POLITECNICO DI TORINO

Master of Science in Electronic engineering

Master Degree Thesis

Testing of switched-mode power supply

Supervisor:
prof. Marcello Chiaberge

Internship tutor:
Davide Di Carlo

Candidate:
Luca Monterisi

Academic year 2018 – 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis was born from the collaboration between the Polytechnic of Turin and Spea, global leader in test equipment for electronics, semiconductor, MEMS and sensor industries.

The main focus of the work was the test of a switched-mode power supply. The work was carried out in the application department of Spea: here, the client's applications are tested in various ways and with different ending goals.

The main reasons that led me to undertake this thesis path were the importance of testing electronic boards and the inevitable importance of power supplies in nowadays electronic. Testing an electronic board, but ultimately any kind of electronic component, is essential in its proper and safe fabrication. No matter how careful the design and fabrication process of an electronic board could be; there will always be the possibility of defects, failures and faults in it. The electronic test presents than itself as the ultimate qualifier step of this complex process. Power supplies have played an important role in the history of electronics and have therefore been protagonists of constant improvement and innovation through time. Often forgotten, power supplies are instead essential for the correct operation of any kind of electronic component.

The main aim of this thesis work is, therefore, to analyse, study and put into practice the testing of a switched-mode Power Supply; to highlight its intrinsic meaning with a closer look to the state-of-the-art techniques created and used in Spea.

The thesis has been completely developed in Spea during an internship period lasted two months followed by a stage of six months ending with a brief installation period carried out in the client's company.

The following dissertation is articulated into seven chapters. In chapter 2 a brief history of the evolution of Power supplies is reported, followed by the modern classification of these devices and the analysis of their different configurations and functions. Chapter 3 investigates the importance and reasons for testing electronic boards focusing on the different kinds of tests such as In-Circuit Test and Functional Test highlighted their different goals and characteristics. Chapter 4 defines the State-of-the-art of test in the Spea 3030 Bed of Nails and Flying Probe Automatic Test Equipment. Chapter 5, 6 and 7 report the work carried out to properly test the Power Supply application going from the analysis of the client's Test Requirement Specification, to the implementation of the Fixture and ending describing the writing of the Test Program and its debug resulting in the actual test of the Power Supply. The final chapter describes the installation process of the implemented application on the customer company site.

Chapter 2

Power Supplies: from Linear to Switching converters

"A Power supply is an electrical device that supplies electric power to an electrical load"[1][1]. Power supplies can be of different types and kinds but in general they perform their required functions by the use of an internal regulator/converter (depending on the characteristics of the input source) which can be linear or switching.

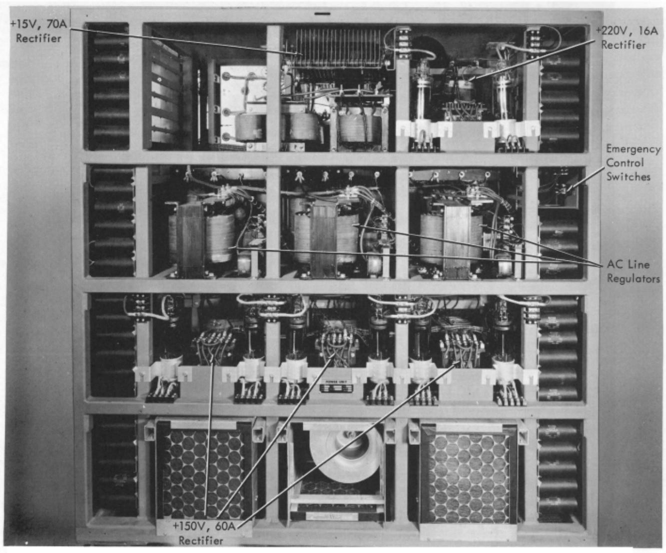
Linear and switching regulator are essentially two different kinds of power supplies.

Historically the first implemented regulators were the linear ones due to the less complexity and more immediate functioning regulation.

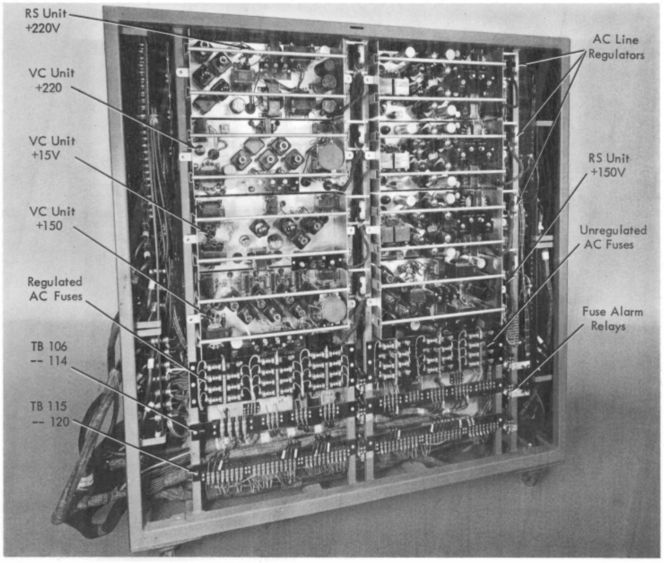
2.1 History and evolution of Power Supplies

It is possible to trace back the starts of the power supply industry in the early 1920s when the main employment of this new technology was as B battery eliminators for powering radio. The first implementation of this devices consisted, as already stated, in linear regulators which were made of vacuum tube. Vacuum tubes were used both for the power and the device control and initially (and almost until the late 1970s) energy dissipation was not addressed as an issue since not only the efficiency reachable was quiet limited but as object itself the vacuum tube returned a visible stimulus to the heat glowing red.

[1] "https://en.wikipedia.org/wiki/Power_supply"



(a) Front view



(b) Rear view

Figure 2.1: IBM 736 Power Supply, October 29 1958[2]

Linear converter remained the prime power supply technology since the late 1950s when the first Switching type regulator was for the first time used [2][2]. This

[2]IBM Customer Engineering Reference Manual - 736 Power Supply - 741 Power Supply - 746

evolution was primarily driven by the introduction in the electronic industry design of the first semiconductors. The link between power supply advancement and transistors improvement was so intertwined that when a new transistor was created after no more than a year a power supply using that technology was on the market.

In the 1960s the linear regulator application already seemed to be put away and from the aerospace field even great incentive to switched mode power supply research surfaced.[3][3]

The second great discovery that further increased the power supply enhancement arrived in the 1970s: new "loss-less" ferrite material were used as core element of the power supply transformer. This, coupled with high speed transistor, allowed to reach for the first time frequency higher than 20 KHz. Trough the 1970s improvements were also achieved in the control end part of these devices as digital control: the very first of this kind of control interface consisted in a series of resistances in parallel to relays. In 1972, the very first Switched Mode Power Supply patent were filed.[4][4]

In the 1980s the most significant parameters of SMPS were further increased thanks to the substitution of bipolar transistors with Field Effect Transistors (FET): efficiency reached the 85 - 90% and the frequency range as well improved from $25 - 50$ KHz to 1 MHz.

From the 1990s, all the improvement, new technologies and new materials developed and used in Power Supplies aimed at increasing efficiency and power density and reducing cost, weight and dimension. One path that the scientific community has chosen to follow in order to achieve this objective is to increase the operational frequency of these devices. Increasing frequency, in fact, can bring to a further reduction of components with although the significant drawback of the increase of all components' switching losses.

Hence, from the 2000s and until nowadays, the research has been focused on

Power Distribution Unit, October 29 1958

[3]The Spacecraft Power Supply System - D.C. Bomberger, D. Feldman, D.E. Trucksess, S.J. Brolin and P.W. Ussery, February 11 1963, section 2.3.

[4]"Switching-mode power supply - Patent Number: US3798531D, Filing date: 1972-06-05, Inventor: R. Allington"

the so called Very High Frequency (VHF) power supply. The VHF devices work in a frequency band of 30 to 300 MHz which is usually characteristic of the radio frequency transmitted domain leading to its merging with the power electronics circuits one.[5][5] This MHz frequency range, however, yields many problems, some of them still to be solved, regarding the electronic components, the circuit architectures and the Electro-magnetic Interference.

As for the architectural issues, high radio frequency devices are designed usually to match defined load (antenna's impedance), power supplies have instead the need to be connected to varying loads meaning that "active and loss-less impedance matching circuits are required"[5]. The EMI problem is self explanatory: the VHF converters will interact with the surrounded environment in a different way than standard power converters increasing the electro-magnetic interaction between components which affects also the device electrical behaviour.

2.2 Linear power supply

A linear power supply, as already stated, owes its name to its exploitation of an internal linear regulator. It is in fact quiet common to refer to power supplies in their totality with the simple name of regulator or converter recognizing, in this way, what is considered their most peculiar functionality.

The main and more general elements of such device are a transformer, a rectifier, a filter and a regulator, as shown in figure 2.2.

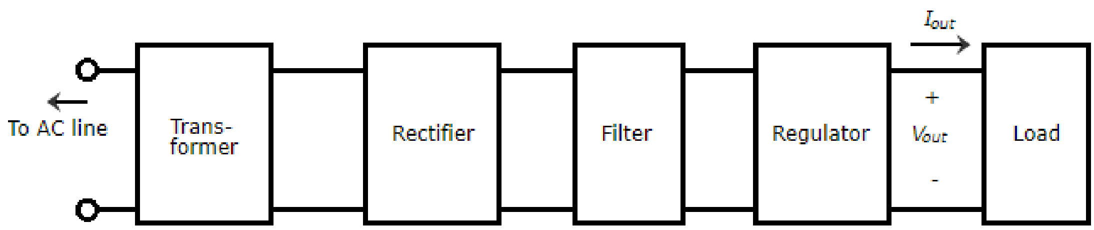


Figure 2.2: Linear power supply generalized block diagram

[5]Evolution of Very High Frequency Power Supply - Arnold Knott, Toke M. Andersen, Peter Kamby, Jeppe A. Pedersen, Mickey P. Madsen, Milovan Kovacevic, Michael A.E. Andersen - Technical University of Denmark, 2013.

The transformer is a device used to modify the mains power value to a lower one properly chosen w.r.t. the needed application. When the initial power source is an AC then a rectifier block is needed in order to convert the AC source to a DC one that will then be modulated to the needed values. The filter, which will be of the utmost importance in the switching regulator, executes a polish of the output signal, removing any kind of unwanted frequency components. Finally, in electronics, "a linear regulator is a system used to maintain a steady voltage"[6][6]. This is usually realized using a feedback loop to bias a pass element (resistance, transistor) to maintain a constant voltage across its output terminals. The aforementioned pass elements of the regulator vary in accordance with the load resulting in a constant output voltage. The working principle of this devices is to adjust the value of the regulating circuital element by dissipating energy (usually as heat) so that the voltage difference between the input and the set voltage will remain constant, hence maintaining the output voltage fixed at the same time. Its main function is to convert a varying DC or AC input voltage to a constant, specific lower DC output voltage. Linear regulators are compulsory step-down converters, meaning that the output voltage will always be less than the input voltage. In fact, there is a minimum voltage difference, called drop-out voltage, between the input and the output that will allow the linear regulator to work.

The downfall of the linear power supplies can be retraced to their dimension and efficiency characteristics. In order to obtain high output values, the physical dimension of a linear regulator can be quite demanding: especially in the aerospace field of application, the constantly increasing need of light weighted, compact and powerful devices have forced the state of the art technology to search for other kind of converter. Last but not least, linear converter has very low efficiency: efficiency largely depends on the voltage difference between input and output; the output voltage is regulated by dissipating excess power as heat resulting in a typical efficiency of 30 - 40% (never higher than 50%).

Still, there are some specific cases in which a linear regulator may be the most suitable choice. It has to be remembered that these devices are extremely cheap in

[6] "https://en.wikipedia.org/wiki/Linear_regulator"

both cost and circuital complexity; moreover, a linear regulator may be preferred for light loads or where the desired output voltage approaches the input source voltage providing, in this cases, a dissipated power lower than other implementations.

Back to the aerospace technology environments, the found substitute to the linear power supply were the Switched mode power-supply.

2.3 Switched Mode Power Supply

A Switched mode power-supply (SMPS), is an electronic power supply that converts electrical power efficiently. Like other power supplies, an SMPS transfers power from a DC or AC source (often mains power) to DC loads. "Voltage regulation is achieved by varying the ratio of on-to-off time"[7][7] the SMPS incorporates a switching regulator which rapidly switch a series device on and off and the duty cycle of the switch sets how much charge is transferred to the load. These devices have generally more complex circuits w.r.t. a traditional linear converter, but have different advantages such as lower physical dimension at lower weight for the same amount of output power, higher efficiency (which can easily reach $80 - 90\%$) and therefore less heat dissipation. On the other hand, SMPS are less suitable for laboratory purposes since they are characterized by more substantial ripple and high frequency components generation which can interfere with the correct functioning of most devices. In other words, a SMPS can be defined as a power converter that utilizes switching devices such as, for instance, MOSFETs that continuously turn on and off at high frequency and storage devices, such as capacitors and inductors, to supply power during its non/conduction state.

The main technological difference between the linear and switched power supply stands in the realization that a transformer, in order to improve its efficiency, needs a smaller and more compact ferromagnetic core (providing the same amount of output power).

As for its internal structure, a switched mode power-supply is composed by almost the same functional blocks of a linear one with, although, a higher focus on the filtering components that will have to execute more severe work. A SMPS

[7] "https://en.wikipedia.org/wiki/Switched-mode_power_supply"

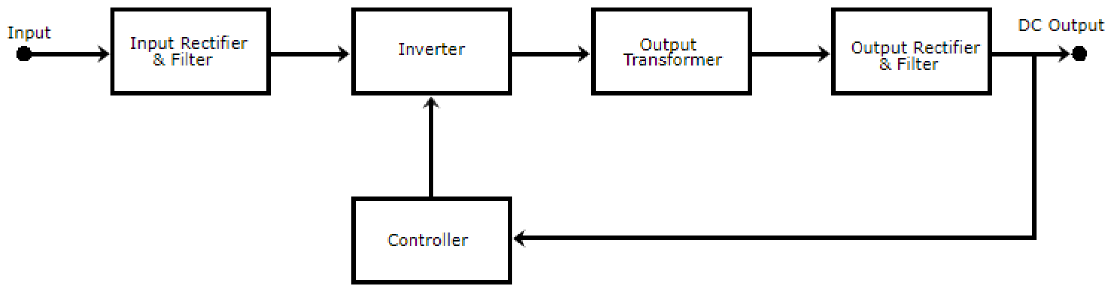


Figure 2.3: Switched Mode Power Supply block diagram

will then generally be composed of an input rectifier and filter, an inverter, an output transformer, an output rectifier and filter and, in some cases, of a feedback controller circuit (figure 2.3).

The input rectifier is logically needed only in the presence of an initial AC input that has to be converted in a DC one with performing an operation called rectification. The rectifier is nothing but an electrical device (which can use various components going from vacuum tube diodes to semiconductor diodes and switches) that converts, as already stated, alternating (bidirectional) current to direct (unidirectional) current. The output of this stage is usually sent to a filter (usually a capacitor or inductor): the pure DC current produced by the rectifier presents itself as a pulsating signal (in the case of a half wave or full wave rectifier) that still needs to be converted into pure DC.

The inverter has the aim of generating, from the direct current provided by the previous stage, an alternate current at very high frequency (always at least higher of $20 - 50$ kHz) through a power oscillator circuit often implemented through a MOSFET amplifier. The stage's output is then sent to a high frequency transformer which will convert the voltage up or down to the required output level (PWM, pulse width modulation). Then the last stage of the chain rectified and filtered the transformer AC output in the required DC current.

The final block of the SMPS is the feedback controller circuit. The controller performs a stabilizing function controlling that the device output voltage remains constant in time. Usually this control is implemented trough a feedback error loop which measures, as the name suggest, the error on the output signal. Set a threshold, if the error exceeds the expected range (in both the positive or negative direction), the energy sent from the oscillator to the transformer is modified accordingly. The stage has been subsequently modified and improved by the addition of protection systems against overload and short circuits.

2.3.1 Classification of SMPSs

A first classification could regroup these devices in two main categories: non-isolated typologies and isolated typologies. The most known SMPSs belonging to the first category are the step-down or buck converter, the step-up or boost converter and a combination of the two simply called the buck-boost converter. The latter category is usually identified with the Flyback converter, forward converter and push-pull converter.

Non-isolated converters

The non-isolated converters have usually the characteristic of not using a transformer and therefore are non-isolated. They are used most commonly for DC-to-DC conversion and are characterized by simpler circuits.

"A buck converter (figure 2.4a) is a DC-to-DC power converter which steps down voltage (while stepping up current) from its input (supply) to its output (load)"[8][8]. It typically contains at least two semiconductors (a diode and a transistor which is described simply by a switch symbol in the figure), "although modern buck converters frequently replace the diode with a second transistor used for synchronous rectification) and at least one energy storage element, a capacitor, inductor, or the two in combination"[8][8]. The idea behind the stepping down of the voltage executed by the converter is simple: the inductor inside the circuit has the role of transferring the energy from the input to the output. During the ON phase, when the switch is closed, the current flows in the external mesh of the circuit (diode is inverse polarized) and the current flowing in the inductor increases as its stored energy does. When the switch is open, the voltage drop on the inductor is equal to the output (with opposite sign) load one and the current and the energy decrease (diode forward polarized). The current increase and decrease can be described through

[8] "https://en.wikipedia.org/wiki/Buck_converter"

the on and off switching time which are proportional to the duty cycle. Hence, modifying properly the duty cycle of the switch results in an according modulation of the output voltage. Below, the mathematical formula useful to follow the said buck converter work explanation are briefly reported.

Inductor energy:

$$E = \frac{1}{2} L * I_L^2 \tag{2.1}$$

Delta current increase:

$$\Delta I_{on} = \int_0^{t_{on}} \frac{V_L}{L} dt = \frac{(V_{in} - V_{out}) * t_{on}}{L} \tag{2.2}$$

Delta current decrease:

$$\Delta I_{off} = \int_{t_{on}}^{T=t_{on}+t_{off}} \frac{V_L}{L} dt = -\frac{V_{out} * t_{off}}{L} \tag{2.3}$$

on time:

$$t_{on} = D * T \tag{2.4}$$

off time:

$$t_{off} = T - D * T \tag{2.5}$$

Where L is the inductor value, D the duty cycle and T the time period. Since the total variation of current has to be constant:

$$\Delta I_{on} + \Delta I_{off} = \frac{(V_{in} - V_{out}) * t_{on}}{L} - \frac{V_{out} * t_{off}}{L} = 0 \tag{2.6}$$

And through proper manipulation, it is verified that:

$$V_{out} = D * V_{in} \tag{2.7}$$

Since the duty cycle is a quantity always lower than 1, the stepping down of the output voltage is achieved.

The buck converter dual component is the boost converter (figure 2.4b) which, still being a DC-to-DC power converter, steps up voltage (while stepping down current) from its input (supply) to its output (load). In both these devices, "to reduce voltage ripples filters made of capacitors (sometimes in combination with inductors) are normally added to such converter's output (load-side filter) and input (supply-side filter)"[9][9]. Both these two configurations have aggressively reduced cost and are characterized by very high efficiency (up to the 95%).

As happens in the buck converter, the boost converter functioning is characterized by two phases, ON and OFF, according to the switch position. During the ON phase, the switch is closed and the inductor increases its current storing energy in the form of an electromagnetic field. During the OFF phase, where the switch is opened, the only path provided to the current is the external circuit mesh. Contrarily to the buck converter, the inductor is in series to the diode and therefore the current does not decrease, charging in fully the capacitor, hence the load.

It is verified, by the following formula, the relation between the current variation and the switch duty cycle which exists in the boost converter as it does in the buck one, providing the same voltage regulation parameter.

Inductor current variation:

$$\frac{\Delta I_L}{\Delta t} = \frac{V_{in}}{L} \tag{2.8}$$

Delta current increase:

$$\Delta I_{on} = \int_0^{t_{on}} \frac{V_{in}}{L} dt = \frac{V_{in} * t_{on}}{L} \tag{2.9}$$

Delta current decrease:

$$\Delta I_{off} = \int_{t_{on}}^{T=t_{on}+t_{off}} \frac{(V_{in} - V_{out})}{L} dt = \frac{(V_{in} - V_{out}) * t_{off}}{L} \tag{2.10}$$

Since the total variation of current has to be constant:

$$\Delta I_{on} + \Delta I_{off} = \frac{V_{in} * t_{on}}{L} + \frac{(V_{in} - V_{out}) * t_{off}}{L} = 0 \tag{2.11}$$

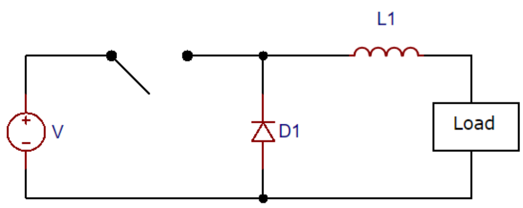
And through proper manipulation, it is verified that:

[9] "https://en.wikipedia.org/wiki/Boost_converter"

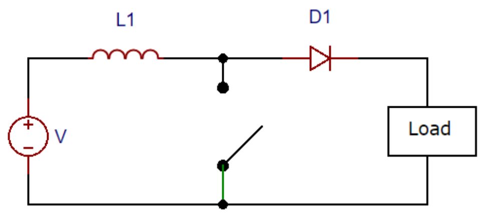
$$V_{out} = \frac{V_{in}}{1 - D} \tag{2.12}$$

Finally obtaining the stepping up of the output voltage ($0 \leq D \leq 1$).

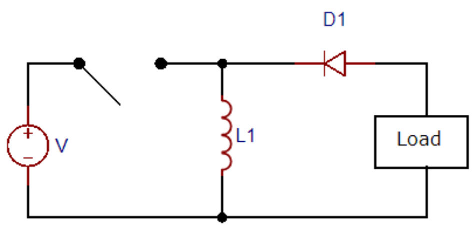
The buck-boost converter (figure 2.4c) is a DC-to-DC power converter which can either step down or step up the voltage from its input to its output. This device can be configured with two different typologies: the inverting topology and the combination between the buck and the boost converters topology. The first one is characterized by an output voltage which is of the opposite polarity of the input one; the second one is the most common configuration in which the device is nothing more than a buck stage followed by a boost one, obtaining the output voltage level initially described above.



(a) Buck converter



(b) Boost converter



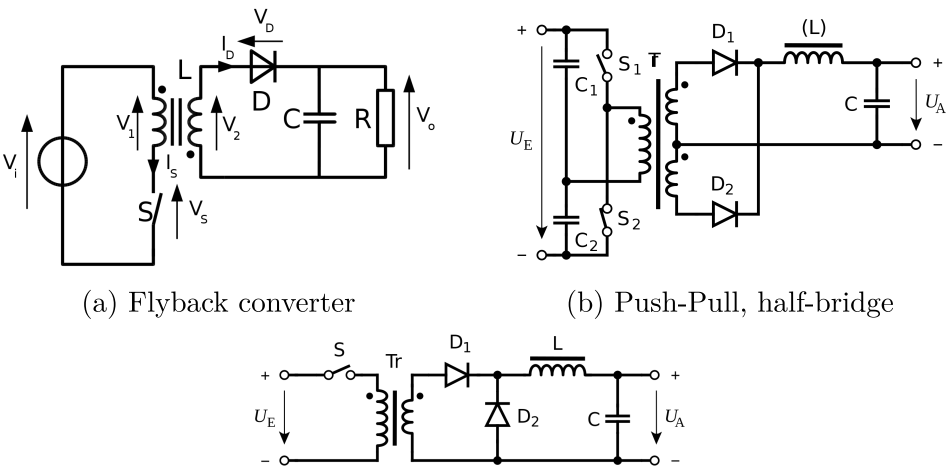
(c) Buck-Boost converter

Figure 2.4: SMPS: non-isolated converter, circuit schematic

Isolated converters

The isolated converters have the same blocks configuration presented in the previous section and therefore they include a transformer being able of always adjusting the output voltage at levels both higher or lower than the input one by simply modifying the turn ratio.

The flyback converter (figure 2.5a) is a galvanic isolated converter used in both AC-to-DC and DC-to-DC conversion. Its base scheme is the same of the buck-boost converter in which although the inductance is replaced by a transformer (two coupled inductors). For this reason, its function is quiet similar to the one of its non-isolated brother and it can be subdivided into two phases: the ON phase and the OFF phase. During the ON phase the first winding of the transformer is powered and the inductor starts to store energy, the diode is reversed biased (blocked) and the output capacitor supplies energy to the load. In the OFF phase, when the switch is opened, the primary winding current and the magnetic flux drops, hence the secondary voltage is positive and the diode is forward-biased allowing current to flow in the transformer which can recharge the capacitor and supply the load. The flyback converter is the most used device in the low power application industry with efficiency of the order of the 90%.



(a) Flyback converter

(b) Push-Pull, half-bridge

(c) Forward converter

Figure 2.5: SMPS: isolated converter, circuit schematic

The forward converter (figure 2.5c) is a DC-to-DC power converter and it may appear very similar to the flyback converter. The main difference lies in the technology used for the transformer: in the forward converter "the transformer is based on same-polarity windings, higher magnetizing inductance and no air-gap"[10][10] and therefore cannot store large amount of energy (as instead happens for the inductors in the flyback converter) which is directly transferred from the input to the output of the device. In this way, the forward converter is usually characterized by higher efficiency than the flyback one, although less exploited. The push-pull converter (figure 2.5b) follows the same behavior of the previously described isolated converter. It is a DC-to-DC power converter which main distinctive characteristic is the use of push-pull transistor circuit as source for the current powering the transformer.

2.4 SMPS and Linear power supply comparison

As previously stated, the main incentive for the outclassed of the linear converter in favor to the Switching one, came from the need, in many field of application (aerospace is just one among others), of more compact, light weighted and efficient converters. Nevertheless, SMPSs still have some disadvantages w.r.t. linear regulator and in peculiar applications they could not be preferred as first or better choice.

The main categories useful to highlight the differences between the two kinds of converters are listed below and afterwards analyzed in detail:

- Physical dimension and weight;
- Output voltage;
- Efficiency;
- Heat and/or power dissipation;
- Circuital complexity;
- Radio frequency interference;
- Electrical noise generation;

[10] "https://en.wikipedia.org/wiki/Forward_converter"

- Power factor and inrush current;
- Economic cost.

One of the most significant parameter influencing the dimension of converters is the operating frequency and in particular the operating frequency of the transformer. Linear converter is characterized by low operating frequency (50 or 60 Hz) and therefore it yields a considerably higher dimension and weight w.r.t. the switching converter which operates commonly in a frequency range between 50 kHz and 1 MHz (capacitors dimension in relation with frequency).

The output voltage in both the application can vary properly and widely depending on the circuit configuration. The real difference between the linear converter and the SMPS lies in the effect of the load on the output voltage fluctuations: in a linear converter, if unregulated, the voltage varies significantly with the load; on the other hand, for switching converter the output voltage varies little with load (theoretically constant until the critical resistance value of the load and still characterized by very low value).

As previously stated, the efficiency of a linear converter critically depends on the difference between the input voltage value and the output one. The normal working condition of this device is to control the output voltage by dissipating excess power as heat. This results not only in unbearable heat and power dissipation, as the output increases, but also in a reduce value of efficiency ($30 - 40\%$ and almost always not higher than 50%). The only situation in which the efficiency of a linear converter can be higher than the aforementioned values occurs when the gap between the input voltage and the needed output is very limited. The SMPS presents a higher efficiency partly due to its reduced dimension and the use of transistors as switches: the transistor is either on or off which means that the only contribution to heat and power dissipation can be approximated of belonging to the non ideality of the circuit components. For the switching converter, efficiency can even reach 95% (more commonly values between $60 - 90\%$).

The most significant disadvantage of SMPS is its circuital complexity. A linear regulator, in fact, not considering the very simple unregulated one that can even be just formed by a diode and a capacitor, presents still simpler circuit than the switching one. Even if the elementary blocks of the two devices perform almost the same functions, the one belonging to the switching regulator yields a higher grade of complexity. Moreover, since the SMPS works at higher frequencies, and therefore is submitted to more severe problems related to ripples and filtering, it contains lots of noise/interference reduction blocks that are absent in the linear converter.

Radio frequency interference and electrical noise are the other critical disadvantages of SMPSs. The generation of both these phenomena is strictly linked to the device's operating frequency. Since the common switching frequency has its lower limit at 50 kHz, noise is generated intensively by the on and off components (transistors) and the current switching. This can cause in an unfiltered output a severe presence of ripples which could damage digital circuits or produce noise in audio circuits. The introduction of EMI (Electro-Magnetic Interference) filters and RF shielding becomes also compulsory due to the radio frequency interference. On the other hand, high frequency interference is unlikely produced by linear converter (mild high frequency interference can occur using AC rectifier under heavy current loading) and noise and ripples are confined to at most frequency components with values double of the operating ones (still not higher than $100 - 120$ Hz).

The power factor is defined as the ration between the real power absorbed by the load and the apparent power flowing in the circuit, and it is a dimensionless number in the closed interval of -1 to 1. When talking about real power, the reference is to the product between the instantaneous voltage and current; apparent power is the average product of voltage and current which is higher than the real one due to energy stored in the load and non-linearity. A negative power factor indicates that the load is generating power which is flowing back to the source. The lower is the power factor value the higher are the Joule losses and the risk of overload and failure of the system. Moreover, the power factor is strictly related to the efficiency of the converter and for this reason should always be as high as possible. Usually both linear and switching converters are characterized by low or medium power factor. Without the implementation of proper solutions, switching converters can even present power factor lower than linear converter (current spikes at the peaks of the AC sinusoid); for this reason, power factor correction, also referred to as PFC, is usually implemented and it supplies or absorbs reactive power, reducing the apparent power component in the power factor.

Another relevant electrical parameter that has to be taken into account when dealing with converter is the inrush current. The inrush current is the maximal instantaneous input current drawn by an electrical device when first turned on. It is especially important in converters since devices such as transformers can draw several times their normal full-load operating current when first activated (in transformers the inrush current can even be 10 to 15 times higher than the normal operating current flowing for several cycles). Hence, knowing the amount of inrush current becomes imperative in order to understand when and if implemented, in the device, protection and control equipment set to preventing failing and over-current problems. Switching regulator yields larger inrush current than linear converter and therefore will present a higher amount of protection circuit solutions. Finally, a parameter which cannot be neglected is the economic cost of the two devices. In the early twentieth century, when the MOSFET technology needed for the switching regulator was still new, SMPS cost was extremely high and in most case prohibited and the linear converter one was, instead, highly affordable. Nowadays, the trend is inverted: component costs have dropped and, especially when trying to reach the same performances of SMPSs, linear converters have become more expensive mainly due to material cost.

In conclusion, the comparison between Linear power supply and Switched Mode Power Supply clearly shows that in terms of size, efficiency, dissipated heat and reachable power and output voltage SMPS is the only possible choice. The circuital complexity needed to reach such performances has to be paid with radio frequency interference, electrical noise and a general higher difficulty in managing the outcoming signal of the regulator with the necessity of further blocks delegated to overcome these drawbacks. There are still few cases in which the linearity and simplicity of the linear regulator can be preferred, but these situations happen more and more often only in laboratory and initial studies environment. Foreshadowing the incessant integration of MOSFET and similar technology, it is logical to suppose that SMPSs will continue to cover an increasing relevant role in the power supplies industry.

2.5 Other SMPSs features

In the previous sections of the dissertation, the term ripple has often been used linked to the drawbacks of the high operating frequency of the SMPS and to the electrical noise by it generated. There are other electrical parameters of the switching converter, other than ripple, which deserve a mention and analysis since can help to understand how SMPSs should properly work. Ripple, current clipping, immunity to power comeback and overcurrent/short circuit protection are some of the many additional features that should be taken into account when talking about power supply.

In the transformation of the AC input of the power supply into a pure DC stream, the rectifier, which is the block responsible of this process, cannot perform it without some kind of error. What usually happen is that the output of the rectifier will be more similar to a pulsing voltage signal (that continues to follow the initial periodical oscillation of the current) that oscillates around the average output voltage value. This oscillation is the mentioned ripple as shown in figure 2.6. In power supplies that have to transit from AC-to-DC, ripple is inevitable and it can only be reduced below tolerable levels that depend on the kind of application in exam. Hence, common practice is to reduce as much as possible this unwanted effect that not only disrupts the linearity of the device's output but also represents a waste energy component. As previously stated, the filter following the rectifier has the aim of eliminating the ripple or at least of reducing it below acceptable levels. In the figure below, the characteristic tooth saw trend of the ripple is shown.

There are situations, involving power supplies, in which there could be the necessity to control and limit the produced current value. When this clipping current process is a necessity, SMPSs have the possibility to set a maximum value of current. Figure 2.7 shows clearly how this process is realized: first there must be the setting of a threshold voltage called V_k, then, through the decrease of a variable load resistance, the current will start to increase reaching a constant value paired with the exact threshold voltage value set previously. As shown in the figure, as the load resistance decreases the I_{max} will remain constant but the V_k will progressively tend to zero (exactly dual trend as the resistance increases above its threshold value R_L).

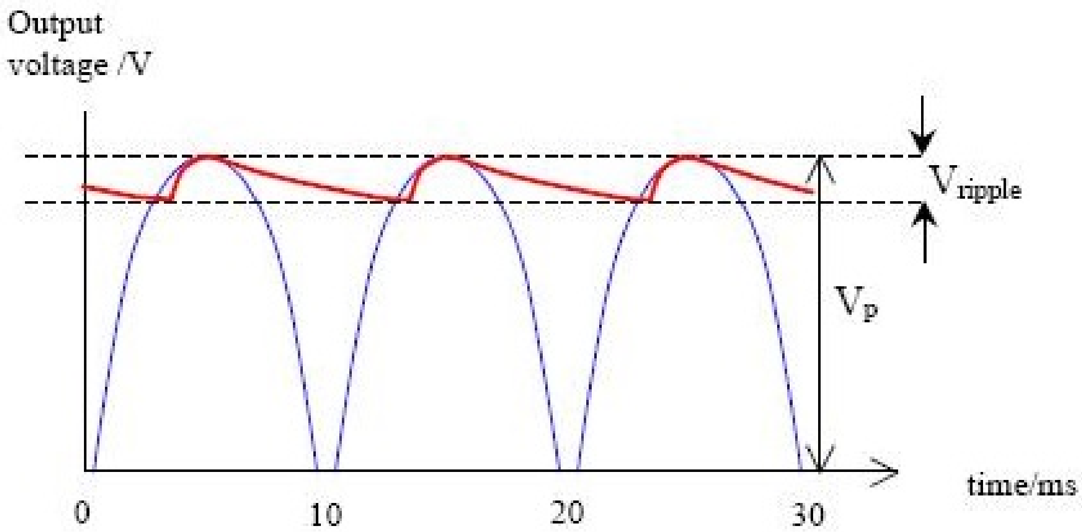


Figure 2.6: Example of ripple in the rectification process

Overcurrent and short circuit protection are probably the simplest protection devices integrable in switching converter. The devoted circuit executes a measure of the current and when its value exceeds a preset limit, the tension is rapidly decreased preventing damages to the rest of the device. The same principle is applicable to the short circuit protection where, when needed, the output is abruptly disconnected preventing hazardous situations.

Protection must not be implemented only for overcurrent and short circuit phenomena. Another effect that can jeopardize the integrity of the SMPS is the possibility of power comeback: a generic amount of power can be reflected back into the circuit damaging it. This power reflection can be caused not only by external sources but also by storage of energy within components already existing inside the SMPS (load, capacitors etc.). The harm that this phenomenon can effect on the device could even be irreparable and for this reason a very common features that SMPS must have is the immunity to power comeback.

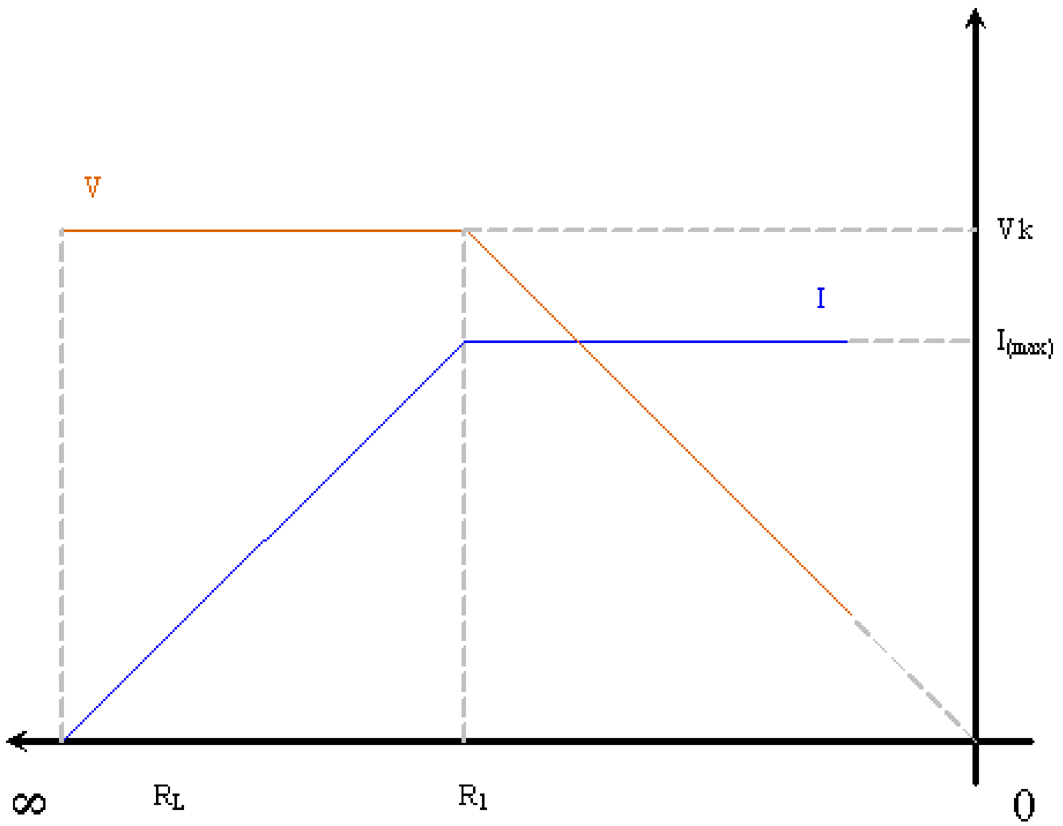


Figure 2.7: Voltage and current behaviour when performing current clipping

Chapter 3

Importance and reasons for testing electronic board

When talking about the test of electronic devices, it can be extremely simple to fall in misunderstanding since the topic is quiet vast and various. An electronic board, of which this section of the dissertation is about, can be tested principally in two diffent stages: a first one used to verify the EMI and electrical stress compatibility and compliance of the board itself; and a second one to be executed after the components forming the device have been mounted on the surface board, called assembly level test.

It is of the utmost importance, common practice nowadays, to execute the assembly level test (to which from now on reference will be made with the simple word test) in the early production stage of the product in order to save possible considerably cost later on the fabrication chain. Detecting a board defect or malfunction could in fact be catastrophic when the product is, for instance, already in the market or in any of the advanced stage of its production. Solving any possible problem when the electronic board has been freshly implemented becomes the best financial and operating choice.

Setting aside the economic aspects linked to electronic board testing, any electronic products, no matter with how much care and attention has been designed and built, is subjected to defects and issues and therefore it is impossible to think of skipping its test.

3.1 In-Circuit Test (ICT) and Functional Test

The importance of testing electronic board has been ascertained. It becomes now imperative to identify the different and more common kinds of tests executable over an electronic component.

Tests can be divided into two main macro categories: In-Circuit test or ICT and functional test. Their most significant difference is the level of control that they perform in terms of logical depth: as designing a circuit it is possible to distinguish a lower logical level and an higher one; following the same analogy, the two tests face the check of the electronic board on two different levels. To clarify, ICT tests the values and functionality of the single components mounted onto the board surface; functional test tries to understand if the board, in its entirety, succeeds in performing the logical and macro actions for which it was designed.

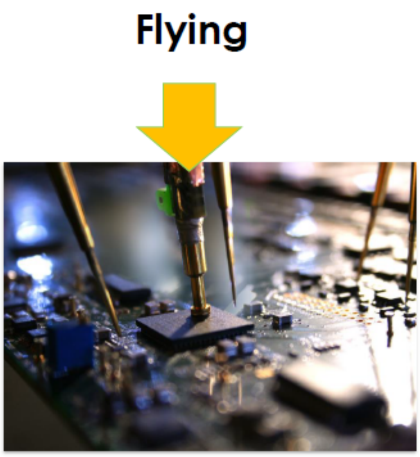
In-Circuit test are characterized by a more technical classification and diversification w.r.t. the parameters of the components under test and therefore can be easily classified and itemized; same thing does not stand for functional test which, for its very nature, changes accordingly to the DUT (Device Under Test) in exam. In the following, the most important ICT are reported:

- In-Circuit Test-Off: test of the electrically testable components. It measures singularly the components mounted on the board evaluating shorts, opens, resistance and capacitance values and other significant quantities which can be useful to understand if the board was assembly correctly and following the wanted specification. This is usually the first implemented test because it can clearly point out whether or not the electronic board was fabricated correctly. Depending on the elements inter-connection, performing the ICT-off on all the needed components could encounter the onset of various problems ranging from the isolation of a certain object from the rest of the circuit (an example can be the test of a resistance value within a voltage divider) to the need of eliminating the contact resistances (when testing very small resistances). The test is carried out with the DUT not powered up.
- In-Circuit Test-On: this test is the *ON* counterpart of the ICT-off. The DUT is powered up in nominal condition and all the components that need power to

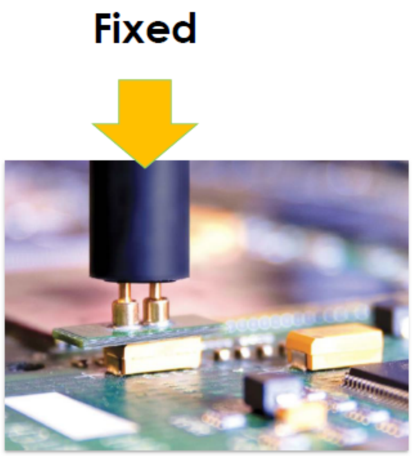
be correctly analyzed are tested (digital gate, operational amplifier, frequency related components). The ICT-on is usually more complex to set up than its off sibling but easier to debug.

- Junction scan: test strictly related to the analysis of diodes. It evaluates the pin welding presence of the components by measuring the junction diode. Without powering the board, the test can evaluate the presence of fabrication defects, such as for instance open pins, by estimating the forward voltage of clamp diodes.

- Electro scan: as the junction scan it evaluates the pin welding of the board components and it is especially used when the junction diode cannot be scanned because of electrical constrains. The electro scan use an antenna with amplifier to measure the signal propagation, increasing the total test coverage of the board since it can reach those component unaffected by the junction test. The tool (antenna) used by the test is strictly required and has to be integrated in the ATE (Automatic Test Equipment) and used solely for this purpose. In figure 3.1, the two kinds of electro-scan probes related to the Flying Probe system and Bed of Nails system are shown.



(a) Flying electro-scan probe



(b) Fixed electro-scan probe

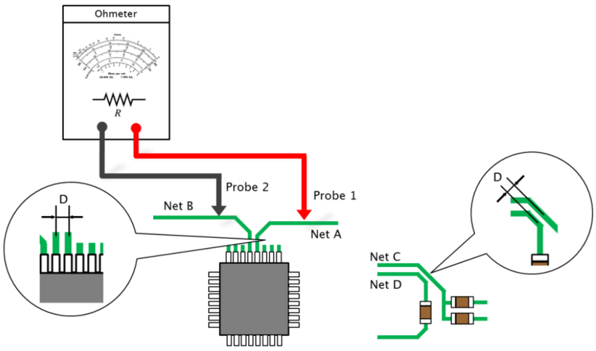
Figure 3.1: Probes for the electro scan test displayed on board[13]

- Short Test based: this test, as suggested by its name, is specifically realized in order to detect the presence of short circuits within the printed board net. It is further divided into Short Test Ohm based (STO), Short Test Capacitance based (STC) and Short Test Impedance based (STZ, where the Z stands of course for impedance). The STO is required by the user normally on small selected portion of the board. The test detects short circuits between the nets in the selected area through the search of resistance value of 10 Ohm. The STO most significant drawback is the limited test speed w.r.t. the dimension of the chosen area and the numbers of pins and nets within it. The test can in fact be configured to measure the resistance value between adjacent pins or nets and it is clear how time demanding this operation can be (an ICs composed of 128 pins will need, in order to cover all the possible identifiable shorts, 8128[1] checks). Since the STO will require an awful amount of time to be executed, the STC is used instead. The Short Test Capacitance is the direct response to the STZ time consuming problem. The shorts search is realized comparing the capacitance value between each pin, net and a referenced ground level. The capacitance value is previously measured on a "golden" board and this results is stored to be afterwards compared with the actual board capacitance measure. If the obtained value differs from the one stored it means that in that position a short could be present. Only for the nets that fails the STC, the STO is performed to check is indeed a defect has occurred. Reducing the number of nets that has to be checked by the STO the test time in considerably reduced. In figure 3.2, the STO (3.2a) and STC (3.2b) measure schematic type are shown.

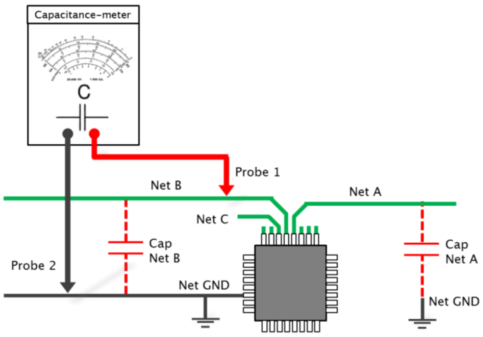
  Hence, STC continues to have the need of run a further STO to identify which specific connection is affected by the defect . It could be taken, for instance, the case of an IC with a used but not connected pin short circuited with

[1]where the total amount has been computed as

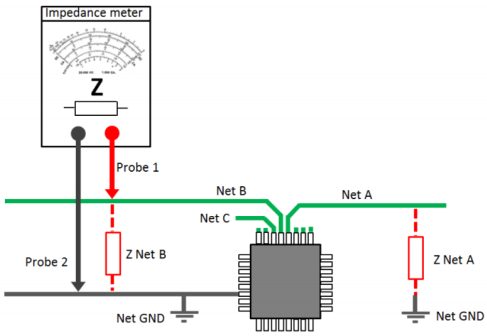
$$C_{(n,k)} = \binom{n}{k} = \frac{n\,!}{k\,!\,(n-k)\,!} \qquad n = 128 \; k = 2 \tag{3.1}$$



(a) STO, measure schematic



(b) STC, measure schematic



(c) STZ, measure schematic

Figure 3.2: Measure schematics of different Short Test based[13]

another pin connected to other devices (figure 3.3). In order to understand the STC process, let assume to have a 2 pF capacitance for the non-connected pin and a 22 pF capacitance for the connected one. In the presence of a short circuit the measured value of the first will yield a 22 pF value instead of a 2 pF, generating an anomalous value completely different from the stored one.

The last Short test technique to be analyzed is the STZ or NZT (Nodal Impedance Test) which, as the name suggests, is based on nodal impedance measurements. The NZT solves a critical issues of the STC which is the inability of said test to obtain a net capacitance measure when the connection from the net itself and the ground level is absent of a capacitance (for instance a ground link through a simple resistance). For this particular cases the NZT, since does not search for a capacitance value but a more general impedance value, can still provide the needed short circuit test. The test follows the same idea behind the STC process: two values of impedance at different frequencies (80 kHz and 160 kHz) are measured and stored. When a non-consistent measure is detected, a further STO is performed on the incriminated net. The other main advantage in performing NZT over STC is that the first one does not required any previous measure of correct parameter values for the next comparison (no "golden" board used).

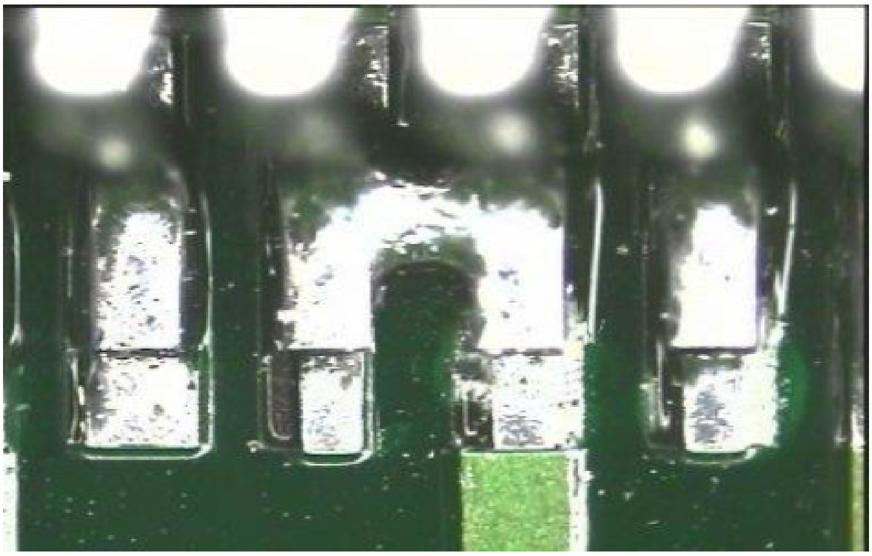


Figure 3.3: Example blow of short circuit[13]

When performing the test of a DUT, a combination of the aforementioned In-Circuit Test is chosen with a special regard towards maximizing its coverage and decreasing as much as possible its time. It has been verified that the optimized run is achieved through a sequence of ICT-Off and STZ. Only if the STZ results in fails than a further STO is performed on a restricted group of nets. After that, depending on the components of which the DUT is composed, an ICT-On test could be required.

Functional tests cover an all other verification area w.r.t. the In-Circuit Test. Functional test, as already stated, cannot be categorized as ICT since their aim changes from one DUT to another according to the required specification. Still, it can be said that they are "based on the static/dynamic behaviour during the normal functioning of the board"[2][13].

Figure 3.4 shows the flow chart explicating the logical process to be followed in designing a functional test.

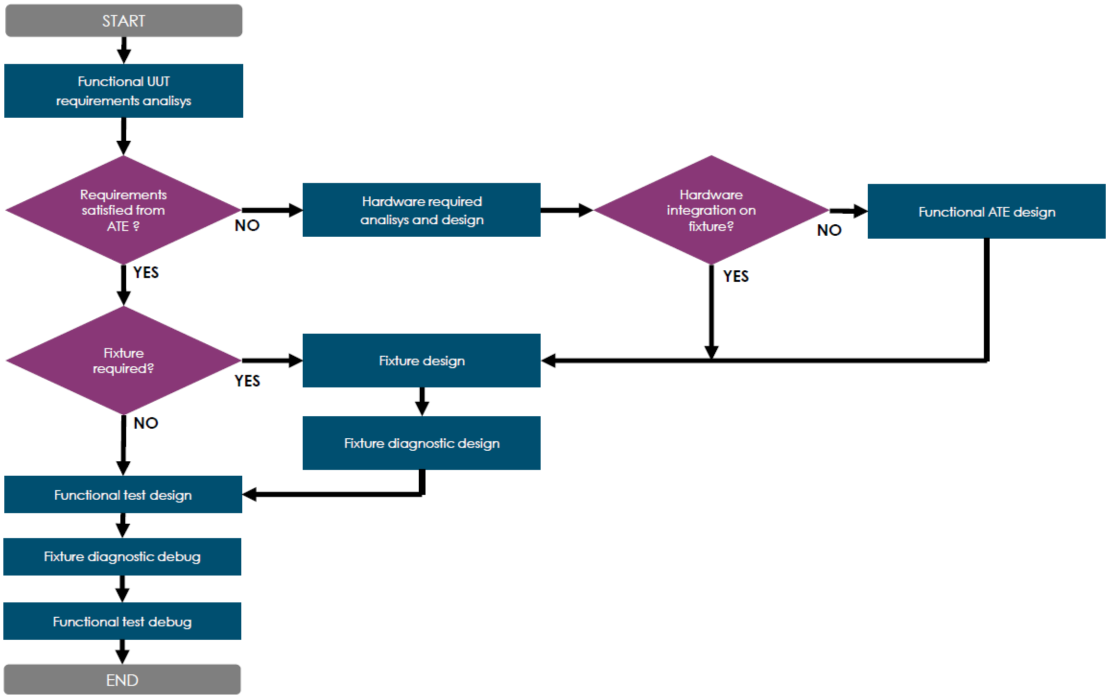


Figure 3.4: Design of a functional test application flow chart[13]

Before performing the actual test, preliminary steps must be analyzed and satisfied. Functional test is originally and usually initiated starting from the test specification (Chapter 5) which are generally provided by the same person who design the DUT (client) and therefore it is essential to preliminary check that the client's requests can be met.

The *feasibility analysis* is responsible for the definition of the responsibilities, equipment and time needed to complete the project. This step is based on the information supplied by the client but it does not compulsory need of taking into account the technical test specification. Then, a *testability* control of the board must be executed: the test required by the customer must be actually performable on

[2]"Test Functions, Spea Academy training book. Author: D. Del Greco, Date: 05.09.2014"

the board meaning that the DUT has to satisfied the requirements needed for that specific test. From the flow chart displayed in figure 3.4, the steps necessary after the UUT (Unit Under Test) requirements analysis are clear: the ATE compatibility with the required tests must be verified as well as the need of a fixture (described in detail in chapter 4). When necessary, modification, both software, hardware or mechanical, can be implemented both on the ATE or the fixture. When all the tools needed for the functional test to be performed are ready, it can be executed and debugged.

Chapter 4

State-of-the-art of test: Flying Probe & Bed of Nails

An *Automatic Test Equipment* or ATE is a set of electronic instruments which main purpose is to verified the correct functionality of a largely kinds of electronic devices. The device inspected by the ATE is usually called *Device Under Test* or DUT (other acronyms used are EUT, Equipment Under Test or UUT, Unit Under Test). An ATE not only provides the execution of the measurements needed to control the correct behaviour of the DUT, but also executes, when required, the diagnostic to such tests. The diagnostic part of the ATE work is probably its most difficult task since it must identify the reason behind the faulty action of the DUT. Lastly, duty of an ATE is to provide a final evaluation of the test results that can immediately highlight if the test is either a pass or a fail.

The complexity of an ATE system is directly proportional to the DUT characteristics and since there are a significant various kind of possible DUT, when talking about ATE we could refer to systems as simply as a digital multimeter to as complex as Multi-core tester architecture. Possible DUT can be:

- Simple components (resistances, capacitors);
- Printed Board Circuits (PCB);
- System on Chip (SoC);
- Integrated Circuits (ICs);

- Packaged electronics components;
- Wafer Testing.

Depending on the type of DUT listed above, the whole testing process could required a non negligible amount of time and considerably cost on the overall electronic component production expenses. For this reason, it has become essential to understand what kind of tests each devices absolutely need and which one could not be performed still leaving a sufficiently coverage on the correctness of its functionality. A cost-benefit analysis must always be implemented in order to understand if the cost of this equipment could prevent, for instance, an higher cost caused by a not detected malfunction and therefore integrated in the mass production of the DUT and ultimately on the market. Truthfully, it must be said that ATE system are implemented with a special regard to repeatability and speed, two factors that significantly decrease the equipment cost.

The most important types of ATE are the Flying Probes systems and the Bed of Nails systems analyzed more in depth in the following sections.

4.1 Flying Probe system

Figure 4.1 shows the latest SPEA's 4080 flying Probe tester. Flying Probe test systems use electro-mechanically controlled test probes to make contact with the board to be tested. This kind of tester is also called FICT which stands for *Fixtureless In-circuit Test*. The name derived from the main characteristic of the Flying Probe which is also its main difference from a Bed of Nails system: the lack of a fixture and, as already stated, the use of flying probes for testing the electronic board. The number of flying probes is variable and can change depending on the sophistication of the tester; the 4080 depicted in figure 4.1 is equipped with 8 probes, less recent versions of the same machine can deployed a configuration of 6 or 4 probes.

The electronic board to be tested can be inserted manually and directly inside the *probe test area* (within which the chassis is placed) or can be inserted inside the *board loading area* equipped with a remotely controlled in-line mechanism. Sometimes when the board to be tested has such a peculiar conformation that results in the impossibility to insert it in the chassis as it is, pallets (mechanical support) can be used. Pallets can be designed ad hoc for the a specific DUT or can be universal.

Figure 4.1: 4080 Flying Probe system[11]

After the positioning of the DUT inside the tester chamber, the actual test can begin. Flying Probe tester logically contacts the electronic board by probes: the probes position themselves on the test pad (contact point designed for the test) and touch it with special pins. The smallest size of test pad to be contacted and the inclination depend on the characteristics described in the data sheet of the test system. The availability of enough test pads, to sufficiently cover the DUT surface, conditions the choice of the probes contact side. Normally the best side to contact the board, and hence to achieved its maximum coverage, is also the one containing the higher number of test points. When this situation cannot be reached exploiting just one side of the DUT, the dual side probing is necessary (of course exploitable only if provided by the tester at disposal).

4.2 Bed of Nails system

Figure 4.2 shows the 3030 Bed of Nails tester. Bed of Nails systems use a mechanical interface between the system and the board to be tested, called a testing adapter or, more commonly, a fixture. As previously said the need of the fixture is the main discriminant between the Flying Probe and Bed of Nails systems.

In Bed of Nails systems, the fixture (figure 4.3a) acts as the interface adapter between the board to be tested and the system. The fixture contains specifically positioned test points (probes), in order to contact the board test points. For an ICT-type test (In-Circuit test), the fixture will contain at least one probe for every board net. Each probe is connected to the system interface of the fixture (which reproduces the system interface) by means of connections made with wire-wrap type cables.[12]

Figure 4.2: 3030 Bed of Nails system[11]

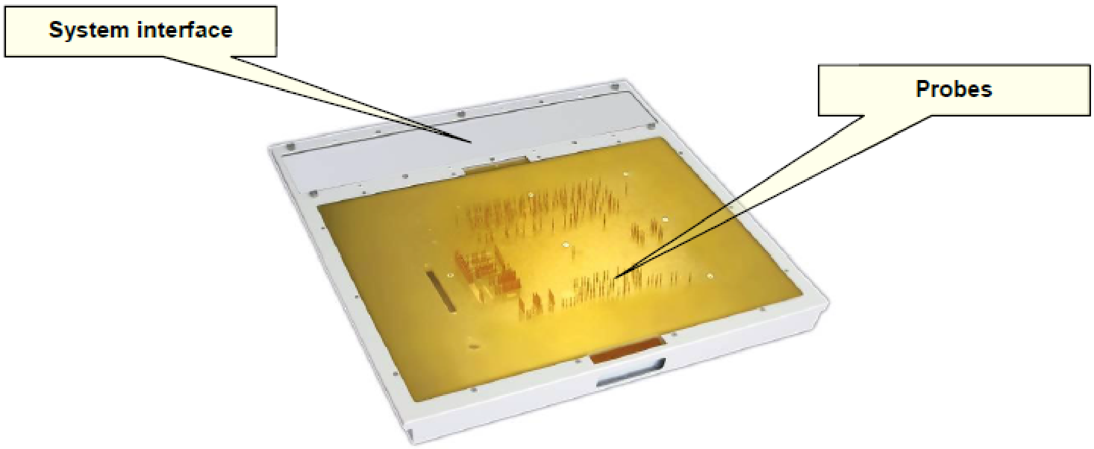
The fixture must be mechanically designed and therefore imply a cost that, depending on the structure and the material, can oscillates between 20000 € and 30000 €. Moreover, it is not the only additional item that need to be designed when testing with the Bed of Nails: a presser plate (figure 4.3b) is a device used for guaranteeing that the contact between the board and the fixture's test points in effective. It can be manually modifiable with mobile pressure rods which run along rigid bars positioned according to the points to be pressed onto the board to be tested, or with fixed rods inserted on a presser plate in Plexiglas, specially-made for a single board.

It is clear, at this point, that the Flying Probe tester, without the need of designing board customized fixtures and presser plates, will result in a cheaper equipment which outclasses the Bed of Nails system. Although the lack of fixture is indeed a strong factor to be taken into account; it is not the only one and further analysis must be carried out in order to correctly identify pros and cons of the two ATEs and if one of them can indeed be classified as utterly better than the other.

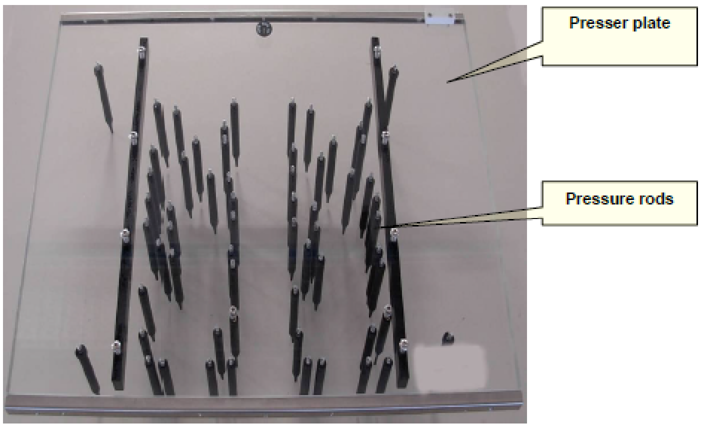
4.3 Flying Probe and Bed of Nails tester comparison

It is quietly correct to state that the Flying Probe tester is born as an evolution of the Bed of Nails. The lack of fixture is indeed a significant advantage in reducing the expenses linked to the Bed of Nails testing and it is not the only gained improvement. In the following all the Flying Probe advantages over the Bed of Nails are listed:

- No board accessibility limits;
- Fixture design not necessary;
- Design modifications can be quickly managed;
- Easier and faster Test application development;
- Increasing in the test coverage;
- In-Circuit and optical test are performed on the same system.



(a) Example of Bed of Nails fixture



(b) Example of Bed of Nails presser plate

Figure 4.3: Additional Bed of Nails mechanical supports[12]

As previously said, contacting by probes allows an higher probability of obtaining a full coverage of the board's surface w.r.t. the exploitation of fixture. It has to be said, however, that even flying probes have their limits: the default "jump" of a flying probe is of 7 mm meaning that if there are PCBs component higher than this height they have to be confined in a so called *no-fly zone*. This portion of area, however, could prevent the access to standard under 7 mm components if they happen to be near its proximity. Another constrained in accessibility linked to flying probes is their inclination: depending on their position on the $x - y$ axes, the probes will be characterized by a different inclination (13° for outer probes and 3.8° for inner ones) which, when contacting components in the shadow of very high ones (even outside the constrains of *no-fly zone*), could cause the impact, and hence breakage, of the probes tips (figure 4.4).

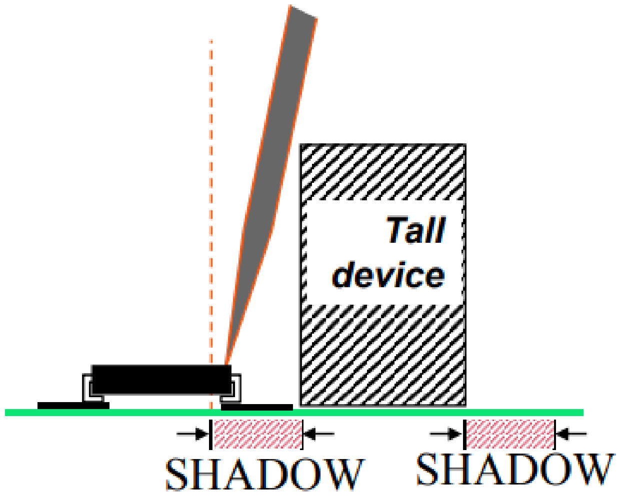


Figure 4.4: Contacting by probes: example of inclination problem

The fixtureless aspect of Flying Probes tester has been exhaustively discussed. Related to this point, however, there is also another advantage over the Bed of Nails which is the possibility to manage quickly any kind of design modifications. This feature cannot logically be exploited when a fixture is participating to the test process since design changes would mean the re-fabrication from scratch of the fixture. Flying Probes tester can instead simply modify by software (test application program) the contacting position adapting to any board layout modifications. Moreover the whole development of the Test application program is considerably simplified and speeded since the mechanical implementation of the fixture is not necessary.

Customize test techniques for Flying Probes tester have also been realized in order to be able to increase the test coverage. In particular the NZT (already described in chapter 3) is a net-oriented test technique developed in order to reduce the test time for flying probe testing up to 80%, while guaranteeing 100% of short circuits coverage.

Last analyzed advantage is the possibility to performed on the same system both the In-Circuit and optical test. Flying Probes system are in fact equipped with a various number of optical cameras (depending on the sophistication of the tester model). In this way, optical inspection and in-circuit test are automatically combined in order to get the maximum coverage.

From the previous dissertation, it may seem how Flying Probes tester has only advantages over the Bed of Nails system and therefore it should always be preferred to it. Obviously this is not true: the main disadvantage of using Flying Probe tester over Bed of Nails one is the test time. For test time we are referring to the amount of time needed only for executed the ICT and all the other possible tests, not considering the fabrication fixture time that will alter the perception of the problem. It is clear why Flying Probe test requires more time to perform the same task than a Bed of Nails: the flying probes have to literally move from one point of the electronic board to another and this requires a non-negligible amount of time especially for particularly complex circuit. The other significant Flying Probe system disadvantage to be taken into account is the amount of sustainable production: most of the Flying Probe testers are not suitable for mass production being more often used for low or medium production.

All this considerations, however, stands correct only for generalization purposes and cannot center which choice of ATE is the correct one. Yet again, this aforementioned choice must be carried out weighting a substantial amount of factor related not only to the DUT characteristic but also to the kind of tests and the time that are necessary and affordable to sustain.

In the specific case of the electronic DUT protagonist of this dissertation, a Flying Probe ATE cannot be used: its particular design, in fact, is not suitable to be tested through the use of flying probes (test points not reachable and in most cases not even presents) and most of the required tests are to be performed with the UUT *packaged in* and therefore not contactable by the flying ATE. This means, obviously, that the ATE to be employed must be the Bed of Nails. Furthermore, the test requirements to be met in order to properly check the DUT correct functioning need the implementation of a series of additional electronic modules which are not usually foregone in a standard fixture, hence the exploitation of the other kind of functional fixture described previously.

4.4 Brief ATE's Hardware overview

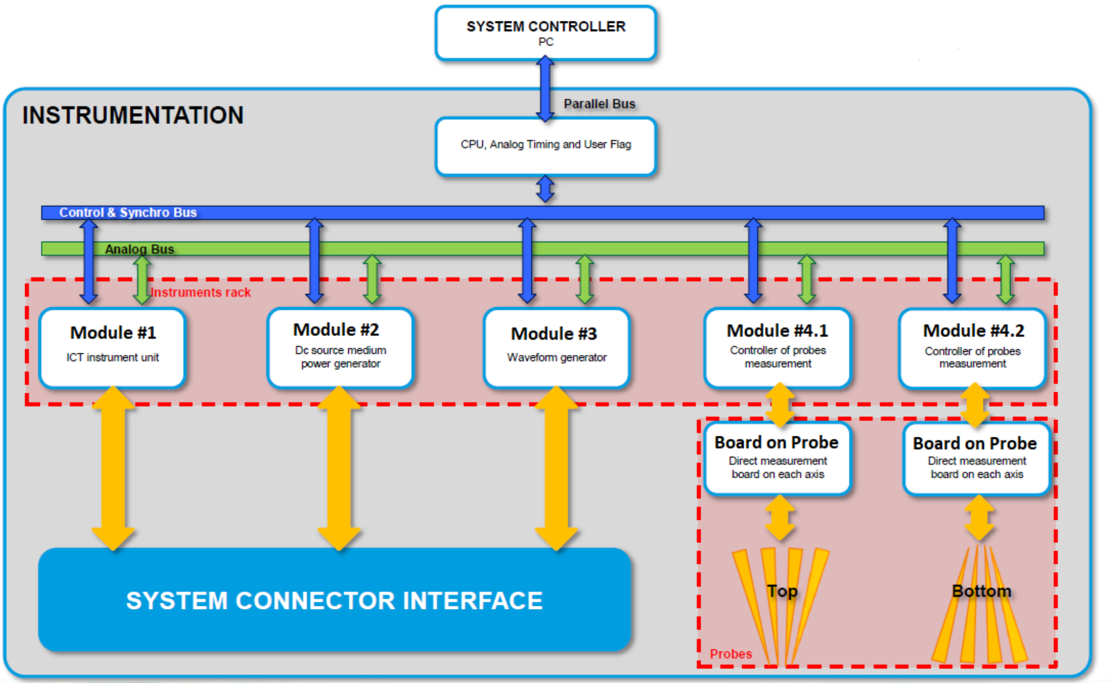


Figure 4.5: Example of ATE system architecture, Flying Probe

Figure 4.5 shows an example of ATE's system architecture schematic. The example is clearly related to a flying Probe tester with the double side contacting by probes. The different modules are usually inserted in an instruments rack and connected between each other (when needed) and the external architectural elements (CPU and system controller) through two buses: the Analog bus (green connection in figure 4.5) and the Control & Syncro bus (blue connection in figure 4.5).

Figure 4.6 shows the Bed of Nails counterpart of Hardware architecture. The connection between modules, as in the previous case, happens through the two Analog and Control & Syncro bus. It can be seen in the figure the dedicated Power Supply Rack with the CAN controller and the programmable power supply.

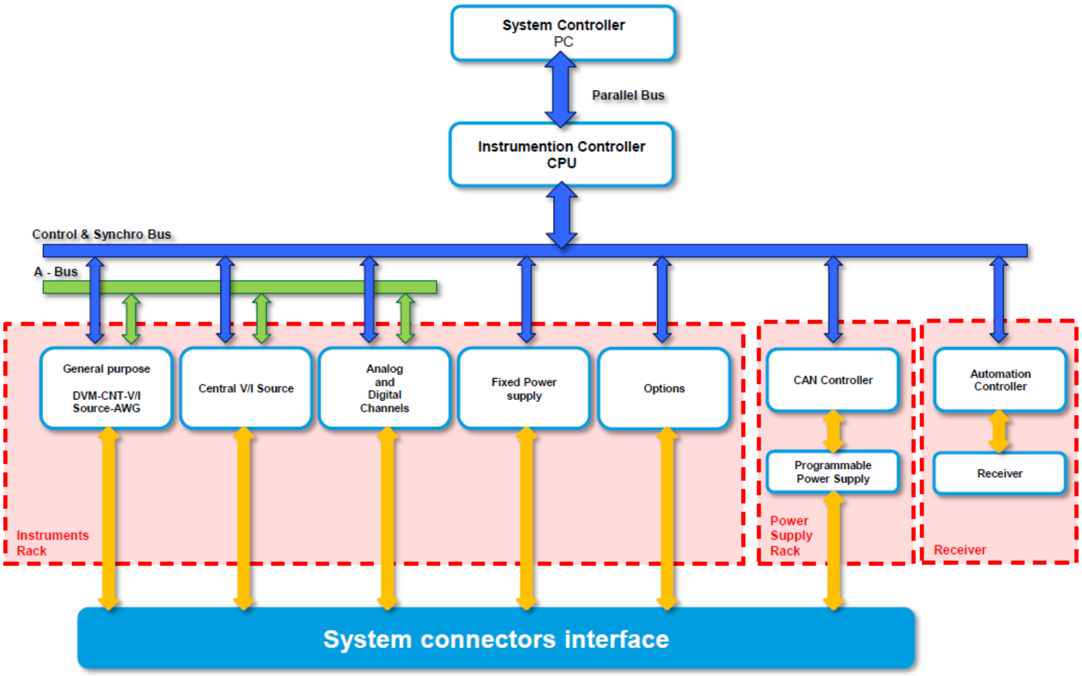


Figure 4.6: Example of ATE system architecture, Bed of Nails

In table 4.1 a summary of possible ATE's electronic modules with their description and function is reported. The content of the aforementioned table is to be considered purely as an example since each ATE could be composed of a largely combination and number of internal modules.

Table 4.1: Electronic Modules

Instruments	Description	Functions
Module #1	ICT Instrument Unit	DC Source, DVM, Digitalizer, Internal Signal Modulation, IC test memory
Module #2	DC source medium power Generator	DC Source, IC Test memory
Module #3	Waveform Generator	AC Source, IC Test memory
Module #4	Controller of probes measurements	/
Module #5	Fixed Power Supply	Application coding signal, UUT code, Fixture Code, FPS1, FPS2, FPS3, FPS4
Module #6	Analog and Digital channel	Channel driver, Sensor, Pattern Genrerator

Chapter 5

Description of a Test Requirement Specification

The fabrication of an electronic board (or any other kind of electronic device) does not end when the single components of which the structure is composed are upon it placed. To correctly verified whether the fabrication process has been performed correctly, further test on the electronic device must be executed. As already said in the previous chapters, there are a largely various type of tests, which can range over from *in circuit* to functional test, each of them dedicated to verified different aspects of the electronic board. Once the importance and inevitable need of performing electronic board test has been stated, it becomes fundamental identifying how this kinds of test must be performed and before what types of standards and regulation they fall.

This section of the dissertation will focus on the functional test and hence on the documents needed in order to perform it: the Test Requirement Specification (TRS).

5.1 Example of a specific power supply Test Requirement Specification

A Test Requirement Specification is a technical document provided in phase of estimation by the client. The prime aim of this paper is to firstly evaluate if the request made by the customer can actually be satisfied. Regarding power supplies, even if, depending on the DUT, the client's requests could change, a general organization of the TRS can almost always be identify and therefore explained.

For proprietary reason the tested Power supply board specific final application will not be disclosed. The DUT, however, still responds to the need of a TRS which is organized as listed below:

- Architecture Requirements;
  - PSM architecture,
  - Input and Output Requirements,
  - Control Signal,
- Test Requirements;
  - Instrumentation,
  - Test set-up,
  - Electrical tests,
- Test Data Record.

The Power Supply comes along with a further TRS relative to the test of the Power Supply Board (PSB) only. The main different between the one mentioned above (Module) and the board one is that some tests must be performed directly on the board and some other, instead, with the DUT *packaged in* and accessible only from its output and input connectors. This last configuration will be referred to with the Module wording and for description reasons the board test will be analyzed formerly to the module one.

As said previously, all the tests related to the PSB will be performed with the top side of the board accessible; all the tests related to the analysis of the PSM will instead be performed accessing only from the two connectors as shown in figure 5.1.

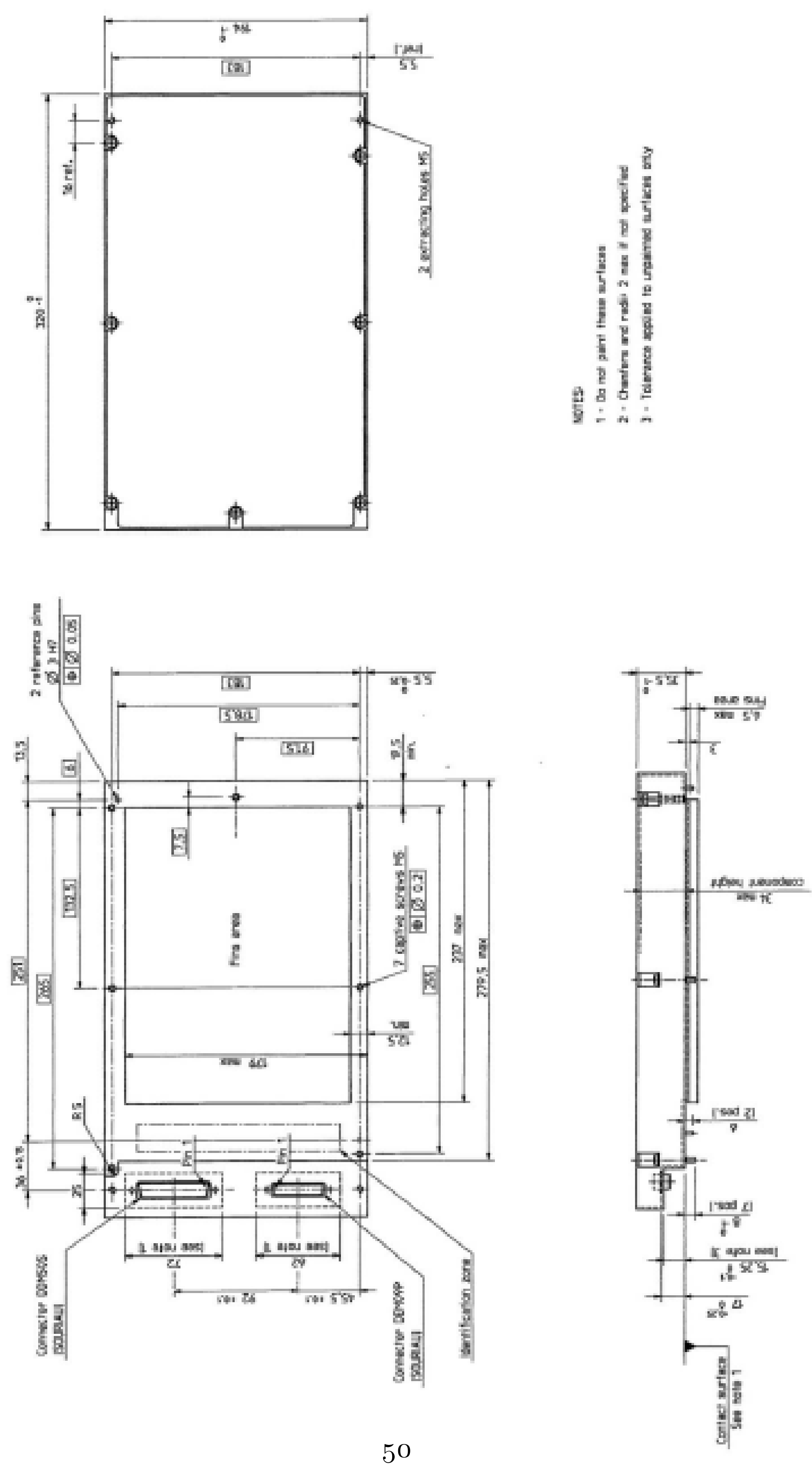


Figure 5.1: PSM Mechanical Interface

5.1.1 PSM (PSB) Architecture Requirements

The Power Supply Unit (PSU) is the major block of a larger Power Supply System (PSS) shown in figure 5.2. It includes the PSB, an Hold-Up assembly, which together form the PSU, and an EMC filter.

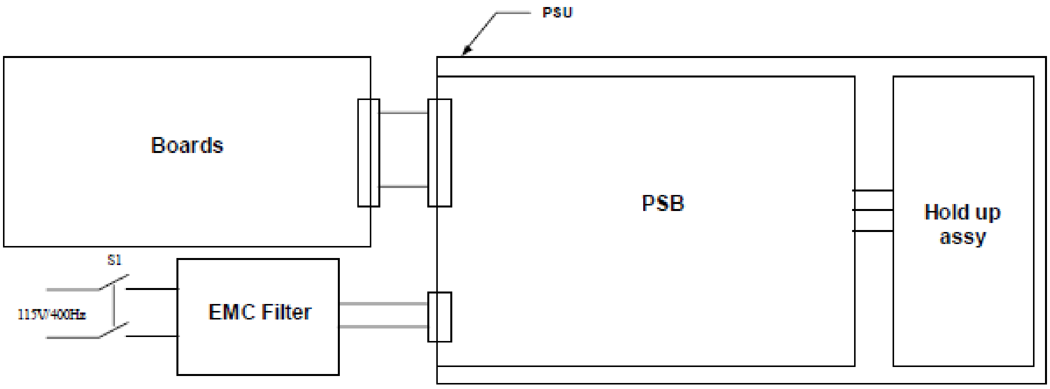


Figure 5.2: PSS schematic blocks architecture

From the above architecture arises the need to perform the PSU test activity with the item configured and interconnected according to its final destination (the PSS level). The PSU is hence provided with electrical and mechanical interfaces designed to allow the plug in installation on the final power supplied equipment. From the electrical point of view the PSU consists of a switching mode converter which provides six output rails (V1 to V6, see in detail later), drawing power, at high power factor, from the 115 V/400 Hz bus.

The main blocks of the PSM architecture are instead shown in figure 5.3 and listed below:

- AC/DC input converter;
- Power Factor Conditioner and the associated hold up capacitor;
- various DC-DC Converter;
- Auxiliary PS;
- Control and Interface Module (CIM);

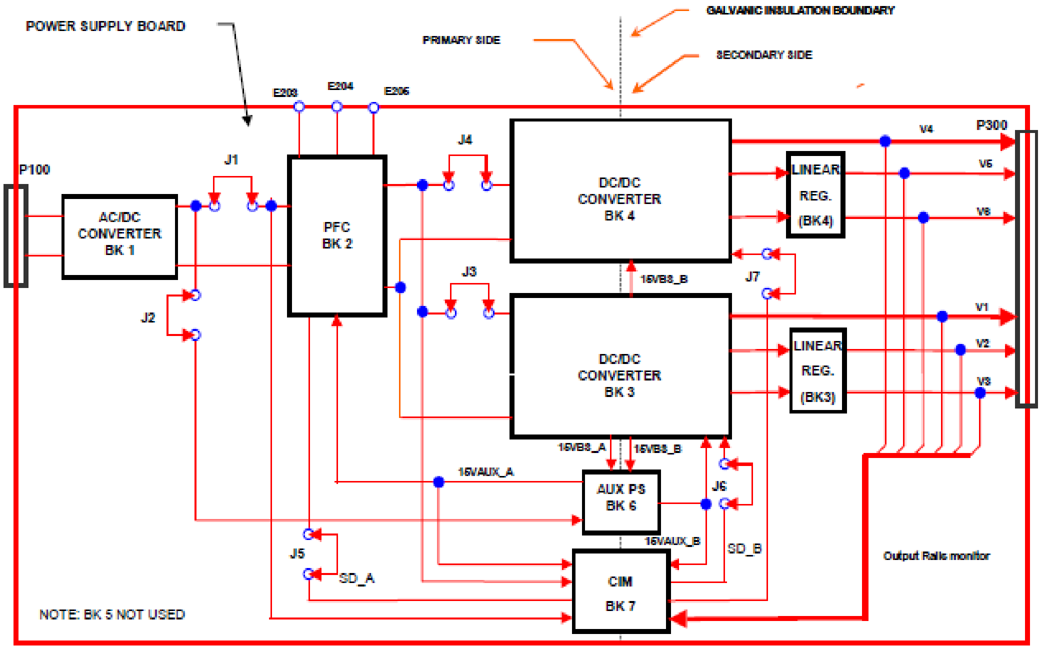


Figure 5.3: PSM schematic blocks architecture

- Block interconnection jumpers (J1 to J7).

For the general description of the filters, DC-DC conversion and PFC refer to chapter 2. Few words must be spent on the Control and Interface Module and the auxiliary PS which have not been mentioned in the dissertation yet.

The CIM block performs the monitoring of all PSB major function, providing both the control signal for the PFC and DC/DC converters enable and the external interface signal (TTL).

The auxiliary Power Supply mainly consists of a low power DC/DC converter arranged in step up configuration, which, drawing power from the rectified AC bus, generates primary side and secondary side start up voltages for the PFC, DC/DC converters and CIM.

A last mention must be made for the block interconnection jumpers (J1 to J7). The PSB is provided with seven jumpers used to interconnect the relevant functional blocks. This allows the functional separation between the blocks, which can be tested separately before being all-together connected. The jumper connection map is shown in figure 5.4.

Jumper	Effect
J1	Connects AC/DC CONVERTER to PFC
J2	Connects AC/DC CONVERTER to AUX PS
J3	Connects PFC to DC/DC CONVERTER #1
J4	Connects PFC to DC/DC CONVERTER #2
J5	Connects PFC to CIM
J6	Connects DC/DC CONVERTER #1 to CIM
J7	Connects DC/DC CONVERTER #2 to CIM

Figure 5.4: Jumpers connection map

Discussing in more detail the design specifications of the individual blocks, from the operating principle of each blocks, their targeted requirements can be logically obtained: The AC/DC converter, as expected, operates the conversion AC-DC of the 115 Vac/400 Hz input bus by means of a bridge rectifier and a Low frequency filter. The Power Factor Conditioner improves the power factor figure by means of a suitable conditioning of the rectified input voltage coming from the AC/AC converter. The aforementioned performance is based on the use of an advanced converter topology which draws a sinusoidal current from the input AC bus and returns a DC stable output voltage supplying the DC/DC converter and storing energy in the external Hold-Up assembly (figure 5.2). The various DC-DC converters present in the PSB are all based on a half bridge, fixed frequency switching mode converter topology, which performs the conversion of a pre-regulated input voltage, coming from the PFC, into a different output voltage. The first DC/DC converter (block three in figure 5.2) converts a 5.18 V input voltage into a ±16.5 V output. This output line supplies two linear regulators allowing the generation of the ±15 V rails. The second DC/DC converter (block four in figure 5.2) converts a 3.37 V input voltage into a ±6.75 V output. This auxiliary line supplies two linear regulators allowing the generation of the ±5 V rails.

Table 5.1: DC-DC converter output voltage rails

Voltage rail	Nominal value [V]	Destination
V1	+5.18	External Load (Digital Rail)
V2	+15	External Load
V3	−15	External Load
V4	+3.3	External Load
V5	+5	External Load(Analog Rail)
V6	−5	External Load

Table 5.1 reports the output voltage rails provided by the DC/DC converters and linear regulators and their corresponding final destination.

Following the above description of the output voltage rail of the PSB, table 5.2 shows the control signals and external intarface signals (TTL) generated by the board.

Table 5.2: PSM Interface/control signals

Signal name	Description
AC_FAIL	Loss of input power
PS_RESET	Power supply reset signal
PWR_FAIL	Power failure alert signal
S/D_A	PFC control enable
S/D_B	DC7DC converter enable

The architecture requirements section of the TRS provides also significant information regarding the input and output voltage and power characteristics before which the UUT shall always submit. For the input, the active power, the apparent power, the power factor and the efficiency are the parameters to be controlled. It can be of interest to analyze the requirement set upon the efficiency: the PSM shall present a minimum efficiency value of 73%. In chapter 2, the value of the efficiency given for the different switching power supplies analyzed was of 80, 90 even 95%. It appears clear now how those values were merely theoretical and, although reachable in some applications, when facing real market devices they are inevitably lower. For completeness, the active power shall never exceed $264,66$ W; the apparent power shall never exceed $294,06$ VA and the power factor shall be better than $0,9$.

The output characteristics section of the Test Requirement Specification provides as many boundaries as they are for the input section. Of all, the most significant one, especially in the perspective of the test to be performed, is the output voltage disturbance which aims to defined the total peak to peak disturbance on each output rail voltage (see table 5.1). By specification, it shall not exceed the values reported in table 5.3 where the listed values refer to figure 5.5 which of course is just an exemplification of the disturbance behaviour. It has also to be pointed out that the values of disturbance reported in table 5.3 refers exclusively to the *Minimum Load Test* which will be analyzed later on.

Table 5.3: Output voltage disturbance

Voltage rail	Nominal value [V]	Max ripple, Vpp [mV]
V1	+5.18	80
V2	+15	150
V3	−15	150
V4	+3.3	80
V5	+5	80
V6	−5	80

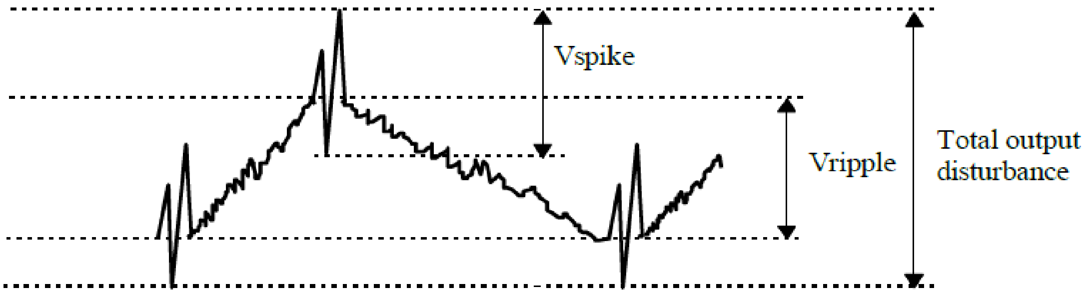


Figure 5.5: Output voltage disturbance components

5.1.2 PSM (PSB) Test requirements

The Test requirements section of the TRS starts by describing the minimum equipment required in order to properly test the PSU. Usually, the TRS provides even an indication of which brand of instrumentation should be used. It is obviously implicit that the necessary information lies only in the specifications of the instrumentation rather than the model or brand of the used tool.

Specifically, the test work station used for the DUT consisted of:

- *Tektronix DPO 5034* Digital phosphor Oscilloscope, 350 MHz, 5 GS/s;
- *UNI-T UT61E* Multimeter AC-DC Meter;
- *AMETEK 2553iX* AC & DC Power Source;
- *Tektronix TPP0500* 500 MHz probe 300 V CAT II 3.9 pF/10 MΩ;
- *Spea YAGEN* Waveform Generator;
- *Spea ACC200* Active load.

The section carries on describing the proper environmental conditions and handling warning that are necessary to follow during the test of the PSB.

Finally, the many electrical tests to be performed on the PSB are described. For sake of accuracy and in order to avoid unnecessary repetitions, the tests reported by the TRS have been analyzed and grouped so that their description would be punctual but not overabundant. Following the aforementioned logic, the electrical tests can be categorized in four main groups: Insulation resistance test, Power setup test, Output Voltages disturbance test and Timing & Signal sequence test.

The insulation resistance test must be performed as the first test before carrying out the functional test. It consists in evaluating the resistance value present between different pins of P100 and P300. P100 is the PSU input connector whose main aim is to provide to the board the 115 Vac required by specification. P300 is the board output connector responsible of providing all the external voltage rails and control signals generated by the PSM. The value to be measured shall not be less then 300 MΩat 500 Vdc.

The Power setup test firstly requires to verify the parameters related to the input voltage rail as of nominal voltage, frequency and current limit. Then, through three different configurations of Load, input parameter such as the nominal voltage and output parameter such as the output power shall be verified. The test is performed for minimum output current (*Minimum load test*), maximum output current (*Maximum/Full load test*) and for normal condition test.

The Output Voltages Disturbance test is linked to the evaluation of ripples and spikes on the voltage rails as previously described in table 5.3. The test is performed at both maximum and minimum load and it is carried out by measuring the values of ripple and spike combined as one total voltage value of output disturbance.

Last macro group of tests is the Timing & Signal sequence test. This tests follow the same logic and aim to verify the correct behaviour of the DUT signals when certain conditions are triggered. The first one of these, the PFC power ON sequence test, must be addressed separately and with particular regard since it cannot be performed for the PSM. The difference between the PSB and PSM has been investigated previously. The reason why the PFC test cannot be performed on the power supply module is due to the test point necessary to stimulate correctly the DUT which are not accessible when the board is packaged as required by the PSM specification. The test points are then only reachable when the top side of the board is exposed and so when the PSB is being tested. Another tests that can only be performed on the PSB are the *AC/DC converter and AUX PS test*, the *PWM Clock test* and the *Power on gate test*.

It is now possible to execute a deeper analysis of each test taking as reference the ones performed on the PSB since allow to gain an higher fulfilling of the test requirements.

AC/DC converter and AUX PS test

In order to perform this test, specific jumper connection must be implemented on the PSB. Following the connection map already shown in figure 5.4, the test requires the presence of the J2 jumper only so that the AC/DC is completely isolated from the rest of the circuit.

In detail the test verifies the values of the AC line RMS current and the input active power. Furthermore, it measures the nominal DC values of the 15VBS_A and 15VBS_B voltage rails. Detailed test data are reported in figure 5.6.

Parameters	value	Accuracy
Ac line RMS current	0.30A	±15%
Ac line input active power	4.2W	±15%

Rail	Test point	Nominal DC value	Accuracy
15VBS_A	CR602 cathode vs E201	12.5V	±12%
15VBS_B	CR603 cathode vs E301	12.5V	±12%

Figure 5.6: AC/DC converter and AUX PS test parameters

PWM clock test

This test verify the PWM (Pulse Width Mode) clock frequencies of various board's components. All the tests requires a measured nominal frequency clock of 125 kHz with an accuracy of ±5%.

PFC test at no load

The execution of this test requires the presence of both jumpers J1 and J2 connecting the PFC to the AC/DC converter. The test verifies the AC line RMS current, the input active power and the Hold-Up capacitor voltage level VH when the board is correctly energised. Specific parameters characteristics are shown in figure 5.7.

Parameters	Nominal value	Accuracy
Ac line RMS current	0.31A	±15%
Input power active power	5.5W	±15%
VH DC level (E202 vs GND_A)	305V	±2%

Figure 5.7: PFC no load test parameters

PFC, DC/DC converters minimum and maximum load test

Rail	Nominal DC Voltage	Nominal load current	Accuracy
V1	5.18V	5.0A	±10%
V2	+15V	0.03A	±10%
V3	-15V	0.07A	±10%
V4	3.37V	5.0A	±10%
V5	+5V	1.0A	±10%
V6	-5V	1.0A	±10%

(a) DC/DC converters @ minimum load

Rail	Nominal DC Voltage	Nominal load current	Accuracy
V1	5.18V	15A	±10%
V2	+15V	0.1A	±10%
V3	-15V	0.2A	±10%
V4	3.37V	15A	±10%
V5	+5V	3.0A	±10%
V6	-5V	3.0A	±10%

(b) DC/DC converters @ maximum load

Figure 5.8: DC/DC converters minimum and maximum load requirement

From this test on, all the jumpers (J1 to J7) must be connected. The test analysed different characteristics of the output voltage rails of the board at both the minimum load condition and the maximum one. *Spea ACC200* Active load are used as loads configured accordingly to the nominal load current required for each line by the specification. Figure 5.8 and shows the nominal load value required for each voltage rail.

With the aforementioned load configuration, the output voltage rails parameters of nominal DC voltage and maximum peak to peak noise ripple are measured. The input parameters of AC line RMS current, input active power, power factor and Hold-Up capacitor voltage level VH are measured as well.

Furthermore, the TRS requires to verify, during the power on, that V1 (5.18 V) and V4 (3.37 V) voltage signals are in accordance with the signal tracks shown in figure 5.9.

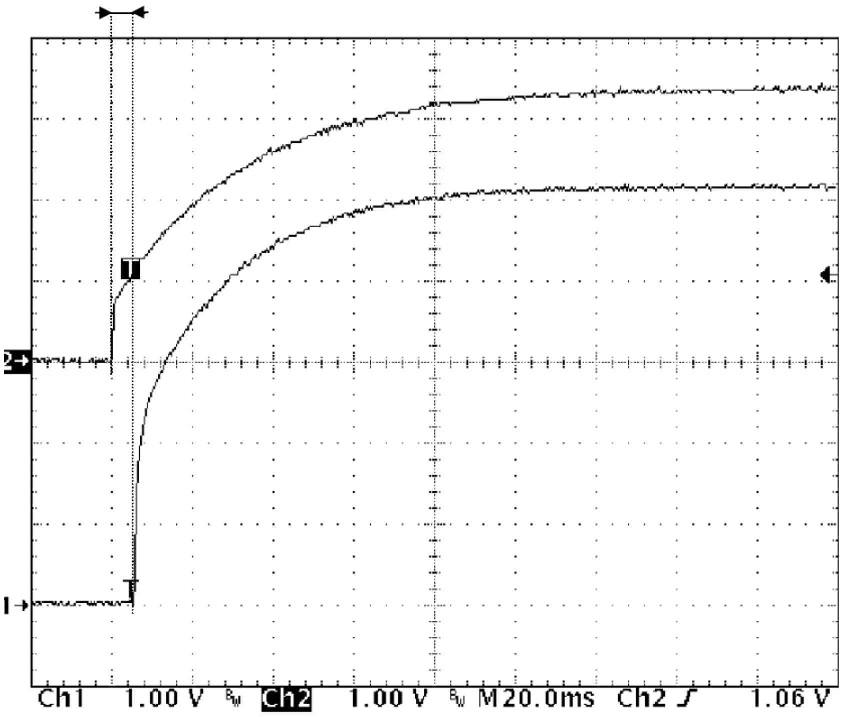


Figure 5.9: V1 and V4 ramps @ power on

The rising time to be measured for V1 is equal to 45 ms with a ±25% accuracy. The rising time to be measured for V4 is equal to 65 ms with a ±25% accuracy. The ΔT between the two rails must also be measure with a requested value of 6 ms and ±100% accuracy.

Timing tests

PFC power on sequence, *Power on gate test*, *Short circuit test* and *Power on sequence* are all tests whose main aim is to verified the timing between different signals at different PSB working condition.

The *PFC power on sequence* aims to verify the sequence shown in figure 5.10 (The S1 signal refers to the actual power on of the PSB; VH is the Hold-up voltage and VH_TH the according logic signal).

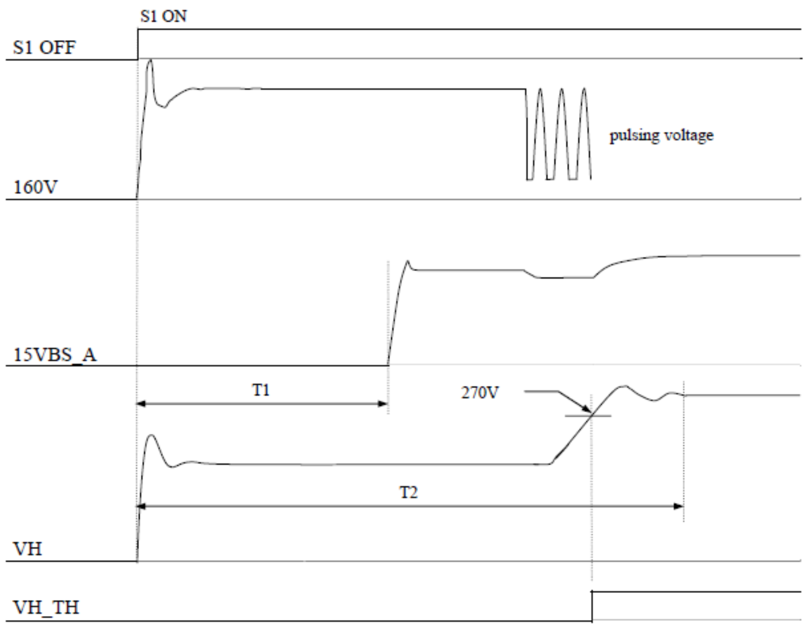


Figure 5.10: PFC power on sequence

The parameters to be measured regarding the above shown figure are reported, with the related test description, in figure 5.11.

Parameter	Nominal value	Accuracy	Remarks
T1	180mS	±20%	time delay between the rising edge of the voltage on E200 vs GND_A and the rising edge of the voltage on CR602 cathode vs GND_A
T2	300mS	±20%	measured at settling time 1% of the voltage on test point E202 vs GND_A
VH_TH rising edge at VH	270V	±12%	voltage on CR703 cathode or on pin 2 of U703 vs GND_A

Figure 5.11: PFC power on sequence test parameters

The *Power on gate test* aims to verify the sequence shown in figure 5.12 where

T1 must be equal to 850 ms with an accuracy of ±15% and T2 must be equal to 1000 ms with an accuracy of ±15%.

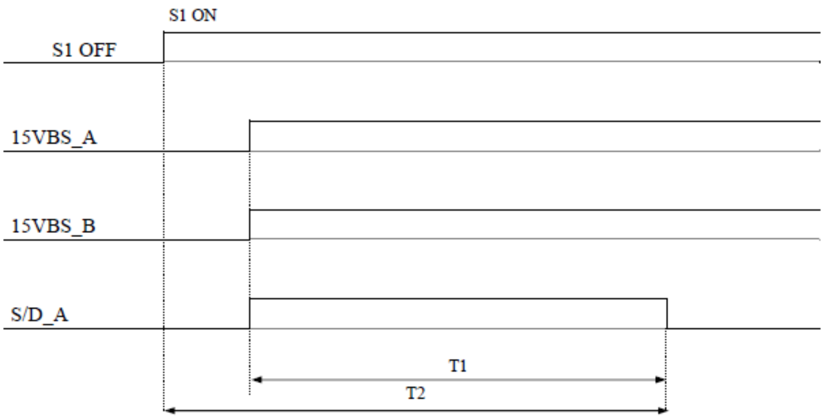


Figure 5.12: Power on gate sequence

The *Short circuit test* verifies that each output voltage rails, when the board is shutdown, short themselves after a proper time interval as shown in figure 5.13.

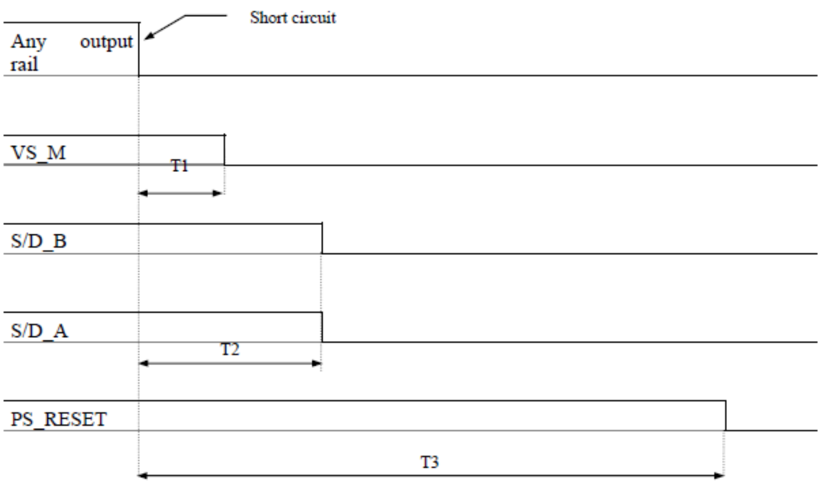


Figure 5.13: Short circuit test sequence

The parameters to be measured, with the related test description, are shown in figure 5.14.

Parameter	Nominal value	Accuracy	Remarks
T1 for rail V1	100µS	±50%	time delay between the falling edge of the short circuited rail and the falling edge of the signal detected on E702 vs GND_B
T1 for rail V4	350µS		
T1 for rail V2, V3, V5, V6	30mS	±20%	
T2 for rail V1	150µS	±20%	time delay between the falling edge of the short circuited rail and the falling edge of the signal detected on E701 or E703 vs GND_B
T2 for rail V4	400µS		
T2 for rail V2, V3, V5, V6	30mS		
T3 for rail V1	800µS	±20%	time delay between the falling edge of the short circuited rail and the falling edge of the signal detected on P300, pin 33 vs GND_B
T3 for rail V4	1000µS		
T3 for rail V2, V3, V5, V6	30mS		

Figure 5.14: Short circuit test parameter

Last timing test is the *Power on sequence* test. As shown in figure 5.15, the test verifies the proper power on of the AC_FAIL, PWR_FAIL and PS_RESET control signals when the PSB is energised.

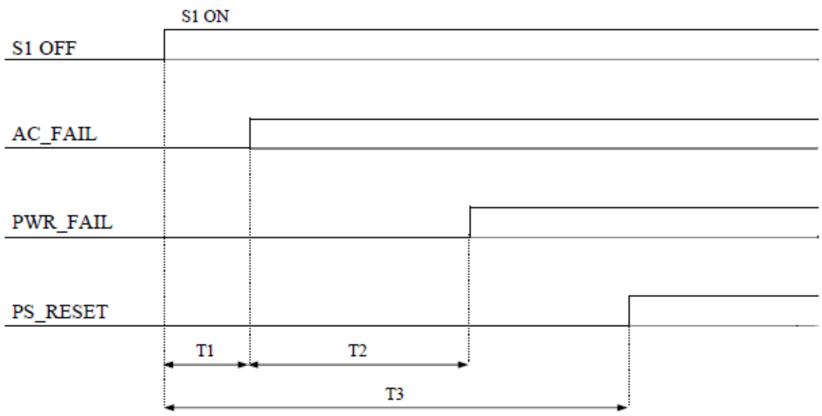
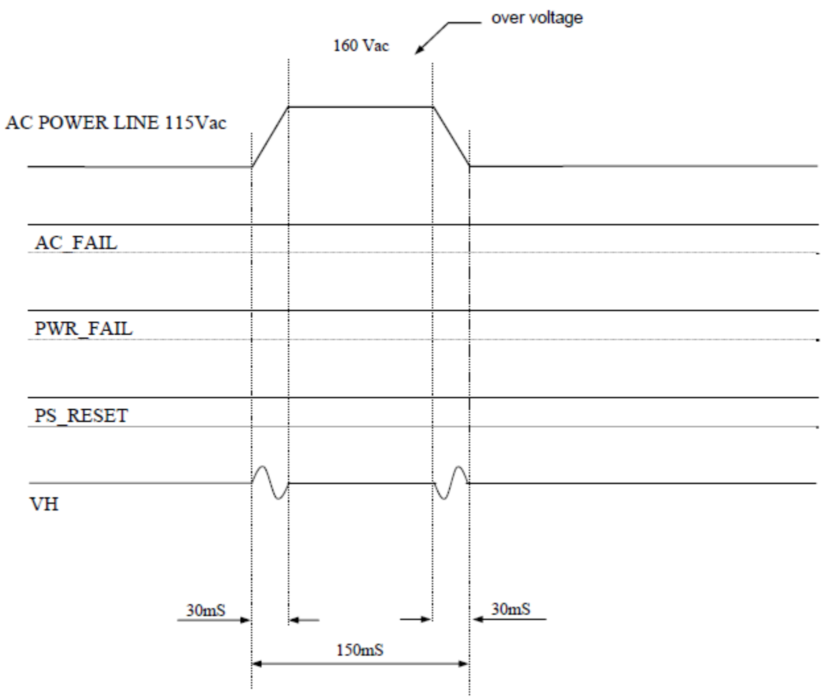


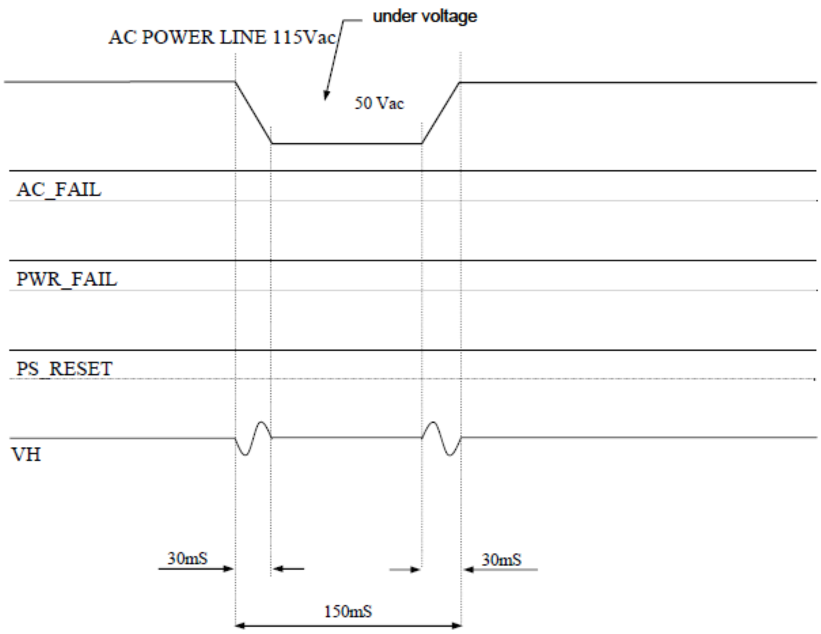
Figure 5.15: Power on sequence

Overvoltage, undervoltage and power interruption tests

The overvoltage, undervoltage and power interruption tests are very similar to the timing test but they still earn a different classification since they required particular action on the power supply of the PSB more complex than a simple on/off.



(a) Input overvoltage surge



(b) Input undervoltage surge

Figure 5.16: Input overvoltage/undervoltage surge

The first two are respectively performed through power line transients: For the overvoltage the AC power line is increased from 115 Vac to 160 Vac for a time interval of 150 ms; for the undervoltage the AC power line is decreased from 115 Vac to 50 Vac for the same time interval. As shown in figure 5.16, during the power line transient, the PSB control signals must always remain at high logic value (5 V). The power interruption sequence on the other hand is realised through an input power interruption of 30 ms as shown in figure 5.17.

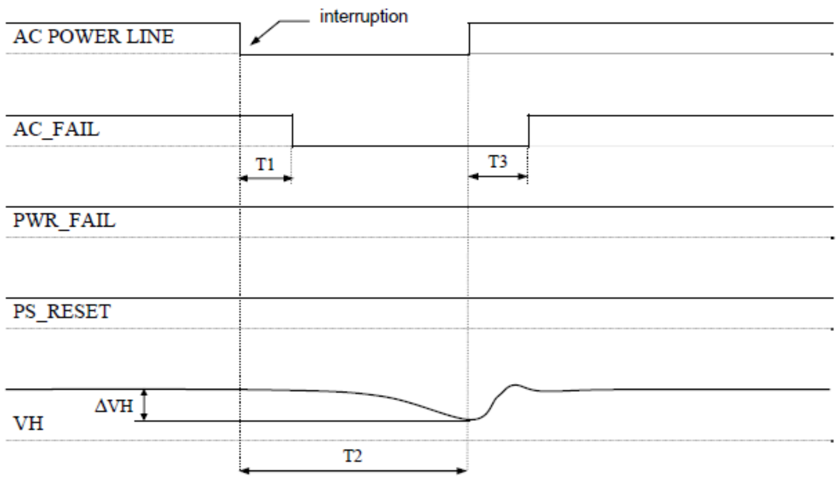


Figure 5.17: Power interruption sequence

As shown in the figure, the test also requires to verify that the VH (Hold-up capacitor voltage) does not fall below a ΔVH of 80 V from its regime value.

Parameter	Nominal value	Accuracy	Remarks
T1	1.5mS	±50%	time delay between the falling edge of the input power line and the falling edge of the signal detected on P300-17 vs GND_B
ΔVH	80V	±20%	hold up capacitors (Hold Up Assy) discharge amount measured on E204 vs E205 (GND_A)
T2	30mS	-0%, to 5%	duration of power interruption
T3	1.5mS	±50%	time delay between the raising edge of the input power line and the rising edge of the signal detected on P300-17 vs GND_B

Figure 5.18: Power interruption test parameter

5.1.3 PSM (PSB) Test Data Record

Last section of the Test Requirement Specification is the Test Data Record. It is simply a Table which acts as a summary contained the results of all the executed measurements and other introductory information related to the item name, its program, the start and end date, the item serial number and similar.

Chapter 6

Implementation of the test adapter

The study of the TRS sets the kind of requirements needed to fulfill the test of the DUT. For the Power Supply Board object of this dissertation, the 3030 Bed of Nails Spea system has been used. As explained in chapter 4, the BoN tester requires the design of the so called fixture which acts as interface between the machine and the board under test. Usually fixtures present a considerable number of test points (probes) in order to contact the board and been able to excite it properly as the tests required. This kind of specific application, however, since the board communicates almost only through the input and output connectors, does not need the presence of test points but only of the respective connector counterparts. These are part of various and different hardware boards that are place inside the fixture whose functions help to carry out the tests. There is for some of the required tests the necessity to directly touch some points on the circuit board which are not reachable trough the DUT's connectors (when testing the PSB rather than the PSM). For this few cases, the contact is made manually by the test operator (using probes and clips).

In this chapter the external and internal composition of the fixture is analyzed and the workflow necessary for its manufacture is described.

6.1 Fixture: external composition

The implemented fixture for the test of the DUT is shown in figure 6.1.

Figure 6.1: Fixture, external

The black upper part of the fixture (visible in figure 6.1) contains the strips that allow the interconnection between the fixture itself and the 3030 system. Each strip is dedicated to the communication with a specific board of the system: drivers, boosters, DVM (Digital Volt Meter), relay boards (for the channels measure) and even external devices such as AC generator and active loads. The interface strips also act as bridge between the internal hardware placed inside the fixture and the system hardware. To exemplify: an output board signal, for instance a voltage rails, runs to the internal fixture hardware through the DUT output connector, then it goes from the aforementioned internal board to its dedicated interface strip (analog channel) which is cabled to the DVM inside the 3030 system which will perform the required voltage measure.

The lower part of the fixture shows the support area of the DUT. The connector P300 and P100 are directly connected to the PSM which is then support by the contrast fingers (black small plastic tube visible in the fixture photo).

Figure 6.2 shows how the PSM is connected to the fixture.

Figure 6.2: Board placement on the fixture

Figure 6.3: Noise Board, Led Board and jumpers close-up

To the left of the board support area (figure 6.1), there are the *SMB NOISE* connector (used for the ripple measures) and the voltage output rails jumpers. The *SMB NOISE* connector is part of a noise board (made by Spea) which is placed vertically inside the fixture connected to another Spea made hardware board called *Specialization*, see figure 6.3.

The last external area of the fixture to be analysed are the P600 and P400 connectors and the PWATT connector (see figure 6.1). The first to connectors are used for the clip cables. These cables are formed at one end by a tray connector, and at the other end by a series of individual clip cables which are used, during the PSB test, to directly contact specif test points on the board surface. The PWATT connector links the external AC generator to the interface of the system and it allows to perform the input in-rush current measure through the used of a current probe.

6.2 Fixture: internal composition

The internal part of the fixture is shown in figure 6.4.

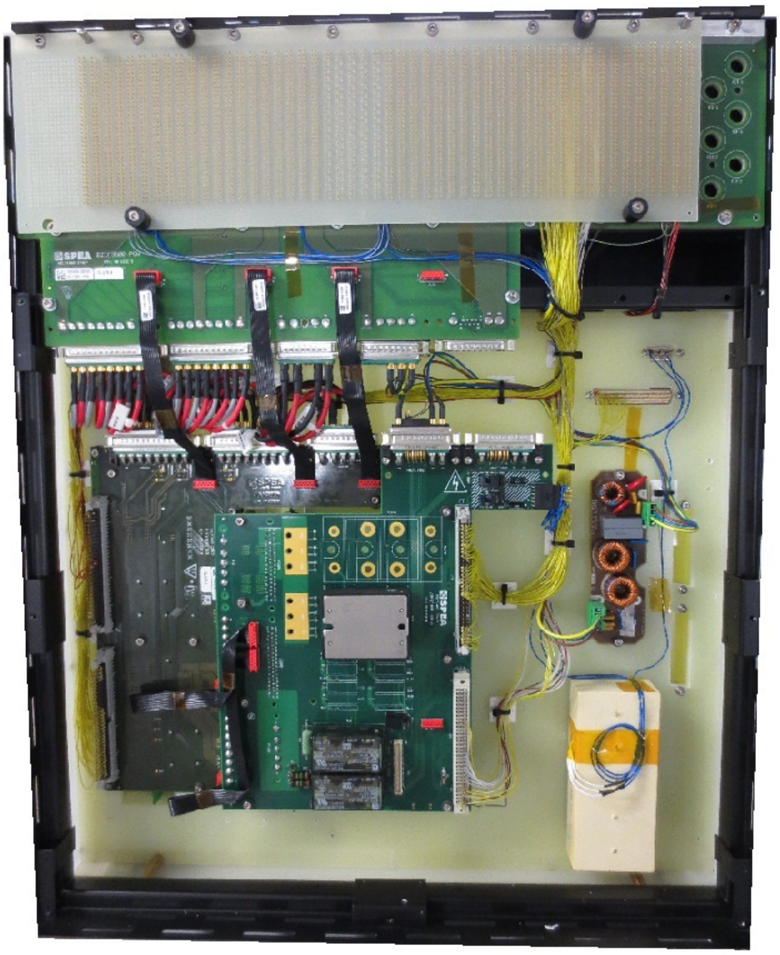


Figure 6.4: Fixture, internal

From figure 6.4, it is possible to have a better appreciation of the fixture interface strips and so how the communication and interconnection with the 3030 system is achieved.

To match the test requirements and be able to perform all the needed measures, the following hardware has been integrated in the fixture composition:

- Board Z0***400 - Interface adapter board;
- Board FIXMB100 - Fixture Mother Board module;
- Board FXFS10A - Specialization board;
- Noise Board;
- Led Board;
- EMC Filter.

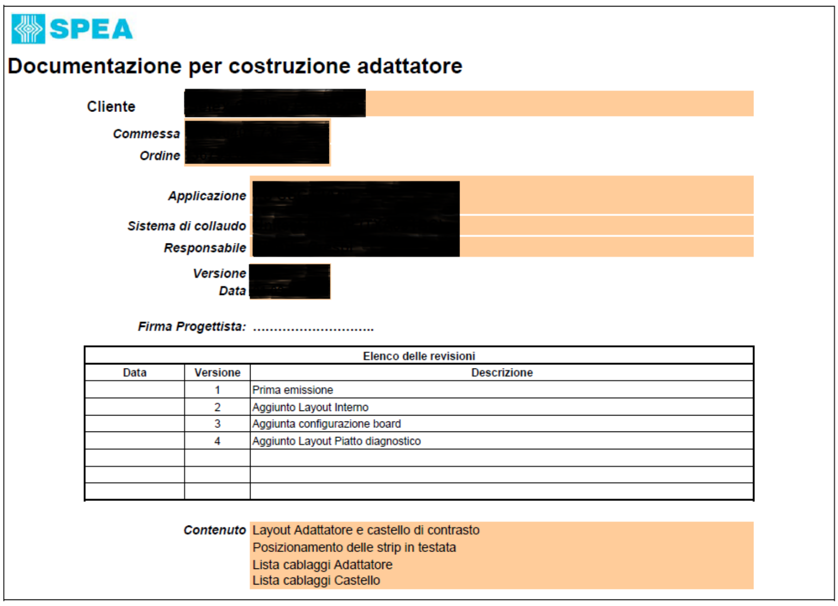
The first three boards are mainly used to create a connection flow between the DUT, the system interface and the other internal fixture hardware. The noise board and led board, as their names suggested, are required in order to perform the noise measure and through the led it is possible to control the state of the board under test at all time.

6.3 Fixture implementation workflow

The fixture, described in detail in the previous sections, is a complex object which manufacture needs to be carefully organized and repeatedly controlled in order to avoid the possibility of having a final product which is defective or not completely conformed to the TRS specification.

First step is the design of the fixture. The designer must take into account the TRS provided by the client in order to have an idea of what kind of components must be integrated in the fixture. The fixture design, however, is not limited to the choice of its internal hardware: the design is also a mechanical design which has to take into account the mechanical properties of the DUT in order to understand how to correctly place it in the fixture. Of course, the actual physical and mechanical project is realized by a mechanical designer which starts, nonetheless, by the "advise" layout made by the test engineer.

The document which is produced in this phase of the fixture project is called *Adapter construction document*. In figure 6.5, the info cover of the document is shown.



SPEA

Documentazione per costruzione adattatore

Cliente [redacted]

Commessa [redacted]
Ordine

Applicazione [redacted]
Sistema di collaudo
Responsabile

Versione [redacted]
Data

Firma Progettista: ....

Elenco delle revisioni		
Data	Versione	Descrizione
	1	Prima emissione
	2	Aggiunto Layout Interno
	3	Aggiunta configurazione board
	4	Aggiunto Layout Piatto diagnostico

Contenuto Layout Adattatore e castello di contrasto
Posizionamento delle strip in testata
Lista cablaggi Adattatore
Lista cablaggi Castello

Figure 6.5: Adapter construction document cover

It can be seen how only fundamental information are reported in the document cover: client, internal Spea project number, name of the application (DUT name), engineer responsible for the test application and the topics included in the document. The external and internal layout is provided in the document as well.

Once the *Adapter construction document* is finished, before sending the data to the mechanical department, the Fixture electrical diagram must be drawn. It is not possible, for proprietary reason, to report in depth the fixture electrical diagram. However, in order to give a general idea of what the diagram contains, it can be said that it reports all the wired, and not wired connection, between the interface strips, the fixture internal hardware, the internal system hardware and the external instruments (AC Generator, active loads).

When all this documents are finished, the mechanical fabrication of the fixture can began. After that, before starting the test of the DUT, the fixture itself is tested in order to verify that all the internal hardware is properly functioning and that all the connections have been realized correctly.

Chapter 7

Implementation of the test program

Once the fixture is ready to be used, the only thing indispensable for the test of the DUT is the test program. The test program is actually divided into two main parts: the first one which is called *Fixture check* and the second one which is the actual program for the debug of the DUT which is called TPGM (Test Program Module). In chapter 3, the different kinds of tests had been explained; for this specific application only the functional test is required and so implemented. This chapter will focus on the implementation of the TPGM and its actual use for the DUT test: snippets of the implemented code will be reported and their function explained (to view the code in its entirety refer to Appendix A).

7.1 Test Program writing

7.1.1 VRAD

The TPGM is written through the auxiliary of the Integrated Development Environment (IDE) *Visual Studio* in *Visual Basic* programming language. Spea has furthermore created a software tool called *VRAD* (Very Rapid Application Development) which helps in the initial setting of the Visual Basic program. VRAD, as shown in figure 7.1, is an excel macro linked to visual studio: it is possible to write the test program as a sequence of tasks and tests where to each test there is the association to a visual basic function and each function can be defined through its internal parameters. In this way, during the test debug on the DUT, it is possible to modify those function's parameters without entering directly inside the visual basic project. It is a powerful tool designed to help its use by operators (client) which are not able to use a programming language. In this way, after the setting up of the program which requires an expert hand, the test of the DUT can be carried out even by less competent operators.

SPEA

Test Plan

Task			Enabled	Fail bin	Pass bin	Datalog	Type	Refresh Site	Location	Description						
System Set Up			Yes	1 - Fail	0 - Pass	Yes	Analog	Yes	Testplan							
Test id	*Test*	*Enabled*	*Functions*	*Fail bin*	*Pass bin*	*Remark*	*Nominal*	*Th low*	*Th high*	*Factor*	*Unit*	*Compare*	*Testpattern*	*Format*	*Datalog*	*Refresh Site*
1	Test 1	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Predisposizione UUT - Modulo	CPASS					In Limits			No	Yes
2	Test 2	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Predisposizione UUT - Scheda	CPASS					In Limits			No	Yes
3	Test 3	Yes	fVRADSystemSetUp	1 - Fail	0 - Pass	_SystemSet_UP	CPASS					In Limits			No	Yes
AC DC Converter and AUX PS TEST			Yes	1 - Fail	0 - Pass	Yes	Analog	No	Testplan	3.8 - AC DC Converter and AUX PS TEST						
Test id	*Test*	*Enabled*	*Functions*	*Fail bin*	*Pass bin*	*Remark*	*Nominal*	*Th low*	*Th high*	*Factor*	*Unit*	*Compare*	*Testpattern*	*Format*	*Datalog*	*Refresh Site*
4	Test 1	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Verifica presenza Jumper	CPASS					In Limits			No	No
5	Test 2	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Jumper	CPASS					In Limits			No	No
6	Test 3	Yes	fVRADSetUpAccensione	1 - Fail	0 - Pass	_Alimentazione UUT su GND_A	CPASS					In Limits			No	Yes
7	Test 4	Yes	fVRADAcGenAmetekMeasureCurrent	1 - Fail	0 - Pass	_AC Line RMS Current @ no Load	0,3	0,255	0,345		A	In Limits			Yes	No
8	Test 5	Yes	fVRADAcGenAmetekMeasurePower	1 - Fail	0 - Pass	_AC Line Input Active Power @ no Load	4,2	3,57	5,13		W	In Limits			Yes	No
9	Test 6	No		1 - Fail	0 - Pass							In Limits			No	No
10	Test 7	Yes	fVRAD_UPto100V_MeasureVoltageABUS	1 - Fail	0 - Pass	_15VBS_A @ no Load	12,5	11	14		V	In Limits			Yes	No
11	Test 8	Yes	fVRAD_PowerOFF	1 - Fail	0 - Pass	_Spegnimento	CPASS					In Limits			No	Yes
12	Test 9	Yes	fVRAD_Discharge	1 - Fail	0 - Pass	_UUT Discharge	CPASS					In Limits			No	Yes
13	Test 10	Yes	fVRADSetUpAccensione	1 - Fail	0 - Pass	_Alimentazione UUT su GND_B	CPASS					In Limits			No	Yes
14	Test 11	No		1 - Fail	0 - Pass							In Limits			No	No
15	Test 12	Yes	fVRAD_UPto100V_MeasureVoltageABUS	1 - Fail	0 - Pass	_15VBS_B @ no Load	12,5	11	14		V	In Limits			Yes	No
16	Test 13	Yes	fVRAD_PowerOFF	1 - Fail	0 - Pass	_Spegnimento	CPASS					In Limits			No	Ignore Yes
17	Test 14	Yes	fVRAD_Discharge	1 - Fail	0 - Pass	_UUT Discharge	CPASS					In Limits			No	Ignore Yes
PWM Clock Test			Yes	1 - Fail	0 - Pass	Yes	Analog	No	Testplan	3.9 - PWM Clok Test						
Test id	*Test*	*Enabled*	*Functions*	*Fail bin*	*Pass bin*	*Remark*	*Nominal*	*Th low*	*Th high*	*Factor*	*Unit*	*Compare*	*Testpattern*	*Format*	*Datalog*	*Refresh Site*
18	Test 1	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Verifica presenza Jumper	CPASS					In Limits			No	No
19	Test 2	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Jumper	CPASS					In Limits			No	No
20	Test 3	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Cavo Hold UP	CPASS					In Limits			No	No
21	Test 4	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Sonde Oscilloscopio	CPASS					In Limits			No	No
22	Test 5	Yes	fVRADOscilloSet	1 - Fail	0 - Pass	_Set Oscilloscopio CH1	CPASS					In Limits			No	No
23	Test 6	Yes	fVRADSetUpAccensione	1 - Fail	0 - Pass	_Alimentazione UUT su GND_A	CPASS					In Limits			No	Yes
24	Test 7	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_U600 pin8 - Clock Frequency	125	118,75	131,25	K	Hz	In Limits			Yes	No
25	Test 8	Yes	fVRADOscilloSet	1 - Fail	0 - Pass	_Set Oscilloscopio CH2	CPASS					In Limits			No	No
26	Test 9	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_U200 pin14 - Clock Frequency	125	118,75	131,25	K	Hz	In Limits			Yes	No
27	Test 10	Yes	fVRAD_PowerOFF	1 - Fail	0 - Pass	_Spegnimento	CPASS					In Limits			No	Yes
28	Test 11	Yes	fVRAD_Discharge	1 - Fail	0 - Pass	_UUT Discharge	CPASS					In Limits			No	Yes
29	Test 12	No		1 - Fail	0 - Pass							In Limits			No	No
30	Test 13	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Sonde Oscilloscopio	CPASS					In Limits			No	No
31	Test 14	Yes	fVRADSetUpAccensione	1 - Fail	0 - Pass	_Alimentazione UUT su GND_A	CPASS					In Limits			No	Yes
32	Test 15	Yes	fVRADOscilloSet	1 - Fail	0 - Pass	_Set Oscilloscopio CH1	CPASS					In Limits			No	No
33	Test 16	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_Q301 pin1 (GATE) - Clock Frequency	125	118,75	131,25	K	Hz	In Limits			Yes	No
34	Test 17	Yes	fVRADOscilloSet	1 - Fail	0 - Pass	_Set Oscilloscopio CH2	CPASS					In Limits			No	No
35	Test 18	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_Q303 pin1 (GATE) - Clock Frequency	125	118,75	131,25	K	Hz	In Limits			Yes	No
36	Test 19	Yes	fVRADOscilloSet	1 - Fail	0 - Pass	_Set Oscilloscopio CH3	CPASS					In Limits			No	No
37	Test 20	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_Q401 pin1 (GATE) - Clock Frequency	125	112,5	137,5	K	Hz	In Limits			Yes	No
38	Test 21	Yes	fVRADOscilloSet	1 - Fail	0 - Pass	_Set Oscilloscopio CH4	CPASS					In Limits			No	No
39	Test 22	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_Q403 pin1 (GATE) - Clock Frequency	125	112,5	137,5	K	Hz	In Limits			Yes	No
40	Test 23	Yes	fVRAD_PowerOFF	1 - Fail	0 - Pass	_Spegnimento	CPASS					In Limits			No	Ignore Yes
41	Test 24	Yes	fVRAD_Discharge	1 - Fail	0 - Pass	_UUT Discharge	CPASS					In Limits			No	Ignore Yes
PFC Test at No Load			Yes	1 - Fail	0 - Pass	Yes	Analog	No	Testplan	3.10 - PFC Test AT NO Load						
Test id	*Test*	*Enabled*	*Functions*	*Fail bin*	*Pass bin*	*Remark*	*Nominal*	*Th low*	*Th high*	*Factor*	*Unit*	*Compare*	*Testpattern*	*Format*	*Datalog*	*Refresh Site*
42	Test 1	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Verifica presenza Jumper	CPASS					In Limits			No	No
43	Test 2	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Jumper	CPASS					In Limits			No	No
44	Test 3	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Cavo Hold UP	CPASS					In Limits			No	No
45	Test 4	Yes	fVRADSetUpAccensione	1 - Fail	0 - Pass	_Alimentazione UUT su GND_A	CPASS					In Limits			No	Yes
46	Test 5	Yes	fVRADAcGenAmetekMeasureCurrent	1 - Fail	0 - Pass	_AC Line RMS Current @ no Load	0,31	0,2635	0,3565		A	In Limits			Yes	No
47	Test 6	Yes	fVRADAcGenAmetekMeasurePower	1 - Fail	0 - Pass	_AC Line Input Active Power @ no Load	5,5	4,675	6,325		W	In Limits			Yes	No
48	Test 7	Yes	fVRADMeasureVoltageABUS	1 - Fail	0 - Pass	_VH DC Level (E202 vs GND_A)	305	298,9	311,5		V	In Limits			Yes	No
49	Test 8	Yes	fVRAD_PowerOFF	1 - Fail	0 - Pass	_Spegnimento	CPASS					In Limits			No	Ignore Yes
50	Test 9	Yes	fVRAD_Discharge	1 - Fail	0 - Pass	_UUT Discharge	CPASS					In Limits			No	Ignore Yes
Minimum Load Test			Yes	1 - Fail	0 - Pass	Yes	Analog	No	Testplan	3.11 / 4.8 - Minimum Load Test						
Test id	*Test*	*Enabled*	*Functions*	*Fail bin*	*Pass bin*	*Remark*	*Nominal*	*Th low*	*Th high*	*Factor*	*Unit*	*Compare*	*Testpattern*	*Format*	*Datalog*	*Refresh Site*
51	Test 1	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Verifica presenza Jumper	CPASS					In Limits			No	No
52	Test 2	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Jumper	CPASS					In Limits			No	No
53	Test 3	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Cavo Hold UP	CPASS					In Limits			No	No
54	Test 4	Yes	fVRADSetUpAccensione	1 - Fail	0 - Pass	_Alimentazione UUT - 115V	CPASS					In Limits			No	Yes
55	Test 5	Yes	fVRADTimeWait	1 - Fail	0 - Pass	_Attesa stabilizzazione corrente	CPASS					In Limits			No	No
56	Test 6	Yes	fVRADAcGenAmetekMeasureCurrent	1 - Fail	0 - Pass	_AC Line RMS Current @ LIGHT Load	0,92	0,828	1,012		A	In Limits			Yes	No
57	Test 7	Yes	fVRADAcGenAmetekMeasurePower	1 - Fail	0 - Pass	_AC Line Input Active Power @ LIGHT Load	90	81	99		W	In Limits			Yes	No

Figure 7.1: VRAD, Test plan section

In order to better understand how VRAD is organized, a detail of the test plan in figure 7.1 is reported in figure 7.2.

17	Test 14	Yes	fVRAD_Discharge	1 - Fail	0 - Pass	_UUT Discharge	CPASS				
PWM_Clock_Test			Yes	1 - Fail	0 - Pass	Yes	Analog	No	Testplan	3.9 - PWM Clok Test	
Test id	*Test*	*Enabled*	*Functions*	*Fail bin*	*Pass bin*	*Remark*	*Nominal*	*Th low*	*Th high*	*Factor*	*Unit*
18	Test 1	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Verifica presenza Jumper	CPASS				
19	Test 2	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Jumper	CPASS				
20	Test 3	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Cavo Hold UP	CPASS				
21	Test 4	Yes	fVRADForm_Message	1 - Fail	0 - Pass	_Connessione Sonde Oscilloscopio	CPASS				
23	Test 6	Yes	fVRADSetUpAccensione	1 - Fail	0 - Pass	_Alimentazione UUT su GND_A	CPASS				
24	Test 7	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_U600 pin8 - Clock Frequency	125	118,75	131,25	K	Hz
26	Test 9	Yes	fVRADOscilloTektronixMeasure	1 - Fail	0 - Pass	_U200 pin14 - Clock Frequency	125	118,75	131,25	K	Hz
27	Test 10	Yes	fVRAD_PowerOFF	1 - Fail	0 - Pass	_Spegnimento	CPASS				
28	Test 11	Yes	fVRAD_Discharge	1 - Fail	0 - Pass	_UUT Discharge	CPASS				

Figure 7.2: VRAD, Test plan section - detail

In the red highlighted area, it is possible to see Test number 6 and 7 of the *PWM_Clock_Test* task. Test number six will run the visual basic function called *fVRADSetUpAccensione* and test number seven the function *fVRADOscilloTektronixMeasure*. Each row also contains a remark field and the data concerning the measure threshold limits and nominal value. The function parameter are accesible through another VRAD excel sheet called *Local Parameter Map* shown in figure .

7.1.2 Visual Basic - TPGM

The help provided by the use of VRAD in the writing of the test program is merely organizational: when VRAD is launched the *Testplan.vb* module is automatically created in Visual Studio. Although of great help, each function recalled by VRAD must still be individually implemented in Visual Studio by the test engineer.

The *Testplan.vb* module is reported below:

```
Option Strict Off
Option Explicit On

Module modTestplan

    Public Function Testplan() As Integer
        Dim FailFlag As Integer

        FailFlag = AtosF.PASS


        ' ---- VRAD TASK EXECUTION BEGIN ---- (do not remove this line)

        ' Read initial site list
        Call GetSiteList(gInitialSiteList(0))

        Call vbRunTask(SYSTEM_SET_UP_ID, AddressOf System_Set_Up)
        Call vbRunTask(AC_DC_CONVERTER_AND_AUX_PS_TEST_ID, AddressOf AC_DC_Converter_and_AUX_PS_TEST)
        Call vbRunTask(PWM_CLOCK_TEST_ID, AddressOf PWM_Clock_Test)
        Call vbRunTask(PFC_TEST_AT_NO_LOAD_ID, AddressOf PFC_Test_at_No_Load)
        Call vbRunTask(MINIMUM_LOAD_TEST_ID, AddressOf Minimum_Load_Test)
```

```
        Call vbRunTask(FULL_LOAD_TEST_ID, AddressOf Full_Load_Test)
        Call vbRunTask(PFC_POWER_ON_SEQUENCE_ID, AddressOf PFC_Power_On_Sequence)
        Call vbRunTask(POWER_ON_GATE_ID, AddressOf Power_On_Gate)
        Call vbRunTask(SHORT_CIRCUIT_OUTPUT_RAIL_BOARD_ID, AddressOf Short_Circuit_Output_Rail_BOARD)
        Call vbRunTask(SHORT_CIRCUIT_OUTPUT_RAIL_MODULE_ID, AddressOf Short_Circuit_Output_Rail_MODULE)
        Call vbRunTask(PSS_POWER_ON_CURRENT_TEST_ID, AddressOf PSS_Power_ON_Current_Test)
        Call vbRunTask(POWER_ON_SEQUENCE_ID, AddressOf Power_ON_Sequence)
        Call vbRunTask(OVER_VOLTAGE_TEST_ID, AddressOf Over_Voltage_Test)
        Call vbRunTask(UNDER_VOLTAGE_TEST_ID, AddressOf Under_Voltage_Test)
        Call vbRunTask(POWER_INTERRUPTION_SEQUENCE_ID, AddressOf Power_Interruption_Sequence)
        Call vbRunTask(POWER_OFF_SEQUENCE_ID, AddressOf Power_OFF_Sequence)


        ' ---- VRAD TASK EXECUTION END ---- (do not remove this line)

        '
        ' --- Test Result management
        '
        AtosF.TplanResultSet(FailFlag)


        Testplan = 1
    End Function

End Module
```

It can be clearly seen how the *Testplan.vb* module simply recalls all the task defined in VRAD and runs them sequentially. It is also very important to highlight how the tasks follow the description given by the TRS in chapter 5. Since it would be impossible to report and analyse completely all the written tasks, the analysis and description of one of them will be made as exemplification for all the rest. As example the *T_Over_Voltage_Test* module will be reported and analysed.

Each task of the *Testplan* module is formed by its internal tests. These tests are characterized by a test number, a call to the function parameters, that the test will run, defined in VRAD and, of course, the function itself. The code of *test 329* of the *T_Over_Voltage_Test* task is reported below.

```
' ---- VRAD TEST BEGIN (329) ---- (do not remove this line)
' Update the site list
Call GetSiteList (gSiteList(0))

      ' #Test - 329
      '  Remark - "_Alimentazione UUT - 115V"
      TestNumber = 329
      While(BeginTest(TestNumber))
        Call SetTaskExecutionInfo (-1, "", TestNumber, "")

        ' ---- VRAD LOCAL TEST PARAMETERS BEGIN (329) ---- (do
           not remove this line)
        Dim parpACVoltage_329 As Double : parpACVoltage_329 =
           GetDoubleLocalParameter (OVER_VOLTAGE_TEST_ID,
           TestNumber, "pACVoltage")
        Dim parpGPIBAddress_329 As Double : parpGPIBAddress_329
           = GetDoubleLocalParameter (OVER_VOLTAGE_TEST_ID,
           TestNumber, "pGPIBAddress")
        Dim parpACAddress_329 As Double : parpACAddress_329 =
           GetDoubleLocalParameter (OVER_VOLTAGE_TEST_ID,
           TestNumber, "pACAddress")
        ' ---- VRAD LOCAL TEST PARAMETERS END ---- (do not
           remove this line)

        ' ---- VRAD LOCAL TEST LIBRARY CALLS (329) ---- (do not
           remove this line)
```

```
        Call fVRADSetUpAccensione (kpGND_REF.kGND_B,
            parpACVoltage_329, kpLoadType.Maximun,
            kmAmetek2253iX_OutputState.kOn, parpGPIBAddress_329,
            parpACAddress_329, TRUE, false)
        ' ---- VRAD LOCAL TEST LIBRARY CALLS END ---- (do not
            remove this line)

        BeforeEndTest
    End While
    EndTest
' ---- VRAD TEST END ---- (do not remove this line)
```

This specific test primarily aim is the Power On of the board under test. The function that fulfill this purpose, as shown in the code snippet, is called *fVRAD-SetUpAccensione* and its code is reported below.

```
Public  Function fSetUpAccensione(ByVal pGNDConnect As
   kpGND_REF, ByVal pACVoltage As Double, ByVal pLoadType As
   kpLoadType, ByVal pACOutputOn As kmAmetek2253iX_OutputState,
   ByVal pGPIBAddress As Double, ByVal pACAddress As Double,
   ByVal pVradStoreEnable As Boolean, ByVal pEnabledAC As
   Boolean,
                              Optional ByRef pSevereError As
                                 Boolean = False) As Long
  'Nome della procedura   : fPowerOn
  'Descrizione            : Alimentazione UUT
  'Parametri              :
  '                         pACVoltage: AC voltage programmed
     component [V].
  '                         pVoltageSteps: numeri di step per
     raggiungere la tensione pACVoltage
  '                         pCurrentLimitProgrammed: current
     programmed as AC generator current limit [A].
  '                         pCurrentLimitChecked: current
     limit used to check if the sinked current is a valid
     value [A].
  '                         pACFrequency: AC programmed
     frequency [Hz]
  '                         pOutputState: output state.
```

```
'                           kmAmetek2253iX_OutputState.kOn = 1        ON
'                           kmAmetek2253iX_OutputState.kOff = 2        OFF
'                         pLoadType: tipo di carico in uscita alla scheda
'                           kpLoadType.kSconnect = 1
'                           kpLoadType.kNoLoad = 2
'                           kpLoadType.kMinimum = 3
'                           kpLoadType.kTypical = 4
'                           kpLoadType.kMaximum = 5
'                           kpLoadType.kDefaultShort = 6
'                           kpLoadType.kDischarge = 7
'                         pGPIBAddress: indirizzo della scheda GPIB
'                         pACAddress: indirizzo GPIB del generatore AC
'Return value parameter :
'                         PASS = Programmazione effettuata correttamente
'                         FAIL = Errore di programmazione
'Release                : 1.00
'Date                   : 06.06.2019
'Maker                  : Luca Monterisi

Dim rTestResult As Long
Dim rActualSite As Integer
Dim rACOvercurrent As Boolean = False
Dim rDCOvercurrent As Boolean = False
Dim rProgrammedVoltage As Double
Dim rMeasuredCurrent As Double
Dim pACCurrentLimitChecked As Double = 10
Dim pACCurrentLimitProgrammed As Double = 3.25
Dim pACFrequency As Double = 400

Dim i As Integer = 1

```

```
sMsgPrintLog("@FG{Blue}" & fGetActualTaskName() & " ; " &
    fGetActualRemark() & " ; Test ID.n." & GetActualTestNum()
     & " --- Vac @" & CStr(pACVoltage) & "V", 0)


'--------------------- Abilitazione sicurezze
    -----------------------
'
    "12345678901234567890123456789012
Call InsulatedChDisconnect() 'Sezionamento Canali HV
rTestResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO,
    kpUserFlagPosition.kRack1_Option2, "
    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC", pSevereError)
If rTestResult <> PASS Then GoTo lblend

'------------- Sconnessione scaricatori Hold UP ----------
rTestResult = fPmxConnect(kpPowerMatrixId.kPMX1,
    kpPowerMatrixSectionId.kHVDISCR, kpPowerMatrixChannel.k_1
    , pSevereError)
If rTestResult <> PASS Then GoTo lblend

'----------------------- Condizioni Iniziali
    ------------------------
rTestResult = fFixtureSetUp(False, pSevereError)
If rTestResult <> PASS Then
    fSetUpAccensione = FAIL
    pSevereError = True
    GoTo lblend
End If

'---------------------------- Chiusura rel  masse
    -----------------------------
rTestResult = fGND_Connection(pGNDConnect, pSevereError)
If rTestResult <> PASS Then
    fSetUpAccensione = FAIL
    pSevereError = True
    GoTo lblend
End If
```

```

'--------------------  Connessione carichi UUT
    ---------------------
rTestResult = fLoadSetUp(pLoadType, pSevereError)
If rTestResult <> PASS Then GoTo lblend

'-------------------    Accencione AC a 10V
    ---------------------
rProgrammedVoltage = 10

rTestResult = fACGenAmetekPowerOn(
    kmAmetek2253iX_PhaseSelected.kPhaseR, rProgrammedVoltage,
     0, kmAmetek2253iX_VoltageRange.k300V, rProgrammedVoltage
     * 1.41 * 1.2, kmAmetek2253iX_VoltageSenseMode.kEnabled,
                             kmAmetek2253iX_VoltageSenseSource
                                .kExternal,
                                pACCurrentLimitProgrammed,
                                pACCurrentLimitChecked,
                                pACOutputOn,
                                kmAmetek2253iX_Waveform.
                                kSinusoid, pACFrequency,
                             pGPIBAddress, pACAddress,
                                parExtACRemoteEnable =
                                parEnabled)


If rTestResult <> PASS Then
    rACOvercurrent = True
    GoTo lblend
End If

'Programmazione Tensione al valore finale 115V
rProgrammedVoltage = pACVoltage
fACGenAmetekVoltageSet(kmAmetek2253iX_PhaseSelected.kPhaseR,
     kmAmetek2253iX_VoltageRange.k300V, rProgrammedVoltage,
    0, rProgrammedVoltage * 1.41 * 1.2,
    kmAmetek2253iX_VoltageSenseMode.kEnabled,
```

```
                    kmAmetek2253iX_VoltageSenseSource.kExternal, pGPIBAddress, pACAddress, parExtACRemoteEnable = parEnabled)

    '-------------------- Verifica assorbimento ----------------------
    Call fACGenAmetekMeasureCurrent(kmAmetek2253iX_PhaseSelected.kPhaseR, kmAmetek2253iX_MeasureCurrentMode.kAC, rMeasuredCurrent, pGPIBAddress, pACAddress, parExtACRemoteEnable = parEnabled)
    If rMeasuredCurrent > pACCurrentLimitChecked Then
        rACOvercurrent = True
        GoTo lblend
    End If

lblend:

    If rTestResult <> PASS Then
        Call fAmetek2253iX_OutputStateSet(kmAmetek2253iX_OutputState.kOff, parExtGPIBAddress, parExtACAddress, True)
    End If
    fSetUpAccensione = rTestResult

    'Stampa tipo di errore a video
    If rACOvercurrent = True Then
        Call MsgDispService("AC overcurrent during power on", 0)
    End If

    'Store risultato
    If pVradStoreEnable = True Then
        Call UseSiteRead(rActualSite)
        fStoreResult(GetActualTestNum, "", CInt(fSetUpAccensione), rActualSite, pSevereError)
    End If

End Function
```

The function sets all the preliminary operations necessary to power on the DUT in safety. This operations are:

- Safeties enable;
- Disconnection of the Hold-Up discharger;
- Initial Condition;
- Ground Connection;
- Active Load connection;
- Preliminary power on at 10 V;
- Power on at 115 V.

The safeties are related to the AC generator which supplies the DUT. The actual operation defined in the visual basic script is the closure of the 32th relay of the Spea user flags board. By closing this relay (properly wired to the fixture interface channels) the high voltage channels of the system are activated. The initial condition are obtained by the run of the *fFixtureSetUp* function which is recalled in line 57 in the script. Then the Ground connection is realized. This is a very significant operation: the DUT has two different ground net a GNDA net and a GNDB ground net. The ground connection defines which of the two will be connected to the system heart providing the common ground for all the measures that will be executed in the task. The active load connection, as the name suggests, create the connection for each output voltage rail to the according load. There are three different configuration allow: nominal load, maximum load and minimum load as reported in the TRS (chapter 5). Finally, when all the previous operation are correctly completed (a fail flag is otherwise set that lead to the end of the function), the actual power on of the DUT is performed. The power on is achieved in two steps: the first one, at a very low AC output voltage of 10 V, is used to verify the current absorption of the board; the second step brings the AC output to the nominal power level of 115 V, energizing completely the DUT.

7.2 Power Supply test (TPGM Debug)

The debug of the test program results in the final test of the Power Supply Module. The debug is usually carried out running step by step the function from the Visual Studio project directly. In this way the control over the running test is higher and it becomes easier to identify problems in the test program or in the board and to sharply stop its execution preventing any serious damage to the DUT. The first debug is also carried out with an oscilloscope which helps to verify that each operation is executed exactly as implemented in the TPGM.

The final goal is to obtain a completely debugged test program which can be used automatically. It also has to be pointed out that the board on which the first debug is executed is called Golden Board and it is assumed utterly functional.

In this section, the most significant debugged tests will be reported and analysed. Each section is identify by the name of the task as it is reported in the VRAD test plan excel sheet (figure 7.1).

7.2.1 Minimum and maximum load test

The minimum and maximum load test refers to the *PFC,DC/DC converters minimum and maximum load test* of the TRS (see chapter 5). The test aims to verify the output voltage rails values at different load configuration. It also verifies the V1 and V4 ramps @ power on (figure 5.9).

The ramps measure has been realized using directly the oscilloscope rather than the system internal DVM and the results is shown in figure 7.3.

7.2.2 Undervoltage, Overvoltage and power interruption test

The Undervoltage test aims to verify the power on of the AC_FAIL, PWR_FAIL and PS_RESET control signals when the AC input voltage drops from the nominal 115 Vac to 50 Vac. In figure 5.16b the waveform that should result from the test are shown.

In figure 7.4 the oscilloscope waveforms obtained during the debug operations are shown.

The external AC generator has been programmed to create a 65 voltage drop with a rising and falling time of 30 ms and an overall hole period of 150 ms (green

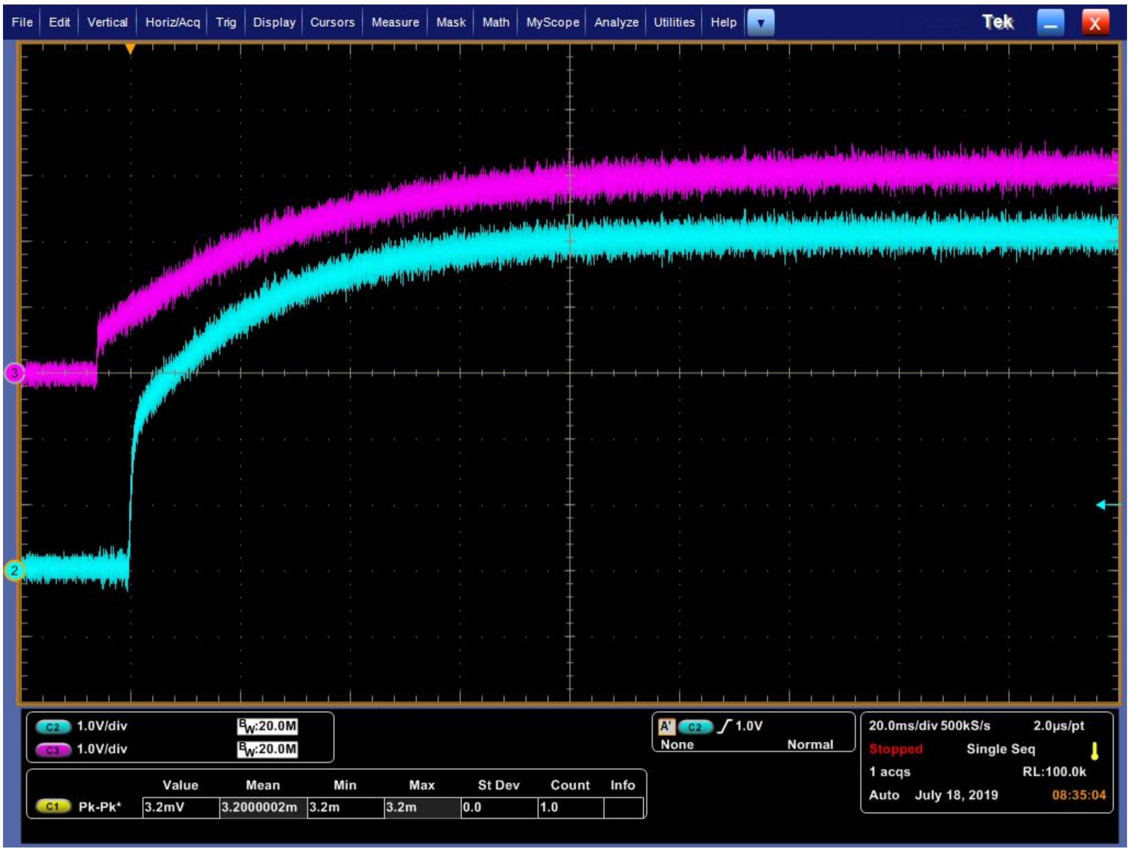


Figure 7.3: Oscilloscope acquired V1 and V4 ramps @ power on

curve in figure 7.4). The blue curve is the DVM timing. When a measure is carried out by the 3030 internal DVM a specific timing must be set. This timing is made of three components: a t_{off} a t_{on} and a second t_{off}. The measure by the DVM is executed at the end of the first off time just when the t_{on} is beginning. The blue curve represents exactly the passage between the t_{off}, the t_{on} and the second t_{off}, where the signal drops at the beginning of the t_{on} because the DVM responds to an inverse pulse. The purple curve represents the control signals and it is clear how it remains high for all the duration of the undervoltage.

Figure 7.5 shows the overvoltage test waveforms acquired by the oscilloscope.

The same considerations made for the undervoltage, concerning the timing curve and the control signal curve, stand still for the overvoltage too. The green curve shows the increase of the AC input voltage from 115 Vac to 160 Vac.

The power interruption test aims to verify the waveform shows in figure 5.17.

Figure 7.4: Oscilloscope undervoltage waveform

The acquired oscilloscope waveforms obtained during the debug are shown in figure 7.6.

The purple curve represents the voltage level VH of the Hold-Up capacitor. The blue curve is the DVM timing curve whose duration in this case is exactly equal to the AC voltage power interruption.

7.2.3 Power on and power on gate tests

The power on and power on gate tests have been defined in chapter 5 as timing tests. They aim to verify the correct timing sequence at the power on of the PSB of various control signals: figure 5.15 and figure 5.12 shows the timing to be verified.

Figure 7.7 and figure 7.8 show the comparison between the TRS test timing requirements and the oscilloscope waveform obtained during the PSM debug.

Figure 7.5: Oscilloscope overvoltage waveform

(a) Power on test TRS sequence

(b) Oscilloscope power on waveform

Figure 7.7: Power on test requirement vs debug waveform comparison

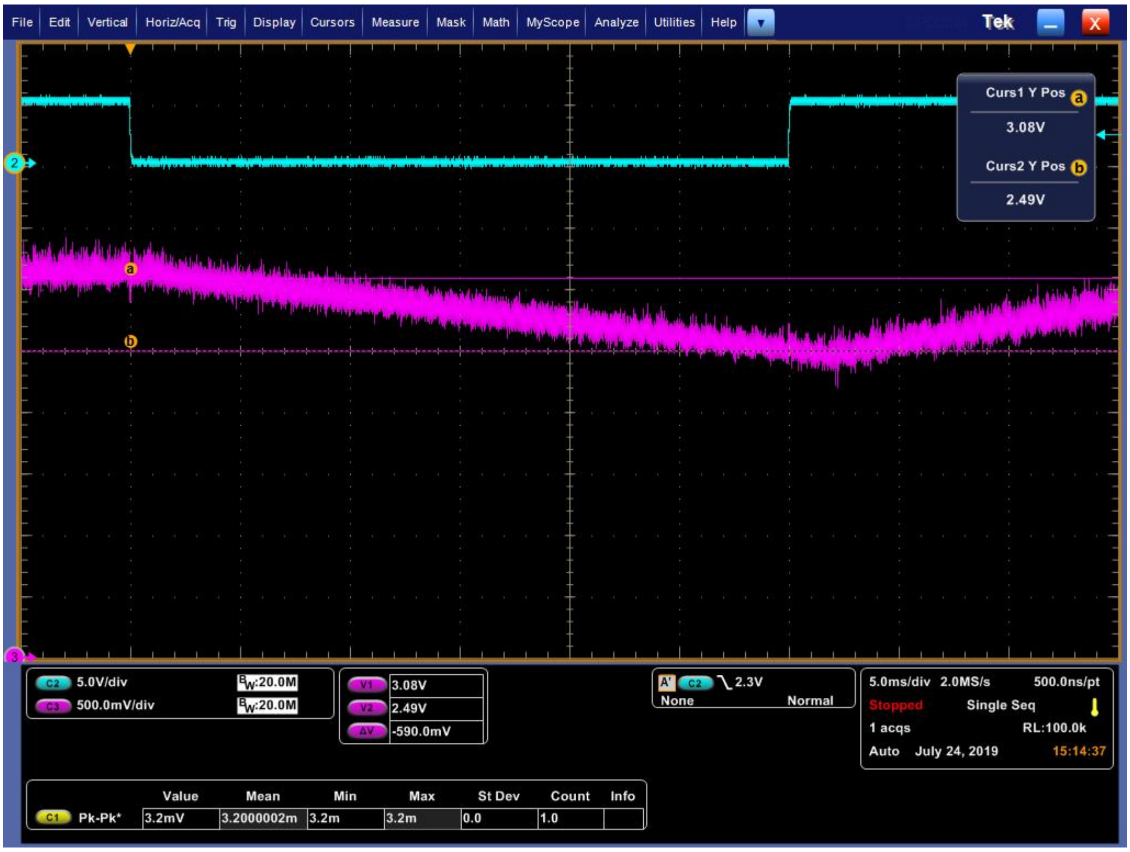
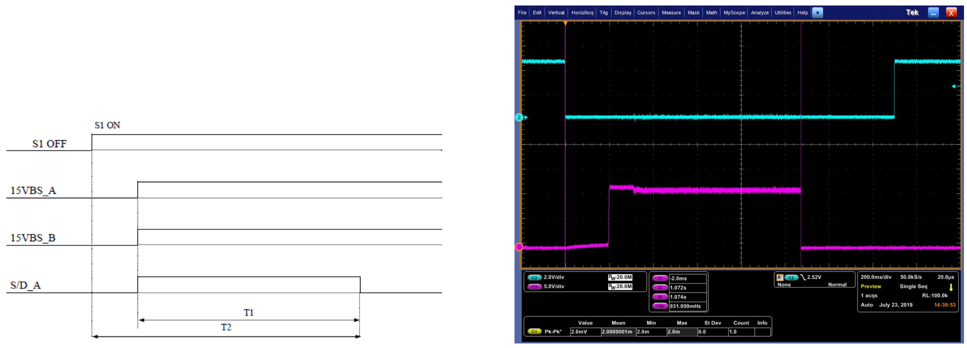


Figure 7.6: Oscilloscope power interruption waveform



(a) Power on gate test TRS sequence

(b) Oscilloscope power on gate waveform

Figure 7.8: Power on gate test requirement vs debug waveform comparison

Since it is quite challenging to obtain simultaneously all the signals reported in the TRS timing graphs, the tests have been repeated as many time as necessary. Specifically, both figure 7.7b and figure 7.8b show the SD_A control signal behaviour as the input voltage is turned off (purple curves). The blue curves, as in the previous tests, represent the DVM timing.

Chapter 8

Conclusion

Once the test program is debugged and verified as completely functioning, two different phases take place ending the application work: Acceptance and Installation. The first one is carried out in Spea: the client is notified that the debug process is complete and he is invited to Spea for the acceptance of the implemented test program. This process can take up to one week (more commonly just a couple of days). It is used to share with the client any non-compliance and deviation of the test program results from the Test Requirement Specification. The client will decide if a changing in the TRS is justifiable or if the TPGM must be modified accordingly. The second phase takes place in loco to the client's company. The installation aims to re-run completely the implemented and debugged test program on the ATE Spea system in the client possession. Depending on the complexity of the test program, this process can take up from one day to one week. In this case the installation has been carried out in Rome with a three days duration. The installation was successful and to date the client is using the implemented TPGM for the debug of various Power Supply Modules.

List of Acronyms

AC Alternating Current

ATE Automatic Test Equipment

BITE Built In Test Equipment

CAN Controller Area Network

CGPM General Conference on Weights and Measures

DC Direct Current

DUT Device Under Test

DVM Digital VoltMeter

EMI Electro-Magnetic Interference

EUT Equipment Under Test

FET Field Effect Transistor

FICT Fixtureless In-Circuit Test

FPS Floating Power Supply

HF High Frequency

ICs Integrated Circuits

ICT In-Circuit Test

IDE Integrated Development Environment

MOSFET Metal Oxide Semiconductor Field Effect Transistor

MSGU Mission Symbol Generator Unit

NZT Nodal Impedance Test

PCB Printed Circuit Board

PFC Power Factor Converter

PSB Power Supply Board

PSM Power Supply Module

PSS Power Supply System

PSU Power Supply Unit

PWM Pulse Width Mode

RF Radio Frequency

SI International System of Units

SMPS Switched Mode Power Supply

SoC System on Chip

STC Short Test Capacitance

STO Short Test Ohm

STZ Short Test Impedance

TRS Test Requirement Specification

UUT Unit Under Test

VHF Very High Frequency

VPM Video Processing Module

VRAD Very Rapid Application Development

Appendix A

```
 Option   Strict Off
Option Explicit On

Module modMain
    Dim AddTestplan As AtosF.TestplanDelegate = AddressOf Testplan
    Dim AddTestplanInit As AtosF.TestplanDelegate = AddressOf TestplanInit
    Dim AddTestplanEnd As AtosF.TestplanDelegate = AddressOf TestplanEnd
    Dim AddTestplanStop As TestplanDelegate = AddressOf TestplanStop

    Public Sub Main()
        Dim l As Integer
        Dim gmCode As Integer

        Call vbTplanSetStopControlFunction(AddTestplanStop)
        Call AtosF.vbTplanSetControlFunctions(AddTestplan, AddTestplanInit, AddTestplanEnd)
        Call AtosF.TplanSetVB()

        l = AtosF.TplanCreateWindow(VB6.GetHInstance.ToInt32(), False)

        Do
            gmCode = AtosF.TplanWinMsgLoop()
        Loop While (gmCode <> 0) And (gmCode <> -1)

        End

    End Sub

End Module

Module modTestplan

    Public Function Testplan() As Integer
        Dim FailFlag As Integer
```

```
        FailFlag = AtosF.PASS


        ' ---- VRAD TASK EXECUTION BEGIN ---- (do not remove this line)

        ' Read initial site list
        Call GetSiteList(gInitialSiteList(0))

        Call vbRunTask(SYSTEM_SET_UP_ID, AddressOf System_Set_Up)
        Call vbRunTask(AC_DC_CONVERTER_AND_AUX_PS_TEST_ID, AddressOf AC_DC_Converter_and_AUX_PS_TEST)
        Call vbRunTask(PWM_CLOCK_TEST_ID, AddressOf PWM_Clock_Test)
        Call vbRunTask(PFC_TEST_AT_NO_LOAD_ID, AddressOf PFC_Test_at_No_Load)
        Call vbRunTask(MINIMUM_LOAD_TEST_ID, AddressOf Minimum_Load_Test)
        Call vbRunTask(FULL_LOAD_TEST_ID, AddressOf Full_Load_Test)
        Call vbRunTask(PFC_POWER_ON_SEQUENCE_ID, AddressOf PFC_Power_On_Sequence)
        Call vbRunTask(POWER_ON_GATE_ID, AddressOf Power_On_Gate)
        Call vbRunTask(SHORT_CIRCUIT_OUTPUT_RAIL_BOARD_ID, AddressOf Short_Circuit_Output_Rail_BOARD)
        Call vbRunTask(SHORT_CIRCUIT_OUTPUT_RAIL_MODULE_ID, AddressOf Short_Circuit_Output_Rail_MODULE)
        Call vbRunTask(PSS_POWER_ON_CURRENT_TEST_ID, AddressOf PSS_Power_ON_Current_Test)
        Call vbRunTask(POWER_ON_SEQUENCE_ID, AddressOf Power_ON_Sequence)
        Call vbRunTask(OVER_VOLTAGE_TEST_ID, AddressOf Over_Voltage_Test)
        Call vbRunTask(UNDER_VOLTAGE_TEST_ID, AddressOf Under_Voltage_Test)
        Call vbRunTask(POWER_INTERRUPTION_SEQUENCE_ID, AddressOf Power_Interruption_Sequence)
        Call vbRunTask(POWER_OFF_SEQUENCE_ID, AddressOf Power_OFF_Sequence)


        ' ---- VRAD TASK EXECUTION END ---- (do not remove this line)

        '
        ' --- Test Result management
        '
```

```
        AtosF.TplanResultSet(FailFlag)


        Testplan = 1
    End Function

End Module

Option Explicit On
Option Strict On

Module modCSGT2

    Public Enum kpLoadType
        Disconnect = 0
        Discharge = 7
        Maximun = 8
        NoLoad = 9
        Typical = 10
        Minimum = 11
    End Enum

    Public Enum kpPowerSupplyConnection
        kNotConnected = 1
        kConnectedAndDisable = 2
        kConnectedAndEnable = 3
    End Enum

    Public Enum kpPowerSupplyStatus
        kEnable = 1
        kDisable = 2
        kSetEnable = 3
        kSetEnableRamp = 4
        kNotChange = 5
    End Enum

    'Costante per richiusura masse a massa di macchina
    Public Enum kpGND_REF
        kGND_A = 1               'Richiusura GND_A a massa di
           macchina
        kGND_B = 2               'Richiusura GND_B a massa di
           macchina
        KNone = 3                'Nessuna GND a massa di
           macchina
    End Enum

    Public Enum kpMeasType
        kRiseTime = 1
        kSettlingTime = 2
    End Enum
```

```
    Public Enum kpNoiseSource
        kpNoiseSource_V1 = 1
        kpNoiseSource_V2 = 2
        kpNoiseSource_V3 = 3
        kpNoiseSource_V4 = 4
        kpNoiseSource_V5 = 5
        kpNoiseSource_V6 = 6
        kpNoiseSource_V7 = 7
        kpNoiseSource_V8 = 8
        kpNoiseSource_VAUX_A = 9
        kpNoiseSource_VAUX_B = 10
        kpNoiseSource_POWER = 100
        kpDisconnectAll = 110
    End Enum

    'Dichiarazione di un Type per il settaggio deglle guardie
       digitali
    Public Enum kpSTAKLevel
        kpNotChange = -999999
        kpSTK_OFF = THREE_STATE
        kpSTK_LOW = LOW
        kpSTK_HIGH = HIGH
    End Enum

    'costante che determina se un eventuale stimolo e On o OFF
    Public Enum KpStato
        KOn = 1
        KOff = 0
        KSconnect = 100
        KConnect = 200
        KNotChange = -555
        KOpen = 100
        KClose = 200
        K_TTL_Low = -100000
        K_TTL_HIGH = -50000
    End Enum

    Public Structure strMultipleMeas
        Dim OutputName As String
        Dim ThLow As Double
        Dim ThHigh As Double
        Dim MeasValue As Double
        Dim MeasUnit As String
    End Structure

    Public rpMeasuredValue As Double = 0
    Public rpResistorValue As Double = 0
    Public rpPowerOffSequencePS_RESETDelay As Double = 0
    Public rpPowerOffSequencePower_FailDelay As Double = 0
```

```
    Public rpPowerInterruption_VHregime As Double = 0
    Public rpPowerInterruption_VHTransient As Double = 0
    Public rpPFCPowerOnSequence_T2 As Double = 0

    Public Function fStoreValue(ByRef pMemoryName As Double) As
       Integer

        'Nome della procedura    : fStoreValue
        'Descrizione             :
        'Parametri               : Variabile dove salvare il
           valore
        'Return value parameter :
        '                          PASS = Programmazione
           effettuata correttamente
        '                          FAIL = Errore di
           programmazione
        'Release                 : 1.00
        'Date                    : 30.07.2019
        'Maker                   : Luca Monterisi

        If pMemoryName = rpPFCPowerOnSequence_T2 Then
            rpMeasuredValue = rpMeasuredValue + 0.004
        End If

        If rpMeasuredValue < 0.000001 Then
            rpMeasuredValue = 0.000001
        End If

        pMemoryName = rpMeasuredValue

        Return PASS

    End Function

    Public Function fPowerOff(Optional ByVal pVradStoreEnable As
        Boolean = True) As Long

        Dim rProgrammingResult As Long = PASS
        Dim rActualSite As Integer
        Dim rErrorMessage As String = ""
        fPowerOff = PASS

        'Disabilitazione AC
        Call fAmetek2253iX_OutputStateSet(
           kmAmetek2253iX_OutputState.kOff, parExtGPIBAddress,
           parExtACAddress, parExtACRemoteEnable = parEnabled)
        If rProgrammingResult <> PASS Then GoTo lblend

        'Disabilitazione WFG1
        rProgrammingResult = fVRADWfgDisable()
```

```
        If rProgrammingResult <> PASS Then GoTo lblend
        'Disabilitazione DRI1
        rProgrammingResult = fDriDisable(kpDriverId.kDRI1)
        If rProgrammingResult <> PASS Then GoTo lblend
        ''Disabilitazione generatore DC
        'rProgrammingResult = fLAMBDA_GENH_SetOutputState(False, parExtGPIBAddress, parExtDC1Address, parExtDC1RemoteEnable = parEnabled)
        'If rProgrammingResult <> PASS Then GoTo lblend
        'Disabilitazione BSTV1
        rProgrammingResult = fBstDisable(kpBoosterId.kBSTV1)
        If rProgrammingResult <> PASS Then GoTo lblend
        'Disabilitazione BSTI1
        rProgrammingResult = fBstDisable(kpBoosterId.kBSTI1)
        If rProgrammingResult <> PASS Then GoTo lblend
        Call sTimeWait(200)

        ' Programmazione 0V
        rProgrammingResult = fWFGClear(kpWFGId.kWFG1)
        If rProgrammingResult <> PASS Then GoTo lblend
        rProgrammingResult = fDriClear(kpDriverId.kDRI1)
        If rProgrammingResult <> PASS Then GoTo lblend
        rProgrammingResult = fBstClear(kpBoosterId.kBSTV1)
        If rProgrammingResult <> PASS Then GoTo lblend
        rProgrammingResult = fBstClear(kpBoosterId.kBSTI1)
        If rProgrammingResult <> PASS Then GoTo lblend
        Call sTimeWait(2000)

        'Sconnessione TP
        fTpDisconnectAllAbus(kpMatrixABUSPoint.kRow1)
        fTpDisconnectAllAbus(kpMatrixABUSPoint.kRow2)
        fTpDisconnectAllAbus(kpMatrixABUSPoint.kRow3)
        fTpDisconnectAllAbus(kpMatrixABUSPoint.kRow4)

        '--------------------- Disabilitazione sicurezze AC -----------------------
        '                                                     "12345678901234567890123456789012
        rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX0")
        If rProgrammingResult <> PASS Then GoTo lblend

lblend:

        'Store misura
        If pVradStoreEnable = True Then
            Call UseSiteRead(rActualSite)
            If rProgrammingResult <> PASS Then
```

```
                rProgrammingResult = fStoreResult(GetActualTestNum, "", CInt(rProgrammingResult), rActualSite)
                'rProgrammingResult = fLAMBDA_GENH_SetOutputState(False, parExtGPIBAddress, parExtDC1Address, parExtDC1RemoteEnable = parEnabled)
            Else
                rProgrammingResult = fStoreResult(GetActualTestNum, "", CInt(fPowerOff), rActualSite)
                'rProgrammingResult = fLAMBDA_GENH_SetOutputState(False, parExtGPIBAddress, parExtDC1Address, parExtDC1RemoteEnable = parEnabled)
            End If
        End If

        If rErrorMessage <> "" Then
            MsgDispService("@FG{Red}" & rErrorMessage, 0)
        End If

        If rProgrammingResult <> PASS Then fPowerOff = rProgrammingResult

    End Function

    Public Function fGND_Connection(ByVal pGNDConnect As kpGND_REF, Optional ByVal pSevereError As Boolean = False) As Long

        'Nome della procedura    : fGND_Connection
        'Descrizione             : Connette GNDA o GNDB a massa di macchina e verifica l'effettiva connessione
        'Parametri               : pGNDConnect: identificatico di cosa connettere a massa di macchina
        '                            k_GND_A = 1    'Richiusura GND_A a massa di macchina
        '                            k_GND_B = 2    'Richiusura GND_B a massa di macchina
        '                            K_None = 3     'Nessuna GND a massa di macchina
        '                          pSevereError: errore grave settato a true dalla funzione se si verificano errori di programmazione
        'Return value parameter :
        '                          PASS = Programmazione effettuata correttamente
        '                          FAIL = Errore di programmazione
```

```
    'Release                  : 1.00
    'Date                     : 28.04.2019
    'Maker                    : Luca Monterisi

    Dim rProgrammingResult As Long
    Dim pConnectionResult As Long
    Dim rGNDConnect As kpSetActionType

    'Impostazione parametro per fSetAction
    Select Case pGNDConnect
        Case kpGND_REF.kGND_A
            rGNDConnect = kpSetActionType.kGNDA
        Case kpGND_REF.kGND_B
            rGNDConnect = kpSetActionType.kGNDB
        Case kpGND_REF.KNone
            rGNDConnect = kpSetActionType.kNoneGND
        Case Else
            Call MsgBox("GND connection mode not identified" & vbCrLf & "The program will be terminate", vbCritical, "fGND_Connection programming message")
            rProgrammingResult = FAIL
    End Select

    If rProgrammingResult <> PASS Then GoTo lblend

    'Connessione a massa di macchina
    rProgrammingResult = fSetAction(rGNDConnect, KpSetActionStatus.KConnect, "GND connection", pSevereError)
    If rProgrammingResult <> PASS Then GoTo lblend

    'Verifica connessione
    If rGNDConnect = kpSetActionType.kGNDA Then
        If rProgrammingResult <> PASS Or pConnectionResult <> PASS Then
            Call MsgBox("GNDA not connected to the system earth" & vbCrLf & "The program will be terminate", vbCritical, "fGND_Connection programming message")
            GoTo lblend
        End If

        'rProgrammingResult = fMeasureOpen(kpDriverId.kDRI1, parTP_GNDB, kpMatrixABUSPoint.kRow1, parTP_GNDA, kpMatrixABUSPoint.kRow4, 5, 0.01, kpDVMFilter.kLPF_25Hz, False, 100000, rpMeasuredValue, pConnectionResult, pSevereError)
        If rProgrammingResult <> PASS Or pConnectionResult <> PASS Then
```

```
                Call MsgBox("GNDB connected to GNDA" & vbCrLf & "The program will be terminate", vbCritical, "fGND_Connection programming message")
                GoTo lblend
            End If

        ElseIf rGNDConnect = kpSetActionType.kGNDB Then

            'rProgrammingResult = fLinkMeasure(30, kpDriverId.kDRI1, kpTPGND_B, kpDriverABUSPoint.kRow1, "", kpDriverABUSPoint.kRow4, 0.1, 0.01, kpDVMFilter.kLPF_25Hz, False, , pConnectionResult, pSevereError)
            rProgrammingResult = fMeasureResistanceVoltage(kpDriverId.kDRI1, parTP_GNDB, kpMatrixABUSPoint.kRow1, "", kpMatrixABUSPoint.kRow4, 0.2, 0.01, kpDVMFilter.kLPF_25Hz, False, 1.2, 10100, rpMeasuredValue, pConnectionResult, pSevereError)
            If rProgrammingResult <> PASS Or pConnectionResult <> PASS Then
                Call MsgBox("GNDB not connected to the system earth" & vbCrLf & "The program will be terminate", vbCritical, "fGND_Connection programming message")
                GoTo lblend
            End If

            'rProgrammingResult = fMeasureOpen(kpDriverId.kDRI1, parTP_GNDA, kpMatrixABUSPoint.kRow1, parTP_GNDB, kpMatrixABUSPoint.kRow4, 0.5, 0.01, kpDVMFilter.kLPF_25Hz, False, 10000, rpMeasuredValue, pConnectionResult, pSevereError)
            If rProgrammingResult <> PASS Or pConnectionResult <> PASS Then
                Call MsgBox("GNDA connected to GNDB" & vbCrLf & "The program will be terminate", vbCritical, "fGND_Connection programming message")
                GoTo lblend
            End If

        ElseIf rGNDConnect = kpSetActionType.kNoneGND Then
            rProgrammingResult = fMeasureOpen(kpDriverId.kDRI1, parTP_GNDB, kpMatrixABUSPoint.kRow1, "", kpMatrixABUSPoint.kRow4, 0.5, 0.01, kpDVMFilter.kLPF_25Hz, False, 100000, rpMeasuredValue, pConnectionResult, pSevereError)
            If rProgrammingResult <> PASS Or pConnectionResult <> PASS Then
```

```
                    Call MsgBox("GNDB not disconnected from the system earth" & vbCrLf & "The program will be terminate", vbCritical, "fGND_Connection programming message")
                    GoTo lblend
                End If

                rProgrammingResult = fMeasureOpen(kpDriverId.kDRI1, parTP_GNDA, kpMatrixABUSPoint.kRow1, "", kpMatrixABUSPoint.kRow4, 0.5, 0.01, kpDVMFilter.kLPF_25Hz, False, 100000, rpMeasuredValue, pConnectionResult, pSevereError)
                If rProgrammingResult <> PASS Or pConnectionResult <> PASS Then
                    Call MsgBox("GNDA not disconnected from the system earth" & vbCrLf & "The program will be terminate", vbCritical, "fGND_Connection programming message")
                    GoTo lblend
                End If

            End If

lblend:

            If rProgrammingResult <> PASS Or pConnectionResult <> PASS Then
                pSevereError = True
            End If

            fGND_Connection = rProgrammingResult + pConnectionResult

        End Function

        Public Function fFixtureSetUp(ByVal pVradStoreEnable As Boolean, Optional ByRef pSevereError As Boolean = False) As Integer

            'Procedure Name          : fFixtureSetUp
            'Description             : Setting della fixture necessari per effettuare il test (sicurezze etc.)
            'Parameter               :
            '                          pVradStoreEnable: Abilita/disabilita l'elaborazione dei risultati con VRAD
            '                          pSevereError: errore grave settato a true dalla funzione se si verificano errori di programmazione
            '
            'Return value parameter :
```

```
        '                          PASS = Programmazione effettuata correttamente
        '                          FAIL = Errore di programmazione
        '
        'Release                  : 1.00
        'Date                     : 06.06.2019
        'Maker                    : Luca Monterisi

        Dim rProgrammingResult As Integer
        Dim rActualSite As Integer

        'Accensione UPS1 per alimentazione rele 5V
        rProgrammingResult = fPpsuOn(kpProgPowerSupplyId.kPPSU1, 5, 1, pSevereError)
        If rProgrammingResult <> PASS Then GoTo lblend

        'Accensione UPS4 per alimentazione rele 12V
        rProgrammingResult = fPpsuOn(kpProgPowerSupplyId.kPPSU4, 12, 0.5, pSevereError)
        If rProgrammingResult <> PASS Then GoTo lblend

        'Accensione UPS3 per alimentazione led
        rProgrammingResult = fPpsuOn(kpProgPowerSupplyId.kPPSU2, 5, 0.5, pSevereError)
        If rProgrammingResult <> PASS Then GoTo lblend

        'Abilitazione UPS2 per alimentazione rele Buffer
        rProgrammingResult = fPpsuOn(kpProgPowerSupplyId.kPPSU3, 12, 0.5, pSevereError)
        If rProgrammingResult <> PASS Then GoTo lblend

        '------------------- Sconnessione scaricatore HV PMX ------------------------
        rProgrammingResult = fSetAction(kpSetActionType.kDischargeHV, KpSetActionStatus.KDisconnect, "fFixtureSetUp", pSevereError)
        If rProgrammingResult <> PASS Then GoTo lblend

        '-------------------- Abilitazione sicurezze AC (RL2 + ponticello su cavo J1) ----------------------
        rProgrammingResult = fSetAction(kpSetActionType.kACSafety, KpSetActionStatus.KOn, "fFixtureSetUp", pSevereError)
        If rProgrammingResult <> PASS Then GoTo lblend

lblend:

        fFixtureSetUp = rProgrammingResult

```

```
        'Store risultato
        If pVradStoreEnable = True Then
            Call UseSiteRead(rActualSite)
            fStoreResult(GetActualTestNum, "", CInt(fFixtureSetUp), rActualSite, pSevereError)
        End If

    End Function

    Public Function fSetAction(ByVal pSetActionType As kpSetActionType, ByVal pSetActionStatus As KpSetActionStatus, ByVal pTaskName As String, Optional ByVal pSevereError As Boolean = False, Optional ByVal pTimeWaitmsec As Integer = 250) As Integer

        Dim rProgrammingResult As Integer
        Dim rAttendi As Double
        Dim rNomeReleSHO As String

        rAttendi = 250

        Select Case pSetActionType

      'Connessione GNDA a massa di macchina
            Case kpSetActionType.kGNDA

                If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                    '    "12345678901234567890123456789012
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXOOXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                    '    "12345678901234567890123456789012
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXCXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
```

```
                Else
                    MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Connessione GND_A a massa di macchina")
                    rProgrammingResult = FAIL
                End If

        'Connessione GNDB a massa di macchina
            Case kpSetActionType.kGNDB

                If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                    '                                   "12345678901234567890123456789012
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXCOXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                    '                                   "12345678901234567890123456789012
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXOXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
                Else
                    MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Connessione GND_B a massa di macchina")
                    rProgrammingResult = FAIL
                End If

        'Disconnessione GNDA e GNDB da massa di macchina (KConnect = Disconnessione)
            Case kpSetActionType.kNoneGND

```

```
                If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                    '                                   "12345678901234567890123456789012
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXXCXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                    '                                   "12345678901234567890123456789012
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXXOXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
                Else
                    MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Disconnessione GND_A e GND_B da massa di macchina")
                    rProgrammingResult = FAIL
                End If

    'Abilitazione USER SERVICE
            Case kpSetActionType.kService
                If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO_NC, kpUserFlagPosition.kRack1_Option1, "XXXXXXXC", pSevereError)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                    rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO_NC, kpUserFlagPosition.kRack1_Option1, "XXXXXXXO", pSevereError)
```

```
            Else
                MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Abilitazione USER SERVICE")
                rProgrammingResult = FAIL
            End If

        'Connessione scaricatore HV PMX
        Case kpSetActionType.kDischargeHV
            If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                'rProgrammingResult = fPmxDisconnect(kpPowerMatrixId.kPMX1, kpPowerMatrixSectionId.kHVDISCR, kpPowerMatrixChannel.k_1, pSevereError) 'HV Discharger non pilotato
                rProgrammingResult = fPmxDisconnect(kpPowerMatrixId.kPMX1, kpPowerMatrixSectionId.kHPREL, kpPowerMatrixChannel.k_1, pSevereError) 'Reset RLHP1: connessione Com-NC
            ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                'rProgrammingResult = fPmxConnect(kpPowerMatrixId.kPMX1, kpPowerMatrixSectionId.kHVDISCR, kpPowerMatrixChannel.k_1, pSevereError) 'HV Discharger non pilotato
                rProgrammingResult = fPmxConnect(kpPowerMatrixId.kPMX1, kpPowerMatrixSectionId.kHPREL, kpPowerMatrixChannel.k_1, pSevereError) 'Set RLHP1: sconnessione Com-NC
            Else
                MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Connessione Scaricatore HV")
                rProgrammingResult = FAIL
            End If

        'Abilitazione sicurezze generatore AC
        Case kpSetActionType.kACSafety
            If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
```

```
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXC", pSevereError)
            ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXO", pSevereError)
            Else
                MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Abilitazione sicurezze AC")
                rProgrammingResult = FAIL
            End If

    'Inhibit per movimentazione castello
        Case kpSetActionType.kPreserPlateInhibit
            If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                '                                                        "12345678901234567890123456789012
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXCX", pSevereError)
            ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXOX", pSevereError)
            Else
                MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Inhibit movimentazione castello")
                rProgrammingResult = FAIL
```

```
            End If

        Case kpSetActionType.kp_SHO_FLAG_V1   'Abilitazione rel  CC per V1 NUFL5

            If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXCXXXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
            ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXOXXXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
            Else
                MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Corto Circuito V1")
                rProgrammingResult = FAIL
            End If

        Case kpSetActionType.kp_SHO_FLAG_V2   'Abilitazione rel  CC per V2 NUFL1

            If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
            ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
```

```
                    rProgrammingResult = fUFLConfigAll(
                        kpUserFlagGroup.kRlyNO,
                        kpUserFlagPosition.kRack1_Option2, "
                        0XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
                        pSevereError)
                Else
                    MsgBox("Stato non riconosciuto= " &
                        pSetActionStatus & "   " & pTaskName,
                        vbCritical, "Corto Circuito V2")
                    rProgrammingResult = FAIL
                End If

            Case kpSetActionType.kp_SHO_FLAG_V3   'Abilitazione
                rel  CC per V3 NUFL2
                If pSetActionStatus = KpSetActionStatus.KOn Or
                    pSetActionStatus = KpSetActionStatus.KConnect
                     Or pSetActionStatus = KpSetActionStatus.
                    KClose Then
                    rProgrammingResult = fUFLConfigAll(
                        kpUserFlagGroup.kRlyNO,
                        kpUserFlagPosition.kRack1_Option2, "
                        XCXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
                        pSevereError)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff
                     Or pSetActionStatus = KpSetActionStatus.
                    KDisconnect Or pSetActionStatus =
                    KpSetActionStatus.KOpen Then
                    rProgrammingResult = fUFLConfigAll(
                        kpUserFlagGroup.kRlyNO,
                        kpUserFlagPosition.kRack1_Option2, "
                        XOXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
                        pSevereError)
                Else
                    MsgBox("Stato non riconosciuto= " &
                        pSetActionStatus & "   " & pTaskName,
                        vbCritical, "Corto Circuito V3")
                    rProgrammingResult = FAIL
                End If

            Case kpSetActionType.kp_SHO_FLAG_V4   'Abilitazione
                rel  CC per V4 NUFL6
                If pSetActionStatus = KpSetActionStatus.KOn Or
                    pSetActionStatus = KpSetActionStatus.KConnect
                     Or pSetActionStatus = KpSetActionStatus.
                    KClose Then
                    rProgrammingResult = fUFLConfigAll(
                        kpUserFlagGroup.kRlyNO,
                        kpUserFlagPosition.kRack1_Option2, "
                        XXXXXCXXXXXXXXXXXXXXXXXXXXXXXXXX",
                        pSevereError)
```

```
            ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXXXXOXXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
            Else
                MsgBox("Stato non riconosciuto= " & pSetActionStatus & "   " & pTaskName, vbCritical, "Corto Circuito V4")
                rProgrammingResult = FAIL
            End If

        Case kpSetActionType.kp_SHO_FLAG_V5   'Abilitazione rel  CC per V5 NUFL3
            If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXCXXXXXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
            ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Or pSetActionStatus = KpSetActionStatus.KOpen Then
                rProgrammingResult = fUFLConfigAll(kpUserFlagGroup.kRlyNO, kpUserFlagPosition.kRack1_Option2, "XXOXXXXXXXXXXXXXXXXXXXXXXXXXXXXX", pSevereError)
            Else
                MsgBox("Stato non riconosciuto= " & pSetActionStatus & "   " & pTaskName, vbCritical, "Corto Circuito V5")
                rProgrammingResult = FAIL
            End If

        Case kpSetActionType.kp_SHO_FLAG_V6   'Abilitazione rel  CC per V6 NUFL4
            If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Or pSetActionStatus = KpSetActionStatus.KClose Then
```

```
                rProgrammingResult = fUFLConfigAll(
                    kpUserFlagGroup.kRlyNO,
                    kpUserFlagPosition.kRack1_Option2, "
                    XXXCXXXXXXXXXXXXXXXXXXXXXXXXXXX",
                    pSevereError)
            ElseIf pSetActionStatus = KpSetActionStatus.KOff
                Or pSetActionStatus = KpSetActionStatus.
                KDisconnect Or pSetActionStatus =
                KpSetActionStatus.KOpen Then
                rProgrammingResult = fUFLConfigAll(
                    kpUserFlagGroup.kRlyNO,
                    kpUserFlagPosition.kRack1_Option2, "
                    XXXOXXXXXXXXXXXXXXXXXXXXXXXXXXX",
                    pSevereError)
            Else
                MsgBox("Stato non riconosciuto= " &
                    pSetActionStatus & "   " & pTaskName,
                    vbCritical, "Corto Circuito V5")
                rProgrammingResult = FAIL
            End If

        Case kpSetActionType.kp_SHO_Tp_V1_100A    '

            rNomeReleSHO = parSHO_TP_V1_100A

            If pSetActionStatus = KpSetActionStatus.KOn Or
                pSetActionStatus = KpSetActionStatus.KConnect
                Then
                Call sTPSetDigitalSTKon(rNomeReleSHO,
                    kpSTAKLevel.kpSTK_HIGH)
            ElseIf pSetActionStatus = KpSetActionStatus.KOff
                Or pSetActionStatus = KpSetActionStatus.
                KDisconnect Then
                Call sTPSetDigitalSTKon(rNomeReleSHO,
                    kpSTAKLevel.kpSTK_OFF)
                Call sTimeWait(rAttendi)
            Else
                MsgBox("Stato non riconosciuto= " &
                    pSetActionStatus & "   " & pTaskName,
                    vbCritical, "Corto Circuito V1")
                rProgrammingResult = FAIL
            End If

        Case kpSetActionType.kp_SHO_Tp_V2_5A    '

            rNomeReleSHO = parSHO_Tp_V2_5A

            If pSetActionStatus = KpSetActionStatus.KOn Or
                pSetActionStatus = KpSetActionStatus.KConnect
                Then
```

```
                    Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_HIGH)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Then
                    Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_OFF)
                    Call sTimeWait(rAttendi)
                Else
                    MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Corto Circuito V2")
                    rProgrammingResult = FAIL
                End If

            Case kpSetActionType.kp_SHO_Tp_V3_5A   '

                rNomeReleSHO = parSHO_Tp_V3_5A

                If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Then
                    Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_HIGH)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Then
                    Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_OFF)
                    Call sTimeWait(rAttendi)
                Else
                    MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Corto Circuito V3")
                    rProgrammingResult = FAIL
                End If

            Case kpSetActionType.kp_SHO_Tp_V4_100A   '

                rNomeReleSHO = parSHO_TP_V4_100A

                If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Then
                    Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_HIGH)
                ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Then
```

```
            Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_OFF)
            Call sTimeWait(rAttendi)
        Else
            MsgBox("Stato non riconosciuto= " & pSetActionStatus & "   " & pTaskName, vbCritical, "Corto Circuito V4")
            rProgrammingResult = FAIL
        End If

    Case kpSetActionType.kp_SHO_Tp_V5_5A    '

        rNomeReleSHO = parSHO_Tp_V5_5A

        If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Then
            Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_HIGH)
        ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Then
            Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_OFF)
            Call sTimeWait(rAttendi)
        Else
            MsgBox("Stato non riconosciuto= " & pSetActionStatus & "   " & pTaskName, vbCritical, "Corto Circuito V5")
            rProgrammingResult = FAIL
        End If

    Case kpSetActionType.kp_SHO_Tp_V6_5A    '

        rNomeReleSHO = parSHO_Tp_V6_5A

        If pSetActionStatus = KpSetActionStatus.KOn Or pSetActionStatus = KpSetActionStatus.KConnect Then
            Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_HIGH)
        ElseIf pSetActionStatus = KpSetActionStatus.KOff Or pSetActionStatus = KpSetActionStatus.KDisconnect Then
            Call sTPSetDigitalSTKon(rNomeReleSHO, kpSTAKLevel.kpSTK_OFF)
            Call sTimeWait(rAttendi)
        Else
```

```
                    MsgBox("Stato non riconosciuto= " & pSetActionStatus & "  " & pTaskName, vbCritical, "Corto Circuito V6")
                    rProgrammingResult = FAIL
                End If

        End Select

        'Impostazione pSevereError
        If rProgrammingResult <> PASS Then
            pSevereError = True
        Else
            'pSevereError = False
        End If

        fSetAction = rProgrammingResult

    End Function

    Public Function fLoadSetUp(ByVal pLoadType As kpLoadType, Optional ByRef pSevereError As Boolean = False) As Integer

        'Nome della procedura   : fLoadConnect
        'Descrizione            : Connette i carichi nella modalit  scelta
        'Parametri              : pLoadType: tipologia di carico
        '                           kSconnect
        '                           kNoLoad
        '                           kMinimum
        '                           kTypical
        '                           kMaximum
        '                           kBoost          Connessione carico esterno BOOST load
        '                         pTestStep: Nome dello step da cui viene chiamata la funzione
        '                         pSevereError: errore grave settato a true dalla funzione se si verificano errori di programmazione
        'Return value parameter :
        '                         PASS = Programmazione effettuata correttamente
        '                         FAIL = Errore di programmazione
        'Release                : 1.00
        'Date                   : 25.04.2019
        'Maker                  : Luca Monterisi

        'Connessione carichi
        Select Case pLoadType
```

```
        Case kpLoadType.Disconnect, kpLoadType.NoLoad
            'Sconnessione carico V1=+5.18V
            fActiveLoadClear(kpActiveLoad.kActiveLoad1)
            'Sconnessione carico V2=+15V
            fActiveLoadClear(kpActiveLoad.kActiveLoad3)
            'Sconnessione carico V3=-15V
            fActiveLoadClear(kpActiveLoad.kActiveLoad6)
            'Sconnessione carico V4=+3.37V
            fActiveLoadClear(kpActiveLoad.kActiveLoad2)
            'Sconnessione carico V5=+5V
            fActiveLoadClear(kpActiveLoad.kActiveLoad4)
            'Sconnessione carico V6=-5V
            fActiveLoadClear(kpActiveLoad.kActiveLoad7)
            'Sconnessione carico Resistivo
            Call sTPSetDigitalSTKon(parTP_cmd_V2_Lmax, kpSTAKLevel.kpSTK_OFF)
            Call sTPSetDigitalSTKon(parTP_cmd_V3_Lmax, kpSTAKLevel.kpSTK_OFF)

            Call sTimeWait(1000)

        Case kpLoadType.Minimum
            'Connessione carico V1=+5.18V 5A
            fActiveLoadProgramming(kpActiveLoad.kActiveLoad1, 0, 5, kpActiveLoadOutFormat.kCONT_ON, pSevereError)
            'Connessione carico V2=+15V 30mA
            fActiveLoadClear(kpActiveLoad.kActiveLoad3)
            sTPSetDigitalSTKon(parTP_cmd_V2_Lmax, kpSTAKLevel.kpSTK_OFF)
            'Connessione carico V3=-15V 70mA
            fActiveLoadClear(kpActiveLoad.kActiveLoad6)
            sTPSetDigitalSTKon(parTP_cmd_V3_Lmax, kpSTAKLevel.kpSTK_OFF)
            'Connessione carico V4=+3.37V 5A
            fActiveLoadProgramming(kpActiveLoad.kActiveLoad2, 0, 5, kpActiveLoadOutFormat.kCONT_ON, pSevereError)
            'Connessione carico V5=+5V 1A
            fActiveLoadProgramming(kpActiveLoad.kActiveLoad4, 0, 1, kpActiveLoadOutFormat.kCONT_ON, pSevereError)
            'Connessione carico V6=-5V 1A
            fActiveLoadProgramming(kpActiveLoad.kActiveLoad7, 0, 1, kpActiveLoadOutFormat.kCONT_ON, pSevereError)

            Call sTimeWait(1000)

```

```
            Case kpLoadType.Maximun
                'Connessione carico V1=+5.18V 15A
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad1
                    , 0, 15, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'Connessione carico V2=+15V 100mA
                fActiveLoadClear(kpActiveLoad.kActiveLoad3)
                sTPSetDigitalSTKon(parTP_cmd_V2_Lmax,
                    kpSTAKLevel.kpSTK_HIGH)
                'Connessione carico V3=-15V 200mA
                fActiveLoadClear(kpActiveLoad.kActiveLoad6)
                sTPSetDigitalSTKon(parTP_cmd_V3_Lmax,
                    kpSTAKLevel.kpSTK_HIGH)
                'Connessione carico V4=+3.37V 15A
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad2
                    , 0, 15, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'Connessione carico V5=+5V 3A
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad4
                    , 0, 3, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'Connessione carico V6=-5V 3A
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad7
                    , 0, 3, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)

                Call sTimeWait(1000)

            Case kpLoadType.Discharge
                'Connessione carico V1=+5.18V
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad1
                    , 0, 15, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'SConnessione carico V2=+15V
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad3
                    , 0, 15, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'SConnessione carico V3=-15V
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad6
                    , 0, 3, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'Connessione carico V4=+3.37V
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad2
                    , 0, 15, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'Connessione carico V5=+5V
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad4
                    , 0, 3, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)
                'Connessione carico V6=-5V
```

```
                fActiveLoadProgramming(kpActiveLoad.kActiveLoad7
                    , 0, 3, kpActiveLoadOutFormat.kCONT_ON,
                    pSevereError)

                Call sTimeWait(1000)


            Case Else
                Call MsgBox("UUT Load type not identified" &
                    vbCrLf & "The program will be terminate",
                    vbCritical, "UUT loads programming message")

        End Select

        'Gestione errore funzione
        If pSevereError = True Then
            fLoadSetUp = FAIL
        Else
            fLoadSetUp = PASS
        End If

    End Function

    Public Function fScaricaCapacita(ByVal pTestTp As String,
        ByVal pReferenceTp As String, ByVal pMeasureHigh As
        Double, ByVal pMeasureLow As Double, ByVal pTimeoutSec As
         Double, Optional ByRef pSevereError As Boolean = False)
        As Double

        'Nome della procedura    : fScaricaCapacita
        'Descrizione             : Effettua la prova di abnormal
            voltage su un elenco di Pin
        'Parametri               : pTestTpList = elenco di punti
            sui cuali effettuare la prova
        '                          pReferenceTpList = punti di
            riferimento (tipicamente GND)
        'Ritorno della funzione : Esito
        '                           1 = FAIL
        '                           0 = PASS
        'Release                 :
        'Autore                  : L. Monterisi

        Dim rTensioneMisurata As Double
        Dim rTestResult As Long
        Dim rActualResult As Long
        Dim rStartTime As Long
        Dim rTimeOutms As Long
        Dim rActualSite As Integer
        Dim pVradStoreEnable As Boolean = True

```

```
        sMsgPrintLog("@FG{Blue}" & fGetActualTaskName() & " ; "
            & fGetActualRemark() & " ; Test ID.n." &
            GetActualTestNum(), 0)

        'Inizializzazione variabili
        rTestResult = PASS
        fScaricaCapacita = FAIL
        rTimeOutms = CLng(pTimeoutSec * 1000)

        'Connessione TP
        TpStrConnectAbus(pReferenceTp, ROW4)
        TpStrConnectAbus(pTestTp, ROW3)

        'Predisposizione Misura
        fDvmSet(DVM1, kpDVMInputStage.kHVLZ, kpDVMCoupling.
            kCPL_DC, kpDVMFilter.kLPF_NONE, kpDVMVRange.kR100V,
            kpDVMMeasMode.kNORMAL, kpDVMMeasType.kDC_MEAS,
            kpDVMEnableAcqRam.kDISABLE) 'F.S. 100V
        fDVMEnable(DVM1)
        fDVMConnectABUS(DVM1, ROW3, ROW4)

        'Attesa scarica sotto la tensione pericolosa
        AnlPhaseSet(APH1, 0.001, 0.001, 0.001, 1)
        AnlPhaseEnable(APH1)
        AnlPhaseSet(APH2, 0.001, 0.001, 0.001, 1)
        AnlPhaseEnable(APH2)
        AnlTimingEnable()
        rStartTime = GetTickCount
        Do
            'Esecuzione misura, con utilizzo comparatore YAPMU
            rActualResult = RunA(0)
            fDvmRead(kpDVMId.kDVM1, rTensioneMisurata,
                kpDVMMeasType.kDC_MEAS) 'Lettura valore misurato
            If rTensioneMisurata > 8 Or rTensioneMisurata < -8
                Then
                 rActualResult = FAIL
            End If
            If GetTickCount - rStartTime > rTimeOutms Then Exit
                Do  'Gestione Timeout
        Loop Until rActualResult = PASS

        'Scarica con strumento esterno
        If rActualResult = PASS Then
            fDvmSet(DVM1, kpDVMInputStage.kLV, kpDVMCoupling.
                kCPL_DC, kpDVMFilter.kLPF_2_5KHz, kpDVMVRange.
                kR10V, kpDVMMeasMode.kNORMAL, kpDVMMeasType.
                kDC_MEAS, kpDVMEnableAcqRam.kDISABLE) 'F.S. 10V,
                input stage protected
            AnlPhaseSet(APH2, 0.001, 0.01, 0.001, 1)

```

```
        'Connessione strumento di scarica
        TpStrConnectAbus(pTestTp, ROW1)
        fVRADDriverProgrammingABUS(kpDriverId.kDRI1, kpDriverABUSPoint.kRow1, kpDriverABUSPoint.kRow4, 0, 0.01, kpDriverCurrentRange.kR100mA, kpDriverOutMode.kDIRECT, kpDriverOutFormat.kCONT_ON, True)

        'Attesa scarica totale (con strumento esterno @ alta corrente)
        Do
            'Esecuzione misura
            rActualResult = RunA(0)
            fDvmRead(kpDVMId.kDVM1, rTensioneMisurata, kpDVMMeasType.kDC_MEAS) 'Lettura valore misurato
            If rTensioneMisurata > pMeasureHigh / 2 Or rTensioneMisurata < pMeasureLow / 2 Then
                rActualResult = FAIL
            End If

            If GetTickCount - rStartTime > rTimeOutms Then Exit Do  'Gestione Timeout
        Loop Until rActualResult = PASS

        If rActualResult = PASS Then

            'Impostazione corrente 100uA per scarica a bassa corrente
            fVRADDriverSourceSet(DRI1, 0, 0.00001, kpDriverCurrentRange.kR100uA, kpDriverOutMode.kDIRECT, kpDriverOutFormat.kCONT_ON)

            'Attesa scarica totale (con strumento esterno @ bassa corrente)
            Do

                'Esecuzione misura
                rActualResult = RunA(0)
                fDvmRead(kpDVMId.kDVM1, rTensioneMisurata, kpDVMMeasType.kDC_MEAS) 'Lettura valore misurato
                If rTensioneMisurata > pMeasureHigh Or rTensioneMisurata < pMeasureLow Then
                    rActualResult = FAIL
                End If

                If GetTickCount - rStartTime > rTimeOutms Then Exit Do  'Gestione Timeout
            Loop Until rActualResult = PASS
```

```

            End If

            TpStrDisconnectAbus(pTestTp, ROW1)

        End If

        'Gestione esito
        If rActualResult <> PASS Then
            If rTensioneMisurata > pMeasureHigh Or
               rTensioneMisurata < pMeasureLow Then
                rActualResult = FAIL
            Else
                rActualResult = PASS
            End If
        End If

        If rActualResult <> PASS Then
            rTestResult = FAIL
        End If

        'Sconnessione Test Point
        TpStrDisconnectAbus(pTestTp, ROW3)    'Scollega il test
           point sotto test da riga 3 di misura

        'Store risultato
        If pVradStoreEnable = True Then
            Call UseSiteRead(rActualSite)
            fStoreMeasures(GetActualTestNum, "",
               rTensioneMisurata, rActualSite, pSevereError)
        End If

        'Reset stimoli di sistema
        Call fDVMClear(DVM1)
        Call fVRADDriverClear(DRI1)
        Call TpStrDisconnectAbus(pTestTp, ROW3)
        Call TpStrDisconnectAbus(pTestTp, ROW1)
        Call TpStrDisconnectAbus(pReferenceTp, ROW4)

    End Function

```

```
    Public Function fMeasureTimeIntervalAfterSCHandlingAC(ByVal pCounterId As kpCounterId, ByVal pCounterIntervalResolution As kpCounterIntervalResolution, ByVal pCounterStartEventRange As kpCounterEventRange, ByVal pCounterStartEventThreshold As Double, ByVal pCounterStartEventSlope As kpCounterEventSlope, ByVal pCounterStopEventRange As kpCounterEventRange, ByVal pCounterStopEventThreshold As Double, ByVal pCounterStopEventSlope As kpCounterEventSlope, ByVal pStartTP As String, ByVal pStartRow As kpDVMABUSPoint, ByVal pStopTP As String, ByVal pStopRow As kpDVMABUSPoint, ByVal pReferenceTP As String, ByVal pReferenceRow As kpDVMABUSPoint, ByVal pCommandTP As String, ByVal pCommandRow As kpDVMABUSPoint, ByVal pToff1 As Double, ByVal pTon As Double, ByVal pToff2 As Double, Optional ByVal pDVMCoupling As kpDVMCoupling = kpDVMCoupling.kCPL_AC, Optional ByVal pFilter As kpDVMFilter = kpDVMFilter.kLPF_NONE, Optional ByVal pVradStoreEnable As Boolean = True, Optional ByRef pMeasuredValue As Double = -99999, Optional ByVal pLoopUntilPass As Integer = 0, Optional ByVal pLowThreshold As Double = -99999, Optional ByVal pHighThreshold As Double = 99999, Optional ByRef pSevereError As Boolean = False) As Long

        'Procedure Name         : fMeasureTimeIntervalAfterSCHandlingAC
        'Description            : Misura di intervallo di tempo sulle righe in seguito ad un corto circuito
        'Return value parameter :
        '                         PASS = Programmazione effettuata correttamente
        '                         FAIL = Errore di programmazione
        '
        'Release                : 2.00
        'Date                   : 05.06.2019
        'Maker                  : Luca Monterisi

        Dim rStartRow As kpMatrixABUSPoint
        Dim rStopRow As kpMatrixABUSPoint
        Dim rReferenceRow As kpMatrixABUSPoint
        Dim rCommandRow As kpMatrixABUSPoint
        Dim rProgrammingResult As Long
        Dim rDVMInputStage As kpDVMInputStage
        Dim rDVMVRange As kpDVMVRange
        Dim rActualSite As Integer
        Dim rCommandRowDriver As kpDriverABUSPoint
        Dim rReferenceRowDriver As kpDriverABUSPoint
        Dim rDummyInteger As Integer
        Dim rEsitoTest As Integer = FAIL
```

```
        Dim rLowThreshold As Double
        Dim rHighThreshold As Double
        Dim rCycles As Integer

        sMsgPrintLog("@FG{Blue}" & fGetActualTaskName() & " ; "
           & fGetActualRemark() & " ; Test ID.n." &
           GetActualTestNum(), 0)

        'Impostazione della riga a cui connettere il TP del CHA
           di misura
        Select Case pStartRow
            Case kpDVMABUSPoint.kRow1
                rStartRow = kpMatrixABUSPoint.kRow1
            Case kpDVMABUSPoint.kRow2
                rStartRow = kpMatrixABUSPoint.kRow2
            Case kpDVMABUSPoint.kRow3
                rStartRow = kpMatrixABUSPoint.kRow3
            Case kpDVMABUSPoint.kRow4
                rStartRow = kpMatrixABUSPoint.kRow4
            Case kpDVMABUSPoint.kRow5
                rStartRow = kpMatrixABUSPoint.kRow5
            Case kpDVMABUSPoint.kRow6
                rStartRow = kpMatrixABUSPoint.kRow6
            Case kpDVMABUSPoint.kRow7
                rStartRow = kpMatrixABUSPoint.kRow7
            Case kpDVMABUSPoint.kRow8
                rStartRow = kpMatrixABUSPoint.kRow8
            Case kpDVMABUSPoint.kNONE
                rStartRow = kpMatrixABUSPoint.kNONE
            Case Else
                Call MsgBox("ERROR! fMeasureTimeIntervalABUS
                   wrong pStartRow parameter" & vbCrLf & "The
                   program will be terminated", vbCritical, "
                   fMeasureTimeIntervalAfterSC")
        End Select

        'Impostazione della riga a cui connettere il TP del CHB
           di misura
        Select Case pStopRow
            Case kpDVMABUSPoint.kRow1
                rStopRow = kpMatrixABUSPoint.kRow1
            Case kpDVMABUSPoint.kRow2
                rStopRow = kpMatrixABUSPoint.kRow2
            Case kpDVMABUSPoint.kRow3
                rStopRow = kpMatrixABUSPoint.kRow3
            Case kpDVMABUSPoint.kRow4
                rStopRow = kpMatrixABUSPoint.kRow4
            Case kpDVMABUSPoint.kRow5
                rStopRow = kpMatrixABUSPoint.kRow5
            Case kpDVMABUSPoint.kRow6
```

```
                rStopRow = kpMatrixABUSPoint.kRow6
            Case kpDVMABUSPoint.kRow7
                rStopRow = kpMatrixABUSPoint.kRow7
            Case kpDVMABUSPoint.kRow8
                rStopRow = kpMatrixABUSPoint.kRow8
            Case kpDVMABUSPoint.kNONE
                rStopRow = kpMatrixABUSPoint.kNONE
            Case Else
                Call MsgBox("ERROR! fMeasureTimeIntervalABUS
                   wrong rStopRow parameter" & vbCrLf & "The
                   program will be terminate", vbCritical, "
                   fMeasureTimeIntervalAfterSC")
        End Select

        'Impostazione della riga a cui connettere il comando del
            rel  di cortocircuito
        Select Case pCommandRow
            Case kpDVMABUSPoint.kRow1
                rCommandRow = kpMatrixABUSPoint.kRow1
            Case kpDVMABUSPoint.kRow2
                rCommandRow = kpMatrixABUSPoint.kRow2
            Case kpDVMABUSPoint.kRow3
                rCommandRow = kpMatrixABUSPoint.kRow3
            Case kpDVMABUSPoint.kRow4
                rCommandRow = kpMatrixABUSPoint.kRow4
            Case kpDVMABUSPoint.kRow5
                rCommandRow = kpMatrixABUSPoint.kRow5
            Case kpDVMABUSPoint.kRow6
                rCommandRow = kpMatrixABUSPoint.kRow6
            Case kpDVMABUSPoint.kRow7
                rCommandRow = kpMatrixABUSPoint.kRow7
            Case kpDVMABUSPoint.kRow8
                rCommandRow = kpMatrixABUSPoint.kRow8
            Case kpDVMABUSPoint.kNONE
                rCommandRow = kpMatrixABUSPoint.kNONE
            Case Else
                Call MsgBox("ERROR! fMeasureTimeIntervalABUS
                   wrong pCommandRow parameter" & vbCrLf & "The
                   program will be terminate", vbCritical, "
                   fMeasureTimeIntervalAfterSC")
        End Select

        'Impostazione parametri DVM
        If pCounterStartEventRange = kpCounterEventRange.kHV Or
           pCounterStopEventRange = kpCounterEventRange.kHV Then
            rDVMInputStage = kpDVMInputStage.kHVHZ
            rDVMVRange = kpDVMVRange.kR100V
            'Divisione dei trigger dovuti alla divisione di
               tensione del DVM
```

```
            pCounterStartEventThreshold =
                pCounterStartEventThreshold / 10
            pCounterStopEventThreshold =
                pCounterStopEventThreshold / 10
        Else
            rDVMInputStage = kpDVMInputStage.kLV
            rDVMVRange = kpDVMVRange.kR10V
        End If

        While rCycles <= pLoopUntilPass And rEsitoTest <> PASS
            And rProgrammingResult = PASS

            'Spegnimento UUT
            rProgrammingResult =
                fVRADAmetek2253iX_OutputStateSet_Off()
            If rProgrammingResult <> PASS Then GoTo lblend
            Call sTimeWait(500)

            'Accensione UUT
            rProgrammingResult =
                fVRADAmetek2253iX_OutputStateSet_On()
            If rProgrammingResult <> PASS Then GoTo lblend
            Call sTimeWait(3000)

            'Impostazione latching
            rProgrammingResult = fAnlRelayModeSet(kpAnlRelayMode
                .kLatching, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Short circuit command point connection
            rProgrammingResult = fTpStrConnectAbus(pCommandTP,
                rCommandRow, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Start point connection
            rProgrammingResult = fTpStrConnectAbus(pStartTP,
                rStartRow, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Stop point connection
            rProgrammingResult = fTpStrConnectAbus(pStopTP,
                rStopRow, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Command Reference point connection
            rProgrammingResult = fTpStrConnectAbus(pReferenceTP,
                 rReferenceRow, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Programmazione DVM1
```

```
rProgrammingResult = fDvmSet(kpDVMId.kDVM1, rDVMInputStage, pDVMCoupling, pFilter, rDVMVRange, kpDVMMeasMode.kNORMAL, kpDVMMeasType.kDC_MEAS, kpDVMEnableAcqRam.kDISABLE, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Connessione DVM 1
rProgrammingResult = fDVMConnectABUS(kpDVMId.kDVM1, pStartRow, pStopRow, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Abilitazione DVM1
rProgrammingResult = fDVMEnable(kpDVMId.kDVM1, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Programmazione counter ***** EVENT RANGE  fisso a LV perch  tensione divisa da DVM ******
rProgrammingResult = fCntIntervalSet(pCounterId, pCounterIntervalResolution, kpCounterEventRange.kLV, pCounterStartEventThreshold, pCounterStartEventSlope, kpCounterEventRange.kLV, pCounterStopEventThreshold, pCounterStopEventSlope, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Connessione measure point
rProgrammingResult = fCntConnectDvm(pCounterId, kpDVMId.kDVM1, kpDVMId.kDVM2, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Enable counter
rProgrammingResult = fCntEnable(pCounterId, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Abilitazione phase 1
rProgrammingResult = fAnlPhaseEnable(kpTimingPhaseId.kAPH1, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Programmazione phase 1
rProgrammingResult = fAnlPhaseSet(kpTimingPhaseId.kAPH1, pToff1, pTon, pToff2, 1, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend

'Abilitazione phase 2 (obbligatoria per la misura)
rProgrammingResult = fAnlPhaseEnable(kpTimingPhaseId.kAPH2, pSevereError)
If rProgrammingResult <> PASS Then GoTo lblend
```

```
            'Programmazione phase 2 (obbligatoria per la misura)
            rProgrammingResult = fAnlPhaseSet(kpTimingPhaseId.
               kAPH2, pToff1, pTon, pToff2, 1, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Abilitazione phase 3 per driver 1
            rProgrammingResult = fAnlPhaseEnable(kpTimingPhaseId
               .kAPH3, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Programmazione phase 3 per driver 1
            rProgrammingResult = fAnlPhaseSet(kpTimingPhaseId.
               kAPH3, pToff1, pTon, pToff2, 1, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Abilitazione phase 4
            rProgrammingResult = fAnlPhaseEnable(kpTimingPhaseId
               .kAPH4, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Programmazione phase 4
            rProgrammingResult = fAnlPhaseSet(kpTimingPhaseId.
               kAPH4, pToff1, pTon, pToff2, 1, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Programmazione timing per driver 1
            rProgrammingResult = fDriTimingSet(kpDriverId.kDRI1,
                kpDriverPhaseId.kAPH3, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Programmazione corto circuito
            rProgrammingResult = fDriverProgrammingABUS(
               kpDriverId.kDRI1, rCommandRowDriver,
               rReferenceRowDriver, 5, 0.1, kpDriverCurrentRange
               .kR1A, kpDriverOutMode.kDIGITAL,
               kpDriverOutFormat.kD_PULSE, kpDriverSlewRate.
               kNORMAL, kpDriverCurrentLimitBypass.kBY_PASS_OFF,
                True, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Esecuzione misura
            RunA(0)

            'Lettura valore misurato
            rProgrammingResult = fCntRead(pCounterId,
               pMeasuredValue, rDummyInteger, pSevereError)
            If rProgrammingResult <> PASS Then GoTo lblend

            'Analisi valore misurato per LoopUntilPass
```

```
            If pVradStoreEnable = True Then
                GetThresholds(GetActualTaskId(), GetActualTestNum(), rLowThreshold, rHighThreshold)
                If pMeasuredValue < rLowThreshold Or pMeasuredValue > rHighThreshold Then
                    rEsitoTest = FAIL
                Else
                    rEsitoTest = PASS
                End If
            Else
                If pMeasuredValue < pLowThreshold Or pMeasuredValue > pHighThreshold Then
                    rEsitoTest = FAIL
                Else
                    rEsitoTest = PASS
                End If
            End If

lblend:

            fMeasureTimeIntervalAfterSCHandlingAC = rProgrammingResult

            'Store del valore misurato
            If pVradStoreEnable = True Then
                If rEsitoTest = PASS Or rCycles >= pLoopUntilPass Or rProgrammingResult <> PASS Then
                    Call UseSiteRead(rActualSite)
                    rProgrammingResult = fStoreMeasures(GetActualTestNum, "", pMeasuredValue, rActualSite, pSevereError)
                End If
            End If
            If rProgrammingResult <> PASS Then fMeasureTimeIntervalAfterSCHandlingAC = rProgrammingResult

            'Rimozione impostazione per programmazione corto circuito
            rProgrammingResult = fDriClear(kpDriverId.kDRI1)

            'Spegnimento UUT
            rProgrammingResult = fVRADAmetek2253iX_OutputStateSet_Off()
            If rProgrammingResult <> PASS Then fMeasureTimeIntervalAfterSCHandlingAC = rProgrammingResult
            Call sTimeWait(500)
```

```
            'DVM1 clear
            rProgrammingResult = fDVMClear(kpDVMId.kDVM1,
               pSevereError)
            If rProgrammingResult <> PASS Then
               fMeasureTimeIntervalAfterSCHandlingAC =
               rProgrammingResult

            'Counter clear
            rProgrammingResult = fCntClear(pCounterId,
               pSevereError)
            If rProgrammingResult <> PASS Then
               fMeasureTimeIntervalAfterSCHandlingAC =
               rProgrammingResult

            'Short circuit command point disconnection
            rProgrammingResult = fTpStrDisconnectAbus(pCommandTP
               , rCommandRow, pSevereError)
            If rProgrammingResult <> PASS Then
               fMeasureTimeIntervalAfterSCHandlingAC =
               rProgrammingResult

            'Start point disconnection
            rProgrammingResult = fTpStrDisconnectAbus(pStartTP,
               rStartRow, pSevereError)
            If rProgrammingResult <> PASS Then
               fMeasureTimeIntervalAfterSCHandlingAC =
               rProgrammingResult

            'Stop point disconnection
            rProgrammingResult = fTpStrDisconnectAbus(pStopTP,
               rStopRow, pSevereError)
            If rProgrammingResult <> PASS Then
               fMeasureTimeIntervalAfterSCHandlingAC =
               rProgrammingResult

            'Reference point disconnection
            rProgrammingResult = fTpStrDisconnectAbus(
               pReferenceTP, rReferenceRow, pSevereError)
            If rProgrammingResult <> PASS Then
               fMeasureTimeIntervalAfterSCHandlingAC =
               rProgrammingResult

            rCycles = rCycles + 1

        End While

    End Function
```

```
    Public Function fTimeComparison(ByVal pTime1 As Double, ByVal pTime2 As Double) As Long

        Dim rEsitoTest As Integer = FAIL
        Dim rActualSite As Integer
        Dim rProgrammingResult As Integer = PASS


        rpMeasuredValue = pTime2 - pTime1

        'Store del valore misurato
        Call UseSiteRead(rActualSite)
        rProgrammingResult = fStoreMeasures(GetActualTestNum, "", rpMeasuredValue, rActualSite)
        sDebugPrint("Meas: T2 - T1 > 0" & " --> " & CStr(Math.Round(rpMeasuredValue, 5)) & " msec --- Test ID.n." & GetActualTestNum)
        MsgPrintLog("@FG{Blue}" & fGetActualTaskName() & " ; " & fGetActualRemark() & " ; Test ID.n." & GetActualTestNum, 0)

        fTimeComparison = rProgrammingResult

    End Function

    Public Function fVoltageComparison(ByVal pVoltage1 As Double, ByVal pVoltage2 As Double) As Long

        Dim rEsitoTest As Integer = FAIL
        Dim rActualSite As Integer
        Dim rProgrammingResult As Integer = PASS

        rpMeasuredValue = pVoltage2 - pVoltage1

        'Store del valore misurato
        Call UseSiteRead(rActualSite)
        rProgrammingResult = fStoreMeasures(GetActualTestNum, "", rpMeasuredValue, rActualSite)
        sDebugPrint("Meas: DeltaVH = " & CStr(Math.Round(rpMeasuredValue, 5)) & " V --- Test ID.n." & GetActualTestNum)
        MsgPrintLog("@FG{Blue}" & fGetActualTaskName() & " ; " & fGetActualRemark() & " ; Test ID.n." & GetActualTestNum, 0)

        fVoltageComparison = rProgrammingResult

    End Function

End Module
```

Bibliography

[1] "https://en.wikipedia.org/wiki/Power_supply"

[2] IBM Customer Engineering Reference Manual - 736 Power Supply - 741 Power Supply - 746 Power Distribution Unit, October 29 1958, page 60-17. "http://www.mirrorservice.org/sites/www.bitsavers.org/pdf/ibm/704/223-6818_704_CE_Manual/736_741_746_PwrSupply_CE_Oct58.pdf"

[3] D.C. Bomberger, D. Feldman, D.E. Trucksess, S.J. Brolin and P.W. Ussery - The Spacecraft Power Supply System, February 11 1963, section 2.3: "A switching type series voltage regulator design was selected rather than quasi-linear series or shunt type regulators because of the higher efficiency that can be obtained with wide variations in battery voltage and load current." Describing the components used fot the Telstar satellite.

[4] "Switching-mode power supply - Patent Number: US3798531D, Filing date: 1972-06-05, Inventor: R. Allington"

[5] Arnold Knott, Toke M. Andersen, Peter Kamby, Jeppe A. Pedersen Mickey P. Madsen, Milovan Kovacevic, Michael A.E. Andersen - Evolution of Very High Frequency Power Supply - Technical University of Denmark, 2013. "http://orbit.dtu.dk/files/88233991/06680598.pdf"

[6] "https://en.wikipedia.org/wiki/Linear_regulator"

[7] "https://en.wikipedia.org/wiki/Switched-mode_power_supply"

[8] "https://en.wikipedia.org/wiki/Buck_converter"

[9] "https://en.wikipedia.org/wiki/Boost_converter"

[10] "https://en.wikipedia.org/wiki/Forward_converter"

[11] "https://www.spea.com/it/"

[12] Spea Handbook, Spea Academy training book. Author: Spea Academy

[13] D. Del Greco - Test Functions, Spea Academy training book, September 5 2015

[14] SI Brochure: The International System of Units (SI) [8th edition, 2006; updated in 2014] - Section 4.1: Non-SI units accepted for use with the SI, and units based on fundamental constants - "https://www.bipm.org/en/publications/si-brochure/chapter4.html"