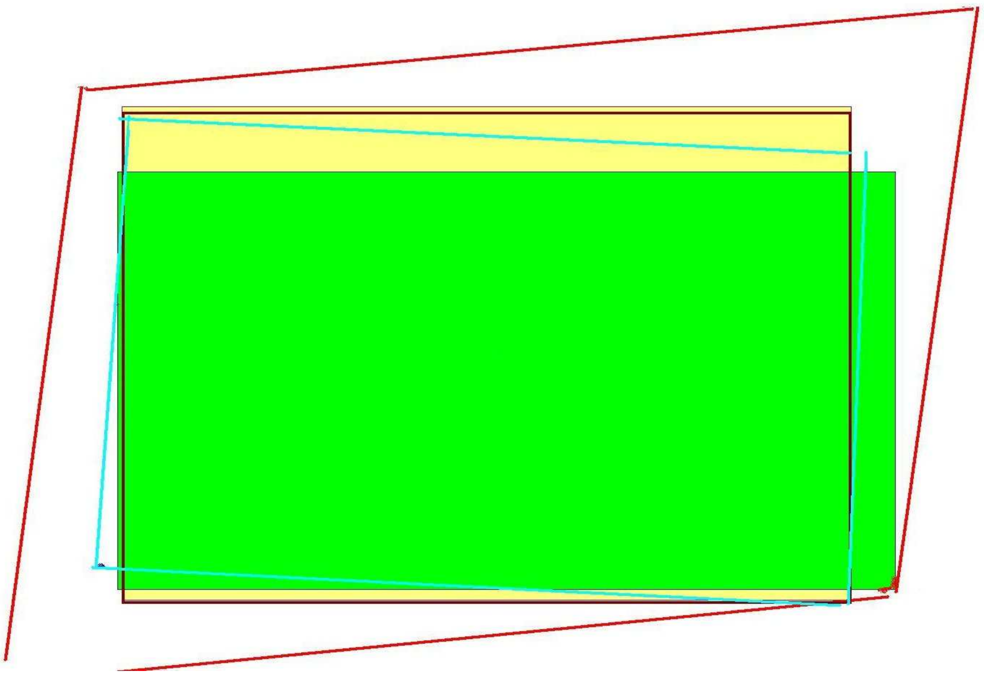
CHALMERS



Universal controller for resistive touch screens

Master of Science Thesis in the Programme Integrated Electronic SystemDesign

YASIN NILS TAIB

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, August 2010

Universal controller for resistive touch screens

YASIN NILS TAIB

Examiner: Rolf Snedsböl

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: Example of square drawn under different calibrations.

Department of Computer Science and Engineering
Göteborg, Sweden August 2010

Abstract

This report describes the development of a universal touch screen controller to be used with a 8-wire resistive touch panels and function as input device for the overlaying system. The controller is implemented in a microcontroller and designed to function with resistive touch screens of different sizes and from different vendors. The controller enables the determination of the touch status of the screen as well as the touchpoints X-Y position. The controller is designed to function in equipments used in rugged environments. Connection to the overlaying host system is provided through a USB connection. It is adapted to the HID protocol which gives the benefit that the drivers needed are provided in most major operating systems. Because it is common for touch screen sensor panels to be misaligned and suffer from problems with linearity, a calibration software for Windows that communicates with the controller has been developed. The resulting prototype is a 8-wire touch controller that utilizes a robust measuring technique in order to provide accurate and calibrated coordinates even in harsh environments.

Summary

This thesis is based on the work performed at Saab Aerotech in 2009. The aim was to develop a universal touch controller that would function with a wide range of different resistive touch panels and function as an input device for the host system. The project goals was divided in to two parts, first to develop a circuit that can communicate with a specific resistive touch panel, as well as provide calibrated data through a USB connection to control the cursor in a Windows environment. Then if these requirements were meet the project should be extended to add features to the controller so it will work with touch panels from different manufacturers and with different sizes. It should also be provided drivers with the ability of calibration in a Windows environment. The prototype was developed uses a microcontroller PIC18 from Microchip and utilizes USB to communicate with the host system. It is implemented as a HID-mouse for easy detection by the host system and making custom drivers redundant. It is designed for use with 8-wire touch panels in different sizes and from different manufacturers. The provided Windows software make it possible to calibrate the touch controller fixing eventual problems with misalignment, scaling and linearity. The comptroller utilizes a robust measuring technique in order to provide accurate and calibrated coordinates even in harsh environments. The final result consists of a touch controller tested with a 15” and 13” touch panels from Mildex Optrical Inc. The prototype was developed in accordance with the initial goals and delivered to Saab in addition to a technical report.

Acknowledgements

I would like to thank everyone at Saab Aerotech, Logtronics for making me feel welcome and my supervisor Johan Ekström for giving me the opportunity to do my thesis at Saab Aerotech. I would also like to thank Joachim Booberg for the nice company and for answering my relevant as well as irrelevant questions. I would also like to thank my examiner at Chalmers for supporting me throughout the work on this project.

Contents

1 Introduction

This master thesis was written as part of the final master project for the program Integrated electronic system design at Chalmers University of Technology. It was preformed at Saab Aerotech division Logtronics in Malmslätt Linköping. The aim of the thesis is to develop a universal touch screen controller to function as the interface between a wide range of touch-screen sensor panels and the cursor of the host system.

1.1 Background

The importance of personal computers is increasing everyday. The defense industry is no exception to this, but rather the opposite. Saab Aerotech division Logtronics develops computers, video systems and displays for extreme environments with applications mainly towards the defense industry but also towards the civil area. These systems can have many different functions and purposes and are often customized by the end customer. Examples of such a systems are the rugged computers and displays developed for BAE Systems Hägglunds vehicle SEP (Splitterskyddad Enhetsplattform) were they are used to monitor the vehicles systems. The main property these different systems have in common is that they are designed to be resistant to outer conditions such as heat, cold, damp, dust etc. By equipping the displays and computers with touch screen a simple and user-friendly way of providing I/O to the systems is achieved. This is often essential considering the type of environments that they are intended to be used in, which often do not allow the space for external input devices. These displays come in several different sizes and the touch panels used are usually supplied with their own touch controller and therefore the surrounding circuit also has to be changed. A preferred solution would be to be able to use a universal touch screen controller for all types of touch panels. It is with this background in mind that the demand for a universal touch controller would be beneficial.

1.2 Aim

The aim of this thesis is to develop a universal touch screen controller for a wide range of resistive touch panels with different sizes and from different manufacturers. The goals are divided into two parts, the main goals that is the primary aim of the project and the optional goals that are an extension of the main once. The controller shall make measurements of the touch panel and provide the overlying host system with calibrated measurements through a USB (Universal Serial Bus) connection. The controller shall function as a I/O device for the Operating system and control the cursors by sending it absolute coordinates. The intentions are that Windows shall detect the controller as a HID-device (Human Interaction Device) and have access to appropriate drivers.

- **Main goals:** The main goal of this project is to develop a circuit that can communicate with a specific resistive touch panel, as well as provide calibrated data through a USB connection to control the cursor in a Windows environment.

- **Optional goals:** If the main goals are completed the project should be extended to add features to the controller so it will work with touch panels from different manufacturers and different sizes. It should also be provided drivers with the ability of calibration in a Windows environment and have a serial (RS-232) connection to the system.

2 Theory

Touch screens are becoming a more and more utilized technique for providing I/O to computer-based systems [2]. One main reason for this is the fact that mobile devices are shrinking in size and there is no longer space for a full size keyboards or external cursors. Another reason is that touch screens offer a user-friendly and easy to use interface. There are several different techniques for implementing touch screens, the four main ones are [2]:

- Surface acoustic wave
- Scanning infrared
- Capacitive
- Resistive

Surface acoustic wave screens send ultrasonic sounds over the touch panel and when it is pressed some of this signal is absorbed. The change in the waves is used to calculate the position. These types of screens are amongst the most advanced types, they provide a very high resolution and because they only consist of one glass layer they also have a very high clarity. But because they are not completely seal able, they are not completely tolerant to outer conditions such as dust and water and therefore not suitable for rugged environments [2].

The scanning infrared technique utilize an array of light emitting infrared diodes on two sides and two other arrays of phototransistors on the opposite sides. The LED's are sequentially emitted to form a grid of IR-light and when the light is obstructed the phototransistors detect an absence of light and the coordinates can be obtained. The design is very robust and resistant to noise but it is limited to small resolutions [2].

Capacitive touch screens consist of a glass panel coated with a charge storing layer. When the screen is pressed with a conductive material such as a finger or a stylus the surrounding circuitry located in the corners measure the current flow. The size of the current flows are then used to calculate the coordinates for where the screen i touched. Capacitive screen have a very high clarity, up to 90 percent but the drawback is that it is not possible to use them with gloves or a non conductive stylus [2].

Resistive touch panels consist of two conductive layers separated from each other. When the panel is pressed the two layers conduct and the coordinates are given by measuring the resistance. Restive touch screens are very robust and fault tolerant but have a very low clarity, some times as low as 75 percent.

Because capacitive and resistive are more tolerant to outer conditions they are the more commonly used techniques in rugged environments. Although resistive screens have inferior clarity, they are still more common for these purposes because they can be used without a conductive stylus [2]. Trade-off has to be made between clarity and the ability to operate the screen with for example gloves.

Saab has chosen to use resistive touch panels for there screens, mainly because they are robust which is crucial in there type of applications. They are also cheaper and have simpler support circuitry [2].

2.1 Resistive touch screens

A basic touch screen system consists of three main parts (figure 2.1).

- Touch panel
- Touch controller
- Host system driver

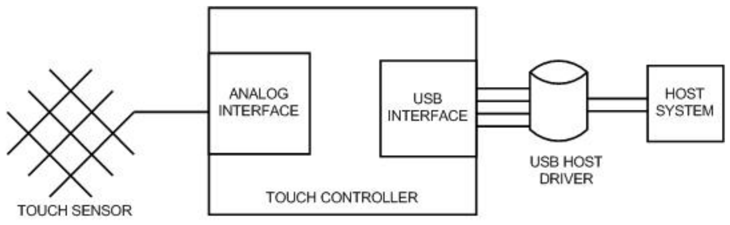
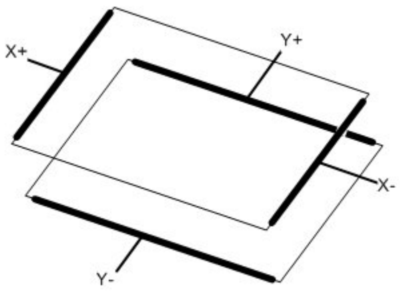


Figure 2.1: Schematic drawing of a resistive touch screen system.

2.1.1 4-wire touch panel

A resistive touch screen sensor panel (often referred to as the touch glass) consists of two conductive layers separated with insulating spacers. Each layer has a fixed resistance defined during fabrication and in the ideal case the resistance between the two layers is infinite. When pressure is applied to one of the layers the two layers connect and by measuring the resistance between the two layers the coordinate of were the touch panel is pressed can be calculated. This is done by applying a known voltage over one of the layers and then measure the voltage over the second layer. The voltage difference corresponds to the coordinate where the touch panel has been pressed in one dimension. When in its idle state the voltage measured over one layer is zero.



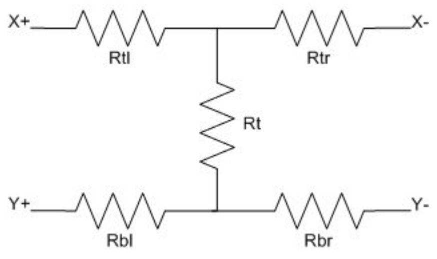


Figure 2.2: a) Illustration of 4-wire touch panel. b) Schematic drawing of Touch panel. [8]

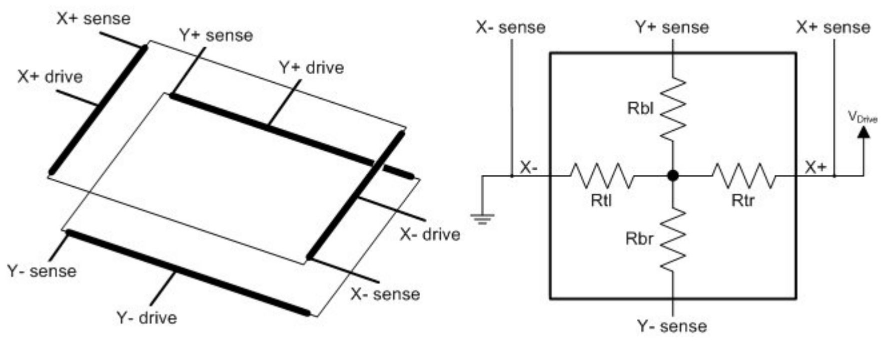
For a 4-wire touch panel this is done twice, once to obtain the X-coordinate and once for Y-coordinate. A equivalent schematic of touch panel can be found in figure 2.2b where X+, X- correspond to the plus and minus pole of X-layer and Y+, Y- correspond to the plus and minus pole of the Y-layer. R_T illustrates the resistance between the two layers and its value is dependent of were the panel is pressed, $R_{tl} + R_{tr}$ form the total resistance in the top layer and $R_{bl} + R_{br}$ for the bottom layer.

2.1.2 8-wire touch panel

A 8-wire touch panel functions in a similar fashion as to a 4-wire. The main difference is that there is one extra wire to each of the nodes (X+, X-, Y+, Y-) giving a total of 8-wires going to the AD of the touch controller. The additional wires are used to be able to subtract the wire resistance from the measured values [9] or offset voltages introduced by the wiring or drive circuitry. This is important because the wire resistance changes with temperature and can therefore lead to corrupted data if the circuit is exposed to environments with large temperature differences. The common error is that the touch controller often needs to be re-calibrated.

The 8-wire touch panel have four drive wires and four sense wires, the drive wires are used to apply the drive voltage to the touch panel. When the voltage is applied the sense wires are used to measure the voltages. First the maximum value for each dimension is determined followed by the minimum value. This is done by applying the drive voltage on one end while the other end is driven low. The sense wire corresponding to the side driven high is measured to obtain the maximum value (V_{XYmax}) for that dimension, while the other sense is measured to obtained the (V_{XYmin}). Figure 2.3 show the design of a 8-wire touch panel with the added wire for Sense and drive.

A coordinate measurement of 8-wire touch panel is done by driving the



t]

Figure 2.3: Schematic drawing of 8-wire touch panel. [14]

$Y{+}_{drive}$ high and driving the $Y{-}_{drive}$ low and then measuring the voltage on the $X{+}_{sense}$ wire. To compensate for eventual voltage offsets equation 2.1 and 2.2 to recalculate the obtained value to adapt to the V_{Xmax}, V_{Ymax} and V_{Xmin}, V_{Ymin} values. The measured value is normalized using V_{Xmin}, V_{Ymin} and then divided by the$V_{Xmax}, V_{Ymax} - V_{Xmin}, V_{Ymin}$ to get a ratio between the measured value and the maximum value. This is then multiplied with $2^{AD-resolution}$ to normalize the value with respect to the AD-converter. The resulting values $X_{coordinate}, Y_{coordinate}$ are the coordinates of the touch point.

$$X_{coordinate} = \frac{V_{x+} - V_{Ymin}}{V_{Ymax} - V_{Ymin}} \times 2^{AD-resolution} \tag{2.1}$$

$$Y_{coordinate} = \frac{V_{y+} - V_{Xmin}}{V_{Xmax} - V_{Xmin}} \times 2^{AD-resolution} \tag{2.2}$$

2.1.3 Touch controller

As the touch panel only consists of discrete components additional circuitry needs to be added in order to perform the previously described measurements. The main part of this is the touch controller that acts as an interface between the touch panel and the host system. The touch controller is usually implemented via an integrated circuit dedicated to this function or by a microcontroller. In this thesis the touch controller is implemented using a microcontroller from Microchip (section 2.2). The touch controller needs to possess two essential properties. The first one is that it needs to be equipped with several A/D converters (or one multiplexed) in order to convert the analog signals from the touch panel. The second one is the ability to provide a connection to the host system supplying it with the measured data. This connection is often established through a RS-232 or USB connection (more on this topic in section 2.3).

2.1.4 Host system driver

The host driver is used by the host system to communicate with the touch controller. The way that the host driver is implemented can differ depending on what system it is implemented for and in some cases be redundant. The host driver implemented in this paper utilizes the USB-HID Mouse driver that is built in most systems along with a custom made software used to configure the touch controller. The drivers is implemented in such a way that the operating system will find it and treat it as a standard HID-mouse.

2.2 Microcontroller uC

The touch controller will be implemented using a Microchip PIC18F4550 microcontroller programed in C. This circuit has been used in previous designs and has been proven adequate for touch screen applications [10]. In addition to this it also has an integrated USB controller hence there is no need for additional circuits to provide a USB connection. The PIC18F4550 also has built in 10bit AD-converters which is a crucial feature the touch panel measurements. This uC also fulfills the requirements for industrial usage of operating in extreme temperatures [1].

2.3 USB

USB has become one of the most common ways of connecting peripherals to a computer based system. By using the USB-HID device class much of the work of developing drivers is already provided [5]. HID class is used mainly for I/O devices such as mice, keyboards and game pads, but it can be used for any hardware as long as it can function under the HID limitations. This means that if a peripheral is developed that is detected as for example a mouse, the drivers are already provided in many of the major operating systems. This dose not only mean that the time consuming work of developing a driver is already done it also means that multiple OS support can easily be achieved.

2.4 Calibration

Because the touch panel is separated from the screen and most often ends up being out of alignment or/and misscaled when mounted and can also suffer from problems of inlinearity, the coordinates of the touch panel and the coordinates of the screen can be mismatched. As a result the input coordinate would be misinterpreted by the host system. To avoid this most touch screens need to be calibrated before they can be properly used. A common way of calibrating is using linear interpolation [6]. The calibration procedure uses three or more predefined points where the coordinates are known, along with the same amount of user inputted coordinates intend to correspond to the predefined points. For larger touchscreens it is more common to used five reference points [11]. By using linear interpolation and the obtained coordinates it is possible to calculate the coefficients in equation 2.3 and 2.4. These new equations then represent an approximate fit of the mismatch between the absolute and the obtained values and can then be used to recalculate future inputs. In equation 2.3 and 2.4 X and Y represent the coordinates where the screen is pressed. A, B, C, D, E and F are calculated using the user inputted values during the calibration procedure. $X_{Calibrated}$ and $Y_{Calibrated}$ are the calibrated values.

$$X_{Calibrated} = AX_{coordinate} + BY_{coordinate} + C \tag{2.3}$$

$$Y_{Calibrated} = DX_{coordinate} + EY_{coordinate} + F \tag{2.4}$$

3 Implementation

3.1 Touch screen sensor panel

The touch panels are manufactured and can be viewed as a pure resistive component. This also gives them similar behavior such as temperature dependencies and a possible drift in resistance size. Because touch screen sensor panels come in several different sizes they also have different resistances, a typical resistances value for the reference panel used in this project are $500\Omega - 1000\Omega$ [12]. The way the touch panels are designed with two plates close to each other results in that the panels get capacitive features. This gives the panels a charging time and will result in faulty data if samples are done before the panel is fully charged. This is avoided by adding a small delay between that the panel is pressed and when the coordinate measurements are made.

3.2 X-Y positioning

Because the displays will be used in rugged environments and can be exposed to large temperature differences, 8-wire touch panels are used. 8-wire technology uses four wires to measure the wire resistance so that they can be subtracted from the calculations used to determining the coordinates of the touch point. This is desirable as resistance is temperature dependent and by removing it from the coordinate calculations it decreases the errors in the read coordinates.

By measuring the voltage induced over the sensor plates and switch the poles of the applied voltage information of the maximum value can be calculated. Equation 3.1 and 3.2 show how to calculate the X and Y coordinates. X_{plus} represents the value obtained with the plus plate as reference and X_{minus} the value with the minus plate as reference. By calculating how large X_{plus} is in comparison with the difference between X_{minus} a calculation of where the screen has been pressed is obtained. This is then multiplied with the AD-resolution to get a value to send to the host system. This small conversion is very useful because the MIN and MAX values of the touch panel do not need to be known. It also gives the benefit that different panels with different resistances can be used. Because the touch screen coordinate measurements and calculation tend to be relatively time demanding, it is desired only to do this as long as the screen is touched. The status of the panel is pulled repeatedly to check if it is pressed or not. Y+ and Y- are set high and X+ X- are measured and compared with the threshold value.

$$X = \frac{X_{plus}}{X_{plus} + X_{minus}} \times 2^{AD-resolution} \tag{3.1}$$

$$Y = \frac{Y_{plus}}{Y_{plus} + Y_{minus}} \times 2^{AD-resolution} \tag{3.2}$$

3.2.1 Filtering

In an ideal environment the methods described in 3.2 would be sufficient, but in order to provide a robust solution the data obtained from the panel needs to be filtered. This is done both in hardware and software. The hardware filters are used to remove the most sever noise, and then software filter is used to ensure reliable data.

In addition to the outer noise the AD converters of the uC suffers from minor problem of leakage. Therefore the obtained values some times tend to drift. If this data would be passed to the host system it would appear as random cursor movement and it is therefor crucial to remove the drift. Because the sample speed (35kHz) is much higher than the speed of which the operator is moving the stylus, it is possible to take several samples of the same touch. The random noise will appear as maximum/minimum extremes in the sample data. "Mean" filtering will remove these extremes and result in more accurate data.

An important problem that is introduced by the human factor is that when the user presses the panel the pressure point and the pressure will not be totally stable. As a result the cursor of the host system will appear to move randomly around the desired point. This error is removed by "average" filtering the sampled data and will result in a smooth behavior of the host system cursor (figure 4.1).

3.3 uC configuration

The touch screen sensor panels X and Y plates are connected to the micro controllers AD converters. The flow chart in figure 3.5 show how a X and Y measurement is conducted. The PIC utilizes one AD-channels that is MUXed into eight different inputs to conduct the X-Y measurement. They are configured using the AD-control registers 0-2 (ADCON0-ADCON2). Each AD-channel is 10-bits wide and stores the converted data in register ADRESH and ADRESL

3.4 USB-Device

When a USB-device is connected to the bus it will be enumerated by the host system. During the enumeration process the devices is assigned an address, the descriptors are read from the device and the appropriate drivers are loaded. After the enumeration process the device is configured and the host is ready to use any of its endpoints.

3.4.1 Descriptors

All USB-devices have a hierarchy descriptors containing information of both the USB-device as a whole and information of elements in the devices. These descriptors need to be stored in the device and sent to the host when requested. The USB-HID device designed in this project is configured using five different descriptors, and they are implemented in the code in Appendix B.

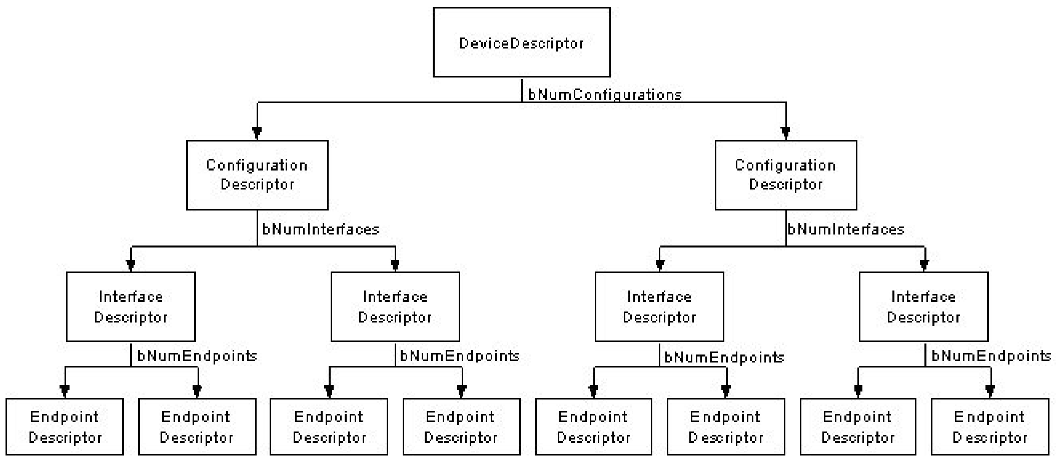


Figure 3.1: USB Descriptors.

- **Device Descriptor**. The Device Descriptor is the first descriptor that is read and it contains information of the device such as Vendor and Product ID. This information is necessary for the operating system when its to find and load the appropriate drivers [5].
- **Configuration Descriptor**. After the retrieval of the Device Descriptor the Configuration Descriptor is exchanged. A device can have several different Configuration Descriptor although the common thing is that it has one. It defines the way the devices is powered and the

number of interfaces. Because it is above the Interface Descriptor in the Descriptor hierarchy it is also possible to have a two configurations defining two different transfer modes.

- **Interface Descriptor** It describes the function or feature that the implements. It contains class, subclass and protocol information and the number of endpoints the interface uses.

- **HID Class-Specific Descriptor** For every endpoint declared in the interface descriptor has to have a endpoint descriptor, except endpoint 0 as this always has to be supported. The endpoints main task is to define the maximum packet size.

- **HID Descriptor** The HID descriptor defines the devices data packets (reports), how large, how many and the purpose of each byte in the packet.

3.4.2 HID-Reports

The USB-device is implemented as a HID-Mouse with an additional feature report used during calibration. The code provided in Appendix B show how this is implemented. The input report sends information from the controller to the host system and has a size of 5 bytes (figure 3.2). It contain two bytes of data for the X-coordinate, two bytes for the Y-coordinate and one byte for the three mouse buttons. In addition to this there is also a "feature report" that also has a size of two bytes. The feature report it is used to send different codes to notify the controller when the user executes a "requests to calibrate". The feature reports register is polled continuously to check if a "request to calibrate" has arrived.

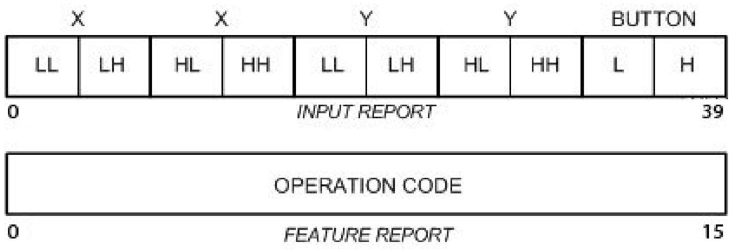


Figure 3.2: Figure 3.2: HID reports.

3.5 Calibration

The calibration procedure was discussed earlier in section 2.4, it describes the basic principles of how to go about in order to calibrate the touch screen. Although this is a commonly used method it is usually elaborated in order to get a more accurate calibration. By using more points as reference more data over the characteristics of the touch panel is gathered. The equation introduced in 2.4 is a general equation that can be modified to be used with the desired numbers of calibration points and is derived from the equations in Appendix A. The default value for this design is five points as this is the number commonly used [11]. For larger touch panels that suffer from a more sever problem with linearity it can easily be increased to use more calibration points.

$$X_{Calibrated} = AX_{coordinate} + BY_{coordinate} + C \tag{3.3}$$

$$Y_{Calibrated} = DX_{coordinate} + EY_{coordinate} + F \tag{3.4}$$

X and Y are predefined calibration points and a typical placement for these during a 5-point measurements is one in the center of the screen and the other four 20 percent from each corner (figure 3.3). Y_D and X_D are values inputted by the user during calibration, each one corresponding to a X and Y value [11].

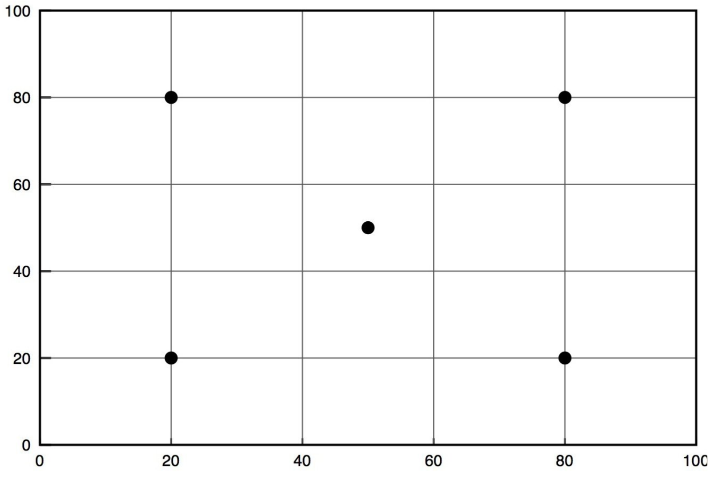


Figure 3.3: 5-point calibration layout.

3.6 Design

The system i portioned into four main parts, one that checks if the touch screen is pressed, one for reading the coordinates, one handling the calibration and one for sending the data to and from the host system. These are show in the flow-chart in figure 3.4.

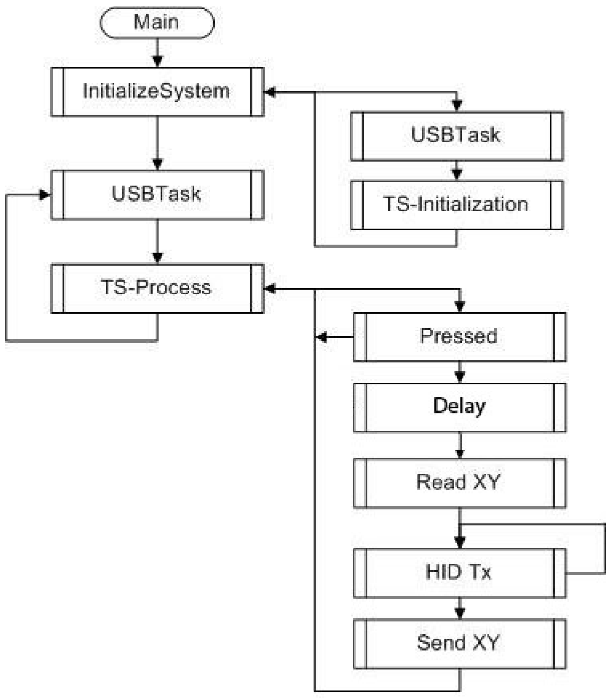


Figure 3.4: Function flow chart

3.6.1 Pressed

Because is not desired to neither send information or make coordinate measurements if the screen is not pressed the function "pressed" is used to check if the touchscreeen is being pressed. It continuously asserts X+ and X- and measures Y+, and if the Y+ is greater then the threshold value the screen is being pressed and the function returns 1. Otherwise it returns 0. When the touch panel is in its idle state this is the only function called on.

3.6.2 Read X-Y

The most important feature is the ability to obtain the coordinate from the touch panel. "Read XY" initializes the panel by asserting the necessary

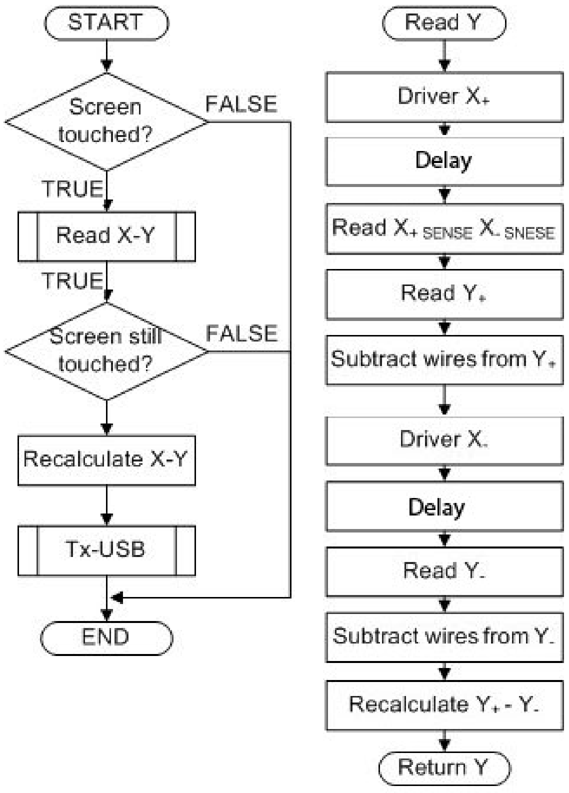


Figure 3.5: Flow chart of X-Y measurement

supply voltages read the coordinates. As described in the section 2.4 the measured values have to be filtered before used to be sure of correct data. This function returns the measured coordinate and it is only called if the panel has been pressed. Figure 3.5 shows a flow chart of how it is operated.

3.6.3 Send X-Y

Send XY is used to transmit the coordinate to the host system. After each time a coordinate is obtained the values are converted to fit in the HID input report and the device checks if the USB channel is free. If not it waits until it can send the report, this being of importance in order to be sure that ”mouse clicks” reach the host system. Every time data is sent it will check if the panel is active or not, if it is ideal state it will send a ”mouse release” report. The ”mouse release” report is the same as a regular input report contain the X-Y coordinates but instead of telling the host system that the mouse button is pressed it tells it to release it.

3.6.4 Calibration software

This calibration software is designed for Windows to be used with the calibration function implemented in the uC. It consist of a simple program that finds an attached USB device that corresponds to a specific product ID. When it detects it it sends a feature report containing an operation associated with the calibration. Then it guides the user in entering the desired references points by displaying them in sequence on the screen. The collecting part is done by the uC and after all reference points are collected it exits.

4 Results

During the course of this project a prototype of the design described in this thesis has been built. Although additional work is needed before it can replace an "off the shell" product, it provides a very good platform for future work.

- The touch controller has been designed with a 15" touch panel from Mildex Optical Inc as reference [12] and has been proved to work properly. It has also been tested with the 13" version of the same touch panel which showed satisfactory results. Test with the 13" panel showed some problems with nonlinearity towards the edges, but this issue is removed after calibration (which is viewed as the normal operating mode). Figure 4.1 shows a 15" touch panel being pressed, and how one set of coordinates (X and Y) is obtained. They gray points represent each sample and the white point is the resulting filtered data. It shows how the correct value is obtained although the measurements are heavily contaminated. One main reason for the noisy measurements is that the operator does not keep a constant pressure in a constant position. In this case the desired point is 556,192.

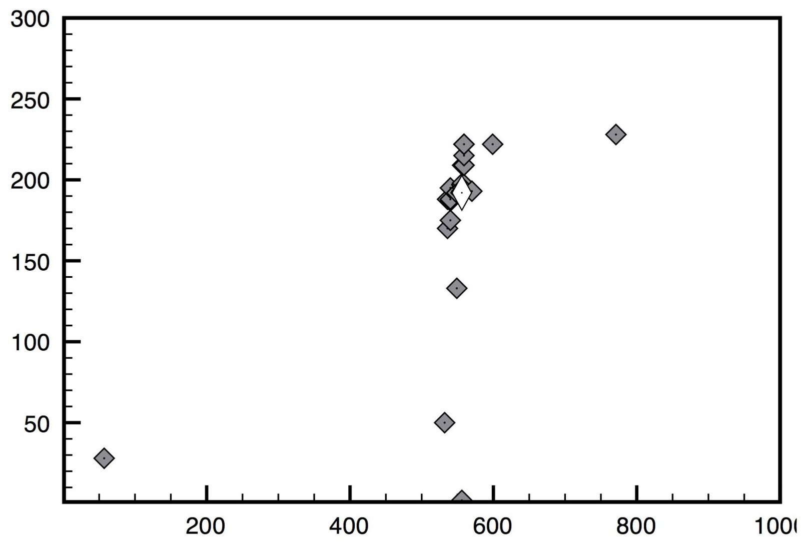


Figure 4.1: Sampling and filtering of the touch point.

- The devices is detected by Windows as a standard HID-mouse and the appropriate drivers are loaded automatically. The established USB connection works and it sends the obtained coordinates to the system and controls the cursor. This is only done as long as the touch panel is pressed. The calibration software can find the device and initialize it for calibration.

- The calibration has been implemented with five-point reference and is implemented in the uC. It is only executed when called from the software designed for a Windows environment. Figure 4.2 shows a freehand drawing of a square under different calibrations. The corners of the different squares represent the same point on the touch panel but with different calibrations. It clearly shows how the coordinate representation changes for different calibration reference points and it compensates for problems with scaling and alignment. Although not shown in the figure it also addresses the problem of non linearity.

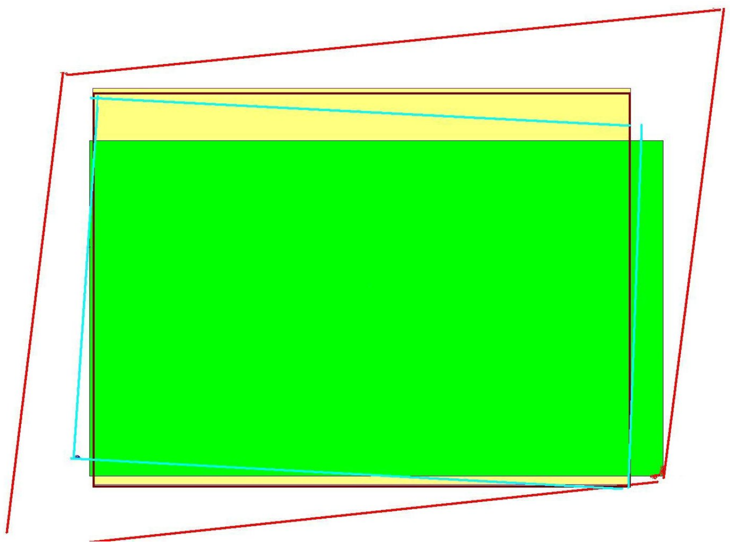


Figure 4.2: Example of square drawn under different calibrations.

The main work load has been put on getting a functioning HID-mouse with bidirectional data transfer support. Creating a proper HID- and USB descriptor without loosing the devices appearance or functionality as a HID-mouse. Another time consuming problem is that when the AD is sample at a high rate and switches fast between the input channels the value of the channels tend to infect each other giving undesired measurements.

5 Further developments

The elaboration of this devices can be done in a wide range of areas. Below is a short list of the most crucial ones in order to provide a robust and divers system.

- **Multi manufacturer support.** The device has not been tested or optimized for touch panels from other manufacturers. I will also need to be tested with a larger amount of different sizes.
- **Serial communication RS-232.** Although USB is a widely used standard that has several advantages it is sometimes desired to provide a more simple interface. A serial interface (RS-232) can be implemented to provide the functionality of the touchscreen to other more low parts of the system.
- **Multiple operating system support** The USB-HID-Mouse protocol is currently supported in several major operating systems, but not all. Another crucial parts is the calibration software that is supplied with the circuit has to be uniquely designed for each operating system.
- **Test** The controller has been designed and tested under close to ideal conditions. The final version would need to go through some extensive testing to assure correct operation under the conditions it is intended to be used in.

References

[1] PIC18F4550 datasheet, Microchip

[2] Bela G. Liptak, Instrument Engineers Handbook, T and F informa, 1995

[3] Rick Downs, Touch screen controller trips, Texas Instruments

[4] *ADS7846 Touch screen controller data sheet.* Texas Instruments

[5] Jan Axelson. *USB Complete.* Lakeview research llc, 2005

[6] Carlos E Vidales. *How to calibrate touch screens.* Embedded Systems Design, 31/02/05

[7] Bonnie C. Baker and Wendy Fang. *Powering resistive touch screens efficiently* . Planet Analog Magazine, 28/05/07

[8] R N Aguilar and G C M Meijer. *Fast electronics for a resistive Touch-screen.* Sensors 2002. Proceedings of IEEE

[9] *PanJitis Cobra 8 Wire controller Interface datasheet.* Panjittouchscreens

[10] *RVD15-S01 datasheet.* SAAB Group

[11] *Calibration procedure for resistive touchscreen system based on the STMPE811.* STMicroelectronics

[12] *S150804 datasheet.* Mildex Optrical Inc

[13] Neal Brenner and Shawn Sullivan. *4-wire and 8-wire Resistive Touch-Screen Controller Using the MSP430* Texas Instruments, February 07

[14] Neal Brenner and Shawn Sullivan. *4-wire and 8-wire Resistive Touch-Screen Controller Using the MSP430* Texas Instruments, February 07

[15] PIC18F4550 Demonstration code, Microchip

A Appendix A

As discussed in section 2.4 and 3.5 A, B, C, D, E and F are used as coefficients to convert the inputted values to calibrated values (equation 1.1 and 1.2).

$$X_{Calibrated} = AX_{coordinate} + BY_{coordinate} + C \tag{A.1}$$

$$Y_{Calibrated} = DX_{coordinate} + EY_{coordinate} + F \tag{A.2}$$

$X_{D1}, Y_{D1} to X_{D5}, Y_{D5}$ are predefined ideal coordinates used as reference when calculating the deviation between the inputted coordinates [11]. $X_1, Y_1 to X_5, Y_5$ are coordinates inputted by the operator. Each one corresponding to one set of ideal coordinates. If the ideal coordinates are equal to the inputted once the conversion is unnecessary.

$$\begin{pmatrix} X_{D1} \\ X_{D2} \\ X_{D3} \\ X_{D4} \\ X_{D5} \end{pmatrix} = M \times \begin{pmatrix} A \\ B \\ C \end{pmatrix} and \begin{pmatrix} Y_{D1} \\ Y_{D2} \\ Y_{D3} \\ Y_{D4} \\ Y_{D5} \end{pmatrix} = M \times \begin{pmatrix} D \\ E \\ F \end{pmatrix} \tag{A.3}$$

Where:

$$M = \begin{pmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \\ X_3 & Y_3 & 1 \\ X_4 & Y_4 & 1 \\ X_5 & Y_5 & 1 \end{pmatrix} \tag{A.4}$$

$$M^{-1} = \frac{1}{det(M)} Adj(M) \tag{A.5}$$

$$P = (M^T \times M)^{-1} \times M^T \tag{A.6}$$

Equation A.5 and A.6 gives:

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} = P \times \begin{pmatrix} X_{D1} \\ X_{D2} \\ X_{D3} \\ X_{D4} \\ X_{D5} \end{pmatrix} and \begin{pmatrix} D \\ E \\ F \end{pmatrix} = P \times \begin{pmatrix} Y_{D1} \\ Y_{D2} \\ Y_{D3} \\ Y_{D4} \\ Y_{D5} \end{pmatrix} \tag{A.7}$$

$$M^T \times M = \begin{pmatrix} \sum_{k=0}^{n} X_k^2 & \sum_{k=0}^{n} X_k Y_k & \sum_{k=0}^{n} X_k \\ \sum_{k=0}^{n} X_k Y_k & \sum_{k=0}^{n} Y_k^2 & \sum_{k=0}^{n} Y_k \\ \sum_{k=0}^{n} X_k & \sum_{k=0}^{n} Y_k & n \end{pmatrix} \tag{A.8}$$

$$M^T \times X_D = \begin{pmatrix} \sum_{k=0}^{n} X_k Y_{Dk} \\ \sum_{k=0}^{n} Y_k Y_{Dk} \\ \sum_{k=0}^{n} Y_{Dk} \end{pmatrix} and M^T \times Y_D = \begin{pmatrix} \sum_{k=0}^{n} X_k X_{Dk} \\ \sum_{k=0}^{n} Y_k X_{Dk} \\ \sum_{k=0}^{n} X_{Dk} \end{pmatrix} \tag{A.9}$$

$$det(M) = a(ei - hf) + b(fg - id) + c(dh - ge) \tag{A.10}$$

The coefficients are given by:

$$A = (j_x(ei - hf) + b(fl_x - ik_x) + c(k_x h - l_x e))/det(M) \tag{A.11}$$

$$B = (a(k_x i - l_x f) + j_x(fg - id) + c(dl_x - gk_x))/det(M) \tag{A.12}$$

$$C = (a(el_x - hk_x) + b(k_x g - l_x d) + j_x(dh - ge))/det(M) \tag{A.13}$$

$$D = (j_y(ei - hf) + b(fl_y - ik_y) + c(k_y h - l_y e))/det(M) \tag{A.14}$$

$$E = (a(k_y i - l_y f) + j_y(fg - id) + c(dl_y - gk_y))/det(M) \tag{A.15}$$

$$F = (a(el_y - hk_y) + b(k_y g - l_y d) + j_y(dh - ge))/det(M) \tag{A.16}$$

B Appendix B

The C code provided below show the USB descriptors. It describes the formate, size and direction of the packages (reports) and how the system should interpret them [15]. As described in 3.4 this is a bidirectional mouse with an five bytes wide input report and a two byte wide feater report.

```
// Device Descriptor
rom USB_DEV_DSC device_dsc=
{
    sizeof(USB_DEV_DSC),// Size of this descriptor in bytes
    DSC_DEV,            // Device descriptor type
    0x0200,             // USB Spec Release Number in BCD format
    0x00,               // Class Code
    0x00,               // Subclass code
    0x00,               // Protocol code
    EP0_BUFF_SIZE,      // Max packet size for EP0
    0x0925,             // Vendor ID
    0x7001,             // Product ID: Mouse in a circle fw demo
    0x0001,             // Device release number in BCD format
    0x01,               // Manufacturer string index
    0x02,               // Product string index
    0x00,               // Device serial number string index
    0x01                // Number of possible configurations
};

/* Configuration Descriptor */
CFG01={
    /* Configuration Descriptor */
    sizeof(USB_CFG_DSC),// Size of this descriptor in bytes
    DSC_CFG,            // CONFIGURATION descriptor type
    sizeof(cfg01),      // Total length of data for this cfg
    1,                  // Number of interfaces in this cfg
    1,                  // Index value of this configuration
    0,                  // Configuration string index
    _DEFAULT|_RWU,      // Attributes
    50,                 // Max power consumption (2X mA)


/* Interface Descriptor */
    sizeof(USB_INTF_DSC),// Size of this descriptor in bytes
    DSC_INTF,           // INTERFACE descriptor type
    0,                  // Interface Number
    0,                  // Alternate Setting Number
    1,                  // Number of endpoints in this intf
    HID_INTF,           // Class code
    BOOT_INTF_SUBCLASS, // Subclass code
    HID_PROTOCOL_MOUSE, // Protocol code
    0,                  // Interface string index
```

```
/* HID Class-Specific Descriptor */
    sizeof(USB_HID_DSC),// Size of this descriptor in bytes
    DSC_HID,            // HID descriptor type
    0x0101,             // HID Spec Release Number in BCD format
    0x00,               // Country Code (0x00 for Not supported)
    HID_NUM_OF_DSC,     // Number of class descriptors
    DSC_RPT,            // Report descriptor type
    sizeof(hid_rpt01),  // Size of the report descriptor


/* Endpoint Descriptor */
    sizeof(USB_EP_DSC), // Size of this descriptor in bytes
    DSC_EP,             //Endpoint Descriptor
    _EP01_IN,           //EndpointAddress
    _INT,               //Attributes
    HID_INT_IN_EP_SIZE, //Interval
    0x0A                //size

};

rom struct{byte bLength;byte bDscType;word string[1];}
sd000={ sizeof(sd000),DSC_STR,0x0409};

rom struct{byte bLength;byte bDscType;word string[26];}
sd001={ sizeof(sd001),DSC_STR,
'S','A','A','B',' ','A','E','R','O','T','E','C','H',' ','-',' ',
'S','A','A','B',' ','G','R','O','U','P'};

rom struct{byte bLength;byte bDscType;word string[23];}
sd002={ sizeof(sd002),DSC_STR,
'T','O','U','C','H',' ','S','C','R','E','E','N',
' ','C','O','N','T','R','O','L','L','E','R'};
\end{lstlisting}


rom struct{byte report[HID_RPT01_SIZE];} hid_rpt01={

    0x05, 0x01, /* Usage Page (Generic Desktop)
    0x09, 0x02, /* Usage (Mouse)
    0xA1, 0x01, /* Collection (Application)
    0x09, 0x01, /*      Usage (Pointer)
    0xA1, 0x00, /*      Collection (Physical)

    0x05, 0x01, /*      Usage Page (Generic Desktop)
    0x09, 0x30, /*      Usage(X)
    0x15, 0x00, /*      Logical Minimum(0)
    0x26, 0xFF, 0x03,/* Lagical Maximum(1024)
```

```
0x35, 0x00, /*       Physical Minimum(0)
0x47, 0xFF, 0x03, 0x00, 0x00,/*       Physical maximum((1024)
0x75, 0x10, /*       Report Size(16)
0x95, 0x01, /*       Report Count(1)
0x81, 0x02, /*       Input (Data, Variable, Absolute)

0x05, 0x01, /*       Usage Page (Generic Desktop)
0x09, 0x31, /*       Usage(Y)
0x15, 0x00, /*       Logical Minimum(0)
0x26, 0xFF, 0x03,/* Lagical Maximum(1024)
0x35, 0x00, /*       Physical Minimum(0)
0x47, 0xFF, 0x03, 0x00, 0x00,/*       Physical maximum((1024)
0x75, 0x10, /*       Report Size (16)
0x95, 0x01, /*       Report Count (1)
0x81, 0x02, /*       Input (Data, Variable, Absolute)

0x05, 0x09, /*       Usage Page(Buttons)
0x19, 0x00, /*       Usage Minimum (0)
0x29, 0x03, /*       Usage Maximum (3)
0x15, 0x00, /*       Logical Minimum(0)
0x25, 0x01, /*       Logical Maximum(0)
0x95, 0x03, /*       Report Count (3)
0x75, 0x01, /*       Report Size (1)
0x81, 0x02, /*       Input (Data, Variable, Absolute)

0x09, 0x33, /* Usage(Receive)
0x75, 0x10, /* Report Size(16)
0x95, 0x02, /* Report Count(2)
0x91, 0x00, /* Output (Data,Array,Absolute)

0xC0,       /* End Collection
0xC0        /* End Collection
};
```