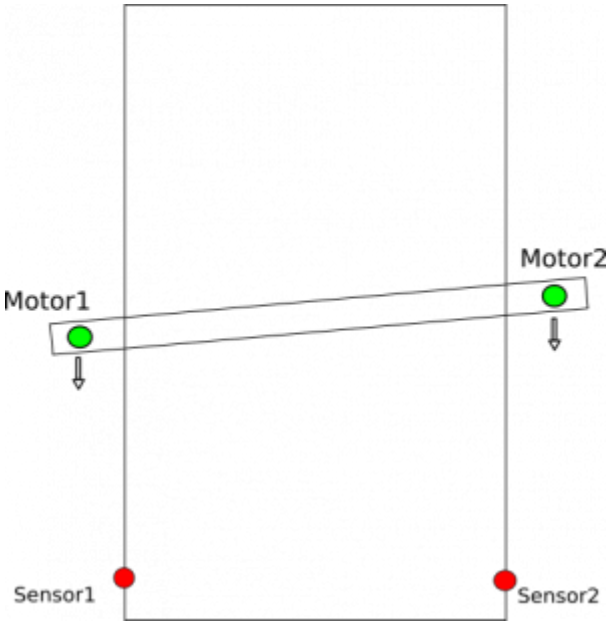
Gantry Alignment

Gantry Alignment macro wizard helps to generate macro for dual motor gantry alignment. There can be several methods for gantry alignment. Almost any of method can be implemented by (advanced) user through macro/PLC layer. We can offer 2 methods for gantry alignment.

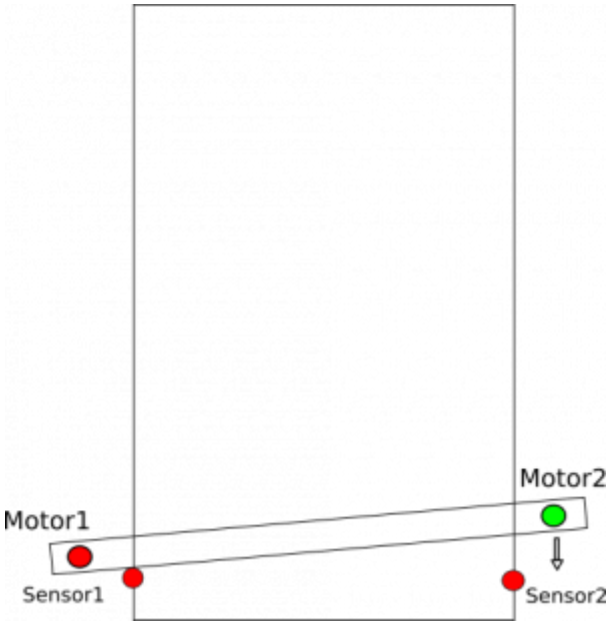
Method 1

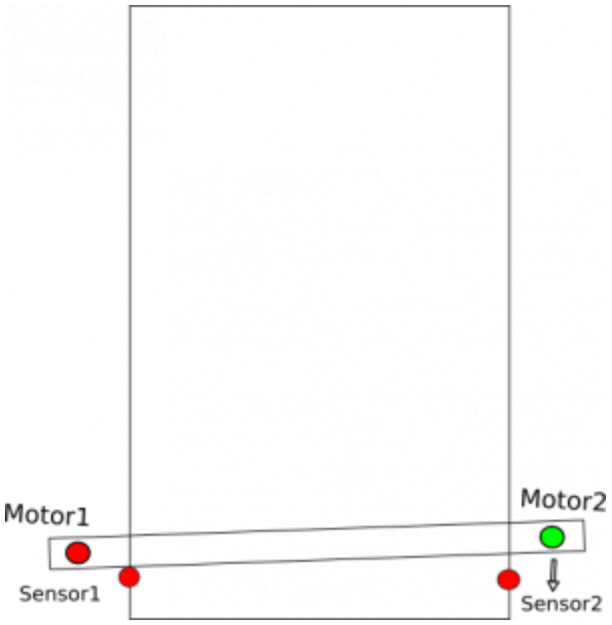
2 sensors on each side of gantry used to find side position. The alignment procedure is described below-

1. CNC activates motors on both sides and moves toward the sensors.

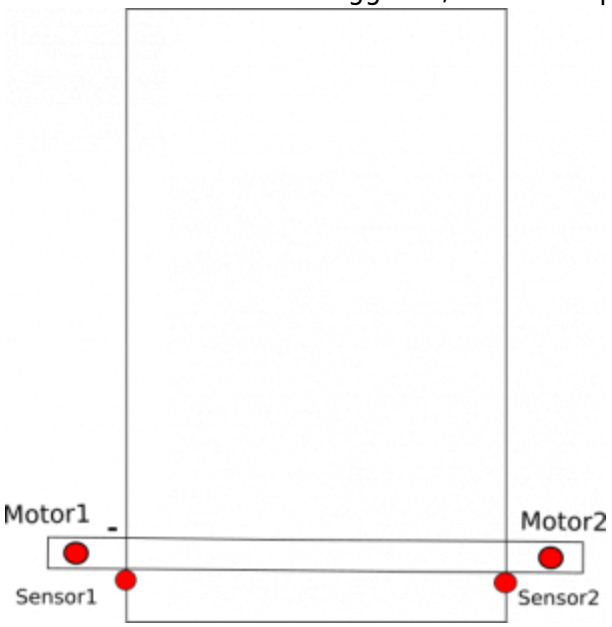


2. When one of the sensors is triggered, CNC control disconnects motor on this side, the 2nd motor continues to move.

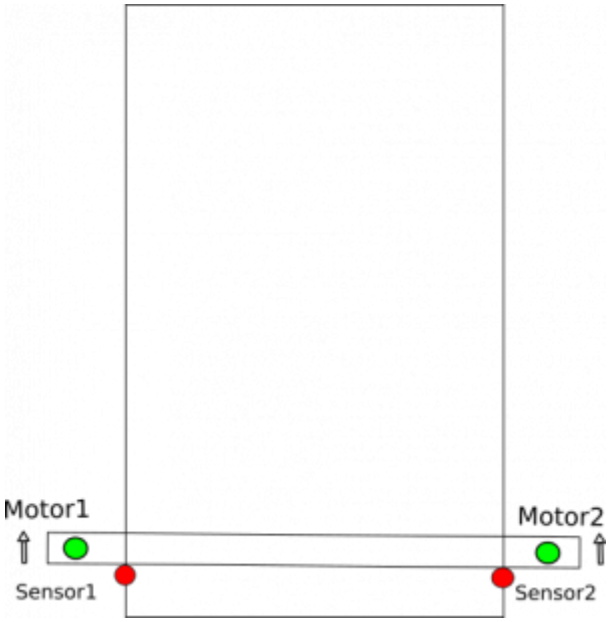




3. When both sensors are triggered, motion stopped.

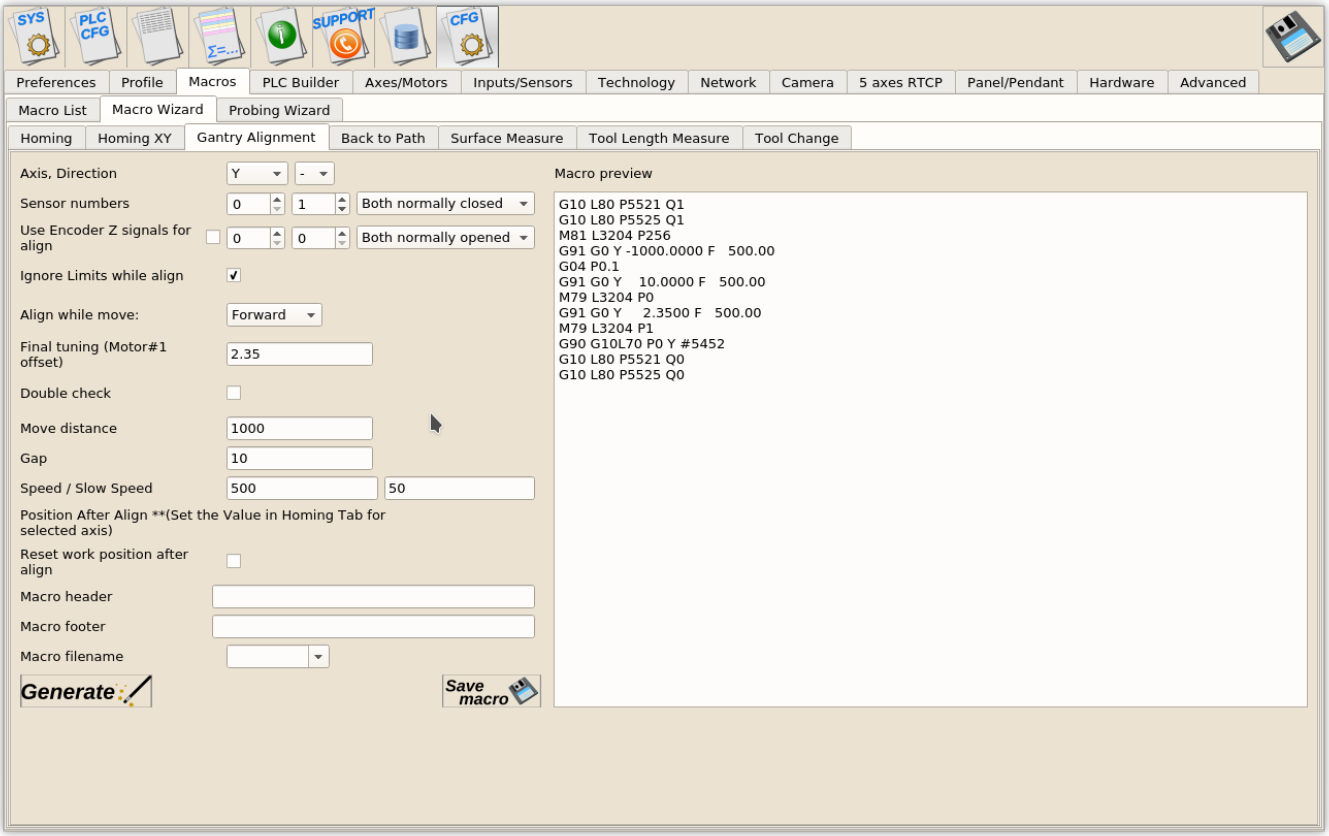


4. In case of sensors, the position should be adjusted at this point can be setup move of 1 motor to given adjustment distance.
5. Both motors are activated and move together.



6. The machine moves out from the sensors for given distance.

To generate this kind of alignment macro can be used the wizard. See below a screenshot of the Gantry Alignment Wizard



On the screen we have

1. **Axis, Direction** - Macro is generated for Y axis
2. While alignment machine moves toward Y-MIN
3. **Sensor numbers** - sensor #0, #1 are used as sensors on each side of gantry, both senors are “Normally Closed”
4. **Use Encoder Z signals for align** - not used
5. **Ignore Limits while align** - If sensors #0 or #1 is used as limit sensor, then Limits should be

ignored (put check-box) while alignment to prevent abort procedure.
6. **Align while move** - Forward
7. **Final tuning (Motor #1 offset)** - set distance to adjust sensors
8. **Double check** (to be described later)
9. **Move distance** Distance expected from start position to alignment sensors. This distance should be set a more than real distance to alignment sensors.
10. **Gap** - move out from the sensors for given distance.
11. **Speed** - speed for motion toward the sensors
12. **Slow Speed** - not used for “Align while move:Forward” method.
13. **Reset work position after align**
14. **Macro filename** - File name to ave macro source to file.

To generate and save macro (As usual for the Macro wizard) 3 buttons should be pressed

1. **Save** (top-right corner - to save macro wizard settings and use them for future macro generate
2. **Generate** - to generate macro source and show it in “Macro preview” window. If Generate button pressed, new macro is shown ONLY in the preview window and is not activated !! At this point, auser can manually edit the macro source in **Macro preview** window
3. **Save Macro** - save macro from **Macro preview** for the macro file with the name given in **Macro filename**, then reload all macros to CNC memory.

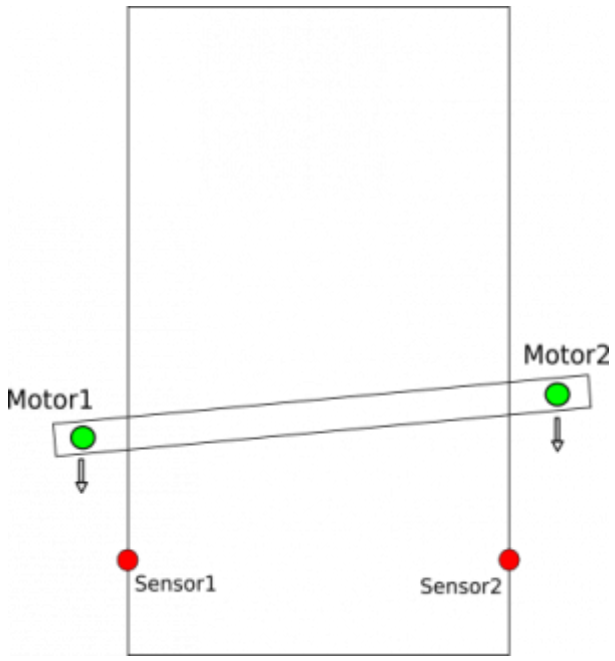
Gantry Alignment procedure either can work simultaneously as a homing procedure (if the alignment sensor is used as a homing sensor as well), or Homing procedure can be run just after gantry alignment procedure.

Method 2

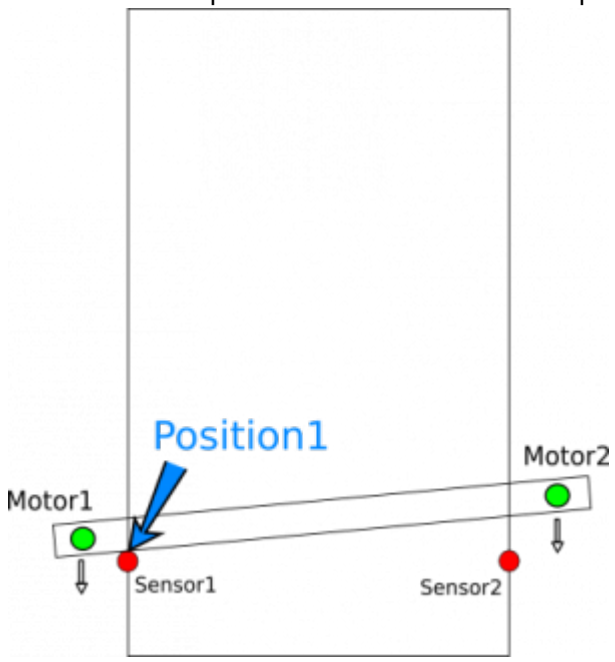
2 sensors on each side of gantry used to find each side positions. When both sensors actual positions found, CNC control moves the only side to get the gantry aligned.

The alignment procedure is described below-

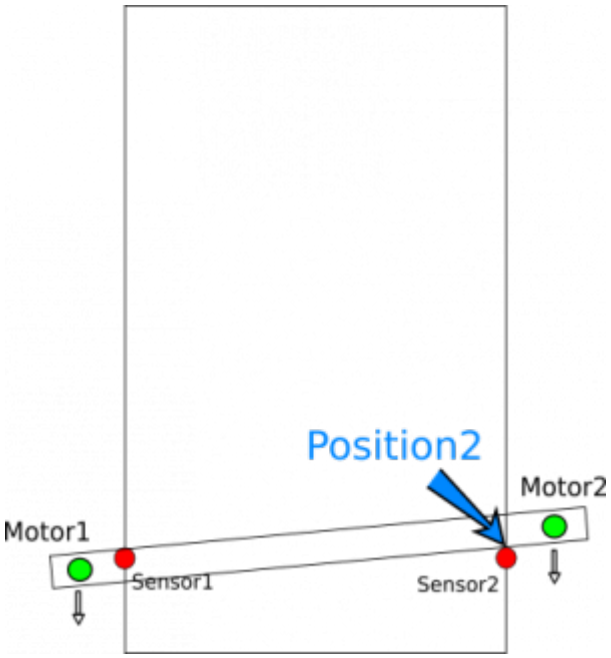
1. CNC activates motors on both sides and moves toward the sensors.



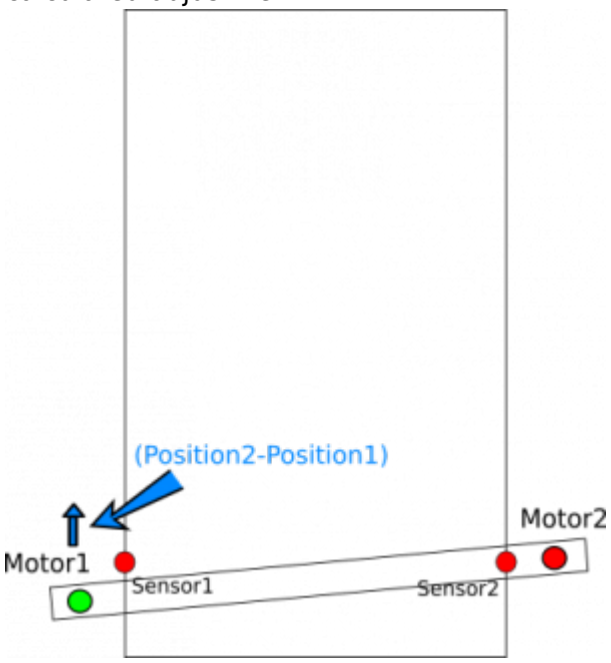
2. PLC controoler procedure latches machine positions at the moment every sensor is triggered.



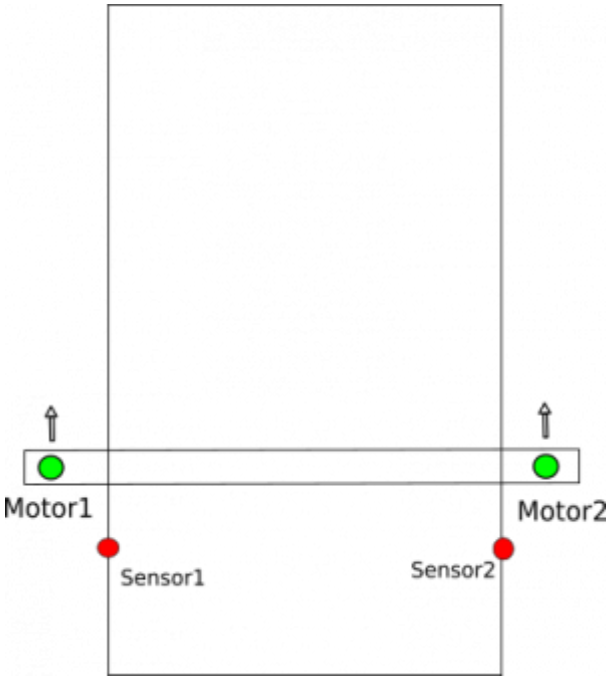
3. Motion stopped at the moment both sensors are triggered.



4. PLC procedure calculates the difference between latched positions and add an adjustment offset
5. The second motor control temporarily disabled, the first motor only is rotated accordingly calculated adjustment



6. Then the second motor control is activated again and both motors moved together



That's the plan. How to implement it? Here is step-by-step implementation description.

- CNC activates motors on both sides and moves toward the sensors.
  Actually both motors for the gantry should be configured in “Axes/Motors” configuration widget by default. However, extra M-code can be used to be sure both motors are turned on. For example, M219.plc connects axis X to both motor output #0 and #1

M219.plc

```
#include vars.h
main()
{

 parameter=0+16; //"0" is axis X; "16" is axis inversion.
 //parameter=1+16; //"1" is axis Y; "16" is axis inversion.
 //parameter=2+16; //"2" is axis Z; "16" is axis inversion.
 //parameter=3+16; //"3" is axis A; "16" is axis inversion.
 //parameter=4+16; //"4" is axis B; "16" is axis inversion.
 //parameter=5+16; //"5" is axis C; "16" is axis inversion.

 command=112+0; //Motor output #0
 message=PLCCMD_SET_CNC_VAR;
 timer=2;do{timer--;}while(timer>0);

 command=112+1; //Motor output #1
 parameter=0+16; //"0" is axis X; "16" is axis inversion.
 message=PLCCMD_SET_CNC_VAR;
 timer=2;do{timer--;}while(timer>0);

 exit(99);
};
```

- PLC controller procedure latches machine positions at the moment every sensor is triggered. For this example, we use procedure M144.plc written specially for the example. Any other implementation can be used as well

M144.plc

```
#include pins.h

// Look after INPUT_GANTRY_S1 & INPUT_GANTRY_S2 input pins
// Position X is stored in register #801 when sensor s1 triggered
// Position X is stored in register #802 when sensor s2 triggered
// If both sensors triggered, then
//  - Motion stopped
//  - Position difference is calculated and stored into register #800

main()
{
 timer=0;

 message=PLCCMD_MOTION_CONTINUE;
 texit=timer+30;do{timer++;}while(timer<texit);

 mode_12=0;

 ready=0;

 s1_old=portget(INPUT_GANTRY_S1);
 s2_old=portget(INPUT_GANTRY_S2);

 do
 {
 timer++;

 s1=portget(INPUT_GANTRY_S1);
 s2=portget(INPUT_GANTRY_S2);

 mode1=mode_12&1;

 if (mode1==0)//if sensor 1 wasn't triggered
 {
  if (s1!=s1_old) //if triggered now
   {
     mode_12=mode_12|1; //set flag for sensor1
     position1=gvarget(5021+0);  //Save Machine X position
   };
 };

 mode2=mode_12&2;
```

```
 if (mode2==0)//if sensor 2 wasn't triggered
 {
  if (s2!=s2_old) //if triggered now
   {
     mode_12=mode_12|2; //set flag for sensor2
     position2=gvarget(5021+0);  //Save Machine X position
   };
 };

 }while(mode_12!=3);

 message=PLCCMD_MOTION_SKIP;
 timer=2;do{timer--;}while(timer>0);

 gvarset(801,position1); //send the position to myCNC software Register
#801
 timer=50;do{timer--;}while(timer>0);
 gvarset(802,position2); //send the position to myCNC software Register
#802
 timer=50;do{timer--;}while(timer>0);

 offset=position1-position2;
 gvarset(800,offset);
 timer=50;do{timer--;}while(timer>0);

 exit(99);
};
```

The M144.plc procedure uses predefined values as the input pin numbers the alignment sensors connected to

```
#include pins.h
...
...

 s1_old=portget(INPUT_GANTRY_S1);
 s2_old=portget(INPUT_GANTRY_S2);


 do
 {
...
...

 s1=portget(INPUT_GANTRY_S1);
 s2=portget(INPUT_GANTRY_S2);

...
...
```

Actual input numbers should be defined in define file “pins.h”. The content of “pins.h” file should be included at the start of each macro. If using this programming style, the “pins.h” is the single file you need to edit to customize your input/output assignments.

pins.h

```
...
...
 #define INPUT_GANTRY_S1 3
 #define INPUT_GANTRY_S2 4
...
...
```

You can put numbers directly to portget/portset functions of PLC source (however, we don't recommend this). It will look like this

```
...

 s1_old=portget(3);
 s2_old=portget(4);

 do
 {
...
 s1=portget(3);
 s2=portget(4);
...
```

- Motion stopped at the moment both sensors are triggered. This is implemented in the M144.plc

PLC sends message to the Motion controller to end the current motion and jump to the next g-code line

```
 message=PLCCMD_MOTION_SKIP;
 timer=2;do{timer--;}while(timer>0);
```

- PLC procedure calculates the difference between latched positions and adds an adjustment offset. This is implemented in M144.plc as well

```
 gvarset(801,position1); //send the position #1 to myCNC software Register
#801
 timer=50;do{timer--;}while(timer>0);
 gvarset(802,position2); //send the position #2 to myCNC software Register
#802
 timer=50;do{timer--;}while(timer>0);

 offset=position1-position2; //calculate the difference
 gvarset(800,offset); //send the difference to myCNC software Register #800
```

```
timer=50;do{timer--;}while(timer>0);
```

- The second motor control temporarily disabled, the first motor only is rotated accordingly calculated adjustment.

Connect/disconnect Axis X from motor output pin is implemented in the M210.plc procedure. Depends on external parameter (which is P-parameter in G-code line), the procedure sends either “15” to disconnect motor output #1 or “16” to assign **Inverted Axis X** to the motor output #1.

M210.plc

```
#include vars.h

main()
{
 connect=eparam;

 message=PLCCMD_SET_CNC_VAR;
 if (connect==0)
 {
  parameter=15;   //OFF
 }else
 {
  parameter=0+16; //"0" is axis X; "16" is axis inversion.
 };
 command=112+1; //Motor output #1
 timer=2;do{timer--;}while(timer>0);

 exit(99);
};
```

As a result,

- Code to disconnect motor output#1 is **M210 P0**
- Code to connect motor output#1 to axis X is **M210 P1**

- Then the second motor control is activated again and both motors moved together.
  G-code G28.9 can be used to move axis accordingly register value.

```
G28.9 X810 (# Move X coordinate to the position stored in register
#810)
```

A complete macro to gantry alignment (Method #2) is shown below

M139

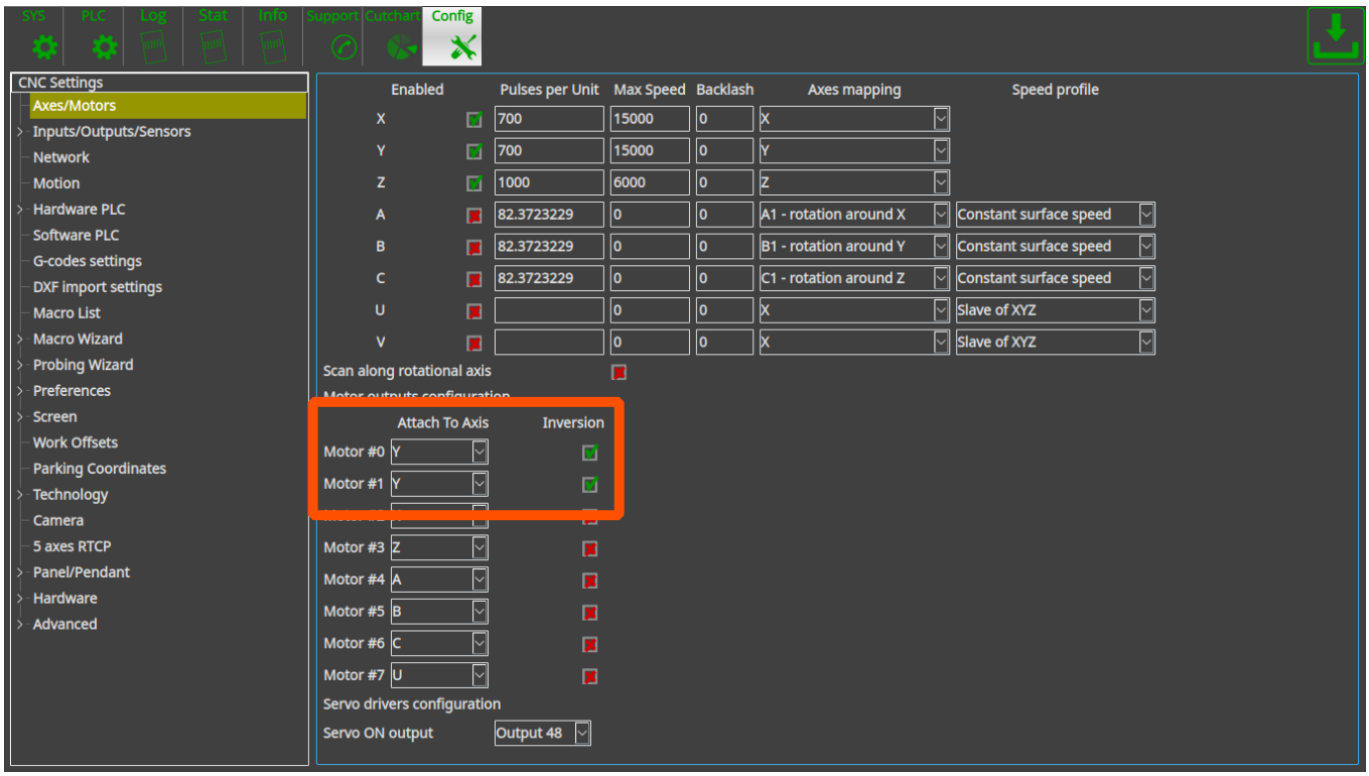
```
M144                    (# Set watching for GANTRY Sensors)
G91 G0 X-1000 F500      (# Move toward Gantry Sensors)

( It takes a time PLC controller from M144 procedure sends )
```

```
( updated register #800,#801,#802 values to myCNC control software )
( Add 0.5 seconds pause to get all the registers updated )

G4 P0.5                 (# Pause)

G10 L81 P810 Q800       (# Get offset in pulses)
G10 L192 P810 Q5531     (# Calculate offset in units/mm)

G10 L190 P810 Q805      (# Add adjustment offset)

M210 P0                 (# OFF the second motor, the Only first motor is ON )

G10 L190 P810 Q5021     (# Add current machine position)
G28.9 X810              (# Move 1 motor to adjust)

M210 P1                 (# ON the second motor, both motors are moved together)
```

Method 2+

What should be changed if Gantry alignment procedure needed for other axis (for example Y axis instead of X)

1) Let's say Motor outputs #0 and #1 configured as Y.



2) Procedure to connect both outputs to Y axis will look as

M219.plc

```
#include vars.h
main()
{

 //parameter=0+16; //"0" is axis X; "16" is axis inversion.
 parameter=1+16; //"1" is axis Y; "16" is axis inversion.
 //parameter=2+16; //"2" is axis Z; "16" is axis inversion.
 //parameter=3+16; //"3" is axis A; "16" is axis inversion.
 //parameter=4+16; //"4" is axis B; "16" is axis inversion.
 //parameter=5+16; //"5" is axis C; "16" is axis inversion.

 command=112+0; //Motor output #0
 message=PLCCMD_SET_CNC_VAR;
 timer=2;do{timer--;}while(timer>0);

 command=112+1; //Motor output #1
 parameter=1+16; //"1" is axis Y; "16" is axis inversion.
 message=PLCCMD_SET_CNC_VAR;
 timer=2;do{timer--;}while(timer>0);

 exit(99);
};
```

3) Need to change PLC procedure that looks after sensors to store Y position instead of X. Change procedure name to M145.plc

M145.plc

```
#include pins.h

// Look after INPUT_GANTRY_S1 & INPUT_GANTRY_S2 input pins
// Position Y is stored in register #801 when sensor s1 triggered
// Position Y is stored in register #802 when sensor s2 triggered
// If both sensors triggered, then
//   - Motion stopped
//   - Position difference is calculated and stored into register #800

main()
{
 timer=0;

 message=PLCCMD_MOTION_CONTINUE;
 texit=timer+30;do{timer++;}while(timer<texit);

 mode_12=0;
```

```
 ready=0;

 s1_old=portget(INPUT_GANTRY_S1);
 s2_old=portget(INPUT_GANTRY_S2);

 do
 {
 timer++;

 s1=portget(INPUT_GANTRY_S1);
 s2=portget(INPUT_GANTRY_S2);

 mode1=mode_12&1;

 if (mode1==0)//if sensor 1 wasn't triggered
 {
  if (s1!=s1_old) //if triggered now
   {
     mode_12=mode_12|1; //set flag for sensor1
     position1=gvarget(5021+1);  //Save the Machine Y position
   };
 };

 mode2=mode_12&2;

 if (mode2==0)//if sensor 2 wasn't triggered
 {
  if (s2!=s2_old) //if triggered now
   {
     mode_12=mode_12|2; //set flag for sensor2
     position2=gvarget(5021+1);  //Save the Machine Y position
   };
 };

 }while(mode_12!=3);

 message=PLCCMD_MOTION_SKIP;
 timer=2;do{timer--;}while(timer>0);

 gvarset(801,position1); //send the position to myCNC software Register
#801
 timer=50;do{timer--;}while(timer>0);
 gvarset(802,position2); //send the position to myCNC software Register
#802
 timer=50;do{timer--;}while(timer>0);

 offset=position1-position2;
 gvarset(800,offset);
 timer=50;do{timer--;}while(timer>0);
```

```
 exit(99);
};
```

4) Fix disable motor PLC procedure. Change procedure name to M211 (instead of M210)

M211.plc

```
#include vars.h

main()
{
 connect=eparam;

 message=PLCCMD_SET_CNC_VAR;
 if (connect==0)
 {
  parameter=15;   //OFF
 }else
 {
  parameter=1+16; //"1" is axis Y; "16" is axis inversion.
 };
 command=112+1; //Motor output #1
 timer=2;do{timer--;}while(timer>0);

 exit(99);
};
```

5) change the alignment macro to use Y axis instead of X axis

M139

```
M145                    (# Set watching for GANTRY Sensors)
G91 G0 Y-1000 F500      (# Move toward Gantry Sensors)

( There is message delivery time from PLC to CNC control software)
( Need to wait some time till registers #800-802 will be updated )
( Add 0.5 seconds pause to get all the registers refreshed )

G4 P0.5                 (# Pause)

G10 L81 P810 Q800       (# Get offset in pulses)
G10 L192 P810 Q5531     (# Calculate offset in units/mm)

G10 L190 P810 Q805      (# Add adjustment offset)

M211 P0                 (# OFF the second motor, the Only first motor is
ON )
```

```
G10 L190 P810 Q5021   (# Add current machine position)
G28.9 Y810            (# Move 1 motor to adjust)

M210 P1               (# ON the second motor, both motors are moved together)
```