2009-01-01

Health Prognisis of Electronics Via Power Profiling

Jonathan Amilcar Cervantes
University of Texas at El Paso, jacervantes2@utep.edu

HEALTH PROGNOSIS OF ELECTRONICS

VIA POWER PROFILING

JONATHAN A. CERVANTES

Department of Electrical and Computer Engineering

APPROVED:

Eric MacDonald, Ph.D., Chair

John Moya, Ph.D.

Jose F. Espiritu, Ph.D.

Patricia D. Witherspoon, Ph.D.
Dean of the Graduate School

This thesis is dedicated to my wife, children, and mother.

HEALTH PROGNOSIS OF ELECTRONICS VIA POWER PROFILING

by

JONATHAN A. CERVANTES, BSECE

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

May 2009

Acknowledgements

My gratitude and most sincere respect goes out to Dr. Eric MacDonald. I consider myself very fortunate for being under his guidance all along this research. Dr. MacDonald's expertise and guidance has in many ways given both shape and purpose not only to this thesis, but to a lot of practical and theoretical aspects of my young career as an electrical engineer. By assimilating Dr. MacDonald's quality of work and dedication to the subject at hand I have come to be exposed to the true nature of analytical research in the greatest extent possible. Not only do I reckon Dr MacDonald as the “Mentor”, but rather as a friend too.

My most honest gratitude also goes out to Dr. Moya. Dr. Moya's sincere manner of lecturing has in many instances led me to the most profound ways of thinking about VLSI. And it was through such, that I became acquainted with the very valuable skills. Much of my thought structure relating to VLSI devices is mirrored in the well learned lessons imparted by Dr. Moya. Throughout my bachelors and masters I always enjoyed the comfort of knowing I knew someone who understood VLSI. Dr. Moya's character and professionalism has left a solid and withstanding impression on my young career as an electrical engineer.

I would also like to thank Dr. Jose F. Espiritu for his involvement and contributions to the shape of this thesis.

In many regards not directly related to the work presented here, my wife has been the pillar of my enthusiasm and commitment to many great projects such as this. A million thanks to her, whose valuable support and natural wisdom has in many instances lead me to the right path.

Last but absolutely not least; my mother. I find it inevitable to digress and look back to the days in which we went through such hardship. I could have parted my future in many directions back in those days. But fortunately my mother was there to point me to the ones with a meaningful future. Thank you Mother.

Abstract

The objective of this research is to investigate a new approach for the early detection of latent defects in electronic devices in the field. Reliability is assessed through the non-traditional approach of recording and evaluating the power profile of electronic devices within a deterministic state of operation. Traditionally, measuring the quiescent current (I_{DDQ}) of a device has been employed in manufacturing tests to detect defective parts prior to deployment to the field. However, the monitoring of the deterministic power signature (i.e. boot up or during a self-test routine) has never been exploited to monitor the health of a device in the field through out the full product life cycle. Critical to the success of this approach is the development of a viable, low-cost monitoring system that can capture a power signature in a digital format and provide a statistical comparison to previous signatures in order to detect deviations in quality from the device in question. Consideration for the engineering trade-offs between

1. improved sensitivity to reliability problems, and
2. reduced size and cost

are both taken into account in this thesis, along with a description of a developed platform that will be employed to evaluate the effectiveness of the proposed approach through the capture of Digital Power Signatures (DPSs).

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The reliability of electronics in space and military applications is critical. Electronic devices specifically dedicated to applications ranging from missile guidance, unattended space mission cruiser controllers and satellite surveillance or geographic reconnaissance are examples of applications in which a high level of reliability is desired. Similarly, electronic devices found inside main frames dedicated to banking applications, personal computers and communication devices also share a great responsibility in safeguarding sensitive information and assuring solid financial transactions. Currently, advances in digital electronics have led to new alternatives to implement non-standard methods to detect reliability problems in electronic devices prior to undesired circumstances. The work presented on this thesis presents a non-traditional approach to discern between electronic devices that could potentially yield a degraded performance by analyzing their corresponding power signature during deterministic operation.

Under this thesis research the definition of a power signature will be the unique pattern of electrical current being drawn by a given device, for a predetermined amount of time under which the behavior of the device is expected to be deterministic (i.e. boot-up or a predefined self-test). In the case of CMOS integrated chips (ICs), for every transistor being active a direct current draw to the main power supply is made. Similar to a finger print, every current signature is unique. By strategically linking together many of the technologies, it is possible to develop a system capable of acquiring digital power profiles/signatures for the purpose of filtering anomalies and predicting hardware failures.

The primary goal of this thesis is to improvise and justify a reliable and inexpensive method for acquiring the digital power signatures of electronic devices. The signatures will be obtained by digitizing a scaled voltage corresponding to the electrical current being consumed, during a period of time under which the devices undergo a deterministic set of events. The net result will be a power signature reflecting a consistent pattern of electrical

current consumption (unless a reliability problem is present). The process is optimized to obtain the maximum information on the power signatures, while simultaneously reducing the storage size needed to allocate the signatures to a practical range for future commercialization. The overarching goal of this research is to develop a platform capable of unobtrusively detecting health declines in electronic devices. Such platform is intended to be efficient in terms of cost, area, data storage and power consumption. For this reason, the methodology and procedures under this thesis have been tailored to consider the future implementation of an economically viable single chip solution. Therefore, careful measures have been taken to efficiently carry out the digital power signature acquisition and processing without compromising unnecessary functionality or resource wasting.

Chapter 2: Previous Work

The previous work relating to the practice of obtaining electrical current signatures (power signatures) or monitoring the behavior of an electronic device via the amount of electrical current being consumed can be found across diverse fields of application. Within this chapter, the details of the previous work relating to some form of electrical current signature acquisition, sensing, or profiling for the purpose of monitoring or evaluating an electric device are presented. The most similar in nature and purpose to the work presented on this thesis is I_{DDQ} testing[1]. I_{DDQ} testing has been standard of the Complementary Metal-Oxide-Semiconductor (CMOS) Integrated Circuit (IC) industry for decades. Currently, as CMOS technology and deep sub-micron Very-Large-Scale Integration (VLSI) embedded devices continue reducing in die area, I_{DDQ} testing is expected to become a deprecated practice in the near future, as described in [15]. However, I_{DDQ} testing has prevailed for a substantial amount of time as industry finds new methods to adapt I_{DDQ} testing to the changing manufacturing environment given the ample range of fault coverage provided.

Section 2.1 introduces the core ideas behind I_{DDQ} testing in the manufacturing environment. Section 2.2 delves into the implementation of I_{DDQ} testing along with a simplified time-line. Second to I_{DDQ} testing is Motor Current Signature Analysis (MCSA). MCSA is a branch of the power signature acquisition practice within the mechanical engineering field. Whereby, the approach of interpreting a real-time electrical current signature to predict the wear of specific machinery has been successful. In the mechanical engineering realm, Motor Current Signature Analysis (MCSA) has had a tremendous impact on the early detection of broken rotor bars for industrial grade squirrel cage induction

1 *In electronics, same double letter subscripts such as cc or dd refer to overall power supply voltages or currents. Originally, the term "IDD" was a designation to reference the flow of electrical current from a power supply connected to the "drain" terminal of a bipolar transistor. However, the notion has expanded to cover not only a single transistor, but rather the entire transistor infrastructure of a chip as a whole. Hence, IDDQ refers to the overall electrical current consumption of a chip under a quiescent state of operation.*

motors (SCIM), as described in [3]. MCSA will be presented in section 2.3 and contrasted to the core of this thesis at the end.

Other fields related in essence to the work presented on this thesis are the power monitoring of electrical systems with the purpose of battery life expansion. Electric current profiling has been intricately related to the optimization of battery usage. Many sequential discharge algorithms have caused a profound impact in the multi-battery power supply systems. Out of which, the most efficient monitor and make use of the electric current load to determine the best possible alternation between the available batteries, as described in [18]. Sequential discharge algorithms, although not a novel practice among industry's trend to maximize battery life expectancy, have had a great impact among fields other than industry related electronic yields. High end solar cars make use of it extensively, high end Radio Control (RC) aircraft and even satellites make use of it as well, as described in [19]. Sequential discharge algorithms will be presented in section 2.4.

Section 2.1: I_{DDQ} Testing in The Manufacturing Environment

Originally embraced by industry in the mid 80's, I_{DDQ} testing has become an integral part of CMOS IC quality control by yielding many benefits at a low price, according to [13]. I_{DDQ} testing is an inescapable phase of CMOS IC evaluation against manufacturing faults of either a design or process origin. The idea behind I_{DDQ} testing is rather simple but effective. Within the complementary metal-oxide semiconductor (CMOS) realm there is an outstanding advantage in terms of power consumption. CMOS devices are known for their close-to-nothing power consumption upon operating in a quiescent state. A quiescent state is a state in which every transistor within the internal switching mesh is allowed to reach a definitive state. Under this state, CMOS devices draw electrical current in the order of a few nano amperes, as described in [13]. This is how a substantial difference between a defective device and a non-defective device arises. If there is a manufacturing fault pertaining to either a process or a design flaw, the device in question will consume an I_{DDQ} electrical current several orders of magnitude greater to the expected quiescent case. Within the manufacturing environment, many faults can alter the expected IC. I_{DDQ} testing is capable of detecting the majority of the processing defects such as, bridging, malformed traces, mask problems, incomplete edging, gate-oxide shorts, inner-gate shorts, stuck-on faults, and logical redundant effects. In terms of design defects, I_{DDQ} testing is capable of detecting floating gates, logic contention, and mask generation errors.

I_{DDQ} testing requires the use of test vectors to initialize flip flops inside the IC. With this, the correspondent p or n field effect transistors (FETs) are provided with a gate bias. The possibility to sensitize and test the great majority of the transistors within a given IC makes I_{DDQ} testing a desirable process for quality control purposes, as described in [4]. A test vector is a predetermined set of logic values. The length of the vector varies according

to the number of available inputs on the IC. The generation of the test vector may be of a random nature (depending on the metrics and standards), though the usual practice is to elaborate a set of test vectors capable of sensitizing the majority of transistors inside the IC. The test vectors are created mainly via a Stuck at Fault (SAF) pattern generator. The SAF pattern generator is a software that utilizes the files generated at the extraction phase in the design process (a netlist describing the circuit under investigation in terms of a gate level model). The software scans the extraction layout files in accordance with predetermined parameters for how probable defects might arise due to the physics behind semiconductor devices and the limits imposed on the proximity of the components. All possible weaknesses and the derived input vector pairs capable of exposing such faults are subsequently recorded. An example of such software is "faultsim". The resulting number of test vectors can be vast depending on the complexity of the IC's transistor level infrastructure. Nevertheless, a single vector is capable of initializing up to 50% of the logic inside the IC to either 0 or 1, as described in [10]. Other tools take advantage of this fact and use the information provided by faultsim to reduce the number of test vectors depending on the localization of the fault and the level of redundancy among the vectors.

Full fault coverage may not always be possible. Specifically when the combinatorial or synchronous sequential logic (or both), is extremely complex. Even though the number of test vectors is very large, the fault coverage provided by them will not always encompass 100%. However, I_{DDQ} testing does an outstanding job for what it costs. Let alone, I_{DDQ} is an effective Statistical Process Monitor and Control (SPMC) tool by itself, according to [13]. Yet another benefit provided by I_{DDQ} testing, is the inadvertent burn-in test being conducted simultaneously to the I_{DDQ} test. Burn-in screening is an inherent part of semiconductor manufacturing in order to assure reliability and long term life expectancy of the IC, as described in [17]. Burn-in testing is advantageous for decreasing the failure rate of the

product during early field life, according to [17]. Burn-in testing is a modified practice of the more general High Voltage Stress (HVS) test performed on all semiconductors, not just CMOS devices. When applying the predetermined test vectors to the target IC, careful measures are taken to set the voltages in order to conduct both an I_{DDQ} test and a High Voltage Stress test.

On the down side, I_{DDQ} testing is not able to detect all possible faults. I_{DDQ} testing is not capable of detecting any defect which does not ultimately lead to a perceptible rise in the supply of electrical current as to surpass a predetermined threshold value. Such faults might include highly resistive interconnects, open defects that do not elevate I_{DDQ}, defects that can possibly render a transistor non conductive, transmission gate faults that can possibly lead to weak logic, and dynamic interconnects that can possibly lead to a capacitive or inductive coupling.

The threshold value for I_{DDQ} (I_{DDQ} threshold) to which the I_{DDQ} of the device under test is compared against, is a parameter of critical importance. If the value for I_{DDQ} threshold is set too low, many good ICs will be discarded under the impression their I_{DDQ} profile is over the threshold. Vice versa, if the the value for I_{DDQ} threshold is set too high many truly bad devices will be overlooked. From many such standpoints as financial, performance, practical, quality, even trademark and prestige, the value imposed on the threshold electrical current I_{DDQ} must yield the highest number of truly reliable good devices without compromising overconfidence. A method used by industry to determine I_{DDQ} threshold is to plot the readings of the individual I_{DDQ} profiles of a large number of ICs chosen at random. The collected data is then analyzed to determine an acceptable value for I_{DDQ} threshold. The next section will introduce the implementation of I_{DDQ} testing from the manufacturing

environment stand point. Technical details and a more in depth description of I_{DDQ} testing are presented.

Section 2.2: I_{DDQ} Testing Implementation

The most common form of I_{DDQ} testing is direct electrical current (DC) I_{DDQ} testing. Under all circumstances, a DC I_{DDQ} test is regulated and controlled by what is known as the tester. The tester is the controlling element over the test environment and procedure. All components within the test environment are subject to the tester. The tester is pre-programmed with the input test vectors before beginning the test and how much time to hold them active before moving on, as to assure a burn-in test simultaneously. The following figure illustrates the rest of the components.

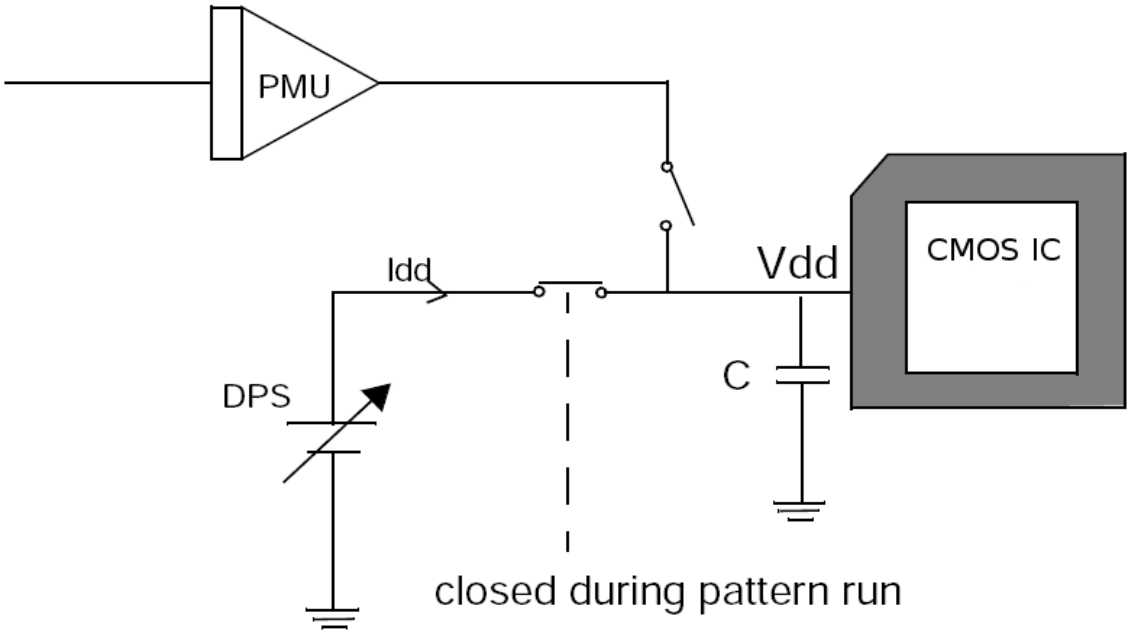


Figure 2.1: Typical circuit used for DC IDDQ testing.

The DPS corresponds to the Device Power Supply. This element is in charge of supplying the necessary electrical power to the CMOS IC under test. The DPS must be

robust enough to sustain all ongoing transients inside the IC and allow for them to settle into a quiescent state once a different input vector has been applied. For the actual reading of I_{DDQ}, the tester decouples the DPS from the IC to allow the capacitor to provide I_{DDQ}. The input vectors do not depend on the DPS, they remain active even though the DPS is disconnected. The device's outputs must not be connected to any load at all. The Phasor Measurement Unit (PMU), is responsible for the reading of I_{DDQ}. Once the tester has disconnected the DPS from the circuit, a predetermined amount of time before reading I_{DDQ} must be allowed. The reason for this is because the RC circuit formed by the capacitor and the IC must be allowed to reach a steady state of electrical current flow. Then, on behalf of the tester the PMU samples the electrical current flow between the capacitor and the target CMOS IC. A recursive Discrete Fourier Transform (DFT) algorithm is used by the PMU to carefully calculate I_{DDQ}.

Therefore, in order to fully test a device the tester loads the IC into a connection chamber. Then, the tester connects the DPS and sets in the first test vector into the input pins of the target. A predetermined amount of time is then given by the tester for the transients inside the device to settle as the input vector stimulates the switching mesh inside. Once the device is given enough time to reach a quiescent state the DPS is disconnected by the tester and the decoupling capacitor takes over providing I_{DDQ}. Another lapse of time must be given to the RC circuit to reach a steady state. When the tester acknowledges this last lapse of time has passed, the PMU is instructed to measure I_{DDQ}. The newly acquired I_{DDQ} is contrasted against I_{DDQ} threshold. If the reading is higher, the device under test (DUT) is discarded. Otherwise, the DPS is connected again to the circuit and the next test vector is loaded. Subsequently, the sequence of events just described are repeated again until the DUT either fails upon any test vector or passes them all.

One of the disadvantages of I_{DDQ} testing is the amount of time it takes to perform. The following enhancement to I_{DDQ} testing addresses this issue. In a Functional I_{DDQ} test, the approach is to augment the DUT with a resistor in series in order to derive a voltage divider circuit and subsequently calculate I_{DDQ} via the former resistor. The following figure illustrates this concept.

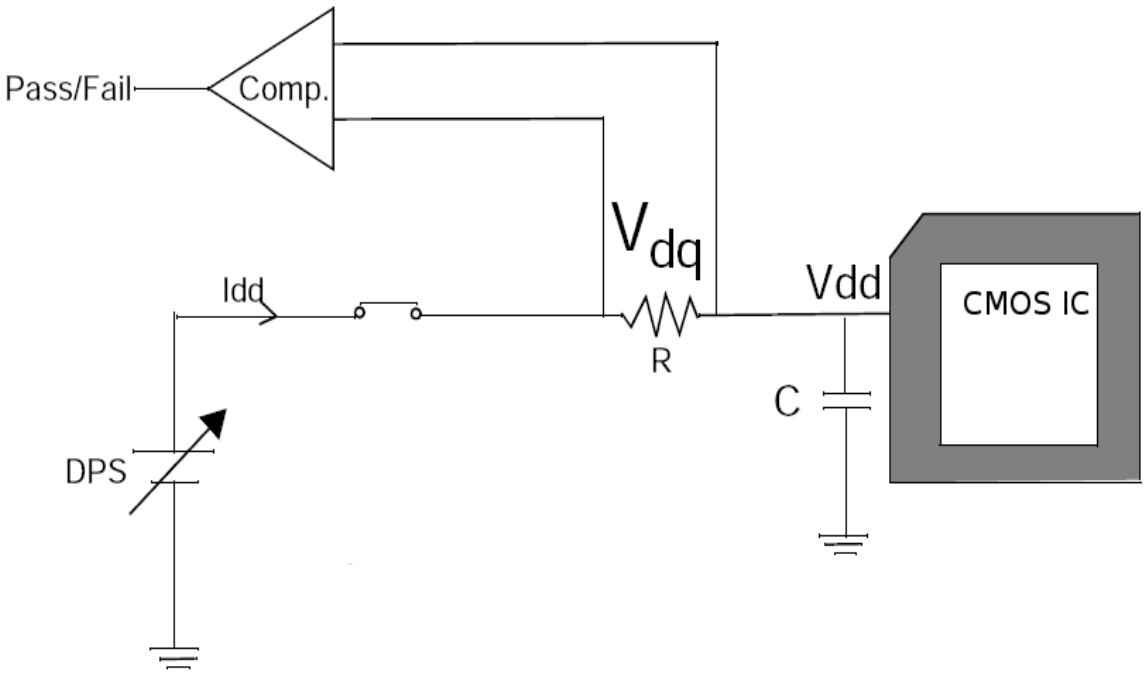


Figure 2.2: Typical circuit used to perform a Functional IDDQ test.

An important remark for this approach is the fact that the DPS is not decoupled from the circuit as the test is being conducted (hence the voltage divider). A voltage comparator is used to detect unacceptable levels of I_{DDQ} passing through R. Therefore, the voltage drop across R must not exceed the voltage imposed by I_{DDQ} threshold multiplied by R. The capacitor now takes on the role of voltage stabilizer rather than voltage source while the

test is being conducted. Regarding the procedure of the test, everything else remains the same. The test vectors are applied as usual one by one and the the DUT is presumed to be non-defective if all yield acceptable levels of I_{DDQ}. Although this scheme is able to speed the process, the resistor R becomes a great point of failure to underestimate as it becomes entirely responsible for the pass or fail of the DUT.

Subsequently, a true improvement came about with the introduction of the Keating-Mayer sensor. This approach came from a voltage differential through time perspective. This scheme was quickly adopted by industry as it increased confidence on the quality of the test itself, and did not consume as much time as previous approaches. The following figure shows the Keating-Mayer sensor being applied to I_{DDQ} testing.

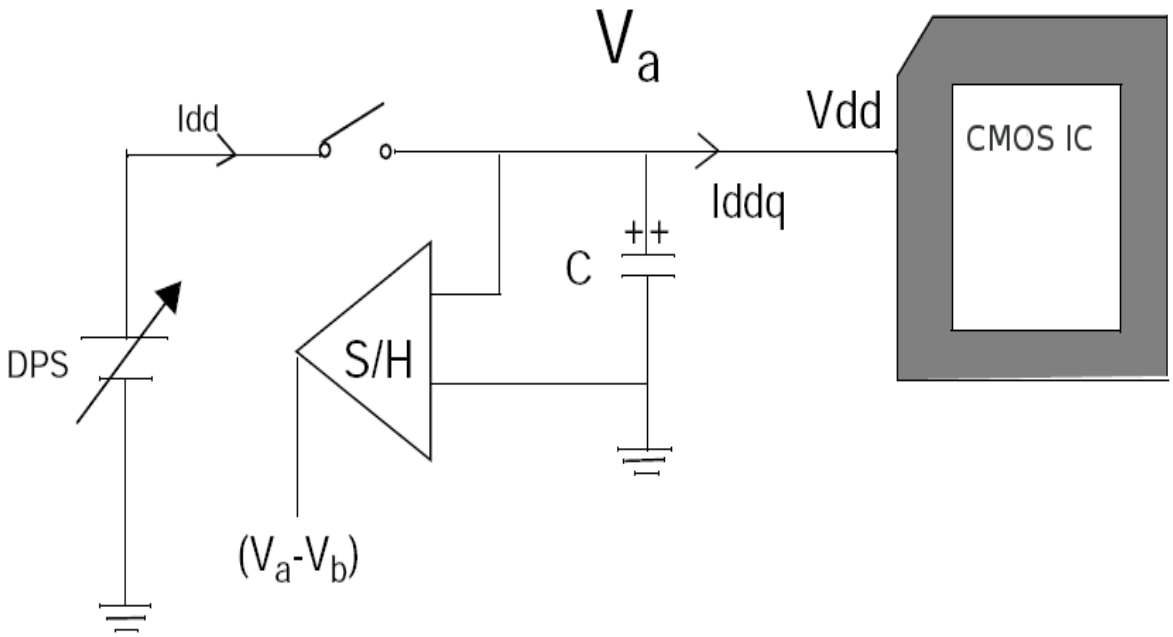


Figure 2.3: Keating-Mayer circuit.

In this scheme, the DPS is isolated once the test vector is loaded and all transients

have settled. Leaving the large decoupling capacitor C responsible for suppling I_{DDQ}. As the voltage across C decays exponentially, the sample and hold circuitry (S/H) acquires two voltage readings Va and Vb respectively. The tester records the time interval in between the two (Δt). I_{DDQ} is then equated using the following formula: $I_{DDQ} = C(Va-Vb)/\Delta t$. The following figure further illustrates the the concept.

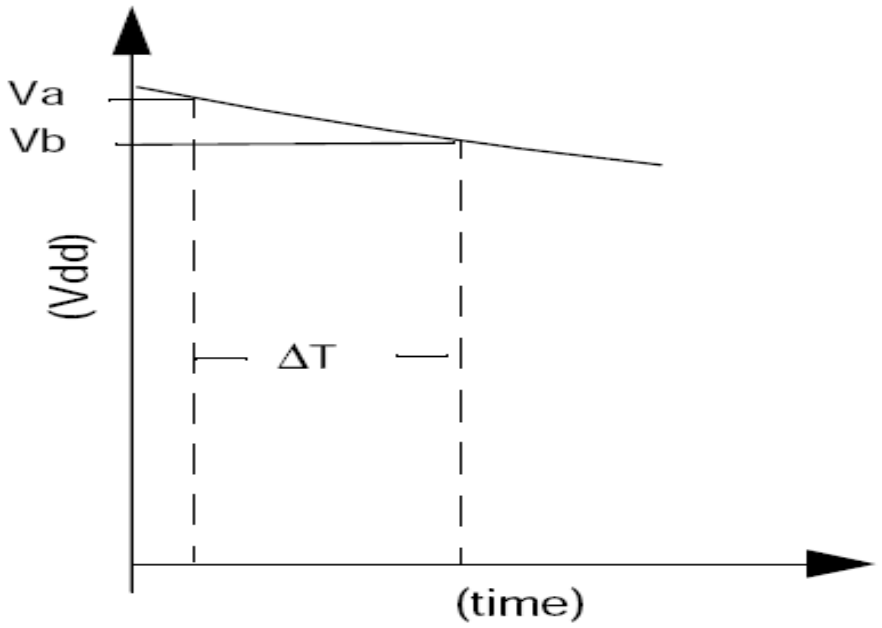


Figure 2.4: Exponential decay of Vdd and associated measurements vs time.

In the early 1990s, the benefits of I_{DDQ} testing had spread across the CMOS IC industry quality community according to [21]. Companies like IBM encouraged I_{DDQ} testing in production. Increasing awareness of off-shore industries now competing in the CMOS IC industry justified the need to raise the quality standards to remain competitive at home and abroad, as described in [21]. In 1996, awareness to bring a new approach to I_{DDQ} testing was established by noting that I_{DDQ} in the traditional form was not enough to strain all

defects, according to [5]. It was discovered that not all CMOS defects are directly reflected to ground in a resistive mode. Rather, two or more nodes within the switching interconnect structure could be resistively or directly tied together as the result of a defect, as described in [5]. According to Gattiker et al:[7], ”We consider “passive defects” to be those defects that involve only non-switching nodes of the circuit. They provide a direct, static electric current path between VDD and GND and produce a constant level of electrical current on all test vectors. Examples include direct shorts between VDD[2] and GND[3] and leaky non-switching reversed-biased pn junctions, such as between the well and substrate.” These were the types of faults captured by traditional I_{DDQ} testing. Whereas according to Gattiker et al:[7], ““Active defects," in our terminology, on the other hand, involve switching nodes of the circuit. Examples of active defects include shorts between a switching node and any other node. Active defects produce abnormal electrical current consumption levels on vectors that activate them and no abnormal electrical current consumption levels on vectors that do not activate them. In other words, they result in multilevel electrical current signatures. Observe now that because passive defects do not involve switching nodes of the circuit, they do not have a direct impact on the quality of information-carrying signals in the circuit and, as such, are unlikely to cause a logical fault. On the other hand, "active" defects do affect switching nodes and therefore may cause performance failure.”

There was now a need to differentiate between these types of faults and elaborate on the traditional way I_{DDQ} testing was carried out to be able to strain these faults as well. The same I_{DDQ} testing strategies explained previously still remained unchanged. The difference now was not to discard the readings of the individual vectors, and build an electrical current consumption histogram per DUT. With this approach, it was now possible to filter out these

2 *In electronics VDD refers to the overall electrical voltage either consumed or specific of a electronic device. VDD is also used to denote the positive (cathode) terminal of an electronic component or device.*

3 *In electronics GND is an abbreviation for ground. The ground of an electric circuit is used to single out the return path of electrical current flow (anode). GND is a point in the electrical circuit used to denote a zero voltage potential. GND is also used as a label for the negative (anode) terminal of a given electronic circuit.*

faults by having the tester scan for steeps or abrupt jumps in I_{DDQ} values. Furthermore, it was then possible to categorize those devices with an acceptable level of passive defects but absolutely no active defects, as possible candidates for the market. As they did not compromised functionality, but rather just consumed a fraction more of electrical current than normal, according to [7].

These changes brought a new perspective to I_{DDQ} testing and a new criteria for isolating defective devices from non-defective devices. For both practical and technical reasons, maintaining a real-time record of the present electrical current signature made the process both more resource demanding and more time consuming to perform on typical testers, as described in [7]. Per [7], "Producing an electrical current signature from a die's I_{DDQ} results, by storing and sorting all measurements is impractical in a production environment; however, it is possible to apply the differential electrical current scheme suggested by the electrical current signature in a way that captures the essential attributes of a die's electrical current signature, without requiring storage of all measurements or sorting of the measurements. The methodology involves applying I_{DDQ} vectors until one of three stopping criteria is met." This methodology quickly became the standard within the manufacturing environment because of the reduction in I_{DDQ} testing time. Per [7], the three criteria are as follows: "As described above, the first, most-easily visible attribute of an electrical current signature is the presence or absence of a "step," where a step indicates the presence of an active defect. The first stopping criterion screens out die with active defects on the basis of the existence of such a step. The criterion involves detection of an I_{DDQ} value that is different from the measurements that proceed it by some threshold value. In practice, this criterion requires storage of only the minimum and maximum values of I_{DDQ} seen prior to the current vector and comparison of the current vector to the stored

minimum and maximum. An approximate means for carrying out this criterion is to store only the first measurement, compare all subsequent I_{DDQ} values to that measurement and stop upon detecting a value that is greater or less than that measurement by a chosen threshold value. The second stopping criterion addresses "no-step" electrical current signatures. In many situations passive defects causing only low levels of constant electrical current may not harm circuit function. Dies with such defects should not be rejected during testing. Other situations may exist, however, where some passive defects are unacceptable, for example if they cause a static power limitation to be violated or present a reliability concern. The second criterion involves detection of a measurement that is above some limit (determined by such power dissipation or reliability concerns). This criterion requires only comparison to a reference electrical current value. The third criterion is detection of a chosen number of measurements of similar magnitude, where the number is chosen to provide acceptable confidence that any further measurements would also produce the same levels of electrical current. Meeting the third, but not the first or second stopping criterion, indicates the die contains no unacceptable defects. The length of the test set should be set to this number."

On the other front, the electric current sensing side of I_{DDQ} (not the handling of the data) was also undergoing drastic changes to speed up the test procedure and increase reliability on the data that needed to be collected: the I_{DDQ} threshold value set for detecting active defects and the one to detect passive defects. The Multi-resolution I_{DDQ} Testing (MIT) scheme depicted by [11] was designed to swiftly provide such readings. The following figure illustrates the circuit.

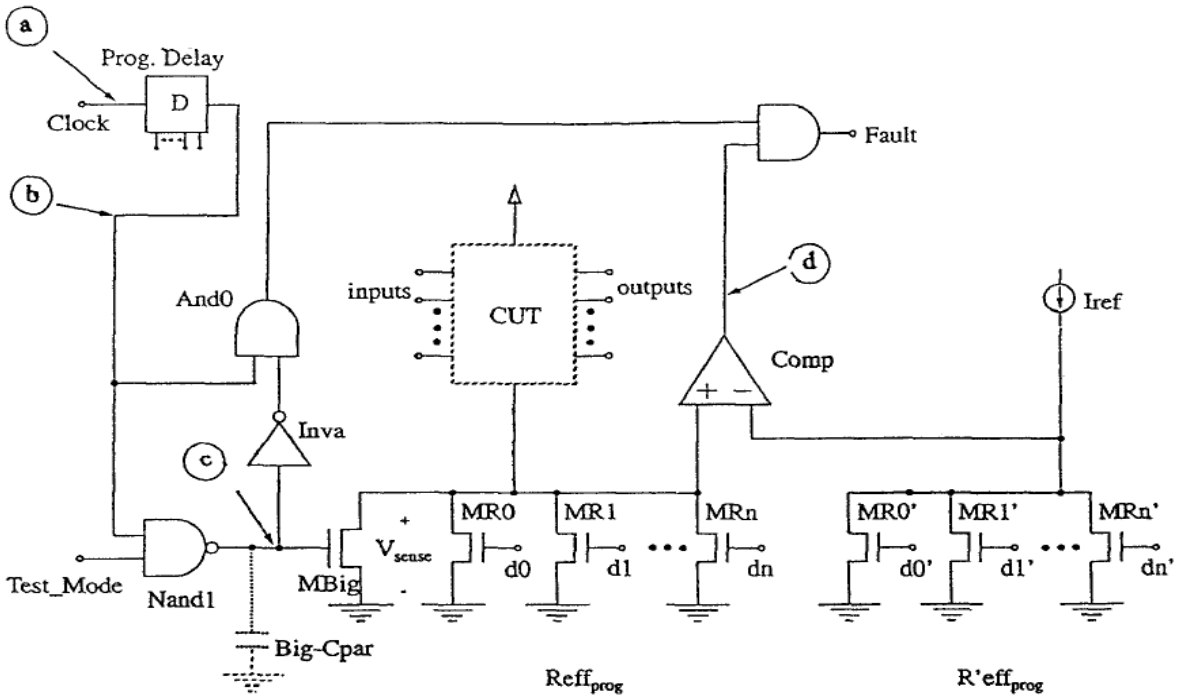


Figure 2.5: The Multi-resolution IDDQ testing circuit.

The details of how this circuit functions are as follow. A clock pulse is used to initialize the test. As this clock pulse is received by the circuit, it will be delayed by the "Prog Delay" bock. The delay lapse can be manipulated by a data word through the input pins shown bellow the block. The purpose of delaying the signal is to allow the electrical current provided by the power supply enough time to settle to I_{DDQ}, per [11]. The purpose of the "Test Mode" NAND gate is to trigger a gradual reduction in the low impedance imposed by Mbig once a clock pulse is received. This means, the tester sets the "Test Mode" line high prior to expecting the clock pulse. Mbig is a non-traditional transistor tailored to have a very low impedance (=< 1 ohm) while the output of the test mode NAND gate is high, and a very high impedance (> 1k ohm) once the NAND gate has ceased to supply any voltage and "Big-Cpar" has gently but quickly dropped the voltage to zero. What this means for the

actual test is less time spent in waiting for the electrical current across the circuit under test (CUT) to settle, per [11]. The other notable aspect of this circuit is the flexibility in sensing various levels of I_{DDQ}, introduced by the word programmable transistors MR0 through MRn (on the CUT side) and MR0' through MRn' (on the Iref side). These transistors are set to operate as large valued resistors, and both sets (CUT and Iref) must receive the same data word to accurately read I_{DDQ}.

The next level to which I_{DDQ} testing was taken was to the wafer level. By the late 1990s new approaches were being developed, as Ultra Large Scale CMOS ICs were being introduced. The introduction of Built In Current Sensors, per [1], was a long term solution to increasingly small feature sizes and even smaller I_{DDQ} differences between bad and good devices. The new order of I_{DDQ} testing was to be decentralized. Hence, the power net inside the area of the die was now to be partitioned and tailored to be capable of electrically isolating the functional blocks inside a given IC. Each functional block in turn, was acquainted with a Built In Current Sensor (BICS) capable of determining the specific I_{DDQ} through such block according to the test vector being applied, as described in [1]. This approach came to be known as the Wafer Oriented Test Evaluation (WOTE). The following circuit illustrates the concept on a partition basis.

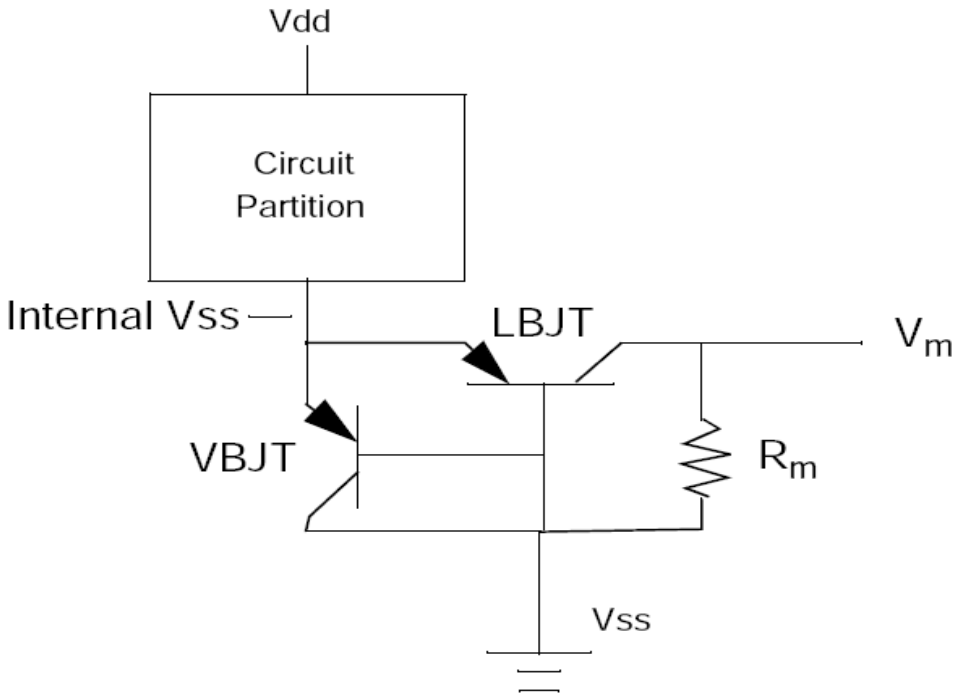


Figure 2.6: The BICS circuit per partition.

From this figure we can observe the voltage Vm being proportional to I_{DDQ} given the partition. This scheme brought two great advantages for the present challenges. First, by decentralizing I_{DDQ} it was now possible to reintroduce a marked difference between the I_{DDQ} from a bad partition and the corresponding I_{DDQ} threshold, as opposed to measuring the bulk of electrical current being consumed by the die. It also provided for a better localization of the faults (should there be one or several). Which in turn speed up the process of addressing such by localizing the fault(s) on a specific partition. Another valuable benefit was the absence of a need to package a defective device, according to [1]. The devices which passed all the BICS tests were diced and packaged. Devices with unacceptable levels of I_{DDQ} could easily be sent back to be processed again. An optional yield from this approach

is how BICS can continuously monitor I_{DDQ} and prompt for action in the field (after testing and packaging). This is because the embedded BICS remained on the IC. From the electrical current profile perspective, consumption histograms were now possible to be elaborated on a partition basis for further quality control. On the down side, BICS introduced overhead caused by processing bipolar transistors over CMOS and the silicon real state dedicated to them, according to [20]. Refinements of BICS approach are presented in the work by [12].

The processing of the data collected by these approaches to I_{DDQ} testing underwent changes too. The quality of the DUT no longer depended on the I_{DDQ} test alone anymore. The differences in I_{DDQ} from vector to vector were now being used to construct a Gaussian distribution. The resulting plot and data, subsequently contrasted against the metrics of a master Gaussian profile previously elaborated (with the aid of the Maximum Likelihood Theorem), as described in [2]. As CMOS ICs tended toward Ultra-Large Scale Integration (ULSI), not discarding devices which merely failed the I_{DDQ} test became a priority, per [22]. The introduction of Gaussian approaches to the electrical current signatures derived from I_{DDQ} test allowed for the consideration of devices with tolerable levels of passive defects only, according to [6]. For such devices, a scrutinized boolean test was conducted next. If the boolean test was passed, they became suitable for the market.

The usefulness of modeling the differences in electrical current signatures as Gaussian distributions had become a requirement to understand and organize strategies to maximize the efficiency of I_{DDQ} testing, per [16]. This entailed a demanding set of processing requirements imposed on the Automated Test Environment (ATE). The tester soon became the center of attention, as it needed to be adapted to these standards. The work load introduced by the test vectors was also being revised. Under [16], a fair degree of discussion is diverted to optimize and subsequently reduce the number of test vectors.

The work presented in [13] serves both as a historical landmark and the birth of a new trend. The introduction of Boundary Scan Chains (BSC) allowed to alleviate the burden imposed on designing the interconnections for a given IC at the substrate level. Specifically because, the designs had to be modified to allow for a given test vector to be placed on a given partition in order to execute the I_{DDQ} test, per [13]. The approach of "snaking-in" a test vector to the appropriate partition became essential to the point of expanding into a scan chain layer specifically designed to interact with the original IC design. The scan chain layer reached levels beyond the scope of a single vendor, as it was implemented differently across industry. At first, there was no agreeable standard by which to route the test vector to the desired partition inputs. An IEEE standard was soon developed, IEEE 1149.1 (known as JTAG) was created to take charge in regulating the language and architecture of boundary scan chains layers.

Early in 2000, a new approach was introduced to estimate I_{DDQ} threshold. Per [14], the approach became to construct regression models suitable enough to compute I_{DDQ} threshold. Regression models are purely computational, and therefore, inherit the advantage of being prone only to process variations and not variations due to manufacturing defects. The extent to which the degree of difference between the measured I_{DDQ} and the one computed by the regression models is acceptable, depends on allowing the process parameters in charge of predicting I_{DDQ} variations to work on the basis of contrasting the regression model's predicted I_{DDQ} against present and past real readings of I_{DDQ}, according to [14]. In [10] a more simple approach is proposed, based on the observation that defect-free devices have a single Gaussian distribution, as opposed to devices with active-defects which exhibit a double Gaussian distribution. However, the distinction between defect-free devices and those with passive-defects is not approached in

[10].

Currently, I_{DDQ} testing is said to become extinct by 2011, per [15]. I_{DDQ} testing faces the challenge of rightfully discerning between defective and non-defective devices, as I_{DDQ} values for both narrow due to continued reductions in feature size and subsequent increases in electrical current leakage. Never the less, Deep Sub-micron VLSI devices must continue to have a means for quality testing before being deployed to the field. I_{DDQ} testing has been a fundamental procedure in CMOS technology quality assurance for decades. The demise of I_{DDQ} testing has been predicted as many times in the past, as industry has rescued it.

The core of this research is related to I_{DDQ} testing in that both pursue an electrical current signature (or power signature). Deviation from I_{DDQ} testing is threefold, as the former work presented

1. is designed to capture power signatures in the field (outside the manufacturing test environment),
2. seeks to convey a reliability prognosis based on the overall deterministic behavior of a possibly larger system constituted by one or more electronic devices (from ipods to super computers), and
3. is intended to be unobtrusive and automatically used on a periodical basis.

The capture of a digitalized power signature also entails a different set of options in terms of processing. Discrete mathematical algorithms implemented by DSP devices can substantially aid in contrasting/analyzing power signatures. The following sections reference other forms of electrical current signature acquisition and evaluation.

Section 2.3: Motor Current Signature Analysis (MCSA)

MCSA is the process of evaluating the condition of induction motor drives (also known as squirrel cage (SC) electrical motors). The efficiency of electrical induction motor drives (IMDs) has made IMDs widely used in many scenarios where alternating electrical current potential is to be converted to mechanical work (from house-hold to industry applications). In the petro-chemistry industry, IMDs are used extensively to carry out petroleum refinement processes. The electrical rating for IMDs within such industry is in the two digit mega watt range. Every time IMDs fail (or have to be repaired) the losses are quantified per millions. The ideal solution to this problem came by introducing a method to monitor their health and accurately predict any conditions leading to their deterioration based on a real-time electrical current signature. As with I_{DDQ}, MCSA has settled into a standard practice. MCSA is not implemented for every IMD out on the field. MCSA is reserved for IMDs of industrial grade (i.e. three phase, 3300 v). Per [23] "Induction motor drives are the most widely used electrical drive system and typically consume 40 to 50 percent of an industrialized nation's total generating capacity."

The concept behind MCSA is based on the observation that any magnetic resonance (aside from the normal operation conditions) of an IMD will be reflected on the electrical current draw. Hence, from the real-time electrical current signature, a frequency spectrum can be reconstructed to differentiate between normal operation conditions and external factors causing any damages. In the case of two or three phase IMDs one phase sufficient as a monitoring point. In accordance with [23], the prime reason for this is because any of the disturbances will have an equal impact on any of the phases involved. The process is carried out by acquiring a scaled version of the real time electrical current signature via the Hall effect of any of the electrical phases involved. The signal is then amplified according to need. The associated frequencies are directly proportional to the electrical current fluxes through the phase. Therefore, a frequency spectrum is reconstructed to identify first and

second order effects. The following figure illustrates the concept.

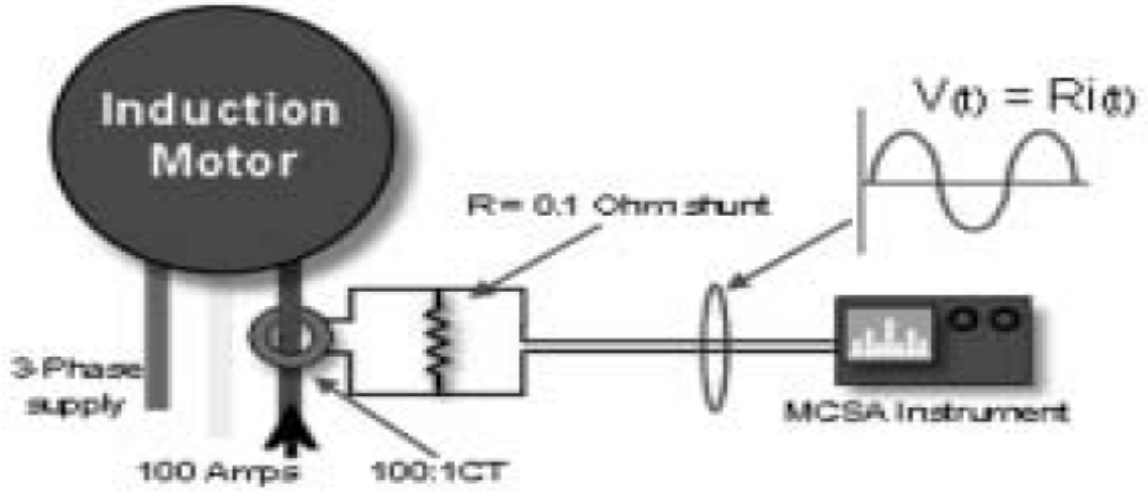


Figure 2.7: Typical MCSA setup.

First order effects, are natural occurring frequencies derived from the electrical current source and the load. Second order effects are due to broken rotor bars, abnormal air gap eccentricity, shorted turns, damaged bearings, damage due to reciprocal loads exceeding the design parameters, etc. In short, second order effects degrade performance of the IMD over time and eventually lead to the overall failure of the system. The severity is categorized according to the intensity of the respective frequency bands, per [24]. In order to evaluate and accurately predict the frequency ranges of first and second order effects, MCSA makes use of predetermined mathematical formulas and algorithms. These allow to predict the frequency bands associated with all of the previously mentioned possible damages inside the IMD. The following figures display the frequency spectra for an IMD in healthy conditions and one damaged by broken rotor bars (taken from [23]).

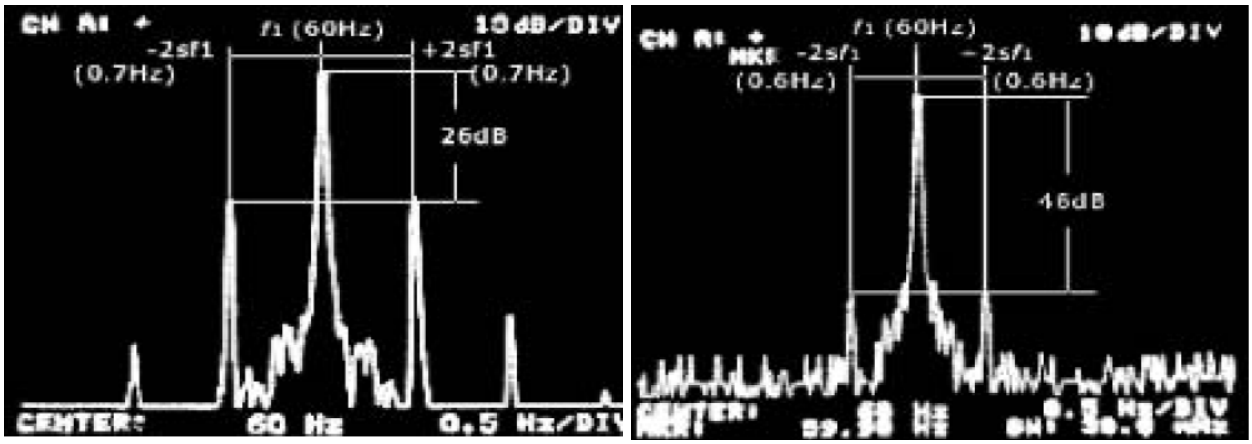


Figure 2.8: Frequency spectra of IMD in good conditions (right) and one in which the broken bars reflect damage beyond tolerance (left).

The two figures shown above depict the difference between the electrical current signature spectrum of an IMD in good conditions and one in which broken bars have caused severe damage. The intensity of the sidebands on the figure to the left indicate the damage done to the rotor of the drive. The prime difference between MCSA and the work presented in this thesis is scope of application. MCSA is utilized to monitor the health of industrial grade IMDs. In contrast, the platform developed on this thesis and the subsequent research aim to integrate the benefits of a reliable health monitor system to all electronic devices. MCSA owes its virtue to the frequency constituents derived from the electrical current consumed by the IMD. In this regard, Digital Power Signatures stand an equal opportunity to be processed and yield similar results. Another similarity, is the use of the Hall effect to perceive changes in electrical current consumption centered at the power supply lines. MCSA utilizes a transformer to scale down a voltage directly proportional to the power being consumed. The work presented on this thesis will make use of a dedicated Hall effect IC to sense fluctuations in the electrical current being drawn at a much lower scale. MCSA typically monitors electrical currents in the 300 to 500 amp range. The dedicated IC used in this thesis is rated for no more than 20 amps.

Section 2.4: Load Current Profile in Battery Life-Time Maximization

The battery pack (or batteries) have been around ever since the need to bring independence to a given electronic device. Such freedom is best represented by the electric circuit inside a simple flash light. The battery pack has evolved to output more efficiently through out time. Currently, the optimization of the chemical constituents conforming the battery pack demand an equal performance on the usage side. The need for a new venue to maximize the output of the chemical battery pack originates from the fact, that the chemical constituents do not efficiently convert the energy stored to electric potential at high electrical current loads, according to [18]. Therefore, the amount of charge a battery can deliver depends substantially on the rate at which it will be drawn out.

Improvements on the efficiency came when the batteries inside a given battery pack were switched on to the load according to an algorithm. This provided the batteries with a lapse of time to recuperate and deliver more efficiently the next time. The common practice became to alternate the batteries within a battery pack according to a sequentially discharge schedule. This approach remained effective through amid the life usage of the battery pack. Beyond this point, the batteries decayed to a single battery with a capacity equal to that of the whole pack. New methods had to be improvised. Per [19], "Since the capacity of a battery is reduced at high discharge electrical currents, it is possible to improve the battery system lifetime by distributing the load among the multiple batteries using a more sophisticated discharge schemes other than a simple sequential approach." Furthermore, according to [18] "In multi-battery systems, the load-dependent capacity of batteries has profound implications. First and foremost, the commonly accepted sequential discharge schedule is a very inefficient policy from a battery lifetime viewpoint, an improved battery discharge policy that selects the battery to be connected to the load based on the absorbed electrical current level is optimal." Hence, a scheme to select the right battery from the pack to be connect to the load depending on the profile of the instantaneous load

electrical current was brought forth in the form of a load-based battery switching algorithm.

The use of batteries of the same capacity along with load-based battery switching was quickly noticed to be an erroneous practice. Per [19], "Unfortunately, load-based battery switching degenerates to sequential discharge for multiple cells that respond similarly to the load (e.g., multiple equal batteries, similar batteries with different size) and for constant loads." According to [18], "This strategy is effective when batteries are highly asymmetric (e.g. one battery is very efficient at low electrical currents, while the other is much better at high electrical currents), and when the electrical current load has high variance." Load-based battery switching was proven to be more effective when the range of the electrical capacities of the individual batteries making up the battery pack was broader. This arose from the fact that a bigger range in battery capacity selection allowed for a greater range of coverage for the possible electrical current load. Therefore, sequential discharge scheduling became affixed to environments in which the electric current load was known to be predictable and almost non-varying. Whereas, load-based battery switching became adopted by many applications requiring the highest degree of efficiency performance where the electrical current load is completely unpredictable. Such applications range from solar cars to the battery packs used on satellites. This technique depicts a form of electrical current profiling for the purpose of decision making. In relation to the work presented on this thesis, decision making is a common purpose for both battery life-time maximization and the former work presented.

Chapter 3: Proposed Approach and Methodology

Chapter two presented some of the variants found in power signature acquisition and analysis. This chapter presents the process for obtaining the digital power signatures, the components involved in the process and their individual function. The process begins by sensing the amount of electrical current consumed by the device in question using a sensor. The analog output of the sensor is then adjusted to exemplify the variations in the amount of electrical current being consumed. This phase in the process will also condition the signal per the specifics of the analog to digital module inside a micro controller. In turn, the micro controller executes a series of routines to deliver either eight or ten bit words of data containing the digitized information about the power signature to a USB module. The USB module in conjunction with the micro controller safeguard the reliable transmission of the data to a computer. Once the data is transferred to the computer, a set of scripts are executed to process the raw signature into a MatLab readable file. It is possible to exert many mathematical algorithms on the digital power signature at this stage. The work presented on this thesis, however, will not perform any mathematical analysis on the digital power signatures as these operations are considered future work. The MatLab file obtained is then plotted for revision and visual analysis. Chapter four exposes the plot of the power signatures from the components inside a target computing environment in detail. The following figure illustrates the components involved and their interconnections at the most basic level. The process involves several pieces of hardware that will be elaborated on individually in the following section.

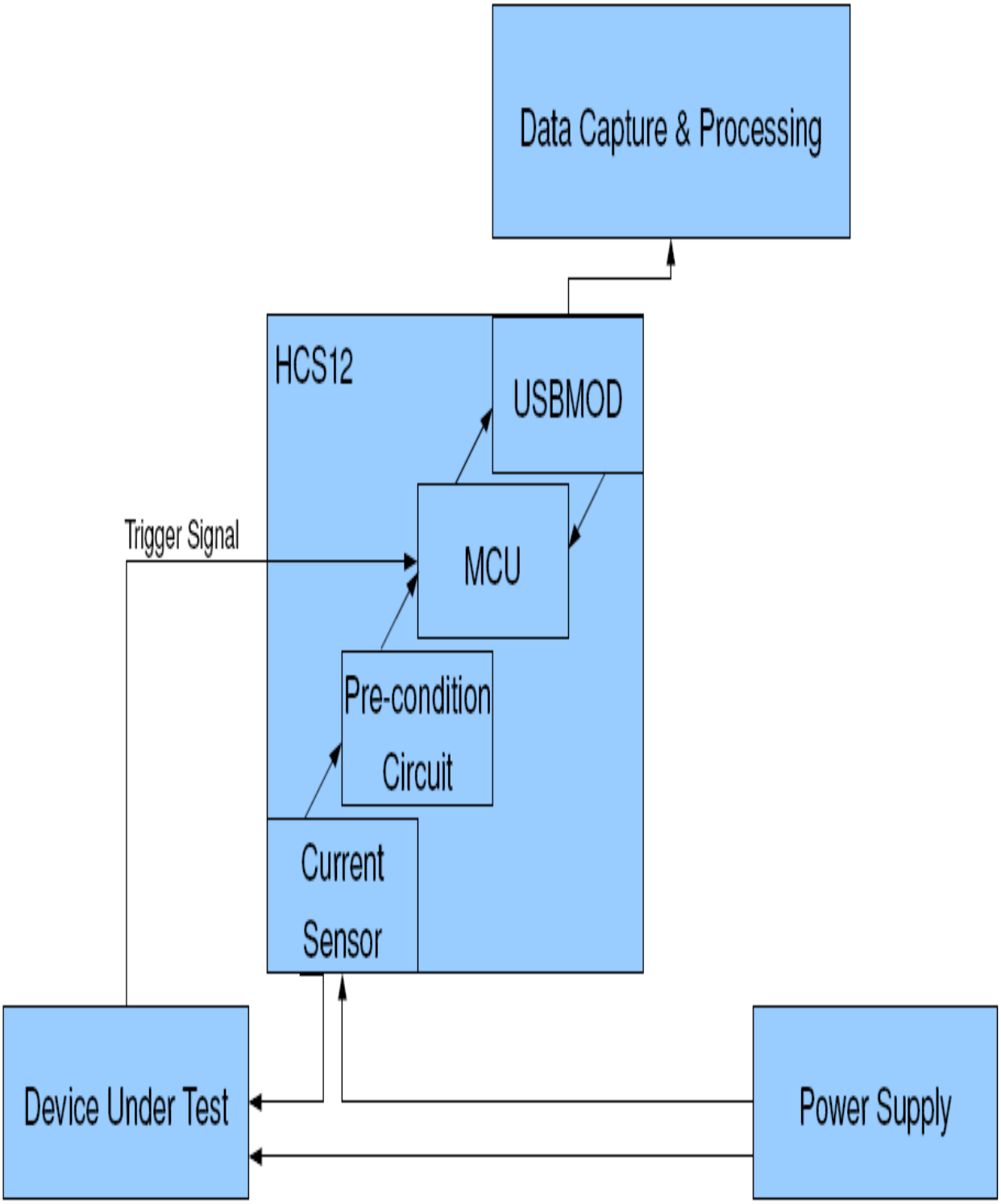


Figure 3.1: Functional diagram of the components involved in acquiring Digital Current Signatures (DCSs).

Section 3.1: Components Description

This section will describe the components and hardware used to pursue the capture of the digital current signatures (DCSs). After describing the components, the following section presents the normal sequence of events that take place in order to acquire a DCS. Hence, complementing both the overall methodology and approach. Per figure 3.1, the following hardware descriptions and functions are better understood by describing them from the bottom up.

Power Supply: The role of the power supply is to provide sufficient electrical current for the device in question to assert a reliable functional state. As long as the metrics pertaining to the electrical current sensor are not violated, the voltage and electrical current supplied can be either AC or DC. Although AC is not handled under this thesis. In the majority of the cases the original power supply is preserved and rather intercepted in series, as to divert the flow of electrical current through the sensor.

Device Under Test: The device under test can be any electronic device that does not draw an electrical current greater than the limit imposed by range covered by the electrical current sensor. The electrical current sensor specifics will be examined shortly (although the range can be incremented by swapping the sensor by one with a greater capacity range). The sensitivity of the sensor can be adjusted in order to accommodate a reasonable sensitivity level per mili-ampere.

Electrical Current Sensor: The electrical current sensor is a key element in acquiring the digital power signatures. The electrical current sensor used under this research is the Allegro ACS706ELC-05C. The ACS706 family of electrical current sensors is a bidirectional one. Therefore, allowing the handling of both DC and AC as previously mentioned. The functional block diagram and specific characteristics for this device are illustrated in the following figures.

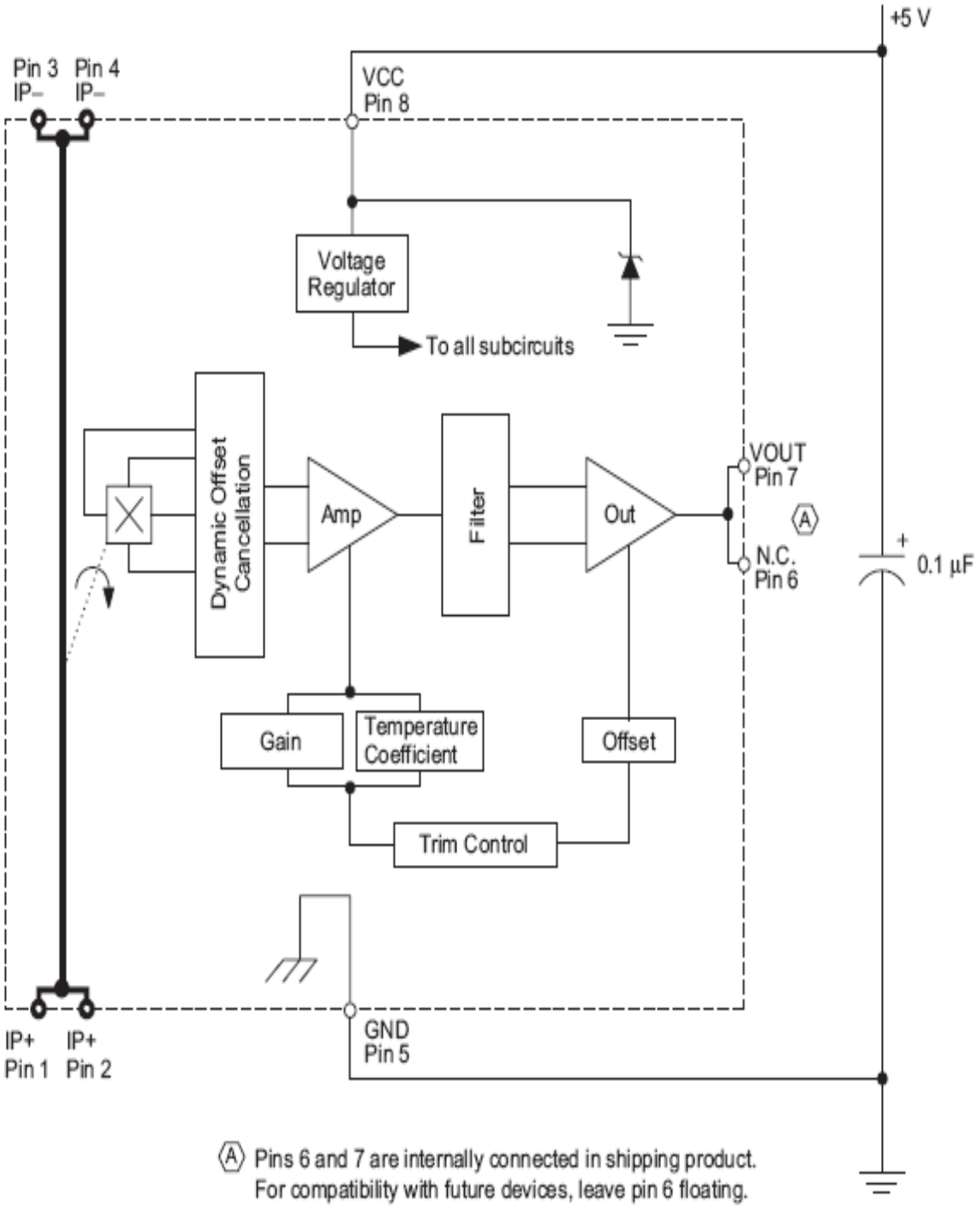


Figure 3.2: Functional diagram for Allegro ACS706ELC-05C electrical current sensor.

Table 3.1: Operating characteristics for Allegro ACS706ELC-05C electrical current sensor.

OPERATING CHARACTERISTICS

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS, over operating ambient temperature range unless otherwise specified						
Optimized Accuracy Range	I_P		–5	–	5	A
Linear Sensing Range	I_R		–15	–	15	A
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Supply Current	I_{CC}	V_{CC} = 5.0 V, output open	5	8	10	mA
Output Resistance	R_{OUT}	I_{OUT} = 1.2 mA	–	1	2	Ω
Output Capacitance Load	C_{LOAD}	VOUT to GND	–	–	10	nF
Output Resistive Load	R_{LOAD}	VOUT to GND	4.7	–	–	kΩ
Primary Conductor Resistance	$R_{PRIMARY}$	T_A = 25°C	–	1.5	–	mΩ
RMS Isolation Voltage	V_{ISORMS}	Pins 1-4 and 5-8; 60 Hz, 1 minute	1600	2500	–	V
DC Isolation Voltage	V_{ISODC}		–	5000	–	V
PERFORMANCE CHARACTERISTICS, over operating ambient temperature range, unless otherwise specified						
Propagation Time	t_{PROP}	I_P =±5 A, T_A = 25°C	–	3.15	–	µs
Response Time	$t_{RESPONSE}$	I_P =±5 A, T_A = 25°C	–	6	–	µs
Rise Time	t_r	I_P =±5 A, T_A = 25°C	–	7.45	–	µs
Frequency Bandwidth	f	–3 dB, T_A = 25°C; I_P is 10 A peak-to-peak; no external filter	–	50	–	kHz
Sensitivity	Sens	Over full range of I_P, I_P applied for 5 ms; T_A = 25°C	–	133	–	mV/A
		Over full range of I_P, I_P applied for 5 ms	124	–	142	mV/A
Noise	V_{NOISE}	Peak-to-peak, T_A = 25°C, no external filter	–	90	–	mV
		Root Mean Square, T_A = 25°C, no external filter	–	16	–	mV
Linearity	E_{LIN}	Over full range of I_P, I_P applied for 5 ms	–	±1	±4.7	%
Symmetry	E_{SYM}	Over full range of I_P, I_P applied for 5 ms	98	100	104.5	%
Zero Current Output Voltage	$V_{OUT(Q)}$	I_P = 0 A, T_A = 25°C	–	V_{CC}/2	–	V
Electrical Offset Voltage	V_{OE}	I_P = 0 A, T_A = 25°C	–15	–	15	mV
		I_P = 0 A	–65	–	65	mV
Magnetic Offset Error	I_{ERROM}	I_P = 0 A, after excursion of 5 A	–	±0.01	±0.05	A
Total Output Error[1]	E_{TOT}	I_P =±5 A, I_P applied for 5 ms;T_A = 25°C	–	±1.5	–	%
		I_P = ±5 A, I_P applied for 5 ms	–	–	±12.5	%

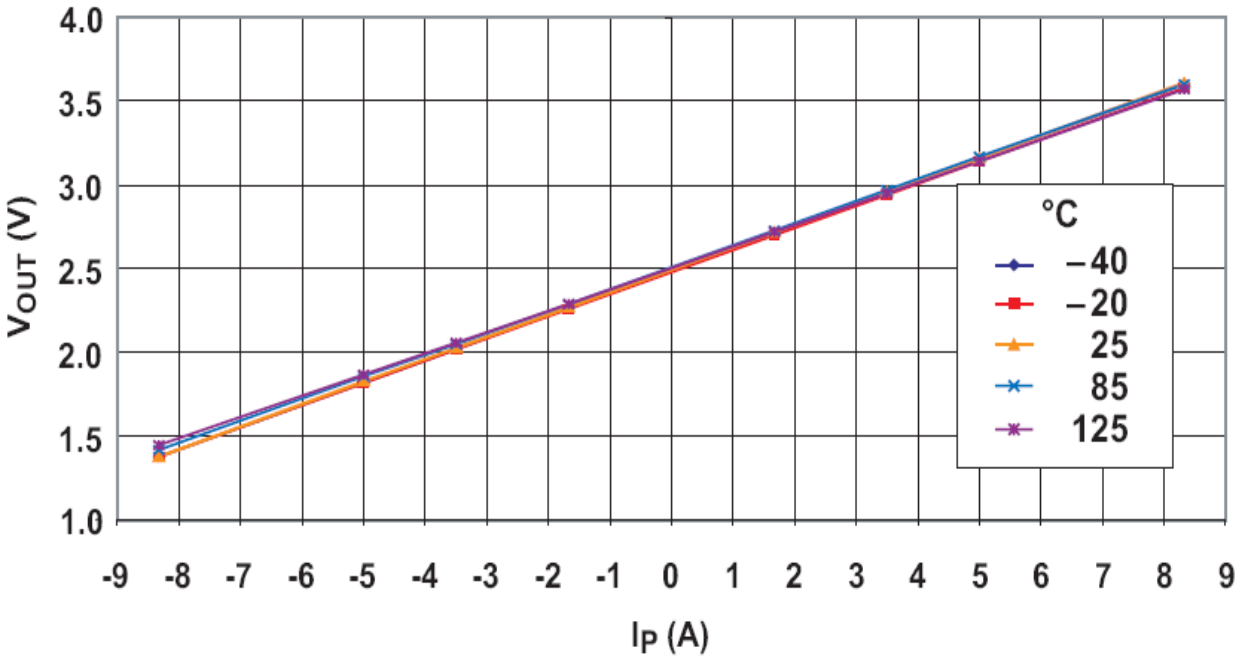


Figure 3.3: Output voltage versus primary electrical current for Allegro ACS706ELC-05C.

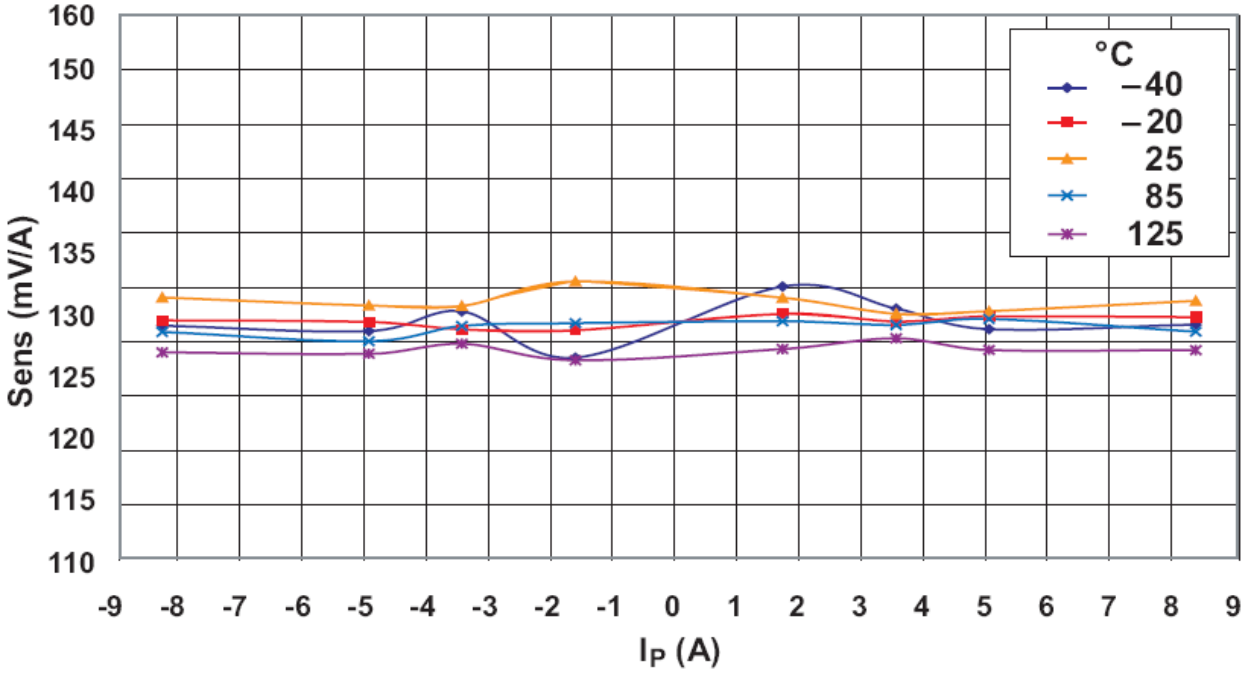


Figure 3.4: Sensitivity versus primary electrical current for Allegro ACS706ELC-05C.

Figure 3.3 illustrates the expected voltage output relative to the input electrical current. The power signature capture for devices consuming DC currents implies the need to adjust and scale the output of the sensor so that 0 volts reflect no electrical current flow, instead of 2.5 volts. The pre-condition circuit is in charge of this task, among others. Figure 3.4 illustrates the sensitivity magnitude of the sensor (133 mV per Ampere). The following picture shows the electrical current sensor and the physical connections used to intercept the power supply of any given electronic device.

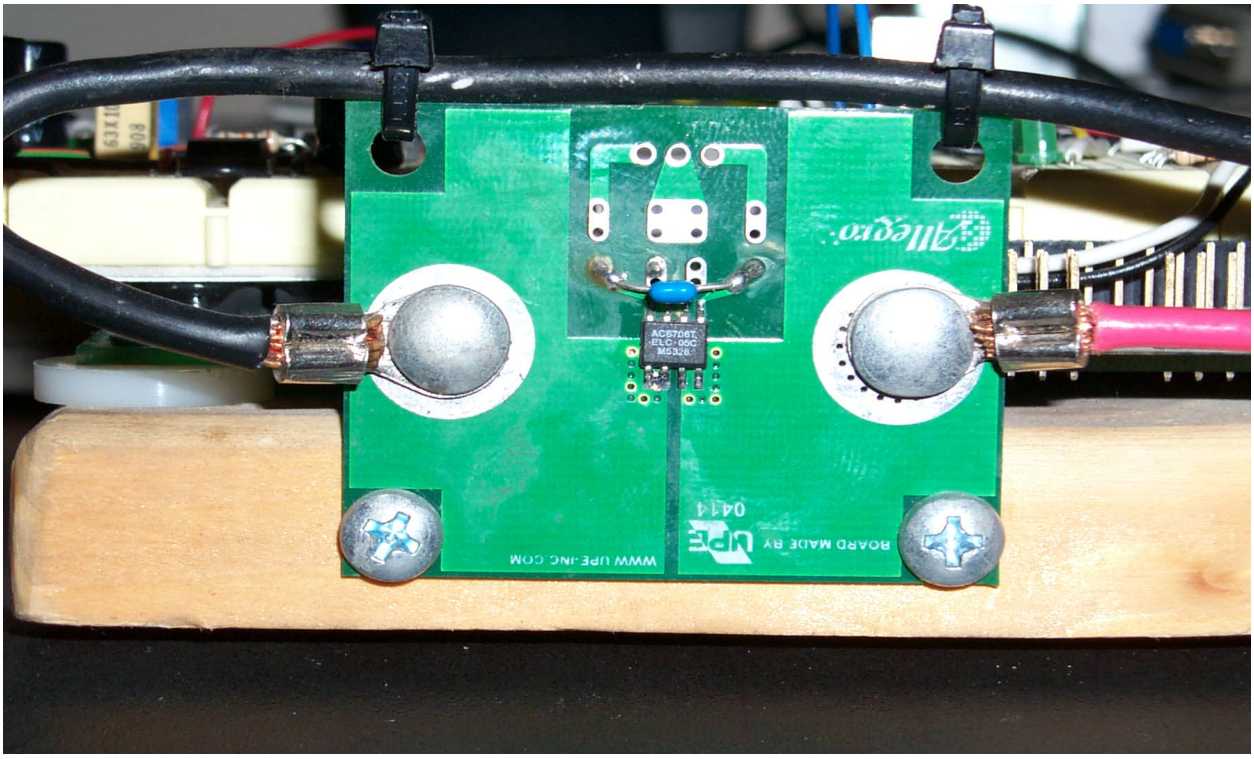


Figure 3.5: Wiring connections to the Allegro Hall effect electrical current sensor (ACS706ELC-05C).

Pre-Condition Circuit: According to the specifications for the electrical current sensor, the output is 2.5 volts when no electrical current is flowing through. The AD in contrast, covers a range from 0 to 5 volts. Therefore, the task of the pre-condition circuit is to transform the output of the electrical current sensor to an adequate one for the AD to convert from. From figure 3.3, the sensor's output for a current flowing only in one direction and fluctuating between 0 and 7.5 amps will be from 2.5 to 3.5 volts respectively. Per the AD specifics, the ideal voltage range corresponding to the power signature fluctuations must be from 0 to 5 volts. The first task of the pre-condition circuit is to add -2.5 volts to the sensor's output in order to lower the voltage to 0 volts. The second job entails scaling the resulting voltage by a factor of 5. Therefore, our target range for power signatures is restricted to electrical currents ranging from 0 to 7.5 amps (flowing into the target -one direction). It is possible to cover different ranges of DC current, as long as they don't

exceed the specifications of figure 3.3. In order to cover AC electrical currents, there is no need to add -2.5 volts to the sensor's output. In this case, the output of the sensor is directly connected to the non-inverting pin of the op-amp, and the gain is re-calibrated to cover the desired electrical current range. The following figures show both the schematic for the pre-condition circuit and the physical implementation, along with a description of the interconnections.

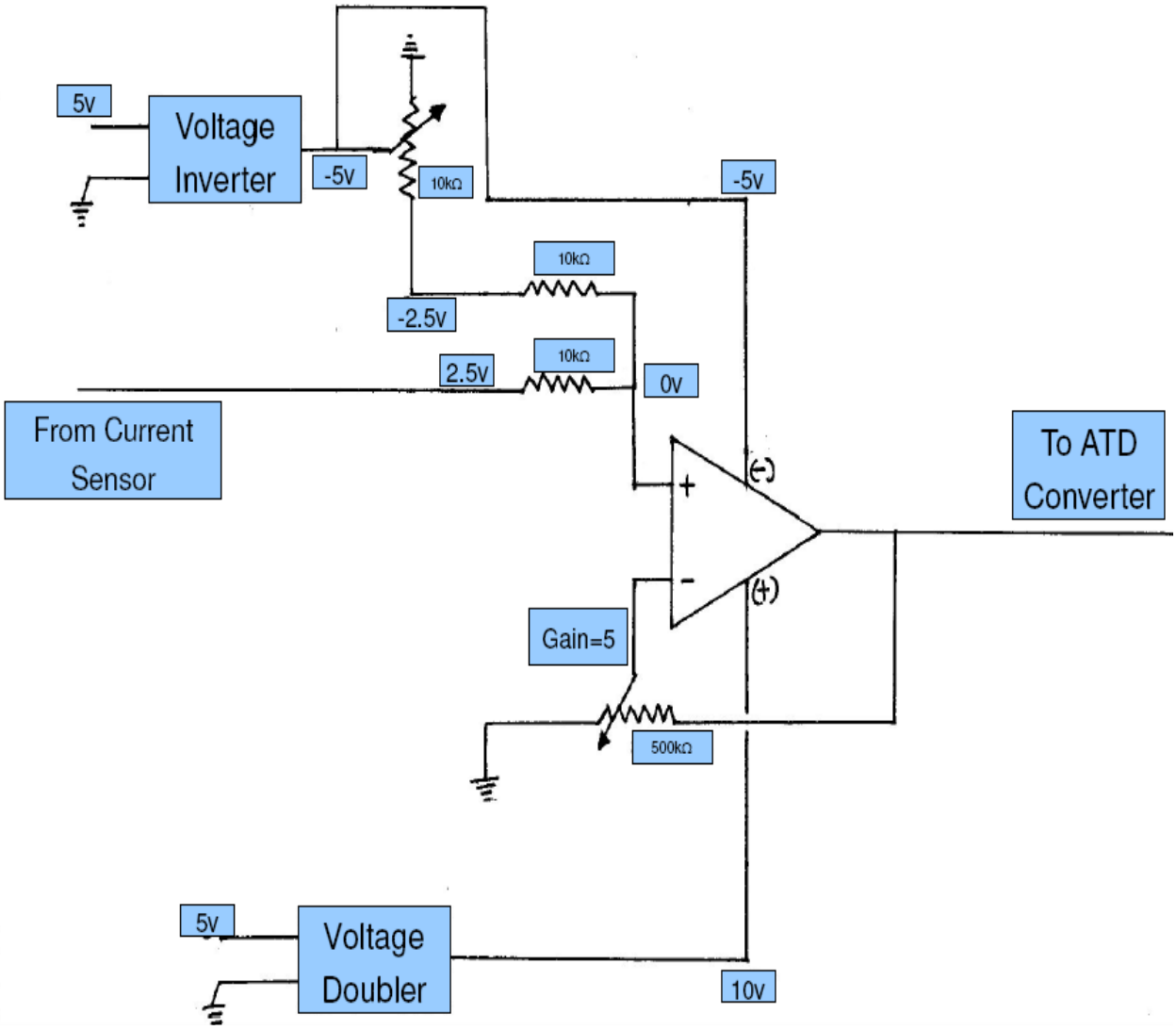


Figure 3.6: Schematic for the pre-condition circuit.

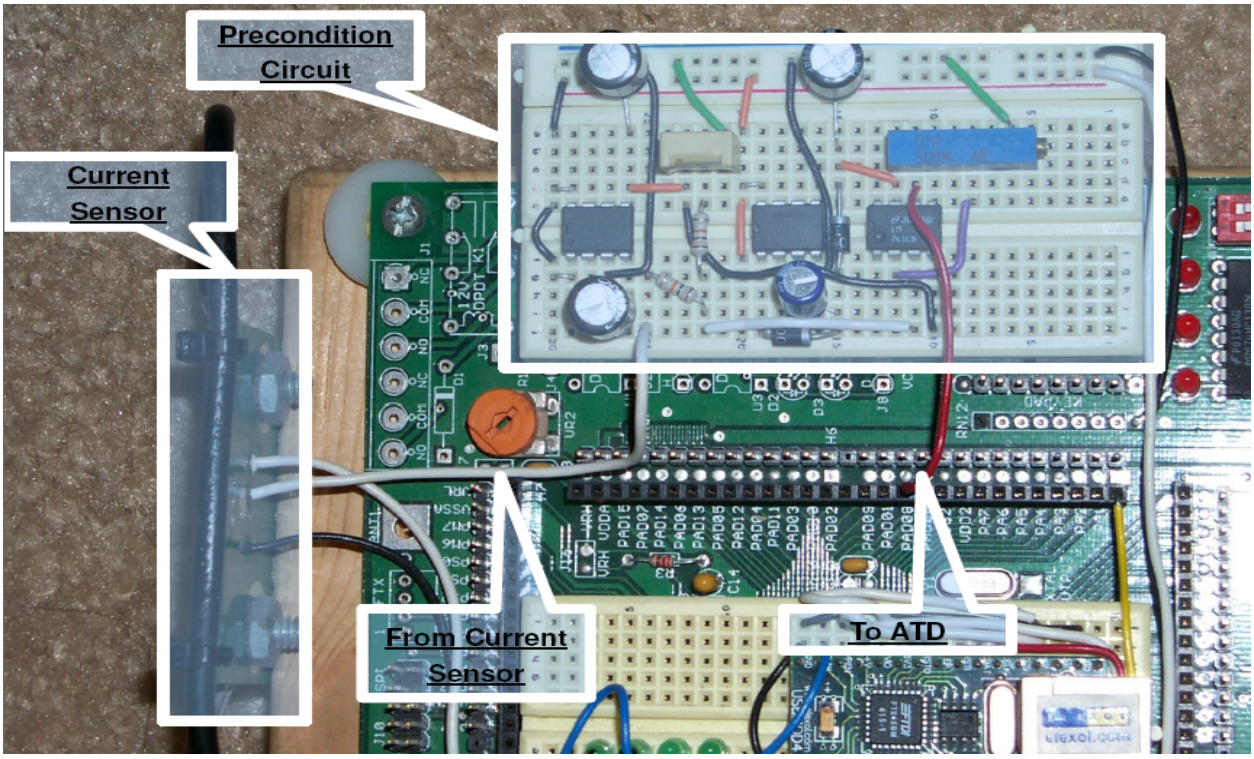


Figure 3.7: Pre-condition circuit and electrical current sensor interconnections.

Micro Controller Unit (MCU): This unit is in charge of regulating the overall function of the process. The HCS12 is a family of micro-controllers with many integrated peripherals. Among these is an enhanced Analog-to-Digital (AD) converter. The AD on the HCS12 is capable of delivering conversions in 8 and 10 bit resolutions. The HCS12 awaits the trigger signal (shown in figure 3.1) before starting the process. Once the trigger signal has been received, the HCS12 enables the AD and commences to digitalize the signal from the pre-condition circuit in accordance to predefined selections (8 or 10 bit resolution). While the HCS12 allows for the AD to digitize the signal coming in from the pre-condition circuit, the USBMOD4 is engaged to initiate the data transmission to the computer. Once the HCS12 starts to internally pass the data delivered by the AD to the USBMOD4, the programmed algorithm will keep guard of when to safely strobe data to the computer (see

appendix for HCS12 code). When the data buffer on the USBMOD4 is full, a signal to the HCS12 is sent. The HCS12 will then take the right steps in order to buffer the data without loosing any or its continuity until the next USB transfer. An 8k byte circular buffer is allocated inside of the HCS12 specifically for this purpose, until the USBMOD4 is ready to transmit more data. The sampling rate of the AD converter is 158khz for 8 bit resolutions and 130khz for 10 bit resolutions (see section 4.1). Therefore, if the Nyquist theorem is applied, this capture scheme is useful for current fluctuations bellow 79khz for 8 bit resolutions and 65khz for 10 bit resolutions respectively. The voltage range to be covered is from 0 to 5 volts, therefore, the voltage resolution is 19.531mv per point of reference (5v/2^(8)).

USBMOD4: The main function of this component is to encapsulate the data received by the HCS12 into a USB protocol and transmit it to the computer on the other end of the USB cable. The USBMOD4 is also responsible for alerting the HCS12 if the USB transmit buffer is full. Therefore, the HCS12 can take the necessary steeps to safeguard the data in the 8k circular buffer inside the HCS12. The USBMOD4 is a device that can be programmed to alternate between USB protocols. The work presented in this thesis uses the bulk transfer protocol rather than the isochronous one, as the former one implements measures to maintain the integrity of the data. Once a signature capture has been completely transferred to the computer (raw data), the HCS12 will reset the USBMOD4 to discard any residual data. The USBMOD4 intakes 8 bit data words at a time, and will strobe out data packets in which 8 bits comprise a piece of information. The following picture illustrates the USBMOD4 with the necessary connections to strobe data to the data capture and processing unit.

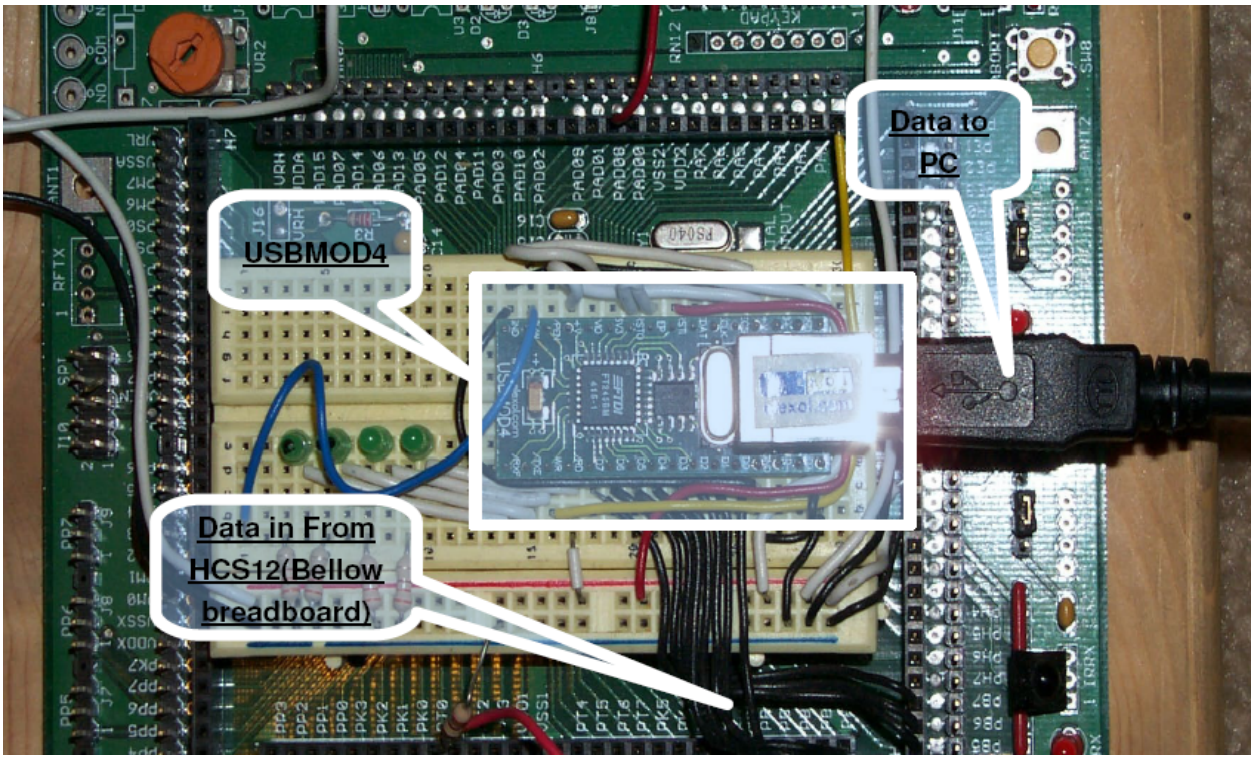


Figure 3.8: Connections to and from USBMOD4.

Data Capture & Processing: This last phase can be conducted differently according to need. In this particular case, the data will be capture to a text file. The file will then be post-processed with pearl scripts (see appendix for pearl scripts) to transform the contents to a decimal form separated by a comma and a new line. The result is a file which either Mat-Lab (or its open source counter part, scilab) can read and plot. The output of the scripts is a running process on behalf of scilab, which in turn depicts a plot of the captured data against the number of samples taken.

Section 3.2: Acquisition of a Digital Current Signature

This section describes the process for acquiring a digital current signature by making use of the hardware described in section 3.1. In accordance with figure 3.1, the current sensor must be placed in series between the power supply of the device in question and the target device (chapter four presents an example). The polarity of the current sensor terminals are denoted by two 10 gauge conductors to reference positive (red) and negative (black) respectively. Once the current sensor has been connected, the next step is to connect the trigger signal cable from the HCS12 board to an element or piece of hardware that can be used to simultaneously start the target device and trigger the HCS12. Next, power the HCS12 board and select the desired resolution, the type of current to be measured (AC-DC), and the time duration of the capture in seconds through PORTH's dip switches (16 sec is the max). The HCS12 has been programmed to display basic instructions on the LCD for reference. Once this is done, a USB cable must be connected from the USBMOD4 on the HCS12 board to the computer where the data is to be captured. The USB port to which the USBMOD4 maps to must be known in order to start a “cat” process to a file referencing such port (Linux). Lastly, trigger the capture.

Section 3.3: 8 Versus 10 bit Signatures

An effective approach to the acquisition of data with a format suitable for an efficient storage unit size is crucial to the successful implementation of the on board chip solution envisioned by this research (see future work). The format of the data has been chosen based on a minimum resolution, effective enough to convey all significant details relating to the signature, and yet be swift to both process and store/recall. For these reasons, the 8 bit resolution signature format has been chosen. 8 bit resolution signatures are more effective for maximizing the use of storage within a practical implementation of this scheme into a dedicated IC. The 10 bit resolution signature format offers a greater range of voltage granularity through a higher resolution. However, the amount of data points collected through the signature acquisition process is cut by half on average when using this format ,because of the delay involved in processing two bytes instead of one. For 10 bit resolution formats, two 8 bit words will comprise a piece of data in the following format: dddddddd ddxxxxxx (the last 6 bits are wasted as don't-cares). Hence, 10 bit resolution signatures are not attractive for practical applications where maximizing data store and execution speed are crucial. In other words, the difference in resolution is compensated by the gain in sampling speed, but most importantly by the efficiency in storage and processing inherited by the 8 bit resolution format. From an analytical perspective, 8 bit resolution signatures contain almost twice as much information in the form of number of samples as their 10 bit counterparts, while both describe an identical wave. From a practical perspective, 8 bit signatures are easier to accommodate for and process, due to the fact that a single data point is stored on a single storage unit (byte). Therefore, 8 bit resolution power signatures are more alluring for the future of this research. The following two figures illustrate an 8 and 10 bit resolution square wave respectively, zoomed in to expose the differences previously mentioned under two cycles.

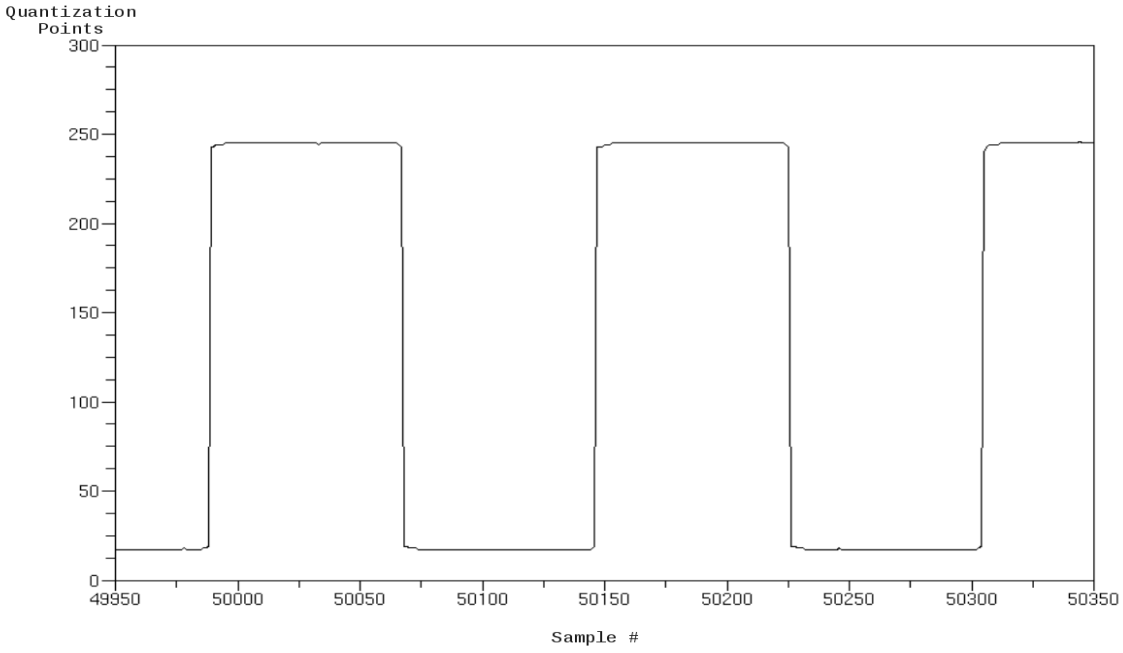


Figure 3.9: 1kh square wave with 8 bit resolution.

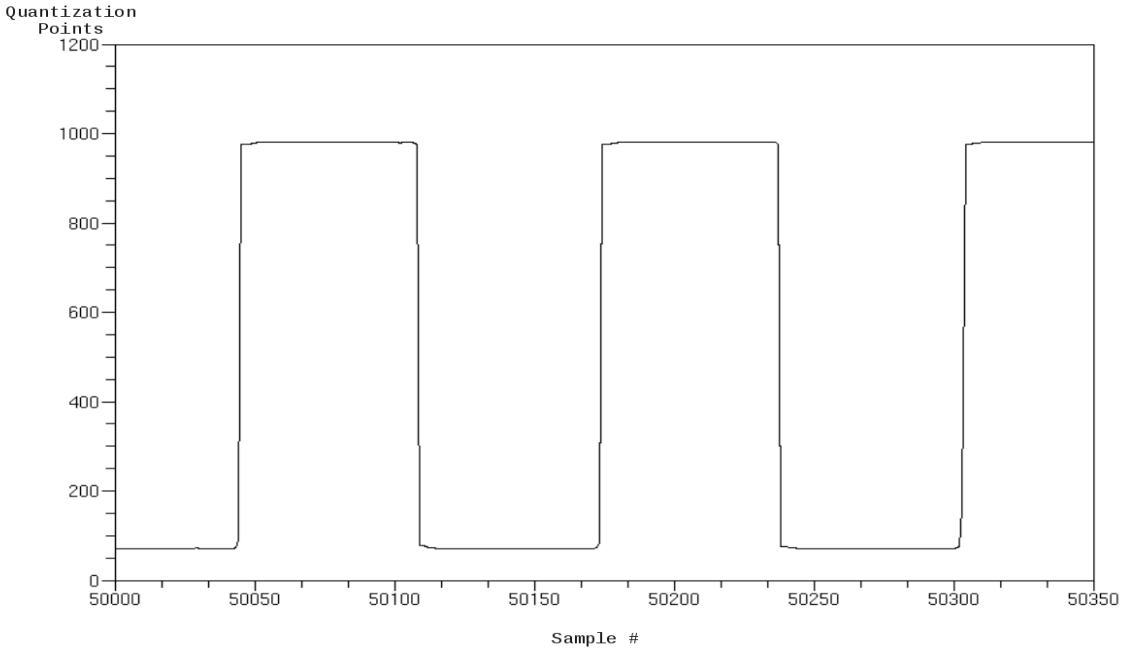


Figure 3.10: 1kh square wave with 10 bit resolution.

Chapter 4: Signature Acquisition For Dell Optiplex G150 Boot Up

The previous chapter introduced an in-depth description of the components and process for acquiring a digital power signature. The next section on this chapter presents the capture of a square wave for the purpose of benchmarking the sampling rate of the hardware platform elaborated to capture the digital current signatures. Section 4.1 will also introduce the metrics to be considered in terms of electrical current and voltage when interpreting the digital power signature plots. As a means to solidify the work presented on this thesis, section 4.2 will present the capture of the digital power signatures for a small computing environment. The target has been selected to be a Dell Optiplex G150 computer. An introduction to the physical setup process is given first. Subsequently, the various signature plots for the internal components are presented. The first component to be analyzed is the CPU. Sets of three signature plots are presented for both 8 and 10 bit signatures for the first component only (CPU). The preceding components are presented with a set of three signature plots with a resolution of 8 bits. After the CPU is presented, the motherboard, fan, and hard drive drive-motor are presented. Along with each set of signature plots, a complementary interpretation for their meaning is also given. The purpose is to illustrate the meaning of the digital power signature plots presented.

Section 4.1: Signature Acquisition Parameters

Chapter three referenced the sampling rate of the AD converter inside the HCS12. This section rightfully justifies the statement previously made. The specifications for the AD converter allow for a sampling rate from 500hz to 176khz (see HCS12 ATD manual). However, to determine the practical sampling rate versus the theoretical one derived from the values programmed into the HCS12, a more realistic and compelling approach must be considered. The approach will be to input a 1khz square wave with a resolution of 8 bits to the AD converter and initiate the capture process to obtain a plot. From the plot, a tally to determine how many samples per cycle were obtained is subsequently computed. The number of samples per cycle have a direct relation with the sample frequency at which the AD is performing. For example, if a 1khz square wave is input to the AD and the plot depicts 8 points of reference per cycle (samples), the AD's sampling rate is eight times faster than the input wave frequency (8khz).

The following images depict the process. Subsequent zooms are presented until the left and right edges of one cycle reveal sufficient closeup to expose the amount of samples per cycle obtained. The plots contain axis labels to further assist the comprehension of the data presented. The range of quantization points possible is presented to the left along with a multiple (in green) by which the true value of a given quantization point can be obtained in volts. The number of samples and a time scale (in red) are presented at the bottom of each plot.

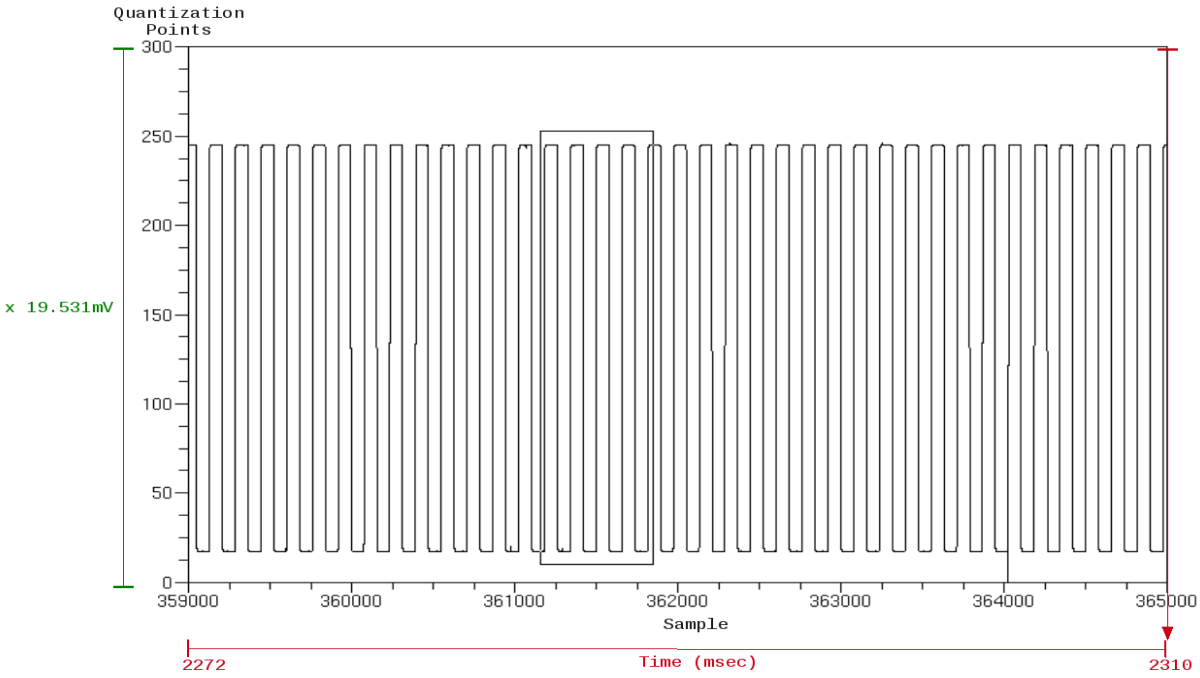


Figure 4.1: Zoom in on a 1khz square.

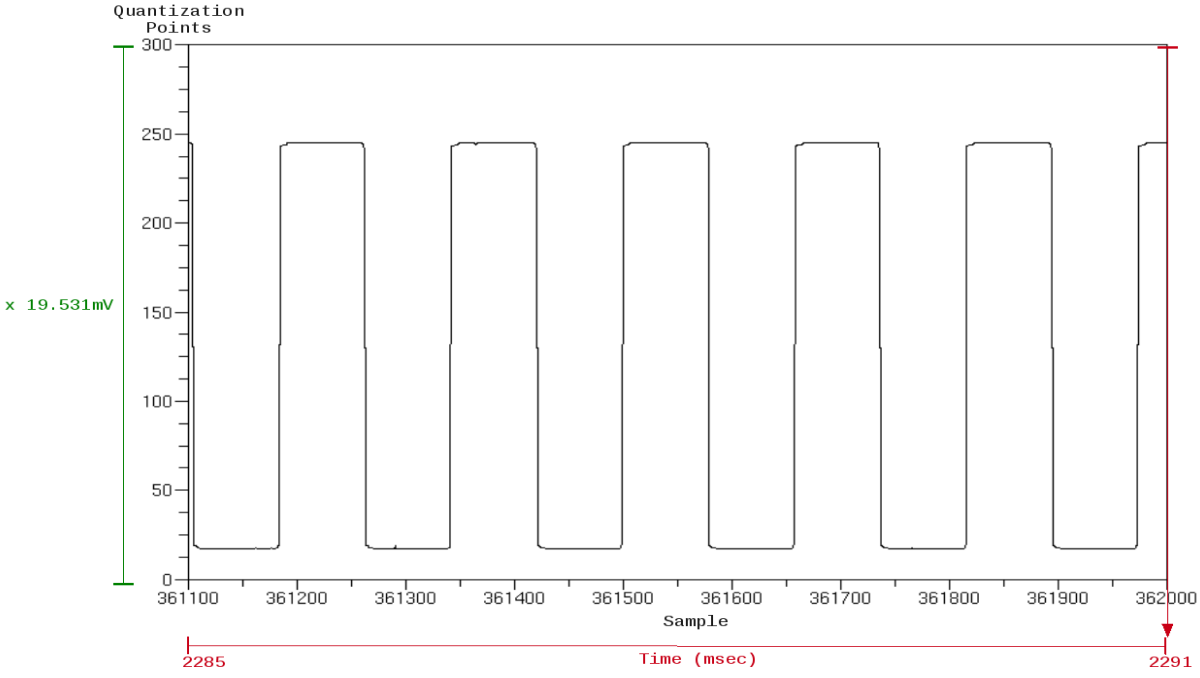


Figure 4.2: Zoom in on Figure 4.1.

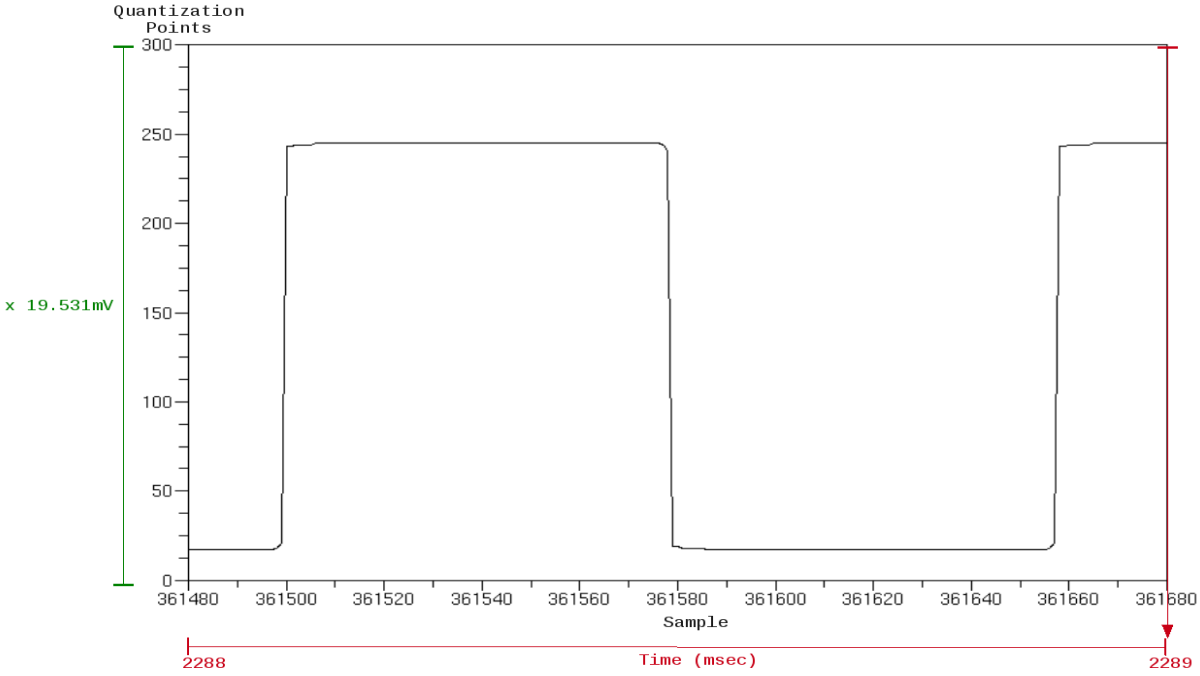


Figure 4.3: Zoom in on Figure 4.2.

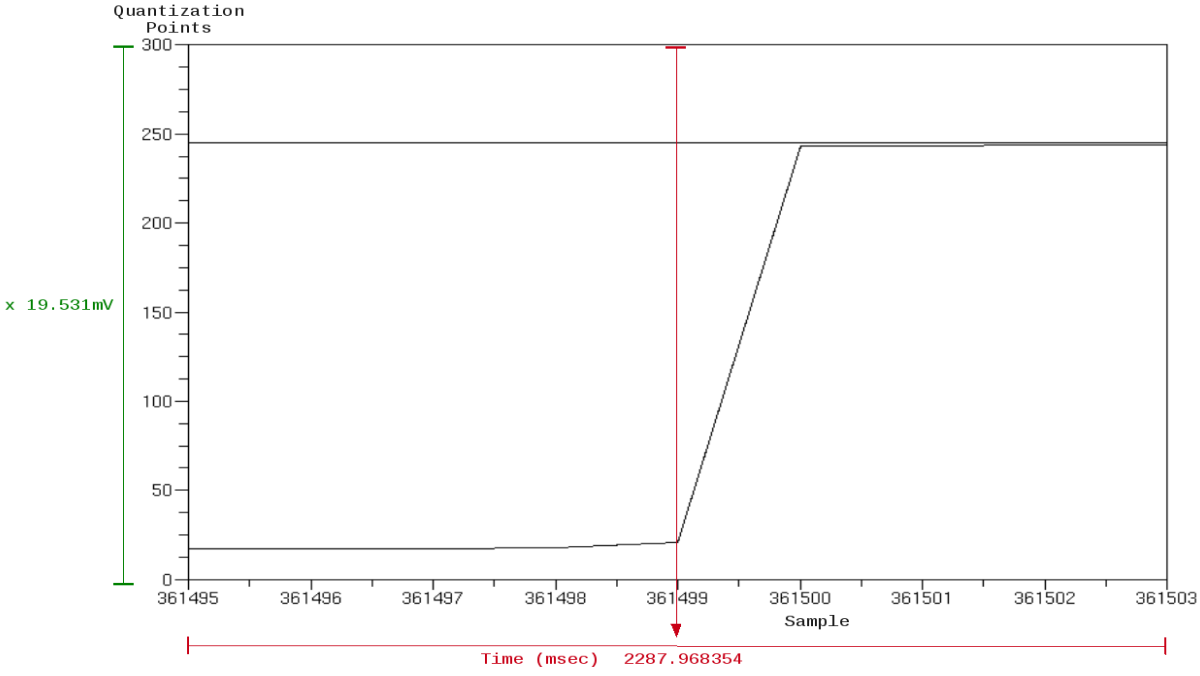


Figure 4.4: Left Zoom in on Figure 4.3.

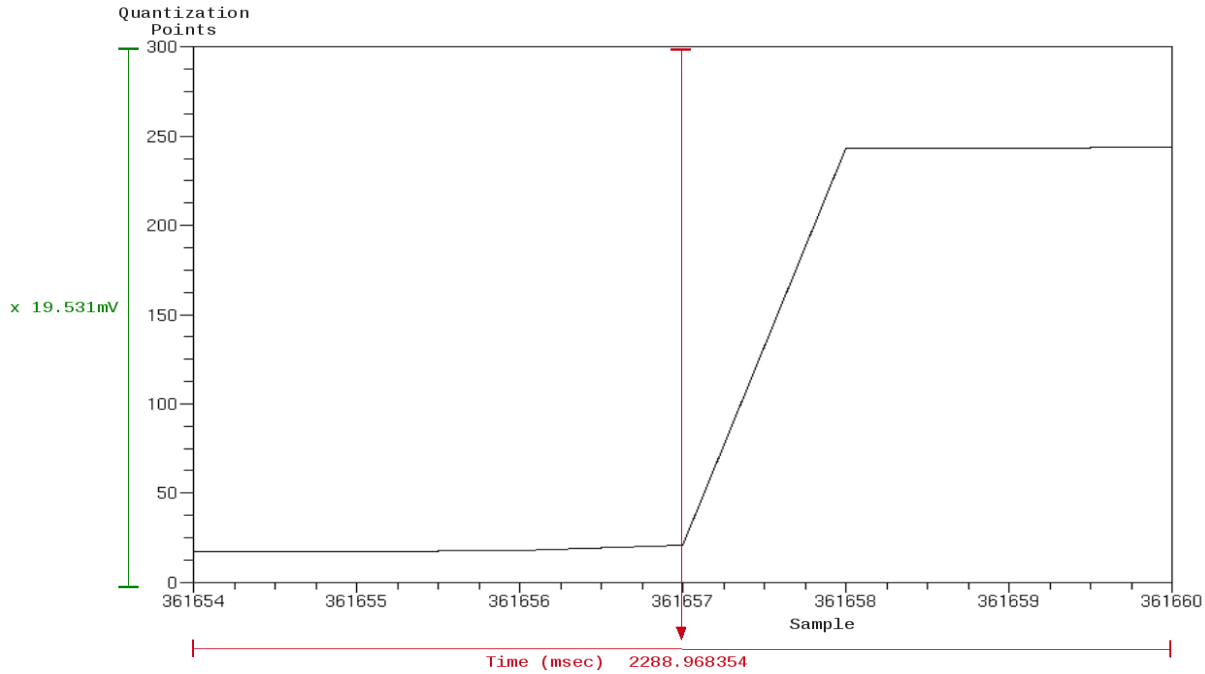


Figure 4.5: Right Zoom in on Figure 4.3.

From figure 4.4, the left point of reference is equal to 361499. From figure 4.5, the right point of reference is equal to 361657. The number of samples per cycle for a 1khz square wave is 158 (361657 – 361499). Therefore, the true sampling rate for the AD converter with a resolution of 8 bits is 158khz. Similar analysis reveals a sampling rate of 130khz for 10 bit resolutions.

Section 4.2: Signature of DELL Optiplex g150 Internal Hardware

In this section, the digital current signatures of the elements inside a small computing environment are presented. The target is a Dell Optiplex g150 computer. As mentioned in section 3.1, the power supply for this target will be preserved and the scope of intervention lies between the power transmission lines, in this particular case, the ones connecting the power supply to the motherboard and hard drive. A point not be omitted for the case being, is the deployment of not only one but several power lines on behalf of the power supply to power all necessary components. The Dell Optiplex g150 power supply outputs three different voltages in oder to fulfill the CPU, mother board, and hard drive requirements via an ATX power connector. +3.3 vdc is used to power mainly the CPU, +5.0 vdc is used to power all TTL and solid state components confined to the motherboard and hard drive, and +12.0 vdc is used to power the drive-motors found inside the hard drive and ventilation fans. Hence, we can obtain a Digital Power Signature (DPS) for the CPU, motherboard, ventilation fan, and drive-motors inside the hard drive. The following figures illustrate the set-up of the hardware described in chapter three to capture the digital current signatures of this small computing environment.

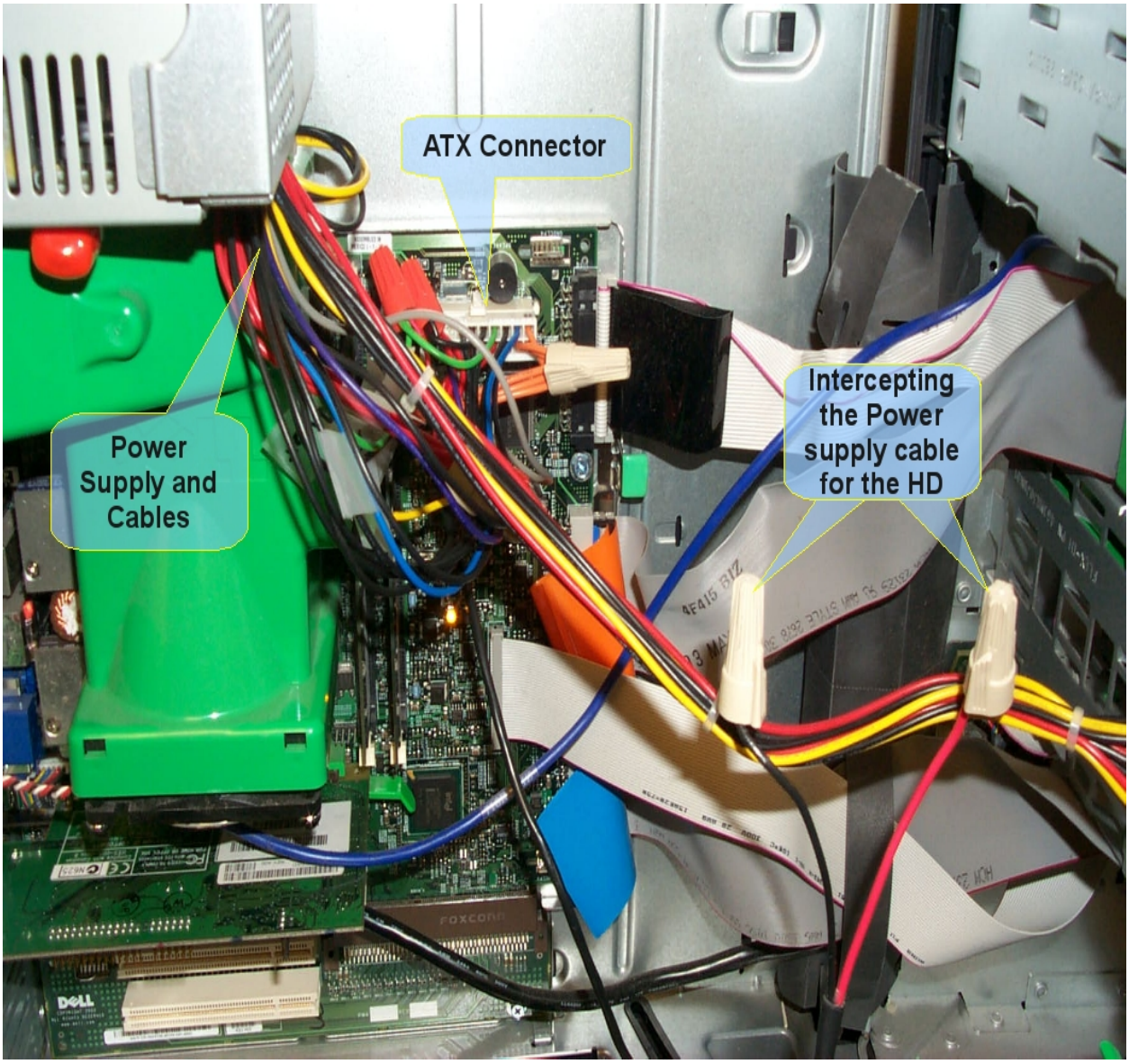


Figure 4.6: Signature acquisition set-up (1 of 3)

Figure 4.6 depicts the set-up on the target side. In this particular example the digital current signature of the hard drive is being obtained. Hence, the intersection of the cables for such is noted. The signature for the motherboard and CPU was obtained in a similar manner (by intercepting the cables from the ATX connector shown above). The following figure further illustrates the set up.

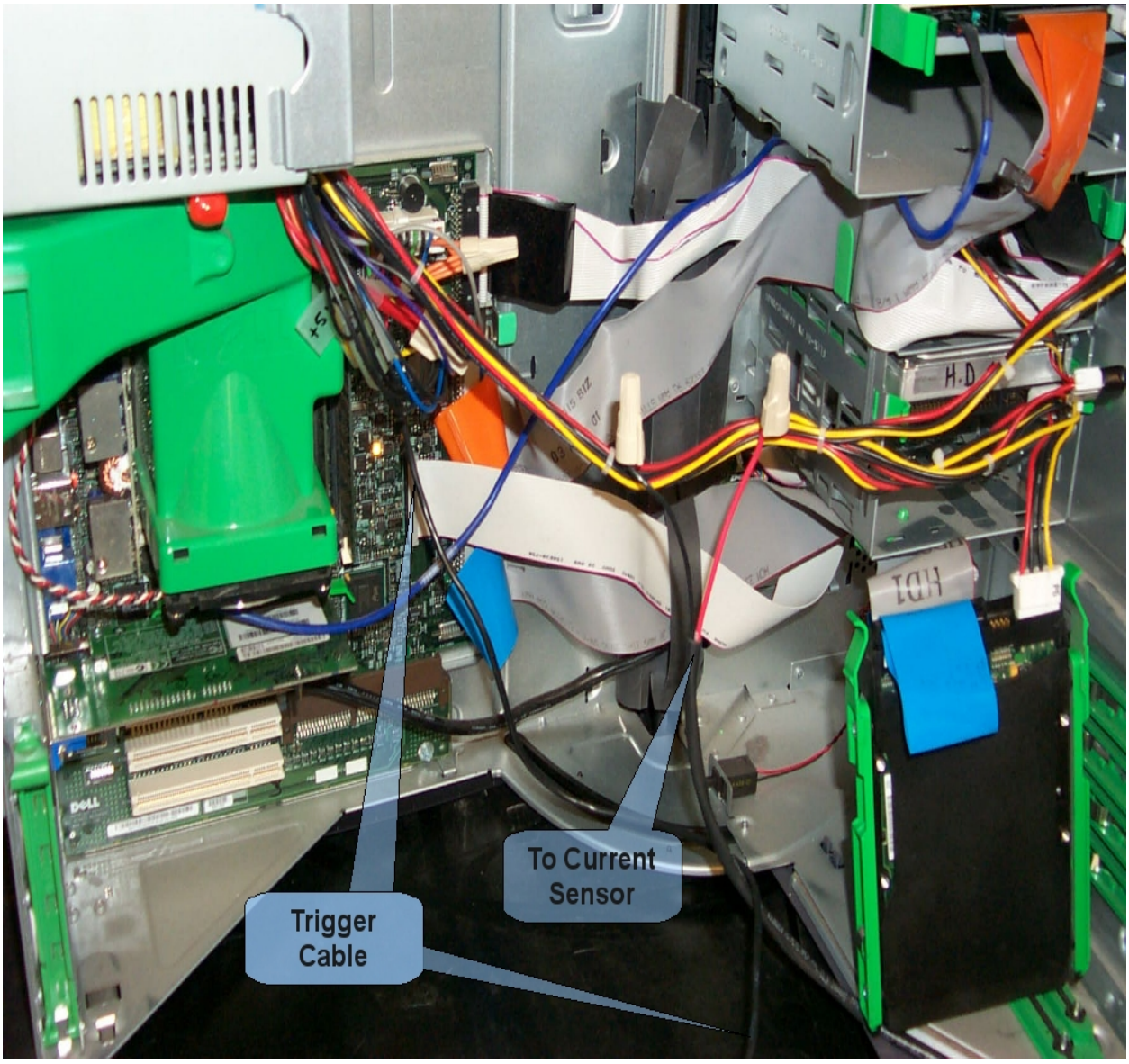


Figure 4.7: Signature acquisition set-up (2 of 3)

Figure 4.7 illustrates the trigger cable used to signal the HCS12 when to initiate the sequence of conversions. The electrical current sensor cable ends on the board, where the sensor is located. The following figure will allow to finally idealize the set up process.

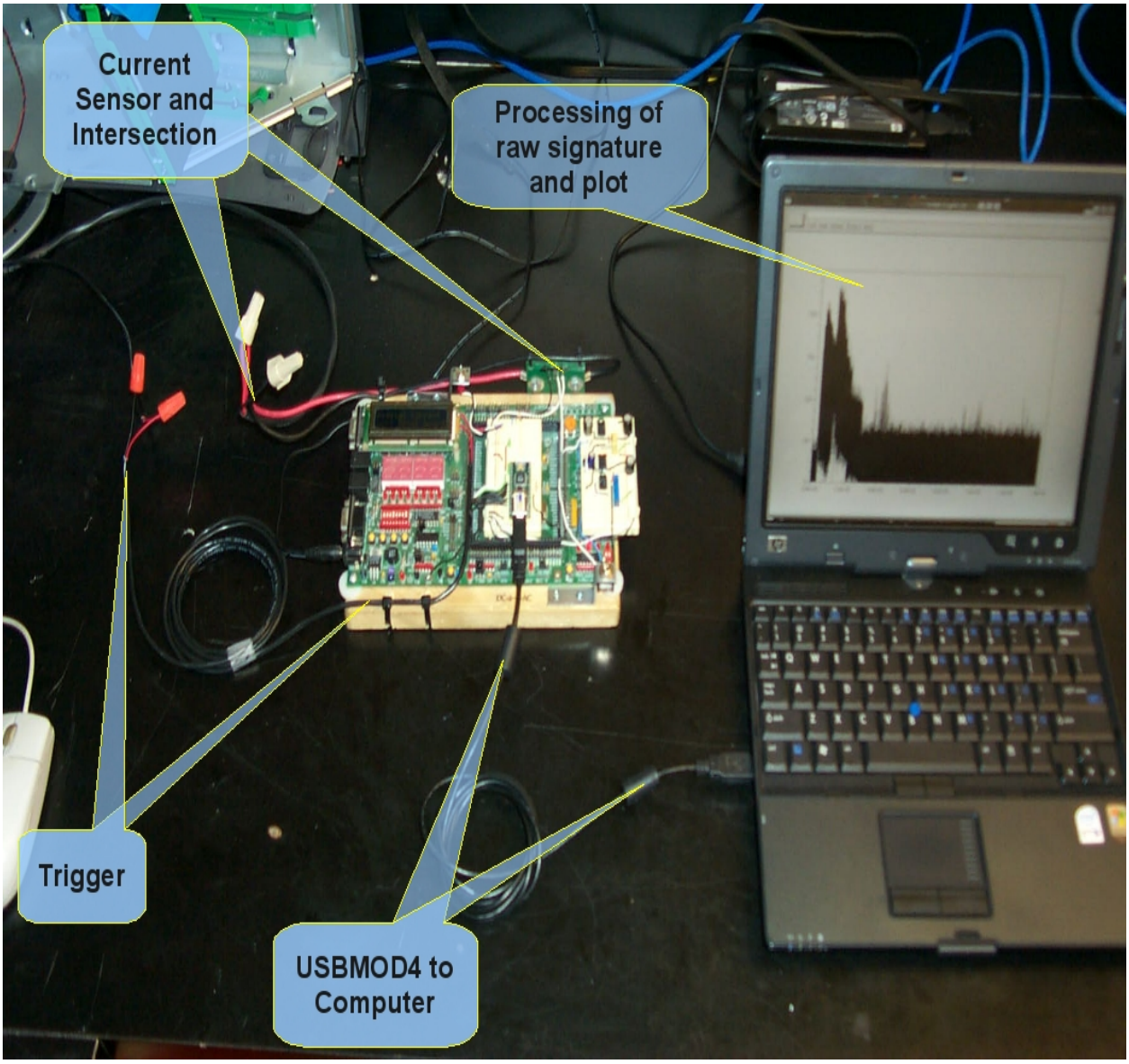


Figure 4.8: Signature acquisition set-up (3 of 3)

This figure illustrates the final stage in the setup. The current sensor on the board is reached via the cables shown in figure 4.6. The USB cable connecting the USBMOD4 and the computer is also depicted along with the trigger signal. The laptop shown is the data processing element under this example, per figure 3.1. As explained in section 3.1, the HCS12 signals the laptop when the sampling cycle has ended via the USBMOD4. Thereafter,

the laptop executes a set of scripts in order to process the raw signature in to a format suitable to be plotted. Digital processing of 8 bit power signatures has already been explored in [25], by making use of the hardware platform developed in this thesis to acquire the signatures. In [25], a statistical approach to determine the correlation between digital power signatures is investigated. [25] utilizes modified boot-up routines on the same computing environment set-up as the one presented here to derive statistical data from boot-up to boot-up, across several alike computing environments in good condition (no damage). The work in [25] preludes a fine correlation on the individual determinism found on each of the systems from boot-up to boot-up. The correlation across the systems in general, however, is found to be inferior. The conclusion in [25] exemplifies the uniqueness of the power signature from a given system.

The remainder of this section exposes the Digital Power Signatures (DPSs) obtained from the devices previously mentioned 16 seconds into the boot up process. These are consider as the startup DPSs for the individual components making up the computing environment. First, the DPSs of the CPU on the motherboard are exposed. The CPU is an Intel Pentium II processor (933 Mhz) A set of three signatures for both 8 and 10 bit resolutions are presented for the purpose of contrasting any differences. From that point onward only 8 bit resolutions are presented. After the CPU signatures are presented, the signatures for the motherboard, fan, and hard drive-motor drive are presented and subsequently interpreted as best as possible. The plots contain axis labels to assist the comprehension of the data presented. The range of quantization points possible is presented to the left along with a multiple (in green) by which the true value of a given quantization point can be obtained in amps. The number of samples and a time scale (in red) are presented at the bottom of each plot.

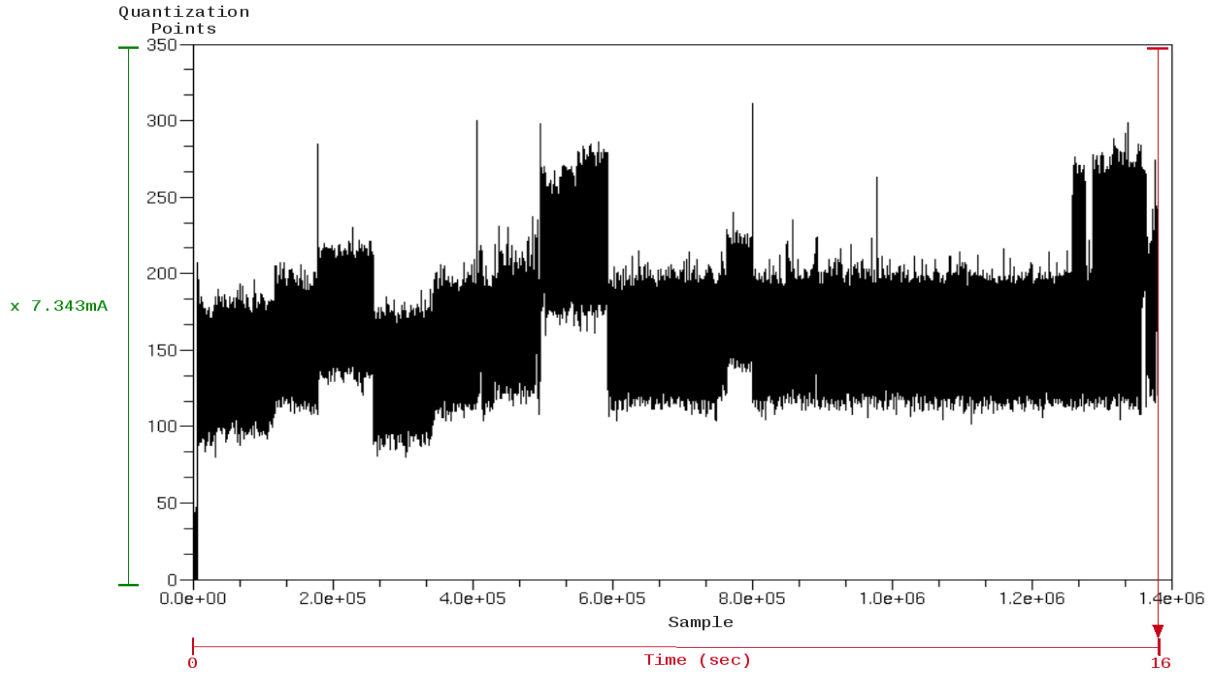


Figure 4.9: CPU Digital Current Signature from Dell Optiplex G150 (10 bit resolution 1 of 3).

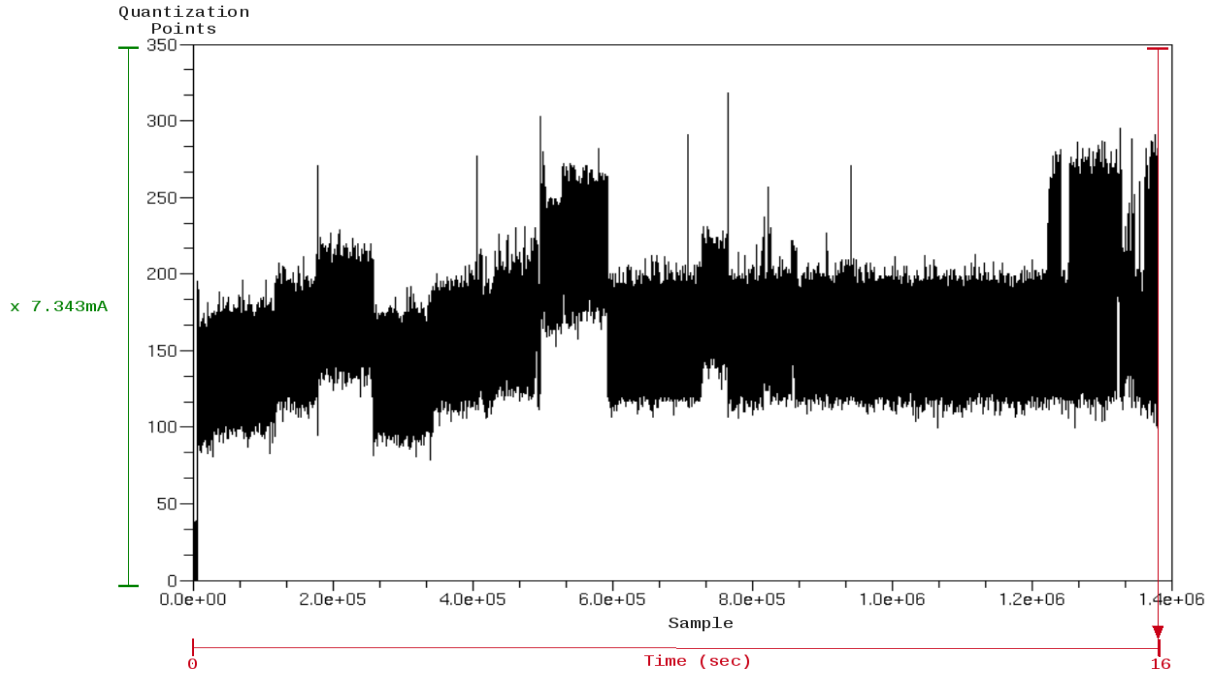


Figure 4.10: CPU Digital Current Signature from Dell Optiplex G150 (10 bit resolution 2 of 3).

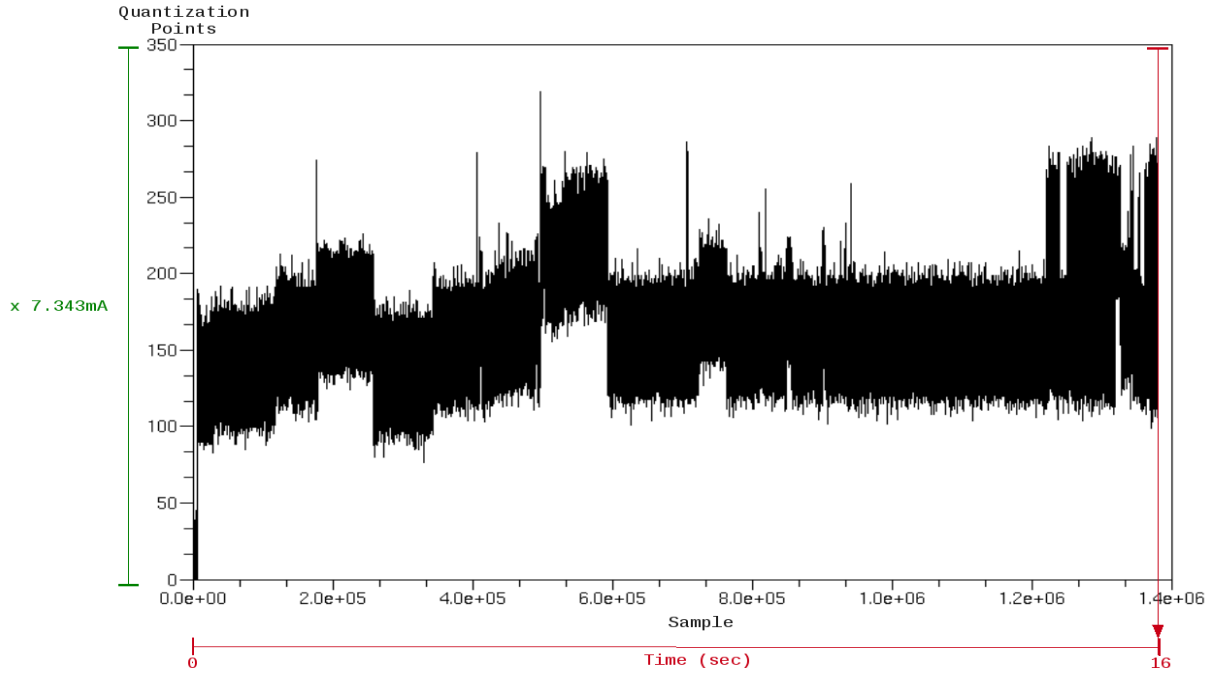


Figure 4.11: CPU Digital Current Signature from Dell Optiplex G150 (10 bit resolution 3 of 3).

The three previous digital power signatures depict the electrical current profile for the CPU 16 seconds into the boot up process. The lapses of time through which the CPU undergoes what can probably be interpreted as code execution rate differences can be appreciated. Code execution depends on cache hits or misses, disk drive data block transfers and subsequent line transfers to and from RAM, data multiplexing between the motherboard bridges, even the manner by which the kernel is coded to "stand up" and give rise to the Operating System (OS). Therefore, an exact interpretation for every single point of data is a burdensome task. However, as intrinsic and complex as the pattern may be, an over all pattern of consumption becomes evident (independent of the many factors affecting the execution of the code). This is primarily due to the determinism inherent in the boot up process. Another remark, is the average amount of data collected through the 16 seconds. On average, each 10 bit resolution digital signature contains 1.378 million reference points (samples) per 16 seconds. The following figures present the 8 bit resolution DPSs for the

same CPU under the exact same operating parameters.

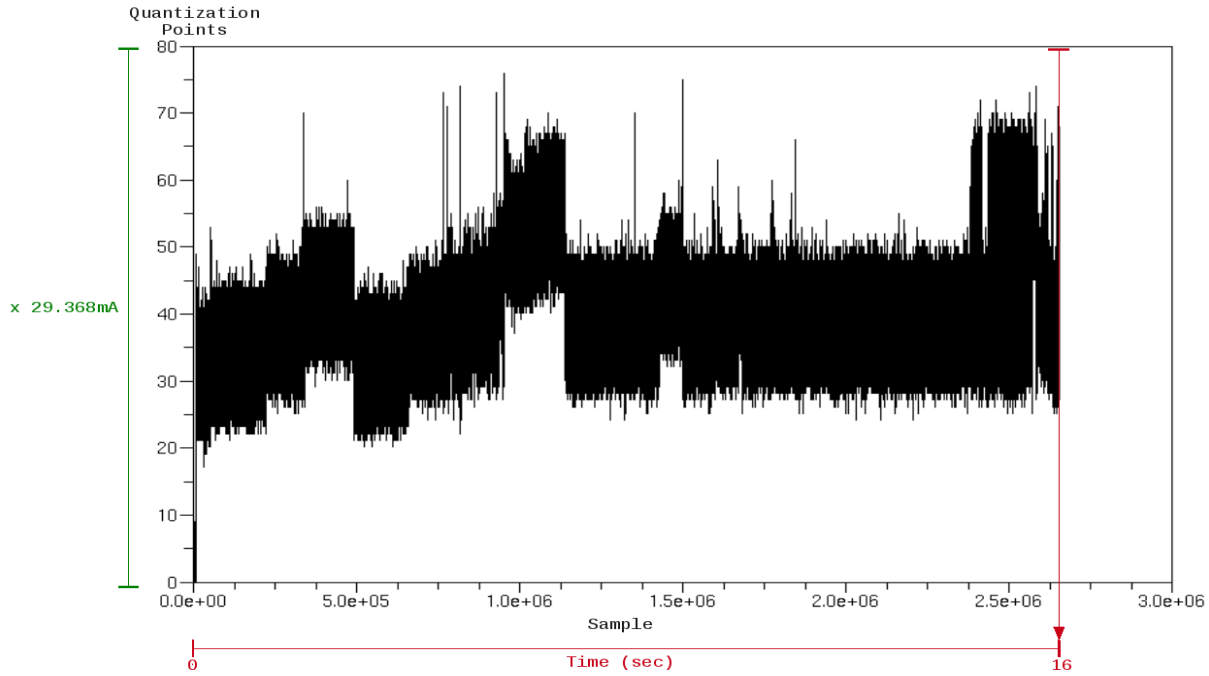


Figure 4.12: CPU Digital Current Signature from Dell Optiplex G150 (8 bit resolution 1 of 3).

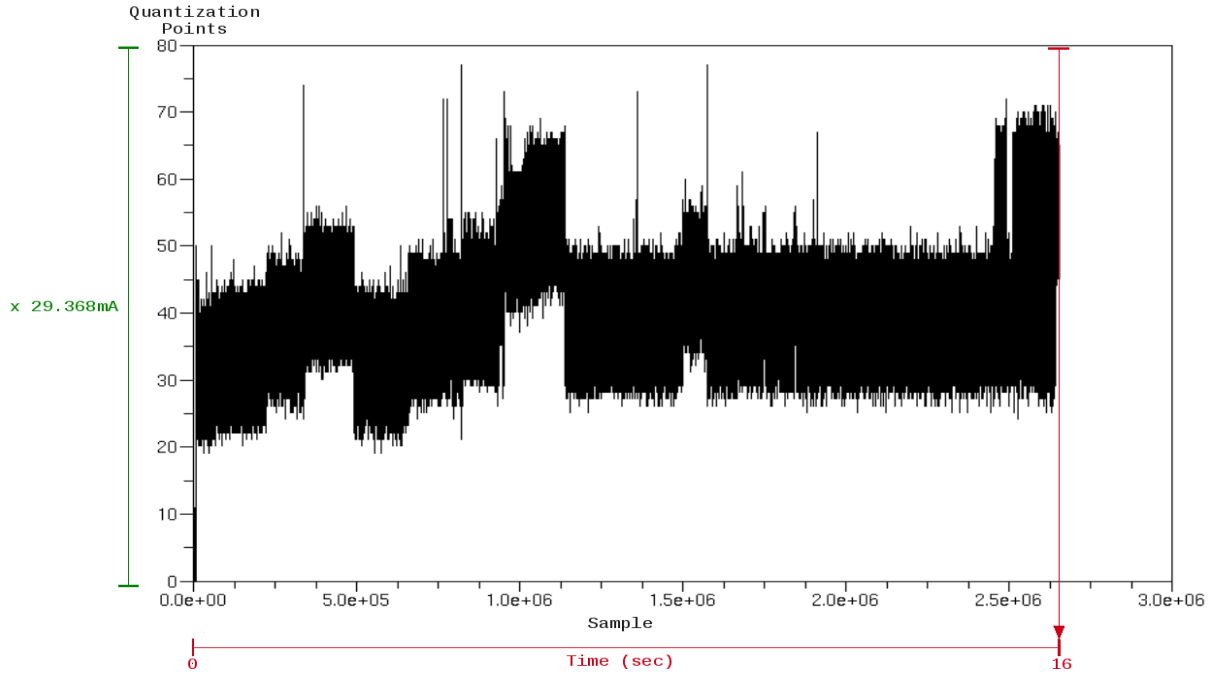


Figure 4.13: CPU Digital Current Signature from Dell Optiplex G150 (8 bit resolution 2 of 3).

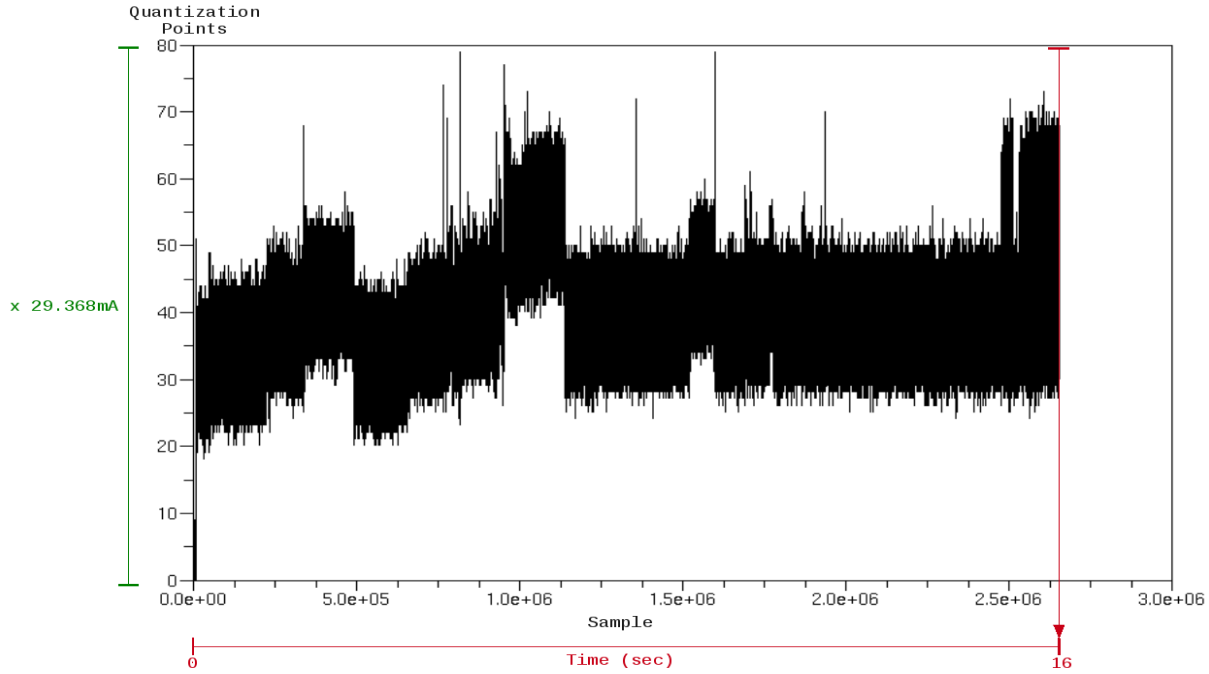


Figure 4.14: CPU Digital Current Signature from Dell Optiplex G150 (8 bit resolution 3 of 3).

The most obvious similarity between the 8 and 10 bit resolution signatures is the symmetry found between them. From the 8 bit power signatures we can also deduce the average minimum and maximum current consumed by the CPU by averaging the upper and lower bound side-bands. In the case of 8 bit signatures, each quantization point is worth 29.368mA. Hence, the minimum electrical current consumed by the CPU on average is 734mA and the average maximum is 1468mA. Another observation is the enormous difference in data points collected for 8 bit resolution signatures versus 10 bit resolution signatures. 8 bit resolution signatures posses on average 2.641 million samples per 16 seconds. Nearly twice as much information as the 10 bit resolution counterparts.

The next sequence of signatures reference the current consumed by the motherboard circuity. When referencing the motherboard signatures there is a variety of components to consider. In this particular case, the motherboard of the Dell Optiplex G150 encompasses a vast set of ICs dedicated to intercommunicate the CPU, RAM, video card,

PCI buss, and USB ports. Hence, when interpreting these signatures, consideration for the current consumption by the previously mentioned hardware must be accounted for. Any peripherals such as a as printer, external hard drives, USB sticks, etc that could cause an external current consumption have bee removed from the computing environment. Only the essential ones have been preserved (keyboard, mouse, monitor).

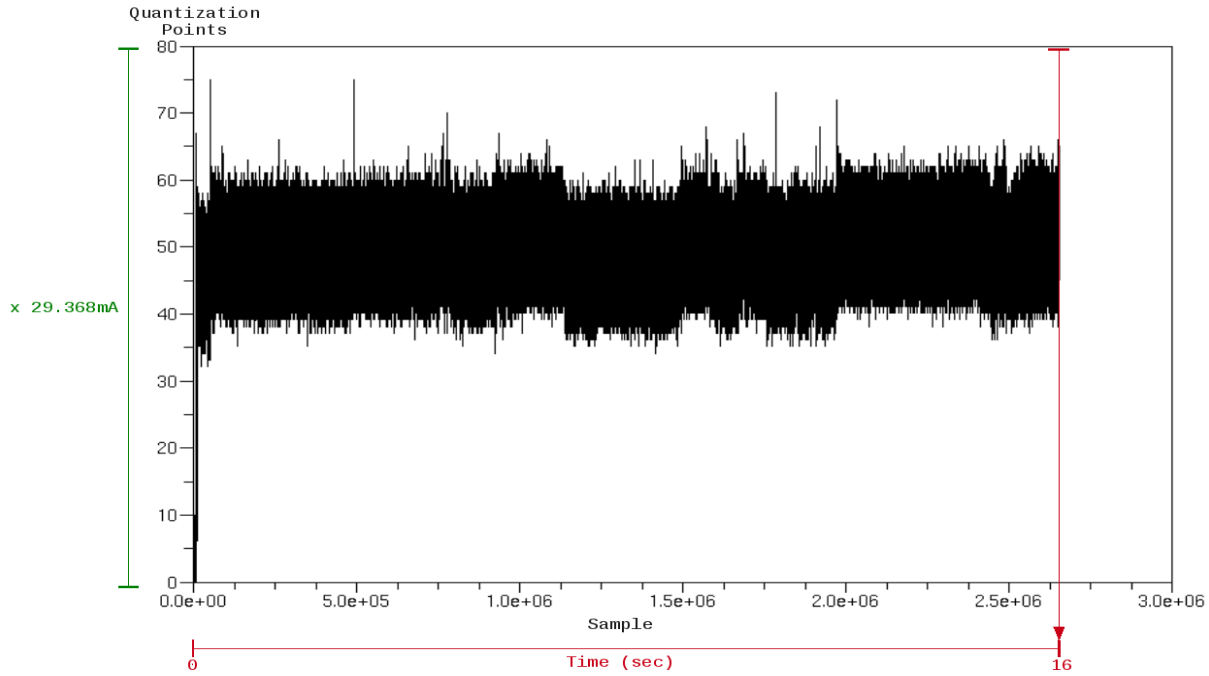


Figure 4.15: MB Digital Current Signature from Dell Optiplex G150 (8 bit resolution 1 of 3).

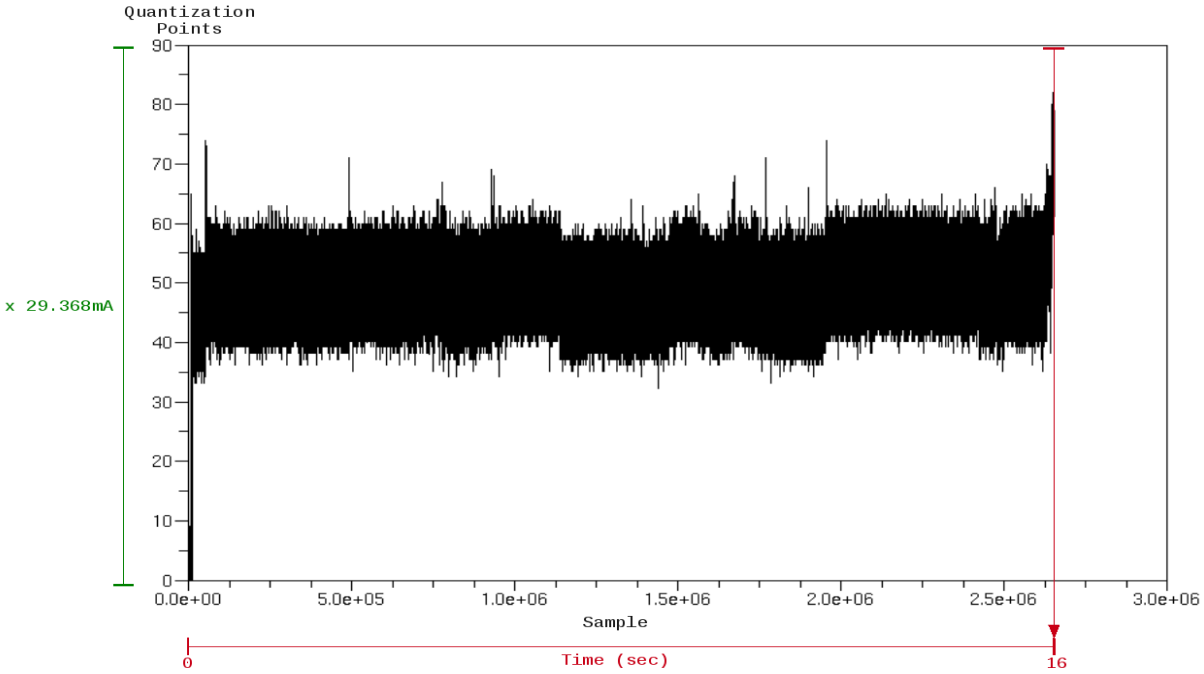


Figure 4.16: MB Digital Current Signature from Dell Optiplex G150 (8 bit resolution 2 of 3).

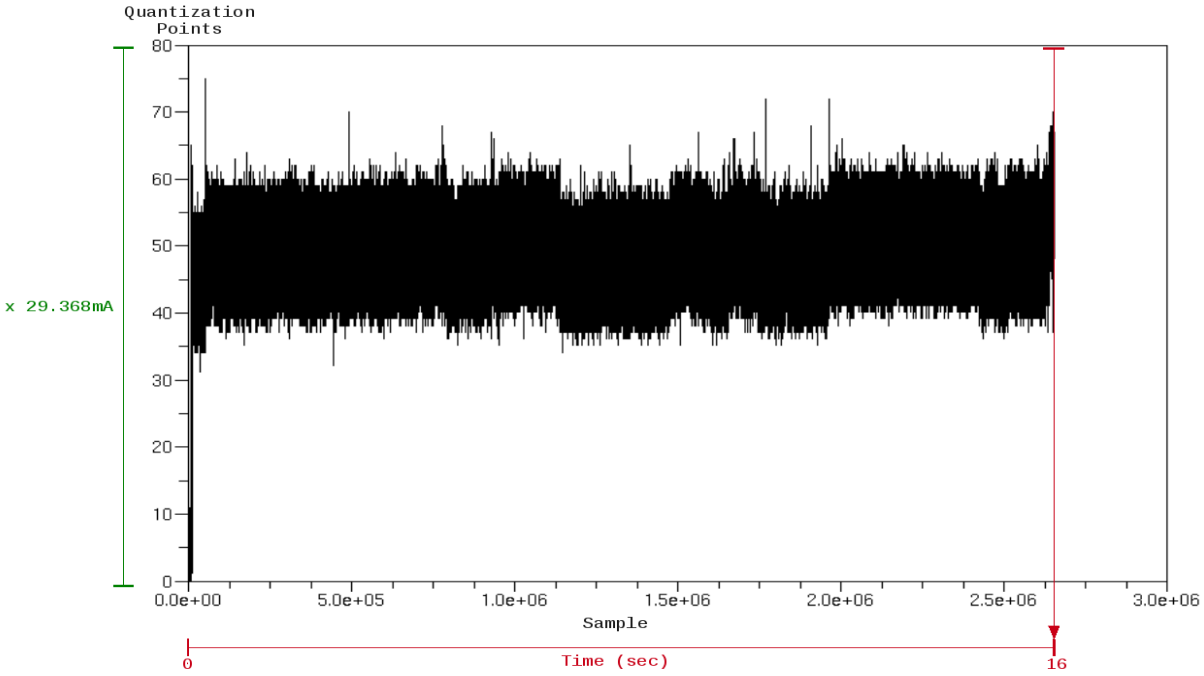


Figure 4.17: MB Digital Current Signature from Dell Optiplex G150 (8 bit resolution 3 of 3).

From the previous motherboard DPSs the following observations are made. The most obvious remark is the steady rate of electrical current being consumed. Careful observation reveals the average minimum to be in the vicinity of 1175mA, whereas, the maximum average is 1762mA. The consumption pattern also is absent of any variance for the great majority of the 16 seconds. This behavior is expected, since the entire set of components integrating the motherboard are TTL logic ICs and solid state devices. There is no reason to expect any of them to be flexible enough to behave as a processing unit capable of displaying an irregular electrical current consumption pattern.

The next sequence of signatures address the pattern of consumption reflected by the fan. Within this small computing environment (as with any other, at any other scale), the fan plays a vital role in safeguarding a steady flow of air inside the case, and therefore preventing the CPU (primarily) from a thermal break-down.

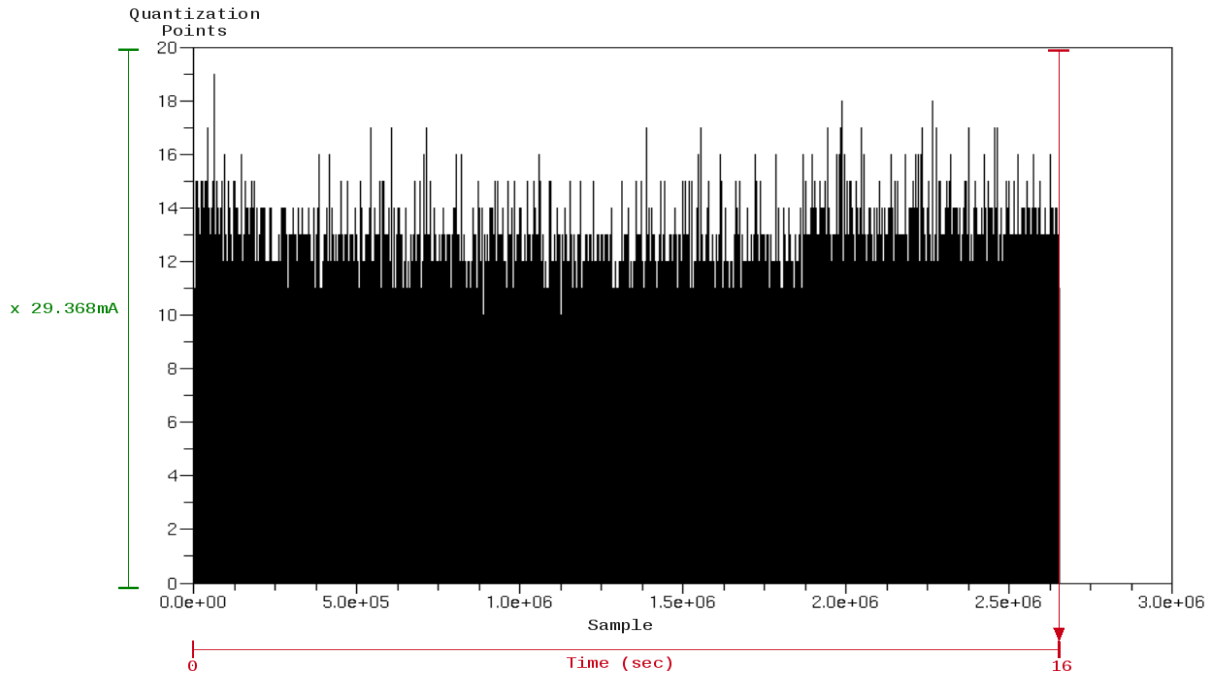


Figure 4.18: FAN Digital Current Signature from Dell Optiplex G150 (8 bit resolution 1 of 3).

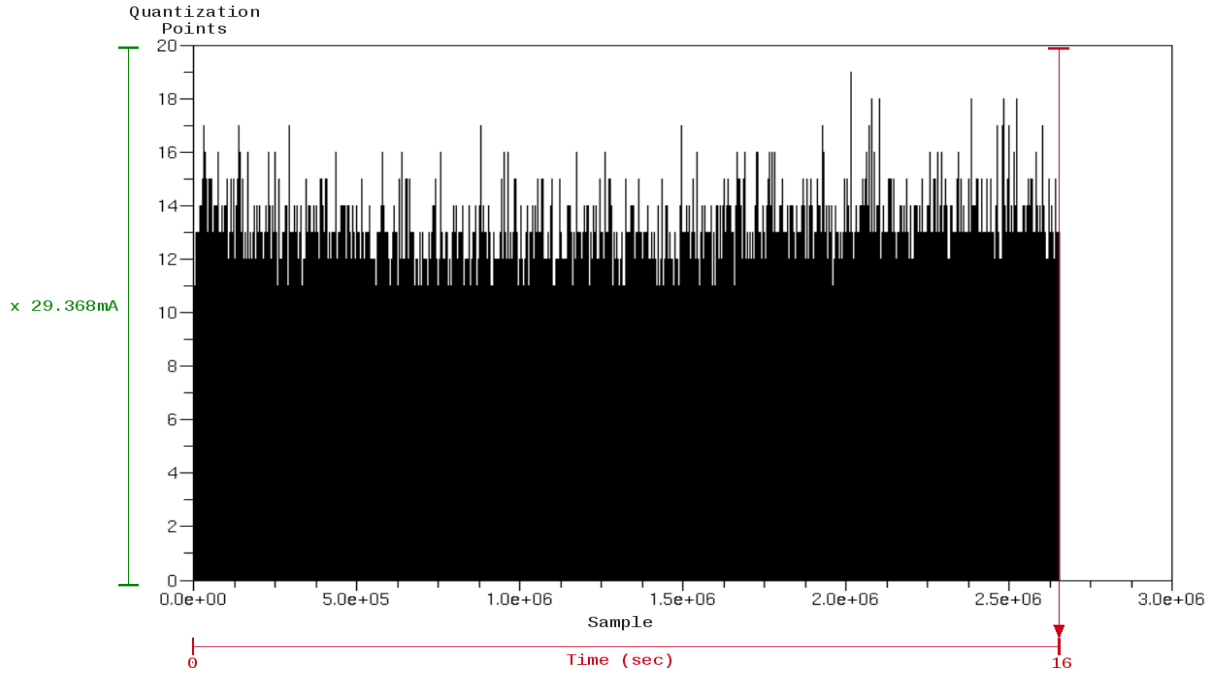


Figure 4.19: FAN Digital Current Signature from Dell Optiplex G150 (8 bit resolution 2 of 3).

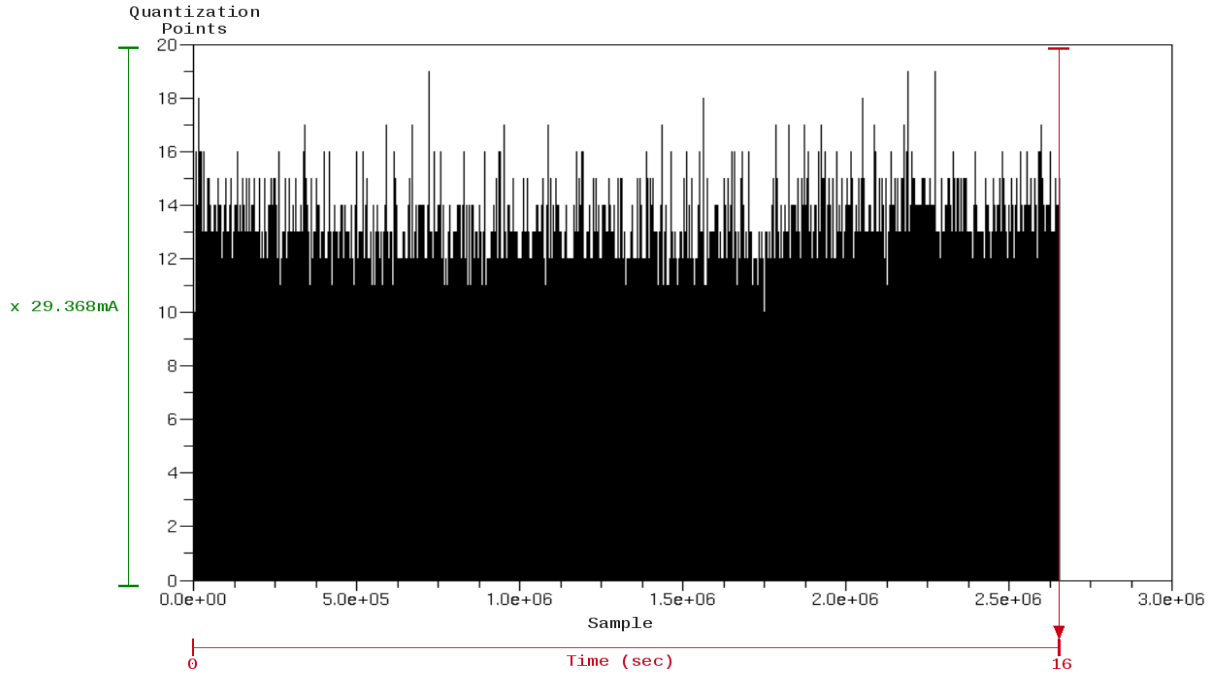


Figure 4.20: FAN Digital Current Signature from Dell Optiplex G150 (8 bit resolution 3 of 3).

The remarks regarding these signatures are the small amount of current being consumed by the fan and an average current consumption of 367.1mA. The over all pattern of consumption is stable because the drive-motor is a brushless one. The last set of current signatures describe the current consumption pattern described by the drive-motor inside the hard drive.

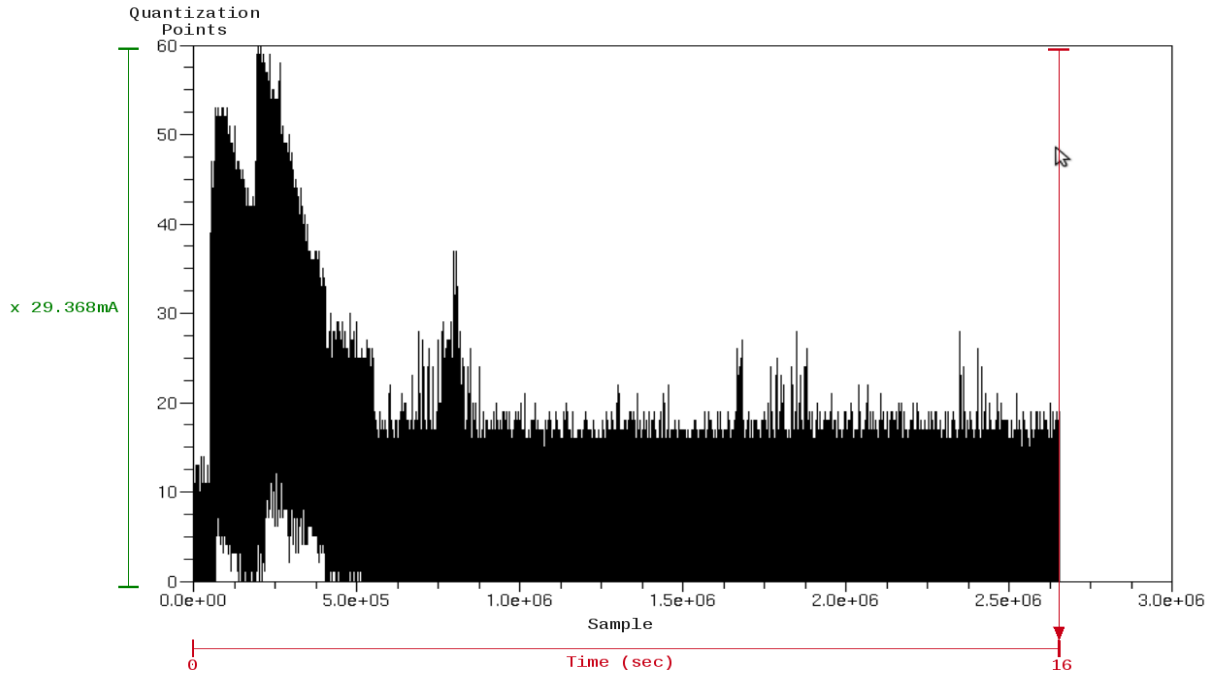


Figure 4.21: HD Digital Current Signature from Dell Optiplex G150 (8 bit resolution 1 of 3).

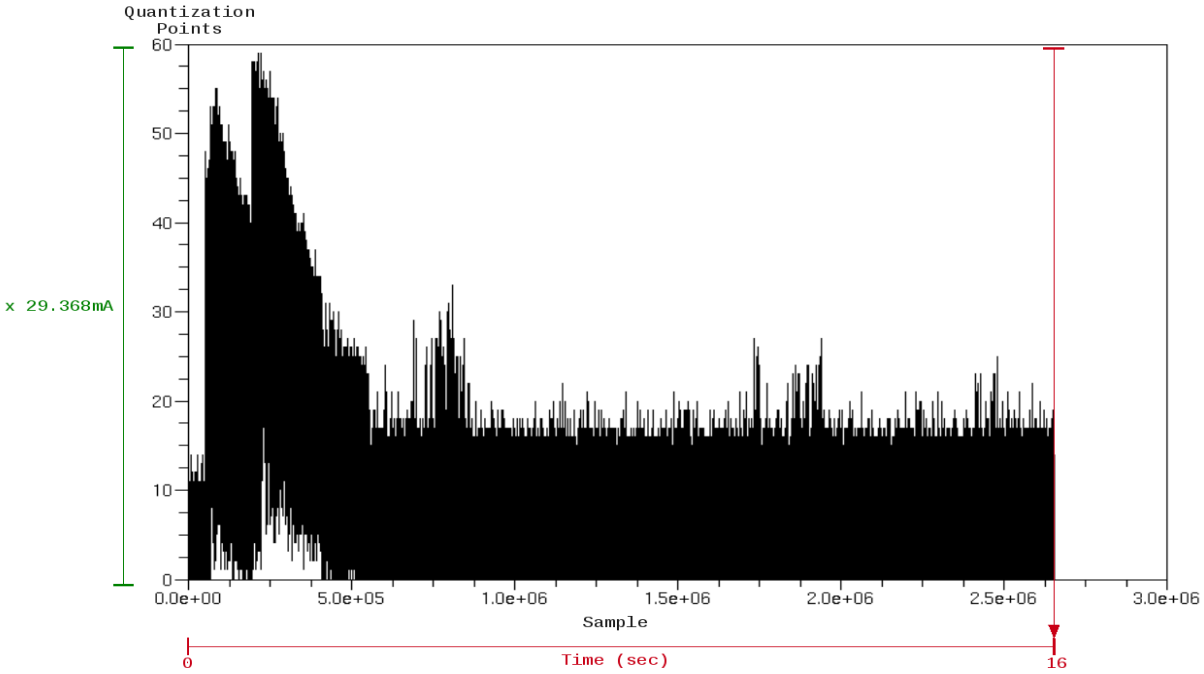


Figure 4.22: HD Digital Current Signature from Dell Optiplex G150 (8 bit resolution 2 of 3).

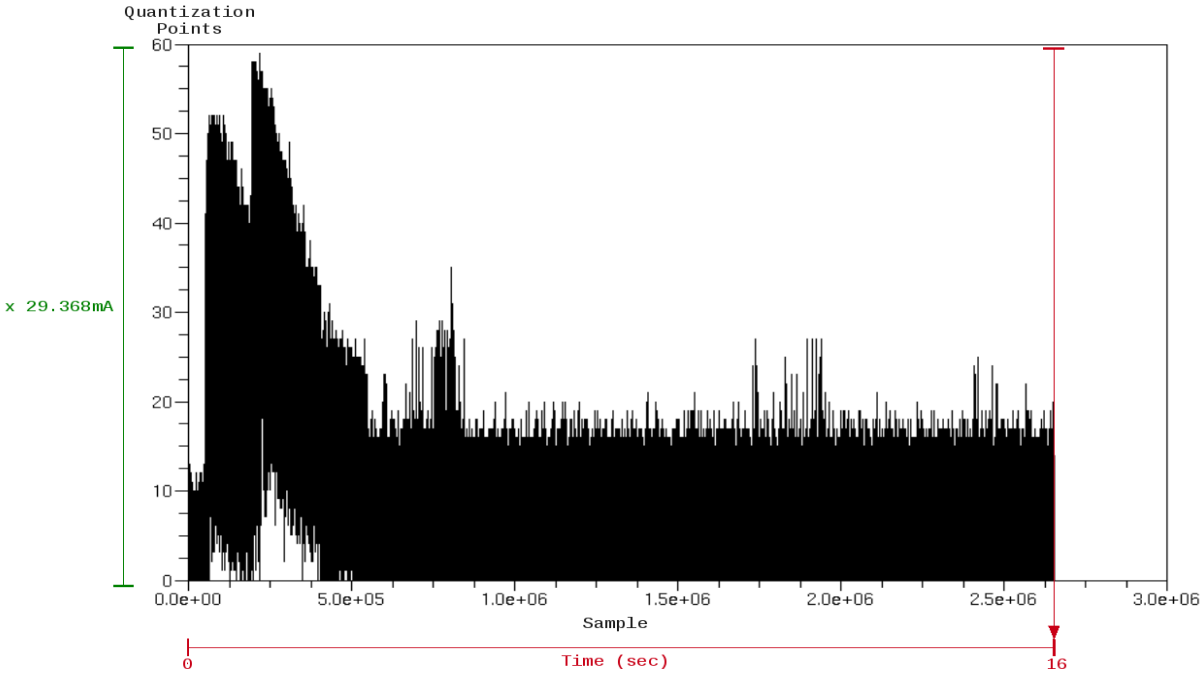


Figure 4.23: HD Digital Current Signature from Dell Optiplex G150 (8 bit resolution 3 of 3).

By observing the current signatures from the drive-motor inside the hard drive, it is feasible to expect a spike of considerable hight and amplitude at the beginning, as the drive-motor strives to bring the platers up to speed from a complete stand still. An over all higher current consumption is noted as a direct consequence. From the signatures, it is possible to determine both the average maximum amount of current consumed to spin-up the platers to the required RPM, and the average time required by the drive-motor to do so by analyzing the surge. In average, the maximum current used in this process is 1732 mA, while the average time the platters take to reach 7500 RPM is 3.164 seconds. The next chapter surmises all of the observations mentioned and the future work to implement a system capable of unobtrusively detect failures on electronic devices.

Chapter 5: Conclusion

The practice of speculating or asserting the functioning of a given electronic device via profiling the electrical current consumption of such, is a natural and feasible approach. Depending on the application and interest the approaches inevitably vary from field to field, however, electrical current profiling in general has pillared solid and withstanding methods to provide a quality prognosis of the target. Chapter two presented the two main variants of electrical current profiling relating to the work presented on this thesis: I_{DDQ} testing and motor current signature analysis (MCSA). I_{DDQ} testing has prevailed and adapted to both reductions in gate feature size and consequent scales in design infrastructure through time, to the point of becoming the leading quality assurance procedure in the CMOS industry. MCSA on the other hand, makes a different use of the electrical current signature to evaluate the condition of industrial grade induction motor drives. MCSA and I_{DDQ} testing are both invaluable tools for quality and reliability assurance purposes.

The signature analysis from the peripherals presented on chapter four and the work presented in [25], suggest the method used on this thesis to acquire Digital Power Signatures encompasses a high level of reliability. Hence, surmising the approach presented in this thesis as a reliable blueprint for general projects involving the capture of a digital power signature. However, the prime goal of the work presented on this thesis, is to provide a pattern to be subsequently used in a platform capable of evaluating defects for a wide range of electronic devices.

Section 5.1 : Future Work

The long term goal of this research is to implement a dedicated device capable of unobtrusively asserting a reliability prognosis for a given electronic device, and to predict as best as possible any present/future defects. Subsequently, the immediate future work is to investigate statistical analysis procedures capable of differentiating between faulty power profiles and non-faulty ones. Determinism will be a key factor in framing the power profiles within a context where random events do not disturb the prognosis process. Parallel to this work, is the integration of a more robust platform capable of processing the power signatures without the aid of the host. This entails the exclusion of the present USBMOD4 device used to transfer the signatures to an external processing element. Limitations on the available processing power provided by the currently used micro-controller (HCS12) will also have to be vanquished by utilizing a more sophisticated micro-controller capable of delivering an opportune real-time prognosis, without economically overthrowing the feasibility of this project.

References

[1] Singh, A.D.; "A Comprehensive Wafer Oriented Test Evaluation (WOTE) Scheme for the I_{DDQ} Testing of Deep sub-micron Technologies", *I_{DDQ} Testing, 1997. Digest of Papers., IEEE International Workshop,* 5-6 Nov. 1997 Page(s):40 – 43.

[2] Thibeault, C.; "A Novel Probabilistic Approach for IC Diagnosis Based on Differential Quiescent Current Signatures", *VLSI Test Symposium, 1997., 15th IEEE, 27 April-1 May 1997 Page(s):80 – 85.*

[3] Thomson, W.T.; Fenger, M.; "Case Histories of Current Signature Analysis to Detect Faults in Induction Motor Drives", *Electric Machines and Drives Conference, 2003. IEMDC'03. IEEE International,* Volume 3, 1-4 June 2003 Page(s):1459 - 1465 vol.3.

[4] Weekly, R.; Sungjun Chun; O'Connell, F.; "Characterization of Current Signatures for Microprocessors", *Electrical Performance of Electronic Packaging, 2004. IEEE 13th Topical Meeting,* 2004 Page(s):95 – 98.

[5] Gattiker, A.E.; Maly, W.; "Current Signatures", *VLSI Test Symposium, 1996., Proceedings of 14th,* 28 April-1 May 1996 Page(s):112 – 117.

[6] Gattiker, A.E.; Maly, W.; "Current Signatures: Application", *Test Conference, 1997. Proceedings., International,* 1-6 Nov. 1997 Page(s):156 – 165.

[7] Gattiker, A.; Nigh, P.; Grosch, D.; Maly, W.; "Current Signatures for Production Testing", *I_{DDQ} Testing, 1996., IEEE International Workshop,* 24-25 Oct. 1996 Page(s):25 – 28.

[8] Arumi, D.; Rodriguez-Montanes, R.; Figueras, J.; Eichenberger, S.; Hora, C.; Kruseman, B.; Lousberg, M.; Maihi, A.K.; "Diagnosis of Bridging Defects Based on Current Signatures at Low Power Supply Voltages", *VLSI Test Symposium, 2007. 25th IEEE,* 6-10 May 2007 Page(s):145 – 150.

[9] Patra, K.; Pal, S.K.; Bhattacharyya, K.; "Drill Wear Monitoring Through Current Signature Analysis Using Wavelet Packet Transform and Artificial Neural Network", *Industrial Technology, 2006. ICIT 2006. IEEE International Conference,* 15-17 Dec. 2006 Page(s):1344 – 1348.

[10] Schat, J.; "Evaluation of the I_{DDQ} Signature in Devices with Gauss-distributed Background Current", *Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop,* 16-18 April 2008 Page(s):1 – 6.

[11] Ramirez-Angulo, J.; Gonzalez-Altamirano, G.; "High Speed I_{DDQ} Current Sensors for VLSI System Testing", *Circuits and Systems, 1996. ISCAS '96., 'Connecting the World'., 1996 IEEE International Symposium,* Volume 1, 12-15 May 1996 Page(s):389 - 392 vol.1.

[12] Chih-Wen Lu; Chung Len Lee; Chen, J.E.; Chauchin Su; "A new I_{DDQ} Testing Scheme Employing Charge Storage BICS Circuit for Deep Submicron CMOS ULSI", *I_{DDQ} Testing, 1998. Proceedings. 1998 IEEE International Workshop,* November 12-13, 1998 Page(s):54 – 58.

[13] Rajsuman, R.; "I_{DDQ} Testing for CMOS VLSI", *Proceedings of the IEEE, Volume 88,Issue 4,*April 2000 Page(s):544 – 568.
[14] Variyam, P.N.; "Increasing the I_{DDQ} Test Resolution Using Current Prediction", *Test Conference, 2000. Proceedings. International,* 3-5 Oct. 2000 Page(s):217 – 224.

[15] Chih-Wen Lu; Chauchin Su; Chung Len Lee; Jwu-E Chen; "Is I_{DDQ} Testing Not Applicable for Deep Submicron VLSI in Year 2011", *Test Symposium, 2000. (ATS 2000). Proceedings of the Ninth Asian,* 4-6 Dec. 2000 Page(s):338 – 343.

[16] C. Thibeault; "On the Comparison of I_{DDQ} and I_{DDQ} Testing", *Proceedings of the 1999 17TH IEEE VLSI Test Symposium,* Page: 143, Year of Publication: 1999, ISBN:0-7695-0146-X.

[17] Kawahara, R.; Nakayama, O.; Kurasawa, T.; "The Effectiveness of I_{DDQ} and High Voltage Stress for Burn-in Elimination", *I_{DDQ} Testing, 1996., IEEE International Workshop,* 24-25 Oct. 1996 Page(s):9 – 13.

[18] Benini, L.; Macii, A.; Macii, E.; Poncino, M.; "Discharge Current Steering for Battery Lifetime Optimization", *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium,* 2002 Page(s):118 – 123.

[19] Rao, R.; Vrudhula, S.; Rakhmatov, D.; "Analysis of Discharge Techniques for Multiple Battery Systems", *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium,* 25-27 Aug. 2003 Page(s):44 – 47.

[20] T.T.T., Inc.; "I_{DDQ} Tutorial", www.ttt.com, Copyright © by T.T.T. (Test Team Technologies), Inc. 1996,1997.

[21] McEuen, S.D.; "I_{DDQ} Benefits", *VLSI Test Symposium, 1991. 'Chip-to-System Test Concerns for the 90's', Digest of Papers,* 15-17 April 1991 Page(s):285 – 290.

[22] Li, J.C.M.; McCluskey, E.J.; "I_{DDQ} Data Analysis Using Current Signature", *I_{DDQ} Testing, 1998. Proceedings. 1998 IEEE International Workshop,* November 12-13, 1998 Page(s):37 – 42.

[23] Thomson, W.T.; Gilmore R.J.; "Motor Current Signature Analysis to Detect Faults in Induction Motor Drives -Fundamentals, Data Interpretation, and Industrial Case Histories", *Proceedings of the thirty-second turbomachinery symposium,* 2003 Page(s):145 – 156.

[24] Howard W Penrose; "Practical Motor Current Signature Analysis -Taking the Mystery out of MCSA", *BJM Corp (www.bjmcorp.com)*, Copyright © by BJM Corp 2003.

[25] Giancarlo R.; "Determinism in Power Signatures of Electronics for Health Monitoring", *The University of Texas at El Paso, Department of Electrical and Computer Engineering*, Copyright © by Giancarlo Rayas, 2008.

Appendix

A. HCS12 Algorithm

The following pages describe the c code used to program the Dragon-12 development board (containing the HCS12 micro-controller). The development environment used for compilation is Eclipse. Eclipse is an open-source application available for Linux operating systems. The following flowchart illustrates the concept behind the algorithm implemented on the HCS12. Special thanks to Navid Mohaghegh for his support, tutorials and outstanding web site.

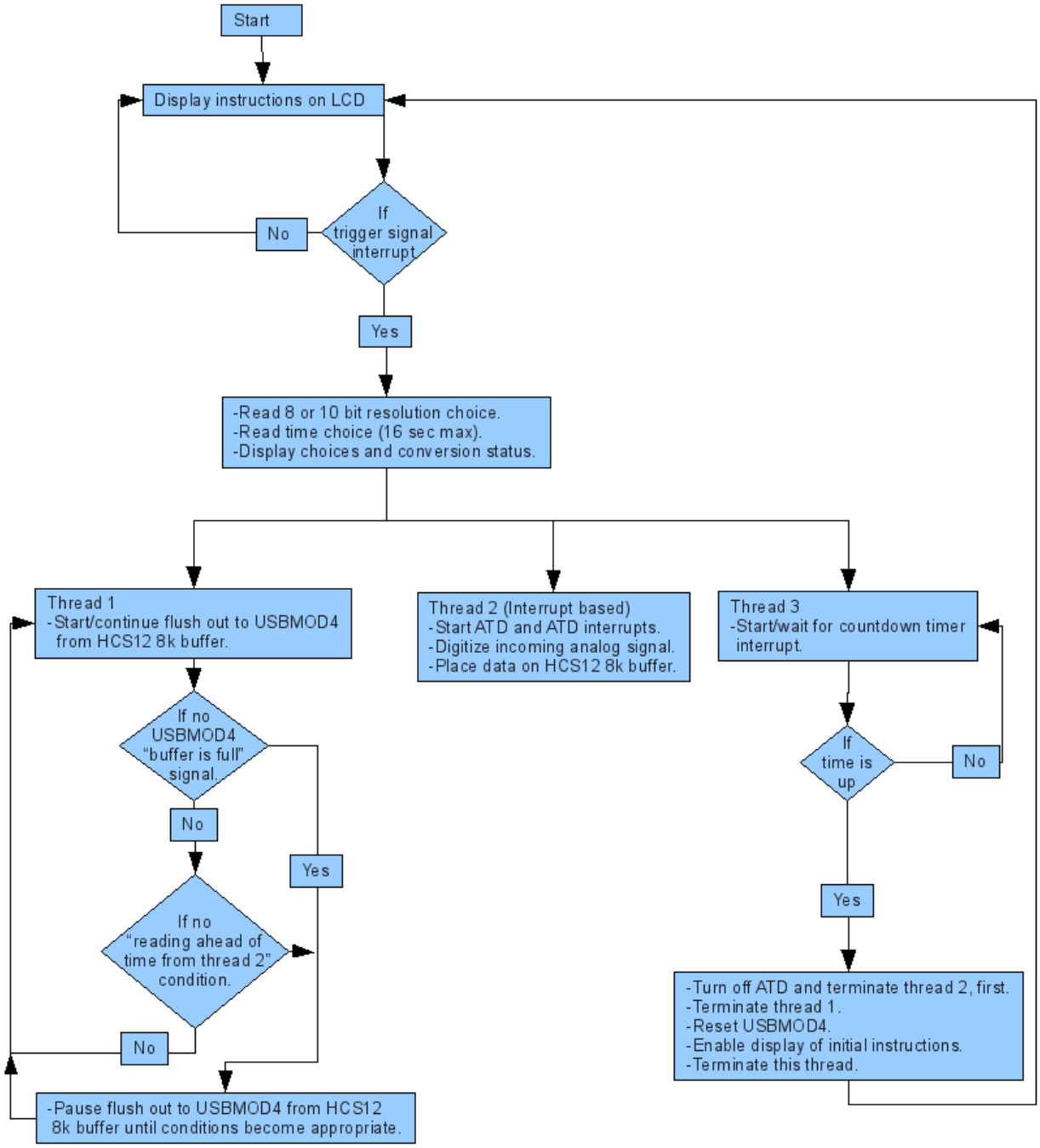


Figure A.1: HCS12 process algorithm.

B. HCS12 c Code

Application programing environments like CodeWarrior and Eclipse automatically include all the necessary files and libraries needed to develop code for a target micro-controller. For this reason, only the sections of c code relative to the algorithm presented in the previous section are presented.

main.c

```
#ifndef _MAIN_C_
#define _MAIN_C_
/*****************************************************************************/
#include "hcs12.h"
#include "atd_init.h"
#include "atd_interr.h"
#include "jobs_and_init.h"
#include "job_exec.h"
#include "rti_init_and_interr.h"
#include "the_jobs.h"
#include "global_vars.h"
#include "pj7_interr.h"

int main ( void ){

        DDRB = 0xFF; /* portb all outputs */
        PORTB = 0x00;/* portb all zeros to be safe */

        DDRJ = 0x01;   /* ptj0 we will strobe (USB-send), ptj1 tells if USBMOD4 is full,
                                ptj7 signals interrupt to start everything */
        RDRJ = 0xFE;   /* full strength on ptj0 */
        PERJ = 0x00;   /* pull ups/downs disabled */
        PPSJ = 0x7F;   /* polarity and interrupt on falling edge trigger for pj7 */
        PIEJ = 0x80;   /* interrupt on pj7*/
                                /* PIFJ = 0x80; clear interrupt flag */
                                /* PTJ = 0x00 */

        DDRA = 0x01;
        PORTA = 0x01;

        init_the_jobs();
        atd_init();
        LCD_Init();

        LCDScrollLine( 1, _str_1 );
        LCDScrollLine( 2, _str_2 );

        /* Define the jobs------------"name"-"per"--"coun"-"exe"--"fcn_ptr" */
        malloc_this_job(&job_list[2], "job2", 300    ,0       ,TRUE, &job2_LCD_scroll);

        rti_init(0x10, 1);
        enable_interrupts();

/* I hate to do this, but, this provides a faster way to flush data out,
 * and is more stable */
for(;;){
while(flush_out == TRUE){
if((tracker_out != tracker_in) && ((PTJ & 0x02) == 0x00)){

               PTJ = 0x01;
                       PORTB = data[tracker_out];          /* in goes the data byte one */
                       PTJ = 0x00;
```

```
                    tracker_out++;
                    tracker_out &= 0x1FFF;

       }/* end if */
}/* end while */
}/* end for ever */

return 0;
}
#endif
```

atd_init.h

```
#ifndef _ATD_INIT_H_
#define _ATD_INIT_H_
/*****************************************************************************/

void BANK8 atd_init (void);
void BANK8 atd_init (void){

ATD0CTL2 = 0xC0;
/*ADPU and ATD Disable / Power Down
(bit15=1) (1)=Enable ATD.
(0)=Disable & power down the ATD.
-------------------------------------------------------------------------------

AFFC and ATD Fast Conversion Complete Flag Clear
(bit14=1) (1)=Fast clear mode (for interrupts).
(0)=Conversion Complete Flag (CCF) clears by reading the status
register -first, reading the result register -second,
and starting a new conversion sequence -third.
-------------------------------------------------------------------------------

AWAI and ATD Wait Mode
(bit13=0) (1)=Enable ATD wait function during MCU Wait mode.
(0)=Disable ATD wait function.
-------------------------------------------------------------------------------

ETRIGLE and External Trigger Level/Edge control

(bit12=0) (1)=Level mode Â– active level gates ATD operation
(0)=Edge mode Â– active edge mode
-------------------------------------------------------------------------------

ETRIGP and External Trigger Polarity

(bit11=0) (1)=Active high level or rising edge active
(0)=Active low level or falling edge active
-------------------------------------------------------------------------------

                     ETRIGE and External Trigger Mode enable
(bit10=0) (1)=Enable external trigger mode
(0)=Disable external trigger mode.
-------------------------------------------------------------------------------

ASCIE and ATD Sequence Complete Interrupt Enable
(bit09=0) (1)=Enables ATD interrupt on Sequence Complete.
(0)=Disables ATD interrupt (for now, this is just initialization).
-------------------------------------------------------------------------------

ASCIF and ATD Sequence Complete Interrupt Flag
(bit08=0) (1)=ATD sequence complete interrupt occurred.
```

```
(0)=No ATD sequence complete interrupt occurred.
-------------------------------------------------------------------------------*/

ATD0CTL3 = 0x08;
/*
       (bit07=x) -No use.
-------------------------------------------------------------------------------

                             S8C/S4C/S2C/S1C - Conversion Sequence Length
       (bit06<->03=0001) One conversion per sequence (we are using only one chanel aniway).

                             S8C S4C S2C S1C Number of Conversions
                             per Sequence
 0 0 0 0 8
 0 0 0 1 1
 0 0 1 0 2
 0 0 1 1 3
 0 1 0 0 4
 0 1 0 1 5
 0 1 1 0 6
 0 1 1 1 7
 1 X X X 8
-------------------------------------------------------------------------------


                             FIFO and Result Register FIFO Mode
 (bit02=0)(1)=Result registers do not map to the conversion sequence.
(0)=Result registers maps to the conversion sequence (Do not use FIFO mode)
-------------------------------------------------------------------------------

FRZ1, FRZ0 and Background Debug Freeze Enable .
 (bit01<->00=00)-Ignore any background signals.

FRZ1 FRZ0ATD-RESPONSE
                             0       0     Ignore the background signal
01Reserved
                             1       0Finish current conversion, then freeze
                             11Freeze Immediately
-------------------------------------------------------------------------------*/

ATD0CTL4 = 0x05;/* see pj7_interr.h for final setup on ATD0CTL4 */
/*                           SRES8 and A/D Resolution Select
 (bit15=1)(1)=8-bit resolution selected.
(0)=10-bit resolution selected.
-------------------------------------------------------------------------------

SMP0, SMP1 - Sample Time Select
 (bit14<->13=10)SMP1 SMP0 Final Sample Time
 00 2 A/D clock periods
 01 4 A/D clock periods
 10 8 A/D clock periods
 11 16 A/D clock periods
-------------------------------------------------------------------------------

PRS0, PRS1, PRS2, PRS3, PRS4 and ATD Clock Prescaler
 (bit12<->8=00101)Divide the ATD module clock (24Mhz) by ((xxxxx+1)*2) to get the ATD
clock frequency.

Formula:

ad clock frequency =(bus freq =24Mhz) / ((xxxxx+1)*2)

in our case:

ad module clock frequency = 24Mhz / ((5+1)*2) = 2Mhz.

Further subdivide the clock frequency by the sample test time to
```

```
get the sampling rate.

Formula:
ad module clock frequency
sampling rate = ------------------------------
A/D clock periods used
in Final Sample Time + 4 + 2

in our case: sampling rate = 2Mhz / (2+4+8) = 142.857Khz.
-------------------------------------------------------------------------------*/

ATD0CTL5 = 0x02;
/*                          DJM - Result Register Data Justification Mode
 (bit07=0)(1)=Right justified mode.
(0)=Left justified mode.
-------------------------------------------------------------------------------

DSGN - Signed/Unsigned Result Data Mode
 (bit06=0)(1)=Signed result register data select.
(0)=Unsigned result register data select.
-------------------------------------------------------------------------------
SCAN - Continuous Conversion Sequence Mode
 (bit05=0)(1)=Perform conversion sequences continuously.
(0)=Perform a conversion sequence - return to idle mode.
-------------------------------------------------------------------------------

MULT - Multi-Channel Sample Mode
 (bit04=0)(1)=Sample across many channels.
(0)=Sample only the specified channel.
-------------------------------------------------------------------------------


 (bit03=x)-No use.
-------------------------------------------------------------------------------

CC, CB, CA - Analog Input Channel Select Code
 (bit02<->00=000)Sample Chanel 00 only for now.

CC CB CA Analog Input Channel
 00 0 AD0
 00 1 AD1
 01 0 AD2
 01 1 AD3
 10 0 AD4
 10 1 AD5
 11 0 AD6
 11 1 AD7
-------------------------------------------------------------------------------*/
}
#endif
```

atd_inter.h

```
#ifndef _ATD_INTERR_H_
#define _ATD_INTERR_H_
/*****************************************************************************/
#include "global_vars.h"

void INTERRUPT atd_isr( void );
void INTERRUPT atd_isr( void ){

       ATD0CTL2 = 0xC0;                          /* block ATD interrups while service */
```

```
        /* 8 bit resolution */
        if((ATD0CTL4 & 0x80) == 0x80 ){

        data[tracker_in] = ATD0DR0H;/* conversion into next buffer slot */
        tracker_in++;                                   /* increment the buffer input pointer */
        tracker_in &= 0x1FFF;                /* compare input pointer to buffe's
                                                                        maximum size and loop back to
the
                                                                        begining of it if limit has
been
                                                                        reached */
        }else{
        /*10 bit resolution */

        data[tracker_in] = ATD0DR0H;/* conversion into next buffer slot */
        tracker_in++;                                   /* increment the buffer input pointer */
        tracker_in &= 0x1FFF;                /* compare input pointer to buffe's
                                                                        maximum size and loop back to
the
                                                                        begining of it if limit has
been
                                                                        reached */
        data[tracker_in] = ATD0DR0; /* conversion into next buffer slot */
        tracker_in++;                                   /* increment the buffer input pointer */
        tracker_in &= 0x1FFF;/* compare input pointer to buffe's
                                                                        maximum size and loop back to
the
                                                                        begining of it if limit has
been
                                                                        reached */


        }

ATD0CTL2 = 0xC2;                                /* enable ATD interrupts */
ATD0CTL5 = 0x02;                                /* start a next ATD sequence */

}
#endif
```

global_vars.h

```
#ifndef _GLOBAL_VARS_H_
#define _GLOBAL_VARS_H_
/*****************************************************************************/
                                /* data buffer */
static volatile
                byte data[8*1024];

                                /* a control variable */
static volatile
                boolean flush_out;

                                /* trackers to the buffer */
static volatile       int
                tracker_in = 0x0000,
                tracker_out = 0x0000;
 #endif
```

```
#ifndef _JOBS_AND_INIT_H_
#define _JOBS_AND_INIT_H_
/*********************************JOB LIST STARTS*****************************/
#define JOB_LIST_SIZE 3
#define NULL ((void *)0)
/**
 * a periodic job will have the following:
 * name (i.e. job0-pwm)
 * its period (i.e. every 3 seconds)
 * counters to check if the time is up to execute it
 * a pointer to a function related to this job
 * functions should ONLY be "void function(void);"
 */
typedef struct{
        unsigned char   *name;
        unsigned long  period;
        unsigned longcounter;
        booleanexecute;
        void (*job_function_pointer)(void);
} JOB;


int job_list_size = JOB_LIST_SIZE;
JOB job_list[JOB_LIST_SIZE];
JOB zero_job = {0,0,0,FALSE,NULL};


/**
 * init the job list and set all the pointers and vars to 0
 */
void BANK8 init_the_jobs(void);
void BANK8 init_the_jobs(void){

        int i = 0;
        for (i = 0; i < job_list_size; i++ ){
                job_list[i].name = zero_job.name;
                job_list[i].period = zero_job.period;
                job_list[i].counter = zero_job.counter;
                job_list[i].execute = zero_job.execute;
                job_list[i].job_function_pointer = zero_job.job_function_pointer;
        }
}


/**
 * allocate the memory for each variable in the JOB struct and assign given values
 */
void BANK8 malloc_this_job(JOB* job, unsigned char* name, unsigned long period,
 unsigned long counter, boolean execute,
 void (*job_function_pointer)(void));
void BANK8 malloc_this_job(JOB* job, unsigned char* name, unsigned long period,
 unsigned long counter, boolean execute,
 void (*job_function_pointer)(void)){

        job->name = name;
        job->period = period;
        job->counter = counter;
        job->execute = execute;
        job->job_function_pointer = job_function_pointer;
}
#endif
```

job_exec.h

```
#ifndef _JOB_EXEC_H_
#define _JOB_EXEC_H_
/*****************************************************************************/
/**
 * helper function for "execute_the_jobs"
 */
void BANK8 execute_a_job(JOB* job);
void BANK8 execute_a_job(JOB* job){

if(job->execute == TRUE){
       if (job->counter == job->period){
              job->counter = 0;
              (* (job->job_function_pointer) )();
       }else{
              job->counter = job->counter + 1;
       }
}
}
/**
 * execute the whole job list IF possible
 * a job will be executed if its period has been reached
 */
void BANK8 execute_the_jobs(JOB job_list[] , int job_list_size);
void BANK8 execute_the_jobs(JOB job_list[] , int job_list_size){

       int i = 0;
       for (i = 0; i < job_list_size; i++ ){

              execute_a_job(&job_list[i]);
       }
}
#endif
```

lcd.h

```
#ifndef _LCD_H_
#define _LCD_H_
/*****************************************************************************/
/* LCD Routines */

typedef struct _scrollData{
 word LCDstartDelay;
 word LCDcharDelay;
 word LCDdelayCounter;
 word LCDlength;
 byte LCDstate;
 char* LCDscrollData;
 int LCDindex;
} ScrollData;


void LCD_Init(void);
void LCDUpdateScroll(void);
void LCDWriteLine(byte line, char* d);
void LCDScrollLine(byte line, char* d );
ScrollData* LCDSetStartDelay( int which, word delay);
ScrollData* LCDSetCharDelay( int which, word delay);
void delay(int ms);

#endif
```

pj7_inter.h

```
#ifndef _PJ7_INTERR_H_
#define _PJ7_INTERR_H_
/*****************************************************************************/
char* _str_3 = "08b resulution";
char* _str_4 = "10b resolution";
char* _str_5 = "!!!USB-->Comp!!!";

void INTERRUPT pj7_isr( void );
void INTERRUPT pj7_isr( void ){

	PIFJ = 0x80;                        /* clear interrupt flag ptj & trackers */

	tracker_in =0x0000;          /* initialize both trackers */
	tracker_out = 0x0000;

	/* 8 bit resolution */
	if((PTJ & 0x40) == 0x00){

		job_list[2].execute = FALSE;  /* no more scrolling */
		LCDWriteLine( 1, _str_3 );           /* print 8 bit choice */
		LCDWriteLine( 2, _str_5 );           /* print alert message */

	/* enable jobs to stop ad per PTH time set up */
		malloc_this_job(&job_list[1], "job1", (PTH * 1927)  , 0, TRUE,     &job1_stop_ad);

		flush_out = TRUE;
		ATD0CTL4 = 0x82;                     /* 8 bit resolution */
		ATD0CTL2 = 0xC2;                     /* enable atd sequence complete interrupts */

	ATD0CTL5 = 0x02;                     /* start a ATD sequence to cache the first ATD
interrupt */

	}else{
	/* 10 bit resolution */

		job_list[2].execute = FALSE;  /* no more scrolling */
		LCDWriteLine( 1, _str_4 );           /* print 10 bit choice */
		LCDWriteLine( 2, _str_5 );           /* print alert message */

	/* enable jobs to stop ad per PTH time set up */
		malloc_this_job(&job_list[1], "job1", (PTH * 1927)  , 0, TRUE,     &job1_stop_ad);

	flush_out = TRUE;
		ATD0CTL4 = 0x02;                     /* 10 bit resolution */
		ATD0CTL2 = 0xC2;                     /* enable atd sequence complete interrupts */

	ATD0CTL5 = 0x02;                     /* start a ATD sequence to cache the first ATD
interrupt */

	}
}
#endif
```

rti_init_and_inter.h

```
#ifndef _RTI_INIT_AND_INTERR_H_
#define _RTI_INIT_AND_INTERR_H_
/*****************************************************************************/
volatile unsigned int counter_for_real_time_interrupt;
unsigned int counter_for_real_time_interrupt_limit;

void INTERRUPT rti_isr( void );
void INTERRUPT rti_isr( void ){
       CRGFLG = 0x80; //clear the RTI
       //for instance if limit is 10, every 10 intrupts do something ...
       if (counter_for_real_time_interrupt == counter_for_real_time_interrupt_limit){
              //reset the counter
              counter_for_real_time_interrupt = 0;
              //do some work
              execute_the_jobs(job_list , job_list_size);
       }else{
       counter_for_real_time_interrupt ++;
       }
}
/*initilize the rti: rti_ctl_value will set the prescaler */
void BANK2 rti_init(unsigned char rti_ctl_value, unsigned int counter_limit);
void BANK2 rti_init(unsigned char rti_ctl_value, unsigned int counter_limit){

       //set the maximum lmit for the counter:
       //if max set to be 10, every 10 intrupts some work will be done
       counter_for_real_time_interrupt_limit = counter_limit;
       // Every 32.768 -- see below --
       RTICTL = rti_ctl_value;
       // How many times we had RTI intruppts
       counter_for_real_time_interrupt = 0;
       //i.e: RTICTL == 0x63 == set rate to 32.768ms:
       //The clock divider is set in register RTICTL and is: (N+1)*2^(M+9),
       //where N is the bit field RTR3 through RTR0
       //and M is the bit field RTR6 through RTR4.
       //0110 0011 = 0x63 ==> 1 / (4MHz / 4*2^15) == 4*2^15 / 4,000 = 32.768ms
       //0111 1111 = 0x7F ==> 1 / (4MHz / 16*2^16) == 2^20/ 4,000 = 262.144ms

       // Enable RTI interrupts
       CRGINT |= 0x80;
       // Clear RTI Flag
       CRGFLG = 0x80;

}
#endif
```

the_jobs.h

```
#ifndef _THE_JOBS_H_
#define _THE_JOBS_H_
/*****************************************************************************/
#include "global_vars.h"
#include "lcd.h"

char* _str_1 = "Digital Current Signature Capture. Jonathan Cervantes (Thesis).";
char* _str_2 = "Connect USB<->Comp; Reset board; Select resolution (8/10 bit); Select duration
(PORTH * 1s); Trigger (twice for 1st time only); cat < /dev/ttyUSBx > [directory]/filename.txt.";
char* _str_6 = "Capture finished";
char* _str_7 = "Use perl script";

void BANK9 job0_finish(void);
void BANK9 job0_finish(void){
```

```
        LCDScrollLine( 1, _str_1 );/* set up scroll initial mesage */
        LCDScrollLine( 2, _str_2 );/* set up scroll initial mesage */
        job_list[2].execute = TRUE;/* enable scrolling */
        job_list[0].execute = FALSE; /* turn job "job3_finish" -mtself) */
}
void BANK9 job1_stop_ad(void);
void BANK9 job1_stop_ad(void){

        ATD0CTL2 = 0xC0;                    /* turn ad off before "flush_out" */
        flush_out = FALSE;                  /* disable some bad programming */

        PORTA = 0x01;
        delay(5000);
        PORTA = 0x00;                       /* reset the USBMOD */
        delay(5000);
        PORTA = 0x01;
        delay(5000);

        LCDWriteLine( 1, _str_6 );
        LCDWriteLine( 2, _str_7 );

 malloc_this_job(&job_list[0], "job0", 4000 ,0       ,TRUE, &job0_finish);
        job_list[1].execute = FALSE; /* turn job "job1_stop_ad" -mtself) */
}
void BANK9 job2_LCD_scroll(void);
void BANK9 job2_LCD_scroll(void){
        LCDUpdateScroll();                  /* update the scrolling of any message */
}
#endif
```

interrupts.c

```
#ifndef _INTERRUPTS_C_
#define _INTERRUPTS_C_
#include "hcs12.h"

/* Reset the COP.*/
extern inline void cop_reset (void);
extern inline void cop_reset (void)
{
ARMCOP = 0x55;
ARMCOP = 0xAA;
}

void fatal_interrupt ()
{
/*
 * Infinite loop for debugging
 * Returning would not help as it's necessary to clear the interrupt flag.
 */
for (;;)
{
cop_reset();
}
}


#ifdef USE_INTERRUPT_TABLE
/*
 * these ISR must be in non-banked memory (near)
 */
```

```
/*******************************************************/
/* Register the handeler unit both here and in below */
/*******************************************************/
void INTERRUPT rti_isr( void );
void INTERRUPT atd_isr( void );
void INTERRUPT pj7_isr( void );
/* void INTERRUPT pp1_isr( void ); */

/*
 * Interrupt vectors table.
 * Note: the `XXX_handler: foo' notation is a GNU extension which is
 * used here to ensure correct association of the handler in the struct.
 * This is why the order of handlers declared below does not follow the MCU order.
 */
const struct interrupt_vectors __attribute__((section(".vectors"))) vectors =
{
pwm_shutdown_handler:
fatal_interrupt,
ptpif_handler:
fatal_interrupt,
can4_tx_handler:
fatal_interrupt,
can4_rx_handler:
fatal_interrupt,
can4_err_handler:
fatal_interrupt,
can4_wake_handler:
fatal_interrupt,
can3_tx_handler:
fatal_interrupt,
can3_rx_handler:
fatal_interrupt,
can3_err_handler:
fatal_interrupt,
can3_wake_handler:
fatal_interrupt,
can2_tx_handler:
fatal_interrupt,
can2_rx_handler:
fatal_interrupt,
can2_err_handler:
fatal_interrupt,
can2_wake_handler:
fatal_interrupt,
can1_tx_handler:
fatal_interrupt,
can1_rx_handler:
fatal_interrupt,
can1_err_handler:
fatal_interrupt,
can1_wake_handler:
fatal_interrupt,
can0_tx_handler:
fatal_interrupt,
can0_rx_handler:
fatal_interrupt,
can0_err_handler:
fatal_interrupt,
can0_wake_handler:
fatal_interrupt,
flash_handler:
fatal_interrupt,
eeprom_handler:
fatal_interrupt,
spi2_handler:
fatal_interrupt,
spi1_handler:
```

```
fatal_interrupt,
iic_handler:
fatal_interrupt,
bdlc_handler:
fatal_interrupt,
selfclk_mode_handler:
fatal_interrupt,
pll_lock_handler:
fatal_interrupt,
accb_overflow_handler:
fatal_interrupt,
mccnt_underflow_handler:
fatal_interrupt,
pthif_handler:
fatal_interrupt,
ptjif_handler:
fatal_interrupt,
atd1_handler:
fatal_interrupt,
atd0_handler:
fatal_interrupt,
sci1_handler:
fatal_interrupt,
sci0_handler:
fatal_interrupt,
spi0_handler:
fatal_interrupt,

// Timer and Accumulator
acca_input_handler:
fatal_interrupt,
acca_overflow_handler:
fatal_interrupt,
timer_overflow_handler:
fatal_interrupt,

// InputCapture/OutputCompare Timers
tc7_handler:
fatal_interrupt,
tc6_handler:
fatal_interrupt,
tc5_handler:
fatal_interrupt,
tc4_handler:
fatal_interrupt,
tc3_handler:
fatal_interrupt,
tc2_handler:
fatal_interrupt,
tc1_handler:
fatal_interrupt,
tc0_handler:
fatal_interrupt,

// External Interrupts
rtii_handler:
fatal_interrupt,
irq_handler:
fatal_interrupt,
xirq_handler:
fatal_interrupt,

illegal_handler:
fatal_interrupt,
cop_fail_handler:
fatal_interrupt,
cop_clock_handler:
```

```
fatal_interrupt,

// Vectors in use
swi_handler:
fatal_interrupt,

/************************************************************/
/**Add yours here:                                          */
/************************************************************/
 rtii_handler:              rti_isr,
 atd0_handler:              atd_isr,
ptjif_handler:              pj7_isr,
/* ptpif_handler:                  pp1_isr, */

/************************************************************/


reset_handler:
_start
};
#endif
#endif
```

lcd.c

```
#ifndef _LCD_C_
#define _LCD_C_
/*****************************************************************************/
#include "hcs12.h"
#include "lcd.h"

#define LCD_ENABLE 2// Strobe data into the LCD module
#define LCD_WRITE_DATA 1// Select command/data
#define LCD_SET_ADDRESS 0x80 // Write address command
#define LCD_CLEAR 0x10 // Write address command
#define SPACE_CHAR0x20
// Adjust delay timers to set scroll rates.
// These are default, use functions to set.
#define LCD_START_COUNTS5
#define LCD_CHAR_COUNTS 2
#define LCD_WIDTH 16
#define NO_SCROLL 255
#define LCD_WRITE_DELAY 250

ScrollData _sd[]={
{LCD_START_COUNTS,LCD_CHAR_COUNTS,0,LCD_WIDTH,-1,(char*)0,0},
{LCD_START_COUNTS,LCD_CHAR_COUNTS,0,LCD_WIDTH,-1,(char*)0,0},
{LCD_START_COUNTS,LCD_CHAR_COUNTS,0,LCD_WIDTH,-1,(char*)0,0},
{LCD_START_COUNTS,LCD_CHAR_COUNTS,0,LCD_WIDTH,-1,(char*)0,0},
{0}};
//===========================================================================
// This is a list of init commands that are sent to the LCD.
// We only have 4 bits - so for the first 4 commands we only send a nibble.
// Then we send two nibble, hi byte first.
//===========================================================================
byte _lcd_init[]={
0x30,  // 1st reset code, must delay 4.1ms after sending
0x30,  // 2nd reset code, must delay 100us after sending
              // all following 10 nibbles must be delay 40us each after sending
0x30, // 3rd reset code,
0x20,  // 4th reset code - we are now in 4 bit mode - start 2 pass transfers.
0x20,  // 4 bit mode, 2 line, 5X7 dot
```

```
0x80,  // 4 bit mode, 2 line, 5X7 dot
0x00, // cursor increment, disable display shift
0x60,  // cursor increment, disable display shift
0x00,  // display on, cursor off, no blinking
0xC0,  // display on, cursor off, no blinking
0x00,  // clear display memory, set cursor to home pos
0x10,  // clear display memory, set cursor to home pos
0x80,// set line1
0x00,// Set line1
};
/* the lenght of a string */
int _strlen(char *s);
int _strlen(char *s){
int i=0;
while(*(s++) )
i++;
return i;
}
/* copy one string into another */
char* _strcat(char* s1, char* s2);
char* _strcat(char* s1, char* s2){
char *s = s1;
while(*s1 ) // move to end of s1
 s1++;
while(*(s2) ) // copy s2 into s1
*(s1++) = *(s2++) ;
 *s1 = 0;
 return s;
}
/* the lenght of a string without spaces */
int _trimlen(char * s);
int _trimlen(char * s){
 int i = _strlen(s) - 1;
 s += i;
 while(*(s--) == 0x20 )
 i--;
 return i;
}
/* delay to allow data to settle in lcd controller */
void delay(int ms);
void delay(int ms){
int i,j;
for( i = 0 ; i < ms ; ++i ){
 for( j = 0 ; j < LCD_WRITE_DELAY ; ++j)
 asm("nop; nop; nop; "); // This foils optimization
}
}
/* write half a byte to the lcd controller */
void LCDWriteNibble(byte n);
void LCDWriteNibble(byte n){
n &= 0xf0;
n >>= 2;
n &= 0x3c;
PORTK &= ~0x3c; // zero out the bits
PORTK |= n; // set the bits.
PORTK |= LCD_ENABLE;// Strobe the data in
PORTK &= ~LCD_ENABLE;
delay(1);
}
/* write one byte to the lcd controller */
void LCDWriteChar( byte d );
void LCDWriteChar( byte d ){
PORTK |=LCD_WRITE_DATA;
LCDWriteNibble(d);// Write the upper nibble.
d <<= 4;// Shift the lower nibble up to the upper.
LCDWriteNibble(d);// Write the upper nibble.
}
```

```
/* lcd controller initialization (do before anything) */
void LCD_Init();
void LCD_Init(){
byte i = 0;
DDRK =0x3f; // PK[0:5] Output
PORTK = 0;
delay(100);// Wait 15ms for first command.
for( ; i < sizeof(_lcd_init) ; ++i ){
LCDWriteNibble(_lcd_init[i]);
delay(50);
}
}
byte _line_control[]={
0x00, // Line 1
0x40, // Line 2
0x14, // Line 3
0x54// Line 4
};
/* LCDWriteLine - Which line 1-4 and a string */
void LCDWriteLine(byte line, char* d);
void LCDWriteLine(byte line, char* d){
byte c = 0;
PORTK &= ~LCD_WRITE_DATA;
c = _line_control[line-1] | LCD_SET_ADDRESS;
LCDWriteNibble(c);
c <<= 4;
LCDWriteNibble(c);
c = 0;
while( *d && (c < LCD_WIDTH) ){
if(*d == '\r' )
break;
LCDWriteChar(*(d++));
c++;
}
/* Clear to the end of the line */
while( c++ < LCD_WIDTH )
LCDWriteChar(SPACE_CHAR);
}
/* External access functions -which - line # 1-4 */
ScrollData* LCDSetStartDelay( int which, word delay){
 _sd[which-1].LCDstartDelay = delay;
 return &_sd[which-1];
}
ScrollData* LCDSetCharDelay( int which, word delay){
 _sd[which-1].LCDcharDelay = delay;
 return &_sd[which-1];
}
/* Write a line to the LCD to be scrolled.
 if d= 0 then stop scolling that line. */
void LCDScrollLine( byte line, char* d );
void LCDScrollLine( byte line, char* d ){
if( 0 == d ){// No more scrolling please.
_sd[line-1].LCDstate = NO_SCROLL;
return;
}
LCDWriteLine(line, d );
// Init the scoll timers.
_sd[line-1].LCDindex = 0;
_sd[line-1].LCDdelayCounter =0;
_sd[line-1].LCDscrollData = d;
 // If it is less than the width, then don't scoll.
if( _trimlen(d) <LCD_WIDTH ){
 _sd[line-1].LCDstate = NO_SCROLL;
}else{
 _sd[line-1].LCDstate = 0;
}
}
```

```
/* LCDUpdateScroll MUST be called if scrolling is to work.
 The time constants depend on how often it is called */
void _LCDUpdateScroll(byte index);
void _LCDUpdateScroll(byte index){
if( NO_SCROLL ==_sd[index].LCDstate )// Means no scrolling.
return;
if( _sd[index].LCDstate == 0 ){// First write delay.
 if(_sd[index].LCDdelayCounter >= _sd[index].LCDstartDelay){
_sd[index].LCDstate = 1;
_sd[index].LCDindex = 1;
_sd[index].LCDdelayCounter =0;
 }
}else if( _sd[index].LCDstate == 1 ){ // Each character delay.
 if( _sd[index].LCDdelayCounter >= _sd[index].LCDcharDelay ){
 if(_sd[index].LCDscrollData[_sd[index].LCDindex] == 0 ){
// hit the end, start over.
_sd[index].LCDindex = 0;
_sd[index].LCDstate = 0;
 }else{ // Move to the next char.
 ++_sd[index].LCDindex;
 }
 // Update the lline on the LCD, shifted.
 LCDWriteLine(index+1, &_sd[index].LCDscrollData[_sd[index].LCDindex]);
 _sd[index].LCDdelayCounter =0;
 }
}
++_sd[index].LCDdelayCounter;
}
/* update the scroll on ALL the possible strings */
void LCDUpdateScroll();
void LCDUpdateScroll(){
byte i;
// For each line update the scoll states.
for( i = 0 ;; ++i){
 if( 0 == _sd[i].LCDstartDelay )
 break;
 _LCDUpdateScroll(i);
}
}
#endif
```

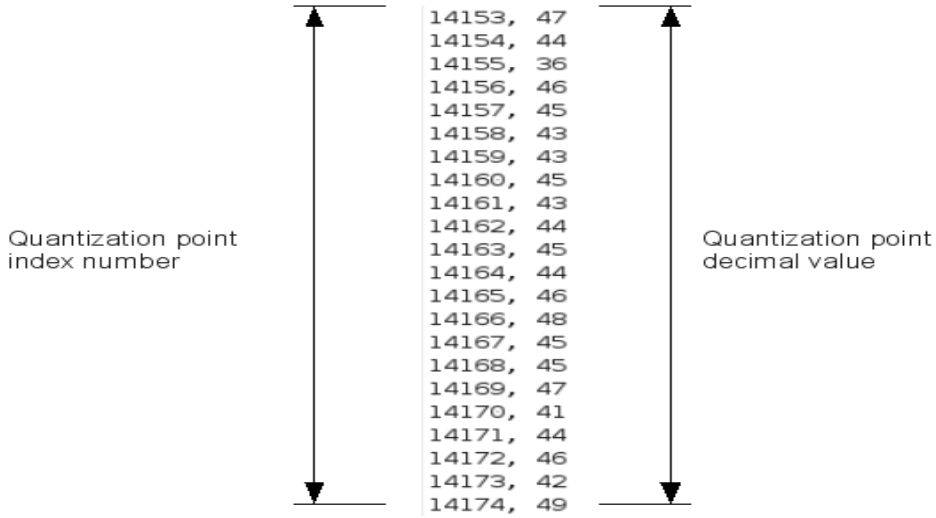
startup.c

```
#include "hcs12.h"

/*
 * The crystal freq. on the DRAGON12 board (Rev D or E MC689S12DP256) is 4 MHz
 * so the default bus speed is 2 MHz.
 *
 * In order to set the bus speed high than 2 MHz the PLL must be initialized
 * The math used to set the PLL frequency is:
 * PLLCLK = CrystalFreq * 2 * (initSYNR+1) / (initREFDV+1)
 * If CrystalFreq = 4Mhz on DRAGON12 board
 * initSYNR = 5PLL multiplier will be 6
 * initREFDV= 0PLL divisor will be 1
 * PLLCLK = 4*2*6/1 = 48MHz
 *
 * The bus speed = PLLCLK / 2 = 24 MHz
 */

void __attribute__((near)) __premain (void);
void __attribute__((near)) __premain (void)
```

```
{
// in case special mode enabled, avoid conflict on PORTE
PEAR |= NECLK;

// bgnd mode stops COP and RTI clocks
COPCTL = RSBCK;

// stops TCNT counter when debugging stops
TSCR1 |= (1<<5);                      // TFRZ

// PLL
CLKSEL = 0;                           // Disable PLL to configure

// xtal 16MHz, bus 24MHz
SYNR= 5;
REFDV = 0;
while (!(CRGFLG & 0x08))      // Wait for PLL LOCK
{
ARMCOP = 0x55;
ARMCOP = 0xAA;
}
CLKSEL |= 0x80;                       // Use PLL

// init switch inputs
PERH = 0xff;                          // Pullups
}
```

C. Perl Scripts

The following scripts intake the raw data conforming the power signature received via the USB link. The output is a file containing the values for the quantization points in a Matlab or Scilab readable format. The format consist of an index number relative to the quantization point, followed by a coma and the decimal value of the quantization point. For example,



To process 8 bit resolution signatures, the following perl script was used.

```
#!/usr/bin/perl
$inputfile= $ARGV[0];
$index = 0;
$data_odd = 0;
$data_even = 0;
$data = 0;
$time = 0;

open(FILE, "<$inputfile");
binmode(FILE);

read(FILE, $buffer, 25000000, 0);
close(FILE);

foreach (split(//, $buffer)) {
@array[$index] = ord($_);
$data = @array[$index];
printf("%d, %d\n", $index, $data);
$index++;
}
```

To process 10 bit resolution signatures, the following perl script was used.

```
#!/opt/local/bin/perl
$inputfile= $ARGV[0];
$index = 0;
$data_odd = 0;
$data_even = 0;
$data = 0;
$time = 0;

open(FILE, "<$inputfile");
binmode(FILE);

read(FILE, $buffer, 25000000, 0);
close(FILE);

foreach (split(//, $buffer)) {
@array[$index] = ord($_);
if($index % 2) {
 $data_odd = @array[$index] / 64;
 $data = $data_odd + $data_even;
 printf("%d, %d\n", $time, $data);
}
else {
 $data_even = @array[$index] * 4;
 }
          $time = $index / 2;
          $index++;
}
```

D. Linux Scripts (for processing automatization)

In order to stream line the power signature processing and plotting, the following two scripts were developed. The first one automatically detects the USB interface connected and transfers control to the second one. The second script makes use of the previous perl scripts to process the raw signatures and deliver a plot on behalf of Scilab. The first script must be placed on /etc/udev/rules.d on linux machines.

```
KERNEL=="ttyUSB0", \
SUBSYSTEMS=="usb", \
ACTION=="add", \
DRIVERS=="usb" \
ATTRS{serial}=="Jonathan_Cervantes", \
ATTRS{product}=="HCS12_Digital_Signature", \
ATTRS{manufacturer}=="THESIS", \
RUN+="/usr/bin/sudo -H -u thesis sh /home/thesis/.Current_Signatures.sh"
```

The script in charge of the processing of the data is the following. The name of the file must be in accordance with the previous script. In this case the name of the file is ".CurrentSignatures.sh".

```
#!/bin/bash
file=$(date +%a_%T)
# must run some program to open the port. like minicom
# include touch in futore

cat < /dev/ttyUSB0 >| /home/thesis/Thesis/Signature-raw/$file
perl /home/thesis/Thesis/Signature-Perl-Scilab/.data_10_bits.pl \
 /home/thesis/Thesis/Signature-raw/$file > \
 /home/thesis/Thesis/Signature-Perl-Scilab/$file.perl.out
scilab -args '-f /home/thesis/Thesis/Signature-Perl-Scilab/.scilab.inst '$file'.perl.out'
exit
```

Scilab is a matrix processing software similar to Matlab. In this case, the automatization process goes one step further as the previous script calls a last one to pass all the necessary data and parameters to Scilab. The name of this file must one more be in accordance with the one given to it on the previous script.

```
cd /home/thesis/Thesis/Signature-Perl-Scilab
args=sciargs()
stacksize(20000000)
D = read(args(4),-1,2)
x = D(:,1)
y = D(:,2)plot2d(x,y)
```

Curriculum Vita

Jonathan A. Cervantes was born in Mexico city (DF). The first out of two sons from Esther A. Garcia and Jaime S. Cervantes. Jonathan was raised in several parts of Mexico through out his early years of life (Mexico DF, Villahermosa Tabasco and Ciudad Juarez Chihuahua). Ever since he was young, Jonathan was known for being avid to take apart electronic devices and appliances, righteously understanding how they work and putting them back together or fixing them. In 1999, Jonathan graduated from CBTIS 128 high school (Cd Juarez) and subsequently entered the Universidad Autonoma de Cd Juarez college for a period of one year and a half. In fall 2001, Jonathan transfered all his credits to The University of Texas at El Paso (UTEP) across the border seeking to continue a Bachelor of Science in Electrical and Computer Engineering. In May 2006, Jonathan was successful in his pursue and decided to extend his knowledge by seeking the degree of Master of Science in the same field. Currently, Jonathan reached the goal previously mentioned.

Permanent Address: 3933 Tierra Marfil Rd

El Paso Texas, 79938

This thesis was typed by Jonathan Cervantes.