ELECTRONIC DESIGN, TESTING & STANDARDS

Testing

Firmware: The Inexpensive Way to Address EMC Issues

In more than ten years working in labs as an EMC engineer, the majority of devices and systems I have tested include two important elements that interact with each other, thereby allowing the equipment under test to work. Electronic boards and firmware. It is difficult to imagine an electronic board without firmware running in a microcontroller. Even a simple wall charger for batteries has integrated firmware to switch the behavior of the charger from constant voltage to constant current, and to switch into trickle charging or to begin the discharge process.

Over the years, I've come to think of firmware as the soul of every electronics board since, without firmware, almost every PCB would be "dead." But because firmware is now so deeply embedded into the working mode of a device, what is the influence, if any, on a device's EMC performances? Could, in theory, at least, a device using different firmware versions behave differently during EMC measurements? We'll explore that question in this article.

Fact: EMC Measurements are Expensive

We all know that EMC measurements can be expensive, especially if issues arise during the test process. Projects have to stay on budget, and layout modifications can be expensive in terms of both time and money. In fact, every new hardware release of a PCB is usually seen as something potentially problematic that should be avoided at all costs. Electronics engineers always want their boards to work right the first time, but sometimes that simply doesn't happen.

During my daily time in our testing laboratory, I often feel the pressure faced by the electronics designer, who is deeply involved in matters like costs and budgets. As a result, EMC measurements should be taken into account inside the budget analysis of a project. As an example, a full compliance EMC measurement evaluation of a medical device with a power cable, one I/O cable, and radiating emissions up to 6GHz could result in an expenditure of thousands of Euros, even if the equipment under test (EUT) meets every testing criteria.

Encountering EMC Measurements: Should You Take a "Hardware" or a "Firmware" Approach?

We are now entering the EMC laboratory with the EUT in hand. What are the common issues an electronics engineer has to face during immunity and emission tests?

According to what I have written above, the great advantage of taking the "firmware approach" is the cost and speed of implementing the solution. So it might be useful here to provide examples of common solutions involving firmware changes. Of course, the list is far from being complete, but I think it represents a good starting point.

Here are four different solutions, two solutions each for emission measurements and immunity testing.

Emissions Solution #1:Clocks and Data Transfers

Let me start with a radical proposition regarding clocks and data transfers. When you are designing an electronics board, assuming that the design requirements allow you to do so, don't set its fastest speed as the default. In other words, the electronics and firmware engineer should estimate the actual speed required to transfer data as related to the requirements and characteristics of the system being designed.

Emissions Solution #2: Unconfigured Pins

Sometimes, leaving default configurations of the unused pins can be dangerous. Read the datasheet and application notes, since often these details are explained. Usually, it is a good idea to apply the information in the documentation of a microcontroller, especially when some PINs require particular treatment (such as speed configuration, idle configuration or not-used state with some kind of high impedance setting, etc.).

Immunity Solution #1: Watchdog

A watchdog is an electronic timer that is updated by the CPU/microcontroller at regular intervals. If the timer is not updated due to a microcontroller or CPU block, a timeout signal is generated by the watchdog. The presence of the timeout signal could be checked by the firmware (running on another microcontroller), and a reset signal could be issued to the CPU/microcontroller in order to reset the state of the device. This is very useful in order to successfully pass pulsed immunity testing, where the performance criterion is usually of B or C type. In some cases, a full reset (with data retention) is permitted for tests like pulsed immunities.

Immunity Solution #2: De-bounce Code and Averaging

In cases in which the firmware has to read the state of a pressed button or a signal coming from a capacitive touch control (or any other sensitive interface), it is recommended to implement a de-bounce algorithm in the code in order to exclude spurious button presses. The de-bounce code can be a useful tool in excluding unwanted effects related to RF injection, which could lead to a microcontroller sensing the activation of a button when no button has been pressed.

Figure 1 shows an example of a de-bounce algorithm based on a timer and interrupt method. A firmware engineer can implement different techniques, depending on the final application and the object of the de-bounce (capacitive touch control, mechanical switch, and so on).

Another technique used to avoid spurious data is averaging. It is used mainly when slow phenomena have to be acquired by sampling a huge amount of data, for example, environmental parameters like temperatures, humidity, etc. In case the injection of a disturbance during immunity tests (both conducted and radiated) alter some data, the averaging keeps the trend stable by reducing unwanted variations.

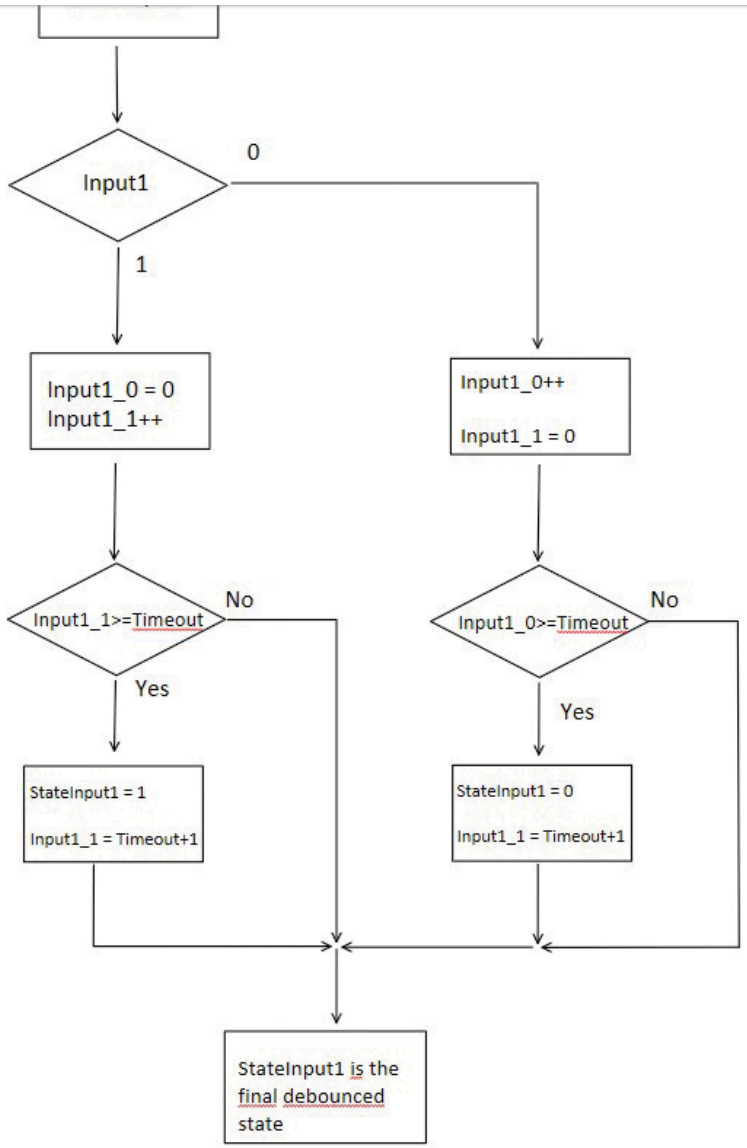


Figure 1: When Input1_0 or Input1_1 exceeds or is equal to Timeout, means that it was at 1 or 0 for at least the time of Timeout. Which means that the logical states 1 and 0 must remain the same for all the Timeout timer.

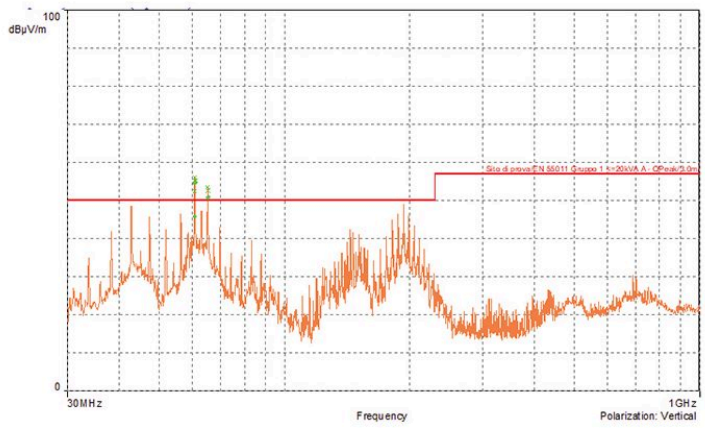
Case Studies

In this final part of the article, I would like to present three case studies I've collected in recent months. They are all about the impact of firmware on EMC measurements, and I'm hoping that you recognize yourself in some of these cases.

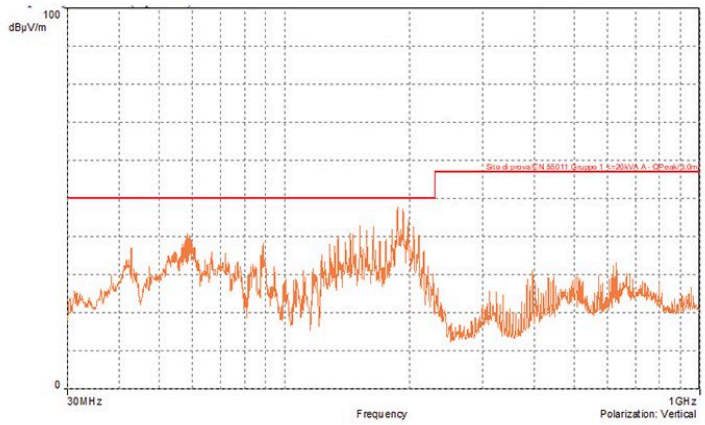
Case Study 1: Medical Device, Inexpensive Ending

This case study is very recent and relates to a medical device that measures biological electrical signals from body nerves (very small signals, hence, huge amplification required). The system has shielded cables and two microcontrollers inside. Radiated emissions were above the limits (see Figure 2a). The person present during the testing process was the developer of both the hardware and firmware of the device.

After some tests (e.g., disconnect cables, switch off one module after the other, etc.), we managed to spot the issue. On one 3 meter length cable, two signals were present: 1) an SPI to read from an ADC; and 2) a serial communication for the link between the two microcontrollers (115200 baud). The datasheet for the microcontrollers noted that, in cases where the PINs are configured as outputs, a speed selection command was available. This command was not set but by default, the compiler set the speed at its maximum rate. By slowing down the two speeds maintaining the overall performance and stability of the



(a)



(b)

Figures 2a and 2b: Figure 2a shows the radiated emissions between 30 and 1000MHz before the firmware modification. Figure 2b shows the radiated emissions of the same board, but different firmware.

Case Study 2: Capacitive Keypad, Inexpensive Ending

Capacitive keypads are extremely sensitive. There are a lot of controllers available on the market, from plug-and-play models (usually cheap, less customization) to those that are more flexible (usually expensive, but offering full customization). Over time, I have noticed that, in many cases, the key issue is traceable to this difference.

A customer brought in the laboratory a system (an access control system for home use) with a capacitive keypad. During testing, we discovered issues with conducted immunity levels, in which the system was taking unwanted commands when a disturbance was applied. Overcoming this issue was as simple as changing the configuration of the capacitive controller by modifying a few lines of code. This modification solved the problem.

Case Study 3: Capacitive Keypad, Expensive Ending

This case study has a very different ending from the previous ones but involved another touch control using capacitive technology embedded into an industrial lighting system. In this case, the controller belonged to the “plug-and-play” category, with no access to the code whatsoever. As a result, every behavior was fully automated by code inside the controller that communicated to the main microcontroller via a serial protocol. There was no way to change code, no way to change anything, in fact, ultimately requiring a change in the layout of the board to improve ground planes and ground connections. The challenge was further complicated by the lack of space inside the enclosure.

I'd like to end this article with answers to two questions about EMC measurements campaign and firmware modifications during the life of a product, as follows:

Q. Who are the most suitable people to assist during the measurements?

A. Electronics engineers and electronic designers with deep knowledge of the board under test, and software engineers who developed the firmware (in some cases, these two are the same person).

Q. If the firmware of a product is changed/updated, do EMC measurements need to be repeated?

A. It depends on the modifications implemented in the firmware. The firmware is very often connected to the results of EMC tests, both for immunity and emission. Taking that into account, it's unlikely that testing two samples of the same equipment loaded and running with different firmware will produce the same test results. Would you take the risk?

I confess that it was hard to find resources for this article, which is mainly based on my experience and daily laboratory life. However, here is a short list of resources where the reader can find some hints and further suggestions on firmware-related issues.

In addition, I would like to give the reader some thoughts about the reactions this article generated on social networks, with some suggestions and hints received from readers and commenters.

Two other ways of solving EMC issues during measurement campaigns using firmware-related techniques are spread spectrum and data scrambling. The first is very well known and often implemented, for example, in switch-mode power supplies (SMPSs), sometimes in a transparent way to the designer. The latter is related to the way of transmitting high data rate digital information. Its purpose is to smooth out the spectrum of the transmitted data from peaks and spikes by acting on the bit sequences in order to resemble the spectrum of white noise instead of the one produced by data transfer. For the curious reader, the last reference is about these two techniques.

References

- *In Compliance Magazine.* (2018, December 7). "What Every Electronics Engineer Needs to Know About EMC Chambers."
- Tim Williams, *EMC for Product Designers* (fifth edition), Newnes, 2017.
- Keith Armstrong, *EMC for Printed Circuit Boards* (second edition), Nutwood UK Limited 2010.
- Keith Armstrong, *EMC Design Techniques*, Nutwood UK Limited, 2010.
- Clemens Valens, "How to Debounce a Mechanical Contact or Switch."
- Some examples of debounce code. The example shown in the article is taken from this website.
- Henry W. Ott, *Electromagnetic Compatibility Engineering*, Wiley 2009: chapter 15.10.
- Keith Armstrong, *Suppressing Emissions by Using Data Scrambling and Spread Spectrum Hardware Design Techniques*.