Universidad de Valladolid

E.T.S.I. Telecomunicación

Trabajo Fin de Grado

Grado en Ingenierías Específicas de
Telecomunicación, Mención en Telemática

# Design and implementation of a standalone CNC controller

# Ontwerp en implementatie van een stand-alone CNC-besturingseenheid

Autor:
**Elias Verstappe**
Tutor de UVa:
**Dr. Jesús M. Hernández Mangas**
Tutor de Universidad Vives (Brugas, Bélgica):
**Mr. Tom Cordemans**

Valladolid, Junio 2019

| | |
|---|---|
| Title: | **Design and implementation of a standalone CNC controller** |
| Author: | **Elias Verstappe** |
| Tutor: | **Dr. Jesús M. Hernández Mangas** |
| Department: | **Department of Electronics** |

**Court**

| | |
|---|---|
| Chairman: | **Dr. Martín Jaraíz Maldonado** |
| Member: | **Dr. Jesús Arias Álvarez** |
| Secretary: | **Dr. Jesús M. Hernández Mangas** |
| Substitute: | **Dr. Ruth Pinacho** |
| Substitute: | **Dr. Pedro López** |

| | |
|---|---|
| Date: | **21 of June 2019** |
| Qualification: | |

# Preface and acknowledgements

Writing a thesis is the final step in obtaining a degree. It is a unique opportunity to apply all acquired knowledge in one large project. Obviously, the development of such a project does not go without a hitch. It is a mountain that is impossible to get over without falling and getting back up. I would not have succeeded without a strong start and necessary support along the way. Therefore I would like to use this opportunity to express my gratitude to the people who have contributed to the realization of this work.

Therefore, first of all, a sincere thanks to my teachers. Your motivated, enthusiastic and communicative way of working and your course of action is a permanent source of inspiration for me.

In addition, a special thanks goes to my internship promoter, Mr. Tom Cordemans. Thank you for the practical support (in connection with Erasmus) and the confidence that you have placed in me from the start. This thesis would not have been realised without your help.

An additional word of thanks goes out to my two promoters at the University of Valladolid, Dr. Jesús M. Hernández Mangas and Dr. Jesús Arias Álvarez. Day after day you have guided me and at the moments that I started going on the wrong path, your expertise broadened my field of vision.

Finally, I want to thank my parents. You have supported me from the beginning. Not just morally and financially, but in many other areas. Thank you for supporting my choice of studying electronics.

Not to be forgotten are the friends with whom I have been able to spend a nice time. Thank you for the experiences we shared together, the moral support and the enjoyable breaks.

# Abstract

## English

This bachelor thesis has been developed by Daan Ver Eecke and Elias Verstappe. We are two students at the 'Hogeschool Vives' in Bruges (Belgium) and both participate in the Erasmus program. The thesis has been worked out at the University of Valladolid in Spain.
The general purpose of a bachelor's thesis is to solve a specific problem (in our case with regard to the field of study electronics).
This thesis describes how we control an old machine with the help of modern electronics. In a world where technological development is happening faster than ever, it is important that the possibility remains to control old devices. Due to the enormous growth, older machines are no longer supported by new operating systems.

Two lecturers from the University of Valladolid, Dr. Jesús M. Hernández Mangas and Dr. Jesús Arias Álvarez, have confronted us with the following challenge: an old CNC machine is currently controlled with the help of an old computer. The program that controls the machine runs on the MS-DOS operating system. Our assignment is to design a custom controller that takes over the task of the old computer. An additional requirement is to also be able to control the (CNC) machine via a website.
A fully custom printed circuit board is designed to solve this problem. The firmware that is responsible for all control is also developed by us. Briefly summarized, the requirements are the following:
With the help of an SD card, a certain PCB design can be read in the machine. The machine can be controlled in two ways: on the one hand everything can be operated using the touchscreen, and on the other hand the machine must be able to be fully controlled via a website.

Now that the project is finished, we can conclude that thanks to the enormous development of technology in recent years, nowadays you can easily and cheaply design a control unit for old devices. We can simulate the operation of a computer that is 25 years old with a small and simple printed circuit board that costs 45 euros (development costs not included).

This result shows that you could create a 'man in the middle' with new hardware for many old machines. This way of working can certainly be applied to other machines. You could even apply this way of working for development of new machines. We were instructed to control a machine with a certain interface. In principle you can compare it with the development of a controller for a machine. In that case, it does not really matter what or how old the machine is.

# Dutch

Deze bachelor-thesis is uitgewerkt door Daan Ver Eecke en Elias Verstappe. Wij zijn twee studenten aan de ‘Hogeschool Vives’ in Brugge (België) en nemen beiden deel aan het Erasmus-programma. De thesis is uitgewerkt aan de Universiteit van Valladolid in Spanje.

Het algemene doel van een bachelor proef is om een concreet probleem (in ons geval met betrekking tot de afstudeerrichting elektronica) op te lossen.

Dit eindwerk beschrijft hoe we een oude machine aansturen met behulp van moderne elektronica. In een wereld waar de technologische ontwikkeling sneller dan ooit verloopt, is het belangrijk dat de mogelijkheid blijft bestaan om oude toestellen aan te sturen. Door de enorme groei worden oudere machines niet meer ondersteund door nieuwe besturingssystemen.

Twee docenten van de Universiteit van Valladolid, Dr. Jesús M. Hernández Mangas en Dr. Jesús Arias Álvarez, hebben ons voor de volgende uitdaging geplaatst: een oude CNC-machine wordt momenteel bestuurd met behulp van een oude computer. Het programma die de machine bestuurt draait op het MS-DOS besturingssysteem. Het is onze opdracht om een op maat gemaakte controller (besturingseenheid) te ontwerpen, die de taak van de oude computer overneemt. Een extra vereiste is om de (CNC-) machine ook via een website te kunnen besturen.

Om dit probleem aan te pakken en op te lossen wordt een volledig op maat gemaakte printplaat ontworpen. De firmware die verantwoordelijk is voor alle sturing wordt ook door ons ontwikkeld. Kort samengevat zijn de vereisten de volgende:

Met behulp van een SD-kaart kan een bepaald PCB-ontwerp in de machine gelezen worden.

De besturing van de machine kan op twee manieren: enerzijds kan alles bediend worden met behulp van het touchscreen, anderzijds moet de machine volledig bestuurd kunnen worden via een website.

Nu het project uitgewerkt is kunnen we besluiten dat je dankzij de enorme ontwikkeling van de technologie, vandaag gemakkelijk en goedkoop een besturingseenheid kan ontwerpen voor oude toestellen. Wij kunnen nu de werking van een computer die 25 jaar oud is nabootsen met een klein en simpel printplaatje dat 45 euro kost (ontwikkelingskost niet inbegrepen).

Dit resultaat toont aan dat met nieuwe hardware voor veel oude machines als het ware een ‘man in the middle’ ontwikkeld kan worden. Deze manier van werken kan zeker toegepast worden op andere machines, zelfs op nieuwe machines. Wij verkregen een opdracht om met een bepaalde interface een machine te besturen. In principe kun je het vergelijken met het ontwikkelen van een besturingseenheid voor een machine. Dan maakt het eigenlijk niet zoveel uit welke of hoe oud de machine is.

# Contents

# List of Figures

# List of Tables

# List of abbreviations and symbols

| | |
|---|---|
| **ARM** | Advanced RISC machine[1] |
| **ASCII** | American standard code for information interchange |
| **CNC (machine)** | Computer numerical control (machine) |
| **FAT** | File allocation table |
| **HPGL** | Hewlett-Packard Graphics Language |
| **IC** | Integrated circuit |
| **LCD** | Liquid crystal display |
| **MCU** | Microcontroller unit |
| **MISO** | Master in slave out |
| **MOSI** | Master out slave in |
| **PCB** | Printed circuit board |
| **PFF** | Petit FAT filesystem |
| **SMD** | Surface-mounted device |
| **SPI** | Serial peripheral bus |

# Chapter 1

# Introduction

## 1.1 Definition of terms

This final settlement was an exam. The comments made during the defence were not included.

This bachelor's thesis was elaborated by Daan Ver Eecke and Elias Verstappe. We are two Belgian students from 'Hogeschool Vives, campus Bruges Station'. We study 'electronics', which is part of the applied engineering and technology field of study. Our mentor from our university in Belgium is Mr. Tom Cordemans. Our two mentors from the University of Valladolid are Dr. Jesús M. Hernández Mangas and Dr. Jesús Arias Álvarez.

This study contains information about the hardware development and firmware that controls the CNC machine. This study was written by Elias Verstappe. Alongside this project, a website was made to control the CNC machine. You can read about the development of the website and the drawing of the 3D printed case in the study that Daan Ver Eecke wrote.

## 1.2 Statement of the problem

The main goal of a bachelor's thesis is to solve a specific problem, with regards to the field of study. Two professors from the University of Valladolid have given us the following problem to solve:

An old CNC machine that is used to manufacture printed circuit boards, is currently controlled by an old personal computer. The communication is done via the parallel port and the application to control the machine runs on the MS-DOS operating system. It is our task to design a controller that controls the CNC machine. This should be done by designing a custom printed circuit board (PCB).

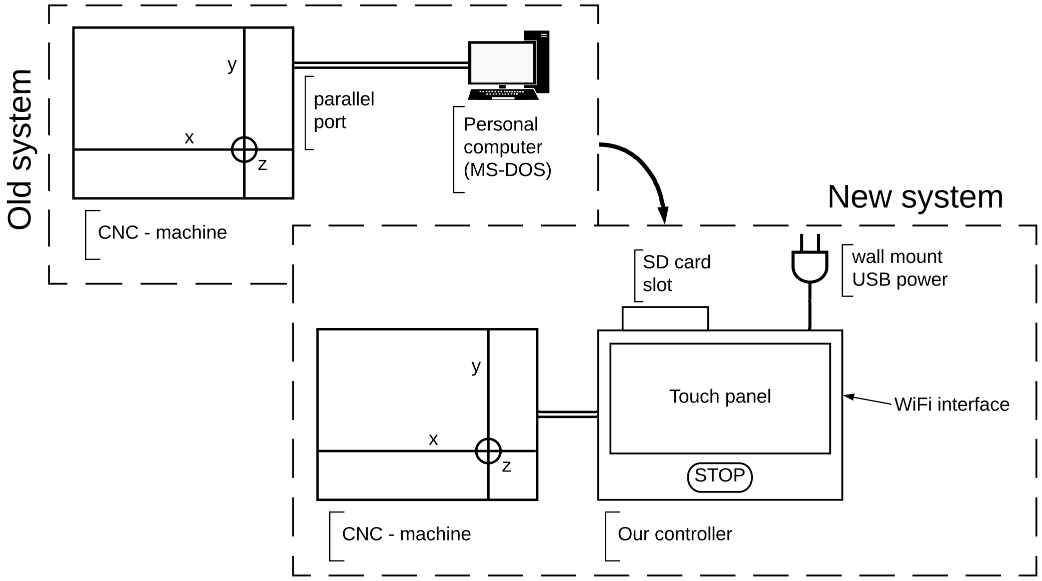


Figure 1.1: Project definition

## 1.3 Description of the chapters

**Chapter 1** has a detailed description of the specific requirements that were set when the idea of this thesis was developed.
**Chapter 2** describes the specific details about the PCB design.
In **Chapter 3** you can read about the firmware that manages the various tasks the controller has.
Furthermore, **Chapter 4** is a user manual that describes all parameters that can be set and describes how the 'operating system' that runs on the board works.
**Chapter 5** gives you an idea of how much the development this project would cost.
Finally, **Chapter 6** contains conclusion of the final work and discusses the achieved results.

## 1.4 Requirements and specification

Specific requirements of the project are the following:

- The controller needs to operate fully independently, no computer will be required when using the CNC machine.
- The design that you wish to manufacture by the CNC machine is supplied to the controller via an SD card.
- The (CNC) machine should be fully controllable using a web page.
- The (CNC) machine should be fully controllable via a touchscreen.
- The firmware will be written in C (reason being that C provides us with a lot of low level control over hardware components)

# Chapter 2

# Hardware

## 2.1 Used software

The software that was used to develop the PCB is Proteus Design Suite. This software tool is used to design and develop schematics for PCB's. It was developed by Labcenter Electronics Ltd. One of the reasons why Proteus was used is that we can easily simulate our design on the computer. This is useful, reason being that we can already start writing firmware for the design while we wait for the manufacturer to develop our circuit board.

## 2.2 Design approach

Figure 2.1 gives you a closer look at how the circuit board is designed and shows which components are present. The following subsections describe the major parts of the board.

We need specific hardware to meet the requirements. A list of all integrated circuits (IC's) and other components can be found in the bill of materials, which is one of the appendices of this study.

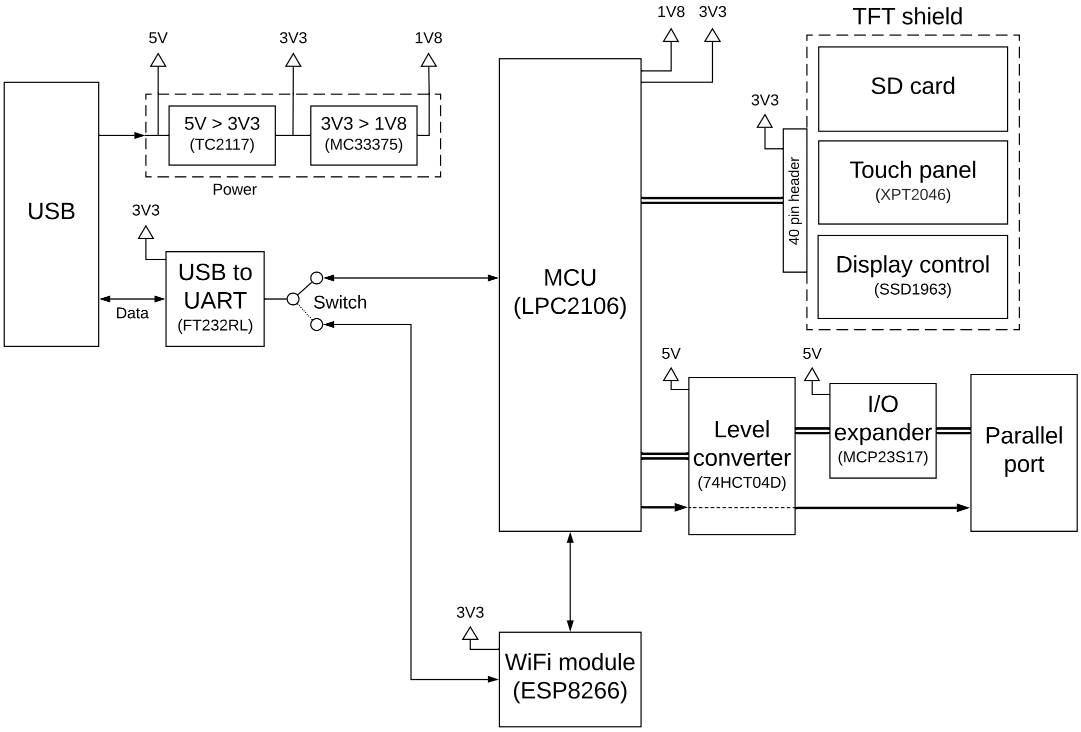


Figure 2.1: Simplified hardware schematic

### 2.2.1 Power supply

Power is provided by the USB connection. Our circuit uses both 5V and 3,3V as well as 1,8V. The 5V is directly derived from the USB connection, the 3,3V and 1,8V are achieved by using voltage regulators. Specific design can be found in the PCB layout, which is included as an appendix.

Every power rail is connected to ground via a capacitor of 22$\mu$F. The datasheet suggest that we use 10$\mu$F capacitors, but for stock reasons 22$\mu$F capacitors were used. This has no negative effect on the working of the power supply.

With the help of two voltage regulators the voltage is converted from 5V to 3,3V and then from 3,3V to 1,8V. The figure below shows how this was implemented.

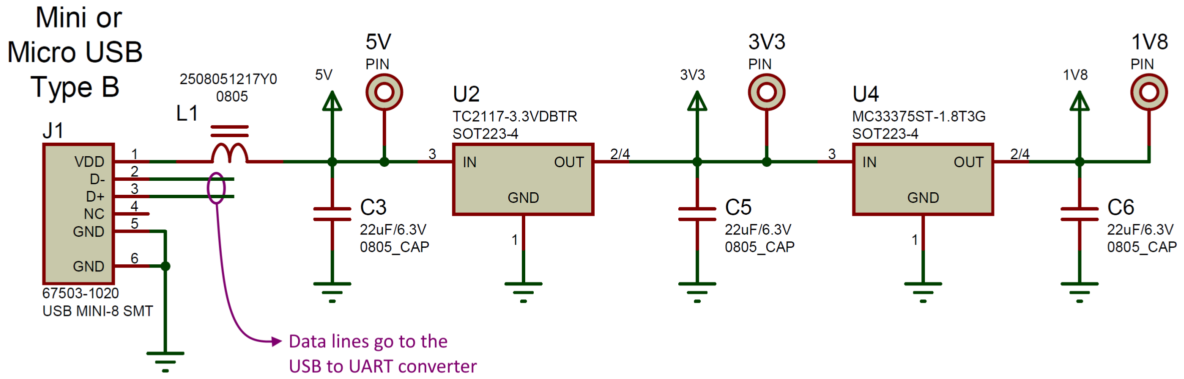


Figure 2.2: Power supply

You may have noticed that there is another component between the power supply of the USB connector and the 5V power rail (component L1). This is a ferrite bead. A ferrite bead[2] is an electric component that suppresses high-frequency noise in electronic circuits. No safety measures have been taken in connection with a potential electrostatic discharge. This is not the main focus of this study and we do not expect problems in that regard.

More information about the power analysis can be found at section 2.4 of this chapter.

### 2.2.2 USB to UART converter and switch between controllers

The microcontroller (when 'the microcontroller' is used, I'm refering to the LPC2106) and WiFi module will be programmed via USB. The board will also be powered via USB. A USB to UART converter is needed to be able to communicate with the microcontroller and WiFi-module.

We only want to use one USB to UART converter (and one USB connection). To be able to choose which microcontroller we program, we use a switch. The schematic below gives a clearer view of how these parts are connected. This can also be found in the schematic itself, which is one of the appendices.

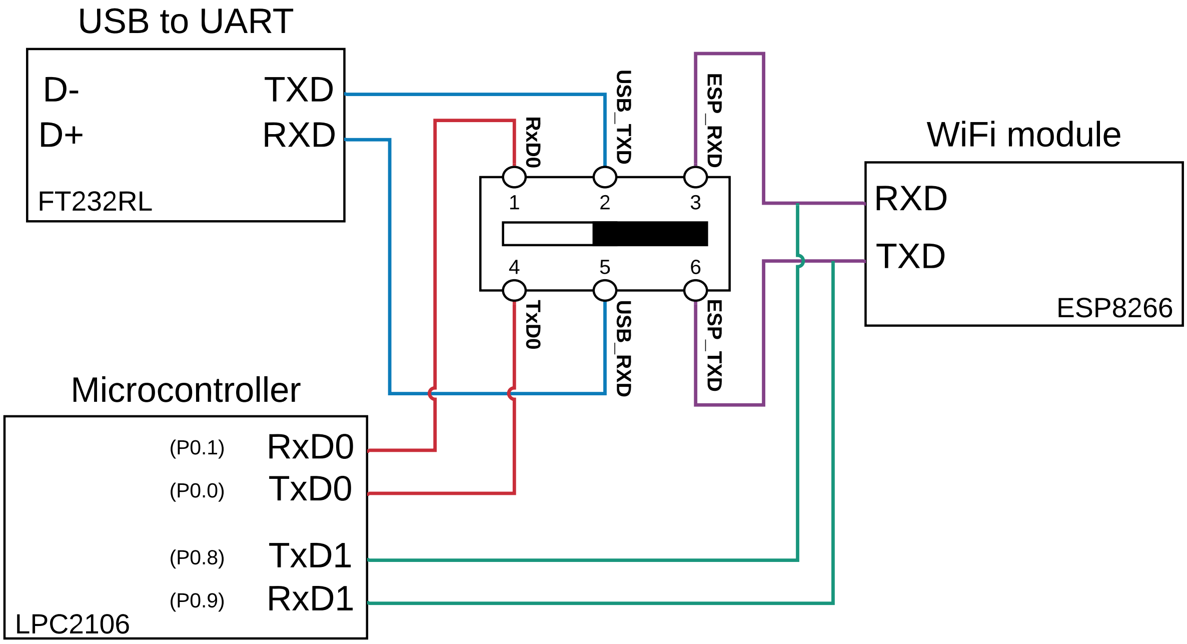


Figure 2.3: Switch between controllers for programming

When the slide switch is in one position, the converter's *transmit* signal is connected to the microcontroller's *receive* pin. Also, the converter's *receive* signal is connected to the microcontroller's *transmit* pin. This results in data communication between converter and microcontroller. When the switch is in the other position, data is beeing shared between the converter and the WiFi module.

You can also see that the WiFi modules UART interface is connected to the second UART interface of the microcontroller. This connection is present because we need to be able to control the CNC machine via a website. Because the microcontroller is responsible for the management and control of the CNC machine, the WiFi module has to tell the main microcontroller what the user wants to do.

### 2.2.3 Microcontroller

The microcontroller that we use is an LPC2106/01. This is an ARM based microcontroller (or MCU). We chose this controller because we have experience using it and the lecturers who developed the idea of this project can help us during development if any major problems would arise.
The microcontroller runs on a frequency of approximately 58,98 MHz. In our application, this frequency is achieved by a clock crystal that produces a frequency of 14,7456 MHz, which is then multiplied by four using a PLL.

In figure 2.4 we can see that our microcontroller has 64kB or RAM memory and 128kB of flash memory.

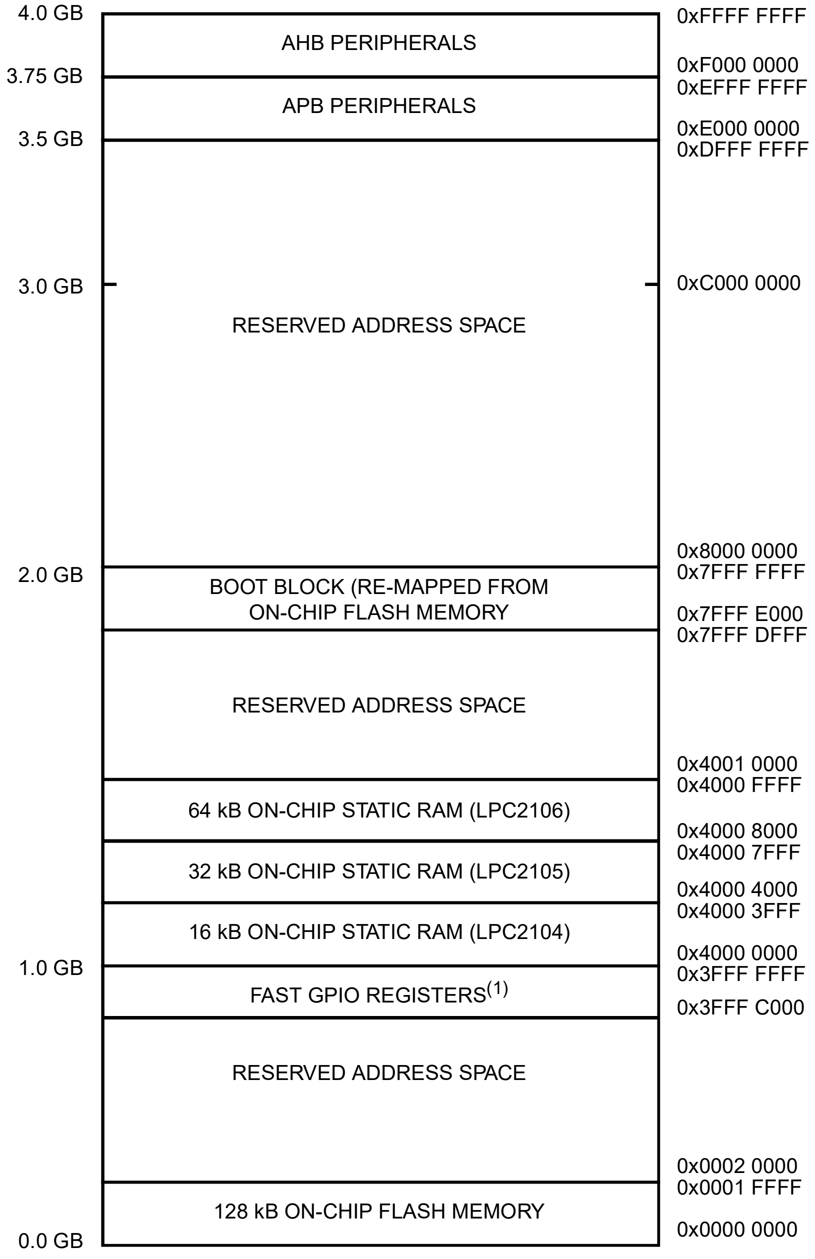


Figure 2.4: The microcontrollers memory map

It might seem that this microcontroller has four gigabyte of storage. In reality, this is not the case. What matters when making the controller is the RAM and flash memory.

### 2.2.4 TFT shield

The TFT shield contains three controllers, one is responsible for the touch management. The second one is responsible for controlling the LCD screen and the last one is the SD card itself. Given that these chips (and SD card slot) were already integrated on the shield, we didn't have to design anything special for this part of our PCB. We only need to correctly connect the pins from the 40 pin connector to the microcontroller.

**Touch controller**

The touchscreen is controlled by the XPT2046 IC. This chip communicates with the microcontroller via SPI. The IC controls a resistive touch screen. A resistive touch screen works by applying a voltage across a resistor network and measuring the change in voltage at a given point on the matrix where a screen is touched by an input stylus, pen or finger. The change in the resistance ratio marks the location on the touch screen.

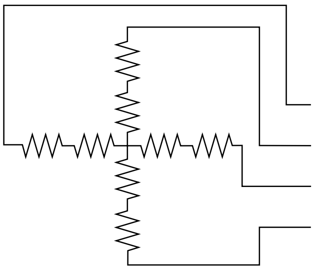


Figure 2.5: Four-wire touch screen circuit

Our touch screen is based on the 'four-wire' architecture. A four-wire touch screen is constructed as shown in figure 2.6. It consists of two transparent resistive layers. The four-wire touch screen panel works by applying a voltage across the vertical or horizontal resistive network. The A/D converts the voltage measured at the point the panel is touched. A measurement of the Y position of the pointing device is made by connecting the X+ input to a data converter chip, turning on the Y+ and Y- drivers and digitizing the voltage seen at the X+ input. The voltage measured is determined by the voltage divider developed at the point of touch. For this measurement, the horizontal panel resistance in the X+ lead doesn't affect the conversion due to the high input impedance of the A/D converter. Voltage is then applied to the other axis, and the A/D converts the voltage representing the X position on the screen through the Y+ input. This provides the X and Y coordinates to the associated processor.

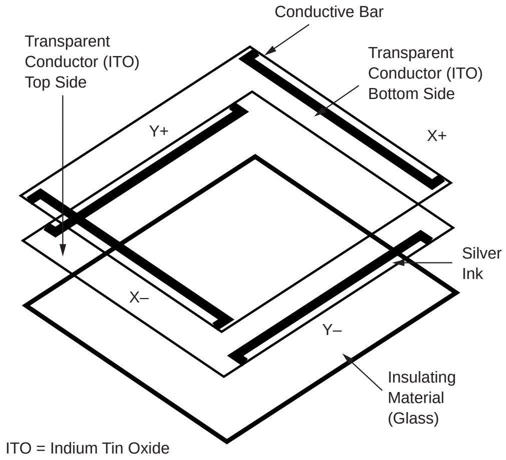

Conductive Bar
Transparent Conductor (ITO) Top Side
Transparent Conductor (ITO) Bottom Side
Y+
X+
Silver Ink
X–
Y–
Insulating Material (Glass)
ITO = Indium Tin Oxide

Figure 2.6: Four-wire touch screen construction

The touch screen is used in differential mode. This basically means that any variation in the touch panels resistance due to change in power supply, changes in driver impedance or changes in temperature will be compensated for by the ratiometric operation of the A/D converter. We use differential mode, as opposed to single-ended mode, because it is more accurate.

### LCD screen

The screen is connected to the microcontroller using *sixteen data* lines, *one read* and *one write* line, *a reset* line and *a chip select* line. Data is sent to the LCD screens controller using the sixteen data lines. For colors, only sixteen bits are used, the RGB565 color representation is used for this. This basically means that five data lines are used for the red pixels, six for the green pixels and the five remaining ones are used for the blue pixels. This was done to be able to send the color of one pixel at once, in stead of splitting this process into multiple data transfers. The result is a faster data rate which helps with the responsive feeling and fast working of the screen.

### SD card

The SD card communicates with the microcontroller via SPI.

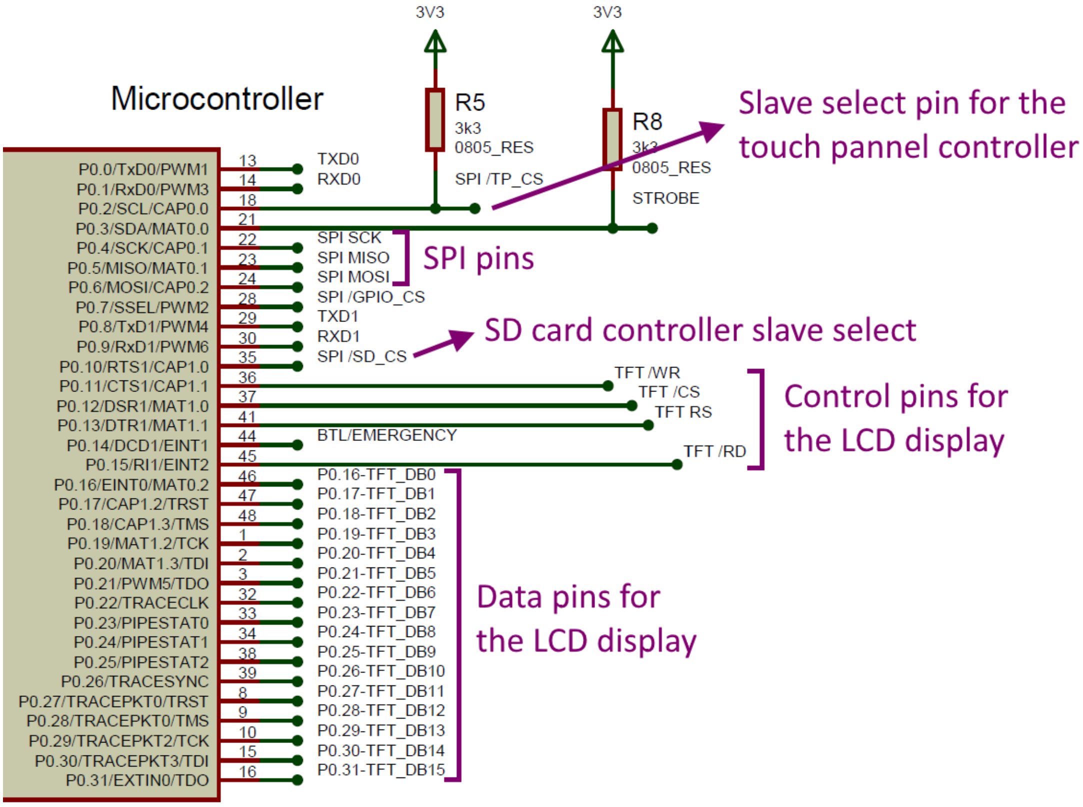


Figure 2.7: Pins related to the TFT shield

Also note that P0.2 (touch panel slave select) and P0.3 (strobe) are connected via a resistor to the 3,3V power supply. This is due to the fact that these pins are open-drain. Pull-up resistors are essential in this case.

### 2.2.5 Parallel port

#### I/O expander

The output of the parallel port is controlled through the I/O expander. We use the MCP23S17 IC. The reason why the parallel port wasn't directly connected to the microcontroller is that we don't have enough I/O pins to our disposal. The microcontroller communicates with the I/O expander via SPI. The I/O expander then controls the pins of the parallel port. Data coming from the CNC machine itself is sent to the microcontroller via the same SPI interface.

#### Level converter

As a result of using the I/O expander and because the parallel port needs to be 5V logic, we need a 'level converter'. The microcontroller works at a voltage level of 3,3V. The I/O expander works with 5V logic. We fixed this problem by placing an integrated circuit that sees a 3,3V signal as a logical '1' between the microcontroller and the I/O expander. The integrated circuit we use is a 74HCT04. This is an inverter circuit that contains six NOT gates. Both the SPI clock and SPI MOSI are connected to the I/O expander via two NOT gates (inverting twice results in the same signal, if we ignore the propagation delay). The two remaining NOT gates are each connected to a different signal, one is connected to the 'chip select' (slave select) pin. The last one is connected to the 'strobe' pin. The strobe signal has its own dedicated pin because we need very precise control over this signal. It is connected directly to the microcontroller. A final point to take into account is the MISO signal. The pins of the microcontroller are 5V tolerant, but the MISO pin is also connected to other integrated circuits that are not 5V tolerant (for example the touch panel controller or the SD card). For this reason, the signal must be converted from a 5V signal to a 3,3V signal. You can see that this is done by using a simple voltage divider in figure 2.8.

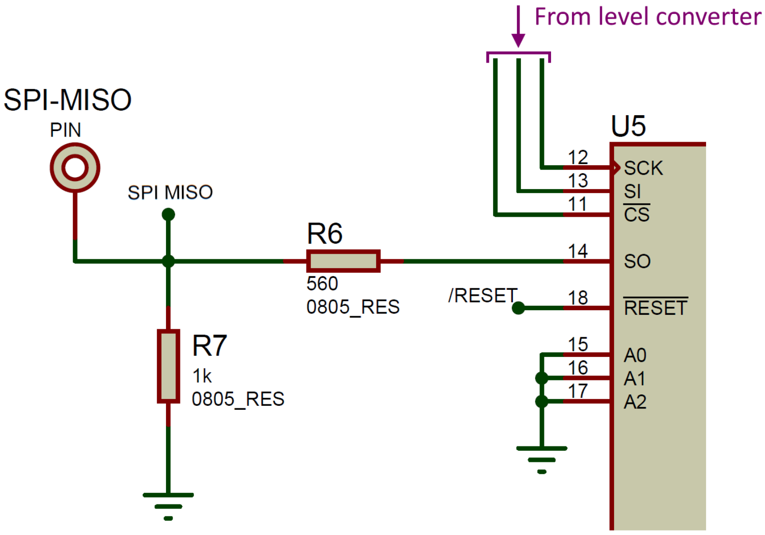


Figure 2.8: Resistor devider for level conversion

When the I/O expanders 'slave out' pin is high (5V), the voltage found on the 'SPI-MISO pin' can be calculated as follows:

$$V_{SPI-MISO} = \frac{R_6}{(R_6 + R_7)} * V_{SO}$$
$$V_{SPI-MISO} = \frac{1k\Omega}{(560 + 1k\Omega)} * 5V$$
$$V_{SPI-MISO} = 3,205V$$

## 2.3 The layout

We managed to use a two layer PCB. This means that conductors are only present on the top and bottom side of the circuit board, in contrast to a more-than-two layered design, where additional layers are inside of the board. Most components are SMD components. The only through-hole components that are used are connectors such as the 40-pin header and the 25-pin SUBD connector.

The schematic itself can be found in appendix B. Figures of the real board can be found on page 11.

We took some options into consideration when placing the components are the following facts:

- All connectors (USB and SUBD) are placed at the edge of the PCB, such that you can easily connect a cable to it.
- Decoupling capacitors are placed as close as possible to the components.
- The button is placed where it is, so that it is located in the centre line of the screen.
- The 40 pin header is placed on the side so that the centre of the TFT shield is as close to the centre of the main PCB as possible.
- As the datasheet for the WiFi module suggests, it is advised to have no copper under and around the antenna. In this figure you can see that no copper can be found under the antenna and 15mm from the edge of the board.

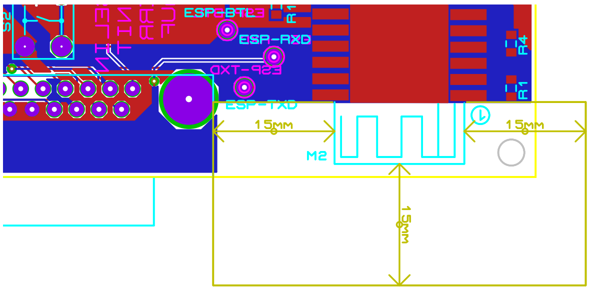
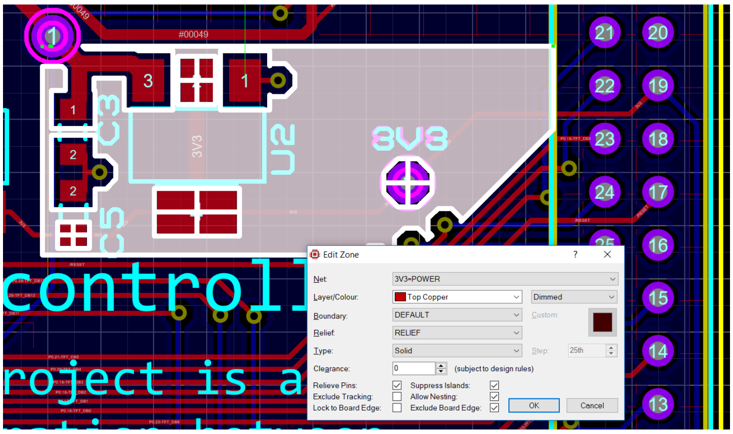


Figure 2.9: Antenna copper cutout

- The power pin of the voltage regulators have a small copper plane. Connecting this pin to a larger surface area is done so that the regulator can dissipate some heat via this plane.



Net: 3V3=POWER

Layer/Colour: Top Copper | Dimmed

Boundary: DEFAULT | Custom:

Relief: RELIEF

Type: Solid | Step: 25th

Clearance: 0 (subject to design rules)

Relieve Pins: ☑ Suppress Islands: ☑

Exclude Tracking: ☐ Allow Nesting: ☑

Lock to Board Edge: ☐ Exclude Board Edge: ☑

Figure 2.10: Copper heat plane connected to 3V3 pin

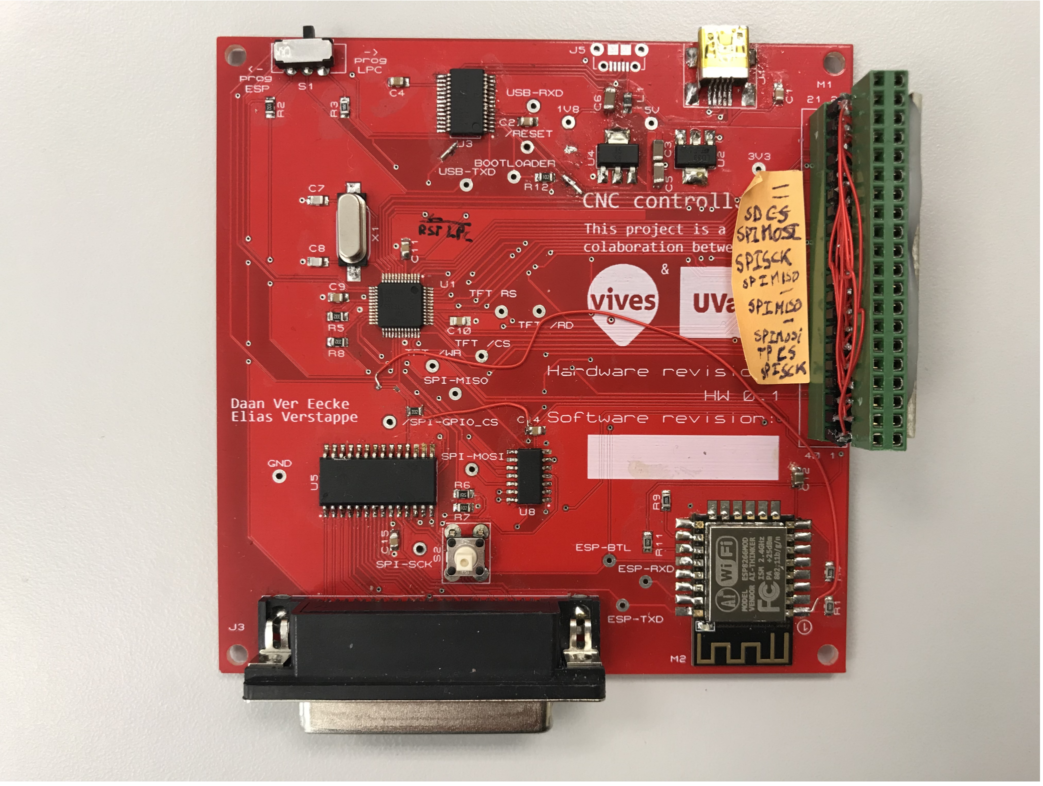


Figure 2.11: Real board top view

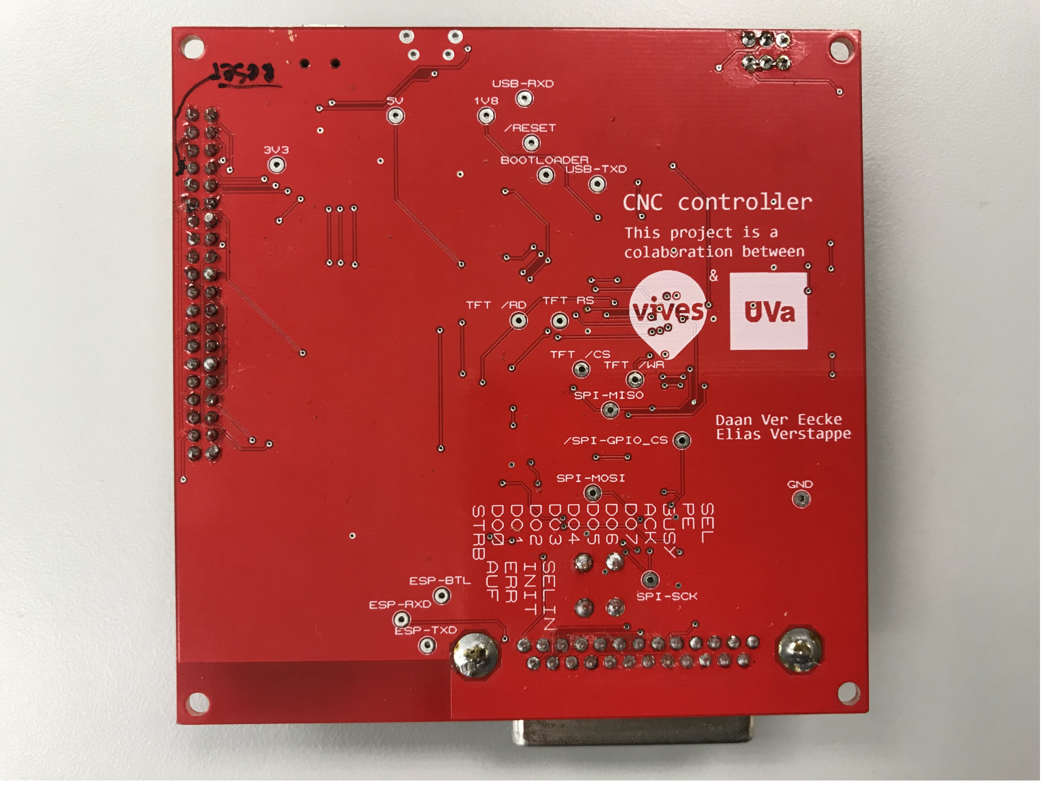


Figure 2.12: Real board bottom view

## 2.4 Power analysis

To power our PCB we use a micro USB a mini USB type B connection. These USB connections provide a voltage of 5V with a maximum current of 0,5A. This equals a power of 2,5W.

| **5V power level** | **3,3V power level** | **1,8V power level** |
|---|---|---|
| 40$\mu$A (level-converter)<br>2mA (I/O expander)<br>420.2mA (3V3)<br>130$\mu$A (3V3 regulator internal) | 15mA (USB to UART converter)<br>200mA (TFT)<br>170mA (WIFI module)<br>35mA (1V8)<br>200$\mu$A (1V8 regulator internal) | 35mA (microcontroller) |
| TOTAL @ 5V = 422.37mA<br>POWER @ 5V = 2.12W | TOTAL @ 3V3 = 420.2mA<br>POWER @ 3V3 = 1.39W | TOTAL @ 1V8 = 35mA<br>POWER @ 1V8 = 0.063W |

Table 2.1: Table with currents drawn by each component

We have three voltage levels in our design: 1,8V, 3,3V and 5V. There is only one component connected to the 1.8V level, this is the LPC2106 (microcontroller). The microcontroller consumes a maximum of 35mA. We can now calculate the required power for the 1,8V power level.

$$P = U * I = 1,8V * 0,035A = 0,063W = 63mW$$

The 3,3V power rail is connected to three components, but we have to take five currents into account. The three components connected to the 3,3V level are the USB to UART converter, the TFT module and the WiFi interface. These consume 15mA, 200mA and 170mA respectively. The current needed in for 1,8V circuit also has to pass through the 3,3V regulator. We also have to take this current into account to calculate the maximum current draw fot the 3,3V power level. Together we get a total of 420,2mA for the 3,3V power level. The power used for this power level can be calculated as follows:

$$P = U * I = 3,3V * 420,2mA = 1,39W$$

Our PCB is powered by the USB cable which provides a voltage of 5V. There are only two components connected to the 5V level, the level converter, which consumes 40$\mu$A and the I/O expander which consumes 2mA. We also have to take into account the current used by the 3,3V level. That is 420,2mA. e also need a current of 130$\mu$A to power the electronics inside of the 3,3V voltage regulator. If we add all of these currents together, we get a total current of 422,37mA in for the 5V level. With these values we can calculate the power used by the entire PCB.

$$P = U * I = 5V * 422,37mA = 2,12W$$

## 2.5 Heat dissipation

It is very important to perform a heat dissipation analysis on the PCB. If the heat analysis indicates that a component may overheat, then we should consider using a heatsink. In our project the two voltage regulators have the biggest chance to overheat.

| Component | IC code | Max. temp [°C] | Conditions |
|---|---|---|---|
| Microcontroller | LPC2016 | 85 | $I_{DD}$ = 45mA @ 60MHz |
| USB to UART | FT232RL | 85 | / |
| 16 bit I/O expander | MCP23S17 | 125 | / |
| Level-converter (NOT port) | 74HCT04D | 70 | Above 70°C $P_{tot}$ drops 8mW/K |
| Wifi interface | ESP-12F | 125 | Operating temp. |
| 1V8 voltage regulator | MC33375ST-1.8T3G | $R_{\theta JA}$ = 80°C/W, 125°C | / |
| 3V3 voltage regulator | TC2117-3.3VDBTR | $R_{\theta JA}$ = 59°C/W, 125°C | / |

Table 2.2: Table with heat dissipated by each component

### 3,3V voltage regulator

The datasheet of the TC2117-3.3VDBTR (3,3V voltage regulator) indicates that the thermal resistance between the junction and the air equals 59°C/W. On the basis of our power analysis we know that the current through the 3,3V regulator is 420,2mA. We can determine the drop-off voltage by subtracting the output voltage from the input voltage.

$$= (5V - 3,3V) = 1,7V$$

Now we can calculate the power generated in the 3V3 regulator. If we then multiply this result with the thermal resistance we get the temperature that the component reaches.

$$420,2mA * (5V - 3,3V) * 59\frac{°C}{W} + 25°C = 67,15°C$$

### 1,8V voltage regulator

The generated heat who this component will emit can be calculated in the same way as shown above. The datasheet of the MC33375ST-1.8T3G(1V8 regulator) indicates that the thermal resistance ($R_{\theta JA}$) is 80°C/W. We can use the same method as with the 3V3 regulator to calculate the stationary temperature.

$$35mA * (3V3 - 1V8) = 0,0525W * 80\frac{°C}{W} + 25°C = 29,2°C$$

The above calculation is not very accurate, but it is not of much importance because the power dissipation of the 1,8V circuit is very low.

# Chapter 3

# Firmware

There are two microcontrollers on the circuit board. One of them controls the screen and CNC machine itself, the other is responsible for the WiFi integration of the controller. The following sections provide information about the firmware that was written for the main microcontroller (LPC2106). The diagram below shows which components are connected to each other and how they communicate with each other.

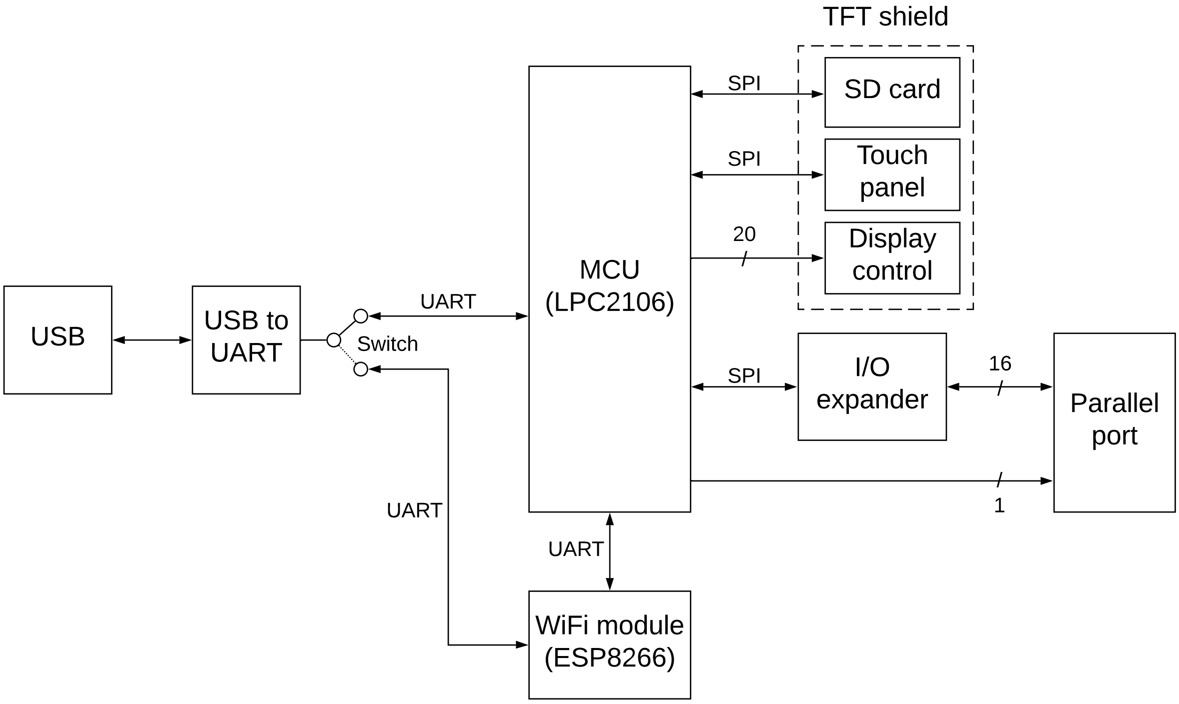


Figure 3.1: Connections and associated protocols

## 3.1 Tasks of the microcontroller

The main microcontroller, which is the LPC2106, is responsible for controlling the CNC machine. This encompasses reading instructions from an SD card and processing instructions from the website. The microcontroller also has to give te user feedback, and the user has to be able to control the machine through the LCD/touch panel.

All firmware that is flashed into the LPC microcontroller, is written in C. C is the language of choice because it is easy to learn, but difficult to master. It warrens an efficient design, to some extent. The compiler that is used is part of the 'YAGARTO' toolchain, which stands for 'Yet Another Gnu Arm Toolchain'.

### 3.1.1 Detailed description

The program starts out by initializing various parts of the circuit board. First, the direction and output state of pins are set, next various UART and SPI interfaces are initialized. Some interrupts, such as the external interrupt for the emergency button, are also configured. Finally, the TFT screen and CNC machine are initialized.
After initialization, some default values for the tools and configuration are set.
Now, the main loop of the program is entered. This loop repeats the following things over and over:

1. Processing the current screen, which involves checking if a screen has been changed, initializing a new screen in the case of a screen change, and checking if any buttons are pressed if the screen stays the same.

2. Check if a full package was received from the WiFi module and decode it if so.

3. If a package was received from the WiFi module, update the screen to update the values.

More information about the package that is received from the WiFi module can be found at section 3.8.

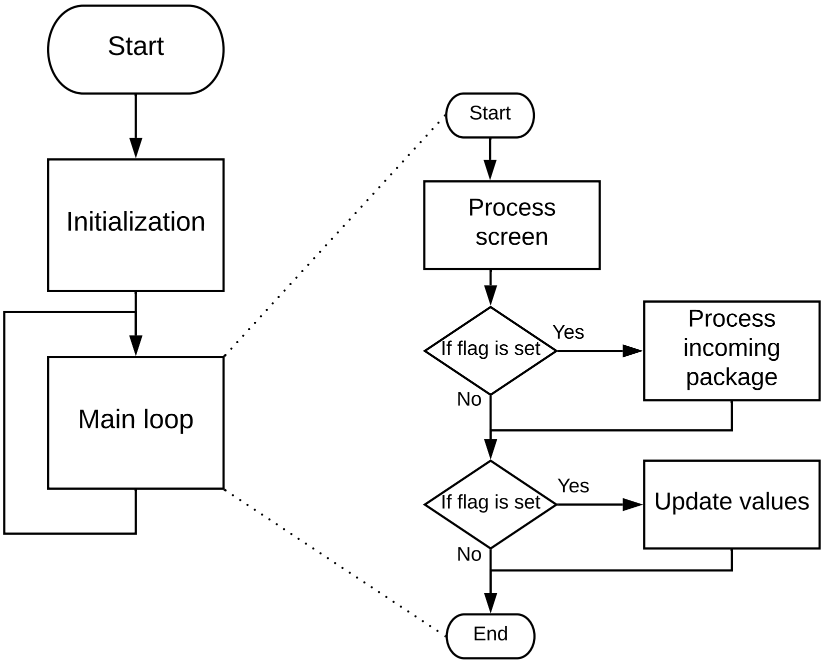


Figure 3.2: Main function

### 3.1.2 Screen handler

The LCD screen is controlled by three functions, an initialize function, a process function and an update function. An enumeration is used to be able to reference each screen. The enumerators (=possible values) of this enumeration are used to set a screen and are also used in a state machine to process the screen. A global variable is used which contains the 'value' of the current screen. The first time the controller is booted, the initialize function is called and sets the first screen.

#### Updating a screen

Whenever the screen needs to be changed (going to another menu), the global variable which indicates the current screen is changed. This is simple, but the program needs to take into account some other factors when doing this.

First, a flag needs to be set that tells other parts of the program (mainly the part that processes the screen) that the screen has been changed. You will read more about why this is in the next section.

Second of all, we need to add some functionality to be able to use a pop-up message. A pop-up message only covers part of the screen. When a such a message is closed, the previous screen (the screen that is partly visible behind the pop-up) needs to be redrawn. A second global variable contains that information. Before setting the global variable that indicates the current screen with the 'new screen', the previous screen variable gets the value of the current screen. To finalize this, a check is required to only change to the pop-up once. If this check is not added, the previous display could become the pop-up screen, which would mean that the menu returns to itself when we expect to resume to the menu that is behind the pop-up.

See the code of this function below.

```
display_states_t g_display_state;       // Global variable indicating current dispay
display_states_t g_previous_display;    // Global variable indicating previous dispay


void display_manager_set_state(display_states_t new_state)
{
    //If new display is warning and warning is already displayed, do nothing.
    if(new_state == e_disp_warning && g_display_state == e_disp_warning)
    {
    } else
    {
        g_previous_display = g_display_state;
        g_display_state = new_state;
        set_display_change_flag ();
    }
}
```

### Processing a screen

A state machine is responsible for displaying and handling the screen. It is important to understand what 'processing' has to be done, before explaining how the state machine works.

Handling a screen is done in two parts. The first part is drawing the screen (so actually displaying it), the second part encompasses checking if any button is being touched. How a touch of the screen is handled will not be discussed in this section. For more information about that, read sections 3.3 and 3.4.

To not keep drawing the main layout of the screen, which would result in the screen flickering, the layout should only be drawn once. Checking the buttons however, should be done periodically.

As you have read in the previous section, a flag is set to indicate that the display has been updated. Inside of the state machine that processes the screens, this flag is checked. If the flag is set, which would mean that there is a change in display, the 'init' function of that display is called as well as clearing the flag. Regardless of the fact that the screen has to be initialized or not, the 'scan' function is called. This function handles all touching of the display. The flowchart in figure 3.3 further explains this state machine.

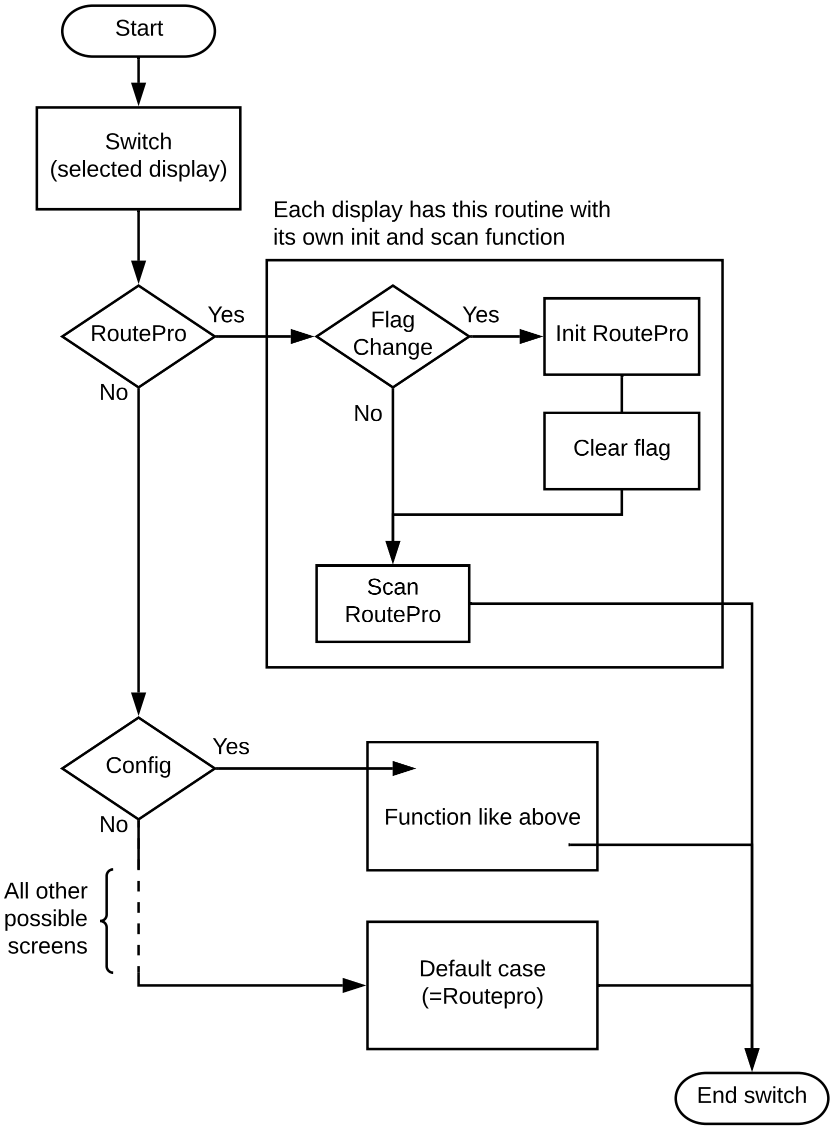


Figure 3.3: Processing the displays

## 3.2 LCD screen

The LCD display needs to display menus, so that the user can interact with the CNC machine. These menus need to be made from scratch. Several functions go into drawing an entire screen. First, some low level functions change the state of output tins of the microcontroller. After that, one function (that uses the low level functions) can select a window in which the pixels can be altered in color. When one pixel can be drawn, this function can be used to draw a line. The function that colors a pixel can also be used to print a character or a bitmap. The screen is 480 pixels by 272 pixels large.

### 3.2.1 Pixel

The *drawPixel* function and its colored function *drawCPixel* use several other funcions to color a pixel. To be able to change the change a pixel, we need some more functions that have more low level control over the screen. First, two commands were written to be able to write a command or data to the LCD controller. These functions are the following:

```
void writeCmd(unsigned short cmd)
{
    TFT_RS_L();
    TFT_DATA_SET( cmd );
    TFT_WR_L();
    TFT_WR_H();
}
```

```
void writeData(unsigned short data)
{
    TFT_RS_H();
    TFT_DATA_SET( data );
    TFT_WR_L();
    TFT_WR_H();
}
```

As you can see, they simply change the state of some pins to the desired value, which is given as an argument to these functions. These abovementioned functions are only directly used by the LCD screen initialization function and the *setXY* function. The setXY function defines a window in which pixels can be altered. Several commands (set_column_address, set_page_address and write_memory_start) as well as the coordinates of two points are sent to the screen controller. One more function that uses the 'writeData' command is the 'setPixel' command. Two (or three) arguments are passed to this command. These arguments are the x and y coordinate, as well as the color that the pixel has to be. Because we only have sixteen data lines, and don't want to split up the pixel's color to take two clock pulses to process, we have decided to use RGB565 color representation. RGB565 dedicates five bits to the intensity of the red color, six for the blue color and 5 for the green color. To color a pixel, a window is set that covers only that one pixel. Next the data lines are set to represent a certain color.

```
void drawCPixel(int x, int y, int color)
{
    setXY(x, y, x, y);
    writeData(color);
}
```

### 3.2.2 Line

Several functions were written to draw a line. Two of them are *drawHLine* and *drawVLine*. These functions draw a horizontal of vertical line, as they suggest. To draw these lines, the 'drawPixel' function is called inside of a loop, with each iteration of the lop incrementing a coordinate which is associated with either the x or y coordinate of the 'drawPixel' function. Other functions like *fillBox* work with the same principle. With the help of a loop this function draws a box that is all one color.

The other function can draw a line is simply *drawLine*. This function calls the simple HLine and VLine functions in case the line is not diagonal. If the line is diagonal, that would mean that the difference between $x_1$ and $x_2$ AND the difference between $y_1$ and $y_2$ is not equal to zero. In this case, the Bresenham algorithm is used to help draw a diagonal line. Drawing a diagonal line is not that self-evident as you'd think. Because we don't want to include floating point arithmetic libraries, the microcontroller only knows integers. See section 3.2.3 for more information about the Bresenham algorithm.

All of these functions have colored variants.

### 3.2.3 Bresenham's line algorithm[3]

Bresenham's line drawing algorithm connects two points with a straight line by calculating all intermediate points on a pixelated canvas.

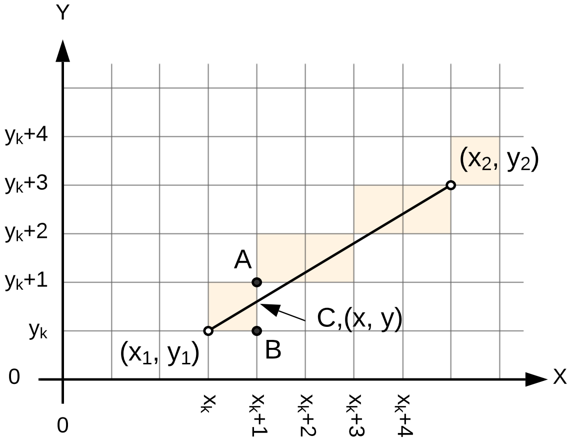


Figure 3.4: Result of Bresenham's line algorithm

The slope of the line can be calculated as follows:

$$\begin{aligned} m &= \frac{(y_2 - y_1)}{(x_2 - x_1)} \\ &= \frac{\Delta y}{\Delta x} \end{aligned}$$

Consider that (m < 1). In this case the change in x is larger than the change in y. We will be incrementing x by one and we will calculate its respected y. If (m > 1) the change in y woud be bigger than the change in x.

In the schematic you can see that the line is going on between the 'A' and 'B' dot.The diagonal line intersects with the grid line at point C, with coordinates (x, y). Two lengths are created, $d_1$ $(A_x - C_x)$ and $d_2$ $(C_x - B_x)$. The next point on the line is now selected by comparing $d_1$ and $d_2$. If $d_1$ is larger than $d_2$, point A will be chosen as the next point, because the line is closer to point A. The ratio between $d_1$ and $d_2$ is known as the decision parameter.

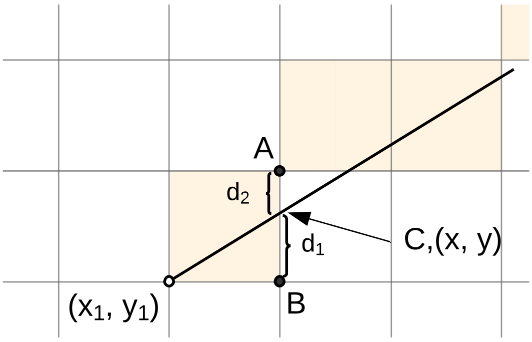


Figure 3.5: Closup of figure 3.4

This is a general explanation. The functions in C code were obtained from the UTFT library[4] for drawing the line on the screen and some code that had already been written to move the CNC machine.

### 3.2.4 Character and string

The *printCChar* function prints out a character on the display. There is a corresponding array that is used to determine what pixels need to be and what pixels don't need to be colored. To help explain how this function works, here is a part of the array that contain one of the used fonts.

```
// Font Size        : 8x12
// Memory usage     : 1144 bytes
// # characters     : 95
fontdatatype SmallFont[1144] PROGMEM={
0x08,0x0C,0x20,0x5F,
.
.
0x00,0x00,0x00,0x00,0x00,0x30,0x48,0x38,0x48,0x3C,0x00,0x00, // a
.
```

As you can see, it is an array of characters (fontdatatype is a user defined datatype and basically just is the same as unsigned char). This example is for the small font, which is eight pixels wide by twelve pixels high. The first four bytes of this array give us some information about the font. At the first and second indices the width and height of a character is stored (in this case 0x08 by 0x0C or just eight by twelve pixels). The third element indicates what the first character is. Here we can see that the first character is index 0x20 or 32 in the ASCII table, which is the space character (not shown). The next element tells us how many characters are 'described' in this array. In this case, 0x5F which is 95 decimal.

To now print a character, one of the 'lines' which describes a character is used to determine where a pixel needs to be colored. One of these 'lines' is checked, bit by bit. When the current bit equals zero, a pixel should be colored in the background color. If the bit is one, that pixel should be colored in the foreground color. Have a look at the following example with the 'a' character. In figure 3.6 you can see how a character is formatted.

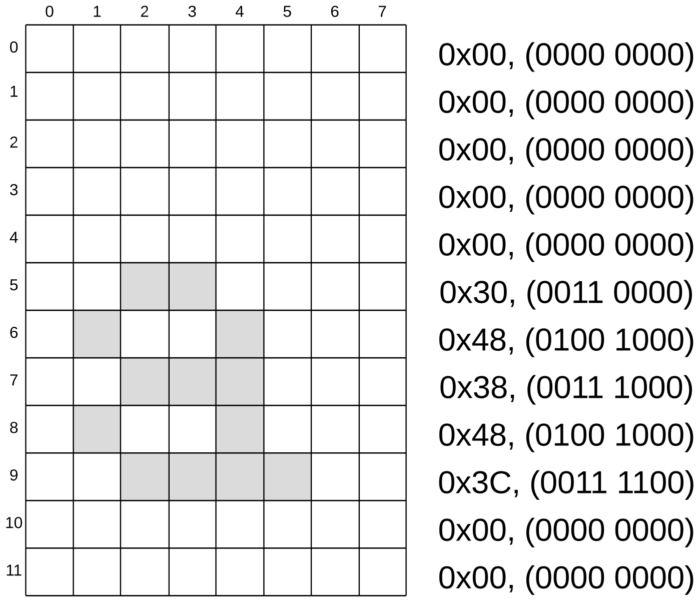


Figure 3.6: 'a' character bitmap

To print out this character, two loops are used to cover all pixels. A first loop is responsible for the x coordinate and increments by one every iteration. The y coordinate increments by one every eight incrementations of the x coordinate. In the case of the small font, eight, in the case of another font with other dimensions this could be different. Every time the y coordinate is incremented, the x coordinate is set to the original value. The data that describes the font (the height and width of a characters 'bitmap') is taken into account to print out a character.

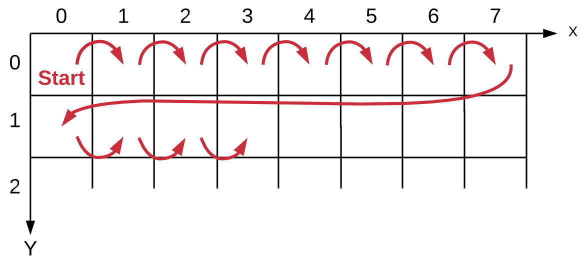


Figure 3.7: 'Line feed' and a 'carriage return'

### Bitmap

A bitmap is printed in much the same way as a character. The first few entries of the array contain information about the graphic. The array structure looks something like this: [width, height, actual amount of color entries, start of color entries, ... ]. The difference between this kind of bitmap and a character is that this each entry of the array represents a full color (RGB565) instead of having each bit represent just black or white. This makes graphic bitmaps much larger than a simple character. In the example below, a five by five bitmap is shown. The full graphic is 56 bytes large. The 'a' character in figure 3.6 only takes up approximately 16 bytes. A graphic that is almost four times the size only takes up about 30% of space.

```
unsigned short arrow_head_y[28] PROGMEM =
{
0x0005, 0x0005, 0x0019,
0x0000, 0x0000, 0x8410, 0x0000, 0x0000,
0x0000, 0x0000, 0xFFFF, 0x0000, 0x0000,
0x0000, 0xC618, 0xFFFF, 0xC618, 0x0000,
0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000,
0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0xC618
};
```

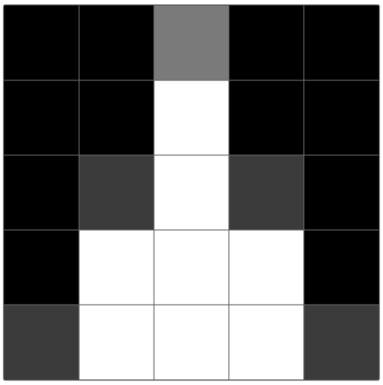


Figure 3.8: Example graphic

## 3.3 Touch control

The touch screen is controlled via SPI. To read an x or y coordinate, a control byte is sent to the touch screens controller. This control byte contains a start bit, a bit to tell the controller to measure in differential mode and some bits that tell the controller which coordinate you want to read. The touch controller responds with a twelve bit number. Both the x and y coordinate will be returned as values between 0 and 4096. The LCD screen that lies underneath the transparent touch conductors works with pixels. For x this value can be between 0 and 479 and for y it can be between 0 and 271.

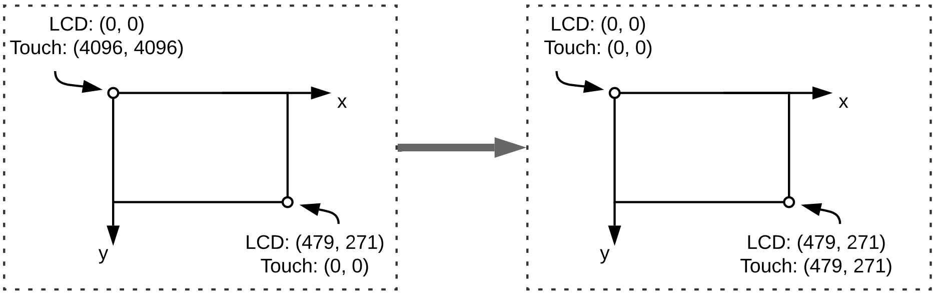


Figure 3.9: Difference in coordinates

It would be handy to have the coordinated from the LCD screen and touch panel match up (and be the same), so that there is no confusion when we check if a certain part of the screen has been touched. For this reason a conversion has to be done. This can easily be done by applying the rule of three.

### 3.3.1 Calibration[5]

Mechanical misalignment and scaling factor can lead to a mismatch between values from a touch panel system (touch screen and touch screen controller) and the display on which the touch screen is mounted.

#### Image transformation

The figure below is an exaggerated view of the distortion that might happen to a circle being displayed on an LCD touch screen display. When a finger is traced around the circle (red line), the touch panel system may give out coordinates of an ellipse (blue line).

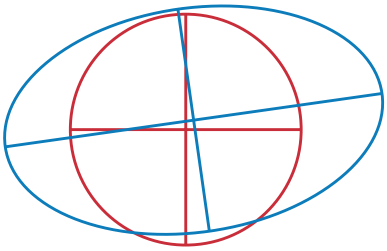


Figure 3.10: Circle transformation due to misalignment and scaling

#### Mathematics

The calculations for the calibration can be found in appendix E.

A calibration function has been used to obtain the transformation matrix elements, to solve equations. The results (the matrix) is stored as fixed values inside of the microcontroller, not to calibrate the screen each time we power the system on.

## 3.4 Interfacing with the machine

The user can control the machine with the aid of the TFT screen. As the touch panel and LCD screen are two separate parts, the firmware has to make sure that they work together. This means that when something that is displayed on the (LCD) screen, is pressed (on the touch panel), something has to happen.

The press of a button, a directory, an input field, etc. will give the user visual feedback and could also trigger some other action. For example, clicking the ‘Start’ button when a file is selected will result in the button changing color on pressing it and changing the color back to the original color when releasing it as well as changing the screen and moving the machine.

The next section describes bow a button press is handled.

### 3.4.1 Pressing a button

Information about the buttons is stored in the microcontrollers memory. At first, the buttons on the screen. After that, an endless loop is entered. This ‘event’-loop checks all of the buttons. Checking the buttons is done in the following way:

The location where the screen is being pressed is compared to the coordinates of the buttons that screen has. If the screen is being pressed on a spot where a button is located, the action associated with the button press is performed. In other words, if the coordinates of the pressed spot are withing the coordinates of a button, the event corresponding to that button is triggered. You can see how this works in figure 3.11.

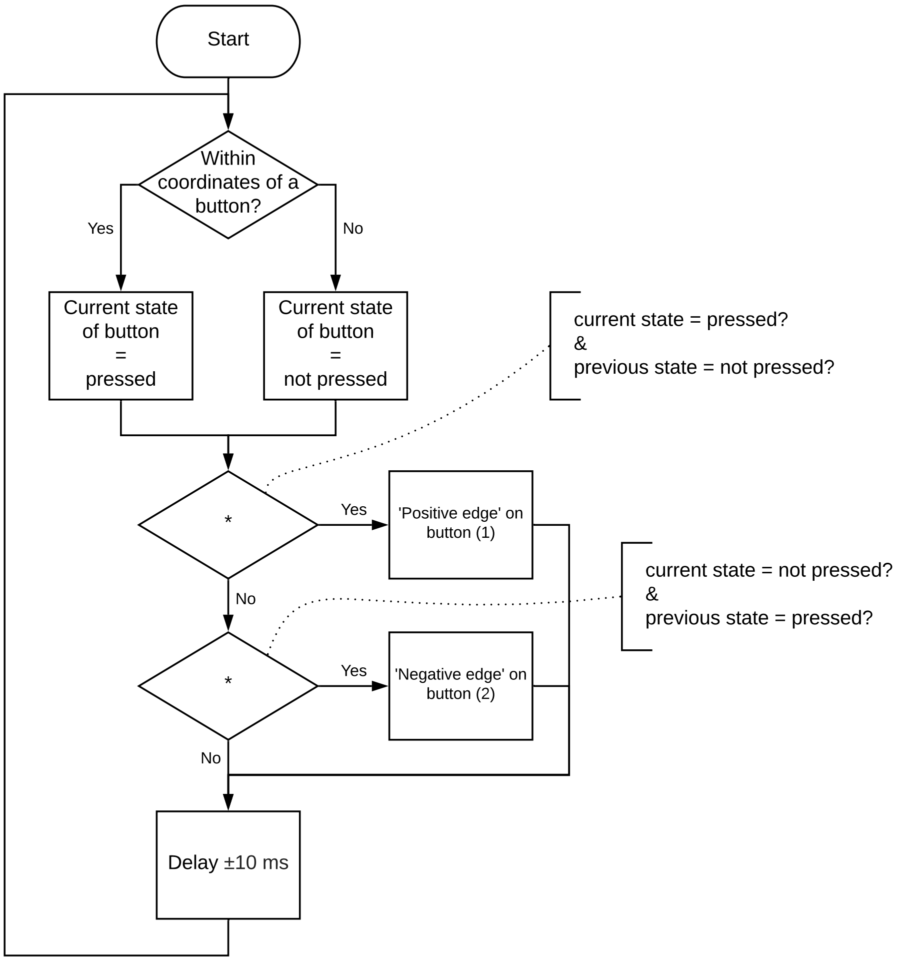
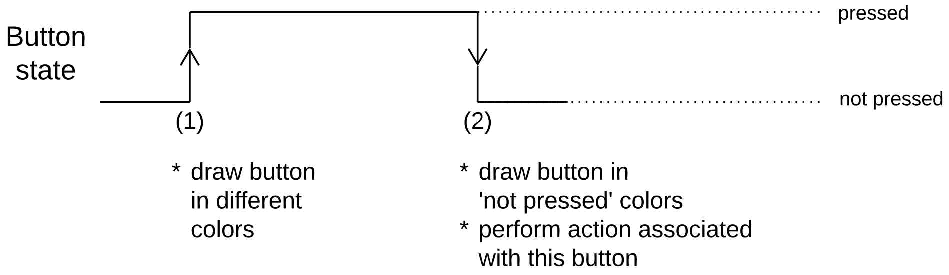

Start
Within coordinates of a button?
Yes
No
Current state of button = pressed
Current state of button = not pressed
current state = pressed?
&
previous state = not pressed?
*
Yes
'Positive edge' on button (1)
No
current state = not pressed?
&
previous state = pressed?
*
Yes
'Negative edge' on button (2)
No
Delay ±10 ms

Figure 3.11: Touching a button

When the button is touched, a certain action can be given to that event. Figure 3.12 shows the principle of how most of the buttons in this project perform some kind of action. This way of controlling a button is flexible, as it allows the programmer to perform an action exactly when he wants. Functions van be called on touching (1) or releasing (2) the button. These functions will also only be called once, not every time the check is performed when the button is pressed.



Button state
pressed
not pressed
(1)
(2)
* draw button in different colors
* draw button in 'not pressed' colors
* perform action associated with this button

Figure 3.12: Perform an action when button is touched

### 3.4.2 Checking if a spot is touched

As you have read in the previous section and as you can see in figure 3.11, a check is performed to see if a button is touched. This function is quite simple, as it just compares coordinates. The coordinates of the spot that is pressed on the screen are compared to coordinates of the buttons that that screen has. If the spot pressed is inside of an area where a button is located, that button is considered ‘pressed’. An error margin is added to make sure that the user does not have to hit a precise spot.

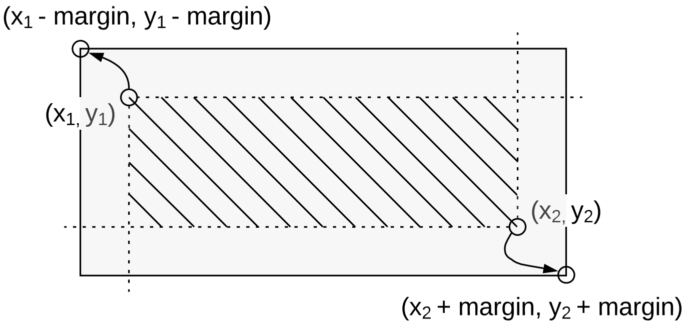


Figure 3.13: Error margin

In firmware this functionality is achieved by first requesting the coordinates of the spot is currently being touched, and afterwards checking if the coordinate is bigger than the left(x1) or top(y1) coordinate minus the margin, and checking if value of the coordinate of the spot is smaller than the right(x2) or bottom(y2) coordinate plus the margin. If these conditions are met, the function returns ‘TRUE’, which indicates that the user is touching a spot within the given area. Else it returns ‘FALSE’, which of course means that the area is not touched.

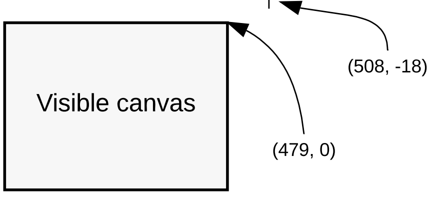


Figure 3.14: ‘Touch’ in idle state

When the screen is not pressed, ‘a touch’ (which is translated to coordinates) is constantly registered. This is a result of not being able to use the interrupt functionality of the touch controller. This because of a lack of IO pins at the microcontroller.
In the idle state (when the screen is not touched), coordinates are still being read, but they are outside of the visible canvas (shown in figure 3.14).

## 3.5 Control the CNC machine

The output of the parallel port is controlled by the I/O expander. The reason why the parallel port wasn't directly connected to the microcontroller is that we don't have enough I/O pins to our disposal.

In figure 3.1 we can see how the microcontroller is connected to the I/O expander and parallel port. If data needs to be written to the parallel port, that data will have to be sent to the I/O expander via SPI. The strobe pin is connected from the microcontroller to one NOT gate (for level conversion), to the parallel port.

The CNC machine has a control unit. The way the machine is controlled is by sending commands to that control unit. By hooking up a logic analyzer to the parallel port, and trying to move the machine via the application on MS-DOS, we were able to extract the commands that need to be sent to the control unit to control the machine. This process of measuring the actual voltage levels is called reverse engineering.

### 3.5.1 CNC routines

For more information about the circular buffer, please refer to section 3.5.4.

One case where we use a circular buffer is to control the CNC-machine. For example, when it is required to move the head a certain distance in one direction (read more at section 3.5.3). Commands have to be sent to the CNC-machine via the parallel port. The way the machine works is as follows:

The axes of the machine are controlled by stepper motors. In our case, one step of te stepper motor results in a movement of 0,0254 mm ($\frac{1}{1000}$th of an inch or also refered to as one thou[6] or one mill) in a certain direction. The speed at which the head moves is determined by how fast commands are sent. The commands control the machine. We have a list of commands that controls the movement in all three axes, spindle control, initialization of the machine etc.

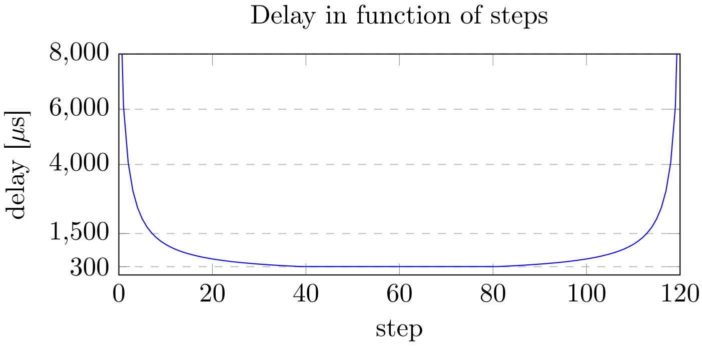
The delays between these commands are in the order of a couple hundred microseconds. The way we are able to control these rather small and precise timings is by means of a timer interrupt. For the implementation two buffers are used. Each command is accompanied by a specific delay.

### 3.5.2 Speed up and slow down

The *InitiateSpeed* function does several things. First the circular buffer is initialized. Also, an initial delay is put into the buffer so that the timer can run once. Afterwards timer 0 is initialized and configured as an interrupt. Finally we come to the 'delay initialization'.

When the head of the machine moves, we can't just tell it to go a certain speed. The machines inertia could and probably would make the stepper motors skip a couple of step. We don't want this. Very precise control over the movement of the head is required. For this reason we need to speed up and slow down all movements.

When we use the InitiateSpeed function, the given argument is the desired maximum speed. Because the speed of movement is determined by the delay in between commands, we need a small delay for fast movement and a big delay for slow movement. If you would plot these delays on a graph, it would look like the graph below. As you can see, the delay is decremented in the first 40 steps and incremented for the last 40 steps. This example was calculated for a speed of 5000 mm/min.



### 3.5.3 Move the head

One function that is central to make the CNC machine move in any direction, is the *setParam_ cmd* function. This function takes two arguments, a command and a delay. The 'command' is an opcode that makes the machine move. An example of such a command would be:

```
#define X_INC (0x4B)
```

Anywhere where X_INC is used, the value will be substituted by 0x4B and the execution of the function will result in the movement of the head in the x-axis. Every movement will call the setParam_cmd command. If this command is executed, a new entry is added to the queue that stores a pair of commands and delays. Figure 3.15 shows this.

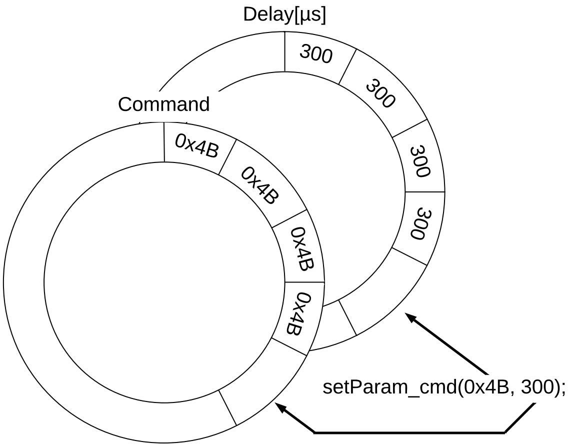


Figure 3.15: Circular buffer - setParam_cmd command

These delays and commands are then processed by a timer interrupt. An interrupt is used to achieve very precise control over timing of commands. The timer is configured in such a way that it will stop (and not restart) counting after a set time.

When the setParam_cmd function is called, the time given as an argument is configured as the time the counter will count and the command is processed. This involves setting the data pins of the parallel port to the correct value (the value depends on the action you want the machine to perform) and putting a low pulse on the strobe pin of the port. When the interrupt triggers, after the period of the previous command, a check is performed to see if there are commands to process. If so, this process of enabling the timer and processing the data is repeated with new values.

### 3.5.4 Circular buffer[7]

Different circular buffers are used in this project. A circular buffer is a data structure that uses a buffer of a fixed size, as well as two 'pointers' to indicate where to read or write data.

The buffer starts out empty. When something is written to the buffer (a command, an instruction, some data...), the write pointer needs to be increased by one, so that the next write operation can write data on a free entry of the buffer. Now assume that the buffer is totally filled with data. We don't want unprocessed data to be overwritten. That is why we also have a read pointer. When data from the buffer is processed, the read pointer gets incremented by one.

If data must now be written to the buffer, a check is first made to see if there is writable entry in the buffer. In other words, data can be written if the entry is empty or if the entries content has been processed. The read pointer needs to be behind the write pointer. If the read pointer is in front of the write pointer, the program has to wait until more data has been processed. Figure 3.16 illustrates this working.

The Wikipedia article about circular buffers [7] has an easy to understand animation that explains this functionality very well.

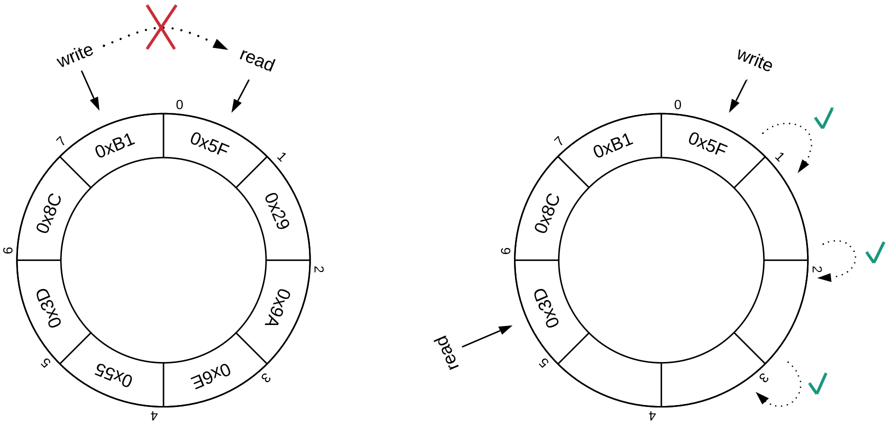


Figure 3.16: No free entries in the buffer (left) and entries can be filled (right)

## 3.6 Read SD-card

### 3.6.1 Read contents of a file

All code that has to do with low level control of the SD card is used from the Petit FAT File (PFF) System Module[8]. This code requires the SD card to have a compatible SPI mode. The PFF functions only work with SD cards that have the FAT12, FAT16 or FAT32 file system. This also means that the controller will only be able to work with SD cards that are formatted with one of the FAT file systems previously mentioned.

To read the content of a file that is located on the SD card, we need to perform the following step:

1. The first step is to *mount* a volume. This function registers a work area to the Petit FatFs module. The volume must be mounted with this function prior to use any other file function. The mount function should also be called after every media change.

2. Second of all, a file needs to be *open*ed. This makes sure that we can use the read function. The open file is valid until next open. One limitation is that we can only open one file at a time.

3. Finally, the file can be read with the *read* function. Writing to existing files or creating new files is not possible. This functionality was not implemented because it is not needed for this application.

The function that is used the most, is the *pf_ read* function. Below you can find what the meaning of each argument is. This function has been used the most because it is implemented in different ways when reading different files. More in depth information about this function can be found in the sections about ‘HPGL’ and ‘Excellon’ files (3.6.2 and 3.6.3).

```
FRESULT pf_read (
  void* buff,  /* [OUT] Pointer to the read buffer */
  UINT btr,    /* [IN]  Number of bytes to read */
  UINT* br     /* [OUT] Number of bytes read */
);
```

The read function basically fills a buffer with characters from a file on an SD card. The number of bytes that are required to be read is given as an argument as well. The last argument is a pointer to an integer that will tell the microcontroller how many bytes have actually been read.

One function that had to be made (modified from the article on the internet) is the function that shows the user all files and directories in a directory. For example, when the device is powered on and a user wants to manufacture a design, the file has to be selected via the touch screen. When first opening the window to load a file, the content of the root directory has to be shown. When the design you wish to make is located within another directory, the directory has to be selected and opened when clicked.

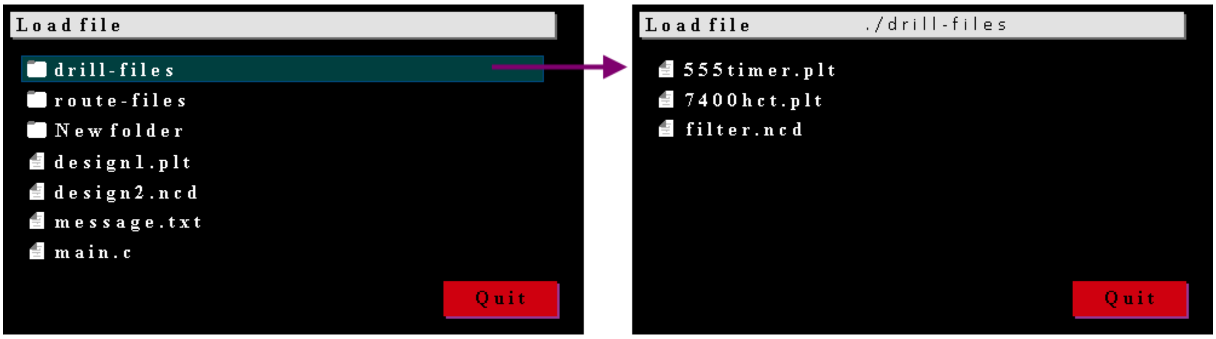


Figure 3.17: Select file procedure

With the help of some additional functions (mainly *pf_ readdir* and *pf_ opendir*), all directory entries can easily be read. The basic principle is that a directory is opened and then, with the help of a loop, all entries within that directory are stored and / or displayed.

### 3.6.2 HPGL files

Now that a file can be read, code needs to be written to process the content of the files. The first file format that needs to be processed is a 'Hewlett-Packard Graphics Language' file that has the '.plt' extension. This type of file contains commands, that are separated by a semicolon. Here is an example:

```
SP1;PU;PA4,0;PD;PA1743,658;PA1743,666;PU;PA1207,1954;PD;PA2446,65
4;PD;PU;PA2263,633;PA1056,337;PA1056,341;PA1064,341;PA1068,345;PA
1150,345;PA1158,341;PU;PA130,239;PD;PA121,247;PA105,247;PA101,251
;PU;PA101,251;PD;PA−4,247;PA−4,560;PU;PA4,349;PD;PA105,349;PA109,
353;PA125,353;PA130,357;PA138,357;PA142,361;PA150,361;PA154,365;P
A162,365;PA174,377;PA174,382;PA373,581; ...
```

We can distinguish three types of commands:

- SPn : This command means 'Set Pen'. The command indicates that the drill bit needs to be changed. The drill with which you need to replace the current drill is the 'n' drill.
- PU / PD : These commands mean 'Pen Up' and 'Pen Down'. As the meaning of these commands implicates, the head of the CNC machine needs to move up or down.
- PAx,y : This command translates to 'Plot Absolute' and gives two coordinates. The x and y coordinate are separated by a comma. These numbers are a multiple of 0,025 mm. These numbers need to be converted to an amount of steps for the machine. one step of the machine is 0,0254 mm. Using the rule of three we can find that the amount of steps the machine has to move is equal to the given coordinate multiplied by 250 and thereafter divided by 254.

```
int xsteps = ((xcoord*250)/254);
int ysteps = ((ycoord*250)/254);
moveToXY(xsteps, ysteps);
```

These commands need to be translated to movements of the CNC machine. The read function reads a part of the string and saves that part in a buffer.

To process a file, a buffer is filled with characters from the SD card. The amount of characters to be read is dependant on the amount of free space in the buffer. The buffer that characters will be put into can in fact still contain some data that should not be overwritten.

This strategy is also used to facilitate the reception of data from the WiFi module, because we don't have enough memory to store a whole file.

The function that parses the commands looks for a semicolon in a string. This value of the semicolon is saved in a variable 'i1'. The 'i1' variable represents the location of the last processed command. A second variable 'i0' contains the index of the first character of a command. The first two characters of a command are than checked to determine which command needs to be sent to the CNC machine. This function returns the location of the first character of the last unfinished command. A command is considered unfinished in the case that is was not finished by a semicolon. When a command was not processed because of being incomplete, this incomplete part will be pasted at the start of the (buffer-) array. After that the buffer is filled ('topped up') with the next characters from the file on the SD card.

This process is repeated while the file has not yet been completely processed. To check this, the amount of read characters is compared to the amount of bytes that should be read. If the last read operation was just executed, the buffer won't be totally filled up but only filled for 40% for example. Here there is a difference between requested to read bytes and actually read bytes. This indicates that all of the file has been read.

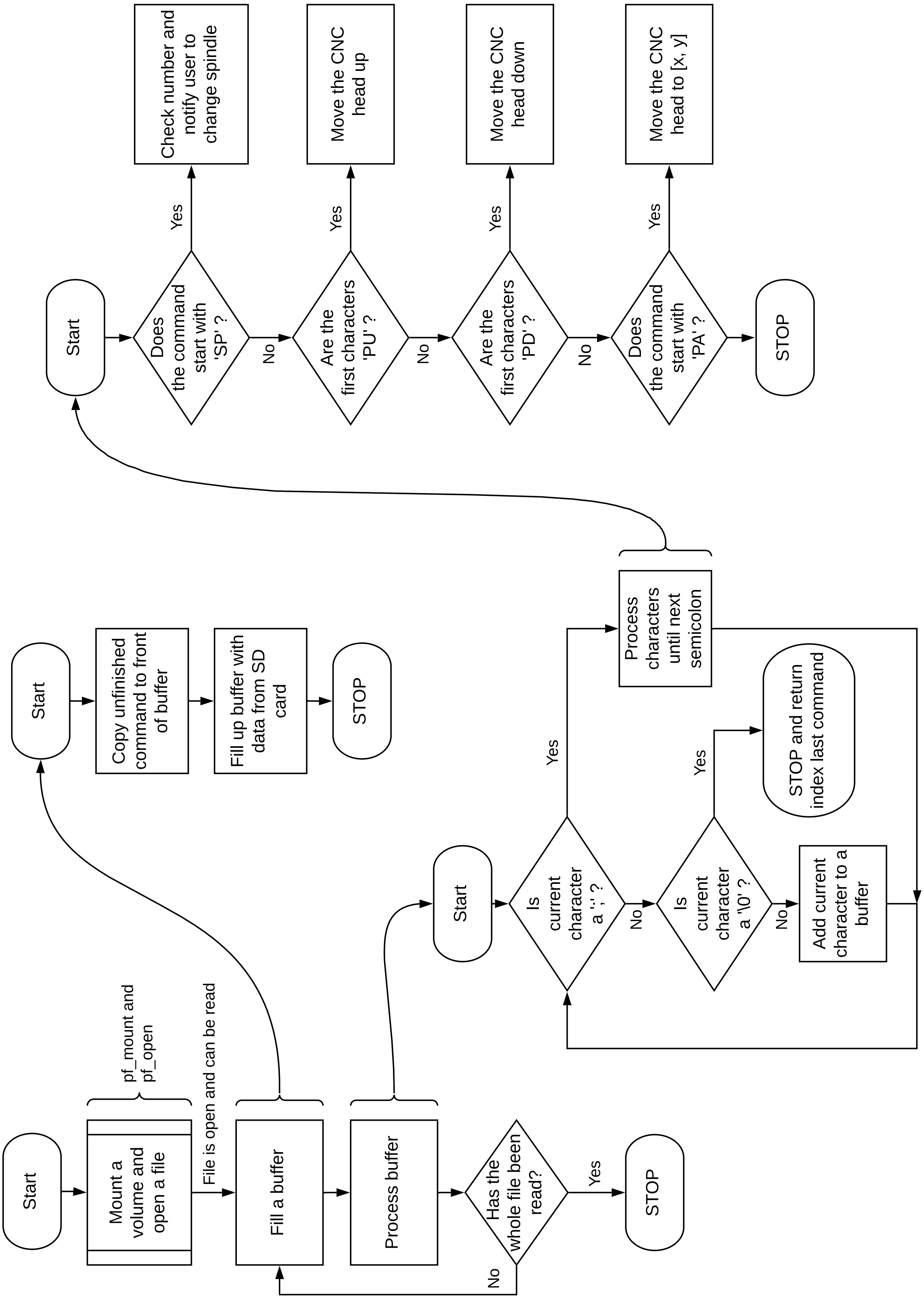

Start
Mount a volume and open a file
pf_mount and pf_open
File is open and can be read
Fill a buffer
Process buffer
Has the whole file been read?
Yes
No
STOP
Start
Copy unfinished command to front of buffer
Fill up buffer with data from SD card
STOP
Start
Is current character a ';' ?
No
Is current character a '\0' ?
No
Add current character to a buffer
Yes
STOP and return index last command
Yes
Process characters until next semicolon
Start
Does the command start with 'SP' ?
Yes
Check number and notify user to change spindle
No
Are the first characters 'PU' ?
Yes
Move the CNC head up
No
Are the first characters 'PD' ?
Yes
Move the CNC head down
No
Does the command start with 'PA' ?
Yes
Move the CNC head to [x, y]
STOP

Figure 3.18: Flowchart explaining how an HPGL file is processed

### 3.6.3 Excellon files

The excellon file has a completely different structure. The extension of an excellon file is '.ncd'. All commands are separated by an 'enter' (carriage return + new line). The file is started by a header. The header runs until a single '%' sign occurs. The header contains some general configuration for the design (for example if the quantities in the file are expressed in metric or imperial units, what diameter a tool is, etc.). Every excellon file starts with the 'M48' string and ends with 'M30'.

Below you can find the content of an excellon file.

```
M48
%
T01
X+019960Y+025640
X+019960Y−001040
M30
```

An explanation on how an excellon file gets processed follows. Because all commands are on a new line and are separated by a new line and carriage return, the file has to be read character per character. A function was made to split the file up in 'lines'. This function returns one command. After extracting one command (= one line) out of the SD card, the command can be executed. The majority of the complexity lies in the function that extracts one command out of the whole file. This function is called *readLine* and can be seen below.

```
int readLine(char *line)
{
    UINT bytes_read, i = 0;

    while(1) {
        pf_read(&line[i], 1, &bytes_read);
        if (line[i] == '\r') continue;
        if (line[i] == '\n') break;

        i++;
        if (i >= (SIZE)) return −1;
    }
    line[i] = 0;
    return 0;
}
```

The function takes a pointer to a character as an argument. Here an array will be given as an argument. Inside of an infinite loop, the array (that was given as an argument) is filled character per character. The *pf_read* function will put a character on the address that is given by the '&line[i]' parameter. This parameter represents the address of the element on the i-th index of the 'line' array. Every time a character is put into the array, a variable that represents the current iteration of the loop is incremented by one. There are two exceptions to this value iterating. Because an '.ncd' file writes each command on a new line and so places a new line and carriage return after each command, we can use this to find when a full command was read. Hence why the following is done to separate each command.

If a carriage return character is found, the loop is restarted without executing the rest of the code in the loop (so without incrementing the 'i' variable). If the current character is a new line character, the loop breaks, terminates the character with a zero and returns a 0, indicating that the command was extracted without any problem. If the character would be larger that the size of the buffer, the function would return a negative one, indicating that something went wrong trying to read a command from the SD card.

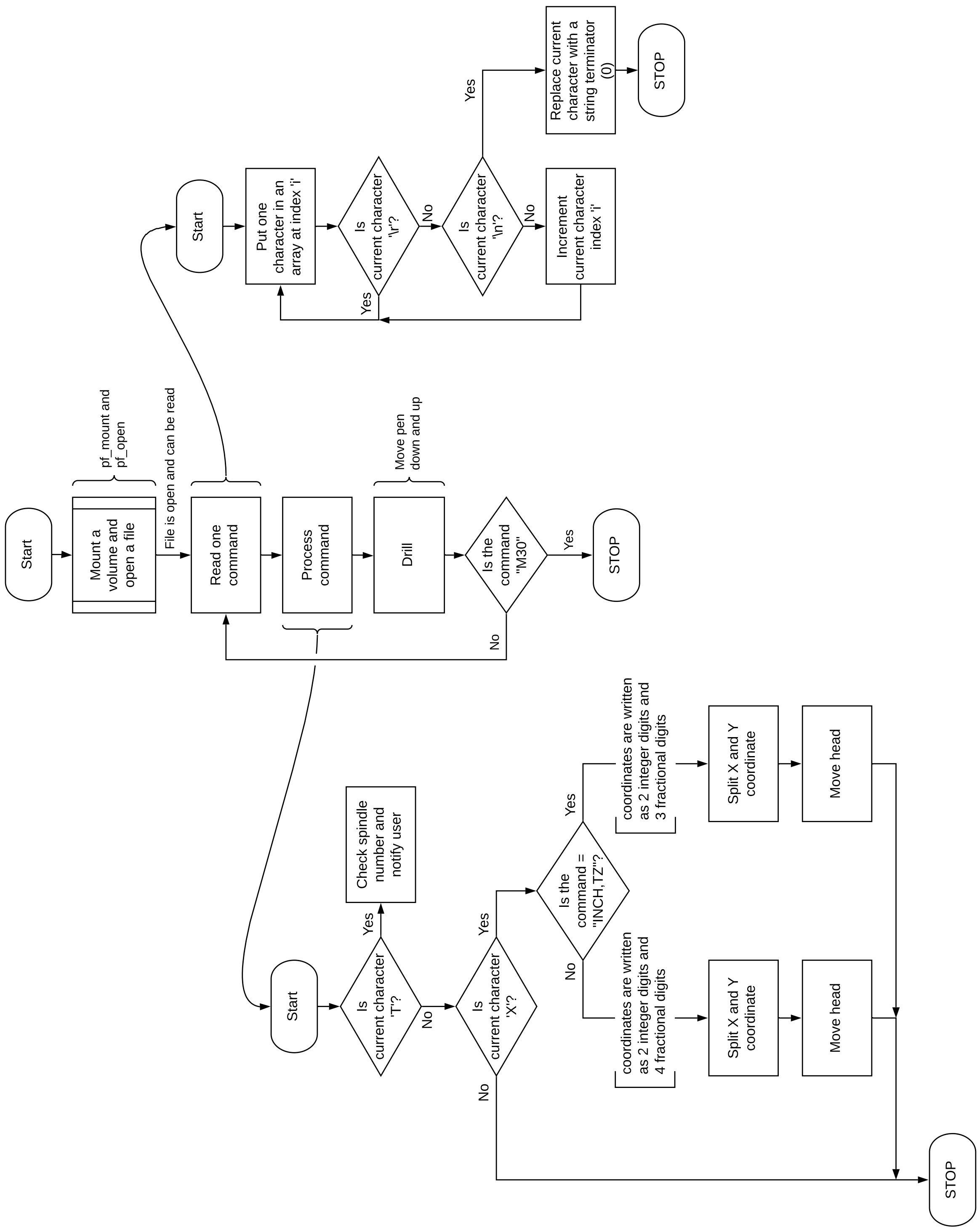

Start
Mount a volume and open a file
pf_mount and pf_open
File is open and can be read
Read one command
Process command
Drill
Move pen down and up
Is the command "M30"
Yes
No
STOP
Start
Put one character in an array at index 'i'
Is current character '\r'?
Yes
No
Is current character '\n'?
Yes
No
Replace current character with a string terminator (0)
STOP
Increment current character index 'i'
Start
Is current character 'T'?
Yes
No
Check spindle number and notify user
Is current character 'X'?
Yes
No
Is the command = "INCH,TZ"?
Yes
No
coordinates are written as 2 integer digits and 3 fractional digits
coordinates are written as 2 integer digits and 4 fractional digits
Split X and Y coordinate
Move head
Split X and Y coordinate
Move head
STOP

Figure 3.19: Flowchart explaining how an excellon file is processed

## 3.7 Emergency button

A mechanical or physical button is used as an emergency stop button. This button is connected to an external interrupt pin of the microcontroller. When the button is pressed, the movement of the head is stopped until the user confirms to restart the machine. On pressing the button, a small message box pops up and asks the user if he wants to resume the current action of the machine.

## 3.8 Communication with WiFi module

The microcontroller and WiFi module communicate with each other via UART. The microcontroller has two UART interfaces. One is used for programming the MCU, the other one is used to communicate with the WiFi module. You can also see the UART connections in figure 2.3.

### 3.8.1 Frame

To successfully transfer data between the microcontrollers, a certain frame format has to be agreed upon. This format is known by both microcontrollers and both microcontrollers send data in that format. For this application, the following frame format has been chosen (figure 3.20).

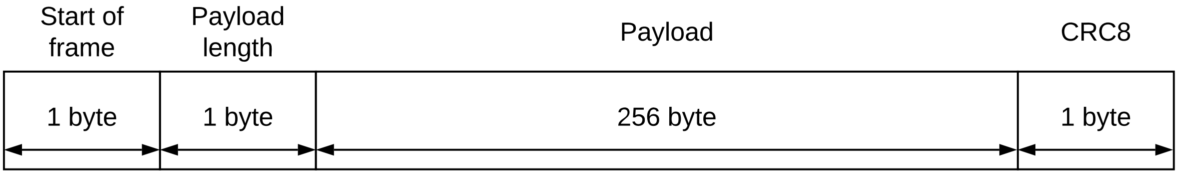


Figure 3.20: Frame format

The start of frame character that is chosen is a dollar sign, '$'. The maximum payload length per frame will be 256 bytes long. The payload can vary in structure depending on the data. An encoding technique known as byte stuffing is applied to the following parts of the frame: the payload length byte, the payload itself and the checksum. The byte stuffing function escapes characters such as the start of frame character or the escape character itself. These characters are escaped if they are present in either the payload length byte, the payload or the checksum. Escaping these characters is essential, if the payload would contain a dollar sign, the state machine used at the receiving end would see this as a start of frame character. More information about the state machine can be found at section 3.8.4.

An error-detection method is also added, to detect changes in raw data between transmitter and receiver. This feature is implemented to check if no data was lost or changed during transmission. If the received data is incorrect, the microcontroller can discard that frame and request to re-transmit that frame. The polynomial that is used is the 'CRC-8-CCITT' polynomial and its normal representation is 0x07.

### 3.8.2 Payload structure

Different payload formats have been defined. Figure 3.21 shows you an example. As you can see, the payload starts with two identification bytes. One the first byte contains information about whether the payload contains data about a file or if the payload has information about a screen. In the case of the example, the payload is filled with data about the tools page. Here the first byte would tell the program that data about a screen is incoming. The second byte tells the program what screen is described. All screens get a number. This is followed by the data on each screen.

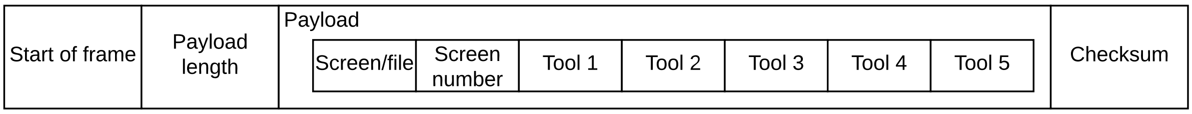


Figure 3.21: Frame filled with tools data

If file data was to be received, the payload would start with a number that would let the microcontroller know that file data is coming. The second byte ('screen number') would tell the controller if a '.plt' or '.ncd' file is being sent. The following eight bytes, because these numbers are sent as integers, contain data about the current frame that is sent and about the total amount of frames that will be sent.

### 3.8.3 Transmission

To send data from the microcontroller (LPC) to the WiFi module (ESP), a function is used to build up a frame and send it. Building up a frame means sending a start of frame byte, sending the payload length, sending the payload itself and sending a checksum. When sending the payload, the program makes sure to ‘escape’ the dollar sign or the escape character itself.

### 3.8.4 Reception

The UART1 interface, which is the connected to the WiFi module, is configured to trigger an interrupt when the receive buffer of that UART interface is filled with one byte. A state machine is used to decode the incoming data. To decode the data we apply inverse byte stuffing, the result is that we end up with the original payload. Figure 3.22 shows the state machine.

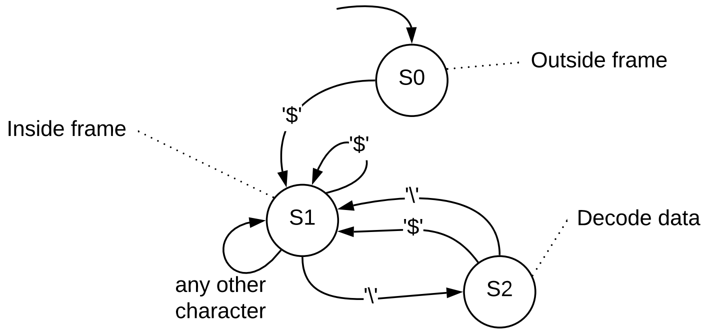


Figure 3.22: State machine that processes the incoming frame

Once the frame is completely received, the payload can be deciphered. Because data from the different screens has a fixed place in the payload, information can easily be extracted from the payload.

Much like how the frame is constructed the codes and numbers that describe the data are used to determine values for parameters. Let us continue working with the example of the tool data. Because a fixed structure has been determined, the program knows that if the first two bytes of the payload are ‘0xC3’ and ‘0x03’ (which tell us that the tools screen information is being sent), that bytes 19 to 22 describe the spindle speed of a tool, whose tool number is specified on the second byte. All payloads can be deciphered in much the same way.

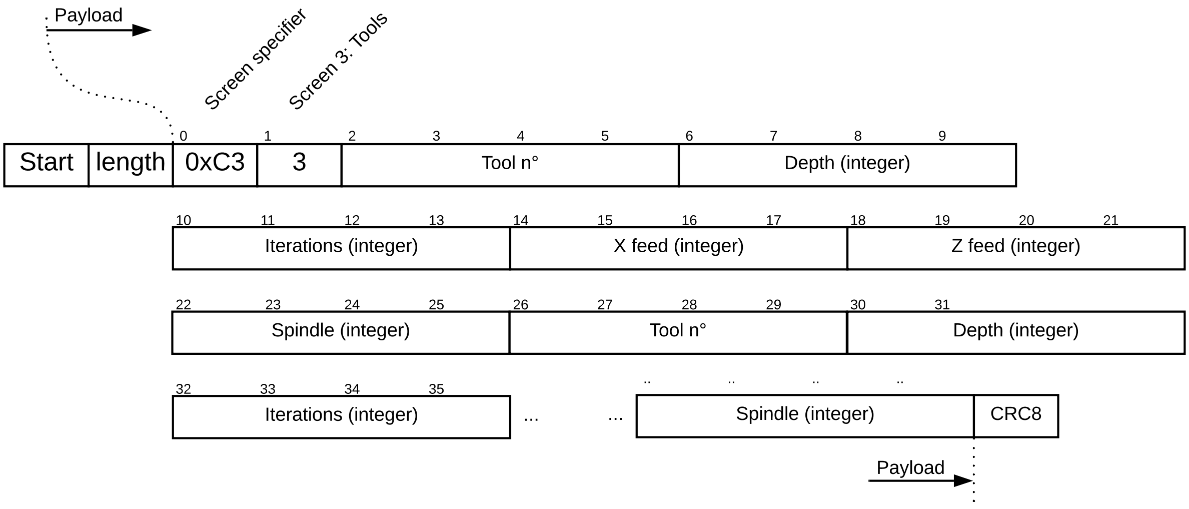


Figure 3.23: Full description of the tools payload

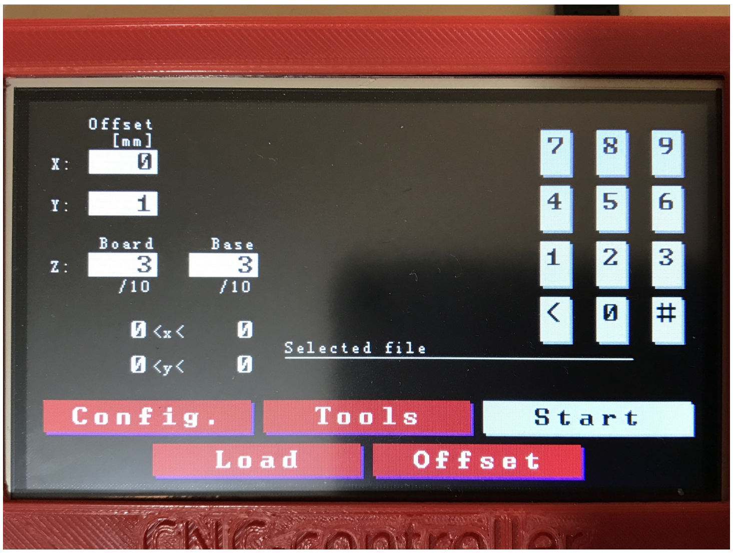
# Chapter 4

# User manual

This chapter describes how to use the controller device. To summarize, the purpose of the controller is to control the CNC machine and ultimately to mill out a certain PCB design using the machine. A design is provided either via an SD card or via the website. Several parameters can be configured with the aid of the touchscreen. In the rest of this chapter you can read an in depth description of the operating system and the meaning of the various adjustable and non-adjustable parameters. The firmware of the controller has a tree-like structure. You can open different screens from the main menu and return to the home screen.

## 4.1 Start up / main screen

When starting up the controller, a screen with the names of the creators of this project is displayed for a short amount of time. After that, you will see the main screen. You can see this screen in figure 4.1.



Offset [mm]

X: 0

Y: 1

Z: Board 3 /10 Base 3 /10

0 <x< 0

0 <y< 0

Selected file

Figure 4.1: Main screen

**Offset**

The offset parameter allows you to move the designs zero point from the machines zero point.

### Board & Base

You can enter the printed circuit boards true thickness in millimeters. This input is necessary for the program to determine the Z-axis movement of the machine and the drilling or milling depth.

The base sheet is a sheet on your machine between the PCB and the machines bed. A base board is necessary to fully drill through the PCB and not damage the bed of the machine.

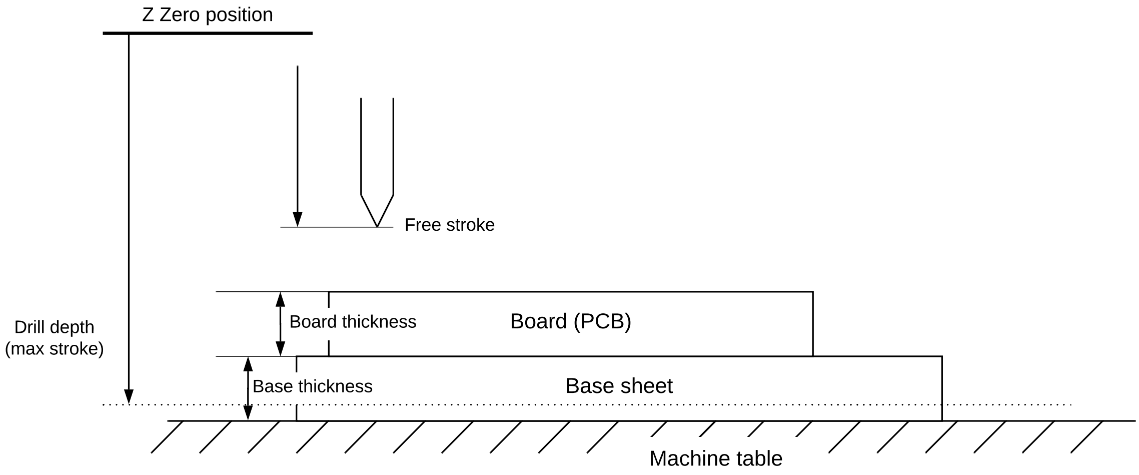


Figure 4.2: Dimensions

### Size display

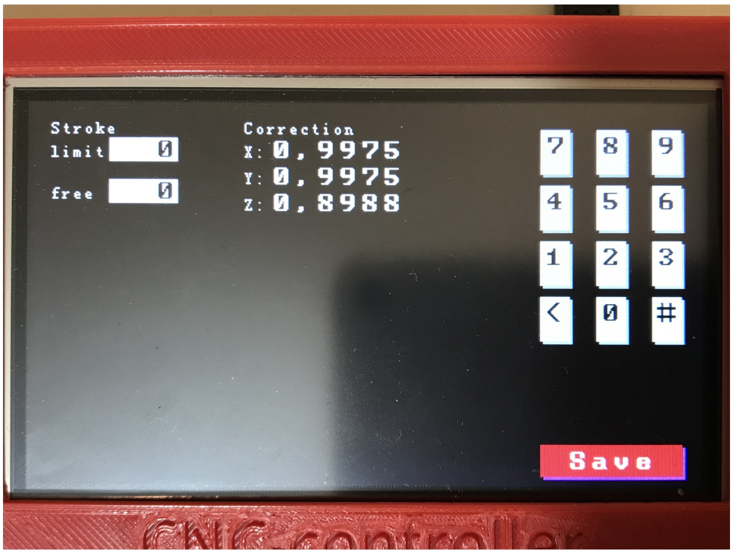
The size display, which is located in the bottom left, shows the minimum and maximum X and Y values. These values are displayed in millimeters. They basically tell you the size of the design.

### Selected file

When the machine is just powered on, the ‘Start’ button is greyed out, because no file is selected. After selecting a file from the SD card, you will see that the start button is colored like the rest of the buttons. It is now clickable. Read more about this in section 4.5.

## 4.2 Configuration

The configuration screen can be accessed by pressing the ‘Config.’ button on the main menu.



Stroke
limit 0
free 0

Correction
X: 0,9975
Y: 0,9975
Z: 0,8988

Figure 4.3: Configuration screen

### Stroke

The Z-axis stroke limit is essential for the software to calculate the correct height over the machine bed and board. The free stroke setting describes the height of the drill bit when moving above the PCB.

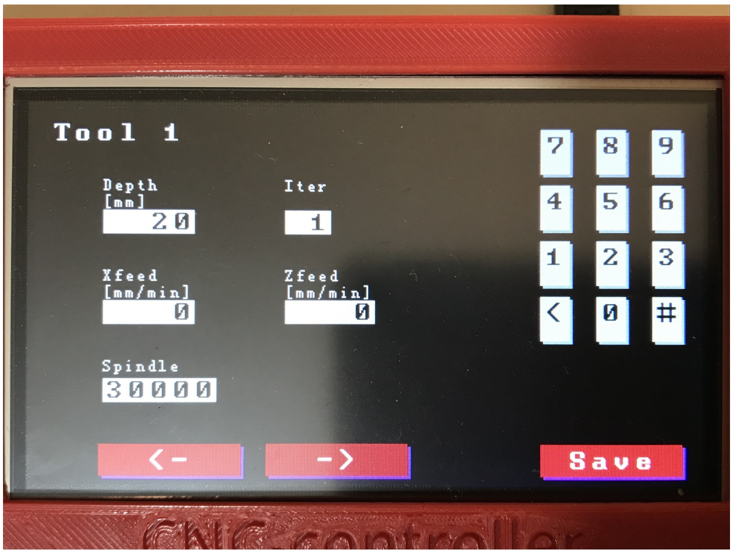
Both of these parameters are visually represented in figure 4.2.

### Correction factor

The correction factors are not changeable but they are necessary because they are related to the gearbox transmission of the spindle and the belt tension which is related to X and Y axis movement. These correction factors take into account the aforementioned problems to ensure accurate operation of the machine.

## 4.3 Tools

The tools page allows you to customize the settings of a particular tool.



Tool 1

Depth [mm] 20

Iter 1

Xfeed [mm/min] 0

Zfeed [mm/min] 0

Spindle 30000

Figure 4.4: Tools screen

### Depth

The depth parameter of a tool specifies how deep the tool will drill into the PCB.

### Xfeed and Zfeed

The X- and Zfeed parameters allow you to configure the speed at which the machine moves in the X- and Y-axis (Xfeed) as well as in the Z-axis (Zspeed).

### Iterations

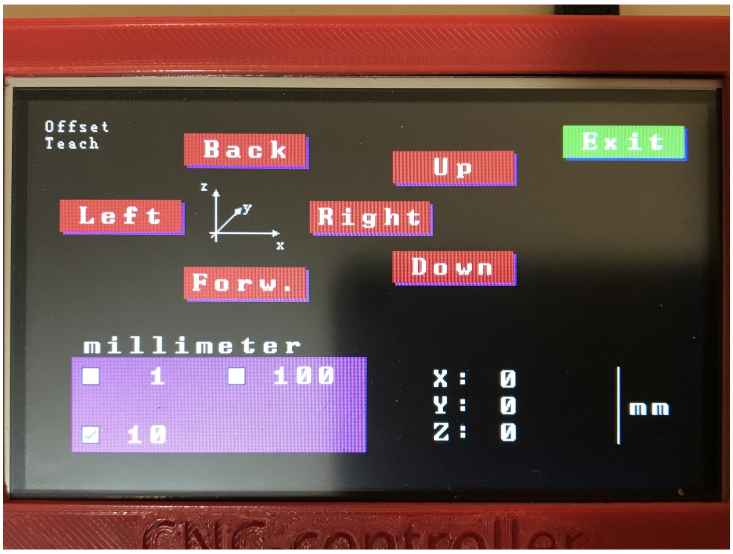
This number is relevant when milling out a board from one big sheet. The parameter indicates in how many times the board will be milled out. Each iteration, the drill will go a little bit deeper.

### Spindle

With the spindle parameter you can configure the speed at which the spindle rotates, expressed in rotations per minute.

## 4.4 Offset

Offset mode allows you to manually move the machine axes. This is useful for general testing or to determine a new board zero position, if necessary. First, select a distance. This selection determines how much the machine moves when pressing a button. The 'coordinate' of the place where the head is currently located is displayed in the bottom right.



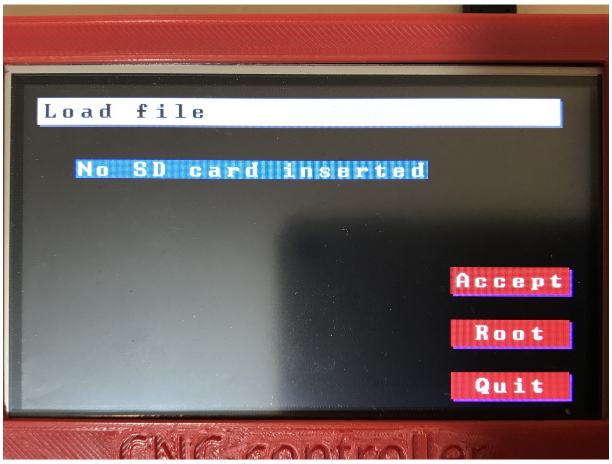
millimeter
☐ 1 ☐ 100
☑ 10

X: 0
Y: 0
Z: 0
mm

Figure 4.5: Offset screen

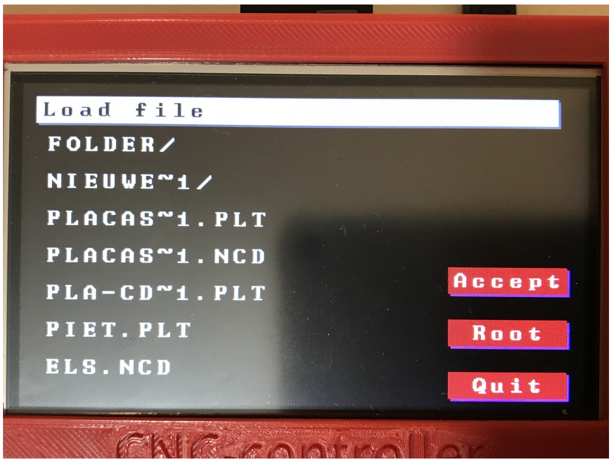
## 4.5 Load and Start

To select a file, press the ‘Load’ button. This will open a new window. If no SD card has been inserted, the controller will tell you that.


Load file

No SD card inserted

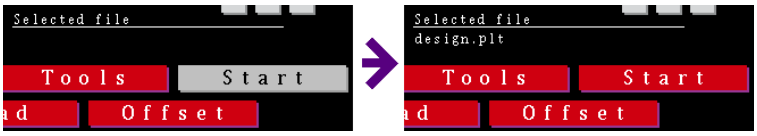
Figure 4.6: Load file screen - no card inserted


Load file

FOLDER/
NIEUWE~1/
PLACAS~1.PLT
PLACAS~1.NCD
PLA-CD~1.PLT
PIET.PLT
ELS.NCD

Figure 4.7: Load file screen - card inserted

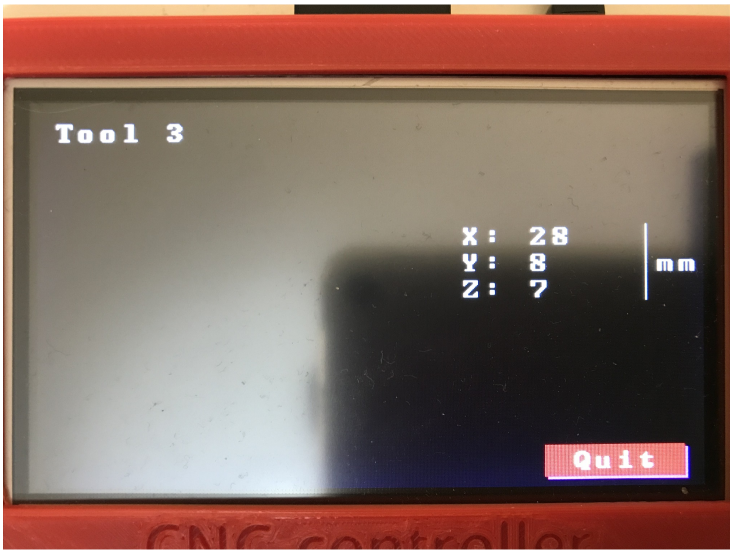
If an SD card has been inserted, you will see the contents of your SD card, but only directories, HPGL and Excellon files. To select a particular file, just press on the name. The file will be highlighted on the screen. If your file is located inside of a directory, open the directory by clicking on it. When you have selected a file you can press accept to leave this window and go back to the main menu. You will see that the ‘selected file’ field will be filled in with the file you have selected. When browsing through the directories of the SD card, you can go back to the main or root directory by pressing the ‘Root’ button. You will also notice that prior to selecting a file, the start button is greyed out and not clickable. When a file is selected, the start button is clickable.


Selected file

Selected file
design.plt

Figure 4.8: Inactive and active start button

When you want the machine to start milling out the design that is selected, press the start button. This will display a new window. Some parameters such as the current tool and the current coordinates of the head are displayed.

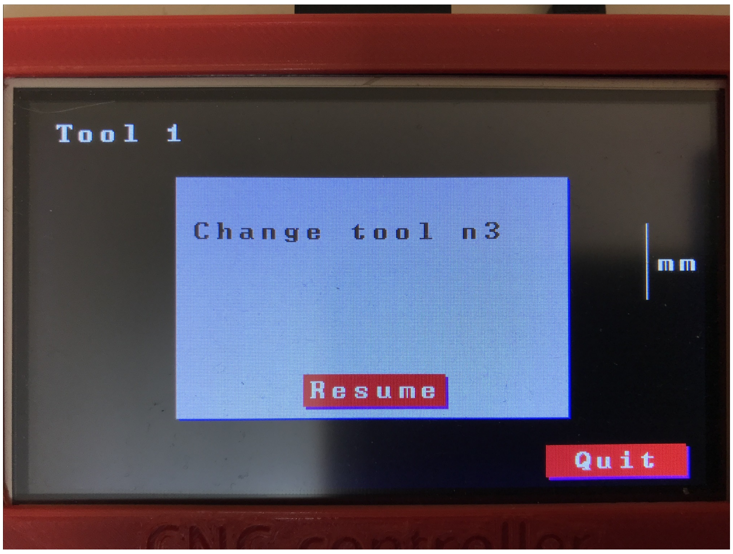


Tool 3

X: 28
Y: 8
Z: 7
mm

Figure 4.9: Machine is running

If a tool needs to be changed during the process of milling out the design, a pop-up screen will notify you to change the tool. The head will stop moving and the spindle will stop rotating. When the tool has been changed, the user can press the resume button on the pup-up, to make the machine resume and continue routing or milling out the design.



Tool 1

Change tool n3

mm

Figure 4.10: Pup-up for tool change

# Chapter 5

# Cost study

## 5.1 Material

Tables 5.1 and 5.2 give you a list of the different components that are needed to make this controller. A more detailed version of this list can be found in one of the appendices.

| **Component** | **Quantity** | **Unit cost** |
|---|---|---|
| *Modules* | | |
| TFT screen | 1 | €19,74 |
| WiFi module | 1 | €2,23 |
| *Subtotal* | | €21,97 |
| *Capacitors* | | |
| 100nF | 8 | €0,04 |
| 22$\mu$F | 3 | €0,03 |
| 33pF | 2 | €0,02 |
| 10$\mu$F | 1 | €0,42 |
| *Subtotal* | | €0,87 |
| *Resistors* | | |
| 1k | 3 | €0,01 |
| 10k | 4 | €0,01 |
| 3k3 | 2 | €0,01 |
| 560 | 1 | €0,14 |
| *Subtotal* | | €0,22 |
| *Integrated circuits* | | |
| LPC2106 (microcontroller) | 1 | €10,33 |
| 3V3 voltage regulator | 1 | €0,48 |
| USB to UART converter | 1 | €2,31 |
| 1V8 voltage regulator | 1 | €0,25 |
| I/O expander | 1 | €0,83 |
| Level converter | 1 | €0,09 |
| *Subtotal* | | €14,30 |

Table 5.1: Bill of materials, part 1

| Component | Quantity | Unit cost |
|---|---|---|
| *Miscellaneous* | | |
| 40pin header | 1 | €3,25 |
| Mini USB type B | 1 | €0,44 |
| Parallel port | 1 | €0,37 |
| Micro USB type B | 1 | €0,16 |
| Ferrite bead | 1 | €0,02 |
| PCB | 1 | €1,76 |
| Slide switch | 1 | €0,24 |
| Emergency button | 1 | €0,16 |
| Clock crystal | 1 | €0,17 |
| *Subtotal* | | €6,57 |
| *TOTAL* | | €43,93 |

Table 5.2: Bill of materials, part 2

Next, the price of the case has to be calculated. The case is 3D printed in PLA (polylactide). PLA is used frequently in the 3D printing world. The price of PLA is 20 euros per kilogram. The price of the case can be calculated with this value. For more information about the case, please refer to the thesis that was developed alongside this thesis. The thesis that is being referred to is titled 'Wireless adaptation for a standalone CNC controller' and was developed by Daan Ver Eecke [9].

| Component | Weight | Unit cost |
|---|---|---|
| Top cover | 25 g | €0,50 |
| Bottom part | 98 g | €1,96 |
| Slide switch | 1 g | €0,02 |
| Stop button | 5 g | €0,10 |
| *TOTAL* | | €2,58 |

Table 5.3: Cost of the case

Because this project is intended as a replacement controller for the CNC machine, the price of the CNC machine is not included.

## 5.2 Labour

Alongside the material cost, the labour cost has to be considered. It is impossible to give a precise price. Many factors influence the price of development of a project such as this one.

Firstly, we cannot assume that we can simply multiply the net pay of an embedded software developer by the total time it took to develop the project. The gross wage is higher. If a company were to develop this product, that price would be even higher than just the gross wages of employees. To nevertheless give an idea, the price calculation is done with the gross salary of an employee.

The gross wage also depends on how many years of experience the person has. A persons salary also depends on the county in which he or she works. Your wage depends on indexation (the technique that is used to adjust income payments by means of a price index).

Different calculations were done to give a broader idea of how much development of a project like this could cost.

A starting employee without working experience and a bachelor's degree in electronics and computer science in Belgium, can expect an average salary of about 2.470,00 euros per month. Given that in Belgium a job is considered a full time job if the person works 38 hours per week. These values come to an hourly rate of €16,25 per hour (gross). Multiplying my time spent on this project by that hourly rate results in the following:

$$610 \text{ hours} * €16{,}25/\text{hour} = €9.912{,}50$$

The average salary for an embedded software engineer in Belgium is about 3.900,00 euros per month. It should also be mentioned here that an experienced software developer could complete this project in a shorter time. An experienced developer uses a microcontroller that he knows, a framework with which he is familiar, etc. Given that the time required for pre-documentation (theoretical elaboration and preparation of the project) is not taken into account, an estimate for the elaboration of a project like this could be between one and a half months and a little more than two months. For convenience, a two-month period is used for the calculation.

$$2 \text{ months} * €3.900/\text{month} = €7.800{,}00$$

Here, of course, no account is taken of the time this person has invested in the past in researching, working out some work principles and becoming familiar with a particular microcontroller and all kinds of other things like this.

## 5.3 Total

The total cost can now be calculated by adding up these values. Adding up the PCB cost, cost of the case and firmware development cost gives us the total cost.

$$€43{,}93 + €2{,}58 + €9.912{,}50 = €9.959{,}01$$

or

$$€43{,}93 + €2{,}58 + €7.800{,}00 = €7.846{,}51$$

From this it can be concluded that the development cost of a project like this can start from about 10.000 euros. Certainly if we take into account that the development cost is higher than the gross salary of the employee(s).

# Chapter 6

# Conclusions

To form my conclusion, I look back at the objectives that were set at the start of the project. The different requirements were the following:

- It must be possible to control the machine completely independently of a computer. A custom PCB needs to be made.
- The machine must be controlled using the touch panel.
- A user must be able to provide a design that he wishes to develop using an SD card.
- The machine must be able to be controlled via a website.

Now that the project is finished I can conclude that all requirements have been met.

The controller can be used completely separately from a computer. The controller must only be connected to the machine and a power supply. A custom PCB has been developed to achieve this. The process of designing the hardware for the controller went relatively well. Some design errors were made when developing the first version of the circuit board but these errors have since been corrected. The schematics and layout that can be found as appendices are updated versions, in which the errors have been fixed. The hardware works without any problems.

The touch panel forms an interface between the user and the machine. The machine can be fully controlled using the touch panel. Different parameters can be edited via the touch panel. Values can be changed, files can be selected, the machine can be controlled etc.

Controlling the machine, which means being able to move the head or turning on the rotation of the spindle at a certain speed was a process of trial and error. The timing of commands has to be very precise. Eventually, we got this working reliably.

If a user wants to have a design developed by the machine, this can be done using an SD card. With the aid of the touch panel, the user can browse through the directories on the SD card and a file can be selected to then be manufactured by the machine.

The last requirement, which is the ability to control the machine via a website, is also operational. ‘Controlling the machine’ can for example mean to move the head of the machine a certain distance in a certain direction. The machine can be controlled by the website. The website also offers the user the possibility to update and change parameters on the controller.

# Bibliography

## References

[1] Wikipedia. *ARM architecture*. Apr. 12, 2019. URL: https://en.wikipedia.org/wiki/ARM_architecture (visited on 04/22/2019).

[2] Wikipedia. *Ferrite bead*. Feb. 17, 2019. URL: https://en.wikipedia.org/wiki/Ferrite_bead (visited on 05/01/2019).

[3] *Bresenham's Line Generation Algorithm*. May 8, 2019. URL: https://www.geeksforgeeks.org/bresenhams-line-generation-algorithm/.

[4] Henning Karlsen. *UTFT library*. 2018. URL: http://www.rinkydinkelectronics.com/library.php?id=51 (visited on 03/03/2019).

[5] Faisal Tariq. *An easy-to-understand explanation of calibration in touch-screen systems*. Nov. 29, 2019.

[6] Wikipedia. *Thousandth of an inch*. Mar. 20, 2019. URL: https://en.wikipedia.org/wiki/Thousandth_of_an_inch (visited on 04/27/2019).

[7] Wikipedia. *Circular buffer*. 2019. URL: https://en.wikipedia.org/wiki/Circular_buffer.

[8] ChaN. *Petit FatFs - FAT file system module R0.03a*. Jan. 30, 2019. URL: http://elm-chan.org/fsw/ff/00index_p.html.

[9] Ver Eecke Daan. "Wireless adaptation for a standalone CNC controller". English. 2019.

[10] Wikipedia. *Bresenham's line algorithm*. Apr. 27, 2019. URL: https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm (visited on 05/01/2019).

[11] Mr. Arnab Chakraborty. *Bresenham's Line Drawing Algorithm*. May 3, 2019. URL: http://elm-chan.org/fsw/ff/00index_p.html (visited on 06/13/2018).

[12] Ezoic. *Ascii table*. URL: http://www.asciitable.com.

[13] William D. Shoaff. *How to Write a Master's Thesis in Computer Science*. Aug. 21, 2001. URL: cs.fit.edu/~wds/guides/howto/ (visited on 04/10/2019).

[14] Wikipedia. *Proteus Design Suite*. Nov. 22, 2018. URL: https://en.wikipedia.org/wiki/Proteus_Design_Suite (visited on 04/10/2019).

[15] *LPC2104/2105/2106 User manual*. Apr. 8, 2009. URL: https://www.nxp.com/docs/en/user-guide/UM10275.pdf (visited on 04/18/2019).

[16] Herbert Schildt. *C++: The Complete Reference, Fourth Edition*. DOI: 10.1036/0072226803.

[17] Directie Onderwijsbeleid VIVES. "Richtlijnen voor de vormgeving van het eindwerk". Dutch. Jan. 2016.

## Data Sheets

[18] *300 mA, Low Dropout Voltage Regulator with On/Off Control*. Version 16. MC33375/D datasheet. ON Semiconductor. Oct. 2013.

[19] *800 mA Fixed Low Dropout Positive Regulator*. TC2117 datasheet. Microchip Technology Incorporated. 2010. ISBN: 978-1-60932-563-3.

[20] *Right Angle PCB Mount D-Sub Connectors 5504F1 Series*. Version 1. 5504F1-25S-02-03-F1 datasheet. Premier Farnell group of companies. July 31, 2014.

[21] *PASS-THROUGH SOCKET*. 40 pin header datasheet. Samtec.

[22] *Hex inverter*. Version 5. 74HC04; 74HCT04 datasheet. NXP Semiconductors N.V. Nov. 27, 2015.

[23] Ferrite bead datasheet. Fair-Rite Products Corp.

[24] *FT232R USB UART IC*. Version 2.11. FT232R datasheet. Future Technology Devices International Ltd. Apr. 2015.

[25] *Quartz Crystal Specification*. HC49/4HSMX datasheet. IQD.

[26] *16-Bit I/O Expander with Serial Interface*. MCP23017/MCP23S17 datasheet. Microchip Technology Incorporated. 2017.

[27] *Micro USB type B connector*. Version D. 10118194-0X0XLF datasheet. Amphenol. May 16, 2017.

[28] *JS Series Sub-Miniature Slide Switches*. JS202011AQN DPDT datasheet. C&K.

[29] CK Ong Skip Osgood and Rick Downs. *TOUCH SCREEN CONTROLLER TIPS*. Burr-Brown. Apr. 2000.

[30] *LPC2104/2105/2106 User manual*. Version 2. NXP B.V. Apr. 8, 2009.

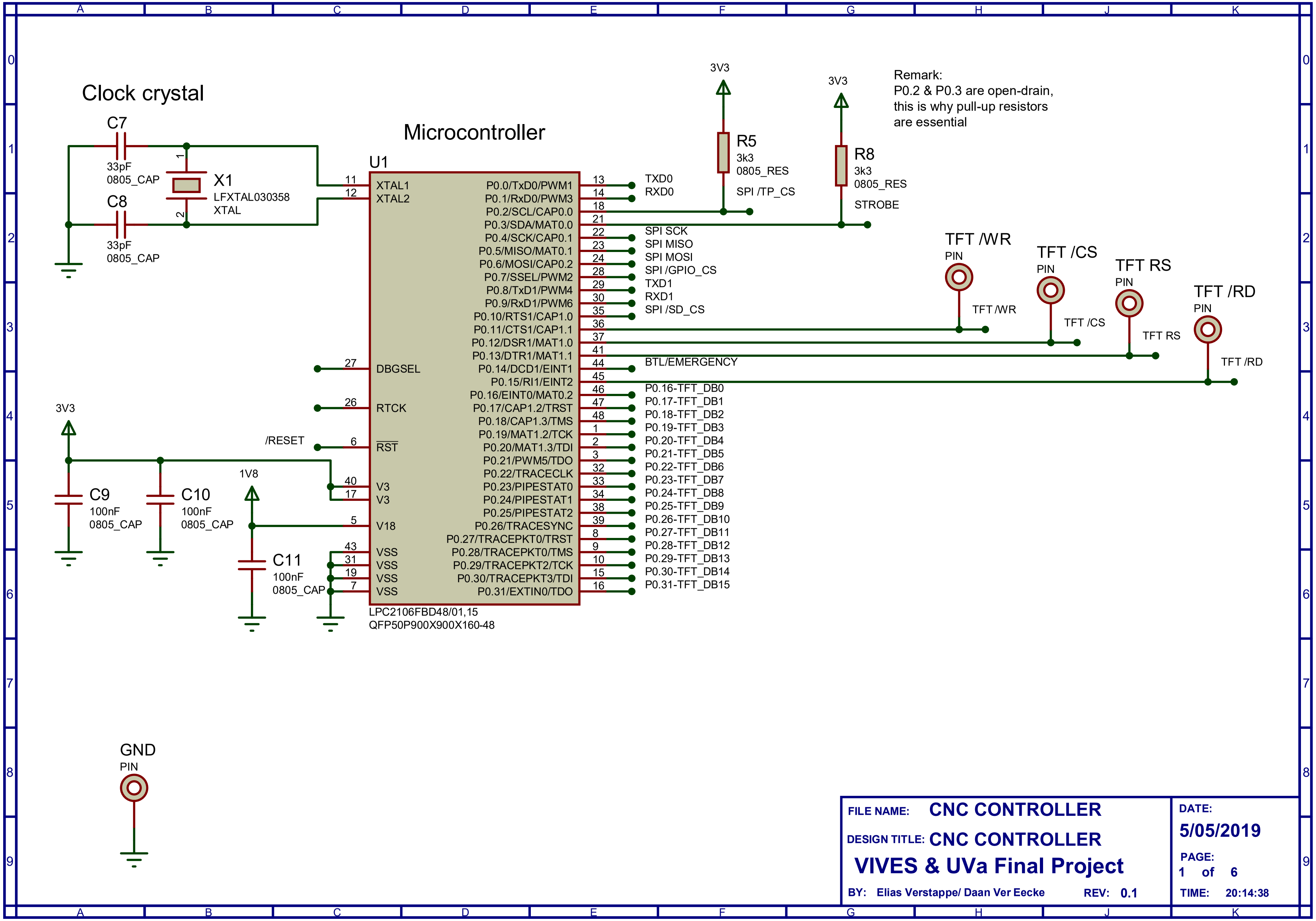
[31] *Micro USB type B*. molex. Oct. 26, 2018.

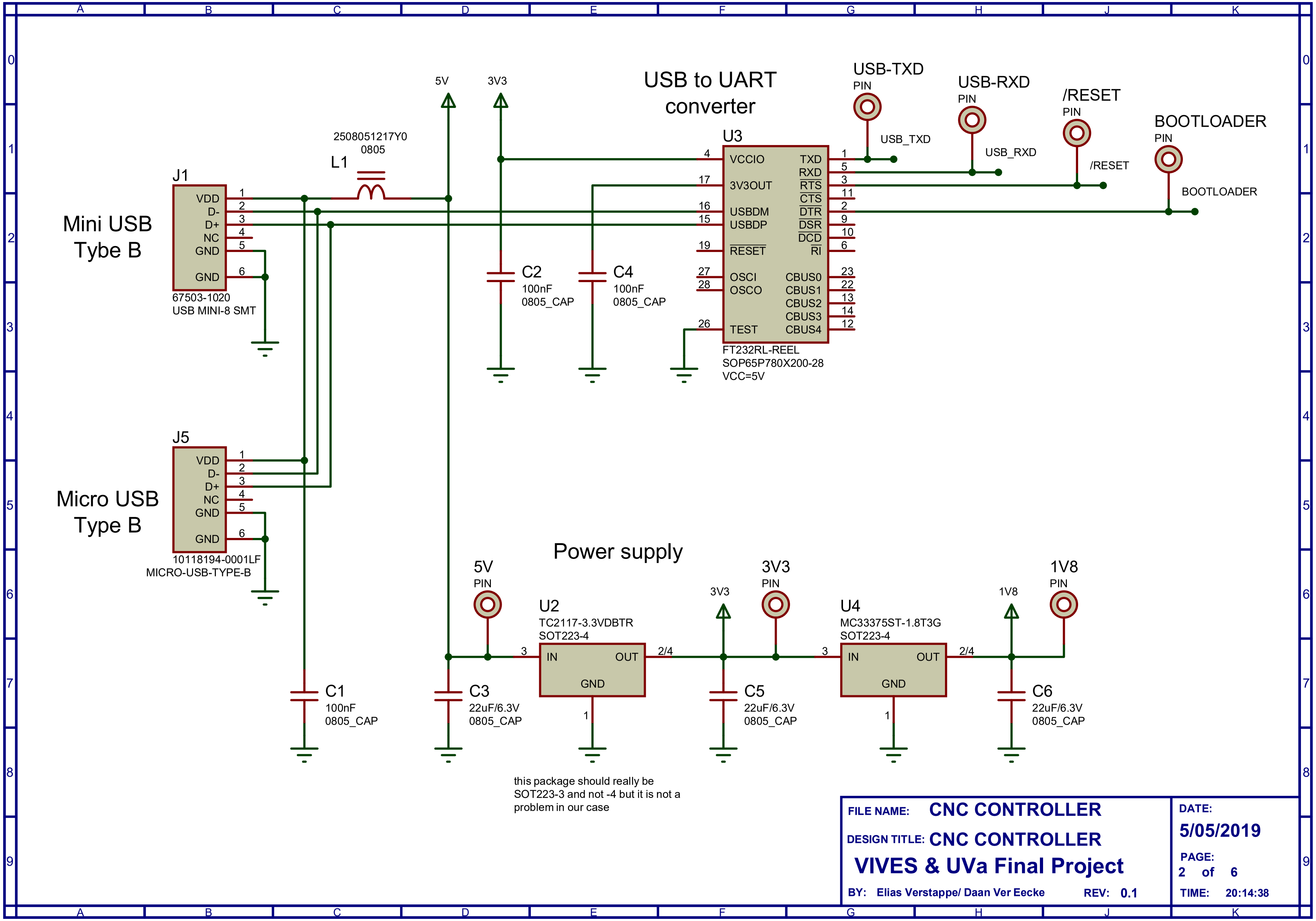
[32] *XPT2046 Data Sheet*. SHENZHEN XPTEK TECHNOLOGY CO.,LTD. 2007.

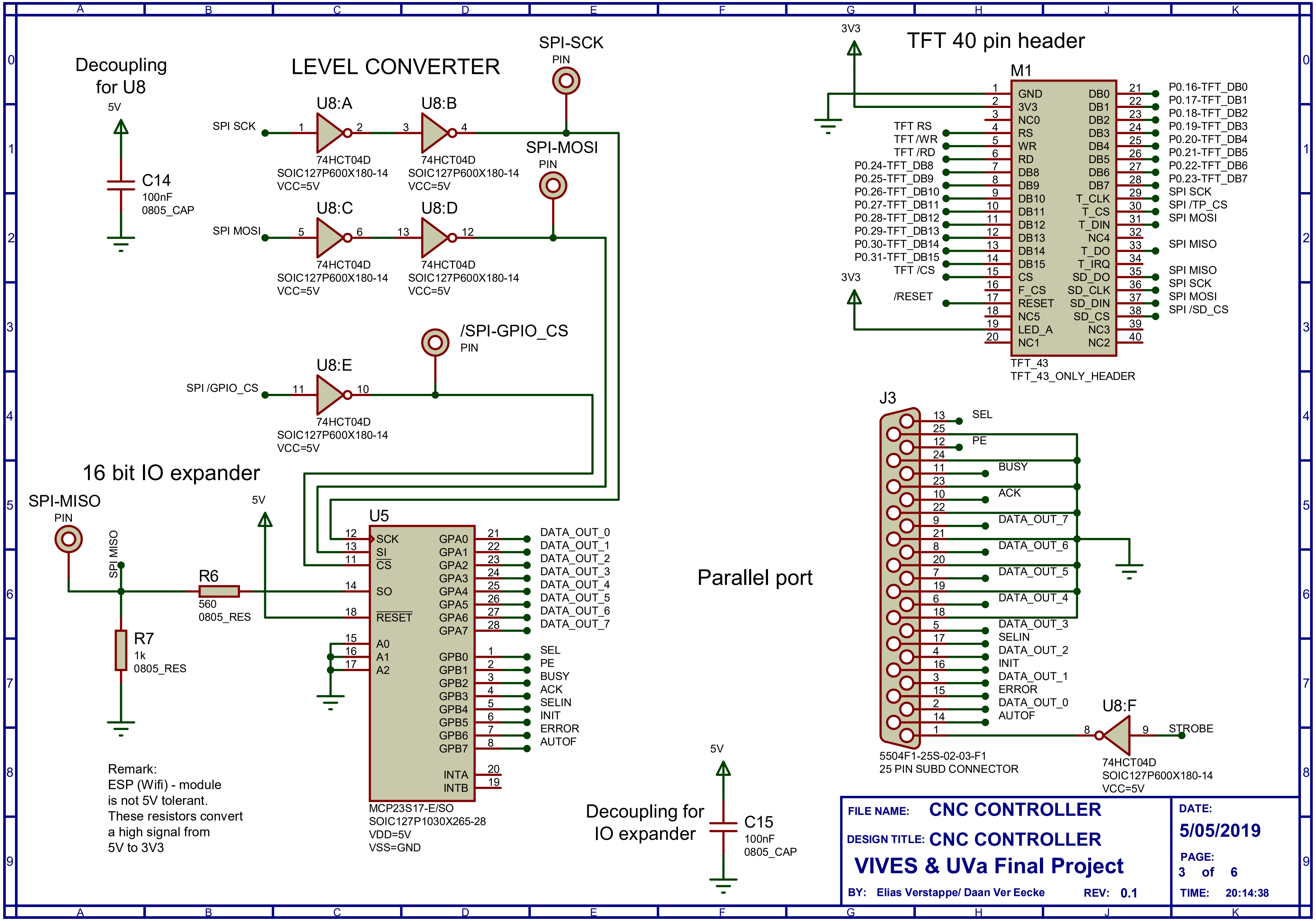
[33] *LCD Display Controller*. Version 0.20. SSD1963 datasheet. SOLOMON SYSTECH. Dec. 2008.

# Appendix A

# PCB schematic

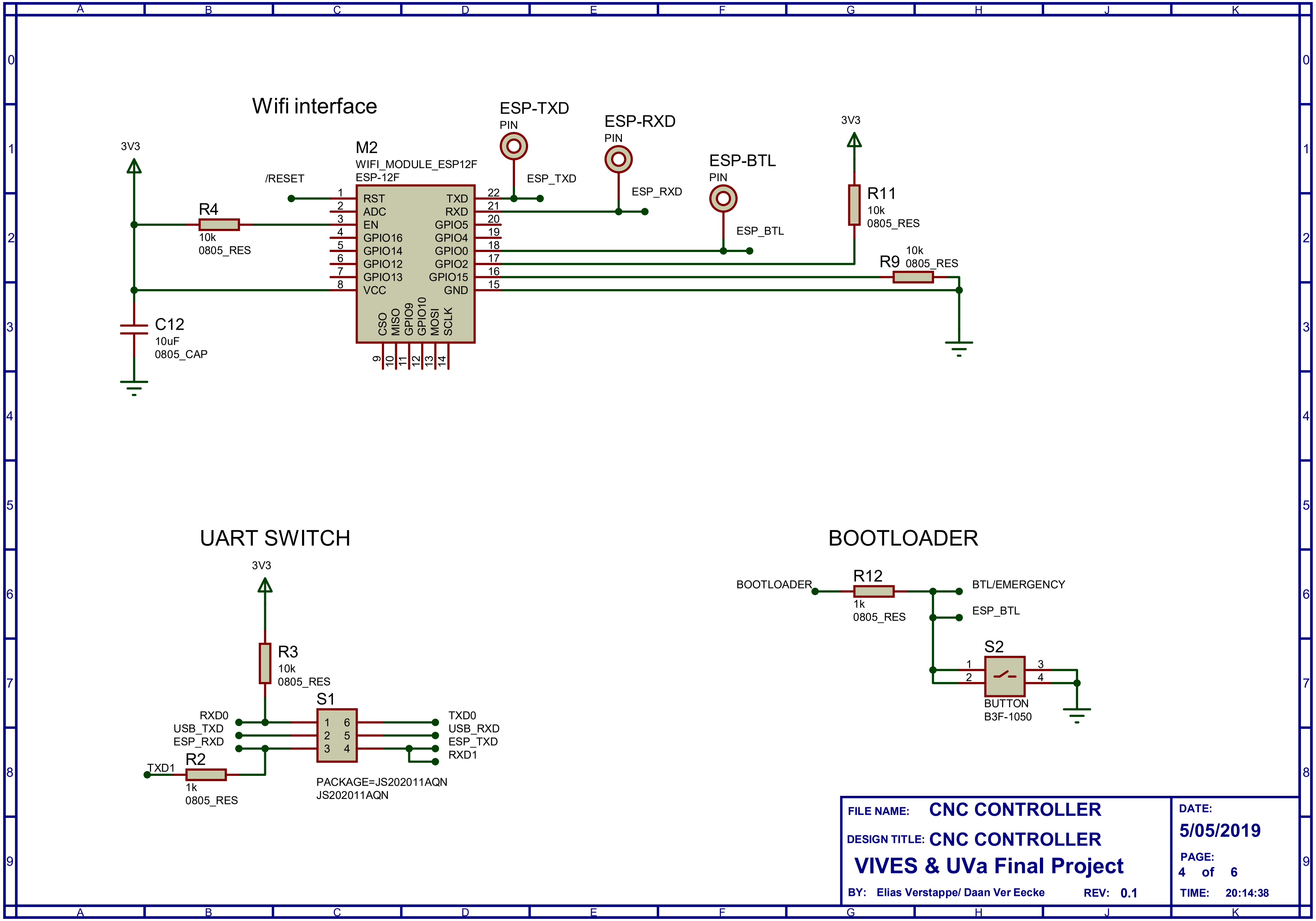

A B C D E F G H J K
0 1 2 3 4 5 6 7 8 9
Clock crystal
C7
33pF
0805_CAP
C8
33pF
0805_CAP
X1
LFXTAL030358
XTAL
1
2
Microcontroller
U1
XTAL1
XTAL2
11
12
DBGSEL
27
RTCK
26
RST
6
/RESET
V3
V3
40
17
V18
5
VSS
VSS
VSS
VSS
43
31
19
7
3V3
1V8
C9
100nF
0805_CAP
C10
100nF
0805_CAP
C11
100nF
0805_CAP
LPC2106FBD48/01,15
QFP50P900X900X160-48
P0.0/TxD0/PWM1 13 TXD0
P0.1/RxD0/PWM3 14 RXD0
P0.2/SCL/CAP0.0 18
P0.3/SDA/MAT0.0 21
P0.4/SCK/CAP0.1 22 SPI SCK
P0.5/MISO/MAT0.1 23 SPI MISO
P0.6/MOSI/CAP0.2 24 SPI MOSI
P0.7/SSEL/PWM2 28 SPI /GPIO_CS
P0.8/TxD1/PWM4 29 TXD1
P0.9/RxD1/PWM6 30 RXD1
P0.10/RTS1/CAP1.0 35 SPI /SD_CS
P0.11/CTS1/CAP1.1 36
P0.12/DSR1/MAT1.0 37
P0.13/DTR1/MAT1.1 41
P0.14/DCD1/EINT1 44 BTL/EMERGENCY
P0.15/RI1/EINT2 45
P0.16/EINT0/MAT0.2 46 P0.16-TFT_DB0
P0.17/CAP1.2/TRST 47 P0.17-TFT_DB1
P0.18/CAP1.3/TMS 48 P0.18-TFT_DB2
P0.19/MAT1.2/TCK 1 P0.19-TFT_DB3
P0.20/MAT1.3/TDI 2 P0.20-TFT_DB4
P0.21/PWM5/TDO 3 P0.21-TFT_DB5
P0.22/TRACECLK 32 P0.22-TFT_DB6
P0.23/PIPESTAT0 33 P0.23-TFT_DB7
P0.24/PIPESTAT1 34 P0.24-TFT_DB8
P0.25/PIPESTAT2 38 P0.25-TFT_DB9
P0.26/TRACESYNC 39 P0.26-TFT_DB10
P0.27/TRACEPKT0/TRST 8 P0.27-TFT_DB11
P0.28/TRACEPKT0/TMS 9 P0.28-TFT_DB12
P0.29/TRACEPKT2/TCK 10 P0.29-TFT_DB13
P0.30/TRACEPKT3/TDI 15 P0.30-TFT_DB14
P0.31/EXTIN0/TDO 16 P0.31-TFT_DB15
3V3
R5
3k3
0805_RES
SPI /TP_CS
3V3
R8
3k3
0805_RES
STROBE
Remark:
P0.2 & P0.3 are open-drain,
this is why pull-up resistors
are essential
TFT /WR
PIN
TFT /WR
TFT /CS
PIN
TFT /CS
TFT RS
PIN
TFT RS
TFT /RD
PIN
TFT /RD
GND
PIN
FILE NAME: CNC CONTROLLER
DESIGN TITLE: CNC CONTROLLER
VIVES & UVa Final Project
BY: Elias Verstappe/ Daan Ver Eecke
REV: 0.1
DATE:
5/05/2019
PAGE:
1 of 6
TIME: 20:14:38


USB to UART converter
Mini USB Tybe B
Micro USB Type B
Power supply
J1
VDD
D-
D+
NC
GND
GND
67503-1020
USB MINI-8 SMT
J5
10118194-0001LF
MICRO-USB-TYPE-B
L1
2508051217Y0
0805
5V
3V3
C2
100nF
0805_CAP
C4
100nF
0805_CAP
U3
VCCIO
3V3OUT
USBDM
USBDP
RESET
OSCI
OSCO
TEST
TXD
RXD
RTS
CTS
DTR
DSR
DCD
RI
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
FT232RL-REEL
SOP65P780X200-28
VCC=5V
USB-TXD
PIN
USB_TXD
USB-RXD
PIN
USB_RXD
/RESET
PIN
/RESET
BOOTLOADER
PIN
BOOTLOADER
5V
PIN
U2
TC2117-3.3VDBTR
SOT223-4
IN
OUT
GND
3V3
PIN
U4
MC33375ST-1.8T3G
SOT223-4
1V8
PIN
C1
100nF
0805_CAP
C3
22uF/6.3V
0805_CAP
C5
22uF/6.3V
0805_CAP
C6
22uF/6.3V
0805_CAP
this package should really be SOT223-3 and not -4 but it is not a problem in our case
FILE NAME: CNC CONTROLLER
DESIGN TITLE: CNC CONTROLLER
VIVES & UVa Final Project
BY: Elias Verstappe/ Daan Ver Eecke
REV: 0.1
DATE: 5/05/2019
PAGE: 2 of 6
TIME: 20:14:38



Remark:
ESP (Wifi) - module
is not 5V tolerant.
These resistors convert
a high signal from
5V to 3V3

FILE NAME: CNC CONTROLLER
DESIGN TITLE: CNC CONTROLLER
VIVES & UVa Final Project
BY: Elias Verstappe/ Daan Ver Eecke REV: 0.1
DATE: 5/05/2019
PAGE: 3 of 6
TIME: 20:14:38



FILE NAME: CNC CONTROLLER

DESIGN TITLE: CNC CONTROLLER

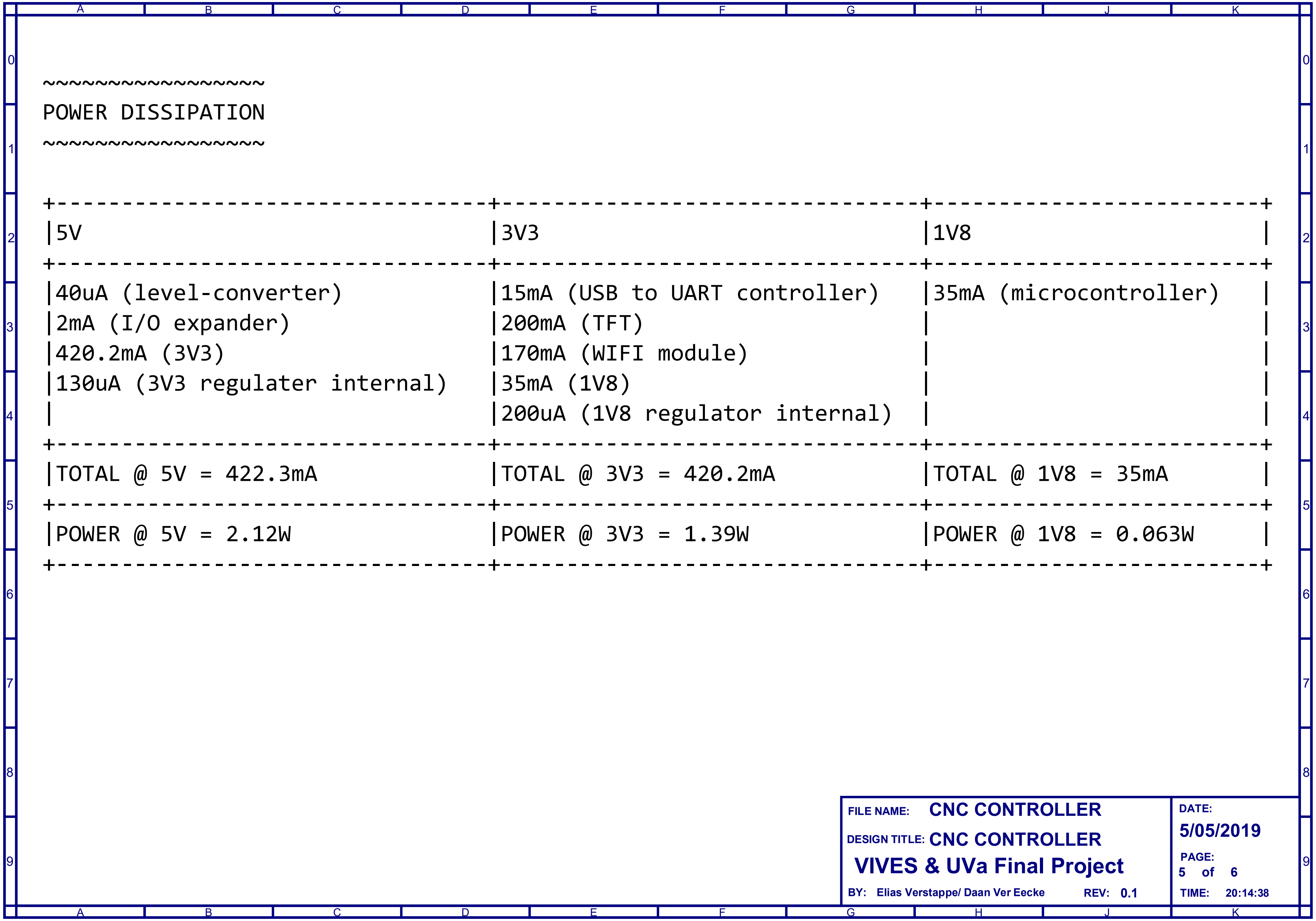
VIVES & UVa Final Project

BY: Elias Verstappe/ Daan Ver Eecke REV: 0.1

DATE: 5/05/2019

PAGE: 4 of 6

TIME: 20:14:38



```
~~~~~~~~~~~~~~~~~
POWER DISSIPATION
~~~~~~~~~~~~~~~~~

+---------------------------------+--------------------------------+-------------------------+
|5V                               |3V3                             |1V8                      |
+---------------------------------+--------------------------------+-------------------------+
|40uA (level-converter)           |15mA (USB to UART controller)   |35mA (microcontroller)   |
|2mA (I/O expander)               |200mA (TFT)                     |                         |
|420.2mA (3V3)                    |170mA (WIFI module)             |                         |
|130uA (3V3 regulater internal)   |35mA (1V8)                      |                         |
|                                 |200uA (1V8 regulator internal)  |                         |
+---------------------------------+--------------------------------+-------------------------+
|TOTAL @ 5V = 422.3mA             |TOTAL @ 3V3 = 420.2mA           |TOTAL @ 1V8 = 35mA       |
+---------------------------------+--------------------------------+-------------------------+
|POWER @ 5V = 2.12W               |POWER @ 3V3 = 1.39W             |POWER @ 1V8 = 0.063W     |
+---------------------------------+--------------------------------+-------------------------+
```

FILE NAME: CNC CONTROLLER

DESIGN TITLE: CNC CONTROLLER

VIVES & UVa Final Project

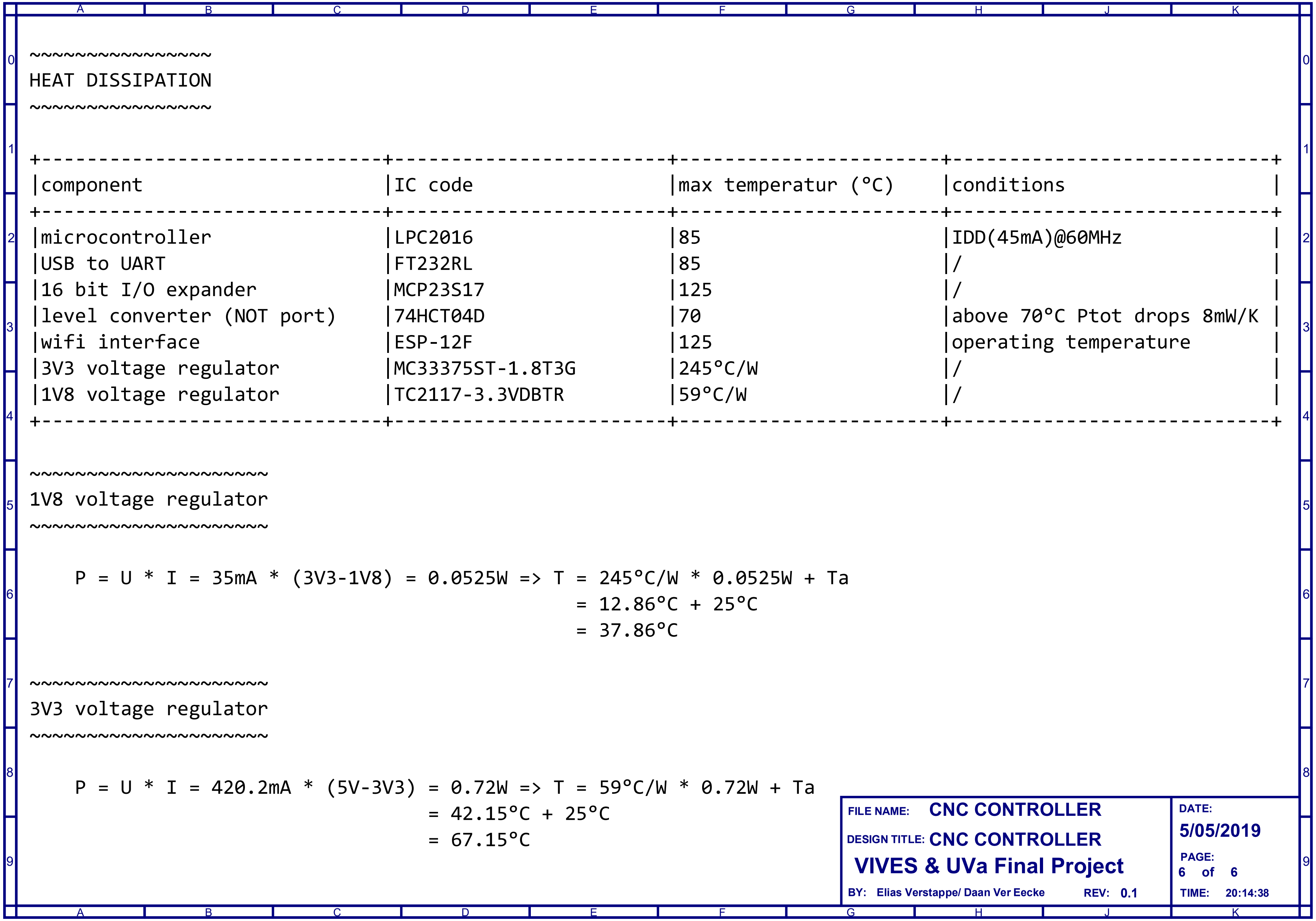
BY: Elias Verstappe/ Daan Ver Eecke

REV: 0.1

DATE: 5/05/2019

PAGE: 5 of 6

TIME: 20:14:38



~~~~~~~~~~~~~~~~
HEAT DISSIPATION
~~~~~~~~~~~~~~~~

| component | IC code | max temperatur (°C) | conditions |
|---|---|---|---|
| microcontroller | LPC2016 | 85 | IDD(45mA)@60MHz |
| USB to UART | FT232RL | 85 | / |
| 16 bit I/O expander | MCP23S17 | 125 | / |
| level converter (NOT port) | 74HCT04D | 70 | above 70°C Ptot drops 8mW/K |
| wifi interface | ESP-12F | 125 | operating temperature |
| 3V3 voltage regulator | MC33375ST-1.8T3G | 245°C/W | / |
| 1V8 voltage regulator | TC2117-3.3VDBTR | 59°C/W | / |

~~~~~~~~~~~~~~~~~~~~~
1V8 voltage regulator
~~~~~~~~~~~~~~~~~~~~~

```
    P = U * I = 35mA * (3V3-1V8) = 0.0525W => T = 245°C/W * 0.0525W + Ta
                                                = 12.86°C + 25°C
                                                = 37.86°C
```

~~~~~~~~~~~~~~~~~~~~~
3V3 voltage regulator
~~~~~~~~~~~~~~~~~~~~~

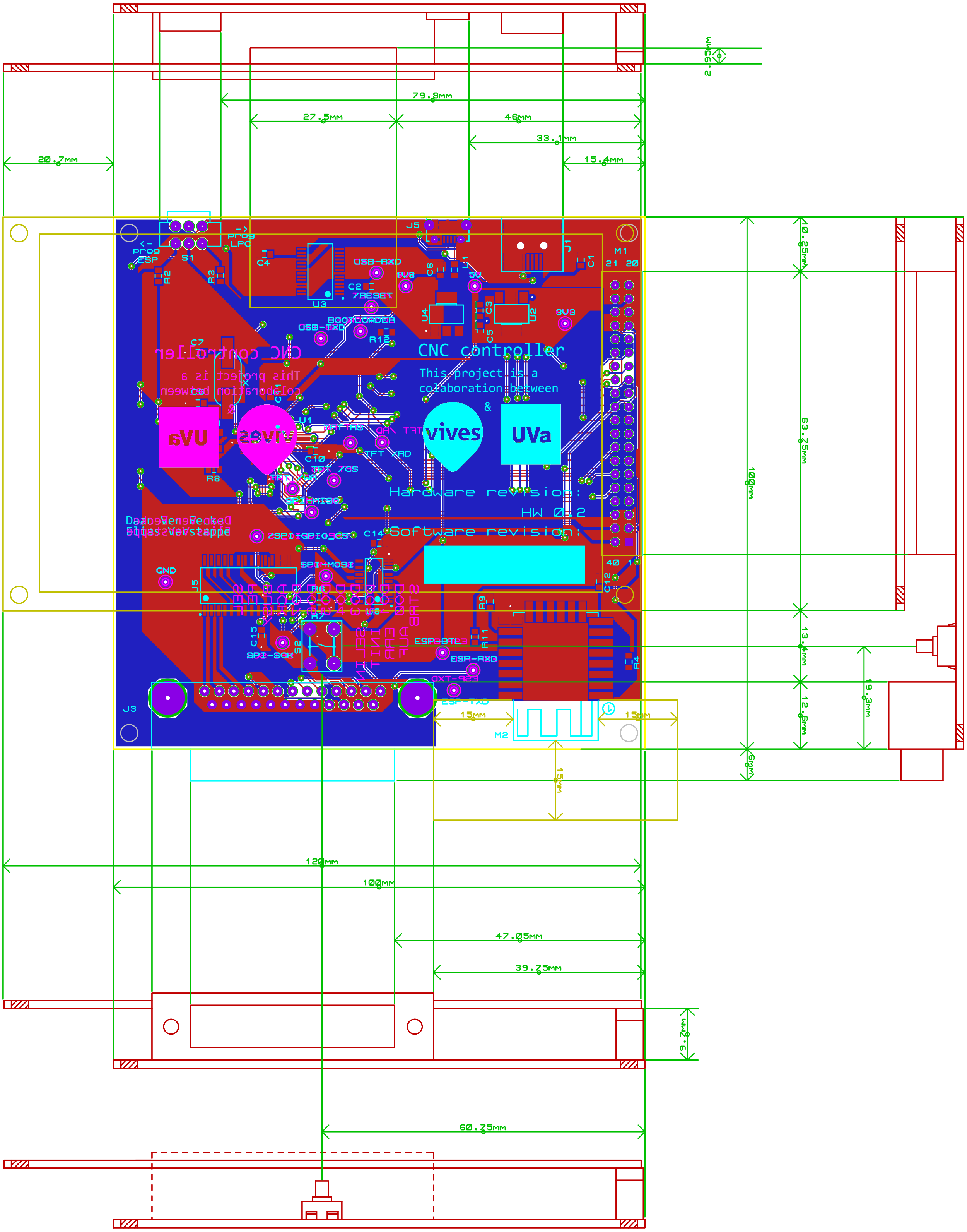
```
    P = U * I = 420.2mA * (5V-3V3) = 0.72W => T = 59°C/W * 0.72W + Ta
                                   = 42.15°C + 25°C
                                   = 67.15°C
```

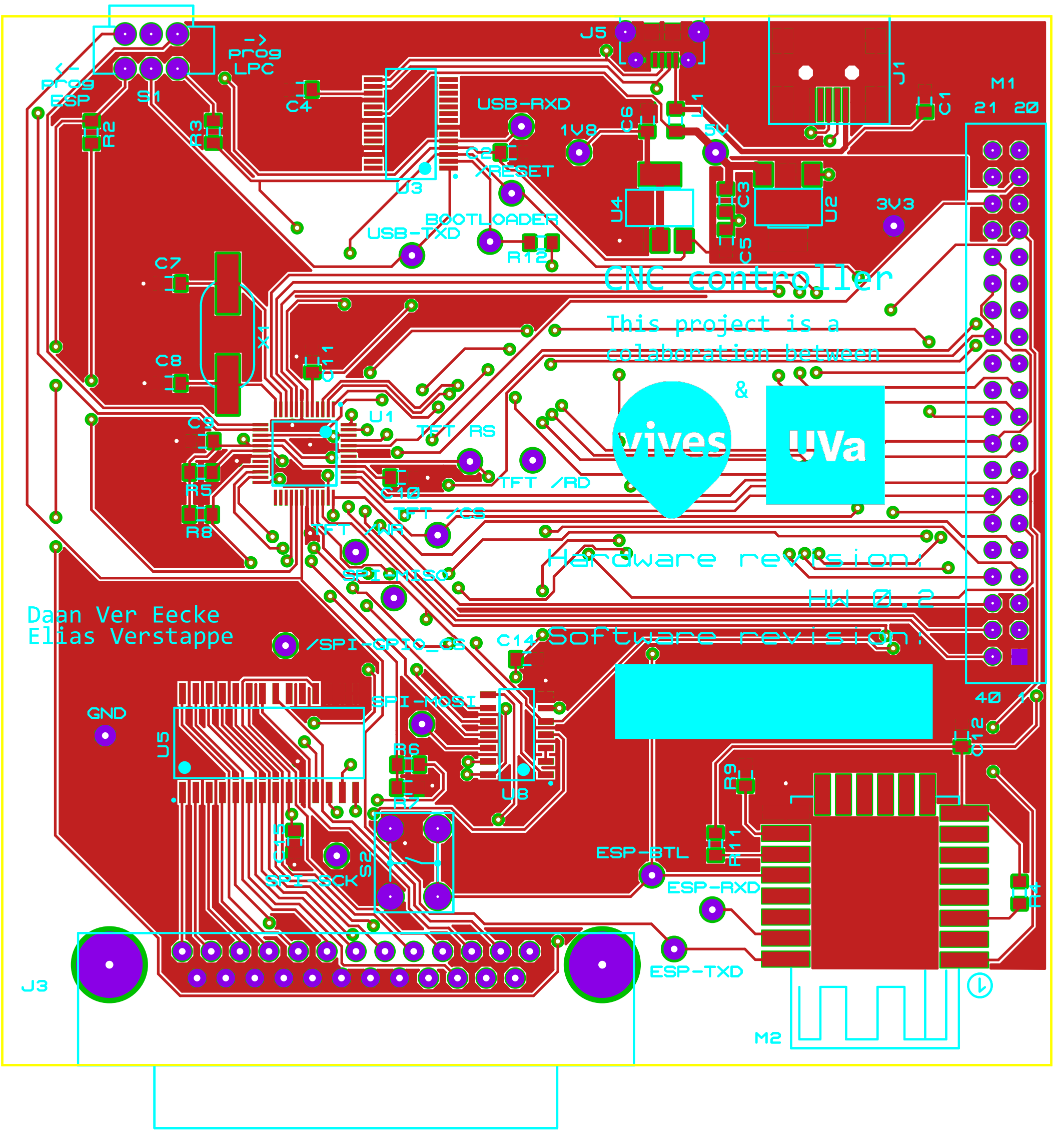
FILE NAME: CNC CONTROLLER
DESIGN TITLE: CNC CONTROLLER
VIVES & UVa Final Project
BY: Elias Verstappe/ Daan Ver Eecke REV: 0.1

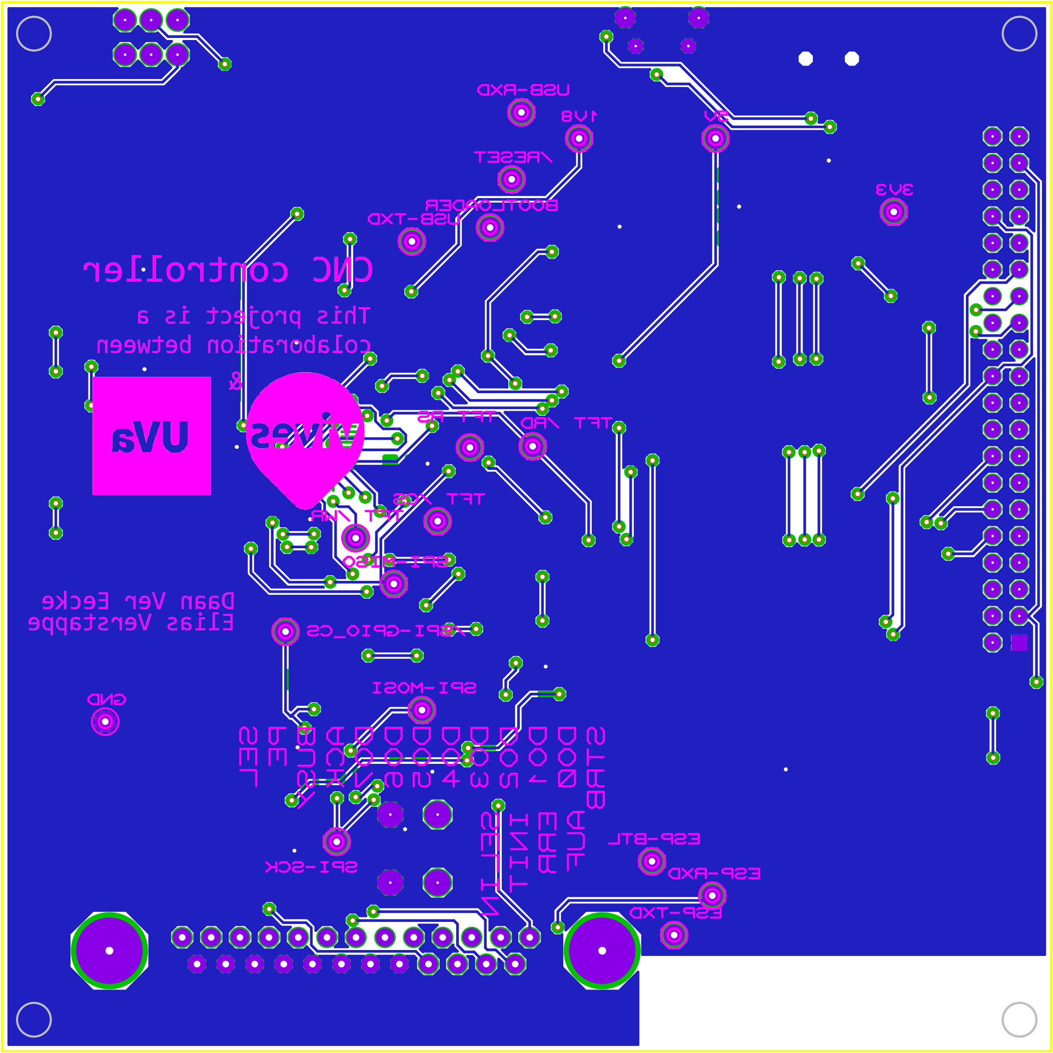
DATE: 5/05/2019
PAGE: 6 of 6
TIME: 20:14:38

# Appendix B

# PCB Layout


2.95mm
79.8mm
27.5mm
46mm
33.1mm
15.4mm
20.7mm
10.25mm
63.75mm
100mm
13.4mm
12.6mm
19.3mm
6mm
15mm
15mm
15mm
120mm
100mm
47.05mm
39.75mm
9.7mm
60.75mm
<- prog ESP
-> prog LPC
S1
J5
M1
J1
USB-RXD
USB-TXD
1V8
5V
3V3
/RESET
BOOTLOADER
U3
CNC controller
This project is a
colaboration between
&
vives
UVa
Hardware revision:
HW 0.2
Software revision:
Dean Vereecke
Elias Verstappe
/SPI-GPIO-CS
SPI-MOSI
SPI-SCK
GND
ESP-BTL
ESP-RXD
ESP-TXD
M2
J3
S2
21 20
40 1


-> prog LPC
<- prog ESP
S1
R2
R3
C4
U3
USB-RXD
J5
J1
C1
M1
21 20
1V8
C6
5V
C2
/RESET
U4
C3
U2
C5
3V3
BOOTLOADER
USB-TXD
R12
C7
CNC controller
This project is a
colaboration between
&
vives
uva
X1
C11
C8
C9
U1
TFT RS
R5
C10
TFT /RD
R8
TFT /CS
TFT /WR
Hardware revision:
SPI-MISO
HW 0.2
Daan Ver Eecke
Elias Verstappe
/SPI-GPIO_CS
C14
Software revision:
40 1
GND
U5
SPI-MOSI
R6
C12
R9
U8
R7
C15
S2
R11
ESP-BTL
SPI-SCK
ESP-RXD
R4
J3
ESP-TXD
M2


CNC controller
This project is a
colaboration between
&
UVa
vives
Daan Ver Eecke
Elias Verstappe
USB-RXD
1V8
5V
/RESET
BOOTLOADER
USB-TXD
3V3
TFT CS
TFT /RD
TFT /WR
TFT RS
SPI-MISO
SPI-GPIO_CS
SPI-MOSI
GND
SPI-SCK
ESP-BTL
ESP-RXD
ESP-TXD

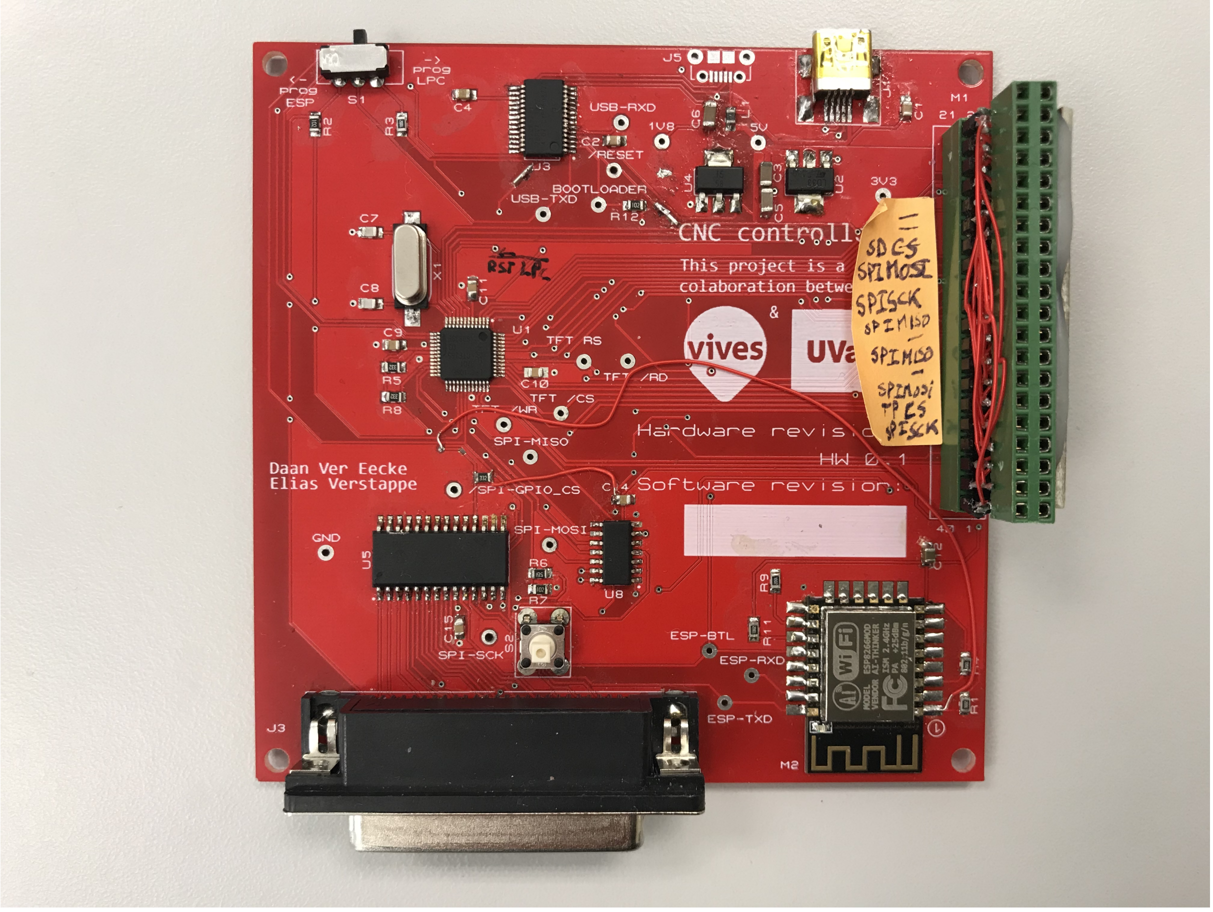


Figure B.1: Real board top view

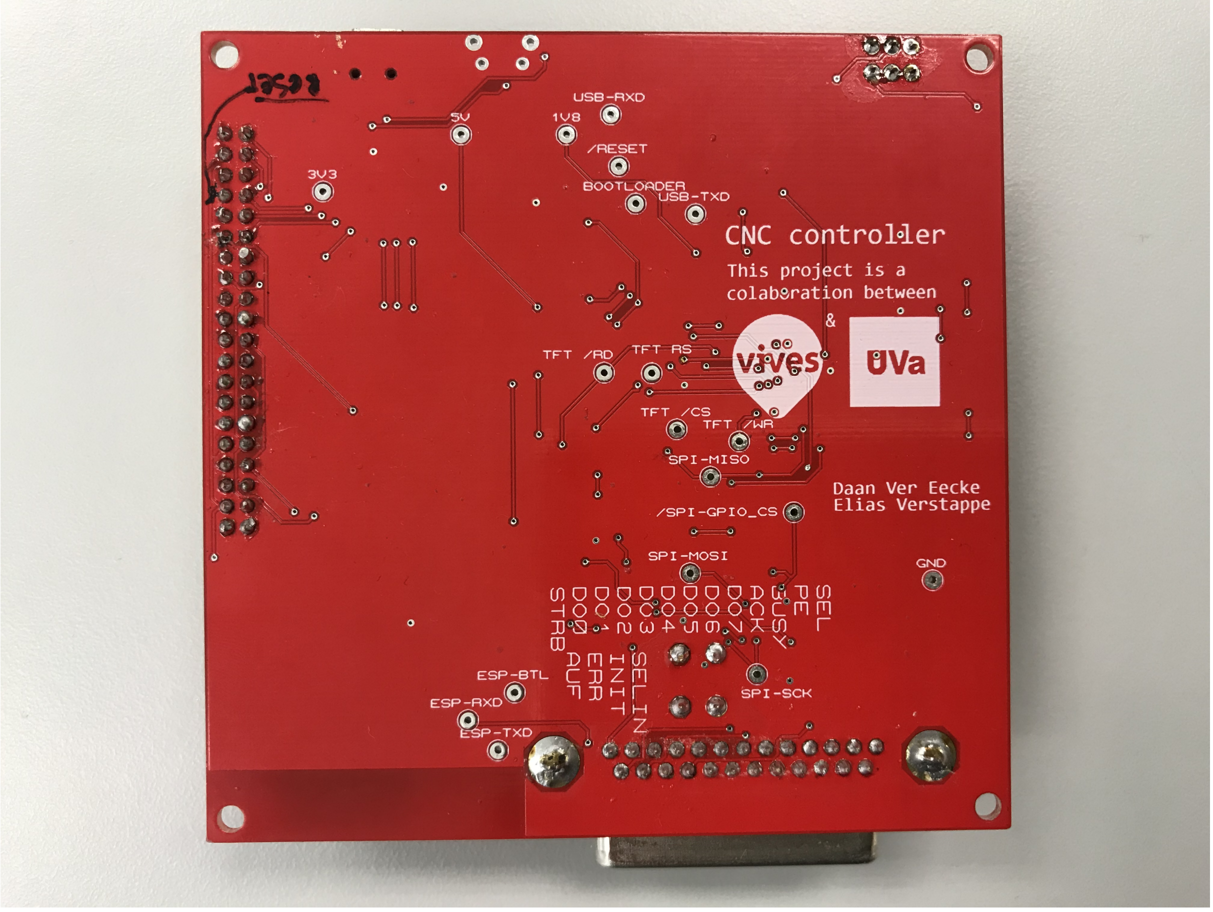


Figure B.2: Real board bottom view

## Appendix C

# Bill of materials

# Bill Of Materials for education_board_remake_v0.1

| | |
|---|---|
| **Design Title** | education_board_remake_v0.1 |
| **Author** | Elias Verstappe |
| **Document Number** | |
| **Revision** | |
| **Design Created** | maandag 11 februari 2019 |
| **Design Last Modified** | zondag 5 mei 2019 |
| **Total Parts In Design** | 60 |

## 2 Modules

| Quantity | References | Value | Stock Code | Unit Cost |
|---|---|---|---|---|
| 1 | M1 | TFT_43 | 2308433 | €19,74 |
| 1 | M2 | WIFI_MODULE_ESP12F | | €2,23 |
| Sub-totals: | | | | €21,97 |

## 14 Capacitors

| Quantity | References | Value | Stock Code | Unit Cost |
|---|---|---|---|---|
| 8 | C1-C2,C4,C9-C11,C14-C15 | 100nF | 2773208RL | €0,04 |
| 3 | C3,C5-C6 | 22uF/6.3V | 2988879 | €0,03 |
| 2 | C7-C8 | 33pF | 2310647 | €0,02 |
| 1 | C12 | 10uF | 2776900RL | €0,42 |
| Sub-totals: | | | | €0,87 |

## 10 Resistors

| Quantity | References | Value | Stock Code | Unit Cost |
|---|---|---|---|---|
| 3 | R2,R7,R12 | 1k | 2688819 | €0,01 |
| 4 | R3-R4,R9,R11 | 10k | 2692284 | €0,01 |
| 2 | R5,R8 | 3k3 | 2123191 | €0,01 |
| 1 | R6 | 560 | 2484004 | €0,14 |
| Sub-totals: | | | | €0,22 |

## 6 Integrated Circuits

| Quantity | References | Value | Stock Code | Unit Cost |
|---|---|---|---|---|
| 1 | U1 | LPC2106FBD48/01,15 | 1832283 | €10,33 |
| 1 | U2 | TC2117-3.3VDBTR | 1605581 | €0,48 |
| 1 | U3 | FT232RL-REEL | 1146032RL | €2,31 |
| 1 | U4 | MC33375ST-1.8T3G | 2724158 | €0,25 |
| 1 | U5 | MCP23S17-E/SO | 1332093 | €0,83 |
| 1 | U8 | 74HCT04D | 1085299 | €0,09 |
| Sub-totals: | | | | €14,30 |

## 0 Transistors

| Quantity | References | Value | Stock Code | Unit Cost |
|---|---|---|---|---|
| Sub-totals: | | | | €0,00 |

## 0 Diodes

| Quantity | References | Value | Stock Code | Unit Cost |
|---|---|---|---|---|
| Sub-totals: | | | | €0,00 |

## 28 Miscellaneous

| Quantity | References | Value | Stock Code | Unit Cost |
|---|---|---|---|---|
| 19 | /RESET,/SPI-GPIO_CS,1V8,3V3,5V,BOOTLOADER,ESP-BTL,ESP-RXD,ESP-TXD,GND,SPI-MISO,SPI-MOSI,SPI-SCK,TFT /CS,TFT /RD,TFT /WR,TFT RS,USB-RXD,USB-TXD | (not a component) | | |
| 1 | HEADER1 | BCS-120-L-D-TE | 2308433 | €3,25 |
| 1 | J1 | 67503-1020 | 2426381 | €0,44 |
| 1 | J3 | 5504F1-25S-02-03-F1 | 1084703 | €0,37 |
| 1 | J5 | 10118194-0001LF | 2668482 | €0,16 |
| 1 | L1 | 2508051217Y0 | 2413130 | €0,02 |
| 1 | PCB1 | ... | ... | €1,76 |
| 1 | S1 | JS202011AQN | 2435145 | €0,24 |
| 1 | S2 | BUTTON | 176433 | €0,16 |
| 1 | X1 | LFXTAL030358 | 2832032RL | €0,17 |
| Sub-totals: | | | | €6,57 |

| | |
|---|---|
| Totals: | €43,93 |

# Appendix D

# Design errors

After separately testing each component of the PCB, We have found some errors in our design. These errors are the following:

1. The reset pin of the WiFi module was not connected correctly. It should have been connected to the general reset signal coming from the USB to UART converter but was just pulled up to 3,3V via a resistor. This resulted in being unable to program the WiFi module.

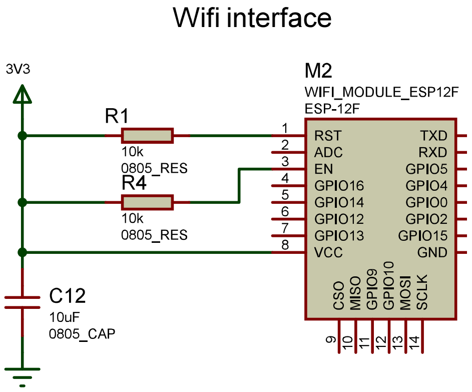


Figure D.1: Faulty connection

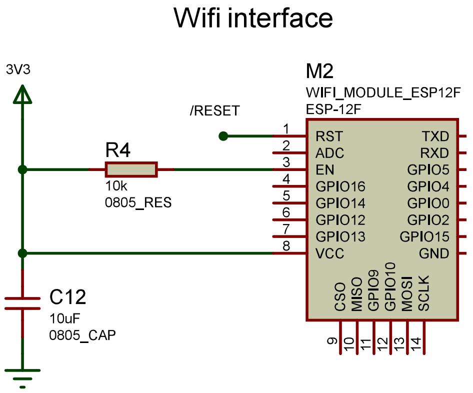


Figure D.2: Error has been fixed

2. The IO expander reset pin was connected to a reset signal, which was only 3,3V. This was too low because the minimum voltage (specified in the datasheet) that is recognized as a logical ‘1’ signal is 4V.

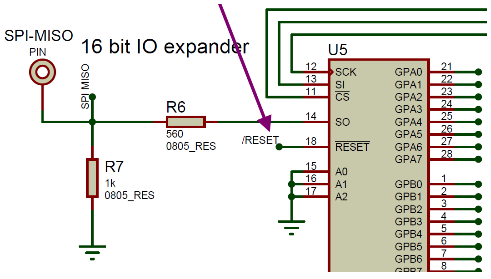


Figure D.3: Faulty reset connection

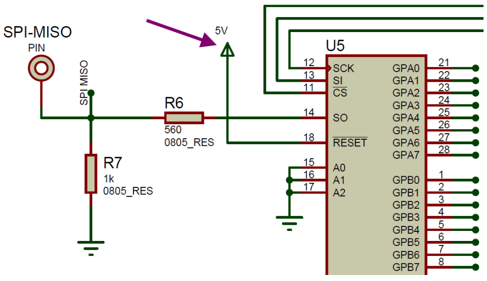


Figure D.4: Reset connected to 5V

3. One major issue is that the footprint of the 40-pin header was incorrect. This header is used to connect the TFT-screen to the main PCB. One side of pins needed to be mirrored. This issue was fixed by making an adaptor that physically connected the pins in a different order. In the picture below you can see how we addressed this issue.

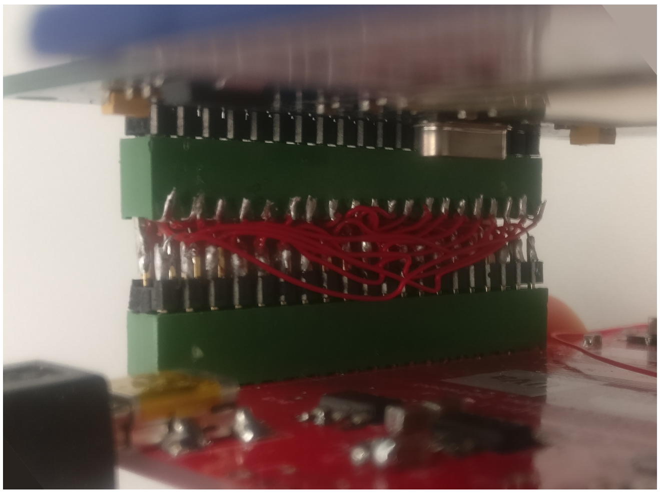


Figure D.5: Solution for the design error regarding the 40-pin header

4. A minor error we made is that two capacitors were placed too close together. As a result, it was hard to solder them.

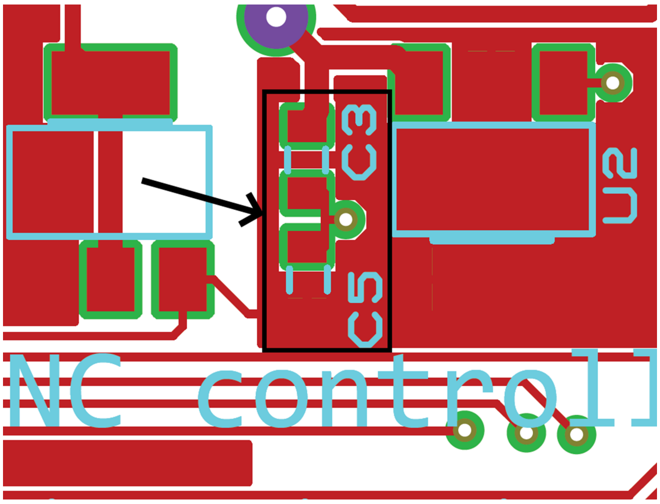


Figure D.6: C3 and C5 are too close together

These errors have since been corrected in the design.

# Appendix E

# Explanation of calibration in touchscreen systems

Maxim > Design Support > Technical Documents > Application Notes > A/D and D/A Conversion/Sampling Circuits > APP 5296
Maxim > Design Support > Technical Documents > Application Notes > Automotive > APP 5296
Maxim > Design Support > Technical Documents > Application Notes > Interface Circuits > APP 5296

Keywords: touch screen panels, touch screen controllers, touch screen, touch panels, N-point calibration, calibration, 3-point calibration, touch systems, resistive
Related Parts

APPLICATION NOTE 5296

# An Easy-to-Understand Explanation of Calibration in Touch-Screen Systems

By: Faisal Tariq

*Abstract: Mechanical misalignment and scaling factors lead to a mismatch between the values coming from a touch screen panel (as translated by a touch screen controller) and the display (typically an LCD) on which the touch screen panel is mounted. This tutorial discusses how to calibrate the touch screen panel to match the display.*

A similar version of this article appeared in the April 20, 2012 issue of *EE Times* magazine.

## Introduction

Mechanical misalignment and scaling factors can lead to a mismatch between the values from a touch-panel system (in this tutorial, the touch-panel system implies a setup comprised of a touch screen and a touch-screen controller) and the display (typically an LCD) on which the touch screen is mounted. This article discusses the mathematical techniques to calibrate the touch-panel system so you can match the graphics on the display to the output from the touch-screen controller.

## Understanding Image Transformation: Translation, Rotation, and Scaling

**Figure 1** is an exaggerated view of the distortion that might happen to a circle being displayed on an LCD touch-screen display. When a finger is traced around the circle (red line), the touch-panel system may give out the coordinates of an ellipse (blue line) instead of the circle, shown below in Figure 1. This change of the shape from a circle to an ellipse can be explained by the following graphic transformations: translation, rotation, and scaling.

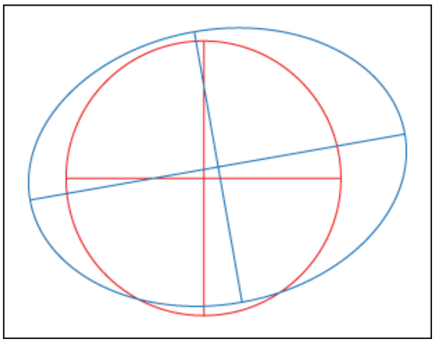


*Figure 1. A circle changes shape on a touch screen because of a mismatch between the display and the touch-panel system.*

Intuition suggests that any coordinate point x, y in an x-y plane that has undergone a transformation should look like:

$$x_{NEW} = f1(x_{OLD}, y_{OLD}) + \text{constant1} \quad \text{(Eq. 1a)}$$
$$y_{NEW} = f2(x_{OLD}, y_{OLD}) + \text{constant2} \quad \text{(Eq. 1b)}$$

Where $x_{NEW}$ and $y_{NEW}$ are the transformed coordinates; $x_{OLD}$ and $y_{OLD}$ are the old coordinates; f1() and f2() are functions that transform the old coordinates; constants1 and 2 are just that, constants.

If the transformation is linear, then functions f1() and f2() can be replaced by the following equations:

$$x_{NEW} = A\ x_{OLD} + B\ y_{OLD} + C \quad \text{(Eq. 2a)}$$
$$y_{NEW} = D\ x_{OLD} + E\ y_{OLD} + F \quad \text{(Eq. 2b)}$$

Where A, B, C, D, E, and F are constant coefficients.

Note that f1() = A $x_{OLD}$ +B $y_{OLD}$ and f2() = D $x_{OLD}$ + E $y_{OLD}$, where constant1 and constant2 are C and F, respectively.

## The Mathematics Behind Translation, Rotation, and Scaling

An example, showing a graphic transformation, will demonstrate that the final transformation equations from this exercise will lead to the above equations 2a and 2b and, therefore, our intuition holds true. (The user can bypass this example and jump directly to the **Three-Point Calibration** and **N-Point Calibration** discussions below.)

The illustration below (**Figure 2**) shows a square (Shape A) that has undergone translation, rotation, and scaling (Shape B). In the process it is transformed into a rectangle.

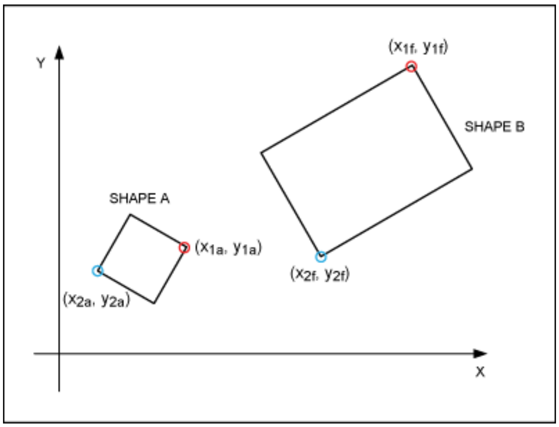


*Figure 2. Translation, rotation, and scaling transform a square into a rectangle.*

To rotate and scale the square (Shape A), it is important to move its center to the origin first (Figure 3) to ensure that the rotation and scaling happen uniformly.

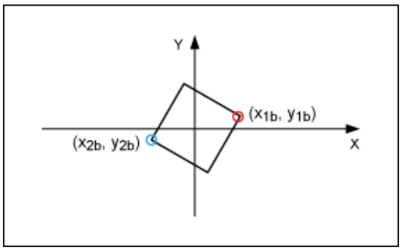


*Figure 3. The square of Figure 2 (Shape A) is moved so that its center lies on the origin of the x-y axis.*

This move to the center, or translation to the center, changes the value of all the coordinates of the square. For simplicity here, only the transformation of $x_1$, $y_1$ in Figure 2 will be explored. Consequently, $x_{1a}$, $y_{1a}$ translates to $x_{1b}$, $y_{1b}$ and the new equation representing this new figure is:

$$x_{1b} = x_{1a} + T_{x0} \quad \text{(Eq. 3a)}$$
$$y_{1b} = y_{1a} + T_{y0} \quad \text{(Eq. 3b)}$$

Rotate the above figure so that proper scaling can be done in the x and y directions. **Figure 4** shows that the figure has been rotated counterclockwise by angle θ.

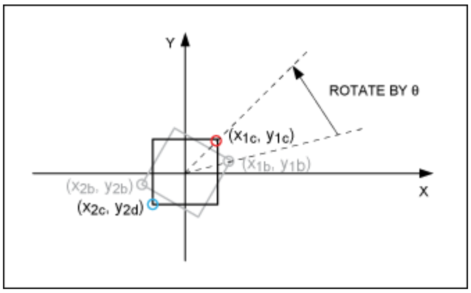


*Figure 4. Following Equation 3, the square is rotated counterclockwise.*

The new coordinates ($x_{1c}$, $y_{1c}$) after the rotation are shown below. (Please see the Appendix for the derivation of the equation for this rotation.)

$$x_{1c} = x_{1b}\,\mathrm{Cos}\theta - y_{1b}\,\mathrm{Sin}\theta \qquad \text{(Eq. 4a)}$$
$$y_{1c} = x_{1b}\,\mathrm{Sin}\theta + y_{1b}\,\mathrm{Cos}\theta \qquad \text{(Eq. 4b)}$$

Substitute Equations 3a and 3b into Equations 4a and 4b, and simplify:

$$x_{1c} = x_{1a}\,\mathrm{Cos}\theta - y_{1a}\,\mathrm{Sin}\theta + K_x \qquad \text{(Eq. 5a)}$$
$$y_{1c} = x_{1a}\,\mathrm{Sin}\theta + y_{1a}\,\mathrm{Cos}\theta + K_y \qquad \text{(Eq. 5b)}$$

Where $K_x = T_{x0}\,\mathrm{Cos}\theta - T_{y0}\,\mathrm{Sin}\theta$ and $K_y = T_{x0}\,\mathrm{Sin}\theta + T_{y0}\,\mathrm{Cos}\theta$.

Now scale the square in both the x and y directions to make it the same size as the rectangle in Figure 2 (Shape B). If we say that $G_x$ is the scaling factor in the x direction and $G_y$ is the scaling factor in the y direction, then the square will transform into a rectangle (**Figure 5**).

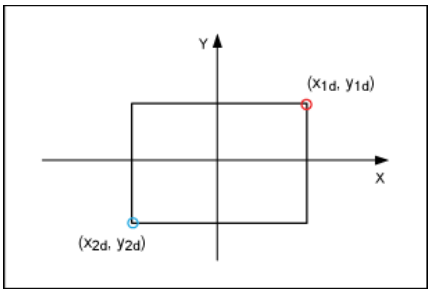


*Figure 5. Transform the square into a rectangle using scaling factors $G_x$ for the x direction, and $G_y$ for the y direction.*

Multiplying Equations 5a and 5b with the scaling factors $G_x$ and $G_y$ yields Equations 6a and 6b, which are the coordinates for the rectangle shown in Figure 5.

$$x_{1d} = x_{1a}\,G_x\,\mathrm{Cos}\theta - y_{1a}\,G_x\,\mathrm{Sin}\theta + K_x\,G_x \qquad \text{(Eq. 6a)}$$
$$y_{1d} = x_{1a}\,G_y\,\mathrm{Sin}\theta + y_{1a}\,G_y\,\mathrm{Cos}\theta + K_y\,G_y \qquad \text{(Eq. 6b)}$$

Rotate the rectangle by a so that it matches the rotational orientation of Shape B in Figure 2. This leads to the transformation of ($x_{1d}$, $y_{1d}$) into ($x_{1e}$, $y_{1e}$), as shown in the **Figure 6**.

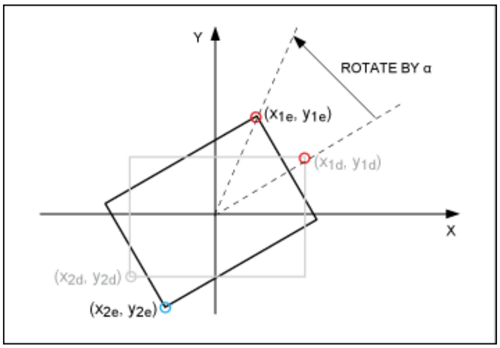


*Figure 6. The rectangle of Figure 5 is rotated so that it matches the orientation of Shape B in Figure 2.*

Once again the rotation is counterclockwise. The equations for ($x_{1e}$, $y_{1e}$) in terms of ($x_{1d}$, $y_{1d}$) are:

$x_{1e} = x_{1d}$ Cosα - $y_{1d}$ Sinα + $K_x$ (Eq. 7a)
$y_{1e} = x_{1d}$ Sinα + $y_{1d}$ Cosα + $K_y$ (Eq. 7b)
Substitute Equations 6a and 6b into Equations 7a and 7b, and simplify:

$x_{1e} = x_{1a}$ A + $y_{1a}$ B + P (Eq. 8a)
$y_{1e} = x_{1a}$ D + $y_{1a}$ E + Q (Eq. 8b)
Where:
A = $G_x$ Cosθ Cosα - $G_y$ Sinθ Cosα
B = - $G_x$ Sinθ Cosα - $G_y$ Cosθ Cosα
P = $K_x$ $G_x$ Cosα - $K_y$ $G_y$ Sinα
D = $G_x$ Cosθ Sinα + $G_y$ Sinθ Cosα
E = -$G_x$ Sinθ Sinα + $G_y$ Cosθ Cosα
Q = $K_x$ $G_x$ Sinα + $K_y$ $G_y$ Cosα

Now a translation (**Figure 7**) needs to be done to move the rotated rectangle to where Shape B appears in Figure 2.

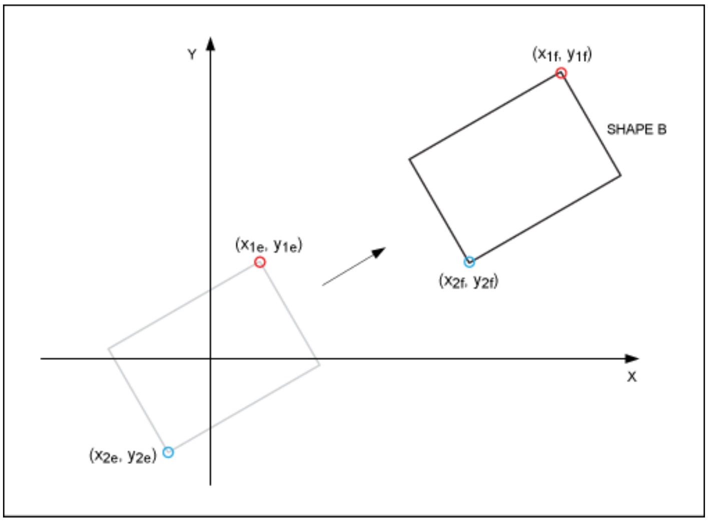


*Figure 7. Translation of the rectangle from the origin to the location where Shape B appears in Figure 2.*

Assume that the translation in the x and y direction is $T_x$ and $T_y$, respectively. Therefore, Equations 8a and 8b will change to:

$x_{1f} = x_{1a}$ A + $y_{1a}$ B + C (Eq. 9a)
$y_{1f} = x_{1a}$ D + $y_{1a}$ E + F (Eq. 9b)
Where C = P + $T_x$ and F = Q + $T_y$.

Equations 9a and 9b are what we want—they match Equations 1a and 1b. Note that the coordinate ($x_{1f}$, $y_{1f}$) is in terms of ($x_{1a}$, $y_{1a}$). In the touch panel mounted on a display the user will have to determine the A, B, C, D, E, and F during calibration.

*Please note that the purpose of this exercise is to give the user an appreciation of the general form of the equation for the transformation, shown by Equations 9a and 9b. This article shows that we will arrive at this general form irrespective of the direction and magnitude of rotation, translation, and scaling.*

## Three-Point Calibration

We start with a pair of equations, the final pair in our mathematical exercise above.

$x_{1f} = x_{1a}$ A + $y_{1a}$ B + C (Eq. 9a)
$y_{1f} = x_{1a}$ D + $y_{1a}$ E + F (Eq. 9b)
The goal of the touch-panel system calibration is to solve the Equations 2a and 2b (or similarly Equations 9a and 9b) to derive values for A, B, C, D, E, and F.

Looking at these equations we know that there are six unknowns. Therefore, we will need six equations to solve for these unknowns and this can be achieved by doing a three-point calibration for a touch-panel system. The user will generate three pairs of (x, y) coordinates by touching the panel at the three pairs of display coordinates: ($x_{1d}$, $y_{1d}$), ($x_{2d}$, $y_{2d}$) and ($x_{3d}$, $y_{3d}$). If their corresponding touch-screen values (as presented by the touch-screen controller) are ($x_1$, $y_1$), ($x_2$, $y_2$), and ($x_3$, $y_3$), then the six unknowns can be solved by the equations shown below. These points must be independent of each other, as shown in **Figure 8**.

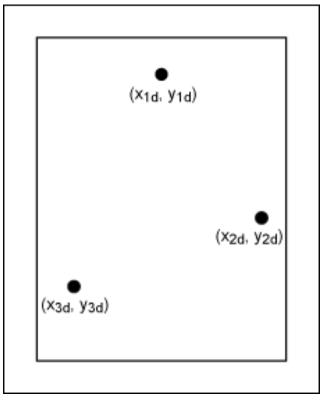


*Figure 8. LCD panel showing three display coordinates for the user to touch.*

$x_{1d} = x_1 A + y_1 B + C$
$x_{2d} = x_2 A + y_2 B + C$
$x_{3d} = x_3 A + y_3 B + C$

$y_{1d} = x_1 D + y_1 E + F$
$y_{2d} = x_2 D + y_2 E + F$
$y_{3d} = x_3 D + y_3 E + F$

Now with six equations for the six unknowns, the above can be written in matrix form as:

$$\begin{bmatrix} x_{1d} \\ x_{2d} \\ x_{3d} \end{bmatrix} = Z \times \begin{bmatrix} A \\ B \\ C \end{bmatrix} \qquad \begin{bmatrix} y_{1d} \\ y_{2d} \\ y_{3d} \end{bmatrix} = Z \times \begin{bmatrix} D \\ E \\ F \end{bmatrix} \qquad \text{Where, } Z = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$

Therefore, a little matrix manipulation will yield A, B, C, D, E, and F, as shown below:

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = Z^{-1} \times \begin{bmatrix} x_{1d} \\ x_{2d} \\ x_{3d} \end{bmatrix} \quad \text{Eq. 10a} \qquad \begin{bmatrix} D \\ E \\ F \end{bmatrix} = Z^{-1} \times \begin{bmatrix} y_{1d} \\ y_{2d} \\ y_{3d} \end{bmatrix} \quad \text{Eq. 10b}$$

Where, $\mathbf{Z^{-1}}$ is the inverse matrix of **Z**.

## Three-Point Calibration Example

We will use the MAX11800 touch-screen controller in this example. Assume that an LCD panel display has a resolution of 256 x 768 and that the three calibration points chosen are (65, 350), (200, 195), and (195, 550). These are the exact points that we want the touch panel to show when it is touched. However, since the resolution for the MAX11800 is 4096 x 4096 (12 bits) and because of mechanical misalignment, the coordinate will come out differently. For this exercise, we assume that these values are: (650, 2000), (2800, 1350), and (2640, 3500), respectively. *Please note: these values are made up and serve as an example only*.

Solving for A, B, C, D, E, and F using Equations 10a and 10b, we get the following:

A = 0.0635
B = 0.0024
C = 18.9116
D = -0.0227
E = 0.1634
F = 37.8887

Therefore, the equations that will generate the x and y coordinates for this particular example are:

$x_d$ = 0.0635 x + 0.0024 y + 18.9116
$y_d$ = -0.0227 x + 0.1634 y + 37.8887
Where (x, y) are the coordinates coming from the touch-panel system and ($x_d$, $y_d$) is the adjusted value.

## N-Point Calibration

The user may choose to calibrate using more points. A generalized form of the sets of equations will look like the following:

$$\begin{aligned} x_{1d} &= x_1 A + y_1 B + C \\ x_{2d} &= x_2 A + y_2 B + C \\ x_{3d} &= x_3 A + y_3 B + C \qquad \text{(Eq. set 11a)} \\ &\ldots \\ x_{nd} &= x_n A + y_n B + C \end{aligned}$$

$$\begin{aligned} y_{1d} &= x_1 D + y_1 E + F \\ y_{2d} &= x_2 D + y_2 E + F \\ y_{3d} &= x_3 D + y_3 E + F \qquad \text{(Eq. set 11b)} \\ &\ldots \\ y_{nd} &= y_n D + y_3 E + F \end{aligned}$$

Where, ($x_{1d}$, $y_{1d}$)...($x_{nd}$, $y_{nd}$) are coordinates generated by the display; ($x_1$, $y_1$)...($x_n$, $y_n$) are the corresponding values (for n points) determined by the MAX11800 from the touch-screen panel. The goal is to determine the coefficients using these values.

In the Equation set 11a there are three unknowns, A, B, and C, but the set of coordinates is more than three. This implies that there are more equations than unknowns. Therefore, in this instance it makes sense to use the least square fit to utilize all the points and derive an average value of the coefficients. This also means that more calibration points would help lower errors. The least square fit is represented in **Figure 9**. The same technique will be applied for determining the unknowns D, E, and F for the y values.

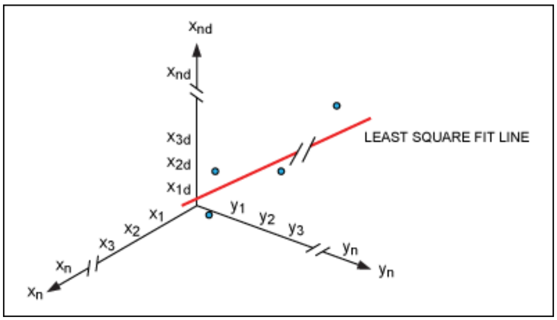


*Figure 9. An average value for the coefficients A, B, and C is found by applying the least square fit to the points.*

Equation sets 11a and 11b can be written in matrix form, as shown:

$$\begin{bmatrix} x_{1d} \\ x_{2d} \\ x_{3d} \\ \ldots \\ x_{nd} \end{bmatrix} = Z \times \begin{bmatrix} A \\ B \\ C \end{bmatrix} \qquad \begin{bmatrix} y_{1d} \\ y_{2d} \\ y_{3d} \\ \ldots \\ y_{nd} \end{bmatrix} = Z \times \begin{bmatrix} D \\ E \\ F \end{bmatrix} \qquad \text{Where, } Z = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ \ldots & & \\ x_{nd} & y_{nd} & 1 \end{bmatrix}$$

By using the least square fit in this matrix form, the coefficients are given by the equations below. (Please refer to a book on regression analysis for the derivation of the matrix form of the least square fit form.)

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = (Z^T \times Z)^{-1} \times Z^T \begin{bmatrix} x_{1d} \\ x_{2d} \\ x_{3d} \\ \ldots \\ x_{nd} \end{bmatrix} \quad \text{Eq. 12a} \qquad \begin{bmatrix} D \\ E \\ F \end{bmatrix} = (Z^T \times Z)^{-1} \times Z^T \begin{bmatrix} y_{1d} \\ y_{2d} \\ y_{3d} \\ \ldots \\ y_{nd} \end{bmatrix} \quad \text{Eq. 12b}$$

## N-Point Calibration Example

Once again, assume that an LCD panel display has a resolution of 256 x 768 and the five calibration points are (100, 350), (50, 200), (200, 200), (210, 600), and (65, 600). These are the exact points that we want the touch panel to show when it is touched. However, since the resolution for the MAX11800 is 4096 x 4096 (12 bits) and because of mechanical misalignment, the points may come out differently. Again assume that they are (1700, 2250), (750, 1200), (3000, 1500), (2500, 3400), and (600, 3000), respectively. *Please note: these values are made up and meant to serve as an example only*.

Solving for A, B, C, D, E, and F using Equations 12a and 12b, we get:

A = 0.0677
B = 0.0190
C = -33.7973
D = -0.0347
E = 0.2100
F = -27.4030

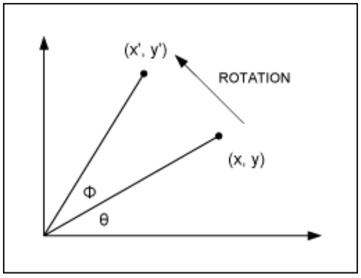
Therefore, the equations to generate the x and y coordinates for this particular example are:

$x_d$ = 0.0677 x + 0.0190 y - 33.7973
$y_d$ = -0.0347 x + 0.2100 y - 27.4030
Where (x, y) are the coordinates coming from the touch-panel controller, and ($x_d$, $y_d$) is the adjusted coordinate value to match what is shown on the display.

## Example Summary

If the least squares technique is used on three points, it will yield the same coefficient as the three-point calibration. It is mathematically easier, therefore, to handle the three-point calibration as three simultaneous linear equations with three unknowns instead of the lengthier calculation with the least squares technique.

## Appendix: Derivation of the Equation for a Counterclockwise Rotation



The original point (x, y) can be written in polar form as:

$$x = r\,\mathrm{Cos}\theta \text{ and } y = r\,\mathrm{Sin}\theta, \qquad \text{(Eq. A)}$$

Where r is the radial distance from the origin to the (x, y) coordinate.

After the rotation the (x, y) coordinates become (x', y'):

$$x' = r\,\mathrm{Cos}(\theta + \Phi) \text{ and } y' = r\,\mathrm{Sin}(\theta + \Phi) \qquad \text{(Eq. B)}$$

Using the trig identities below:

$$\begin{aligned} \mathrm{Cos}(\theta + \Phi) &= \mathrm{Cos}\Phi\,\mathrm{Cos}\theta - \mathrm{Sin}\Phi\,\mathrm{Sin}\theta \\ \mathrm{Sin}(\theta + \Phi) &= \mathrm{Sin}\Phi\,\mathrm{Cos}\theta - \mathrm{Cos}\Phi\,\mathrm{Sin}\theta \end{aligned} \qquad \text{(Eq. C)}$$

And substituting in Equation B, we get:

$$x' = r \cos\Phi \cos\theta - r \sin\Phi \sin\theta$$
$$y' = r \sin\Phi \cos\theta - r \cos\Phi \sin\theta \qquad \text{(Eq. D)}$$

Substituting Equation A into Equation D:

$$x' = x \cos\Phi - y \sin\Phi$$
$$y' = x \sin\Phi - y \cos\Phi \qquad \text{(Eq. E)}$$

The derivation for the clockwise rotation is left as an exercise for the user.

## Related Parts

| | |
|---|---|
| MAX11800 | Low-Power, Ultra-Small Resistive Touch-Screen Controllers with I²C/SPI Interface |
| MAX11801 | Low-Power, Ultra-Small Resistive Touch-Screen Controllers with I²C/SPI Interface |
| MAX11802 | Low-Power, Ultra-Small Resistive Touch-Screen Controllers with I²C/SPI Interface |
| MAX11803 | Low-Power, Ultra-Small Resistive Touch-Screen Controllers with I²C/SPI Interface |
| MAX11811 | TacTouch™, Low-Power, Ultra-Small, Resistive Touch-Screen Controller with Haptic Driver |

## Next Steps

| | |
|---|---|
| EE-Mail | Subscribe to EE-Mail and receive automatic notice of new documents in your areas of interest. |
| Download | Download, PDF Format (247.4kB) |
| Share | G+1 Tweet Me gusta 0 Other Channels Email this page to an associate or friend. |

APP 5296: May 07, 2012
APPLICATION NOTE 5296, AN5296, AN 5296, APP5296, Appnote5296, Appnote 5296

# Appendix F

# Source code

main.c

```c
#include "lib/system.h"

extern const unsigned char * CurrentFont;
extern const unsigned char SmallFont[];
extern const unsigned char BigFont[];
extern const unsigned char SevenSegNumFont[];

BOOLEAN motor_state = FALSE;


void IRQ_Button_Routine (void) {
    if (appRunning)
    {
        set_button_interrupt_flag();
    }
        EXTINT = (1<<1);
    VICVectAddr=-1;
}


void IRQ_Spurious_Routine (void) {
    VICVectAddr=-1;
}

char* itoa(int num, char* str, int base);
void reverse(char *s, int lengthOfString);
unsigned int myPow(int base, int exponent);



int main (void)
{
    _printf("Starting_application\n");

    init_io_config();
    UART1_Init();
    stateMachine_Init();
    TFTInit();
    SPI_init();
    SPI_CLK_normal();
    init_external_interrupt();
    init_spurious_interrupt();

        drawStartupScreen();
    _delay_ms(1000);

    InitiateSpeed(5000);    // set maximum speed to 5000 mm/min

    setToolData(1, 20, 1, 0, 0, 30000);
    setToolData(2, 569, 1, 0, 0, 30000);
    setToolData(3, 569, 1, 0, 0, 60000);
    setToolData(4, 1170, 1, 0, 0, 60000);
    setToolData(5, 800, 1, 0, 0, 45000);
    setRouteDrill(0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5);
    runApplication();
}


char* itoa(int num, char* str, int base)
{
    int i = 0;
    BOOLEAN isNegative = FALSE;
    /* Handle 0 explicitely, otherwise empty string is printed for 0 */
    if (num == 0)
    {
        str[i++] = '0';
        str[i] = '\0';
        return str;
    }
    // In standard itoa(), negative numbers are handled only with
    // base 10. Otherwise numbers are considered unsigned.
    if (num < 0 && base == 10)
    {
        isNegative = TRUE;
        num = -num;
    }
    // Process individual digits
    while (num != 0)
    {
        int rem = num % base;
        str[i++] = (rem > 9)? (rem-10) + 'a' : rem + '0';
        num = num/base;
    }
    // If number is negative, append '-'
    if (isNegative)
        str[i++] = '-';

    str[i] = '\0'; // Append string terminator
    // Reverse the string
    reverse(str, i);

    return str;
}


void reverse(char *s, int lengthOfString)
{
   int length, c;
   char *begin, *end, temp;

   length = lengthOfString;
   begin  = s;
   end    = s;

   for (c = 0; c < length - 1; c++)
      end++;

   for (c = 0; c < length/2; c++)
   {
      temp   = *end;
      *end   = *begin;
      *begin = temp;

      begin++;
      end——;
   }
}

unsigned int myPow(int base, int exponent)
{
    int i = 0;
    unsigned int number = 1;

    for (i = 0; i < exponent; ++i)
        number *= base;

    return(number);
}
```

lib/calibrate.h

```
/*
 *   Copyright (c) 2001, Carlos E. Vidales. All rights reserved.
 *
 *   This sample program was written and put in the public domain
 *    by Carlos E. Vidales.  The program is provided "as is"
 *    without warranty of any kind, either expressed or implied.
 *   If you choose to use the program within your own products
 *    you do so at your own risk, and assume the responsibility
 *    for servicing, repairing or correcting the program should
 *    it prove defective in any manner.
 *   You may copy and distribute the program's source code in any
 *    medium, provided that you also include in each copy an
 *    appropriate copyright notice and disclaimer of warranty.
 *   You may also modify this program and distribute copies of
 *    it provided that you include prominent notices stating
 *    that you changed the file(s) and the date of any change,
 *    and that you do not charge any royalties or licenses for
 *    its use.
 *
 *   File Name:  calibrate.h
 *
 *   Definition of constants and structures, and declaration of functions
 *    in Calibrate.c
 */
#ifndef _CALIBRATE_H_
#define _CALIBRATE_H_
#include <math.h>
#ifndef _CALIBRATE_C_
 #define EXTERN extern
#else
 #define EXTERN
#endif

#ifndef OK
 #define OK                     0
 #define NOT_OK            -1
#endif

#define INT32 long


#define MAX_SAMPLES     5


typedef struct Point { INT32 x,y ;} POINT ;
typedef struct Matrix {
                        INT32    An,     /* A = An/Divider */
                                 Bn,     /* B = Bn/Divider */
                                 Cn,     /* C = Cn/Divider */
                                 Dn,     /* D = Dn/Divider */
                                 En,     /* E = En/Divider */
                                 Fn,     /* F = Fn/Divider */
                                 Divider ;
                     } MATRIX ;
```

```
EXTERN int setCalibrationMatrix( POINT * display,
                                 POINT * screen,
                                 MATRIX * matrix);

EXTERN int getDisplayPoint( POINT * display,
                            POINT * screen,
                            MATRIX * matrix );


EXTERN MATRIX  matrix;


#endif  /* _CALIBRATE_H_ */
```

lib/calibrate.c

```
/*
 *
 *   Copyright (c) 2001, Carlos E. Vidales. All rights reserved.
 *
 *   This sample program was written and put in the public domain
 *    by Carlos E. Vidales.  The program is provided "as is"
 *    without warranty of any kind, either expressed or implied.
 *   If you choose to use the program within your own products
 *    you do so at your own risk, and assume the responsibility
 *    for servicing, repairing or correcting the program should
 *    it prove defective in any manner.
 *   You may copy and distribute the program's source code in any
 *    medium, provided that you also include in each copy an
 *    appropriate copyright notice and disclaimer of warranty.
 *   You may also modify this program and distribute copies of
 *    it provided that you include prominent notices stating
 *    that you changed the file(s) and the date of any change,
 *    and that you do not charge any royalties or licenses for
 *    its use.
 *
 *
 *
 *   File Name:  calibrate.c
 *
 *
 *   This file contains functions that implement calculations
 *    necessary to obtain calibration factors for a touch screen
 *    that suffers from multiple distortion effects: namely,
 *    translation, scaling and rotation.
 *
 *   The following set of equations represent a valid display
 *    point given a corresponding set of touch screen points:
 *
 *
 *                                               /-     -\
 *              /-    -\     /-            -\   |       |
 *              |      |     |              |   |   Xs  |
 *              |  Xd  |     | A    B    C  |   |       |
 *              |      |  =  |              | * |   Ys  |
 *              |  Yd  |     | D    E    F  |   |       |
 *              |      |     |              |   |   1   |
 *              \-    -/     \-            -/   |       |
 *                                               \-     -/
 *
 *
 *    where:
 *
 *           (Xd,Yd) represents the desired display point
 *                    coordinates,
 *
 *           (Xs,Ys) represents the available touch screen
 *                    coordinates, and the matrix
 *
 *           /-   -\
 *           |A,B,C|
 *           |D,E,F| represents the factors used to translate
 *           \—   —/  the available touch screen point values
 *                    into the corresponding display
 *                    coordinates.
 *
 *
 *    Note that for practical considerations, the utilitities
 *     within this file do not use the matrix coefficients as
 *     defined above, but instead use the following
 *     equivalents, since floating point math is not used:
 *
 *            A = An/Divider
 *            B = Bn/Divider
 *            C = Cn/Divider
 *            D = Dn/Divider
 *            E = En/Divider
 *            F = Fn/Divider
 *
 *
 *
 *    The functions provided within this file are:
 *
 *          setCalibrationMatrix() — calculates the set of factors
 *                                    in the above equation, given
 *                                    three sets of test points.
 *               getDisplayPoint() — returns the actual display
 *                                    coordinates, given a set of
 *                                    touch screen coordinates.
 * translateRawScreenCoordinates() — helper function to transform
 *                                    raw screen points into values
 *                                    scaled to the desired display
 *                                    resolution.
 *
 *
 */


#define _CALIBRATE_C_




/****************************************************/
/*                                                  */
/* Included files                                   */
/*                                                  */
/****************************************************/

#include "calibrate.h"




/****************************************************/
/*                                                  */
/* Local Definitions and macros                     */
/*                                                  */
/****************************************************/
```

```
/****************************************************/
/*                                                  */
/* Global variables                                 */
/*                                                  */
/****************************************************/



/****************************************************/
/*                                                  */
/* Forward Declaration of local functions           */
/*                                                  */
/****************************************************/






/**********************************************************************
 *
 *     Function: setCalibrationMatrix()
 *
 *  Description: Calling this function with valid input data
 *                in the display and screen input arguments
 *                causes the calibration factors between the
 *                screen and display points to be calculated,
 *                and the output argument - matrixPtr - to be
 *                populated.
 *
 *               This function needs to be called only when new
 *                calibration factors are desired.
 *
 *
 *  Argument(s): displayPtr (input) - Pointer to an array of three
 *                                     sample, reference points.
 *               screenPtr (input) - Pointer to the array of touch
 *                                    screen points corresponding
 *                                    to the reference display points.
 *               matrixPtr (output) - Pointer to the calibration
 *                                     matrix computed for the set
 *                                     of points being provided.
 *
 *
 *  From the article text, recall that the matrix coefficients are
 *   resolved to be the following:
 *
 *
 *      Divider =  (Xs0 - Xs2)*(Ys1 - Ys2) - (Xs1 - Xs2)*(Ys0 - Ys2)
 *
 *
 *
 *                 (Xd0 - Xd2)*(Ys1 - Ys2) - (Xd1 - Xd2)*(Ys0 - Ys2)
 *            A = ---------------------------------------------------
 *                                  Divider
 *
 *
 *                  (Xs0 - Xs2)*(Xd1 - Xd2) - (Xd0 - Xd2)*(Xs1 - Xs2)
 *             B = ----------------------------------------------------
 *                                  Divider
 *
 *
 *                  Ys0*(Xs2*Xd1 - Xs1*Xd2) +
 *                              Ys1*(Xs0*Xd2 - Xs2*Xd0) +
 *                                            Ys2*(Xs1*Xd0 - Xs0*Xd1)
 *             C = ----------------------------------------------------
 *                                  Divider
 *
 *
 *                  (Yd0 - Yd2)*(Ys1 - Ys2) - (Yd1 - Yd2)*(Ys0 - Ys2)
 *             D = ----------------------------------------------------
 *                                  Divider
 *
 *
 *                  (Xs0 - Xs2)*(Yd1 - Yd2) - (Yd0 - Yd2)*(Xs1 - Xs2)
 *             E = ----------------------------------------------------
 *                                  Divider
 *
 *
 *                  Ys0*(Xs2*Yd1 - Xs1*Yd2) +
 *                              Ys1*(Xs0*Yd2 - Xs2*Yd0) +
 *                                            Ys2*(Xs1*Yd0 - Xs0*Yd1)
 *             F = ----------------------------------------------------
 *                                  Divider
 *
 *
 *       Return: OK - the calibration matrix was correctly
 *                     calculated and its value is in the
 *                     output argument.
 *               NOT_OK - an error was detected and the
 *                         function failed to return a valid
 *                         set of matrix values.
 *                        The only time this sample code returns
 *                        NOT_OK is when Divider == 0
 *
 *
 *
 *                 NOTE!    NOTE!    NOTE!
 *
 *  setCalibrationMatrix() and getDisplayPoint() will do fine
 *  for you as they are, provided that your digitizer
 *  resolution does not exceed 10 bits (1024 values).  Higher
 *  resolutions may cause the integer operations to overflow
 *  and return incorrect values.  If you wish to use these
 *  functions with digitizer resolutions of 12 bits (4096
 *  values) you will either have to a) use 64-bit signed
 *  integer variables and math, or b) judiciously modify the
 *  operations to scale results by a factor of 2 or even 4.
 *
 *
 */
int setCalibrationMatrix( POINT * displayPtr,
                          POINT * screenPtr,
                          MATRIX * matrixPtr)
{

    int  retValue = OK ;



    matrixPtr->Divider = ((screenPtr[0].x - screenPtr[2].x) * (screenPtr[1].y -
    screenPtr[2].y)) -
                         ((screenPtr[1].x - screenPtr[2].x) * (screenPtr[0].y -
    screenPtr[2].y)) ;

    if( matrixPtr->Divider == 0 )
    {
        retValue = NOT_OK ;
    }
    else
    {
        matrixPtr->An = ((displayPtr[0].x - displayPtr[2].x) * (screenPtr[1].y -
    screenPtr[2].y)) -
                        ((displayPtr[1].x - displayPtr[2].x) * (screenPtr[0].y -
    screenPtr[2].y)) ;

        matrixPtr->Bn = ((screenPtr[0].x - screenPtr[2].x) * (displayPtr[1].x -
    displayPtr[2].x)) -
                        ((displayPtr[0].x - displayPtr[2].x) * (screenPtr[1].x -
    screenPtr[2].x)) ;

        matrixPtr->Cn = (screenPtr[2].x * displayPtr[1].x - screenPtr[1].x *
    displayPtr[2].x) * screenPtr[0].y +
                        (screenPtr[0].x * displayPtr[2].x - screenPtr[2].x *
    displayPtr[0].x) * screenPtr[1].y +
                        (screenPtr[1].x * displayPtr[0].x - screenPtr[0].x *
    displayPtr[1].x) * screenPtr[2].y ;

        matrixPtr->Dn = ((displayPtr[0].y - displayPtr[2].y) * (screenPtr[1].y -
    screenPtr[2].y)) -
                        ((displayPtr[1].y - displayPtr[2].y) * (screenPtr[0].y -
    screenPtr[2].y)) ;

        matrixPtr->En = ((screenPtr[0].x - screenPtr[2].x) * (displayPtr[1].y -
    displayPtr[2].y)) -
                        ((displayPtr[0].y - displayPtr[2].y) * (screenPtr[1].x -
    screenPtr[2].x)) ;

        matrixPtr->Fn = (screenPtr[2].x * displayPtr[1].y - screenPtr[1].x *
    displayPtr[2].y) * screenPtr[0].y +
                        (screenPtr[0].x * displayPtr[2].y - screenPtr[2].x *
    displayPtr[0].y) * screenPtr[1].y +
                        (screenPtr[1].x * displayPtr[0].y - screenPtr[0].x *
    displayPtr[1].y) * screenPtr[2].y ;
    }

    return( retValue ) ;

} /* end of setCalibrationMatrix() */
```

```
/**********************************************************************
 *
 *     Function: getDisplayPoint()
 *
 *  Description: Given a valid set of calibration factors and a point
 *                value reported by the touch screen, this function
 *                calculates and returns the true (or closest to true)
 *                display point below the spot where the touch screen
 *                was touched.
 *
 *
 *
 *  Argument(s): displayPtr (output) – Pointer to the calculated
 *                                      (true) display point.
 *               screenPtr (input) – Pointer to the reported touch
 *                                    screen point.
 *               matrixPtr (input) – Pointer to calibration factors
 *                                    matrix previously calculated
 *                                    from a call to
 *                                    setCalibrationMatrix()
 *
 *
 *  The function simply solves for Xd and Yd by implementing the
 *   computations required by the translation matrix.
 *
 *                                              /–     –\
 *              /–    –\     /–            –\   |       |
 *              |      |     |              |   |   Xs  |
 *              |  Xd  |     | A    B    C  |   |       |
 *              |      |  =  |              | * |   Ys  |
 *              |  Yd  |     | D    E    F  |   |       |
 *              |      |     |              |   |   1   |
 *              \–    –/     \–            –/   |       |
 *                                              \–     –/
 *
 *  It must be kept brief to avoid consuming CPU cycles.
 *
 *
 *       Return: OK – the display point was correctly calculated
 *                     and its value is in the output argument.
 *               NOT_OK – an error was detected and the function
 *                         failed to return a valid point.
 *
 *
 *
 *                 NOTE!    NOTE!    NOTE!
 *
 *  setCalibrationMatrix() and getDisplayPoint() will do fine
 *  for you as they are, provided that your digitizer
 *  resolution does not exceed 10 bits (1024 values).  Higher
 *  resolutions may cause the integer operations to overflow
 *  and return incorrect values.  If you wish to use these
 *  functions with digitizer resolutions of 12 bits (4096
 *  values) you will either have to a) use 64–bit signed
 *  integer variables and math, or b) judiciously modify the
 *  operations to scale results by a factor of 2 or even 4.
 *
 *
 */
int getDisplayPoint( POINT * displayPtr,
                     POINT * screenPtr,
                     MATRIX * matrixPtr )
{
    int  retValue = OK ;


    if( matrixPtr->Divider != 0 )
    {
            /* Operation order is important since we are doing integer */
            /*  math. Make sure you add all terms together before      */
            /*  dividing, so that the remainder is not rounded off     */
            /*  prematurely.                                           */

        displayPtr->x = ( (matrixPtr->An * screenPtr->x) +
                          (matrixPtr->Bn * screenPtr->y) +
                           matrixPtr->Cn
                        ) / matrixPtr->Divider ;

        displayPtr->y = ( (matrixPtr->Dn * screenPtr->x) +
                          (matrixPtr->En * screenPtr->y) +
                           matrixPtr->Fn
                        ) / matrixPtr->Divider ;
    }
    else
    {
        retValue = NOT_OK ;
    }

    return( retValue ) ;

} /* end of getDisplayPoint() */



void calibrate()
{

POINT displaySample[MAX_SAMPLES] =      {
                                        {  72,  41 },
                                        { 240, 231 },
                                        { 408, 136 },
                                                                        } ;



POINT perfectScreenSample[3] =  {
                                        {  72,  41 },
                                        { 240, 231 },
                                        { 408, 136 },
                                } ;


```

```
POINT perfectDisplaySample[3] = {
                                        {  72,  41 },
                                        { 240, 231 },
                                        { 408, 136 },
                                    } ;


    int retValue = OK ;

    // MATRIX  matrix ;
    POINT   display ;


    int     n ;



    setCalibrationMatrix( &perfectDisplaySample[0],
                                                  &perfectScreenSample[0],
                                                  &matrix ) ;



    _printf("\n\nLook␣at␣the␣unity␣matrix:\n\n"
           "matrix.An␣=␣%d␣␣matrix.Bn␣=␣%d␣␣matrix.Cn␣=␣%d\n"
           "matrix.Dn␣=␣%d␣␣matrix.En␣=␣%d␣␣matrix.Fn␣=␣%d\n"
           "matrix.Divider␣=␣%d\n",
           matrix.An,matrix.Bn,matrix.Cn,
           matrix.Dn,matrix.En,matrix.Fn,
           matrix.Divider ) ;


        int clbPointArray[10] = {0};

        readPoints(&clbPointArray);


    POINT screenSample[MAX_SAMPLES] =   {
                                            { clbPointArray[0], clbPointArray[1] },
                                            { clbPointArray[2], clbPointArray[3] },
                                            { clbPointArray[4], clbPointArray[5] },
                                            { clbPointArray[6], clbPointArray[7] },
                                            { clbPointArray[8], clbPointArray[9] },
                                        } ;


    setCalibrationMatrix( &displaySample[0], &screenSample[0], &matrix ) ;



    _printf("\n\nThis␣is␣the␣actual␣calibration␣matrix␣that␣we␣will␣use\n"
           "for␣all␣points␣(until␣we␣calibrate␣again):\n\n"
           "matrix.An␣=␣%d␣␣matrix.Bn␣=␣%d␣␣matrix.Cn␣=␣%d\n"
           "matrix.Dn␣=␣%d␣␣matrix.En␣=␣%d␣␣matrix.Fn␣=␣%d\n"
           "matrix.Divider␣=␣%d\n",
           matrix.An,matrix.Bn,matrix.Cn,
           matrix.Dn,matrix.En,matrix.Fn,
           matrix.Divider ) ;


    _printf("\n\nShow␣the␣results␣of␣our␣work:\n\n"
           "␣␣Screen␣Sample␣␣␣␣Translated␣Sample␣␣␣␣Display␣Sample\n\n" ) ;


    for( n = 0 ; n < MAX_SAMPLES ; ++n )
    {
        getDisplayPoint( &display, &screenSample[n], &matrix ) ;
        _printf("␣␣%␣6d,%-6d␣␣␣␣␣␣%␣6d,%-6d␣␣␣␣␣␣␣%␣6d,%-6d\n",
                screenSample[n].x,  screenSample[n].y,
                display.x,          display.y,
                displaySample[n].x, displaySample[n].y ) ;
    }

}
```

lib/cncroutines.h

```c
#pragma once
#include "system.h"


#define CNC_DATA        0
#define CNC_STATUS      1
#define CNC_CONTROL     2

#define CNC_OK          0x00
#define CNC_Home        0x20
#define CNC_Overload    0x10
#define CNC_Haube       0x30
#define CNC_OverTemp    0x08
#define CNC_Tool        0x28

// Signal gets inverted, clear the pin at the LPC to het a logic '1' at the port
#define STROBE_PIN      (1<<3)      //Strobe = P0.3
#define STROBE_H() {FIOCLR=STROBE_PIN;}
#define STROBE_L() {FIOSET=STROBE_PIN;}

extern BOOLEAN CNC_Motor;


unsigned char readStrobePin();

void outportb(int port, int data);
int inportb(int port);
int CNC_Status();

// -----------------------------------------------------
#define C_INIT      0xD0
#define C_MOTOR_ON  0x90
#define C_XINC      0x4B
#define C_XDEC      0x09
#define C_YINC      0x4E
#define C_YDEC      0x06
#define C_XINCYINC  0x4F
#define C_XDECYDEC  0x05
#define C_XINCYDEC  0x07
#define C_XDECYINC  0x0D
#define C_ZINC      0x50
#define C_ZDEC      0x30
#define C_NOTBRAKE  0x80
#define C_BRAKE     0xC2

#define C_STATUS    0x00
#define C_DELAY     0xFF

#define N_OUT       45
#define N_Vel       40

extern int delay_buffer[N_OUT];
extern int command_buffer[N_OUT];
extern int speed_buffer[N_OUT];
extern volatile int read_pointer, write_pointer;

extern int CNC_XPos_n, CNC_YPos_n, CNC_ZPos_n;
extern int CNC_Step;
extern unsigned char CNC_SpeedIndex;

extern int isr_cnc_status;
extern volatile int dataCommandReturn;


int dataCommand(int parameter);
void moveX(int steps, int speed);
void moveY(int steps, int speed);
void moveZ(int steps, int speed);
void moveHome();
void moveToXYZ(int x1, int y1, int z1);
char CNC_XYZStep(int x, int y, int z, int max);
void moveToXY(int x1, int y1);
char CNC_XYStep(int x, int y, int max);
void InitiateSpeed(unsigned int vel);
void setParam_cmd(unsigned int command, unsigned int delay);
void moveZUp();
void moveZDown();
void drill();
BOOLEAN CNC_MotorOn_n(int rpm);
void CNC_MotorOff_n();
void CNCInit();
void moveAxis(int command, int distance);
void keepTimerAlive();
```

lib/cncroutines.c

```
#include "cncroutines.h"

BOOLEAN CNC_Motor = FALSE;

unsigned char readStrobePin()
{
    return ((~((FIOPIN & 0b1000) >> 3)) & 1);
}

void outportb(int port, int data)
{
    int b_translated = 0;

    switch (port)
    {
        case CNC_DATA:
            MCPSetPort(PORTA, data);
            #ifdef ROUTINE_DEBUG
            _printf("port:␣%d,␣data:␣0x%x,␣b_translated:␣0x%x\r\n", port, data,
    b_translated);
            #endif
            break;

        case CNC_CONTROL:
            b_translated =  (((data & 0b10) >> 1 ) << 7 ) |     // LINEFEED - AUTOF
                            (((data & 0b1000) >> 3 ) << 4 ) |   // RESET - INIT
                            (((data & 0b100) >> 2 ) << 5 );     // SELECT - SELIN
            MCPSetPort(PORTB, b_translated);
            if ((data & 0b00000001) == 1)
            {
                STROBE_H();
                #ifdef ROUTINE_DEBUG
                _printf("Setting␣strobe␣signal␣high\r\n");
                _printf("bit␣three␣shoulf␣be␣'1':␣0x%x\r\n", FIODIR);
                _printf("(that␣means␣it␣should␣be␣between␣8␣and␣F,␣and␣more␣
    specifically␣it␣should␣be␣0xD)\r\n");
                #endif
            } else {
                STROBE_L();
                #ifdef ROUTINE_DEBUG
                _printf("Clearing␣strobe␣signal␣high\r\n");
                _printf("bit␣three␣shoulf␣be␣'1':␣0x%x\r\n", FIODIR);
                _printf("(that␣means␣it␣should␣be␣between␣8␣and␣F,␣and␣more␣
    specifically␣it␣should␣be␣0xD)\r\n");
                #endif
            }

            #ifdef ROUTINE_DEBUG
            _printf("outportb␣->␣CNC_CONTROL;␣setting␣the␣following␣values:\r\n");
            _printf("AUTOF␣␣:␣%d\r\n", (((data & 0b10) >> 1 ) << 7 ));
            _printf("INIT␣␣␣:␣%d\r\n", (((data & 0b1000) >> 3 ) << 4 ));
            _printf("SELIN␣␣:␣%d\r\n", (((data & 0b100) >> 2 ) << 5 ));
            _printf("STROBE␣:␣%d\r\n\r\n", (data & 0b00000001));
            #endif
            break;
    }
}

int inportb(int port)
{
    unsigned char value, strobeValue = 0;

    int portAData = MCPReadPort(PORTA);
    int portBData = MCPReadPort(PORTB);

    switch (port)
    {
        case CNC_DATA:
            value = portAData;
            #ifdef ROUTINE_DEBUG
            _printf("Converted␣CNC_DATA:␣0x%x\r\n", value);
            #endif
            break;

        case CNC_CONTROL:
            value = (((portBData & 0b10000000) >> 7 ) << 3 ) |  // AUTOF(B7) –
    LINEFEED
                    (((portBData & 0b100000) >> 5 ) << 2) |     // INIT(B5) – RESET
                    (((portBData & 0b10000) >> 4 ) << 3 ) |     // SELIN(B4) –
    SELECT
                    readStrobePin();
            #ifdef ROUTINE_DEBUG
            _printf("Value␣converted␣for␣CNC_CONTROL:␣0x%x,\r\nthat␣means:\r\n",
    value);
            _printf("AUTOF␣␣:␣%d\r\n", ((portBData & 0b10000000) >> 7 ));
            _printf("INIT␣␣␣:␣%d\r\n", ((portBData & 0b100000) >> 5 ));
            _printf("SELIN␣␣:␣%d\r\n", ((portBData & 0b10000) >> 4 ));
            _printf("STROBE␣:␣%d\r\n", (value & 0b00000001));
            #endif
            break;

        case CNC_STATUS:
            value = (((portBData & 0b1000000) >> 6 ) << 3 ) |   // ERROR – GPB6
                    (((portBData & 0b1) >> 0 ) << 4 ) |         // SELECT – GPB0
                    (((portBData & 0b10) >> 1 ) << 5 ) |        // PAPER–OUT – GPB1
                    (((portBData & 0b1000) >> 3 ) << 6 ) |      // ACK – GPB3
                    (((portBData & 0b100) >> 2 ) << 7 );        // BUSY – GPB2
            #ifdef ROUTINE_DEBUG
            _printf("Converted␣CNC_STATUS:␣0x%x\r\n", value);
            #endif
            break;
    }
    return value;
}

int CNC_Status()
{
    unsigned char Reg;

    Reg = inportb(CNC_STATUS);
    Reg = Reg & 0x38;
    return Reg;
```

```
}



// ————————————————————————————————————————————————————————
int delay_buffer[N_OUT];
int command_buffer[N_OUT];
int speed_buffer[N_OUT];

volatile int read_pointer, write_pointer;
int CNC_XPos_n, CNC_YPos_n, CNC_ZPos_n;
int CNC_Step;
unsigned char CNC_SpeedIndex;

int isr_cnc_status;
volatile int dataCommandReturn;



/**
 * @brief Timer0 ISR
 *
 */
void __attribute__ ((interrupt("IRQ"))) IRQ_CNC_Timer_Routine (void) {
    // _printf("RD: %d, WR: %d\n", read_pointer, write_pointer);
    if ( read_pointer != write_pointer )
    {
    T0MR0 = (delay_buffer[read_pointer]);
    T0TC=0;

        if( command_buffer[read_pointer] == C_STATUS )
        {
            isr_cnc_status = CNC_Status();
        }
        else if( command_buffer[read_pointer] == C_DELAY )
        {
            // Do nothing
        }
        else
        {
            dataCommand(command_buffer[read_pointer]);
        }
        read_pointer++;
        if( read_pointer >= N_OUT ) read_pointer = 0;
    }
    T0IR = 1;
    VICVectAddr=-1;
}




// ——> New functions

/**
 * @brief Outputs an 8-bit command to the parallel port, after which the
 * @brief strobe line is but low for 5 Âţs (strobe is high during idle state)
 *
 * @param parameter is the command that will be output to the port
 * @return int serts a global variable to the value of the status register
 */
int dataCommand(int parameter)
{
    outportb(CNC_DATA, parameter);

    STROBE_L();
    _delay_us(5);
    STROBE_H();

    dataCommandReturn = inportb(CNC_STATUS);
    return dataCommandReturn;
}



/**
 * @brief Moves the CNC-head in the X-axis
 *
 * @param steps is the amount of steps the motor will turn, each step is 0,0254 mm
 * @param speed is the delay between this command and the next (determines speed)
 */
void moveX(int steps, int speed)
{
    int w;
    for (w = 0; w < abs(steps); w++)
    {
        if (steps < 0)
        {
            setParam_cmd(C_XDEC, speed);
        } else
        {
            setParam_cmd(C_XINC, speed);
        }
    }
}

void moveY(int steps, int speed)
{
    int w;
    for (w = 0; w < abs(steps); w++)
    {
        if (steps < 0)
        {
            setParam_cmd(C_YDEC, speed);
        } else
        {
            setParam_cmd(C_YINC, speed);
        }
    }
}

void moveZ(int steps, int speed)
{
    int w;
    for (w = 0; w < abs(steps); w++)
    {
        if (steps < 0)
        {
            setParam_cmd(C_ZDEC, speed);
        } else
        {
            setParam_cmd(C_ZINC, speed);
        }
    }
}

// TODO: Not ready yet
/**
 * @brief Moves the spindle up
 *
 */
void moveZUp()
{
    // _printf("PEN UP\n");
    moveAxis(C_ZDEC, 300);  // TODO: figure out what value of steps has to be (
    NEGATIVE, how much??) and how fast?
}

/**
 * @brief Moves the spindle down
 *
 */
void moveZDown()
{
    // _printf("PEN DOWN\n");
    if (CNC_ZPos_n+300 < 1180)
    {
        moveAxis(C_ZINC, 300);  // TODO: figure out what value of steps has to be (
    POSITIVE, how much??) and how fast?
    }

}

void drill()
{
    moveZDown();
    moveZUp();
}

void moveAxis(int command, int distance)
{
    CNC_Step = 0;
    int i = 0;

    switch (command)
    {
    case C_XDEC:
        CNC_XPos_n -= distance;
        break;

    case C_XINC:
        CNC_XPos_n += distance;
        break;

    case C_YDEC:
        CNC_YPos_n -= distance;
        break;

    case C_YINC:
        CNC_YPos_n += distance;
        break;

    case C_ZDEC:
        CNC_ZPos_n -= distance;
        break;

    case C_ZINC:
        CNC_ZPos_n += distance;
        break;

    }
    if (CNC_XPos_n<0) CNC_XPos_n=0;
    if (CNC_YPos_n<0) CNC_YPos_n=0;
    if (CNC_ZPos_n<0) CNC_ZPos_n=0;
// enable timer && check initial delay


    // T0MR0 = 20;
    // enableTimer(0);


    // _printf("RD: %d, WR: %d\n", read_pointer, write_pointer);
    for (i = 0; i < distance; i++)
    {
        if ( i < N_Vel ) CNC_SpeedIndex = i;
        if ( i > distance-N_Vel ) CNC_SpeedIndex = distance-i;
        setParam_cmd( command, (unsigned int)(speed_buffer[CNC_SpeedIndex]) );
        // _printf("t: %d, q: %d , p: %d, c: %d | ", T0TCR, T0MR0, &speed_buffer[i],
     speed_buffer[CNC_SpeedIndex]);
        // _printf("RD: %d, WR: %d\n", read_pointer, write_pointer);
    }
    while(read_pointer != write_pointer){
        if ((delay_buffer[read_pointer]) != speed_buffer[(N_OUT-1)])
        {
            enableTimer(0);
        }
    }
}

/**
 * @brief Moves the CNC-head to [0-0-0]
 *
 */
void moveHome()
{
    int speed = 800;
    do
    {
        setParam_cmd((C_XDECYDEC | C_ZDEC), (speed));
    } while (!(CNC_Status() == CNC_Home || CNC_Status() == CNC_Haube));

    if( CNC_Status() == CNC_Home )
    {
        CNC_XPos_n = 0; CNC_YPos_n = 0; CNC_ZPos_n = 0;
    }
    _printf("Homing␣complete:␣[%3d,␣%3d,␣%2d]\n", CNC_XPos_n, CNC_YPos_n, CNC_ZPos_n
    );
    setParam_cmd(C_DELAY, 50000);
}


void moveToXYZ(int x1, int y1, int z1)
{
    int x = CNC_XPos_n, y = CNC_YPos_n, z = CNC_ZPos_n;
    int dx = x1 - x, dy = y1 - y, dz = z1 - z;
    int sx = (dx > 0 ? 1 : (dx < 0 ? -1 : 0));
    int sy = (dy > 0 ? 1 : (dy < 0 ? -1 : 0));
    int sz = (dz > 0 ? 1 : (dz < 0 ? -1 : 0));
    int ax, ay, az, decx, decy, decz, max;
    unsigned char var;

    CNC_Step = 0;

    dx = abs(dx); dy = abs(dy); dz = abs(dz);
    ax = dx<<1, ay = dy<<1, az = dz<<1;

    max = dx;
    var = 0;
    if ( dy > max ) { max = dy; var = 1; }
    if ( dz > max ) { max = dz; var = 2; }

    switch ( var )
    {
        case 0 :
            for(decy = ay-dx, decz = az-dx; ; x += sx, decy += ay, decz += az)
            {
                if ( !CNC_XYZStep(x, y, z, max) ) break;
                if ( x == x1 ) break;
                if ( decy >= 0 ) { decy -= ax, y += sy; }
                if ( decz >= 0 ) { decz -= ax, z += sz; }
            }
            break;

        case 1 :
            for(decx = ax-dy, decz = az-dx; ; y += sy, decx += ax, decz += az)
            {
                if ( !CNC_XYZStep(x, y, z, max) ) break;
                if ( y == y1 ) break;
                if ( decx >= 0 ) { decx -= ay, x += sx; }
                if ( decz >= 0 ) { decz -= ay, z += sz; }
            }
            break;

        case 2 :
            for(decx = ax-dz, decy = ay-dz; ; z += sz, decx += ax, decy += ay)
            {
                if ( !CNC_XYZStep(x, y, z, max) ) break;
                if ( z == z1 ) break;
                if ( decx >= 0 ) { decx -= az, x += sx; }
                if ( decz >= 0 ) { decy -= az, y += sy; }
            }
            break;
    }
    if (CNC_XPos_n<0) CNC_XPos_n=0;
    if (CNC_YPos_n<0) CNC_YPos_n=0;
    if (CNC_ZPos_n<0) CNC_ZPos_n=0;

    while(read_pointer != write_pointer){
        if ((delay_buffer[read_pointer]) != speed_buffer[(N_OUT-1)])
        {
            enableTimer(0);
        }
    }
}

char CNC_XYZStep(int x, int y, int z, int max)
{
    int dx, dy, dz;
    int delta;
    char command;
    char state = CNC_Status();

    dx = x - CNC_XPos_n; CNC_XPos_n = x;
    dy = y - CNC_YPos_n; CNC_YPos_n = y;

    if ( !dx && !dy && !dz )    return 1;

    command = 0x00;
    if ( dy == 0 )
    {
        if ( dx > 0 )   command = C_XINC;
        if ( dx < 0 )   command = C_XDEC;
    }
    else
    if ( dx == 0 )
    {
        if ( dy > 0 )   command = C_YINC;
        if ( dy < 0 )   command = C_YDEC;
    }
    else
    if ( dx > 0 )
    {
        if ( dy > 0 )   command = C_XINCYINC;
        if ( dy < 0 )   command = C_XDECYDEC;
    }
    else
    if ( dx < 0 )
    {
        if ( dy > 0 )   command = C_XDECYINC;
        if ( dy < 0 )   command = C_XDECYDEC;
    }

    if ( dz > 0 )   command |= C_ZINC;
    if ( dz < 0 )   command = command | C_ZDEC;

    CNC_Step++;

    if ( CNC_Step < N_Vel ) CNC_SpeedIndex = CNC_Step;
    if ( CNC_Step > max-N_Vel ) CNC_SpeedIndex = max-CNC_Step;

    setParam_cmd(command, (speed_buffer[CNC_SpeedIndex]));


    if (state != CNC_Status())
    {   // TODO: Check if these values are correct (0x01 & 0x03)
        if ( (CNC_Status() == 0x01) | (CNC_Status() == 0x03) ) {
            return 0;
        } else return 1;

    }
    else return 1;
}

char CNC_XYStep(int x, int y, int max)
{
    int dx, dy;
    int delta;
    char command;
    char state = CNC_Status();

    dx = x - CNC_XPos_n; CNC_XPos_n = x;
    dy = y - CNC_YPos_n; CNC_YPos_n = y;

    if ( !dx && !dy )    return 1;

    command = 0x00;
    if ( dy == 0 )
    {
        if ( dx > 0 )   command = C_XINC;
        if ( dx < 0 )   command = C_XDEC;
    }
    else
    if ( dx == 0 )
    {
        if ( dy > 0 )   command = C_YINC;
        if ( dy < 0 )   command = C_YDEC;
    }
    else
    if ( dx > 0 )
    {
        if ( dy > 0 )   command = C_XINCYINC;
        if ( dy < 0 )   command = C_XDECYDEC;
    }
    else
    if ( dx < 0 )
    {
        if ( dy > 0 )   command = C_XDECYINC;
        if ( dy < 0 )   command = C_XDECYDEC;
    }

    CNC_Step++;

    if ( CNC_Step < N_Vel ) CNC_SpeedIndex = CNC_Step;
    if ( CNC_Step > max-N_Vel ) CNC_SpeedIndex = max-CNC_Step;

    setParam_cmd(command, (speed_buffer[CNC_SpeedIndex]));


    if (state != CNC_Status())
    {   // TODO: Check if these values are correct (0x01 & 0x03)
        if ( (CNC_Status() == 0x01) | (CNC_Status() == 0x03) ) {
            return 0;
        } else return 1;

    }
    else return 1;
}


void moveToXY(int x1, int y1)
{
    // _printf("Move to function entered. [%2d, %2d], RD{%d} WR{%d}\n", x1, y1,
    read_pointer, write_pointer);
    int x = CNC_XPos_n, y = CNC_YPos_n;
    int dx = x1 - x, dy = y1 - y;
    int sx = (dx > 0 ? 1 : (dx < 0 ? -1 : 0));
    int sy = (dy > 0 ? 1 : (dy < 0 ? -1 : 0));
    int ax, ay, decx, decy, max;
    unsigned char var;

    CNC_Step = 0;

    dx = abs(dx); dy = abs(dy);
    ax = dx<<1, ay = dy<<1;

    max = dx;
    var = 0;
    if ( dy > max ) { max = dy; var = 1; }
    // _printf("1\n");
    switch ( var )
    {
        case 0 :
            for(decy = ay-dx; ; x += sx, decy += ay)
            {
                    // _printf("2\n");
                if ( !CNC_XYStep(x, y, max) ) break;
                if ( x == x1 ) break;
                if ( decy >= 0 ) { decy -= ax, y += sy; }
            }
            break;

        case 1 :
            for(decx = ax-dy; ; y += sy, decx += ax)
            {
                    // _printf("3\n");
                if ( !CNC_XYStep(x, y, max) ) break;
                if ( y == y1 ) break;
                if ( decx >= 0 ) { decx -= ay, x += sx; }
            }
            break;
    }
    if (CNC_XPos_n<0) CNC_XPos_n=0;
    if (CNC_YPos_n<0) CNC_YPos_n=0;



    // _printf("Before enabling, RD{%d} != WR{%d}\n", read_pointer, write_pointer);
    while(read_pointer != write_pointer){
        if ((delay_buffer[read_pointer]) != speed_buffer[(N_OUT-1)])
        {
            // _printf("enabling timer\n");
            enableTimer(0);
        }
    }
}



/**
 * @brief This function should be called once. It is responible for speeding up and slowing down the CNC-head when making a movement.
 *
 * @param vel is a chosen speed, measured in mm/min
 */
void InitiateSpeed(unsigned int vel)
{
    unsigned int i;
    long V;

    // FIFO buffer initialization
    read_pointer = 0;
    write_pointer = 0;
    delay_buffer[read_pointer] = 30;

    // Configure interrupt for TIMER0
    #define TIMER0 (4)
    VICVectAddr1 = (unsigned int)IRQ_CNC_Timer_Routine;
    VICVectCntl1 = (1<<5)|TIMER0; // Enable vector
    VICVectAddr=-1;
    VICIntEnable = (1<<TIMER0); // Enable interrupt

    // Configure timer
    resetTimer(0);     // T0TCR = 2; // Reset TIMER0
        // Time = (T0PR+1)/PCLK
        // 1e-3 = (T0PR+1//(14745600*4)
        // T0PR = 58981
    //T0PR = 58981; // every step is 1 ms
    T0PR = 59; // every step is 1 us
    T0MCR = 7; // Stop and reset counter and generate interrupt
    T0MR0 = 1000; // Period for Timer0 interrupt (first time 1 ms)

    // Enable interrupts
        asm volatile ("mrs␣r0,cpsr\n␣bic␣r0,r0,#0x80\n␣msr␣cpsr,r0");
    STROBE_H();

    // Delay initialization
    // _printf("Delay calculation for a speed of %4d mm/min\r\n", vel);
    // _printf("Delay is measured in us\r\n");
    for ( i = 0; i < N_Vel; i++ )
    {
        V = (long)vel*(i+1)/(long)N_Vel;
        // Velocity = V mm/min x 1 step/0.0254 mm x 1 min/60 sec
                // The inverse is in sec/step ––> 1524000 (usec/step) / V(mm/min)
        V = 1524000/V; // usec/step
        speed_buffer[i] = (int) (V); // measured in usec
        // _printf("[%2d]: %5d\r\n", i, speed_buffer[i]);
    }
}


/**
 * @brief Puts a command and a corresponding delay in the buffers
 *
 * @param command is an 8–bit hex code that controls the CNC machine
 * @param delay is the delay between this command and the next command, measured in Âţs
 */
void setParam_cmd(unsigned int command, unsigned int delay)
{
    int isig;

    command_buffer[write_pointer] = command;
    delay_buffer[write_pointer] = delay;
    // enableTimer(0);

    isig = write_pointer + 1;
    if (isig >= N_OUT) isig = 0;
    do {/*_printf("waiting\n");*/} while (isig == read_pointer); // Wait for space at buffers
    write_pointer = isig;
    enableTimer(0);
    // _printf("ISR0 RD{%d}, WR{%d}\n", read_pointer, write_pointer );
}

void CNCInit()
{
    MCPsetIODirCNC();
    _delay_ms(200);
    STROBE_H();

    outportb(CNC_CONTROL, 4);
    _delay_ms(10);
    outportb(CNC_CONTROL, 0);
    _delay_ms(250);
    outportb(CNC_CONTROL, 4);
    _delay_ms(10);

    setParam_cmd(C_INIT, 20000);
    setParam_cmd(C_INIT, 2000);
    // moveHome();
}

BOOLEAN CNC_MotorOn_n(int rpm)
{
    unsigned char command;
    if(rpm < 30000 || rpm > 60000) return FALSE;
    if(rpm == 60000) command = 0x9F;
    else command = 0x90 | (rpm-30000)/1875;

    setParam_cmd(C_NOTBRAKE, 2000);
    setParam_cmd(command, 50000);                               // 0,05 second delay

    int i;
    for (i = 0; i < 40; i++) setParam_cmd(C_DELAY, 50000);      // 40 times 50ms = 2
     second delay


    if(CNC_Status() == CNC_OK)  CNC_Motor = TRUE;
    else                        CNC_Motor = FALSE;

    return CNC_Motor;
}

void CNC_MotorOff_n()
{
    setParam_cmd(C_BRAKE, 2000);
    setParam_cmd(C_INIT, 50000);

    CNC_Motor = FALSE;
}


void keepTimerAlive()
{
    while(read_pointer != write_pointer){
        if ((delay_buffer[read_pointer]) != speed_buffer[(N_OUT-1)])
        {
            enableTimer(0);
        }
    }
}
```

lib/display_manager.h

```
#ifndef DISPLAY_MANAGER_H
#define DISPLAY_MANAGER_H

typedef enum e_display_states
{
    e_disp_route_pro,
    e_disp_config,
    e_disp_tools,
    e_disp_offset,
    e_disp_warning,
    e_disp_load,
    e_disp_execute_file
} display_states_t;

extern display_states_t g_display_state;
extern display_states_t g_previous_display;
extern char button_interrupt_flag;


void display_manager_init_state();
void display_manager_set_state(display_states_t new_state);
void display_manager_process_state();

void set_display_change_flag();
void clear_display_change_flag();

void set_button_interrupt_flag();
void clear_button_interrupt_flag();

#endif //DISPLAY_MANAGER_H
```

lib/display_manager.c

```
#include "display_manager.h"

display_states_t g_display_state;
display_states_t g_previous_display;
char display_change_flag = 0;
char button_interrupt_flag = 0;


void display_manager_init_state()
{
    g_display_state = e_disp_route_pro;
    display_manager_set_state(g_display_state);
}

void display_manager_set_state(display_states_t new_state)
{
    //If new display is warning and warning is already displayed, do nothing.
    if(new_state == e_disp_warning && g_display_state == e_disp_warning)
    {
    } else
    {
        g_previous_display = g_display_state;
        g_display_state = new_state;
        set_display_change_flag ();
    }

}


void display_manager_process_state()
{
    switch (g_display_state)
    {
        case e_disp_route_pro:
            if(display_change_flag==1) {clear_display_change_flag(); initRP();}
            scanRP();
            break;

        case e_disp_config:
            if(display_change_flag==1) {clear_display_change_flag(); initConf();}
            scanConf();
            break;

        case e_disp_tools:
            if(display_change_flag==1) {clear_display_change_flag(); initTD();}
            scanTD();
            break;

        case e_disp_offset:
            if(display_change_flag==1) {clear_display_change_flag(); initOTS();}
            scanOTS();
            break;

        case e_disp_warning:
            if(display_change_flag==1) {clear_display_change_flag(); initWRNG("
    Emergency␣stop");}
            scanWRNG("Emergency␣stop");
            break;

        case e_disp_load:
            if(display_change_flag==1) {clear_display_change_flag(); initLF();}
            scanLF();
            break;

        case e_disp_execute_file:
            if(display_change_flag==1) {clear_display_change_flag(); initEF();}
            scanEF();
            break;

        default:
            if(display_change_flag==1) {clear_display_change_flag(); initLF();}
            scanRP();
            break;

    }
}

void set_display_change_flag()
{
    display_change_flag = 1;
}

void clear_display_change_flag()
{
    display_change_flag = 0;
}

void set_button_interrupt_flag()
{
    button_interrupt_flag = 1;
}

void clear_button_interrupt_flag()
{
    button_interrupt_flag = 0;
}
```

lib/execute_file.h

```
#pragma once

#include "system.h"
#define SZ                  (80)

extern char fileArray[amountoffiles][amountofcharacters];
extern char dirArray[amountoffiles][amountofcharacters];
extern int fileNumber;
extern int dirNumber;

extern char fileAndInf[amountoffiles][amountofcharacters+7];
extern int faiNumber;


int SDVolumeInit();

void printFile(char filename[]);
void printExcellon(char filename[]);
int readLine(char *line);

void processFile(char filename[]);
void retvalToConsole(int retval);
int parse_hpgl(char * s);
int parse_excellon(char *BUF);

int scanFiles (char* path);
void printFiles();
void clearFileBuffer();
```

lib/execute_file.c

```
#include "execute_file.h"
#define EXCBUFFER (64)

FATFS filesysobj;

char fileArray[amountoffiles][amountofcharacters];
char dirArray[amountoffiles][amountofcharacters];
int fileNumber;
int dirNumber;

char fileAndInf[amountoffiles][amountofcharacters+7];
int faiNumber;

int SDVolumeInit()
{
    int returnV = 0;

    returnV = pf_mount(&filesysobj);
    // if (returnV != 0) retvalToConsole(returnV);
    return returnV;
}


void printFile(char filename[])
{
    CHAR buff[SZ];
    UINT bytes_read, i = 0;
    int returnV = 0;

    _printf("Open␣[%s]\n", filename);
    returnV = pf_open(filename);
    if (returnV != 0) retvalToConsole(returnV);

    while (buff[i] != '\0')
    {
        pf_read(buff, SZ, &bytes_read);
        for (i = 0; i < bytes_read; i++){
            _putch(buff[i]);
        }
    }
}

void printExcellon(char filename[])
{
    char buff[SZ*2];
    UINT bytes_read, i = 0;
    int returnV = 0;

    _printf("Open␣[%s]\n", filename);
    returnV = pf_open(filename);
    if (returnV != 0) retvalToConsole(returnV);

    do {
        returnV = readLine(buff);
        _printf(">%s<\n", buff);

    } while ((strcmp(buff, "M30")) && (returnV != -1));
}

int readLine(char *line)
{
    UINT bytes_read, i = 0;

    while(1) {
        pf_read(&line[i], 1, &bytes_read);
        if (line[i] == '\r') continue;
        if (line[i] == '\n') break;

        i++;
        if (i >= (SZ*2)) return -1;
    }
    line[i] = 0;
    return 0;
}


int scanFiles (char* path)
{
    FRESULT res;
    FILINFO fno;
    DIR dir;
    int i;
    fileNumber = 0;
    dirNumber = 0;

    res = pf_opendir(&dir, path);
    if (res == FR_OK) {
        i = strlen(path);
        for (;;) {
            res = pf_readdir(&dir, &fno);
            if (fno.fattrib & AM_DIR) {
                // Directorys
                if ( !strstr(fno.fname, "SYSTEM") ) strcpy(dirArray[dirNumber++],
    fno.fname);
            } else {
                // Files
                if ( strstr(fno.fname, ".PLT") || strstr(fno.fname, ".NCD") ) strcpy
    (fileArray[fileNumber++], fno.fname);
            }
            if (res != FR_OK || fno.fname[0] == 0) break;
        }
    }
    return res;
}

void printFiles()
{
    int i;
    for (i = 0; i < amountoffiles; i++)
    {
        if (fileArray[i] != 0)
        {
            _printf("%s\n", fileArray[i]);
        }
    }
}


void processFile(char filename[])
{
    UINT bytes_read, i = 0;
    int returnV = 0;

    _printf("Open␣[%s]\n", filename);
    returnV = pf_open(filename);
    if (returnV != 0) retvalToConsole(returnV);

        char buff[SZ*2]="";
        int to_read = 0;

    int len = strlen(filename);

    if (!(strcmp(&filename[len-4], ".PLT")))
    {
        _printf("\nParsing␣HPGL␣file\n");
        do {
            to_read = (SZ*2 - i);
            // Read from file or uart (from ESP) and concatenate No more than SZ
            pf_read(&buff[i], to_read, &bytes_read);
            // Parse. Be careful about parse errors
            i = parse_hpgl(buff); if(i==-1) _printf("HPGL␣format␣error\r\n");
            // Process remainder string
            strncpy(&buff[0],&buff[i],strlen(buff)-i+1);
            // _printf("REMAINDER = %s\r\n", buff);
        } while (bytes_read == to_read );
    } else if (!(strcmp(&filename[len-4], ".NCD")))
    {
        _printf("\nParsing␣excellon␣file\n");
        do {
        returnV = readLine(buff);
        parse_excellon(buff);
        drill();
        } while ((strcmp(buff, "M30")) && (returnV != -1));
    }
    _printf("End␣of␣file␣reached\n");
}


void retvalToConsole(int retval)
{
    switch (retval)
    {
    case 0:
        _printf(">FR_OK␣=␣0,␣␣␣␣␣␣␣␣/*␣0␣-␣The␣function␣succeeded␣*/\n");
        break;
    case 1:
        _printf(">FR_DISK_ERR,␣␣␣␣␣␣/*␣1␣-␣An␣error␣occured␣in␣the␣disk␣read␣
    function␣*/\n");
        break;
    case 2:
        _printf(">FR_NOT_READY,␣␣␣␣␣/*␣2␣-␣The␣storage␣device␣could␣not␣be␣
    initialized␣due␣to␣a␣hard␣error␣or␣no␣medium␣*/\n");
        break;
    case 3:
        _printf(">FR_NO_FILE,␣␣␣␣␣␣␣/*␣3␣-␣Could␣not␣find␣the␣file␣or␣path.*/\n");
        break;
    case 4:
        _printf(">FR_NOT_OPENED,␣␣␣␣/*␣4␣-␣The␣file␣has␣not␣been␣opened.*/\n");
        break;
    case 5:
        _printf(">FR_NOT_ENABLED,␣␣␣/*␣5␣-␣The␣volume␣has␣not␣been␣mounted.*/\n");
        break;
    case 6:
        _printf(">FR_NO_FILESYSTEM␣␣/*␣6␣-␣There␣is␣no␣valid␣FAT␣partition␣on␣the␣
    drive␣*/\n");
        break;
    }
}


int parse_hpgl(char * s)
{
    int vector_counter = 0;
    int pen;
    int xcoord, ycoord, xsteps, ysteps;
    int i0,i1, ic;
    i0 = 0;
    i1 = 0;

    while(1)
    {
        // Search for semicolon. Result at i1
        while(s[i1]!=';')
        if(s[i1++]==0) return i0;
        s[i1]=0;
        // _printf("CMD = %s \r\n", &s[i0]);
        // Process CMD
        if(s[i0]=='P' && s[i0+1]=='U')
        { // Pen Up
            // _printf("-> Pen up\r\n");
            moveZUp();
        }
        else if(s[i0]=='P' && s[i0+1]=='D')
        { // Pen Down
            // _printf("-> Pen down\r\n");
            moveZDown();
        }
        else if(s[i0]=='S' && s[i0+1]=='P')
        { // Set Pen
            pen = atoi(&s[i0+2]);
            _printf("->␣Pen␣%d\r\n", pen);
            // ------------------------------------
            CNC_MotorOff_n();


            char wrng[] = "Change␣tool␣n";
            char tool_n[1];
            BOOLEAN quit = FALSE;
            itoa(pen, tool_n, 10);
            strcat(wrng, tool_n);

            // scanStopScreen(wrng);
            drawWrnSc(wrng);
            while(1){
                // _printf("Hellooooo\n");
                buttonHandler(BUTTON_StopSc, StopScSZ, 8);
                if (controlCommand.state == READY)
                {
                    if (strcmp(controlCommand.command, "Resume") == 0)
                    {
                        controlCommand.state = NOT_READY;
                        drawStartS(303, 84);
                        quit = TRUE;
                    }
                }
                if(quit) break;
            }

            CNC_MotorOn_n(45000);
            updateTool(pen);
            // ------------------------------------------
        }
        else if(s[i0]=='P' && s[i0+1]=='A')
        { // Pen At
            ic=i0+2;
            while(s[ic]!=',') if(s[ic++]==0) return -1; // Error
            xcoord = atoi(&s[i0+2]);
            ycoord = atoi(&s[ic+1]);

            // _printf("-> Pen at (%3d,%3d)\r\n", xcoord,ycoord);
            xsteps = ((xcoord*250)/254);
            ysteps = ((ycoord*250)/254);
            moveToXY(xsteps, ysteps);

            if(vector_counter == 10) {drawPos(303, 84); vector_counter = 0;}
            else vector_counter++;
        }
        s[i1]=';';
        i1++;
        i0=i1;
    }
}


int parse_excellon(char *BUF)
{
    int tool;
    int i,k;
    int x, y;
    char s[9];
    static int format24 = 0; // Needs to be static

    // At excellon files each command is in a complete line ended with newline
    character
    if     (!strcmp(BUF,"M48")) _printf ("->Start␣of␣file\r\n");
    else if(!strcmp(BUF,"M30")) _printf ("->End␣of␣file\n\n");
    else if(BUF[0]=='%') _printf("->End␣of␣header\r\n");
    else if(BUF[0]=='T')
    {
        if(strlen(BUF)==3) // Process Txx tool change commands
        {
            tool= atoi(&BUF[1]);
            _printf("->Select␣tool␣:␣%02d\r\n", tool );
            updateTool(tool);
        } // Ignore commands as T01F00S00C0.020
    }
    else if(BUF[0]=='X') // Drill at
    {
        if(format24) // 24 means 2 digits are integer part and 4 digits are
    fractiona part
        {
            for(i=0;i<7;i++) s[i]=BUF[i+1];
            s[8]=0;
            x = atoi(s);
            for(i=0;i<7;i++) s[i]=BUF[i+9];
            y = atoi(s);
            _printf("->␣Drill␣at␣%3d,%3d\r\n",x,y);
            moveToXY(x,y);
        }
        else // Trailing zeroes included (TZ) INCH 2.3
        {
            k=0;
            s[6]=0;
            for(i=0;BUF[i]!='Y';i++,k++) s[k]=BUF[i+1];
            s[k-1]=0;
            x = atoi(s);

            k=0;
            for(;i<strlen(BUF);i++,k++) s[k]=BUF[i+1];
            s[k]=0;
            y = atoi(s);

            _printf("->␣Drill␣at␣%3d,%3d\r\n", x, y);
            moveToXY(x,y);
        }
    }
    else if(!strcmp(BUF,"INCH,TZ"))
    {
        format24 = 0;
        _printf("->␣Format␣2.3\r\n");
    }
    // Ignore rest of commands
}

void clearFileBuffer()
{
    int i;
    for (i = 0; i < amountoffiles; i++)
    {
        dirArray[i][0] = 0;
        fileArray[i][0] = 0;
        fileAndInf[i][0] = 0;
        yCoords[i] = 300;
    }
}
```

lib/font.c

```
// #pragma once

#if defined(__AVR__)
        #include <avr/pgmspace.h>
        #define fontdatatype const uint8_t
#elif defined(__PIC32MX__)
        #define PROGMEM
        #define fontdatatype const unsigned char
#elif defined(__arm__)
        #define PROGMEM
        #define fontdatatype const unsigned char
#endif

// SmallFont.c
// Font Size    : 8x12
// Memory usage : 1144 bytes
// # characters : 95

fontdatatype SmallFont[1144] PROGMEM={
0x08,0x0C,0x20,0x5F,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // <Space>
0x00,0x00,0x20,0x20,0x20,0x20,0x20,0x20,0x00,0x20,0x00,0x00, // !
0x00,0x28,0x50,0x50,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // "
0x00,0x00,0x28,0x28,0xFC,0x28,0x50,0xFC,0x50,0x50,0x00,0x00, // #
0x00,0x20,0x78,0xA8,0xA0,0x60,0x30,0x28,0xA8,0xF0,0x20,0x00, // $
0x00,0x00,0x48,0xA8,0xB0,0x50,0x28,0x34,0x54,0x48,0x00,0x00, // %
0x00,0x00,0x20,0x50,0x50,0x78,0xA8,0xA8,0x90,0x6C,0x00,0x00, // &
0x00,0x40,0x40,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // '
0x00,0x04,0x08,0x10,0x10,0x10,0x10,0x10,0x10,0x08,0x04,0x00, // (
0x00,0x40,0x20,0x10,0x10,0x10,0x10,0x10,0x10,0x20,0x40,0x00, // )
0x00,0x00,0x00,0x20,0xA8,0x70,0x70,0xA8,0x20,0x00,0x00,0x00, // *
0x00,0x00,0x20,0x20,0x20,0xF8,0x20,0x20,0x20,0x00,0x00,0x00, // +
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x40,0x80, // ,
0x00,0x00,0x00,0x00,0x00,0xF8,0x00,0x00,0x00,0x00,0x00,0x00, // -
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x00,0x00, // .
0x00,0x08,0x10,0x10,0x10,0x20,0x20,0x40,0x40,0x40,0x80,0x00, // /
0x00,0x00,0x70,0x88,0x88,0x88,0x88,0x88,0x88,0x70,0x00,0x00, // 0
0x00,0x00,0x20,0x60,0x20,0x20,0x20,0x20,0x20,0x70,0x00,0x00, // 1
0x00,0x00,0x70,0x88,0x88,0x10,0x20,0x40,0x80,0xF8,0x00,0x00, // 2
0x00,0x00,0x70,0x88,0x08,0x30,0x08,0x08,0x88,0x70,0x00,0x00, // 3
0x00,0x00,0x10,0x30,0x50,0x50,0x90,0x78,0x10,0x18,0x00,0x00, // 4
0x00,0x00,0xF8,0x80,0x80,0xF0,0x08,0x08,0x88,0x70,0x00,0x00, // 5
0x00,0x00,0x70,0x90,0x80,0xF0,0x88,0x88,0x88,0x70,0x00,0x00, // 6
0x00,0x00,0xF8,0x90,0x10,0x20,0x20,0x20,0x20,0x20,0x00,0x00, // 7
0x00,0x00,0x70,0x88,0x88,0x70,0x88,0x88,0x88,0x70,0x00,0x00, // 8
0x00,0x00,0x70,0x88,0x88,0x88,0x78,0x08,0x48,0x70,0x00,0x00, // 9
0x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,0x00,0x20,0x00,0x00, // :
0x00,0x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,0x20,0x20,0x00, // ;
0x00,0x04,0x08,0x10,0x20,0x40,0x20,0x10,0x08,0x04,0x00,0x00, // <
0x00,0x00,0x00,0x00,0xF8,0x00,0x00,0xF8,0x00,0x00,0x00,0x00, // =
0x00,0x40,0x20,0x10,0x08,0x04,0x08,0x10,0x20,0x40,0x00,0x00, // >
0x00,0x00,0x70,0x88,0x88,0x10,0x20,0x20,0x00,0x20,0x00,0x00, // ?
0x00,0x00,0x70,0x88,0x98,0xA8,0xA8,0xB8,0x80,0x78,0x00,0x00, // @
0x00,0x00,0x20,0x20,0x30,0x50,0x50,0x78,0x48,0xCC,0x00,0x00, // A
0x00,0x00,0xF0,0x48,0x48,0x70,0x48,0x48,0x48,0xF0,0x00,0x00, // B
```

```
0x00,0x00,0x78,0x88,0x80,0x80,0x80,0x80,0x88,0x70,0x00,0x00, // C
0x00,0x00,0xF0,0x48,0x48,0x48,0x48,0x48,0x48,0xF0,0x00,0x00, // D
0x00,0x00,0xF8,0x48,0x50,0x70,0x50,0x40,0x48,0xF8,0x00,0x00, // E
0x00,0x00,0xF8,0x48,0x50,0x70,0x50,0x40,0x40,0xE0,0x00,0x00, // F
0x00,0x00,0x38,0x48,0x80,0x80,0x9C,0x88,0x48,0x30,0x00,0x00, // G
0x00,0x00,0xCC,0x48,0x48,0x78,0x48,0x48,0x48,0xCC,0x00,0x00, // H
0x00,0x00,0xF8,0x20,0x20,0x20,0x20,0x20,0x20,0xF8,0x00,0x00, // I
0x00,0x00,0x7C,0x10,0x10,0x10,0x10,0x10,0x10,0x90,0xE0,0x00, // J
0x00,0x00,0xEC,0x48,0x50,0x60,0x50,0x50,0x48,0xEC,0x00,0x00, // K
0x00,0x00,0xE0,0x40,0x40,0x40,0x40,0x40,0x44,0xFC,0x00,0x00, // L
0x00,0x00,0xD8,0xD8,0xD8,0xD8,0xA8,0xA8,0xA8,0xA8,0x00,0x00, // M
0x00,0x00,0xDC,0x48,0x68,0x68,0x58,0x58,0x48,0xE8,0x00,0x00, // N
0x00,0x00,0x70,0x88,0x88,0x88,0x88,0x88,0x88,0x70,0x00,0x00, // O
0x00,0x00,0xF0,0x48,0x48,0x70,0x40,0x40,0x40,0xE0,0x00,0x00, // P
0x00,0x00,0x70,0x88,0x88,0x88,0x88,0xE8,0x98,0x70,0x18,0x00, // Q
0x00,0x00,0xF0,0x48,0x48,0x70,0x50,0x48,0x48,0xEC,0x00,0x00, // R
0x00,0x00,0x78,0x88,0x80,0x60,0x10,0x08,0x88,0xF0,0x00,0x00, // S
0x00,0x00,0xF8,0xA8,0x20,0x20,0x20,0x20,0x20,0x70,0x00,0x00, // T
0x00,0x00,0xCC,0x48,0x48,0x48,0x48,0x48,0x48,0x30,0x00,0x00, // U
0x00,0x00,0xCC,0x48,0x48,0x50,0x50,0x30,0x20,0x20,0x00,0x00, // V
0x00,0x00,0xA8,0xA8,0xA8,0x70,0x50,0x50,0x50,0x50,0x00,0x00, // W
0x00,0x00,0xD8,0x50,0x50,0x20,0x20,0x50,0x50,0xD8,0x00,0x00, // X
0x00,0x00,0xD8,0x50,0x50,0x20,0x20,0x20,0x20,0x70,0x00,0x00, // Y
0x00,0x00,0xF8,0x90,0x10,0x20,0x20,0x40,0x48,0xF8,0x00,0x00, // Z
0x00,0x38,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x38,0x00, // [
0x00,0x40,0x40,0x40,0x20,0x20,0x10,0x10,0x10,0x08,0x00,0x00, // <Backslash>
0x00,0x70,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x70,0x00, // ]
0x00,0x20,0x50,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ^
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFC, // _
0x00,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // '
0x00,0x00,0x00,0x00,0x00,0x30,0x48,0x38,0x48,0x3C,0x00,0x00, // a
0x00,0x00,0xC0,0x40,0x40,0x70,0x48,0x48,0x48,0x70,0x00,0x00, // b
0x00,0x00,0x00,0x00,0x00,0x38,0x48,0x40,0x40,0x38,0x00,0x00, // c
0x00,0x00,0x18,0x08,0x08,0x38,0x48,0x48,0x48,0x3C,0x00,0x00, // d
0x00,0x00,0x00,0x00,0x00,0x30,0x48,0x78,0x40,0x38,0x00,0x00, // e
0x00,0x00,0x1C,0x20,0x20,0x78,0x20,0x20,0x20,0x78,0x00,0x00, // f
0x00,0x00,0x00,0x00,0x00,0x3C,0x48,0x30,0x40,0x78,0x44,0x38, // g
0x00,0x00,0xC0,0x40,0x40,0x70,0x48,0x48,0x48,0xEC,0x00,0x00, // h
0x00,0x00,0x20,0x00,0x00,0x60,0x20,0x20,0x20,0x70,0x00,0x00, // i
0x00,0x00,0x10,0x00,0x00,0x30,0x10,0x10,0x10,0x10,0x10,0xE0, // j
0x00,0x00,0xC0,0x40,0x40,0x5C,0x50,0x70,0x48,0xEC,0x00,0x00, // k
0x00,0x00,0xE0,0x20,0x20,0x20,0x20,0x20,0x20,0xF8,0x00,0x00, // l
0x00,0x00,0x00,0x00,0x00,0xF0,0xA8,0xA8,0xA8,0xA8,0x00,0x00, // m
0x00,0x00,0x00,0x00,0x00,0xF0,0x48,0x48,0x48,0xEC,0x00,0x00, // n
0x00,0x00,0x00,0x00,0x00,0x30,0x48,0x48,0x48,0x30,0x00,0x00, // o
0x00,0x00,0x00,0x00,0x00,0xF0,0x48,0x48,0x48,0x70,0x40,0xE0, // p
0x00,0x00,0x00,0x00,0x00,0x38,0x48,0x48,0x48,0x38,0x08,0x1C, // q
0x00,0x00,0x00,0x00,0x00,0xD8,0x60,0x40,0x40,0xE0,0x00,0x00, // r
0x00,0x00,0x00,0x00,0x00,0x78,0x40,0x30,0x08,0x78,0x00,0x00, // s
0x00,0x00,0x00,0x20,0x20,0x70,0x20,0x20,0x20,0x18,0x00,0x00, // t
0x00,0x00,0x00,0x00,0x00,0xD8,0x48,0x48,0x48,0x3C,0x00,0x00, // u
0x00,0x00,0x00,0x00,0x00,0xEC,0x48,0x50,0x30,0x20,0x00,0x00, // v
0x00,0x00,0x00,0x00,0x00,0xA8,0xA8,0x70,0x50,0x50,0x00,0x00, // w
0x00,0x00,0x00,0x00,0x00,0xD8,0x50,0x20,0x50,0xD8,0x00,0x00, // x
0x00,0x00,0x00,0x00,0x00,0xEC,0x48,0x50,0x30,0x20,0x20,0xC0, // y
0x00,0x00,0x00,0x00,0x00,0x78,0x10,0x20,0x20,0x78,0x00,0x00, // z
0x00,0x18,0x10,0x10,0x10,0x20,0x10,0x10,0x10,0x10,0x18,0x00, // {
```

```
0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10, // |
0x00,0x60,0x20,0x20,0x20,0x10,0x20,0x20,0x20,0x20,0x60,0x00, // }
0x40,0xA4,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ~
};

// BigFont.c (C)2010 by Henning Karlsen
// Font Size    : 16x16
// Memory usage : 3044 bytes
// # characters : 95

fontdatatype BigFont[3044] PROGMEM={
0x10,0x10,0x20,0x5F,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
     <Space>
0x00,0x00,0x00,0x00,0x07,0x00,0x0F,0x80,0x0F,0x80,0x0F,0x80,0x0F,0x80,0x0F,0x80,0x07
    ,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x00,0x00, //
    !
0x00,0x00,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x06,0x30,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    "
0x00,0x00,0x0C,0x30,0x0C,0x30,0x0C,0x30,0x7F,0xFE,0x7F,0xFE,0x0C,0x30,0x0C,0x30,0x0C
    ,0x30,0x0C,0x30,0x7F,0xFE,0x7F,0xFE,0x0C,0x30,0x0C,0x30,0x0C,0x30,0x00,0x00, //
    #
0x00,0x00,0x02,0x40,0x02,0x40,0x0F,0xF8,0x1F,0xF8,0x1A,0x40,0x1A,0x40,0x1F,0xF0,0x0F
    ,0xF8,0x02,0x58,0x02,0x58,0x1F,0xF8,0x1F,0xF0,0x02,0x40,0x02,0x40,0x00,0x00, //
    $
0x00,0x00,0x00,0x00,0x00,0x00,0x0E,0x10,0x0E,0x30,0x0E,0x70,0x00,0xE0,0x01,0xC0,0x03
    ,0x80,0x07,0x00,0x0E,0x70,0x0C,0x70,0x08,0x70,0x00,0x00,0x00,0x00,0x00,0x00, //
    %
0x00,0x00,0x00,0x00,0x0F,0x00,0x19,0x80,0x19,0x80,0x19,0x80,0x0F,0x00,0x0F,0x08,0x0F
    ,0x98,0x19,0xF8,0x18,0xF0,0x18,0xE0,0x19,0xF0,0x0F,0x98,0x00,0x00,0x00,0x00, //
    &
0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x0E,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    '
0x00,0x00,0x00,0x00,0x00,0xF0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x0E,0x00,0x0E
    ,0x00,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC0,0x00,0xF0,0x00,0x00,0x00,0x00, //
    (
0x00,0x00,0x00,0x00,0x0F,0x00,0x03,0x80,0x01,0xC0,0x00,0xE0,0x00,0x70,0x00,0x70,0x00
    ,0x70,0x00,0x70,0x00,0xE0,0x01,0xC0,0x03,0x80,0x0F,0x00,0x00,0x00,0x00,0x00, //
    )
0x00,0x00,0x00,0x00,0x01,0x80,0x11,0x88,0x09,0x90,0x07,0xE0,0x07,0xE0,0x3F,0xFC,0x3F
    ,0xFC,0x07,0xE0,0x07,0xE0,0x09,0x90,0x11,0x88,0x01,0x80,0x00,0x00,0x00,0x00, //
    *
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x80,0x01,0x80,0x01,0x80,0x0F,0xF0,0x0F
    ,0xF0,0x01,0x80,0x01,0x80,0x01,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    +
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x0E,0x00,0x00,0x00, //
    ,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xF8,0x1F
    ,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    —
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x00,0x00,0x00,0x00, //
    ,
0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x06,0x00,0x0E,0x00,0x1C,0x00,0x38,0x00,0x70,0x00
    ,0xE0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x1C,0x00,0x00,0x00,0x00,0x00, //
    /

0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x78,0x1C,0xF8,0x1C,0xF8,0x1D,0xB8,0x1D
    ,0xB8,0x1F,0x38,0x1F,0x38,0x1E,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, //
    0
0x00,0x00,0x00,0x00,0x01,0x80,0x01,0x80,0x03,0x80,0x1F,0x80,0x1F,0x80,0x03,0x80,0x03
    ,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x1F,0xF0,0x00,0x00,0x00,0x00, //
    1
0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x38,0x00,0x38,0x00,0x70,0x00,0xE0,0x01
    ,0xC0,0x03,0x80,0x07,0x00,0x0E,0x38,0x1C,0x38,0x1F,0xF8,0x00,0x00,0x00,0x00, //
    2
0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x38,0x00,0x38,0x00,0x70,0x03,0xC0,0x03
    ,0xC0,0x00,0x70,0x00,0x38,0x1C,0x38,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, //
    3
0x00,0x00,0x00,0x00,0x00,0xE0,0x01,0xE0,0x03,0xE0,0x06,0xE0,0x0C,0xE0,0x18,0xE0,0x1F
    ,0xF8,0x1F,0xF8,0x00,0xE0,0x00,0xE0,0x00,0xE0,0x03,0xF8,0x00,0x00,0x00,0x00, //
    4
0x00,0x00,0x00,0x00,0x1F,0xF8,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1F,0xE0,0x1F
    ,0xF0,0x00,0x78,0x00,0x38,0x1C,0x38,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, //
    5
0x00,0x00,0x00,0x00,0x03,0xE0,0x07,0x00,0x0E,0x00,0x1C,0x00,0x1C,0x00,0x1F,0xF0,0x1F
    ,0xF8,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, //
    6
0x00,0x00,0x00,0x00,0x1F,0xFC,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x00,0x1C,0x00,0x38,0x00
    ,0x70,0x00,0xE0,0x01,0xC0,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00,0x00, //
    7
0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1F,0x38,0x07,0xE0,0x07
    ,0xE0,0x1C,0xF8,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, //
    8
0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1F,0xF8,0x0F
    ,0xF8,0x00,0x38,0x00,0x38,0x00,0x70,0x00,0xE0,0x07,0xC0,0x00,0x00,0x00,0x00, //
    9
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00
    ,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    :
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x00,0x00,0x00
    ,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    ;
0x00,0x00,0x00,0x70,0x00,0xE0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x1C,0x00,0x1C
    ,0x00,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC0,0x00,0xE0,0x00,0x70,0x00,0x00, //
    <
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3F,0xFC,0x3F,0xFC,0x00,0x00,0x00
    ,0x00,0x3F,0xFC,0x3F,0xFC,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, //
    =
0x00,0x00,0x1C,0x00,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC0,0x00,0xE0,0x00,0x70,0x00
    ,0x70,0x00,0xE0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x00,0x1C,0x00,0x00,0x00, //
    >
0x00,0x00,0x03,0xC0,0x0F,0xF0,0x1E,0x78,0x18,0x38,0x00,0x38,0x00,0x70,0x00,0xE0,0x01
    ,0xC0,0x01,0xC0,0x00,0x00,0x00,0x00,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0x00, //
    ?

0x00,0x00,0x0F,0xF8,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0xFC,0x1C,0xFC,0x1C
    ,0xFC,0x1C,0xFC,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1F,0xF0,0x07,0xF8,0x00,0x00, //
    @
0x00,0x00,0x00,0x00,0x03,0xC0,0x07,0xE0,0x0E,0x70,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C
```

```
,0x38,0x1F,0xF8,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x00,0x00,0x00,0x00, // A
0x00,0x00,0x00,0x00,0x1F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1F,0xF0,0x00,0x00,0x00,0x00, // B
0x00,0x00,0x00,0x00,0x07,0xF0,0x0E,0x38,0x1C,0x38,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0x38,0x0E,0x38,0x07,0xF0,0x00,0x00,0x00,0x00, // C
0x00,0x00,0x00,0x00,0x1F,0xE0,0x0E,0x70,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x70,0x1F,0xE0,0x00,0x00,0x00,0x00, // D
0x00,0x00,0x00,0x00,0x1F,0xF8,0x0E,0x18,0x0E,0x08,0x0E,0x00,0x0E,0x30,0x0F,0xF0,0x0F,0xF0,0x0E,0x30,0x0E,0x00,0x0E,0x08,0x0E,0x18,0x1F,0xF8,0x00,0x00,0x00,0x00, // E
0x00,0x00,0x00,0x00,0x1F,0xF8,0x0E,0x18,0x0E,0x08,0x0E,0x00,0x0E,0x30,0x0F,0xF0,0x0F,0xF0,0x0E,0x30,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x1F,0x00,0x00,0x00,0x00,0x00, // F
0x00,0x00,0x00,0x00,0x07,0xF0,0x0E,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x00,0x1C,0x00,0x1C,0x00,0x1C,0xF8,0x1C,0x38,0x1C,0x38,0x0E,0x38,0x07,0xF8,0x00,0x00,0x00,0x00, // G
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1F,0xF0,0x1F,0xF0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x00,0x00,0x00,0x00, // H
0x00,0x00,0x00,0x00,0x0F,0xE0,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x0F,0xE0,0x00,0x00,0x00,0x00, // I
0x00,0x00,0x00,0x00,0x01,0xFC,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x38,0x70,0x38,0x70,0x38,0x70,0x38,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // J
0x00,0x00,0x00,0x00,0x1E,0x38,0x0E,0x38,0x0E,0x70,0x0E,0xE0,0x0F,0xC0,0x0F,0x80,0x0F,0x80,0x0F,0xC0,0x0E,0xE0,0x0E,0x70,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, // K
0x00,0x00,0x00,0x00,0x1F,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x08,0x0E,0x18,0x0E,0x38,0x1F,0xF8,0x00,0x00,0x00,0x00, // L
0x00,0x00,0x00,0x00,0x1C,0x1C,0x1E,0x3C,0x1F,0x7C,0x1F,0xFC,0x1F,0xFC,0x1D,0xDC,0x1C,0x9C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x00,0x00,0x00,0x00, // M
0x00,0x00,0x00,0x00,0x1C,0x1C,0x1C,0x1C,0x1E,0x1C,0x1F,0x1C,0x1F,0x9C,0x1D,0xDC,0x1C,0xFC,0x1C,0x7C,0x1C,0x3C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x00,0x00,0x00,0x00, // N
0x00,0x00,0x00,0x00,0x03,0xE0,0x07,0xF0,0x0E,0x38,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x0E,0x38,0x07,0xF0,0x03,0xE0,0x00,0x00,0x00,0x00, // O

0x00,0x00,0x00,0x00,0x1F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0F,0xF0,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x1F,0x00,0x00,0x00,0x00,0x00, // P
0x00,0x00,0x00,0x00,0x03,0xE0,0x0F,0x78,0x0E,0x38,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x7C,0x1C,0xFC,0x0F,0xF8,0x0F,0xF8,0x00,0x38,0x00,0xFC,0x00,0x00, // Q
0x00,0x00,0x00,0x00,0x1F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0F,0xF0,0x0E,0x70,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, // R
0x00,0x00,0x00,0x00,0x0F,0xF0,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x00,0x0F,0xE0,0x07,0xF0,0x00,0x38,0x1C,0x38,0x1C,0x38,0x1C,0x38,0x0F,0xF0,0x00,0x00,0x00,0x00, // S
```

```
0x00,0x00,0x00,0x00,0x1F,0xFC,0x19,0xCC,0x11,0xC4,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x07,0xF0,0x00,0x00,0x00,0x00, // T
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // U
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC0,0x03,0x80,0x00,0x00,0x00,0x00, // V
0x00,0x00,0x00,0x00,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x1C,0x9C,0x1C,0x9C,0x1C,0x9C,0x0F,0xF8,0x0F,0xF8,0x07,0x70,0x07,0x70,0x00,0x00,0x00,0x00, // W
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC0,0x03,0x80,0x03,0x80,0x07,0xC0,0x0E,0xE0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x00,0x00,0x00,0x00, // X
0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC0,0x03,0x80,0x03,0x80,0x03,0x80,0x03,0x80,0x0F,0xE0,0x00,0x00,0x00,0x00, // Y
0x00,0x00,0x00,0x00,0x1F,0xF8,0x1C,0x38,0x18,0x38,0x10,0x70,0x00,0xE0,0x01,0xC0,0x03,0x80,0x07,0x00,0x0E,0x08,0x1C,0x18,0x1C,0x38,0x1F,0xF8,0x00,0x00,0x00,0x00, // Z
0x00,0x00,0x00,0x00,0x07,0xF0,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0xF0,0x00,0x00,0x00,0x00, // [
0x00,0x00,0x00,0x00,0x10,0x00,0x18,0x00,0x1C,0x00,0x0E,0x00,0x07,0x00,0x03,0x80,0x01,0xC0,0x00,0xE0,0x00,0x70,0x00,0x38,0x00,0x1C,0x00,0x07,0x00,0x00,0x00,0x00, // <Backslash>
0x00,0x00,0x00,0x00,0x07,0xF0,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x07,0xF0,0x00,0x00,0x00,0x00, // ]
0x00,0x00,0x01,0x80,0x03,0xC0,0x07,0xE0,0x0E,0x70,0x1C,0x38,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // ^
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0xFF,0x7F,0xFF, // _
0x00,0x00,0x00,0x00,0x1C,0x00,0x1C,0x00,0x07,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, // '
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x00,0x70,0x00,0x70,0x0F,0xF0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xD8,0x00,0x00,0x00,0x00, // a
0x00,0x00,0x00,0x00,0x1E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0F,0xF0,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1B,0xF0,0x00,0x00,0x00,0x00, // b
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x70,0x1C,0x00,0x1C,0x00,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // c
0x00,0x00,0x00,0x00,0x00,0xF8,0x00,0x70,0x00,0x70,0x00,0x70,0x0F,0xF0,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xD8,0x00,0x00,0x00,0x00, // d
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C,0x70,0x1F,0xF0,0x1C,0x00,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, // e
0x00,0x00,0x00,0x00,0x03,0xE0,0x07,0x70,0x07,0x70,0x07,0x00,0x07,0x00,0x1F,0xE0,0x1F,0xE0,0x07,0x00,0x07,0x00,0x07,0x00,0x07,0x00,0x1F,0xC0,0x00,0x00,0x00,0x00, //
```

```
    f
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xD8,0x1C,0x70,0x1C
    ,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xF0,0x07,0xF0,0x00,0x70,0x1C,0x70,0x0F,0xE0, //
    g
0x00,0x00,0x00,0x00,0x1E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0xF0,0x0F,0x38,0x0F
    ,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, //
    h
0x00,0x00,0x00,0x00,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0x00,0x0F,0xC0,0x01,0xC0,0x01
    ,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x0F,0xF8,0x00,0x00,0x00,0x00, //
    i
0x00,0x00,0x00,0x00,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x00,0x03,0xF0,0x00,0x70,0x00
    ,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x00,0x70,0x1C,0x70,0x0C,0xF0,0x07,0xE0, //
    j
0x00,0x00,0x00,0x00,0x1E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x38,0x0E,0x70,0x0E
    ,0xE0,0x0F,0xC0,0x0E,0xE0,0x0E,0x70,0x0E,0x38,0x1E,0x38,0x00,0x00,0x00,0x00, //
    k
0x00,0x00,0x00,0x00,0x0F,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01
    ,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x0F,0xF8,0x00,0x00,0x00,0x00, //
    l
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xF8,0x1C,0x9C,0x1C
    ,0x9C,0x1C,0x9C,0x1C,0x9C,0x1C,0x9C,0x1C,0x9C,0x1C,0x9C,0x00,0x00,0x00,0x00, //
    m
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xE0,0x1C,0x70,0x1C
    ,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x00,0x00,0x00,0x00, //
    n
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x70,0x1C
    ,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, //
    o

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1B,0xF0,0x0E,0x38,0x0E
    ,0x38,0x0E,0x38,0x0E,0x38,0x0E,0x38,0x0F,0xF0,0x0E,0x00,0x0E,0x00,0x1F,0x00, //
    p
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xB0,0x38,0xE0,0x38
    ,0xE0,0x38,0xE0,0x38,0xE0,0x38,0xE0,0x1F,0xE0,0x00,0xE0,0x00,0xE0,0x01,0xF0, //
    q
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1E,0xF0,0x0F,0xF8,0x0F
    ,0x38,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x0E,0x00,0x1F,0x00,0x00,0x00,0x00,0x00, //
    r
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0F,0xE0,0x1C,0x30,0x1C
    ,0x30,0x0F,0x80,0x03,0xE0,0x18,0x70,0x18,0x70,0x0F,0xE0,0x00,0x00,0x00,0x00, //
    s
0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x03,0x00,0x07,0x00,0x1F,0xF0,0x07,0x00,0x07
    ,0x00,0x07,0x00,0x07,0x00,0x07,0x70,0x07,0x70,0x03,0xE0,0x00,0x00,0x00,0x00, //
    t
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C
    ,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x1C,0x70,0x0F,0xD8,0x00,0x00,0x00,0x00, //
    u
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x70,0x1C,0x70,0x1C
    ,0x70,0x1C,0x70,0x1C,0x70,0x0E,0xE0,0x07,0xC0,0x03,0x80,0x00,0x00,0x00,0x00, //
    v
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x1C,0x1C,0x1C
    ,0x1C,0x1C,0x9C,0x1C,0x9C,0x0F,0xF8,0x07,0x70,0x07,0x70,0x00,0x00,0x00,0x00, //
    w
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0xE0,0x1C,0xE0,0x0F
    ,0xC0,0x07,0x80,0x07,0x80,0x0F,0xC0,0x1C,0xE0,0x1C,0xE0,0x00,0x00,0x00,0x00, //
    x
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0E,0x38,0x0E,0x38,0x0E
```

```
    ,0x38,0x0E,0x38,0x0E,0x38,0x07,0xF0,0x03,0xE0,0x00,0xE0,0x01,0xC0,0x1F,0x80, //
    y
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xE0,0x18,0xE0,0x11
    ,0xC0,0x03,0x80,0x07,0x00,0x0E,0x20,0x1C,0x60,0x1F,0xE0,0x00,0x00,0x00,0x00, //
    z
0x00,0x00,0x00,0x00,0x01,0xF8,0x03,0x80,0x03,0x80,0x03,0x80,0x07,0x00,0x1C,0x00,0x1C
    ,0x00,0x07,0x00,0x03,0x80,0x03,0x80,0x03,0x80,0x01,0xF8,0x00,0x00,0x00,0x00, //
    {
0x00,0x00,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01
    ,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0x00, //
    |
0x00,0x00,0x00,0x00,0x1F,0x80,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x00,0xE0,0x00,0x38,0x00
    ,0x38,0x00,0xE0,0x01,0xC0,0x01,0xC0,0x01,0xC0,0x1F,0x80,0x00,0x00,0x00,0x00, //
    }
0x00,0x00,0x00,0x00,0x1F,0x1C,0x3B,0x9C,0x39,0xDC,0x38,0xF8,0x00,0x00,0x00,0x00,0x00
    ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00  //
    ~
};

// SevenSegNumFont.c
// Font Size    : 32x50
// Memory usage : 2004 bytes
// # characters : 10

// fontdatatype SevenSegNumFont[2004] PROGMEM={
// 0x20,0x32,0x30,0x0A,
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x60,0x0C,0xFF,0xFE,0xF0,0x1E,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3E,0x00,0x00,0x78,0x38,0x00,0x00,0x18,0x20,0x00,0x00,0x08,0
    x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,0x38,0x00,0x00,0x18,0x3E,0x00,0x00,0x78,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x1E,0x00,0x00,0xF0,0x0C,0xFF,0xFE,0x60,0
    x01,0xFF,0xFF,0x00,0x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x00,0xFF,0xFE,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 0
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x60,0x00,0x00,0x00,0xF0,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x00,0x78,0x00,0x00,0x00,0x18,0x00,0x00,0x00,0x08,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x00,0x00,0x00,0x78,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x00,0xF0,0x00,0x00,0x00,0x60,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 1
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x60,0x00,0xFF,0xFE,0xF0,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x00,0x78,0x01,0xFF,0xFE,0x18,0x03,0xFF,0xFF,0x88,0
    x0F,0xFF,0xFF,0xE0,0x27,0xFF,0xFF,0xC0,0x39,0xFF,0xFF,0x00,0x3E,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x1E,0x00,0x00,0x00,0x0C,0xFF,0xFE,0x00,0
    x01,0xFF,0xFF,0x00,0x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x00,0xFF,0xFE,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 2
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x60,0x00,0xFF,0xFE,0xF0,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x00,0x78,0x01,0xFF,0xFE,0x18,0x03,0xFF,0xFF,0x88,0
    x0F,0xFF,0xFF,0xE0,0x07,0xFF,0xFF,0xC0,0x01,0xFF,0xFF,0x18,0x00,0x00,0x00,0x78,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x00,0xF0,0x00,0xFF,0xFE,0x60,0
    x01,0xFF,0xFF,0x00,0x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x00,0xFF,0xFE,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 3
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x60,0x0C,0x00,0x00,0xF0,0x1E,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3E,0x00,0x00,0x78,0x39,0xFF,0xFE,0x18,0x23,0xFF,0xFF,0x88,0
    x0F,0xFF,0xFF,0xE0,0x07,0xFF,0xFF,0xC0,0x01,0xFF,0xFF,0x18,0x00,0x00,0x00,0x78,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x00,0xF0,0x00,0x00,0x00,0x60,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 4
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x0C,0xFF,0xFE,0x00,0x1E,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3E,0x00,0x00,0x00,0x39,0xFF,0xFE,0x00,0x23,0xFF,0xFF,0x80,0
    x0F,0xFF,0xFF,0xE0,0x07,0xFF,0xFF,0xC0,0x01,0xFF,0xFF,0x18,0x00,0x00,0x00,0x78,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x00,0xF0,0x00,0xFF,0xFE,0x60,0
    x01,0xFF,0xFF,0x00,0x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x00,0xFF,0xFE,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 5
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x0C,0xFF,0xFE,0x00,0x1E,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0
    x3F,0x00,0x00,0x00,0x3E,0x00,0x00,0x00,0x39,0xFF,0xFE,0x00,0x23,0xFF,0xFF,0x80,0
    x0F,0xFF,0xFF,0xE0,0x27,0xFF,0xFF,0xC0,0x39,0xFF,0xFF,0x18,0x3E,0x00,0x00,0x78,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x1E,0x00,0x00,0xF0,0x0C,0xFF,0xFE,0x60,0
    x01,0xFF,0xFF,0x00,0x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x00,0xFF,0xFE,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 6
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x60,0x00,0xFF,0xFE,0xF0,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x00,0x78,0x00,0x00,0x00,0x18,0x00,0x00,0x00,0x08,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x00,0x00,0x00,0x78,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x00,0xF0,0x00,0x00,0x00,0x60,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 7
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x60,0x0C,0xFF,0xFE,0xF0,0x1E,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3E,0x00,0x00,0x78,0x39,0xFF,0xFE,0x18,0x23,0xFF,0xFF,0x88,0
    x0F,0xFF,0xFF,0xE0,0x27,0xFF,0xFF,0xC0,0x39,0xFF,0xFF,0x18,0x3E,0x00,0x00,0x78,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x1E,0x00,0x00,0xF0,0x0C,0xFF,0xFE,0x60,0
    x01,0xFF,0xFF,0x00,0x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x00,0xFF,0xFE,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 8
// 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,0x00,0x01,0xFF,0xFF,0x00,0
    x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x60,0x0C,0xFF,0xFE,0xF0,0x1E,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0x3F,0x00,0x01,0xF8,0
    x3F,0x00,0x01,0xF8,0x3E,0x00,0x00,0x78,0x39,0xFF,0xFE,0x18,0x23,0xFF,0xFF,0x88,0
    x0F,0xFF,0xFF,0xE0,0x07,0xFF,0xFF,0xC0,0x01,0xFF,0xFF,0x18,0x00,0x00,0x00,0x78,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0
    x00,0x00,0x01,0xF8,0x00,0x00,0x01,0xF8,0x00,0x00,0x00,0xF0,0x00,0xFF,0xFE,0x60,0
    x01,0xFF,0xFF,0x00,0x03,0xFF,0xFF,0x80,0x01,0xFF,0xFF,0x00,0x00,0xFF,0xFE,0x00,0
    x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,  // 9
// };
```

lib/graphics.c

```
// #pragma once

#if defined(__AVR__)
        #include <avr/pgmspace.h>
        #define fontdatatype const uint8_t
#elif defined(__PIC32MX__)
        #define PROGMEM
        #define fontdatatype const unsigned char
#elif defined(__arm__)
        #define PROGMEM
        #define fontdatatype const unsigned char
#endif

// [0]: width
// [1]: height
// [2]: amount of color data indexes...
// [3]... : color data
unsigned short arrow_head_y[28] PROGMEM ={
0x0005, 0x0005, 0x0019,
0x0000, 0x0000, 0x8410, 0x0000, 0x0000,
0x0000, 0x0000, 0xFFFF, 0x0000, 0x0000,
0x0000, 0xC618, 0xFFFF, 0xC618, 0x0000,
0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000,
0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0xC618
};

unsigned short arrow_head_x[28] PROGMEM ={
0x0005, 0x0005, 0x0019,
0xC618, 0x0000, 0x0000, 0x0000, 0x0000,
0xFFFF, 0xFFFF, 0xC618, 0x0000, 0x0000,
0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x8410,
0xFFFF, 0xFFFF, 0xC618, 0x0000, 0x0000,
0xC618, 0x0000, 0x0000, 0x0000, 0x0000
};

unsigned short arrow_head_45[39] PROGMEM ={
0x0006, 0x0006, 0x0024,
0x0000, 0x0000, 0x0000, 0x0000, 0x8410, 0x8410,
0x0000, 0x8410, 0xC618, 0xFFFF, 0xFFFF, 0x8410,
0x0000, 0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000,
0x0000, 0x8410, 0xFFFF, 0xFFFF, 0xC618, 0x0000,
0x0000, 0xFFFF, 0x8410, 0xC618, 0x8410, 0x0000,
0xFFFF, 0x8410, 0x0000, 0x0000, 0x0000, 0x0000
};

// unsigned short file[259] PROGMEM={
// 0x0010, 0x0010, 0x0100,
// 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0
    x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0010 (16)
// 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x8410, 0xFFFF, 0xFFFF, 0
    xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0020 (32)
// 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x8410, 0x8410, 0xFFFF, 0xFFFF, 0
    xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0030 (48)
// 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x8410, 0x8410, 0x8410, 0xFFFF, 0xFFFF, 0
    xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0040 (64)
```

```
// 0x0000, 0x0000, 0x0000, 0x0000, 0x8410, 0x8410, 0x8410, 0x8410, 0xC618, 0xC618, 0xC618, 0xC618, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0050 (80)
// 0x0000, 0x0000, 0x0000, 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0060 (96)
// 0x0000, 0x0000, 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0xFFFF, 0xFFFF, 0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0070 (112)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0080 (128)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x0090 (144)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x00A0 (160)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x00B0 (176)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x00C0 (192)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xC618, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x00D0 (208)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x00E0 (224)
// 0x0000, 0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000, 0x0000,   // 0x00F0 (240)
// 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0100 (256)
// };

// unsigned short folder[259] PROGMEM={
// 0x0010, 0x0010, 0x0100,
// 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0010 (16)
// 0x0000, 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0020 (32)
// 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000,   // 0x0030 (48)
// 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0x8410, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0040 (64)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0050 (80)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0060 (96)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0070 (112)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0080 (128)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x0090 (144)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00A0 (160)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00B0 (176)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00C0 (192)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00D0 (208)
// 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,   // 0x00E0 (224)
// 0x0000, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0
```

```
    xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF, 0x0000,   // 0x00F0 (240)
// 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0
    x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // 0x0100 (256)
// };
```

lib/initialization.h

```
#ifndef INITIALIZATION_H
#define INITIALIZATION_H

void init_io_config();

void init_external_interrupt();
void init_spurious_interrupt();

void IRQ_Button_Routine (void)      __attribute__ ((interrupt("IRQ")));
void IRQ_Spurious_Routine (void)    __attribute__ ((interrupt("IRQ")));

#endif //INITIALIZATION_H
```

lib/initialization.c

```c
#include "initialization.h"
#include "system.h"

void init_io_config()
{
    #ifdef REAL_SYSTEM
        SCS = 1; // Enable FAST GPIO
        #endif

    FIODIR  = 0b11111111111111111011110111011101;
    FIOSET  = 0xFFFFFFFF;

    PINSEL0 = 0x20051505;
    PINSEL1 = 0x00000000;

    EXTMODE     = (1<<1);
    EXTPOLAR    = (0<<1);
    EXTINT      = (1<<1);

    FIOCLR = (1<<7);
}

void init_external_interrupt()
{
    VICIntEnable |= (1<<15);
    VICVectAddr0=(unsigned int)IRQ_Button_Routine;
    VICVectCntl0=(1<<5)|15;
}

void init_spurious_interrupt()
{
    VICDefVectAddr=(unsigned int)IRQ_Spurious_Routine;
    VICVectAddr=-1;
}
```

lib/integer.h

```
/*
 *  File: integer.h
 *  Author: Nelson Lombardo
 *  Year: 2015
 *  e–mail: nelson.lombardo@gmail.com
 *  License at the end of file.
 */

/*****************************************************************************/
/* Integer type definitions                                                  */
/*****************************************************************************/
#ifndef _INTEGER_H_
#define _INTEGER_H_

#include <stdint.h>

/* 16–bit, 32–bit or larger integer */
typedef int16_t         INT;
typedef uint16_t        UINT;

/* 8–bit integer */
typedef int8_t          CHAR;
typedef uint8_t         UCHAR;
typedef uint8_t         BYTE;
typedef uint8_t         BOOL;

/* 16–bit integer */
typedef int16_t         SHORT;
typedef uint16_t        USHORT;
typedef uint16_t        WORD;
typedef uint16_t        WCHAR;

/* 32–bit integer */
typedef int32_t         LONG;
typedef uint32_t        ULONG;
typedef uint32_t        DWORD;

/* Boolean type */
typedef enum { FALSE = 0, TRUE } BOOLEAN;
typedef enum { LOW = 0, HIGH } THROTTLE;

#endif

// Âńinteger.hÂż is part of:
/*----------------------------------------------------------------------------/
/  ulibSD – Library for SD cards semantics            (C)Nelson Lombardo, 2015
/-----------------------------------------------------------------------------/
/ ulibSD library is a free software that opened under license policy of
/ following conditions.
/
/ Copyright (C) 2015, ChaN, all right reserved.
/
/ 1. Redistributions of source code must retain the above copyright notice,
/    this condition and the following disclaimer.
/
```

```
/ This software is provided by the copyright holder and contributors "AS IS"
/ and any warranties related to this software are DISCLAIMED.
/ The copyright owner or contributors be NOT LIABLE for any damages caused
/ by use of this software.
/----------------------------------------------------------------------------*/

// Derived from Mister Chan works on FatFs code (http://elm–chan.org/fsw/ff/00
   index_e.html):
/*----------------------------------------------------------------------------/
/  FatFs – FAT file system module  R0.11                 (C)ChaN, 2015
/-----------------------------------------------------------------------------/
/ FatFs module is a free software that opened under license policy of
/ following conditions.
/
/ Copyright (C) 2015, ChaN, all right reserved.
/
/ 1. Redistributions of source code must retain the above copyright notice,
/    this condition and the following disclaimer.
/
/ This software is provided by the copyright holder and contributors "AS IS"
/ and any warranties related to this software are DISCLAIMED.
/ The copyright owner or contributors be NOT LIABLE for any damages caused
/ by use of this software.
/----------------------------------------------------------------------------*/
```

lib/lpc2106.h

```
/***********************************************************************/
/*                                                                     */
/*  LPC2103.H:  Header file for Philips LPC2101 / LPC2102 / LPC2103    */
/*                                                                     */
/***********************************************************************/

#ifndef __LPC2103_H
#define __LPC2103_H

/* Vectored Interrupt Controller (VIC) */
#define VICIRQStatus   (*((volatile unsigned long *) 0xFFFFF000))
#define VICFIQStatus   (*((volatile unsigned long *) 0xFFFFF004))
#define VICRawIntr     (*((volatile unsigned long *) 0xFFFFF008))
#define VICIntSelect   (*((volatile unsigned long *) 0xFFFFF00C))
#define VICIntEnable   (*((volatile unsigned long *) 0xFFFFF010))
#define VICIntEnClr    (*((volatile unsigned long *) 0xFFFFF014))
#define VICSoftInt     (*((volatile unsigned long *) 0xFFFFF018))
#define VICSoftIntClr  (*((volatile unsigned long *) 0xFFFFF01C))
#define VICProtection  (*((volatile unsigned long *) 0xFFFFF020))
#define VICVectAddr    (*((volatile unsigned long *) 0xFFFFF030))
#define VICDefVectAddr (*((volatile unsigned long *) 0xFFFFF034))
#define VICVectAddr0   (*((volatile unsigned long *) 0xFFFFF100))
#define VICVectAddr1   (*((volatile unsigned long *) 0xFFFFF104))
#define VICVectAddr2   (*((volatile unsigned long *) 0xFFFFF108))
#define VICVectAddr3   (*((volatile unsigned long *) 0xFFFFF10C))
#define VICVectAddr4   (*((volatile unsigned long *) 0xFFFFF110))
#define VICVectAddr5   (*((volatile unsigned long *) 0xFFFFF114))
#define VICVectAddr6   (*((volatile unsigned long *) 0xFFFFF118))
#define VICVectAddr7   (*((volatile unsigned long *) 0xFFFFF11C))
#define VICVectAddr8   (*((volatile unsigned long *) 0xFFFFF120))
#define VICVectAddr9   (*((volatile unsigned long *) 0xFFFFF124))
#define VICVectAddr10  (*((volatile unsigned long *) 0xFFFFF128))
#define VICVectAddr11  (*((volatile unsigned long *) 0xFFFFF12C))
#define VICVectAddr12  (*((volatile unsigned long *) 0xFFFFF130))
#define VICVectAddr13  (*((volatile unsigned long *) 0xFFFFF134))
#define VICVectAddr14  (*((volatile unsigned long *) 0xFFFFF138))
#define VICVectAddr15  (*((volatile unsigned long *) 0xFFFFF13C))
#define VICVectCntl0   (*((volatile unsigned long *) 0xFFFFF200))
#define VICVectCntl1   (*((volatile unsigned long *) 0xFFFFF204))
#define VICVectCntl2   (*((volatile unsigned long *) 0xFFFFF208))
#define VICVectCntl3   (*((volatile unsigned long *) 0xFFFFF20C))
#define VICVectCntl4   (*((volatile unsigned long *) 0xFFFFF210))
#define VICVectCntl5   (*((volatile unsigned long *) 0xFFFFF214))
#define VICVectCntl6   (*((volatile unsigned long *) 0xFFFFF218))
#define VICVectCntl7   (*((volatile unsigned long *) 0xFFFFF21C))
#define VICVectCntl8   (*((volatile unsigned long *) 0xFFFFF220))
#define VICVectCntl9   (*((volatile unsigned long *) 0xFFFFF224))
#define VICVectCntl10  (*((volatile unsigned long *) 0xFFFFF228))
#define VICVectCntl11  (*((volatile unsigned long *) 0xFFFFF22C))
#define VICVectCntl12  (*((volatile unsigned long *) 0xFFFFF230))
#define VICVectCntl13  (*((volatile unsigned long *) 0xFFFFF234))
#define VICVectCntl14  (*((volatile unsigned long *) 0xFFFFF238))
#define VICVectCntl15  (*((volatile unsigned long *) 0xFFFFF23C))

/* Pin Connect Block */
```

```
#define PINSEL0        (*((volatile unsigned long *) 0xE002C000))
#define PINSEL1        (*((volatile unsigned long *) 0xE002C004))

/* General Purpose Input/Output (GPIO) */
#define IOPIN          (*((volatile unsigned long *) 0xE0028000))
#define IOSET          (*((volatile unsigned long *) 0xE0028004))
#define IODIR          (*((volatile unsigned long *) 0xE0028008))
#define IOCLR          (*((volatile unsigned long *) 0xE002800C))

/* Fast General Purpose Input/Output (GPIO) */
#define FIODIR         (*((volatile unsigned long *) 0x3FFFC000))
#define FIOMASK        (*((volatile unsigned long *) 0x3FFFC010))
#define FIOPIN         (*((volatile unsigned long *) 0x3FFFC014))
#define FIOSET         (*((volatile unsigned long *) 0x3FFFC018))
#define FIOCLR         (*((volatile unsigned long *) 0x3FFFC01C))

/* Memory Accelerator Module (MAM) */
#define MAMCR          (*((volatile unsigned char *) 0xE01FC000))
#define MAMTIM         (*((volatile unsigned char *) 0xE01FC004))
#define MEMMAP         (*((volatile unsigned char *) 0xE01FC040))

/* Phase Locked Loop (PLL) */
#define PLLCON         (*((volatile unsigned char *) 0xE01FC080))
#define PLLCFG         (*((volatile unsigned char *) 0xE01FC084))
#define PLLSTAT        (*((volatile unsigned long *) 0xE01FC088))
#define PLLFEED        (*((volatile unsigned char *) 0xE01FC08C))

/* APB Divider */
#define APBDIV         (*((volatile unsigned char *) 0xE01FC100))

/* Power Control */
#define PCON           (*((volatile unsigned char *) 0xE01FC0C0))
#define PCONP          (*((volatile unsigned long *) 0xE01FC0C4))

/* External Interrupts */
#define EXTINT         (*((volatile unsigned char *) 0xE01FC140))
#define EXTWAKE        (*((volatile unsigned char *) 0xE01FC144))
#define EXTMODE        (*((volatile unsigned char *) 0xE01FC148))
#define EXTPOLAR       (*((volatile unsigned char *) 0xE01FC14C))

/* Timer 0 */
#define T0IR           (*((volatile unsigned char *) 0xE0004000))
#define T0TCR          (*((volatile unsigned char *) 0xE0004004))
#define T0TC           (*((volatile unsigned long *) 0xE0004008))
#define T0PR           (*((volatile unsigned long *) 0xE000400C))
#define T0PC           (*((volatile unsigned long *) 0xE0004010))
#define T0MCR          (*((volatile unsigned long *) 0xE0004014))
#define T0MR0          (*((volatile unsigned long *) 0xE0004018))
#define T0MR1          (*((volatile unsigned long *) 0xE000401C))
#define T0MR2          (*((volatile unsigned long *) 0xE0004020))
#define T0MR3          (*((volatile unsigned long *) 0xE0004024))
#define T0CCR          (*((volatile unsigned long *) 0xE0004028))
#define T0CR0          (*((volatile unsigned long *) 0xE000402C))
#define T0CR1          (*((volatile unsigned long *) 0xE0004030))
#define T0CR2          (*((volatile unsigned long *) 0xE0004034))
#define T0CR3          (*((volatile unsigned long *) 0xE0004038))
#define T0EMR          (*((volatile unsigned long *) 0xE000403C))
```

```
#define T0CTCR         (*((volatile unsigned char *) 0xE0004070))
#define T0PWMCON       (*((volatile unsigned long *) 0xE0004074))

/* Timer 1 */
#define T1IR           (*((volatile unsigned char *) 0xE0008000))
#define T1TCR          (*((volatile unsigned char *) 0xE0008004))
#define T1TC           (*((volatile unsigned long *) 0xE0008008))
#define T1PR           (*((volatile unsigned long *) 0xE000800C))
#define T1PC           (*((volatile unsigned long *) 0xE0008010))
#define T1MCR          (*((volatile unsigned long *) 0xE0008014))
#define T1MR0          (*((volatile unsigned long *) 0xE0008018))
#define T1MR1          (*((volatile unsigned long *) 0xE000801C))
#define T1MR2          (*((volatile unsigned long *) 0xE0008020))
#define T1MR3          (*((volatile unsigned long *) 0xE0008024))
#define T1CCR          (*((volatile unsigned long *) 0xE0008028))
#define T1CR0          (*((volatile unsigned long *) 0xE000802C))
#define T1CR1          (*((volatile unsigned long *) 0xE0008030))
#define T1CR2          (*((volatile unsigned long *) 0xE0008034))
#define T1CR3          (*((volatile unsigned long *) 0xE0008038))
#define T1EMR          (*((volatile unsigned long *) 0xE000803C))
#define T1CTCR         (*((volatile unsigned char *) 0xE0008070))
#define T1PWMCON       (*((volatile unsigned long *) 0xE0008074))

/* Universal Asynchronous Receiver Transmitter 0 (UART0) */
#define U0RBR          (*((volatile unsigned char *) 0xE000C000))
#define U0THR          (*((volatile unsigned char *) 0xE000C000))
#define U0IER          (*((volatile unsigned long *) 0xE000C004))
#define U0IIR          (*((volatile unsigned long *) 0xE000C008))
#define U0FCR          (*((volatile unsigned char *) 0xE000C008))
#define U0LCR          (*((volatile unsigned char *) 0xE000C00C))

#define U0LSR          (*((volatile unsigned char *) 0xE000C014))
#define U0SCR          (*((volatile unsigned char *) 0xE000C01C))
#define U0DLL          (*((volatile unsigned char *) 0xE000C000))
#define U0DLM          (*((volatile unsigned char *) 0xE000C004))
#define U0ACR          (*((volatile unsigned long *) 0xE000C020))
#define U0FDR          (*((volatile unsigned long *) 0xE000C028))
#define U0TER          (*((volatile unsigned char *) 0xE000C030))

/* Universal Asynchronous Receiver Transmitter 1 (UART1) */
#define U1RBR          (*((volatile unsigned char *) 0xE0010000))
#define U1THR          (*((volatile unsigned char *) 0xE0010000))
#define U1IER          (*((volatile unsigned long *) 0xE0010004))
#define U1IIR          (*((volatile unsigned long *) 0xE0010008))
#define U1FCR          (*((volatile unsigned char *) 0xE0010008))
#define U1LCR          (*((volatile unsigned char *) 0xE001000C))
#define U1MCR          (*((volatile unsigned char *) 0xE0010010))
#define U1LSR          (*((volatile unsigned char *) 0xE0010014))
#define U1MSR          (*((volatile unsigned char *) 0xE0010018))
#define U1SCR          (*((volatile unsigned char *) 0xE001001C))
#define U1DLL          (*((volatile unsigned char *) 0xE0010000))
#define U1DLM          (*((volatile unsigned char *) 0xE0010004))
#define U1ACR          (*((volatile unsigned long *) 0xE0010020))
#define U1FDR          (*((volatile unsigned long *) 0xE0010028))
#define U1TER          (*((volatile unsigned char *) 0xE0010030))

/* Inter-Integrated Circuit interface 0 (I2C0) */
```

```
#define I2C0CONSET     (*((volatile unsigned char *) 0xE001C000))
#define I2C0STAT       (*((volatile unsigned char *) 0xE001C004))
#define I2C0DAT        (*((volatile unsigned char *) 0xE001C008))
#define I2C0ADR        (*((volatile unsigned char *) 0xE001C00C))
#define I2C0SCLH       (*((volatile unsigned long *) 0xE001C010))
#define I2C0SCLL       (*((volatile unsigned long *) 0xE001C014))
#define I2C0CONCLR     (*((volatile unsigned char *) 0xE001C018))

/* Serial Peripheral Interface 0 (SPI0) */
#define S0SPCR         (*((volatile unsigned long *) 0xE0020000))
#define S0SPSR         (*((volatile unsigned char *) 0xE0020004))
#define S0SPDR         (*((volatile unsigned long *) 0xE0020008))
#define S0SPCCR        (*((volatile unsigned char *) 0xE002000C))
#define S0SPINT        (*((volatile unsigned char *) 0xE002001C))

/* Real Time Clock (RTC) */
#define ILR            (*((volatile unsigned char *) 0xE0024000))
#define CTC            (*((volatile unsigned long *) 0xE0024004))
#define CCR            (*((volatile unsigned char *) 0xE0024008))
#define CIIR           (*((volatile unsigned char *) 0xE002400C))
#define AMR            (*((volatile unsigned char *) 0xE0024010))
#define CTIME0         (*((volatile unsigned long *) 0xE0024014))
#define CTIME1         (*((volatile unsigned long *) 0xE0024018))
#define CTIME2         (*((volatile unsigned long *) 0xE002401C))
#define SEC            (*((volatile unsigned char *) 0xE0024020))
#define MIN            (*((volatile unsigned char *) 0xE0024024))
#define HOUR           (*((volatile unsigned char *) 0xE0024028))
#define DOM            (*((volatile unsigned char *) 0xE002402C))
#define DOW            (*((volatile unsigned char *) 0xE0024030))
#define DOY            (*((volatile unsigned long *) 0xE0024034))
#define MONTH          (*((volatile unsigned char *) 0xE0024038))
#define YEAR           (*((volatile unsigned long *) 0xE002403C))
#define ALSEC          (*((volatile unsigned char *) 0xE0024060))
#define ALMIN          (*((volatile unsigned char *) 0xE0024064))
#define ALHOUR         (*((volatile unsigned char *) 0xE0024068))
#define ALDOM          (*((volatile unsigned char *) 0xE002406C))
#define ALDOW          (*((volatile unsigned char *) 0xE0024070))
#define ALDOY          (*((volatile unsigned long *) 0xE0024074))
#define ALMON          (*((volatile unsigned char *) 0xE0024078))
#define ALYEAR         (*((volatile unsigned long *) 0xE002407C))
#define PREINT         (*((volatile unsigned long *) 0xE0024080))
#define PREFRAC        (*((volatile unsigned long *) 0xE0024084))

/* Analog/Digital Converter (ADC) */
#define ADCR           (*((volatile unsigned long *) 0xE0034000))
#define ADGDR          (*((volatile unsigned long *) 0xE0034004))
#define ADINTEN        (*((volatile unsigned long *) 0xE003400C))
#define ADDR0          (*((volatile unsigned long *) 0xE0034010))
#define ADDR1          (*((volatile unsigned long *) 0xE0034014))
#define ADDR2          (*((volatile unsigned long *) 0xE0034018))
#define ADDR3          (*((volatile unsigned long *) 0xE003401C))
#define ADDR4          (*((volatile unsigned long *) 0xE0034020))
#define ADDR5          (*((volatile unsigned long *) 0xE0034024))
#define ADDR6          (*((volatile unsigned long *) 0xE0034028))
#define ADDR7          (*((volatile unsigned long *) 0xE003402C))
#define ADSTAT         (*((volatile unsigned long *) 0xE0034030))
```

```
/* Inter-Integrated Circuit interface 1 (I2C1) */
#define I2C1CONSET      (*((volatile unsigned char *) 0xE005C000))
#define I2C1STAT        (*((volatile unsigned char *) 0xE005C004))
#define I2C1DAT         (*((volatile unsigned char *) 0xE005C008))
#define I2C1ADR         (*((volatile unsigned char *) 0xE005C00C))
#define I2C1SCLH        (*((volatile unsigned long*) 0xE005C010))
#define I2C1SCLL        (*((volatile unsigned long*) 0xE005C014))
#define I2C1CONCLR      (*((volatile unsigned char *) 0xE005C018))

/* Synchronous Serial Port interface (SSP) (Direcciones MAL)
#define SSPCR0          (*((volatile unsigned long *) 0xE0068000))
#define SSPCR1          (*((volatile unsigned char *) 0xE0068004))
#define SSPDR           (*((volatile unsigned long *) 0xE0068008))
#define SSPSR           (*((volatile unsigned char *) 0xE006800C))
#define SSPCPSR         (*((volatile unsigned char *) 0xE0068010))
#define SSPIMSC         (*((volatile unsigned char *) 0xE0068014))
#define SSPRIS          (*((volatile unsigned char *) 0xE0068018))
#define SSPMIS          (*((volatile unsigned char *) 0xE006801C))
#define SSPICR          (*((volatile unsigned char *) 0xE0068020))

*//* Synchronous Serial Port interface (SSP) */ //0xE005C000
#define SSPCR0          (*((volatile unsigned long *) 0xE005C000))
#define SSPCR1          (*((volatile unsigned char *) 0xE005C004))
#define SSPDR           (*((volatile unsigned long *) 0xE005C008))
#define SSPSR           (*((volatile unsigned char *) 0xE005C00C))
#define SSPCPSR         (*((volatile unsigned char *) 0xE005C010))
#define SSPIMSC         (*((volatile unsigned char *) 0xE005C014))
#define SSPRIS          (*((volatile unsigned char *) 0xE005C018))
#define SSPMIS          (*((volatile unsigned char *) 0xE005C01C))
#define SSPICR          (*((volatile unsigned char *) 0xE005C020))

/* Timer 2 */
#define T2IR            (*((volatile unsigned char *) 0xE0070000))
#define T2TCR           (*((volatile unsigned char *) 0xE0070004))
#define T2TC            (*((volatile unsigned long *) 0xE0070008))
#define T2PR            (*((volatile unsigned long *) 0xE007000C))
#define T2PC            (*((volatile unsigned long *) 0xE0070010))
#define T2MCR           (*((volatile unsigned long *) 0xE0070014))
#define T2MR0           (*((volatile unsigned long *) 0xE0070018))
#define T2MR1           (*((volatile unsigned long *) 0xE007001C))
#define T2MR2           (*((volatile unsigned long *) 0xE0070020))
#define T2MR3           (*((volatile unsigned long *) 0xE0070024))
#define T2CCR           (*((volatile unsigned long *) 0xE0070028))
#define T2CR0           (*((volatile unsigned long *) 0xE007002C))
#define T2CR1           (*((volatile unsigned long *) 0xE0070030))
#define T2CR2           (*((volatile unsigned long *) 0xE0070034))
#define T2EMR           (*((volatile unsigned long *) 0xE007003C))
#define T2CTCR          (*((volatile unsigned char *) 0xE0070070))
#define T2PWMCON        (*((volatile unsigned long *) 0xE0070074))

/* Timer 3 */
#define T3IR            (*((volatile unsigned char *) 0xE0074000))
#define T3TCR           (*((volatile unsigned char *) 0xE0074004))
#define T3TC            (*((volatile unsigned long *) 0xE0074008))
#define T3PR            (*((volatile unsigned long *) 0xE007400C))
#define T3PC            (*((volatile unsigned long *) 0xE0074010))
#define T3MCR           (*((volatile unsigned long *) 0xE0074014))
```

```c
#define T3MR0          (*((volatile unsigned long *) 0xE0074018))
#define T3MR4          (*((volatile unsigned long *) 0xE007401C))
#define T3MR2          (*((volatile unsigned long *) 0xE0074020))
#define T3MR3          (*((volatile unsigned long *) 0xE0074024))
#define T3CCR          (*((volatile unsigned long *) 0xE0074028))
#define T3CR0          (*((volatile unsigned long *) 0xE007402C))
#define T3CR1          (*((volatile unsigned long *) 0xE0074030))
#define T3CR2          (*((volatile unsigned long *) 0xE0074034))
#define T3EMR          (*((volatile unsigned long *) 0xE007403C))
#define T3CTCR         (*((volatile unsigned char *) 0xE0074070))
#define T3PWMCON       (*((volatile unsigned long *) 0xE0074074))

/* Reset Source Identification */
#define RSIR           (*((volatile unsigned char *) 0xE01FC180))

/* Code Security Protection */
#define CPSR           (*((volatile unsigned long *) 0xE01FC184))

/* Syscon Miscellaneous */
#define SCS            (*((volatile unsigned long *) 0xE01FC1A0))

/* Watchdog timer */
#define WDMOD          (*((volatile unsigned char *) 0xE0000000))
#define WDTC           (*((volatile unsigned long *) 0xE0000004))
#define WDFEED         (*((volatile unsigned char *) 0xE0000008))
#define WDTV           (*((volatile unsigned long *) 0xE000000C))

#endif  // __LPC2103_H
```

```
#pragma once

#include "system.h"

#define MCP_CS (1<<7)

#define IODIR_PORTA    0x00
#define IODIR_PORTB    0x4F


typedef enum {
        DATA_OUT_0,
        DATA_OUT_1,
        DATA_OUT_2,
        DATA_OUT_3,
        DATA_OUT_4,
        DATA_OUT_5,
        DATA_OUT_6,
        DATA_OUT_7,
    SEL,
    PE,
    BUSY,
    ACK,
    SELIN,
    INIT,
    ERROR,
    AUTOF
} MCPPIN;

typedef enum {
    PORTA,
    PORTB
} MCPPORT;


int MCP23S17Read(int address);
void MCP23S17Write(int address, int data);
void MCPsetIODirCNC();

void MCPSetPort(MCPPORT port, int data);
int MCPReadPort(MCPPORT port);


#define MCP_CS_L() {FIOSET=MCP_CS;}
#define MCP_CS_H() {FIOCLR=MCP_CS;}

/* ---------------------------------
               REGISTERS
--------------------------------- */
#define IODIRA          (0x00)
#define IODIRB          (0x01)
#define IPOLA           (0x02)
#define IPOLB           (0x03)
#define GPINTENA        (0x04)
#define GPINTENB        (0x05)
```

```
#define DEFVALA         (0x06)
#define DEFVALB         (0x07)
#define INTCONA         (0x08)
#define INTCONB         (0x09)
#define IOCONA          (0x0A)
#define IOCONB          (0x0B)
#define GPPUA           (0x0C)
#define GPPUB           (0x0D)
#define INTFA           (0x0E)
#define INTFB           (0x0F)
#define INTCAPA         (0x10)
#define INTCAPB         (0x11)
#define GPIOA           (0x12)
#define GPIOB           (0x13)
#define OLATA           (0x14)
#define OLATB           (0x15)
```

lib/mcp23s17.c

```c
#include "mcp23s17.h"
// #define MCP_DEBUG

/**
 * @brief Read function for the I/O expander
 *
 * @param address is the address of a register in the I/O expander
 * @return int returns data that is at the provided address
 */
int MCP23S17Read(int address)
{
        int data;
        MCP_CS_L();                             // lower CS
        SPI_byte(0x41);                 // transmit read opcode
        SPI_byte(address);      // send MSByte address first
        data=SPI_byte(0);
        MCP_CS_H();                             // set CS high
        return data;
}

/**
 * @brief Writes data to a register in the I/O expander
 *
 * @param address is the address of a register in the I/O expander
 * @param data is the data you want to write to the provided register address
 */
void MCP23S17Write(int address, int data)
{
        MCP_CS_L();                             // lower CS
        SPI_byte(0x40);                 // write command
        SPI_byte(address);      // address
        data=SPI_byte(data);    // data byte(s)
        MCP_CS_H();                             // set CS high
}

/**
 * @brief Sets the correct IO directiosn for port A and B for the CNC controller application
 *
 */
void MCPsetIODirCNC()
{
        // Set GPIO A as output pins, these are all 'data out'-pins
        MCP23S17Write(IODIRA, IODIR_PORTA);
        // Set GPIO B IO direction, these contain in- and outputs
        MCP23S17Write(IODIRB, IODIR_PORTB);
}

/**
 * @brief Writes data to a particular port
 *
 * @param port can either be 'PORTA' or 'PORTB', is the port you want to write data to
 * @param data is the data you want to write to the provived port
 */
```

```
void MCPSetPort(MCPPORT port, int data)
{
        switch (port)
        {
                case PORTA:
                        MCP23S17Write(OLATA, data);
                        #ifdef MCP_DEBUG
                        _printf("MCP␣-␣I␣have␣just␣written␣0x%x␣to␣OLATA\r\n");
                        #endif
                        break;

                case PORTB:
                        MCP23S17Write(OLATB, data);
                        #ifdef MCP_DEBUG
                        _printf("MCP␣-␣I␣have␣just␣written␣0x%x␣to␣OLATB\r\n");
                        #endif
                        break;
        }
}

/**
 * @brief Reads state of a certain port
 *
 * @param port can either be 'PORTA' or 'PORTB', is the port you want to read
 * @return int is the state of the pins of that port
 */
int MCPReadPort(MCPPORT port)
{
        #ifdef MCP_DEBUG
        _printf("MCP␣-␣I'm␣about␣to␣read␣one␣of␣the␣port\r\n");
        #endif

        switch (port)
        {
                case PORTA:             return MCP23S17Read(GPIOA);
                case PORTB:             return MCP23S17Read(GPIOB);
        }
}
```

lib/pff.h

```c
/*---------------------------------------------------------------------------/
/  Petit FatFs – FAT file system module include file  R0.03   (C)ChaN, 2014
/----------------------------------------------------------------------------/
/ Petit FatFs module is an open source software to implement FAT file system to
/ small embedded systems. This is a free software and is opened for education,
/ research and commercial developments under license policy of following trems.
/
/  Copyright (C) 2014, ChaN, all right reserved.
/
/ * The Petit FatFs module is a free software and there is NO WARRANTY.
/ * No restriction on use. You can use, modify and redistribute it for
/   personal, non–profit or commercial use UNDER YOUR RESPONSIBILITY.
/ * Redistributions of source code must retain the above copyright notice.
/
/----------------------------------------------------------------------------*/
#pragma once

#include "system.h"

#ifndef _PFATFS
#define _PFATFS 4004    /* Revision ID */

#ifdef __cplusplus
extern "C" {
#endif

#include "integer.h"
/*---------------------------------------------------------------------------/
/  Petit FatFs – Configuration file  R0.03 (C)ChaN, 2014
/---------------------------------------------------------------------------*/

#ifndef _PFFCONF
#define _PFFCONF 4004   /* Revision ID */

/*---------------------------------------------------------------------------/
/ Function Configurations
/---------------------------------------------------------------------------*/

#define _USE_READ       1       /* Enable pf_read() function */
#define _USE_DIR        1       /* Enable pf_opendir() and pf_readdir() function */
#define _USE_LSEEK      0       /* Enable pf_lseek() function */
#define _USE_WRITE      0       /* Enable pf_write() function */

#define _FS_FAT12       0       /* Enable FAT12 */
#define _FS_FAT16       1       /* Enable FAT16 */
#define _FS_FAT32       1       /* Enable FAT32 */

/*---------------------------------------------------------------------------/
/ Locale and Namespace Configurations
/---------------------------------------------------------------------------*/

#define _USE_LCC        1       /* Allow lower case characters for path name */

#define _CODE_PAGE      437
/* The _CODE_PAGE specifies the code page to be used on the target system.
```

```
/  SBCS code pages with _USE_LCC == 1 requiers a 128 byte of case conversion
/  table. This might occupy RAM on some platforms, e.g. avr-gcc.
/  When _USE_LCC == 0, _CODE_PAGE has no effect.
/
/   932  - Japanese Shift_JIS (DBCS, OEM, Windows)
/   936  - Simplified Chinese GBK (DBCS, OEM, Windows)
/   949  - Korean (DBCS, OEM, Windows)
/   950  - Traditional Chinese Big5 (DBCS, OEM, Windows)
/   1250 - Central Europe (Windows)
/   1251 - Cyrillic (Windows)
/   1252 - Latin 1 (Windows)
/   1253 - Greek (Windows)
/   1254 - Turkish (Windows)
/   1255 - Hebrew (Windows)
/   1256 - Arabic (Windows)
/   1257 - Baltic (Windows)
/   1258 - Vietnam (OEM, Windows)
/   437  - U.S. (OEM)
/   720  - Arabic (OEM)
/   737  - Greek (OEM)
/   775  - Baltic (OEM)
/   850  - Multilingual Latin 1 (OEM)
/   858  - Multilingual Latin 1 + Euro (OEM)
/   852  - Latin 2 (OEM)
/   855  - Cyrillic (OEM)
/   866  - Russian (OEM)
/   857  - Turkish (OEM)
/   862  - Hebrew (OEM)
/   874  - Thai (OEM, Windows)
*/

/*---------------------------------------------------------------------------/
/ System Configurations
/---------------------------------------------------------------------------*/

#define _WORD_ACCESS    0
/* The _WORD_ACCESS option is an only platform dependent option. It defines
/  which access method is used to the word data on the FAT volume.
/
/   0: Byte-by-byte access. Always compatible with all platforms.
/   1: Word access. Do not choose this unless under both the following conditions.
/
/  * Address misaligned memory access is always allowed for ALL instructions.
/  * Byte order on the memory is little-endian.
/
/  If it is the case, _WORD_ACCESS can also be set to 1 to improve performance and
/  reduce code size. Following table shows an example of some processor types.
/
/   ARM7TDMI    0           ColdFire    0           V850E       0
/   Cortex-M3   0           Z80         0/1         V850ES      0/1
/   Cortex-M0   0           RX600(LE)   0/1         TLCS-870    0/1
/   AVR         0/1         RX600(BE)   0           TLCS-900    0/1
/   AVR32       0           RL78        0           R32C        0
/   PIC18       0/1         SH-2        0           M16C        0/1
/   PIC24       0           H8S         0           MSP430      0
/   PIC32       0           H8/300H     0           x86         0/1
*/
```

```
#endif /* _PFFCONF */

#if _PFATFS != _PFFCONF
#error Wrong configuration file (pffconf.h).
#endif

#if _FS_FAT32
#define CLUST   DWORD
#else
#define CLUST   WORD
#endif

/* File system object structure */

typedef struct {
        BYTE    fs_type;        /* FAT sub type */
        BYTE    flag;           /* File status flags */
        BYTE    csize;          /* Number of sectors per cluster */
        BYTE    pad1;
        WORD    n_rootdir;      /* Number of root directory entries (0 on FAT32) */
        CLUST   n_fatent;       /* Number of FAT entries (= number of clusters + 2)
    */
        DWORD   fatbase;        /* FAT start sector */
        DWORD   dirbase;        /* Root directory start sector (Cluster# on FAT32)
    */
        DWORD   database;       /* Data start sector */
        DWORD   fptr;           /* File R/W pointer */
        DWORD   fsize;          /* File size */
        CLUST   org_clust;      /* File start cluster */
        CLUST   curr_clust;     /* File current cluster */
        DWORD   dsect;          /* File current data sector */
} FATFS;

/* Directory object structure */

typedef struct {
        WORD    index;          /* Current read/write index number */
        BYTE*   fn;                     /* Pointer to the SFN (in/out) {file[8],ext
    [3],status[1]} */
        CLUST   sclust;         /* Table start cluster (0:Static table) */
        CLUST   clust;          /* Current cluster */
        DWORD   sect;           /* Current sector */
} DIR;

/* File status structure */

typedef struct {
        DWORD   fsize;          /* File size */
        WORD    fdate;          /* Last modified date */
        WORD    ftime;          /* Last modified time */
        BYTE    fattrib;        /* Attribute */
        char    fname[13];      /* File name */
} FILINFO;

/* File function return code (FRESULT) */

```

```
typedef enum {
        FR_OK = 0,                      /* 0 – The function succeeded */
        FR_DISK_ERR,            /* 1 – An error occured in the disk read function */
        FR_NOT_READY,           /* 2 – The storage device could not be initialized
    due to a hard error or no medium */
        FR_NO_FILE,                     /* 3 */
        FR_NOT_OPENED,          /* 4 */
        FR_NOT_ENABLED,         /* 5 */
        FR_NO_FILESYSTEM        /* 6 – There is no valid FAT partition on the drive
    */
} FRESULT;

/*--------------------------------------------------------------*/
/* Petit FatFs module application interface                     */

FRESULT pf_mount (FATFS* fs);
            /* Mount/Unmount a logical drive */
FRESULT pf_open (const char* path);
            /* Open a file */
FRESULT pf_read (void* buff, UINT btr, UINT* br);                       /* Read data
     from the open file */
FRESULT pf_write (const void* buff, UINT btw, UINT* bw);        /* Write data to the
     open file */
FRESULT pf_lseek (DWORD ofs);
            /* Move file pointer of the open file */
FRESULT pf_opendir (DIR* dj, const char* path);                         /* Open a
    directory */
FRESULT pf_readdir (DIR* dj, FILINFO* fno);                                     /*
    Read a directory item from the open directory */
FRESULT scan_files (char* path);

/*--------------------------------------------------------------*/
/* Flags and offset address                                     */

/* File status flag (FATFS.flag) */

#define FA_OPENED       0x01
#define FA_WPRT         0x02
#define FA__WIP         0x40

/* FAT sub type (FATFS.fs_type) */

#define FS_FAT12        1
#define FS_FAT16        2
#define FS_FAT32        3

/* File attribute bits for directory entry */

#define AM_RDO  0x01    /* Read only */
#define AM_HID  0x02    /* Hidden */
#define AM_SYS  0x04    /* System */
#define AM_VOL  0x08    /* Volume label */
#define AM_LFN  0x0F    /* LFN entry */
#define AM_DIR  0x10    /* Directory */
#define AM_ARC  0x20    /* Archive */
#define AM_MASK 0x3F    /* Mask of defined bits */
```

```
/*---------------------------------*/
/* Multi-byte word access macros  */

#if _WORD_ACCESS == 1   /* Enable word access to the FAT structure */
#define LD_WORD(ptr)            (WORD)(*(WORD*)(BYTE*)(ptr))
#define LD_DWORD(ptr)           (DWORD)(*(DWORD*)(BYTE*)(ptr))
#define ST_WORD(ptr,val)        *(WORD*)(BYTE*)(ptr)=(WORD)(val)
#define ST_DWORD(ptr,val)       *(DWORD*)(BYTE*)(ptr)=(DWORD)(val)
#else                                   /* Use byte-by-byte access to the FAT structure */
#define LD_WORD(ptr)            (WORD)(((WORD)*((BYTE*)(ptr)+1)<<8)|(WORD)*(BYTE*)(ptr))
#define LD_DWORD(ptr)           (DWORD)(((DWORD)*((BYTE*)(ptr)+3)<<24)|((DWORD)*((BYTE*)(ptr)+2)<<16)|((WORD)*((BYTE*)(ptr)+1)<<8)|*(BYTE*)(ptr))
#define ST_WORD(ptr,val)        *(BYTE*)(ptr)=(BYTE)(val); *((BYTE*)(ptr)+1)=(BYTE)((WORD)(val)>>8)
#define ST_DWORD(ptr,val)       *(BYTE*)(ptr)=(BYTE)(val); *((BYTE*)(ptr)+1)=(BYTE)((WORD)(val)>>8); *((BYTE*)(ptr)+2)=(BYTE)((DWORD)(val)>>16); *((BYTE*)(ptr)+3)=(BYTE)((DWORD)(val)>>24)
#endif

#ifdef __cplusplus
}
#endif

#endif /* _PFATFS */
```

lib/pff.c

```
/*----------------------------------------------------------------------------/
/  Petit FatFs - FAT file system module  R0.03                (C)ChaN, 2014
/-----------------------------------------------------------------------------/
/ Petit FatFs module is a generic FAT file system module for small embedded
/ systems. This is a free software that opened for education, research and
/ commercial developments under license policy of following trems.
/
/  Copyright (C) 2014, ChaN, all right reserved.
/
/ * The Petit FatFs module is a free software and there is NO WARRANTY.
/ * No restriction on use. You can use, modify and redistribute it for
/   personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY.
/ * Redistributions of source code must retain the above copyright notice.
/
/-----------------------------------------------------------------------------/
/ Jun 15,'09  R0.01a  First release.
/
/ Dec 14,'09  R0.02   Added multiple code page support.
/                     Added write funciton.
/                     Changed stream read mode interface.
/ Dec 07,'10  R0.02a  Added some configuration options.
/                     Fixed fails to open objects with DBCS character.

/ Jun 10,'14  R0.03   Separated out configuration options to pffconf.h.
/                     Added _USE_LCC option.
/                     Added _FS_FAT16 option.
/----------------------------------------------------------------------------*/

#include "pff.h"                /* Petit FatFs configurations and declarations */

FATFS FatFS[1];

/* Declarations of low level disk I/O functions */
/* Status of Disk Functions */
typedef BYTE    DSTATUS;

/* Results of Disk Functions */
typedef enum {
        RES_OK = 0,             /* 0: Function succeeded */
        RES_ERROR,              /* 1: Disk error */
        RES_NOTRDY,             /* 2: Not ready */
        RES_PARERR              /* 3: Invalid parameter */
} DRESULT;

#define STA_NOINIT              0x01    /* Drive not initialized */
#define STA_NODISK              0x02    /* No medium in the drive */

/*-----------------------------------------------------------------------*/
/* Initialize Disk Drive                                                 */
/*-----------------------------------------------------------------------*/
DSTATUS disk_initialize (void)
{
 DSTATUS stat;
 stat = SD_init();
 return stat;
}
/*-----------------------------------------------------------------------*/
/* Read Partial Sector                                                   */
/*-----------------------------------------------------------------------*/
DRESULT disk_readp (
        BYTE* buff,             /* Pointer to the destination object */
        DWORD sector,   /* Sector number (LBA) */
        UINT offset,    /* Offset in the sector */
        UINT count              /* Byte count (bit15:destination) */
)
{
        DRESULT res;
    unsigned char B[512];
    int i;
        // Put your code here
    res = SD_read_Block( sector, B );
        for(i=offset;i<offset+count;i++) buff[i-offset] = B[i];
//     _printf("disk_readp (%d) sector %d, ofs = %d, cnt = %d\r\n",res,sector,offset,count);
        return res;
}
/*-----------------------------------------------------------------------*/
/* Write Partial Sector                                                  */
/*-----------------------------------------------------------------------*/
DRESULT disk_writep (
        BYTE* buff,             /* Pointer to the data to be written, NULL:Initiate/Finalize write operation */
        DWORD sc                /* Sector number (LBA) or Number of bytes to send */
)
{
        DRESULT res=0;
        if (!buff) {
                if (sc) {
                        // Initiate write process
                } else {
                    // Finalize write process
                }
        } else {
                // Send data to the disk
        }
        return res;
}
/*-----------------------------------------------------------------------

   Module Private Definitions

-----------------------------------------------------------------------*/
#if _PFATFS != 4004     /* Revision ID */
#error Wrong include file (pff.h).
#endif

#if _FS_FAT32
#if !_FS_FAT16 && !_FS_FAT12
#define _FS_32ONLY 1
#else
#define _FS_32ONLY 0
#endif
```

```
#else
#if !_FS_FAT16 && !_FS_FAT12
#error Wrong _FS_FATxx setting.
#endif
#define _FS_32ONLY 0
#endif

#define ABORT(err)      {fs->flag = 0; return err;}

/*--------------------------------------------------------*/
/* DBCS code ranges and SBCS extend char conversion table */
/*--------------------------------------------------------*/

#if _CODE_PAGE == 932   /* Japanese Shift-JIS */
#define _DF1S   0x81    /* DBC 1st byte range 1 start */
#define _DF1E   0x9F    /* DBC 1st byte range 1 end */
#define _DF2S   0xE0    /* DBC 1st byte range 2 start */
#define _DF2E   0xFC    /* DBC 1st byte range 2 end */
#define _DS1S   0x40    /* DBC 2nd byte range 1 start */
#define _DS1E   0x7E    /* DBC 2nd byte range 1 end */
#define _DS2S   0x80    /* DBC 2nd byte range 2 start */
#define _DS2E   0xFC    /* DBC 2nd byte range 2 end */

#elif _CODE_PAGE == 936 /* Simplified Chinese GBK */
#define _DF1S   0x81
#define _DF1E   0xFE
#define _DS1S   0x40
#define _DS1E   0x7E
#define _DS2S   0x80
#define _DS2E   0xFE

#elif _CODE_PAGE == 949 /* Korean */
#define _DF1S   0x81
#define _DF1E   0xFE
#define _DS1S   0x41
#define _DS1E   0x5A
#define _DS2S   0x61
#define _DS2E   0x7A
#define _DS3S   0x81
#define _DS3E   0xFE

#elif _CODE_PAGE == 950 /* Traditional Chinese Big5 */
#define _DF1S   0x81
#define _DF1E   0xFE
#define _DS1S   0x40
#define _DS1E   0x7E
#define _DS2S   0xA1
#define _DS2E   0xFE

#elif _CODE_PAGE == 437 /* U.S. (OEM) */
#define _EXCVT {0x80,0x9A,0x90,0x41,0x8E,0x41,0x8F,0x80,0x45,0x45,0x45,0x49,0x49,0
    x49,0x8E,0x8F,0x90,0x92,0x92,0x4F,0x99,0x4F,0x55,0x55,0x59,0x99,0x9A,0x9B,0x9C,0
    x9D,0x9E,0x9F, \
                                0x41,0x49,0x4F,0x55,0xA5,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0x21,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
```

```
xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 720 /* Arabic (OEM) */
#define _EXCVT {0x80,0x81,0x45,0x41,0x84,0x41,0x86,0x43,0x45,0x45,0x45,0x49,0x49,0x8D,0x8E,0x8F,0x90,0x92,0x92,0x93,0x94,0x95,0x49,0x49,0x98,0x99,0x9A,0x9B,0x9C,0x9D,0x9E,0x9F, \
                                0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 737 /* Greek (OEM) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0x8D,0x8E,0x8F,0x90,0x92,0x92,0x93,0x94,0x95,0x96,0x97,0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87, \
                                0x88,0x89,0x8A,0x8B,0x8C,0x8D,0x8E,0x8F,0x90,0x91,0xAA,0x92,0x93,0x94,0x95,0x96,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0x97,0xEA,0xEB,0xEC,0xE4,0xED,0xEE,0xE7,0xE8,0xF1,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 775 /* Baltic (OEM) */
#define _EXCVT {0x80,0x9A,0x91,0xA0,0x8E,0x95,0x8F,0x80,0xAD,0xED,0x8A,0x8A,0xA1,0x8D,0x8E,0x8F,0x90,0x92,0x92,0xE2,0x99,0x95,0x96,0x97,0x97,0x99,0x9A,0x9D,0x9C,0x9D,0x9E,0x9F, \
                                0xA0,0xA1,0xE0,0xA3,0xA3,0xA5,0xA6,0xA7,0xA8,0xA9,0xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xB5,0xB6,0xB7,0xB8,0xBD,0xBE,0xC6,0xC7,0xA5,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE5,0xE5,0xE6,0xE3,0xE8,0xE8,0xEA,0xEA,0xEE,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 850 /* Multilingual Latin 1 (OEM) */
#define _EXCVT {0x80,0x9A,0x90,0xB6,0x8E,0xB7,0x8F,0x80,0xD2,0xD3,0xD4,0xD8,0xD7,0xDE,0x8E,0x8F,0x90,0x92,0x92,0xE2,0x99,0xE3,0xEA,0xEB,0x59,0x99,0x9A,0x9D,0x9C,0x9D,0x9E,0x9F, \
                                0xB5,0xD6,0xE0,0xE9,0xA5,0xA5,0xA6,0xA7,0xA8,0xA9,0xAA,0xAB,0xAC,0x21,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC7,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
```

```
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE5,0xE5,0xE6,0xE7,0xE7,0xE9,0
    xEA,0xEB,0xED,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0
    xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 852 /* Latin 2 (OEM) */
#define _EXCVT {0x80,0x9A,0x90,0xB6,0x8E,0xDE,0x8F,0x80,0x9D,0xD3,0x8A,0x8A,0xD7,0
    x8D,0x8E,0x8F,0x90,0x91,0x91,0xE2,0x99,0x95,0x95,0x97,0x97,0x99,0x9A,0x9B,0x9B,0
    x9D,0x9E,0x9F, \
                                0xB5,0xD6,0xE0,0xE9,0xA4,0xA4,0xA6,0xA6,0xA8,0xA8,0
    xAA,0x8D,0xAC,0xB8,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBD,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC6,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD1,0xD1,0xD2,0xD3,0xD2,0xD5,0xD6,0xD7,0xB7,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE3,0xD5,0xE6,0xE6,0xE8,0xE9,0
    xE8,0xEB,0xED,0xED,0xDD,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0
    xFA,0xEB,0xFC,0xFC,0xFE,0xFF}

#elif _CODE_PAGE == 855 /* Cyrillic (OEM) */
#define _EXCVT {0x81,0x81,0x83,0x83,0x85,0x85,0x87,0x87,0x89,0x89,0x8B,0x8B,0x8D,0
    x8D,0x8F,0x8F,0x91,0x91,0x93,0x93,0x95,0x95,0x97,0x97,0x99,0x99,0x9B,0x9B,0x9D,0
    x9D,0x9F,0x9F, \
                                0xA1,0xA1,0xA3,0xA3,0xA5,0xA5,0xA7,0xA7,0xA9,0xA9,0
    xAB,0xAB,0xAD,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB6,0xB6,0xB8,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBE,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC7,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD1,0xD1,0xD3,0xD3,0xD5,0xD5,0xD7,0xD7,0xDD,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xE0,0xDF, \
                                0xE0,0xE2,0xE2,0xE4,0xE4,0xE6,0xE6,0xE8,0xE8,0xEA,0
    xEA,0xEC,0xEC,0xEE,0xEE,0xEF,0xF0,0xF2,0xF2,0xF4,0xF4,0xF6,0xF6,0xF8,0xF8,0xFA,0
    xFA,0xFC,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 857 /* Turkish (OEM) */
#define _EXCVT {0x80,0x9A,0x90,0xB6,0x8E,0xB7,0x8F,0x80,0xD2,0xD3,0xD4,0xD8,0xD7,0
    x98,0x8E,0x8F,0x90,0x92,0x92,0xE2,0x99,0xE3,0xEA,0xEB,0x98,0x99,0x9A,0x9D,0x9C,0
    x9D,0x9E,0x9E, \
                                0xB5,0xD6,0xE0,0xE9,0xA5,0xA5,0xA6,0xA6,0xA8,0xA9,0
    xAA,0xAB,0xAC,0x21,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC7,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE5,0xE5,0xE6,0xE7,0xE8,0xE9,0
    xEA,0xEB,0xDE,0x59,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0
    xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 858 /* Multilingual Latin 1 + Euro (OEM) */
#define _EXCVT {0x80,0x9A,0x90,0xB6,0x8E,0xB7,0x8F,0x80,0xD2,0xD3,0xD4,0xD8,0xD7,0
    xDE,0x8E,0x8F,0x90,0x92,0x92,0xE2,0x99,0xE3,0xEA,0xEB,0x59,0x99,0x9A,0x9D,0x9C,0
    x9D,0x9E,0x9F, \
                                0xB5,0xD6,0xE0,0xE9,0xA5,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0x21,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC7,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD1,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE5,0xE5,0xE6,0xE7,0xE7,0xE9,0
    xEA,0xEB,0xED,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0
    xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 862 /* Hebrew (OEM) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0
    x9D,0x9E,0x9F, \
                                0x41,0x49,0x4F,0x55,0xA5,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0x21,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0
    xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0
    xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 866 /* Russian (OEM) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0
    x9D,0x9E,0x9F, \
                                0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0
    x8A,0x8B,0x8C,0x8D,0x8E,0x8F,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0x90,0x91,0x92,0x93,0x9d,0x95,0x96,0x97,0x98,0x99,0
    x9A,0x9B,0x9C,0x9D,0x9E,0x9F,0xF0,0xF0,0xF2,0xF2,0xF4,0xF4,0xF6,0xF6,0xF8,0xF9,0
    xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 874 /* Thai (OEM, Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0
    x9D,0x9E,0x9F, \
                                0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0
    xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0
    xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 1250 /* Central Europe (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x8A,0x9B,0x8C,0
    x8D,0x8E,0x8F, \
                                0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xA3,0xB4,0xB5,0xB6,0xB7,0xB8,0xA5,0
    xAA,0xBB,0xBC,0xBD,0xBC,0xAF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xF7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xFF}

#elif _CODE_PAGE == 1251 /* Cyrillic (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x82,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x80,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x8A,0x9B,0x8C,0
    x8D,0x8E,0x8F, \
                                0xA0,0xA2,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB2,0xA5,0xB5,0xB6,0xB7,0xA8,0xB9,0
    xAA,0xBB,0xA3,0xBD,0xBD,0xAF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF}

#elif _CODE_PAGE == 1252 /* Latin 1 (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0xAd,0x9B,0x8C,0
    x9D,0xAE,0x9F, \
                                0xA0,0x21,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xF7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0x9F}

#elif _CODE_PAGE == 1253 /* Greek (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0
    x9D,0x9E,0x9F, \
                                0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xA2,0xB8,0xB9,0xBA, \
                                0xE0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xF2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xFB,0xBC,0xFD,0xBF,0xFF}

#elif _CODE_PAGE == 1254 /* Turkish (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x8A,0x9B,0x8C,0
    x9D,0x9E,0x9F, \
                                0xA0,0x21,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                                0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xF7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0x9F}

#elif _CODE_PAGE == 1255 /* Hebrew (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0
    x9D,0x9E,0x9F, \
                   0xA0,0x21,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                   0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                   0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0
    xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0
    xFA,0xFB,0xFC,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 1256 /* Arabic (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x8C,0
    x9D,0x9E,0x9F, \
                   0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                   0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                   0x41,0xE1,0x41,0xE3,0xE4,0xE5,0xE6,0x43,0x45,0x45,0
    x45,0x45,0xEC,0xED,0x49,0x49,0xF0,0xF1,0xF2,0xF3,0x4F,0xF5,0xF6,0xF7,0xF8,0x55,0
    xFA,0x55,0x55,0xFD,0xFE,0xFF}

#elif _CODE_PAGE == 1257 /* Baltic (Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0
    x9D,0x9E,0x9F, \
                   0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xA8,0xB9,0
    xAA,0xBB,0xBC,0xBD,0xBE,0xAF, \
                   0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                   0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xF7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xFF}

#elif _CODE_PAGE == 1258 /* Vietnam (OEM, Windows) */
#define _EXCVT {0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0
    x8D,0x8E,0x8F,0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0xAC,0
    x9D,0x9E,0x9F, \
                   0xA0,0x21,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0
    xAA,0xAB,0xAC,0xAD,0xAE,0xAF,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0
    xBA,0xBB,0xBC,0xBD,0xBE,0xBF, \
                   0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xCC,0xCD,0xCE,0xCF,0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xDE,0xDF, \
                   0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0
    xCA,0xCB,0xEC,0xCD,0xCE,0xCF,0xD0,0xD1,0xF2,0xD3,0xD4,0xD5,0xD6,0xF7,0xD8,0xD9,0
    xDA,0xDB,0xDC,0xDD,0xFE,0x9F}
```

```
#else
#error Unknown code page.

#endif

/* Character code support macros */

#define IsUpper(c)      (((c)>='A')&&((c)<='Z'))
#define IsLower(c)      (((c)>='a')&&((c)<='z'))

#ifndef _EXCVT  /* DBCS configuration */

#ifdef _DF2S    /* Two 1st byte areas */
#define IsDBCS1(c)      (((BYTE)(c) >= _DF1S && (BYTE)(c) <= _DF1E) || ((BYTE)(c) >=
     _DF2S && (BYTE)(c) <= _DF2E))
#else                   /* One 1st byte area */
#define IsDBCS1(c)      ((BYTE)(c) >= _DF1S && (BYTE)(c) <= _DF1E)
#endif

#ifdef _DS3S    /* Three 2nd byte areas */
#define IsDBCS2(c)      (((BYTE)(c) >= _DS1S && (BYTE)(c) <= _DS1E) || ((BYTE)(c) >=
     _DS2S && (BYTE)(c) <= _DS2E) || ((BYTE)(c) >= _DS3S && (BYTE)(c) <= _DS3E))
#else                   /* Two 2nd byte areas */
#define IsDBCS2(c)      (((BYTE)(c) >= _DS1S && (BYTE)(c) <= _DS1E) || ((BYTE)(c) >=
     _DS2S && (BYTE)(c) <= _DS2E))
#endif

#else                   /* SBCS configuration */

#define IsDBCS1(c)      0
#define IsDBCS2(c)      0

#endif /* _EXCVT */

/* FatFs refers the members in the FAT structures with byte offset instead
/ of structure member because there are incompatibility of the packing option
/ between various compilers. */

#define BS_jmpBoot                      0
#define BS_OEMName                      3
#define BPB_BytsPerSec          11
#define BPB_SecPerClus          13
#define BPB_RsvdSecCnt          14
#define BPB_NumFATs                     16
#define BPB_RootEntCnt          17
#define BPB_TotSec16            19
#define BPB_Media                       21
#define BPB_FATSz16                     22
#define BPB_SecPerTrk           24
#define BPB_NumHeads            26
#define BPB_HiddSec                     28
#define BPB_TotSec32            32
#define BS_55AA                         510

#define BS_DrvNum                       36
#define BS_BootSig                      38
```

```
#define BS_VolID                        39
#define BS_VolLab                       43
#define BS_FilSysType           54

#define BPB_FATSz32                     36
#define BPB_ExtFlags            40
#define BPB_FSVer                       42
#define BPB_RootClus            44
#define BPB_FSInfo                      48
#define BPB_BkBootSec           50
#define BS_DrvNum32                     64
#define BS_BootSig32            66
#define BS_VolID32                      67
#define BS_VolLab32                     71
#define BS_FilSysType32         82

#define MBR_Table                       446

#define DIR_Name                        0
#define DIR_Attr                        11
#define DIR_NTres                       12
#define DIR_CrtTime                     14
#define DIR_CrtDate                     16
#define DIR_FstClusHI           20
#define DIR_WrtTime                     22
#define DIR_WrtDate                     24
#define DIR_FstClusLO           26
#define DIR_FileSize            28

/*---------------------------------------------------------------------------

   Private Functions

---------------------------------------------------------------------------*/


static
FATFS *FatFs;   /* Pointer to the file system object (logical drive) */

/* Fill memory */
static
void mem_set (void* dst, int val, int cnt) {
        char *d = (char*)dst;
        while (cnt--) *d++ = (char)val;
}

/* Compare memory to memory */
static
int mem_cmp (const void* dst, const void* src, int cnt) {
        const char *d = (const char *)dst, *s = (const char *)src;
        int r = 0;
        while (cnt-- && (r = *d++ - *s++) == 0) ;
        return r;
}

/*-----------------------------------------------------------------------*/
/* FAT access - Read value of a FAT entry                                */
/*-----------------------------------------------------------------------*/
```

```
static
CLUST get_fat ( /* 1:IO error, Else:Cluster status */
        CLUST clst      /* Cluster# to get the link information */
)
{
        BYTE buf[4];
        FATFS *fs = FatFs;

        if (clst < 2 || clst >= fs->n_fatent)   /* Range check */
                return 1;

        switch (fs->fs_type) {
#if _FS_FAT12
        case FS_FAT12 : {
                UINT wc, bc, ofs;

                bc = (UINT)clst; bc += bc / 2;
                ofs = bc % 512; bc /= 512;
                if (ofs != 511) {
                        if (disk_readp(buf, fs->fatbase + bc, ofs, 2)) break;
                } else {
                        if (disk_readp(buf, fs->fatbase + bc, 511, 1)) break;
                        if (disk_readp(buf+1, fs->fatbase + bc + 1, 0, 1)) break;
                }
                wc = LD_WORD(buf);
                return (clst & 1) ? (wc >> 4) : (wc & 0xFFF);
        }
#endif
#if _FS_FAT16
        case FS_FAT16 :
                if (disk_readp(buf, fs->fatbase + clst / 256, ((UINT)clst % 256) *
    2, 2)) break;
                return LD_WORD(buf);
#endif
#if _FS_FAT32
        case FS_FAT32 :
                if (disk_readp(buf, fs->fatbase + clst / 128, ((UINT)clst % 128) *
    4, 4)) break;
                return LD_DWORD(buf) & 0x0FFFFFFF;
#endif
        }

        return 1;       /* An error occured at the disk I/O layer */
}

/*-----------------------------------------------------------------------*/
/* Get sector# from cluster# / Get cluster field from directory entry    */
/*-----------------------------------------------------------------------*/

static
DWORD clust2sect (      /* !=0: Sector number, 0: Failed - invalid cluster# */
        CLUST clst              /* Cluster# to be converted */
)
{
        FATFS *fs = FatFs;
        clst -= 2;
        if (clst >= (fs->n_fatent - 2)) return 0;               /* Invalid cluster#
    */
        return (DWORD)clst * fs->csize + fs->database;
}

static
CLUST get_clust (
        BYTE* dir               /* Pointer to directory entry */
)
{
        FATFS *fs = FatFs;
        CLUST clst = 0;


        if (_FS_32ONLY || (_FS_FAT32 && fs->fs_type == FS_FAT32)) {
                clst = LD_WORD(dir+DIR_FstClusHI);
                clst <<= 16;
        }
        clst |= LD_WORD(dir+DIR_FstClusLO);

        return clst;
}

/*-----------------------------------------------------------------------*/
/* Directory handling - Rewind directory index                           */
/*-----------------------------------------------------------------------*/

static
FRESULT dir_rewind (
        DIR *dj                 /* Pointer to directory object */
)
{
        CLUST clst;
        FATFS *fs = FatFs;

        dj->index = 0;
        clst = dj->sclust;
        if (clst == 1 || clst >= fs->n_fatent)  /* Check start cluster range */
                return FR_DISK_ERR;
        if (_FS_FAT32 && !clst && (_FS_32ONLY || fs->fs_type == FS_FAT32))      /*
    Replace cluster# 0 with root cluster# if in FAT32 */
                clst = (CLUST)fs->dirbase;
        dj->clust = clst;                                               /* Current
    cluster */
        dj->sect = (_FS_32ONLY || clst) ? clust2sect(clst) : fs->dirbase;       /*
    Current sector */

        return FR_OK;   /* Seek succeeded */
}

/*-----------------------------------------------------------------------*/
/* Directory handling - Move directory index next                        */
/*-----------------------------------------------------------------------*/

static
```

```
FRESULT dir_next (      /* FR_OK:Succeeded, FR_NO_FILE:End of table */
        DIR *dj                 /* Pointer to directory object */
)
{
        CLUST clst;
        WORD i;
        FATFS *fs = FatFs;


        i = dj->index + 1;
        if (!i || !dj->sect)    /* Report EOT when index has reached 65535 */
                return FR_NO_FILE;

        if (!(i % 16)) {                /* Sector changed? */
                dj->sect++;                     /* Next sector */

                if (dj->clust == 0) {   /* Static table */
                        if (i >= fs->n_rootdir) /* Report EOT when end of table */
                                return FR_NO_FILE;
                }
                else {                                  /* Dynamic table */
                        if (((i / 16) & (fs->csize - 1)) == 0) {        /* Cluster changed? */
                                clst = get_fat(dj->clust);              /* Get next cluster */
                                if (clst <= 1) return FR_DISK_ERR;
                                if (clst >= fs->n_fatent)               /* When it reached end of dynamic table */
                                        return FR_NO_FILE;                      /* Report EOT */
                                dj->clust = clst;                               /* Initialize data for new cluster */
                                dj->sect = clust2sect(clst);
                        }
                }
        }

        dj->index = i;

        return FR_OK;
}

/*-----------------------------------------------------------------------*/
/* Directory handling - Find an object in the directory                  */
/*-----------------------------------------------------------------------*/

static
FRESULT dir_find (
        DIR *dj,                /* Pointer to the directory object linked to the file name */
        BYTE *dir               /* 32-byte working buffer */
)
{
        FRESULT res;
        BYTE c;

        res = dir_rewind(dj);                   /* Rewind directory object */
        if (res != FR_OK) return res;

        do {
                res = disk_readp(dir, dj->sect, (dj->index % 16) * 32, 32)      /* Read an entry */
                        ? FR_DISK_ERR : FR_OK;
                if (res != FR_OK) break;
                c = dir[DIR_Name];      /* First character */
                if (c == 0) { res = FR_NO_FILE; break; }        /* Reached to end of table */
                if (!(dir[DIR_Attr] & AM_VOL) && !mem_cmp(dir, dj->fn, 11)) /* Is it a valid entry? */
                        break;
                res = dir_next(dj);                                     /* Next entry */
        } while (res == FR_OK);

        return res;
}

/*-----------------------------------------------------------------------*/
/* Read an object from the directory                                     */
/*-----------------------------------------------------------------------*/
#if _USE_DIR
static
FRESULT dir_read (
        DIR *dj,                /* Pointer to the directory object to store read object name */
        BYTE *dir               /* 32-byte working buffer */
)
{
        FRESULT res;
        BYTE a, c;

        res = FR_NO_FILE;
        while (dj->sect) {
                res = disk_readp(dir, dj->sect, (dj->index % 16) * 32, 32)      /* Read an entry */
                        ? FR_DISK_ERR : FR_OK;
                if (res != FR_OK) break;
                c = dir[DIR_Name];
                if (c == 0) { res = FR_NO_FILE; break; }        /* Reached to end of table */
                a = dir[DIR_Attr] & AM_MASK;
                if (c != 0xE5 && c != '.' && !(a & AM_VOL))     /* Is it a valid entry? */
                        break;
                res = dir_next(dj);                     /* Next entry */
                if (res != FR_OK) break;
        }

        if (res != FR_OK) dj->sect = 0;

        return res;
}
#endif
```

```
/*-----------------------------------------------------------------------*/
/* Pick a segment and create the object name in directory form           */
/*-----------------------------------------------------------------------*/

static
FRESULT create_name (
        DIR *dj,                        /* Pointer to the directory object */
        const char **path       /* Pointer to pointer to the segment in the path
    string */
)
{
        BYTE c, ni, si, i, *sfn;
        const char *p;
#if _USE_LCC
#ifdef _EXCVT
        static const BYTE cvt[] = _EXCVT;
#endif
#endif

        /* Create file name in directory form */
        sfn = dj->fn;
        mem_set(sfn, '␣', 11);
        si = i = 0; ni = 8;
        p = *path;
        for (;;) {
                c = p[si++];
                if (c <= '␣' || c == '/') break;        /* Break on end of segment
    */
                if (c == '.' || i >= ni) {
                        if (ni != 8 || c != '.') break;
                        i = 8; ni = 11;
                        continue;
                }
#if _USE_LCC
#ifdef _EXCVT
                if (c >= 0x80)                                  /* To upper extended
     char (SBCS) */
                        c = cvt[c - 0x80];
#endif
                if (IsDBCS1(c) && i < ni - 1) { /* DBC 1st byte? */
                        BYTE d = p[si++];                       /* Get 2nd byte */
                        sfn[i++] = c;
                        sfn[i++] = d;
                } else
#endif
                {                                               /* Single byte code
    */
                        if (_USE_LCC && IsLower(c)) c -= 0x20;  /* toupper */
                        sfn[i++] = c;
                }
        }
        *path = &p[si];                                         /* Rerurn pointer to
     the next segment */

        sfn[11] = (c <= '␣') ? 1 : 0;           /* Set last segment flag if end of
    path */
```

```
        return FR_OK;
}

/*-----------------------------------------------------------------------*/
/* Get file information from directory entry                             */
/*-----------------------------------------------------------------------*/
#if _USE_DIR
static
void get_fileinfo (              /* No return code */
        DIR *dj,                        /* Pointer to the directory object */
        BYTE *dir,                      /* 32-byte working buffer */
        FILINFO *fno             /* Pointer to store the file information */
)
{
        BYTE i, c;
        char *p;

        p = fno->fname;
        if (dj->sect) {
                for (i = 0; i < 8; i++) {       /* Copy file name body */
                        c = dir[i];
                        if (c == '␣') break;
                        if (c == 0x05) c = 0xE5;
                        *p++ = c;
                }
                if (dir[8] != '␣') {            /* Copy file name extension */
                        *p++ = '.';
                        for (i = 8; i < 11; i++) {
                                c = dir[i];
                                if (c == '␣') break;
                                *p++ = c;
                        }
                }
                fno->fattrib = dir[DIR_Attr];                           /* Attribute
    */
                fno->fsize = LD_DWORD(dir+DIR_FileSize);        /* Size */
                fno->fdate = LD_WORD(dir+DIR_WrtDate);          /* Date */
                fno->ftime = LD_WORD(dir+DIR_WrtTime);          /* Time */
        }
        *p = 0;
}
#endif /* _USE_DIR */

/*-----------------------------------------------------------------------*/
/* Follow a file path                                                    */
/*-----------------------------------------------------------------------*/

static
FRESULT follow_path (   /* FR_OK(0): successful, !=0: error code */
        DIR *dj,                        /* Directory object to return last directory
    and found object */
        BYTE *dir,                      /* 32-byte working buffer */
        const char *path         /* Full-path string to find a file or directory */
)
{
        FRESULT res;
        while (*path == '␣') path++;            /* Strip leading spaces */
        if (*path == '/') path++;                       /* Strip heading separator if exist */
        dj->sclust = 0;                                         /* Set start directory (always root dir) */

        if ((BYTE)*path < '␣') {                        /* Null path means the root directory */
                res = dir_rewind(dj);
                dir[0] = 0;

        } else {                                                        /* Follow path */
                for (;;) {
                        res = create_name(dj, &path);   /* Get a segment */
                        if (res != FR_OK) break;
                        res = dir_find(dj, dir);                /* Find it */
                        if (res != FR_OK) break;                /* Could not find the object */
                        if (dj->fn[11]) break;                  /* Last segment match. Function completed. */
                        if (!(dir[DIR_Attr] & AM_DIR)) { /* Cannot follow path because it is a file */
                                res = FR_NO_FILE; break;
                        }
                        dj->sclust = get_clust(dir);    /* Follow next */
                }
        }

        return res;
}

/*-----------------------------------------------------------------------*/
/* Check a sector if it is an FAT boot record                            */
/*-----------------------------------------------------------------------*/

static
BYTE check_fs ( /* 0:The FAT boot record, 1:Valid boot record but not an FAT, 2:Not a boot record, 3:Error */
        BYTE *buf,      /* Working buffer */
        DWORD sect      /* Sector# (lba) to check if it is an FAT boot record or not */
)
{
        if (disk_readp(buf, sect, 510, 2))              /* Read the boot record */
                return 3;
        if (LD_WORD(buf) != 0xAA55)                             /* Check record signature */
                return 2;

        if (!_FS_32ONLY && !disk_readp(buf, sect, BS_FilSysType, 2) && LD_WORD(buf) == 0x4146)  /* Check FAT12/16 */
                return 0;
        if (_FS_FAT32 && !disk_readp(buf, sect, BS_FilSysType32, 2) && LD_WORD(buf) == 0x4146)  /* Check FAT32 */
                return 0;
        return 1;
```

```
}

/*---------------------------------------------------------------------------

   Public Functions

----------------------------------------------------------------------------*/

/*-----------------------------------------------------------------------*/
/* Mount/Unmount a Locical Drive                                         */
/*-----------------------------------------------------------------------*/

FRESULT pf_mount (
        FATFS *fs               /* Pointer to new file system object */
)
{
        BYTE fmt, buf[36];
        DWORD bsect, fsize, tsect, mclst;

        FatFs = 0;

        if (disk_initialize() & STA_NOINIT)     /* Check if the drive is ready or not */
                return FR_NOT_READY;

        /* Search FAT partition on the drive */
        bsect = 0;
        fmt = check_fs(buf, bsect);                     /* Check sector 0 as an SFD format */
        if (fmt == 1) {                                         /* Not an FAT boot record, it may be FDISK format */
                /* Check a partition listed in top of the partition table */
                if (disk_readp(buf, bsect, MBR_Table, 16)) {    /* 1st partition entry */
                        fmt = 3;
                } else {
                        if (buf[4]) {                                   /* Is the partition existing? */
                                bsect = LD_DWORD(&buf[8]);      /* Partition offset in LBA */
                                fmt = check_fs(buf, bsect);     /* Check the partition */
                        }
                }
        }
        if (fmt == 3) return FR_DISK_ERR;
        if (fmt) return FR_NO_FILESYSTEM;       /* No valid FAT patition is found */

        /* Initialize the file system object */
        if (disk_readp(buf, bsect, 13, sizeof (buf))) return FR_DISK_ERR;

        fsize = LD_WORD(buf+BPB_FATSz16-13);                            /* Number of sectors per FAT */
        if (!fsize) fsize = LD_DWORD(buf+BPB_FATSz32-13);

        fsize *= buf[BPB_NumFATs-13];                                   /* Number of sectors in FAT area */
        fs->fatbase = bsect + LD_WORD(buf+BPB_RsvdSecCnt-13); /* FAT start sector (
    lba) */
        fs->csize = buf[BPB_SecPerClus-13];                                     /*
    Number of sectors per cluster */
        fs->n_rootdir = LD_WORD(buf+BPB_RootEntCnt-13);         /* Nmuber of root
    directory entries */
        tsect = LD_WORD(buf+BPB_TotSec16-13);                           /* Number of
     sectors on the file system */
        if (!tsect) tsect = LD_DWORD(buf+BPB_TotSec32-13);
        mclst = (tsect                                          /* Last cluster# + 1
     */
                - LD_WORD(buf+BPB_RsvdSecCnt-13) - fsize - fs->n_rootdir / 16
                ) / fs->csize + 2;
        fs->n_fatent = (CLUST)mclst;

        fmt = 0;                                                        /* Determine
     the FAT sub type */
        if (_FS_FAT12 && mclst < 0xFF7)
                fmt = FS_FAT12;
        if (_FS_FAT16 && mclst >= 0xFF8 && mclst < 0xFFF7)
                fmt = FS_FAT16;
        if (_FS_FAT32 && mclst >= 0xFFF7)
                fmt = FS_FAT32;
        if (!fmt) return FR_NO_FILESYSTEM;
        fs->fs_type = fmt;

        if (_FS_32ONLY || (_FS_FAT32 && fmt == FS_FAT32))
                fs->dirbase = LD_DWORD(buf+(BPB_RootClus-13));  /* Root directory
    start cluster */
        else
                fs->dirbase = fs->fatbase + fsize;                              /*
    Root directory start sector (lba) */
        fs->database = fs->fatbase + fsize + fs->n_rootdir / 16;        /* Data
    start sector (lba) */

        fs->flag = 0;
        FatFs = fs;

        return FR_OK;
}

/*-----------------------------------------------------------------------*/
/* Open or Create a File                                                 */
/*-----------------------------------------------------------------------*/

FRESULT pf_open (
        const char *path        /* Pointer to the file name */
)
{
        FRESULT res;
        DIR dj;
        BYTE sp[12], dir[32];
        FATFS *fs = FatFs;

        if (!fs) return FR_NOT_ENABLED;         /* Check file system */

        fs->flag = 0;
        dj.fn = sp;
        res = follow_path(&dj, dir, path);      /* Follow the file path */

        if (res != FR_OK) return res;           /* Follow failed */
        if (!dir[0] || (dir[DIR_Attr] & AM_DIR))        /* It is a directory */
                return FR_NO_FILE;

        fs->org_clust = get_clust(dir);         /* File start cluster */
        fs->fsize = LD_DWORD(dir+DIR_FileSize); /* File size */
        fs->fptr = 0;                                           /* File pointer */
        fs->flag = FA_OPENED;

        return FR_OK;
}

/*-----------------------------------------------------------------------*/
/* Read File                                                             */
/*-----------------------------------------------------------------------*/
#if _USE_READ

FRESULT pf_read (
        void* buff,             /* Pointer to the read buffer (NULL:Forward data to
    the stream)*/
        UINT btr,               /* Number of bytes to read */
        UINT* br                /* Pointer to number of bytes read */
)
{
        DRESULT dr;
        CLUST clst;
        DWORD sect, remain;
        UINT rcnt;
        BYTE cs, *rbuff = buff;
        FATFS *fs = FatFs;

        *br = 0;
        if (!fs) return FR_NOT_ENABLED;         /* Check file system */
        if (!(fs->flag & FA_OPENED))            /* Check if opened */
                return FR_NOT_OPENED;

        remain = fs->fsize - fs->fptr;
        if (btr > remain) btr = (UINT)remain;                   /* Truncate btr by
    remaining bytes */

        while (btr)     {
                    /* Repeat until all data transferred */
                if ((fs->fptr % 512) == 0) {                            /* On the
    sector boundary? */
                        cs = (BYTE)(fs->fptr / 512 & (fs->csize - 1));  /* Sector
    offset in the cluster */
                        if (!cs) {
                    /* On the cluster boundary? */
                                if (fs->fptr == 0)
            /* On the top of the file? */
                                        clst = fs->org_clust;
                                else
                                        clst = get_fat(fs->curr_clust);
                                if (clst <= 1) ABORT(FR_DISK_ERR);
                                fs->curr_clust = clst;                          /*
    Update current cluster */
                        }
                        sect = clust2sect(fs->curr_clust);              /* Get
    current sector */
                        if (!sect) ABORT(FR_DISK_ERR);
                        fs->dsect = sect + cs;
                }
                rcnt = 512 - (UINT)fs->fptr % 512;                      /* Get
    partial sector data from sector buffer */
                if (rcnt > btr) rcnt = btr;
                dr = disk_readp(!buff ? 0 : rbuff, fs->dsect, (UINT)fs->fptr % 512,
    rcnt);
                if (dr) ABORT(FR_DISK_ERR);
                fs->fptr += rcnt; rbuff += rcnt;                        /* Update
    pointers and counters */
                btr -= rcnt; *br += rcnt;
        }

        return FR_OK;
}
#endif

/*-----------------------------------------------------------------------*/
/* Write File                                                            */
/*-----------------------------------------------------------------------*/
#if _USE_WRITE

FRESULT pf_write (
        const void* buff,       /* Pointer to the data to be written */
        UINT btw,                       /* Number of bytes to write (0:Finalize the
    current write operation) */
        UINT* bw                        /* Pointer to number of bytes written */
)
{
        CLUST clst;
        DWORD sect, remain;
        const BYTE *p = buff;
        BYTE cs;
        UINT wcnt;
        FATFS *fs = FatFs;

        *bw = 0;
        if (!fs) return FR_NOT_ENABLED;         /* Check file system */
        if (!(fs->flag & FA_OPENED))            /* Check if opened */
                return FR_NOT_OPENED;

        if (!btw) {             /* Finalize request */
                if ((fs->flag & FA__WIP) && disk_writep(0, 0)) ABORT(FR_DISK_ERR);
                fs->flag &= ~FA__WIP;
                return FR_OK;
        } else {                /* Write data request */
                if (!(fs->flag & FA__WIP))              /* Round-down fptr to the
    sector boundary */
                        fs->fptr &= 0xFFFFFE00;
        }
        remain = fs->fsize - fs->fptr;
        if (btw > remain) btw = (UINT)remain;                   /* Truncate btw by
    remaining bytes */

        while (btw)     {
                    /* Repeat until all data transferred */
                if ((UINT)fs->fptr % 512 == 0) {                        /* On the
    sector boundary? */
                        cs = (BYTE)(fs->fptr / 512 & (fs->csize - 1));  /* Sector
    offset in the cluster */
                        if (!cs) {
                    /* On the cluster boundary? */
                                if (fs->fptr == 0)
            /* On the top of the file? */
                                        clst = fs->org_clust;
                                else
                                        clst = get_fat(fs->curr_clust);
                                if (clst <= 1) ABORT(FR_DISK_ERR);
                                fs->curr_clust = clst;                          /*
    Update current cluster */
                        }
                        sect = clust2sect(fs->curr_clust);              /* Get
    current sector */
                        if (!sect) ABORT(FR_DISK_ERR);
                        fs->dsect = sect + cs;
                        if (disk_writep(0, fs->dsect)) ABORT(FR_DISK_ERR);      /*
    Initiate a sector write operation */
                        fs->flag |= FA__WIP;
                }
                wcnt = 512 - (UINT)fs->fptr % 512;                      /* Number of
     bytes to write to the sector */
                if (wcnt > btw) wcnt = btw;
                if (disk_writep(p, wcnt)) ABORT(FR_DISK_ERR);   /* Send data to the
    sector */
                fs->fptr += wcnt; p += wcnt;                            /* Update
    pointers and counters */
                btw -= wcnt; *bw += wcnt;
                if ((UINT)fs->fptr % 512 == 0) {
                        if (disk_writep(0, 0)) ABORT(FR_DISK_ERR);      /* Finalize
    the currtent secter write operation */
                        fs->flag &= ~FA__WIP;
                }
        }

        return FR_OK;
}
#endif

/*-----------------------------------------------------------------------*/
/* Seek File R/W Pointer                                                 */
/*-----------------------------------------------------------------------*/
#if _USE_LSEEK

FRESULT pf_lseek (
        DWORD ofs               /* File pointer from top of file */
)
{
        CLUST clst;
        DWORD bcs, sect, ifptr;
        FATFS *fs = FatFs;

        if (!fs) return FR_NOT_ENABLED;         /* Check file system */
        if (!(fs->flag & FA_OPENED))            /* Check if opened */
                        return FR_NOT_OPENED;

        if (ofs > fs->fsize) ofs = fs->fsize;   /* Clip offset with the file size */
        ifptr = fs->fptr;
        fs->fptr = 0;
        if (ofs > 0) {
                bcs = (DWORD)fs->csize * 512;   /* Cluster size (byte) */
                if (ifptr > 0 &&
                        (ofs - 1) / bcs >= (ifptr - 1) / bcs) { /* When seek to same
     or following cluster, */
                        fs->fptr = (ifptr - 1) & ~(bcs - 1);    /* start from the
    current cluster */
                        ofs -= fs->fptr;
                        clst = fs->curr_clust;
                } else {                                                        /*
    When seek to back cluster, */
                        clst = fs->org_clust;                   /* start from the
    first cluster */
                        fs->curr_clust = clst;
                }
                while (ofs > bcs) {                             /* Cluster following
     loop */
                        clst = get_fat(clst);           /* Follow cluster chain */
                        if (clst <= 1 || clst >= fs->n_fatent) ABORT(FR_DISK_ERR);
                        fs->curr_clust = clst;
                        fs->fptr += bcs;
                        ofs -= bcs;
                }
                fs->fptr += ofs;
                sect = clust2sect(clst);                /* Current sector */
                if (!sect) ABORT(FR_DISK_ERR);
                fs->dsect = sect + (fs->fptr / 512 & (fs->csize - 1));
        }

        return FR_OK;
}
#endif

/*-----------------------------------------------------------------------*/
/* Create a Directroy Object                                             */
/*-----------------------------------------------------------------------*/
#if _USE_DIR

FRESULT pf_opendir (
        DIR *dj,                        /* Pointer to directory object to create */
        const char *path        /* Pointer to the directory path */
)
{
        FRESULT res;
        BYTE sp[12], dir[32];
        FATFS *fs = FatFs;

        if (!fs) {                                      /* Check file system */
                res = FR_NOT_ENABLED;
        } else {
                dj->fn = sp;
                res = follow_path(dj, dir, path);               /* Follow the path
    to the directory */
                if (res == FR_OK) {                                             /*
    Follow completed */
                        if (dir[0]) {                                           /*
    It is not the root dir */
                                if (dir[DIR_Attr] & AM_DIR)             /* The
    object is a directory */
                                        dj->sclust = get_clust(dir);
                                else
            /* The object is not a directory */
                                        res = FR_NO_FILE;
                        }
                        if (res == FR_OK)
                                res = dir_rewind(dj);                   /* Rewind
    dir */
                }
        }

        return res;
}

/*-------------------------------------------------------------------------*/
/* Read Directory Entry in Sequense                                        */
/*-------------------------------------------------------------------------*/

FRESULT pf_readdir (
        DIR *dj,                        /* Pointer to the open directory object */
        FILINFO *fno            /* Pointer to file information to return */
)
{
        FRESULT res;
        BYTE sp[12], dir[32];
        FATFS *fs = FatFs;

        if (!fs) {                                      /* Check file system */
                res = FR_NOT_ENABLED;
        } else {
                dj->fn = sp;
                if (!fno) {
                        res = dir_rewind(dj);
                } else {
                        res = dir_read(dj, dir);        /* Get current directory
    item */
                        if (res == FR_NO_FILE) res = FR_OK;
                        if (res == FR_OK) {                             /* A valid
    entry is found */
                                get_fileinfo(dj, dir, fno);     /* Get the object
    information */
                                res = dir_next(dj);                     /* Increment
     read index for next */
                                if (res == FR_NO_FILE) res = FR_OK;
                        }
                }
        }

        return res;
}

#endif /* _USE_DIR */
```

lib/screens.h

```c
#pragma once

#include "system.h"

extern int STARTCOORDX;
extern int yCoords[amountoffiles];

extern Button BUTTON_OTS[];
extern Button BUTTON_SS[];
extern Button BUTTON_RP[];
extern NumberBox NUMBERS_RP[];
extern Button BUTTON_QW[];
extern Button BUTTON_TD[];
extern Button BUTTON_SvSet[];
extern Button BUTTON_LF[];
extern Numpad NUMPAD[];
extern Numpad NUMPAD2[];
extern InputValue NumpadInput[];
extern Button BUTTON_CONF[];
extern InputValue ConfigInput[];

extern InputValue ToolInput1[];
extern InputValue ToolInput2[];
extern InputValue ToolInput3[];
extern InputValue ToolInput4[];
extern InputValue ToolInput5[];
extern Button BUTTON_EXEC[];
extern Button BUTTON_StopSc[];


void drawOffsetTeachB();
void drawStartupScreen();
void drawRouteProB();
void drawHomingXY();
void drawToolData();
void drawConfig();

void drawLoadFile();
void drawFiles();

void drawStartS(int x, int y);
void drawPos(int x, int y);
void updateTool(int currentTool);

void updateOffsetDim();

void drawNumPad(Numpad * layout, int x, int y);
void drawNumpNumb(Numpad button, int x, int y);
void pressNumpNumb(Numpad button, int x, int y);

void drawToolInfo(int toolnumber);
void fileField(int x, int y, char *str, BOOLEAN pressed);

void drawWrnSc(char * wrng);
```

lib/screens.c

```
#include "screens.h"

int STARTCOORDX = 15;
int yCoords[amountoffiles] = {300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
     300};

extern const unsigned char * CurrentFont;
extern const unsigned char SmallFont[];
extern const unsigned char BigFont[];
extern const unsigned char SevenSegNumFont[];


extern unsigned short arrow_head_y;
extern unsigned short arrow_head_x;
extern unsigned short arrow_head_45;


void drawStartupScreen()
{
        fillCBox(0, 0, 479, 271, COLOR5);
        CurrentFont = BigFont;
        printCString("CNC-controller", 128, 5, COLOR1, COLOR5);

        printCString("Daan␣Ver␣Eecke", 20, 80, COLOR1, COLOR5);
        printCString("Elias␣Verstappe", 20, 60, COLOR1, COLOR5);
}


void drawRouteProB()
{
        int foreground_color = COLOR_WHITE;
        int background_color = COLOR_BLACK;

        fillCBox(0, 0, 479, 271, COLOR5);
        int i = 0;

        //-------------------BOTTOM BUTTONS------------------//
        CurrentFont = BigFont;
        for (i = 0; i < RPBSIZE; i++) drawTButton(BUTTON_RP[i]);

        //-------------------INPUT BOXES------------------//
        for (i = 0; i < (NUMINPSZ-2); i++) drawInputNumber(NumpadInput[i]);

        //-------------------LABELS------------------//
        CurrentFont = SmallFont;

        printCString("Offset", 39, 6, COLOR_WHITE, COLOR_BLACK);
        // printCString("Clip", 119, 6, COLOR_WHITE, COLOR_BLACK);
        printCString("[mm]", 55, 18, COLOR_WHITE, COLOR_BLACK);
        // printCString("[mm]", 127, 18, COLOR_WHITE, COLOR_BLACK);

        printCString("Board", 47, 92, COLOR_WHITE, COLOR_BLACK);
        printCString("Base", 127, 92, COLOR_WHITE, COLOR_BLACK);

        printCString("␣X:", 4, 35, COLOR1, COLOR5);
        printCString("␣Y:", 4, 65, COLOR1, COLOR5);
        printCString("␣Z:", 4, 109, COLOR1, COLOR5);

        printCString("/10", 61, 123, COLOR_WHITE, COLOR_BLACK);
        printCString("/10", 133, 123, COLOR_WHITE, COLOR_BLACK);

        printCString("<x<", 85, 156, COLOR_WHITE, COLOR_BLACK);
        printCString("<y<", 85, 181, COLOR_WHITE, COLOR_BLACK);

        // printCString("Vectors", 230, 6, COLOR_WHITE, COLOR_BLACK);
        // printCString("from", 173, 35, COLOR_WHITE, COLOR_BLACK);
        // printCString("to", 173, 65, COLOR_WHITE, COLOR_BLACK);

        CurrentFont = BigFont;
        printCString("0", 67, 152, COLOR_WHITE, COLOR_BLACK); // x min
        printCString("0", 67, 177, COLOR_WHITE, COLOR_BLACK); // y min
        printCString("0", 143, 152, COLOR_WHITE, COLOR_BLACK); // x max
        printCString("0", 143, 177, COLOR_WHITE, COLOR_BLACK); // y max
        // printCString("0", 271, 31, COLOR_WHITE, COLOR_BLACK); // vec from
        // printCString("0", 271, 61, COLOR_WHITE, COLOR_BLACK); // vec to


        CurrentFont = SmallFont;
        drawHLine(180, 180, 250);
        printCString("Selected␣file", 180, 167, COLOR_WHITE, COLOR_BLACK);
        printCString(fullPath, 180, 184, COLOR_WHITE, COLOR_BLACK);
}


void drawHomingXY()
{
        strcpy((lastScreen), "Homing");
        fillCBox(0, 0, 479, 271, COLOR5);

        drawCBox(10, 109, 470, 171, COLOR2);

        CurrentFont = BigFont;

        printCString("Homing␣X/Y␣axis", 120, 116, COLOR1, COLOR5);

        fillCBox(80, 140, 400, 166, COLOR4);
        printCString("Press␣STOP␣to␣stop", 96, 145, COLOR1, COLOR4);
}


void drawToolData()
{
        int i;
        fillCBox(0, 0, 479, 271, COLOR5);

        CurrentFont = SmallFont;
        printCString("Depth", 50, 50, COLOR_WHITE, COLOR_BLACK);
        printCString("[mm]", 50, 62, COLOR_WHITE, COLOR_BLACK);
        printCString("Iter", 180, 50, COLOR_WHITE, COLOR_BLACK);
        printCString("Xfeed", 50, 115, COLOR_WHITE, COLOR_BLACK);
        printCString("[mm/min]", 50, 127, COLOR_WHITE, COLOR_BLACK);
        printCString("Zfeed", 180, 115, COLOR_WHITE, COLOR_BLACK);
        printCString("[mm/min]", 180, 127, COLOR_WHITE, COLOR_BLACK);
        printCString("Spindle", 50, 180, COLOR_WHITE, COLOR_BLACK);

        CurrentFont = BigFont;
        for(i = 0; i < TDBSIZE; i++) drawTButton(BUTTON_TD[i]);

        drawToolInfo(1);
}


void drawToolInfo(int toolnumber)
{
        CurrentFont = BigFont;
        int i;

        char tool[1];
        itoa(toolnumber, tool, 10);

        printCString("Tool", 11, 10, COLOR_WHITE, COLOR_BLACK);
        printCString(tool, 91, 10, COLOR_WHITE, COLOR_BLACK);

        switch (toolnumber)
        {
        case 1:
                for (i = 0; i < TOOLINPSZ; i++) drawInputNumber(ToolInput1[i]);
                break;
        case 2:
                for (i = 0; i < TOOLINPSZ; i++) drawInputNumber(ToolInput2[i]);
                break;
        case 3:
                for (i = 0; i < TOOLINPSZ; i++) drawInputNumber(ToolInput3[i]);
                break;
        case 4:
                for (i = 0; i < TOOLINPSZ; i++) drawInputNumber(ToolInput4[i]);
                break;
        case 5:
                for (i = 0; i < TOOLINPSZ; i++) drawInputNumber(ToolInput5[i]);
                break;
        }
}



void drawOffsetTeachB()
{
        int foreground_color = COLOR_WHITE;
        int background_color = COLOR_BLACK;

        fillCBox(0, 0, 479, 271, COLOR5);

        CurrentFont = SmallFont;
        printCString("Offset", 8, 8, foreground_color, background_color);
        printCString("Teach", 8, 20, foreground_color, background_color);

        CurrentFont = BigFont;

        // x/y buttons
```

```
    drawTButton(BUTTON_OTS[0]);
    drawTButton(BUTTON_OTS[1]);
    drawTButton(BUTTON_OTS[2]);
    drawTButton(BUTTON_OTS[3]);
    // // z axis buttons
    drawTButton(BUTTON_OTS[4]);
    drawTButton(BUTTON_OTS[5]);

    // Option buttons
    drawTButton(BUTTON_OTS[6]);
    drawTButton(BUTTON_OTS[7]);


    // centre axis
    CurrentFont = SmallFont;
    int x_top_left = 119;
    int y_top_left = 50;

    printCString("z", x_top_left, y_top_left, foreground_color, background_color
);
    printCString("y", (x_top_left+31), (y_top_left+15), foreground_color,
background_color);
    printCString("x", (x_top_left+55), (y_top_left+42), foreground_color,
background_color);

    drawHCLine((x_top_left+7), (y_top_left+40), 46, foreground_color);
    drawVCLine((x_top_left+11), (y_top_left+14), 33, foreground_color);
    drawCLine((x_top_left+8), (y_top_left+43), (x_top_left+26), (y_top_left+25),
 COLOR1);

    drawBitmap((x_top_left+9), (y_top_left+9), &arrow_head_y);
    drawBitmap((x_top_left+52), (y_top_left+38), &arrow_head_x);
    drawBitmap((x_top_left+24), (y_top_left+22), &arrow_head_45);


    // Precicion selection
    CurrentFont = BigFont;
    printCString("millimeter", 34, 163, foreground_color, background_color);

    fillCBox(28, 180, 238, 247, COLOR4);

    printCString("␣␣1", 50, 185, foreground_color, COLOR4);
    emptyCMBox(33, 185);

    printCString("␣10", 50, 227, foreground_color, COLOR4);
    emptyCMBox(33, 227);

    printCString("␣100", 154, 185, foreground_color, COLOR4);
    emptyCMBox(137, 185);


    // current dimensions
    CurrentFont = BigFont;

    x_top_left = 285;
```

```
        y_top_left = 187;

        printCString("X:", x_top_left, y_top_left, foreground_color,
    background_color);
        printCString("Y:", x_top_left, (y_top_left+19), foreground_color,
    background_color);
        printCString("Z:", (x_top_left), (y_top_left+38), foreground_color,
    background_color);

        drawVCLine((x_top_left+133), (x_top_left+133), 55, foreground_color);
        drawVCLine((x_top_left+134), (y_top_left-1), 55, foreground_color);
        printCString("mm", (x_top_left+140), (y_top_left+19), foreground_color,
    background_color);
}


void drawLoadFile()
{
        fillCBox(0, 0, 479, 271, COLOR_BLACK);

        buttonBox(7, 7, 457, 29, COLOR226, COLOR107);
        CurrentFont = BigFont;
        printCString("Load␣file", 10, 10, COLOR_BLACK, COLOR226);

        int i = 0;
        for (i = 0; i < LFSIZE; i++) drawTButton(BUTTON_LF[i]);
}


void drawFiles()
{
        int startCoordY = 40;
        fillCBox(STARTCOORDX, startCoordY, STARTCOORDX+260, startCoordY+281,
    COLOR_BLACK);

        int XSPACING = 32;
        int YSPACING = 32;

    int i, j = 0;
        char numbersBuff[3];

        // TODO: '-1' for the empty entry
    for (i = 0; i < (dirNumber); i++)
        {
                strcat(dirArray[i], "/");
                fileField(STARTCOORDX, (startCoordY+(i*YSPACING)), dirArray[i],
    FALSE);

                strcpy(fileAndInf[i], dirArray[i]);
                // strcat(fileAndInf[i], "/");
                yCoords[i] = (startCoordY+(i*YSPACING));

        }
    for (j = 0; j < (fileNumber); j++)
        {
                fileField(STARTCOORDX, (startCoordY+(i*YSPACING)+(j*YSPACING)),
    fileArray[j], FALSE);
```

```
                strcpy(fileAndInf[i+j],  fileArray[j]);
                yCoords[i+j] = (startCoordY+(i*YSPACING)+(j*YSPACING));
        }
}


void fileField(int x, int y, char *str, BOOLEAN pressed)
{
        if (pressed)
        {
                fillCBox(x, y, x+280, y+16, COLOR_GREY);
                printCString(str, x, y, COLOR_RED, COLOR_GREY);
        } else {
                fillCBox(x, y, x+280, y+16, COLOR_BLACK);
                printCString(str, x, y, COLOR_WHITE, COLOR_BLACK);
        }
}


void updateOffsetDim()
{
        CurrentFont = BigFont;
        int x_top_left = 285;
        int y_top_left = 187;

        char snum[5];

        fillCBox(x_top_left+46, y_top_left, x_top_left+112, y_top_left+56,
    COLOR_BLACK);

        itoa(((CNC_XPos_n*100)/3937), snum, 10);
        printCString(snum, x_top_left+48, y_top_left, COLOR_WHITE, COLOR_BLACK);
            // X

        itoa(((CNC_YPos_n*100)/3937), snum, 10);
        printCString(snum, x_top_left+48, y_top_left+19, COLOR_WHITE, COLOR_BLACK);
         // Y

        itoa(((CNC_ZPos_n*100)/3937), snum, 10);
        printCString(snum, x_top_left+48, y_top_left+38, COLOR_WHITE, COLOR_BLACK);
    // Z
}


void drawNumPad(Numpad * layout, int x, int y)
{
        int i = 0;
        for (i = 0; i < NUMPADSZ; i++) drawNumpNumb(layout[i], x, y);
}


void drawNumpNumb(Numpad button, int x, int y)
{
        CurrentFont = BigFont;

        buttonBox(x+button.origin_x, y+button.origin_y, x+button.origin_x+20, y+
    button.origin_y+31, COLOR_GREY, COLOR_DGREY);

        printCString( button.number,
                                x+button.origin_x+2, y+button.origin_y+3,
                                COLOR_BLACK, COLOR_GREY);
}


void pressNumpNumb(Numpad button, int x, int y)
{
        CurrentFont = BigFont;

        buttonBox(x+button.origin_x, y+button.origin_y, x+button.origin_x+20, y+
    button.origin_y+31, ~COLOR_GREY, ~COLOR_DGREY);

        printCString( button.number,
                                x+button.origin_x+2, y+button.origin_y+3,
                                ~COLOR_BLACK, ~COLOR_GREY);
}


void drawConfig()
{
        fillCBox(0, 0, 479, 271, COLOR_BLACK);

        int i = 0;
        for (i = 0; i < CONFSZ; i++) drawTButton(BUTTON_CONF[i]);

        for (i = 0; i < (CONFINPSZ); i++) drawInputNumber(ConfigInput[i]);

        //-----------------LABELS-----------------//
        CurrentFont = SmallFont;
        printCString("Stroke", 12, 9, COLOR_WHITE, COLOR_BLACK);
        printCString("limit", 12, 26, COLOR_WHITE, COLOR_BLACK);
        printCString("free", 12, 56, COLOR_WHITE, COLOR_BLACK);

        // printCString("Chord angle", 11, 88, COLOR_WHITE, COLOR_BLACK);

        printCString("Correction", 151, 9, COLOR_WHITE, COLOR_BLACK);
        printCString("X:", 151, 26, COLOR1, COLOR5);
        printCString("Y:", 151, 45, COLOR1, COLOR5);
        printCString("Z:", 151, 64, COLOR1, COLOR5);

        CurrentFont = BigFont;
        printCString("0,9975", 169, 22, COLOR_WHITE, COLOR_BLACK);
        printCString("0,9975", 169, 41, COLOR_WHITE, COLOR_BLACK);
        printCString("0,8988", 169, 60, COLOR_WHITE, COLOR_BLACK);
}


void drawStartS(int x, int y)
{
        fillCBox(0, 0, 479, 271, COLOR_BLACK);

        CurrentFont = BigFont;
        drawTButton(BUTTON_EXEC[0]);
        drawPos(x, y);
        updateTool(1);
}


void drawPos(int x, int y)
{
        CurrentFont = BigFont;

        printCString("X:", x, y, COLOR_WHITE, COLOR_BLACK);
        printCString("Y:", x, (y+19), COLOR_WHITE, COLOR_BLACK);
        printCString("Z:", (x), (y+38), COLOR_WHITE, COLOR_BLACK);

        drawVCLine((x+133), (x+133), 55, COLOR_WHITE);
        drawVCLine((x+134), (y-1), 55, COLOR_WHITE);
        printCString("mm", (x+140), (y+19), COLOR_WHITE, COLOR_BLACK);

        char snum[5];
        fillCBox(x+46, y, x+112, y+56, COLOR_BLACK);
        itoa(((CNC_XPos_n*100)/3937), snum, 10);
        printCString(snum, x+48, y, COLOR_WHITE, COLOR_BLACK);  // X
        itoa(((CNC_YPos_n*100)/3937), snum, 10);
        printCString(snum, x+48, y+19, COLOR_WHITE, COLOR_BLACK);       // Y
        itoa(((CNC_ZPos_n*100)/3937), snum, 10);
        printCString(snum, x+48, y+38, COLOR_WHITE, COLOR_BLACK); // Z
}


void updateTool(int currentTool)
{
        char tool[1];
        itoa(currentTool, tool, 10);

        printCString("Tool", 11, 10, COLOR_WHITE, COLOR_BLACK);
        printCString(tool, 91, 10, COLOR_WHITE, COLOR_BLACK);
}


void drawWrnSc(char * wrng)
{
        CurrentFont = BigFont;

        fillCBox(100, 50, 379, 221, COLOR_GREY2);
        drawVCLine(380, 52, 171, COLOR_BLUE);
        drawHCLine(102, 222, 278, COLOR_BLUE);

        drawTButton(BUTTON_StopSc[0]);

        printCString(wrng, 110, 80, COLOR_BLACK, COLOR_GREY2);
}
```

lib/sd.c

```c
#include "system.h"

// #define SDDEBUG

//////////////////////// SD ROUTINES //////////////////////////////
//SD commands, many of these are not used here
#define GO_IDLE_STATE            0
#define SEND_OP_COND             1
#define SEND_IF_COND                     8
#define SEND_CSD                 9
#define STOP_TRANSMISSION        12
#define SEND_STATUS              13
#define SET_BLOCK_LEN            16
#define READ_SINGLE_BLOCK        17
#define READ_MULTIPLE_BLOCKS     18
#define WRITE_SINGLE_BLOCK       24
#define WRITE_MULTIPLE_BLOCKS    25
#define ERASE_BLOCK_START_ADDR   32
#define ERASE_BLOCK_END_ADDR     33
#define ERASE_SELECTED_BLOCKS    38
#define SD_SEND_OP_COND                  41   //ACMD
#define APP_CMD                                  55
#define READ_OCR                                 58
#define CRC_ON_OFF               59

static struct {
        unsigned int  res32;                    // 32-bit response to commands
        unsigned char SDHC_flag;                // false: byte addresses, true: sector addresses
        unsigned char tskdelay;                 // switch task on delays if true
} SDvars;

#define WAITRESP        2400            // max number of bytes waiting for cmd response
#define WAITDATA        24000           // max number of bytes waiting for read data
#define RETRIES         2400            // max number of cmd retries during init
#if 1
//*****************************************************************
//Function      : to send a command to SD card
//Arguments     : cmd: (8-bit command value)
//                        arg: (32-bit command argument)
//return        : response byte
//notes     : CS is left low (for read/write commands)
//*****************************************************************
unsigned int SD_cmd(unsigned int cmd, unsigned int arg)
{
        unsigned int i, R1;

        if(SDvars.SDHC_flag == 0)
                if(cmd == READ_SINGLE_BLOCK ||
                cmd == WRITE_SINGLE_BLOCK )
//     ||
//                 cmd == READ_MULTIPLE_BLOCKS  ||
//                 cmd == WRITE_SINGLE_BLOCK    ||
//                 cmd == WRITE_MULTIPLE_BLOCKS ||
//                 cmd == ERASE_BLOCK_START_ADDR||
//                 cmd == ERASE_BLOCK_END_ADDR )
                {
                        arg <<= 9;
                }
        SD_CS_L();

        SPI_byte(cmd | 0x40);   //send command, first two bits always are '01'
        SPI_byte(arg>>24);              //only 8 LSB bits are actually sent
        SPI_byte(arg>>16);
        SPI_byte(arg>>8);
        SPI_byte(arg);

        //it is compulsory to send correct CRC for CMD8 (CRC=0x87) & CMD0 (CRC=0x95)
        //for remaining commands, CRC is ignored in SPI mode
        SPI_byte((cmd)?0x87:0x95);

        i=WAITRESP;
        while((R1 = SPI_byte(0xFF)) == 0xff) { //wait response
                if(!(--i)) break; //time out error
                //if (SDvars.tskdelay) asm volatile ("svc #0");
        }

        // commands with additional 32-bit response
        if( cmd == READ_OCR || cmd == SEND_IF_COND)
        {
                i=SPI_byte(0xff);
                i=(i<<8)|SPI_byte(0xff);
                i=(i<<8)|SPI_byte(0xff);
                i=(i<<8)|SPI_byte(0xff);
                SDvars.res32=i;
        }

        SPI_byte(0xff); //extra 8 CLK

        return R1; //return state
}

#ifdef SDDEBUG
//******************************************************************
//Function      : to read CSD / CID registers
//arguments : cmd (9: CSD, 10:CID),
//                        pointer to 16-byte buffer
//return        : 0 if no error,
//                        otherwise the response byte will be sent
//******************************************************************
unsigned int SD_read_reg16(unsigned int cmd, unsigned char *buffer)
{
        unsigned int i;

        SD_CS_L();

        SPI_byte(cmd | 0x40);   //send command, first two bits always are '01'
        SPI_byte(0); // Address (dummy)
        SPI_byte(0);
        SPI_byte(0);
        SPI_byte(0);
        SPI_byte(0); // CRC

        i = WAITDATA;
        while(SPI_byte(0xff) != 0xfe) //wait for start block token 0xfe (0x11111110)
                if(!(--i)){SD_CS_H(); return 1;} //return if time-out

        //read 16 bytes of data
        for(i=16; i; i--) *buffer++ = SPI_byte(0xff);

        // 3 extra bytes ignored: 2 for CRC, 1 for extra clocks
        for (i=3;i;i--) SPI_byte(0xff);

        SD_CS_H();

        return 0;
}
#endif

//*******************************************************************
//Function      : to initialize the SD/SDHC card in SPI mode
//Arguments     : flagnt (0: espera activa, 1: llama a siguiente tarea)
//return        : 0 if no error,
//                        otherwise the response byte will be sent
//*******************************************************************
unsigned int SD_init()
{
        unsigned int i, response, SD_version;

        SPI_init();

        SD_CS_H();
        for(i=10;i;i--) SPI_byte(0xff);   //80 clock pulses before sending the first
     command

        SD_CS_L();
        i=32;
        do {
                response = SD_cmd(GO_IDLE_STATE, 0); //send 'reset & go idle'
    command
                SD_CS_H();
                if(!(--i)) return 1;   //time out, card not detected
                //if (SDvars.tskdelay) asm volatile("svc #0");
        } while(response != 0x01);

        SPI_byte(0xff);
        SPI_byte(0xff);

        SD_version = 2; //default set to SD compliance with ver2.x;
                                        //this may change after checking the next
    command
        i=RETRIES;
        do {
                response = SD_cmd(SEND_IF_COND,0x000001AA); //Check power supply
    status, mendatory for SDHC card
                SD_CS_H();
                if(!(--i)) {
                        SD_version = 1;
                        break;
                } //time out
                //if (SDvars.tskdelay) asm volatile("svc #0");
        }while(response != 0x01);
#ifdef SDDEBUG
        _printf("CMD8.RES7=%08x\r\n",SDvars.res32);
        _printf("SD␣version:␣%d\r\n",SD_version);
#endif
        i = RETRIES;
        do {
                response = SD_cmd(APP_CMD,0); //CMD55, must be sent before sending any ACMD command
                SD_CS_H();
                response = SD_cmd(SD_SEND_OP_COND,0x40000000); //ACMD41
                SD_CS_H();
                if(!(--i))  return 2;  //time out, card initialization failed
                //if (SDvars.tskdelay) asm volatile("svc #0");
        }while(response != 0x00);

        i = RETRIES;
        SDvars.SDHC_flag = 0;
        if (SD_version == 2) {
                do {
                        response = SD_cmd(READ_OCR,0);
                        SD_CS_H();
                        if(!(--i)) break;       //time out
                        //if (SDvars.tskdelay) asm volatile("svc #0");
                }while(response != 0x00);
#ifdef SDDEBUG
                _printf("OCR=%08x\r\n",SDvars.res32);
#endif
                SDvars.SDHC_flag = (SDvars.res32>>24) & 0x40;
        }

        SPI_CLK_fast(); // Set a fast SPI clock

#ifdef SDDEBUG
        static unsigned char buf[16];
        i=SD_read_reg16(9,buf);
        _printf("CSD:␣");
        if (i) _printf("Error=%d",i);
        else {
                for (i=0;i<16;i++) _printf("%02x␣",buf[i]);
                for (i=0;i<16;i++) _printf("%c",(buf[i]>'␣' && buf[i]<128)?buf[i]:'.');
        }
        _printf("\r\nCID:␣");
        i=SD_read_reg16(10,buf);
        if (i) _printf("Error=%d",i);
        else {
                for (i=0;i<16;i++) _printf("%02x␣",buf[i]);
                for (i=0;i<16;i++) _printf("%c",(buf[i]>'␣' && buf[i]<128)?buf[i]:'.');
        }
        _printf("\r\n");
#endif
        return 0; //successful return
```

```
}

//*****************************************************************
//Function      : to read a single block from SD card
//Arguments     : Block address (LBA),
//                      : destination buffer address (pointer)
//return        : 0 if no error,
//                        otherwise the response byte will be sent
//*****************************************************************
unsigned int SD_read_Block(unsigned int addr, unsigned char *buffer)
{
        unsigned int i;

        i = SD_cmd(READ_SINGLE_BLOCK, addr); //read a Block command

        //_printf("addr=%x stat=%02x\r\n",addr,i);
        if(i) {SD_CS_H(); return i;}    //check for SD status: 0x00 - OK (No flags set)

        i = WAITDATA;
        while(SPI_byte(0xff) != 0xfe) { //wait for start block token 0xfe (0x11111110)
                if(!(--i)){SD_CS_H(); return 1;} //return if time-out
        }
        //read 512 bytes of data

        #ifndef SD_SSP
        // Sin usar la FIFO del mÃşdulo SSP o con SPI
        for(i=512; i; i--) *buffer++ = SPI_byte(0xff);
        #else
        // Usando la FIFO del mÃşdulo SSP
        SSPDR=0xff; // a la cola TX
        for (i=511;i;i--) {
                SSPDR=0xff;
                while(!(SSPSR&0x4)); // Espera por datos en RX
                *buffer++=SSPDR;
        }
        while(!(SSPSR&0x4));    // Espera por el Ãžltimo dato;
        *buffer++=SSPDR;
        #endif

        // 3 extra bytes ignored: 2 for CRC, 1 for extra clocks
        for (i=3;i;i--) SPI_byte(0xff);

        SD_CS_H();

        return 0;
}
#endif
```

lib/spi.h

```c
#include "system.h"

#define SD_SSP

#ifdef SD_SSP
 #define SPI_init SSP_init
 #define SPI_byte SSP_byte
 #define SPI_CLK_fast()     {SSPCPSR = 4;}      // 14.750 MHz
 #define SPI_CLK_normal()   {SSPCPSR = 30;}     //  2.000 MHz
 #define SPI_CLK_slow()     {SSPCPSR = 148;}    //    400 kHz
 #define SPI_FREQ_LOW()     {S0SPCCR = 30;};    // sets SPI clock to 921,6 kHz
#else
 #define SPI_CLK_fast()     {S0SPCCR = 8;}          // 7.370 MHz
 #define SPI_CLK_normal()   {S0SPCCR = 30;}     // 2.000 MHz
 #define SPI_CLK_slow()     {S0SPCCR = 148;}    //   400 kHz
#endif


#define SD_CS_L() {FIOCLR=(1<<10);}
#define SD_CS_H() {FIOSET=(1<<10);}

void SPI_init();
unsigned int SPI_byte(unsigned int d);

void SSP_init();
void SSP_stop();
unsigned int SSP_byte(unsigned int d);
```

lib/spi.c

```
#include "spi.h"

#ifndef SD_SSP
// SPI routines ––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
void SPI_init() // init SPI interface with a slow clock
{
        // PINSEL0|=(1<<8)|(1<<10)|(1<<12); // P0.4=SCK, P0.5=MISO, P0.6=MOSI
        PINSEL0|=0x00001500;
  // FIODIR|=(1<<3);                                     // SD_CS (P0.3) como salida
        SD_CS_H();

        S0SPCR=(1<<5);        // configure SPI device as master
        S0SPCCR=((PCLK+799999)/800000)*2;       // divisor para 400kHz (redondeo
    arriba, par)
}

inline unsigned int SPI_byte(unsigned int d) // 8–bit data transfer
{
        S0SPDR=d;
        while (!(S0SPSR&0x80));
        return S0SPDR;
}
#else
// SSP routines ––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
void SSP_init()
{
  // PINSEL0|=(1<<8)|(1<<10)|(1<<12); // P0.4=SCK , P0.5=MISO, P0.6=MOSI
  PINSEL0|=0x00001500;
  // FIODIR|=(1<<3);                               // SD_CS (P0.3) como salida
  SD_CS_H();

#define PCSPI (8)
#define PCSSP (21)
  PCONP &= ~(1<<PCSPI); // Disable SPI
  PCONP |=  (1<<PCSSP); // Enable SSP
  //         1111110000000000
  //         5432109876543210
  SSPCR0 = 0b0000000000000111 | (0 << 8);
  //                     ....   DSS Data Size Select   0111     8–bit transfer
  //                   ..           FRF Frame Format         00     SPI
  //                  .         CPOL                    0     clock low
  //                 .          CPHA                    0     first clock edge
  //         ........           SCR Serial Clock Rate   0

  //         00000000
  //         76543210
  SSPCR1 = 0b00000000;
  //                .   LBM Loop Back Mode      0 no loopback
  //               .    SSE SSP Enable      0 disable
  //              .     MS  Master/Slave    0 master
  //             .      SOD Slave Output Disabled 0 only relevant in slave mode
  //         ––––       reserved
  SSPCPSR = 148; // Entre 2 y 254 y PAR,   400000 bps // SD_lento

  // f_BIT = PCLK/(SSPCPSR*(SCR+1))
  // f_BIT = 4*14745600 / (148*(0+1)) =   398529.7 Hz // Modo lento
  // f_BIT = 4*14745600 / (  4*(0+1)) = 14745600.0 Hz // Modo rÃąpido

  SSPCR1 = 0b00000010; // enable SSP
}

void SSP_stop()
{
  volatile int r;
  while(SSPSR & (1<<2)) r = SSPDR; // VacÃ■o la FIFO
  SD_CS_H();
  SSPCR1 = 0b00000000;
  //SSPCPSR = 30; // Entre 2 y 254 y PAR
  //SSPCPSR = 44; // Entre 2 y 254 y PAR
  SSPCPSR = 148; // Entre 2 y 254 y PAR
  SSPCR1 = 0b00000010;
}

inline unsigned int SSP_byte(unsigned int d)
{
 SSPDR = d;
 while(SSPSR&(1<<4)); // Busy
 return SSPDR;
}
#endif
```

lib/ssd1963.h

```c
#pragma once

#include "system.h"

#define TFT_CS (1<<12)
#define TFT_WR (1<<11)
#define TFT_RD (1<<15)
#define TFT_RS (1<<13)

#define TFT_CS_H() {FIOSET=TFT_CS;}
#define TFT_WR_H() {FIOSET=TFT_WR;}
#define TFT_RD_H() {FIOSET=TFT_RD;}
#define TFT_RS_H() {FIOSET=TFT_RS;}

#define TFT_CS_L() {FIOCLR=TFT_CS;}
#define TFT_WR_L() {FIOCLR=TFT_WR;}
#define TFT_RD_L() {FIOCLR=TFT_RD;}
#define TFT_RS_L() {FIOCLR=TFT_RS;}

#define TFT_DATABUS_INPUT  FIODIR = (FIODIR & 0x0000FFFF)
#define TFT_DATABUS_OUTPUT FIODIR = (FIODIR | 0xFFFF0000)

#define TFT_DATA_SET(data) FIOCLR = 0xFFFF0000; FIOSET = (data<<16)


#define DISP_HOR_RESOLUTION 480
#define DISP_VER_RESOLUTION 272

struct _current_font
{
 unsigned char width;
 unsigned char height;
 unsigned char firstC;
 unsigned char n_char;
 unsigned char font[];
};


void writeCmd(unsigned short cmd);
void writeData(unsigned short data);
void TFTInit(void);
void setXY(int x1, int y1, int x2, int y2);
void clrXY();
void clrBox(int x1, int y1, int x2, int y2);

void drawPixel(int x, int y);
void drawCPixel(int x, int y, int color);

void drawHLine(int x, int y, int length);
void drawVLine(int x, int y, int length);
void drawHCLine(int x, int y, int length, int color);
void drawVCLine(int x, int y, int length, int color);

void drawBox(int x1, int y1, int x2, int y2);
void drawCBox(int x1, int y1, int x2, int y2, int color);
void fillBox(int x1, int y1, int x2, int y2);
void fillCBox(int x1, int y1, int x2, int y2, int color);

void drawLine(int x1, int y1, int x2, int y2);
void drawCLine(int x1, int y1, int x2, int y2, int color);
void drawCircle(int x, int y, int radius);
void fillCircle(int x, int y, int radius);

void printChar(char character, int xpos, int ypos);
void printString(char *st, int xpos, int ypos);

void printCChar(char character, int xpos, int ypos, int fgc, int bgc);
void printCString(char *st, int xpos, int ypos, int fgc, int bgc);

void printCharF(char character, int xpos, int ypos, int font);
void printStringF(char *st, int xpos, int ypos);

void buttonBox(int x1, int y1, int x2, int y2, int boxColor, int lineColor);
void drawButton(char *st, int x, int y, int length);
void drawBButton(char *st, int x, int y);

void drawTButton(Button button);
void pressTButton(Button button);

void emptyCMBox(int x, int y);
void checkMark(int x, int y);

void calibrationPoint(int x, int y);

void clrScr(void);

void drawBitmap(int x, int y, unsigned short *bitmap);

void drawInputNumber(InputValue box);



//--------------------------------//
//-------- COMMAND TABLE ---------//
//--------------------------------//

#define nop                           (0x00)
#define soft_reset                    (0x01)
#define get_power_mode                (0x0A)
#define get_address_mode              (0x0B)
#define get_pixel_format              (0x0C)
#define get_display_mode              (0x0D)
#define get_signal_mode               (0x0E)
#define enter_sleep_mode              (0x10)
#define exit_sleep_mode               (0x11)
#define enter_partial_mode            (0x12)
#define enter_normal_mode             (0x13)
#define exit_invert_mode              (0x20)
#define enter_invert_mode             (0x21)
#define set_gamma_curve               (0x26)
#define set_display_off               (0x28)
#define set_display_on                (0x29)
```

```
#define set_column_address             (0x2A)
#define set_page_address               (0x2B)
#define write_memory_start             (0x2C)
#define read_memory_start              (0x2E)
#define set_partial_area               (0x30)
#define set_scroll_area                (0x33)
#define set_tear_off                   (0x34)
#define set_tear_on                    (0x35)
#define set_address_mode               (0x36)
#define set_scroll_start               (0x37)
#define exit_idle_mode                 (0x38)
#define enter_idle_mode                (0x39)
#define set_pixel_format               (0x3A)
#define write_memory_continue          (0x3C)
#define read_memory_continue           (0x3E)
#define set_tear_scanline              (0x44)
#define get_scanline                   (0x45)
#define read_ddb                       (0xA1)
#define set_lcd_mode_                  (0xB0)
#define get_lcd_mode                   (0xB1)
#define set_hori_period                (0xB4)
#define get_hori_period                (0xB5)
#define set_vert_period                (0xB6)
#define get_vert_period                (0xB7)
#define set_gpio_conf                  (0xB8)
#define get_gpio_conf                  (0xB9)
#define set_gpio_value                 (0xBA)
#define get_gpio_status                (0xBB)
#define set_post_proc                  (0xBC)
#define get_post_proc                  (0xBD)
#define set_pwm_conf                   (0xBE)
#define get_pwm_conf                   (0xBF)
#define set_lcd_gen0                   (0xC0)
#define get_lcd_gen0                   (0xC1)
#define set_lcd_gen1                   (0xC2)
#define get_lcd_gen1                   (0xC3)
#define set_lcd_gen2                   (0xC4)
#define get_lcd_gen2                   (0xC5)
#define set_lcd_gen3                   (0xC6)
#define get_lcd_gen3                   (0xC7)
#define set_gpio0_rop                  (0xC8)
#define get_gpio0_rop                  (0xC9)
#define set_gpio1_rop                  (0xCA)
#define get_gpio1_rop                  (0xCB)
#define set_gpio2_rop                  (0xCC)
#define get_gpio2_rop                  (0xCD)
#define set_gpio3_rop                  (0xCE)
#define get_gpio3_rop                  (0xCF)
#define set_dbc_conf                   (0xD0)
#define get_dbc_conf                   (0xD1)
#define set_dbc_th                     (0xD4)
#define get_dbc_th                     (0xD5)
#define set_pll                        (0xE0)
#define set_pll_mn                     (0xE2)
#define get_pll_mn                     (0xE3)
#define get_pll_status                 (0xE4)
#define set_deep_sleep                 (0xE5)
```

```
#define set_lshift_freq                (0xE6)
#define get_lshift_freq                (0xE7)
#define set_pixel_data_interface       (0xF0)
#define get_pixel_data_interface       (0xF1)
```

lib/ssd1963.c

```c
#include "ssd1963.h"

struct _current_font * CurrentFont;


/**
 * @brief Writes a command to the display controller
 *
 * @param cmd is the command that will be written
 */
void writeCmd( unsigned short cmd )
{
        TFT_RS_L();
        TFT_DATA_SET( cmd );
        TFT_WR_L();
        TFT_WR_H();
}

/**
 * @brief Writes data to the display controller
 *
 * @param data is the data that will be written
 */
void writeData( unsigned short data )
{
        TFT_RS_H();
        TFT_DATA_SET( data );
        TFT_WR_L();
        TFT_WR_H();
}

/**
 * @brief Initializes the TFT screen
 *
 */
void TFTInit(void)
{
        TFT_CS_L();

        writeCmd(0xE2);         //PLL multiplier, set PLL clock to 120M
        writeData(30);      //N=0x36 for 6.5M, 0x23 for 10M crystal
        writeData(3);
        writeData(0x54);

        writeCmd(0x01);         // software reset
        _delay_ms(100);

        writeCmd(0xE0);         // PLL enable
        writeData(0x01);
        _delay_ms(10);

        writeCmd(0xE0);
        writeData(0x03);
        _delay_ms(10);

        _delay_ms(100);

        writeCmd(0xE6);         //PLL setting for PCLK, depends on resolution
        writeData(0x01);
        writeData(0x1F);
        writeData(0xFF);

        writeCmd(0xB0);         //LCD SPECIFICATION
        writeData(0x20);
        writeData(0x00);
        writeData(0x01);                //Set HDP       479
        writeData(0xDF);
        writeData(0x01);                //Set VDP       271
        writeData(0x0F);
        writeData(0x00);

        writeCmd(0xF0);         //pixel data interface
        writeData(0x03);

        writeCmd(0x3A);         //pixel format
        writeData(0x50);        // RGB565

        writeCmd(0xB4);         //HSYNC
        writeData(0x02);                //Set HT        531
        writeData(0x13);
        writeData(0x00);                //Set HPS       8
        writeData(0x08);
        writeData(0x2B);                //Set HPW       43
        writeData(0x00);                //Set LPS       2
        writeData(0x02);
        writeData(0x00);

        writeCmd(0xB6);         //VSYNC
        writeData(0x01);                //Set VT        288
        writeData(0x20);
        writeData(0x00);                //Set VPS       4
        writeData(0x04);
        writeData(0x0c);                //Set VPW       12
        writeData(0x00);                //Set FPS       2
        writeData(0x02);

        _delay_ms(1);

        writeCmd(0x29);         //display on
}

/**
 * @brief Sets a window in which pixels can be set a particular color
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 */
void setXY(int x1, int y1, int x2, int y2)
{
        writeCmd(0x2a);
        writeData(x1>>8);
        writeData(x1);
        writeData(x2>>8);
        writeData(x2);
        writeCmd(0x2b);
        writeData(y1>>8);
        writeData(y1);
        writeData(y2>>8);
        writeData(y2);
        writeCmd(0x2c);
}

/**
 * @brief Configures the window in which pixels can be modified to be the whole
   screen
 *
 */
void clrXY()
{
        setXY(0, 0, DISP_HOR_RESOLUTION, DISP_VER_RESOLUTION);
}


/**
 * @brief Draws a pixel at a provided coordinate in white
 *
 * @param x: x coordinate
 * @param y: y coordinate
 */
void drawPixel(int x, int y)
{
    setXY(x, y, x, y);
    writeData(0xFFFF);
}

/**
 * @brief Draws a pixel at a provided coordinate in a provided color
 *
 * @param x: x coordinate
 * @param y: y coordinate
 * @param color in which the pixel will be drawn. This color should be a 16-bit
   number. We use the RGB565 color model.
 */
void drawCPixel(int x, int y, int color)
{
    setXY(x, y, x, y);
    writeData(color);
}

/**
 * @brief Draws a one pixel wide horizontal line, in white
 *
 * @param x: x coordinate
 * @param y: y coordinate
 * @param length of the line
 */
void drawHLine(int x, int y, int length)
```

```
{
        int i = 0;

        if (length<0)
        {
                length = –length;
                x –= length;
        }

    for(i = 0; i < length; i++)
        {
        drawPixel((x+i), y);
    }

        clrXY();
}

/**
 * @brief Draws a one pixel wide vertical line, in white
 *
 * @param x: x coordinate
 * @param y: y coordinate
 * @param length of the line
 */
void drawVLine(int x, int y, int length)
{
        int i = 0;

        if (length<0)
        {
                length = –length;
                y –= length;
        }

    for(i = 0; i < length; i++)
        {
        drawPixel(x, (y+i));
    }

        clrXY();
}

/**
 * @brief Draws a one pixel wide horizontal line, in a provided color
 *
 * @param x: x coordinate
 * @param y: y coordinate
 * @param length of the line
 * @param color in which the line will be drawn. This color should be a 16–bit
    number. We use the RGB565 color model.
 */
void drawHCLine(int x, int y, int length, int color)
{
        int i = 0;

        if (length<0)
        {
                length = –length;
                x –= length;
        }

    for(i = 0; i < length; i++)
        {
        drawCPixel((x+i), y, color);
    }

        clrXY();
}

/**
 * @brief Draws a one pixel wide vertical line, in a provided color
 *
 * @param x: x coordinate
 * @param y: y coordinate
 * @param length of the line
 * @param color in which the line will be drawn. This color should be a 16–bit
    number. We use the RGB565 color model.
 */
void drawVCLine(int x, int y, int length, int color)
{
        int i = 0;

        if (length<0)
        {
                length = –length;
                y –= length;
        }

    for(i = 0; i < length; i++)
        {
        drawCPixel(x, (y+i), color);
    }

        clrXY();
}

/**
 * @brief Draws only the outline of a box, in white
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 */
void drawBox(int x1, int y1, int x2, int y2)
{
        int xleft, xright, yleft, yright, xlen, ylen = 0;

        if(x2 < x1){
                xleft = x2;
                xright = x1;
                xlen = (x1–x2);
        } else {
                xleft = x1;
                xright = x2;
                xlen = (x2-x1);
        }

        if(y2 < y1){
                yleft = y2;
                yright = y1;
                ylen = (y1-y2);
        } else {
                yleft = y1;
                yright = y2;
                ylen = (y2-y1);
        }

        drawHLine(xleft, yleft, xlen);
        drawHLine(xleft, yright, (xlen + 1));
    drawVLine(xleft, yleft, ylen);
    drawVLine(xright, yleft, (ylen + 1));

        clrXY();
}

/**
 * @brief Draws only the outline of a box, in a provided color
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 * @param color in which the outline will be drawn. This color should be a 16-bit
    number. We use the RGB565 color model.
 */
void drawCBox(int x1, int y1, int x2, int y2, int color)
{
        int xleft, xright, yleft, yright, xlen, ylen = 0;

        if(x2 < x1){
                xleft = x2;
                xright = x1;
                xlen = (x1-x2);
        } else {
                xleft = x1;
                xright = x2;
                xlen = (x2-x1);
        }

        if(y2 < y1){
                yleft = y2;
                yright = y1;
                ylen = (y1-y2);
        } else {
                yleft = y1;
                yright = y2;
                ylen = (y2-y1);
        }

        drawHCLine(xleft, yleft, xlen, color);
        drawHCLine(xleft, yright, (xlen + 1), color);
    drawVCLine(xleft, yleft, ylen, color);
    drawVCLine(xright, yleft, (ylen + 1), color);

        clrXY();
}

/**
 * @brief Draws and fills a box in white
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 */
void fillBox(int x1, int y1, int x2, int y2)
{
        setXY(x1, y1, x2, y2);

        int i = 0;
        for (i = 0; i <= ((x2-x1+1)*(y2-y1+1)); i++){
                writeData( 0xFFFF );
        }

        clrXY();
}

/**
 * @brief Draws and fills a box in a provided color
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 * @param color in which the box will be drawn. This color should be a 16-bit number
    . We use the RGB565 color model.
 */
void fillCBox(int x1, int y1, int x2, int y2, int color)
{
        setXY(x1, y1, x2, y2);

        int i = 0;
        for (i = 0; i <= ((x2-x1+1)*(y2-y1+1)); i++){
                writeData( color );
        }

        clrXY();
}

/**
 * @brief Draws a one pixel wide line (that can be diagonal) in white
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 */
```

```
void drawLine(int x1, int y1, int x2, int y2)
{
        if (y1==y2)
                drawHLine(x1, y1, x2-x1);
        else if (x1==x2)
                drawVLine(x1, y1, y2-y1);
        else
        {
                unsigned int    dx = (x2 > x1 ? x2 - x1 : x1 - x2);
                short                   xstep =  x2 > x1 ? 1 : -1;
                unsigned int    dy = (y2 > y1 ? y2 - y1 : y1 - y2);
                short                   ystep =  y2 > y1 ? 1 : -1;
                int                             col = x1, row = y1;

                if (dx < dy)
                {
                        int t = - (dy >> 1);
                        while (1)
                        {
                                drawPixel(col, row);
                                if (row == y2)
                                        return;
                                row += ystep;
                                t += dx;
                                if (t >= 0)
                                {
                                        col += xstep;
                                        t   -= dy;
                                }
                        }
                }
                else
                {
                        int t = - (dx >> 1);
                        while (1)
                        {
                                drawPixel(col, row);
                                if (col == x2)
                                        return;
                                col += xstep;
                                t += dy;
                                if (t >= 0)
                                {
                                        row += ystep;
                                        t   -= dx;
                                }
                        }
                }
        }
        clrXY();
}

/**
 * @brief Draws a one pixel wide line (that can be diagonal) in a provided color
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 * @param color in which the line will be drawn. This color should be a 16-bit number. We use the RGB565 color model.
 */
void drawCLine(int x1, int y1, int x2, int y2, int color)
{
        if (y1==y2)
                drawHCLine(x1, y1, x2-x1, color);
        else if (x1==x2)
                drawVCLine(x1, y1, y2-y1, color);
        else
        {
                unsigned int    dx = (x2 > x1 ? x2 - x1 : x1 - x2);
                short                   xstep =  x2 > x1 ? 1 : -1;
                unsigned int    dy = (y2 > y1 ? y2 - y1 : y1 - y2);
                short                   ystep =  y2 > y1 ? 1 : -1;
                int                             col = x1, row = y1;

                if (dx < dy)
                {
                        int t = - (dy >> 1);
                        while (1)
                        {
                                drawCPixel(col, row, color);
                                if (row == y2)
                                        return;
                                row += ystep;
                                t += dx;
                                if (t >= 0)
                                {
                                        col += xstep;
                                        t   -= dy;
                                }
                        }
                }
                else
                {
                        int t = - (dx >> 1);
                        while (1)
                        {
                                drawCPixel(col, row, color);
                                if (col == x2)
                                        return;
                                col += xstep;
                                t += dy;
                                if (t >= 0)
                                {
                                        row += ystep;
                                        t   -= dx;
                                }
                        }
                }

        }
        clrXY();
}
```

```
/**
 * @brief Draws a one pixel wide outline of a circle in white
 *
 * @param x coordinate of the middle point of the circle
 * @param y coordinate of the middle point of the circle
 * @param radius of the circle
 */
void drawCircle(int x, int y, int radius)
{
        int f = 1 – radius;
        int ddF_x = 1;
        int ddF_y = –2 * radius;
        int x1 = 0;
        int y1 = radius;

        drawPixel(x, y + radius);
        drawPixel(x, y – radius);
        drawPixel(x + radius, y);
        drawPixel(x – radius, y);


        while(x1 < y1)
        {
                if(f >= 0)
                {
                        y1––;
                        ddF_y += 2;
                        f += ddF_y;
                }
                x1++;
                ddF_x += 2;
                f += ddF_x;

                drawPixel(x + x1, y + y1);
                drawPixel(x – x1, y + y1);
                drawPixel(x + x1, y – y1);
                drawPixel(x – x1, y – y1);
                drawPixel(x + y1, y + x1);
                drawPixel(x – y1, y + x1);
                drawPixel(x + y1, y – x1);
                drawPixel(x – y1, y – x1);

        }
        clrXY();
}

/**
 * @brief Draws and fills circle in white
 *
 * @param x coordinate of the middle point of the circle
 * @param y coordinate of the middle point of the circle
 * @param radius of the circle
 */
void fillCircle(int x, int y, int radius)
{
        int x1, y1 = 0;
        for(y1=-radius; y1<=0; y1++)
                for(x1=-radius; x1<=0; x1++)
                        if(x1*x1+y1*y1 <= radius*radius)
                        {
                                drawHLine(x+x1, y+y1, 2*(-x1));
                                drawHLine(x+x1, y-y1, 2*(-x1));
                                break;
                        }
}

/**
 * @brief Erase all pixels on the screen. The screen is set to black.
 *
 */
void clrScr(void)
{
        setXY(0, 0, DISP_HOR_RESOLUTION-1, DISP_VER_RESOLUTION-1);

        int h = 0;
        for (h = 0; h < DISP_HOR_RESOLUTION* DISP_VER_RESOLUTION; h++){
                writeData( COLOR(0,0,0) );
        }
}

/**
 * @brief Erase only a window of the screen. Pixels within the window are set to
   black.
 *
 * @param x1 top left x coordinate
 * @param y1 top left y coordinate
 * @param x2 bottom right x coordinate
 * @param y2 bottom right y coordinate
 */
void clrBox(int x1, int y1, int x2, int y2)
{
        setXY(x1, y1, x2-1, y2-1);

        int h = 0;
        for (h = 0; h < x2 * y2; h++){
                writeData( COLOR(0,0,0) );
        }
}

/**
 * @brief Prints out a string on the LCD screen
 *
 * @param st is a provided string
 * @param xpos is the top left 'x' coordinate of the first character of the string
 * @param ypos is the top left 'y' coordinate of the first character of the string
 */
void printString(char *st, int xpos, int ypos)
{
        int i = 0;
        for(i = 0; i < strlen(st); i++){
                printChar(st[i], (xpos+(i*CurrentFont->width)), (ypos));
        }
```

```
}

/**
 * @brief Prints out a single character to the LCD screen. The character will be
    white while the background around this character will be black.
 *
 * @param character to display
 * @param xpos is the top left 'x' coordinate of the character
 * @param ypos is the top left 'y' coordinate of the character
 */
void printChar(char character, int xpos, int ypos)
{
        int foregroundColor = 0xFFFF;
        int backgroundColor = 0x0000;

        int dir = ( ( ((int)character) - CurrentFont->firstC ) * ( (CurrentFont->
    width/8) * CurrentFont->height ) );
        int y, i = 0;

        setXY(xpos, ypos, xpos+ CurrentFont->width-1, ypos+CurrentFont->height-1);
        for(y = 0; y < (CurrentFont->height*CurrentFont->width/8); y++){

                for(i = 0; i < 8; i++){
                        if( (CurrentFont->font[dir + y]) & (1 << (7-i)) ){
                                writeData(foregroundColor);
                        } else {
                                writeData(backgroundColor);
                        }
                }
        }
}

/**
 * @brief Prints out a single character to the LCD screen. The character will be the
     provided 'foreground color' while the color around this character will be the
    provided 'background color'.
 *
 * @param character to display
 * @param xpos is the top left 'x' coordinate of the character
 * @param ypos is the top left 'y' coordinate of the character
 * @param fgc is the color you want the character to have
 * @param bgc is the color you want the pixels around the character to have
 */
void printCChar(char character, int xpos, int ypos, int fgc, int bgc)
{
        int foregroundColor = (fgc);
        int backgroundColor = (bgc);

        int dir = ( ( ((int)character) - CurrentFont->firstC ) * ( (CurrentFont->
    width/8) * CurrentFont->height ) );
        int y, i = 0;

        setXY(xpos, ypos, xpos+ CurrentFont->width-1, ypos+CurrentFont->height-1);
        for(y = 0; y < (CurrentFont->height*CurrentFont->width/8); y++){

                for(i = 0; i < 8; i++){
                        if( (CurrentFont->font[dir + y]) & (1 << (7-i)) ){
                                writeData(foregroundColor);
                        } else {
                                writeData(backgroundColor);
                        }
                }
        }
}

/**
 * @brief Prints out a string in the colors of your choise.
 *
 * @param st is the string that will be displayed
 * @param xpos is the top left 'x' coordinate of the first character of the string
 * @param ypos is the top left 'y' coordinate of the first character of the string
 * @param fgc is the color you want the text to have
 * @param bgc is the color you want the pixels around the character to have
 */
void printCString(char *st, int xpos, int ypos, int fgc, int bgc)
{
        int i = 0;
        for(i = 0; i < strlen(st); i++){
                printCChar(st[i], (xpos+(i*CurrentFont->width)), (ypos), fgc, bgc);
        }
}

/**
 * @brief Is a template that I made, draws a box with a shadow effect.
 *
 * @param x1 top left x coordinateof the box
 * @param y1 top left y coordinateof the box
 * @param x2 bottom right x coordinateof the box (without the shadow)
 * @param y2 bottom right y coordinateof the box (without the shadow)
 * @param boxColor is the color the button will have
 * @param lineColor is the color of the shadow trail
 */
void buttonBox(int x1, int y1, int x2, int y2, int boxColor, int lineColor)
{
        fillCBox(x1, y1, x2, y2, boxColor);
        drawHCLine(x1+2, y2+1, x2-x1, lineColor);
        drawHCLine(x1+2, y2+2, x2-x1, lineColor);
        drawVCLine(x2+1, y1+2, y2-y1, lineColor);
        drawVCLine(x2+2, y1+2, y2-y1+1, lineColor);
}

/**
 * @brief Draws a button, following my template. Puts text inside of the button.
    This button will be sized for the small font
 *
 * @param st is the text you want to be inside of the button
 * @param x is the top left 'x' coordinate of the BOX
 * @param y is the top left 'y' coordinate of the BOX
 * @param length the button should have
 */
void drawButton(char *st, int x, int y, int length)
{
        int boxColor = COLOR3;
        int lineColor = COLOR4;
        int txtColor = COLOR1;

        int padding = 3;
        int SF_HEIGHT = 12;

        buttonBox(x, y, (x+length), (y+(2*padding)+SF_HEIGHT), boxColor, lineColor);
        printCString(st, (x+padding), (y+padding), txtColor, boxColor);
}

void drawBButton(char *st, int x, int y)
{
        int boxColor = COLOR3;
        int lineColor = COLOR4;
        int txtColor = COLOR1;

        int padding = 3;
        int BF_HEIGHT = 16;
        int BF_WIDTH = 16;

        int st_len = (strlen(st)*BF_WIDTH);
        int x_mid = (x + (st_len/2));
        int y_mid = (y + (BF_HEIGHT/2) + padding);
        int x_st_mid = (st_len/2);


        buttonBox((x_mid-(st_len/2)-padding), (y_mid-(BF_HEIGHT/2)-padding),
                                (x_mid+(st_len/2)+padding), (y_mid+(BF_HEIGHT/2)+
    padding),
                                boxColor, lineColor);
        printCString(st, (x_mid-x_st_mid), (y+padding), txtColor, boxColor);
}

/**
 * @brief Draws a button that is in its non-pressed state
 *
 * @param button is the button that should be drawn. This datatype contains all
    information the buttn has.
 */
void drawTButton(Button button)
{
        int padding = 3;

        buttonBox       (
                                (button.origin_x - padding),
                                (button.origin_y - padding),
                                ((button.origin_x + button.size_x) + padding),
                                ((button.origin_y + button.size_y) + padding),
                                button.box_color_np,
                                button.shade_color_np
                                );
        int n_charac = strlen(button.text);
        int x_to_print = ( button.origin_x + ((button.size_x/2)  - ((n_charac * 8) )
    ));

        printCString(button.text, x_to_print, button.origin_y, button.text_color_np,
     button.box_color_np);
}
```

```
/**
 * @brief Draws a button that is in its pressed state. The button is basically the
    same but inverts the colors of the normal button.
 *
 * @param button is the button that should be drawn. This datatype contains all
    information the buttn has.
 */
void pressTButton(Button button)
{
        int padding = 3;

        buttonBox       (
                                (button.origin_x - padding),
                                (button.origin_y - padding),
                                ((button.origin_x + button.size_x) + padding),
                                ((button.origin_y + button.size_y) + padding),
                                button.box_color_p,
                                button.shade_color_p
                                );
        int n_charac = strlen(button.text);
        int x_to_print = ( button.origin_x + ((button.size_x/2)  - ((n_charac * 8) )
    ));

        printCString(button.text, x_to_print, button.origin_y, button.text_color_p,
    button.box_color_p);
}


void pressBButton(char *st, int x, int y)
{
        int boxColor = ~COLOR3;
        int lineColor = ~COLOR4;
        int txtColor = ~COLOR1;

        int padding = 3;
        int BF_HEIGHT = 16;
        int BF_WIDTH = 16;

        int st_len = (strlen(st)*BF_WIDTH);
        int x_mid = (x + (st_len/2));
        int y_mid = (y + (BF_HEIGHT/2) + padding);
        int x_st_mid = (st_len/2);


        buttonBox((x_mid-(st_len/2)-padding), (y_mid-(BF_HEIGHT/2)-padding),
                                (x_mid+(st_len/2)+padding), (y_mid+(BF_HEIGHT/2)+
    padding),
                                boxColor, lineColor);
        printCString(st, (x_mid-x_st_mid), (y+padding), txtColor, boxColor);
}

void emptyCMBox(int x, int y)
{
        fillCBox(x+2, y+2, x+14, y+14, COLOR_WHITE);
        drawCBox(x+2, y+2, x+14, y+14, COLOR51);
}
```

```

void checkMark(int x, int y)
{
        // 72
        drawCLine((x+4), (y+8), (x+6), (y+10), COLOR72);
        drawCLine((x+7), (y+10), (x+12), (y+5), COLOR72);
        // 107
        drawCLine((x+4), (y+9), (x+6), (y+11), COLOR107);
        drawCLine((x+7), (y+11), (x+12), (y+6), COLOR107);
        // 226
        drawCLine((x+5), (y+8), (x+6), (y+9), COLOR226);
        drawCLine((x+7), (y+9), (x+11), (y+5), COLOR226);
}

// Note that background will need to be set BLACK for the calibration
// these points are beeing drawn in WHITE
void calibrationPoint(int x, int y)
{
        drawPixel(x, y);
        drawCircle(x, y, 1);
        drawCircle(x, y, 10);
}

void drawBitmap(int x, int y, unsigned short *bitmap)
{
        int i;

    short g_width = (bitmap[0]);
    short g_height = (bitmap[1]);
        short g_size = (bitmap[2]);

    setXY(x, y, (x+g_width-1), (y+g_height-1));
    for (i=0; i<g_size; i++)
    {
        writeData((int)(bitmap[(i+3)]));
    }

        clrXY();
}


void drawInputNumber(InputValue box)
{
        char value[5];
        itoa(box.value, value, 10);
        unsigned short write_color = COLOR_BLACK;

        fillCBox(box.x, box.y, box.x+(box.length*16), box.y+16, COLOR_WHITE);

        if (box.state == READY) write_color = COLOR_BLUE;
        else write_color = COLOR_BLACK;

        if (box.value <10)                      printCString(value, ( box.x+((box.
    length-1)*16) ), box.y, write_color, COLOR_WHITE);
        else if (box.value <100)        printCString(value, ( box.x+((box.length-2)
    *16) ), box.y, write_color, COLOR_WHITE);
        else if (box.value <1000)       printCString(value, ( box.x+((box.length-3)
    *16) ), box.y, write_color, COLOR_WHITE);
        else if (box.value <10000)      printCString(value, ( box.x+((box.length-4)
    *16) ), box.y, write_color, COLOR_WHITE);
        else                                        printCString(value, ( box.x
    +((box.length-5)*16) ), box.y, write_color, COLOR_WHITE);
}
```

lib/system.h

```c
#pragma once


void _puts(char *);
void _printf(char *,...);
extern void (*_vputch)(int);
extern int (*_vgetch)();
#define _putch(d) ((*_vputch)(d))
#define _getch() ((*_vgetch)())
int     _gets(unsigned char *,int);
void _U0putch(unsigned char);
unsigned char _U0getch();
void _delay_loop(unsigned int);

extern void EINT1_On();
extern void PLL_On();


/**********************************************************
                  Header files
 *********************************************************/
#define FOSC    14745600        // Crystal frequency
#define MSEL    4               // PLL multiplier
#define CCLK    (FOSC*MSEL)         // CPU clock
#define PCKDIV  1               // APB divider
#define PCLK    (CCLK/PCKDIV)   // Peripheral clock

// delays
#define _delay_us(n) _delay_loop(CCLK/400*n/10000-1)
#define _delay_ms(n) _delay_loop(CCLK/4000*n-1)

#include "types.h"          // should be included as one of the first things
#include "integer.h"
#include "vardec.h"
#include "initialization.h"

#include "calibrate.h"
#include "cncroutines.h"
#include "lpc2106.h"
#include "mcp23s17.h"
#include "timerinterrupt.h"
#include "pff.h"
#include "execute_file.h"
#include "screens.h"
#include "spi.h"
#include "ssd1963.h"
#include "touchcontrol.h"
#include "xpt2046.h"
#include "uart1.h"
#include "display_manager.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
#ifndef REAL_SYSTEM
#undef FIODIR
#undef FIOPIN
#undef FIOSET
#undef FIOCLR

#define FIODIR IODIR
#define FIOPIN IOPIN
#define FIOSET IOSET
#define FIOCLR IOCLR
#endif


#define COLOR(r,g,b) ( (unsigned short) ((r >> 3)<<11) | ((g >> 2)<<5) | (b >> 3) )

#define COLOR1 0xFFFF
#define COLOR_WHITE 0xFFFF
#define COLOR2 COLOR(241, 90, 34)
#define COLOR3 COLOR(0xCF,0x00,0x0F)
#define COLOR4 COLOR(0x96,0x36,0x94)
#define COLOR5 0x0000
#define COLOR_BLACK 0x0000
#define COLOR_GREY  COLOR(213, 215, 216)
#define COLOR_DGREY COLOR(147, 149, 150)
#define COLOR_BLUE  COLOR(0, 0, 255)
#define COLOR_RED   COLOR(255, 0, 0)
#define COLOR_GREY2   COLOR(0xC0, 0xC0, 0xC0)
#define COLOR_GREY3   COLOR(0xA0, 0xA0, 0xA0)


#define COLOR51     COLOR(51, 51, 51)
#define COLOR72     COLOR(72, 72, 72)
#define COLOR107    COLOR(107, 107, 107)
#define COLOR226    COLOR(226, 226, 226)
#define COLOR246    COLOR(246, 246, 246)

#define COLOREX1    COLOR(0, 204, 102)
#define COLOREX2    COLOR(0, 102, 153)
```

lib/timerinterrupt.h

```c
#pragma once
#include "system.h"

#define TIMER0  (4)
#define TIMER1  (5)

void enableTimer(char timer);
void resetTimer(char timer);
void irqSettingsTimer(char timer, int value);
void setTimer0Delay(unsigned char delay);

void stepScaleTimer(char timer, int time);

void clearIntFlag(char timer);
```

lib/timerinterrupt.c

```c
#include "timerinterrupt.h"


void enableTimer(char timer)
{
    switch (timer)
    {
    case 0:
        T0TCR = 1; // enable timer to process commands
        break;
    case 1:
        T1TCR = 1;
        break;
    }
}


void resetTimer(char timer)
{
    switch (timer)
    {
    case 0:
        T0TCR = 2;
        break;
    case 1:
        T1TCR = 2;
        break;
    }
}


/**
 * @brief Configure settings for Timer0
 *
 * @param timer is the timer you want to configure (0–3)
 * @param value that is set to T0MCR register (Match Control Register)
 * Bit 0:
 * '1' –> Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.
 * '0' –> This interrupt is disabled.
 *
 * Bit 1:
 * '1' –> Reset on MR0: the TC will be reset if MR0 matches it.
 * '0' –> Feature disabled.
 *
 * Bit 2:
 * '1' –> Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.
 * '0' –> Feature disabled.
 */
void irqSettingsTimer(char timer, int value)
{
    switch (timer)
    {
    case 0:
        T0MCR = value;
        break;
    case 1:
        T1MCR = value;
        break;
    }
}


void setTimer0Delay(unsigned char delay)
{
    T0MR0 = delay;
}


/**
 * @brief Configures the duration of one step
 *
 * @param timer is the timer you want to configure (0–3)
 * @param time is the time pet step, measured in Âţs
 */
void stepScaleTimer(char timer, int time)
{
    switch (timer)
    {
    case 0:
        T0PR = ((CCLK/1000000)*time);
        break;
    case 1:
        T1PR = ((CCLK/1000000)*time);
        break;
    }
}

/**
 * @brief Configures the period for the timer
 *
 * @param timer is the timer you want to configure (0–3)
 * @param period is the period for the timer, measured as an amount of steps
    configured in with the 'stepScaleTimer' function
 */
void periodTimer(char timer, int period)
{
    switch (timer)
    {
    case 0:
        T0MR0 = period;
        break;
    case 1:
        T1MR0 = period;
        break;
    }
}


void clearIntFlag(char timer)
{
    switch (timer)
    {
    case 0:
        T0IR = 1;
        break;
    case 1:
        T1IR = 1;
        break;
    }
    VICVectAddr=-1;
}
```

lib/touchcontrol.h

```c
#pragma once

#include "system.h"

#define OTSBSIZE    8
#define SSBSIZE     2
#define RPBSIZE     5
#define RPNSIZE     14
#define QWBSIZE     2
#define TDBSIZE     3
#define SvSetBSIZE  2
#define LFSIZE      3
#define NUMPADSZ    12
#define CONFSZ      1
#define EXECSZ      1
#define StopScSZ    1

#define NUMINPSZ    6
#define CONFINPSZ   2
#define TOOLINPSZ   5

#define EX_FILE_X   303
#define EX_FILE_Y   84

#define TOOLBUFFERSZ    122


extern BOOLEAN appRunning;

extern char lastScreen[];
extern AVAILABILITY saveState;
extern int selectedStep;
extern ScreenControl controlCommand;
extern char fullPath[50];
extern int startCondition;
extern int current_tool;


BOOLEAN areaTouched(int x1, int y1, int x2, int y2, int error_margin);
void buttonHandler(Button *inputButton, int arraySize, int error_margin);

void scanOTS();
void scanRP();
void scanTD();
void scanLF();
void scanConf();
void scanWRNG();
void scanEF();

void initOTS();
void initRP();
void initTD();
void initLF();
void initConf();
void initWRNG(char * wrng);
void initEF();

void runApplication();

void numpadHandler(Numpad * layout, int x_offset, int y_offset);
void inputBoxHandler(InputValue * box, int size);
void fileFieldHandler();
void startButton();
```

lib/touchcontrol.c

```
#include "touchcontrol.h"

// #define CNC_MOVEMENT
// #define TOUCH_DEBUG
#define x_offset    364
#define y_offset    16
#define ERROR_2    2

BOOLEAN appRunning = FALSE;

int default_error = 8;
char lastScreen[20];
int selectedStep;
int num_input = 0;

char ready_button = 0;

char *selectedEntry;
char currentDir[50];
char currentFile[50];
char fullPath[50];

int startCondition = 0;
int current_tool = 1;

BOOLEAN fileSelected = FALSE;

AVAILABILITY saveState = NOT_READY;
AVAILABILITY inputState = NOT_READY;


ScreenControl controlCommand = { "empty", NOT_READY };

Button BUTTON_OTS [OTSBSIZE] = {
    { "Back", 111, 22, 80, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Forw.", 111, 118, 80, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Left", 22, 70, 80, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Right", 201, 70, 80, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Up", 261, 34, 80, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Down", 261, 106, 80, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Exit", 383, 16, 80, 16, COLOREX1, COLOREX2, COLOR1, ~COLOREX1, ~COLOREX2, ~
    COLOR1, NOT_PRESSED, NOT_PRESSED }

};

Checkbox MILPER1_OTS [] = {
    { "␣␣␣1", 33, 185, NOT_PRESSED, 1 },
    { "␣␣10", 33, 227, NOT_PRESSED, 10 },
    { "␣100", 137, 185, NOT_PRESSED, 100 }
```

```
};

Button BUTTON_SS [SSBSIZE] = {
    { "RoutePro", 45, 175, 150, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~
    COLOR1, NOT_PRESSED, NOT_PRESSED },
    { "DrillPro", 296, 175, 150, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~
    COLOR1, NOT_PRESSED, NOT_PRESSED }
};

Button BUTTON_RP [RPBSIZE] = {
    { "Config.", 10, 214, 142, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1
    , NOT_PRESSED, NOT_PRESSED },
    { "Tools", 168, 214, 142, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
     NOT_PRESSED, NOT_PRESSED },
    { "Load", 89, 245, 142, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Offset", 247, 245, 142, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1
    , NOT_PRESSED, NOT_PRESSED },
    { "Start", 326, 214, 142, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
     NOT_PRESSED, NOT_PRESSED }
};

Button BUTTON_TD [TDBSIZE] = {
    { "<–", 50, 245, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "–>", 170, 245, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Save", 367, 245, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED }
};

Button BUTTON_LF [LFSIZE] = {
    { "Quit", 367, 245, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Root", 367, 200, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "Accept", 367, 155, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
     NOT_PRESSED, NOT_PRESSED }
};

Numpad NUMPAD [NUMPADSZ] = {
    { '7', 0, 0, NOT_PRESSED, NOT_PRESSED },
    { '8', 40, 0, NOT_PRESSED, NOT_PRESSED },
    { '9', 80, 0, NOT_PRESSED, NOT_PRESSED },
    { '0', 40, 120, NOT_PRESSED, NOT_PRESSED },
    { '4', 0, 40, NOT_PRESSED, NOT_PRESSED },
    { '5', 40, 40, NOT_PRESSED, NOT_PRESSED },
    { '6', 80, 40, NOT_PRESSED, NOT_PRESSED },
    { '<', 0, 120, NOT_PRESSED, NOT_PRESSED },
    { '1', 0, 80, NOT_PRESSED, NOT_PRESSED },
    { '2', 40, 80, NOT_PRESSED, NOT_PRESSED },
    { '3', 80, 80, NOT_PRESSED, NOT_PRESSED },
    { '#', 80, 120, NOT_PRESSED, NOT_PRESSED }
};

Numpad NUMPAD2 [NUMPADSZ] = {
    { '6', 0, 0, NOT_PRESSED, NOT_PRESSED },
    { '7', 39, 0, NOT_PRESSED, NOT_PRESSED },
    { '8', 80, 0, NOT_PRESSED, NOT_PRESSED },
    { '9', 119, 0, NOT_PRESSED, NOT_PRESSED },
    { '0', 158, 0, NOT_PRESSED, NOT_PRESSED },
    { '<', 199, 0, NOT_PRESSED, NOT_PRESSED },
    { '1', 0, 40, NOT_PRESSED, NOT_PRESSED },
    { '2', 39, 40, NOT_PRESSED, NOT_PRESSED },
    { '3', 80, 40, NOT_PRESSED, NOT_PRESSED },
    { '4', 119, 40, NOT_PRESSED, NOT_PRESSED },
    { '5', 158, 40, NOT_PRESSED, NOT_PRESSED },
    { '#', 199, 40, NOT_PRESSED, NOT_PRESSED }
};

Button BUTTON_CONF [CONFSZ] = {
    { "Save", 367, 245, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED }
};

Button BUTTON_EXEC [EXECSZ] = {
    { "Quit", 367, 245, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED }
};

Button BUTTON_StopSc [StopScSZ] = {
    { "Resume", 193, 194, 95, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
     NOT_PRESSED, NOT_PRESSED }
};

InputValue NumpadInput [NUMINPSZ] = {
    { 0, 3, 39, 31, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // x offset
    { 0, 3, 39, 61, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // y offset

    { 0, 3, 39, 105, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // z board
    { 0, 3, 111, 105, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // z base

    { 0, 3, 111, 31, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // x clip
    { 0, 3, 111, 61, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // y clip
};

InputValue ConfigInput [CONFINPSZ] = {
    { 0, 3, 54, 22, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Stroke limit
    { 0, 3, 54, 52, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Stroke free

    // { 0, 3, 11, 101, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Chord angle
};

InputValue ToolInput1 [TOOLINPSZ] = {
    { 0, 4, 50, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED },  // Depth
    { 0, 2, 180, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Iter
    { 0, 4, 50, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Xfeed
    { 0, 4, 180, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED },// Yfeed
    { 0, 5, 50, 195, NOT_READY, NOT_PRESSED, NOT_PRESSED } // Spindle
};

InputValue ToolInput2 [TOOLINPSZ] = {
    { 0, 4, 50, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED },  // Depth
    { 0, 2, 180, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Iter
    { 0, 4, 50, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Xfeed
    { 0, 4, 180, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED },// Yfeed
    { 0, 5, 50, 195, NOT_READY, NOT_PRESSED, NOT_PRESSED } // Spindle
};

InputValue ToolInput3 [TOOLINPSZ] = {
    { 0, 4, 50, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED },  // Depth
    { 0, 2, 180, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Iter
    { 0, 4, 50, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Xfeed
    { 0, 4, 180, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED },// Yfeed
    { 0, 5, 50, 195, NOT_READY, NOT_PRESSED, NOT_PRESSED } // Spindle
};

InputValue ToolInput4 [TOOLINPSZ] = {
    { 0, 4, 50, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED },  // Depth
    { 0, 2, 180, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Iter
    { 0, 4, 50, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Xfeed
    { 0, 4, 180, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED },// Yfeed
    { 0, 5, 50, 195, NOT_READY, NOT_PRESSED, NOT_PRESSED } // Spindle
};

InputValue ToolInput5 [TOOLINPSZ] = {
    { 0, 4, 50, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED },  // Depth
    { 0, 2, 180, 74, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Iter
    { 0, 4, 50, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED }, // Xfeed
    { 0, 4, 180, 139, NOT_READY, NOT_PRESSED, NOT_PRESSED },// Yfeed
    { 0, 5, 50, 195, NOT_READY, NOT_PRESSED, NOT_PRESSED } // Spindle
};

const int y_1_a = 24;
const int y_2_a = 48;
const int y_3_a = 84;
const int y_4_a = 68;

const int y_1_b = 32;
const int y_2_b = 56;
const int y_3_b = 92;
const int y_4_b = 76;

NumberBox NUMBERS_RP [RPNSIZE] = {
    { "x_scale", 32,        24 },
    { "x_offset", 136,      24 },
    { "x_clip", 256,        24 },
    { "vectors_from", 400,  24 },
    { "y_scale", 32,        48 },
    { "y_offset", 136,      48 },
    { "y_clip", 256,        48 },
    { "vectors_to", 400,    48 },
    { "z_board", 32,    84 },
    { "z_base", 136,    84 },
    { "x_min", 256,         68 },
    { "x_max", 384,         68 },
    { "y_min", 256,     84 },
    { "y_max", 384,     84 }
};

Button BUTTON_QW [QWBSIZE] = {
    { "Yes", 130, 155, 100, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "No", 250, 155, 100, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED }
};

Button BUTTON_SvSet [SvSetBSIZE] = {
    { "Yes", 130, 155, 100, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED },
    { "No", 250, 155, 100, 16, COLOR3, COLOR4, COLOR1, ~COLOR3, ~COLOR4, ~COLOR1,
    NOT_PRESSED, NOT_PRESSED }
};


BOOLEAN areaTouched(int x1, int y1, int x2, int y2, int error_margin)
{
    if ((calibratedX() >= (x1−error_margin))
     && (calibratedY() >= (y1−error_margin))
     && (calibratedX() <= (x2+error_margin))
     && (calibratedY() <= (y2+error_margin)))
    {
        return TRUE;
    }
}

void buttonHandler(Button *inputButton, int arraySize, int error_margin)
{
    int i = 0;

    for (i = 0; i < arraySize; i++)
    {
        if ( (areaTouched(inputButton[i].origin_x,
                          inputButton[i].origin_y,
                          (inputButton[i].origin_x + inputButton[i].size_x),
                          (inputButton[i].origin_y + inputButton[i].size_y),
    error_margin )) == TRUE )
        {
            inputButton[i].current_state = PRESSED;

        } else {
            inputButton[i].current_state = NOT_PRESSED;
        }

        if ((inputButton[i].current_state == PRESSED) && (inputButton[i].
    previous_state == NOT_PRESSED))
        {
            pressTButton(inputButton[i]);
            #ifdef TOUCH_DEBUG
            _printf("You␣are␣pressing␣");
            _printf(inputButton[i].text);
            _printf("\r\n");
            #endif
        }
        if ((inputButton[i].current_state == NOT_PRESSED) && (inputButton[i].
    previous_state == PRESSED))
        {
            drawTButton(inputButton[i]);
            strcpy(controlCommand.command, inputButton[i].text);
            controlCommand.state = READY;

            #ifdef TOUCH_DEBUG
            _printf("You␣have␣just␣released␣%s\r\n", inputButton[i].text);
            _printf("%s␣saved␣in␣controlCommand.command\r\n", controlCommand.command
    );
            #endif

        }
        inputButton[i].previous_state = inputButton[i].current_state;

    _delay_ms(10);          // TODO: UGLY DELAY
    }
}

void initOTS()
{
    getToolData();
    strcpy((lastScreen), "OffsetTeach");
    drawOffsetTeachB();
}


void scanOTS()
{
    int i, j = 0;

    buttonHandler(BUTTON_OTS, OTSBSIZE, default_error);

    for (i = 0; i < (sizeof(MILPER1_OTS) / sizeof(Checkbox)); i++)
    {
        if ( (areaTouched(MILPER1_OTS[i].origin_x,
                MILPER1_OTS[i].origin_y,
                (MILPER1_OTS[i].origin_x + 40),
                (MILPER1_OTS[i].origin_y + 40), default_error )) == TRUE )
        {
            #ifdef TOUCH_DEBUG
            // _printf("BUTTON_OTS iteration: %d | ", i);
            _printf(MILPER1_OTS[i].text); _printf("␣[]␣has␣been␣touched\r\n");
            #endif

            for (j = 0; j < (sizeof(MILPER1_OTS) / sizeof(Checkbox)); j++)
            {
                #ifdef TOUCH_DEBUG
                _printf("Clearing␣button␣[␣"); _printf(MILPER1_OTS[j].text); _printf
    ("␣]\r\n");
                #endif
                emptyCMBox(MILPER1_OTS[j].origin_x, MILPER1_OTS[j].origin_y);
                MILPER1_OTS[j].state = NOT_PRESSED;
            }
            #ifdef TOUCH_DEBUG
            _printf("Checking␣button␣[␣"); _printf(MILPER1_OTS[i].text); _printf("␣
    ]\r\n");
            #endif
            checkMark(MILPER1_OTS[i].origin_x, MILPER1_OTS[i].origin_y);
            MILPER1_OTS[i].state = PRESSED;

            selectedStep = (int)((MILPER1_OTS[i].value * 3937) / 100);
                                                // translate millieters to steps
        }
    _delay_ms(15);          // TODO: UGLY DELAY
    }

    if (controlCommand.state == READY)
    {
        if (strcmp(controlCommand.command, "Exit") == 0)
        {
            controlCommand.state = NOT_READY;
            // scanRP();
            display_manager_set_state(e_disp_route_pro);
        }
    int speed = 300;
        if (strcmp(controlCommand.command, "Left") == 0)   //Move X–
        {
            controlCommand.state = NOT_READY;
            moveAxis(C_XDEC, selectedStep);
        }
        if (strcmp(controlCommand.command, "Right") == 0)   //Move X+
        {
            controlCommand.state = NOT_READY;
            moveAxis(C_XINC, selectedStep);
        }
        if (strcmp(controlCommand.command, "Back") == 0)   //Move Y+
        {
            controlCommand.state = NOT_READY;
            moveAxis(C_YINC, selectedStep);
        }
        if (strcmp(controlCommand.command, "Forw.") == 0)   //Move Y–
        {
            controlCommand.state = NOT_READY;
            moveAxis(C_YDEC, selectedStep);
        }
        if (strcmp(controlCommand.command, "Up") == 0)      //Move Z–
        {
            controlCommand.state = NOT_READY;
            moveAxis(C_ZDEC, selectedStep);
        }
        if (strcmp(controlCommand.command, "Down") == 0)    //Move Z+
        {
            controlCommand.state = NOT_READY;
            moveAxis(C_ZINC, selectedStep);
        }
    }
    updateOffsetDim();
}


void initRP()
{
    getRouteDrill();
    drawRouteProB();
    drawNumPad(NUMPAD, x_offset, y_offset);
    strcpy((lastScreen), "RoutePro");
    startCondition = 0;
}

void scanRP()
{
    if (!fileSelected) {startButton(); startCondition = RPBSIZE-1;} // If no file is selected, display a grey button
    else startCondition = RPBSIZE;

    buttonHandler(BUTTON_RP, startCondition, ERROR_2);
    numpadHandler(NUMPAD, x_offset, y_offset);
    inputBoxHandler(NumpadInput, (NUMINPSZ-2));

    if (controlCommand.state == READY)
    {
        if (strcmp(controlCommand.command, "Config.") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_config);
        }
        if (strcmp(controlCommand.command, "Tools") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_tools);
        }
        if (strcmp(controlCommand.command, "Start") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_execute_file);
        }


        if (strcmp(controlCommand.command, "Load") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_load);
        }
        if (strcmp(controlCommand.command, "Offset") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_offset);
        }

        // Numpad
        num_input = NumpadInput[ready_button].value;
        if (strcmp(controlCommand.command, "1") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+1;

        }
        if (strcmp(controlCommand.command, "2") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+2;
        }
        if (strcmp(controlCommand.command, "3") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+3;
        }
        if (strcmp(controlCommand.command, "4") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+4;
        }
        if (strcmp(controlCommand.command, "5") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+5;
        }
        if (strcmp(controlCommand.command, "6") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+6;
        }
        if (strcmp(controlCommand.command, "7") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+7;
        }
        if (strcmp(controlCommand.command, "8") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+8;
        }
        if (strcmp(controlCommand.command, "9") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+9;
        }
        if (strcmp(controlCommand.command, "0") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10);
        }
        if (strcmp(controlCommand.command, "<") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input > 0) num_input = (num_input/10);
        }

        // Display value
        int i;
        for (i = 0; i < (NUMINPSZ-2); i++)
        {
            NumpadInput[i].state = NOT_READY;
            drawInputNumber(NumpadInput[i]);
        }
        NumpadInput[ready_button].state = READY;
        NumpadInput[ready_button].value = num_input;
        drawInputNumber(NumpadInput[ready_button]);
```

```
    }
}

void initTD()
{
    strcpy((lastScreen), "ToolData");
    int current_tool = 1;
    getToolData();
    drawToolData();
    drawNumPad(NUMPAD, x_offset, y_offset);
}

void scanTD()
{
    int i;
    buttonHandler(BUTTON_TD, TDBSIZE, ERROR_2);
    numpadHandler(NUMPAD, x_offset, y_offset);
    switch (current_tool)
    {
        case 1: inputBoxHandler(ToolInput1, TOOLINPSZ); break;
        case 2: inputBoxHandler(ToolInput2, TOOLINPSZ); break;
        case 3: inputBoxHandler(ToolInput3, TOOLINPSZ); break;
        case 4: inputBoxHandler(ToolInput4, TOOLINPSZ); break;
        case 5: inputBoxHandler(ToolInput1, TOOLINPSZ); break;
    }

    if (controlCommand.state == READY)
    {
        if (strcmp(controlCommand.command, "Save") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_route_pro);
        }
        if (strcmp(controlCommand.command, "<−") == 0)
        {
            controlCommand.state = NOT_READY;
            current_tool−−;
            if(current_tool==0) current_tool = 5;
            drawToolInfo(current_tool);
        }
        if (strcmp(controlCommand.command, "−>") == 0)
        {
            controlCommand.state = NOT_READY;
            current_tool++;
            if(current_tool==6) current_tool = 1;
            drawToolInfo(current_tool);
        }

        // Numpad
        unsigned int num_length = 100;    // numlen > 100
        switch (current_tool)
        {
            case 1: num_input = ToolInput1[ready_button].value; num_length = myPow
    (10, (int)(ToolInput1[ready_button].length−1)); break;
            case 2: num_input = ToolInput2[ready_button].value; num_length = myPow
    (10, (int)(ToolInput2[ready_button].length−1)); break;
            case 3: num_input = ToolInput3[ready_button].value; num_length = myPow
```

```
(10, (int)(ToolInput3[ready_button].length-1)); break;
        case 4: num_input = ToolInput4[ready_button].value; num_length = myPow
(10, (int)(ToolInput4[ready_button].length-1)); break;
        case 5: num_input = ToolInput5[ready_button].value; num_length = myPow
(10, (int)(ToolInput5[ready_button].length-1)); break;
        default:    _printf("Error␣at␣current␣tool\n"); break;
    }

    if (strcmp(controlCommand.command, "1") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+1;

    }
    if (strcmp(controlCommand.command, "2") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+2;
    }
    if (strcmp(controlCommand.command, "3") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+3;
    }
    if (strcmp(controlCommand.command, "4") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+4;
    }
    if (strcmp(controlCommand.command, "5") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+5;
    }
    if (strcmp(controlCommand.command, "6") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+6;
    }
    if (strcmp(controlCommand.command, "7") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+7;
    }
    if (strcmp(controlCommand.command, "8") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+8;
    }
    if (strcmp(controlCommand.command, "9") == 0)
    {
        controlCommand.state = NOT_READY;
        if (num_input < num_length) num_input = (num_input*10)+9;
    }
    if (strcmp(controlCommand.command, "0") == 0)
    {
        controlCommand.state = NOT_READY;
```

```
            if (num_input < num_length) num_input = (num_input*10);
        }
        if (strcmp(controlCommand.command, "<") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input > 0) num_input = (num_input/10);
        }

        // Display value
        switch (current_tool)
        {
            case 1:
                for (i = 0; i < TOOLINPSZ; i++)
                {
                    ToolInput1[i].state = NOT_READY;
                    drawInputNumber(ToolInput1[i]);
                }
                ToolInput1[ready_button].state = READY;
                ToolInput1[ready_button].value = num_input;
                drawInputNumber(ToolInput1[ready_button]);
                setToolData(1, ToolInput1[0].value, ToolInput1[1].value, ToolInput1
    [2].value, ToolInput1[3].value, ToolInput1[4].value);
                break;
            case 2:
                for (i = 0; i < TOOLINPSZ; i++)
                {
                    ToolInput2[i].state = NOT_READY;
                    drawInputNumber(ToolInput2[i]);
                }
                ToolInput2[ready_button].state = READY;
                ToolInput2[ready_button].value = num_input;
                drawInputNumber(ToolInput2[ready_button]);
                setToolData(2, ToolInput2[0].value, ToolInput2[1].value, ToolInput2
    [2].value, ToolInput2[3].value, ToolInput2[4].value);

                // Set tool data
                break;
            case 3:
                for (i = 0; i < TOOLINPSZ; i++)
                {
                    ToolInput3[i].state = NOT_READY;
                    drawInputNumber(ToolInput3[i]);
                }
                ToolInput3[ready_button].state = READY;
                ToolInput3[ready_button].value = num_input;
                drawInputNumber(ToolInput3[ready_button]);
                setToolData(3, ToolInput3[0].value, ToolInput3[1].value, ToolInput3
    [2].value, ToolInput3[3].value, ToolInput3[4].value);
                break;
            case 4:
                for (i = 0; i < TOOLINPSZ; i++)
                {
                    ToolInput4[i].state = NOT_READY;
                    drawInputNumber(ToolInput4[i]);
                }
                ToolInput4[ready_button].state = READY;
                ToolInput4[ready_button].value = num_input;
                drawInputNumber(ToolInput4[ready_button]);
                setToolData(4, ToolInput4[0].value, ToolInput4[1].value, ToolInput4
    [2].value, ToolInput4[3].value, ToolInput4[4].value);
                break;
            case 5:
                for (i = 0; i < TOOLINPSZ; i++)
                {
                    ToolInput5[i].state = NOT_READY;
                    drawInputNumber(ToolInput5[i]);
                }
                ToolInput5[ready_button].state = READY;
                ToolInput5[ready_button].value = num_input;
                drawInputNumber(ToolInput5[ready_button]);
                setToolData(5, ToolInput5[0].value, ToolInput5[1].value, ToolInput5
    [2].value, ToolInput5[3].value, ToolInput5[4].value);
                break;
        }
        fillPayloadWithTool();
    }
}


void initLF()
{
    strcpy((lastScreen), "LoadFile");
    drawLoadFile();
    fullPath[0] = 0;
    currentDir[0] = 0;

    int mounting = SDVolumeInit();
    if(mounting == 2) {printCString("No␣SD␣card␣inserted", 40, 60, COLOR_WHITE,
    COLOR_BLUE);}
    else if(mounting == 1) {printCString("Disk␣read␣error", 40, 60, COLOR_WHITE,
    COLOR_BLUE);}
    else if(mounting == 6) {printCString("There␣is␣no␣FAT␣partition␣on␣the␣drive",
    40, 60, COLOR_WHITE, COLOR_BLUE);}
    else {          // If SD card is found and all is well
        scanFiles(fullPath);
        drawFiles();
    }
}

void scanLF()
{
    int i;

    buttonHandler(BUTTON_LF, LFSIZE, default_error);
    fileFieldHandler();
    if (controlCommand.state == READY)
    {
        if (strcmp(controlCommand.command, "Quit") == 0)
        {
            controlCommand.state = NOT_READY;
            fullPath[0] = 0;
            display_manager_set_state(e_disp_route_pro);
        }
        if (strcmp(controlCommand.command, "Accept") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_route_pro);
        }
        if (strcmp(controlCommand.command, "Root") == 0)
        {
            controlCommand.state = NOT_READY;
            fullPath[0] = 0;
            fileSelected = FALSE;
            display_manager_set_state(e_disp_load);
        }
    }
}



void runApplication()
{
    appRunning = TRUE;

    display_manager_init_state();

    #ifdef CNC_MOVEMENT
    CNCInit();
    drawHomingXY();
    moveHome();
    #endif

    while (1)
    {
        display_manager_process_state();
        if(button_interrupt_flag==1) {clear_button_interrupt_flag();
    display_manager_set_state(e_disp_warning);}
        if(buffer_flag==1) {decode_payload(payload); clear_buffer_flag();}
        if(uart1_isr_flag==1) {clear_uart1_isr_flag(); display_manager_init_state()
    ;}
    }
}


void numpadHandler(Numpad * layout, int x, int y)
{
    int i = 0;
    for (i = 0; i < 12; i++)
    {
        if ( (areaTouched(x+layout[i].origin_x,
                          y+layout[i].origin_y,
                          (x+layout[i].origin_x + 20),
                          (y+layout[i].origin_y + 31), 5 ) == TRUE ) )
        {
            layout[i].current_state = PRESSED;
        } else {
            layout[i].current_state = NOT_PRESSED;
        }

        if ((layout[i].current_state == PRESSED) && (layout[i].previous_state ==
    NOT_PRESSED))
```

```
        {
            pressNumpNumb(layout[i], x, y);
        }
        if ((layout[i].current_state == NOT_PRESSED) && (layout[i].previous_state ==
     PRESSED))
        {
            drawNumpNumb(layout[i], x, y);

            strcpy(controlCommand.command, layout[i].number);
            controlCommand.state = READY;
        }
        layout[i].previous_state = layout[i].current_state;
    _delay_ms(5);
    }
}

void inputBoxHandler(InputValue * box, int size)
{
    int i, j = 0;
    for (i = 0; i < size; i++)
    {
        if ( (areaTouched(box[i].x, box[i].y,
                          box[i].x+box[i].length*16, box[i].y+16,
                           5 ) == TRUE ) )
        {
            box[i].current_state = PRESSED;
        } else {
            box[i].current_state = NOT_PRESSED;
        }

        if ((box[i].current_state == PRESSED) && (box[i].previous_state ==
    NOT_PRESSED))
        {

        }
        if ((box[i].current_state == NOT_PRESSED) && (box[i].previous_state ==
    PRESSED))
        {
            for (j = 0; j < size; j++)
            {
                box[j].state = NOT_READY;
                drawInputNumber(box[j]);
            }

            ready_button = i;
            box[i].state = READY;
            drawInputNumber(box[i]);
        }
        box[i].previous_state = box[i].current_state;
    _delay_ms(8);
    }
}

void initConf()
{
    strcpy((lastScreen), "Config");
    drawConfig();
    drawNumPad(NUMPAD, x_offset, y_offset);
}

void scanConf()
{
    int i;
    buttonHandler(BUTTON_CONF, CONFSZ, default_error);
    numpadHandler(NUMPAD, x_offset, y_offset);
    inputBoxHandler(ConfigInput, CONFINPSZ);
    if (controlCommand.state == READY)
    {
        if (strcmp(controlCommand.command, "Save") == 0)
        {
            controlCommand.state = NOT_READY;
            // scanRP();
            display_manager_set_state(e_disp_route_pro);
        }

        // Numpad
        num_input = ConfigInput[ready_button].value;
        if (strcmp(controlCommand.command, "1") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+1;

        }
        if (strcmp(controlCommand.command, "2") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+2;
        }
        if (strcmp(controlCommand.command, "3") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+3;
        }
        if (strcmp(controlCommand.command, "4") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+4;
        }
        if (strcmp(controlCommand.command, "5") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+5;
        }
        if (strcmp(controlCommand.command, "6") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+6;
        }
        if (strcmp(controlCommand.command, "7") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+7;
        }
        if (strcmp(controlCommand.command, "8") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+8;
        }
        if (strcmp(controlCommand.command, "9") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10)+9;
        }
        if (strcmp(controlCommand.command, "0") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input < 100) num_input = (num_input*10);
        }
        if (strcmp(controlCommand.command, "<") == 0)
        {
            controlCommand.state = NOT_READY;
            if (num_input > 0) num_input = (num_input/10);
        }

        // Display value
        for (i = 0; i < CONFINPSZ; i++)
        {
            ConfigInput[i].state = NOT_READY;
            drawInputNumber(ConfigInput[i]);
        }
        ConfigInput[ready_button].state = READY;
        ConfigInput[ready_button].value = num_input;
        drawInputNumber(ConfigInput[ready_button]);
    }
}



void fileFieldHandler()
{
    int i, y, j;
    int error_margin = 8;

    BUTTON_STATE c, p;
    p = NOT_PRESSED;

    for (i = 0; i < amountoffiles; i++)
    {
        y = yCoords[i];

        if ( (areaTouched(STARTCOORDX, y, STARTCOORDX + 280, y+16, error_margin ) ==
     TRUE))
        {
            c = PRESSED;
        } else {
            c = NOT_PRESSED;
        }

        if ((c == PRESSED) && (p == NOT_PRESSED))   // You are pressing
        {
            for (j = 0; j < amountoffiles; j++)
            {
                fileField(STARTCOORDX, yCoords[j], (char*)&fileAndInf[j], FALSE);
            }
            fileField(STARTCOORDX, y, (char*)&fileAndInf[i], TRUE);

        }
        if ((c == NOT_PRESSED) && (p == PRESSED))   // You have just released
        {
            selectedEntry = (char*)&fileAndInf[i-1];         // -1 because file was selected in previous iteration
            int sL = strlen(selectedEntry);

            if (selectedEntry[sL-1] == '/')  // If entry is a directory
            {
                if (currentDir[0] != 0)       // if current dir is a subdir
                {
                    strcat(currentDir, "/");
                    strcat(currentDir, selectedEntry);
                    sL = strlen(currentDir); currentDir[sL-1] = 0;  // Remove last backslash
                    strcpy(fullPath, currentDir);

                } else                        // Current DIR is not a subDir
                {
                    strcpy(currentDir, selectedEntry);
                    currentDir[sL-1] = 0;
                    strcpy(fullPath, currentDir);
                }
                clearFileBuffer();
                scanFiles(fullPath);
                drawFiles();
                fileSelected = FALSE;
                return;

            } else                           // File
            {
                strcpy(currentFile, selectedEntry);
                strcpy(fullPath, currentDir);
                strcat(fullPath, "/");
                strcat(fullPath, currentFile);
                fileSelected = TRUE;
            }
        }
        p = c;
    _delay_ms(8);
    }
}


void startButton()
{
    int x1 = 326;
    int x2 = 468;
    int y1 = 214;
    int y2 = 230;
    buttonBox ( (x1-3), (y1-3), (x2+3), (y2+3), COLOR_GREY2, COLOR_GREY3 );
    int n_charac = strlen("Start");
    int x_to_print = ( x1 + (((x2-x1)/2) - ((n_charac * 8) )));
    printCString("Start", x_to_print, y1, COLOR_BLACK, COLOR_GREY2);
}


void initWRNG(char * wrng)
{
    drawWrnSc(wrng);
}

void scanWRNG()
{
    buttonHandler(BUTTON_StopSc, StopScSZ, 8);
    if (controlCommand.state == READY)
    {
        if (strcmp(controlCommand.command, "Resume") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(g_previous_display);
        }
    }
}


void initEF()
{
    drawStartS(EX_FILE_X, EX_FILE_Y);
    strcpy((lastScreen), "StartScreen");

    CNC_MotorOn_n(45000);
    processFile(fullPath);
    CNC_MotorOff_n();
    fullPath[0] = 0;
    fileSelected = FALSE;

    moveToXY(11810, 10236);
}

void scanEF()
{
    buttonHandler(BUTTON_EXEC, EXECSZ, default_error);
    if (controlCommand.state == READY)
    {
        if (strcmp(controlCommand.command, "Quit") == 0)
        {
            controlCommand.state = NOT_READY;
            display_manager_set_state(e_disp_route_pro);
        }
    }
    drawPos(EX_FILE_X, EX_FILE_Y);
}
```

lib/types.h

```c
#pragma once

#include "system.h"

typedef enum {PRESSED, NOT_PRESSED} BUTTON_STATE;
typedef enum {READY, NOT_READY} AVAILABILITY;

typedef struct {
    char text[20];
    int origin_x;
    int origin_y;
    int size_x;
    int size_y;
    int box_color_np;
    int shade_color_np;
    int text_color_np;
    int box_color_p;
    int shade_color_p;
    int text_color_p;
    BUTTON_STATE current_state;
    BUTTON_STATE previous_state;
} Button;

typedef struct {
    char number[1];
    int origin_x;
    int origin_y;
    BUTTON_STATE current_state;
    BUTTON_STATE previous_state;
} Numpad;

typedef struct {
    char text[8];
    int origin_x;
    int origin_y;
    BUTTON_STATE state;
    int value;
} Checkbox;

typedef struct {
    char command[20];
    AVAILABILITY state;
} ScreenControl;

typedef struct {
    char name[20];
    int origin_x;
    int origin_y;
} NumberBox;

typedef struct {
    char command[16];
    int spindle;
    int x;
    int y;
} CommandFromFile;

typedef struct {
    int terminator;
    int lastSemicolon;
} CmdProc;

typedef struct {
    unsigned int value;
    int length;
    int x;
    int y;
    AVAILABILITY state;
    BUTTON_STATE current_state;
    BUTTON_STATE previous_state;
} InputValue;

typedef struct {
    unsigned char tool_number;
    int depth;
    int iteration;
    int x_feed;
    int z_feed;
    int spindle;
} ToolData;

typedef struct {
    int x_offset;
    int x_clip;
    int x_vectors;
    int y_offset;
    int y_clip;
    int y_vectors;
    int z_board;
    int z_base;
    int x_min_value;
    int x_max_value;
    int y_min_value;
    int y_max_value;
    int total;
    int done;
} RouteDrill;
```

lib/uart1.h

```c
#pragma once

#include "system.h"
#define SOF 0x24    // Start of frame character
#define ESC 0x5C    // Escape character

#define THR_E       (1<<5)
#define DR          (1<<0)
#define UARTBFRSZ   256

// Will place an integer on the pace where 'INTEG' is called.
#define INTEG(N)    (payload[N]|(payload[N+1]<<8)|(payload[N+2]<<16)|(payload[N
    +3]<<24))

// Used for sending data, will fill buffer with an integer.
// This function will split the integer up into bytes
#define I2B(i, N)   \
    send_payload[N]=i & 0xFF;   \
    send_payload[N+1]=(i>>8) & 0xFF;   \
    send_payload[N+2]=(i>>16) & 0xFF;   \
    send_payload[N+3]=(i>>24) & 0xFF;

#define C2B(c, N)   (send_payload[N]=c & 0xFF)


extern char payload[UARTBFRSZ];
extern char send_payload[UARTBFRSZ];
extern ToolData tool[5];
extern RouteDrill randd;

extern char buffer_flag;
extern char uart1_isr_flag;


void set_buffer_flag();
void clear_buffer_flag();

void set_uart1_isr_flag();
void clear_uart1_isr_flag();

void __attribute__ ((interrupt("IRQ"))) IRQ_UART1_Routine (void);

void UART1_Init();
void UART1_putch(char c);
char UART1_getch();
void UART1_print(char * s);

void UART1_putch_bs(char c);
void sendFrame(char * s, char len);
void stateMachine_Init();

int decode_payload(char * buffer);

void setToolData(unsigned char tool_number, int depth, int iteration, int x_feed,
    int z_feed, int spindle);
```

```
void printToolData(unsigned char tool_number);
void fillPayloadWithTool();

void setRouteDrill(int x_offset, int x_clip, int x_vectors, int y_offset, int y_clip
    , int y_vectors, int z_board, int z_base, int x_min_value, int x_max_value, int
    y_min_value, int y_max_value /*, int total, int done*/);
void fillPayloadWithRandD();

void getToolData();
void getRouteDrill();
```

lib/uart1.c

```c
#include "uart1.h"

char payload[UARTBFRSZ];
volatile char uart1_i, salir, state;
volatile char rawLength = 0;
unsigned int CRC32 = 0;
char send_payload[UARTBFRSZ];

#define TOOLFRAMESZ     122

char buffer_flag = 0;
char uart1_isr_flag = 0;

ToolData tool[5];
RouteDrill randd;


void __attribute__ ((interrupt("IRQ"))) IRQ_UART1_Routine (void) {
    char c;

    while (U1LSR & 1)
    {
        c = U1RBR;
        // _printf("%x ", c);
        switch(state)
        {
            case 0: // Outside frame
                if(c==SOF) {uart1_i=0; state = 1; };
                break;
            case 1: // Inside frame
                if(c==ESC) {state = 2; break; };
                if(c==SOF) {state = 0; salir = 1; break;};
                if(uart1_i == 0) {rawLength = c; uart1_i++;}
                else{payload[(uart1_i++)-1] = c;};
                if (uart1_i >= rawLength+2) {set_buffer_flag(); state = 0;}
                break;
            case 2: // Decoding datum
                if(c==ESC) {payload[(uart1_i++)-1] = ESC; state = 1; break;};
                if(c==SOF) {payload[(uart1_i++)-1] = SOF; state = 1; break;};
                state = 0;
                break;
        }

    }
    set_uart1_isr_flag();
    VICVectAddr=-1;
}

int decode_payload(char * buffer)
{
    int i = 0;
    int k = 0;

    if (buffer[0] == 0xC3)          // Payload contains screen data
    {          // buffer[0]==0xC3? -> after this line 'i' is '1'
        switch (buffer[1])
        {
            case 1:         // Config screen
                break;

            case 2:         // Route–/Drillpro screen
                setRouteDrill(INTEG(2), INTEG(6), INTEG(10), INTEG(14), INTEG(18),
    INTEG(22), INTEG(26), INTEG(30), INTEG(34), INTEG(38), INTEG(42), INTEG(46) /*,
    INTEG(50), INTEG(56)*/);
                getRouteDrill();
                break;

            case 3:         // Tools screen
                setToolData(INTEG(2+(k*24)), INTEG(6+(k*24)), INTEG(10+(k*24)),
    INTEG(14+(k*24)), INTEG(18+(k*24)), INTEG(22+(k*24)));

                k=1;setToolData(INTEG(2+(k*24)), INTEG(6+(k*24)), INTEG(10+(k*24)),
    INTEG(14+(k*24)), INTEG(18+(k*24)), INTEG(22+(k*24)));
                k=2;setToolData(INTEG(2+(k*24)), INTEG(6+(k*24)), INTEG(10+(k*24)),
    INTEG(14+(k*24)), INTEG(18+(k*24)), INTEG(22+(k*24)));
                k=3;setToolData(INTEG(2+(k*24)), INTEG(6+(k*24)), INTEG(10+(k*24)),
    INTEG(14+(k*24)), INTEG(18+(k*24)), INTEG(22+(k*24)));
                k=4;setToolData(INTEG(2+(k*24)), INTEG(6+(k*24)), INTEG(10+(k*24)),
    INTEG(14+(k*24)), INTEG(18+(k*24)), INTEG(22+(k*24)));

                getToolData();

                break;

            case 4:         // Start screen
                break;

            case 5:         // Load screen
                break;

            case 6:         // Offset screen
                switch (buffer[2])
                {
                    case 'L':
                        moveAxis(C_XDEC, (buffer[3] * 3937 / 100));
                        break;
                    case 'B':
                        moveAxis(C_YINC, (buffer[3] * 3937 / 100));
                        break;
                    case 'F':
                        moveAxis(C_YDEC, (buffer[3] * 3937 / 100));
                        break;
                    case 'R':
                        moveAxis(C_XINC, (buffer[3] * 3937 / 100));
                        break;
                    case 'U':
                        moveAxis(C_ZDEC, (buffer[3] * 3937 / 100));
                        break;
                    case 'D':
                        moveAxis(C_ZINC, (buffer[3] * 3937 / 100));
                        break;
                }
```

```
                break;
        }

    } else if (buffer[i] == 0xAF)   // Payload contains PLT file data
    {

    } else if (buffer[i] == 0xA0)   // Payload contains NCD file data
    {

    } else
    {
        return -1;
    }
    stateMachine_Init();
}


void UART1_Init()
{
    #define UART1IRQ (7)
    VICVectAddr2 = (unsigned int)IRQ_UART1_Routine;
    VICVectCntl2 = (1<<5)|UART1IRQ; // Enable vector
    VICVectAddr=-1;
    VICIntEnable = (1<<UART1IRQ); // Enable interrupt

    unsigned char lcr;
    lcr = 0x03;     // 00000011 Line Control Register
                    //       .. Character length 11 = 8 bits
                    //      .   Stop bits         0 = 1 bit
                    //     .    Parity enable     0 = No parity
                    //    .     Even parity       non sense
                    //   .      Stick parity      non sense
                    //  .       Break control     0 = No break
                    // .        DLAB              0 = No Access to DLM,DLL
    U1LCR = 0x80 | lcr; // DLAB = 1, Access to DLM,DLL
    U1DLL = 32;
    U1DLM =  0;     // Baud = 4*14745600/(16* 32) = 115200
    U1LCR = lcr;    // Same a above, but DLAB = 0 = Normal access to RBR/THR & IER
    U1FCR = 0x07;   // 00000111 FIFO Control Register
                    //        . FIFO enable       1 = TX&RX FIFOs enabled
                    //       .  RX FIFO Reset     1 = Reset RX FIFO
                    //      .   TX FIFO Reset     1 = Reset TX FIFO
                    // ..       RX interrupt trigger level 00 = 1 byte
    U1IER = 1;    // Enable the RX line status interrupts
}


void UART1_putch(char c)
{
    while(!(U1LSR & THR_E));    // while U1LSR.THR_E == 0
    U1THR = c;
}


char UART1_getch()
{
    while(!(U1LSR & DR));       // while U1LSR.DR == 0
    return U1RBR;
}


void UART1_print(char * s)
{
    int i;
    i= 0;
    while(s[i]) UART1_putch(s[i++]);
}


void UART1_putch_bs(char c)
{
 if(c==SOF)
 {
  UART1_putch(ESC);
  UART1_putch(SOF);
 }
 else if(c==ESC)
 {
  UART1_putch(ESC);
  UART1_putch(ESC);
 }
 else
  UART1_putch(c);
}


void sendFrame(char * s, char len)
{
    int i;
    char chk = 5;

    UART1_putch(SOF); _printf("%2x␣", SOF);
    // UART1_putch(len+3);  _printf("%2x ", (len+3));
    UART1_putch_bs(len);  _printf("%2x␣", (len));
    //frame length
    for(i=0;i<len; i++) {UART1_putch_bs(s[i]);    _printf("%2x␣", s[i]);}
    // chk = CRC8(s, 0x18, 8); //
    UART1_putch_bs(chk);  _printf("%2x", chk);
    // UART1_putch(SOF);
}


void stateMachine_Init()
{
    uart1_i = 0;
    salir = 0;
    state = 0;
    clear_buffer_flag();
}


void setToolData(unsigned char tool_number, int depth, int iteration, int x_feed,
    int z_feed, int spindle)
{
    tool[tool_number-1].tool_number = tool_number;
    tool[tool_number-1].depth = depth;
    tool[tool_number-1].iteration = iteration;
    tool[tool_number-1].x_feed = x_feed;
    tool[tool_number-1].z_feed = z_feed;
    tool[tool_number-1].spindle = spindle;
}


void getToolData()
{
    ToolInput1[0].value = tool[0].depth;
    ToolInput1[1].value = tool[0].iteration;
    ToolInput1[2].value = tool[0].x_feed;
    ToolInput1[3].value = tool[0].z_feed;
    ToolInput1[4].value = tool[0].spindle;

    ToolInput2[0].value = tool[1].depth;
    ToolInput2[1].value = tool[1].iteration;
    ToolInput2[2].value = tool[1].x_feed;
    ToolInput2[3].value = tool[1].z_feed;
    ToolInput2[4].value = tool[1].spindle;

    ToolInput3[0].value = tool[2].depth;
    ToolInput3[1].value = tool[2].iteration;
    ToolInput3[2].value = tool[2].x_feed;
    ToolInput3[3].value = tool[2].z_feed;
    ToolInput3[4].value = tool[2].spindle;

    ToolInput4[0].value = tool[3].depth;
    ToolInput4[1].value = tool[3].iteration;
    ToolInput4[2].value = tool[3].x_feed;
    ToolInput4[3].value = tool[3].z_feed;
    ToolInput4[4].value = tool[3].spindle;

    ToolInput5[0].value = tool[4].depth;
    ToolInput5[1].value = tool[4].iteration;
    ToolInput5[2].value = tool[4].x_feed;
    ToolInput5[3].value = tool[4].z_feed;
    ToolInput5[4].value = tool[4].spindle;
}

void printToolData(unsigned char tool_number)
{
    _printf("---␣Tool␣%d␣---\ndepth:␣%7d\niteration:␣%3d\nx␣feed:␣%6d\nz␣feed:␣%6d\
    nspindle:␣%5d\n", tool[tool_number-1].tool_number, tool[tool_number-1].depth,
    tool[tool_number-1].iteration, tool[tool_number-1].x_feed, tool[tool_number-1].
    z_feed, tool[tool_number-1].spindle);
}

void fillPayloadWithTool()
{
    send_payload[0] = 0xC3;
    send_payload[1] = 3;

    int i, j, k;
    char * p;
    k = 2;

    for (j = 0; j < 5; j++)
    {
        p = (char*)&tool[j];
        // sizeof(struct ToolData) = 24 (not 21, because we have 5 ints and one char
    , there are three bytes padding to fill up the char)
        for (i = 0; i < sizeof(ToolData); i++)
        {
            send_payload[k] = p[i];

            k++;
        }
    }
}


void setRouteDrill(int x_offset, int x_clip, int x_vectors, int y_offset, int y_clip
    , int y_vectors, int z_board, int z_base, int x_min_value, int x_max_value, int
    y_min_value, int y_max_value /*, int total, int done*/)
{
    randd.x_offset=x_offset;
    randd.x_clip=x_clip;
    randd.x_vectors=x_vectors;
    randd.y_offset=y_offset;
    randd.y_clip=y_clip;
    randd.y_vectors=y_vectors;
    randd.z_board=z_board;
    randd.z_base=z_base;
    randd.x_min_value=x_min_value;
    randd.x_max_value=x_max_value;
    randd.y_min_value=y_min_value;
    randd.y_max_value=y_max_value;
}


void getRouteDrill()
{
    NumpadInput[0].value=randd.x_offset;
    NumpadInput[1].value=randd.y_offset;
    NumpadInput[2].value=randd.z_board;
    NumpadInput[3].value=randd.z_base;
    NumpadInput[4].value=randd.x_clip;
    NumpadInput[5].value=randd.y_clip;
}


void fillPayloadWithRandD()
{
    send_payload[0] = 0xC3;
    send_payload[1] = 2;

    int i, j, k;
    char * p;
    k = 2;
    p = (char*)&randd;

    for (i = 0; i < sizeof(RouteDrill); i++, k++)
    {
        send_payload[k] = p[i];
    }
}

void set_buffer_flag()
{
  buffer_flag = 1;
}

void clear_buffer_flag()
{
  buffer_flag = 0;
}

void set_uart1_isr_flag()
{
  uart1_isr_flag = 1;
}

void clear_uart1_isr_flag()
{
  uart1_isr_flag = 0;
}
```

lib/vardec.h

```c
#pragma once

#include "system.h"

#define amountoffiles       (12)
#define amountofcharacters  (14)
```

lib/xpt2046.h

```
#pragma once

#include "system.h"

#define T_CLK   (1<<4)
#define T_CS    (1<<2)
#define T_DIN   (1<<6)
#define T_DOUT  (1<<5)

#define T_CS_L() {FIOCLR=T_CS;}
#define T_CS_H() {FIOSET=T_CS;}

#define X_READ  (0x50)
#define Y_READ  (0x10)


int TP_read(int ctrlByte);
unsigned int getX();
unsigned int getY();
int uncalibratedX();
int uncalibratedY();
int calibratedX();
int calibratedY();

void readPoints(int * points);
void measurePoint(int x, int y, int error, int x_store, int y_store, int * points);


#define MATRIX_AN       (-42336)
#define MATRIX_BN       (1344)
#define MATRIX_CN       (1047648)
#define MATRIX_DN       (0)
#define MATRIX_EN       (-42940)
#define MATRIX_FN       (719132)
#define MATRIX_DIVIDER  (-37968)
```

```
#include "xpt2046.h"

extern const unsigned char * CurrentFont;
extern unsigned char SmallFont[];
extern unsigned char BigFont[];
extern unsigned char SevenSegNumFont[];

/**
 * @brief Reads a value from the touch controller, at a certain address
 *
 * @param ctrlByte is the address at which data will be read
 * @return int that is the data read at the provided address
 */
int TP_read(int ctrlByte)
{
        SPI_CLK_normal();
        T_CS_L();
        SPI_byte(0x80 | ctrlByte);

        int result = 0;
        result = SPI_byte(0)<<8;
        result |= SPI_byte(0);
        T_CS_H();

        SPI_CLK_fast();
        return ((result>>3) & 0x0FFF);
}


/**
 * @brief Reads an 'x' coordinate from the touch controller
 *
 * @return unsigned int
 */
unsigned int getX()
{
        return (TP_read(X_READ));
}

/**
 * @brief Reads a raw 'y' coordinate from the touch controller
 *
 * @return unsigned int
 */
unsigned int getY()
{
        return (TP_read(Y_READ));
}


/**
 * @brief Returns a converted, but uncalibrated value between '0' and '479'
 *
 * @return int
 */
```

```
int uncalibratedX()
{
        return (((4096-getX())*1000)/8533);
}

/**
 * @brief Returns a converted, but uncalibrated value between '0' and '271'
 *
 * @return int returns an uncalibrated 'y' value read from the touch controller
 */
int uncalibratedY()
{
        return (((4096-getY())*1000)/15059);
}


/**
 * @brief Calibrates the read 'x' coordinate, so that we take the misalignment of
    the screen into account
 *
 * @return int is a value between '0' and '479'
 */
int calibratedX()
{
        return ((uncalibratedX()*MATRIX_AN/MATRIX_DIVIDER)+(uncalibratedY()*
    MATRIX_BN/MATRIX_DIVIDER)+(MATRIX_CN/MATRIX_DIVIDER));
}

/**
 * @brief Calibrates the read 'y' coordinate, so that we take the misalignment of
    the screen into account
 *
 * @return int is a value between '0' and '271'
 */
int calibratedY()
{
        return ((uncalibratedX()*MATRIX_DN/MATRIX_DIVIDER)+(uncalibratedY()*
    MATRIX_EN/MATRIX_DIVIDER)+(MATRIX_FN/MATRIX_DIVIDER));
}


/**
 * @brief
 *
 * @param points
 */
void readPoints(int * points)
{
        int X1 = 72;
        int Y1 = 41;
        int X2 = 240;
        int Y2 = 231;
        int X3 = 408;
        int Y3 = 136;
        int X4 = 288;
        int Y4 = 109;
        int X5 = 120;
```

```c
        int Y5 = 245;

        int ERROR = 20;

        CurrentFont = SmallFont;

        printCString("Calibrate␣screen,␣touch␣the␣points", 5, 5, 0xFFFF, 0x0000);
        printCString(":␣1/5", 272, 5, 0xFFFF, 0x0000);
        measurePoint(X1, Y1, ERROR, 0, 1, points);
        printCString(":␣2/5", 272, 5, 0xFFFF, 0x0000);
        measurePoint(X2, Y2, ERROR, 2, 3, points);
        printCString(":␣3/5", 272, 5, 0xFFFF, 0x0000);
        measurePoint(X3, Y3, ERROR, 4, 5, points);
        printCString(":␣4/5", 272, 5, 0xFFFF, 0x0000);
        measurePoint(X4, Y4, ERROR, 6, 7, points);
        printCString(":␣5/5", 272, 5, 0xFFFF, 0x0000);
        measurePoint(X5, Y5, ERROR, 8, 9, points);
}


/**
 * @brief
 *
 * @param x
 * @param y
 * @param error
 * @param x_store
 * @param y_store
 * @param points
 */
void measurePoint(int x, int y, int error, int x_store, int y_store, int * points)
{
        calibrationPoint(x, y);

        while(1){

                if ( (uncalibratedX() < (x + error)) && (uncalibratedX() > (x - error)) )
                {
                        if ( (uncalibratedY() < (y + error)) && (uncalibratedY() > (y - error)) )
                        {
                                points[x_store] = uncalibratedX();
                                points[y_store] = uncalibratedY();
                        }
                }
                if ( (points[x_store] < (x + error)) && (points[x_store] > (x - error)) )
                {
                        if ( (points[y_store] < (y + error)) && (points[y_store] > (y - error)) )
                        {
                                if (calibratedX() == 480)
                                {
                                        if (calibratedY() == 0)
                                        {
                                                break;
                                        }
                                }
                        }
                }
                _delay_ms(150);
        }
        fillCBox((x-error), (y-error), (x+error), (y+error), 0x0000);
}
```