# Validation of a Touch Screen Interface using an Oscilloscope

Roland Gamper

## Introduction

A Touch Screen Interface is examined using chained **computational functions** on a LeCroy digital oscilloscope. This case study spans 4 functions: the SPI real time decoding, the MessageToValue function, the Trend and the XY plot. This setup is used to develop and validate a Touch Screen hardware and software driver.

The technique shown is very efficient in detecting errors or validating the design, but also in **diagnosing the root cause** of these errors. **The main advantage of using the oscilloscope** resides in the fact that the instrument shows the physical transmission layers, the digital encoding, as well as the interpretation of the data embedded in the serial data at the same time.

The same type observations could be conducted on **many other serial streams** , such as UART, MIL-STD-1553, SPI, ARINC 429, LIN, FlexRay, MIPI, I2C, etc. and provide rapid insights into the quality and structure of the data at both the analog and digital levels, as well as the inner workings of the interface.

## Context, Hardware and Software

The 240x320 pixel screen and its associated touch screen are part of a multi peripheral board used by the Laboratory of Numerical Systems (LSN) to train students at the Geneva's University of Applied Sciences. The board interconnects the peripherals over a variety of serial lines (I2C, SPI, CAN, UART ) and has dozens of probing pods easing the observation of signals.

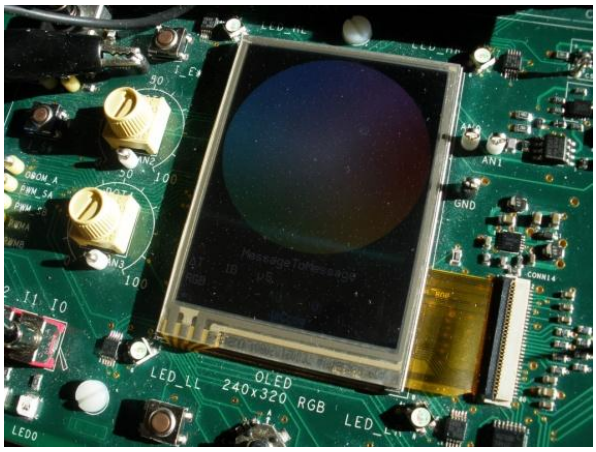


Figure 1: The TSC2046 Touch screen is mounted on a 240x320 pixel OLED screen.

The Touch Screen is a TSC2046, slaved to a LPC2292 microcontroller (NXP), over an SPI bus. The touch screen detects hits with an 8 bit resolution, over the entire span of the screen. Whenever the touch screen detects a contact, an interrupt service request (IRQ) is posted to the controller. In response, the microcontroller activates the touch screen driver, which interrogates the device over the SPI interface at a rate of 33 250 kb/s. The driver reads out the hit coordinates and forwards them to the application program layer.

## Setting up the oscilloscope for monitoring the Touch Screen interaction

The setup required to reach the XY plot is a **4 step processing chain** shown in Fig.2. Each of the components of the chain feeds its results into the next component. We will look at the processing chain, from the read out of the SPI signals on Channel 1 and 2 (see Fig 3), downstream to the XY plot showing the 2D image of the screen hits (see Fig. 5 and 6). The intermediate steps are documented in Fig. 4.

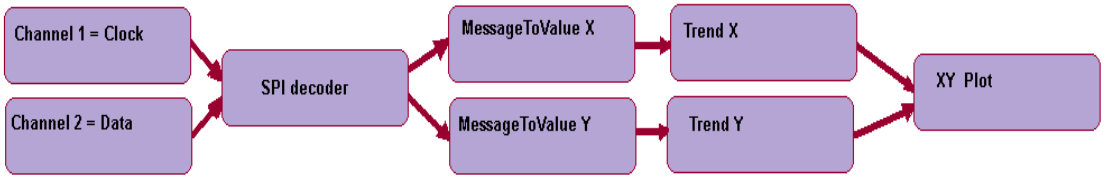


Figure 2: The 4 step processing chain of the oscilloscope, from raw data on the Channel to the XY display

## Using the SPI decoding

The decoder has to be setup in order to decode the SPI message flow between the Touch Screen and the microcontroller. In this case we use Channel 2 for the Data line and Channel 3 for the Clock line. The Frame mode, combined with the Inter Frame Timeout allows the logical interpretation of the messages as complete transactions containing X and Y for the contact point. This is important for the MessageToValue Parameters used later. We chose a decimal viewing to ease comparison between positions and values. The settings are shown in detail on Fig. 3.

## Configuring two MessageToValue Serial Parameters

This serial Data Parameter belongs to the ProtoBusMAG option set. It extracts a value from a given message at a known position and can also rescale it if necessary. For the practical case studied here we need 2 MessageToValue Parameters to extract X and Y coordinates from each message. The only difference between both Parameters is the Start Position of the value. It is 32 bits for the X position and 8 bits for the Y position.

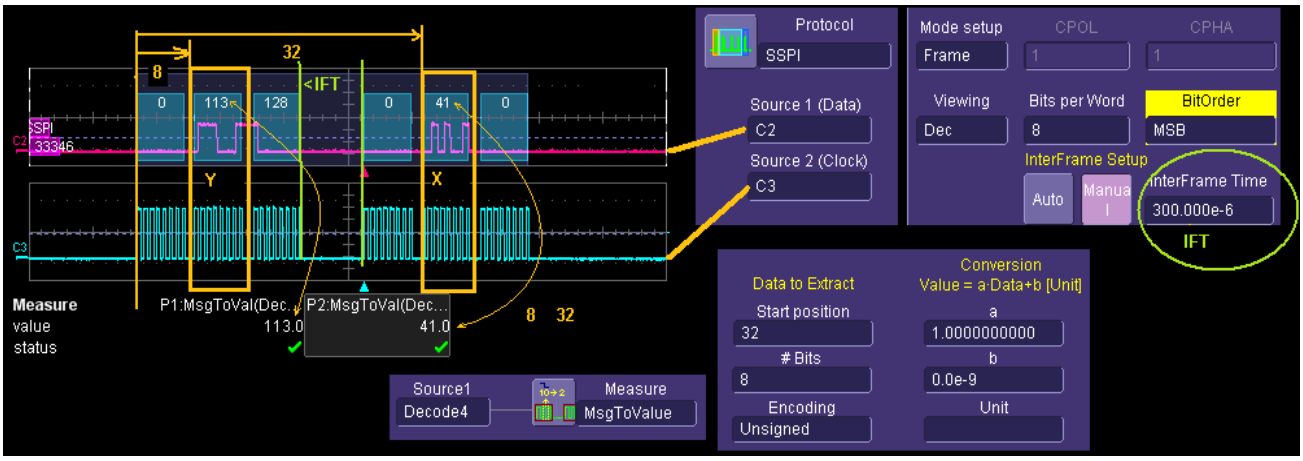


Figure 3: MessageToValue Setup. Observe the match between the values annotated on the trace and those extracted by the MessageToValue Parameter. In this message the X position is 41 while the Y position is 113. Also note the The Inter Frame Timeout , essential to keep the logical messages together.

## Trending both results of MessageToValue

The next processing components are 2 Trend Functions shown in Fig. 4. F1 accumulates the P2 values, while F2 accumulates the P1 values. Both Trends are setup at the same scales and offsets so that they can be combined into an XY graph reflecting the geometry and orientation of the screen. Each pair of values on the Trends represents the coordinates of an impact point on the Touch Screen. When all the <x,y> pairs are drawn into the XY plot, the trajectory of the hits become visible.

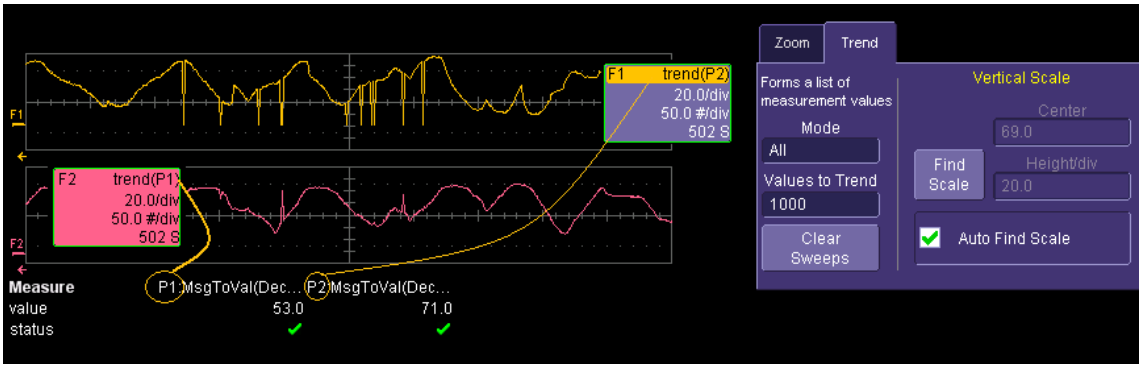


**Figure 4: The exact setup of the Trend functions keeping a memory of all impact points on the Touch Screen.**

## Reconstructing the 2D image of the Touch Screen

Finally, at the end of the processing chains, both Trends are used as sources for an XY graph. This graph shows the <x,y> decoded pairs in an appropriate coordinate system and reconstructs exactly the positions of the impact points on the Touch Screen, as shown in Fig. 5.

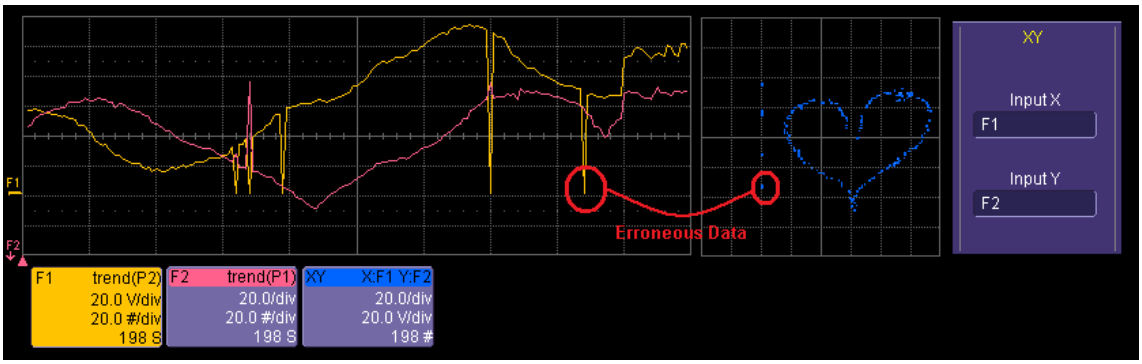


**Figure 5: The Trends Functions and their XT plot, with XY settings. The spikes on the Trend F1 cause stray dots on the XY plot, and reflect an error on the X position read out.**

As a comparison, the XY plot on the oscilloscope is shown along with a screen photograph in Fig. 6. Both images exhibit the very same hits in the heart shape. The DSO can therefore reconstruct the 2D image **directly from the analog signals** measured on Channels 1 and 2. This technique allows a quick assessment of the correct functioning of the system. Furthermore, if and when errors appear, the tools help in rapidly diagnosing the root cause of the error.

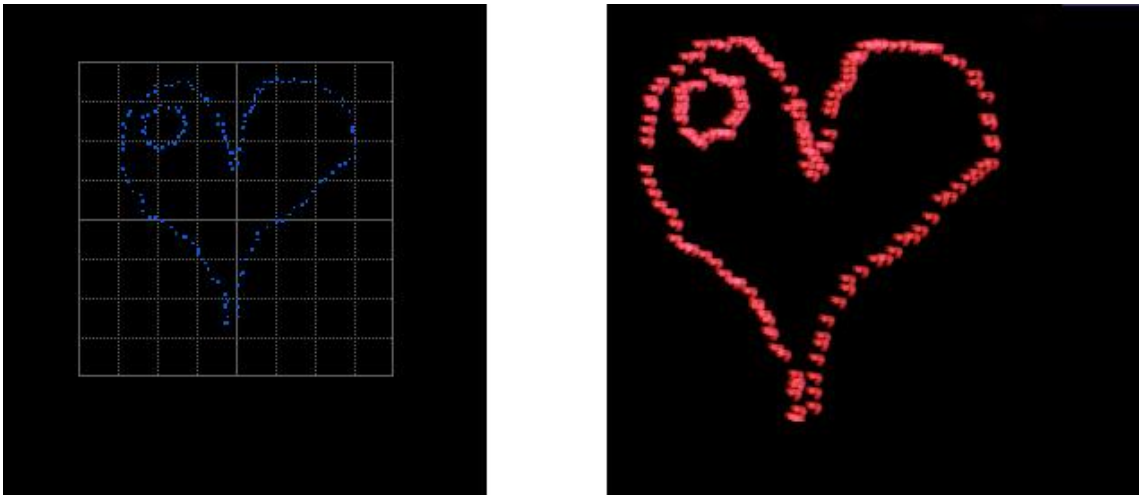

Figure 6: On the left, the XY plot observed on the DSO, and on the right a photograph of the screen with the corresponding trajectory of the hits.

## Other possible Measurements

Other serial data tools allow the qualification of the **responsiveness** of the driver, by measuring delays between Touch Screen messages.

Using the Decoder and the BusLoad Parameter, it is easy to hunt **for useless bus transfers** and therefore optimize the handshake. In this example, the third byte of the X and Y transfer could be dropped from the transmission, as it does not carry meaningful information.

Parameters such as DeltaMessages help assessing the **bus activity when multiple peripheral** share a common bus.

The Mask and Pass Fail system allows the validation of extrema on the X and Y values, in order to identify **unrealistic readings** of the values caused by the hardware or the software.

## Conclusions

The development and validation of hardware drivers is eased by using the right tools. A digital oscilloscope of the latest generation allows the observation of various signals at different points of the numerical processing chain. The simultaneous observation and correlation of analog, digital and encoded data allows rapid analysis of the defect and shortens the development cycles. As an example, erroneous hits shown in the XY plot **can have several causes**, such as the Touch Screen hardware, serial transmission problems or computational errors in the driver. The observation of the results of the processing steps explained above will quickly help the engineer in identifying one of the 3 causes listed.

## Thanks

Many thanks to F. Vannel at the LSN for the opportunity to capture live signals on the system described here and to Y. Leoni for the firmware development needed to generate the proper signals and pictures. Thanks as well to G. Ritter and U. Schroffenegger at LeCroy for the collaboration, insights and exchanges on the ProtoBusMAG module embedded in the LeCroy oscilloscopes.

## The author

Roland Gamper is Senior Software Engineer and was employed 22 years by LeCroy in Geneva. He currently undertakes contract developments, in particular in the field of Protocol analysis, through Lahniss Ltd. He also teaches microcontroller programming at Geneva's University of Applied Sciences.