

## Function description for CNC mill, V1.00

2021-12-24, Armin Rehberger

## Table of content

General description.....	2
Used hardware with tasks.....	3
Used software.....	3
Principle structure.....	3
Softwarestructure.....	4
Raspberry Pi4, task GUI (execute event triggered).....	4
Raspberry Pi4, task Mill (cyclic task).....	4
Raspberry Pi4, programstructure Mill.....	5
Teensy4.0, programstructure.....	6
Teensy4.0 programstructure handler.....	7
Used libarys.....	8
Data shared memory Pi4 tasks GUI <-> Mill.....	9
Data Raspberry → Teensy, I2C.....	11
Data Teensy → Raspberry, I2C.....	12
Input / Output assignment Teensy 4.0.....	13
I2C busconfiguration and wiring.....	14
Teensy, settings mill.....	15
Timer interrupt in us.....	15
Hardware mechanic.....	15
Speeds axis X,Y,Z.....	15
Speed calculation spreadsheet.....	16
Explanation spreadsheet speed calculation.....	17
Speed calculation, function „SpeedMM_Min_ProgramScans“.....	17
Amount of steps start - stopramp calculation, function „Y = m*x+b“.....	17
Function start – stopramp spreadsheet.....	18
Explanation spreadsheet function start - stopramp.....	18
Calculation linear movement.....	19
Linear movement spreadsheet.....	19
Calculation circular movement.....	20
Circular movement spreadsheet.....	21
Abort movement.....	22
Abort inching.....	22
Abort measure tool length.....	22
Abort automatic mode.....	22
Layout circuit board Teensy 4.0.....	23
Pictures.....	28

# General description

Implementation of a mechatronic CNC mill.

Raspberry PI is used to operate the mill, a graphical GUI is implemented.

Teensy4.0 is used to control the real time CNC machine.

A G-Code file can be loaded with the GUI, the file will be readed and then transferred to Teensy4.0 (max. 7500 steps). After loading the file, the CNC-program can be executed.

In manual mode of the CNC machine you can do following:

Inche the axis (X, Y, Z) forwards and backwards.

Four digital outputs (four relay outputs) for spindle, coolant, clamp and light can be forced.

The 0-10V PWM signal for the spindle speed can be forced.

The tool length can be automatically measured with a tool length sensor.

The state of the digital inputs are displayed (Reference sensors X, Y, Z, emergency stop and tool length sensor).

Four bytes, Hexadecimal, are used to transfer the actual positions of the axis from the Teensi to the Pi (0.0000 to 9999.9999 mm).

# Used hardware with tasks

## Raspberry Pi4

### Task name Usage

GUI Graphic user interface  
Mill Data exchange GUI <-> Mill <-> Teensy

## Teensy 4.0

### Task name Usage

loop Main program  
handler Timer interrupt. Realtime system to control the CNC machine

# Used software

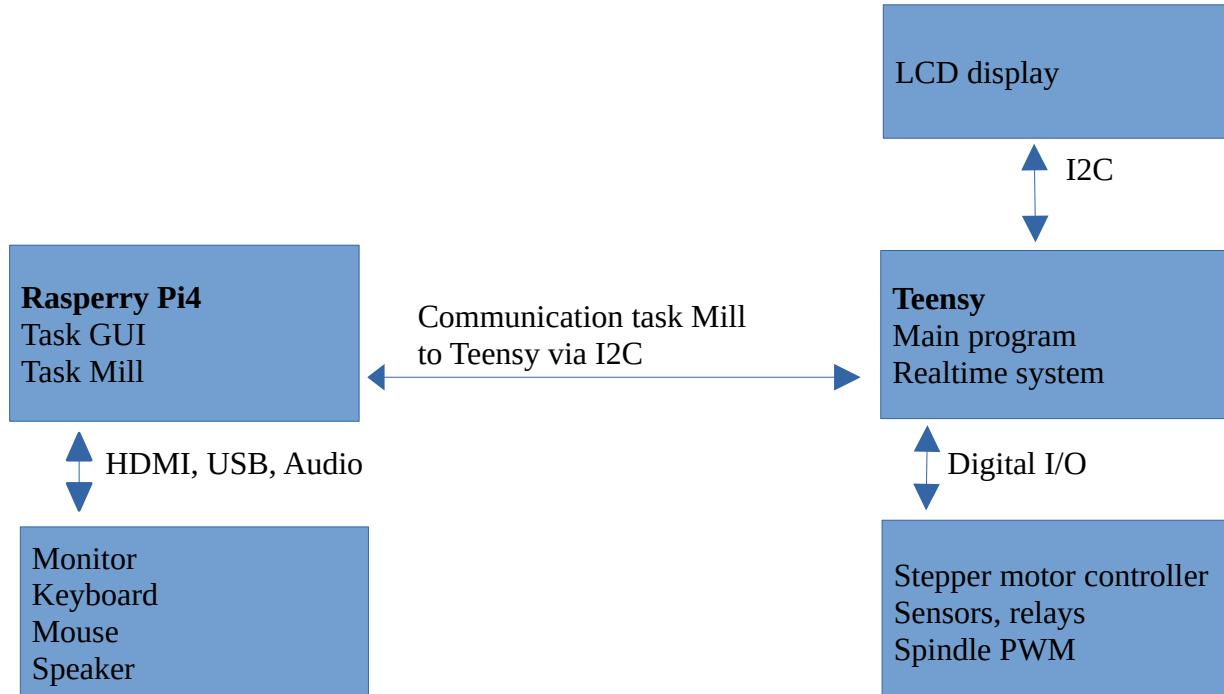
## Raspberry Pi4

Spyder3 V3.3.3  
Python V3.7

## Teensy 4.0

Arduino 1.8.15  
Teensy Loader 1.54

# Principle structure



# Softwarestructure

## Raspberry Pi4, task GUI (execute event triggered)

PythonProgramGUIPortalfraese\_V1\_00.py

This is the main task.

Class APPGUI, initialize

Function poll100, zyclic call every 100ms

Function OnResizemainWindow, place the main frame

Function showFrame, places the visible frame with his objects

Function hideFrame, hide the specific frame

Function OpenFileDialog, function for the OpenFileDialog, reads the filename

Function ButtonStartPressed, transfers the start command to Spider3 Mill

Function ButtonManualPressed, process of all the commands in frame manual

Function ButtonDiagnosticPressed, process of all the commands in frame diagnostic

Function SliderSpindleSpeedChange, function to transfer the manual spindle speed

Function Quit, quit program

Function Main, Starts multitask mill multiprocessing with data exchange to mill (shared memory),  
mainloop tkinter for GUI

## Raspberry Pi4, task Mill (cyclic task)

PythonProgrammPortalfraese\_V1\_00.py

Function evaluate\_line, reads a line from the G-Code file

Function read\_file, opens the G-Code file and calls the function evaluate\_line for each line

Function read\_bit\_motor, reads a byte and returns 8 bits

Function hexadecimalToDecimal, reads a hexadecimal-value and returns a decimal-value

Function decToHexToDec, converts a decimal value to 4 bytes decimal

Function main, multitask started from GUI, with loop forever (while True:)

    Read shared memory from GUI

    Read data from Teensy, I2C

    Case execution: Commands to Teensy, feedbacks from Teensy

    Write data to Teensy, I2C

    Write shared memory to GUI

## Raspberry Pi4, programstructure Mill

Read data GUI -> Mill, Shared memory

Read data Teensy, I2C

CaseIdle	Idle
CaseTransferFilename	Transfer filename
CaseReadFile	Read file
CaseTransferFile	Transfer stepdata to teensy
CaseMotHandshakeCommandDone	Handshake command done
CaseMotOperateOn	Operate on
CaseMotOperateOff	Operate off, go back to idle
CaseMotInching	Inching
CaseMeasureToolLength	Measure tool length
Write data Teensy, I2C	
Write data Mill -> GUI, Shared memory	

## **Teensy4.0, programstructure**

Function SendDataI2C, event triggered to send data to Raspberry Mill

Function ReciveDataI2C, event triggered to read data from Raspberry Mill

Function Step, toggl output step for X, Y anz Z axis

Function  $y=m*x+b$ , linear scaling function

Function SpeedMM\_Min\_ProgramScans, calculate speed from mm/min -> ProgramScans

Function Positioning, positioning sequence for axis X, Y and Z

Setup, initialize pin mode Teensy, starts timer interrupt, configures I2C slave for raspberry, initializes I2C master for LCD display

loop, transfers string to LCD display, toggels LED on Teensy board

handler, cyclic real time timer interrupt. executes all CNC commands

## Teensy4.0 programstructure handler

Copy I2C receive data, to achieve consistent data from I2C

Communication to Raspberry, fill StepData array

Read digital inputs

Handle immediate stop

```
switch (MotorAll.StepNo)
    case StepNoAction           Load next command
        switch (ActualStepData.Code)
            case CodeG00/G01:      Initialize G00/G01
            case CodeG02/G03:      Initialize G02/G03
            case CodeInching:     Initialize inching
            case CodeMeasureToolLength: Initialize measure tool length
            case CodeG04:          Initialize G04
            case CodeG52:          Execute G52
            case CodeG54:          Execute G54
            case CodeG74:          Initialize G74
            case CodeG90:          Execute G90
            case CodeG91:          Execute G91
            case CodeM03:          Execute M03
            case CodeM05:          Execute M05
            case CodeM08:          Execute M08
            case CodeM09:          Execute M09
            case CodeM10:          Execute M10
            case CodeM11:          Execute M11
            case CodeM35:          Execute M35
            case CodeM36:          Execute M36
            case CodeM02/M30:       Execute M02/M30
            case StepPositioningMain: Execute G74, calls function Positioning
            case StepLinearMovement: Execute G00/G01
            case StepCircularMovement: Execute G02/G03
            case StepInching:       Execute inching
            case StepMeasureToolLength: Execute measure tool length
            case StepDwellTime:     Execute G04
            case StepHandshakeCommandDone: Command done, goto StepNoAction
```

Transfer digital outputs (spindle, coolant, clamp, light) and status to Mill

Switch enable, transfer output enable

Transfer digital outputs motor X, Y and Z

Transfer digital output PWM spindle speed

Copy I2C send data, to achieve consistent data to I2C

# Used libarys

## Spyder3

time  
datetime  
tkinter for GUI  
multiprocessing for communication GUI <-> Mill  
smbus for I2C communication Mill <-> Teensy

## Arduino

[https://github.com/Richard-Gemmell/teensy4\\_i2c](https://github.com/Richard-Gemmell/teensy4_i2c)  
i2c\_driver.h  
i2c\_driver\_wire.h

<https://www.arduinolibraries.info/libraries/liquid-crystal-i2-c>  
LiquidCrystal\_I2C.h

Required changes in LiquidCrystal:

In file: LiquidCrystal\_I2C.H --> #include <Wire.h> replaced to #include <i2c\_driver\_wire.h>  
LiquidCrystal\_I2C.CPP --> #include <Wire.h> replaced to #include <i2c\_driver\_wire.h>

LiquidCrystal\_I2C.H

```
//YWROBOT
#ifndef LiquidCrystal_I2C_h
#define LiquidCrystal_I2C_h

#include <inttypes.h>
#include "Print.h"
//#include <Wire.h> // 2020-12.02, because of Teensy I2C
#include <i2c_driver_wire.h>
```

LiquidCrystal\_I2C.CPP

```
#endif
//#include "Wire.h" // 2020-12.02, because of Teensy I2C
#include "i2c_driver_wire.h"
```

# Data shared memory Pi4 tasks GUI <-> Mill

## ##### GUI <-> Mill, Shared memory, overview

ComGUIArrayInt = multiprocessing.Array('i', range(32))  
ComGUIArrayIntStringActualStep = multiprocessing.Array('i', range(100))  
ComGUIArrayIntStringActualFile = multiprocessing.Array('i', range(256))

## ##### ComGUIArrayInt

ComGUIArrayInt[0] GUI -> Mill Command. 1=read file, 2=start program. Mill -> GUI 0 = command read

ComGUIArrayInt[1] GUI -> Mill ManualCommands. Bit 0 = Manual command Spindle on  
Bit 1 = Manual command Coolant on  
Bit 2 = Manual command Clamp on  
Bit 3 = Manual command Light on  
Bit 4 = Manual command X+  
Bit 5 = Manual command X-  
Bit 6 = Manual command Y+  
Bit 7 = Manual command Y-  
Bit 8 = Manual command Z+  
Bit 9 = Manual command Z-  
Bit 10 = ClearDiagnosticCounterI2C  
Bit 11 = Manual mode selected  
Bit 12 = Measure tool length

ComGUIArrayInt[2] GUI -> Mill Spindle speed in manual mode

ComGUIArrayInt[3]

ComGUIArrayInt[4]

ComGUIArrayInt[5]

ComGUIArrayInt[6]

ComGUIArrayInt[7]

ComGUIArrayInt[8] Mill -> GUI Actual line number in file

ComGUIArrayInt[9] Mill -> GUI State. 0=Idle, 1=Program active

ComGUIArrayInt[10] Mill -> GUI Digital states. Bit 0 = Reference sensor X  
Bit 1 = Reference sensor Y  
Bit 2 = Reference sensor Z  
Bit 3 = Emergency stop button  
Bit 4 = Spindle on  
Bit 5 = Coolant on  
Bit 6 = Clamp on  
Bit 7 = Light on  
Bit 8 = Tool length sensor  
Bit 9 = Measure tool length done

ComGUIArrayInt[11] Mill -> GUI Diagnostic I2C Recive good

ComGUIArrayInt[12] Mill -> GUI Diagnostic I2C Recive checksum error

ComGUIArrayInt[13] Mill -> GUI Diagnostic I2C Recive hardware error

ComGUIArrayInt[14] Mill -> GUI Diagnostic I2C Send good

ComGUIArrayInt[15] Mill -> GUI Diagnostic I2C Send hardware error

ComGUIArrayInt[16] Mill -> GUI actual position X HighHigh

ComGUIArrayInt[17] Mill -> GUI actual position X High

ComGUIArrayInt[18] Mill -> GUI actual position X Low

ComGUIArrayInt[19] Mill -> GUI actual position X LowLow  
ComGUIArrayInt[20] Mill -> GUI actual position Y HighHigh  
ComGUIArrayInt[21] Mill -> GUI actual position Y High  
ComGUIArrayInt[22] Mill -> GUI actual position Y Low  
ComGUIArrayInt[23] Mill -> GUI actual position Y LowLow  
ComGUIArrayInt[24] Mill -> GUI actual position Z HighHigh  
ComGUIArrayInt[25] Mill -> GUI actual position Z High  
ComGUIArrayInt[26] Mill -> GUI actual position Z Low  
ComGUIArrayInt[27] Mill -> GUI actual position Z LowLow  
ComGUIArrayInt[28] Mill -> GUI actual position Reserve  
ComGUIArrayInt[29] Mill -> GUI actual position Reserve  
ComGUIArrayInt[30] Mill -> GUI actual position Reserve  
ComGUIArrayInt[31] Mill -> GUI actual position Reserve

##### **ComGUIArrayIntStringActualStep** Mill -> GUI  
Value 0 = Stringlen  
Value 1..99 = Stringdata

##### **ComGUIArrayIntStringActualFile** GUI -> Mill  
Value 0 = Stringlen  
Value 1..255 = Stringdata

# Data Raspberry → Teensy, I2C

##### I2C Master --> Slave, Struct I2C recive, max. 32 Byte possible  
Raspberry → Teensy, Byte 0..27, 28 Byte

Byte 0

Bit 0 = Operate (0 - Stepper disabled 1 - Stepper enabled)  
Bit 1 = Reserve 1  
Bit 2 = Reserve 2  
Bit 3 = Reserve 3  
Bit 4 = Reserve 4  
Bit 5 = Reserve 5  
Bit 6 = Reserve 6  
Bit 7 = Reserve 7

Byte 1

Bit 0 = ActivateCommand  
Bit 1 = Reserve 1  
Bit 2 = Reserve 2  
Bit 3 = Reserve 3  
Bit 4 = Reserve 4  
Bit 5 = Reserve 5  
Bit 6 = Reserve 6  
Bit 7 = Reserve 7

Byte 2

Bit 0 = Manual command Spindle on  
Bit 1 = Manual command Coolant on  
Bit 2 = Manual command Clamp on  
Bit 3 = Manual command Light on  
Bit 4 = Manual command X+  
Bit 5 = Manual command X-  
Bit 6 = Manual command Y+  
Bit 7 = Manual command Y-

Byte 3

Bit 0 = Manual command Z+  
Bit 1 = Manual command Z-  
Bit 2 = Clear diagnostic counter I2C  
Bit 3 = Measure tool length  
Bit 4 = Reserve 4  
Bit 5 = Reserve 5  
Bit 6 = Reserve 6  
Bit 7 = Reserve 7

Byte 4, 5 Spindle speed in manual mode

Byte 6 G Code / M Code

Byte 7, 8, 9, 10 Value 0 (with four decimal places)  
Byte 11, 12, 13, 14 Value 1 (with four decimal places)  
Byte 15, 16, 17, 18 Value 2 (with four decimal places)  
Byte 19, 20, 21, 22 Value 3 (with four decimal places)  
Byte 23, 24, 25, 26 Value 4 (with four decimal places)  
Byte 27 Checksum (XOR Byte 0..26)

## Data Teensy → Raspberry, I2C

##### Slave --> I2C Master, Struct I2C send, max. 32 Byte possible

Teensy → Raspberry, Byte 0..22, 23 Byte

Byte 0

- Bit 0 = OperateOn
- Bit 1 = Reserve 1
- Bit 2 = Reserve 2
- Bit 3 = Reserve 3
- Bit 4 = Reserve 4
- Bit 5 = Reserve 5
- Bit 6 = Reserve 6
- Bit 7 = Reserve 7

Byte 1

- Bit 0 = CommandDone
- Bit 1 = ReadyForCommand
- Bit 2 = ProgramDone
- Bit 3 = Reserve 3
- Bit 4 = Reserve 4
- Bit 5 = Reserve 5
- Bit 6 = Reserve 6
- Bit 7 = Reserve 7

Byte 2

- Bit 0 = Reference sensor X
- Bit 1 = Reference sensor Y
- Bit 2 = Reference sensor Z
- Bit 3 = Emergency stop button
- Bit 4 = Spindle on
- Bit 5 = Coolant on
- Bit 6 = Clamp on
- Bit 7 = Light on

Byte 3

- Bit 0 = Tool length sensor
- Bit 1 = Measure tool length done
- Bit 2 = Reserve 2
- Bit 3 = Reserve 3
- Bit 4 = Reserve 4
- Bit 5 = Reserve 5
- Bit 6 = Reserve 6
- Bit 7 = Reserve 7

Byte 4, 5, 6, 7 Actual position motor 0 (with four decimal places)

Byte 8, 9, 10, 11 Actual position motor 1 (with four decimal places)

Byte 12, 13, 14, 15 Actual position motor 2 (with four decimal places)

Byte 16, 17, 18, 19 Actual position motor 3 (with four decimal places)

Byte 20, 21 ActualStepProgram

Byte 22 Checksum (XOR Byte 0..21)

# Input / Output assignment Teensy 4.0

## Pin      Used for

13 LED buildin  
14 Ref. 1 Y-Axis (Input)  
15 Ref. 2 Z-Axis (Input)  
16 SCL1 Raspberry  
17 SDA1 Raspberry  
18 SDA0 Liquid Christal  
19 SCL0 Liquid Christal  
20 Ref. 0 X-Axis (Input)  
21 Emergency stop (Input Pullup)  
22 PWM Spindle Speed (Output)  
23 Tool length sensor (Input Pullup)  
3.3V     Output 250mA  
GND 0V Power supply 0V  
Vin 5V    Power supply 5V

12 Enable 0 (Output)  
11 Step 0 (Output)  
10 Dir 0 (Output)  
9 Enable 1 (Output)  
8 Step 1 (Output)  
7 Dir 1 (Output)  
6 Enable 2 (Output)  
5 Step 2 (Output)  
4 Dir 2 (Output)  
3 Spindle on (Output)  
2 Coolant on (Output)  
1 Clamp on (Output)  
0 Light on (Output)  
GND 0V (Not connected)

## Input Emergency stop and tool length sensor

These input are switches, therefore use Input\_Pullup.

If the buttons are not pressed, the switches are closed (normally closed, NC), the voltage at the input pin = 0.0V, the input is logically false.

If the buttons are pressed, the switches are open, the voltage at the input pin = 3.3V, the input is logically true.

## Inputs reference sensors axis X, Y and Z

NPN – inductive sensor

Input to Teensy via SFH 618A-3 VIS optocoupler

Sensor outputable to 24V via 4.7K resistor and to input optocoupler.

Output optocoupler via 665R resistor to 3.3V and to input Teensy.

# I2C busconfiguration and wiring

## Raspberry I2C master to Teensy:

Used bus 3

Edit config.txt

sudo nano /boot/config.txt

Available I2C buses: 1, 3, 4, 5, 6, 7

Don't use bus 0 and 2

Bus 1 = Pin 3 (GPIO2) SDA1

Pin 5 (GPIO3) SCL1

Bus 1 with internal pull up resistors, 3.3V

Bus 3 = Pin 11 (GPIO17) SDA

Pin 13 (GPIO27) SCL

Bus 3 without internal pull up resistors, 3.3V

dtoverlay=i2c-gpio,bus=3,i2c\_gpio\_delay\_us=2,i2c\_gpio\_sda=17,i2c\_gpio\_scl=27

Bus 4 = Pin 16 (GPIO23) SDA

Pin 18 (GPIO24) SCL

Bus 4 without internal pull up resistors, 3.3V

dtoverlay=i2c-gpio,bus=4,i2c\_gpio\_delay\_us=2,i2c\_gpio\_sda=23,i2c\_gpio\_scl=24

config.txt

Uncomment some or all of these to enable the optional hardware interfaces

dtparam=i2c\_arm=on

dtparam=i2c1=on

#dtparam=i2s=on

#dtparam=spi=on

dtoverlay=i2c-gpio,bus=3,i2c\_gpio\_delay\_us=2,i2c\_gpio\_sda=17,i2c\_gpio\_scl=27

dtoverlay=i2c-gpio,bus=4,i2c\_gpio\_delay\_us=2,i2c\_gpio\_sda=23,i2c\_gpio\_scl=24

## Teensy, I2C slave to Raspberry:

Teensy slave address 8

Teensy Pin 16 SCL1      Raspberry SCL GPIO 27, Pin 13

Teensy Pin 17 SDA1      Raspberry SDA GPIO 17, Pin 11

Pull up resistors 2K2 Ohm

## Teensy, I2C master to LCD display LiquidChristal:

Liquid Christal slave address 39

Teensy Pin 18 SDA0

Teensy Pin 19 SCL0

# **Teensy, settings mill**

## **Timer interrupt in us**

```
#define TimeInterrupt 15.5625
```

## **Hardware mechanic**

```
const double StepsPerTurn = 400.0;  
const double SpindlePitch = 3.0; // 3.0 mm  
const double MM_Step = 0.0075; // mm per step  
const double ToolLengthSensor = 32.2; // Switching point of tool length sensor in mm
```

## **Speeds axis X,Y,Z**

```
const double SpeedG00 = 2400.0; // Speed rapid movement in mm/min, 2400.0mm/min = 40mm/s  
const double SpeedG74 = 1200.0; // Speed reference run in mm/min, 1200.0mm/min = 20mm/s  
const double SpeedInching = 150.0; // Speed inching in mm/min, 150.0mm/min = 2.5mm/s  
const double SpeedMeasureToolLength = 300.0; // Speed measure tool length in mm/min,  
300.0mm/min = 5mm/s  
const double SpeedMeasureToolLengthUpToReferenceSensor = 450.0; // Speed measure tool length  
in mm/min, 450.0mm/min = 7.5mm/s
```

# Speed calculation spreadsheet

1	Fill in green fields only							
2								
3	Input speed must be min. 2, because one cycle output on and one cycle output off							
4	Timer Interrupt in us	15,5625						
5								
6	Steps per turn motor	400	Deg per step	0,9				
7								
8								
9								
10	Speed prog. scan	ms per step	ms/turn	Turns/min	Turns/s	Step frequency / Hz	mm/s	mm/min
11	2	0,031125	12,45	4819,3	80,32	32128,51	240,96	14457,83
12	3	0,0466875	18,675	3212,9	53,55	21419,01	160,64	9638,55
13	4	0,06225	24,9	2409,6	40,16	16064,26	120,48	7228,92
14	5	0,0778125	31,125	1927,7	32,13	12851,41	96,39	5783,13
15	6	0,093375	37,35	1606,4	26,77	10709,50	80,32	4819,28
16	7	0,1089375	43,575	1376,9	22,95	9179,58	68,85	4130,81
17	8	0,1245	49,8	1204,8	20,08	8032,13	60,24	3614,46
18	9	0,1400625	56,025	1071,0	17,85	7139,67	53,55	3212,85
19	10	0,155625	62,25	963,9	16,06	6425,70	48,19	2891,57
20	11	0,1711875	68,475	876,2	14,60	5841,55	43,81	2628,70
21	12	0,18675	74,7	803,2	13,39	5354,75	40,16	2409,64
22	13	0,2023125	80,925	741,4	12,36	4942,85	37,07	2224,28
23	14	0,217875	87,15	688,5	11,47	4589,79	34,42	2065,40
24	15	0,2334375	93,375	642,6	10,71	4283,80	32,13	1927,71
25	16	0,249	99,6	602,4	10,04	4016,06	30,12	1807,23
26	17	0,2645625	105,825	567,0	9,45	3779,83	28,35	1700,92
27	18	0,280125	112,05	535,5	8,92	3569,83	26,77	1606,43
28	19	0,2956875	118,275	507,3	8,45	3381,95	25,36	1521,88
29	20	0,31125	124,5	481,9	8,03	3212,85	24,10	1445,78
30	21	0,3268125	130,725	459,0	7,65	3059,86	22,95	1376,94
31	22	0,342375	136,95	438,1	7,30	2920,77	21,91	1314,35
32	23	0,3579375	143,175	419,1	6,98	2793,78	20,95	1257,20
33	24	0,3735	149,4	401,6	6,69	2677,38	20,08	1204,82
34	25	0,3890625	155,625	385,5	6,43	2570,28	19,28	1156,63
35	26	0,404625	161,85	370,7	6,18	2471,42	18,54	1112,14
36	27	0,4201875	168,075	357,0	5,95	2379,89	17,85	1070,95
37	28	0,43575	174,3	344,2	5,74	2294,89	17,21	1032,70
38	29	0,4513125	180,525	332,4	5,54	2215,76	16,62	997,09
39	30	0,466875	186,75	321,3	5,35	2141,90	16,06	963,86
40	31	0,4824375	192,975	310,9	5,18	2072,81	15,55	932,76
41	32	0,498	199,2	301,2	5,02	2008,03	15,06	903,61
42	33	0,5135625	205,425	292,1	4,87	1947,18	14,60	876,23
43	34	0,529125	211,65	283,5	4,72	1889,91	14,17	850,46
44	35	0,5446875	217,875	275,4	4,59	1835,92	13,77	826,16
45	36	0,56025	224,1	267,7	4,46	1784,92	13,39	803,21
46	37	0,5758125	230,325	260,5	4,34	1736,68	13,03	781,50
47	38	0,591375	236,55	253,6	4,23	1690,97	12,68	760,94
48	39	0,6069375	242,775	247,1	4,12	1647,62	12,36	741,43
49	40	0,6225	249	241,0	4,02	1606,43	12,05	722,89
50	41	0,6380625	255,225	235,1	3,92	1567,24	11,75	705,26
51	42	0,653625	261,45	229,5	3,82	1529,93	11,47	688,47
52	43	0,6691875	267,675	224,2	3,74	1494,35	11,21	672,46
53	44	0,68475	273,9	219,1	3,65	1460,39	10,95	657,17
54	45	0,7003125	280,125	214,2	3,57	1427,93	10,71	642,57
55	46	0,715875	286,35	209,5	3,49	1396,89	10,48	628,60
56	47	0,7314375	292,575	205,1	3,42	1367,17	10,25	615,23
57	48	0,747	298,8	200,8	3,35	1338,69	10,04	602,41
58	49	0,7625625	305,025	196,7	3,28	1311,37	9,84	590,12
59	50	0,778125	311,25	192,8	3,21	1285,14	9,64	578,31
60	51	0,7936875	317,475	189,0	3,15	1259,94	9,45	566,97
61	52	0,80925	323,7	185,4	3,09	1235,71	9,27	556,07
62	53	0,8248125	329,925	181,9	3,03	1212,40	9,09	545,58
63	54	0,840375	336,15	178,5	2,97	1189,94	8,92	535,48
64	55	0,8559375	342,375	175,2	2,92	1168,31	8,76	525,74
65	56	0,8715	348,6	172,1	2,87	1147,45	8,61	516,35
66	57	0,8870625	354,825	169,1	2,82	1127,32	8,45	507,29
67	58	0,902625	361,05	166,2	2,77	1107,88	8,31	498,55
68	59	0,9181875	367,275	163,4	2,72	1089,10	8,17	490,10
69	60	0,93375	373,5	160,6	2,68	1070,95	8,03	481,93

## **Explanation spreadsheet speed calculation**

Speed prog. scan 2 means, that the step output is one cycle high and one cycle low.  
Speed prog. scan 10 means, that the step output is one cycle high and nine cycle low.

The higher the number of steps per turn, the lower the max. achived speed.  
The higher the number of steps per turn, the lower the torque.  
The smaller the number of steps per turn, the finer the gradations in mm/s.  
The lower the TimeInterrupt, the higher the max. achived speed  
The higher the power supply of the stepper motor controller, the higher the achived speed.

ms per step = TimerInterrupt us / 1000 \* Speed prog. scan

ms per turn = ms per step \* Steps per turn motor

Turns per min = (1 / ms per turn) \* 60000

Turns per s = 1000 / ms per turn

Step frequency Hz = 1 / ms per step \* 1000

mm/s = 1000 / ms per turn \* Spindle pitch

mm/min = mm/s \* 60

## **Speed calculation, function „SpeedMM\_Min\_ProgramScans“**

inline unsigned long SpeedMM\_Min\_ProgramScans(double MM\_Min)

Input: double mm/min, output programscans

// ##### Function calculate speed from mm/min -> ProgramScans

// e.g.

// 2400mm/min :60 = 40mm/s

// 40mm/s :3 = 13,3 Turns/s (SpindlePitch 3,0mm)

// 13,3 Turns/s = 13,3 Turns/1000ms = 1000ms/13,3 Turns = 75 ms/Turn

// 75ms/Turn / 400 = 0,1875 ms/Step (Steps per turn motor = 400)

// 0,1875 ms/Step \* 1000 = 187,5 us/Step

// 187,5 us/Step : 15,5625 = 12,048 ProgramScans

→ 2400mm/min = 12,048 program scans (Spindle pitch 3mm, timer interrupt 15,5625us)

## **Amount of steps start - stopramp calculation, function „Y = m\*x+b“**

Unit speeds in mm/min

// Actual Speed >=200 and <= 2000, Steps Ramp = 0..100

// Actual Speed < 200, Steps Ramp = 0

// Actual Speed > 2000, Steps Ramp = 100

// Y = m\*x+b

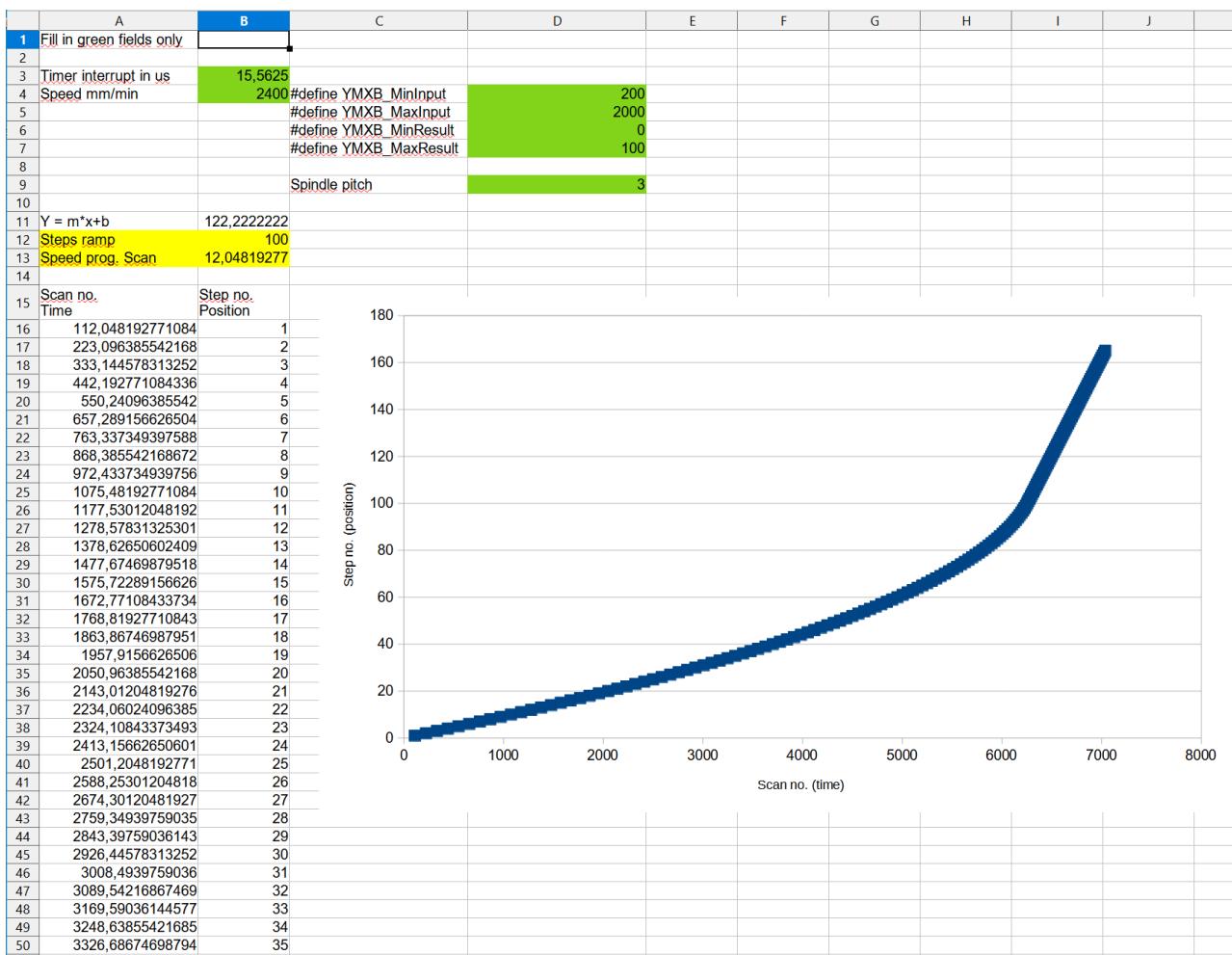
#define YMXB\_MinInput 200.0

#define YMXB\_MaxInput 2000.0

#define YMXB\_MinResult 0.0

#define YMXB\_MaxResult 100.0

## Function start – stopramp spreadsheet



## Explanation spreadsheet function start - stopramp

Speed from G-Code file 2400 (2400 mm/min or 60mm/min)

Calculation steps ramp: 100

Calculation speed prog. Scan 12

Timer interrupt = 15,5625 us

### Movement:

First step after 112 (100+12) prog. Scans ( $112 * 15,5625 \text{ us} = 1743 \text{ us}$  or 1.7ms)

Second step after 223 (112 + 111) prog. Scans ( $223 * 15,5625 \text{ us} = 3470 \text{ us}$  or 3.4ms)

Third step after 333 (223 + 110) prog. Scans ( $333 * 15,5625 \text{ us} = 5182 \text{ us}$  or 5.1ms)

Fourt step after 442 (333 + 109) prog. Scans ( $442 * 15,5625 \text{ us} = 6878 \text{ us}$  or 6.8ms)

...

100th step after 6254 (6241 + 13) prog. Scans ( $6254 * 15,5625 \text{ us} = 97,327\text{ms}$ )

**101th step after 6266 (6254 + 12) prog. Scans ( $6266 * 15,5625 \text{ us} = 97,514\text{ms}$ ) at linear speed**

102th step after 6278 (6266 + 12) prog. Scans ( $6278 * 15,5625 \text{ us} = 97,701\text{ms}$ )

...

# Calculation linear movement

There is a virtual time master (step no.). When a step has to be executed, for example after each 12th prog. Scan, there is a decision, which axis has to make a step.

The axis with the longest distance is making at every decision a step.

The other axis are making it's steps when the difference between the required position and the actual position is  $\geq$  mm/step (0,0075mm).

## Example:

Axis Y has got the longest distance with 533,333 steps

### Distance X: 3,0mm

Amount steps X: 400  $\rightarrow$  3,0/0,0075

Distance per step: 3,0 / 533,3 = 0,005625

### Distance Y: 4,0mm

Amount steps Y: 533,333  $\rightarrow$  4,0/0,0075

Distance per step: 4,0 / 533,333 = 0,0075

### Distance Z: -1mm

Amount steps Z: 133,333  $\rightarrow$  1,0/0,0075

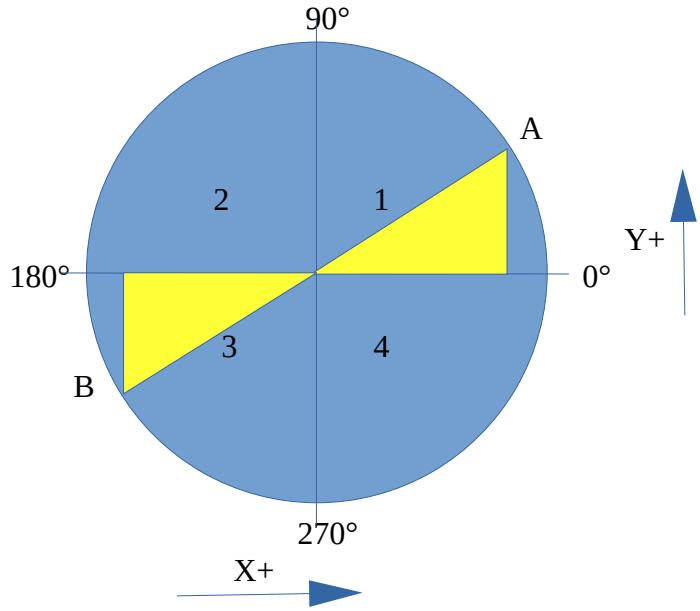
Distance per step: 1,0 / 533,333 = -0,001875

# Linear movement spreadsheet



# Calculation circular movement

There is a virtual time master (step no.). When a step has to be executed, for example after each 12th prog. Scan, there is a decision, which axis has to made a step.



## Example:

Zero point is in the middle of the circle (X0 Y0), radius 50,0mm, G03 counterclockwise

Quadrant no. 1  
X+ Y+

Quadrant no. 2  
X- Y+

Quadrant no. 3  
X- Y-

Quadrant no. 4  
X+ Y-

Calculate coordinates for G-commands:

Startpoint A:

$$X = \cos 30^\circ * 50,0\text{mm} = 43,301\text{mm}$$

$$Y = \sin 30^\circ * 50,0\text{mm} = 25,0\text{mm}$$

Endpoint B:

$$X = \cos 30^\circ * 50,0\text{mm} = -43,301\text{mm}$$

$$Y = \sin 30^\circ * 50,0\text{mm} = -25,0\text{mm}$$

G90

G01 X43,301 Y25,0

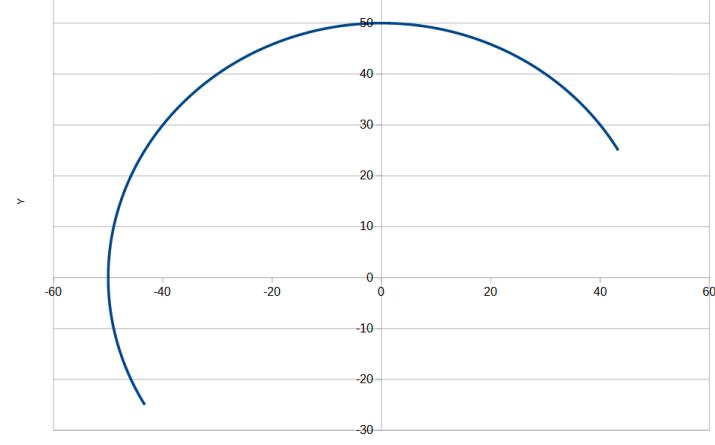
G03 X-43,301 Y-25,0 I-43,301 J-25,0 (I = center of the circle X, J = center of the circle Y)

## Calculations in program:

$$\begin{aligned} \text{Radius} &= \sqrt{(|I|^2 + |J|^2)} & &= \sqrt{((-43,301)^2 + (-25,0)^2)} = 50,0 \\ \text{Center circle } X &= \text{Startposition } X + I & &= 43,301 + (-43,301) = 0,0 \\ \text{Center circle } Y &= \text{Startposition } Y + J & &= 25,0 + (-25,0) = 0,0 \\ \text{Quadrant startpos} &= 1 & & \text{because } I < 0,0 \text{ and } J \leq 0,0 \\ \text{Quadrant endpos} &= 3 & & \text{because } X < \text{CenterCircleX} \text{ and } Y \leq \text{CenterCircleY} \\ \text{Startangle in quadrant 1} &= \arcsin(|J| / \text{Radius}) = \arcsin(|-25,0| / 50,0) = 30,0^\circ \\ \text{Endangle in quadrant 3} &= \arcsin(|(CenterCircleY - (Y)) / \text{Radius}|) + 180,0^\circ = \arcsin(|(0 - (-25,0)) / 50,0|) + 180,0^\circ = 210,0^\circ \\ \text{Distance angle} &= \text{endangle} - \text{startangle} = 210,0^\circ - 30,0^\circ = 180,0^\circ \\ \text{Angle (distance) per step} &= \arcsin(\text{mm per step} / \text{Radius}) = \arcsin(0,0075 / 50,0) = 0,0085^\circ \\ \text{RequiredPosX} &= \cos(\text{ActualAngle}) * \text{Radius} + \text{CenterCircleX} = \text{e.g. } \cos(60,0^\circ) * 50,0 + 0,0 = 25,0 \\ \text{RequiredPosY} &= \sin(\text{ActualAngle}) * \text{Radius} + \text{CenterCircleY} = \text{e.g. } \sin(60,0^\circ) * 50,0 + 0,0 = 43,301 \end{aligned}$$

# Circular movement spreadsheet

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	G03												
2	Start X	Start Y	End X	End Y	I (CenterCircle X)	J (CenterCircle Y)							
3	43,301		25	-43,301	-25	-43,301							
4													
5	Radius												
6	49,9998 * i^2 + j^2												
7													
8	Center of the circle												
9	0 X		Start X + CenterCircle X										
10	0 Y		Start Y + CenterCircle Y										
11													
12	Quadrant startpos												
13	Quadrant 1	1											
14	Quadrant 4	0											
15													
16	Quadrant endpos												
17	Quadrant 1	0											
18	Quadrant 2	0											
19	Quadrant 3	1											
20	Quadrant 4	0											
21													
22	Angel startpoint in Quadrant												
23	30,0001610491016 Quadrant I	SIN(J / Radius)											
24	59,9998576748706 Quadrant IV	SIN(I / Radius)											
25													
26	Angel endpoint in Quadrant												
27	30,0001610491016 Quadrant I + III	SIN((CenterCircleY - EndY) / Radius)											
28	59,9998576748706 Quadrant II + IV	SIN((CenterCircleX - EndX) / Radius)											
29													
30	Startangle	Quadrant I → +0 deg											
31	30	Quadrant II → +90 deg											
32		Quadrant III → +180 deg											
33	Endangle	Quadrant IV → +270 deg											
34	210												
35													
36	Angel (deg)	Radian	X (mm)	Y (mm)									
37	30	0.523598666667	43,3010702707414	24,9998782878701									
38	30,0085	0.52374701962	43,2973609864249	25,0063018544938									
39	30,017	0.52389537258	43,2936507531933	25,0127248707638									
40	30,0255	0.52404372553	43,2899395651295	25,019473365387									
41	30,034	0.52419207849	43,2862274243148	25,0255692516773									
42	30,0425	0.52434043144	43,2825143308309	25,0319906160381									
43	30,051	0.5244887844	43,2788002847594	25,0384114294799									
44	30,0595	0.52463713736	43,2750852861822	25,0448316918613									
45	30,068	0.52478549031	43,2713693351809	25,0512514030411									
46	30,0765	0.52493384327	43,27136933518374	25,0576705628778									
47	30,085	0.52508219622	43,2639345762335	25,0640891712304									
48	30,0935	0.52523054918	43,260215768451	25,0705072279574									
49	30,102	0.52537890213	43,25649600085717	25,0769247329178									
50	30,1105	0.52552725509	43,2527752966775	25,0833416859701									
51	30,119	0.52567560804	43,24005363238503	25,0897580869722									
52	30,1275	0.525823961	43,2453310171721	25,0961739357858									
53	30,136	0.52597231396	43,2416074497246	25,1025892322669									
54	30,1445	0.52612066691	43,23788293059	25,109003976275									
55	30,153	0.52626901987	43,2341574598501	25,1154181676692									
56	30,1615	0.52641737282	43,2304310375869	25,1218318063082									
57	30,17	0.52656572578	43,2267036638826	25,1282448920509									
58	30,1785	0.52671407873	43,2229753338819	25,1346574247561									
59	30,187	0.52686243169	43,2192460624782	25,1410694042827									
60	30,1955	0.52701078464	43,2155158349424	25,1474808304896									
61	30,204	0.5271591376	43,2117846562936	25,1538917032356									
62	30,2125	0.52730749056	43,208052526614	25,1603020223797									
63	30,221	0.52745584351	43,2043194459856	25,1667117877808									
64	30,2295	0.52760419647	43,2005854144907	25,1731209992979									
65	30,238	0.52775254942	43,1968504322114	25,1795296567898									
66	30,2465	0.52790090238	43,19311449923	25,185937760155									
67	30,255	0.5280492553	43,1893776156285	25,192345309134									
68	30,2635	0.52819760829	43,1856397814894	25,198752307042									
69	30,272	0.52834596124	43,1819009968948	25,2051587436653									
70	30,2805	0.5284943142	43,1781612619271	25,2115646289361									
71	30,289	0.52864266716	43,1744205766689	25,2179699593156									



# **Abort movement**

## **Abort inching**

In manual mode, inching of the axis can be aborted due to pressing the inching off button or toggle the active axis by pressing the button of the active axis again. Inching will also be stopped, when the hardware emergency stop button is pressed.

## **Abort measure tool length**

In manual mode, measure tool length can be aborted due to pressing the measure tool length off button via the GUI or pressing the hardware emergency stop button.

## **Abort automatic mode**

If the automatic mode is active, the start button changes to abort button. By pressing the abort button, the automatic mode will be aborted.

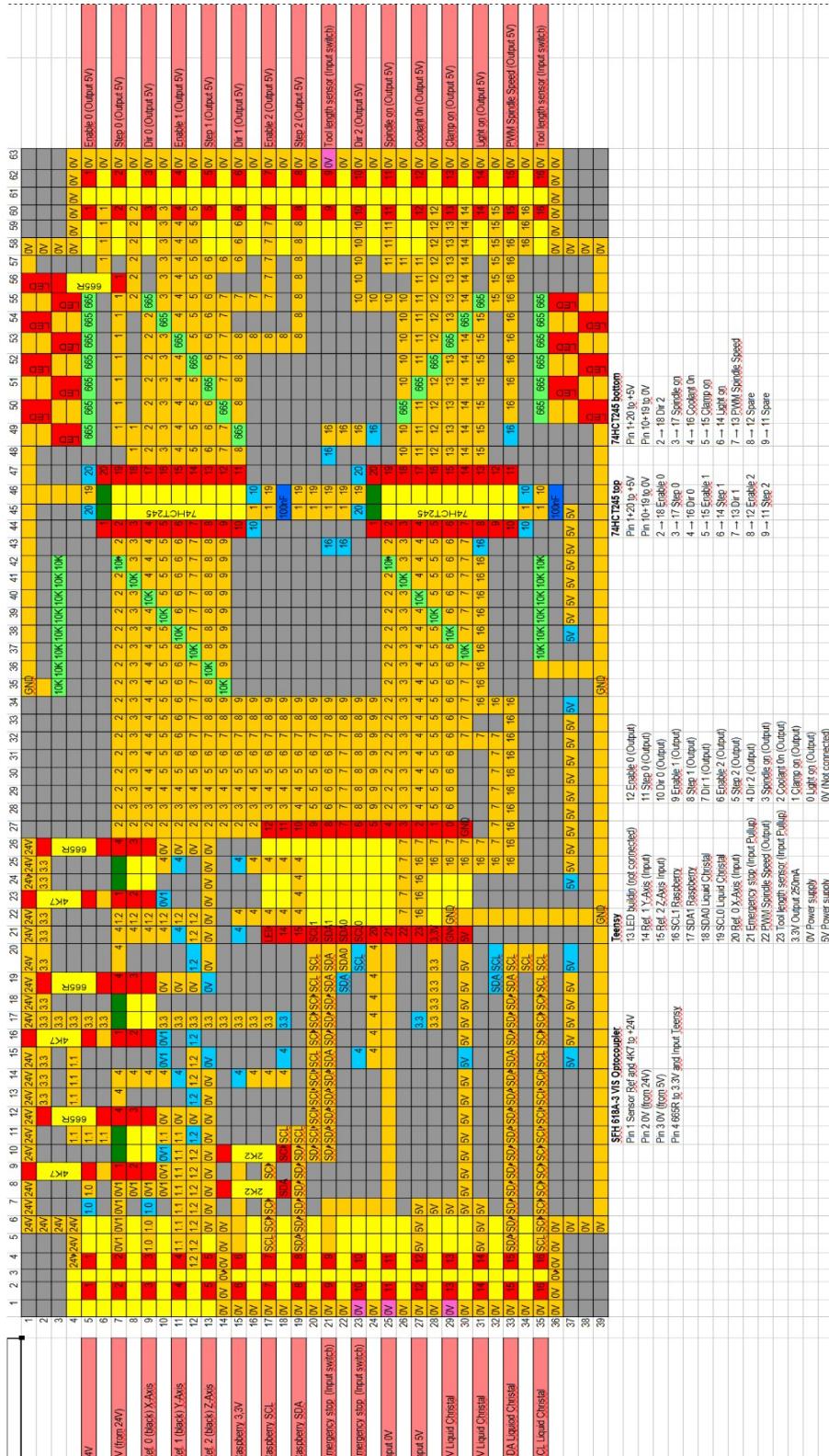
After aborting the automatic mode, the abort button changes back to start. After aborting, the automatic mode can be continued by pressing the start button.

If a movement is active during pressing the abort button, the movement goes to stopramp and then the movement will be stopped. When the movement is stopped, the spindle will be switched off.

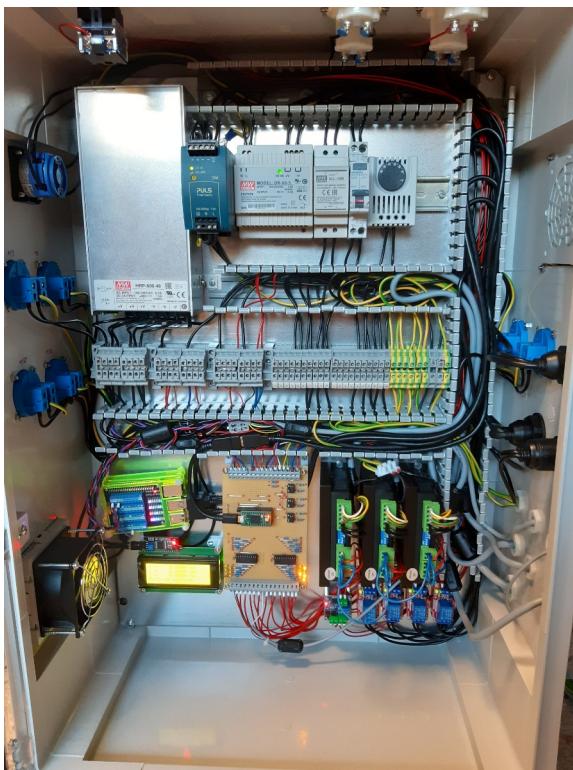
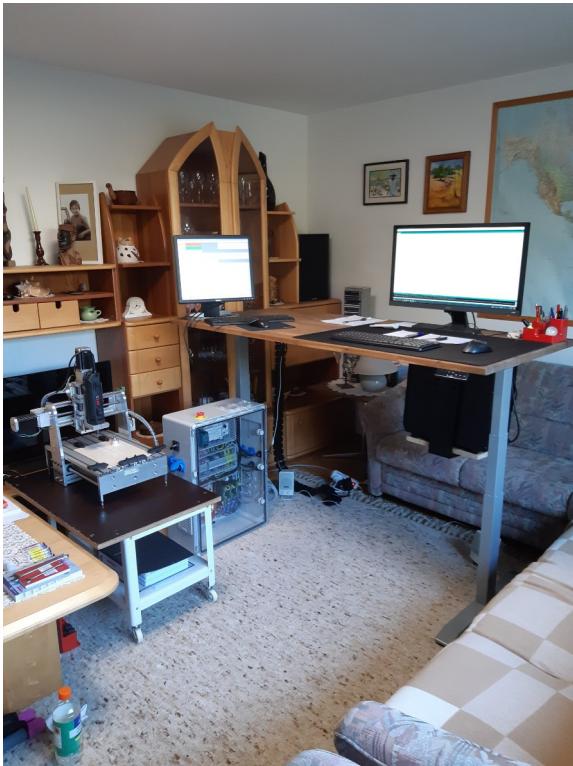
If a movement is already in stopramp when the abort button is pressed, the stopramp will be executed and then the movement will be stopped. When the movement is stopped, the spindle will be switched off.

After aborting, the movement always continues with the startramp.

# Layout circuit board Teensy 4.0

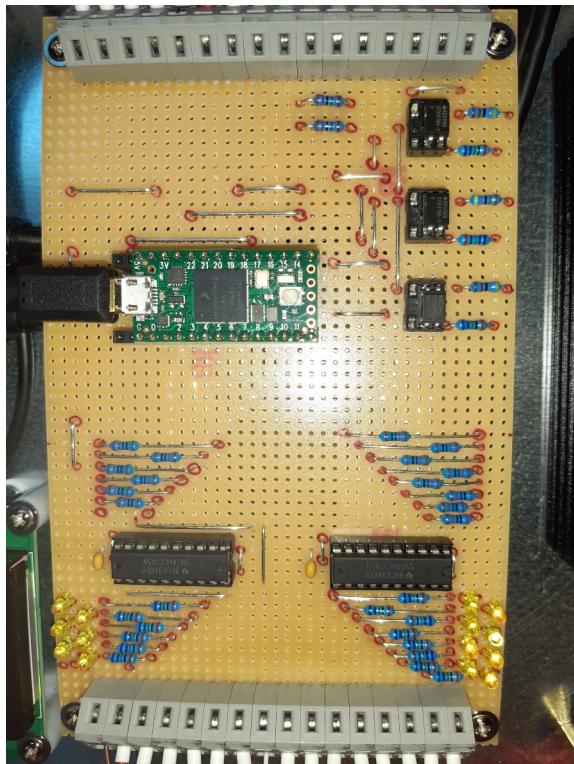
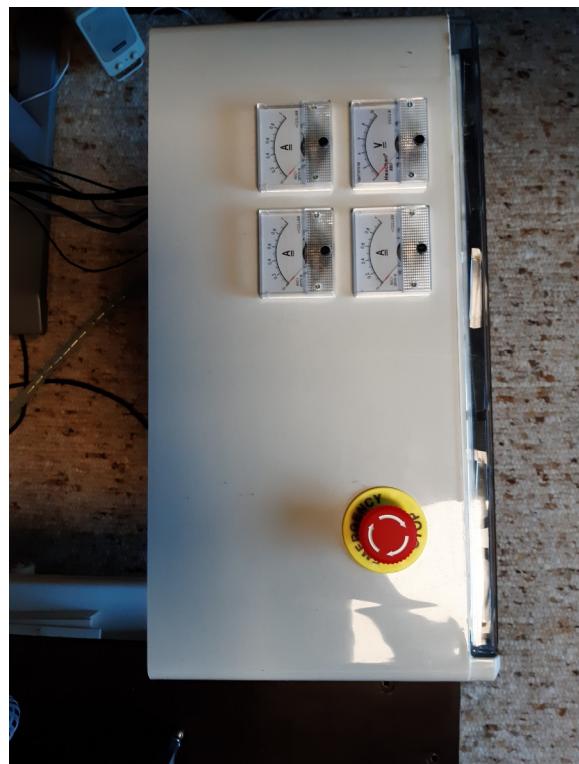


# Pictures



Top: Power supplies 48V, 24V, 5V  
Bottom: Pi4, LCD, circuit board, motor controls  
PWM 0-10V, four output relays

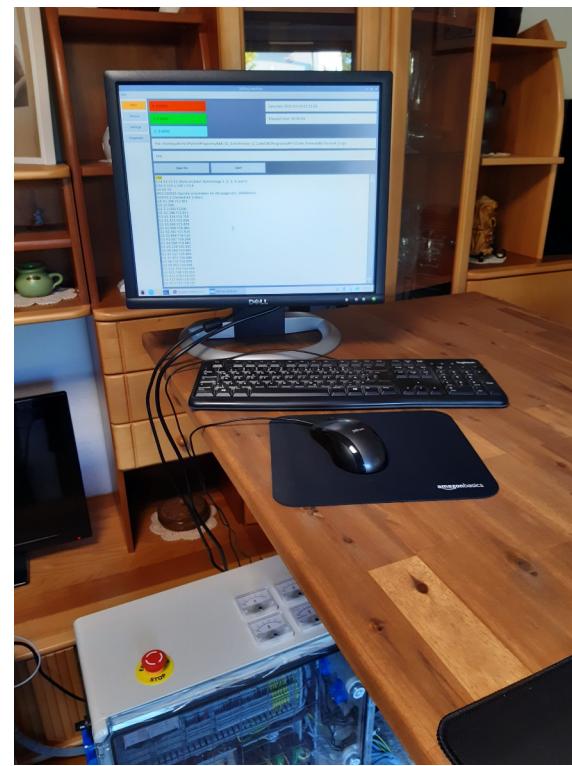
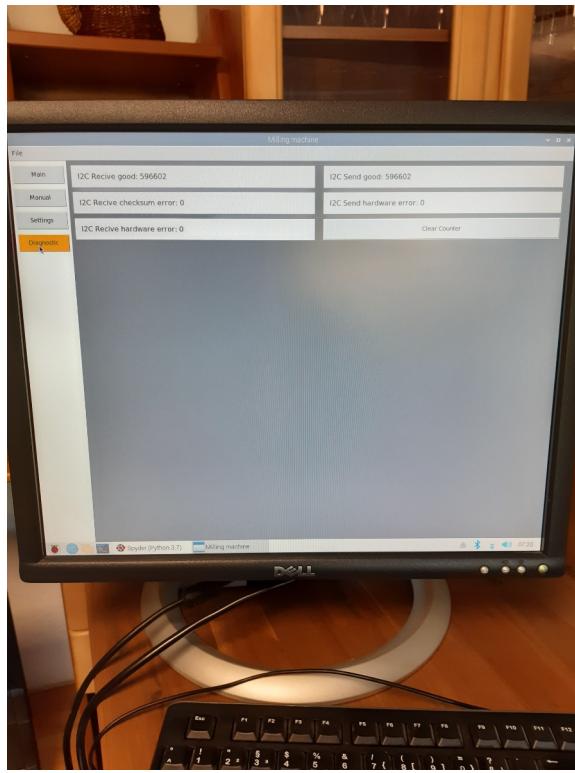
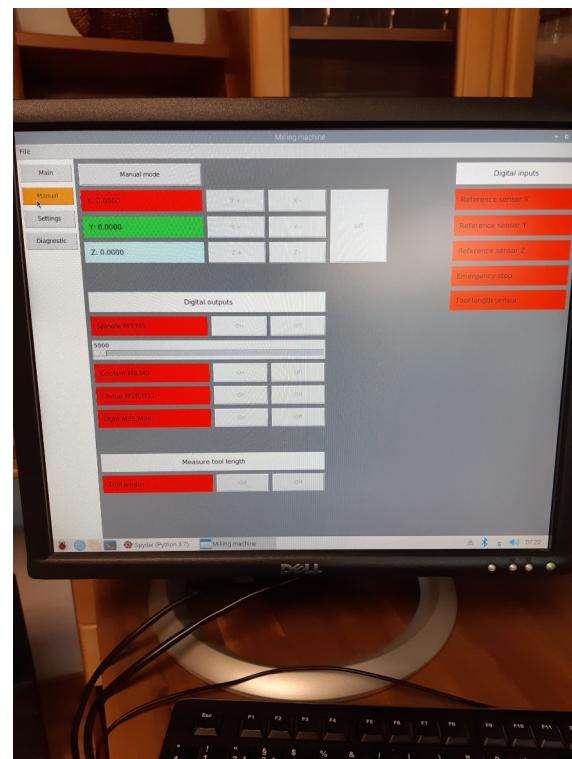
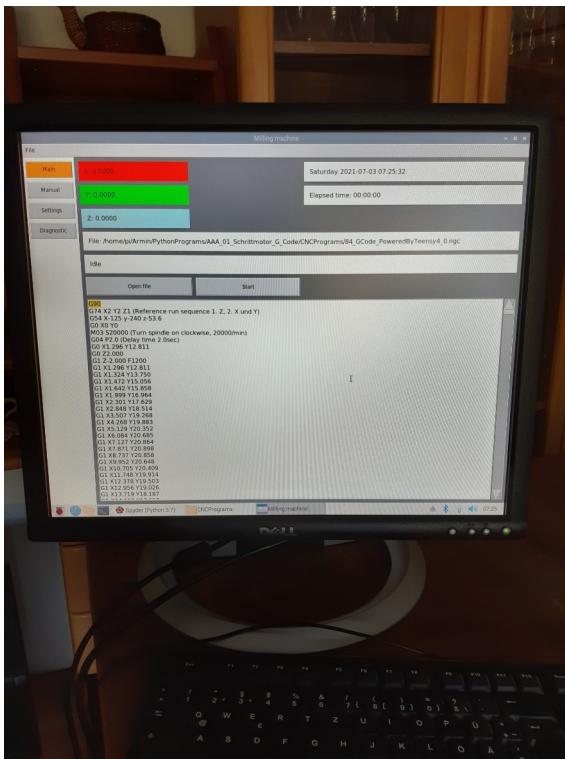
USB programming port Teensy  
Connections for PI4 (USB memory, audio,  
keyboard, HDMI, mouse)



Circuit board with Teensy 4.0



LCD display and Raspberry Pi4, both connectet to Teensy via I2C



GUI on Raspberry Pi4

