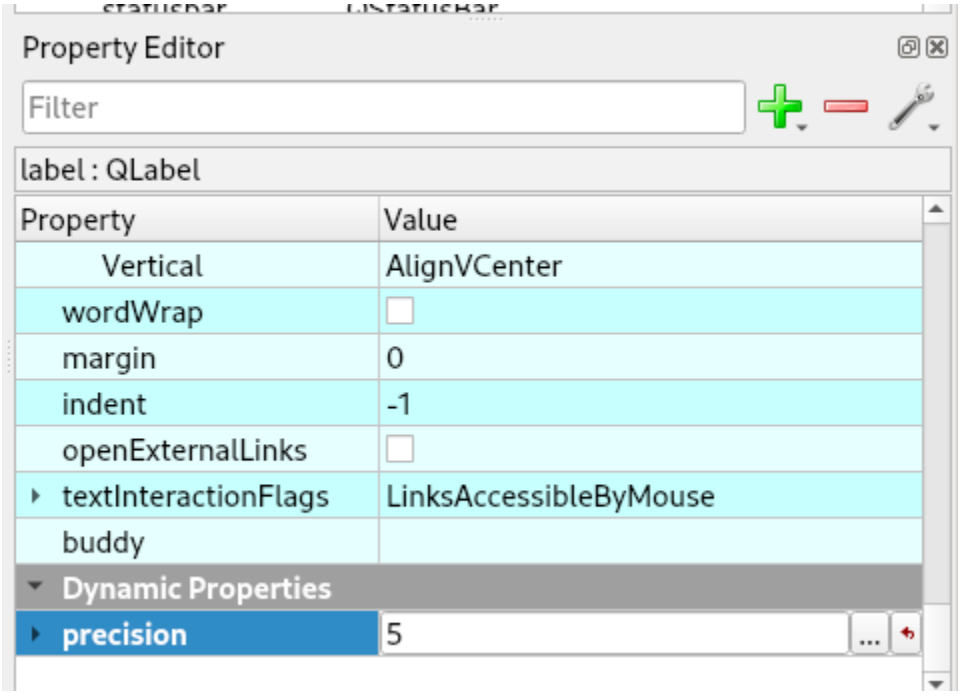
To override the default, select the label then click on the Green Plus sign in the Property Editor to add a Dynamic Property and select String. See *Dynamic Properties*



Set the Property Name to *precision*:



Set the Value to how many decimal places you want for that status label



For more information about status labels read the LinuxCNC Python Interface Status Attributes

7.2 Status Labels

Status Labels are created by adding a QLabel and changing the Object Name to one of the following. Some status labels use a dictionary to look up the name instead of displaying the integer the name is displayed.

```
acceleration_lb - returns float
reflects the INI entry [TRAJ]DEFAULT_LINEAR_ACCELERATION, if that entry is not
found it returns 1e+99

active_queue_lb - returns integer
number of motions blending

actual_position_lb - returns tuple of floats
current trajectory position, (x y z a b c u v w) in machine units
```

See the *Axis Position* labels for individual axis positions.

```
adaptive_feed_enabled_lb - returns boolean
status of adaptive feedrate override

ain_lb - returns tuple of floats
current value of the analog input pins

angular_units_lb - returns float - precision can be set
machine angular units per deg, reflects [TRAJ]ANGULAR_UNITS

aout_lb - returns tuple of floats
current value of the analog output pins

axis_lb - returns tuple of dicts
reflecting current axis values

axis_mask_lb - returns integer
sum of the axes by [TRAJ]COORDINATES X=1, Y=2, Z=4, A=8, B=16, C=32, U=64, V=128, W=256

block_delete_lb - returns boolean
block delete current status

call_level_lb - returns integer
current subroutine depth. - 0 If not in a subroutine

command_lb - returns string
currently executing command

current_line_lb - returns integer
currently executing line

current_vel_lb - returns float - precision can be set
current velocity in user units per second

cycle_time_lb - returns float - precision can be set
thread period

```

(continues on next page)

(continued from previous page)

```
debug_lb - returns integer
debug flag from the INI file

delay_left_lb - returns float - precision can be set
remaining time on G4 dwell command in seconds

din_lb - returns tuple of integers
current value of the digital input pins

distance_to_go_lb - returns float - precision can be set
remaining distance of current move as reported by trajectory planner

echo_serial_number_lb - returns integer
The serial number of the last completed command sent by a UI to task.
All commands carry a serial number. Once the command has been executed,
its serial number is reflected in echo_serial_number

enabled_lb - returns boolean
trajectory planner enabled flag

estop_lb - returns integer
Returns either STATE_ESTOP = 1) or not = 0)

exec_state_lb - returns integer that is used to lookup the state name.
task execution state. One of EXEC_ERROR = 1, EXEC_DONE = 2,
EXEC_WAITING_FOR_MOTION = 3, EXEC_WAITING_FOR_MOTION_QUEUE = 4,
EXEC_WAITING_FOR_IO = 5, EXEC_WAITING_FOR_MOTION_AND_IO = 7,
EXEC_WAITING_FOR_DELAY = 8, EXEC_WAITING_FOR_SYSTEM_CMD = 9,
EXEC_WAITING_FOR_SPINDLE_ORIENTED = 10).

feed_hold_enabled_lb - returns boolean
enable flag for feed hold

feed_override_lb - returns boolean
 enable flag for feed override

file_lb - returns string
currently loaded G-code filename with path

flood_lb - returns integer that is used to lookup the state of OFF or ON
Flood status, either FLOOD_OFF = 0) or FLOOD_ON = 1)

g5x_index_lb - returns integer that is used to lookup the coordinate system name
currently active coordinate system, G54=1, G55=2 etc

g5x_offset_lb - returns tuple of floats
offsets of the currently active coordinate system X, Y, Z, U, V, W, A, B, C

gcodes_lb - returns tuple of integers
Active G-codes for each modal group.
The integer values reflect the nominal G-code numbers multiplied by 10.
(Examples: 10 = G1, 430 = G43, 923 = G92.3)
```

(continues on next page)

(continued from previous page)

```
homed_lb - returns tuple of integers
currently homed joints, 0 = not homed, 1 = homed
```

See the *Axis Homed* labels for individual axis home status.

```
ini_filename_lb - returns string
path to the INI file passed to linuxcnc

inpos_lb - returns boolean
machine-in-position flag

input_timeout_lb - returns boolean
flag for M66 timer in progress

interp_state_lb - returns integer that is used to lookup the state name
current state of RS274NGC interpreter. One of INTERP_IDLE = 1,
INTERP_READING = 2, INTERP_PAUSED = 3, INTERP_WAITING = 4

interpreter_errcode_lb - returns integer that is used to lookup the error name
current RS274NGC interpreter return code
        INTERP_OK = 0,
        INTERP_EXIT = 1,
        INTERP_EXECUTE_FINISH = 2,
        INTERP_ENDFILE = 3,
        INTERP_FILE_NOT_OPEN = 4,
        INTERP_ERROR = 5

joint - returns tuple of dicts
reflecting current joint values
```

See the *Joint Status* labels for individual joint status items.

```
joint_actual_position - returns tuple of floats
actual joint positions

joint_position - returns tuple of floats
desired joint positions

joints_lb - returns integer
number of joints. Reflects [KINS]JOINTS INI value

kinematics_type_lb - returns integer that is used to lookup the kinematics name
The type of kinematics
        KINEMATICS_IDENTITY = 1
        KINEMATICS_FORWARD_ONLY = 2
        KINEMATICS_INVERSE_ONLY = 3
        KINEMATICS_BOTH = 4

limit - returns tuple of integers
axis limit masks. minHardLimit=1, maxHardLimit=2, minSoftLimit=4, maxSoftLimit=8
```

(continues on next page)

(continued from previous page)

```
linear_units_lb - returns float - precision can be set
machine linear units per mm, reflects [TRAJ]LINEAR_UNITS INI value

lube_lb - returns integer
lube on flag

lube_level_lb - returns integer
reflects iocontrol.0.lube_level

max_acceleration_lb - returns float - precision can be set
maximum acceleration. Reflects [TRAJ]MAX_ACCELERATION

max_velocity_lb - returns float - precision can be set
maximum velocity. Reflects the current maximum velocity. If not modified by
halui.max-velocity or similar it should reflect [TRAJ]MAX_VELOCITY

min_jog_vel_lb - returns int
minimum jog velocity slider setting. Reflects the [DISPLAY] MIN_LINEAR_VELOCITY
setting in user units per minute.

max_jog_vel_lb - returns int
maximum jog velocity slider setting. Reflects the [DISPLAY] MAX_LINEAR_VELOCITY
setting in user units per minute.

mcodes_lb - returns tuple of 10 integers
currently active M-codes

mist_lb - returns integer
Mist status, either MIST_OFF = 0 or MIST_ON = 1

motion_line_lb - returns integer
source line number motion is currently executing

motion_mode_lb - returns integer that is used to lookup the motion mode name
This is the mode of the Motion controller.
        TRAJ_MODE_FREE = 1
        TRAJ_MODE_COORD = 2
        TRAJ_MODE_TELEOP = 3

motion_type_lb - returns integer that is used to lookup the motion type name
The type of the currently executing motion. One of:
        MOTION_TYPE_TRAVERSE = 1
        MOTION_TYPE_FEED = 2
        MOTION_TYPE_ARC = 3
        MOTION_TYPE_TOOLCHANGE = 4
        MOTION_TYPE_PROBING = 5
        MOTION_TYPE_INDEXROTARY = 6
        Or 0 if no motion is currently taking place.

optional_stop_lb - returns integer
option stop flag
```

(continues on next page)

(continued from previous page)

```
paused_lb - returns boolean
motion paused flag

pocket_prepped_lb - returns integer
A Tx command completed, and this pocket is prepared. -1 if no prepared pocket

position - returns tuple of floats
trajectory position

probe_tripped_lb - returns boolean
True if probe has tripped

probe_val_lb - returns integer
reflects value of the motion.probe-input pin

probed_position_lb - returns tuple of floats
position where probe tripped

probing_lb - returns boolean
True if a probe operation is in progress

program_units_lb - returns integer that is used to lookup the units name
        CANON_UNITS_INCHES = 1,
        CANON_UNITS_MM = 2,
        CANON_UNITS_CM = 3

queue_lb - returns integer
current size of the trajectory planner queue

queue_full_lb - returns boolean
the trajectory planner queue is full

rapid_override_lb - returns percent
rapid override percent

rapidrate_lb - returns float - precision can be set
rapid override scale, 1.0 = 100%

read_line_lb - returns integer
line the RS274NGC interpreter is currently reading

rotation_xy_lb - returns float - precision can be set
current XY rotation angle around Z axis

settings_lb - returns tuple of floats
current interpreter settings
settings[0] = sequence number
settings[1] = feed rate
settings[2] = speed
settings[3] = G64 P blend tolerance
settings[4] = G64 Q naive CAM tolerance
```

(continues on next page)

(continued from previous page)

```
spindles_lb - returns tuple of dicts
returns the current spindle status

state_lb - returns integer that is used to lookup the state name
current command execution status
One of RCS_DONE = 1, RCS_EXEC = 2, RCS_ERROR = 3

task_mode_lb - returns integer that is used to lookup the task mode name
current task mode
One of  MODE_MANUAL = 1, MODE_AUTO = 2, MODE_MDI = 3

task_paused_lb - returns integer
task paused flag, not paused = 0, paused = 1

task_state_lb - returns integer that is used to lookup the task state name
current task state
One of STATE_ESTOP = 1, STATE_ESTOP_RESET = 2, STATE_OFF = 3 STATE_ON = 4
STATE_OFF is never seen

tool_in_spindle_lb - returns integer
current tool number in spindle (0 if no tool loaded)

tool_from_pocket_lb - returns integer
pocket number for the currently loaded tool (0 if no tool loaded)

tool_offset_lb - returns tuple of floats
offset values of the current tool

tool_table_lb - returns tuple of tool_results
list of tool entries. Each entry is a sequence of the following fields: id,
xoffset, yoffset, zoffset, aoffset, boffset, coffset, uoffset, voffset,
woffset, diameter, frontangle, backangle, orientation. The id and orientation
are integers and the rest are floats.
If id = -1 no tools are in the tool table.
```

Note: You don't have to use all the labels; only use the ones you need.

7.3 Axis Status

The Axis status contains status items for all 9 axes. Replace the *n* with the number of the axis. Axis numbers start at 0 and go through 8. Returns a float

Table 1: Axis Status Labels

axis_n_max_position_limit_lb	axis_n_min_position_limit_lb
axis_n_velocity_lb	axis_n_vel_per_min_lb

Note: The Axis velocity label only reports back *jogging* speed; use the joint velocity label for *linear* speed.

7.4 Joint Status Labels

The Joint status contains status items for 16 joints. Replace the *n* with the number of the joint. Joint numbers start at 0 and go through 15

Table 2: Joint Status Labels

joint_n_backlash_lb	joint_n_enabled_lb
joint_n_fault_lb	joint_n_ferror_current_lb
joint_n_ferror_highmark_lb	joint_n_homed_lb
joint_n_homing_lb	joint_n_min_soft_limit_lb
joint_n_inpos_lb	joint_n_input_lb
joint_n_jointType_lb	joint_n_max_ferror_lb
joint_n_max_hard_limit_lb	joint_n_max_position_limit_lb
joint_n_max_soft_limit_lb	joint_n_min_ferror_lb
joint_n_min_hard_limit_lb	joint_n_min_position_limit_lb
joint_n_output_lb	joint_n_override_limits_lb
joint_n_units_lb	joint_n_vel_min_lb
joint_n_vel_sec_lb	

7.4.1 Joint Descriptions

backlash (returns float) - Backlash in machine units. configuration parameter, reflects [JOINT_n]BACKLASH.

enabled (returns integer) - non-zero means enabled

fault (returns integer) - non-zero means axis amp fault.

ferror_current (returns float) - current following error.

ferror_highmark (returns float) - magnitude of max following error.

homed (returns integer) - non-zero means has been homed.

homing (returns integer) - non-zero means homing in progress.

inpos(returns integer) - non-zero means in position.

input (returns float) - current input position.

jointType (returns integer) - type of axis configuration parameter, reflects [JOINT_n]TYPE. LINEAR=1, ANGULAR=2. See Joint INI configuration for details.

max_ferror (returns float) - maximum following error. configuration parameter, reflects [JOINT_n]FERROR.

max_hard_limit (returns integer) - non-zero means max hard limit exceeded.

max_position_limit (returns float) - maximum limit (soft limit) for joint motion, in machine units. configuration parameter, reflects [JOINT_n]MAX_LIMIT.

max_soft_limit non-zero means max_position_limit was exceeded, int

min_ferror (returns float) - configuration parameter, reflects [JOINT_n]MIN_FERROR.

min_hard_limit (returns integer) - non-zero means min hard limit exceeded.

min_position_limit (returns float) - minimum limit (soft limit) for joint motion, in machine units. configuration parameter, reflects [JOINT_n]MIN_LIMIT.

min_soft_limit (returns integer) - non-zero means min_position_limit was exceeded.

output (returns float) - commanded output position.

override_limits (returns integer) - non-zero means limits are overridden.

units (returns float) - joint units per mm, or per degree for angular joints. (joint units are the same as machine units, unless set otherwise by the configuration parameter [JOINT_n]UNITS)

velocity (returns float) - current velocity.

7.5 Special Labels

Run from line label *start_line_lb*

7.6 Axis Position Labels

Axis machine position labels (no offsets.) Returns a float

Table 3: Machine Absolute Position Status Labels

actual_lb_x	actual_lb_y	actual_lb_z
actual_lb_a	actual_lb_b	actual_lb_c
actual_lb_u	actual_lb_v	actual_lb_w

Axis position labels *including* all offsets. Returns a float

Table 4: DRO Relative Status Labels

dro_lb_x	dro_lb_y	dro_lb_z
dro_lb_a	dro_lb_b	dro_lb_c
dro_lb_u	dro_lb_v	dro_lb_w

7.7 Axis Distance to Go labels

Table 5: Distance to Go Labels

dtg_lb_x	dtg_lb_y	dtg_lb_z
dtg_lb_a	dtg_lb_b	dtg_lb_c
dtg_lb_u	dtg_lb_v	dtg_lb_w

7.8 Axis Homed Labels

Table 6: Axis Homed Labels

home_lb_0	home_lb_1	home_lb_2
home_lb_3	home_lb_4	home_lb_5
home_lb_6	home_lb_7	home_lb_8

7.9 Offset Labels

Offsets for the currently active G5x coordinate system. Returns a float

Table 7: G5x Status Labels

g5x_lb_x	g5x_lb_y	g5x_lb_z
g5x_lb_a	g5x_lb_b	g5x_lb_c
g5x_lb_u	g5x_lb_v	g5x_lb_w

Offsets for G92. Returns a float

Table 8: G92 Status Labels

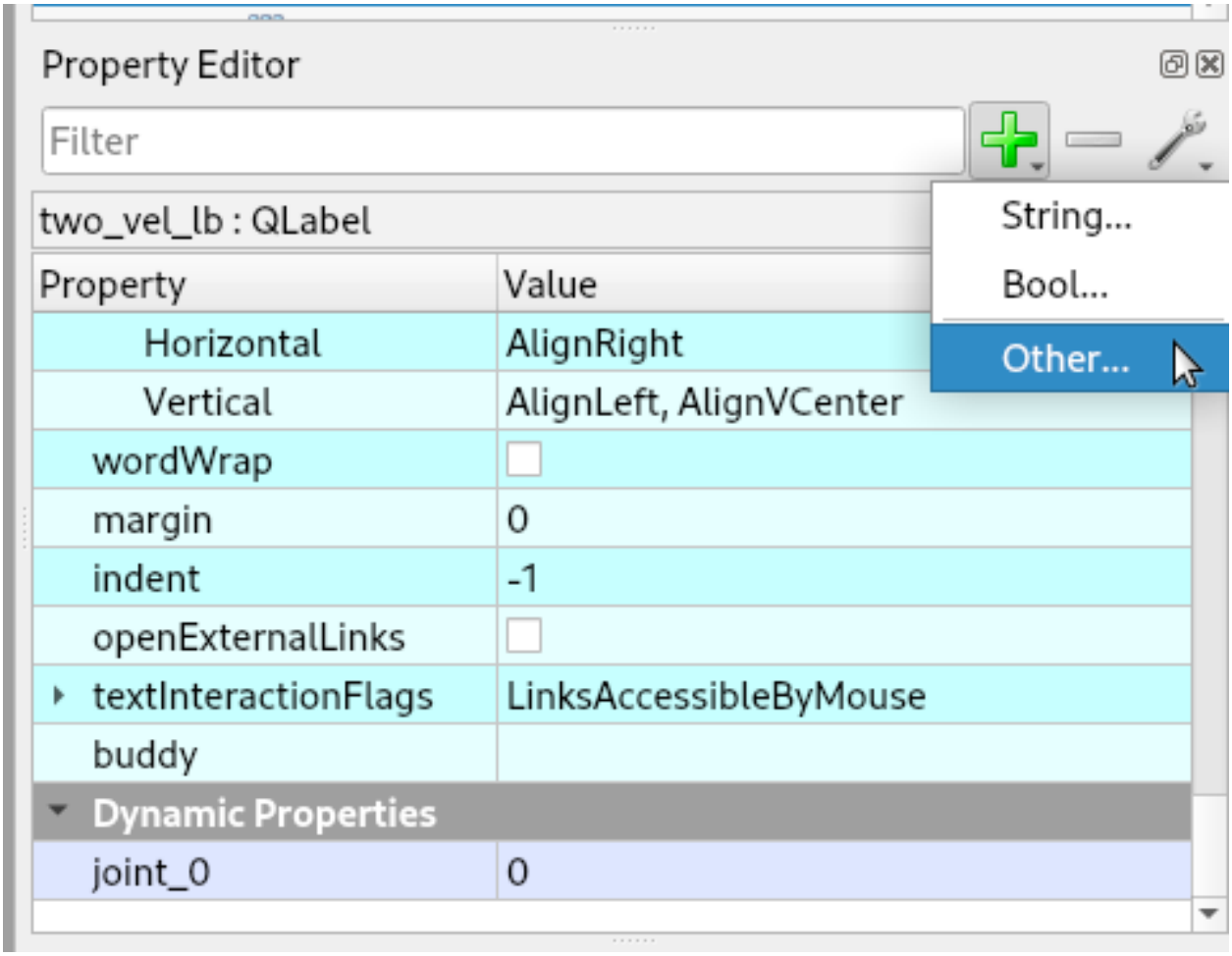
g92_lb_x	g92_lb_y	g92_lb_z
g92_lb_a	g92_lb_b	g92_lb_c
g92_lb_u	g92_lb_v	g92_lb_w

7.10 Velocity Labels

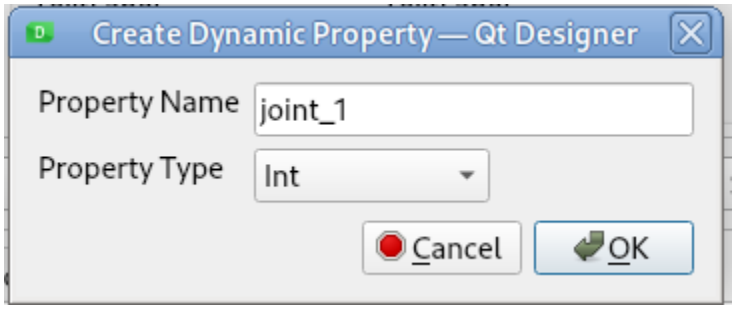
Tool velocity using two perpendicular joint velocities.

Name the label *two_vel_lb* and add two int type Dynamic Properties called *joint_0* and *joint_1* and set the values to the perpendicular joint numbers you want to calculate. Typically this would be for the X and Y axes.

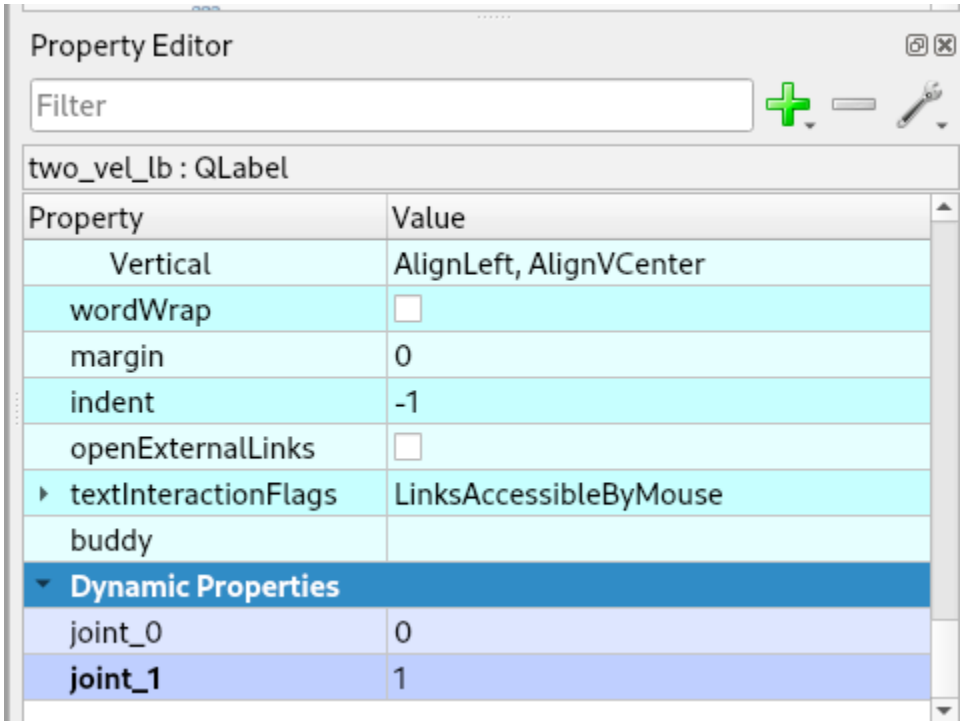
To select an int type of Dynamic Property, select *Other* after clicking on the green plus sign



Then select the Property Type of *int*



The two Dynamic Properties should look like this



Tool velocity using *three* perpendicular joint velocities.

Name the label *three_vel_lb* and add three int type Dynamic Properties called *joint_0*, *joint_1* and *joint_2* and set the values to the perpendicular joint numbers you want to calculate. Typically this would be for the X, Y and Z axes.

7.11 I/O Status

The I/O status contains status items for 64 I/O's. Replace the *n* with the number of the I/O. I/O numbers start at 0 and go through 63. Analog I/O returns a float. For example a QLabel with an object name of din_5_lb will show the status of the *motion.digital-in-05* HAL pin

Table 9: I/O Status Labels

HAL Pin	Label Name
motion.analog-in-nn	ain_n_lb
motion.analog-out-nn	aout_n_lb
motion.digital-in-nn	din_n_lb
motion.digital-out-nn	dout_n_lb

CHAPTER

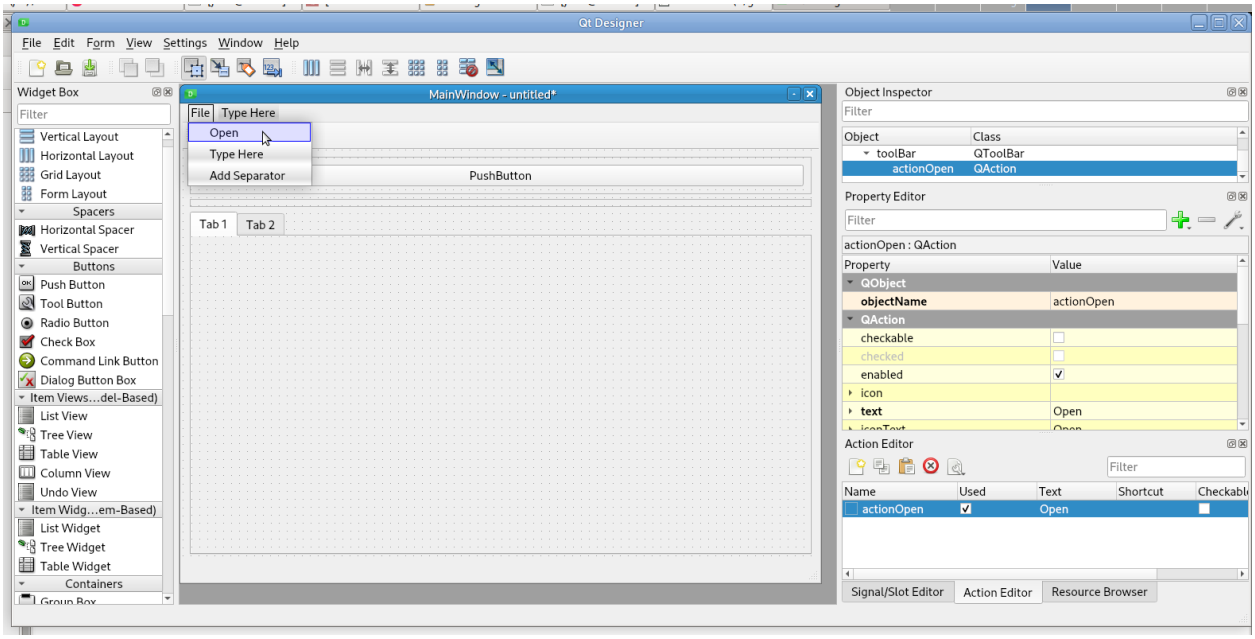
EIGHT

MENU

Adding Menu Items Tutorial Tool Bar Buttons Tutorial

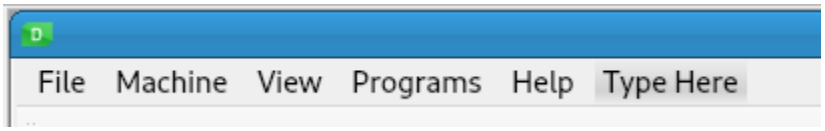
Note: Every menu item has a command button, so you don't need to use any menu items if you don't want to.

Adding a menu item creates an action. When you create File > Open menu, the *actionOpen* action is created.



Warning: If you use the full-screen option, you will not be able to exit the application if you don't have the Exit action or an Exit Push Button or Press ALT-F4 to close the GUI.

This shows the typical menu categories which are the first items in each menu. The image is from the Qt Designer.



The following table shows the menu name you type into Qt Designer and the action name that is created by the Qt Designer. Menu categories like *File* don't create an action name.

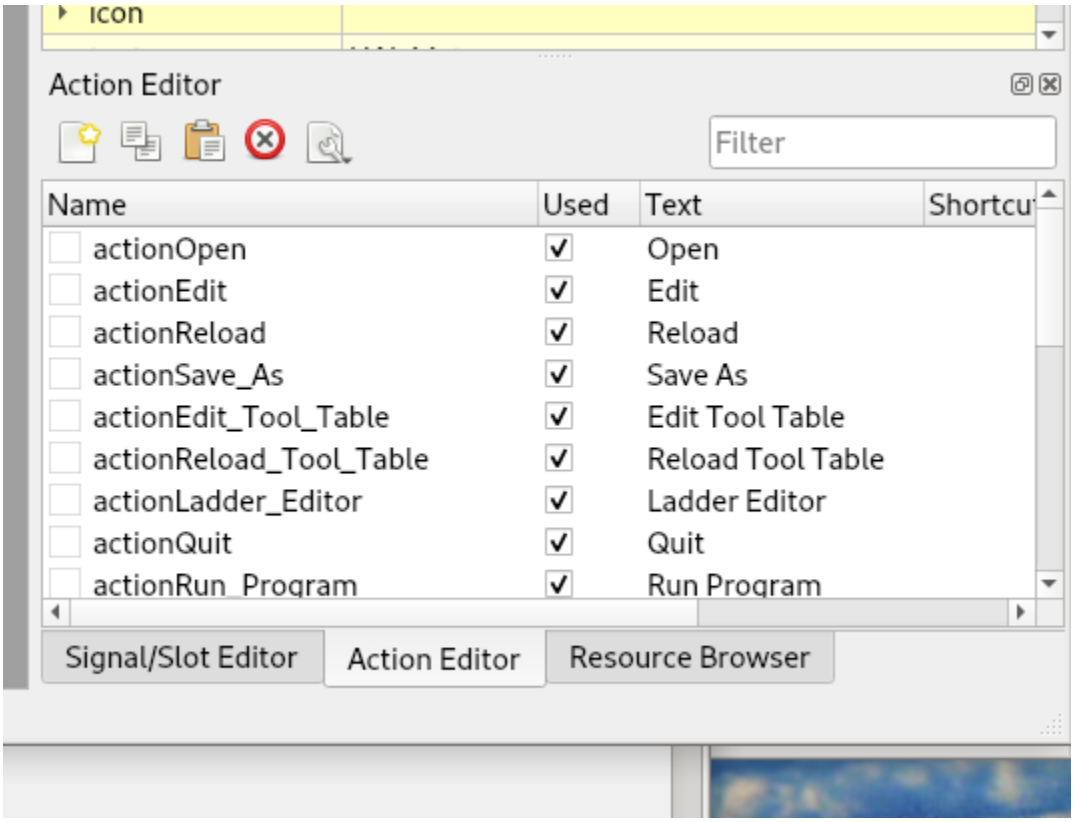
Table 1: Menu Items

File	**Action Name**
Open	actionOpen
Edit	actionEdit
Reload	actionReload
Save As	actionSave_As
Edit Tool Table	actionEdit_Tool_Table
Reload Tool Table	actionReload_Tool_Table
Ladder Editor	actionLadder_Editor
Quit	actionQuit
Machine	**Action Name**
E Stop	actionE_Stop
Power	action_Power
Run	actionRun
Run From Line	actionRun_From_Line
Step	actionStep
Pause	actionPause
Resume	actionResume
Stop	actionStop
Clear MDI History	actionClear_MDI_History
Copy MDI History	actionCopy_MDI_History
Homing	this creates a home menu item for each axis
Unhoming	this creates a unhome menu item for each axis
Clear Offsets	this creates a clear offsets for each coordinate system
Programs	**Action Name**
Show HAL	actionShow_HAL
HAL Meter	actionHAL_Meter
HAL Scope	actionHAL_Scope
View	**Action Name**
DRO	actionDRO
Limits	actionLimits
Extents Option	actionExtents_Option
Live Plot	actionLive_Plot
Velocity	actionVelocity
Metric Units	actionMetric_Units
Program	actionProgram
Rapids	actionRapids
Tool	actionTool
Lathe Radius	actionLathe_Radius
DTG	actionDTG
Offsets	actionOffsets
Overlay	actionOverlay
Clear Live Plot	actionClear_Live_Plot
Help	**Action Name**
About	actionAbout
Quick Reference	actionQuick_Reference

8.1 Action Names

When you add a menu item, it creates an action and the Object Name is created from the menu name automatically.

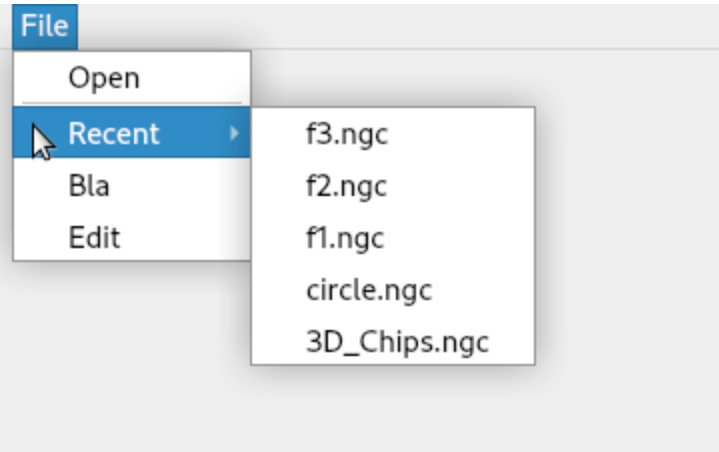
The Object Name must match the above items *exactly* in order to be discovered by Flex GUI:



8.2 Recent Files

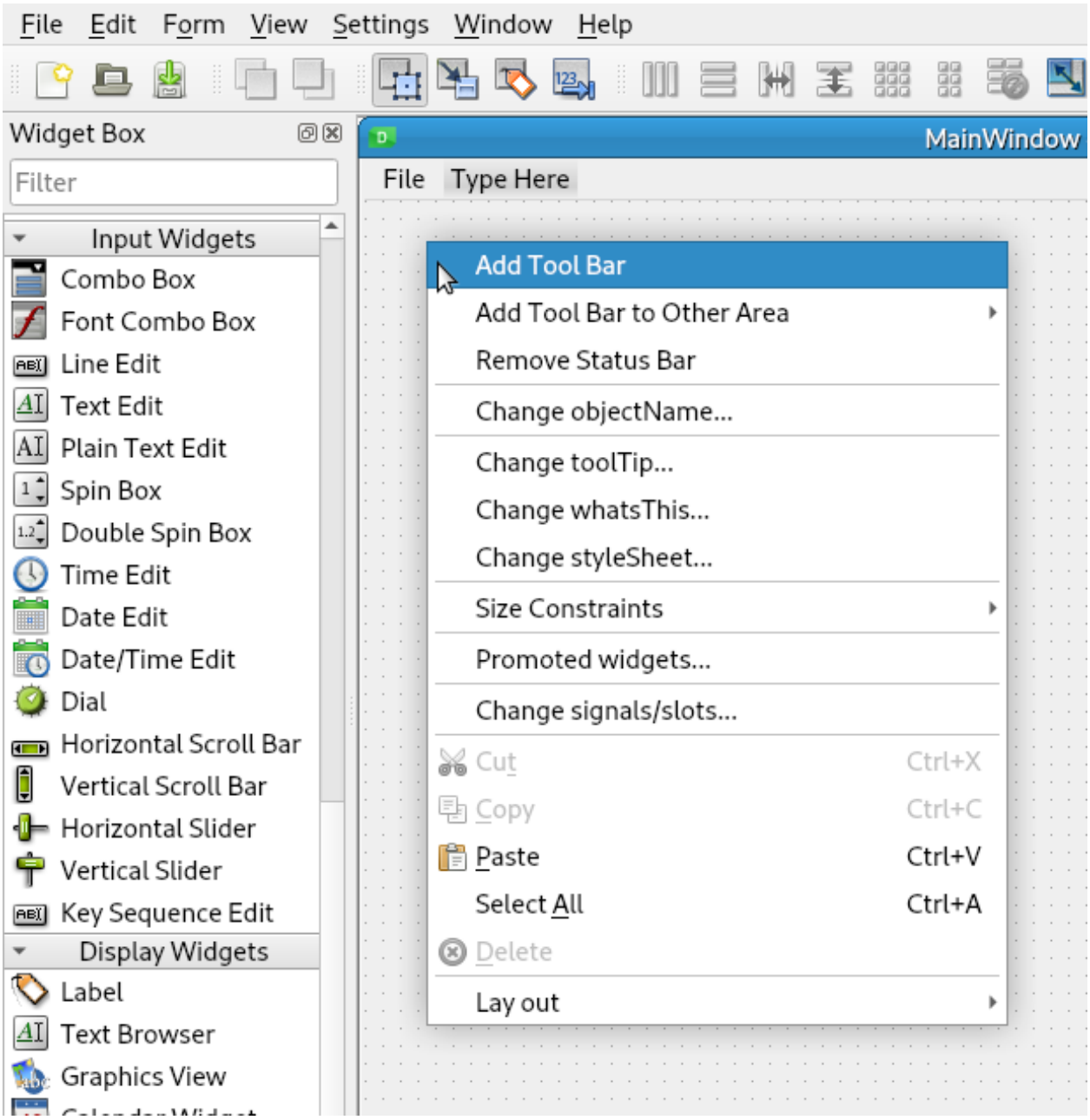
Note: The Recent menu item is added after the Open menu. There must be at least one menu item after Open for the Recent menu to be added.

Location of the Recent menu after the Open menu:

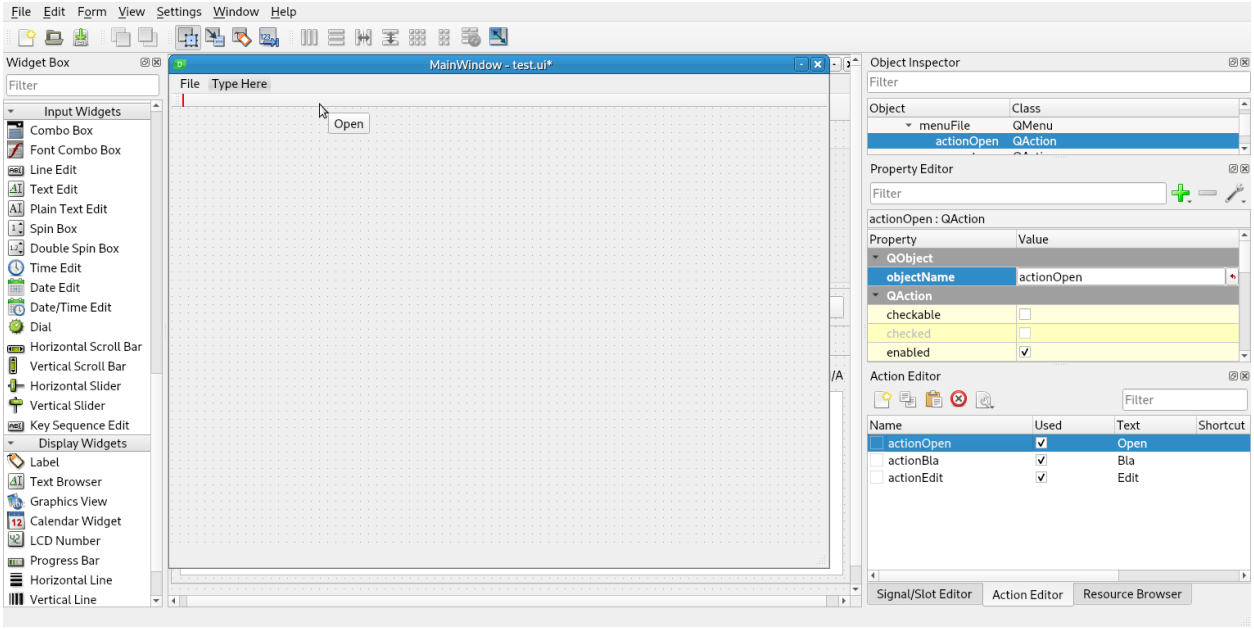


8.3 Tool Bars

If you right-click on the main window, you can add a Tool Bar:



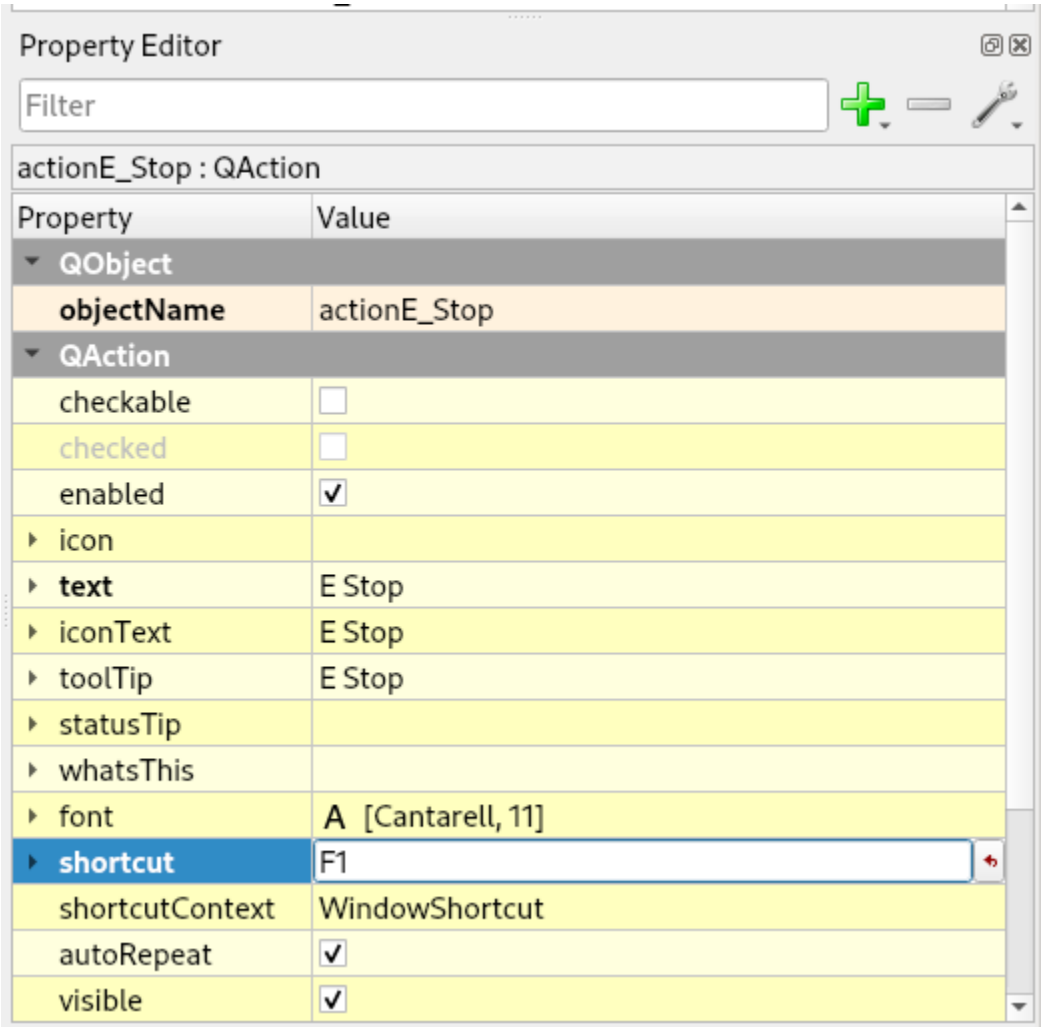
To add actions to the Tool Bar, drag them from the Action Editor and drop them in the Tool Bar:



To set the style of a Tool Bar Button, use the action name and replace action with *flex_* for example the actionQuit would be *flex_Quit*. See *Tool Bar Buttons* in the stylesheet examples.

8.4 Shortcut Keys

Shortcut keys can be added in the Property Editor by clicking in the shortcut Value box and pressing the key or key combination you want to use. You can change text, icon Text, or tool Tip also.



Note: A toolTip can be handy, however they might not work on touchscreens.

CHAPTER

NINE

CONTROLS

Command Buttons Tutorial Home Controls Tutorial

9.1 Push Buttons

Controls are QPushButtons that can be placed anywhere you like. Use the Name from the list below for each control widget objectName. Replace the *(0-8)* with the joint number or axis index. More controls are in *Tools*.

Table 1: Control Push Buttons

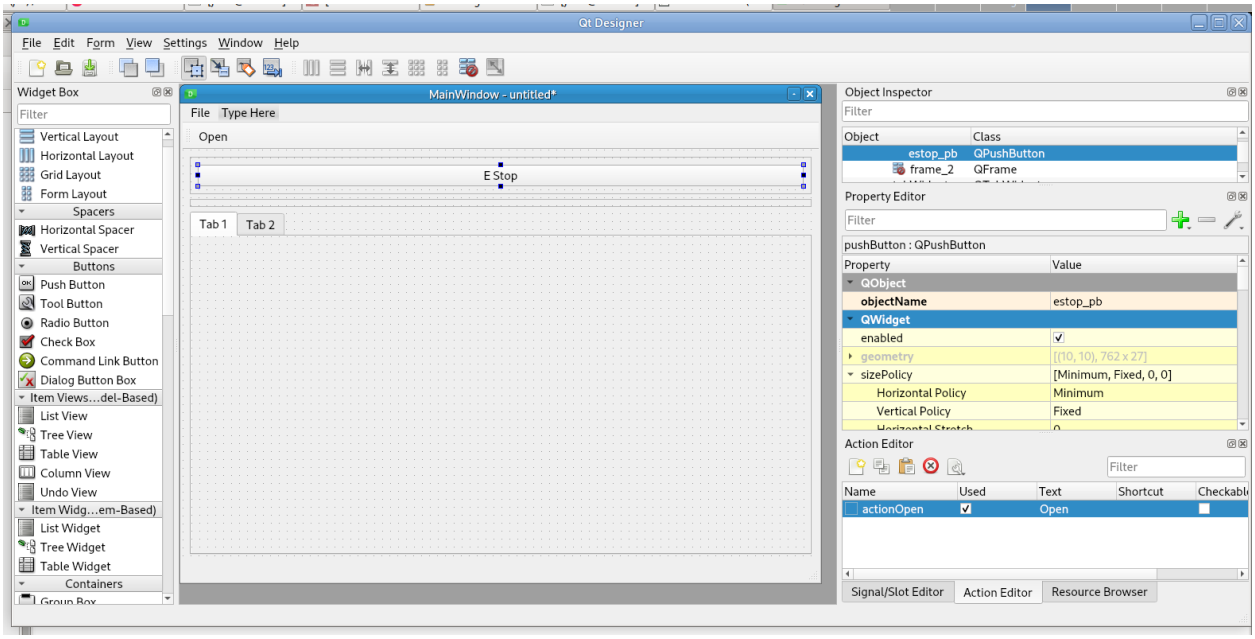
Control Function	**Object Name**
Open a G-code File	open_pb
Edit a G-code File	edit_pb
Reload a G-code File	reload_pb
Edit Tool Table	edit_tool_table_pb
Edit Ladder	edit_ladder_pb
Reload Tool Table	reload_tool_table_pb
Save	save_pb
Save As a New Name	save_as_pb
Quit the Program	quit_pb
E-Stop Toggle	estop_pb
Power Toggle	power_pb
Run a Loaded G-code File	run_pb
Run From Line	run_from_line_pb
Step one Logical Line	step_pb
Pause a Running Program	pause_pb
Resume a Paused Program	resume_pb
Stop a Running Program	stop_pb
Home All Joints	home_all_pb
Home a Joint (0-8)	home_pb_(0-8)
Unhome All Joints	unhome_all_pb
Unhome a Joint (0-8)	unhome_pb_(0-8)
Clear an Axis Offset	clear_(axis letter)_pb
Manual Mode	manual_mode_pb
Flood Toggle	flood_pb
Mist Toggle	mist_pb
Clear Error History	clear_errors_pb
Copy Error History	copy_errors_pb
Clear Information History	clear_info_pb

continues on next page

Table 1 – continued from previous page

Show HAL	show_hal_pb
HAL Meter	hal_meter_pb
HAL Scope	hal_scope_pb
Help About	about_pb
Quick Reference	quick_reference_pb

Note: You don't have to use any of these controls; Flex GUI is flexible.

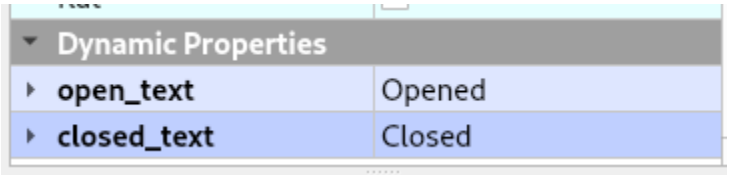


Note: Touch-Off buttons require a Double Spin Box named *touchoff_dsb*

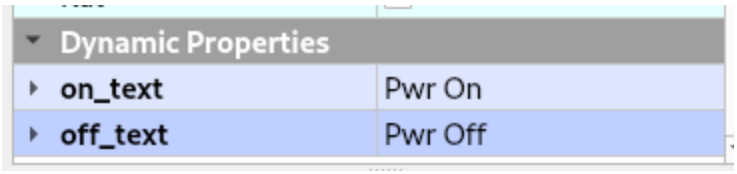
Note: Tool-Touch-Off buttons require a Double Spin Box named *tool_touchoff_dsb*

9.2 E Stop and Power

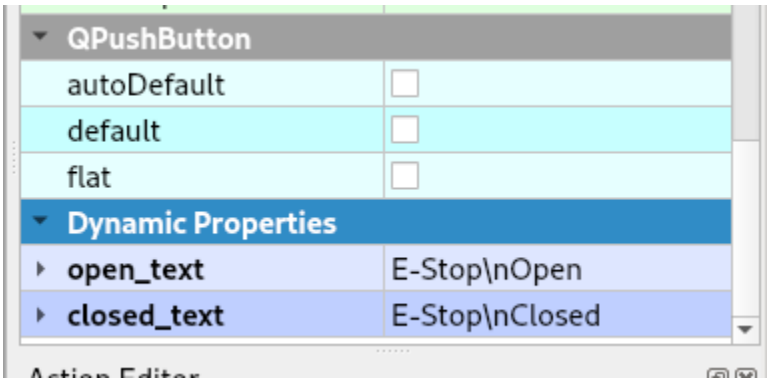
The E Stop push button Open/Closed state text can be set by adding two string type Dynamic Properties *open_text* and *closed_text*. The text in those two properties will be used if found. See *Dynamic Properties*



The Power push button On/Off state text can be set by adding two string type Dynamic Properties *on_text* and *off_text*. The text in those two properties will be used if found. The default text is *Power Off* and *Power On*.



Note: To have two words be above and below insert a backslasgh and n between the words.



This is how the above looks in the GUI

9.3 LED Buttons

Important: This is implmented in version 1.2.0.

Some push buttons can have a LED to indicate on and off states. The LED is added to the push button with a Bool type Dynamic Property called *led_indicator*. All other properties are optional.

Table 2: LED Buttons

Control Name	Object Name	Function
Save	save_pb	NC Code Save Button
Reload	reload_pb	Reload NC Code from current file
E Stop	estop_pb	E Stop Toggle Button
Power	power_pb	Power Toggle Button
Run	run_pb	Runs a loaded NC file
Pause	pause_pb	Pauses a running NC file
Manual Mode	manual_mode_pb	Puts the control into Manual Mode
MDI Mode	mdi_mode_pb	Puts the control into MDI Mode
Flood	flood_pb	Turns on the flood coolant
Mist	mist_pb	Turns on the mist coolant
Probe Enable	probing_enable_pb	Enables Probing and disables other controls
Home All	home_all_pb	Homes all the joints

Adding the Bool type Dynamic Property *led_indicator* to the above control buttons will add the default LED to that button. Each control button can have different options. The default colors are Red when Off and Green when On. The default shape is round.

Table 3: LED Button Dynamic Properties

Property Type	Property Name	Function
Bool	led_indicator	Creates a LED
	Optional	
Int	led_diameter	Sets the Diameter of the LED in pixels
Int	led_right_offset	Sets the offset from the right edge in pixels
Int	led_top_offset	Sets the offset from the top edge in pixels
Color	led_on_color	Sets the color of the LED when on
Color	led_off_color	Sets the color of the LED when off
String	led_shape	square

To change the LED default options they can be set in the INI file. See *LED Defaults*

Tip: A space after the button text gives more room for the LED

9.4 Coordinate System Controls

A QPushButton can be used to clear the current coordinate system by using 0 as the index or any one of the 9 coordinate systems with (1-9).

To clear the G92 coordinate system use 10 as the index.

To clear the current G54x coordinate system rotation use 11 as the index.

Table 4: Clear Coordinate System Offsets

Control Function	Object Name
Clear Current G5x	clear_coord_0
Clear G5x Coordinate System	clear_coord_(1-9)
Clear G92 Coordinate System	clear_coord_10
Clear Current G5x Rotation clear_coord_11	

To clear an axis offset in the current G5x coordinate system use the following.

Table 5: Clear Axis Offsets

Control Function	Object Name
Clear Current G5x X axis	clear_(axis letter)_pb

9.5 Options

The QPushButton options are toggle-type buttons; press to turn on, press again to turn off. They are normal push buttons but Flex automatically makes them *checkable*.

Table 6: Options

Function	Widget	Name
Flood Toggle	QPushButton	flood_pb
Mist Toggle	QPushButton	mist_pb
Optional Stop at M1	QPushButton	optional_stop_pb
Block Delete line that starts with /	QPushButton	block_delete_pb
Feed Override Enable/Disable	QPushButton	feed_override_pb

9.6 Axis Index

```
X 0
Y 1
Z 2
A 3
B 4
C 5
U 6
V 7
W 8
```

CHAPTER

TEN

JOGGING

Jogging requires a *Jog Velocity Slider* and *Jog Mode Selector*. If either is not found, Jogging will be disabled.

Jogging increments are from the ini entry *INCREMENTS* in the [DISPLAY] section. See *Jog Increments* for more information.

Table 1: Required Jog Widgets

Function	Widget	Name
Jog Velocity Slider	QSlider	jog_vel_sl
Jog Mode Selector	QComboBox	jog_modes_cb

The Jog Velocity Label shows the current jog velocity setting from the Jog Velocity Slider

Table 2: Optional Jog Widgets

Function	Widget	Name
Jog Velocity Label	QLabel	jog_vel_lb

10.1 Keyboard Jogging

To enable keyboard jogging a QCheckbox is used. When checked the right/left arrow keys jog the X axis and the up/down arrow keys jog the Y axis and the page up/down keys jog the Z axis. When not checked the keys function as normal keys.

Table 3: Keyboard Jogging

Function	Widget	Name
Jog Enable	QCheckBox	keyboard_jog_cb

10.2 Jog Button Controls

Jog Controls Tutorial

This type of jog controls provides a button for each axis and jog direction.

Table 4: Jog Button Widgets

Function	Widget	Name
Jog Plus Axis (0-8)	QPushButton	jog_plus_pb_(0-8)
Jog Minus Axis (0-8)	QPushButton	jog_minus_pb_(0-8)

Note: Jog Plus/Minus buttons use the *Axis Index*. So *Jog Y Plus* is *jog_plus_pb_1*.

Note: *Jog Mode Selector* reads the ini entry [DISPLAY] INCREMENTS and if not found, only *Continuous* will be an option.

10.3 Jog Selected Axis Controls

To add Axis style jog controls where you select an axis then the plus/minus buttons jog the selected axis add a QRadioButton for each axis and a QPushButton for Plus and Minus. Axes are 0-8 for X, Y, Z, A, B, C, U, V, W.

Table 5: Jog Selected Widgets

Function	Widget	Name
Axis Select (0-8)	QRadioButton	axis_select_(0-8)
Jog Plus	QPushButton	jog_selected_plus
Jog Minus	QPushButton	jog_selected_minus

10.4 Overrides

Overrides Tutorial

A QSlider is used to control the following functions and the corresponding label shows the value of the slider:

Table 6: Overrides

Function	Widget	Object Name
Feed Override Slider	QSlider	feed_override_sl
Feed Override Percent	QLabel	feed_override_lb
Rapid Override Slider	QSlider	rapid_override_sl
Rapid Override Percent	QLabel	rapid_override_lb
Spindle Override Slider	QSlider	spindle_override_sl
Spindle Override Percent	QLabel	spindle_override_0_lb
Maximum Velocity	QSlider	max_vel_sl
Override Limits	QCheckBox	override_limits_cb

The following settings can be used in the DISPLAY section of the ini file:

```
Feed Override maximum              MAX_FEED_OVERRIDE
Spindle Override maximum           MAX_SPINDLE_OVERRIDE
```

10.5 Override Presets

Feed, Rapid and Spindle overrides can have a preset button(s) for different preset amounts. Replace the nnn with the percent of override you want that button to use.

Table 7: Override Presets

Function	Widget	Object Name
Feed Override Preset	QPushButton	feed_percent_nnn
Rapid Override Preset	QPushButton	rapid_percent_nnn
Spindle Override Preset	QPushButton	spindle_percent_nnn

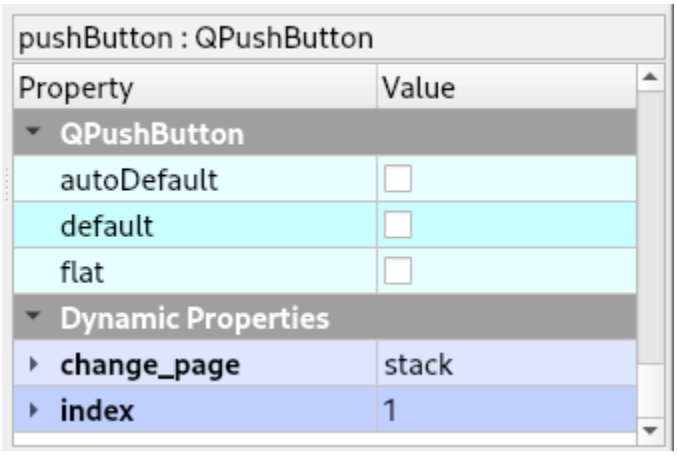
Note: The maximum override for Rapid is 100

10.6 Stacked Widget

To change to a specific page on a QStackedWidget add a QPushButton on each page and set a couple of Dynamic Properties. See *Dynamic Properties*

Table 8: Stacked Widget Change Page

Dynamic Property Name	Value
change_page	QStackedWidget Object Name
index	index of page to change to



To create a Next Page and Previous Page buttons for a QStackedWidget add two QPushButtons with the following Dynamic Properties. See *Dynamic Properties*

Table 9: Stacked Widget Next/Previous Page

Button Function	Dynamic Property Name	Value
Next Page	next_page	QStackedWidget Object Name
Previous Page	previous_page	QStackedWidget Object Name

Note: The Forward and Backward Buttons should not be in the QStackedWidget

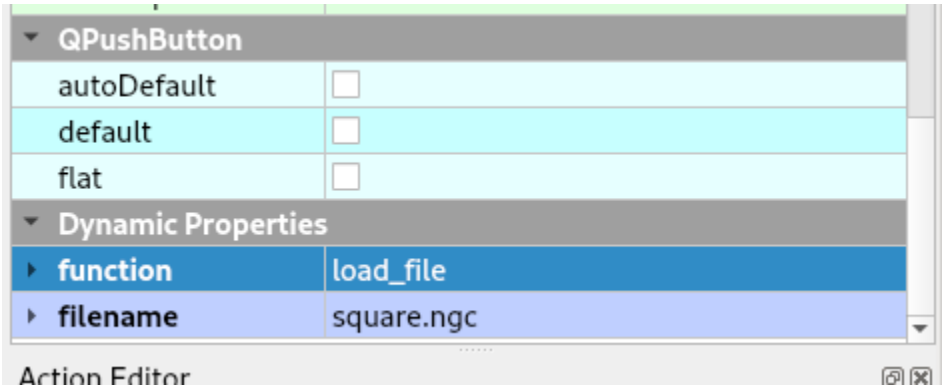
10.7 File Load Buttons

To create a QPushButton to load a specific file add two Dynamic Properties.

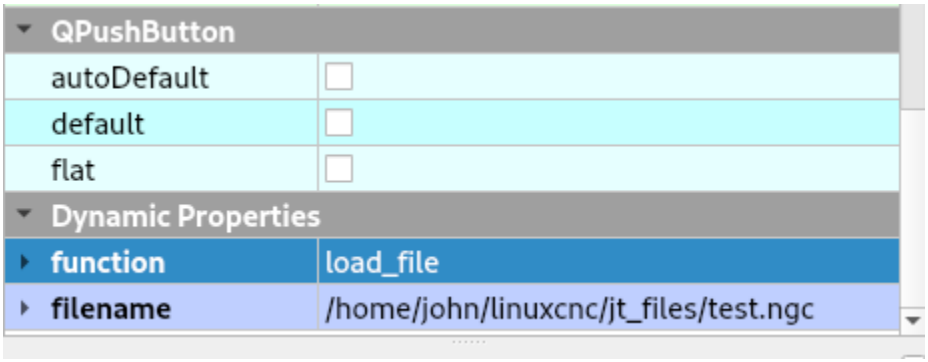
Table 10: File Load Buttons

Dynamic Property Name	Value
function	load_file
filename	file to load

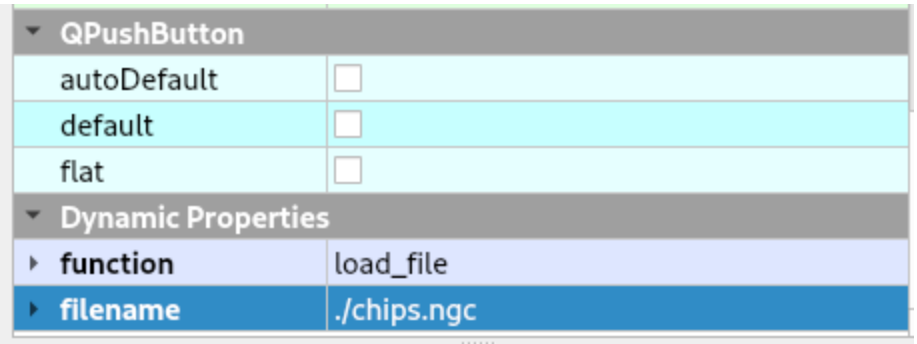
File Name in the PROGRAM_PREFIX path.



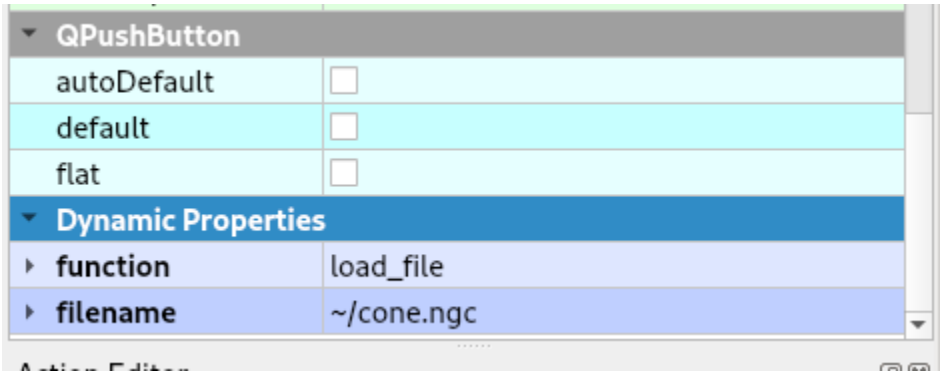
File Name with Full Path



File Name in the Configuration Directory



File Name relative to the Users Home Directory



The file name can be just the name and extension if it's in the PROGRAM_PREFIX path. Or it can be any valid path and file name.

File Name Examples

```
A file in the PROGRAM_PREFIX path
somefile.ngc

A file in the configuration directory
./anotherfile.ngc

A file up one directory from the configuration directory
../up_one.ngc

A file relative to the users home directory
/home/fred/linuxcnc/my_files/afile.ngc
could be
~/linuxcnc/my_files/afile.ngc
```

Warning: The file must be in the directory specified by the INI entry PROGRAM_PREFIX in the [DISPLAY] section or have a valid path.

This is useful for probe routine buttons to load the nc code so the path can be viewed in the plotter and for programs that are ran frequently.

Note: The file is not added to the Recent Files list.

CHAPTER

ELEVEN

PLOTTER

Plotter Tutorial

To add a live G-code plotter, add a QWidget or QFrame and name it *plot_widget*.

11.1 Controls

If you're using a touch-screen, add pan, zoom, and rotate controls for the plotter

Table 1: Display Controls

Control	**Widget**	**Object Name**
Rotate View Up	QPushButton	view_rotate_up_pb
Rotate View Down	QPushButton	view_rotate_down_pb
Rotate View Left	QPushButton	view_rotate_left_pb
Rotate View Right	QPushButton	view_rotate_right_pb
Pan View Up	QPushButton	view_pan_up_pb
Pan View Down	QPushButton	view_pan_down_pb
Pan View Left	QPushButton	view_pan_left_pb
Pan View Right	QPushButton	view_pan_right_pb
Zoom View In	QPushButton	view_zoom_in_pb
Zoom View Out	QPushButton	view_zoom_out_pb

The following controls set-predefined views

Table 2: Display Views

Control	**Widget**	**Object Name**
View Perspective	QPushButton	view_p_pb
View X	QPushButton	view_x_pb
View Y	QPushButton	view_y_pb
View Y2	QPushButton	view_y2_pb
View Z	QPushButton	view_z_pb
View Z2	QPushButton	view_z2_pb

To clear the Live plot

Table 3: Display Functions

Control	Widget	Name
Clear Live Plot	QPushButton	view_clear_pb

11.2 Display

The DRO overlaid onto the plotter can be customized by turning on or off various features. Use either a QCheckbox or a QPushButton to toggle these

Table 4: Display Checkbox Options

Function	Widget	Object Name
View DRO	QCheckBox	view_dro_cb
View Machine Limits	QCheckBox	view_limits_cb
View Extents Option	QCheckBox	view_extents_option_cb
View Live Plot	QCheckBox	view_live_plot_cb
View Velocity	QCheckBox	view_velocity_cb
Use Metric Units	QCheckBox	view_metric_units_cb
View Program	QCheckBox	view_program_cb
View Rapids	QCheckBox	view_rapids_cb
View Tool	QCheckBox	view_tool_cb
View Lathe Radius	QCheckBox	view_lathe_radius_cb
View Distance to Go	QCheckBox	view_dtg_cb
View Offsets	QCheckBox	view_offsets_cb
View Overlay	QCheckBox	view_overlay_cb

Table 5: Display PushButton Options

Function	Widget	Object Name
View DRO	QPushButton	view_dro_pb
View Machine Limits	QPushButton	view_limits_pb
View Extents Option	QPushButton	view_extents_option_pb
View Live Plot	QPushButton	view_live_plot_pb
View Velocity	QPushButton	view_velocity_pb
Use Metric Units	QPushButton	view_metric_units_pb
View Program	QPushButton	view_program_pb
View Rapids	QPushButton	view_rapids_pb
View Tool	QPushButton	view_tool_pb
View Lathe Radius	QPushButton	view_lathe_radius_pb
View Distance to Go	QPushButton	view_dtg_pb
View Offsets	QPushButton	view_offsets_pb
View Overlay	QPushButton	view_overlay_pb

Note: Don't set the checked property to checked in Qt Designer as this is already handled in the code. Once you check an option it is remembered.

11.3 Menu

The following menu items can set display options. *Menu Name* is what you type when creating the Menu, then press enter. All the items are checkbox type menu items that stay coordinated with the checkbox of the same option.

Table 6: Plot Menu Options

Function	Menu Name	Object Name
View DRO	DRO	actionDRO
View Machine Limits	Limits	actionLimits
View Extents Option	Extents Option	actionExtents_Option
View Live Plot	Live Plot	actionLive_Plot
View Velocity	Velocity	actionVelocity
Use Metric Units	Metric Units	actionMetric_Units
View Program	Program	actionProgram
View Rapids	Rapids	actionRapids
View Tool	Tool	actionTool
View Lathe Radius	Lathe Radius	actionLathe_Radius
View Distance to Go	DTG	actionDTG
View Offsets	Offsets	actionOffsets
View Overlay	Overlay	actionOverlay

Note: Once a view selection has been set, Flex GUI remembers it.

The live plot can be cleared from the menu with this menu item.

Table 7: Plot Menu

Function	Menu Name	Object Name
Clear Live Plot	Clear Live Plot	actionClear_Live_Plot

11.4 DRO

The font size can be set in the ini file by adding in the [FLEXGUI] section DRO_FONT_SIZE = n where n is an integer. The default size is 12.

CHAPTER

TWELVE

MANUAL DATA INPUT (MDI)

MDI Tutorial

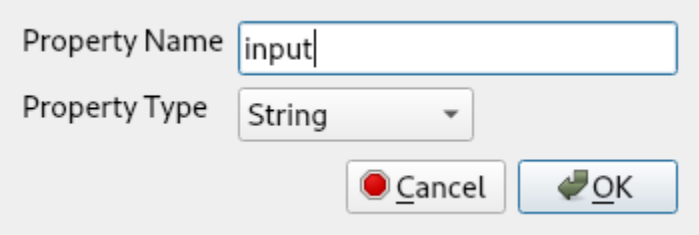
12.1 MDI Interface

The MDI Interface uses a QLineEdit named *mdi_command_le* to enter commands.

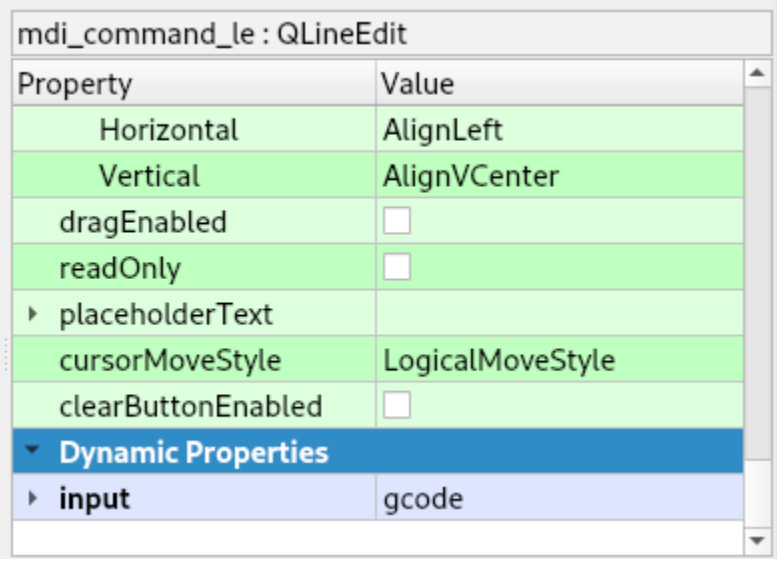
For touch screens there are two options a *NC Popup* is a touch screen that has G and M words and a number keypad or a *Keyboard Popup* has a full keyboard.

To enable a popup add a Dynamic string type Property to the *mdi_command_le* QLineEdit and name it *input* and set the value to either *nccode* or *keyboard*.

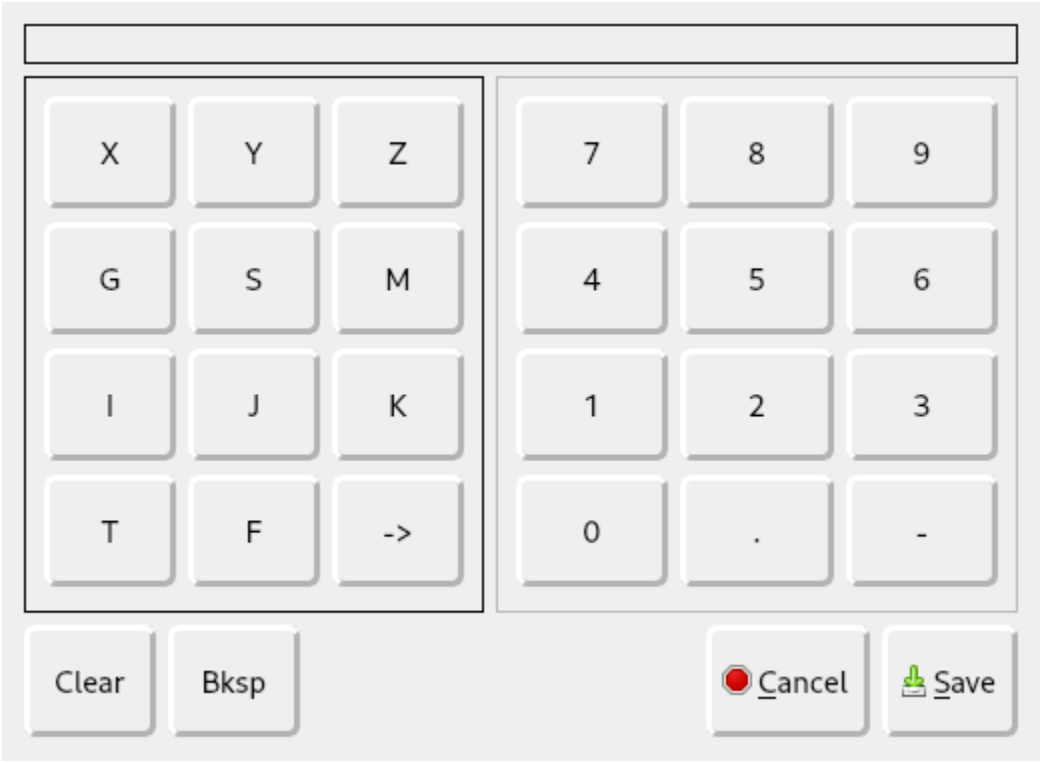
Dynamic Property



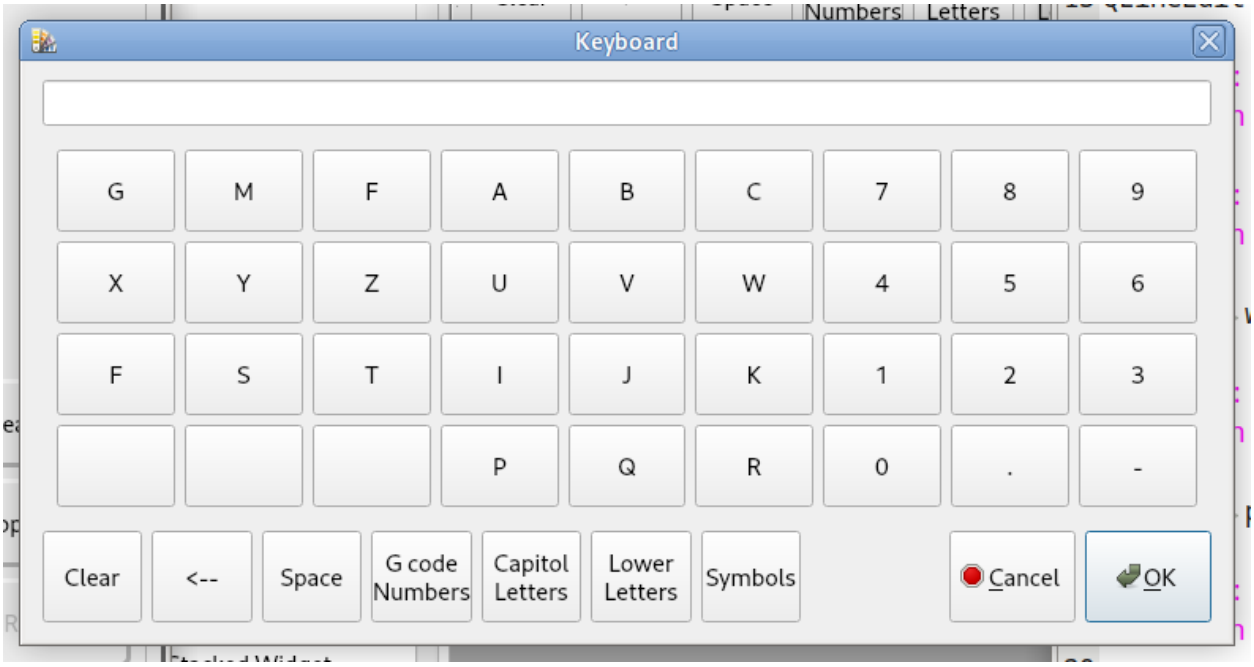
Setting the value



NC code popup window



Keyboard popup window



12.2 MDI History

MDI history uses a QListWidget named *mdi_history_lw* to display the MDI history. You can click on a line in the history display to copy the command to the MDI Interface, ready for running.

The MDI history is kept in a file named *mdi_history.txt* in the configuration directory.

12.3 MDI Controls

The following QPushButtons can be used to execute, copy, and clear MDI command history

Table 1: MDI Push Buttons

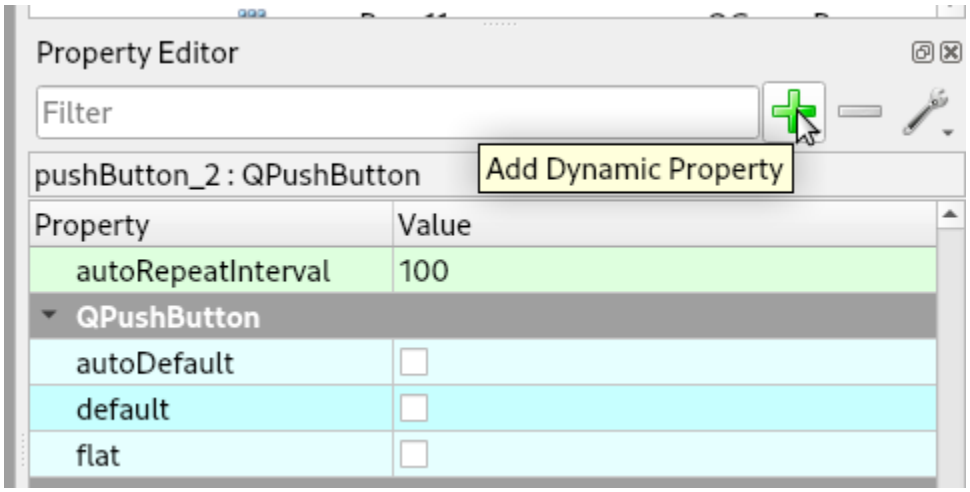
Function	Widget	Object Name
Run MDI Command	QPushButton	run_mdi_pb
Copy the MDI History to the Clipboard	QPushButton	copy_mdi_history_pb
Save the MDI History to a file	QPushButton	save_mdi_history_pb
Clear the MDI History	QPushButton	clear_mdi_history_pb

12.4 MDI Button

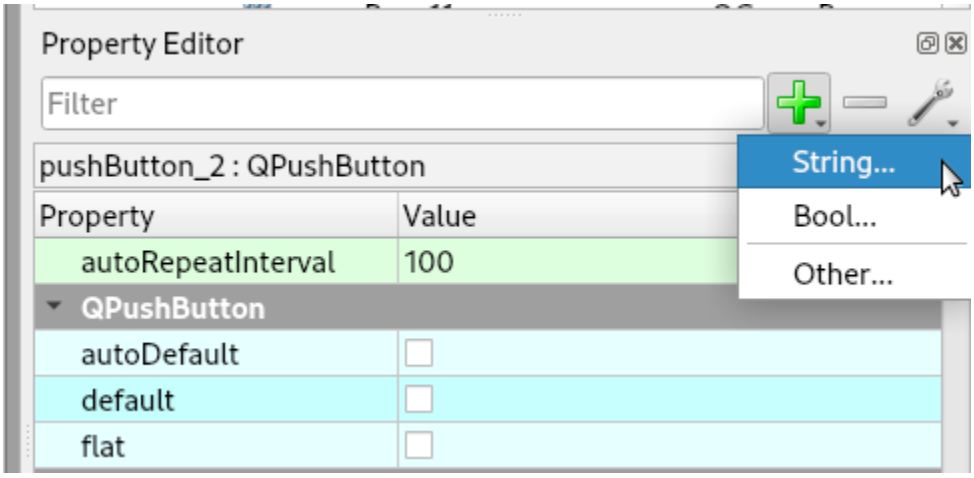
MDI buttons execute a MDI command when the button is pressed. These are created by adding two dynamic properties called *function* and *command* to a QPushButton.

Note: If the *command* property is not found, the button will not be enabled!

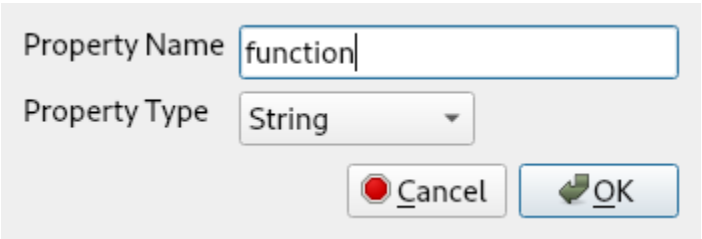
Select the button then create a dynamic property by pressing the green plus sign in the Property Editor



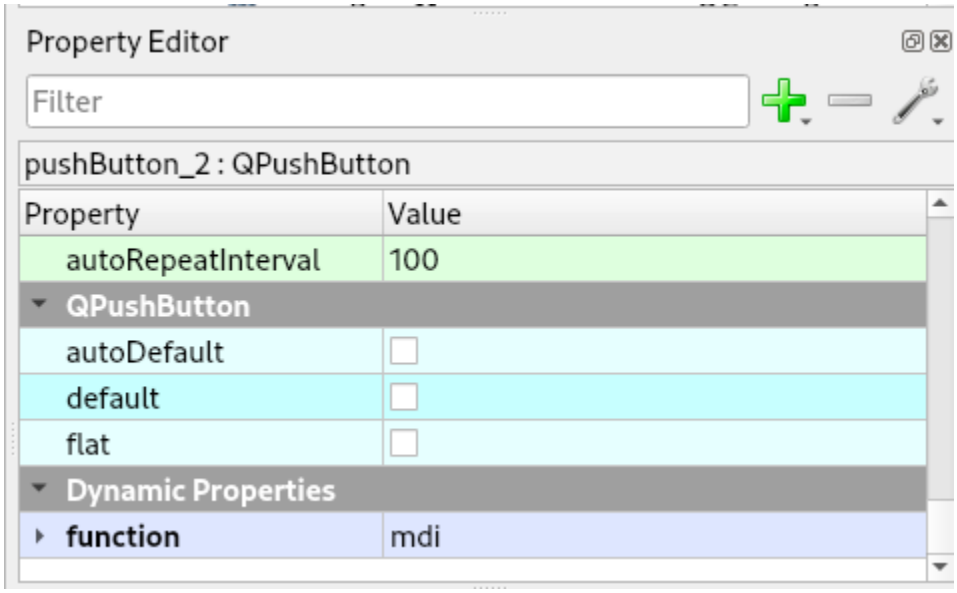
Then select *string*:



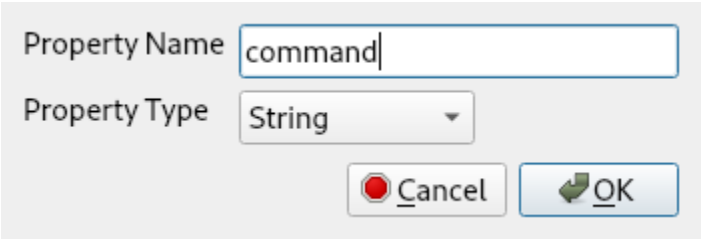
Name the property *function* and click OK



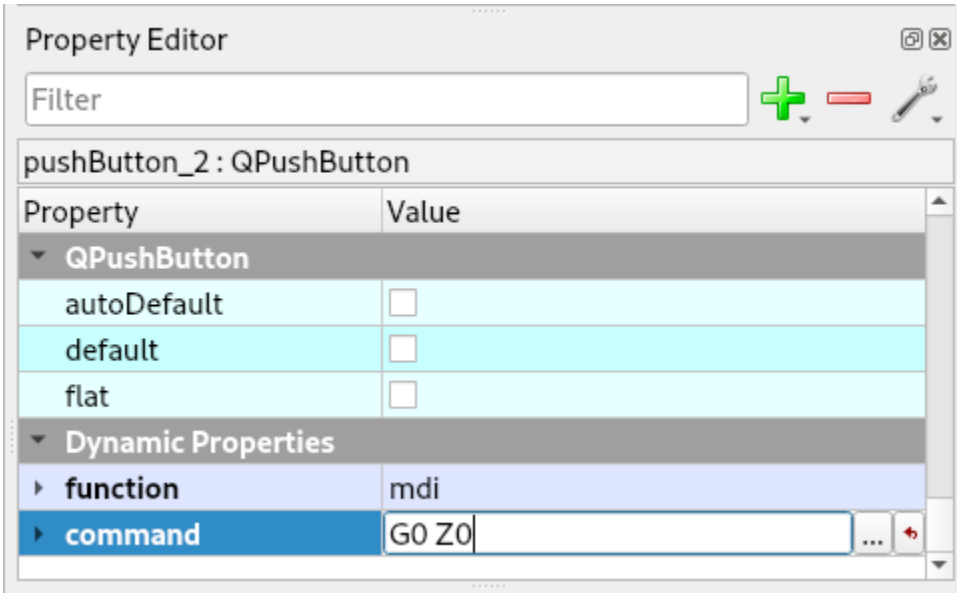
Set the value of the property to *mdi*



Add a property called *command*



Set the value of the property to your valid MDI command



CHAPTER

THIRTEEN

SPINDLE

Video Tutorial

13.1 Spindle Status

Spindle status labels show the current status of the item.

Table 1: Spindle Status Labels

Control Function	Object Type	Object Name
Spindle Brake	QLabel	spindle_brake_0_lb
Spindle Direction	QLabel	spindle_direction_0_lb
Spindle Enabled	QLabel	spindle_enabled_0_lb
Spindle Override Enabled	QLabel	spindle_override_enabled_0_lb
Spindle Commanded Speed	QLabel	spindle_speed_0_lb
Spindle Speed LCD	QLCDNumber	spindle_speed_0_lcd
Spindle Override Percent	QLabel	spindle_override_0_lb
Spindle Homed	QLabel	spindle_homed_0_lb
Spindle Orient State	QLabel	spindle_orient_state_0_lb
Spindle Orient Fault	QLabel	spindle_orient_fault_0_lb
Current S word Setting	QLabel	settings_speed_lb
Spindle Actual Speed	QLabel	spindle_actual_speed_lb
Spindle Speed Setting	QLabel	spindle_speed_setting_lb

Note: Spindle Commanded Speed does not show override. Spindle Actual Speed is actual speed including override.

Note: The digitCount property of the LCD must be large enough to display the whole number.

On start-up, Flex will check for the following items in the [SPINDLE_0] section of the .ini file

If *INCREMENT* is not found, Flex will look in the .ini [DISPLAY] section for *SPINDLE_INCREMENT* and if not there will default the increment to 10 for spindle faster/slower control buttons and spindle step for the QSpinBox.

If *MIN_FORWARD_VELOCITY* is found, it will be used to set the QSpinBox minimum setting. If not found, the minimum setting will be 0.

If *MAX_FORWARD_VELOCITY* is found, it will set the QSpinBox maximum setting. If not found, the maximum setting will be 1000.

INCREMENT will also set the QSpinBox single step when using the up/down arrows.

13.2 Spindle Controls

The following items control the spindle on/off direction and speed

Table 2: Spindle Status Labels

Control Function	Object Type	Object Name
Spindle Forward	QPushButton	spindle_fwd_pb
Spindle Reverse	QPushButton	spindle_rev_pb
Spindle Stop	QPushButton	spindle_stop_pb
Spindle Faster	QPushButton	spindle_plus_pb
Spindle Slower	QPushButton	spindle_minus_pb
Spindle Speed	QSpinBox	spindle_speed_sb

Note: The spindle can not be started with a spindle speed of zero.

13.3 Spindle Overrides

The spindle speed override is set using a QSlider. See the Status Labels above for spindle override status labels.

Table 3: Spindle Override

Control Function	Object Type	Object Name
Spindle Override	QSlider	spindle_override_sl

CHAPTER

FOURTEEN

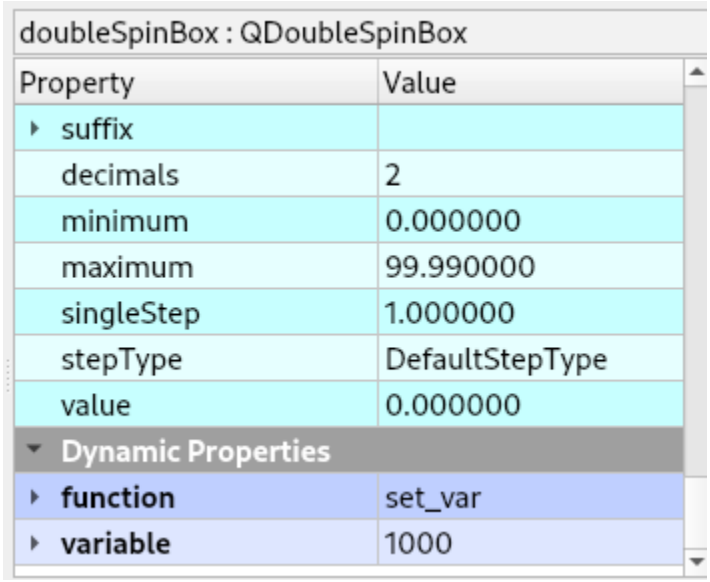
PARAMETERS

Parameter values in the var file can be set and watched for changes.

14.1 Setting Parameters

To set a user parameter value in the var file with a QDoubleSpinBox add a couple of string type Dynamic Properties. See *Dynamic Properties* The parameters 31 - 5000 are available for use in NC code programs. Replace *nnnn* with the variable number.

```
function set_var
variable `nnnn`
```



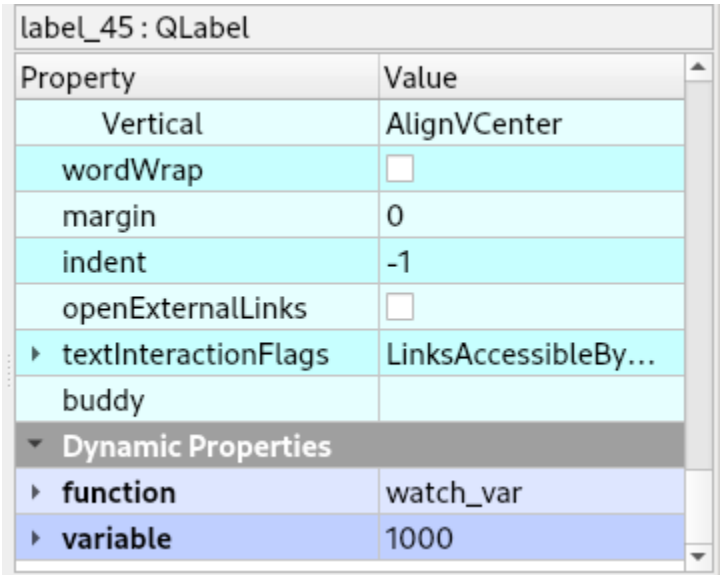
The user parameter must be in the var file. On startup Flex reads the var file and sets the value of the QDoubleSpinBox to that value. When you change the value of the QDoubleSpinBox the var file is updated with the new value. There is a 0.5 second timeout before the var file is updated to give time to type in a number.

The configuration must be out of E-Stop, Power On and Homed before the QDoubleSpinBox is enabled.

14.2 Watching Parameters

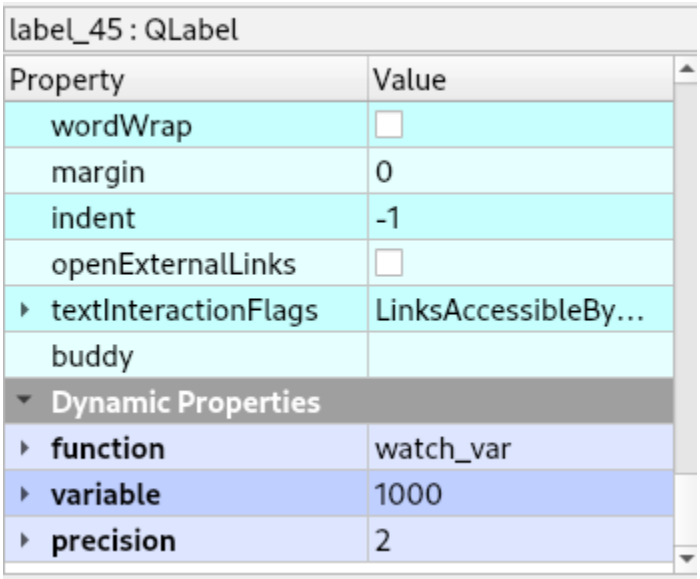
To watch the value of a user parameter a QLabel can be used with the following string type Dynamic Properties. See *Dynamic Properties*

```
function watch_var
variable `nnnn`
```



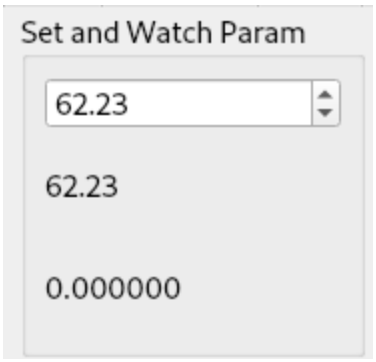
The value is updated at startup and any time the var file is updated.

The default precision for a parameter value is 6, to change the precision add a Dynamic Property called *precision* and set the value to the number of digits after the dot.



Note: The var file is not updated until the end of a NC file.

Example of setting and watching parameters


Set and Watch Param
62.23
62.23
0.000000

CHAPTER

FIFTEEN

PYTHON MODULE

Python Import Tutorial

To import a python module add the following to the INI [FLEXGUI] section using the name of the python file without the.py extension. The file name must be unique and can not be any python module name. You can have as many imports as you need to simplify your code.

```
[FLEXGUI]
IMPORT_PYTHON = testpy
```

Note: The module requires the .py extension to be able to import so the above module would be named testpy.py.

In each python file you import you must have a *startup* function where you make any connections from objects in the ui file to code in your module. The parent is passed to the startup function to give you access to all the objects in the GUI.

```
from functools import partial

def startup(parent):
        # connect a pushbutton without passing parent
        parent.my_test_pb.clicked.connect(test_1)
        parent.get_names_pb.clicked.connect(partial(get_names, parent))

        # connect a pushbutton and pass parent to the function
        parent.another_test_pb.clicked.connect(partial(test_2, parent))

def test_1():
        print('test 1')

def test_2(parent):
        # in this function you have access to all the objects in parent
        print(f'test 2 {parent.another_test_pb.text()}')

def get_names(parent):
        # get all the object names from the parent
        print(dir(parent))
```

15.1 Timer

A user timer is provided for use in the user python module.

```
from functools import partial

def startup(parent):
        parent.user_timer.timeout.connect(testit)
        parent.conn_pb.setEnabled(False) # prevent another connection
        parent.disc_pb.clicked.connect(partial(disc, parent))
        parent.conn_pb.clicked.connect(partial(conn, parent))
        parent.start_pb.clicked.connect(partial(start, parent))
        parent.stop_pb.clicked.connect(partial(stop, parent))

def testit():
        print('testing')

def disc(parent):
        parent.user_timer.timeout.disconnect(testit)
        parent.conn_pb.setEnabled(True) # allow a connection
        parent.disc_pb.setEnabled(False) # prevent trying to disconnect

def conn(parent):
        parent.user_timer.timeout.connect(testit)
        parent.conn_pb.setEnabled(False) # prevent trying to connect
        parent.disc_pb.setEnabled(True) # allow a disconnect

def start(parent):
        parent.user_timer.start(1000) # milliseconds

def stop(parent):
        parent.user_timer.stop()
```

CHAPTER

SIXTEEN

PROBING

16.1 Probe Enable

Add a QPushButton named *probing_enable_pb* and if it is found it will be set as a toggle button. The button will only be enabled when the machine is homed and not running a program. The button is set to checkable in code so it can be styled with :checked and :enabled pseudo-states among others.

Note: The *probing_enable_pb* requires at least one object that starts with *probe_* to be enabled.

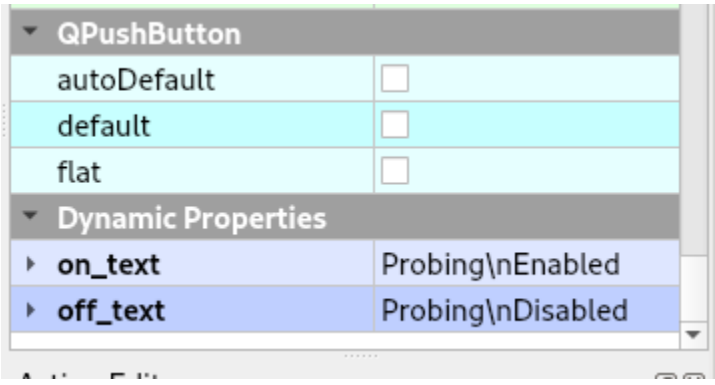
```
QPushButton#probing_enable_pb:enabled:checked {
        color: white;
        background-color: red;
}

QPushButton#probing_enable_pb:enabled {
        color: green;
        background-color: yellow;
}
```

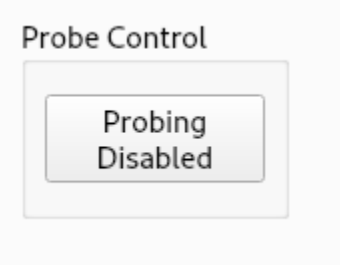
Note: A QPushButton with checkable set to true does not have an unchecked pseudo-state

For more style sheet options see the *StyleSheet* section

The text on the Probe Enable button can be set by adding two Dynamic Properties named *on_text* and *off_text*. Both must be present or no change will take place as there is no default text for the Probe Enable button. See *Dynamic Properties*



This is what the button would look like with the above settings.



The Probe Enable button can have a LED indicator. See *LED Buttons* for information on LED buttons

16.2 Function

When the *probing_enable_pb* is toggled *OFF*, any widget with an object name that starts with *probe_* will be disabled.

When the *probing_enable_pb* is toggled *ON* the widgets that start with *probe_* will be enabled. In addition, spindle controls will disabled, spindle speed set to 0, run controls will be disabled, and MDI controls will be disabled.

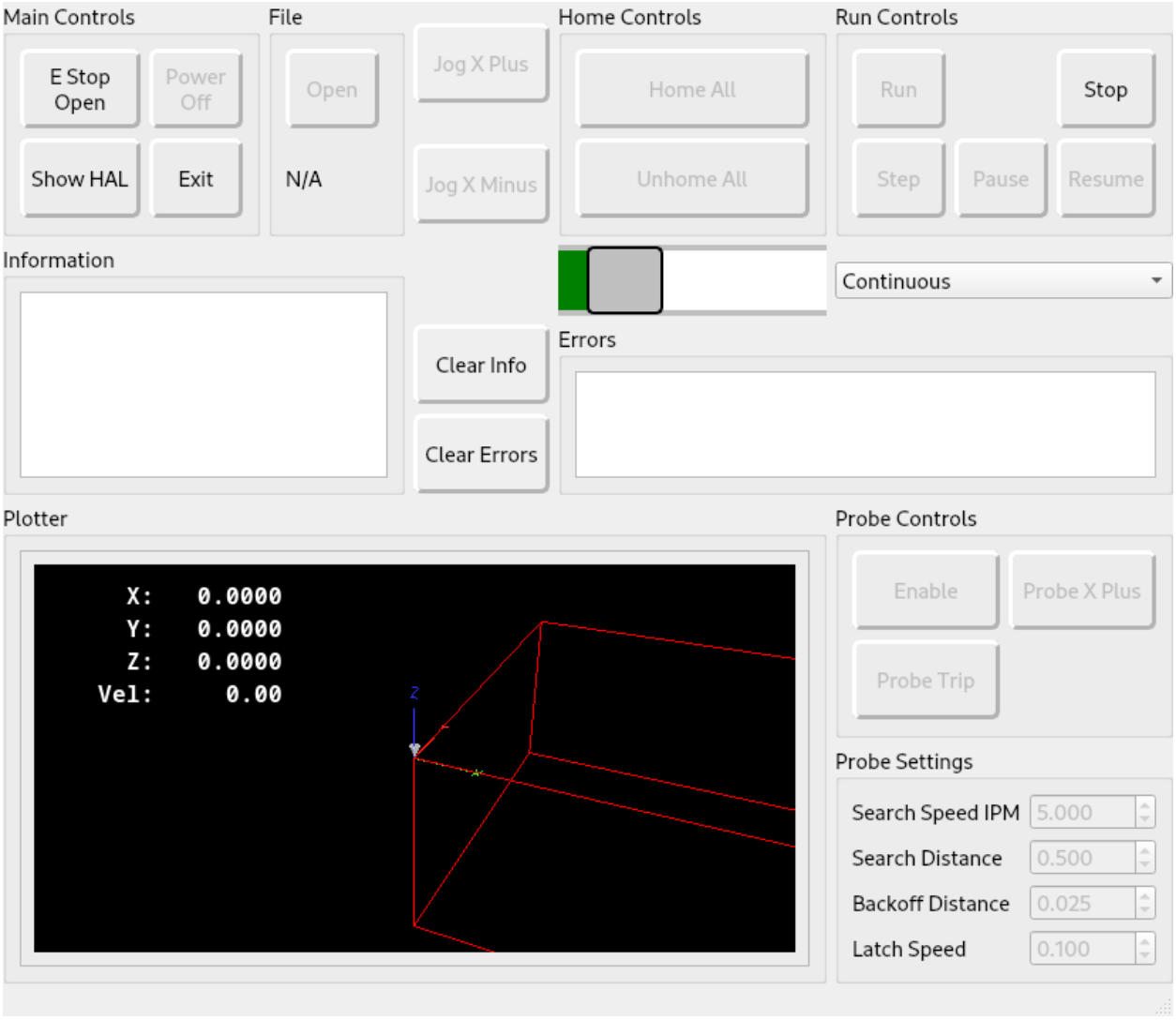
QPushButtons with an objectName that start with *probe_* and configured as a *MDI Button* (to launch the probing subroutines) will be enabled when probing is enabled and disabled when probing is disabled.

You can create a HAL *HAL LED Button* to use in your probing subroutine. Set the objectName to start with *probe_* and it will be enabled and disabled with the probe buttons.

If you're using a touch-screen, add a Dynamic Property named *input* and set the value to *touch*.

16.3 Example

A minimal example is in the Flex Examples in the Features directory



To run the example, close the E Stop, turn Power on, Home all, then toggle the Enable button. When probing is enabled many other controls are disabled including the spindle.

To test the probe routine press the Probe X Plus button and the X axis will start to move in the positive direction. If you do nothing when the Search Distance is reached you will get an error that the G38.2 move finished without making contact, which is expected.

Press the Probe X Plus button again and after it starts to move, press the Probe Trip button, the X axis will back off the Backoff Distance and start to move in the positive direction again. Pressing the Probe Trip button again will end the probe simulation. The debug information will show up in the Information window.

16.4 Subroutine

The probe subroutines use the values from the Probe Settings spin boxes. To use these values, you need to make the spin box a HAL pin. See the *HAL LED Button* example in the HAL section.

The subroutine is located in a directory called *subroutines* that is in the configuration directory. The ini's [RS274NGC] SUBROUTINE_PATH sets the path that LinuxCNC looks for subroutines. Notice the leading ./ specifices that the path to the current directory is where the subroutine directory is.

```
SUBROUTINE_PATH = ./subroutines
```

The example files used are the following; notice that the xplus.ngc is in the ./subroutines directory

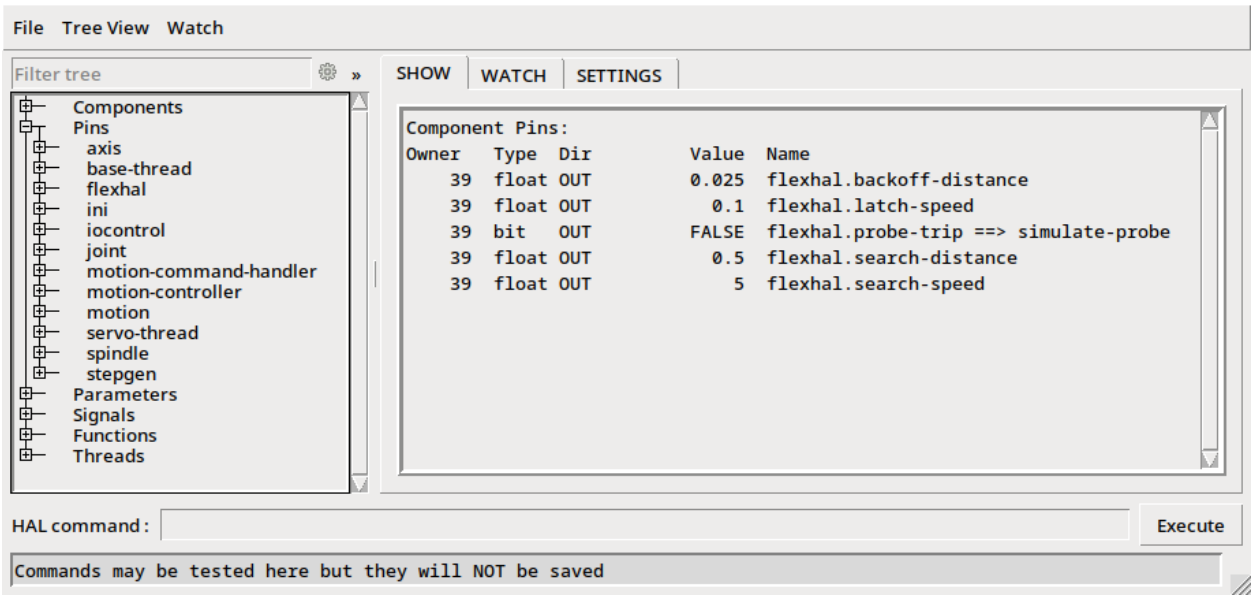
```
├── main.hal
├── parameters.var
├── parameters.var.bak
├── postgui.hal
├── probe.ini
├── probe.ui
├── README
├── sim_axis_probe.ini
├── subroutines
│   └── xplus.ngc
└── tool.tbl
```

In your subroutine you can use user parameters instead of using HAL pins. See the *Parameters* section.

The subroutine is a normal LinuxCNC subroutine. The magic is how you get the values from HAL pins with *#<_hal[pin_name]>* where pin_name is the actual pin name in HAL.

```
(filename xplus.ngc)
(HAL pins #<_hal[pin_name]>)
(G90 absolute distance mode G91 incremental distance mode)
o<xplus> sub
        (msg, xplus subroutine)
        G20
        ; initial search
        G91 G38.2 F#<_hal[flexhal.search-speed]> X#<_hal[flexhal.search-distance]>
        ;5061-5069 - Coordinates of a G38 probe result (X, Y, Z, A, B, C, U, V & W)
        (debug, Probe Contact at #5061)
        ; back off using #5061 to compensate for over travel on the probe
        G90 G0 X[#5061-#<_hal[flexhal.backoff-distance]>]
        ; final probe at latch speed
        G91 G38.2 F#<_hal[flexhal.latch-speed]> X[#<_hal[flexhal.backoff-distance]> + 0.
↪02]
        (debug, Probe Contact at #5061)
o<xplus> endsub
M2
```

Looking at the Halshow window which pops up when you press the Show HAL button, you can see the flexhal pin names for each spin box and for the Probe Trip button. Also notice that the Probe Trip button is connected to a signal which is connected to motion.probe-input in the postgui.hal file


File
Tree View
Watch
Filter tree
»
Components
Pins
axis
base-thread
flexhal
ini
iocontrol
joint
motion-command-handler
motion-controller
motion
servo-thread
spindle
stepgen
Parameters
Signals
Functions
Threads
SHOW
WATCH
SETTINGS
Component Pins:
Owner Type Dir Value Name
39 float OUT 0.025 flexhal.backoff-distance
39 float OUT 0.1 flexhal.latch-speed
39 bit OUT FALSE flexhal.probe-trip ==> simulate-probe
39 float OUT 0.5 flexhal.search-distance
39 float OUT 5 flexhal.search-speed
HAL command :
Execute
Commands may be tested here but they will NOT be saved

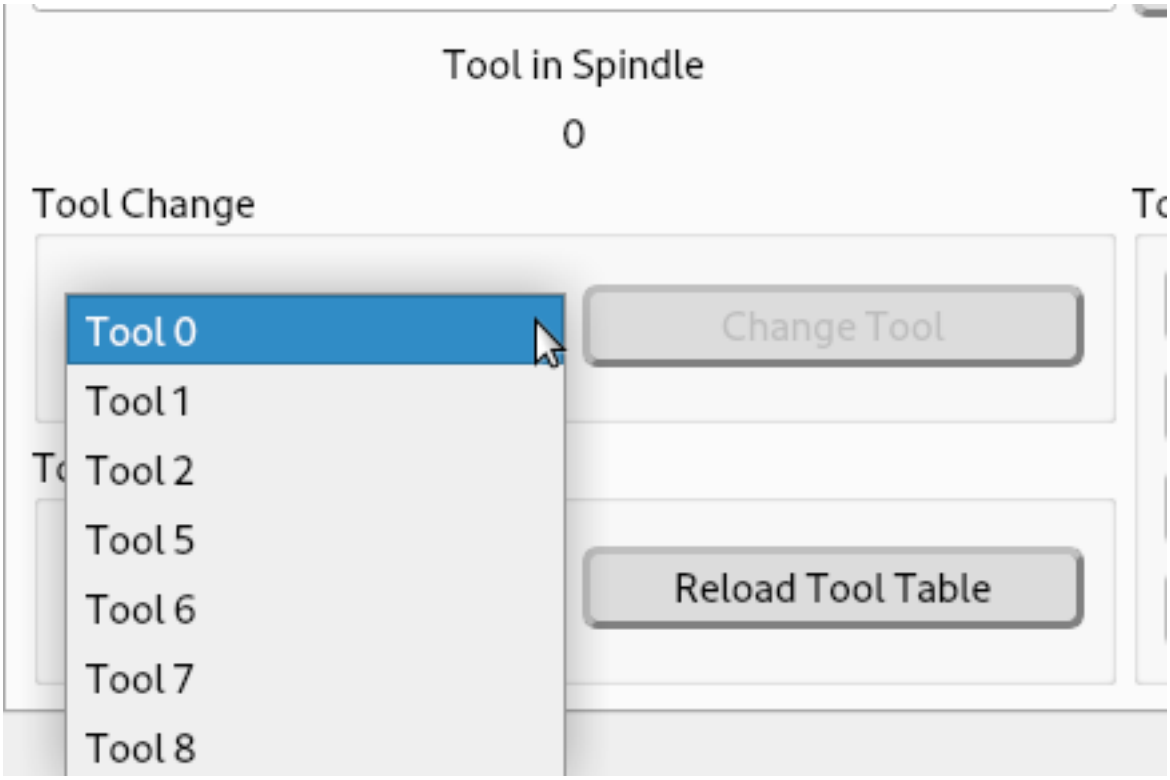
CHAPTER

SEVENTEEN

TOOLS

Tools Tutorial

17.1 Tool Change

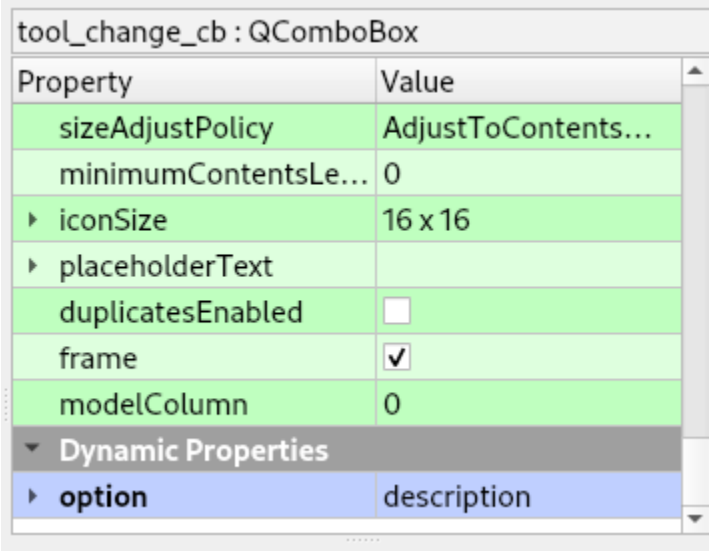


A tool change QPushButton, with a QComboBox to select the tool number to change to, is done with QPushButton named *tool_change_pb* and a QComboBox named *tool_change_cb*. The tool change combobox will automatically be populated with all the tools found in the tool table.

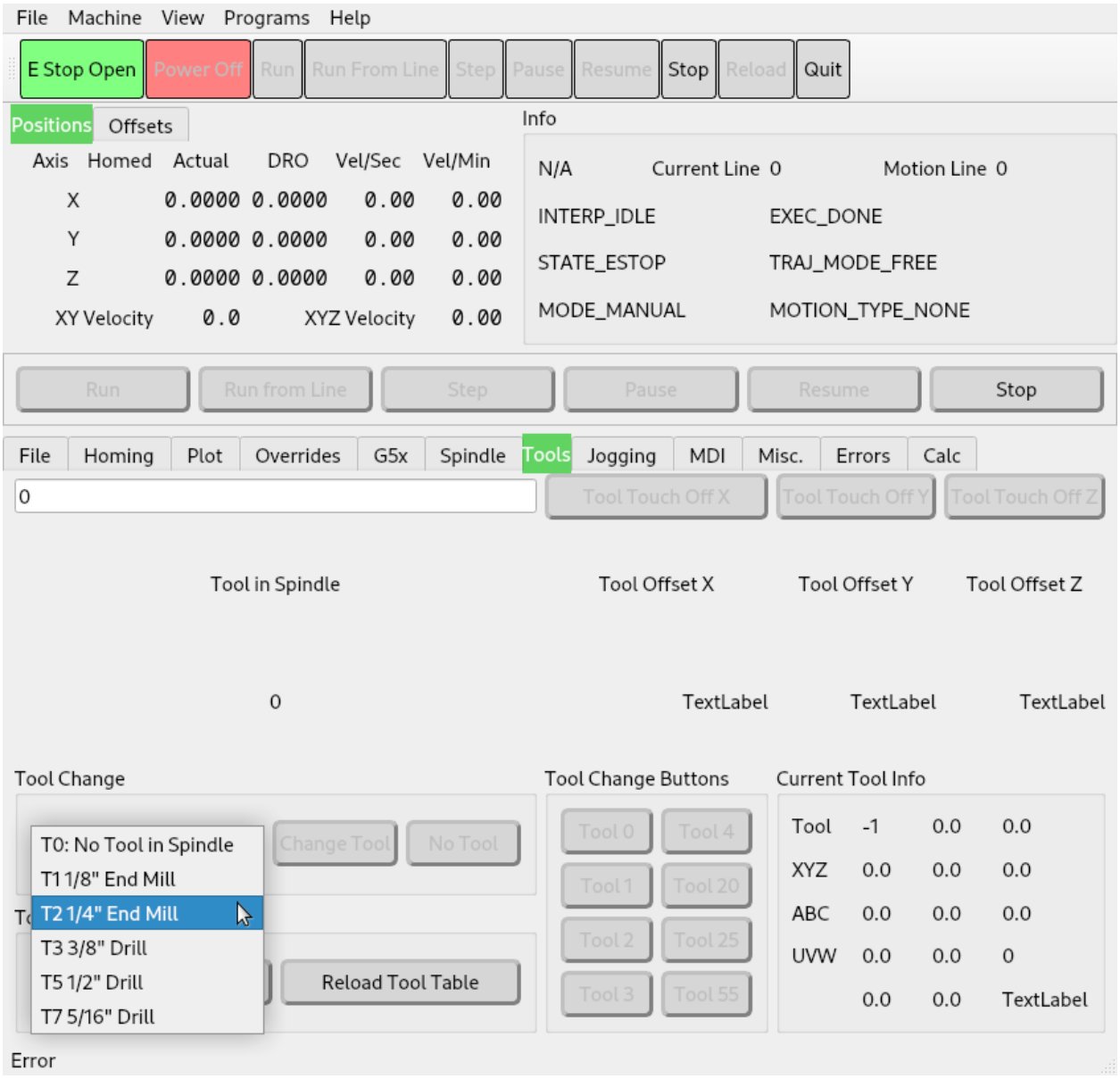
Table 1: Tool Change Controls

Control Function	Object Type	Object Name
Tool Change	QPushButton	tool_change_pb
Tool Selector	QComboBox	tool_change_cb

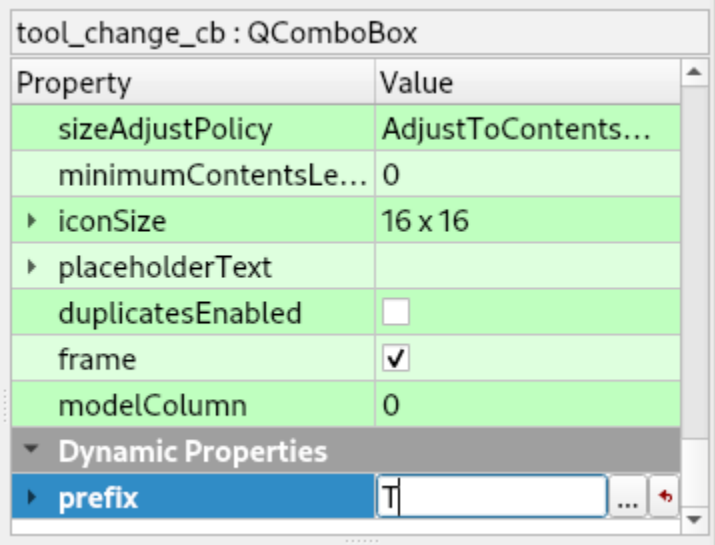
To add the description of the tools to the tool change combo box add a Dynamic Property named *option* and set the value to *description*. See *Dynamic Properties*



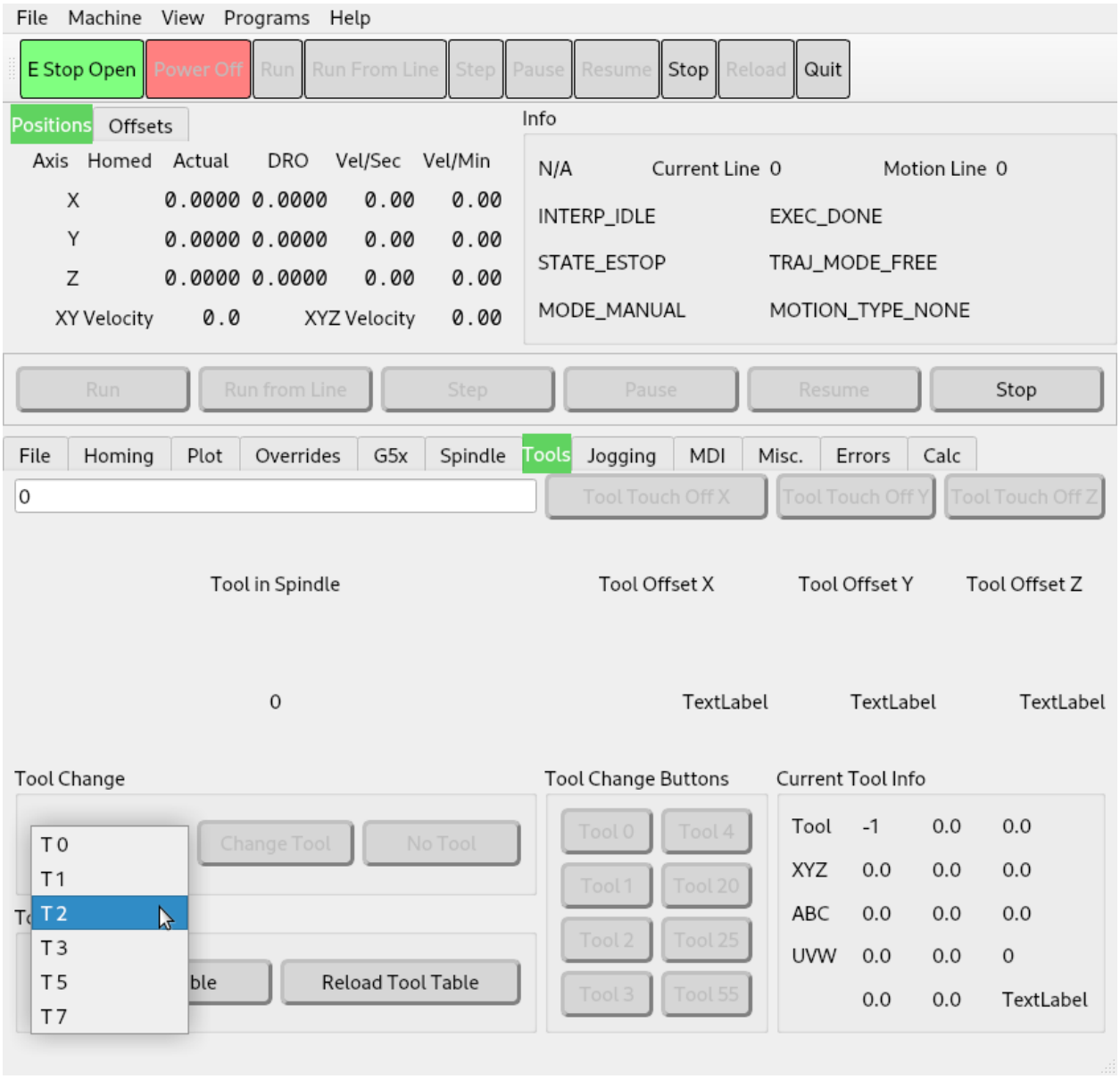
The description from the tool table will be appended to the tool number.



If you have limited space you can define the tool prefix by adding a Dynamic Property named *prefix* and set the value to the prefix you want.



The tool number will follow the prefix.



Note: Only one option can be used, if option is found it is used and prefix will be ignored.

17.2 Manual Tool Change

All that is needed to add a manual tool change is to add the following to the ini file in the [FLEXGUI] section.

```
[FLEXGUI]
MANUAL_TOOL_CHANGE = True
```

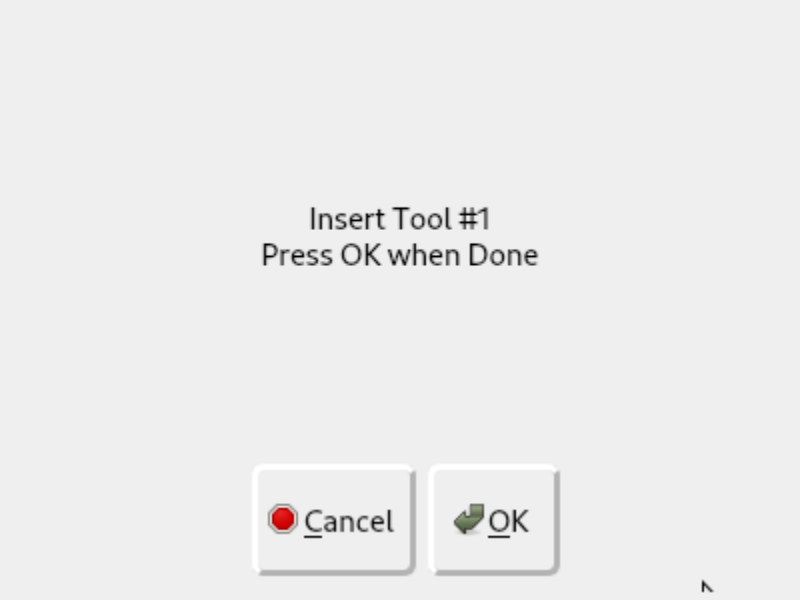


Fig. 1: This is without using a theme.

Fig. 2: This is the blue-touch theme.

Fig. 3: This is the dark-touch theme.

Warning: You can't use the hal_manualtoolchange at the same time as the built in Flex Manual Tool Change, you must comment out all the hal_manualtoolchange lines or remove them.

17.3 Manual Tool Change Error

If you get an error that hal_manualtoolchange component exists look in your hal files for the hal_manualtoolchange lines in the Manual Tool Change Option below.

If you're using a copy of one of the Axis sims the hal_manualtoolchange component can be hard to find. It's recommended that you start with a simple configuration like the Flex example simple-sim.

17.4 Manual Tool Change Option

The HAL Manual Tool Change requires at least the following HAL code in the main hal file if not using the builtin Flex Manual Tool Change above.

```
# manual tool change
loadusr -W hal_manualtoolchange
net tool-change iocontrol.0.tool-change => hal_manualtoolchange.change
net tool-changed iocontrol.0.tool-changed <= hal_manualtoolchange.changed
net tool-number iocontrol.0.tool-prep-number => hal_manualtoolchange.number
net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
```

17.5 Tool Change Button

Tool change QPushButtons can be used to change tools without a spinbox by adding up to 99 QPushButtons named *tool_change_pb_n*. With *n* being the number of the tool you wish to change to using that button

Table 2: Tool Change Buttons

Control Function	Object Type	Object Name
Tool Change Button	QPushButton	tool_change_pb_(n)

17.6 Tool Touchoff

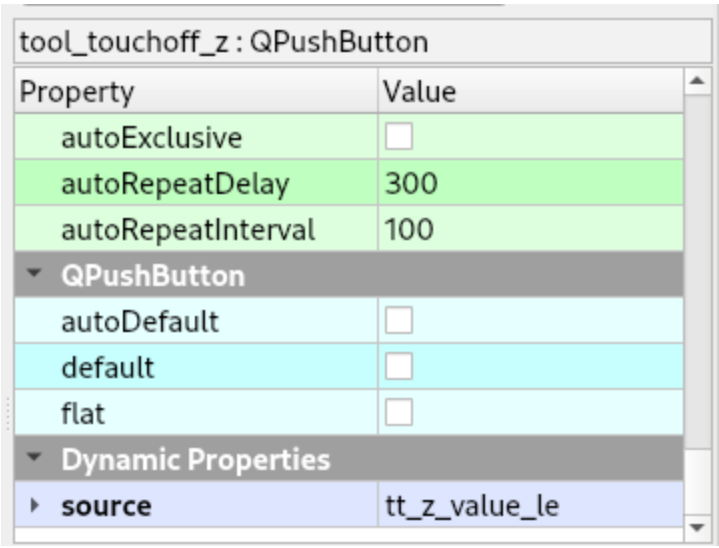
To touch-off a tool to an axis, use a tool-touch-off QPushButton and a QLineEdit to enter the value of the touch off.

Table 3: Tool Touchoff Controls

Control Function	Object Type	Object Name
Tool Touch Off Value	QLineEdit	tool_touchoff_le
Tool Touch Off	QPushButton	tool_touchoff_(axis letter)

Optionally you can have a QLineEdit for each axis for tool touch off. Add a Dynamic Property named *source* to the tool touch off button and set the value to the name of the QLineEdit that is the source for that touch off button. See *Dynamic Properties*

Tool touch off QLineEdit for each axis.



17.7 Tool Touchoff Selected Axis

To have Axis style tool touch off add a QPushButton named *tool_touchoff_selected_pb*. You must have at least one QRadiobutton for an axis to select.

Table 4: Tool Touchoff Selected Widgets

Function	**Widget**	**Name**
Axis Select (0-8)	QRadioButton	axis_select_(0-8)
Tool Touchoff	QPushButton	tool_touchoff_selected_pb

17.8 Current Tool Status

Current Tool status of the tool loaded in the spindle. All the labels can have a Dynamic Property called *precision* with the number of digits you wish to show. The *tool_id_lb* and the *tool_orientation_lb* are integers.

Table 5: Tool Table Status Labels

tool_id_lb	tool_xoffset_lb	tool_yoffset_lb
tool_zoffset_lb	tool_aoffset_lb	tool_boffset_lb
tool_coffset_lb	tool_uoffset_lb	tool_voffset_lb
tool_woffset_lb	tool_diameter_lb	tool_frontangle_lb
tool_backangle_lb	tool_orientation_lb	

CHAPTER

EIGHTEEN

COORDINATE SYSTEMS

Coordinate System Tutorial

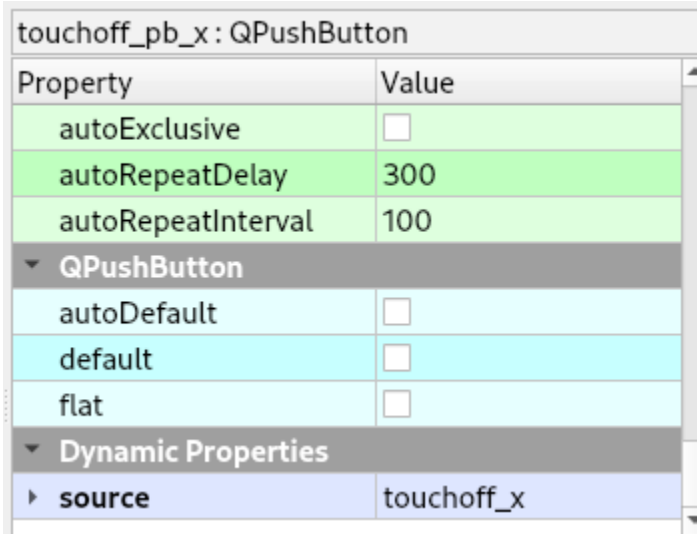
18.1 Coordinate System Touchoff

To touch-off an axis, use a QPushButton and QLineEdit to set the touch-off value. Optionally you can have a QComboBox to select the Coordinate System to touch off to.

Table 1: Coordinate System Touch Off Controls

Control Function	Object Type	Object Name
Touch Off Axis	QPushButton	touchoff_pb_(axis letter)
Touch Off Value	QLineEdit	touchoff_le
Coordinate System	QComboBox	touchoff_system_cb

Optionally you can have a QLineEdit for any axis by adding a string type Dynamic Property named *source* to the QPushButton and the value contains the object name of the QLineEdit that you want to use. See *Dynamic Properties*



As you can see you can have a QLineEdit for each axis.

18.2 Change Coordinate System

To change the coordinate system via a button, use a change_cs_`n` QPushButton where *n* is 1-9 for G54 through G59.3

Table 2: Coordinate System Change Buttons

Control Function	Object Type	Object Name
Change Coordinate System	QPushButton	change_cs_(n)

CHAPTER

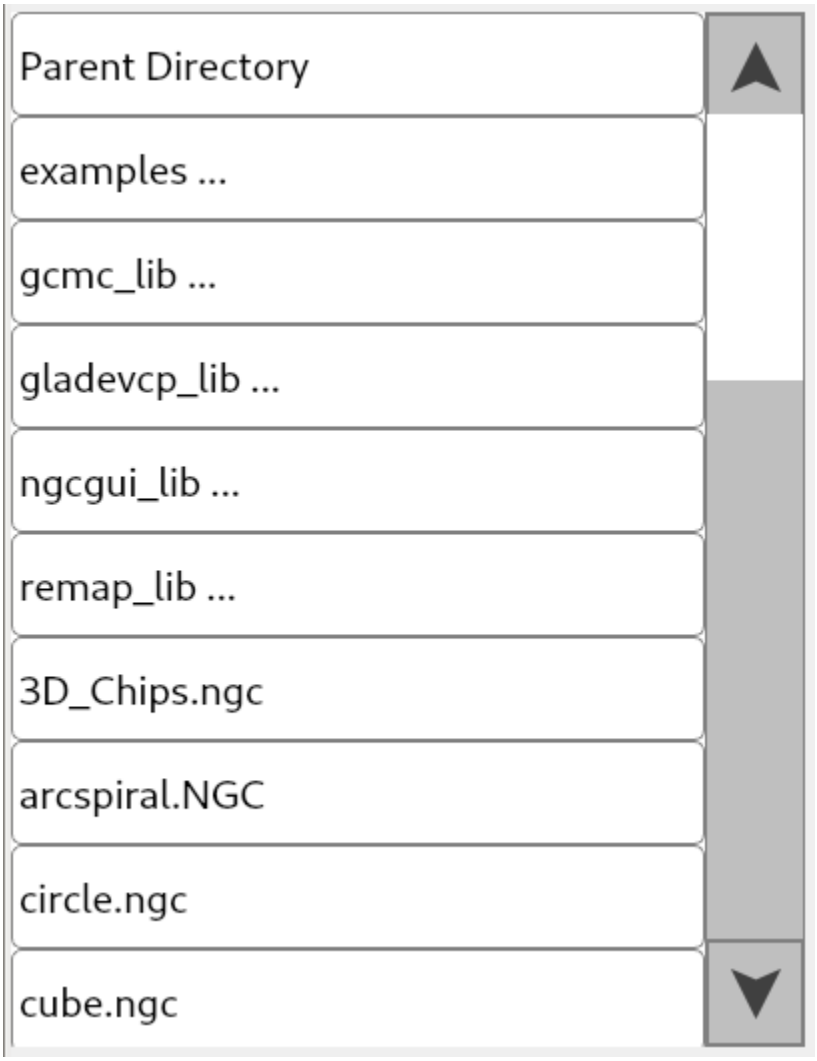
NINETEEN

MISCELLANEOUS ITEMS

19.1 File Selector

Add a QListWidget and name it *file_lw*, this can be used with a touch screen by specifying the touch input. A single left-click or touch is all that's needed to use the *File Selector*. A left-click or touch on a directory will change to that directory. A left-click or touch on the up or down arrow will move the list by one. A left-click or touch in between an arrow and the slider will move the list by one page. Touch-and-hold to move the slider.

If you use the touch input, the selector looks like this



Note: Make sure you use a QListWidget and not a QListView for the file selector.

File, Error and Information Tutorial

19.2 Code Viewer

To add a code viewer, add a *QPlainTextEdit* from Input Widgets and name it *gcode_pte*

```
( AXIS "splash G-code" Not intended for actual milling )
( To run this code anyway you might have to Touch Off the Z axis)
( depending on your setup. As if you had some material in your mill... )
( Hint jog the Z axis down a bit then touch off )
( Also press the Toggle Skip Lines with "/" to see that part )
( If the program is too big or small for your machine, change the scale below )
( LinuxCNC 19/1/2012 2:13:51 PM )
#<depth>=2.0
#<scale>=1.0
G21 G90 G64 G40
G0 Z3.0
( engraving )
```

19.3 Code Viewer Controls

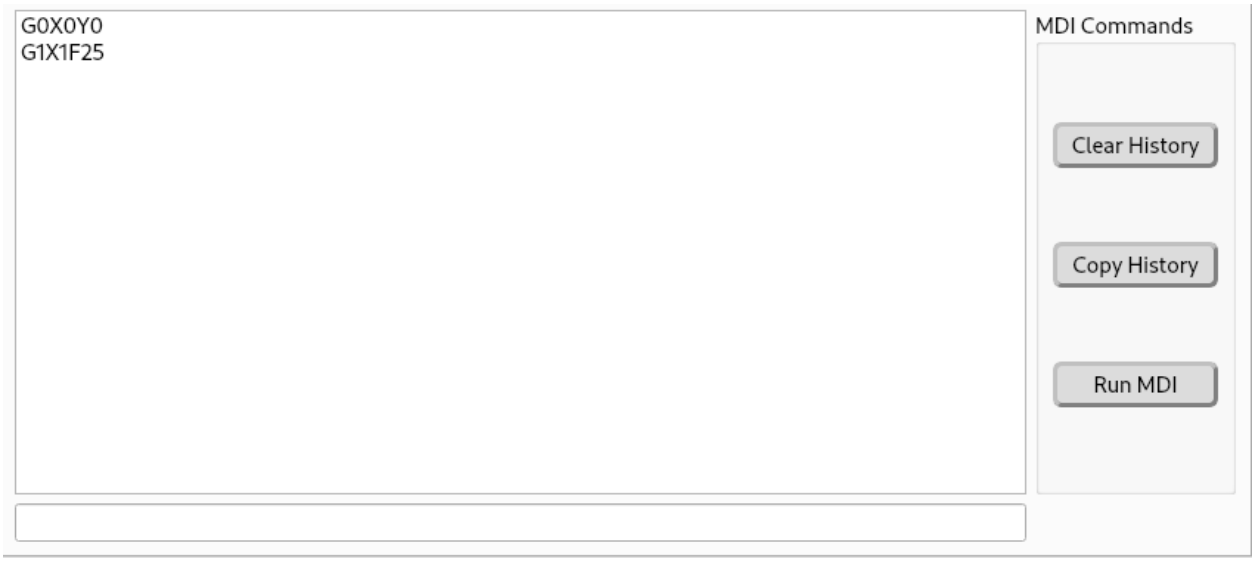
The Code Viewer allows you to edit the file in Flex GUI without using an external text editor. You can save the current code to the current file name, save the current code with a new file name and you can search the code.

Table 1: Code Viewer Controls

Function	Type	Object Name
Save	QPushButton	save_pb
Save As	QPushButton	save_as_pb
Search	QPushButton	search_pb

19.4 MDI Viewer

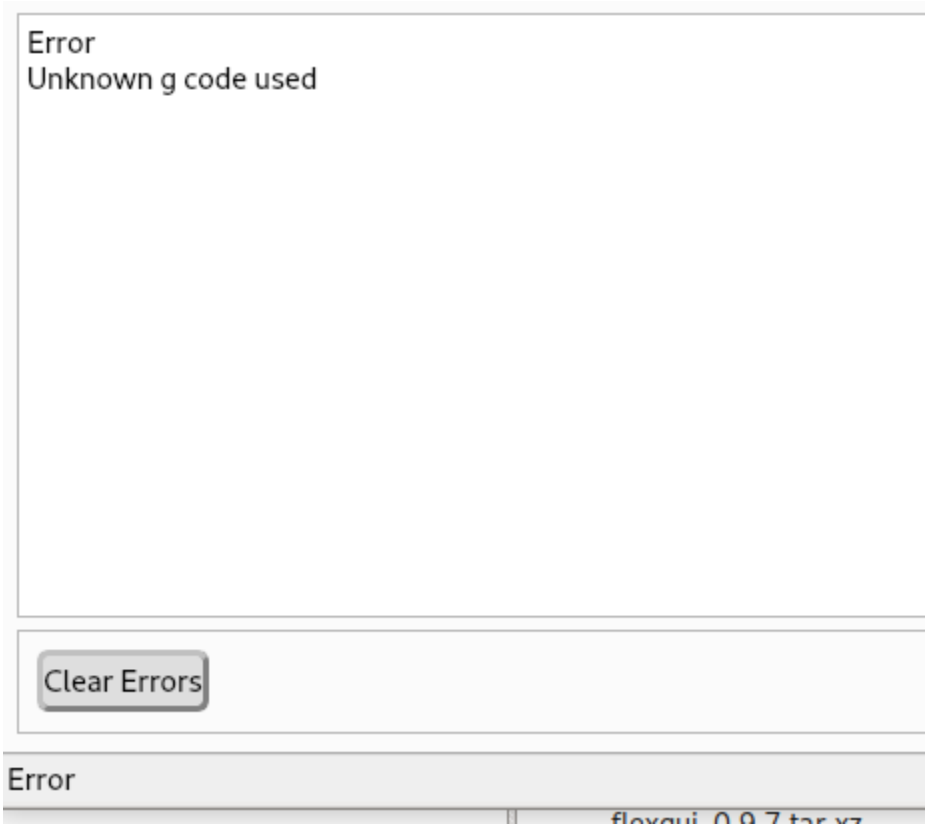
To add a MDI viewer, add a *QListWidget* from Item Widgets and name it *mdi_history_lw*



To enter MDI commands, add a Line Edit and name it *mdi_command_le*.

19.5 Error Viewer

To add an error viewer, add a *QPlainTextEdit* from Input Widgets and name it *errors_pte*



To clear the error history, add a QPushButton and set the objectName to *clear_errors_pb*.

To copy the errors to the clipboard, add a QPushButton and set the object name to *copy_errors_pb*.

Warning: The error viewer must be a QPlainTextEdit not a QTextEdit.

19.6 Information Viewer

To add an information viewer, add a *QPlainTextEdit* from Input Widgets and name it *info_pte*. Information messages from MSG, DEBUG and PRINT will show up in the Information Viewer if it exists.

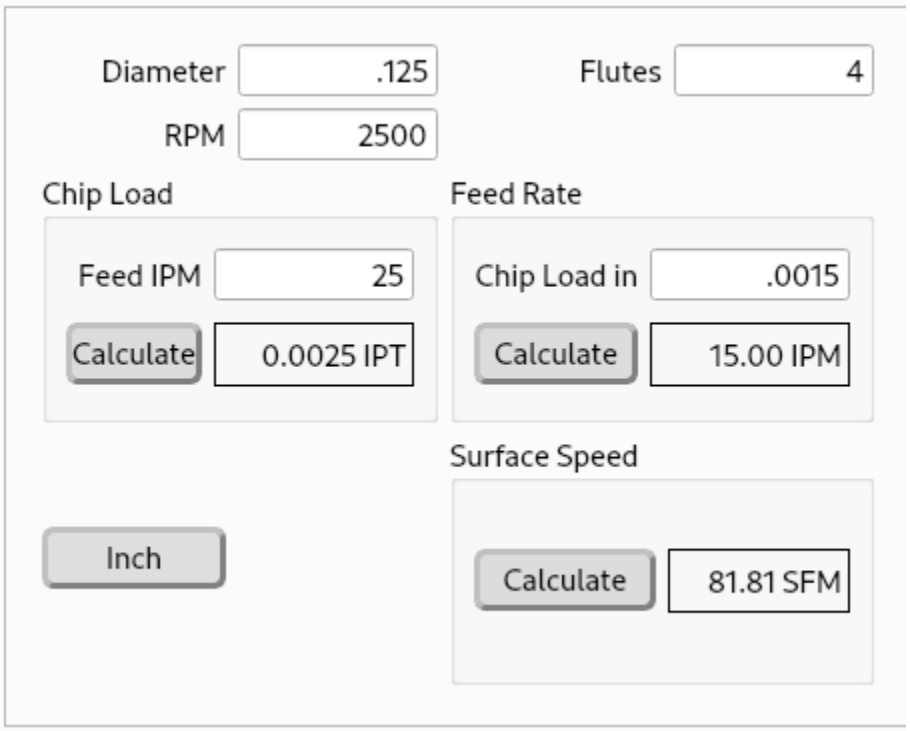
If *info_pte* is not found and the *errors_pte* is found, then information messages will show up in the Error Viewer.

To clear the information viewer, add a QPushButton and name it *clear_info_pb*.

Warning: The information viewer must be a QPlainTextEdit not a QTextEdit.

19.7 Speed & Feed Calculators

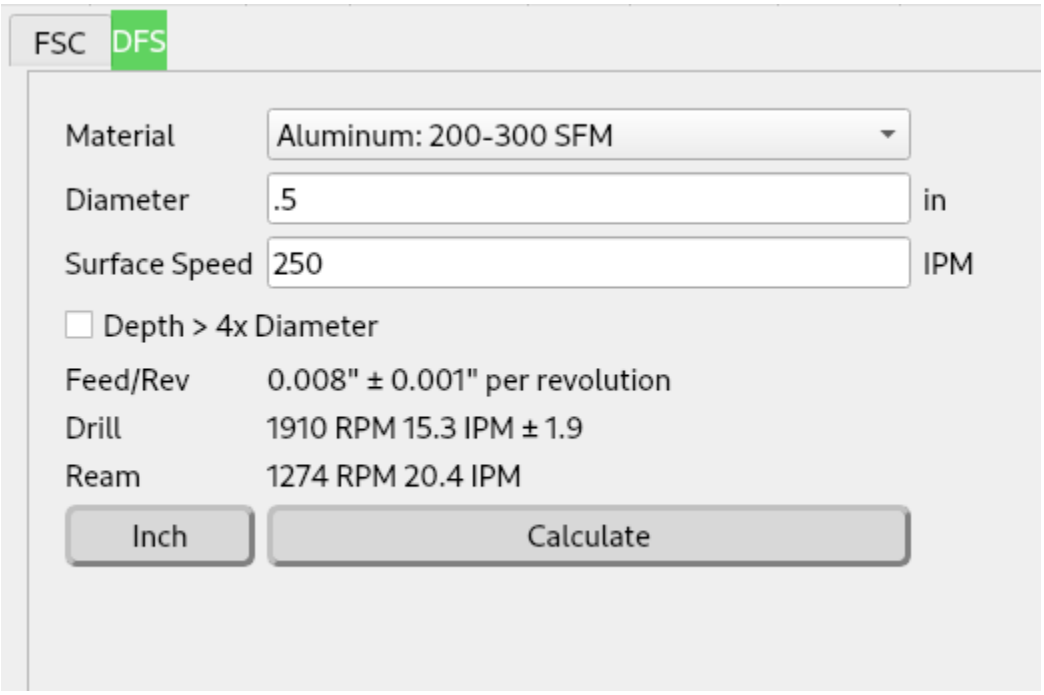
To add a milling Speeds and Feeds Calculator, add a *QFrame* or *QWidget* and set the Object Name to *fsc_container*



To make the entry boxes touch-screen aware, add a Dynamic Property called *mode* and set the value to *touch*. Then when you touch an entry field, a numeric popup will show up to allow you to enter the value without a keyboard. See *Dynamic Properties*

To add a Drill Feed and Speed calculator, add a *QFrame* or *QWidget* and set the Object Name to *dsf_container*.

To make the entry boxes touch-screen aware, add a Dynamic Property called *mode* and set the value to *touch*. Then when you touch it, a numeric popup will appear, allowing you to enter the numbers



19.8 Help System

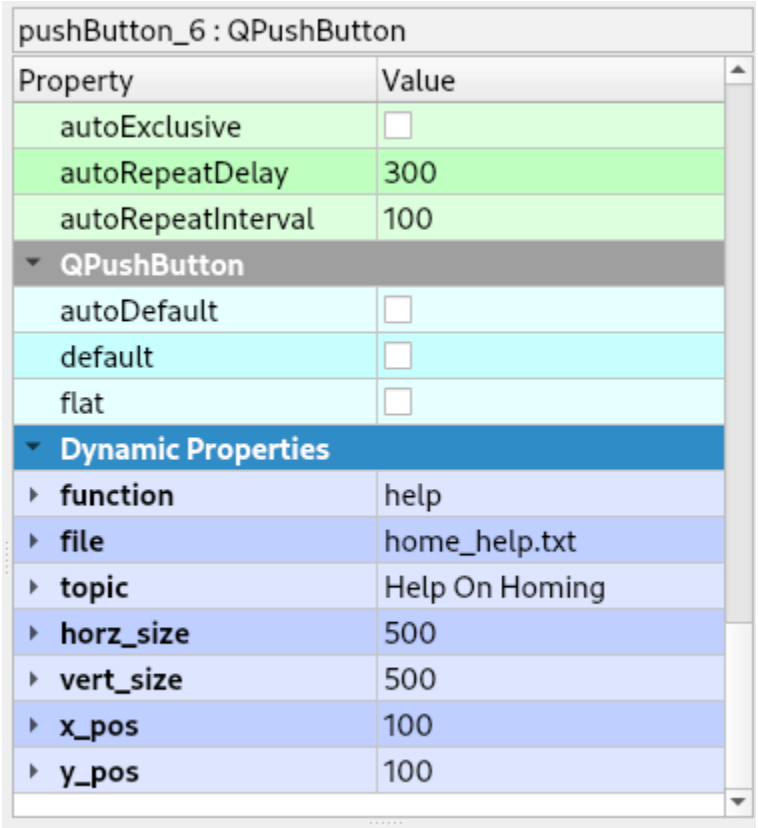
A QPushButton can be setup to launch a Help dialog which contains text from a file in the configuration directory. A help button can be placed on multiple places with different file names. Only one Help dialog can be open at a time.

Table 2: Help Button Dynamic Properties

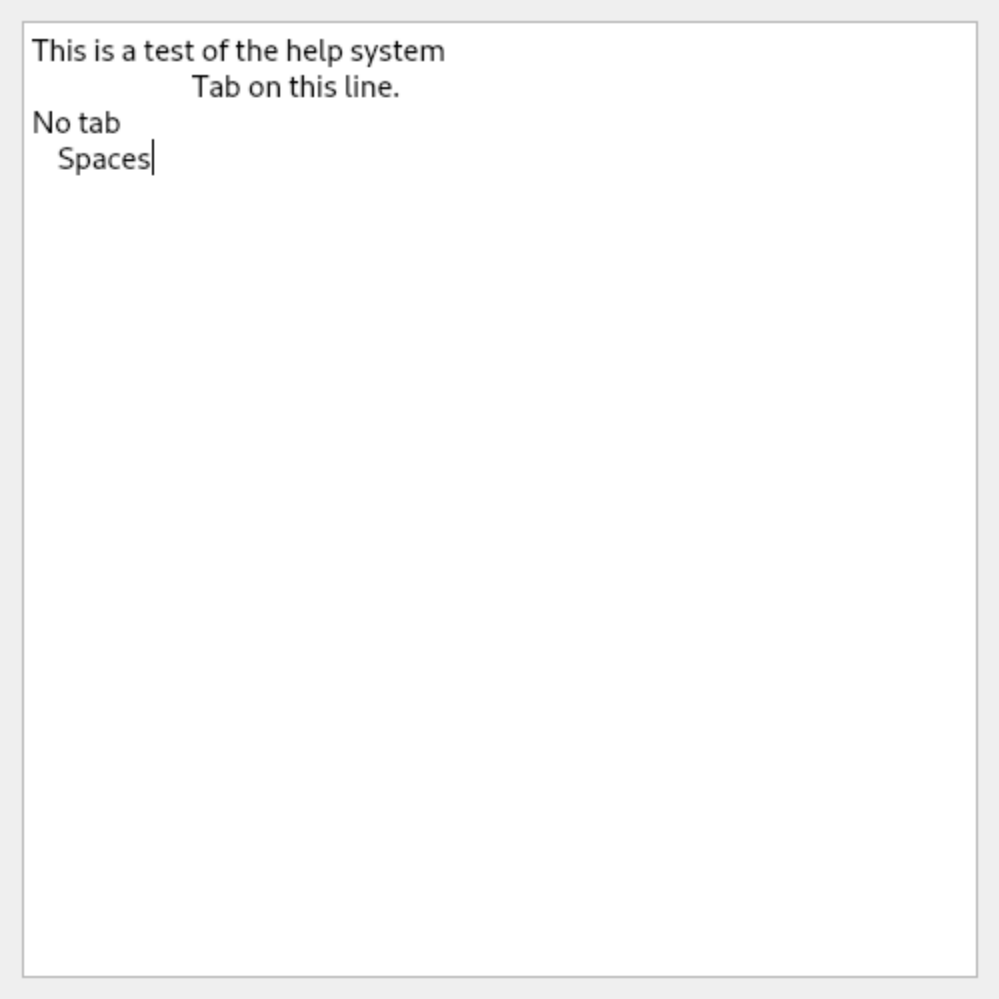
Property Type	Property Name	Value
String	function	help
String	file	file name
String	topic	title of topic
String	x_pos	x location of upper left corner
String	y_pos	y location of upper left corner
String	horz_size	width
String	vert_size	height

Note: The x_pos is from the left edge of the screen and the y_pos is from the top of the screen.

Dynamic Properties


pushButton_6 : QPushButton
Property
Value
autoExclusive
autoRepeatDelay
300
autoRepeatInterval
100
QPushButton
autoDefault
default
flat
Dynamic Properties
function
help
file
home_help.txt
topic
Help On Homing
horz_size
500
vert_size
500
x_pos
100
y_pos
100

Help Dialog


This is a test of the help system
Tab on this line.
No tab
Spaces

CHAPTER

TWENTY

HAL PINS

HAL Tutorial

Creating widgets that connect to HAL (Hardware Abstract Layer) is as simple as adding a few Dynamic Properties. See *Dynamic Properties* for step by step instructions to add a Dynamic Property.

Typically the Dynamic Properties are String type with some exceptions being Bool and Color. Connections from Flex HAL objects to other HAL objects must be done in the file assigned to the POSTGUI_HALFILE variable in the [HAL] section typically named *postgui.hal*.

```
[HAL]
HALFILE = main.hal
POSTGUI_HALFILE = postgui.hal
```

The property *pin_name* defines the HAL pin name that is prefixed with *flexhal*. A pin_name of my-button would be *flexhal.my-button* in HAL.

Note: Dynamic Property names are case sensitive and must be all lower case. Hal types and directions are case sensitive and must be all caps. The function value must be lower case.

Note: Hal pin names can containe a-z, A-Z, 0-9, underscore _, or dash -.

20.1 HAL Button

A QPushButton, QCheckBox or QRadioButton can be assigned to a HAL *bit* pin by adding two string type Dynamic Properties. A pin_name of my-button would be *flexhal.my-button* in HAL. HAL pins can be connected in the postgui.hal file.

Table 1: HAL Push Button

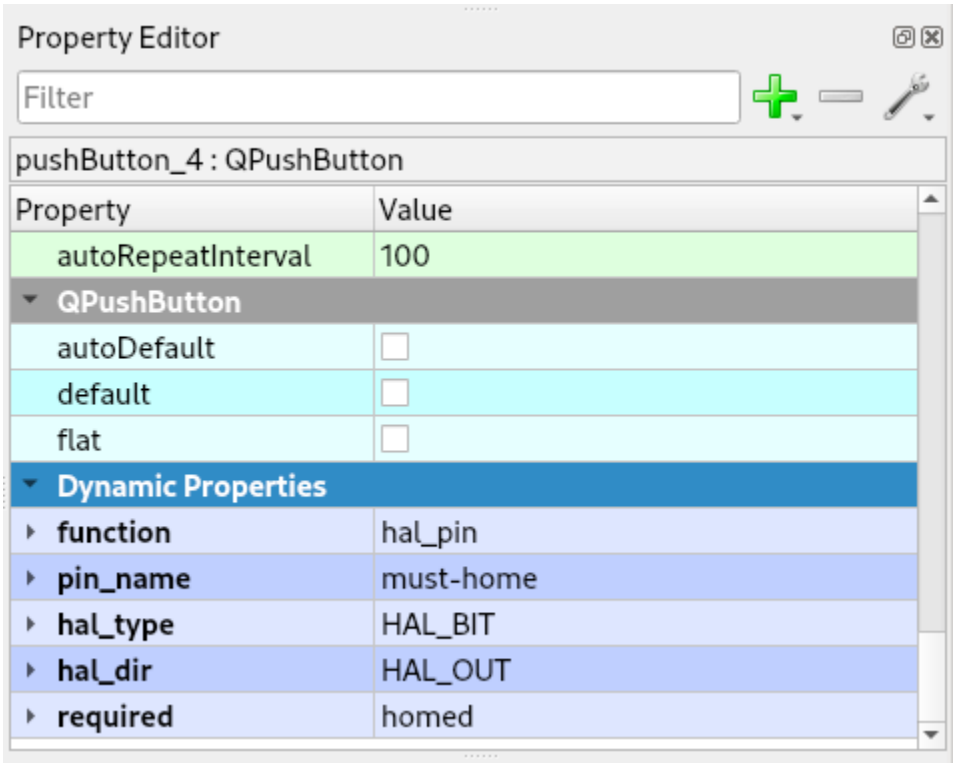
Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name

Optionally the button can be disabled when the power is off by adding a Dynamic Property named *state_off* and setting the value to *disabled*.

Optionally if the HAL button requires all joints to be homed before being enabled, you can specify that by adding a Dynamic Property named *required* and set the value to *homed*.

Table 2: HAL Button Options

Property Type	Property Name	Pin Value
String	state_off	disabled
String	required	homed



Warning: By default, no QRadioButtons are checked unless you set one checked in the Designer. Starting up with none checked could be a problem if you expect one to be selected at startup.

20.2 HAL LED Button

A QPushButton can be a HAL LED button by adding two dynamic properties. The *pin_name* property will be the HAL pin name you use to connect the button state in the postgui.hal file. The button can be momentary or checkable. The default colors are Red when Off and Green when On. The default shape is round.

Table 3: HAL LED Button Dynamic Properties

Property Type	Property Name	Pin Value
String	function	hal_led_button
String	pin_name	any unique name
	Optional	
String	led_shape	square

20.3 HAL Spinbox

A QSpinBox can be a HAL *number* pin by adding three string type Dynamic Properties. The pin_name used will create a HAL pin prefixed with *flexhal.* A pin_name of my-spinbox would be in HAL *flexhal.my-spinbox*. The spinbox is an Out type that will set the value of the HAL pin to match the value of the spinbox.

Table 4: HAL Spin Box

Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name
String	hal_type	HAL_S32 or HAL_U32

20.4 HAL Double Spinbox

A QDoubleSpinBox can be a HAL *number* pin by adding two string type Dynamic Properties. The pin_name used will create a HAL pin prefixed with *flexhal.* A pin_name of my-spinbox would be in HAL *flexhal.my-spinbox*. The spinbox is an Out type that will set the value of the HAL pin to match the value of the spinbox.

Table 5: HAL Spin Box

Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name

20.5 Slider

A QSlider can be a HAL pin by adding these three string type Dynamic Properties. The pin_name used will create a HAL pin prefixed with *flexhal.* A pin_name of my-slider would be in HAL *flexhal.my-slider*. A QSlider supports only integers so to connect it to a float HAL pin use conv_s32_float or conv_u32_float.

See *Dynamic Properties* for step by step instructions to add a Dynamic Property

Table 6: HAL Slider

Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name
String	hal_type	HAL_S32 or HAL_U32

20.6 HAL I/O

A HAL I/O object has an input and output on the same pin. The pin can set an input pin of another HAL object and the pin can be set by another HAL object output pin. The HAL I/O will stay synchronized with the pin it's connected to.

A QPushButton (set to checkable), QCheckBox, QRadioButton, QSpinBox, QDoubleSpinBox or a QSlider can be a HAL I/O object.

Table 7: HAL I/O

Property Type	Property Name	Pin Value
String	function	hal_io
String	pin_name	any unique name
String	hal_type	HAL_S32 or HAL_U32 for a QSpinBox or QSlider

20.7 Label

A QLabel can be used to monitor HAL pins. HAL connections must be made in the post gui HAL file. The pin_name used will create a HAL pin prefixed with *flexhal.* A pin_name of my-reader would be in HAL *flexhal.my-reader*.

Table 8: HAL Label

Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name
String	hal_type	HAL_BIT or HAL_FLOAT or HAL_S32 or HAL_U32

Note: A HAL_FLOAT QLabel can have a string Dynamic Property called *precision* with a value of the number of decimal digits.

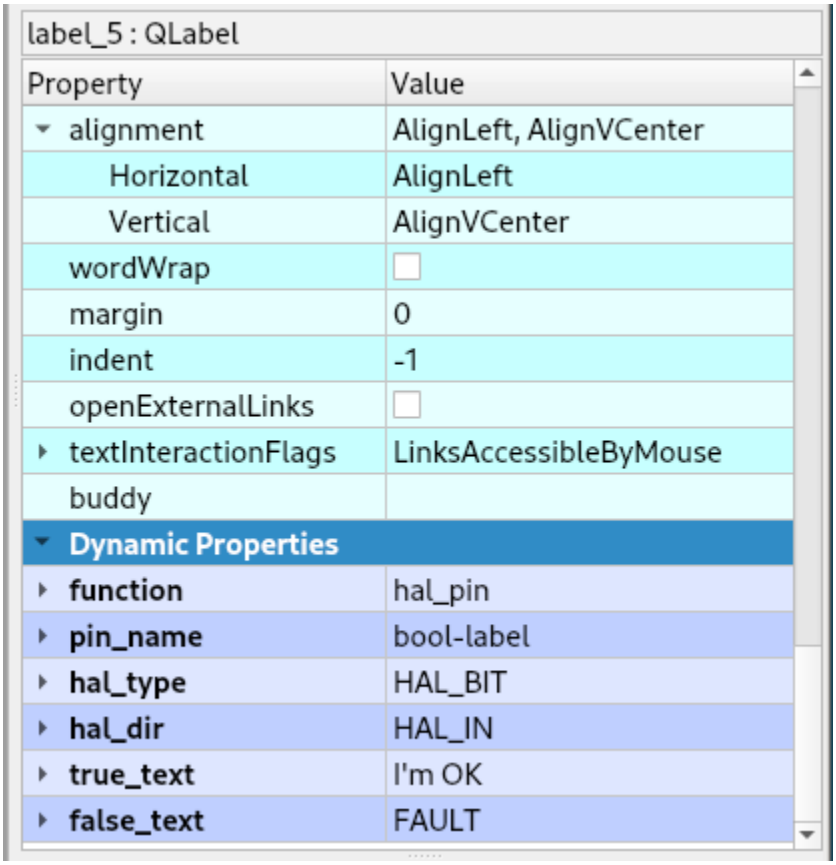
20.8 Bool Label

A QLabel of hal_type HAL_BIT can have True and False text by adding two additional Dynamic Properties.

See *Dynamic Properties* for step by step instructions to add a Dynamic Property

Table 9: HAL Bool Label

Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name
String	true_text	text to display when True
String	false_text	text to display when False



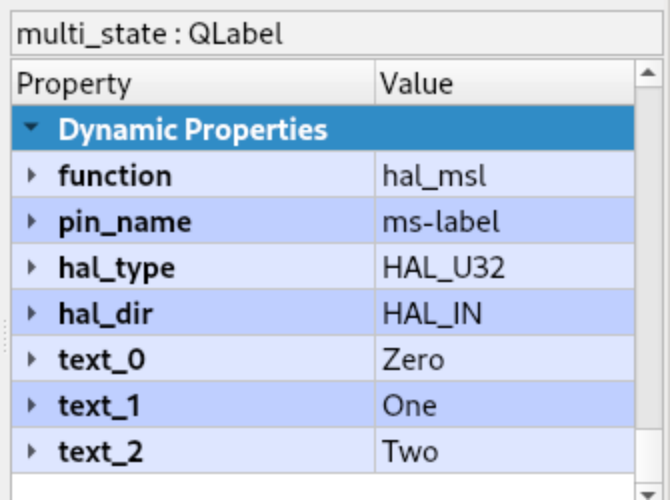
20.9 Multi-State Label

A QLabel of hal_type HAL_U32 can have multiple text by adding as many Dynamic Properties as needed. The *text_n* starts at 0 for example text_0, text_1 etc.

Table 10: HAL Multi-State Label

Property Type	Property Name	Pin Value
String	function	hal_msl
String	pin_name	any unique name
String	text_n	text to display when value is equal to n

Note: The text values must start at 0 and be sequencial.



20.10 HAL LED

A QLabel can be used as a HAL LED indicator by adding the following properties to a blank label. The default colors are Red when Off and Green when On. The pin_name is the hal name the LED will have. The default shape is round.

The HAL LED needs to be connected in the postgui.hal file and can only be connected to a HAL pin of type bit with a HAL direction of OUT or a signal that is connected to a HAL pin of type bit with a HAL direction of OUT. Only one OUT direction can be connected to a signal while multiple IN directions can be connected to a signal.

Table 11: HAL LED

Property Type	**Property Name**	**Pin Value**
Bool	hal_led	True
String	function	hal_led
String	pin_name	any unique name
	Optional	
Color	on_color	color of your choice
Color	off_color	color of your choice
Int	edge_margin	space between circle and edge of the label
String	led_shape	square

Choosing 'rectangular' for LED shape will fill the entire control as one giant indicator.

Note: Select Other to get the list and select Color. You can copy and paste the hex color value into the color picker.

20.11 HAL LED Label

Similar to the HAL LED except the LED is in the upper right corner so the label can have text. The default colors are Red when Off and Green when On. The default shape is round.

Table 12: HAL LED Label

Property Type	Property Name	Pin Value
Bool	hal_led_label	True
String	function	hal_led
String	pin_name	any unique name
	Optional	
Color	led_on_color	color of your choice
Color	led_off_color	color of your choice
Int	led_diameter	diameter of led
Int	led_right_offset	offset from right edge
Int	led_top_offset	offset from top edge
String	led_shape	square

20.12 LCD

A QLCDNumber can be used to monitor HAL pins. HAL connections must be made in the post gui HAL file. The pin_name used will create a HAL pin prefixed with *flexhal.* A pin_name of my-reader would be in HAL *flexhal.my-reader*.

Table 13: HAL LCD

Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name
String	hal_type	HAL_FLOAT or HAL_S32 or HAL_U32

Note: A HAL_FLOAT QLCDNumber can have a string Dynamic Property called *precision* with a value of the number of decimal digits.

20.13 Progress Bar

A QProgressBar can be used to monitor HAL pins. HAL connections must be made in the post gui HAL file. The pin_name used will create a HAL pin prefixed with *flexhal.* A pin_name of my-bar would be in HAL *flexhal.my-bar*.

Table 14: HAL Progressbar

Property Type	Property Name	Pin Value
String	function	hal_pin
String	pin_name	any unique name
String	hal_type	HAL_S32 or HAL_U32

20.14 Step by Step

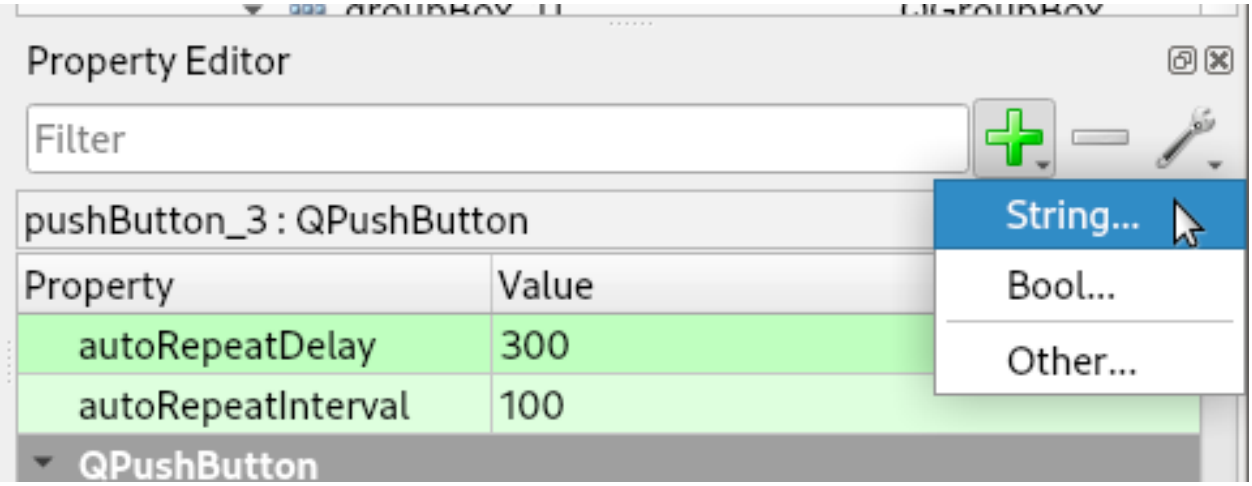
Note: This example is for a QPushButton

You can use a QPushButton as a momentary output, or with *checkable* selected for a toggle type output, or QCheckBox or QRadioButton for a HAL output control.

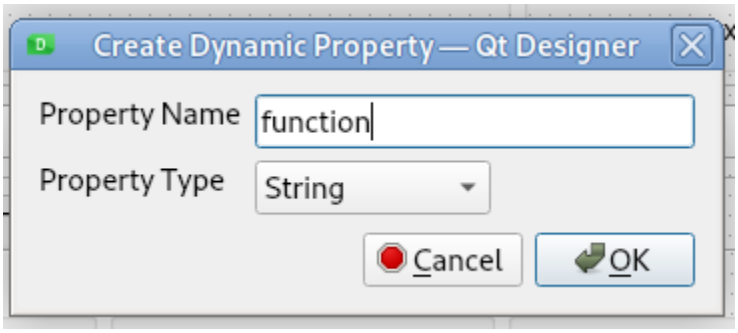
Drag the widget into the GUI and the widget can have any name you like; names are not used by HAL controls in Flex GUI - it is the following that matters.

Click on the widget to select it then click on the green plus sign in the Property Editor for that widget to add a Dynamic Property and select String.

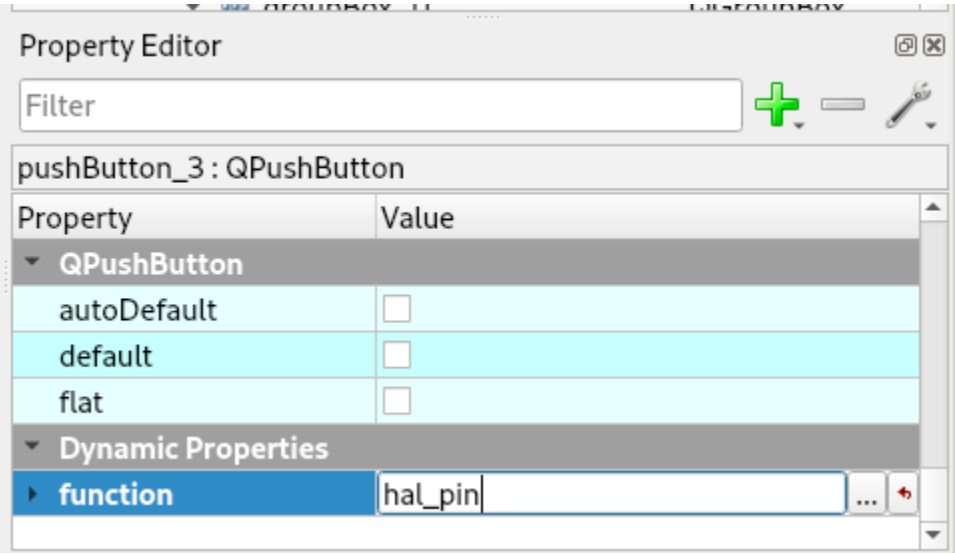
See *Dynamic Properties* for step by step instructions to add a Dynamic Property



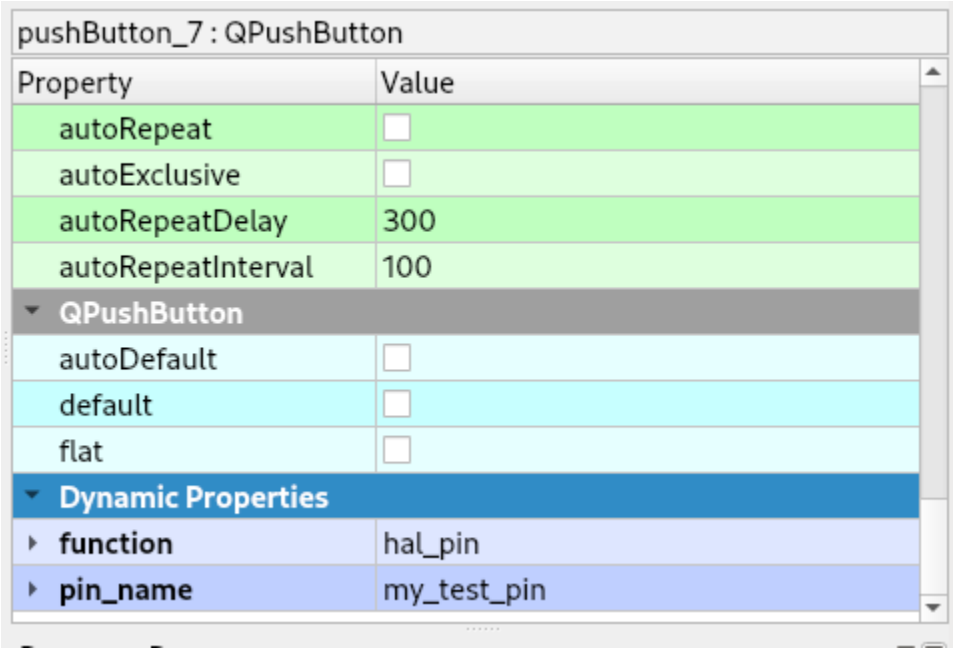
Set the Property Name to *function* and click Ok



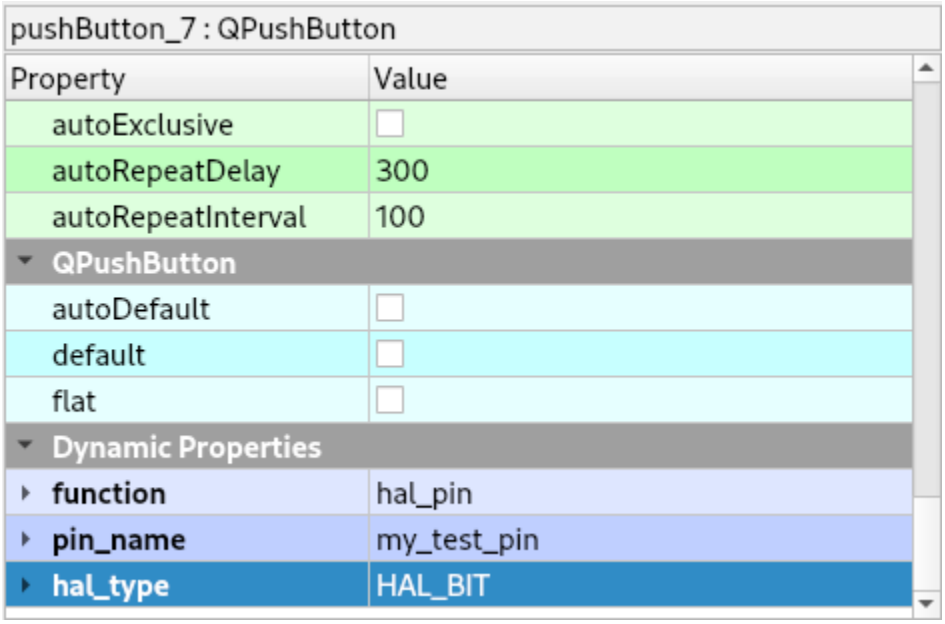
Set the Value to *hal_pin*; this tells Flex GUI that this widget is going to be for a HAL pin



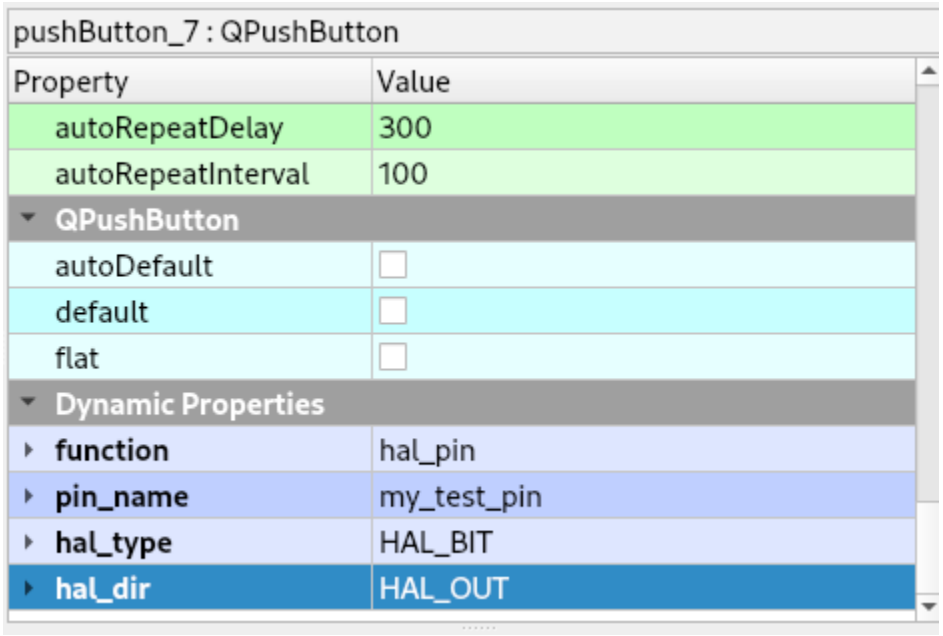
Add another string Dynamic Property named *pin_name* and set the value to any unique name



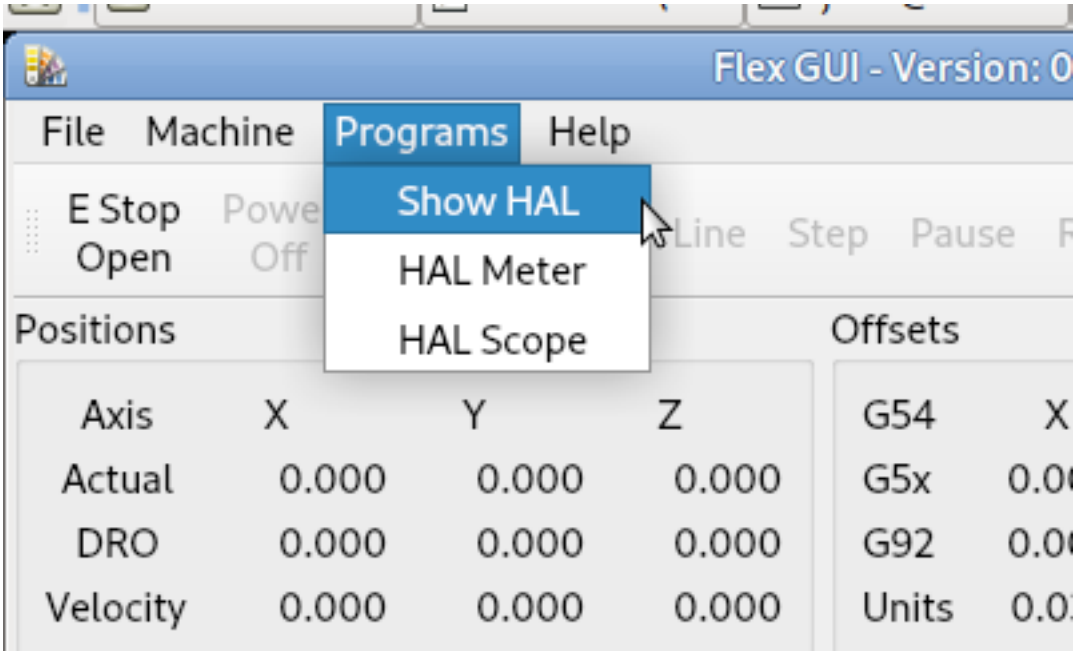
Add another Dynamic Property named *hal_type* and set the value to HAL_BIT



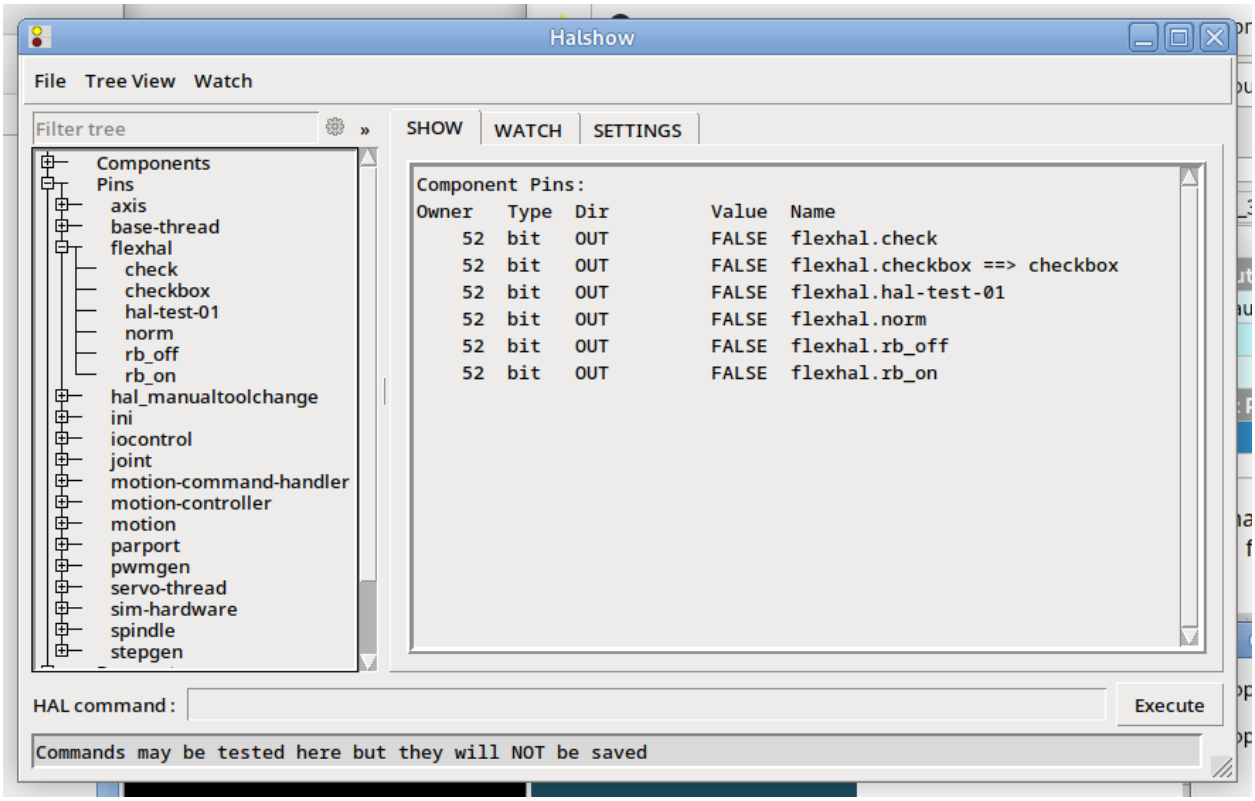
Add another Dynamic Property named *hal_dir* and set the value to HAL_OUT



If you added Show HAL to your menu, you can open up the *Halshow* program and view the pin names



The pin names will all start with *flexhal* plus the unique name you gave them



Now you can connect the Flex HAL pin in the postgui.hal file like normal

```
net some-signal-name flexhal.hal-test-01 => some-other-pin-in
```

After installing Flex GUI, from the CNC menu, you can copy the Flex GUI examples and look at the hal-btn example.

HAL Pin Types:

```
HAL_BIT
HAL_FLOAT
HAL_S32
HAL_U32
```

HAL Pin Directions:

```
HAL_IN
HAL_OUT
HAL_IO
```

CHAPTER

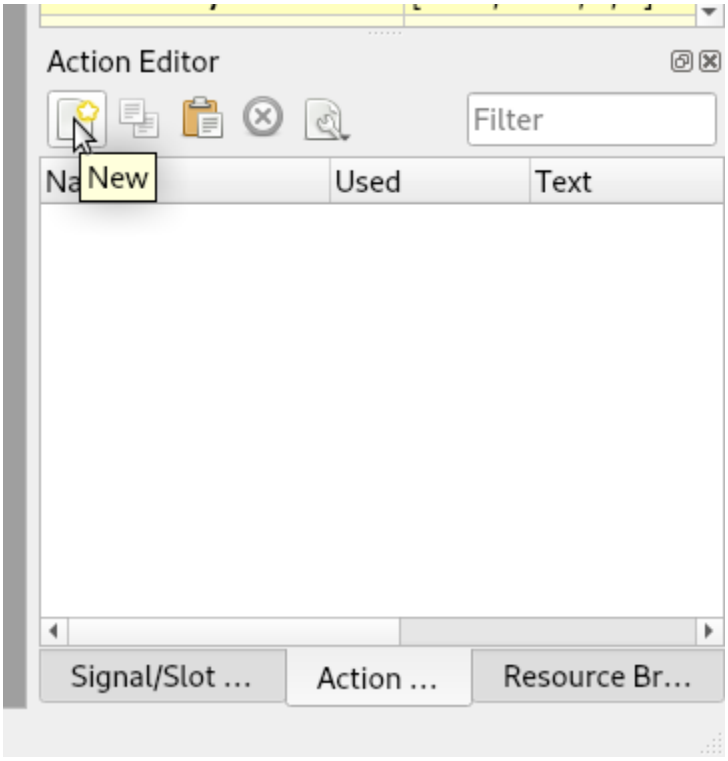
TWENTYONE

TOUCH SCREENS

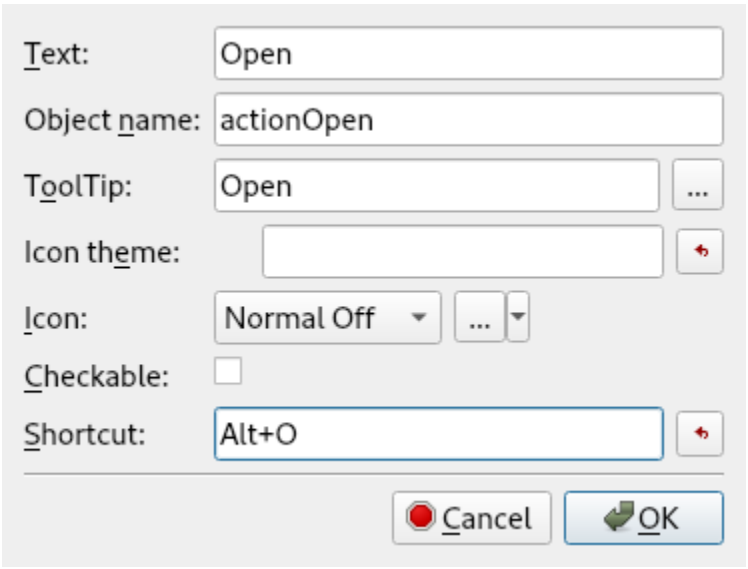
Some entry widgets like MDI and Touch-Off have a touch-screen popup available to make it easier for those users to enter the data.

21.1 Tool Bar

To add a button to the tool bar without having a menu item that creates an action, you just have to create the action yourself

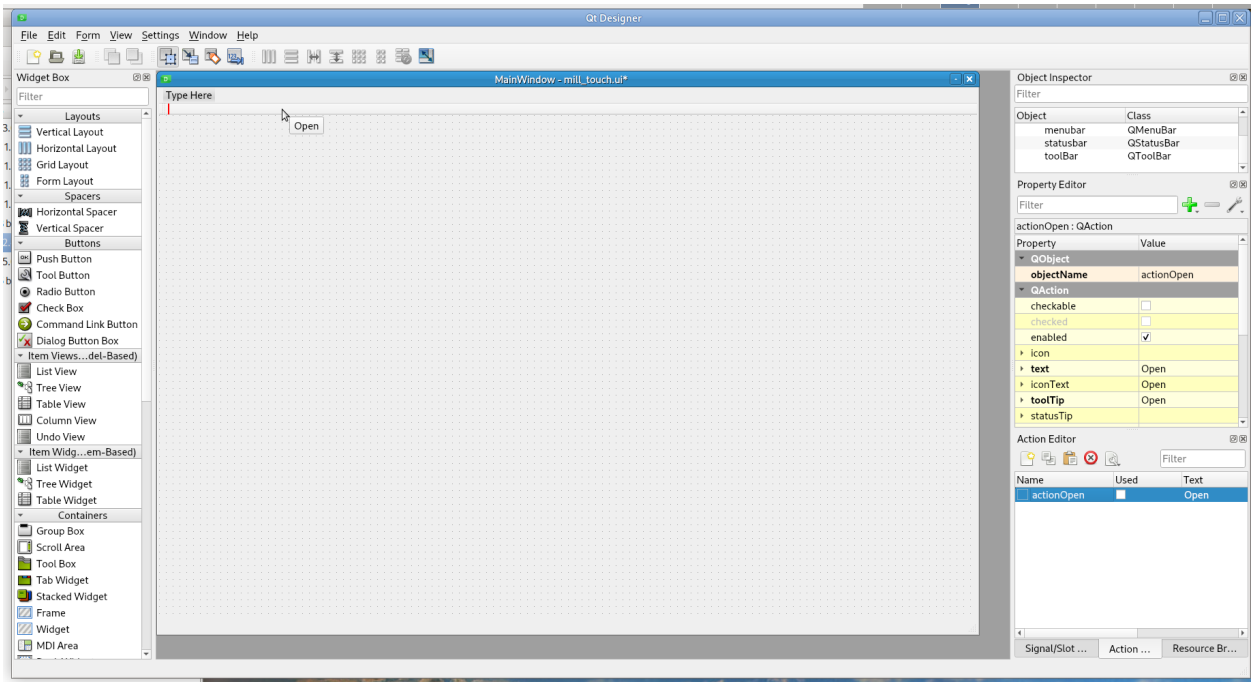


The action creating window, when you type in the Text name, the Object name is created for you



Warning: Make sure the Object name matches the Action Name created when you create a menu item - see the *Menu* section for the full list of Action Names.

Now you just drag the action into the tool bar to create a new tool bar button



Another option is to just use QPushButtons in a QFrame, as every menu action has a QPushButton as well that executes the same function.

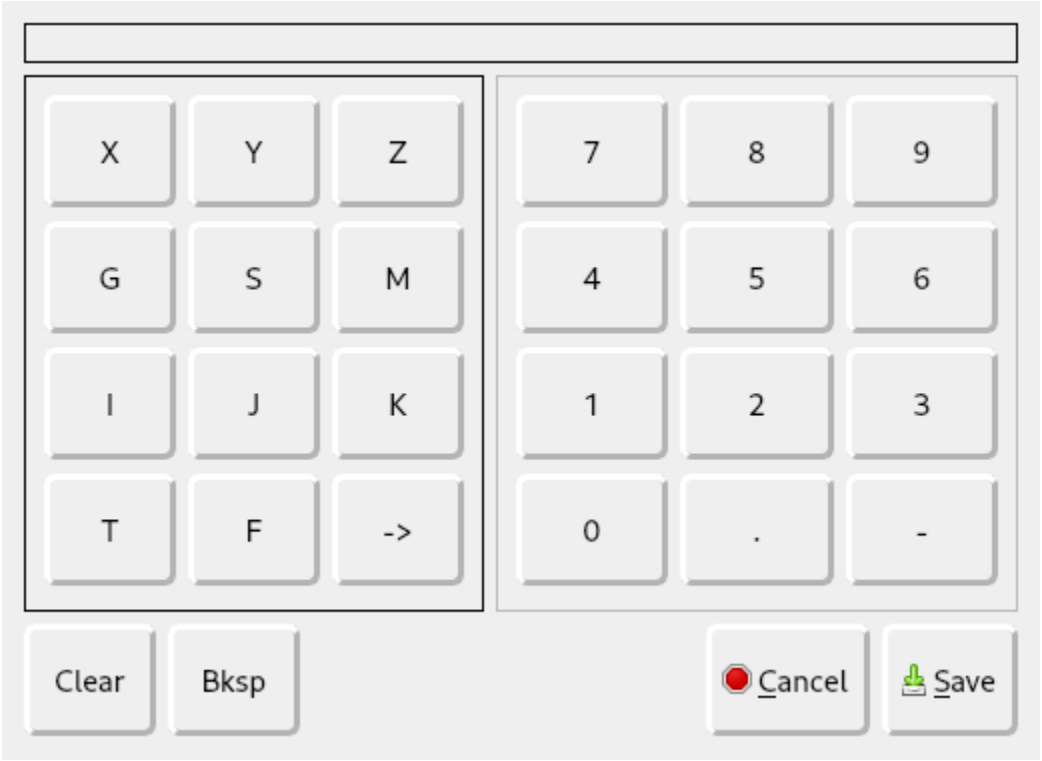
21.2 Popup Keypads

If you drag a popup keypad to a different location, next time you use the keypad for that object the popup keypad will remember the last location and move to that location.

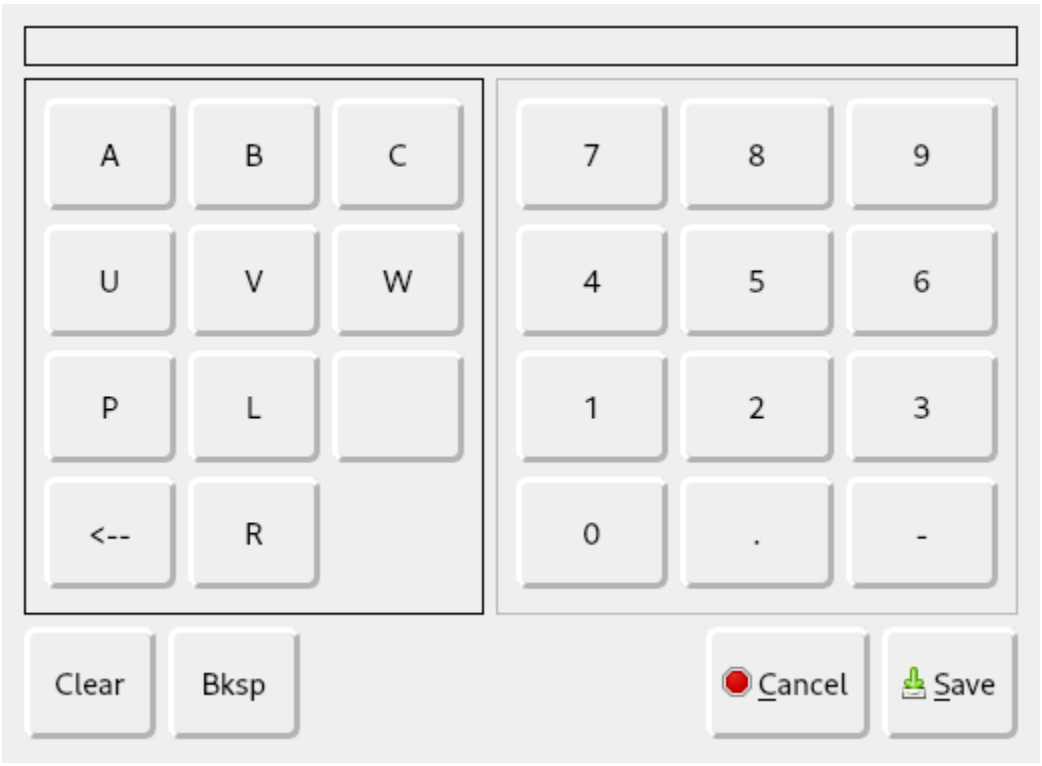
21.3 MDI

To enable the popup entry dialogs for the MDI entry, the QLineEdit object name must be *mdi_command_le* and the Dynamic Property *input* must be *nccode* for the NC codes popup or *keyboard* for a full keyboard popup.

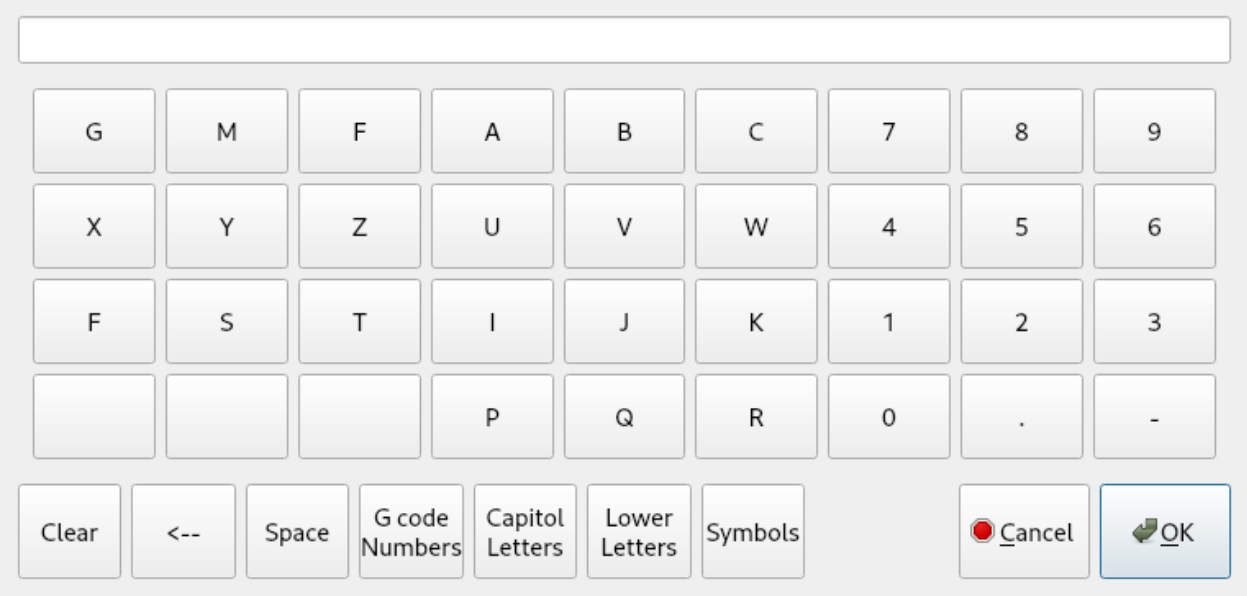
The G codes dialog will appear when you touch the MDI entry box



The arrow buttons change the letters section to different letters



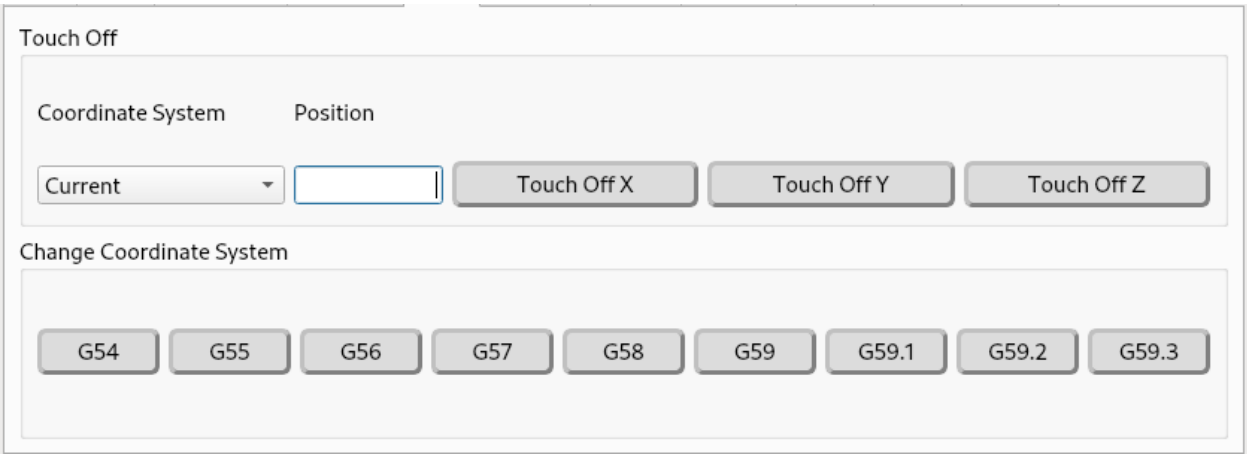
The full keyboard



21.4 Touch Off

The Coordinate System Touch-Off offset is a QLineEdit named *touchoff_le*. To enable the number pad popup for the offset entry, add a Dynamic Property named *input* and set the value to *number*

Touch-Off:



21.5 Tool Touch-Off

The Tool Touch-Off offset is a QLineEdit named *tool_touchoff_le*. To enable the number pad popup for the offset entry, add a Dynamic Property named *input* and set the value to *number*.

21.6 Spin Boxes

QDoubleSpinBox and QSpinBox can use the popup numbers keypad by adding a Dynamic Property named *input* and setting the value to *number*. If you enter a float value for a QSpinBox the value will get converted to an integer

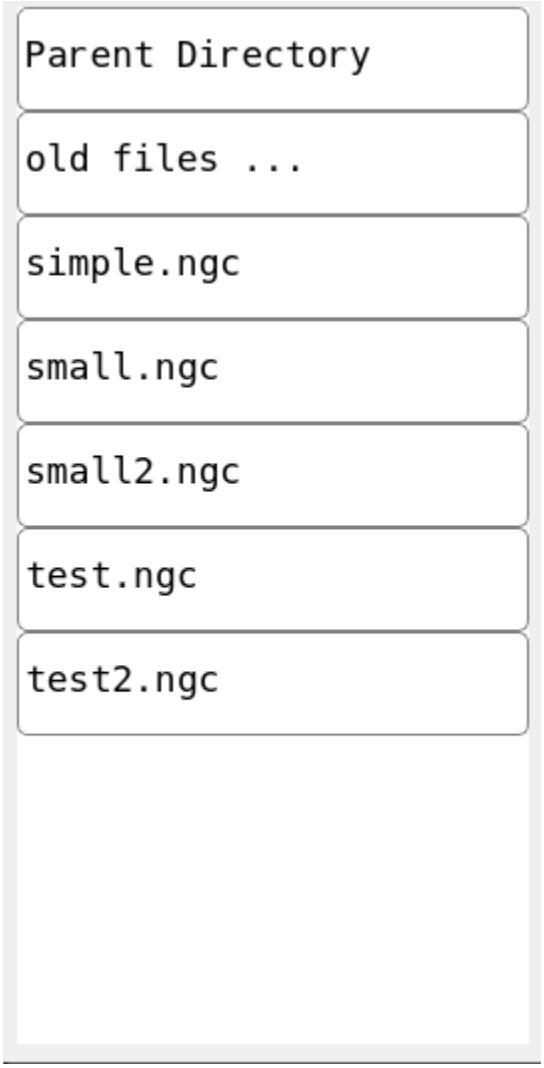
21.7 Line Edits

A QLineEdit can have a popup entry for numbers, G codes, or a full keyboard. Add a Dynamic Property named *input* and set the value to one of these *number*, *nccode*, or *keyboard*.

21.8 File Navigator

If a QListWidget with an objectName of *file_lw* is found, a touch-friendly file selector is added. A Parent Directory and possibly a directory name with an ellipsis can be used to change directories. Touch a file name and it is loaded into the GUI.

If PROGRAM_PREFIX is specified, that will be the starting directory:

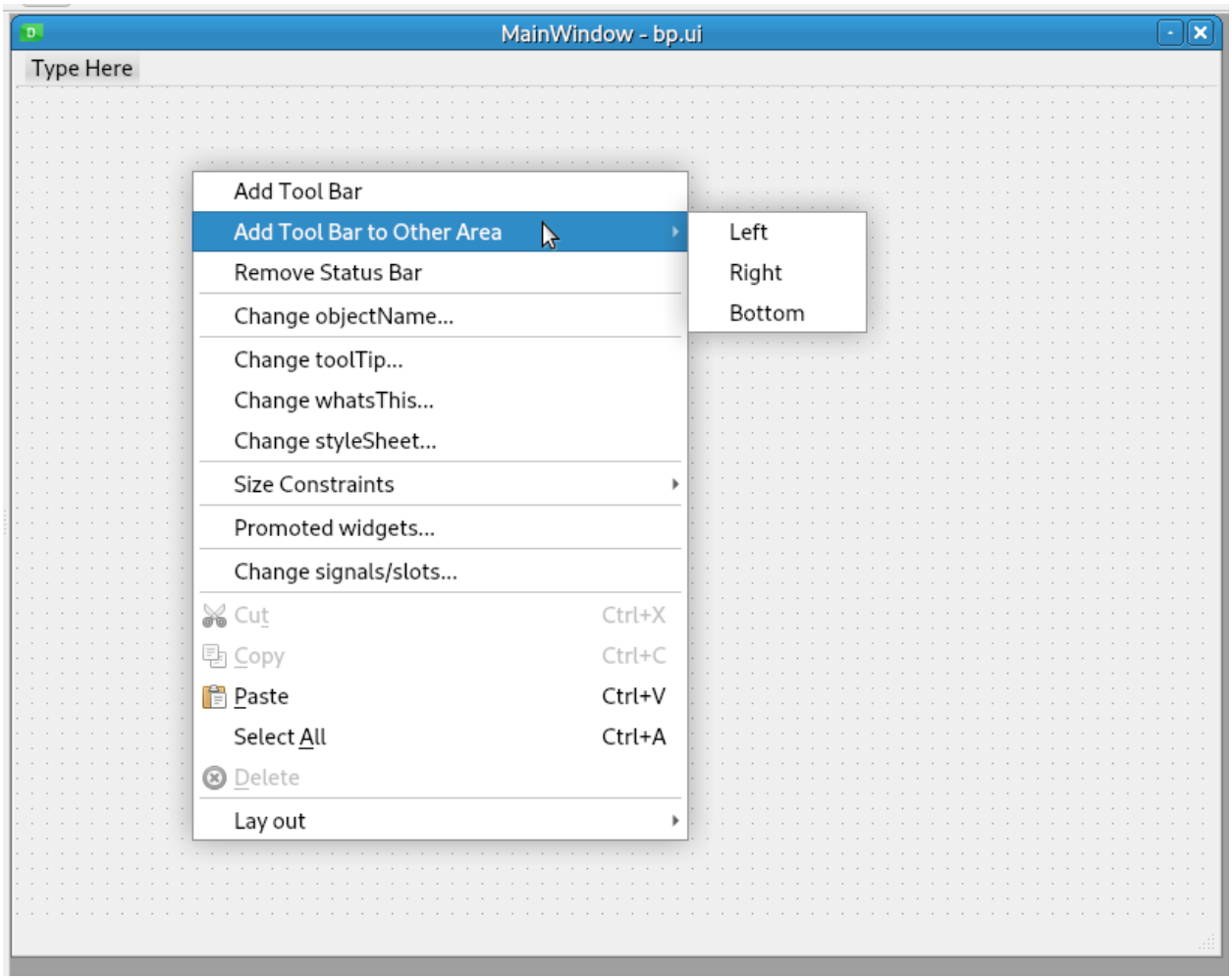

Parent Directory
old files ...
simple.ngc
small.ngc
small2.ngc
test.ngc
test2.ngc

CHAPTER

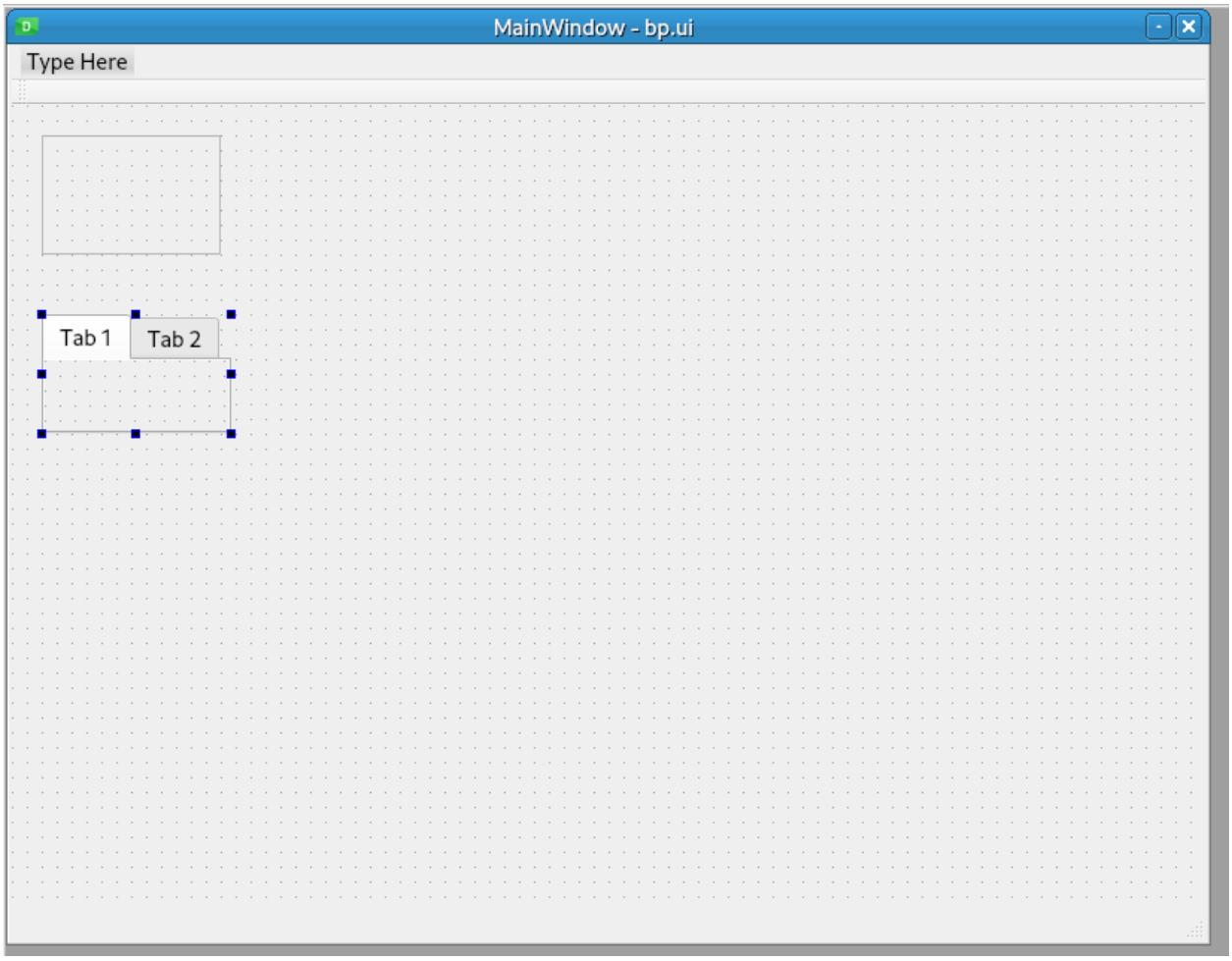
TWENTYTWO

MASTER LAYOUT

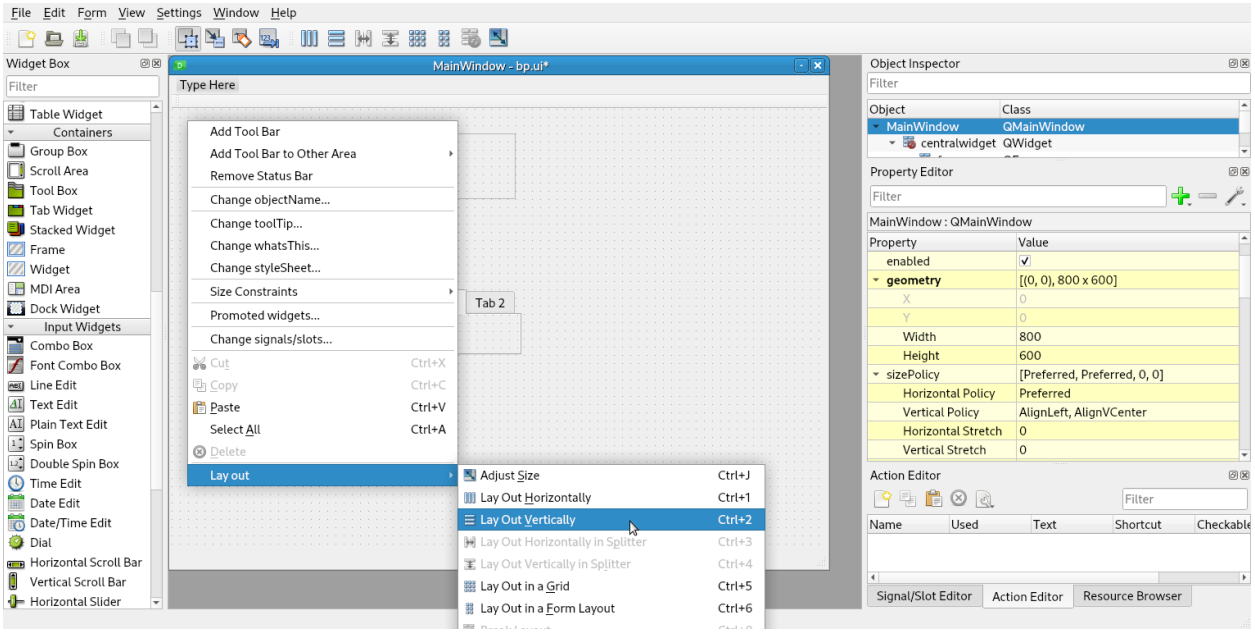
Starting with an empty Main Window, if you right-click in it you can add a tool bar or remove the status bar



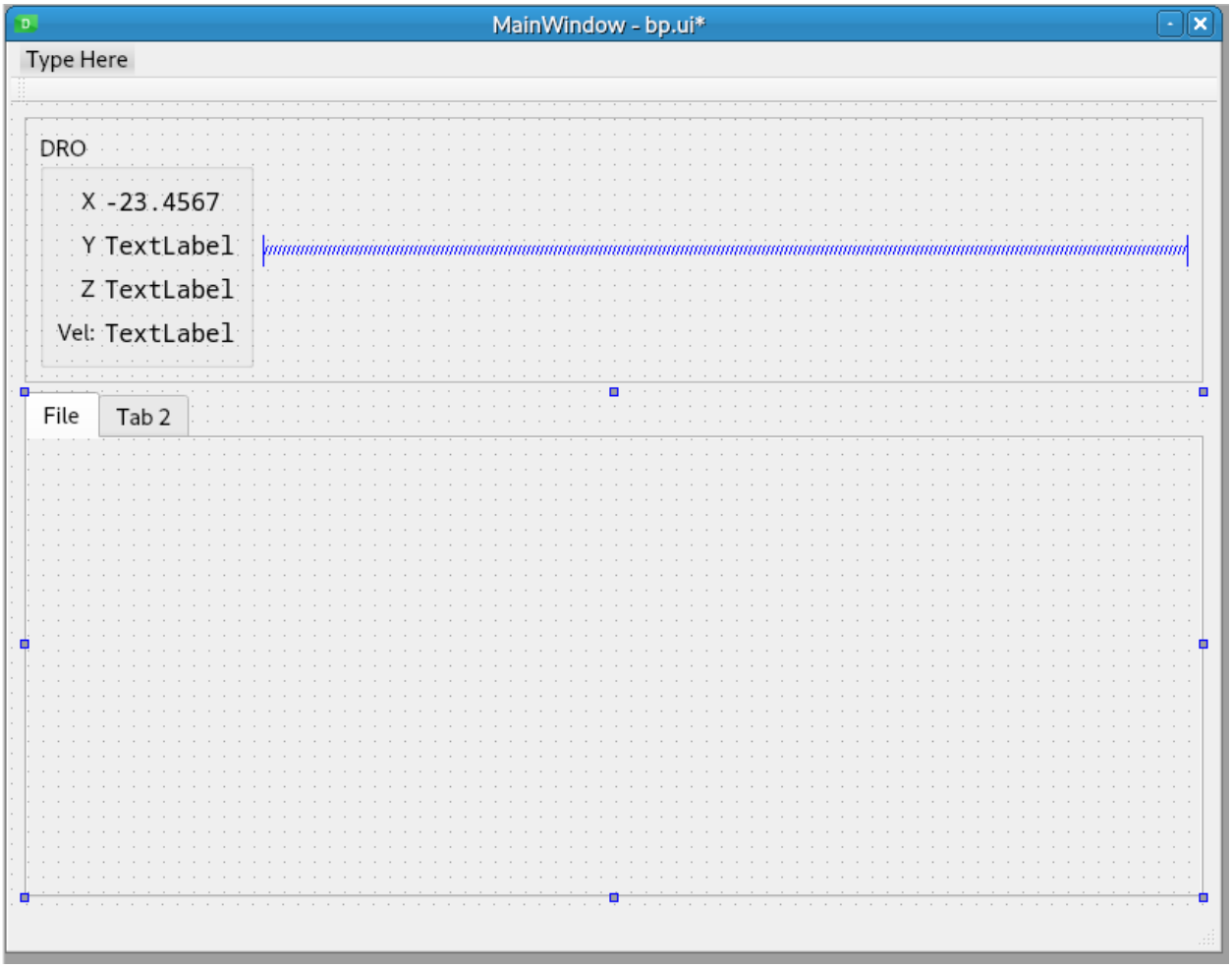
Next, if you have some items you want visible all the time you can add a QFrame or QWidget then below that, add a QTabWidget



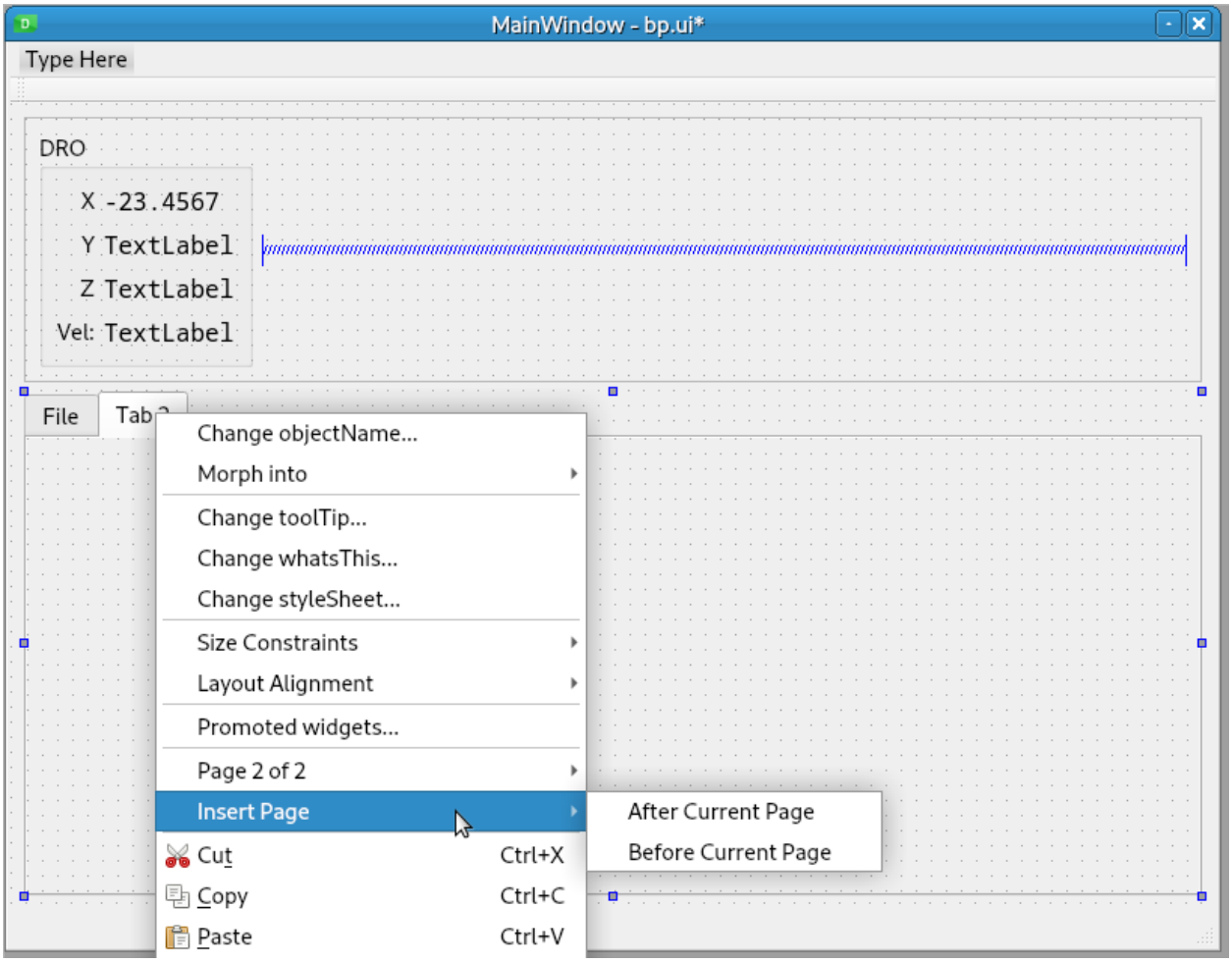
Now that you have at least one widget in the main window, you can right-click and select the layout you want to use



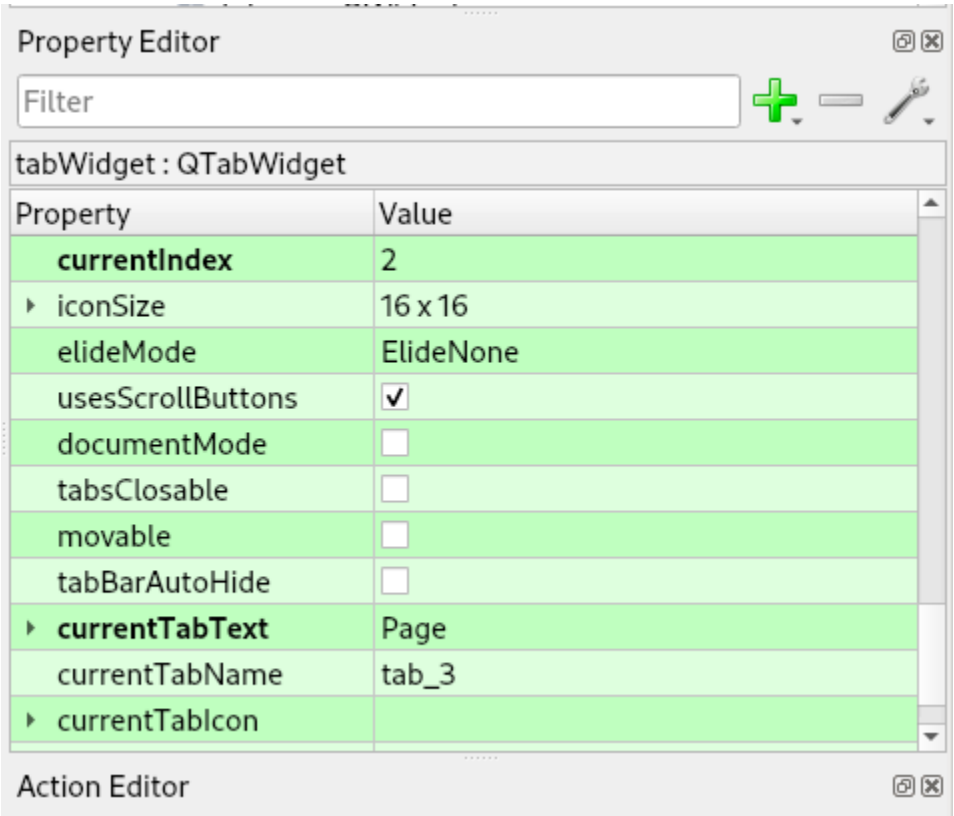
Example layout:



To add more tabs to a tab widget, right click on the tab then select Insert Page and where you want it to be inserted



To change the tab name, in the Property Editor QTabWidget section, change the currenTabText value to the new desired name.


Property Editor
Filter
tabWidget : QTabWidget
Property
Value
currentIndex
2
iconSize
16 x 16
elideMode
ElideNone
usesScrollButtons
documentMode
tabsClosable
movable
tabBarAutoHide
currentTabText
Page
currentTabName
tab_3
currentTabIcon
Action Editor

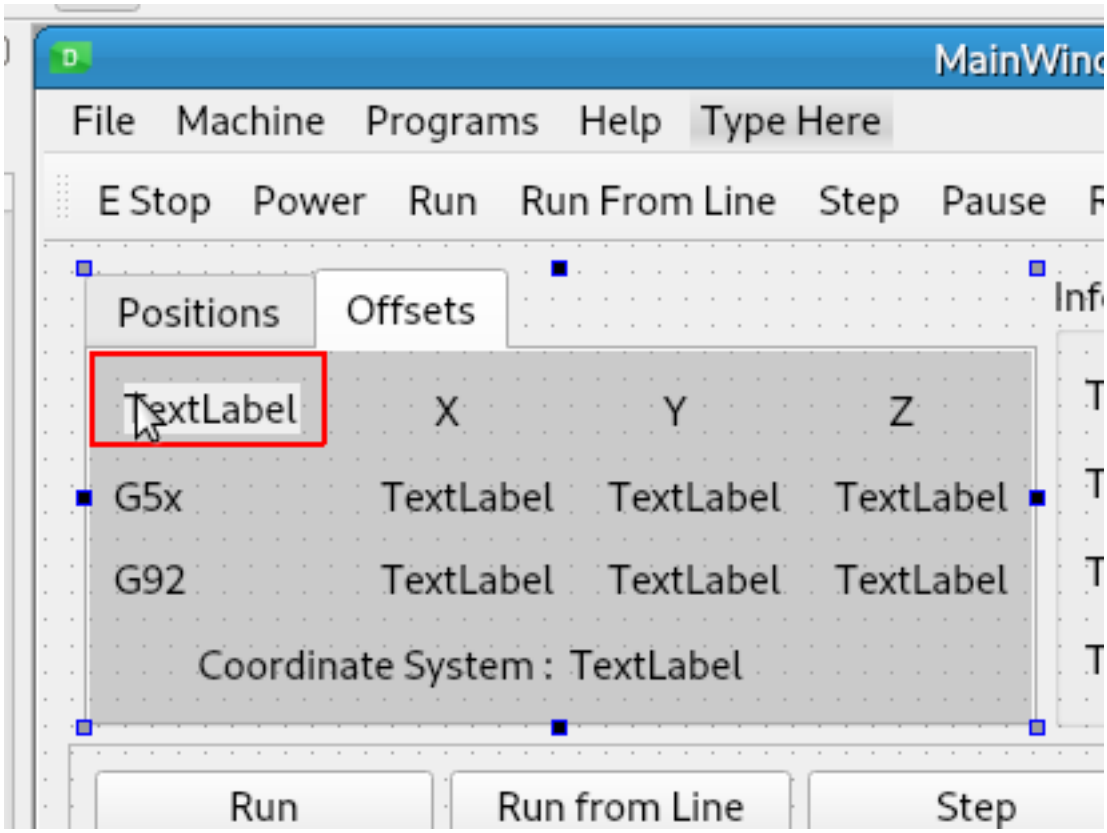
CHAPTER

TWENTYTHREE

GUI TIPS

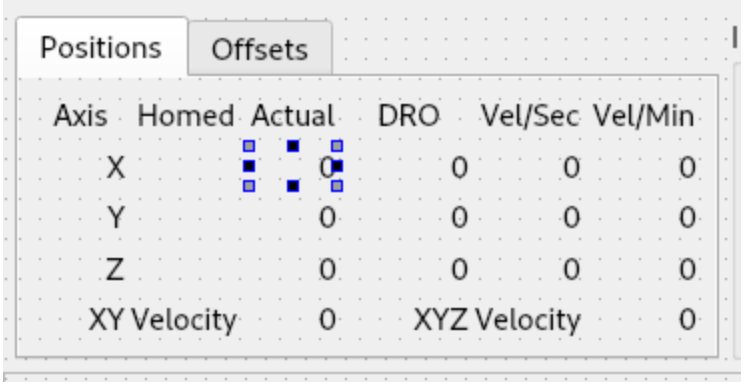
To group items together, use a container like a QFrame, QGroupBox or a QTabWidget.

If the contents of a container are rows and columns after adding at least one widget, right click and select *Layout* then *Lay Out in a Grid*. Now you can drag and drop widgets into the container. The blue line or red box indicate where it will be placed in the grid.

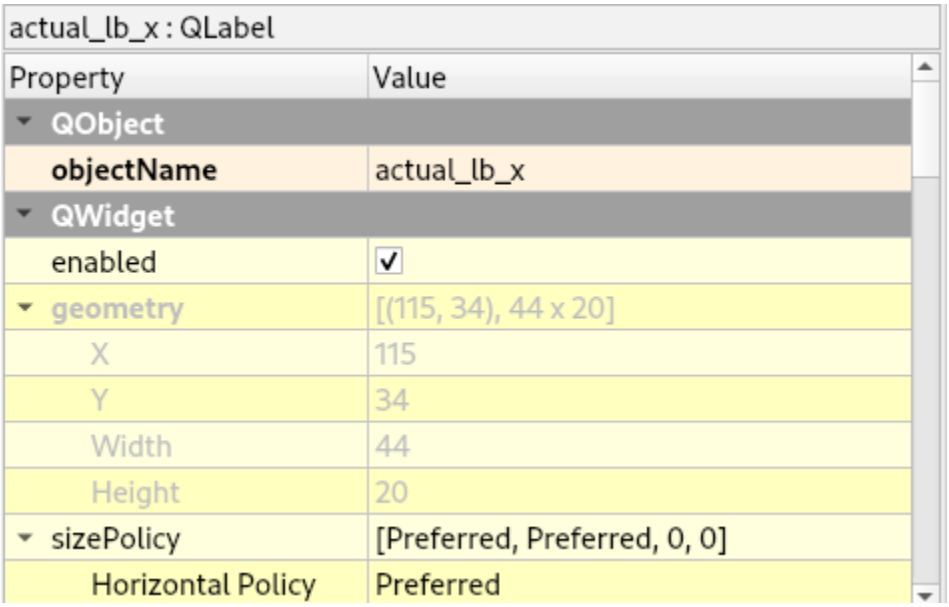


When using a grid layout for items that may change like Dro labels, change the text to represent the longest number including a minus sign. Next, in the property editor look at the Width and select (I usually use the column title) a widget and set the minimum width a tad bigger than the widest widget in that column. This will prevent the column from resizing as the values change.

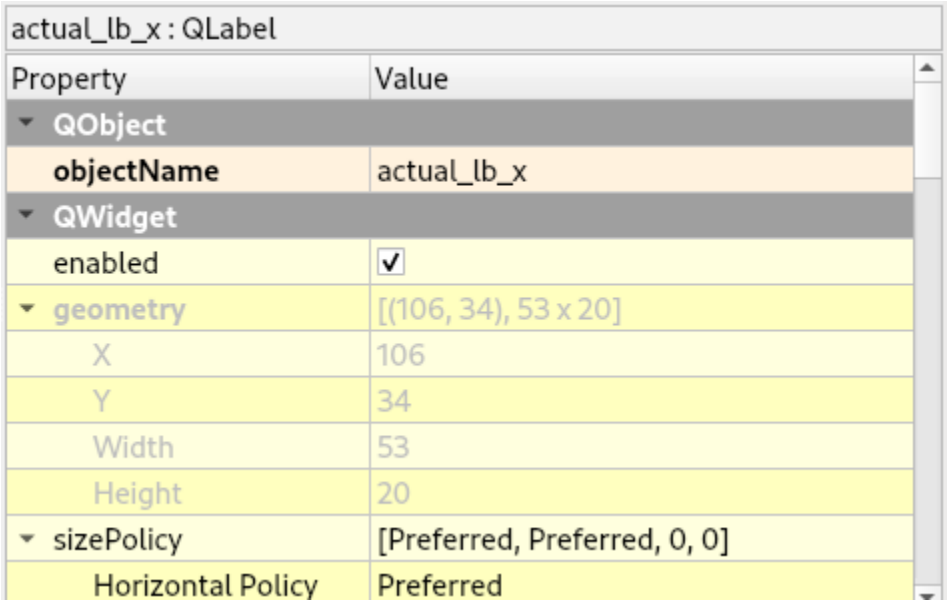
For example, the numbers in the Actual column can contain up to 8 characters like -23.4567. In the next image no minimum width has been set



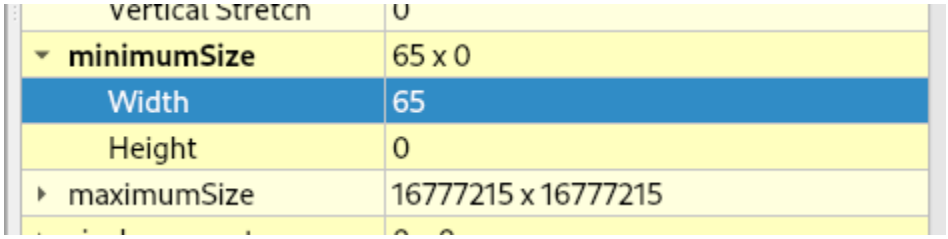
All the cells in the column will have the same width - here you can see it has a width of 44



If we double-click in the label and add -23.4567 the width changes to 61



I usually set the title of a column width to be a bit wider than the widest widget in the column



If you drag a container into another container that has a layout and it's real short, just set the minimum height to make it larger and easier to drag and drop into.

Ctrl + left click to select several widgets at once to change all their properties.

The Monospace font is good for numbers that need a fixed width like DRO values.

CHAPTER

TWENTYFOUR

STYLESHEET

You can use your own .qss style sheet by creating a valid .qss file in the configuration directory and setting it in the *INI Settings*.

```
[DISPLAY]
QSS = name_of_file.qss
```

Note: If a THEME is found in the ini file the QSS entry is ignored

The Qt Style Sheets Reference and the Style Sheet Syntax and the Style Sheet Examples are good references to use when creating your own stylesheets.

Note: If there is an error in the stylesheet syntax, no warning is issued, it is just ignored. So don't forget the ; at the end of each setting. And do not accidentally use any backslashes it will break the whole file.

Warning: If you only set a background-color on a QPushButton, the background may not appear unless you set the border property to some value, even if border is set to none.

24.1 Colors

Most colors can be specified using Hex, RGB or RGBA color model. RGB is Red, Green, Blue and A means Alpha or transparency. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all). Hex is red, green blue in hexadecimal number pairs from 00 to ff.

```
#0000ff
rgb(0, 0, 255) Blue
rgba(0, 0, 255, 25%) Light Blue
```

24.2 Examples

```
/* Set the background color for all QPushButtons, border is required * /
QPushButton {
        background-color: rgba(224, 224, 224, 50%);
        border: 1px;
}

/* Set the background color and style for all QPushButtons when Pressed * /
QPushButton:pressed {
        background-color: rgba(192, 192, 192, 100%);
        border-style: inset;
}

/* Set settings for a QPushButton named exit_pb * /
QPushButton#exit_pb {
        border: none;
        background-color: rgba(0, 0, 0, 0);
}

/* Using sub controls * /
QAbstractSpinBox::up-button {
        min-width: 30px;
}

/* Combining sub controls and state * /
QTabBar::tab:selected {
        background: lightgray;
}

/* Target by Object Name starts with something common*/
QLabel[objectName*="dro"] {
        font-family: Courier;
        font-size: 14pt;
        font-weight: 700;
}
```

24.3 Tool Bar Buttons

A tool bar button created from a menu action can be styled by using the QToolButton` selector:

```
QToolButton:hover {
        background-color: rgba(255, 0, 0, 75%);
}
```

To set the style of a single tool bar button, you need to use the widget name for that action. The tool bar button must exist in the tool bar.

Table 1: Tool Button Names

Menu Item	Action Name	Widget Name
Open	actionOpen	flex_Open
Edit	actionEdit	flex_Edit
Reload	actionReload	flex_Reload
Save As	actionSave_As	flex_Save_As
Edit Tool Table	actionEdit_Tool_Table	flex_Edit_Tool_Table
Reload Tool Table	actionReload_Tool_Table	flex_Reload_Tool_Table
Ladder Editor	actionLadder_Editor	flex_Ladder_Editor
Quit	actionQuit	flex_Quit
E Stop	actionE_Stop	flex_E_Stop
Power	action_Power	flex_Power
Run	actionRun	flex_Run
Run From Line	actionRun_From_Line	flex_Run_From_Line
Step	actionStep	flex_Step
Pause	actionPause	flex_Pause
Resume	actionResume	flex_Resume
Stop	actionStop	flex_Stop
Clear MDI History	actionClear_MDI_History	flex_Clear_MDI_History
Copy MDI History	actionCopy_MDI_History	flex_Copy_MDI_History
Show HAL	actionShow_HAL	flex_Show_HAL
HAL Meter	actionHAL_Meter	flex_HAL_Meter
HAL Scope	actionHAL_Scope	flex_HAL_Scope
About	actionAbout	flex_About
Quick Reference	actionQuick_Reference	flex_Quick_Reference

The syntax to select a tool bar button by name (here the flex_Quit button) is:

```
QToolButton#flex_Quit:hover {
        background-color: rgba(255, 0, 0, 75%);
}
```

CHAPTER

TWENTYFIVE

RESOURCES

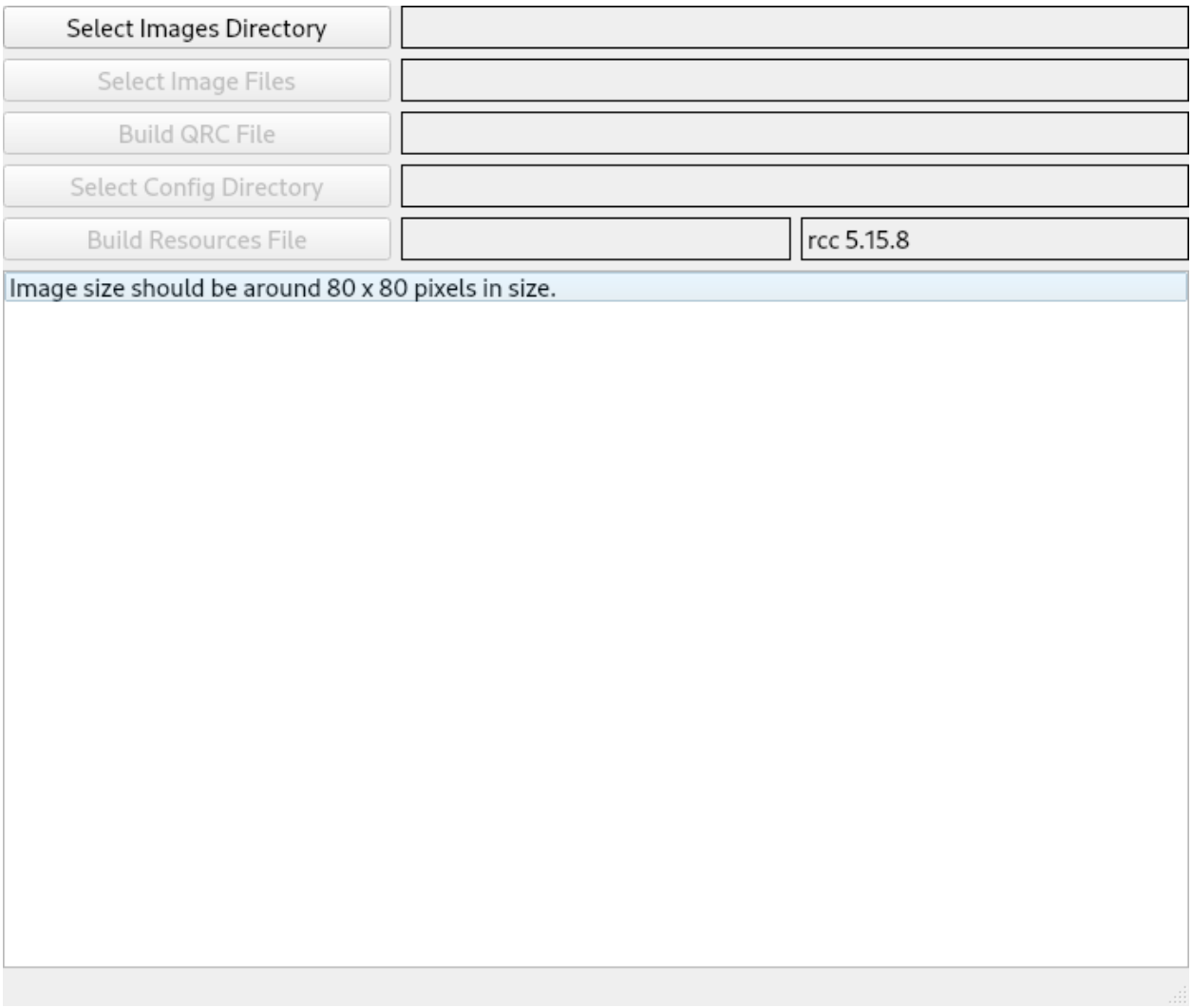
To create a resources.py file with images to use with the .qss stylesheet, start by placing all the images in a different directory than the configuration files. A subdirectory in the configuration directory is fine

```
└── configs
    └── my_mill
        └── images
```

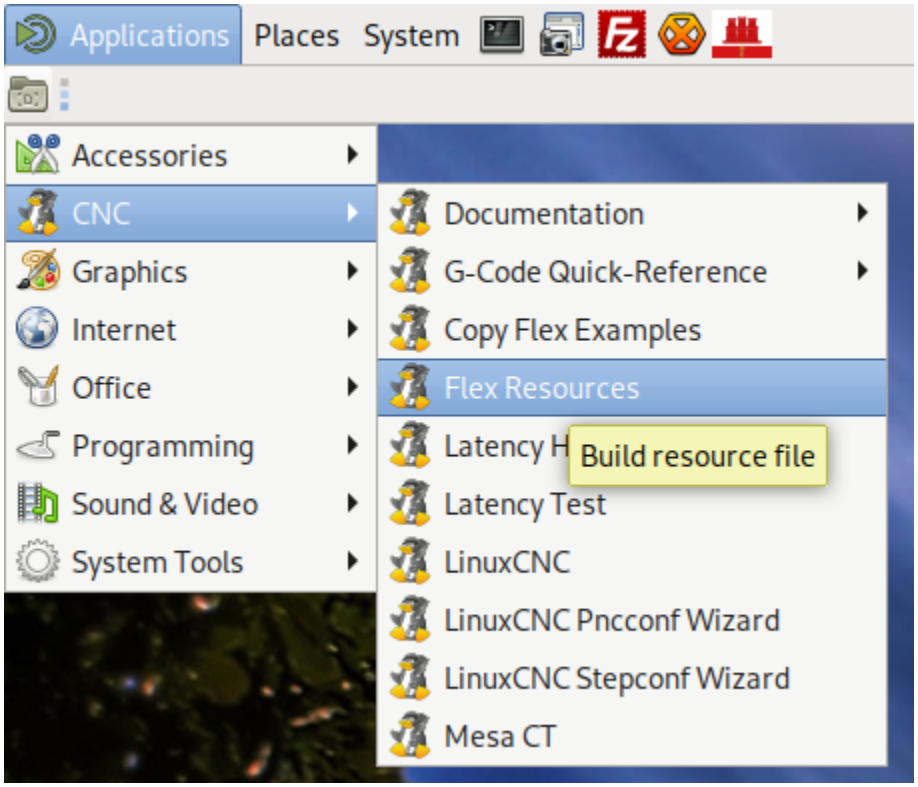
Add the following library if not installed

```
sudo apt install qtbase5-dev-tools
```

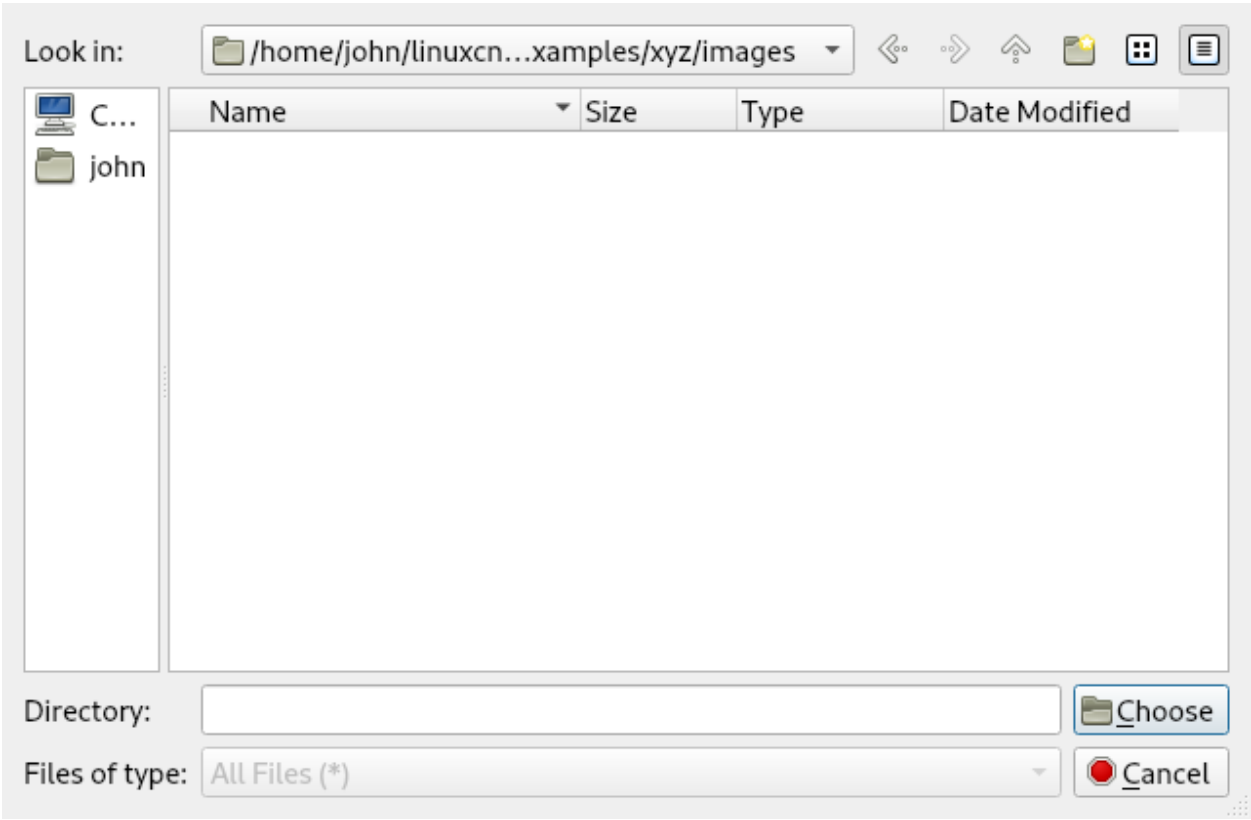
After installing Flex GUI on the CNC menu run *Flex Resources*


Select Images Directory
Select Image Files
Build QRC File
Select Config Directory
Build Resources File
rcc 5.15.8
Image size should be around 80 x 80 pixels in size.

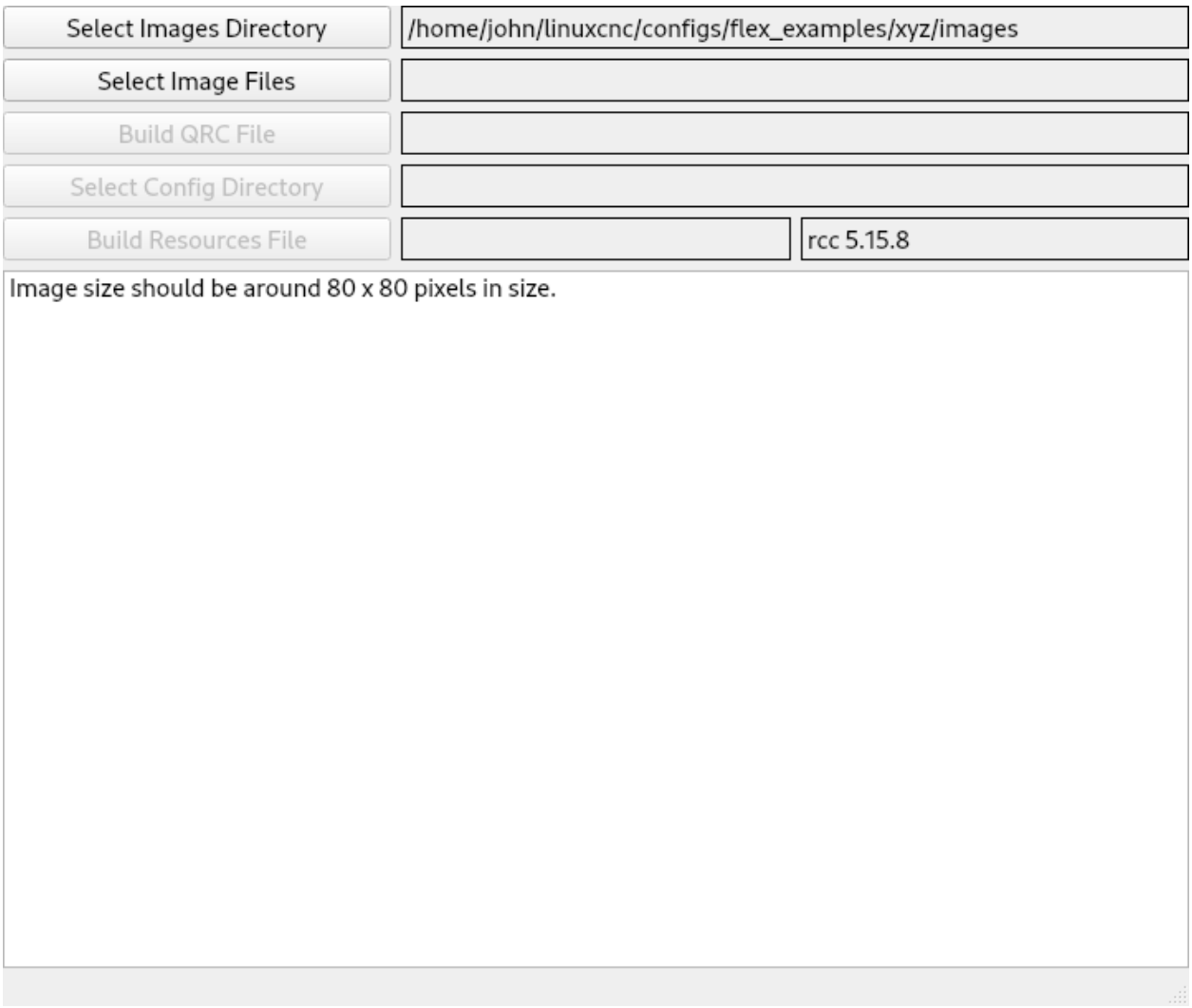
Startup



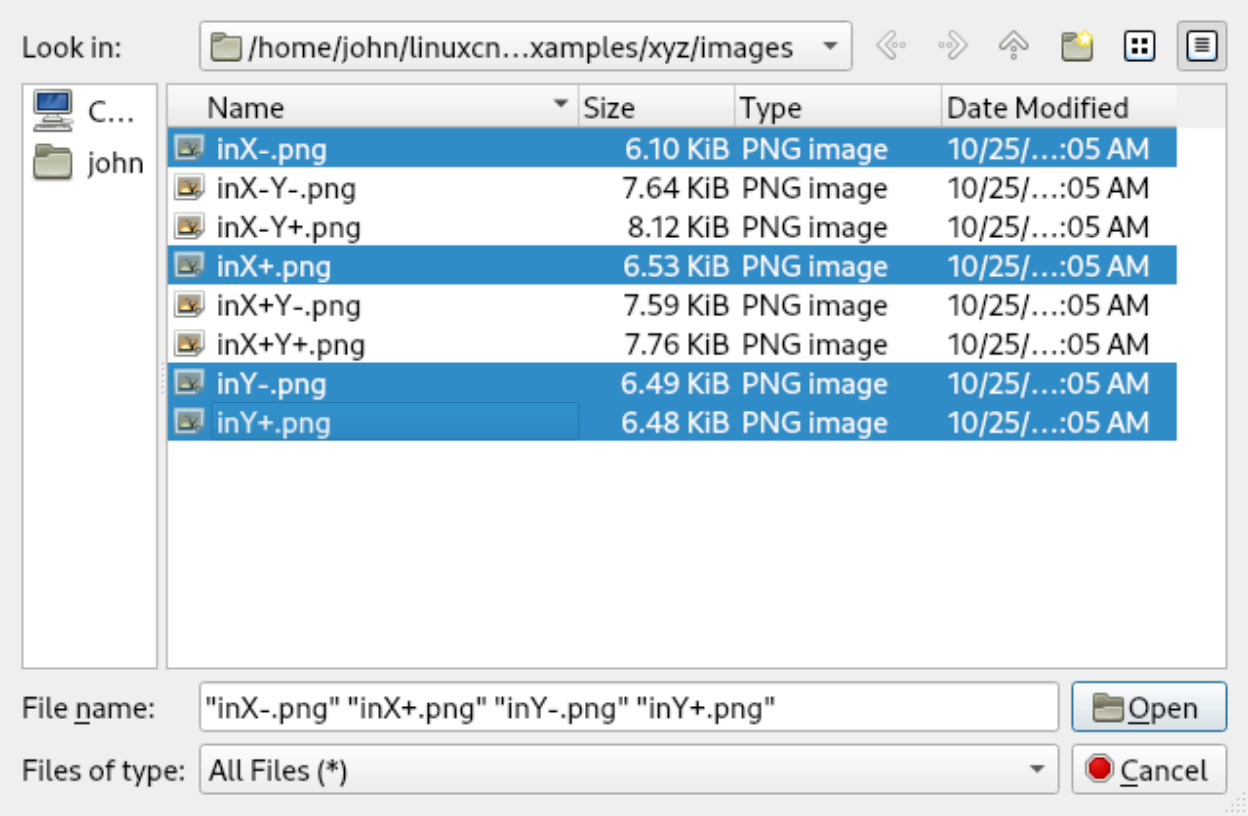
Next Select Images Directory



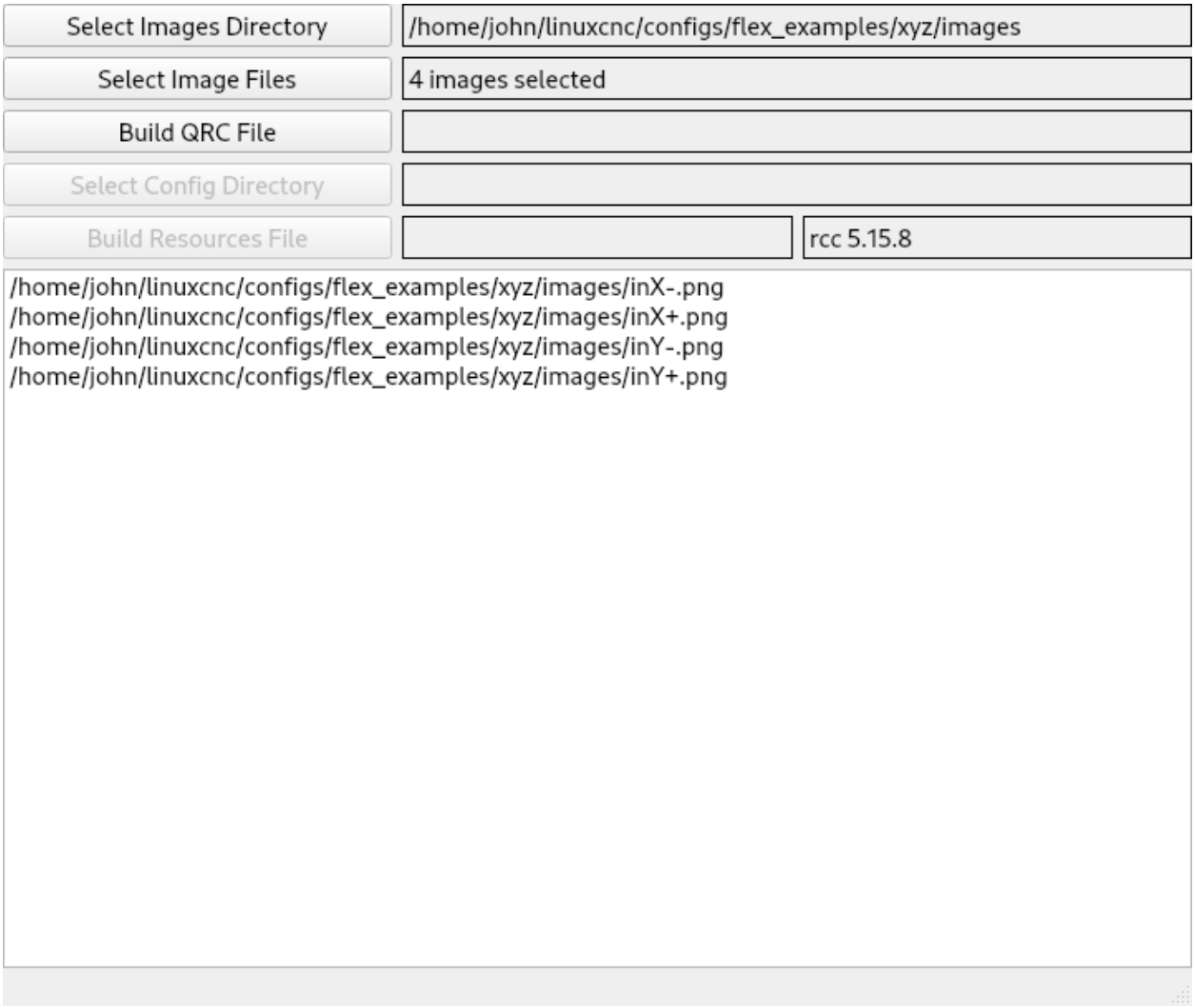
The selected directory is shown in the label



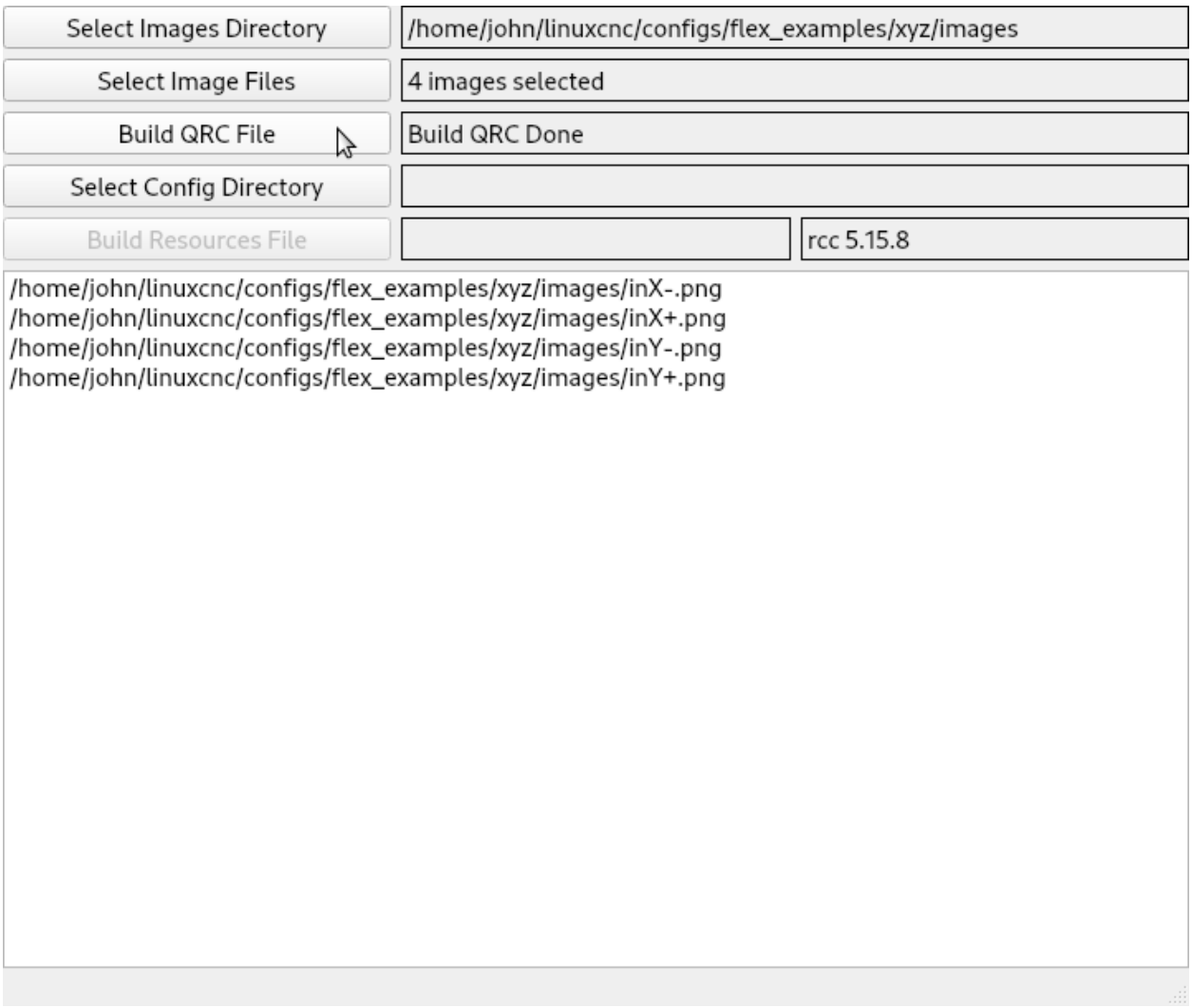
Next Select Image Files. To select all the images left click on the first one and hold down the shift key and left click on the last one. To pick several images but not all hold down the ctrl key while you left click on each one.



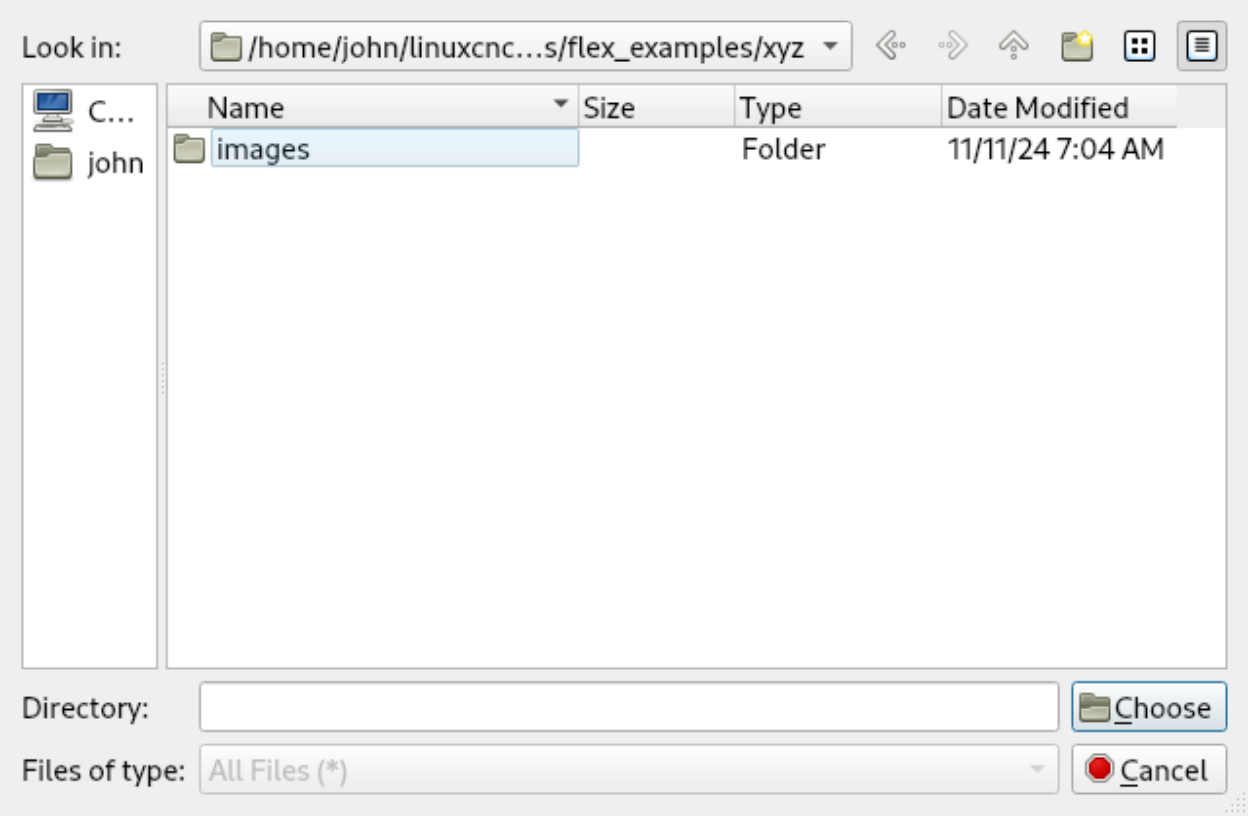
The images selected are shown below



Next Build QRC File

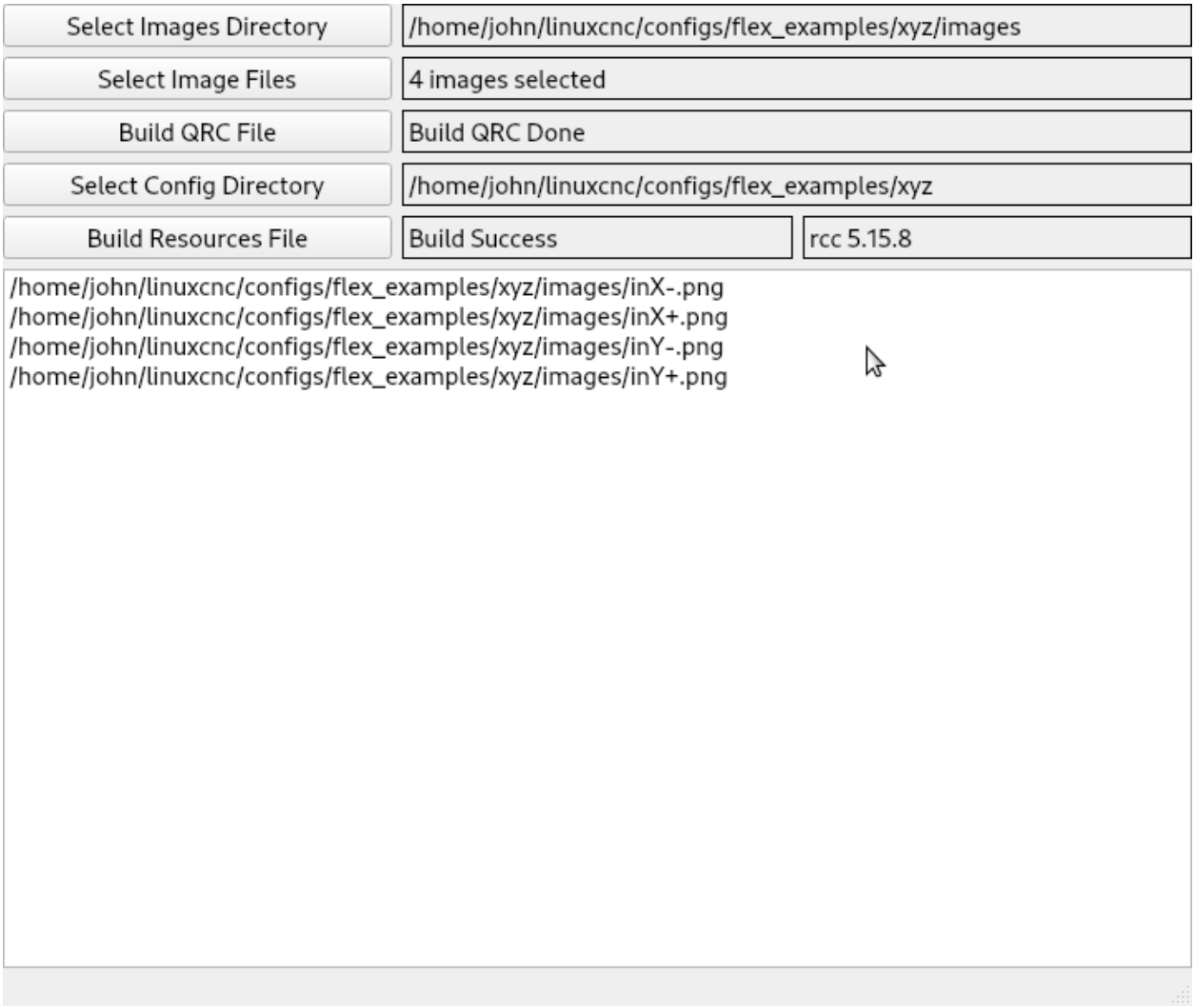


Next Select Config Directory



Note: The Image directory and the configuration directory must be different

Next Build Resources File



The Flex Resource Builder can be closed now. In the configuration directory you will have a resources.py file that contains the images used by the stylesheet.

Next edit the ini file and in the [DISPLAY] section add the following line

```
RESOURCES = resources.py
```

In the [DISPLAY] section add the style sheet

```
QSS = xyz.qss
```

To add an image named my-image.png to a QPushButton with an object name of my_pb add the following to the qss file

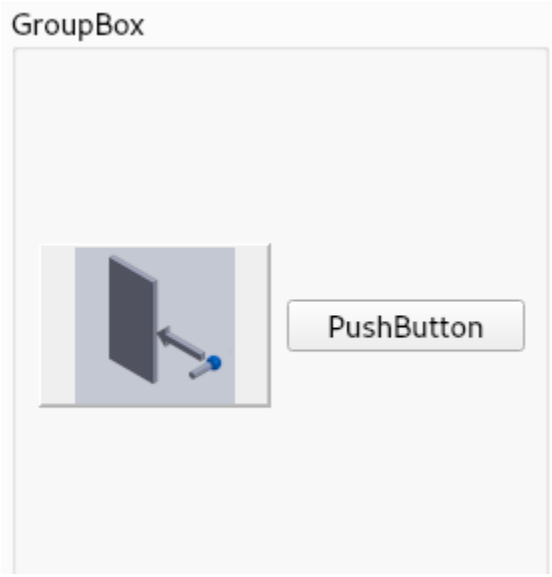
```
QPushButton#my_pb {
        min-height: 80px;
        min-width: 80px;
        margin: 2px;
        background-position: center;
        background-origin: content;
        background-clip: padding;
```

(continues on next page)

(continued from previous page)

```
        background-repeat: no-repeat;
        background-image: url(:my-image.png);
}
```

Now when you run the configuration the image will be on the QPushButton



Note: Delete any text in the QPushButton or it will be on top of the image