Real-time efficient scaling mechanism for high speed processing of analog measurements

Yashas Nagaraj Udupa B.Eng

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's Degree Programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisors

Univ.-Prof. Dr. Marcel Carsten Baunach
Institute of Technical Informatics (ITI)

Dr. Benjamin Steinwender
KAI GmbH

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

(Ort, Datum) (Unterschrift)

Abstract

Real-time processing of data is of utmost importance in embedded systems since the micro-computing device present in the embedded system handles a tremendous amount of data coming from environmental events and internal events. The current trending micro-computing device in embedded systems is a lightweight high-performance system on a chip. This system on a chip, along with the microcomputer environments contains smart power devices for providing diagnostic and protection features.

At Kompetenzzentrum fur Automobil- und Industrieelektronik GmbH there exists a reliability test system concept for testing of semiconductor power devices called Modular Power Stress test architecture [1]. The performance of the reliability test during the design of semiconductor power devices and its system plays a vital role as in real-world, semiconductor devices are subjected to numerous stresses. These stresses are electrical stress, mechanical stress or thermal stress. In response to the mentioned stresses, several electrical signals need to be measured in the semiconductor device. Measurement and interpretation of these electrical signals are performed using a microcontroller in the Modular Power Stress test architecture. The Microcontroller acts as a controlling and measuring agent of the stress that is undergone by the semiconductor power devices. To measure the analog signals, the microcontroller makes use of a data processing mechanism. In this work, high-speed analog measurement processing concepts for a microcontroller are delivered and processing of analog measurements is automated irrespective of the type of stress tests that are performed.

Acknowledgements

I heartfully thank Dr. Prof. Marcel Carsten Baunach for being my university supervisor and for providing the directions and technical knowledge to perform the thesis. I also thank Dr. Benjamin Steinwender for providing the opportunity to carry out this research work. I carry immense pleasure for the exposure, learning, challenges and knowledge transfer that I have received during this work. I obtained the right freedom, encouragement, criticism, and guidance to do this project.

I further thank my colleagues of KAI GmbH and Embedded Automotive Software Graz, ITI, Graz University of Technology, who supported in multiple ways that kept me progressing in my thesis. I thank Mr. Keerthi Datta Konanur Ramanna for assisting me in my academics and also providing helpful insights many times. I most prominently thank my family for their moral support, encouragement, and lessons that have led me to be persistent in my endeavors. Finally, I would like to thank everyone who has supported me to pursue my studies.

This work was funded by the Austrian Research Promotion Agency (FFG, Project No. 874907). The work in this thesis has been sponsored by KAI GmbH and has been carried out under the joint supervision by KAI GmbH and Embedded Automotive Software Group, ITI, Graz University of Technology. Their support is hereby greatly acknowledged.

Contents

List of Figures

List of Tables

1 Introduction

System on Chip (SoCs) play a prominent role in the leap forward of today's technological evolution. The ability to perform complex functionalities within a small object created the demand for the extensive need of SoCs in almost every semiconductor industry. SoCs are leading to provide smart solutions for the dominant, real-life complex problems because of their intelligent features. To utilize the benefits of the capabilities of SoC and to ensure the operation of such micro computing environments and power devices, it is essential to predict the life span of the SoC or Device Under Test (DUT) that helps in determining the reliability to use the DUT in the real world embedded applications. The determination of reliability of DUT involves a classical procedure of introducing the semiconductor into various stress scenarios. From the obtained behavior to predict the SoCs' lasting capabilities and reliability. This is a very essential procedure for semiconductor manufacturers during the development and release of any new semiconductor products. Various types of stress tests [2] are

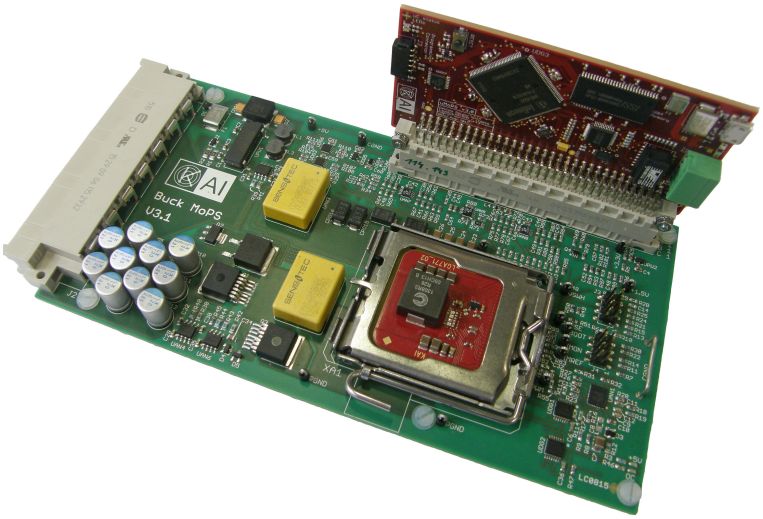

Figure 1.1: MicroMoPS, BuckMoPS and LGA771 arrangement.

performed to extract the characteristics of the DUT. These stress tests obey a flexible test approach called the Modular test system. This test system is split into two instances, namely the host system and the hardware module. The hardware module is termed as MicroMoPS in this test environment. The host system is named as Software Architecture for MoPS (SAM) and controls the overall test flow and communicates with the MicroMoPS[1]. Many MicroMoPS units may be connected to the host computer via an Ethernet network. The host computer forwards the stress pattern to the MicroMoPS and receives preprocessed (digitized and filtered) measurement data application and status information. The inclusion of the application module between MicroMoPS and the stress subjected semiconductor device is for integrating the test circuit, to monitor vital parameters of DUT during test runs as well as to provide protection circuits to avoid catastrophic destruction of the test setup in case of a DUT failure. Each application module is connected to one MicroMoPS, which controls the application test, performs measurement data acquisition and logs device status information. The application modules are tailored to an individual type of test. In this work, scaling functions such as Reverse Polish Notation (RPN) and linear scaling are incorporated into the data processing module of MicroMoPS for time and memory-efficient processing of analog measurements. Measurement circuits are analyzed to deduce the linear scaling equation that fits for the data processing system of the Modular test system. Also, the processing of data is automated via SAM regardless of the type of stress test application that is performed. The attainment of high speed processing of analog measurements and the automation of processing of data are investigated on a particular stress test type called a Low Voltage stress test system. The application module and DUT that are associated (Figure 1.1) to the Low Voltage stress test system are termed as BuckMoPS and LGA771 respectively.

1.1 Problem Statement and Motivation

In this sophisticated test environment, the stress exertion on to the DUT is achieved by running so-called *test plans* in MicroMoPS. This test plan[3] is created by test engineers. When creating a test plan, it is necessary to initialize all the hardware parameters and test parameters. Also, the calculation of measurement related scaling parameters is described in the test plan. This method of manual scaling is error-prone and the scaling values may vary based on the type of stress test that is performed. Semiconductor

[1]MoPS stands for Modular Power Stress

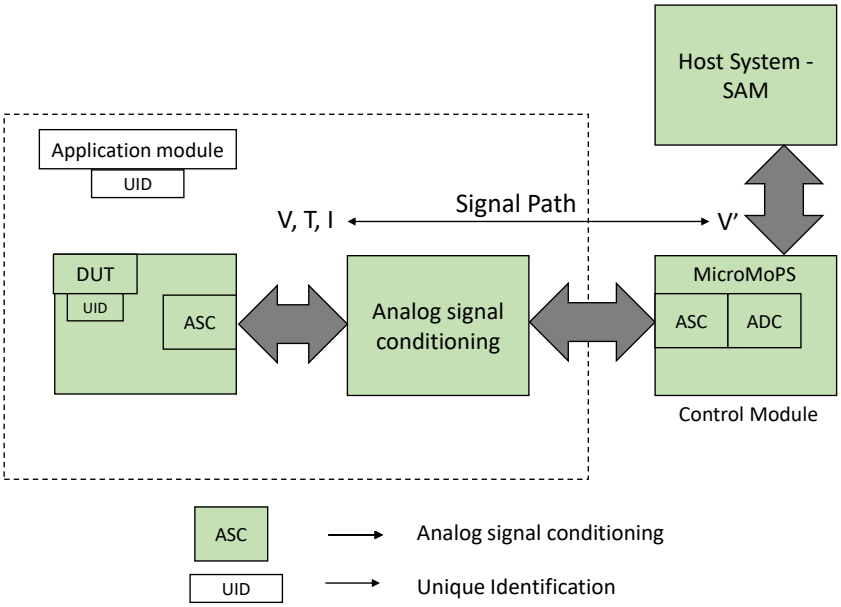

Figure 1.2: Measurement environment.

devices are subjected to various application specific stress tests [2] such as *power cycling, automotive repetitive short circuit testing, and inductive clamping*. In order to avoid manual scaling, the procedure of scaling needs to be automated. This way, writing the test plan is simplified and human error possibilities are much reduced. Using the SAM [1] and the Unique Identification (UID) of application module and DUT module, the possibility of automation of data processing is identified. Another concern is that, the data processing system of MicroMoPS uses a standard scaling mechanism to process the analog measurements [2] into MicroMoPS operating voltage values. But, from the analog signal conditioning circuit (see Figure 3.13) that lies between the source of the analog to digital converter of control module and source of the analog signal at the application module, there is a scope of deriving a linear scaling equation that processes the digital data in the control module. Introducing linear scaling over standard scaling function improves the speed of processing of analog measurements that are generated on SoC. Also, in a few other scenarios, the scaling function need not have to be linear because of the complex analog signal conditioning circuit that could exist in the measurement environment. To address this scenario, a memory-efficient mathematical notation called RPN is introduced to the data processing system of the control module. Hence, the linear scaling mechanism and RPN mechanism are investigated for scaling in this work.

1.2 Research Questions

- Is it possible to incorporate a robust scaling function to boost the performance of measurement data processing?
- Is it possible to derive a linear scaling equation in the test environment that involves complex measurement circuits?
- Is it possible to automate the method of scaling in the test environment?

1.3 Thesis Structure

The rest of the thesis is structured as follows:

1. **Chapter 2** provides a brief overview of the software tools, programming languages, and hardware peripherals that are used in this work.

2. **Chapter 3** presents the underlying concepts of the Modular stress test environment, hardware and software modules in the test environment, measurement data acquisition and processing concepts of MicroMoPS.

3. **Chapter 4** explores the milestones of this project, analysis of the measurement circuit and the approach at which the milestones are achieved.

4. **Chapter 5** explains the performed software implementations concerning to the mentioned milestones.

5. **Chapter 6** describes the results that are obtained from performing this work and some analysis with the existing systems.

6. **Chapter 7** concludes the research work of this thesis and provides some recommendations for future works.

2 Fundamentals

To perform this thesis, the pre-existing firmware image of MicroMoPS and the host application software of the MicroMoPS are extended. The extension of the Micro-MoPS firmware is achieved by utilizing the associated Integrated and Development Environment(IDE), Digital Application Virtual Engineer (DAVE), and the extension of host application software is achieved by developing applications in the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) Actor framework of the host application. The overview of the relevant software tools, programming languages, hardware peripherals, and scripts that are associated with this work is provided in the following sections:

2.1 Programming languages

2.1.1 Embedded C

This is a programming language which is used most widely to develop microcontroller based applications (low level and high level). The reason for the usage of Embedded C [4] is that programs for embedded systems become more complex. The complexity arises from the processor operation where the processors are bound to perform complex functionalities such as analog signals analysis, processing of analog signals by applying filtering algorithms, low level I/O operations, fixed-point operators, usage of different memory spaces. Syntactically and semantically, Embedded C inherits concepts from standard C along with some real-time programming concerns such as dynamic memory allocation and de-allocation, mutexes and semaphores.

2.1.2 Lua

Lua is a lightweight, high-level programming language, written in C and utilizes a minimum of RAM while still performing as well as expected. It is designed to be used as an extension language, which means Lua has no idea of a "main" function. Instead, it can be incorporated into any C based program to enhance its functionality. Lua is therefore majorly used for providing customizable applications where for the complex applications, macros and scripts may be integrated using Lua. Lua is not very verbose but a very expressive language. The example of a Lua program is as shown in Listing 2.1

Listing 2.1: Lua example

```
function swap(a, b)
        return b, a
end

a, b = swap(200, 300)
```

As it can be seen from lua example code Listing 2.1, functions can return multiple values. Lua is a dynamically typed language: variables do not have types; only values do. Lua also supports object-oriented programming.

2.1.3 LabVIEW

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a graphical programming environment used for testing, measuring and controlling of any computer based data acquisition. This software offers a Virtual Instrument (VI) component, which enables users to create their own application in the form of VIs and this application could be interfaced with the microcontroller to acquire, process and store the data. Applications in LabVIEW are usually built using well-known design patterns [5] which allows the developed VIs to appear much more organized and flexible to provide software extension. LabVIEW uses the concepts of object-oriented programming such as classes, inheritance, and encapsulation. These concepts provide advantage to LabVIEW to build any software application since the VIs which are created using these concepts help to modify the code in respective VIs and the modification does not affect the other sections of code. A powerful feature of LabVIEW is the Actor framework. The Actor Framework [6], [7] is used to design and build scalable multi-actor systems [8] to solve problems requiring a high level of concurrency.

2.2 Hardware peripherals

2.2.1 I2C

I^2C communication protocol [9] is used in communication between multiple slave devices and one or more master devices. It is a serial communication interface used for short distance intra board communication. The data exchange using I^2C must adhere a certain protocol for valid I2C communications. Data transfers are achieved using a data line Serial Data Line (SDA) and a clock signal Serial Clock Line (SCL). The bus states that are communicated using the I2C protocol are:

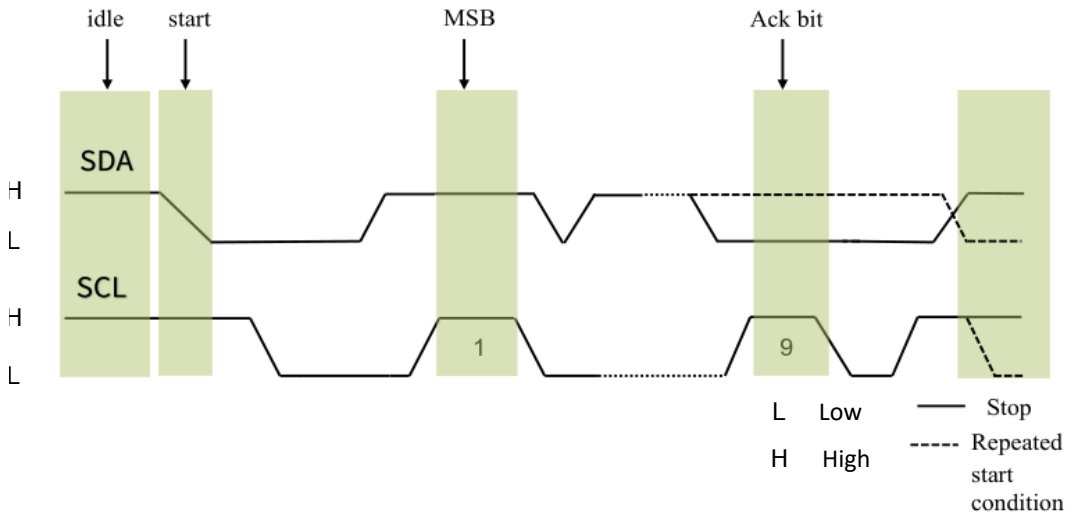

Figure 2.1: I2C protocol overview.

Idle Bus is idle when both, SDA and SCL, are inactive, i.e they are logic HIGH.

START START condition is initiated when there is a change in the state of SDA from HIGH to LOW while SCL is still HIGH.

Repeated START Repeated START condition is initiated when there is a transition of SDA from HIGH to LOW while SCL is still HIGH. The master uses this condition to repeat the transfer of data immediately at the end of the data transfer.

MS bit The data that the master is intended to transfer is sent in accordance with the generation of SCL pulses. The data on SDA remains unchanged during the entire high pulse of SCL; Transitions of SDA occur only during the LOW state of SCL. Subsequently, data is sent to a receiving device bit wise during the rising edge of SCL. As shown in the Figure 2.1, the Most Significant Bit (MSB) of data remains unchanged during the first clock pulse of SCL, it follows that the MSB is shifted into the receiving device during the rising edge of the SCL pulse. Likewise, the data is sent bit wise until the Least Significant Bit (LSB) of the data is received at the receiver.

Acknowledge bit For every byte data transmission, an acknowledgement bit must be received at the master. The master must generate a separate clock pulse associated to the reception of an acknowledgement bit.

STOP A STOP condition is initiated when there is a change in state of SDA from LOW to HIGH while SCL is still HIGH.

Using I^2C communication, data can be read by a master device from slave device. DS28CM00 [10] acts as a slave in this work. DS28CM00 is a UID chip that contains UID of a slave device.

2.2.2 UDP

User Datagram Protocol (UDP) [11] is a Transport Layer protocol [12] used for low-latency connectionless communication in IP-based networks. Communication is achieved by transmitting information in the form of datagrams from source to destination without verifying the state of the receiver. UDP provides two services such as the port number and checksum capability. Port numbers are used to distinguish different user requests and checksum is an optional feature that is used to detect errors in the data that has arrived.

A UDP header has four fields, each of these fields is of length 2 bytes. The fields of UDP are:

- **Source port**: The Source port represents the port of the sender.

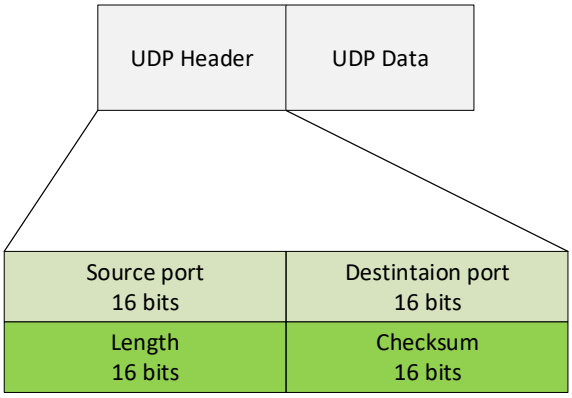

Figure 2.2: User Datagram Protocol.

- **Destination port**: The Destination port represents the port the datagram is addressed to.
- **UDP length**: The UDP length represents the length in bytes of the UDP header.
- **Checksum**: The Checksum is used for checking error.

UDP, unlike other transmission protocols, does not use handshaking dialogues to provide a guarantee that data delivers to its destination. However, it has very low overhead. UDP works by encapsulating data inside the header field and these data are sent as packets to their destinations. UDP protocol is most commonly used as a basic protocol in client-server application protocols such as TFTP, DNS, etc.

2.2.3 Analog-to-digital converter

Analog-to-digital converter (ADC) is an electronic component [13] that converts an analog electrical signal (mostly voltage signal) into a digital value. Analog signals are the continuous-time and continuous-amplitude signals that could mean physical quantities such as sound, light, temperature, and motion.

Microcontroller consists of an ADC for the conversion of these physical quantities into digital values because microcontrollers can only work with discrete values. ADC follows certain signal processing concepts as shown in the Figure 2.3 in performing the conversion of analog signals into digital values.

Figure 2.3: Sampled Signal.

Signal sampling Sampling of a signal is performed in order to reduce the continuous-time signal to a discrete-time signal where, any physical quantity such as sound wave or light wave is converted into a sequence of samples. Sampling within an ADC obeys a fixed sampling rate that depends on the input signal, known as sampling frequency (f_s). The sampled signal (see Figure 2.4) is used in the digitization of its every slice. The digitization follows a fundamental principle known as the *Nyquist theorem* [14] to successfully construct a digital signal from the input signal caused by sampling. According to the Nyquist theorem, the sampling frequency needs to be at least twice the highest analog frequency component (f_{max}) of the input signal in order to construct a digital representation.

The equation that represents the Nyquist criterion is given by:

$$f_s >= 2f_{max}$$

Resolution The ADC's resolution is the smallest change in voltage that can be detected and thus causes a change in the digital input. This change is also known as the step size of the ADC. The resolution (N) of the ADC can be determined by a bit length that is specific to the ADC. An ADC which has 'n' bit digital output provides 2^n digital values (resolution), i.e.,

$$N = 2^n \tag{2.1}$$

For example, if the digital output is of a bit length of 12-bit for an ADC, the resolution of this ADC is 2^{12}, i.e., 4096. This also means that for ADC with the resolution 4096, the digital values for each sample of an ADC range from 0 to 4095.

Scaling The processing of the digital data into microcontroller specific discrete voltage values is known as Scaling. Scaling is performed by using the bit length, digital

data, and reference voltage information. The standard scaling function of an ADC is given by:

$$f(x) = \left(\frac{V_{max} - V_{min}}{N}\right) \cdot d - V_{min}, \tag{2.2}$$

f(x) = Scaling function,
V_{max} = Maximum analog voltage to be converted to digital output,
V_{min} = Minimum analog voltage to be converted to digital output,
N = Resolution (see equation 2.1) of the ADC,
d = digital value.

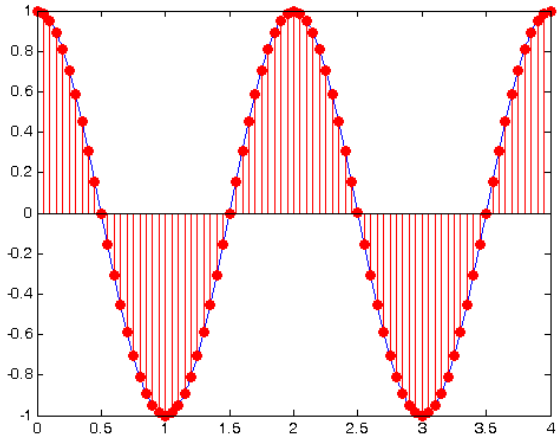

Figure 2.4: Sampled Signal.

2.3 Tools

2.3.1 DAVE

DAVE [1] is an Eclipse based IDE using the GNU C-compiler, designed for firmware development of ARM-based 32-bit XMC Infineon processors. Using the C/C++ language, firmware is developed. DAVE includes the building blocks of software development such as code editor, compiler, and debugger. The configuration wizard of DAVE provides an overview over the hardware peripherals, control units, and modules. DAVE provides a graphical user interface and wizard, which gets easy for the beginner to

[1] https://www.infineon.com/dave

get acquainted with the development tool. DAVE's interactive user interface provides a configuration window which allows the designer to select and configure a specific product and then automatically generate system initialization code for that product, including its core, memory, peripherals, driver functions, and interrupts. In DAVE, user-specific functionality can be added to the automatically generated code without having overwritten the parts when applying further changes to the microcontroller configurations.

2.3.2 Git

Git [15] is a distributed version control tool that supports distributed non-linear workflows. It is designed for facilitating the co-ordination between the connected programmers in developing software and, track or reverse the changes that are done in the development stages.

2.4 Markup languages for data interchange

2.4.1 JSON

JSON [16] is short for JavaScript Object Notation, is a standard file format used primarily for serialization and de-serialization. Serialization converts an Object-oriented programming's (OOP) object into a JSON string and de-serializing is to do the inverse operation. It is represented in key-value pairs and is used widely in web applications to generate and parse data.

3 Related Work

This chapter describes the underlying concepts that are involved in this sophisticated stress test environment and their correspondence in meeting the objectives of this work. This chapter explains the stress test procedure that is developed and incorporated at KAI, hardware and software systems that are used for the stress test architectures, the measurement Data AcQuisition (DAQ) system of MicroMoPS, the scaling mechanism that exists for processing the acquired measurement data, communication channels' specific analog signal conditioning circuits and their role in the acquisition of measurement data.

3.1 MTS architecture

At KAI, there is a pre-developed modular, flexible and adaptable test system architecture incorporated for reliability stress testing of power semiconductors, which is called Modular Power Stress (MoPS) architecture [1]. This stress test system as shown in Figure 3.1 follows a certain flow in delivering a smart approach of executing a reliability stress test on DUT to estimate their lifetime. In this stress test system, multiple DUTs are subjected to stress test patterns under automotive environment conditions. The MTS architecture is split into two parts, namely the host computer and control module (MicroMoPS). The host computer is the central unit system which controls the overall test flow and communicates with the control modules. The host computer forwards the test patterns termed as test plans generated by Test Plan Builder [3]. Based on the test patterns that are received by control module from a host computer, signals are generated by the control module to provide stimuli to the semiconductor device under test. Thereby, the DUT undergoes stress as mentioned in the test pattern. The control module further senses the output responses and closes the control loop from a PI controller integrated with the application module, which leads the control module

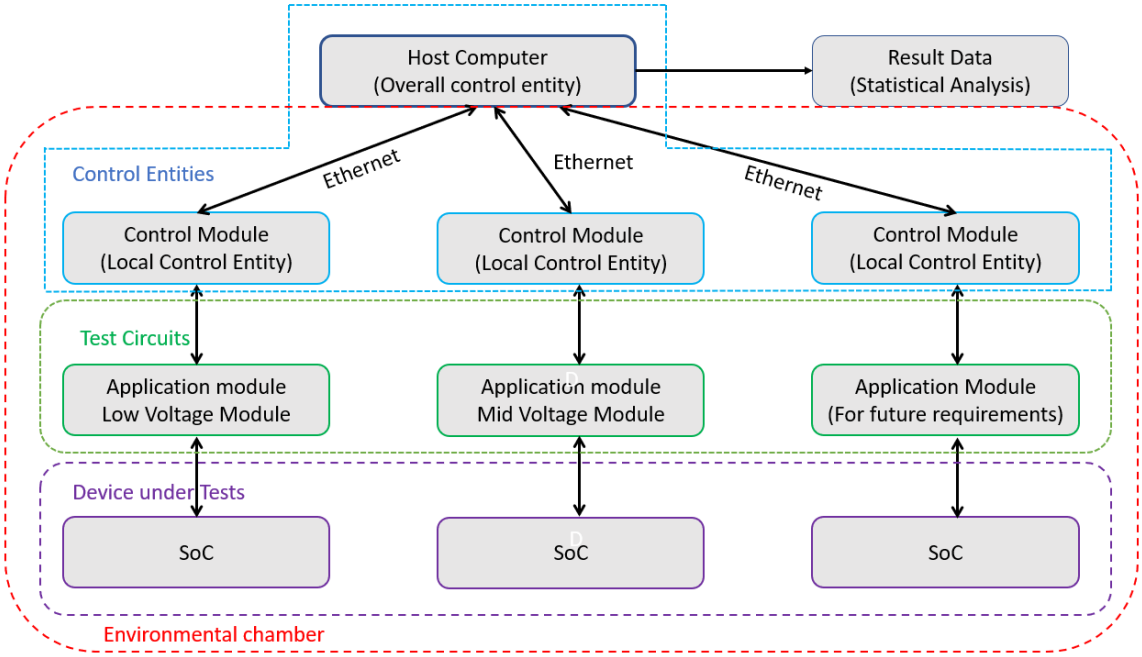

Figure 3.1: MTS architecture.

to start recording the measurement parameters of the test. The key advantage of MTS architecture is that there is a total separation of control and measurement part of the test system from the test circuits. The test circuits are built within the application module and further, the application module is connected to a semiconductor device under test by a special connector. Thus, by the integration of application modules to control modules, it is possible to subject DUTs into various types of stress tests. In order to meet reliability stress testing of power semiconductors under automotive environment conditions, multiple DUTs and local control modules are attached to the application module and placed inside an environmental chamber as shown in Figure 3.2. These application modules are designed for subjecting DUTs into specific types of stress test.

Currently, the types of stress tests are:

1. Low Voltage Test System: A board called the Buck MoPS is chosen as a Point-of-Load (PoL), which is a DC-DC power converter application. The DUTs such as power switches are subjected to an application-specific stress test called *power*

Figure 3.2: Climate chamber that exerts stress on semiconductor power devices.

cycling. DUTs are heated to reach a temperature between 85 °C and 125 °C for consumer devices, 150 °C for automotive devices and are supplied with a supply voltage that is set to the maximum level which still complies the datasheet specification of DUT. Intermittent electronic loads are submitted to DUTs which toggle between a high load of 100% that nearly reaching DUTs to their maximum operating temperature and a low load of 10 % [2].

2. Mid Voltage Test System [17]: Board such as the PoL converter is chosen as an application module. The DUTs are power transistors which are subjected to medium voltage stress up to 600V.

3. High Voltage Test System: Devices such as Insulated Gate Bipolar Transistor and Metal-Oxide Semiconductor Field Effect Transistor (MOSFET) transistors are tested by subjecting to a high voltage of up to 1.5 kV.

The analog measurement data from a DUT is acquired by the data acquisition system of MicroMoPS via the dedicated application module communication channels. The analog measurements [17] such as voltage, current and temperature signals are acquired

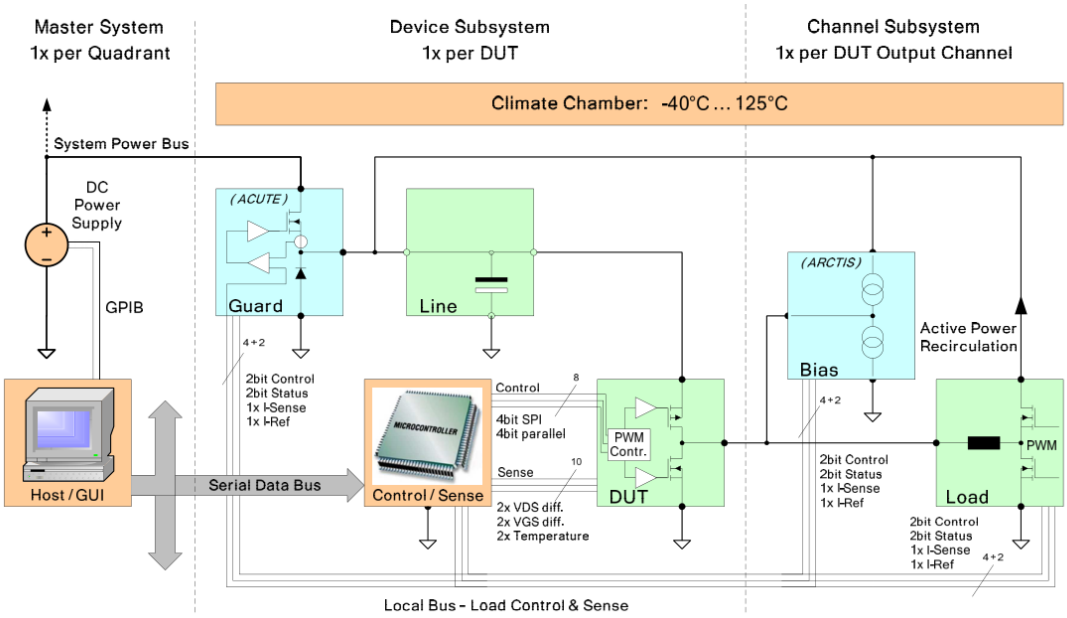

Figure 3.3: Modular Power Stress Distributed system Architecture [18].

and go through several stages of attenuation at the application module because of the presence of signal conditioning circuits. These signal conditioning circuits are meant for conditioning the measurement data signals into control module specific operating voltage ranges (0 V to 3 V).

3.2 MoPS Distributed System

The Modular Power Stress (MoPS) is a system architecture concept [1], [18] developed to provide a modular infrastructure for customizable stress test applications. The most essential components of MoPS system architecture as shown in Figure 3.3 are:

Device subsystem The components other than control module that are present in the device subsystem act as supporting elements to perform a stress test on a semiconductor device under test. For example, a guard module protects the DUT from damage by shutting down the power from the control module when there is a device failure. The control module acts as an intermediate agent to trigger application specific stress tests on the DUT. The control module is situated close to DUT to establish nearly a lossless one-to-one communication. From the stress

tests, the electrical signals that are generated in the DUT module are measured in the control module via communication channels of the control module.

Master system This is a centralized host, single control entity (see Section 3.2.2) which forwards the stress pattern created by test engineers to a control module (MicroMoPS), to execute stress on the DUT module.

Serial Data Bus This component helps to establish back and forth stress test associated communication between host computer software and the control module. It is also called global bus. Some of the mandatory data exchanges that occur between the host computer software and the control module include:

1. Reception of test plan by control module from SAM.
2. Reporting of status and measurement results from control module to SAM via Ethernet.

Channel subsystem The stress test correspondent electronic circuitry namely ARCTIS is present in this subsystem. An electronic load is driven towards the DUT via channel subsystem. Also, the real-time controller is part of this system which provides closed loop control logic for the acquisition of the analog measurements.

3.2.1 Hardware architecture

This section describes the hardware modules of MoPS distribution system.

MicroMoPS

The Control module that is used for testing the semiconductor device under test is called MicroMoPS. It is a XMC4700 [19] microcontroller based on the 32bit ARM Cortex-M4 processor core. The following are the hardware features that are available in MicroMoPS:

- Ethernet communication supported with automatic calculation of Medium Access Control (MAC) address deduced from the unique identification number (UID) of the XMC Micro Controller (µC).

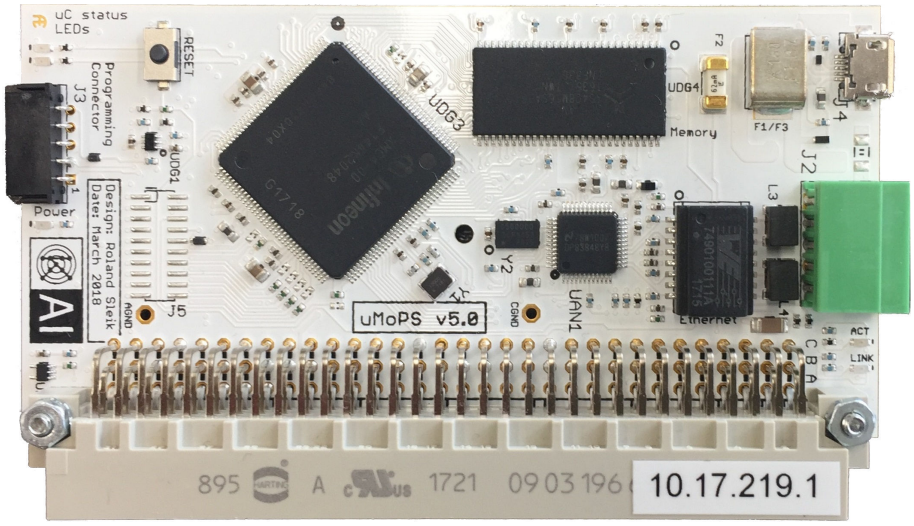

Figure 3.4: MicroMoPS

- 4 analog input modules capable of acquiring up to 24 input signals with a resolution of 12 bits at rates up to 1.8 MHz.
- 4 analog output channels with a 12 bit resolution.
- 4 Delta-Sigma Demodulator (DSD) channels to gather analog data from galvanically isolated sensors.
- 6 Pulse Width Modulation (PWM) outputs are divided into 6 units, in which 2 units are provided with an inverted output for half-bridge control.
- 1 Serial Peripheral Interface (SPI) for communicating to a variety of peripherals, such as real-time clocks, memory and control devices such as DAC.
- 1 Inter-Integrated Circuit (I^2C) for communicating at low-speed to a variety of peripherals. A Key feature of I2C is the capability to control the network of device chips with two General Purpose Input Output (GPIOs) pins.
- 2 Light Emitting Diode (LEDs) to indicate run and error states.
- 12 GPIO pins.
- Synchronous DRAM of size 8 MB to store analog measurement data.
- Ethernet and Universal Serial Bus (USB) communication interfaces.
- Up to 4 software timers of 16-bit width with 1ms resolution for triggering custom events in the test Finite-State Machine (FSM).

BuckMoPS

BuckMoPS is the application module which carries the control module - MicroMoPS and DUT - a power switch (IC) to indulge into a low-voltage application stress test system. BuckMoPS is a DC-DC power stage application. The reason behind the addition of an application module together with a control module in the test environment is that it provides a clear separation of stress application from DUT and control module. Also, by introducing an application module into the test system, the DUTs are subjected to application-equivalent conditions, which helps in the exclusive measurement of power semiconductor device parameters. The measurement parameters which are described for the low voltage application stress test system are: input voltage or input current of the DUT, output voltage or output current of the DUT and temperature of the DUT [17].

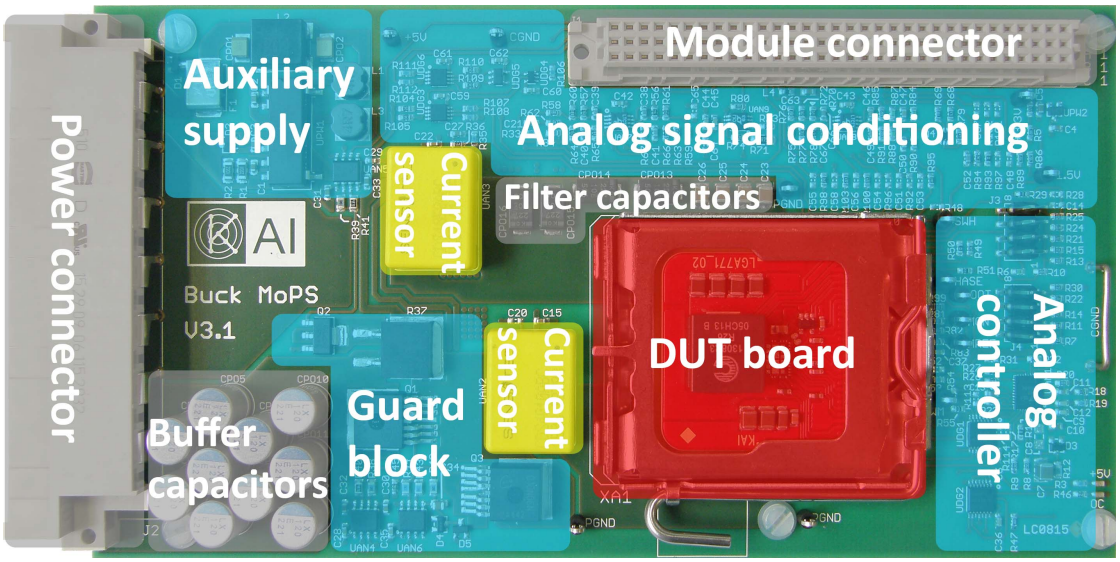

Figure 3.5: Low Voltage application board - BuckMoPS.

Auxiliary supply: This module is meant for supplying power to the control module and application module.

Guard Block: This provides smart protection functions that are used to protect the DUT board from over-voltage, which would avoid the damage of the device.

Analog controller: This component plays a role in providing a suitable control logic to drive and measure test on DUT board.

DUT board socket: This is the area where the DUT board is held and tested.

Analog signal conditioning: This component consists of Op-Amps to amplify the signal received from the control module. Op-amps are used for conditioning the analog measurement signal i.e. to attenuate analog differential voltage signals into single-ended voltage values that lie within the control module's range.

Power connector: This is the medium by which the power is supplied to application module.

Buffer and Filter capacitors: These are useful to support the power supply at high transient events.

Module connector: This is the connector that is used to interface control modules.

Current sensors: This component is useful for input and output current measurement.

Device under test

The DUT is a pair of MOSFETs transistors connected as shown in Figure 3.6. The Low Voltage application board i.e. BuckMoPS consists of a mentioned DUT. These DUTs internally have passive inductors and capacitors to achieve application-equivalent behavior. It requires only a PWM signal from the control module for the operation of DUT.

3.2.2 Software architecture

This section describes the software layer of MoPS distribution system.

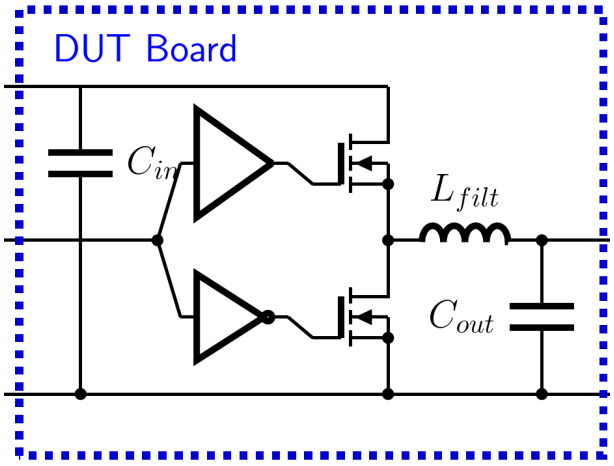

Figure 3.6: DUT board plugged to Low voltage application board - BuckMoPS.

MoPS-CORE Microcontroller Firmware

The MoPS-CORE Microcontroller Firmware is a firmware version of XMC-based microcontroller hardware targets with some CPU resource limitations and memory constraints. The major handlers that essentially run the entire operation of the MicroMoPS microcontroller are the ones that are included within the main loop. By means of measure_time() handler the time of the main loop is measured for statistical and investigation purposes. The handler comm_handle_msg() is used in communication between the internal modules of the microcontroller and also to exchange information between the host application i.e. SAM via Ethernet. The handler check_uplink() is used in testing the connection status between the host and the control module before the test plans are executed on the control module. Furthermore, the guard_feed() is a handler that provides a watch-dog guard mechanism, which is a hardware timer capable of resetting the microcontroller, unless it is periodically reset by the software. Through the led_active() handler, the status of the controller is easily detected where there are two LEDs that are dedicated to give visual indicators to the user about the working state of the controller software. Finally, the firmware runs the FSM handler that is developed by test engineers as test procedures and the Lua code that is part of FSM diagram is executed to interface the hardware modules of the microcontroller.

Listing 3.1: MoPS test loop

```
void test_loop(void) {
        test_identify_boards();
```

```
        while (1) {
                measure_time();
                comm_handle_msg();
                check_uplink();
                handle_fsm();
                guard_feed();
                led_active();
        }
}
```

There are three essential components that are associated to MoPS-CORE Firmware, they are:

Electronic Data Sheet The Electronic Data Sheet (EDS) is the configuration information of control modules that are necessary for performing tests. This configuration information includes Hardware version, peripheral information, and device scaling parameters. Fundamentally, the EDS is a JSON string, which is generated upon microcontroller booting. The microcontroller firmware uses the pre-defined compiled hardware configuration to generate the JSON string. After the generation of EDS, the same is uploaded to the MoPS project web server by SAM so, that other MoPS applications can access the information.

MoPS web server The MoPS web server is the server software dedicated for storing Hardware information of the MicroMoPS and delivering Hardware information of the MoPS microcontroller to the corresponding host software application such as TestPlan Builder.

Lua interpreter The Lua interpreter [20], [21] is to provide flexibility to the hardware designers and product engineers of the company in configuring a test sequence directly without having to deal with the low-level programming of microcontrollers. A set of Lua commands are available to provide a hardware interface and they are simple to use. These Lua commands contain the implementation of custom modules and hardware access routines. They access hardware modules via defined C-APIs. Each Lua command (enclosed within FSM) present in the test plan accesses it's corresponding C-Application Programming Interface (API) from the Lua space of MicroMoPS firmware and thus, allows the MicroMoPS to execute the entire test plan.

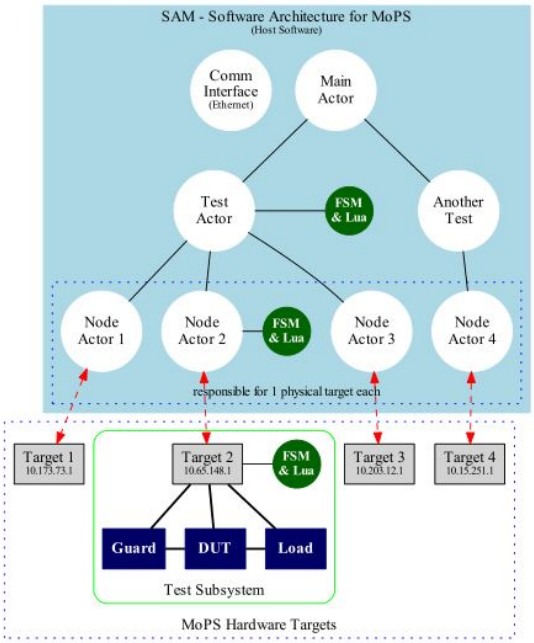

Figure 3.7: Software Architecture for MoPS

Software Architecture for MoPS

Software Architecture for MoPS is a host layer software [1] developed using LabVIEW. This software is designed for test engineers to perform any type of stress testing from their respective host computers. The stress tests are applied by loading a test plan into SAM software. This software makes use of the Actor framework [6] of LabVIEW to essentially create multiple independent software agents called as Actors. These actors have a separate GUI window displaying their concerned attributes. Furthermore, these actors also run some tasks in the background and accordingly their corresponding GUI is refreshed and displayed to provide test engineers a user-friendly interface to interact with.

SAM is organized in a hierarchical manner as shown in the Figure 3.7 where each of these software agents is meant for particular tasks that are fundamentally connected to a target system and responsible for the acquisition of measurement of electrical characteristics of semiconductor power device such as voltage, current, and temperature. SAM also facilitates in applying stress to DUT via control module and to automate the MicroMoPS related host software functionalities.

The Actors that play a major role in the entire operation of SAM are:

Main Actor This actor is initiated as soon as the SAM application begins to run and during the start-up of this application, the associated child actors i.e. log actor and communication interface are initialized. The Main actor is a root actor that is responsible for facilitating the coordination between its sibling actors using queues of LabVIEW's Actor Framework.

Log Actor This actor is used by every other actors to print and log error messages at log actor's Graphical User Interface (GUI) display window of SAM. Thereby, these messages help SAM users and developers to identify and fix bugs that arise while reconfiguring or extending the SAM software.

Communication Interface The Communication interface deals with sending messages to and receiving messages from the control modules via the Ethernet or USB interface.

Node Actor The Node actor is responsible for managing a hardware target that is connected to a host computer. In the operational case of SAM, the moment the hardware target is connected to SAM, the instance of the hardware node is created and the hardware node's GUI window is counter created to display the hardware node's corresponding attributes. The GUI of the Node Actor displays information about the node and the test. Measurement data acquired by the hardware target is also viewed in this actor.

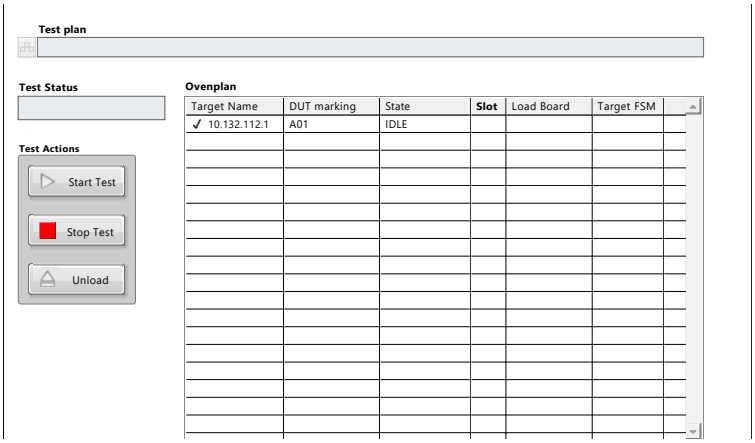

Figure 3.8: Test Actor

Test Actor Test actor provides a GUI platform to display the test status to the user and also provides the interface to send test events to the control module. GUI

display of Test Actor is as shown in the Figure 3.8. To start the test, "start" is the event that is executed. To stop the test, "stop" is the event that is executed. Once, when the test starts running, the analog measurements can be acquired by an event "meas", which is executed from the FSM window of Test actor. Test actor in much general sense executes the finite state machine given in the test plan (see Section 3.2.2).

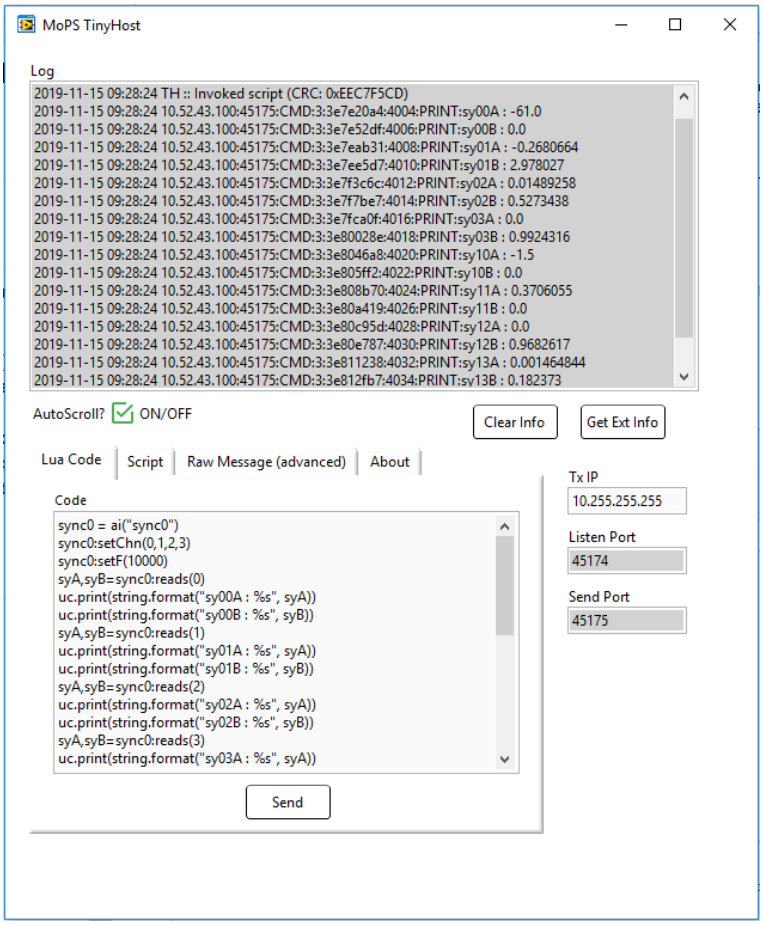

Figure 3.9: TinyHost application

TinyHost

A small application called TinyHost exists to test the communication interface of the MicroMoPS and also to send messages to the hardware targets. Through, this application the messages can be sent to the hardware targets manually and thereby,

commands that are part of Lua interpreter which are present in the microcontroller's firmware are executed. To get a sense of TinyHost's purpose, an example of one of the tests that are performed during this thesis is considered. In this work, to test the functionality of hardware target's analog input communication modules and their channels' value (see Section 3.2.3), Lua code (see Section 2.1.2) is written to invoke Lua commands that are present at the Lua space of microcontroller's firmware, as shown in the Figure 3.9. Subsequently, the result is displayed on the log field of the TinyHost as shown in the Figure 3.9.

Test Plan Builder

Test plan builder [3] is an application used to create a test plan in the form of a FSM diagram where each state of the FSM diagram is fundamentally a set of Lua commands that are defined in the microcontroller's firmware. Test plan builder also downloads the EDS from the MoPS web server to know the details of the latest hardware configuration of MicroMoPS. Consequently, from the knowledge of hardware configurations, Test-plan builder enables the test engineers to create a test plan. The development procedure of test plans and the syntactic rules that are applicable to successfully build test plans are described in detail in Plankensteiner's Master thesis [3]. EDS configuration information is necessary for verifying the respective hardware capabilities before a test starts. The Test-Plan Builder requires this information to provide knowledge to test design engineers about the available hardware and software modules that could be used in creating test plans. A sample of the test plan is as shown in the Figure 3.10. In principle, there are two sections in the layout of a Test plan tab of Test plan builder, they are:

1. **HOST:** The FSM diagram in Host is meant for providing control over the operation of the Target FSM diagram. As soon as the start button is pressed after loading the test plan to SAM, there occurs a transition from the IDLE state of Host FSM to the START state in test execution. This START state machine sends a "start" event to the IDLE state machine of Target FSM which, leads Target FSM to propagate further from the IDLE state.

2. **TARGET:** The FSM diagram in Target is the starting point for the control module to subject DUT into stress, where the functionality of Target FSM is totally controlled by the Host FSM. Intuitively, the running of all the state machines

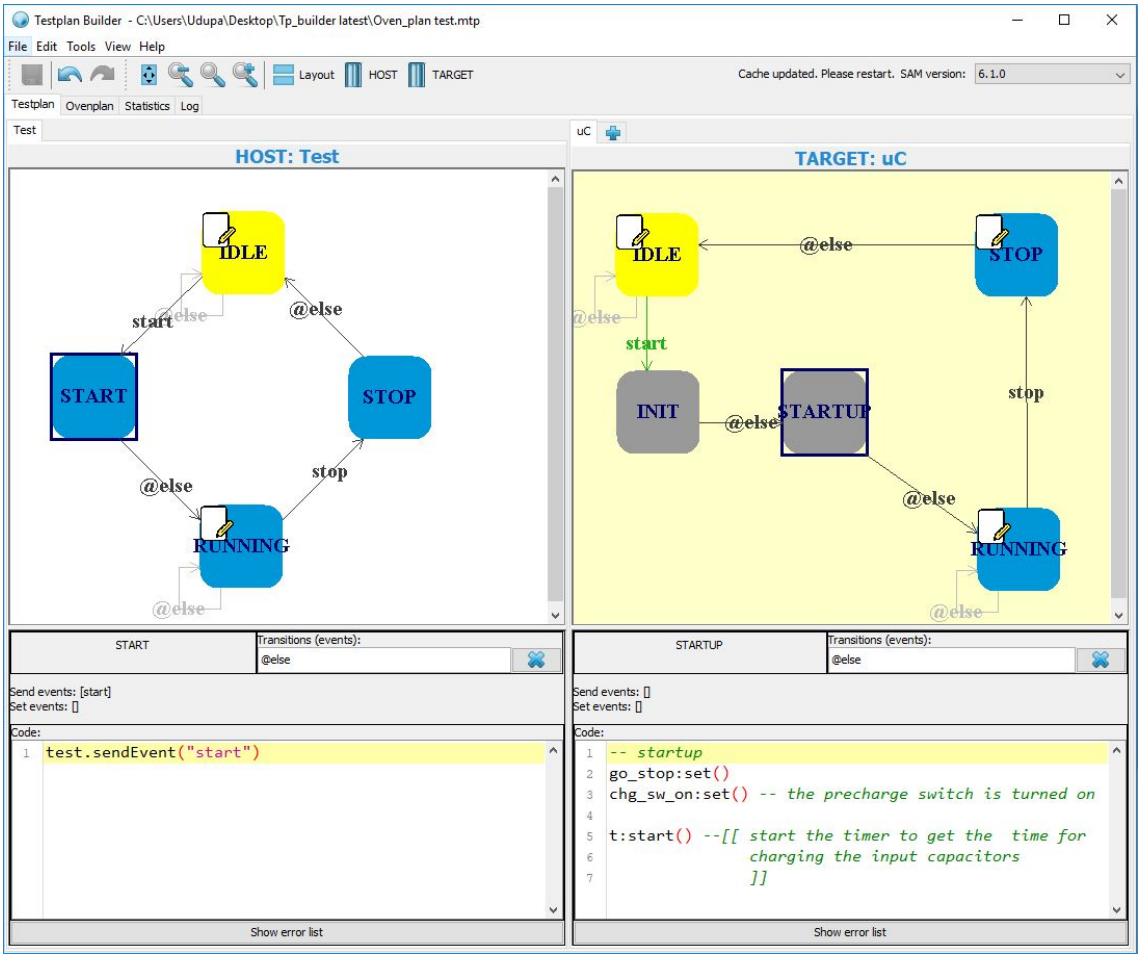

Figure 3.10: Test plan builder application

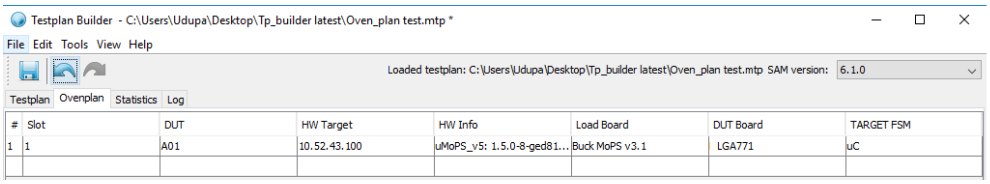

Figure 3.11: Oven plan window

of Target FSM except INIT is nothing but the invoking of Lua commands that are specific to microcontroller's firmware modules ([1], p.39). As soon as Target FSM is triggered by Host FSM by receiving a start event, there occurs a transition from the IDLE state to the INIT state in Target FSM. Consequently, at INIT a digital stimuli is generated towards the DUT because of the PWM signal from

the control module. Furthermore, Target FSM reaches to running state which, is an indication that the test is running. Finally, the test is stopped by pressing a stop button in SAM, which results in the Target FSM diagram to reach to its IDLE state from any given state.

Oven plan The oven plan of the Test plan builder tool provides an interface to select desired DUT, application module and Hardware target. Test engineers select the board names based on the stress type they want to perform. The selection of boards is done by a drop-down present in the oven plan window of the test plan builder. A variant of MicroMoPS (classified by their IP address), Application module and DUT are chosen as shown in the Figure 3.11. The test FSM diagram in the test plan window and hardware selection in the oven plan window fulfills the protocol of the creation of the test plan. Further, to which the test plan is generated as a JSON file. The JSON file is loaded to SAM to execute the stress test.

3.2.3 Measurement environment of MoPS

This section describes the measurement data acquisition system of MicroMoPS, measurements communicating channels of MicroMoPS, signal conditioning circuits present in the measurement environment and processing concepts.

Measurement data acquisition of MicroMoPS

Data AcQuisition (DAQ) of MicroMoPS is responsible for acquiring, displaying and storing the analog measurements that are obtained from tests. The analog measurements such as temperature, voltage, and current of the test are sent to Analog-to-digital converter (ADC) of the MicroMoPS via dedicated analog input channels. These measurements that are received at the ADC of MicroMoPS, are sampled and quantized to convert into a certain range of digital values. In general, the ADC of a MicroMoPS is of 12 bit resolution, therefore the analog values that are converted into digital values range from 0 and 4095 and represent the microcontroller's specific voltage values range from 0 V to 3 V. These measurements go through some signal conditioning (see Figure 3.13) before they are used by the ADC to perform digital conversion. The analog measurements that are communicated using analog communication channels are classified into two types:

1. **Single-ended multi-channel measurements:** This type of measurement requires the analog value to be referenced to the well chosen common-mode i.e. Ground.

2. **Differential measurements:** This type of measurement does not have common-mode reference but have a difference in analog value between the two input signal leads.

The communication channels which conduct analog measurements follow a certain hierarchy in their design (see Figure 3.12) and these measurements are converted from differential measurements/single-ended channel measurements to single-ended values by allowing them to go through signal conditioning. Because of this, the measurements are achieved in delimiting them into the microcontroller operating voltage range i.e., 0 V to 3.30 V.

Channels hierarchy and their purpose

Analog-to-digital conversions in MicroMoPS can be executed in two modes. They are:

1. **Synchronous mode** - Channels belonging to a synchronous group can do an analog-to-digital conversion in parallel as the ADC kernels of these channels are synchronized with each other.

2. **Scan mode** - Channels belonging to the scan group follow a configurable linear sequence that allows the dedicated channels to do analog-to-digital conversion one after another.

With respect to the modes of analog-to-digital conversion, there are dedicated channel modules such as sync0, sync1, scan0, scan1, and scan2. The mentioned channel modules as they are named, correspond to synchronous and scan conversion modes. These channel modules are further classified as follows:

1. **sync[0,1][0,3]B** - Unipolar Analog-to-Digital conversion synchronous channels, where first index, i.e, [0,1] represents the module number, second index, i.e, [0,3] represents the module's specific channel number and third index represents the channel's specific group name. The group name assigned to this channel is 'B'.

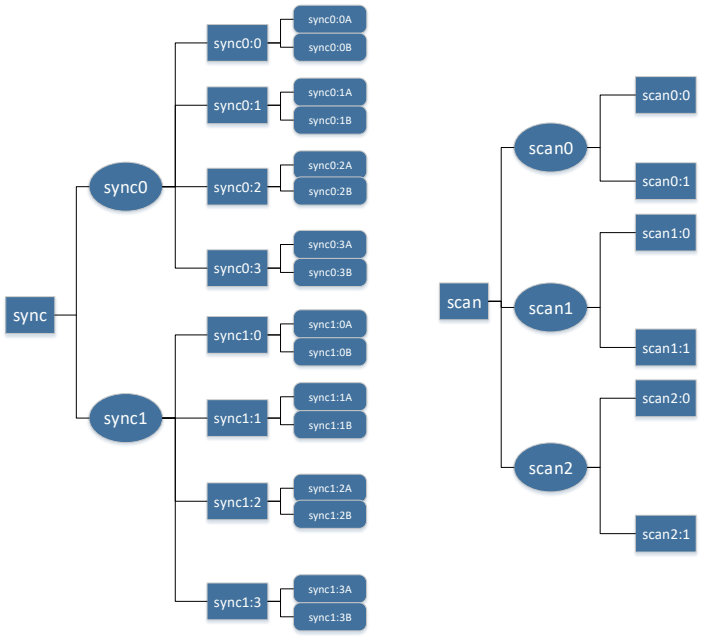

Figure 3.12: Channels hierarchy

2. **sync[0,1][0,3]A[N,P]** - Bipolar Analog-to-Digital conversion synchronous channels, where first index, i.e, [0,1] represents the module number, second index, i.e, [0,3] represents the module's specific channel number and third index represents the channel's specific group together with the polarity, i.e, -/+ of the elelctrical signal. The group name assigned to this channel is 'A'.

3. **scan[0,1,2][0,1]** - Analog-to-Digital conversion scan channels, where first index, i.e, [0,1] represents the module number, second index, i.e, [0,1] represents the module's specific channel number.

The Differential Voltage measurements such as Switch Node Voltage (Vswh), Measurement voltage from Auxiliary MicroMoPS node 1 (Aux1), IC current monitor (Imon), Converter input voltage (Vin) and Converter output voltage (Vout) are acquired via bipolar Analog-to-Digital conversion channels i.e. **sync[0,1][0,3]A[N,P]**. Aux1 and Aux2 are the auxiliary MicroMoPS that are used to cover multiple functions available on the test device.

The single ended Voltage or Current or Temperature measurements such as Converter input current (Iin), Converter output current (Iout), DUT case temperature (Tc), DUT

board temperature (Tbrd), Driver current (Idrv), Driver Voltage (Vdrv), IC current monitor (Imon) and IC temperature monitor (Tmon) are acquired via unipolar analog-to-digital conversion synchronous channels i.e. **sync[0,1][0,3]B**.

The above measurement parameters are part of the Low Voltage MTS environment [17].

Analog signal conditioning circuits

To most of the measurement signals, the analog signal conditioning circuit setting is merely the differential amplifier with a series resistor and standard voltage divider circuits convention as shown in the Figure 3.13. This signal conditioning circuit is designed by the hardware designers of KAI.

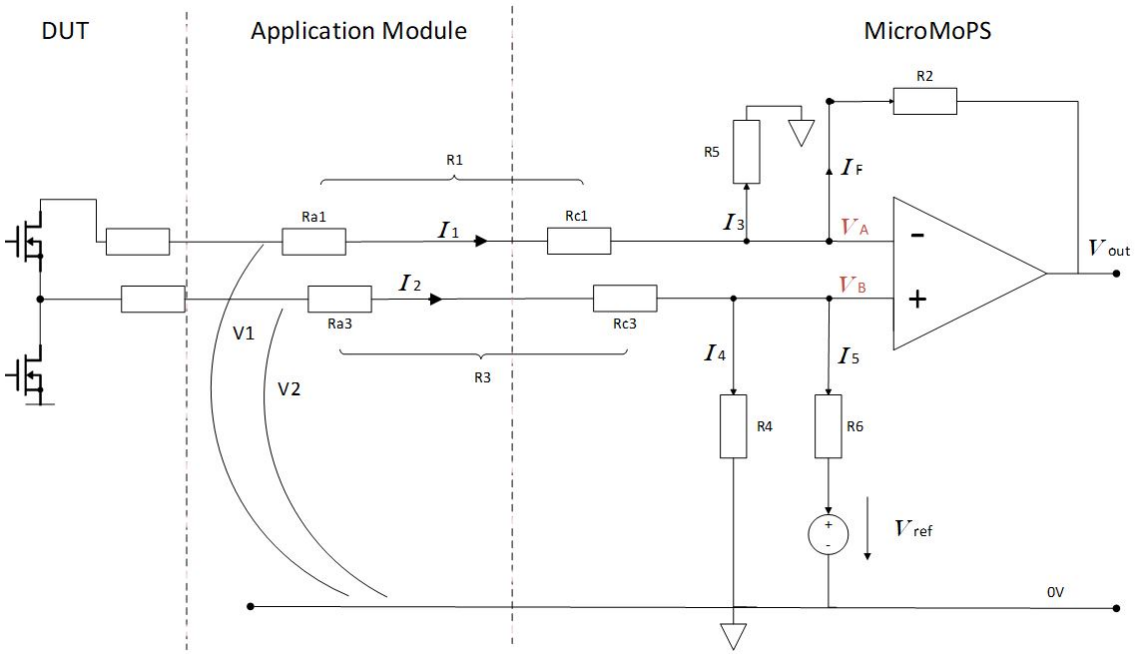

Figure 3.13: Analog signal conditioning circuit

Conditioning of the voltage, temperature, and current signals is achieved by using an analog signal conditioning circuit (see Figure 3.13) with some standard component values. The analog signal conditioning circuit for Converter input current (Iin) and Converter output current (Iout) are placed on the application module and the resulting

conditioned signal is directly fed to analog-to-digital converters of the control module. The analog signal conditioning circuit for Converter input voltage (Vin), Converter output voltage (Vout) and IC current monitor (Imon) are placed on the control module. For the DUT case temperature measurement i.e. DUT case temperature (Tcase), the resistive sensor is supplied from a constant current source with a provision of voltage amplification. IC temperature monitor (Tmon) in an application module to monitor the DUT temperature. Driver voltage (Vdrv) in the application module consists of a Voltage divider circuit to drive MicroMoPS. Finally, Driver current (Idrv) in the application module to supply current to MicroMoPS.

Processing concepts

Finally, in order to convert the digital measurements (representative of the analog measurements of the DUT) into microcontroller's specific analog values which range from 0 V to 3 V, the respective digital values need to be processed. The method of processing digital values is called a scaling function. This scaling function can vary from channel to channel because of the varying microcontroller specific analog value range for particular test measurements. For an instance, the microcontroller specific analog value range for the channel **sync[1][0]A** ranges from –1.50 V to 1.50 V whereas, the microcontroller specific analog value bandwidth for the channel **sync[0][0]B** ranges from 0 V to 3 V. Standard scaling function (see Equation (2.2)) is used to process the analog measurements.

3.3 State of the art

Stress test application on a high level goes at a particular flow. This flow essentially provides the state of the art that exists in this stress test environment. The flow of the stress test as depicted in Figure 3.14 is explained accordingly in steps as follows:

1. The starting point of the stress test application is from the control module. A control module is flashed with a stress test application integrated firmware image to execute the test plan. Upon booting of microcontroller, an EDS is generated in the MoPS CORE firmware project folder.

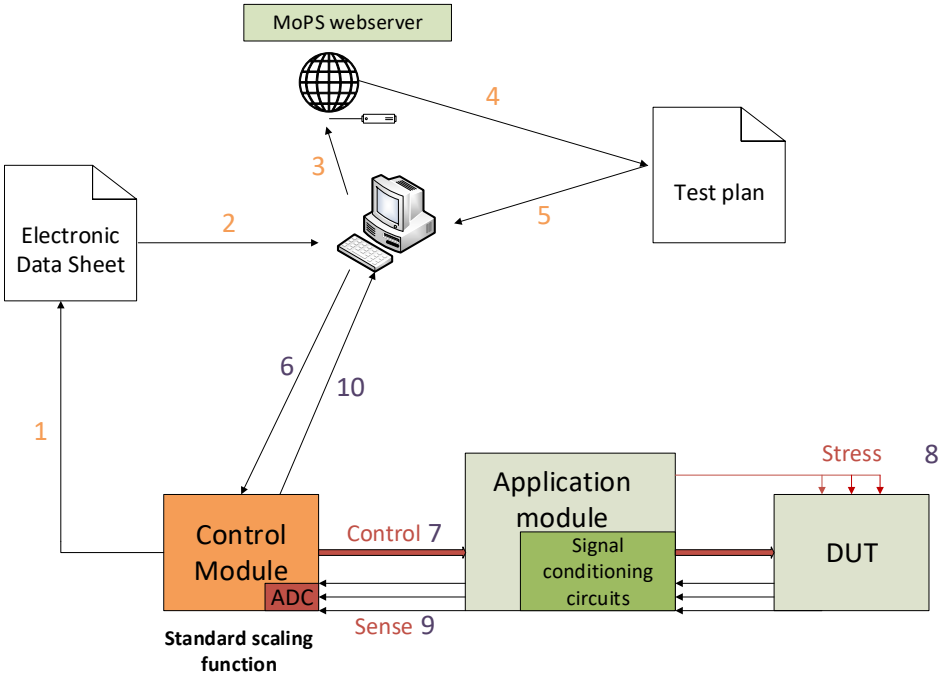

Figure 3.14: State of the art

2. SAM accesses the latest EDS that is generated in the MoPS CORE firmware project folder.

3. SAM further updates the latest EDS to MoPS web server.

4. Test plan builder tool downloads the EDS to create and validate the test plans that are created by test engineers.

5. A Semiconductor is not subjected to stress test until this point. SAM further keeps waiting to receive test plans to subject DUT into an application-specific stress test. Subsequently, Test plans (see Section 3.2.2) that are created by test engineers are loaded to SAM.

6. Test plans are forwarded by SAM to MicroMoPS for it's execution.

7. Test plans are executed and as a part of execution, the PWM signal is sent from the control Module to DUT. The control (PWM) signal makes the DUT go into an operational mode.

8. Electrical stresses are applied onto DUT through the application module.

9. Electrical signals are generated in the DUT because of the electrical stresses that are exerted on the semiconductor device.

10. The generated electrical signals are measured through the analog communication channels of MicroMoPS.

11. The analog measurements are digitized using an ADC of MicroMoPS and the digitized values are later processed using a standard scaling function (see Section 2.2.3) to convert them into discrete MicroMoPS specific voltage values.

12. The processed data are acquired by SAM to retain the original values of analog measurements. The retained values are used in the SAM for further interpretation such as, for calculating the end of life of DUT.

4 Approach and Methodology

4.1 Overview

After understanding the scope of improvement in the existing analog measurement signals processing methodology, a new approach is worked out and implemented within the Low Voltage MTS setup. The new approach involves some action items that are needed to be completed at a certain chronology. The sequence of action items is depicted in Figure 4.1. Section 4.2 describes the data processing concepts that are derived for processing analog measurements in MicroMoPS. Section 4.3 describes the identification of boards in the stress test environment and preparation of board related information in the MoPS web server. Section 4.4 describes the automated procedure of processing analog measurements regardless of the board combinations.

4.2 Derive data processing concepts for stress test environment

4.2.1 Acquiring a signal model

In the modular stress test environment, the application module is used for submitting stresses on DUT. The used application module involves its own application specific characteristic circuitry. In this work, since it is the Low-Voltage stress test system that is used, the discrete power transistors are submitted to *power cycling* ([2], p.2) test. This is a test in which DUTs are pulled to their extreme operating conditions by introducing them into various scenarios. Further, in response to a "power cycling" test, certain MTS parameters are measured. These parameters of the Low Voltage MTS are recorded based on the control logic implemented by a PI controller at BuckMoPS (Section 3.2.1).

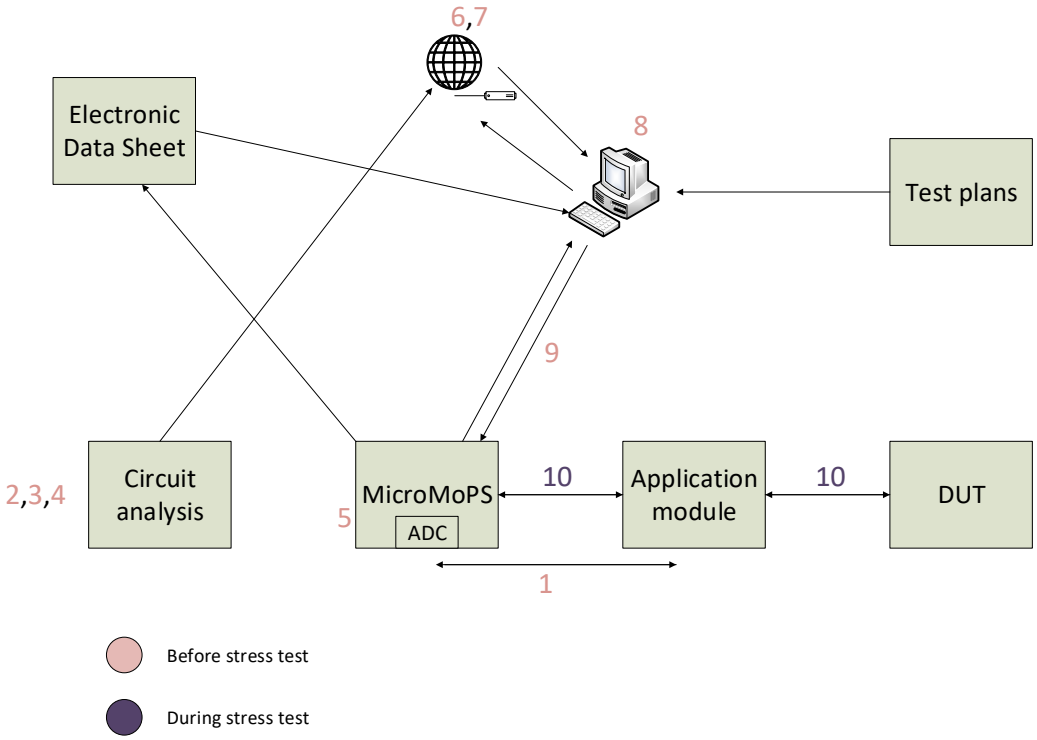

Figure 4.1: Enhancements.

As a result, the measurement signals are automatically monitored by a control module throughout the entire test. These measurement signals go through conditioning before reaching the ADC, as explained in Section 3.2.3. To obtain the real time efficient scaling mechanism for MicroMoPS, the path that the measurement signals traverse from the source of the signal conditioning circuits to the source of the analog-to-digital converter of the control module as shown in Figure 4.3, are analysed in this work. The path that the measurement signals travel consist of the measurement circuits that are associated with acquisition of measurements. Further, the signal path information is obtained by following the hardware schematics of MicroMoPS and application module. The values of every active and passive device that are present in the measurement circuitry is important to know because it helps in obtaining scaling factor associated parameters which are specific to the respective type of measurements.

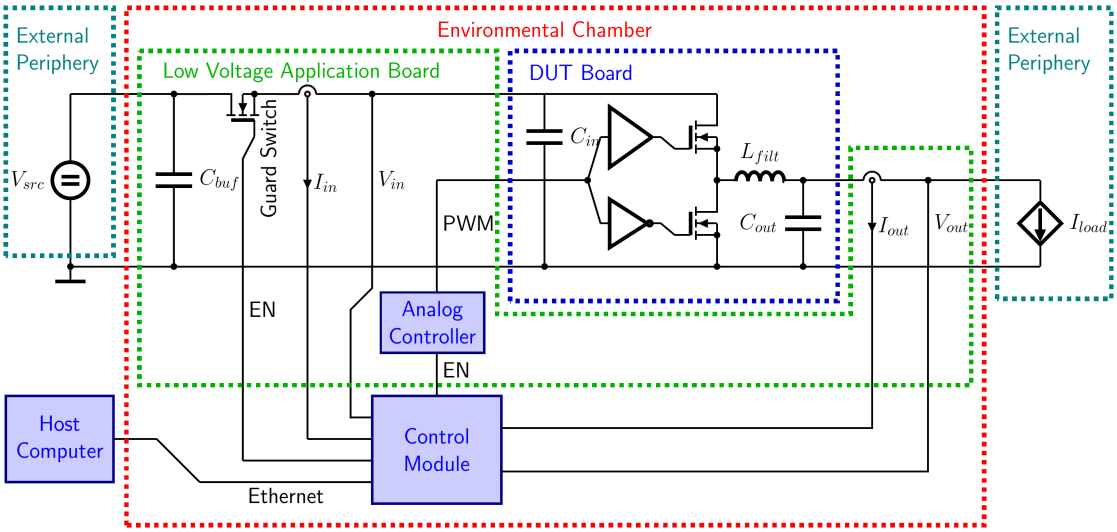

Figure 4.2: Low Voltage Modular test system.

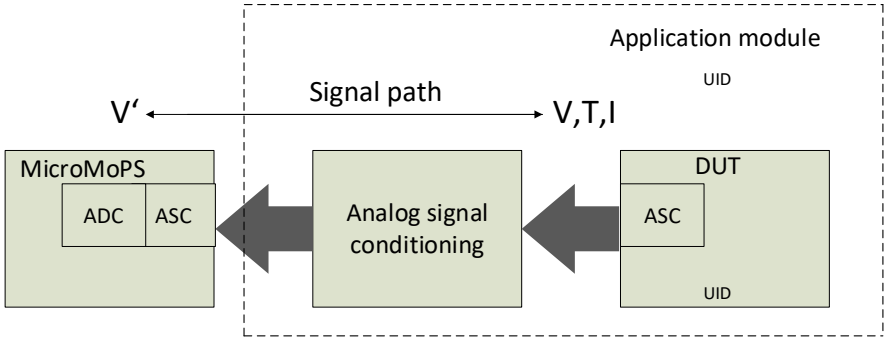

Figure 4.3: Signal path of measurement data acquisition.

The series resistors (R1, R3), differential operational amplifier and the standard voltage divider circuit (R2, R4, R5 and R6) conventions (see Figure 3.13) are part of the analog signal conditioning circuit as described in Section 3.2.3. They are defined by Hardware designers at KAI to appropriately attenuate the electrical signals into MicroMoPS operating voltage range from 0 V to 3 V. The defined circuit component values are listed as shown in Table 4.1.

Table 4.1: Component values

Signal	R1	R2	R3	R4	R5	R6
Iin	18 kΩ	60kΩ	18kΩ	60kΩ	15kΩ	15kΩ
Iout	27kΩ	60kΩ	27kΩ	60kΩ	75kΩ	75kΩ
Vin	26.9kΩ	4.02KΩ	26.9kΩ	2kΩ	NP	NP
Vout	6.7kΩ	4.02KΩ	6.7kΩ	2kΩ	NP	NP
Imon	2kΩ	2kΩ	2kΩ	2kΩ	4.02kΩ	2kΩ

4.2.2 Derive transfer characteristics to find scaling parameters.

After obtaining a signal model which provides information about the measurement circuit of the Low Voltage stress test environment, the circuits that are present in the signal path are analysed to derive the transfer function between the source of the signal conditioning circuits and the source of the analog-to-digital converter of the MicroMoPS. Derivation of the transfer function is done using one of the standard circuit analysis techniques (e.g. Superposition theorem). Measurements such as Vin, Vout, Iin, Imon and Iout with differential amplifier circuits (see Section 3.2.3) are analysed using Superposition theorem. Using the Superposition theorem, three equations of input/output relations are obtained from which, the transfer function is derived. The derived transfer function is further simplified. The steps that are involved in deriving the transfer function are as follows:

1. V_1 and V_2 are shorted.

$$V_{o1} = V_{ref} \cdot \frac{R_3||R_4}{R_6 + R_3||R_4} \cdot \left(1 + \frac{R_2}{R_1 + R_5}\right)$$

2. V_1 and V_{ref} are shorted.

$$V_{o2} = V_2 \cdot \frac{R_4||R_6}{R_3 + R_4||R_6} \cdot \left(1 + \frac{R_2}{R_1 + R_5}\right)$$

3. V_2 and V_{ref} are shorted.

$$V_{o3} = V_1 \cdot \left(-\frac{R_2}{R_1}\right)$$

V_{out} results in $V_{out} = V_{o1} + V_{o2} + V_{o3}$, which equals the expression of nodal analysis, i.e.

$$V_{out} = \frac{R_3R_4R_6}{R_3R_4 + R_3R_6 + R_4R_6}\left(\frac{R_2}{R_1 + R_5} + \frac{V_{ref}}{R_6}\right)\left(1 + \frac{R_2}{R_1} + \frac{R_2}{R_5}\right) - \frac{R_2}{R_1}V_1.$$

The resistor relations assumptions are applied:

$R_1 = R_3$
$R_2 = R_4$
$R_5 = R_6$

This further simplifies the V_{out} equation to

$$V_{out} = \frac{R_2}{R_1}(V_1 - V_2) + \frac{R_2}{R_5}V_{ref}, \qquad \text{where } (V_1 - V_2) = V_{in}$$

Further, the information of MicroMoPS' voltage range and resistor values as described in Section 3.2.3 and Table 4.1, respectively, are specific to measurement channels. These values are plugged into the derived transfer function to obtain the measurement range of the voltage or current or temperature signal before conditioning. The algebraic method of performing the above explained calculation is demonstrated by considering a 'Vin' measurement parameter's signal path as an example.

For 'V_{in}' measurement parameter the V_{out} equation simplifies to,

$$V_{out} = \frac{R_2}{R_1}.V_{in}, \tag{4.1}$$

To find 'V_{in}' measurement range, the maximum voltage condition of MicroMoPS is considered and the values corresponding to V_{in} are put to Equation (4.1),

$$3V = \frac{4.02k}{26.9k}V_{imax},$$

-> V_{imax} = 20.17500 V

Similarly, minimum voltage condition corresponding to V_{in} becomes,

$$0V = \frac{4.02k}{26.9k}V_{imin},$$

Table 4.2: Measurements range

Signals	**min**	**max**
Iin	-5A	10A
Iout	-5A	70A
Vin	0V	20.175V
Vout	0V	5.020V
Imon	0V	0.305V

-> V_{imin} = 0 V

Likewise, the measurement range of the signal at source is found for all the measurement parameters. They are listed in Table 4.2.

Also, the measurement range of other parameters are found by either simple nodal analysis method or they are present in hardware schematics of application module.

Using the obtained measurement ranges, the measurement specific linear equation fitting scaling parameters are found out. The algebraic method of finding out scaling parameters is explained by reconsidering the same measurement parameter, i.e., Vin.

Linear scaling function is given by
y = k*x + d,
where, y = analog value,
k = scaling factor,
x = digital representation of analog measurement,
d = offset

For maximum voltage condition of Vin, the digital value is 4095 (12 bit resolution, see Section 3.2.3), the analog value is the maximum Vin measurement from the Table 4.2

$$20.175 = k(4095) + d \tag{4.2}$$

Similarly, for minimum voltage condition of Vin, the digital value is 0, the analog value is the minimum Vin measurement from the Table 4.2

$$0 = k(0) + d \tag{4.3}$$

Solving Equation (4.2) and Equation (4.3) using substitution method, the values of scaling parameters that are associated to Vin measurement parameter are obtained as below.

k = 0.0049267399267399,
d = 0

From the obtained scaling parameters 'k' and 'd', it is evident that the found scaling parameters could be used for processing the digitized stress measurements to original signal value (signal before conditioning) but not to MicroMoPS specific voltage range. Therefore, the linear equation is further improved by considering additional parameters in the scaling function.

4.2.3 Find a linear scaling function from obtained scaling parameters.

The improvement of the linear scaling function as discussed in the previous section is achieved by considering the changing factor inside the linear scaling function. The mentioned changing factor influences in determining the appropriate scaling values that precisely process analog measurements into MicroMoPS specific voltage values. It is given by,

$$y = \left[\left(k - \frac{V_{max} - V'_{max}}{4096}\right) * x + d\right] V$$

If the measurement parameter is temperature or voltage signal then, the improved linear scaling equation is as shown below,

$$y = \left[\left(k - \frac{(I|T)_{max} * g - V'_{max}}{4096}\right) * x + d\right] V$$

where, 'I' is the current measurement signal and 'T' is the temperature measurement signal. The changing factor represents the change in maximum value of the original (V/T/I) signal and the maximum voltage of the attenuated signal (V') as represented in the Figure 4.3. The value '4096' in the equation represents the factor that contributes in the occurring change. Finally, the transconductance parameter 'g' does the necessary conversion of temperature or current signals into respective voltage values (V').

For the measurement parameter Tcase [17], since it is the resistive sensor that is fed directly on to ADC of the MicroMoPS instead of a differential amplifier, the scaling parameters for T_{case} are found in a differently instead of deriving transfer function. The DUT board is let to go into temperature ranging between –55 °C and 37 °C. Consequently, the corresponding voltage that is generated because of voltage amplification for every different board temperature is measured. Using the generated voltage information, scaling parameters ('k', 'd') and transconductance parameter ('g') are found. Scaling parameters of all the measurements and the measurements respective units are listed in Table 4.3.

Table 4.3: Measurement parameters

Scaling parameters associated to Measurements			
Signals	**k**	**d**	**Units**
Iin	0.000732601	0.0	A
Iout	0.000732601	0.0	A
Vin	0.000363	0.0	V
Vout	0.000363	0.0	V
Imon	0.000732601	0.0	A
Vdrv	0.000732609	0.0	V
Idrv	0.000733	0.0	A
Imon	0.000732601	-1.5	V
Tcase	0.000734056	0.0	C
Vswh	0.000732601	0.0	V
Tmon	0.000732601	0.0	V
Aux1	0.000732601	-1.5	V
Aux2	0.000732601	0.0	V

Supposedly, if the stress test is of any application other than Low Voltage MTS, then the respective test application specific processing parameters such as **k**, **d**, V_{max}, V'_{max} and **g** are the only information that are needed to precisely process the analog measurements.

4.2.4 Extended scaling mechanism.

Reverse Polish Notation (RPN) is a memory-efficient method that could be used in performing the scaling function. The working principle of RPN is explained by

considering the scaling function of one of the channels (see Section 3.2.3) of MicroMoPS as an example.

For example, the Scaling function of channel sync[0][0]B is (d/4096)*3 where d is a digital value. RPN method of computing this scaling function is:

Step1: Convert the scaling function into a string i.e "d 4096 / 3 *". For the sake of understanding, if d is given a digital value of 2048 then, the RPN notation becomes "2048 4096 / 3 *".

Step2: Push 2048 onto Lua stack.

Step3: Push 4096 onto Lua stack.

Step4: Pop from lua stack twice, use the operator '/' to calculate 2048/4096 and push the result i.e. 0.5 onto Lua stack.

Step5: Push 3 onto Lua stack.

Step6: Pop from Lua stack twice, use operator '*' to calculate 0.5*3 and push the result i.e 1.5 onto Lua stack.

This way, the scaling is performed for every communication channel of MicroMoPS using RPN.

4.3 Identify the boards and prepare LUTs in web server

4.3.1 Board UID extraction.

The starting point of the Low-Voltage MTS setup is the operation of the test sequence in MicroMoPS. The test sequence is executed by the central control entity i.e SAM as a test plan. Before, the DUT is subjected to *power cycling* (see Section 3.1), the extraction of UID of the boards is an important step. It is considered important because it helps in meeting one of the objectives of this work, which is to automate the processing of analog measurements in MicroMoPS. Identification of the UID of the application module and DUT is done by pursuing I2C communication between the control module and a UID provider chip (see Section 5.1) of application module and DUT, respectively. To achieve I2C communication, the concerned functional pins of the control module are used and standard steps of the communication procedure are followed (see Section 2.2.1). Thereby, the UID of MicroMoPS itself is read from a register.

4.3.2 Prepare board ID database in MoPS web server.

The board UIDs which are obtained from the MicroMoPS are communicated through Ethernet to its host application system called SAM. Proactively, every board combination which corresponds to a particular type of stress test along with the board UIDs are stored as a lookup database as shown in the Listing 4.1, in MoPS web server (see Section 3.2.2).

Listing 4.1: MoPS Boards

```
{
    "date": "2019-09-29T18:15:11+02:00",
    "applicationModule": [
        {
            "name": "Buck MoPS v3.1",
            "uids": [
                "0000000000d8d3c7"
            ]
        }
    ],
    "dutBoard": [
        {
            "name": "LGA771",
            "uids": [
                "0000000001052523"
            ]
        }
    ]
}
```

4.3.3 Store the scaling values meaningfully in MoPS web server database.

Based on the communication channels' configuration information provided in the EDS and mathematically obtained scaling parameters for each of these communi cation channels, a LUT is created as shown in the Listing 4.2, in the MoPS web server. The created LUT contains the scaling related information.

Listing 4.2: MoPS Scaling example

```
{
    "date": "2019-09-28T10:18:02+02:00",
    "scaling": [
        {
            "hwTarget": "uMoPS_v5",
            "appModule": "Buck MoPS v3.1",
            "dutBoard": "LGA771",
            "channel": "sync0:0B",
            "rpn": "",
            "description": "V_drv",
            "unit": "V",
            "scale": 0.000732609,
            "offset": 0
        }
    ]
}
```


4.3.4 Send UIDs before MicroMoPS enters into real-time mode.

The extracted board UIDs along with their directional pins are embedded as a single string and communicated as a payload to SAM using Ethernet communication. The board UIDs communication is ensured to take place before the firmware enters into the main loop. The reason for the UIDs communication before main loop is to avoid redundant communication of UIDs and also to keep short the time that elapses from entering into main loop till the test pattern starts to execute in MicroMoPS. The time at which the control module enters inside an infinite loop is considered as an entrance towards the real-time mode of this stress test environment. In the real-time mode, the role of the control module is to endlessly perform tasks that are associated with the stress test system.

4.4 Resolve UIDs in SAM and send the scaling values to MicroMoPS

4.4.1 Resolution of UIDs in SAM

Before the control module enters into a real-time mode the EDS is generated. SAM reads the latest EDS from the MicroMoPS firmware project. The EDS that is read by

SAM is uploaded to the MoPS web server (see Section 3.2.2). SAM becomes functional when test engineers at KAI start to perform the application specific stress test. As soon as SAM starts running, it downloads LUTs associated to scaling and boards (see Figure 1.1) that are created in MoPS web server. The LUTs associated to scaling and boards are as shown in Listing 4.2 and Listing 4.1 respectively. The LUTs associated to scaling and boards together help the SAM to automate the scaling mechanism regardless of the type of stress tests that are performed. Test plans are created by test engineers and are loaded to SAM.

The flow at which the resolution of board UIDs and the communication of associated channel and scaling related parameters to MicroMoPS is depicted in Figure 4.4. The Board UIDs that SAM receives from MicroMoPS are verified with the board UID LUT which is downloaded by SAM. Subsequently, board UIDs are searched in the board UID LUT to filter the respective board names. The board names that are fetched after

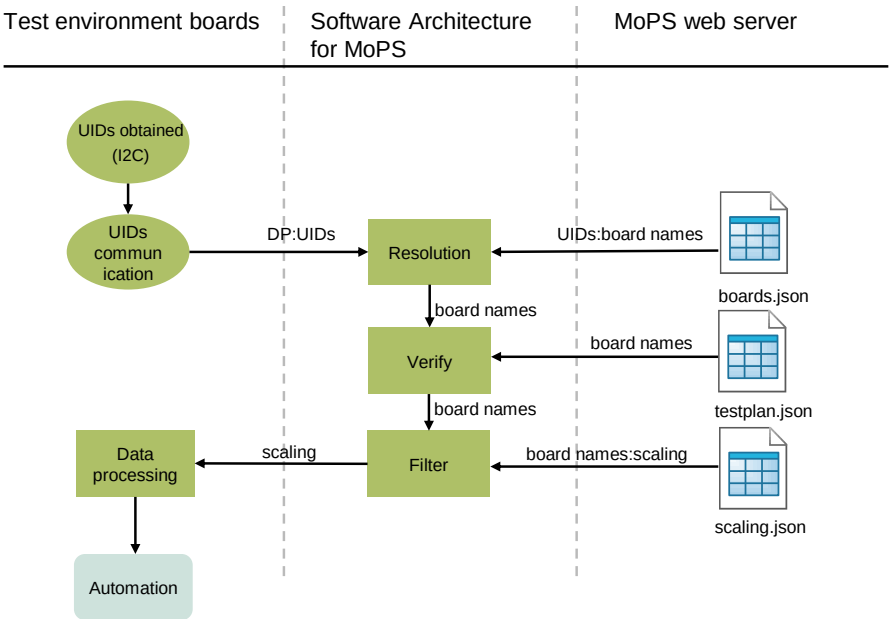

Figure 4.4: Resolution of board UIDs and scaling values communication

the resolution of the board UIDs, are further authenticated with the board names that are present in the test plan. The board names that are present in the test plan are the hardware information provided by test engineers. Verification of boards are done in order to maintain consistency between the actual board connections and the test

engineers selection of boards in the test plan. After the successful verification of board names in SAM, the verified board names are searched in the scaling connected LUT to fetch the appropriate MicroMoPS channels and their scaling values.

4.4.2 Send the scaling values to MicroMoPS.

The fetched communication channels and their corresponding scaling values are sent in queue as a string back to data processing system of MicroMoPS, using Ethernet communication. The communication channels and their scaling values that are received at scaling related receiver handler of MicroMoPS are dynamically parsed to appropriately adjust the scaling values to the channels specific scaling variables (see Listing 5.1).

4.4.3 Execute Low Voltage Test System

After successful adjustment of the scaling parameters in the MicroMoPS, the test sequence which is forwarded by SAM is executed on MicroMoPS. Finally, after all the above phases, the linear scaling mechanism and the Reverse Polish Notation get successfully integrated into the data processing system of MicroMoPS. The entire procedure of resolving the board UIDs supports to the automation of processing of analog measurements in MicroMoPS. Measurements that are processed using linear scaling and Reverse Polish Notation are acquired at SAM and manipulated to retain the original values of analog measurements for any interpretation.

5 Implementation and Realization

The discussed approach is brought into existence in this sophisticated test environment by accessing MicroMoPS and its peripherals, by extending the pre-developed MicroMoPS firmware and by effectively utilizing the Actor framework of SAM. The implementation of the proposed approach is done for a Low Voltage stress test system. The boards that are associated with the Low voltage stress test system (Figure 1.1) are termed in the MoPS environment as follows:

- Application module - BuckMoPS
- DUT module - LGA771

5.1 Extraction of UID of DUT and application board

Board IDs are selected by using relevant GPIO pins of MicroMoPS, which are meant for I2C (see Section 2.2.1) communication. In this work, MicroMoPS is treated as a Master block, DS28CM00 [10] UID chip of application module and DUT as slave modules. The operation of reading the board UID is succeeded by initializing and utilizing accordingly the appropriate digital IO pin of MicroMoPS. The digital pin corresponding to the Master Module is set to appropriate mode at the very beginning of the operation. Setting digital pin into an appropriate mode drives the Serial interface clock input (SCL) (see Section 2.2.1). Two PCA9515A [22] ICs are embedded in the BuckMoPS to facilitate the MicroMoPS in reading UID of application board and DUT module. With the support of PCA9515A ICs and UID chip, the corresponding DUT and application module are interacted from MicroMoPS using I2C communication.

Obtainment of the board IDs is primarily controlled as per the programmer's requirement by utilizing the select line of the UIN chip of BuckMoPS. PCA9515A ICs

are part of UIN chip present in BuckMoPS, which facilitates accessing the DS28CM00 module. When the select line is set as logic 0, PCA9515A of application module is enabled, which further establishes communication between MicroMoPS and UID chip of application module. Application module's UID can be requested using this communication. Similarly, when the select line is chosen as logic 1, PCA9515A of DUT is enabled, which further establishes communication between MicroMoPS and UID chip of DUT. DUT's UID can be requested using this communication. There are two communication lines which need to be synchronously used by the master to write to UID chip or read from UID chip. They are, SDA and SCL lines (see Section 2.2.1). SDA line is used for sending the data from master to slave or receiving the data from slave to master and SCL is used for providing synchronization in communication between master and slave. Because the concern is to read the board id of DUT and application module from MicroMoPS, systematic sequential communication procedure is followed (see Section 2.2.1). The received data is CRC verified using standard 8-bit polynomial i.e X8 + X5 + X4 + 1 [10], to authenticate the corruption of received data.

The above operation is implemented inside the test_identify_boards() handler (see Section 3.2.2) of MicroMoPS firmware to successfully extract the board UID of BuckMoPS and LGA771.

The board UIDs extracted from slave devices are listed in the Table 5.1:

Table 5.1: Boards and their respective UID

Board type	**Board name**	**Unique Identity Number**	**Select line**
Application module	BuckMoPS	0x0d8d3c7	0
Device under test	LGA771	0x1052523	1

5.2 Update the microcontroller's configuration file for linear scaling computation

The configuration file of a MicroMoPS contains the initializations of MicroMoPS associated peripheral modules. These modules are configurable based on the firmware

developer's needs. In the MicroMoPS firmware project, there is a dedicated configuration file to manage the configurations of a MicroMoPS called "config.c".

To incorporate the linear scaling mechanism to MicroMoPS, the scaling module i.e MoPS scaling, is extended by adding two parameters as shown in the Listing 5.1. The added parameters are *scaling factor* and *offset* in adjacent to the respective analog channel (.name) parameter. The scaling values that are assigned are the default scaling values. These values are meaningful to measurement circuits of the Low-Voltage stress test system. These scaling values are updated automatically with different values when the stress test system that is performed is different from the Low-Voltage test system.

The EDS is generated every time when the MicroMoPS firmware starts. The EDS contains the communication channels and channels' specific scaling parameters. The DUT creation in the mops web server based on the EDS information and downloading of the LUTs when SAM starts running was carried out by summer students of KAI.

Listing 5.1: MoPS scaling

```
MoPS_scale MoPS_scaling[] = {

   {.name = "sync1:2A", .scale = 0.0003662109375f, .offset = 0.f,},

   {.name = "sync0:0B", .scale = 0.000732421875f, .offset = 0.f,},

   {.name = "sync0:2A", .scale = 0.014892578125f, .offset = 0.f,},

   {.name = "sync1:0A", .scale = 0.000732421875f, .offset = -1.5f,},

   {.name = "sync0:0A", .scale = 0.02978515625f, .offset = -61.0f,},
   END_OF_LIST,

};
```

5.3 Board UIDs communication to SAM

The byte-wise data of UID, which are received from the UID chip is collected and combined inside test_identify_boards() (see Section 3.2.2) handler to reproduce UID (64-bit length) of the board. In this work, UID of BuckMoPS and LGA771 are restored. The restored UIDs are further communicated to SAM using the dedicated MicroMoPS' test handlers [1].

5.4 Resolution of UIDs in SAM and scaling values communication from SAM to MicroMoPS

The resolution of Board IDs in SAM is implemented by utilizing the LabVIEW software packages. The basic step before the implementation of the algorithm which mainly resolves the board ids is the serialization of the application module's objects and dutBoard's objects into LabVIEW data structures as shown in the Figure 5.1. The application module's objects and the dutBoard's objects are present in the "boards.json". The board UIDs are cached offline using I2C communication and communicated to SAM. The derived scaling values and board UIDs are stored in a database of MoPS web server. As described in the Section 4.4.1, the board database, and scaling database are downloaded by SAM from the MoPS web server. Intuitively, "boards.json" and "scaling.json" are downloaded by the main actor of SAM to facilitate the board UIDs resolution. The resolution of board UIDs is done in the Node actor (see Section 3.2.2) of SAM. To achieve board UIDs resolution, variables such as "name" and "uids" are declared as string and array of string data members of class "Board IDs", respectively as shown in the Figure 5.1. Similarly, variables such as applicationBoards and dutBoards are declared as data members of "Board ID export" class. The data members of "Board ID export" i.e applicationBoards and dutBoards, are an instance of the class "Board IDs".

The classes that are created by the name "Board IDs" and "Board ID export" are the LabVIEW classes. The serialization methods that are present in the classes do the conversion of JSON objects present in "boards.json" into strings. These strings are nothing but the key-value pair of applicationModule and dutBoard as shown in the Listing 4.1. Thus, by means of serialization, the board name and uid values of applicationModule JSON object and dutBoard JSON object are assigned to applicationBoards and dutBoards data members of "Board ID export", respectively. In the same way, the board UIDs along with their directional pin that are received by SAM from MicroMoPS, are assigned to data members of "Board Info item". Based on the directional pin value, the received board UIDs are classified into appropriate board type and accordingly, the board UIDs are assigned to UID data member of "Board Info item". After a successful serialization, the UIDs of data members (applicationBoards and dutBoards) of "Board Info item" are compared with the UID of data members (applicationModules and dutModules) of "Board ID export".

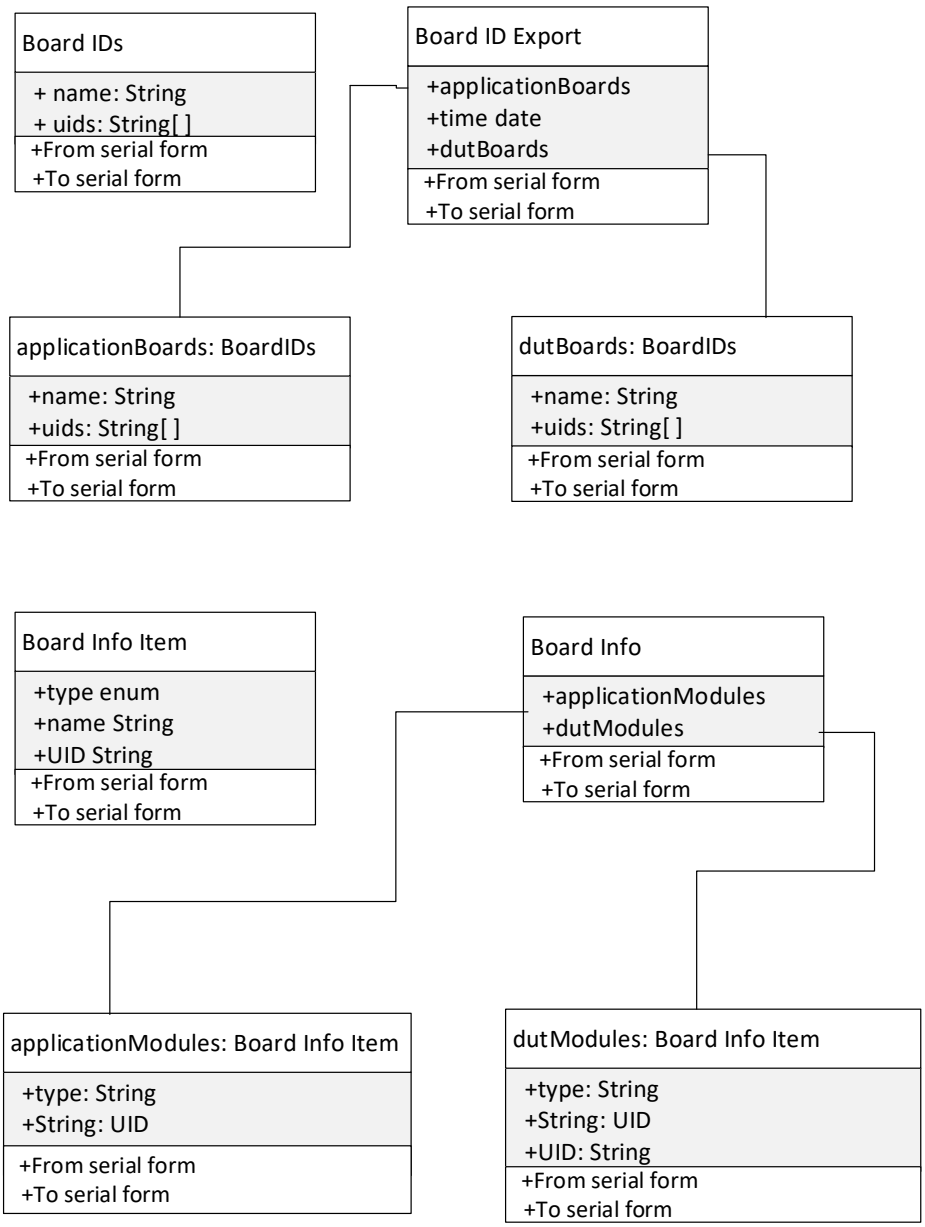

Figure 5.1: Serialization of JSON objects into LabVIEW data structures

This, comparison if found matched, the respective board names such as BuckMoPS and LGA771 are updated to application module and dutModule objects which are

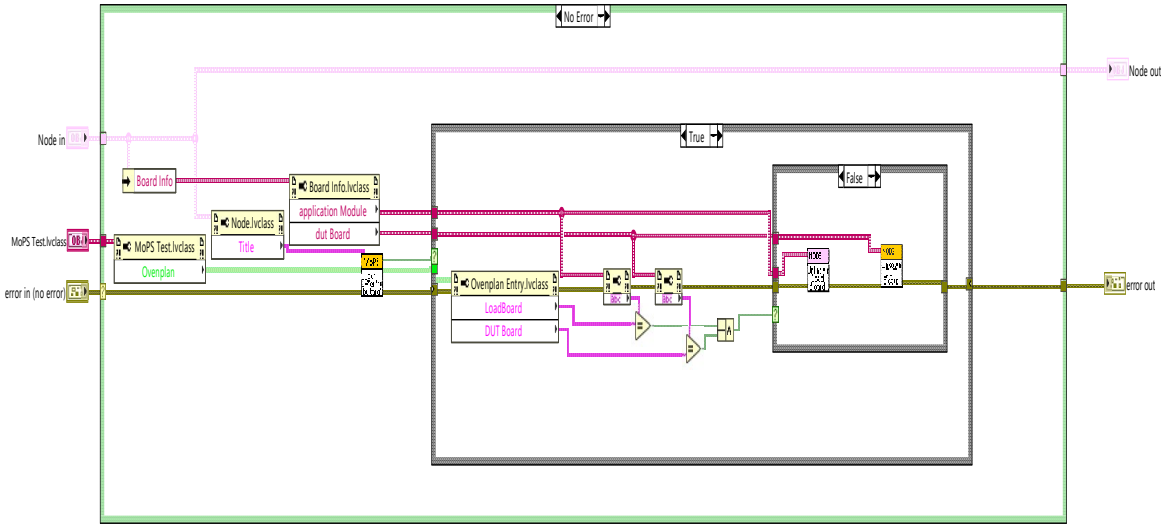

Figure 5.2: Verification of boards with oven plan entry.

instances of "Board Info". Earlier to this update, only a UID information and the type was known to application module and dutModule objects, through the MicroMoPS communication. Successful comparison leads to the resolution of board UIDs. The updated board names along with their respective types in "Board Info" are logged on to the log GUI display of SAM. Logging of board names with their type helps the test engineers or SAM developers to know the status of detection of boards by SAM.

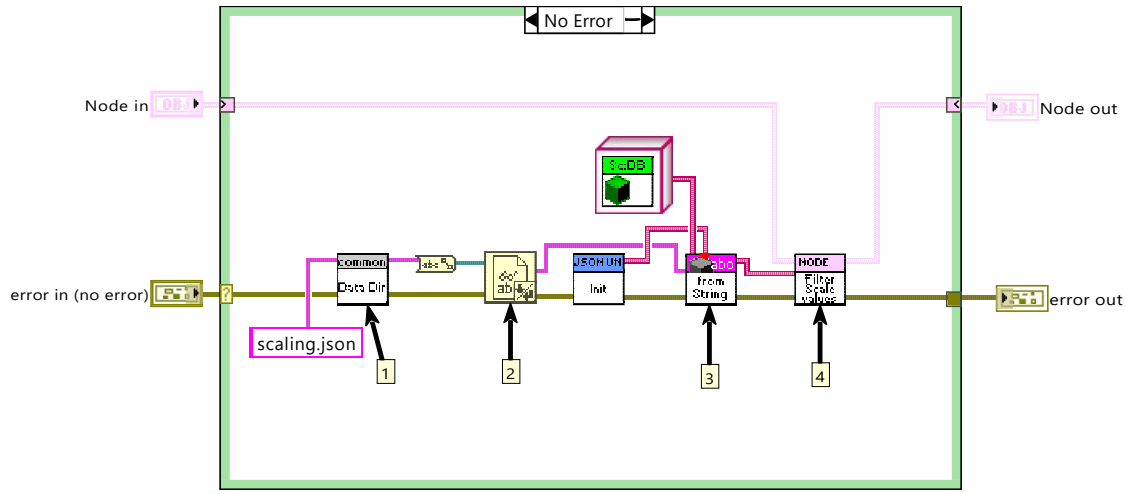

Figure 5.3: Communication of channel name and associated scaling values to MicroMoPS.

Further, the board names that are present in the application module and dutModule objects are verified with the Hardware information present in the oven plan as shown in the Figure 5.2. The Serialization procedure as described earlier is undergone by test plan (JSON) to un-flatten the oven plan JSON objects into LabVIEW data structures. The Hardware information which is present in the oven plan entry class (due to serialization) is verified with the board names that are present in the application modules and dutModules. If the verification is unsuccessful, the conflicting board information is reported to the MoPS web server.

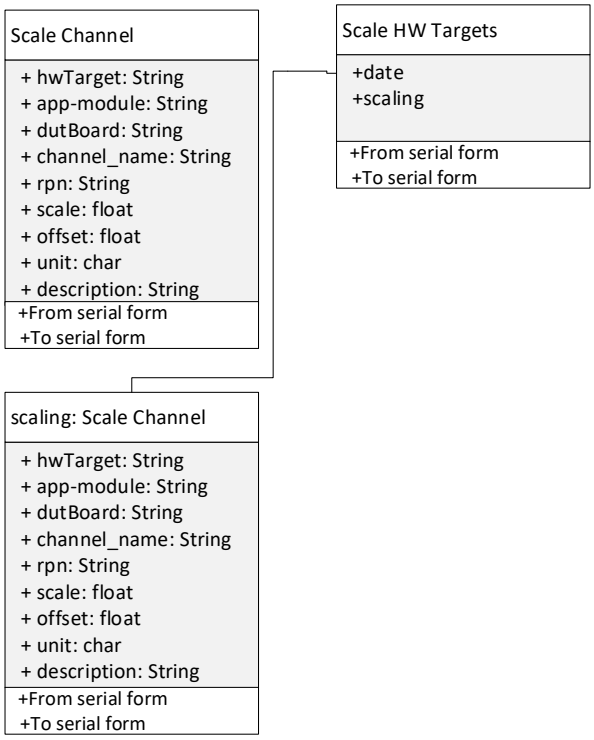

Figure 5.4: Serialization of "scaling.json" objects into LabVIEW data structures

After successful verification, the board names correspondent combination is searched from the "Scale HW Targets" LabVIEW object. "Scaling" object is a result of serialization of "scaling.json" (Figure 5.3) into LabVIEW objects. Serialization of "scaling.json" (Figure 5.3) into LabVIEW objects enables the conversion of JSON objects (Listing 4.2) present in the "scaling.json" into strings. After Serialization, filtering of channel names, scaling values and offset parameters, and communication of the same are done at a

particular sequence as shown in the Figure 5.3. Below are the operations that are done at the mentioned sequence:

1. Read the configuration path.
2. The text present in the text file is read.
3. De-serialization of JSON objects into LabVIEW objects is performed.
4. The data members of the "Scaling" object i.e Hardware Target, application module and DUT board combination are compared with the applicationModule and dutModule objects. In the case of successful match, the board combination specific scaling value, offset and channel name are filtered from the "Scaling" object. The filtered parameters are further communicated to the board ID receiver handler [1] of MicroMoPS.

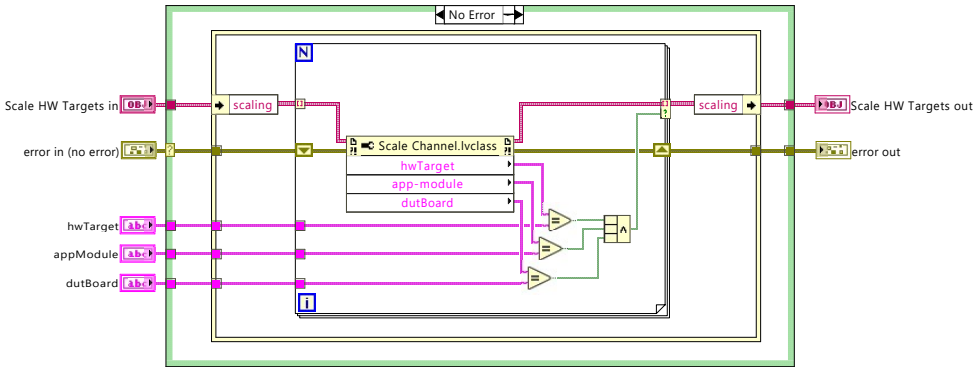

Figure 5.5: Filtering of scaling values

The filtering of the scaling values are implemented as shown in the Figure 5.5. The "Scaling" object is an instance of "Scale Channel" class which contains the hwTarget, app-module, dutBoard, channel_name, rpn, scale, offset, unit and description data members. Data members such as scale, offset and channel_name are filtered by performing a comparison between the board names that are obtained after resolution and the hwTarget, app-module and dutBoard data members, respectively. The control module's board name is obtained from the hardware information present in EDS.

Finally, the filtered parameters are communicated to the board ID receiver handler of MicroMoPS. The filtered parameters are specific to every individual channel. The parameters are channel-wise communicated in a queue to MicroMoPS. The VI implementation of communication of scaling values is as shown in the Figure 5.6.

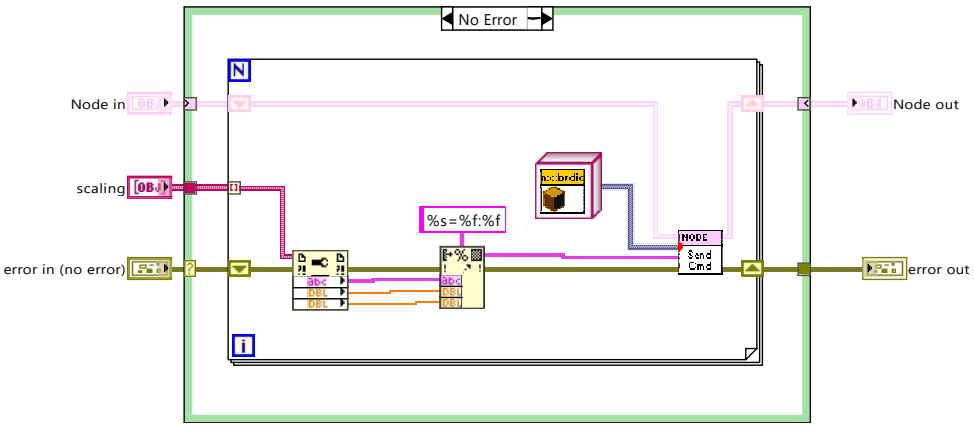

Figure 5.6: Communication of scaling values

5.5 Parse the scaling parameters dynamically in MicroMoPS

Sequentially, the board combination specific communication channels and their respective scaling parameters are received as a payload (string data) from SAM to board ID receiver handler (MoPS_cmd_handler_BOARDID()) of MicroMoPS (see Figure 5.7). The string data that is obtained at board ID receiver handler of MicroMoPS is delimited by ':' between channel name and scaling parameters. It as shown in the example: "sync0:0B:0.0007326007326:0". Where, sync0:0B is a channel name and module (see Section 3.2.3), 0.0007326007326 is a value of the scaling factor denoted by 'k' and 0 is a value of offset denoted by 'd'. This string is dynamically parsed and assigned to channel-specific scale (.scale) and offset (.offset) variables (see Figure 5.7) present in the "MoPSscaling" module. Other parameters such as .func and .name are presesnt in order to define the RPN function and the communication channel module, respectively. These parameters are globally used in MoPS firmware, which are defined in the configuration file of MoPS firmware. At present, the RPN string corresponding to the communication channels is not communicated from SAM.

5.6 Compute scaling mechanisms in MicroMoPS

In this work, linear scaling formula (see Section 4.2.3) and Reverse Polish notation (see Section 4.2.4) are implemented within the Lua handler of analog input module of MicroMoPS. The linear scaling formula uses the updated scaling parameters to process

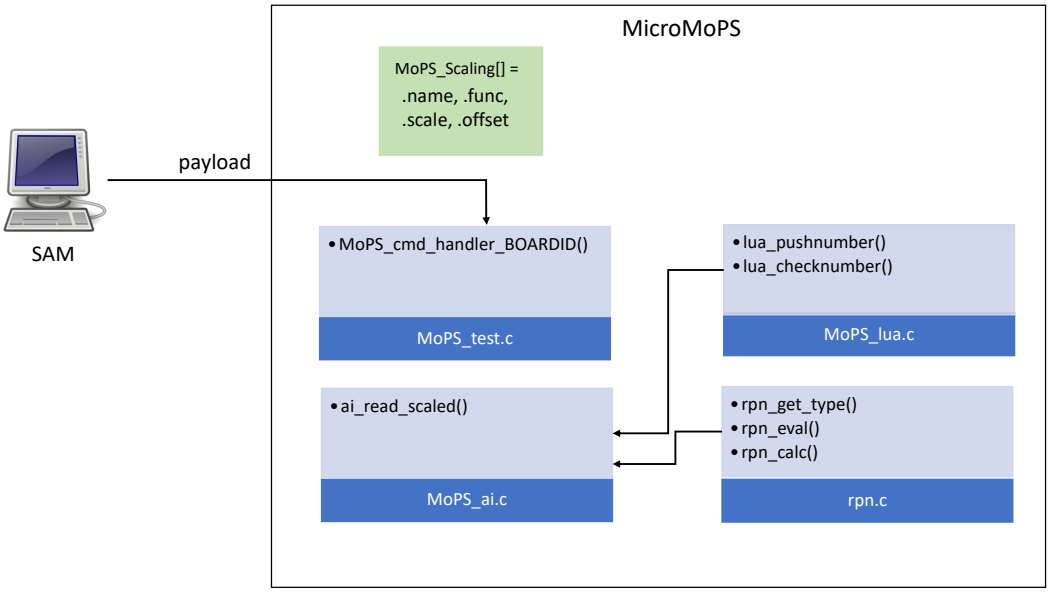

Figure 5.7: Implementation of data processing module

the analog measurements. RPN computation is achieved by making use of three fundamental handlers, such as 'rpn_get_type()', 'rpn_eval()' and 'rpn_calc()'. These handlers are defined in the 'rpn.c' file (see Figure 5.7). The 'rpn_get_type()' handler is meant for decoding the respective operations or tokens based on the string data that is received from the SAM. The allowed operations in the RPN module are float variables, addition, subtraction, multiplication, division, square root, minimum function, and maximum function. From the decoded type the respective operation is performed through the 'rpn_eval()' handler. The 'rpn_calc()' handler makes use of 'rpn_get_type()' and 'rpn_eval()' handlers to perform the RPN computation. Along with the mentioned handlers the 'rpn_calc()' handler makes use of some of the functions that are part of Lua interpreter to memory efficiently (see Section 4.2.4) perform the RPN computation. Finally, the MoPS_scaling module (ai_read_scaled()) makes use of either 'rpn_calc()' or linear scaling formula to process the data using dynamically parsed scaling parameters (see Section 5.5). The operation of the RPN scaling mechanism is tested by hard coding the RPN string (see Section 4.2.4) to channel-specific .func variable (updated in the config file). Hard coding of RPN string is achieved by communicating the RPN string from TinyHost to MicroMoPS.

6 Evaluation and Analysis

The enhancement in the existing software architecture to improve the processing speed of DUT electrical measurements are evaluated and analysed in this chapter. The processing capabilities of Reverse Polish Notation and linear scaling function are evaluated by measuring their time and memory consumption in performing the scaling computation. In order to compute Reverse Polish Notation or linear scaling function, a piece of code is written in the digital processing module of MicroMoPS. Scaling operation is performed as many times as the number of communication channels that are present in the MicroMoPS (see Section 5.5). The scaling operation comes as a part of the execution of test plans, and the Lua modules or classes that are specific to scaling are used in the test plan script to invoke the corresponding Lua C-API of MoPS-CORE microcontroller firmware. The Lua class that is meant for invoking the scaling function corresponding C-API i.e., 'ai_read_scaled()' (see Section 5.6) is 'ai' [1]. The real-time performance of the scaling operations is tested by invoking the scaling function multiple times from the TinyHost (see Section 3.2.2), while the test plan is not running. Before invoking scaling functions setting some of the configurations (see Figure 3.9) that are specific to the communication channel is necessary. These test runs of a scaling operation are done at a very indeterministic interval to determine the stable values of the time that is consumed by the scaling operation. The time measurements that are recorded during the test runs are approximated to the closest simpler representation. The real-time performance of Reverse Polish Notation and linear scaling function is discussed in the following sections.

Real-Time performance of RPN function

As described in the Section 5.6, the 'ai_read_scaled()' handler makes use of 'rpn_calc()' handler to perform RPN computation. The Memory consumption of the RPN is

calculated by investigating the code that is associated in performing the RPN. The code snippet which performs the data processing (RPN) computation is built by using several local variables that are specific to 'rpn_calc()' (see Figure 5.7) handler. The memory consumption of each of the local variables that are part of 'rpn_calc()' is as follows:

1. rpn_string = 32 bytes (max)
2. *temp(char) = 1 byte
3. *delim(char) = 1 byte
4. is valid(bool) = 1 byte
5. *save(char) = 1 byte
6. *snippet(char) = 1 byte
7. lua_number – res, a, b (float) = 4bytes*3 = 12 bytes

The sum of the memory consumption of the local variables that are part of 'rpn_calc()' is **49 bytes**.

Consumption of time in performing RPN function is calculated by making use of the 'time_it()' handler from the 'MoPS_time.c' module. The calculation of the consumption of time is done for around 50 test runs. After around 50 test runs, the range of consumption of time to perform RPN is calculated. The time consumed by the RPN function approximately ranges from **45 µs to 66 µs**.

Real-Time performance of Linear scaling mechanism

Similarly, the real-time performance of "Linear scaling mechanism" is measured by calculating the time and memory consumption of the code that performs the computation. The Memory consumption of the linear scaling mechanism is calculated by investigating the code that is associated in performing the linear scaling function. Mere linear scaling equation is implemented within a 'ai_read_scaled()' handler to perform the processing (linear scaling function) of data. The Memory consumption of each of the local variables that are part of linear scaling mechanism is as follows:

1. scale (float) = 4 bytes

2. digital_result (float) = 4 bytes

3. offset (float) = 4 bytes

The sum of the memory consumption of the local variables that are part of linear scaling function is **12 bytes**.

The method of calculation of time consumption remains the same as how it is done for RPN function. Also, for the calculation of consumption of time to compute linear scaling function, around 50 samples of measurements are taken. The time consumed by linear scaling mechanism approximately ranges from **29 ns to 134 ns**.

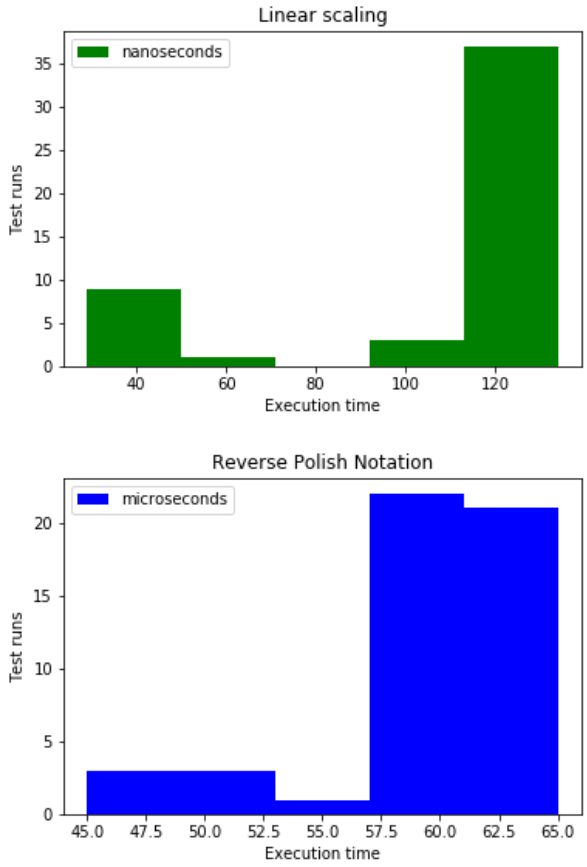

Figure 6.1: Distribution of execution time of the scaling functions

The distribution of the execution time of the linear scaling and Reverse Polish Notation function is as shown in the Figure 6.1. The test runs of scaling computation are performed in this sophisticated stress test environment to record the time that is consumed by the scaling operation. At rare times the execution time of the scaling computation lies closer towards the Best-Case Execution Time (BCET) of the given scaling computation because of the less interrupt load during the scaling operation. However, most of the other times the execution time of the scaling computation varies around the Worst Case Execution Time (WCET) of the given scaling computation because of the substantial interrupt load during the scaling operation. In the case of a linear scaling mechanism, out of 50 disordered test runs, 8 to 10 test runs that records the execution time lies between 29 and 70 nanoseconds, and the rest of the test runs that records the execution time lies between 95 and 134 nanoseconds. Similarly, in the case of Reverse Polish Notation, out of 50 disordered test runs, 2 to 4 test runs that records the execution time lies between 45 and 57.5 microseconds, and the rest of the test runs that records the execution time lies between 57.5 and 65 microseconds. From the obtained execution time distribution, it can be inferred that there exists the stable region around the WCET, which means almost all the times regardless of the increase or decrease of test runs from 50, the execution time of scaling operation only keeps moving within the stable region and at rare times towards the BCET. The stable region for Reverse Polish Notation and linear scaling function lies between 95 and 134 nanoseconds, and 57.5 and 65 microseconds, respectively.

The real-time performance comparison of Reverse Polish Notation and linear scaling mechanism is as shown in the Table 6.1.

Table 6.1: Real-time performance comparision

Scaling mechanism	**Time consumption**	**Memory consumption**	**Test runs**
RPN	45 μs - 66 μs	49 bytes	≈50
Linear scaling	29 ns - 134 ns	12 bytes	≈50

From Table 6.1, it can be concluded that the linear scaling mechanism consumes less time and memory in comparison with the Reverse Polish Notation function. This, further implies that the linear scaling mechanism is more real-time efficient scaling mechanism compared to Reverse Polish Notation.

On the other hand, the entire procedure of resolution of board UIDs from multiple different sources such as the test plan, board and scaling database in web server, board

UIDs communication from MicroMoPS, and subsequent communication of scaling parameters to the data processing system of MicroMoPS supports to an automation of processing of analog measurements in the stress test environment regardless of any plausible combination of hardware target, application module and DUT. Every plausible combination of boards correspond to an individual type of stress test application. The main benefit of the automation of the processing of data is the avoidance in the maintenance of the scaling database of every application-specific stress test system in the control module. This results in a conservation of quite an amount of memory of the MicroMoPS. Also, by automating the processing of data, the method of manual feed of scaling values is avoided, which helps in minimizing human error.

6.1 Answers to the Research questions

The above analysis and evaluation of the results bring a conclusion to the raised research questions (see Section 1.2).

- By analysis of the various existing application boards, a linear model solution could be found. The performance of the linear model over the Reverse Polish Notation model is greatly enhanced.

- The derivation of a linear scaling mechanism and their precise conversion of analog measurements into control module-specific voltages, proves the possibility of a linear scaling mechanism in this stress test environment.

- The resolution of board UIDs and the filtering of scaling values establishes the automation of data processing. This exhibits the possibility of automated processing of analog measurements in this stress test environment.

7 Summary and Outlook

7.1 Summary

In this sophisticated stress test environment to improve the data processing mechanism, the electrical circuits that are associated with measuring the physical quantities that generate in a semiconductor, are analyzed. A classical linear scaling mechanism is introduced to the data processing system of a control module by analyzing the measurement circuits. Also, the Reverse Polish Notation scaling computation function is incorporated into the data processing system of the control module which opens up the possibility of performing a complex level of scaling functions.

The boards that are present in the stress test environment are identified by the extraction of the respective board ids. The extraction of board ids and their correspondence in the resolution of board information coming from different sources such as test plan and MoPS web server, led to the automation of processing of analog measurements that generate in the semiconductor, irrespective of the stress test application.

The automation in the processing of analog measurements has simplified the writing procedure of test plans and manual scaling is no more involved within the test plans. Human error is minimized because of the automated method of scaling. The improved data processing mechanism is implemented and tested in the KAI lab.

7.2 Outlook

Real-time efficient scaling mechanism and automation of processing of analog measurements provide gateway for conducting advanced application stress testing in the stress test environment. The newly introduced scaling mechanism known as Reverse

Polish Notation can be used for further improvement of a scaling function. The gain function introduced by a differential operational amplifier itself could be a non-linear function. The differential input/output impedance of an operational amplifier can be of a significant effect in calculation of discrete voltage values specfic to control module. The internal circuits of the differential operational amplifier can be analysed and the contribution from its gain function in performing the scaling can be described using Reverse Polish Notation. The implementation of the Lua interpreter influences in the real-time performance of the Reverse Polish Notation because the RPN uses Lua stack to perform the scaling operation. Lua interpreter even though it utilizes a minimum of RAM because of its implementation in Lua, there is a scope to optimize the implementation of the Lua interpreter. Optimization of the implementation of the Lua interpreter results in the improved execution time performance of the Reverse Polish Notation. Similarly, the execution time performance of Reverse Polish Notation could significantly be improved by replacing the division operation that takes place during the scaling computation by right shifting 12 times the digital output along with some operation to handle the decimal result.

Bibliography

[1] B. Steinwender, "A Distributed Controller Network for Modular Power Stress Tests," Ph.D. dissertation, Alpen-Adria-Universität Klagenfurt, Jun. 2016 (cit. on pp. III, 3, 14, 17, 24, 28, 51, 56, 59).

[2] R. Sleik, M. Glavanovics, S. Einspieler, A. Muetze, and K. Krischan, "Modular Test System Architecture for Device, Circuit and System Level Reliability Testing," in *Proceedings of the 31st annual IEEE Applied Power Electronics Conference and Exposition (APEC 2016)*, Long Beach Convention Center, California, USA: IEEE, Apr. 2016, pp. 759–765. DOI: 10.1109/APEC.2016.7467957 (cit. on pp. 1, 3, 16, 36).

[3] K. Plankensteiner, "Test Plan Generation and Verification for a Modular Power Stress Test System," M.S. thesis, Graz University of Technology, 2015 (cit. on pp. 2, 14, 27).

[4] B. W. Kernighan and D. M. Ritchie, *The C programming language*, 2nd ed. London, Englewood Cliffs, NJ: Prentice Hall, 1988 (cit. on p. 6).

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Addison-Wesley Professional, 1994, p. 416, ISBN: 978-0201633610 (cit. on p. 7).

[6] *Actor Framework*, National Instruments, Nov. 2012. [Online]. Available: http://ni.com/actorframework (visited on 11/21/2012) (cit. on pp. 7, 24).

[7] S. R. Mercer and A. C. Smith, *Using the Actor Framework in LabVIEW*, 2011. [Online]. Available: https://readthedocs.web.cern.ch/download/attachments/5571215/Using%20the%20Actor%20Framework%203.0%20in%20LabVIEW.pdf?version=1&modificationDate=1390810919000&api=v2 (visited on 11/08/2019) (cit. on p. 7).

[8] C. Hewitt, *Actor Model of Computation: Scalable Robust Information Systems*, arXiv:1008.1459v38 [cs.PL], Jan. 21, 2015. [Online]. Available: https://arxiv.org/abs/1008.1459v38 (visited on 09/06/2018) (cit. on p. 7).

[9] *I2C-bus specification and user manual*, 6th ed., NXP Semiconductors, 2014 (cit. on p. 8).

[10] *DS28CM00 - I²C/SMBus Silicon Serial Number*, Rev. 072406, Maxim Integrated, 2006 (cit. on pp. 9, 49, 50).

[11] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5 ed., H. Michael, Ed. Prentice Hall, 2010, ISBN: 0-13-212695-8 (cit. on p. 9).

[12] J. D. Day and H. Zimmermann, "The OSI reference model," *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1334–1340, Dec. 1983, ISSN: 0018-9219. DOI: 10.1109/PROC.1983.12775 (cit. on p. 9).

[13] D. A. Rauth and V. T. Randal, "Analog-to-digital conversion. part 5," *IEEE Instrumentation & Measurement Magazine*, vol. 8, no. 4, pp. 44–54, Oct. 2005. DOI: 10.1109/mim.2005.1518622 (cit. on p. 10).

[14] F. Miller, A. Vandome, and J. McBrewster, *Nyquist-Shannon Sampling Theorem: Aliasing, Sine Wave, Signal Processing, Nyquist Rate, Nyquist Frequency, Sampling Rate, Shannon-Hartley Theorem, Whittaker-Shannon Interpolation Formula, Reconstruction from Zero Crossings*. Alphascript Publishing, 2010, ISBN: 9786130045814 (cit. on p. 11).

[15] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development*. O'Reilly Media, Inc., 2012 (cit. on p. 13).

[16] *The JSON Data Interchange Format*, ECMA International, 2013. [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf (visited on 11/02/2017) (cit. on p. 13).

[17] R. Sleik, "System Level Reliability Testing Under Application Stress Conditions," Ph.D. dissertation, Graz University of Technology, Jun. 2018 (cit. on pp. 16, 20, 32, 43).

[18] B. Steinwender, S. Einspieler, M. Glavanovics, and W. Elmenreich, "Distributed power semiconductor stress test & measurement architecture," English, in *Proceedings of the 11th IEEE International Conference on Industrial Informatics*, Bochum, Germany, Jul. 2013, pp. 129–134. DOI: 10.1109/INDIN.2013.6622870 (cit. on p. 17).

[19] *XMC4700/XMC4800 Reference Manual*, Infineon Technologies, 2016 (cit. on p. 18).

[20] R. Ierusalimschy, L. H. de Figueiredo, and W. C. Filho, "Lua—an extensible extension language," *Software: Practice and Experience*, vol. 26, no. 6, pp. 635–652, 1996, ISSN: 1097-024X. DOI: 10.1002/(SICI)1097-024X(199606)26:6<635::AID-SPE26>3.0.CO;2-P (cit. on p. 23).

[21] B. Steinwender, M. Glavanovics, and W. Elmenreich, "Executable Test Definition for a State Machine Driven Embedded Test Controller Module," in *Proceedings of the 13th IEEE International Conference on Industrial Informatics*, 2015. DOI: 10.1109/INDIN.2015.7281729 (cit. on p. 23).

[22] *PCA951A - Dual Bidirectional I2C Bus and SMBus Repeater*, Texas Instruments, 2014 (cit. on p. 49).

Acronyms

µC Micro Controller.

API Application Programming Interface.

ARCTIS Advanced Repetitive Clamping Test Integrated System.

BCET Best-Case Execution Time.

DAQ Data AcQuisition.

DAVE Digital Application Virtual Engineer.

DSD Delta-Sigma Demodulator.

DUT Device Under Test.

EDS Electronic Data Sheet.

FSM Finite-State Machine.

GPIO General Purpose Input Output.

GUI Graphical User Interface.

I^2C Inter-Integrated Circuit.

Idrv Driver current.

Iin Converter input current.

Imon IC current monitor.

Iout Converter output current.

LabVIEW Laboratory Virtual Instrument Engineering Workbench.

LED Light Emitting Diode.

LSB Least Significant Bit.

LUT LookUp-Table.

MoPS Modular Power Stress.

MOSFET Metal-Oxide Semiconductor Field Effect Transistor.

MSB Most Significant Bit.

MTS Modular Test System.

PoL Point-of-Load.

PWM Pulse Width Modulation.

RPN Reverse Polish Notation.

SAM Software Architecture for MoPS.

SCL Serial Clock Line.

SDA Serial Data Line.

SoC System on Chip.

SPI Serial Peripheral Interface.

Tcase DUT case temperature.

Tmon IC temperature monitor.

UDP User Datagram Protocol.

UID Unique Identification.

USB Universal Serial Bus.

Vdrv Driver voltage.

VI Virtual Instrument.

Vin Converter input voltage.

Vout Converter output voltage.

WCET Worst Case Execution Time.