2019

Upgrading the ISEL EP1090

REPLACING THE ORIGINAL CONTROLLS WITH GRBL

SEBASTIAN SCHÜTZ

Content

1. Introduction 2

2. Plan 2

3. Reverse engineering the circuit board 2

4. The actual conversion 4

5. Connect to the ISEL 5

6. Settings for grbl 7

1. Introduction

For quite some time now, we had an ISEL EP1090 standing at home, unused. Having a CNC-machine this close opens many possibilities, but in the current state of the machine, it´s possibilities were limited, due to:

- It is just 2.5D (Not bad, but real 3D would have been nice)
- Acceleration and deceleration were catastrophic. Although I can´t say it for sure, it sounded like every movement started with maximum speed and ended with an abrupt halt.
- No probing. Every time I used the ISEL I used a precision paper in order to lover the drill bit onto the workpiece, till I could feel a light drag on the paper
- Finally: The control software. It was ridiculously hard finding a piece of software, which could communicate with the ISEL[1] and the ones I found were pretty old and outdated.

So, it was time for an upgrade, because the mechanics of this machine are great.

2. Plan

The original plan was to swap the main processor with an Arduino UNO running grbl. This turned out to be a bad idea. I managed to cram a bareduino into the ISEL but had tons of problems with electrical interference. So, I decided to route the signal cables (for STEP, DIR and the endstops) outside and only let the power electronics, responsible for controlling the motors in the machine base. But first I had to do some digging.

3. Reverse engineering the circuit board

Figure 1 shows the circuit board of the ISEL. In the red circle you can see the socket of the old processor. I removed it, so that it doesn't interfere with our new brain. The unpopulated socket next to it (green circle) is home to the MC74HC245A. This is a transceiver chip, used to pass the signals from the processor to the driver IC of the stepper motor control. We will use this socket to communicate with the stepper motor control und repurpose the original chip (Although it should also work without it).

Additional info:

- Blue circle: The L297 stepper controller. This chip converts the DIR and STEP signals into signals usable by a motor. It supports Full and Halfstep mode.
- Black circle: The L298N motor driver. For each motor two chips are used in order to increase the power
- Yellow circle: I know, it is a little bit hard to see, but this is the place where the connectors for the endstops originally were plugged in.

[1] Due to a propriatery softwareprotocoll used by the ISEL. If they had just stick to g-code….

- The molex connectors are running from the board to the motors. It starts with the motor for the z-axis at the top. The middle connector is for Y (left, right) and the one on the bottom for the x-axis (front, back)

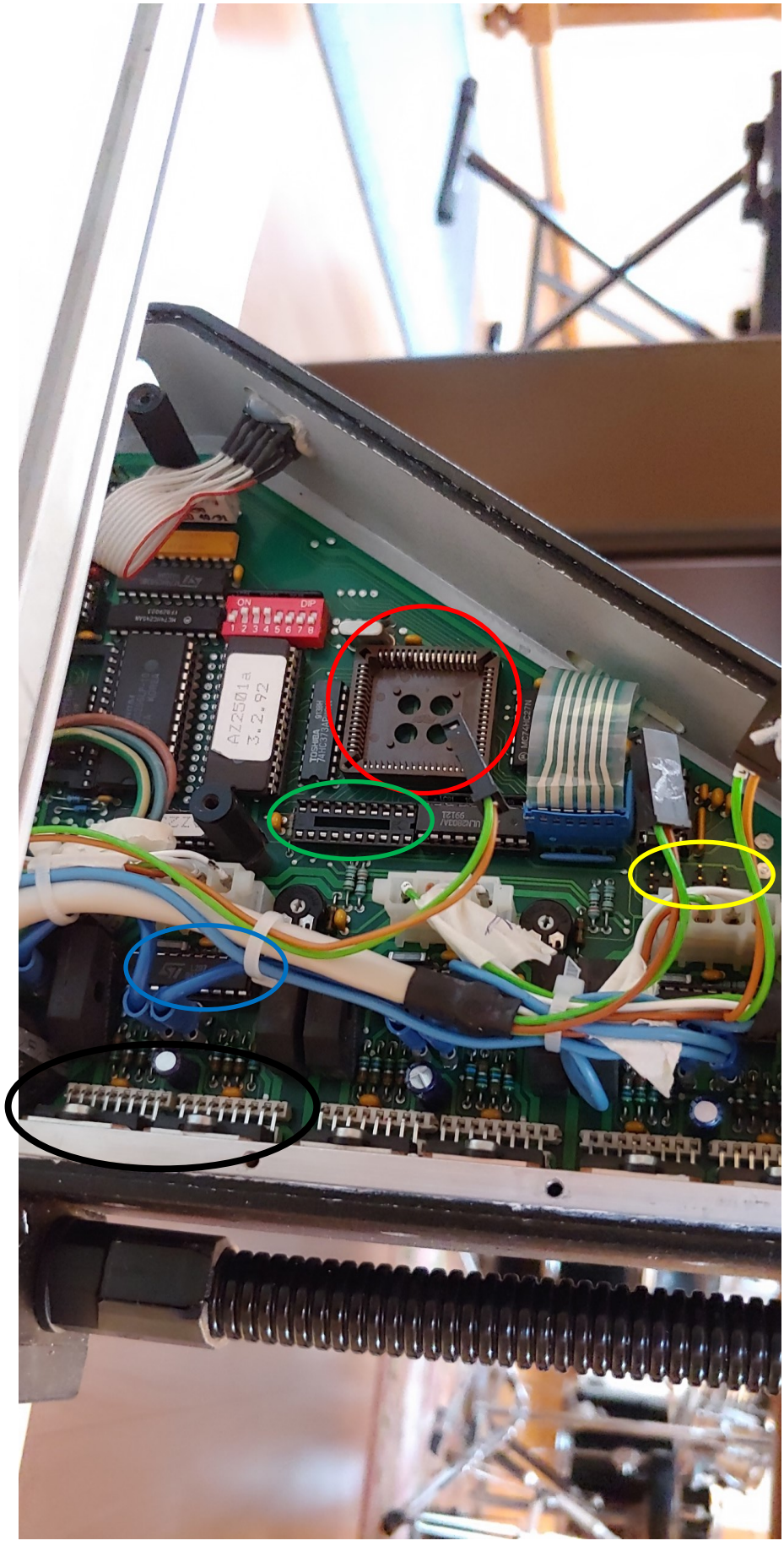

Figure 1: Mainboard of the ISEL EP1090

4. The actual conversion

The new brain of the ISEL will be an Arduino Nano, sitting on a PCB by the Arduinoclub[2]. This PCB does all the work of isolating the inputs and outputs and routing all the pins to screw terminals for easy access.

The PCB is mounted in a control box, wiring is done according to the following plan: https://www.arduinoclub.de/wp-content/uploads/2019/01/EN_AC-CNC2018-N-GRBL_R2.x_Schema.pdf

There are only a few things to be noted, which I changed:

- I added a filter in front of the 12V power supply, which is sitting on the rail
- On the output side of the solid state relay a varistor was added, following the connection diagram in the datasheet of the relay. (red circle)
- The relay is used to switch the socket located at the front of the control box, which is used to power up the spindle. Always have an electrician do the high voltage work!!!
- I used a CAT 7 LAN cable to bring the signals from the PCB to the inside of the ISEL, because it has good shielding. Due to the signals being digital, the shielding is grounded on both sides of the cable.

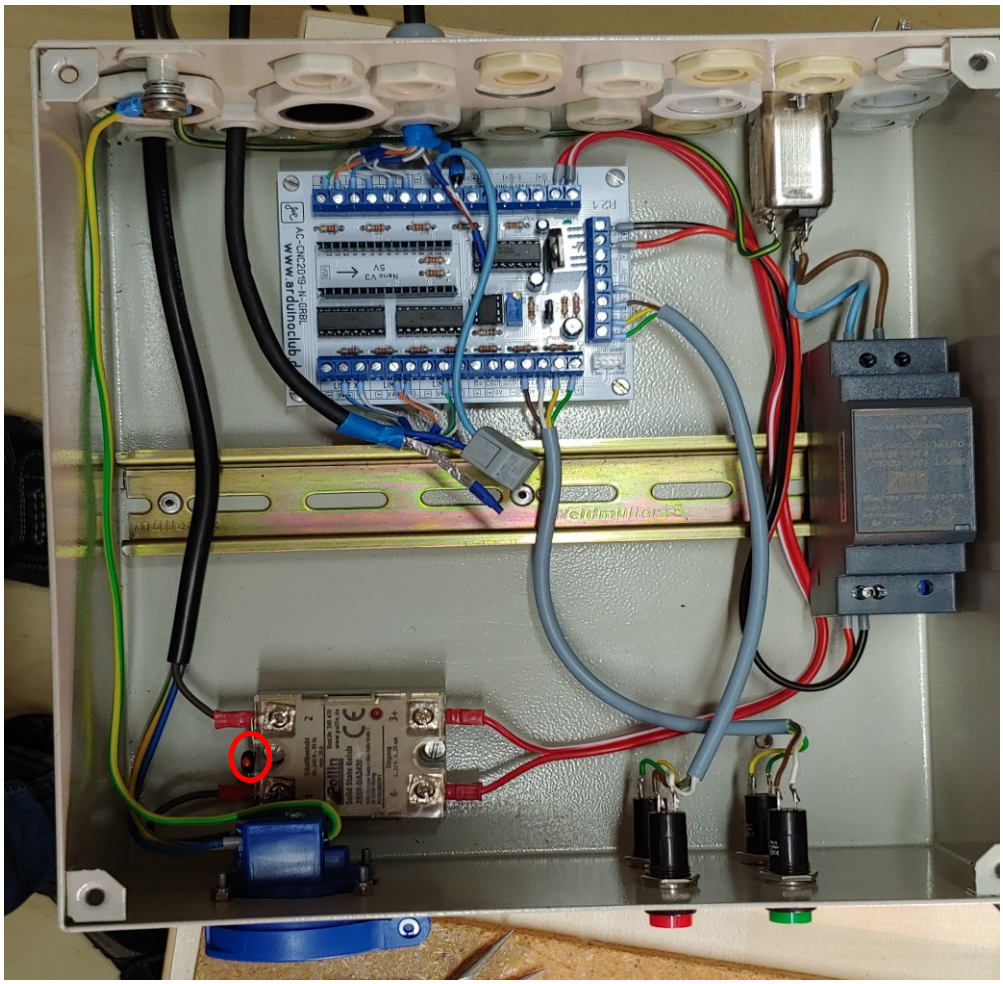

Figure 3: The new cable cabinet. In red the varistor

[2] https://www.arduinoclub.de/2019/01/15/cnc-arduino-nano-grbl-1-1-en/

I run the two CAT7 LAN cables through the old serial port, in order to avoid drilling holes into the body of the ISEL. I 3D printed a small adapter, which can be screwed in place. This is the red piece[3]. On the inside, one cable tie on each cable makes sure, that the cable won´t slip out. If you want to replicate this, remember to pull the cables through the red adapter bevor you start to solder pins to the cables.

Figure 4: The new front view of the ISEL.

5. Connect to the ISEL

This took quite some time, till I found a solution, which I could easily reverse in case something went wrong. As mentioned in the beginning, we will be using the socket of the MC74HC245A in order to interfere with the power electronics. Luckily for us,

normal headers fit perfectly inside the socket, so I made the following board:

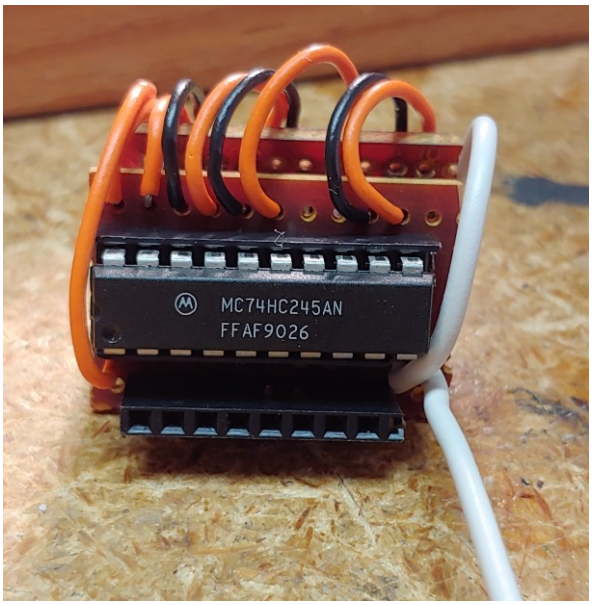

Figure 6: The upper board of the two connection boards

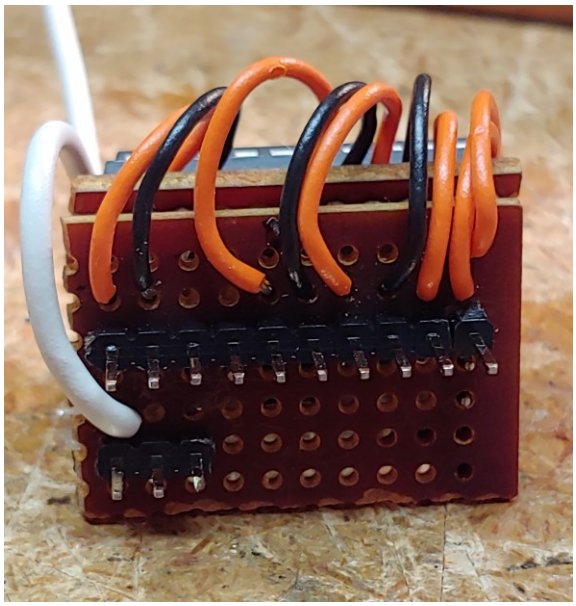
Figure 5: The lower board of the two connection boards

We have 2 PCBs connected back to back. On the underside we have the pins, which go into the socket, where the MC74HC245A once was located on the ISEL board. Till now I have had no problems with it coming lose. The top side reuses the MC74HC245A in order to bring the signals to the Stepper motor drivers. You don´t have to use this chip, because in theory the output of the Arduinoclub-PCB[2] should be sufficient, but I wanted to be safe. The MC74HC245A is wired in as in the following picture:

[3] You can download the file here: https://a360.co/2ZOcWAF Depending on your cable, you might have to drill out the two holes.

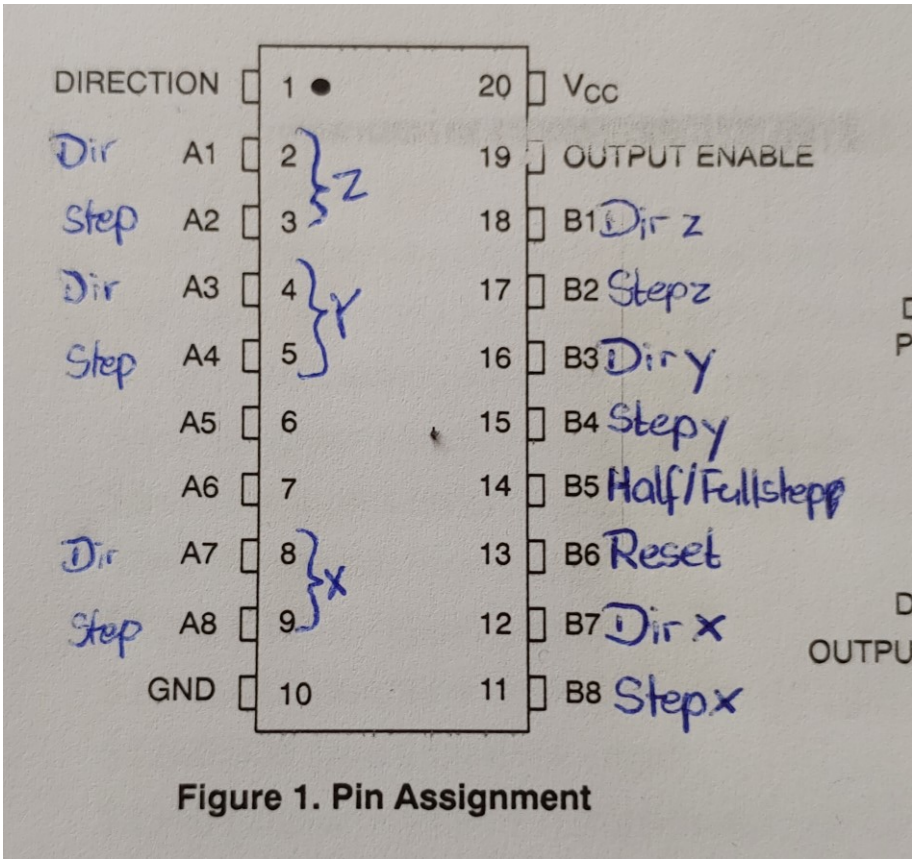

Figure 7: The pin Assignment of the MC74HC245[4]

Let's start with the left side:

- The direction pin is connected to V_{CC}. The MC74HC245A can be used in both directions and normally you could switch those. This isn´t needed here, so we keep it just the way we need it.
- After that the pins for the Z, Y and X axis are routed out to the header. As you can see A5 and A6 aren´t connected. I will explain why, when we come to the right side.
- GND is obvious. The only thing to note is the white cable running out in Figure 6. This is used as ground for the shielding of the cables running from the cable cabinet to the ISEL.

And the right side (output side):

- The OUTPUT ENABLE pin is also connected to V_{CC}. Otherwise the output wouldn´t work.
- Now we have the same configuration as on the left side. Just two pins are labelled here:
    - B5 - Half/Full step: The motor driver of the ISEL allows half and full step mode. Originally you could set this with a small DIP switch on the ISEL mainboard, but this switch is running to the processor and I didn´t wanted to make an Adapter for this one, so I decided, that I will be using half step mode as a good enough compromise between precision and performance.
    - B6 - Reset: This pin must be pulled to high, because otherwise the motordrivers won´t work

[4] Taken from the datasheet of the MC74HC245A

- Because I have no intention of changing the state of these to pins, I used two 13k pullup resistors on the back of the lower board. These two pins run to the corresponding pins on all three L297 stepper controller.

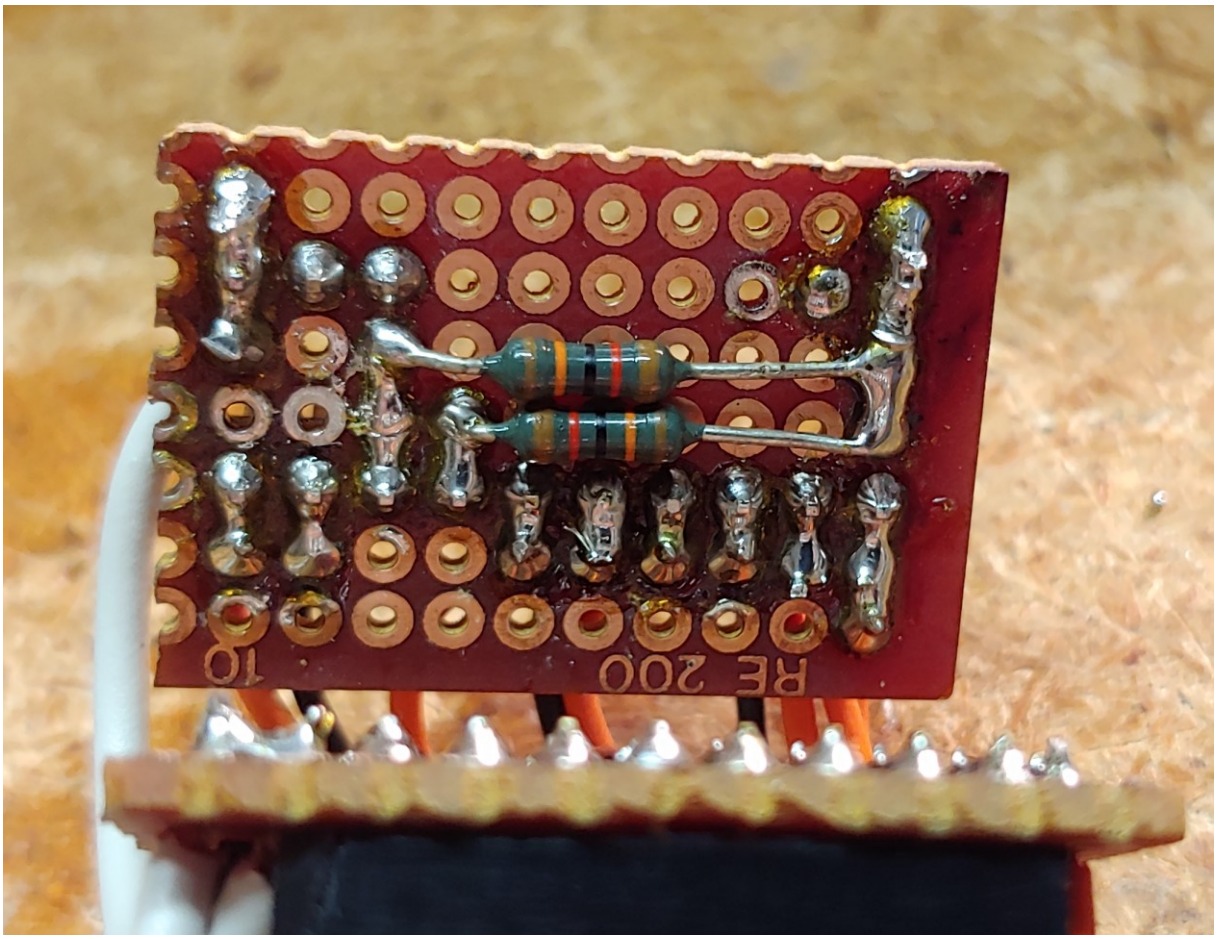

Figure 8: The underside of the lower connection board. Not a beauty, but it works

And remember to put a insulator between the two boards to prevent shorts.

6. Settings for grbl

Flashing grbl on to the Arduino Nano is pretty easy and you can follow these instructions: https://www.arduinoclub.de/wp-content/uploads/2019/01/DE_Manual_AC-CNC2019-N-GRBL_R2.0_install_GRBL1.1.docx.pdf

You just have to uncomment the following line: #define USE_SPINDLE_DIR_AS_ENABLE_PIN

If you need additional help, the grbl wiki is a great place to start.

For my ISEL used the following settings:

$0=10 (unchanged – Length of step impulse)
$1=25 (unimportant)
$2=7 (Invert stepsignal on all axis. This is because of the motor drivers in the ISEL. Normally the pin is high and an incoming Puls/Step is low)

$3=7 (Invert all axis)
$4=0 (Stepper enable invert – doesn´t matter, because not used)
$5=1 (Limit pins invert – because oft he NC-switches)
$6=0 (Probe pin invert – not used till now)
$10=1 (Status report – standard value)
$11=0.010 (Junction deviation - standard value)
$12=0.002 (Arc tolerance - standard value)
$13=0 (Report inches – set to mm. What else?)
$20=1 (soft limits – activated, because the ISEL only has limit switches on one side)
$21=1 (Hard limits – activated, because otherwise we wouldn´t be using the limit switches)
$22=1 (Homing – start with $H)
$23=0 (Homing dir invert)
$24=25.000 (Homing feed - standard value)
$25=500.000 (Homing seek – standard value)
$26=250 (homing debounce - standard value)
$27=1.000 (homing pull off – standard value – 1mm away from the limit switch)
$30=1000 (Max spindle speed - standard value)
$31=0 (Min spindle speed - standard value)
$32=0 (Laser mode – No, because we are running a milling machine)
$100=100.000 (Steps per mm [x]. Because we used the half steps [Microstepping = 2])
$101=100.000 (Steps per mm [y]. Because we used the half steps [Microstepping = 2])
$102=100.000 (Steps per mm [z]. Because we used the half steps [Microstepping = 2])
$110=2400.000 (Max rate [x] – The ISEL could go faster, but I keep it safe. Maximum: 3600 mm/min)
$111=2400.000 (Max rate [y] – The ISEL could go faster, but I keep it safe. Maximum: 3600 mm/min)
$112=2400.000 (Max rate [z] – The ISEL could go faster, but I keep it safe. Maximum: 3600 mm/min)
$120=10.000 (Acceleration in mm/sec² [x])
$121=10.000 (Acceleration in mm/sec² [y])
$122=10.000 (Acceleration in mm/sec² [z])
$130=230.000 (Max travel. Needed for soft limits [x])
$131=190.000 (Max travel. Needed for soft limits [y])
$132=110.000 (Max travel. Needed for soft limits [z])

With this setup the ISEL runs quite well. It is important to know, that negative movement will be moving the axis away from the limit switches/home position.