These changes complete those proposed (thread #1775)

```
# ----------------------------------------------------------------------
    # code to change manually tool
    # ----------------------------------------------------------------------
    def toolChange(self, tool=None):
        if tool is not None:
            # Force a change
            self.tool = tool
            self._lastTool = None

        # check if it is the same tool
        if self.tool is None or self.tool == self._lastTool:
            return []

        # create the necessary code
        lines = []
 -      # remember state and populate variables,
 -      # FIXME: move to ./controllers/_GenericController.py
 -      lines.append(
 -          "$g"
 -      )
        lines.append("m5")                                  # stop spindle
        lines.append("%wait")
        lines.append("%_x,_y,_z = wx,wy,wz")  # remember position
+       lines.append("g49")                                # cancel TLO
        lines.append("g53 g0 z[toolchangez]")
        lines.append("g53 g0 x[toolchangex] y[toolchangey]")
        lines.append("%wait")
+       lines.append("m0")                                 # feed hold
+
        if CNC.comment:
            lines.append(
                f"%msg Tool change T{int(self.tool):02} ({CNC.comment})")
        else:
            lines.append(f"%msg Tool change T{int(self.tool):02}")
-       lines.append("m0")  # feed hold

        if CNC.toolPolicy < 4:
            lines.append("g53 g0 x[toolprobex] y[toolprobey]")
            lines.append("g53 g0 z[toolprobez]")                    # fast approach
+           lines.append("g91")
-           # fixed WCS
-           if CNC.vars["fastprbfeed"]:
-               prb_reverse = {"2": "4", "3": "5", "4": "2", "5": "3"}
-               CNC.vars["prbcmdreverse"] = (
-                   CNC.vars["prbcmd"][:-1]
-                   + prb_reverse[CNC.vars["prbcmd"][-1]]
-              )
-               currentFeedrate = CNC.vars["fastprbfeed"]
-               while currentFeedrate > CNC.vars["prbfeed"]:
-                   lines.append("%wait")
-                   lines.append(
-                       f"g91 [prbcmd] {CNC.fmt('f', currentFeedrate)} "
-                        f"z[toolprobez-mz-tooldistance]"
-                   )
+           lines.append("[prbcmd] z-[tooldistance] f[fastprbfeed]") # switch search
+           lines.append("g0 z[1]")                              # switch clearence
+           lines.append("[prbcmd] z-[2] f[prbfeed]")                # measure
+           lines.append("g90")                              # restore mode
-                   lines.append("%wait")
-                   lines.append(
-                       f"[prbcmdreverse] {CNC.fmt('f', currentFeedrate)} "
-                       f"z[toolprobez-mz]"
-                   )
-                   currentFeedrate /= 10
-           lines.append("%wait")
-           lines.append(
-               "g91 [prbcmd] f[prbfeed] z[toolprobez-mz-tooldistance]")

            if CNC.toolPolicy == 2:
                # Adjust the current WCS to fit to the tool
                # FIXME could be done dynamically in the code
                p = WCS.index(CNC.vars["WCS"]) + 1
                lines.append(f"g10l20p{int(p)} z[toolheight]")
                lines.append("%wait")

            elif CNC.toolPolicy == 3:
                # Modify the tool length, update the TLO
                lines.append("g4 p1")  # wait a sec to get the probe info
                lines.append("%wait")
                lines.append("%global TLO; TLO=prbz-toolmz")
                lines.append("g43.1z[TLO]")
                lines.append("%update TLO")

            lines.append("g53 g0 z[toolchangez]")
            lines.append("g53 g0 x[toolchangex] y[toolchangey]")

        if CNC.toolWaitAfterProbe:
            lines.append("%wait")
            lines.append("%msg Restart spindle")
            lines.append("m0")  # feed hold

+       if CNC.comment:
+           lines.append("%%msg T%02d (%s)" %(self.tool,CNC.comment)) # tool in action
+       else:
+           lines.append("%%msg T%02d" %(self.tool))
+
        # restore state
-       lines.append("g90")  # restore mode
        lines.append("g0 x[_x] y[_y]")  # ... x,y position
        lines.append("g0 z[_z]")  # ... z position
        lines.append("f[feed] [spindle]")  # ... feed and spindle
        lines.append("g4 p5")  # wait 5s for spindle to speed up

        # remember present tool
        self._lastTool = self.tool
        return lines
```

probePage.py

```
 # ------------------------------------------------------------------------
-   def updateTool(self):
+    def updateToolHeight(self):
        state = self.toolHeight.cget("state")
        self.toolHeight.config(state=NORMAL)
        self.toolHeight.set(CNC.vars["toolheight"])
        self.toolHeight.config(state=state)
+        self.event_generate("<<StateTool>>")

# ------------------------------------------------------------------------
    def calibrate(self, event=None):
        self.set()
        if self.check4Errors():
            return
        lines = []
        lines.append("g53 g0 z[toolchangez]")
        lines.append("g53 g0 x[toolchangex] y[toolchangey]")
        lines.append("g53 g0 x[toolprobex] y[toolprobey]")
        lines.append("g53 g0 z[toolprobez]")
+        lines.append("g91")
-        if CNC.vars["fastprbfeed"]:
-            prb_reverse = {"2": "4", "3": "5", "4": "2", "5": "3"}
-            CNC.vars["prbcmdreverse"] = (
-                CNC.vars["prbcmd"][:-1] + prb_reverse[CNC.vars["prbcmd"][-1]]
-            )
-            currentFeedrate = CNC.vars["fastprbfeed"]
-            while currentFeedrate > CNC.vars["prbfeed"]:
-                lines.append("%wait")
-                lines.append(
-                    f"g91 [prbcmd] {CNC.fmt('f', currentFeedrate)} "
-                    + "z[toolprobez-mz-tooldistance]"
-                )
+        lines.append("[prbcmd] z-[tooldistance] f[fastprbfeed]")   # switch search
+        lines.append("g0 z[1]")                                    # Switch clearence
+        lines.append("[prbcmd] z[-2] f[prbfeed]")                  # Measure
+        lines.append("g90")                                        # restore initial state
-                lines.append("%wait")
-                lines.append(
-                    f"[prbcmdreverse] {CNC.fmt('f', currentFeedrate)} "
-                    + "z[toolprobez-mz]"
-                )
-                currentFeedrate /= 10
-        lines.append("%wait")
-        lines.append("g91 [prbcmd] f[prbfeed] z[toolprobez-mz-tooldistance]")
        lines.append("g4 p1")  # wait a sec
        lines.append("%wait")
        lines.append("%global toolheight; toolheight=wz")
        lines.append("%global toolmz; toolmz=prbz")
+        lines.append("%global Zsensor; Zsensor=wz")
        lines.append("%update toolheight")
+        lines.append("%update Zsensor")
        lines.append("g53 g0 z[toolchangez]")
        lines.append("g53 g0 x[toolchangex] y[toolchangey]")
-        lines.append("g90")
        self.app.run(lines=lines)
```

bmain.py

```
    def __init__(self, master, **kw):

        # Canvas X-bindings
        self.bind("<<ViewChange>>", self.viewChange)
        self.bind("<<AddMarker>>", self.canvas.setActionAddMarker)
        self.bind("<<MoveGantry>>", self.canvas.setActionGantry)
        self.bind("<<SetWPOS>>", self.canvas.setActionWPOS)

        frame = Page.frames["Probe:Tool"]
        self.bind("<<ToolCalibrate>>", frame.calibrate)
        self.bind("<<ToolChange>>", frame.change)

+       self.bind('<<ProbeTLO>>', frame.updateTLO)
+       self.bind('<<StateTLO>>', self.gstate.updateTLO)
+       self.bind('<<ProbeTool>>', frame.updateToolHeight)
+
        self.bind("<<AutolevelMargins>>", self.autolevel.getMargins)
        self.bind("<<AutolevelZero>>", self.autolevel.setZero)
        self.bind("<<AutolevelClear>>", self.autolevel.clear)
        self.bind("<<AutolevelScan>>", self.autolevel.scan)
        self.bind("<<AutolevelScanMargins>>", self.autolevel.scanMargins)

        self.bind("<<CameraOn>>", self.canvas.cameraOn)
        self.bind("<<CameraOff>>", self.canvas.cameraOff)

def _monitorSerial(self):

        # Update probe and draw point
        if self._probeUpdate:
            Page.frames["Probe:Probe"].updateProbe()
            Page.frames["ProbeCommon"].updateSensor() # see discussion #1775
+           Page.frames["State"].updateTLO()
            self.canvas.drawProbe()
            self._probeUpdate = False

        # Update any possible variable?
        if self._update:
            if self._update == "toolheight":
                Page.frames["Probe:Tool"].updateToolHeight()
            elif self._update == "TLO":
                Page.frames["ProbeCommon"].updateSensor() # see discussion #1775
+           elif self._update == "TLO":
+               Page.frames["State"].updateTLO()
            self._update = None
```

controlPage.py

```
   # ----------------------------------------------------------------------
    def setTLO(self, event=None):
        try:
            tlo = float(self.tlo.get())
            self.sendGCode(f"G43.1Z{tlo:g}")
            self.app.mcontrol.viewParameters()
            self.event_generate("<<CanvasFocus>>")
+           self.event_generate("<<ProbeTLO>>")
        except ValueError:
            pass

    # ----------------------------------------------------------------------
    def setTool(self, event=None):

   # ----------------------------------------------------------------------
+   def updateTLO(self, event=None):
+               state = self.tlo.cget("state")
+               self.tlo.config(state=NORMAL)
+          self.tlo.set(f'{float(CNC.vars["TLO"]):6.3f}')
+               self.tlo.config(state=state)
```