EUT/DUT Software Requirements for EMC Testing

James Pawson - Unit 3 Compliance – hello@unit3compliance.co.uk

V1 – May 2023 – Initial publication - JP

Introduction

This guide is one of our series of articles on preparing your product or equipment for Electro-Magnetic Compatibility (EMC) testing.

In this one we are going to look at the **software and firmware requirements for your Equipment (or Device) Under Test (EUT or DUT**) to get the most out of testing.

Our goal with EMC testing is to fully characterise the performance of the equipment (i.e. a full functional test) in the presence of a variety of electromagnetic phenomena or under specific conditions.

We need to consider maximising the electrical activity (including power consumption) within the equipment to be able to measure **EM Emissions** accurately.

We also need to consider how we can verify performance of the equipment during **EM Immunity** testing to ensure that application of EM noise to the product does not result in interference to the operation.

EMC testing should be carried **thoroughly** (to reduce compliance risks) and in a **timely** (cost effective) fashion. Useful software is essential to meet these needs.

There is a **commercial vs technical risk dimension** to the provision of test software. The balance is one of spending time developing test software on one hand vs the cost saving from faster testing and more complete test coverage. The advantage of developing this software is that it is highly reusable for other development activities.

We would estimate that around **50%** of our customers suffer some **cost-incurring delays** to testing **due to test software problems or availability**.

This guide is meant to be **thought provoking and not comprehensive**. It is impossible to write a generic guide that covers all specific cases. You as the designers will know the operation of your equipment better than anyone, therefore understand what modes and monitoring needs to be done. You should be able to apply the **principles** within this guide to your specific case.

This guide does not exist in isolation. If you **spot errors, have opinions, or suggestions** then please let us know. We would be happy to revise this guide and give full credit for any submissions.

Quick Guide

All major equipment functions should be working and:

- **Monitorable** – visually or remotely, as automatic as possible
- **Easily** – with minimal effort, easy to spot failed results or errors, maybe not boredom inducing lines of scrolling text
- **Remotely** – to allow testing inside the RF shielded room
- **Continually** – automatic repeat until stopped, no user input required
- **Quickly** – all functions exercised and checked in less than 3 seconds
- **Simply** – achieve the right balance of cost and benefits during development of the software

(Just memorise this simple acronym MERCQS (think of the famous cyclist) and you'll be fine!)

For each hardware function, you should be able to **draw up a table of software requirements**. Below is an example for a networked temperature sensor with Ethernet functionality.

Function	Modes	Emissions	Immunity	Monitoring	Performance Criteria A
Temperature sensor	Type K thermocouple	Measuring temperature	Measuring temperature	On screen display	±0.5°C
Temperature sensor	Platinum RTD	Measuring temperature	Measuring temperature	On screen display	±0.25°C
Display	Highest resolution and pixel clock	Normal display	As emissions, add visual heartbeat signal	On screen display	No distortion, blocking, or freezing
Ethernet	10Base-T	Continuous data via iperf	Continuous data via iperf	Iperf readout on remote terminal	>=1Mbps
Ethernet	100Base-TX	Continuous data via iperf	Continuous data via iperf	Iperf readout on remote terminal	>=10Mbps

Some tips (and in document links)

- Prior Planning Prevents Poor Performance – schedule the development of this software at the start of the project, and consider the costs of poor quality software.
- Consider other uses for the software during development (hardware testing, temperature measurements, radio qualification, etc)
- Try to design functions that fulfil both emissions and immunity requirements to save complexity
- Aim for a dwell / cycle time of less than 3 seconds to speed up testing, reduce costs, and reduce the need to compromise during testing.
- Consider how you will capture data during monitoring. Capture the minimum and maximum values during testing. Graphing values can be useful alongside raw data and error flags.
- Add changing data or a heartbeat to spot if the software freezes or crashes
- Enable the watchdog timer and disable any other timeouts
- Bias all analogue values to 50% of their range where possible
- Do the best you can in the time you have available
- Ask for help if needed – hello@unit3compliance.co.uk

Contents