PLC Examples

***NOTE**: The myCNC team recommends utilizing the examples provided in this manual (as well as other manuals in this documentation) as a starting point for your machine setup. When possible (and applicable), it is recommended to keep changes to a mininum. In general, using these examples as the basis for your PLCs/macro commands allows for an easier setup process.*

Spindle Speed control for DAC

SPN.plc

```
#define command     var00
#define parameter   var01
//set Spindle speed control via DAC channel #1
//Spindle Speed is given in **eparam** register

main()
{
  value=eparam;
  if (value>0xFFF) {value=0xFFF;};
  //fix if given value is out of range 0...0xfff
  if (value<0) {value=0;};

  dac01=value;   //setup DAC value


  //**Set Spindle Speed** is asynchronous operation.
  //It's better to inform myCNC Software New Spindle Speed applied.
  //Send information about new Spindle Speed to myCNC Software
  message=PLCCMD_REPLY_TO_MYCNC;
  //Command code to send to myCNC software
  command=PLC_MESSAGE_SPINDLE_SPEED_CHANGED;
  //Message code
  parameter=eparam;     //New Spindle Speed information
  timeout=timer+10; do { timer++; } while (timer<timeout);
  //Delay to push the Message to myCNC Software

  gvarset(7371,eparam);
  //myCNC register #7371 contains actual Spindle Speed.
  //Another way to inform myCNC software about new Spindle Speed
  //(to display on it DRO for example)

  exit(99);  //normal exit.
};
```

Spindle Control through Triggers

In this case, a couple of lines are added to standard M03 (spindle ON), M05 (spindle OFF) and M02 (End program) PLCs, typically to allow the system to interpret some spindle feedback signal. Spindle control is done through a trigger, with the trigger flag indicating whether the trigger is ON or OFF. If the spindle is ON and the trigger is activated, then the program will be stopped.

The following code can be used to enable the trigger:

```
  message=PLCCMD_TRIGGER1_ON;
  timer=10;do{timer--;}while(timer>0);
```

The following code can be used to disable the trigger:

```
  message=PLCCMD_TRIGGER1_OFF;
  timer=10;do{timer--;}while(timer>0);
```

The code to enable the trigger should be inserted into M03, while the code to disable the trigger should be inserted into M05 and M02. The full resultant PLC procedure code can be found below:

Click to expand the M03 code

```
//Turn on Spindle clockwise
#include pins.h
#include vars.h
main()
{
  timer=0;
  proc=plc_proc_spindle;

  val=eparam;
  if (val>0xfff) {val=0xfff;};
  if (val<0) {val=0;};

  dac01=val;

  portclr(OUTPUT_CCW_SPINDLE);
  portset(OUTPUT_SPINDLE);

  gvarset(7370,1); timer=30;do{timer--;}while (timer>0); //Spindle State
  gvarset(7371,eparam); timer=30;do{timer--;}while (timer>0); //Spindle
Speed Mirror register

  //gvarset(7372,0);//Mist State
  //timer=30;do{timer--;}while (timer>0); //
  //gvarset(7373,0);//Flood State
  //timer=30;do{timer--;}while (timer>0); //

  //delay after spindle started
  timer=spindle_on_delay;
```

```
  do{timer--;}while (timer>0); //delay for Spindle reach given speed

  message=PLCCMD_TRIGGER1_ON;
  timer=10;do{timer--;}while(timer>0);

  exit(99); //normal exit
};
```

Click to expand the M05 code

```
//Spindle Stop
//set Spindle speed control via DAC & PWM1 channel

#include pins.h
#include vars.h


main()
{
  portclr(OUTPUT_SPINDLE);
  portclr(OUTPUT_CCW_SPINDLE);
  proc=plc_proc_idle;

  message=PLCCMD_TRIGGER1_OFF;
  timer=10;do{timer--;}while(timer>0);

  if (spindle_off_delay!=0)
  {
    timer=spindle_off_delay;
    do { timer--; } while (timer>0);
  };

  dac01=0x0;

  gvarset(7370,0);  timer=30;do{timer--;}while(timer>0); //Spindle State
  gvarset(7371,0);  timer=30;do{timer--;}while(timer>0); //Spindle Speed

  exit(99);  //normal exit
};
```

Click to expand the M02 code

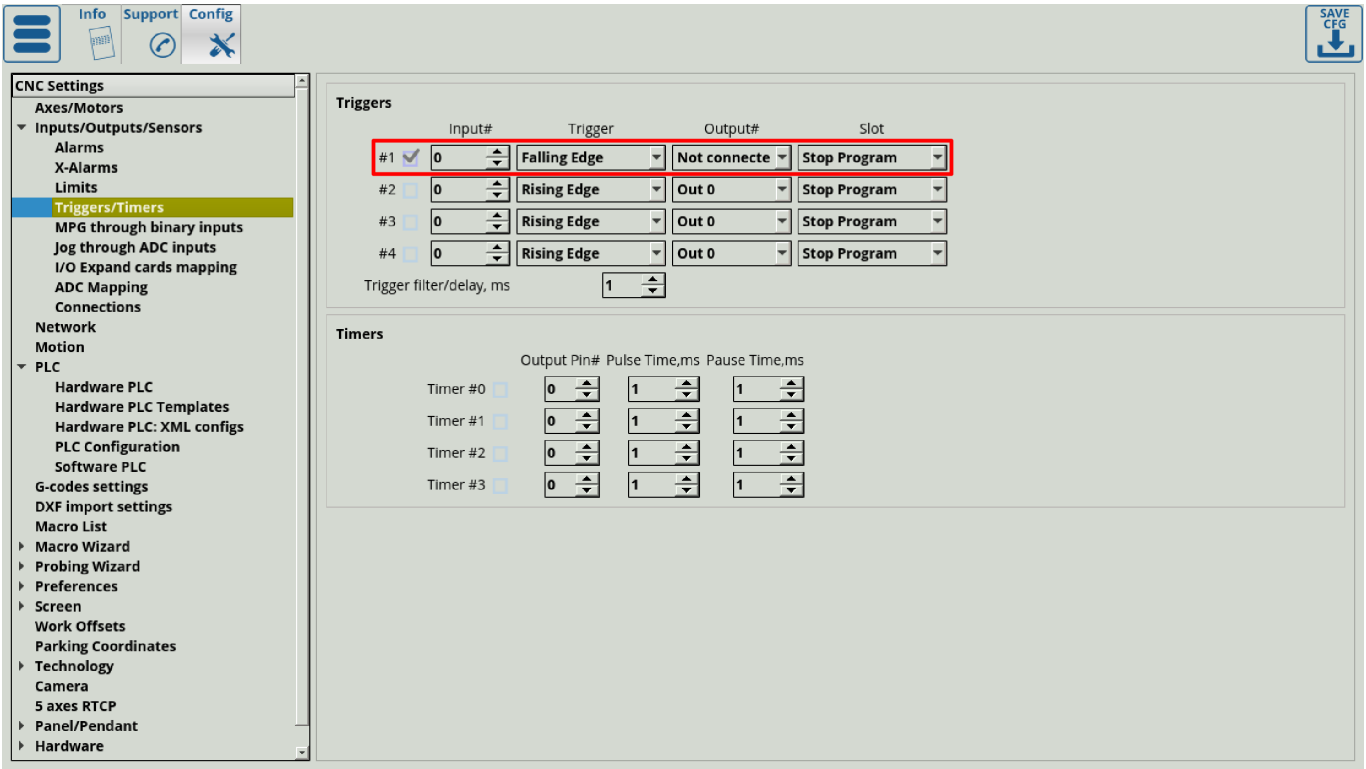
```
#include pins.h
#include vars.h

// g0moveA - start motion:
// flags -
// bit 0 - absolute programming
```

```
// bit 1 - machine coordinates
// bit 7 - delayed start.
// axes mask
// bit 0 - X axis; bit 1 - Y axis;bit 2 - Z axis;bit 3 - A axis;bit 4 - B
axis;bit 5 - C axis

lift_up()
{

  if (proc==plc_proc_spindle)
  {
    z1=gvarget(17003);
    timer=10; do{timer--;}while (timer>0);//wait till motion started

    z2=gvarget(7020);
    z2=z2*100;

    if (absolute==0) {  z2=z1+z2;  };

    z1=z1+100;  //add 1mm gap

    if (z2>z1)
    {   //position coordinate in given axis in 0.01 units (mm)
        gvarset(7080,speed_z);  //set speed
        g0moveA(1,0x4,z2);        //absolute programming; Z axis;
        timer=300; do{timer--;}while (timer>0);  //wait motion started
        //wait motion stopped
        do
        { ex=0; code=gvarget(6060);
          if (code==0x4d) {ex=1;};
          if (code==0x57) {ex=1;};
        } while(ex==0);
    };
  };

};

main()
{
  message=PLCCMD_TRIGGER4_ON;
  timer=2;do{timer--;}while(timer>0);

  if (absolute!=0) { absolute=1; };

  portclr(OUTPUT_MIST);
  portclr(OUTPUT_FLOOD);
  gvarset(7372,0);//Reset Mist State
  timer=30;do{timer--;}while(timer>0);
  gvarset(7373,0);//Reset Flood State
  timer=30;do{timer--;}while(timer>0);
```

```
  lift_up();

  dac01=0x0;    //off DAC output

  portclr(OUTPUT_SPINDLE);
  portclr(OUTPUT_CCW_SPINDLE);
  gvarset(7370,0);//Spindle State
  gvarset(7371,0);//Spindle Speed Mirror register

  message=PLCCMD_TRIGGER1_OFF;
  timer=10;do{timer--;}while(timer>0);

  proc=plc_proc_idle;
  exit(99);
};
```

More info on the M02 procedure available here: Stop/End program button commands.

The trigger itself can be set up in Settings > Config > Inputs/Outputs/Sensors > Triggers/Timers to look the following way:



- **Input Number** can be set to the port which the spindle feedback signal is using (#0 in this case)
- **Trigger** will be set to Falling Edge
- **Output** set to Not Connected
- **Slot** set to Stop Program

Spindle Speed control for ET10_DAC

SPN.plc

```
#define command     var00
#define parameter   var01
//set Spindle speed control via ET10 DAC channel #1
//Spindle Speed is given in **eparam** register

main()
{
  command=0x32;
  //EXT_ET10_DAC_OFFSET; set ADC offset register address
  parameter=0x800-(eparam/2)+(1<<12);
  //0x800 - is the middle of 12bits range - represents 0V
  //Eparam contains 12bits DAC value in 0V range, ET10 DAC setup in
+10V...-10V range, so need to /2
  //Encoder channel number is given in high 12 bits of 16bit word.

  message=PLCCMD_SET_CNC_EXTVAR;
  //setup Message register with command for access to [[External CNC
Variables]]
  texit=timer+2;do{timer++;}while(timer<texit);
  //2ms delay to push the command from PLC to myCNC Core


  //**Set Spindle Speed** is asynchronous operation.
  //It's better to inform myCNC Software New Spindle Speed applied.
  //Send information about new Spindle Speed to myCNC Software
  message=PLCCMD_REPLY_TO_MYCNC;                  //Command code to
send to myCNC software
  command=PLC_MESSAGE_SPINDLE_SPEED_CHANGED;      //Message code
  parameter=eparam;                               //New Spindle Speed
information
  timeout=timer+10; do { timer++; } while (timer<timeout); //Delay to
push the Message to myCNC Software

  gvarset(7371,eparam);
  //myCNC register #7371 contains actual Spindle Speed.
  //Another way to inform myCNC software about new Spindle Speed (to
display on it DRO for example)


exit(99);  //normal exit.
};
```

M03, Spindle On, Relay and ET10 DAC

M03.plc

```
//Turn on Spindle clockwise
//set Spindle speed control ET10 DAC channel #2
#include pins.h  //defines for pins numbers
#include vars.h  //defines for variable names

main()
{
  timer=0;
  value=eparam;

  command=0x32;//EXT_ET5_DAC_OFFSET
  parameter=0x800-(eparam/2)+(2<<12);//channel #2
  message=PLCCMD_SET_CNC_EXTVAR;
  texit=timer+2;do{timer++;}while(timer<texit);

  portclr(OUTPUT_CCW_SPINDLE);
  portset(OUTPUT_SPINDLE);


  gvarset(7370,1);
  //Global Register #7370 shown actual Spindle state (0=OFF, 1=ON).
  //Set Register value when Spindle is ON
  gvarset(7371,eparam);
  //myCNC register #7371 contains actual Spindle Speed.
  //Another way to inform myCNC software about new Spindle Speed (to
display on it DRO for example)


  //**Set Spindle Speed** is asynchronous operation.
  //It's better to inform myCNC Software New Spindle Speed applied.
  //Send information about new Spindle Speed to myCNC Software
  message=PLCCMD_REPLY_TO_MYCNC;                    //Command code to
send to myCNC software
  command=PLC_MESSAGE_SPINDLE_SPEED_CHANGED;        //Message code
  parameter=eparam;                                 //New Spindle Speed
information
  timeout=timer+10; do { timer++; } while (timer<timeout); //Delay to
push the Message to myCNC Software



  //Wait till Spindle Rotation Speed comes to good values before next
motion started
  timeout=timeout_on_delay+timer;
  do{timer++;}while (timer<timeout);                //delay for Spindle
reach given speed

  exit(99);                                         //normal exit.
};
```

Water Fill and Drain control

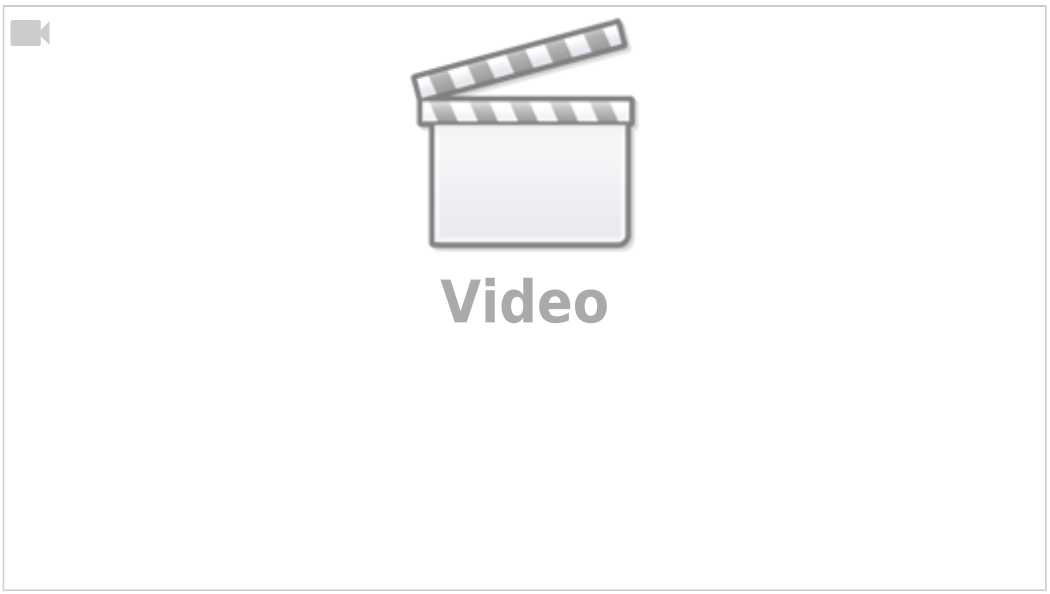
Procedure M240 is used in some plasma cutting machines to control Water Table Fill & Drain.

Running procedure with parameter “1” toggles Water Filling, running with parameter “0” toggles Water Draining. If try to ON both Fill & Drain, the procedure will turn off the previous relay to prevent conflicts.

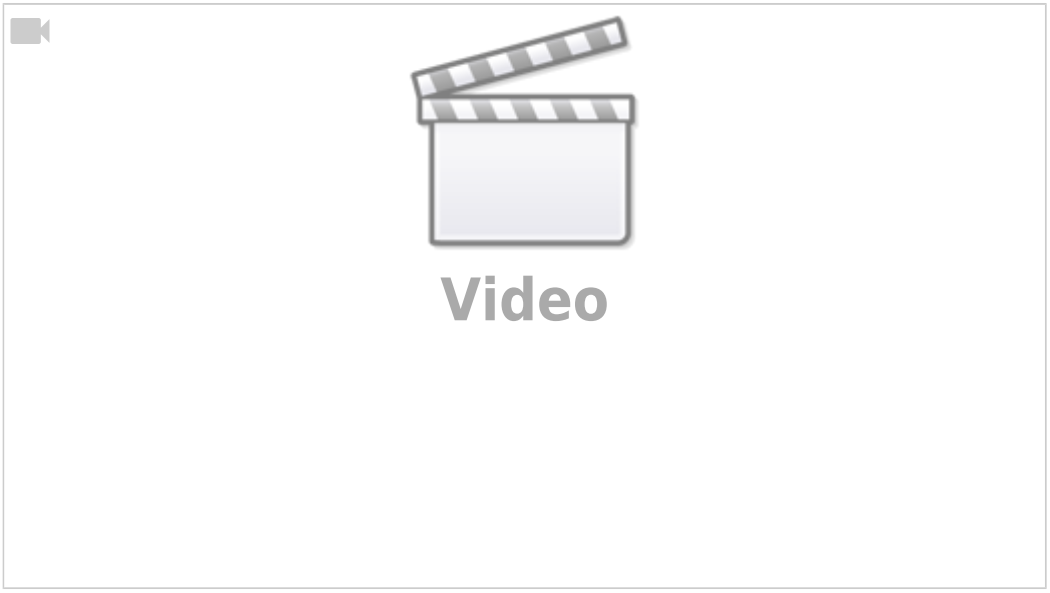
M240.plc

```
#define OUTPUT_FILL       13
#define OUTPUT_DRAIN      12

main()
{
  o=gvarget(7184); //read OUTPUT PORT 0 state (pins 0...31)
  drain_state=o&(1<<OUTPUT_DRAIN);
  fill_state=o&(1<<13);

  if (eparam==0) //toggle drain
  {
   if (drain_state==0)
   {
    portset(OUTPUT_DRAIN);
    portclr(OUTPUT_FILL);    //to prevent both are open
   }else
   {
    portclr(OUTPUT_DRAIN);
   };
  }else //toggle fill
  {
   if (fill_state==0)
   {
    portset(OUTPUT_FILL);
    portclr(OUTPUT_DRAIN);    //to prevent both are open
   }else
   {
    portclr(OUTPUT_FILL);
   };
  };
exit(99);
};
```

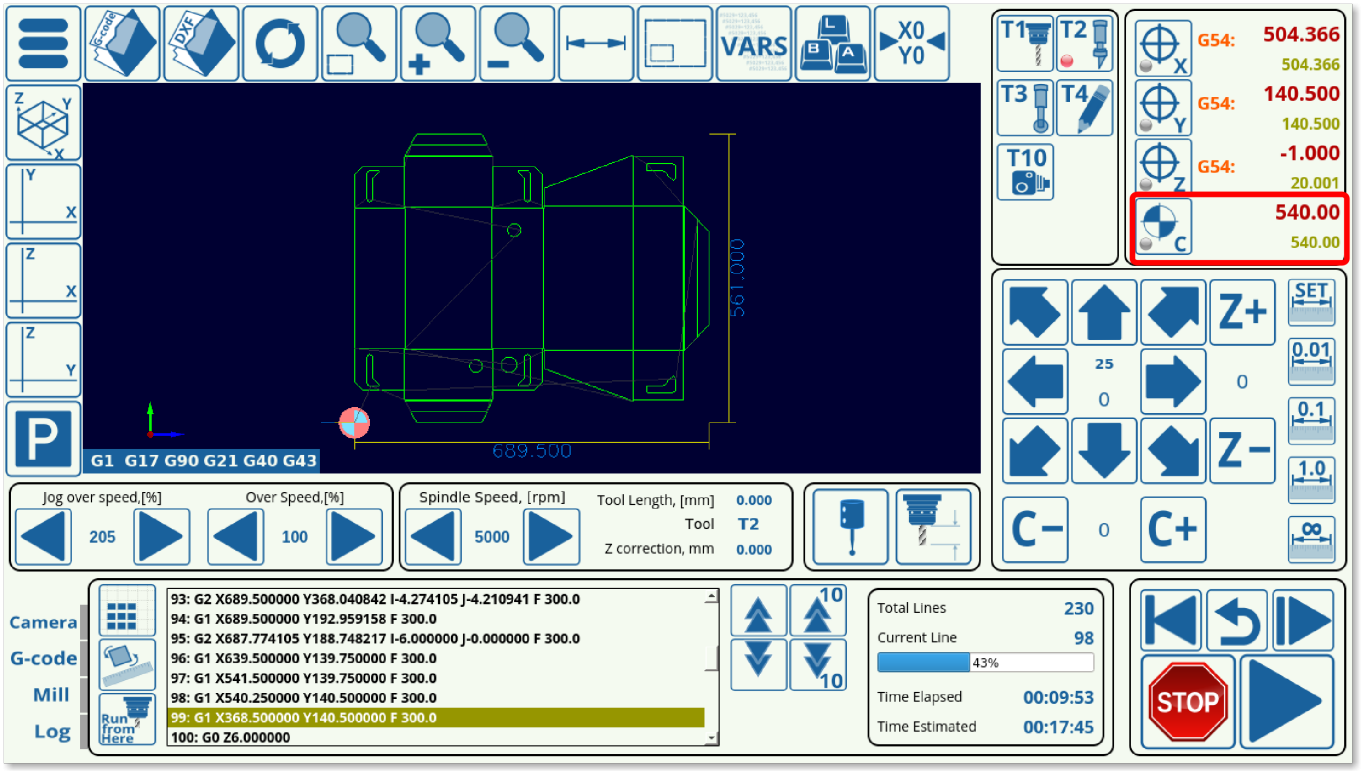
A tutorial on how to use global variable 7184 present in the code above is available on our YouTube channel:



Eliminating tangential knife spin at the start of the program (M212)



Because of how the system records angles, the software shows angles larger than 360 degrees (one full revolution) if a number of turns in the same direction have been taken by the knife. For example, if the knife has turned around its axis from 0 degrees twice in the positive direction, the angle now will be recorded as 720 degrees (2 full revolutions). After the program completes, and the angle is left at this number, the next time the program starts, the knife will rotate back until the angle is equal to zero. This behaviour is not ideal for some users, as it can extend the cutting process time.



The M212 PLC exists to remove this positive/negative degree turn that is larger than 360 degrees at the program start. This is useful if the user wants to stop the knife from spinning back multiple times to its 0 position on the c-axis as the program is starting (however, this will still allow the knife to rotate an angle less than a full revolution in order to align itself properly).

This PLC is provided with the myCNC software, and looks as follows:

```
main()
{
 c=gvarget(17006); //get C-position in PLC units (0.01 degree)

 m=18000; //180 degree in PLC units (0.01 degree)
 if (c>m)
 {
  do{ c=c-36000; }while (c>m); //remove the whole positive turns
 };

 m=0-m; //-180 degree in PLC units (0.01 degree)
 if (c<m)
 {
   do{ c=c+36000; }while (c<m); //remove the whole negative turns
 };

 gvarset(7080,1000); //set speed 3000 degree/s;
 timer=10;do{timer--;}while(timer>0);
 g0moveA(0x0,0x20,0-c); //C axis, move to C=0
 timer=200;do{timer--;}while(timer>0);
 do { code=gvarget(6060); }while(code!=0x4d);//wait till motion finished

 timer=100;do{timer--;}while(timer>0); //delay for any case
```

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

This PLC can be added to the DXF footer in **Settings > Config > DXF Import Settings** to run every time when the program generated from an imported DXF file finishes running.

NOTE: Since M212 changes the current coordinate value, it is necessary to always have a positioning command of the format G0C_ after M212. If, for example, the coordinate is reset to 0 inside the PLC, then the next code after the PLC must be G90G0A0. This will result in no movement along the axis, but the coordinates will be synchronized.

When analyzing the program, by default the software does not know that the coordinate inside the PLC has changed, and thus the subsequent G0 command must be used to update it (G1/G2/G3 are NOT suitable for this task). However, the G0 code is only necessary if the M212 is embedded into a G-code program. If the M212 PLC is started independently (for example by clicking a the button which launches the M212 alone), then the system will automatically see that the coordinate has been changed.

Gantry Alignment Procedure (with Homing)

M132

```
G10 L80 P5521 Q1
G10 L80 P5525 Q1

M146 P0 L1028
M88 L0 P5(Soft stop when sensor triggered)
G91 G0 Y  -300.0000 F   600.00
G04 P0.1
M89 L1 P5(Quick stop when sensor triggered)
G91 G0 Y   300.0000 F    30.00
G04 P0.1
M135
```

M135

```
G10 L80 P5521 Q1
G10 L80 P5525 Q1

M146 P0 L1028

M144
G91 G0 Y100 F30
G04 P0.1

G90 G10 L70 P0 Y0
G04 P0.1
M145 P0 L1028
G90 G10 L193 P97 Q5531
```

```
debug #98
G90 G10 L192 P98 Q7525
debug #98
debug #97
G90 G10 L190 P97 Q98
debug #97

G90 G28.9 Y97 F200

M146 P0 L1028

G90 G10 L70 P0 Y0

G90 G10 L80 P5521 Q0
G90 G10 L80 P5525 Q0
G90 G10 L80 P7395 Q0 (Homing Flag)
```

M144.plc

```
//Look after input1 & input2 sensors, remember position, when triggered

main()
{

timer=0;

message=PLCCMD_MOTION_CONTINUE;
texit=timer+2;do{timer++;}while(timer<texit);

ready=0;

state1=0;
state2=0;

e9000=portget(13);//gvarget(9000);
e9001=portget(14);//gvarget(9001);

state0=0;

m1=0;
m2=0;

do
{
 timer++;

if (state0==0)
{
 a=portget(13);//gvarget(9000);
```

```
 if (a!=e9000)
 {
        m1=1;
        position1=gvarget(5021+1);    //Machine Y
state0=1;
 };
 a=portget(14);//gvarget(9100);
 if (a!=e9001)
 {
        m1=2;
        position1=gvarget(5021+1);    //Machine Y
state0=1;
 };
}else
{
if (m1==2)
{
 a=portget(13);//gvarget(9000);
 if (a!=e9000)
 {
        m2=1;
        position2=gvarget(5021+1);    //Machine Y
state0=2;
 };
}else
{
 a=portget(14);//gvarget(9100);
 if (a!=e9001)
 {
        m2=2;
        position2=gvarget(5021+1);    //Machine Y
state0=2;
 };
};
};


}while(state0<2);


b=position1-position2;



if (b>25000)
{
b=50000-b;
};
c=0-25000;
if (b<c)
{
```

```
b=50000+b;
};



gvarset(97,b);
texit=timer+30;do{timer++;}while(timer<texit);

gvarset(7230,m1);
if (m1==1) { gvarset(98,1);}
else { x=0-1; gvarset(98,x);};

message=PLCCMD_MOTION_SKIP;
//message=PLCCMD_MOTION_SOFT_SKIP;
texit=timer+2;do{timer++;}while(timer<texit);

exit(99);
};
```

M145.plc

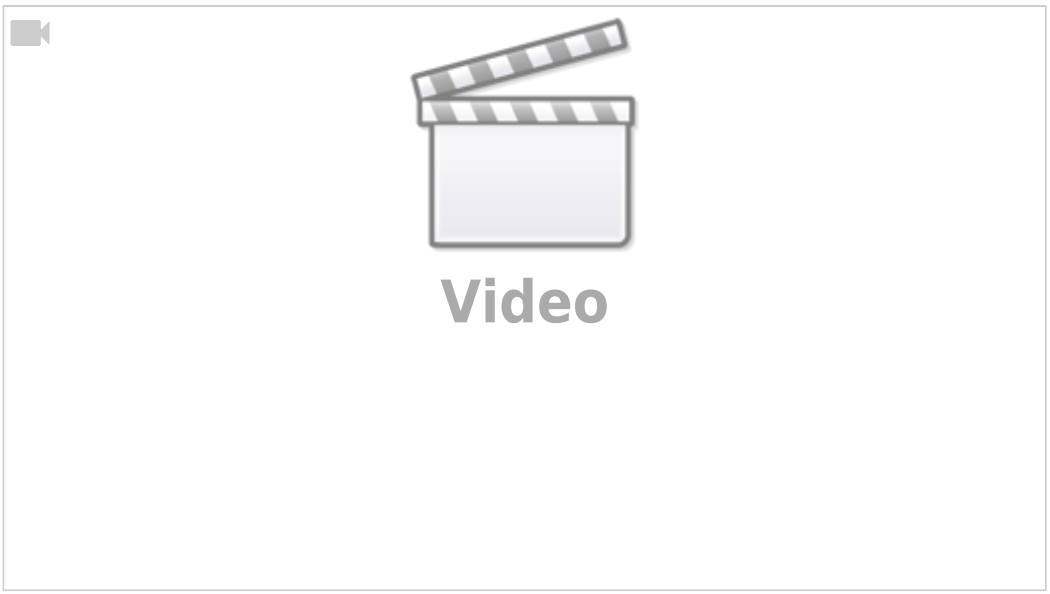
```
#define var_address var00
#define var_value   var01

main()
{
    timer=0;

    lparam=eparam>>16;

  axis=1;     //
    n=gvarget(7230);

    channel=0xff;
    if (n==1) {channel=0;};
    if (n==2) {channel=1;};
    if (n==4) {channel=2;};
    if (n==8) {channel=3;};

    if (channel>8)
    {
    message=PLCCMD_MOTION_ABORT;
    texit=timer+2;do{timer++;}while(timer<texit);
    exit(99);
    };


    var_value=15;
    var_address=112+channel;//channel turn off
    message=PLCCMD_SET_CNC_VAR;
```

```
    texit=timer+2;do{timer++;}while(timer<texit);

    exit(99);

};
```

M146.plc

```
#define var_address var00
#define var_value   var01

main()
{
    timer=0;

  dir=0;

  axis=1;
  channel=0;

    var_address=112+channel;//channel 0 set up
    var_value=axis;
    if (dir!=0) {  var_value=16+axis;  };
    message=PLCCMD_SET_CNC_VAR;
    texit=timer+10;do{timer++;}while(timer<texit);


  channel=1;

    var_address=112+channel;//channel 0 set up
    var_value=axis;
    if (dir!=0) {  var_value=16+axis;  };
    message=PLCCMD_SET_CNC_VAR;
    texit=timer+10;do{timer++;}while(timer<texit);

    gvarset(7230,1);


    exit(99);
};
```

Automatic plate rotation and offset adjustment




```
#include pins.h
#define INPUT_PROBE 7
#define ROLLBACK 2000
wait_move()
{
    do { code=gvarget(6060); }while(code!=0x4d);  //wait till motion
finished
};
show_error()
{
    gvarset(9121,1);
    timer=50;do{timer--;}while(timer>0);
    message=PLCCMD_MOTION_BREAK;  //1033
    exit(99);
};
find_start()
{
  gvarset(8631,50); //acceleration time 100ms = 0.1s
  gvarset(8634,(1<<24)|speed); //Jog speed for X
  gvarset(8634,(2<<24)|speed); //Jog speed for Y
  gvarset(8635,3); //X+ Y+
  timer=0;
  do
   {
      sensor0=portget(INPUT_PROBE);
      if ((timer&0xff)==0)   {     gvarset(8635,3);    };
      timer++;
    }while(sensor0==0);
  timer=2000;
  do   {      if ((timer&0xff)==0)   {     gvarset(8635,3);    };
timer--;    }while(timer>0);
   gvarset(8635,0); wait_move();
};
do_rollback_y()
{
```

```
  sensor0=portget(INPUT_PROBE);
  if (sensor0==0)
  {
    do{
    g0moveA(0,0x02,ROLLBACK); wait_move();
    sensor0=portget(INPUT_PROBE);
    }while(sensor0==0);
  };
};
do_rollback_x()
{
  gvarset(8632,speed);
  g0moveA(0,0x01,length+(length>>2));
  do
  {
    sensor0=portget(INPUT_PROBE);
    if (sensor0==0) { message=stop_code;  };
    code=gvarget(6060);
  }while(code!=0x4d);//wait till motion finished
};
find_edge()
{
  gvarset(8632,speed);
  sensor0=portget(INPUT_PROBE);
  len=length;
  if (sensor0!=0) {len=0-len;};
  g0moveA(0,axis,len);
  edge=100;
  do{
   code=gvarget(6060);
   if (code==0x4d)
    {
      edge=0;
    }else
    {
      sensor1=portget(INPUT_PROBE);
      if (sensor1!=sensor0)
       {
        edge=1;
        message=stop_code;
        do { code=gvarget(6060); }while(code!=0x4d);  //wait till motion
finished
       };
    };
  }while (edge==100);
};
move_x()
{
  gvarset(8632,speed_fast);
  g0moveA(0,0x1,0-length);
  do
```

```
  {
    sensor1=portget(INPUT_PROBE);
    if (sensor1==0)
    {
        edge=0;
        message=stop_code;
        wait_move();
    };
    code=gvarget(6060);
  }while(code!=0x4d);//wait till motion finished
};
move_y()
{
  gvarset(8632,speed_fast);
  g0moveA(0,0x2,length);
  do { code=gvarget(6060); }while(code!=0x4d);//wait till motion finished
};
move_y0()
{
  gvarset(8632,speed_fast);
  g0moveA(0,0x2,ROLLBACK);
  do { code=gvarget(6060); }while(code!=0x4d);//wait till motion finished
};
save_pos()
{
    if (edge==0)  {   show_error();    };
    //y_pos=gvarget(5042);
    //gvarset(400+point, gvarget(5041));  timer=30;do{timer--
;}while(timer>0); //
    //gvarset(401+point, y_pos);  timer=30;do{timer--;}while(timer>0); //
    //point+=2;
    gvarset(5730, 1);  timer=30;do{timer--;}while(timer>0); //
    gvarset(8632,speed_fast);  g0moveA(0,axis,ROLLBACK); wait_move();
};
main()
{
  gvarset(5740, 1);  timer=30;do{timer--;}while(timer>0); //
  portset(7);
  gvarset(300,300); timer=30;do{timer--;}while(timer>0); //
  gvarset(301,1500); timer=30;do{timer--;}while(timer>0); //
  point=0;
  gvarset(8631,50); //acceleration time 100ms = 0.1s
  gvarset(5539,1); //switch to fast g0moveA implementation
  edge=0;
  stop_code=PLCCMD_LINE_SOFT_STOP;//skip line
  length=6000;
  speed_probe=gvarget(300);
  speed_fast=gvarget(301);
  speed=speed_fast;
  find_start();
  axis=0x1; //X axis
```

```
  do_rollback_x();
  do
  {
    axis=0x2; //Y axis
    speed=speed_fast;
    do_rollback_y();
    do{  find_edge(); }while((edge==0)&(sensor1!=0));
    if (edge!=0)
    {
      speed=speed_probe;
      find_edge();
      save_pos();
      move_x();
    };
  }while(edge!=0);
    axis=0x1; //X axis
    speed=speed_fast;
    find_edge();
    if (edge==0)  {   show_error();    };
    speed=speed_probe;
    find_edge();
    save_pos();
    move_y();
    speed=speed_fast;
    find_edge();
    if (edge==0)  {   show_error();    };
    speed=speed_probe;
    find_edge();
    if (edge==0)  {   show_error();    };
    save_pos();
    gvarset(5740, 200);  timer=30;do{timer--;}while(timer>0); //
    exit(99);
};
```

Simultaneous homing for two axes

The code below allows simultaneous homing for both the X and the Y axes. The example is using a lot of the functionality described in the Jog from PLC section of the PLC manual (PLC. Please refer to the full PLC manual for further explanation on the jog from PLC functionality and its associated global variables.

```
#include pins.h

wait_move()
{
    do { code=gvarget(6060); }while(code!=0x4d);  //wait till motion
finished
};

show_error()
```

```
{
    gvarset(9121,1); //bring up popup message 21
    timer=50;do{timer--;}while(timer>0);
    message=PLCCMD_MOTION_BREAK;  //1033
    exit(99);
};

find_home_xy()
{

  gvarset(5521,1); //disable hardware limits
  gvarset(5525,1); //disable software limits

  gvarset(8631,50); //acceleration time 100ms = 0.05s

  statex=0;
  statey=0;
  speed=500;
  speed_slow=50;

  direction=(1<<8)+(2<<8); //set the direction variable to X- Y-

  gvarset(8634,(1<<24)|speed); //Jog speed for X
  gvarset(8634,(2<<24)|speed); //Jog speed for Y

  gvarset(8635,direction); //jog in the set direction (X- Y-)
  timer=0;
  do
   {
      changed=0;
      if (statex==0) //to home X
       {
          sens=portget(INPUT_HOME_X); //get state of home x sensor
          if (sens!=0)
          {
            statex=1;
            gvarset(8634,(1<<24)|speed_slow); //Jog speed for X
            changed=1;
          };
       };
      if (statex==1) //rollback from home X
       {
          sens=portget(INPUT_HOME_X);
          if (sens==0)
          {
            statex=2;
            changed=1;
          };
       };

      if (statey==0) //to home Y
```

```
       {
          sens=portget(INPUT_HOME_Y); //get state of home y sensor
          if (sens!=0)
          {
            statey=1;
            gvarset(8634,(2<<24)|speed_slow); //Jog speed for Y
            changed=1;
          };
       };
      if (statey==1) //rollback from home X
       {
          sens=portget(INPUT_HOME_Y); //get state of home y sensor
          if (sens==0)
          {
            statey=2;
            changed=1;
          };
       };

  if (changed!=0) //if any of the sensors state is changed
  {
    direction=0; //set direction to 0 (no movement) before flipping
    if (statex==0) { direction=direction | (1<<8);  };  //direction set to
X-
    if (statex==1) { direction=direction | 1;  }; //direction set to X+
    if (statey==0) { direction=direction |  (2<<8);  };  //direction set to
Y-
    if (statey==1) { direction=direction |  2;  }; //direction set to Y+
    gvarset(8635,direction); //jog in new direction for the axes
  };



      if ((timer&0xff)==0)   {     gvarset(8635,direction);    };
      timer++;
      ready=(statex==2)&(statey==2);

    }while(ready==0);


   gvarset(8635,0); wait_move(); //stop jog

};

main()
{

  gvarset(8631,50); //acceleration time 50ms = 0.05s
  gvarset(5539,1); //switch to fast g0moveA implementation

  find_home_xy();
```

```
  exit(99);  //normal exit
};
```

Mouse Jog (M320)

To enable mouse jog, we need to set global variable #7085 to “1”. Afterwards, a SHIFT + Left Mouse Click (hold for ~ 1 second) will activate the jog and the machine should start to move to the selected point.

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

//Move to Mouse Jog position
//Position is 7991, 7992

wait_motion_end()
{
  timer=2; do{timer--;}while (timer>0);  //wait motion started
  do
   {
      ex=0; code=gvarget(6060);
      if (code==0x4d) {ex=1;};
      if (code==0x57) {ex=1;};
    } while(ex==0);
};

main()
{
    x=gvarget(7991); //get X coordinate
    y=gvarget(7992); //get Y coordinate
    gvarset(5539,1); //set new model of PLC move API
    gvarset(8631,100); //Set PLC acceleration time in ms
    speed=gvarget(7041); //get rapid-speed-x
    gvarset(8632,speed); //Set PLC speed in units/min

    g0moveA(0x81,1,x); //Set X (machine, delayed start)
    g0moveA(0x81,2,y); //Set Y (machine, delayed start)
    g0moveA(0x41,0x3,0); //Start move to Parking

    wait_motion_end();

    exit(99);
};
```