Touch sensitive user interface module
for ARM Cortex-M3

OSKAR RÖNNQVIST

Master of Science Thesis
Stockholm, Sweden 2011
Touch sensitive user interface module for ARM Cortex-M3

Oskar Rönnqvist

Touch-Controller
Display-Controller
MidRange-Platform
Display

Touch-Panel

Master of Science Thesis MMK 2011:24 MDA403
KTH Industrial Engineering and Management
Machine Design
SE-100 44 STOCKHOLM
Examensarbete MMK 2011:24 MDA403

Pekskärmsanvändargränssnittsmodul till ARM Cortex-M3

Oskar Rönqvist

Godkänt
2011-03-23

 Examiner Mats Hanson
Handledare Bengt Eriksson

Uppdragsgivare
Syntronic AB
Kontaktperson Andreas Källkvist

Sammanfattning

Denna rapport utreder hur ett pekskärmsanvändargränssnitt kan implementeras på Syntronic AB’s hård- och mjukvaruplattform baserad på ARM Cortex-M3.

Ett pekskärmsanvändargränssnitt kräver två system, ett tryckkänsligtsystem för att detektera och positionera när användaren ger input till systemet och en skärm för att presentera ett grafiskt användargränssnitt.

Rapporten börjar med att beskriva de tillgängliga teknikerna för pekpaneler och skärmar. Detta syftar till att ge djupare kunskap om de olika teknologierna och deras för- och nackdelar. Denna kunskap används sedan för att i samarbete med Syntronic AB utvärdera de olika teknologierna utifrån Syntronic AB’s krav. Utvärderingen leder till val av teknik för både pekpanel och skärm som köps in och implementeras på plattformen.

Implementationen består av både hårdvarunära programmering för kommunikationen med pekpanel och skärm och API mjukvaror för att hantera hårdvaran på ett enkelt sätt.

Både hård- och mjukvara är dokumenterad för att visa hur systemet är uppbyggt. Denna dokumentation skall också kunna användas som en manual till mjukvaran vid vidareutveckling eller tillämpning.

Abstract

This report investigates how a touch sensitive user interface can be implemented on Syntronic AB’s hardware and software platform based on an ARM Cortex-M3 microcontroller.

A touch sensitive user interface consists of two systems, one touch sensitive system to detect and position user input and one display to present a graphical user interface.

The first part of the report describes available touch and display technologies. The purpose is to deepen the knowledge about the different technologies and their advantages and disadvantages. This knowledge is then used to evaluate the technologies in collaboration with Syntronic AB based on their needs. The evaluation leads to a choice of technology both for the touch sensitive system and display technology. Hardware corresponding to the technology choices is ordered and implemented.

The implementation consists of both software that is strongly connected to the hardware to handle the communication with the touch panel and display and API software to enable easier interfacing to the hardware.

Both hard- and software are documented to describe the system structure. This documentation can also be used as a software manual to ease further development or implementation.

The result of the implementation is a touch panel and a display with a simple GUI. The implementation clearly shows how a touch sensitive interface can be implemented on the platform and provides software that is reusable.
# Table of Contents

1. Introduction ......................................................................................................................... 1
   1.1. Background.......................................................................................................................... 1
   1.2. System Components .......................................................................................................... 2
   1.3. Requirement Specification .............................................................................................. 3
       1.3.1. Touch Requirements .................................................................................................. 3
       1.3.2. Display Requirements .............................................................................................. 3
   1.4. MidRange.......................................................................................................................... 4
       1.4.1. Introduction ............................................................................................................... 4
       1.4.2. Hardware .................................................................................................................. 4
       1.4.3. Software ................................................................................................................... 5
2. Pre-study................................................................................................................................. 6
   2.1. Touch................................................................................................................................. 6
      2.1.1. Touch Technologies .................................................................................................. 6
   2.2. Display............................................................................................................................... 12
      2.2.1. Display Technologies ................................................................................................ 12
3. Specification............................................................................................................................ 16
   3.1. Parameter Weighting ........................................................................................................ 16
   3.2. Technology Evaluation ..................................................................................................... 18
   3.3. Technology Specification ................................................................................................ 20
      3.3.1. Touch ....................................................................................................................... 20
      3.3.2. Display .................................................................................................................... 20
   3.4. Preliminary System Draft .............................................................................................. 20
      3.4.1. Touch ....................................................................................................................... 20
      3.4.2. Display .................................................................................................................... 23
4. Implementation....................................................................................................................... 25
   4.1. High Level system ........................................................................................................... 25
      4.1.1. Hardware .................................................................................................................. 25
      4.1.2. Software ................................................................................................................... 26
   4.2. Touch Module.................................................................................................................. 28
1. Introduction

1.1. Background

Today Syntronic is working with several different platforms and technologies. The latest platform called MidRange is based on an ARM Cortex-M3 controller. The MidRange platform is a cost efficient and versatile alternative to Syntronic's more advanced platforms. A library of both software and hardware modules for the platform has been built by Syntronic to enable faster development of new projects.

The development process and projects based on MidRange have shown that the platform needs to be extended and further developed in some areas. One of those areas is the user-interface.

When the MidRange platform is used in projects today the primary way of getting input from users is by buttons and other button-like hardware. This input method can be limiting in some type of applications. Because of this, Syntronic wants to investigate how a touch-interface could be used with the MidRange platform. A touch-interface would improve the user-interface possibilities and make the platform more versatile.

A touch-interface includes two separate modules, one display and one touch sensitive system. These modules will act as one system in user-applications but can individually be separated in hardware and software.

The display is used to present interface items and the touch sensitive system is used to register where the user gives input. The display used today is a simple monochrome-LCD. This display could be sufficient but preferably a color display should be used. This type of display is not supported today so hardware must be chosen and software developed. The touch sensitive system technology is not specified and could be of any technology.

The modules should be evaluated regarding environmental parameters such as temperature, humidity, lighting conditions etc. This is important since the platform is and will be used in different climates, the focus should also be at durability and user friendliness.

This report will describe different technologies and evaluate these in relation to the MidRange platform to enable the best possible implementation. The implementation will be described, both in hardware and software. The process will also be described throughout the report.
1.2. System Components

The system will consist of two separate parts, the display and the touch panel. These systems will be connected to the MidRange via controllers. These systems are able to function individually but a touch interface requires them both.

The display is used to display information to the user and the touch-panel to receive input from the user.
1.3. **Requirement Specification**

This section will discuss the requirements that are set for the different systems. The requirements are few and vague and its purpose is to set some basic guidelines to aid the pre-study and choice of technology.

1.3.1. **Touch Requirements**

The requirements for the touch panel are not restricting or high performance since the MidRange in itself has quite weak computation capabilities. These requirements are intended to be vague and should be seen as pointers to support the choice of technology.

- Resolution: Good enough to sense a touch of finger size i.e. the touch panel should recognize a touch in the area of the object touching it.
- Response time: 100 ms shouldn’t be an issue since the response time most likely will not be a bottleneck.
- Interface: Any suitable interface on MidRange.

1.3.2. **Display Requirements**

The display requirements

- Size: 5-7”
- Color: RGB
- Resolution: minimum QVGA(320x240)
- Interface: Any suitable interface on MidRange, preferably prepared I80/M68 FSMC interface(see section4.3.3.3Interfacing (FSMC)).
1.4. MidRange

1.4.1. Introduction

The MidRange platform was developed by Syntronic AB to be used in projects with lower performance demands and low cost demands. The platform consists of both hardware and software libraries. These libraries are continuously expanded with new material. These additions can come from projects, thesis students or in-house platform development.

This makes it possible to reuse a lot of previous work in new projects and by that shorten development times and reduce costs.

MidRange is based on a Cortex M3 (STM32F103) processor running at 72 MHz. The processor is a low cost 32-bit processor with a lot of I/O functionality.

To make real-time critical applications possible an RTOS can optionally be used on the platform. The chosen RTOS is FreeRTOS, this is a license free RTOS which is developed for small embedded systems. The cortex-M3 port of FreeRTOS is simple to implement and use. It supports the common functions of a RTOS and is used in most parts of the software library.

1.4.2. Hardware

The MidRange hardware is a development board constructed by Syntronic. The development board enables development on common peripheral hardware such as:

- 2xRS232 + 1xUSART
- 8xA/D + 2xD/A
- 1xUSB
- 1xSD card reader(SPI/SDIO)
- 1xCAN
- 1xRS485
- 1xEthernet
- 1xI2C + 1xSPI
- Display interface(FSMC)
- GPIOs

Syntronic's projects start out with development on the development board and if needed additional expansion PCB:s are developed. This can then be presented to the customer for demonstration and evaluation. If the customer wants a larger volume a project specific PCB is constructed. The project specific PCB consists of the required components from the development board and if any associated expansion boards.
1.4.3. Software

Since large parts of the hardware are reused in different projects, much of the code written to them can also be reused. The software is stored in a library with source code and basic projects with common software setups are predefined. Below is a breakdown of some of the drivers that are included in the library.

- RS232
- Ethernet
- CAN
- SD-Card
- I2C
- SPI
- RTOS
2. Pre-study

2.1. Touch

The touch technology literature study aims to deepen the knowledge about different touch technologies to be able to evaluate them according to the specifications developed in cooperation with Syntronic employees. The technology that best suits Syntronic’s demands will then be further analyzed. The further study will handle both the technology itself and the ways of implementing it in a prototype.

2.1.1. Touch Technologies

There are many different touch sensitive technologies. The most common are described below. These have been chosen by looking at the commonly used technologies in embedded systems.

2.1.1.1. Capacitive

2.1.1.1.1. Surface

In surface capacitive touch a thin transparent conductive layer is placed on a glass substrate. This conductive layer has linear resistance which is important for the positioning of the touch location. To sense a touch a small voltage is induced over the conductive layer creating a uniform electrical field. When a conductive object (i.e. a finger) comes close to the conductive layer and electrical field a capacitor is formed. The new capacitor creates a current flow which leads to a voltage drop that can be measured from four corners. The readings from the different corners can then be used to calculate the position of the touch. This method is not that accurate and can be affected by scratches and other false capacitances.

---

1 (Gary L. Berrett u.d.)
2.1.1.2. Projected

Projected capacitance touch is different from surface capacitance touch since there are two conductive layers each with a “pattern” engraved. This pattern can have different designs but the purpose is to divide the layer into sections and when two layers are placed on top of each other with an insulation layer in between a grid is formed. There are two ways to read this grid and by that determine where the touch was made, they are explained below.

Self capacitance

The self capacitance method induces a small voltage over one of the layers and then reads the voltage drops on each of the sections on the second layer. The same is done the other way around and when these readings are combined coordinates to the touch is achieved. This method to read the grid gives stronger signals but does not enable recognition of more than one touch.

Mutual capacitance

This method reads the change in capacitance “from” each grid intersection individually. Practically this is done by reading the voltage drop on all horizontal-axis sections when voltage is induced on one section on the vertical-axis. This will result in different voltage drops depending on the presence of a conductive object. When both axis have been scanned in this manner a projection of the touch is achieved and from that the touch position can be determined. While each intersection is read individually it is possible to recognize and separate more than one conductive object which enables multi-touch.
2.1.1.2. Resistive

Resistive touch comes in different shapes but relies on the same base, two separated conductive layers with a linear resistive coating. The bottom layer is usually applied on a glass body to ensure stability and protection. The top layer is placed on top of the bottom layer with a spacing material separating them. This spacing material can be of different types but must be non-conductive. On top of the top layer a protective layer is placed to protect the sensitive layer from damage. There are two main technologies to measure where a touch was made: 4-wire and 5-wire explained below.

\(^2\) (Atmel 2007)
2.1.1.2.1. 4-wire

4-wire technology is named after the number of wires that needs to be connected to the physical touch interface. Two wires are connected to the top layer and two to the bottom layer. The wires are usually named X-, X+, Y- and Y+, the names simplify the interpretation of the coordinates. To read the position of the touch, different wire functionality is cycled, in Table 1 is a description of the cycles.

**Table 1 – 4-wire positioning cycles**

<table>
<thead>
<tr>
<th>Function</th>
<th>X-</th>
<th>X+</th>
<th>Y-</th>
<th>Y+</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-coordinate</td>
<td>GND</td>
<td>VCC</td>
<td>HI-IMPEDANCE</td>
<td>ADC</td>
</tr>
<tr>
<td>Y-coordinate</td>
<td>HI-IMPEDANCE</td>
<td>ADC</td>
<td>GND</td>
<td>VCC</td>
</tr>
</tbody>
</table>

The X-coordinate cycle induces a voltage over the bottom layer and while the conductive layer is linear resistive there will be a linear voltage drop. When a touch occurs the top layer will act as a voltage probe measuring the voltage division at the touch point. The ratio between measured voltage and induced voltage will correspond to the relative distance from X- and X+. The same principle is used to measure the Y-coordinate but a voltage is induced to the top layer and the bottom layer is used as a voltage probe.

![Schematic of a 4-wire touch screen when pressure is applied.](image)

**Figure 2 - “Schematic” of a 4-wire touch screen when pressure is applied.**

---

3 AVR341: Four and five-wire Touch Screen Controller

4 (Atmel 2007)
2.1.1.2.2. 5-wire

5-wire is as 4-wire named after the number of wires that needs to be connected to the touch interface. The 5-wire implementation is very similar to 4-wire implementation but has a significant difference that affects the durability and accuracy over time. The difference is that all voltage probing is handled by the top layer. One issue with resistive touch is that whenever a touch is made the top layer flexes down to make contact with the bottom layer. This flexing leads to mechanical tear on the top layer which over time can affect the resistive linearity. This will affect the voltage drop across the top layer which leads to inaccurate interpretation of the coordinate.

In the 5-wire technology this is not a problem since the top layer is only used for voltage probing. The ability to sense both X- and Y-coordinates is achieved by applying voltage to the four corners of the bottom layer in different patterns. The four corners are often named UL (Upper Left), UR (Upper Right), LL (Lower Left) and LR (Lower Right) and the top layer Sense. In the cycles that are used to determine X- and Y-coordinates are described.

**Table 2 - 5-wire positioning cycles**

<table>
<thead>
<tr>
<th>Function</th>
<th>UL</th>
<th>UR</th>
<th>LL</th>
<th>LR</th>
<th>Sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-coordinate</td>
<td>GND</td>
<td>VCC</td>
<td>GND</td>
<td>VCC</td>
<td>ADC</td>
</tr>
<tr>
<td>Y-coordinate</td>
<td>GND</td>
<td>GND</td>
<td>VCC</td>
<td>VCC</td>
<td>ADC</td>
</tr>
</tbody>
</table>

This will lead to a linear voltage drop first in X-direction and then in Y-direction.
2.1.1.3. Surface Acoustic\(^5\)

Surface acoustic touch screens use ultrasonic waves traveling in a glass body to determine where a touch was made. This is achieved by placing wave sources in/on a glass screen and reflector elements that bounces the waves across the screen. Receiving units can then read the wave and detect if there is a change in the wave. This change could be made by pressing a finger against the screen and by that absorbing some wave energy. This change in the wave can then be translated to a touch coordinate by comparing the reference wave to the altered wave.

This technology has very low or no optical effect since the only material between user and display is glass. A limitation is that touches must be made with “soft” objects since hard objects do not absorb wave energy.

2.1.1.4. Infrared\(^6\)

Infrared touch technology uses arrays of infrared LEDs and photosensitive receivers. These are placed on the edge of the display in pairs, on one side the infrared LED’s and on the other the photosensitive receivers. If a finger blocks one of the beams the photosensitive receiver connected to that IR-LED will detect this. When these arrays are placed both on the x- and y-axis a finger that touches the display will block at least one beam in x-direction and one in y-direction. These signals can then be used to calculate a coordinate for the touch location.

This technology can sense any object that can block IR-light (i.e. most physical objects). A drawback is that the LED’s require a lot of power in this context. This can be countered by turning off some LED’s when the touch panel is not being used and turning them all on when a user starts to use it to increase accuracy. Another way to decrease power consumption is to pulse the LED’s when that specific led should be measured instead of having them turned on continuously.

\(^5\) (Densitron Corporation 2007)  
\(^6\) (Densitron Corporation 2007)
2.2. Display

2.2.1. Display Technologies

There are many different display technologies that suit embedded systems. To get a better understanding of them a brief explanation of the technologies will be made. Since Syntronic wants a graphical color display only technologies supporting this will be considered. These brief explanations will give means to decide which technology suits Syntronic the best and further research will be carried out on that technology.

2.2.1.1. Organic Light Emitting Diode (OLED)

OLED’s are LED’s which include an organic material (carbon based). The advantages with OLEDs are that they can be made very thin, low power consumption, good contrast and good viewing angle.

A significant difference between LCD’s and OLED-displays is that OLED’s are emitting their own light and therefore removes the need for a backlight. Without a backlight a true black can be achieved since a pixel that shows the color black is turned off and emits no light. This also adds to the low power consumption value because the pixel only requires power when it is activated.

The problems that the OLED technology struggles with are high production costs and degradation of performance.

2.2.1.2. LCD

LCD (liquid crystal display) is the most common technology for color displays in embedded systems today. There are a number of different ways to use liquid crystals to represent graphics and pros and cons with different techniques. Some aspects of the display are separated from the technology itself such as addressing and interfacing, which will be handled in a separated section.

Twisted Nematic (TN)

TN LCD displays use the twist of liquid crystals to control light flow. The cell is built from different parts that together have an advanced functionality. If viewed from the bottom-up (Bottom where the light source is placed) the first component is a polarizing filter that polarizes the light, the next layer is an electrode that is coated in the same direction as the underlying polarization filter. On top of the electrode there are liquid crystals held between two glass layers. Next is the top electrode which is coated 90° from the lower electrode and on top of that a polarization filter which have the same 90° twist from the lower polarization filter. This is the cell build-up and the functionality is explained below.

---

7 (Anoniadis 2003)
8 (Carrein 2007)
In the “off-state” when no voltage is applied over the electrodes, light is able to pass through the liquid crystal cell. This is achieved with help from the coated electrodes. Since the electrodes are coated in a 90° twist the liquid crystals forms a spiral that twists the polarized light 90° so that it is able to pass through the top polarizing filter.

In the “on-state” when voltage is applied over the electrodes the crystals align themselves according to the electrical field. This disrupts the spiral which stops the twist of polarized light and thus stops the light from passing through the top polarization filter. The spiral disruption can be controlled with the voltage applied giving the possibility to control the amount of light that passes through the cell.

![Figure 3 - LCD TN, open and closed pixel](http://commons.wikimedia.org/wiki/File:TN-LCD-schematic-MS-208kB.png)

---

In Plane Switching (IPS)

IPS is another technology very similar to TN but with important differences that overcomes some of the limitations with TN. The most noticeable limitation in TN technology is the viewing angle which is greatly increased with IPS technology.

In IPS technology the LC’s are rotated in the same plane to either let light through or block it. This leads to better light flow through the LC’s which results in better viewing angle properties. The drawback with the technology is that it is slower to turn the LC’s resulting in slow response times. To be able to twist the LC’s in parallel to the plane two electrodes are used, these electrodes must be placed in the light flow and therefore blocks some of the light. Since TN only needs one electrode an IPS panel needs a stronger backlight to achieve the same output light. Improvement to the design of the cell and electrode placing has resulted in less light blockage and also faster response times.
2.2.1.3. Cathode-Ray Tube (CRT)\textsuperscript{10}

CRT is a display technology that displays an image by sending electrons on fluorescent segments of different colors i.e. red, green and blue. This is the technology that made television possible and come in both color and monochrome.

The technology is based on an electron gun which sends out electrons at high speed from the back of the display. These electrons are then shot through a control grid that regulates the flow of electrons through the grid. This makes it possible to regulate how much light a display segment emits. The electrons are then shot through a focusing anode that focuses the electron beam so that it can be directed consequently. These parts are all parts of the electron gun and the following parts are the parts that combined with the electron gun create the CRT.

To be able to control the electron beam to hit specific segments four plates are used. Two for horizontal direction and two for vertical direction. The electron beam first passes through the horizontal directing plates, a magnetic field is created between the plates and by regulating the power of the field the electron beam can be directed to the desired horizontal position. The electrons are then passed between the vertical directing plates that work with the same principle as the horizontal plates.

This makes it possible to control the amount of light emitted from each segment. This technology is considered old today, as it has been replaced primarily by LCD panels.

\textsuperscript{10} (Pearson Education u.d.)
3. Specification

3.1. Parameter Weighting

To be able to choose the best technology both for the display and touch-interface a list of parameters was produced in cooperation with Syntronic and then a weighting of these parameters was done also in cooperation with Syntronic since only they can answer which parameters that are important and which are not. After the rating a the ratings were discussed with Syntronic which led to a numbered 1-5 rating on how important a parameter is and also when a rating was difficult to do a way to reason was derived. The ratings and comments can be found in Appendix A. A graphical representation of the ratings and a discussion can be found below.

As seen in Figure 4 the most important parameters are those associated with environmental effects. This comes from Syntronic’s need to build a versatile platform which can be used in different climates and rough environments. Parameters associated with high performance parameters like multi-touch,
responsiveness and touch accuracy are of less importance. These high performance parameters are less important because the Cortex M3 is a relatively weak controller and cannot really take advantage of these features.

**Figure 5** Graphical representation of Display parameter rating

The display parameter weighting can be seen in Figure 5 and just as the touch-interface environmental parameters are rated high and high-performance parameters are rated low. The reasons for this weighting are the same as for the touch-interface and the focus is on functionality.
3.2. Technology Evaluation

The technologies that were found relevant were evaluated considering the parameters that were weighted in 3.1 and Appendix A. The evaluation was made on the technology in general and not for specific products and should be seen as a point in the right direction. Each technology was evaluated considering the different parameters and were rated in the range:

- “++” - Very good (5)
- “+” – Good (4)
- “*” – Neutral (3)
- “-” – Not good (2)
- “- -” – Bad (1)

Each rating is also associated with a numeric value, in parenthesis. This numerical value is then used in combination with the parameters importance to achieve a weighted rating. The formula used to calculate this weighted rating is:

\[
\text{weighted rating} = \text{importance} \times \text{rating}
\]

The sum of these weighted ratings for each technology can be seen in Figure 6, Figure 7 and Appendix B.
Figure 6 - Weighted ratings of relevant touch technologies

Figure 7 - Weighted evaluation of relevant display technologies
3.3. **Technology Specification**

When reviewing the weighted evaluations of the different technologies it is clear that the technologies are very similar in performance but with some key differences.

3.3.1. **Touch**

The choice of touch technology is based on some key differences that make the chosen technology more versatile and suitable for use with the MidRange platform. The chosen technology is resistive and the key features leading to this decision is the weighted ratings outcome. The features that had a big impact on resistive technology having the highest rating were *power consumption, cost, dust, smudges and dirt, durability and usability with gloves.*

3.3.2. **Display**

The outcome of the weighted parameter rating was very even but there were differences on some parameters. The parameters that differed were *power consumption, cost, and angle of vision.* The outcome of the evaluation is that any of the technologies are suitable for implementation and that the specific products have to be evaluated since the technologies themselves are so similar.

3.4. **Preliminary System Draft**

This section will describe the system draft that was created before the ordered hardware had been delivered. The purpose was to prepare a system design so that the implementation could start without the hardware since lead time can be long.

3.4.1. **Touch**

When this draft was made hardware was ordered but information about the protocol was deficient. This draft helped when building the code draft that later became the finished implementation.

3.4.1.1. **Hardware**

The hardware had the structure shown in Figure 8, a touch panel connected to the MidRange via a touch controller. The hardware structure simplifies implementation since no handling of the signals “in” the touch panel has to be interpreted. The interpretation of the signals on the touch panel is handled by the touch controller.
3.4.1.2. Software

The touch driver will communicate with the touch controller with RS232. An RS232 interface has already been implemented on the MidRange platform and enables a straightforward implementation. The idea is that when the touch controller senses a touch the coordinates will be sent with RS232 to the Cortex M3 controller and handled by the touch driver.

How this will be handled by the driver is not determined but an example is shown below:

![Touch State machine](image.png)

Figure 8 - Touch hardware structure

Figure 9 - Touch State machine
3.4.1.3. Process

The touch interface development is separated into different parts to give overview and enable early and continuous testing early and throughout the development. Below is a list of the steps in the development that will lead to the finished interface.

1. Apply voltage to appropriate wires and measure voltage on appropriate wire to check basic functionality (if possible).
2. Connect controller to touch-film and connect to HyperTerminal and ensure that the signals that are transmitted are correct.
3. Write code to the microcontroller that can receive communication from the touch controller and present it to the user.
4. Write code that calls appropriate functions when a touch is sensed.
3.4.2. Display

3.4.2.1. Hardware
The display hardware is very similar to the touch hardware in structure but not in function. The display is connected to MidRange via a controller just as the touch hardware, but the communication protocols differ.

![Display hardware structure](image)

Figure 10 - Display hardware structure

3.4.2.2. Software
The software controlling the display could be divided into three main parts:

- The layout build-up with different components such as buttons, text field, images etc.
- A part that converts the components to pixel information
- A part that sends the pixel information to the display controller
3.4.2.3. Process

To increase productivity and ensure efficient development a list of different steps that leads to the finished implementation is presented. These steps start out very basic and functionality is added for each step.

1. Connect power and communication cables to controller and display
2. Initialize FSMC\textsuperscript{11} interface on cortex M3 and do basic communication with controller.
3. Draw display in different colors
4. Draw geometric shapes on display in different colors
5. Draw text on display
6. Draw images on display
7. Develop simple GUI builder with simple components (i.e. geometric shapes with actions)
8. Develop more complex GUI builder that support multiple components of varying complexity

\textsuperscript{11} 4.3.3 Interfacing (FSMC)
4. Implementation

This section describes the implementation in four parts: High Level system, Touch Module, Display Module and Graphical User Interface Module. Together these modules create the high level system that the user comes in contact with.

4.1. High Level system

4.1.1. Hardware

The hardware can be divided into three modules; MidRange, touch and display. They are connected to each other with different interfaces and communicate with different protocols. The physical connections and modules are shown in Figure 11. The module setup makes it possible to develop software in two different layers; hardware dependant and non-dependant. This simplifies implementation of other hardware and additional features to the API. More about this in the following section.

![Figure 11 - System hardware overview](image)
4.1.2. Software

The software for the touch module and display module is divided into two layers, hardware dependant (driver) and non hardware dependant (API). These parts are well separated in different source files for easier implementation of new hardware and additional features. A demonstration on how the separation is done in code can be seen below.

```c
void lcd_init(){
    WF70ATIBGDA_init();
}
```

Code Example 1 - software layers example

The `lcd_init()` function is a typical API function while the `WF70ATIBGDA_init()` is a typical driver function in this case for the display with product name `WF70ATIBGA`. The driver function that is called in the API implementation function can easily be changed to another driver without having to change the code that is using the API.

The system functionality with touch panel and display is achieved with just one main loop. This task handles the communication with the touch panel and the display via the API functions. This task should run as often as possible to achieve a responsive system but the functionality will not be compromised if it run less frequently. The following sections will explain the three parts (touch, GUI and display) closer.
Figure 12 - System software overview
4.2. Touch Module

4.2.1. Hardware

The hardware that was used in the implementation was one capacitive and one resistive touch panel. Both were bought from forenex Technology Ltd. through Multicomponent AB. This distributor was chosen since Syntronic has an ongoing business relationship with them. Compromises had to be made considering specifications since ordering of these products in small quantities is expensive and sometimes not possible at all.

The touch panels were bought in a set with attached controller boards. These controller boards handle the interpretation of the physical signals from the touch panel and delivers coordinates and some other information over a RS232 interface. This simplifies the implementation and does not affect the control of touch events much. Both controller boards use the same communication protocol which also simplifies the implementation.

The touch panel performance is very good according to the datasheet considering the platform it is connected to. It is specified to deliver:

<table>
<thead>
<tr>
<th>Panel</th>
<th>Report rate(points/sec)</th>
<th>Tolerance</th>
<th>Resolution(&quot;pixels&quot;)</th>
<th>Response time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacitive</td>
<td>180</td>
<td>+/- 1%</td>
<td>2048x2048</td>
<td>25</td>
</tr>
<tr>
<td>Resistive</td>
<td>160</td>
<td>+/- 0,5%</td>
<td>2048x2048</td>
<td>35</td>
</tr>
</tbody>
</table>

In theory this resolution and tolerance gives a touch resolution of:

\[
\text{resolution}_{x_{axis}} = \frac{\text{panel width}}{\text{pixels}_{x_{axis}}} = \frac{153}{2048} \approx 0,075 \text{ mm/pixel}
\]

\[
\text{resolution}_{y_{axis}} = \frac{\text{panel height}}{\text{pixels}_{y_{axis}}} = \frac{92}{2048} \approx 0,045 \text{ mm/pixel}
\]

Number of touch points = panel height × panel width = 2048 × 2048 = 4 194 304

This means that the touch panel can be divided into 4 194 304 segments with the size0,075 x 0,045mm. These segments are significantly smaller than a finger so much of the accuracy lies in the software evaluating the signals.
4.2.2. Connecting Hardware

The connections with the MidRange that are required to give the controllers/touch panels power and enable communication can be seen in the figures below in Figure 13 and Figure 14.

As can be seen the connections are very basic thanks to the controller board. The number of connections required from the MidRange to the controller board is just 4 wires; RS232 communication and power supply. The only difference between the capacitive and resistive connections is that one more wire is required from the controller board to the touch panel in the capacitive case. The capacitive panel needs an extra wire that probes the touch panel while the resistive uses different existing wires for probing.
4.2.3. **Touch Controller Communication**

The touch-panel controller communicates with RS232/USART and use a protocol that is explained in the datasheet\(^{12}\) that comes with the controller. An explanation of how the protocol is built and how it is handled in the cortex M3 follows below.

4.2.4. **Communication Parsing**

The data that are sent from the touch controller cannot be used as it is formatted in the packets so they have to be parsed. The parsing routine aims to extract and store all data in an easy accessed data type. The parsing can be divided into different phases:

![Packet parsing diagram]

*Figure 15 - Packet parsing*

\(^{12}\)(Forenex Technology Co., Ltd. 2010)
4.2.4.1. Diagnostic Packets

Diagnostic packets are used to ask the controller questions and getting a response. The diagnostic questions are: Check active and Get firmware. The structure of these commands are described in the table below.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3-Byte (3+LengthInByte)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0A</td>
<td>LengthInByte +1</td>
<td>Command</td>
<td>Response</td>
</tr>
</tbody>
</table>

Table 3 - Diagnostic Packet

4.2.4.2. Report Packets

Report packets are the packets that are sent when a touch is sensed by the touch controller. The controller sends report packets 160-180 times per second. The report packets contain information about the touch and a breakdown of the information obtained can be seen below.

- Pressure information (if obtainable) or Player ID (if obtainable)
- Resolution of touch point coordinates (11-14 bits)
- A-axis raw data
- B-axis raw data
- Touch status (touch/lift off)

This information is sent in a packet of 5-6 bytes, 5 bytes if no pressure information is obtainable. The bytes are built up as seen in Table 4 - Report Packet.

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td>1</td>
<td>Z</td>
<td>M</td>
<td>0</td>
<td>0</td>
<td>AD1</td>
<td>AD0</td>
<td>STATUS</td>
</tr>
<tr>
<td>Byte 1</td>
<td>0</td>
<td>A13</td>
<td>A12</td>
<td>A11</td>
<td>A10</td>
<td>A09</td>
<td>A08</td>
<td>A07</td>
</tr>
<tr>
<td>Byte 2</td>
<td>0</td>
<td>A06</td>
<td>A05</td>
<td>A04</td>
<td>A03</td>
<td>A02</td>
<td>A01</td>
<td>A00</td>
</tr>
<tr>
<td>Byte 3</td>
<td>0</td>
<td>B13</td>
<td>B12</td>
<td>B11</td>
<td>B10</td>
<td>B09</td>
<td>B08</td>
<td>B07</td>
</tr>
<tr>
<td>Byte 4</td>
<td>0</td>
<td>B06</td>
<td>B05</td>
<td>B04</td>
<td>B03</td>
<td>B02</td>
<td>B01</td>
<td>B00</td>
</tr>
<tr>
<td>Byte 5</td>
<td>0</td>
<td>P06</td>
<td>P05</td>
<td>P04</td>
<td>P03</td>
<td>P02</td>
<td>P01</td>
<td>P00</td>
</tr>
</tbody>
</table>

Table 4 - Report Packet

**Byte 0:** Packet header contains information about how the rest of the packet should be handled and if it was a touch or lift off.

Bit 0 (STATUS): Touch status touch/lift off

0: Lift off
1: touch

Bit 1-2(AD0,AD1): Touch raw data resolution.

- AD0:AD1 = 0:0 11-bits
- AD0:AD1 = 1:0 12-bits
- AD0:AD1 = 0:1 13-bits
- AD0:AD1 = 1:1 14-bits

Bit 5(M): Indicates if there are player ID information in the packet

- 0: No player ID information
- 1: Player ID information present and sent as byte 5

Bit 6(Z): Indicates if there are pressure information in the packet

- 0: No pressure information
- 1: Pressure information present and sent as byte 5

**Byte 1:** A-axis raw data, high bits. How many bits that are data are obtained from AD0 and AD1 bits in Byte 0.

- 0-6: A07-A13

**Byte 2:** A-axis raw data, low bits.

- 0-6: A00-A06

**Byte 3:** B-axis raw data, high bits. How many bits that are data are obtained from AD0 and AD1 bits in Byte 0.

- 0-6: B07-B13

**Byte 4:** B-axis raw data, low bits.

- 0-6: B00-B06

**Byte 5:** Pressure/Player ID data, which one can be resolved from bit 6 and 7 in byte 0.

- 0-6: P00-P06
4.2.5. Touch Interpretation

The touch controller board sends packets of data at 160-180Hz when a touch is identified. The packets can either indicate that a touch is being made or that a liftoff was made (for further information see Touch Controller Communication).

The rate of data transmission induce some problems.

- In most cases only the first touch position and/or release position is relevant. In some implementations the “drag” motion of a touch can be relevant.
- “touch bouncing” can occur if a touch is made with a trembling finger/item or due to signal irregularities. This can lead to unintentional inputs.
- Very small differences in touch coordinates gets registered, most of the time a much lower resolution is enough.

To decrease the load on the microprocessor these irrelevant packets should be identified and dismissed as soon as possible. The way that this is done is by looking at the type of touch event (touchdown / liftoff), distance between touch event coordinates and time between touch events. By using these parameters a third touch event can be identified “move”. A touch on the touch panel will then result in minimum two touch events and in theory infinite touch events.

- A touch on a single point without “dragging” results in a “touchdown”-event and a “liftoff”-event.
- A touch on one point then a “dragging” movement with continuous contact with the touch panel results in a “touchdown”-event, multiple “move”-events and a “liftoff”-event.

The purpose of the “move”-event is to further increase the flexibility in the touch API. The move event can be used for “mouse-over” functionality to make an interface more like a regular computer controlled with a computer-mouse. The process of determining if packets are relevant or not can be seen in the flow visualized in Figure 16. The time and position difference threshold values are set to 50 ms and 50 pixels. These values are produced empirically.

![Figure 16 - Touch event determination flow](image-url)
In Figure 17 a demonstration of the different touch events can be seen. The touch was made by touching the touch panel at the green square and then dragging with continuous contact with the touch panel to the red square. The different color of the squares indicates which touch event that was triggered at that position. The green square is a “touchdown”-event, the grey squares are “move”-events and the red square is a “liftoff”-event. The distance between the grey “move” squares is the defined minimum distance between touch events coordinates. The defined time threshold is hard to illustrate but can be noticed when dragging fast across the touch panel.

![Figure 17 - Different touch events demonstration](image)

This interpretation of touch events is very basic and could be further developed. There are some functions that many users take for granted in modern touch interfaces and some of these can be implemented on this system. A breakdown of some interesting functions for further work is listed below.

- Double click – make the software able to recognize when two touches are made at the same position with a defined (small) time interval.
- Sweeping – Recognize if the user does a dragging touch in multiple directions. This is commonly used in touch interfaces to substitute next/previous buttons.
- Gesture – add gesture recognition so that input can be made without specific buttons. These gestures could be combined with the Sweeping actions or be more advanced freeform gestures.
4.2.6. Touch Calibration

The touch panel needs to be calibrated to ensure a good match between touch panels coordinates and display coordinates. This calibration can be done on each startup or when the touch panel starts to report touch coordinates that doesn’t correspond to the display coordinates.

The principle is that during the calibration some transformation constants are calculated. After the calibration when a touch is sensed the coordinate reported from the touch controller gets recalculated using the earlier obtained calibration constants. By doing this the un-calibrated coordinates from the touch controller becomes calibrated display coordinates and the touches become accurate.

There are different sources to the inaccuracy and these can be handled and/or prevented in different ways. Below is a breakdown of the most common inaccuracy sources.

4.2.6.1.1. Mechanical misalignment

When attaching the touch panel to the display it is very likely that it will be misaligned both in translation and rotation.

4.2.6.1.2. Touch panel scaling

Many touch panels suffer from a non uniform scaling in both x- and y-axis. This can lead to errors.

4.2.6.1.3. Person specific touch technique

Each person has a unique touch technique and this can make a well calibrated touch panel appear as poor calibrated for another person. This can be handled by letting people recalibrate the touch panel if they feel that the calibration does not fit their touch technique. Another approach is to calibrate the touch panel and let people “learn” how to touch the touch panel correctly.

4.2.6.2. Two-point Calibration

The two-point calibration is easy to understand for the user and does not require heavy computations. The way it is done is by presenting an image on the display with two points that should be touched in a specific order. The calibration sequence can be divided into different steps:

1. Display two targets on the display, one in the lower left (LL) and one in the upper right (UR). The coordinates of these targets should be the distance 1/8th display width/height from respective corner. The equations below shows the touch panel raw data associated with these points.

   \[
   LL = \left( \frac{1}{8} \times \text{display width}, \frac{1}{8} \times \text{display height} \right) = (256_{\text{raw data}}, 256_{\text{raw data}}) \\
   UR = \left( \frac{7}{8} \times \text{display width}, \frac{7}{8} \times \text{display height} \right) = (1792_{\text{raw data}}, 1792_{\text{raw data}}) 
   \]

2. The user is then prompted to touch first the LL and then the UR target. The raw data from those touches are then saved as \( LL_{\text{calibration}} \) and \( UR_{\text{calibration}} \) as follows
3. Now the calibration is finished. Now when a touch is registered the raw data will be recalculated using these formulas

\[ \begin{align*}
LL_{\text{calibration}} &= (LL_{\text{calibration} x}, LL_{\text{calibration} y}) \\
UR_{\text{calibration}} &= (UR_{\text{calibration} x}, UR_{\text{calibration} y})
\end{align*} \]

\[ \begin{align*}
X_{\text{calibrated}} &= \left( \frac{X_{\text{raw data}} - LL_{\text{calibration} x}}{UR_{\text{calibration} x} - LL_{\text{calibration} x}} \right) \times 1792 \\
Y_{\text{calibrated}} &= \left( \frac{Y_{\text{raw data}} - LL_{\text{calibration} y}}{UR_{\text{calibration} y} - LL_{\text{calibration} y}} \right) \times 1792
\end{align*} \]

The resulting \( X_{\text{calibrated}} \) and \( Y_{\text{calibrated}} \) will then be calibrated touch values. These calculations are performed on the micro controller for all touch coordinates that are received from the touch controller. This calibration algorithm will not be able to compensate for a rotational mechanical misalignment but translation errors in both x and y will be corrected.

Figure 18 - 2-point calibration image

4.2.6.3. \textit{N}-point Calibration

To compensate for rotational misalignment a calibration routine with more touch points are required. There are typical of 3-point or 5-point type and uses a more advanced rotational matrix to recalculate the raw data obtained from the touch panel. This method requires matrix operations and thus is more calculation heavy on the processor. This calibration routine has not been implemented but could lead to better calibration results. For further reading and reference see (Wendy Fang and Tony Chang 2007).
4.2.7. Firmware&API

To make the touch panel usable for simple implementation both in this and other projects the firmware has been developed with a small API that can be understood and used fairly easy. Below is a short description of the firmware structure and basic API.

4.2.7.1. Firmware

The touch-loop consists of three functions that are run in a loop. This touch loop pseudo code can be seen in Code Example 2.

```plaintext
while(1){
    //Parse the received byte (if present) and resolve what type of communication and data it is
    parse_byte(receivedByte);
    //Analyze the received data packet (if present) and set active touch event
    analyse_new_touch_data();
    //Call functions according to touch event type
    handle_touch();
}
```

**Code Example 2 - Touch loop**

**parse_byte()**

This function checks if there are any bytes in the rs232 receive queue, if there is it starts to analyze that byte. The function keeps track on which packet that is currently being received, how many bytes are left in that packet and some other parameters to ensure that the data communication never get out of sync. The information that is received is stored in specific data types to increase understandability and enable easier access. When a complete packet has been received this function recognizes this and sets the flag `touch_data_flag`.

**analyse_new_touch_data()**

This function runs when the `touch_data_flag` is set. First of all the packet type is determined (diagnostic or report) after that the information is analyzed. If it is a diagnostic packet nothing is done since the information there is irrelevant for the implementation. If it is a report packet the information is analyzed in order to determine which touch event that should be set and the coordinates for that event is saved. When the touch event type has been set the flag `touch_flag` is set.

**handle_touch()**

This function is only executed if the `touch_flag` is set indicating that a new touch event is active. The function then calls appropriate functions according to the current touch event.
4.2.7.2. API

All the information about the state of the touch panel and the current touch event are collected in a struct seen in Code Example 3.

```c
typedef struct
{
    uint16_t coords[2]; // touch coordinates in display coordinates (800x480)
    uint16_t panel_coords[2]; // touch coordinates in touch panel coordinates (2048x2048)
    uint8_t touch_flag; // indicates that a touch event has been triggered
    uint8_t touch_trace; // Activates or de-activates trace printouts
    uint8_t status; // Indicates the last touch events type (TOUCH/MOVE/RELEASE)
    uint8_t touch_data_flag; // New touch data is available
    uint8_t touch_calibration_flag; // The touch panel is in calibration state
    uint8_t calibration_step; // Current calibration step (LL/UR)
    uint16_t LLx, LLy, URx, URy; // Calibration constants
    // Touch interpretation thresholds
    uint8_t touch_time_interval; // Defined time threshold between touches
    uint8_t touch_distance_interval; // Defined distance threshold between touches
} Touch;
```

The variable values in this struct can be accessed with `get` and `set` functions. A breakdown of the most commonly used can be seen below.

**get_touch_status()**

Returns a pre-defined value (TOUCH/MOVE/RELEASE) that indicates what type of touch event that was triggered.

**uint16_t get_touch_coord(char axis)**

Returns the display coordinate of last touch event on a specified axis (x/y). This is the function that is used to see where the touch actually occurred.
4.3. Display Module

4.3.1. Hardware

The hardware used for the display implementation was just as the touch panels ordered through Multicomponent AB. The display itself is produced by WINSTAR Display Co., Ltd. and its module number is WF70ATIBGDA#. The display consists of two components: one LCD panel with connected panel driving PCB and one LCD controller mounted on a PCB. The microcontroller is only communicating with the LCD controller which in turn communicates with the LCD panel. This solution is widely used and in this case it is necessary because the memory on the Cortex M3 is limited. The memory on the LCD controller is enough to store one frame so in theory only one pixels data has to be stored on the microcontroller. More about the interfacing in section 4.3.3 Interfacing (FSMC).

The selection of hardware was made with focus on the parameters that were specified during the pre-study but in the end some compromises had to be made. It was important to get a good LCD controller that had good documentation and preferably a widely used one to make troubleshooting easier. It was also important that the display had an interface that was prepared with pinouts on the MidRange. The display size is 7” and was chosen because it is a fairly common size and suits the possible target implementations good. The resolution on the LCD is 800x480 which is quite high considering the microcontroller specifications.

4.3.2. Connecting Hardware

Connections between MidRange and the display are needed to provide power and enable communication to the display driver. The connections that are needed are voltage and ground for power and a 16-bit FSMC interface that requires in total 20 connection wires.

The display backlight also needs power and that is in this implementation provided by a DC power supply.

The connections are visualized in Figure 19.
The connections points on MidRange are pin connectors with 2,54 mm pitch and the connector on the LCD controller board is a flexible flatcable connector with pitch 0,5 mm. An adaptor PCB solves this problem by converting a 2x15 pin connector to a 30 pin flexible flat cable connector.

**Figure 19 - Physical connections to the display**
4.3.3. Interfacing (FSMC)

The protocol that is used to communicate with the LCD controller is Flexible Static Memory Controller (FSMC). This is in fact as the name indicates a protocol that is designed to interface different memory types, but it is also used in a lot of LCD controllers since it in many cases can be seen as a memory. The controller used in this implementation is handled just as a memory bank with some command features.

The flexible part in FSMC makes it able to interface many different memory types but also makes it more complex to set it up. The FSMC interface to this LCD controller is of intel-8080 type and the memory type is SRAM. The settings that can be made are memory timings and can be found in the LCD controller datasheet. The settings can be seen in Table 5.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Type</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{\text{Cyc}} )</td>
<td>Reference Clock Cycle Time</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{\text{PWCSL}} )</td>
<td>Pulse width CS# low</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>( t_{\text{Cyc}} )</td>
</tr>
<tr>
<td>( t_{\text{PWCSH}} )</td>
<td>Pulse width CD# high</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>( t_{\text{Cyc}} )</td>
</tr>
<tr>
<td>( t_{\text{FDRD}} )</td>
<td>First Read Data Delay</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>( t_{\text{Cyc}} )</td>
</tr>
<tr>
<td>( t_{\text{AS}} )</td>
<td>Address Setup Time</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{\text{AH}} )</td>
<td>Address Hold Time</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{\text{DSW}} )</td>
<td>Data Setup Time</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{\text{DHW}} )</td>
<td>Data Hold Time</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{\text{DSR}} )</td>
<td>Data Access Time</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>ns</td>
</tr>
<tr>
<td>( t_{\text{DHR}} )</td>
<td>Output Hold Time</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
</tbody>
</table>

Table 5 - FSMC intel-8080 timing

4.3.3.1. Display Communication

When the timings have been setup the communication to the LCD controller can start. The communication is divided into two types command and data. Commands are used to setup the LCD controller to function as indented and data is used to transfer data associated with commands and pixel data to the frame buffer. The way that the LCD controller separates between commands and data communication is by actively setting a pin on the LCD controller high/low.

Settings

During the setup function some important settings are needed so that the LCD controller communicates correctly with the panel. The settings are

- Resolution – Sets the resolution of the LCD panel
- Horizontal- and vertical period – Sets the period for vertical and horizontal sync.

---

(Systech 2008, 79)
- Pixel data format – sets in which format the pixel color data will be sent.
- Column address – sets start and end horizontal pixel
- Page address – sets start and end vertical pixel

These settings are made by first sending a command to the LCD controller followed by the data that should be set. These settings can be tricky to find since the documentation for LCD display is often very limited.

**Pixel painting**

When the setup of the LCD controller is correctly done, pixels can be controlled to display different graphics. The way that pixel information is sent depends on the chosen pixel format but they all follow the same principle.

The procedure starts with selecting the window that pixels should be painted in and activating memory writes, this is set with LCD controller commands. Then the actual pixel information can be sent. The data is sent according to selected pixel format and for each complete pixel information received the LCD controller will jump to the next pixel. This continues until the whole window is filled. Code to display a solid color on the display can be seen in Code Example 4.

```c
void lcd_draw_solid(uint32_t color){
    lcd_set_window(0, 799, 0, 479);
    lcd_send_command(LCD_CMD_WRITE_MEMORY_START);
    int x,y;

    for(y = 0; y <= 479; y++) {
        for(x = 0; x <= 799; x++) {
            send_pixel_data(color);
        }
    }
}
```

**Code Example 4 - Draw display in solid color**

By writing functions with different complexity different objects can be displayed on the display. The display API supports circles, rectangles, rings, crosses, text and bitmaps more about these in the API section.
4.3.4.  Firmware & API

4.3.4.1. Firmware

The firmware is the hardware specific code that handles setup of LCD controller and low level communication. The function names start with the name of the display to indicate that they are hardware specific. These functions are:

**WF70ATIBGDA_init(void)**

Runs all functions that initialize GPIO, FSMC interface and LCD controller using these functions:

- **WF70ATIBGDA_set_gpio()**
- **WF70ATIBGDA_set_fsmc()**
- **WF70ATIBGDA_reset()**

This function needs to be run before any other display function.

**WF70ATIBGDA_set_window(uint16_t sx, uint16_t ex, uint16_t sy, uint16_t ey)**

Sets the parameters on the window that is going to be filled with pixel data.

**WF70ATIBGDA_start_memory_write(void)**

Sets the LCD controller in receive pixel data mode.

**WF70ATIBGDA_send_command(uint8_t cmd)**

Sends a command(cmd) to the LCD controller.

**WF70ATIBGDA_send_data(uint8_t data)**

Sends data(data) to the LCD controller.

4.3.4.2. API

The display API is built as hardware independent as possible and includes basic functionality to draw different shapes, text and patterns on the display. The API also includes functions to communicate with an LCD controller on a higher level, access flags and a display update handler. The gain with this structure is that software that uses the display API does not have to be altered when new hardware is used.
4.3.4.2.1. Controller Communication

**lcd_init**(void)

Calls the hardware dependant initialization routine.

**lcd_set_window**(uint16_t sx, uint16_t ex, uint16_t sy, uint16_t ey)

Calls the hardware dependant function that sets the window to be filled with pixel data.

**lcd_start_memory_write**(void)

Calls the hardware dependant function that activates data writes to the controller.

**lcd_send_command**(uint8_t cmd)

Calls the hardware dependant function that sends commands to the controller.

**lcd_send_data**(uint8_t data)

Calls the hardware dependant function that sends data to the controller.

4.3.4.2.2. Update Handler

**update_display**(void)

Calls desired function when the *display_update_flag* is set.

4.3.4.2.3. Patterns

**lcd_draw_solid**(uint32_t color)

Draws the whole display in a selected color,

**lcd_draw_many_colors**(void)

Draws a demonstration screen of the color palette that the display can handle.

4.3.4.2.4. Text

**lcd_print_char**(char c, uint16_t x_offset, uint16_t y_offset, uint16_t transparent)

Prints a single char on coordinates specified. There is also an option to print the string with transparent background or not.

**lcd_print_char_size**(char c, uint16_t x_offset, uint16_t y_offset, uint16_t transparent, uint16_t size)

This function is similar to the *lcd_print_char* function but has the extra parameter *size* that specified the size of the char.
**lcd_print_string**(const char* s, uint16_t len, uint16_t x, uint16_t y, uint16_t transparent)

Prints a string starting on coordinates specified. The number of chars that should be printed is specified in *len*. There is also an option to print the string with transparent background or not.

**lcd_print_string_size**(const char* s, uint16_t len, uint16_t x, uint16_t y, uint16_t transparent, uint16_t size)

This function is similar to the *lcd_print_string* function but has the extra parameter *size* that selects the size of the chars in the string.

4.3.4.2.5. Image

**lcd_draw_image**(char * filename, char* directory, uint16_t x, uint16_t y)

Draws a BMP image on the display. The image will be printed on the coordinates specified and must be located in the specified directory on a SD-card connected to the MidRange.

4.3.4.2.6. Geometric Shapes

**lcd_draw_rectangle**(uint16_t sx, uint16_t ex, uint16_t sy, uint16_t ey, uint32_t c)

Draws a rectangle with top-left corner coordinates (sx, sy) and bottom-right corner coordinates (ex, ey). The color is selected with parameter *c*.

**lcd_draw_circle**(uint16_t c_x, uint16_t c_y, uint16_t c_r, uint32_t c)

Draws a circle with center coordinates (c_x, c_y) and radius c_r. The color is selected with parameter *c*.

**lcd_draw_ring**(uint16_t r_x, uint16_t r_y, uint16_t r_r, uint32_t c);

Draws a ring with center coordinates (r_x, r_y) and radius r_r. The color is selected with parameter *c*.

**lcd_draw_cross**(uint16_t c_x, uint16_t c_y, uint16_t c_w, uint16_t c_h, uint32_t c)

Draws a cross with center coordinates (c_x, c_y), width c_w and height c_h. The color is selected with parameter *c*.
4.4. Graphical User Interface Module

To simplify display and touch testing a graphical user interface module was developed. This software aids the programmer when building GUI’s. The following sections will explain the structure, API and how it can be used.

4.4.1. Structure

The GUI’s structure is built around `Gui_object`, a struct that can hold all information about a GUI component.

```
typedef struct
{
    uint8_t text_size, label, has_action, type;
    uint16_t x, y, height, width, radius;
    uint32_t color;
    char text [30];
    char file_name [30];
    char directory [30];
} Gui_object;
```

The GUI components that are needed for the GUI is created using this struct which can act as different types: rectangle, circle, ring, cross, point and image. The types represent the geometric shape of the GUI component. The type of a specific GUI component is specified in the variable `type`. This variable is then used when for example drawing components on the display with generic functions. With this structure some object oriented programming can be simulated even though C lacks support of that.

The settings that can be used for a GUI component are:

- Text size, the size of the text can be specified in the variable `size`.
- Size, the size of the GUI component is specified with `height`, `width` and `radius` depending on `type`.
- `label`, is a unique predefined identification number for each `gui_object`. This is used to easily access GUI objects.
- `has_action`, this variable should be `TRUE` if this component have an action connected to it i.e. if it is a button otherwise `FALSE`.
- `type`, this specifies which type of component this is circle, rectangle, cross, ring, point or image. This is set by using the defines `CIRCLE`, `RECTANGLE`, `CROSS`, `RING`, `POINT` and `IMAGE`.
- `x`, specifies the x offset from display origo in display coordinates.
- `y`, specifies the y offset from display origo in display coordinates.
- **color**, the GUI component color.
- **text**, the text (if any) that should be drawn on the GUI component. In this implementation the limit is 30 chars but this can be changed by increasing or decreasing the array size.
- **file_name & directory**, is used to specify the path to the bmp for the image GUI component.

The workflow when creating a new GUI is shown below:

1. Create the **Gui_objects** that is needed in the GUI and add them to the panel

   ```c
   gui_object.label = MENU_NUMPAD;
   gui_object.width = 200;
   gui_object.height= 75;
   gui_object.x = 300;
   gui_object.y = 50;
   gui_object.type =RECTANGLE;
   gui_object.has_action =TRUE;
   strcpy(gui_object.text, "Numpad");
   gui_object.text_size = 3;
   gui_object.color = 0x000FFF;
   add_gui_component_to_panel(gui_object);
   ```

2. Add the actions that you want your GUI components to have. This is done in **void action_performed**(*Action_event *e)*. In this function the action is connected to a specific GUI component by the label. Each component can have different actions depending on the touch event (TOUCH/MOVE/RELEASE).

   ```c
   void action_performed(Action_event *e){
      if(e->event_type == TOUCH){
         switch(e->o->label){
            case MENU_NUMPAD:
               //<<----- Menu button action here!
               break;
         }
      }
   }
   ```
3. Repeat 1 and 2 until all your GUI components are added.

4. This is what a simple GUI example can look like.

![GUI Example](image)

**Figure 20 - GUI example**

Now the GUI API can be used to handle input from the user and display the GUI to the user. See section nedan.

### 4.4.2. API

**Gui_object* get gui_component_label(uint8_t l)**

Function that returns a pointer to a by label specified GUI object. Can be used to get access to a GUI object and make changes to it.

**void draw_component_on_lcd(Gui_object *object)**

Draws the specified GUI object on the display. Can be used if just a single object needs to be updated.

**void draw_gui_components_on_lcd(void)**

Draws all the active GUI objects on the LCD. Used when drawing the GUI for the first time or for updating the display.

**void clear_all_components(void);**

Clears all the active GUI objects so that a new GUI can be specified.

**void add_gui_component_to_panel(Gui_object object)**

Adds a GUI object to the panel.

**int gui_action(uint16_t x, uint16_t y);**
Checks if there are any actions associated with coordinates \((x, y)\) and if so calls functions that executes them. This function can be used when new touch events are obtained to carry out the GUI functions for that specific coordinates.

**void component_action(Gui_object *object)**

Performs the action associated with the specified GUI object.

**int gui_inside(Gui_object *object, uint16_t x, uint16_t y)**

Returns *TRUE* if coordinates \((x, y)\) are inside the GUI object.

**void action_performed(Action_event *e)**

This function acts as an event handler that uses information in the *action_event* to determine what action that should be carried out. The information in the *action_event* is the *gui_object* that triggered the action and the type of touch event that was the source.
5. Results

The implementation has been tested in various ways to see how well the implementation works and how the actual hardware stands against the parameters that were rated in the technology evaluation (See 3.2 Technology Evaluation). Unfortunately the resistive touch-panel stopped working before the tests could be carried out so most of the touch panel test results will be on the capacitive touch-panel.

5.1. Environmental

The environmental parameters; humidity durability, dust, smudges and dirt durability, temperature durability and water durability were hard to test. It was challenging to stage the test environment, test procedure and how the result should be measured.

5.1.1. Humidity Durability

The humidity durability could not be tested since no humidity test equipment was available. The datasheets for the touch panels are very limited and give no information about durability against humidity. The Display datasheet specifies that the display can be used in up to 75% RH.

5.1.2. Dust, Smudges and Dirt Durability

This was tested by soiling the touch-panel with smudges, and placing dust and similar (from an unclean floor) on the touch panel.

The touch panel was then used and the accuracy and responsiveness was estimated. The results showed that the capacitive touch-panel did not get affected by these foreign substances. The touch panel behaved as usual but the visibility of the display was reduced since there were foreign objects in the way.

5.1.3. Temperature Durability

The durability in different temperatures could be tested for one extreme temperature, cold. This was easy because the temperature outside during this period was very low the test setup could be placed just inside an open window. The hot extreme temperature could not be tested since no equipment was available.

When the touch panel was chilled to a low temperature the performance was unaffected. The panel was just as responsive and accurate from a user perspective as it usually is in room temperature.
5.1.4. Water Durability
This aspect was very delicate to test since water and electronics don’t mix very well and the test setup was not enclosed in a protective enclosure. The test could be carried out by separating the touch panel from the rest of the electronics. The touch panel was then sprayed with water and used to test the functionality. The display durability could not be tested since it definitely would have destroyed the display. In real implementations water durability would most likely be achieved by building a good enclosure that prevents water from coming in contact with the electronics.

During the test there were no accuracy or responsiveness problems even when there were small pools of water on the panel.

5.1.5. Readability in Intense Sunlight
To test how the readability of the display is altered during intense sunlight an overhead-projector was used. The display was placed in the light from the overhead-projector.

First the whole system with both display and touch panel were tested. There were a lot of reflections on the touch panel that made the display unreadable in many angles. The touch panel was then removed and the display was easier to read but still hard to read in some angles.

5.2. Vandalism Durability
Testing the vandalism durability would end in destroying the hardware and since the hardware should be available after this project, more of an assessment has been made. While working with this hardware it has been obvious that it is durable to everyday use and misuse such as touching with foreign objects. But if someone wanted to damage the equipment it would be very easy. A simple punch would probably break the touch panel and the display. This means that the vandalism durability must be handled by enclosing the touch panel and display as good as possible.
5.3. Touch Accuracy Test

5.3.1. Setup
To test the accuracy of the touch-interface a series of circular targets were drawn on the display. The targets were placed in a strategic manner to cover as much as possible of the touch panel area without making the test too complex to carry out. The test screen can be seen in Figure 21.

![Figure 21 - Accuracy Test Screen](image)

The targets were labeled t1->t17 according to Figure 22 to enable better data interpretation.

![Figure 22 - Target labeling](image)

5.3.2. Execution
The test was done using two test persons who were told to touch at the location of the targets in a specific order starting at t1 then t2 and so on up to t17. The coordinates for the touches were saved. This test procedure was done with the touch interface at different angles to simulate different implementation assemblies. The data that was obtained is presented and analyzed in the next section.
5.3.3. Results

First the data was analyzed to see if the calibration increased the accuracy of the touch panel. In Figure 23 the absolute error between the recorded touch position and the actual position of the target is displayed. Based on the data collected the calibration significantly increases the accuracy. In some cases the difference in absolute error is over 10mm. Since the touch panel is positioned by hand on top of the display the alignment is not that good and this is a factor in the large differences. This error could be lowered by assembling the touch panel and display very precise. But this also shows that if a calibration routine is used the alignment can be done with less precision.
The data was then analyzed to see if there were any accuracy changes on different areas of the touch panel. The result shows no significant changes in accuracy, see Figure 24. This data can be somewhat misleading since at the very edges and about 1-2mm in on the panel the accuracy is very bad, this is based on experience from use during development. The targets used in this test was placed outside these regions for more usable data but this inaccuracy must be taken into account when designing GUI’s.
The test included different assembly angles and in Figure 25 the mean absolute error for all targets at different angles can be seen. The data indicates that it is easier to do accurate touches if the panel is assembled at 90°. This can be explained by optical differences with different viewing angles. Since there is a gap between the touch panel and the display you don’t actually touch the touch panel directly above the target when viewing it at angle, see Figure 26. This might be the source of the difference but it can also be that there is more comfortable to use the touch panel at an angle and by that increasing the accuracy.
To understand how large a target (i.e. button, GUI component) has to be to ensure that when a user tries to touch a target it is recognized as a hit the absolute errors were analyzed in relation to target sizes. The data that is visualized in Figure 27 shows the percentage of all the touches made that would have hit a circular target with the specified radius. First of all it can be seen that the hit percentage is greatly increased when the touch panel is calibrated, this was not a surprise but the results from the not calibrated panel can be used if for example an implementation makes it impossible to have a calibration routine. The calibrated panel’s results are quite good and show that a circular target with radius 11 mm is hit 100% of the touches. This can then be used as a pointer for GUI making.
5.4. Display

5.4.1. Panel Performance
The display has been tested by drawing different patterns to display color palette and resolution. These tests have been made as a quick and easy way to evaluate different displays against each other. These test results are hard to measure and is better used for comparison to other displays in a test setup. See Figure 28 and Figure 29 for examples of test screens.

Figure 28 - Chessboard

Figure 29 - Color representation
5.4.2. Refresh Rate
The refresh rate of the display is good but the user experienced refresh rate is low. This is because the display controller draws the frame buffer on screen at a much higher rate than the microcontroller fills the frame buffer with new data. The result of this is that the sweep can be seen on screen. This makes the interface look a bit low quality but can be helped with clever GUI programming.

To fill the whole frame buffer with a simple pattern (solid color) is fastest since it requires minimal computation. Drawing more complex patterns such as images or GUIs is slower since the amount of computations that are needed to compute which pixels to paint and which not is much larger.

By clever GUI building and only updating small sections of the display at a time the user perceived refresh can be increased. Another solution to this problem is to use a display controller that has a double frame buffer that can be filled with the next frame data and then switch quickly between them.
6. Discussion

The process of choosing and implementing a touch interface on the MidRange platform has been much more than just hardware interfacing. When initial tests were made to the touch panel it was obvious that what we call a touch on a touch panel is so much more than just coordinates. The touch has to be interpreted in the way that the user assumes it will be. This can for example be double click, gestures etc.

More work can be done on this interpretation part to make the touch panel more functional.

When choosing which display to use the technology is important but more important factors are:

- Does the display have an attached controller PCB? How do you interface this?
- Which display interfaces are available on the platform you are using?
- What are the physical connections needed to connect to the display?

These factors are very important to check before ordering. This can be a problem because the documentation that is obtainable from the manufacturer is often very limited.

The display that was used in the implementation had very good specifications and delivered very good results. One problem with the display was the high resolution. The cortex-M3 processor was in a way to weak to handle it but in another it was not. Updating the whole frame buffer takes about 3 seconds and during that time a sweep can be seen on the display. This can be avoided by updating small areas of the display. This requires a more complex GUI-handler and the result is probably a less intuitive GUI handler. This will have to be evaluated considering the future implementations, is high resolution a requirement? If not, choose a display with lower resolution for easier implementation.

The GUI software that was developed in this project is restricted in the number of GUI components but dynamic in the handling of these. This makes it possible to develop simple GUIs with buttons of different shapes quite easily. The structure also makes it easy to extend the software with new types of GUI objects.
7. References


0Whitepaper.pdf (använd den 07 March 2011).


Pearson Education. “Pearson education.”


### TOUCH

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rating</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost</strong></td>
<td>4</td>
<td>Always an important parameter but is relative to the performance i.e. if a cost increase gives far better performance it is defendable.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>3</td>
<td>i.e. how large touch surface can the technology support.</td>
</tr>
<tr>
<td><strong>Resolution(touch accuracy)</strong></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><strong>Multi touch</strong></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Usability with stylus</strong></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td><strong>Usability with gloves</strong></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Humidity durability</strong></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>Dust, smudges, dirt durability</strong></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>Transparency</strong></td>
<td>3</td>
<td>Not important if small effect, very important if large effect.</td>
</tr>
<tr>
<td><strong>Lifespan</strong></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Low temperature durability</strong></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>High temperature durability</strong></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>Power consumption</strong></td>
<td>3</td>
<td>Depends on if the device is mobile or stationary i.e. battery driven or not.</td>
</tr>
<tr>
<td><strong>Responsiveness</strong></td>
<td>3</td>
<td>Since the MidRange platform is “slow” the response time only needs to be faster than that.</td>
</tr>
<tr>
<td><strong>Water durability</strong></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>Vandalism durability</strong></td>
<td>4</td>
<td>Vandalism in the term of daily-use and daily-misuse. Not intended brutal vandalism.*</td>
</tr>
</tbody>
</table>

*Better encapsulation can increase vandalism durability but that might affect the functionality, so a weighing has to be made between durability and functionality.

### DISPLAY

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Betyg</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Rating</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Size</td>
<td>1</td>
<td>Touch will set the limit.</td>
</tr>
<tr>
<td>Resolution</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Angle of vision</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Power consumption</td>
<td>3</td>
<td>Depends on if the device is mobile or stationary i.e. battery driven or not.</td>
</tr>
<tr>
<td>Brightness</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Readability in intense sunlight</td>
<td>5</td>
<td>Look at transflective displays.</td>
</tr>
<tr>
<td>Low temperature durability</td>
<td>5</td>
<td>Same as for touch</td>
</tr>
<tr>
<td>High temperature durability</td>
<td>5</td>
<td>Same as for touch</td>
</tr>
<tr>
<td>Humidity durability</td>
<td>5</td>
<td>Same as for touch</td>
</tr>
<tr>
<td>Vandalism durability</td>
<td>4</td>
<td>See note *</td>
</tr>
<tr>
<td>Color depth</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Response time</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix B  Technology weighting

<table>
<thead>
<tr>
<th>Capacitive(Projected)</th>
<th>Product</th>
<th>Capacitive(Surface)</th>
<th>Prod.</th>
<th>Resistive</th>
<th>Prod.</th>
<th>SAW</th>
<th>Prod.</th>
<th>IR</th>
<th>Prod.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>4</td>
<td>12</td>
<td>5</td>
<td>15</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>12</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>15</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>12</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>15</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>20</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>20</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>4</td>
<td>16</td>
<td>3</td>
<td>12</td>
<td>5</td>
<td>20</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>4</td>
<td>20</td>
<td>3</td>
<td>15</td>
<td>5</td>
<td>25</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>25</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>4</td>
<td>20</td>
<td>5</td>
<td>25</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>5</td>
<td>25</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>25</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>54</td>
<td>203</td>
<td>48</td>
<td>193</td>
<td>59</td>
<td>221</td>
<td>49</td>
<td>193</td>
<td>52</td>
<td>201</td>
</tr>
<tr>
<td>LCD(TN)</td>
<td>Product</td>
<td>LCD(IPS)</td>
<td>Product</td>
<td>OLED</td>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>----------</td>
<td>---------</td>
<td>------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>12</td>
<td>5</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>4</td>
<td>16</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>3</td>
<td>12</td>
<td>3</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>5</td>
<td>25</td>
<td>5</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>4</td>
<td>20</td>
<td>5</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>5</td>
<td>25</td>
<td>5</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>2</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

46 175 46 178 44 177