Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik

Lehrstuhl für Messsystem- und Sensortechnik
Prof. Dr.-Ing. habil. Dr. h.c. Alexander W. Koch

Master's Thesis

3-Channel Solar Array Simulator for CubeSat Power Budget Verification

Daniel Nagy

September 2018

Supervisors:
Dr.-Ing. Markus P. Plattner
Jonis Kiesbye

Acknowledgments

The results presented in this thesis are the outcome of multiple people’s work. In the following, I would like to mention those who had an important role in delivering these results and I would like to express my gratitude to them. I thank:

- Amine Ben Mhenni, who introduced me to the MOVE-II project.
- Lucas Krempel, who was my supervisor in the MOVE-II Payload team and provided valuable information about the satellite’s solar cells. He also organized the purchase of the hardware components used in this project.
- Rupert Amann, who created the first solar array simulator for MOVE-II and gave me useful explanations about the satellite’s electrical power system.
- The HiL Team, that is, David Meßmann, Maximilian Preislinger, Jonis Kiesbye, Tejas Kale and Martin Mostad, who created the HiL environment and thus provided the necessary conditions for the solar array simulator to have a useful role. Their weekly updates about the system and their feedback on my work were indispensable to the project. Maximilian Preislinger made many of the changes to the Simulink model that were necessary for realizing the network communication with the solar array simulator. Tejas Kale provided me with several Linux-related hints which were extremely useful when writing the software for the device. Jonis Kiesbye coordinated the HiL team, made the necessary tools available and co-supervised the thesis.

Finally, I would also like to say thank you to the entire MOVE-II team who built the satellite. Without their effort there would never have been room for the development of the solar array simulator.

Zusammenfassung

Das Power Budget eines Weltraumfahrzeugs ist die Bilanz seiner eingesammelten und verbrauchten Energie und muss für eine erfolgreiche Mission genau geplant werden. Wenn die eingesammelte Energie, die üblicherweise durch Solarzellen erzeugt wird, über der benötigten Energie liegt, wird das Power Budget als positiv bezeichnet. Jedoch ist es oft sehr kompliziert zu beweisen, dass das Power Budget eines Gerätes positiv ist, wegen der vielen relevanten Faktoren, wie z.B. der Leistungsverbrauch, die Menge der verfügbaren Sonnenenergie oder die Effizienz, mit der diese Energie eingesammelt wird. Deshalb ist eine Methode, die die Verifizierung des Power Budgets durch Messungen statt mathematischer Berechnungen erlaubt, sehr nützlich.

In dieser Masterarbeit wird ein Solarzellensimulator entwickelt, der die Verifikation des Power Budgets von MOVE-II ermöglicht, einem an der Technischen Universität München von Studenten gebauten Mikrosatelliten. Ein Solarzellensimulator ist ein spezielles Netzteil, das das Verhalten von Solarzellen nachahmt, statt feste Spannungen und Ströme zu erzeugen. Der hier entwickelte Solarzellensimulator wird Teil einer Simulink-basierten Hardware-in-the-Loop (HiL) Testumgebung, welche direkt mit dem Engineering Model von MOVE-II zusammenwirkt. Ein Computer im HiL-System simuliert die Umlaufbahn des Satelliten und der Solarzellensimulator reproduziert dementsprechend das elektrische Verhalten der Solarzellen, von denen MOVE-II im normalen Betrieb durch Strom versorgt wird. So kann das Power Budget durch Beobachtungen überprüft werden.

Ein HiL-System für MOVE-II ist bereits vorhanden, und es wurde ursprünglich konstruiert, um das Lageregelungssystem des Satelliten zu testen. Zunächst wird es geplant, wie dieses System erweitert werden kann, um Power-Budget-Tests ausführen zu können. Danach wird die Spezifikation des Solarzellensimulators definiert, basierend auf den Eigenschaften des Stromversorgungssystems von MOVE-II und der zu simulierenden Solarzellen. Als nächster Schritt werden die einzelnen logischen Einheiten des Geräts entworfen, wie z.B. die analoge Ausgangsstufe, oder die Hardware und die Software des digitalen Controllers. Das Gerät wird dann gebaut, programmiert und getestet.

Um das Verhalten der Solarzellen von MOVE-II möglichst genau nachzubilden, wird eine mathematische Methode für die Berechnung der Strom-Spannung-Charakteristik der Zellen vorgestellt, implementiert und evaluiert. Danach wird der Simulator mit einem Netzwerkinterface versehen, das die Kommunikation mit dem Simulationsrechner ermöglicht.

Das Gerät wird dann in die HiL-Umgebung integriert und es wird zum Ausführen von Tests benutzt, welche ein positives Power Budget für die zwei wichtigsten Betriebsmodi des Satelliten verifizieren. Schließlich werden Möglichkeiten zur künftigen Verbesserung des Gerätes diskutiert.

Abstract

The power budget of a spacecraft is a plan about the balance of its incoming and consumed energy and needs careful planning for a successful mission. The required energy is usually collected using solar cells. If the harvested solar energy is more than what is needed to complete the mission, then the spacecraft is called power positive. However, proving power positivity is not an easy task due to the complexity of the factors that influence the amount of incoming and outgoing energy, such as the spacecraft's power consumption profile, the amount of available solar energy and the efficiency of energy harvesting. Therefore, a tool for verifying a power budget through physical measurements rather than mathematical computations, is extremely useful.

In this thesis project, a solar array simulator is developed, which supports the power budget verification of MOVE-II, a microsatellite built by students at the Technical University of Munich. A solar array simulator (SAS) is a special power supply, which, instead of providing constant voltage or current, tries to mimic the behavior of an array of photovoltaic cells (a solar array). The SAS developed here integrates into a Simulink-based hardware-in-the-loop (HiL) simulation environment that physically interacts with the satellite's engineering model. A computer in the HiL setup simulates the satellite orbit and the SAS reproduces the electrical behavior of the solar cells that MOVE-II is normally powered from. This way, the power budget of the satellite can be verified by observations.

First, the structure of the test setup that should be used for the power budget verification is planned based on an existing HiL environment, which was originally created to test the attitude determination and control system of MOVE-II. Then, requirements against the SAS are determined by investigating the properties of the MOVE-II electrical power system (EPS) and the satellite's solar cells. Next, the problem is broken down into the design of multiple submodules of the simulator, such as its analog output stage, its digital controller unit and different levels of software running on the controller. The device is then constructed, programmed and tested.

In order to accurately reproduce the behavior of the MOVE-II solar cells, a mathematical method to determine the cells' voltage-current characteristic curve is presented, implemented and evaluated. As the final design step, the SAS is equipped with a network interface that allows it to interact with the HiL simulation.

The device is then integrated into the HiL setup and is used to execute power budget verification tests, which show power positive results for two of the most important operational modes of the satellite. Finally, possibilities for future improvement of the device are discussed.

Table of Contents

List of Figures

List of Tables

Symbols

Symbol	**Unit**	**Description**
A	m^2	Surface area
a	–	Diode ideality factor
β	–	Transistor DC current gain
$\epsilon_{\mathrm{I,rel}}$	–	Relative current error
$\epsilon_{\mathrm{V,rel}}$	–	Relative voltage error
η	–	Efficiency
$\eta_{\mathrm{MPPT,a}}$	–	Accumulated maximum power point tracking efficiency
$\eta_{\mathrm{MPPT,i}}$	–	Instantaneous maximum power point tracking efficiency
$f_{\mathrm{u,VI}}$	Hz	VI curve update frequency
G	$\mathrm{W/m}^2$	Solar intensity
I	A	Current
I_0	A	Diode reverse saturation current
I_{MPP}	A	Maximum power point current
I_{pv}	A	Photovoltaic current
I_{sc}	A	Short-circuit current
k	J/K	Boltzmann constant, $k = 1.38 \cdot 10^{-23}$ J/K
K_{I}	A/°C	Temperature coefficient of short-circuit current
K_{V}	V/°C	Temperature coefficient of open-circuit voltage
N_{VI}	–	Number of slots for VI curves in memory
P	W	Power
q	C	Unit charge, $q = 1.602 \cdot 10^{-19}$ C
R_{p}	Ω	Equivalent parallel output resistance
R_{s}	Ω	Equivalent series output resistance
R_{θ}	°C/W	Thermal resistance
T	K	Absolute temperature
T_{J}	K	Junction temperature
τ_{s}	s	Settling time
V	V	Voltage
$V_{\mathrm{GS(th)}}$	V	Gate-to-source threshold voltage
V_{MPP}	V	Maximum power point voltage
V_{oc}	V	Open-circuit voltage
V_{os}	V	Input offset voltage
V_{T}	V	Thermal voltage

Abbreviations

1U	Single Unit
3J	3-Junction
4J	4-Junction
ADC	Analog-to-Digital Converter
ADCS	Attitude Determination and Control System
AM	Air Mass
BCR	Battery Charging Regulator
BoL	Beginning of Life
CDH	Command and Data Handling
COM	Communication Subsystem
CS	Current Source
DAC	Digital-to-Analog Converter
DUT	Device Under Test
EM	Engineering Model
EoC	End of Charge
EPS	Electrical Power System
FM	Flight Model
FP	Flappanel
FS	Full Scale
GPIO	General-Purpose Input/Output
HiL	Hardware in the Loop
HRC	High Rupturing Capacity
LDO	Low Dropout
LSB	Least Significant Bit
LUT	Lookup Table
MISO	Master In Slave Out
MOSFET	Metal-Oxide Semiconductor Field Effect Transistor
MOSI	Master Out Slave In
MPP	Maximum Power Point
MPPT	Maximum Power Point Tracking
op-amp	Operational Amplifier
PCB	Printed Circuit Board
PL	Payload
PRU	Programmable Real-Time Unit
PSU	Power Supply Unit
PV	Photovoltaic
PWM	Pulse Width Modulation
RRIO	Rail-to-Rail Input-Output
SA	Solar Array
SAS	Solar Array Simulator
SCK	Serial Clock
SoC	System on a Chip
SP	Sidepanel
SPI	Serial Peripheral Interface
SS	Slave Select
SSH	Secure Shell
TCP	Transmission Control Protocol
TLE	Two-Line Element Set
UDP	User Datagram Protocol
VI	Voltage-Current
VS	Voltage Source

1 Motivation

1.1 The MOVE-II CubeSat Project

The development of MOVE-II, the second satellite produced at the Technical University of Munich, started in April 2015. It is built by around 60 members of the Scientific Workgroup for Rocketry and Space Flight[1], a student organization within the Chair for Astronautics[2]. MOVE-II is the successor of the university's first satellite, called First MOVE, and just like its predecessor, it is based on the CubeSat model, a standard for small satellites primarily intended for educational purposes. Being a single-unit (1U) CubeSat, MOVE-II is nearly cubical, with approximate dimensions of $10 \times 10 \times 10\ \mathrm{cm}$ and a total mass of $1.2\ \mathrm{kg}$. The purpose of the project is to provide hands-on experience for students interested in building space hardware, and to verify the satellite bus (the means of data transport between the components) of MOVE-II in order to employ it in larger-scale projects later. Moreover, the satellite carries a set of newly-developed solar cells as its payload and it is going to characterize their behavior in space for the first time. The flight model (FM) of the device, shown in Fig. 1.1, is going to be launched in November 2018, while an electrically identical instance, the engineering model (EM) will be kept at the ground station for further testing [1].

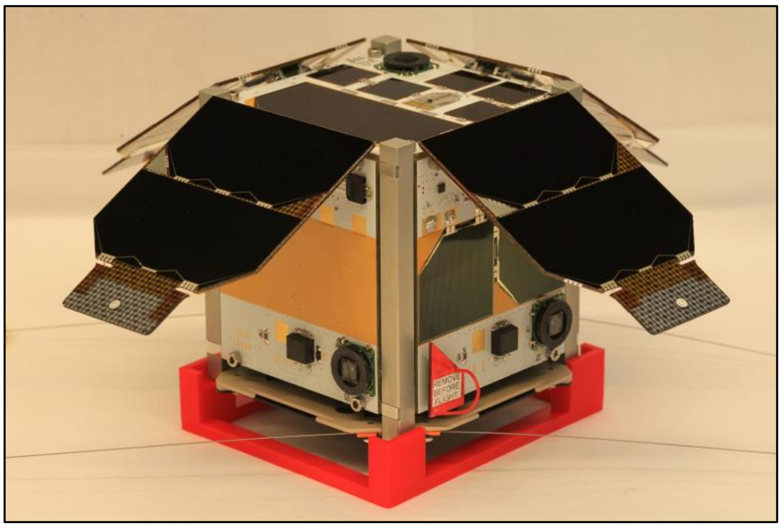


Figure 1.1 – The flight model of MOVE-II

MOVE-II consists of different subsystems. These include, amongst others, the Communication subsystem (COM) for interacting with the ground station from which the spacecraft is operated; the Attitude Determination and Control System (ADCS) for controlling the satellite's orientation in 3D space; the Payload (PL); and the Command and Data Handling subsystem (CDH) which controls all other subsystems. In the focus of this thesis is the Electrical Power System (EPS), which is responsible for harvesting energy from the satellite's solar cells, charging this energy into the device's battery and supplying power to all other subsystems.

[1] *Wissenschaftliche Arbeitsgemeinschaft für Raketentechnik und Raumfahrt* (WARR)
[2] *Lehrstuhl für Raumfahrttechnik* (LRT)

The satellite has been designed with a pre-defined power budget in mind, which means that the maximal allowed average power consumption of each subsystem was determined based on the anticipated solar energy collected by the EPS during the orbit. The collected solar energy and the exact amount of power consumed by the subsystems depend on several factors. The most important ones are the satellite's orientation relative to the Sun in its orbit, the characteristics of the solar cells, the exact behavior of all subsystems and the efficiency of the EPS. Due to the high number and complexity of the relevant factors, it is difficult to make precise, purely computation-based estimates about the power budget. Hence, an alternative approach is needed, which will be presented next.

1.2 The Hardware-in-the-Loop Simulation Setup

When testing a satellite, it is nearly impossible to reproduce all aspects of the space environment in a laboratory on Earth (for example, we cannot eliminate gravity). A commonly employed solution to this problem is to build a test system in which the hard-to-replicate aspects of the space environment are modeled and simulated in software, but several, preferably all components of the satellite are inserted into the system as they are, i.e. in hardware. Usually, it is not possible to include all satellite components in such a test environment in hardware – those directly interacting with the simulated aspects of the space environment may have to be modeled and simulated too. For example, if the temperatures that the device under test (DUT) would experience in its orbit are simulated, and not generated physically, then the temperature sensors of the DUT (if any) also need to be modeled and must be substituted by emulator modules which imitate the sensors' output based on the simulated temperature values. Nevertheless, this approach eliminates the need to model and simulate those components of the DUT which are included in the setup as hardware. Such a setup is called a Hardware-in-the-Loop (HiL) simulation environment and it allows a much more precise analysis of a satellite's behavior than purely software-based simulations, since parts of the actual hardware are included in the system.

A HiL environment for MOVE-II already exists and it was built to verify its ADCS (Attitude Determination and Control) subsystem. This subsystem uses various sensors, such as sun sensors and magnetometers to determine the satellite's attitude (i.e. its rotation angle) e.g. by measuring the incident angle of sunlight and the magnitude and direction of the Earth's magnetic field. Then, magnetorquers (coils) are used to generate a magnetic field, which, by interacting with the Earth's own magnetic field, adjusts the orientation of the satellite to a desired angle, e.g. such that the solar cells are pointed against the Sun for maximal illumination. In the HiL environment, the sensors and the magnetorquers are simulated in software, since it would be very difficult to replicate the excitation of the sensors and to create a suspension on which the magnetorquers can rotate the satellite around all three axes. However, the controller unit of the ADCS, which interprets the sensor outputs and generates the magnetorquer actuation commands, is included as hardware. This means that the ADCS peripherals, i.e. the sensors and the actuators are detached from the satellite and are substituted by emulators, which behave based on models of the peripherals implemented in software. Meanwhile, the ADCS control unit is left in place and now communicates with the aforementioned peripheral emulators. This provides a reliable means of testing the ADCS control algorithms [2].

A similar approach can be used to verify the power budget of MOVE-II. It is possible to insert the EPS and the cores of the rest of the subsystems as hardware into the HiL environment. This eliminates the necessity to model how and with what efficiency the EPS charges energy into the satellite's battery and how the other subsystems consume this energy, removing considerable complexity from the power budget verification problem. However, it is hard to use the actual solar cells of the satellite for energy input during the test, since the in-orbit illumination conditions are not easily reproduced on Earth. Hence, the quantities influencing the power output of the solar cells (their illumination and temperature) need to be simulated in software and a device emulating the cells' output under the simulated conditions has to be used in their place.

This motivates the application of a programmable solar array simulator (SAS)[3] which can receive data from the existing HiL setup and imitate the behavior of the solar cells of MOVE-II. A solar array simulator would make it possible to verify the satellite's power budget with the HiL approach, eliminating the need to model its power consumption and the behavior of its EPS. This would provide more reliable estimations about the power budget than those that currently exist. The construction of such a device is the topic of this thesis.

1.3 Behavior of a Solar Array and its Simulation

A solar or photovoltaic (PV) cell is a semiconductor (e.g. Si or GaAs) based device which converts the energy of light into electrical energy. Individual solar cells are most often connected in series (sometimes in parallel) to form a solar or photovoltaic panel with a higher output voltage (or current, respectively). Commercially available solar panels are usually packaged assemblies with two terminals. When illuminated, the photovoltaic effect generates a so-called photo current within the panel, part of which will flow through the load connected to its output terminals, hence liberating electric power. Solar panels can again be combined in series and/or parallel to form a so-called solar array. For instance, two strings connected in parallel, where each string consists of two serially connected solar panels, form a solar array (see Fig. 1.2). It should be noted that in practice, the terms "solar cell", "panel" and "module" are often used interchangeably.

Photovoltaic cells exhibit nonlinear voltage-current characteristics. This means that the instantaneous output current of an illuminated solar cell is a nonlinear function of its instantaneous output voltage. Under a fixed illumination level and temperature, the output voltage and current (the operating point) of the PV cell is determined by its connected load. If the output voltages and the corresponding currents are measured under different loads, the current-versus-voltage characteristic curve (VI curve) of the cell can be drawn. The VI curve of a solar cell depends on the cell's physical attributes (material, size, etc.) as well as on the level of its illumination and on its temperature. Physically, the illuminating light is an electromagnetic wave whose energy transport rate (power) per unit area, i.e. its energy flux is represented by its Poynting vector. The illumination level (also called solar flux, solar intensity, insolation or irradiation) is the magnitude of this vector, is marked with G and measured in $\mathrm{W/m^2}$.

[3] The usual name for a device emulating an array of solar panels is solar array simulator, although they should more precisely be called solar array emulators. The standard term 'simulator' will be used throughout this text too.

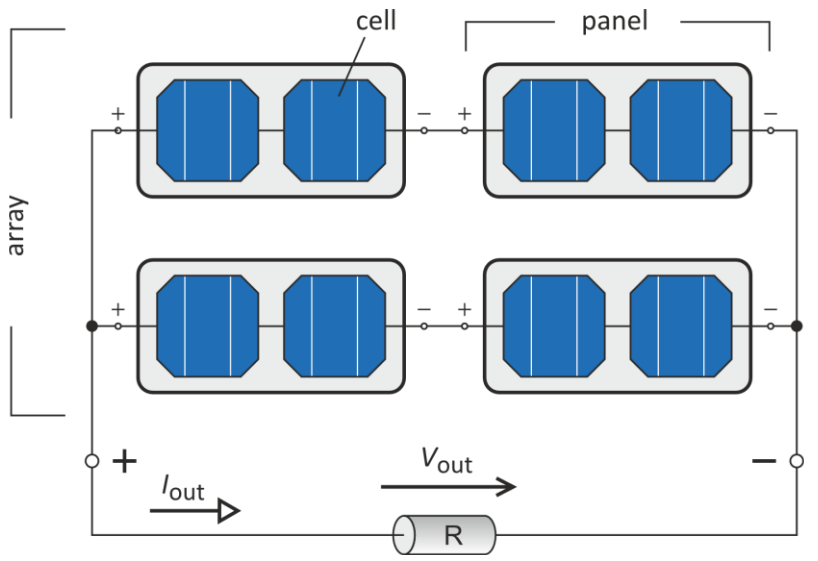


Figure 1.2 – Example for the structure of a solar array

VI curves of PV panels are similar to those of PV cells, but with higher voltages or currents, depending on how the cells are connected. Likewise, VI curves of solar arrays depend on the characteristics of its individual panels and their connection topology. As an example, the VI curve of the commercially available Sunpower E20/435 solar panel [3] is shown in Fig. 1.3 for irradiation $G = 1\,\mathrm{kW/m^2}$ and cell temperature $T = 25\,°\mathrm{C}$.[4] The operating point for a given load, as well as the curve's remarkable points are also marked on the graph.

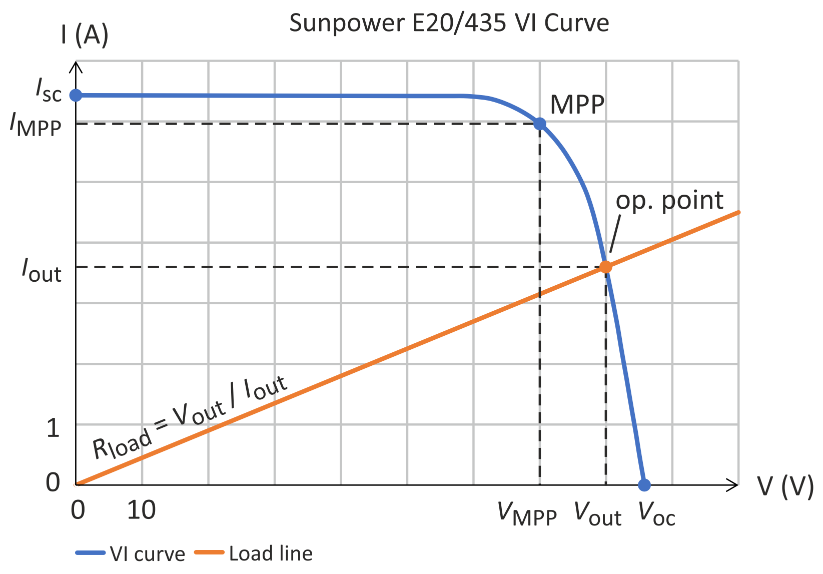


Figure 1.3 – VI curve of the Sunpower E20/435 panel

There are three remarkable points on a VI curve: the open-circuit point, the short-circuit point and the maximum power point (MPP). At the open-circuit point, the PV device is unloaded, hence its output current is zero and its output voltage

[4] The output of a PV device also depends on the spectrum of illuminating light. The data on Fig. 1.3 is given for the AM1.5 (air mass 1.5) spectrum, which is the spectrum of sunlight experienced by a panel tilted by 37° from the surface of Earth while facing the sunrays perpendicularly [5][36].

equals to its maximal, or open-circuit voltage (V_{oc}). At the short-circuit point the device is short-circuited and delivers its maximum or short-circuit current (I_{sc}) with zero output voltage. The maximum power point is where the product of the device's output voltage and current, i.e. the power delivered to the load, is maximal. The voltage and the current at the MPP are marked with V_{MPP} and I_{MPP}. The region around the MPP is referred to as maximum power region. Between the open-circuit point and this region, the PV device behaves roughly like a constant current source, with output current close to I_{sc}. Between the maximum power region and the open-circuit point it behaves roughly like a constant voltage source, with output voltage close to V_{oc}. Both V_{oc} and I_{sc}, and thus the entire VI curve depend on temperature and irradiation, with I_{sc} approximately being a linear function of irradiation and V_{oc} increasing roughly linearly with decreasing temperature [4]. It should be noted that the VI characteristics of a solar array whose panels are illuminated at different intensities (a partially shaded solar array) can have a considerably different shape than the one shown in Fig. 1.3. VI curves and partial shading will be further discussed in Section 6.2.

Since the instantaneous output of a PV device depends on its operating point, which is governed by the load, a solar array simulator should be able to determine and constantly track its load and adjust its output accordingly. Moreover, a flexible solar array simulator must be able to change its VI curve depending on the programmed environmental variables (illumination and temperature) and the type of PV device it is simulating. An SAS compatible with the HiL environment discussed above should have all of these capabilities, because the environmental variables change during the orbit and MOVE-II is equipped with different types of solar cells, connected into arrays of different topologies.

2 State of the Art

2.1 Introduction

Solar array simulators or PV simulators are specialized power supplies which imitate the behavior of photovoltaic devices. Generally, they are used to test systems which are normally powered from some arrangement of solar cells. In particular, the capability of a device to efficiently harness the available solar power under various conditions can be tested with them, without the need to physically recreate these conditions (e.g. produce different lightings). In this chapter, existing solar array simulators will be reviewed to serve as a comparison to the design presented later.

2.2 Figures of Merit of Solar Array Simulators

Before examples of solar array simulators are presented, let us define some figures of merit of these devices so that we can evaluate and compare their performance later. Four of the most important features of a photovoltaic simulator are the following:

- *VI curve modelling accuracy and flexibility*: The VI curve modelling accuracy of an SAS shows how closely the generated VI curve matches that of the simulated PV device. The modelling flexibility shows how freely the output curve can be defined, e.g. can the device output any arbitrary VI curve or just a certain class of curves, etc.
- *Settling time* (τ_s): This is the time needed for an SAS to stabilize its output voltage and current onto a new operating point on the VI curve after its load has changed abruptly. The shorter this time, the faster the simulator is and the better it resembles a real PV device in terms of transient response.
- *VI curve update frequency* ($f_\mathrm{u,VI}$): This attribute shows how quickly the device can change its simulated curve. For example, if the simulated curve can be changed every $100\ \mathrm{m}s$, then $f_\mathrm{u,VI} = 10\ \mathrm{Hz}$.
- *Maximum ratings*: These values specify the maximum open-circuit voltage ($V_\mathrm{oc,max}$) and short-circuit current ($I_\mathrm{sc,max}$) the device can simulate, as well as its maximum output power (P_max).

2.3 The Current MOVE-II Solar Array Simulator

A simple solar array simulator for MOVE-II, called SaS V1 already exists, and it is used primarily to keep the batteries of the satellite's engineering model charged. Internally, it is a voltage source based on a switch-mode DC to DC converter in series with a fixed output resistor. This setup results in a single straight-line approximation of a VI curve with V_oc and I_sc determined by the voltage of the voltage source (V_0) and the resistance of the series resistor (R_s), as shown in Fig. 2.1. The average power delivered to the load can be set by turning the voltage source on and off in a pulse-width modulated (PWM) fashion. In the actual implementation, the device has 3 output channels with $V_0 = 15\ \mathrm{V}$ and $R_\mathrm{s} = 3.3\ \Omega$, and the PWM period is 1 minute [6][7].

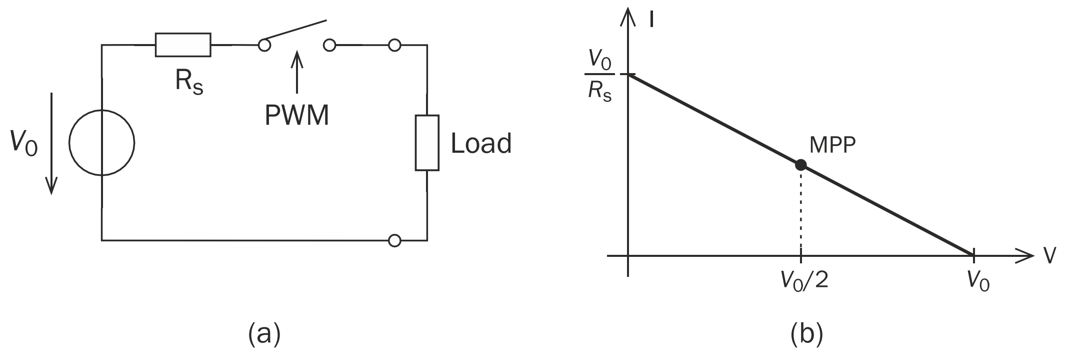


Figure 2.1 – Operating principle (a) and on-state output curve (b) of the SaS V1

The device is programmable and can change its average output power based on a user-defined list of values which specify time intervals and the PWM duty cycles that should be used during each of them. This allows different orbital scenarios to be simulated, i.e. orbits with more or less amount of sunlight reaching the solar cells. The SaS V1 is controlled via a web-based interface where also the state of its outputs can be monitored. The device is depicted in Fig. 2.2.

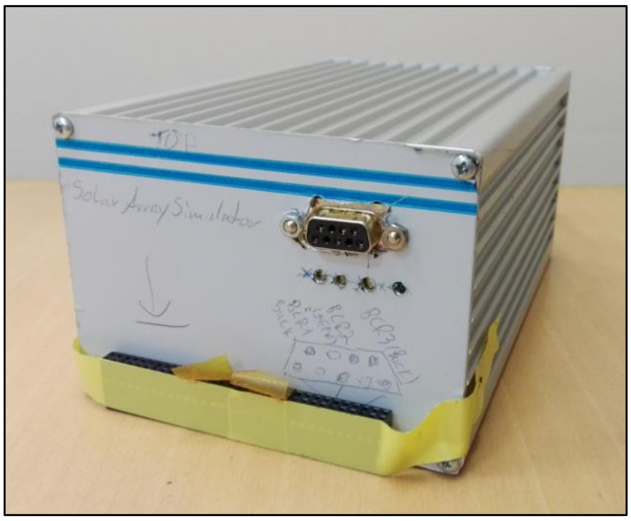


Figure 2.2 – Photo of the SaS V1

2.4 Commercially Available Models

Several companies offer solar array simulators in their product portfolio. Notable examples are the E4360 series from Keysight, the 62000H-S series from Chroma and the model SAS12010 from Aplab. The key features of the base models from each manufacturer are summarized in Table 2.1 [8][9][10]. Photos of the simulators are displayed in Fig. 2.3.

Table 2.1 – Comparison of commercial solar array simulators

Model	$V_{\mathrm{oc,max}}$	$I_{\mathrm{sc,max}}$	P_{max}	$f_{\mathrm{u,VI}}$	N_{VI}
Keysight E4361A	65 V	8.5 A	510 W	33 Hz	512
Chroma 62020H-150S	150 V	40 A	2 kW	1 Hz	100
Aplab SAS12010	121 V	10.5 A	1.2 kW	0.5 Hz	50

In the table above, $f_{\mathrm{u,VI}}$ is the VI curve update frequency (see earlier), while N_{VI} shows how many VI curves can be held in the device's memory at a time that can be chosen from when simulating varying environmental circumstances. For all three models, the user can define the VI characteristics based on either the remarkable points of the curve or a table of arbitrary voltage-current pairs. The average asking price of the units listed above is in the order of ten thousand US dollars.

The MOVE-II HiL system simulates new environmental variables with a frequency of $5\ \mathrm{Hz}$, meaning that the SAS from Chroma and Aplab would not be able to keep up with the pace of the simulation directly, since in their case $f_{\mathrm{u,VI}}$ is below $5\ \mathrm{Hz}$. Moreover, to cover all possible combinations of the environmental variables, the number of different VI curves stored by the devices at a time (N_{VI}) may not suffice. In spite of these limitations, the devices might be useable to verify the power budget of MOVE-II. On the one hand, an acceptably accurate verification might be achievable even with lower update speeds and with only a moderate resolution between the covered VI curves, and on the other hand the numbers given above might result from limitations in the devices' user interfaces only. This means that better performance might be possible if the instruments are commanded programmatically via their various I/O ports (Ethernet, USB, etc.).

Figure 2.3 – Commercial solar array simulators
From left to right: Keysight E4361A, Chroma 62020H-150S and Aplab SAS12010.
Source: keysight.com, chromaate.com, aplab.com

2.5 Simulators Developed in Scientific Research

Let us review some solar array simulator designs presented in research papers. The purpose of these simulators is usually to prove the feasibility of a new method which results in improved characteristics in one certain aspect of VI curve emulation, e.g. high accuracy, high speed or low cost and ease of construction of the simulator.

The design shown in [11] uses a programmable laboratory power supply (PSU) to implement an SAS. The power supply is used as the output stage of the SAS and for sensing the instantaneous output voltage and current. The control algorithm, which programs the desired output of the power supply, runs on a PC. The advantage of the design is that it may be implemented with any laboratory power supply that can be commanded from a computer with no additional components other than the software running on the controlling PC. However, tests have shown that the approach results in high settling times (slow VI curve tracking) with τ_{s} in the order of $500\ \mathrm{ms}$, mainly due to the communication latency between the PC and the power supply.

To mitigate the above problem, the authors of [12] have shown how to modify a laboratory power supply so that its output voltage can be set by injecting an external voltage into its internal feedback loop. This results in direct and instantaneous control over the output voltage and therefore much lower settling times can be achieved, with τ_s being some tens of milliseconds. Since there is no control over the output current of the PSU with this approach, the design suffers from decreased accuracy and potential instability near the short-circuit point.

In some form, all solar array simulators need to have information about the VI curve they simulate, in order to know what output to set for a certain load condition. The designs described so far reference the VI curve in form of a lookup table (LUT), so for instance, they store the currents corresponding to each voltage on the VI curve digitally with a certain resolution. Analog-to-digital and digital-to-analog conversions necessary to utilize this LUT, as well as reading the LUT itself, slow down the regulation of the output.

Extremely fast PV simulators (ones with $\tau_s = 1\ \mathrm{ms}$ or lower) can be constructed by implementing the VI curve reference as an analog circuit. Since a diode's current-voltage characteristics resemble the VI curve of a solar cell, diodes are often used as the base of analog VI curve references. The design presented in [17] uses a so-called amplified diode chain to "query" the corresponding voltage for any current of the simulated VI curve. While the design is fast thanks to its fully analog nature, the output curve is completely governed by the diodes and hence it is inflexible. The ability to simulate at least different solar intensities is introduced by the method shown in [13], where photodiodes are used instead of traditional diodes. The simulation of different solar insolation levels is achieved by illuminating the photo diodes with LEDs of varying brightness.

The authors of [18] implemented an analog VI curve reference circuit using operational amplifiers (op-amps). This adds flexibility to the design and it was shown that using the proposed circuit topology, the open-circuit voltage, the short-circuit current, and to some extent the shape of the VI curve can be adjusted independently. Unfortunately, these parameters are controlled by resistances within the circuit, which can be easily changed mechanically (by using potentiometers) but are usually difficult to change programmatically. Another weakness of the design is that it relies on an explicit mathematical expression of the V-I relationship, and as a consequence cannot model the parasitic resistances of a solar cell.

An interesting class of solar array simulators use a physical instance of the simulated PV module as the reference. These designs not only offer high speed, but they are also the most accurate ones, since with this approach there is no need to model the solar module behavior or even to obtain data about it. The work presented in [15] shows how the output of a small illuminated solar cell can be amplified to simulate a much larger PV device. In [14] a similar approach is shown, but a miniature solar array is used as the reference so that the peculiar behavior of partially shaded solar arrays can be simulated. In both designs, the reference cells have to be illuminated according to the desired solar insolation. The method introduced in [16] avoids this complication by substituting the light-generated photo current with an external adjustable current source, which, however, was shown to reduce the simulation accuracy near the open-circuit point.

The main drawback of the approach used in these designs is limited flexibility, since in order to simulate another type of PV device, the reference module has to be replaced. Moreover, the temperature dependence of the PV devices is not included in the simulation unless the reference cells are physically brought to the desired temperature.

Research articles in this field make it clear that it is not easy to merge the advantages of different methods, e.g. LUT-based implementations are very flexible but almost never as fast as more rigid analog designs. The key is to identify the demands against the simulator based on the task it should perform and select implementation methods accordingly.

3 Thesis Goals and Methodology

3.1 Introduction

In this chapter, the goals that should be reached at the end of the thesis work will be defined. After that, the steps to be used to achieve them will be outlined.

3.2 Thesis Goals

The goal of this thesis is to extend the existing HiL setup so that it can be utilized for an accurate verification of the MOVE-II power budget. It is important that the HiL philosophy should be obeyed during this process, i.e. the EPS and the power-consuming subsystems should appear in the HiL simulation as hardware. This necessitates the development of a solar array simulator, to serve as an interface between the computer-simulated space environment data and the physical solar array connectors of the satellite EPS. While the SAS should be tailored specifically to MOVE-II, it should be flexible enough to be used in similar CubeSat projects. The design and realization of this device is the most important task within this project. Following the naming scheme begun with the current MOVE-II SAS, the SaS V1, the new simulator will be called SaS V2 and will be referred to as such (or simply as SAS) from now on.

Once the SaS V2 is part of the HiL environment, a number of mission scenarios should be simulated and distinct statements on the power budget of MOVE-II should be formulated.

3.3 Methodology

In the process of reaching the defined goals, the following steps will be carried out:

1. *Specification*: First, existing conditions, such as the properties of the EPS, the MOVE-II solar cells and the HiL environment will be analyzed to build up a detailed specification of the SaS V2.
2. *Design and construction*: After a brief review of possible solutions, the design steps of the device will be laid out and the device will be constructed.
3. *Interfacing to HiL*: Once the design is ready and tested, it will be integrated into the HiL environment and the whole system will be tested again.
4. *Power budget verification*: Finally, different orbital scenarios will be simulated with the extended HiL setup, and it will be declared whether the MOVE-II power budget is positive (spacecraft has enough power) or negative (spacecraft runs out of power) in each of the assessed cases.

4 The MOVE-II Electrical Power System

4.1 Introduction

In order to specify requirements against the solar array simulator, we must first review the properties of the system it is going to be connected to (the MOVE-II EPS) and the PV devices it will simulate. This review will be provided next.

4.2 The EPS Subsystem

A crucial element of any satellite is its electrical power system (EPS), whose role is to collect energy from the solar cells of the spacecraft and provide its components with the power necessary to operate. MOVE-II uses the model 25-02452 3rd Generation EPS from Clyde Space, which is a commercially available product intended for CubeSats and comes as a single $10 \times 10\ \mathrm{cm}$ printed circuit board (PCB). The EPS is connected to the following other units of the satellite: the solar arrays to receive power from them; a battery to store the harvested energy; and to the rest of the subsystems through various regulated voltage buses to deliver power to them.

The EPS connects to the solar arrays via 3 battery charge regulators (BCRs) which are responsible for routing the solar power into the satellite's battery or to other subsystems. Therefore, the system can host up to 3 solar arrays. The BCRs are effectively switch mode DC to DC converters. Two of them employ the buck topology (BCR buck 1 and BCR buck 2) and hence they can provide a lower voltage on the output than that on the input. The third BCR is a SEPIC converter (BCR SEPIC), whose output voltage can either be lower or higher than its input.

Each BCR has 2 solar array connectors, which are internally connected in parallel so that constructing arrays of parallel PV panels can be done without additional external wiring. Therefore, the 2 sockets wired to the same BCR are electrically and logically one connector and will be handled as such from now on. These solar array connectors have a positive and a negative pole to match the positive and negative output terminals of the connected PV device. The negative pole of each socket is connected to common ground. Additional pins found on the same connectors allow adding sensors to the system which can provide information about the temperature and the illumination of the respective solar array. As these pins are not relevant from the power delivery's point of view, they will be omitted from the discussion and diagrams that follow. Fig. 4.1 demonstrates the connection of the EPS to its peripherals.

The Clyde Space EPS datasheet defines the maximum values of voltages and currents that can be applied to the BCR inputs via the solar array connectors. These are summarized in Table 4.1. The maximum input current (I_{max}) of a BCR depends on its input voltage (V_{in}): it is constant at the value given in column 3 of Table 4.1 until about 80% of the maximum input voltage (V_{max}) and then starts to roll off to the value given in column 4. The DC voltages and currents applied to the BCRs by the solar arrays (or an SAS) must not exceed the values shown in the table.

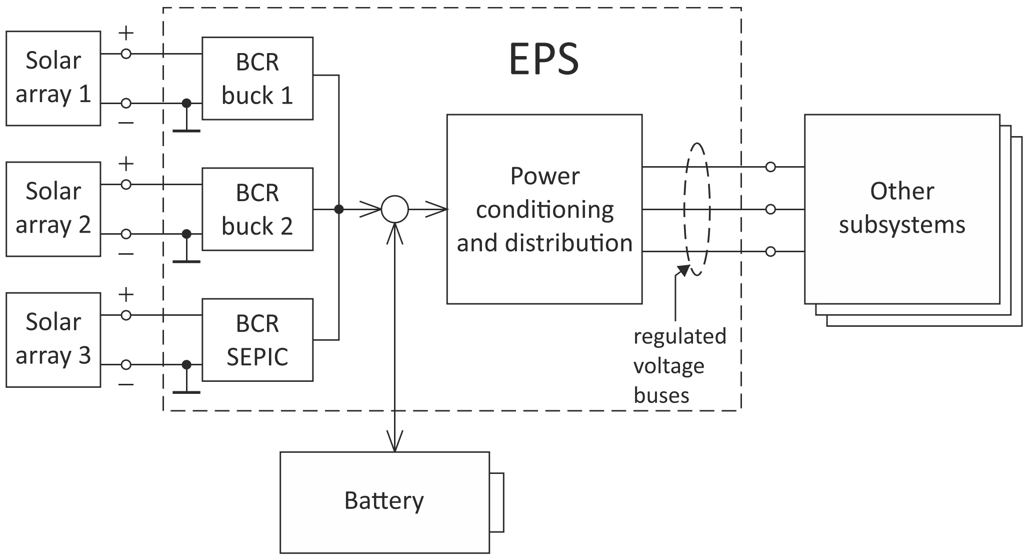


Figure 4.1 – EPS connections to other satellite elements

BCR	$V_{\max}$	$I_{\max}$ at $0 \leq V_{\text{in}} \leq 0.8 \cdot V_{\max}$	$I_{\max}$ at $V_{\text{in}} = V_{\max}$
Buck 1, Buck 2	$30\ \text{V}$	$0.5\ \text{A}$	$0.4\ \text{A}$
SEPIC	$9.5\ \text{V}$	$0.5\ \text{A}$	$0.4\ \text{A}$

Table 4.1 – Maximum ratings of the EPS battery charge regulators

Most circuits powered from PV devices employ some form of maximum power point tracking (MPPT) mechanism, which ensures that the operating point of the PV device coincides with its maximum power point so as to transfer all of its available power to the load. The BCRs on the Clyde Space EPS utilize a simple method: approximately every 2.5 seconds they disconnect all load from the solar arrays for around $100\ \text{ms}$ to measure their open-circuit voltage. Then, they regulate their input voltage to 83% of the measured value, which is assumed to give an operating point very close to the MPP. Fig. 4.2 shows how the input voltage of a BCR changes during MPPT in an oscilloscope screen capture.

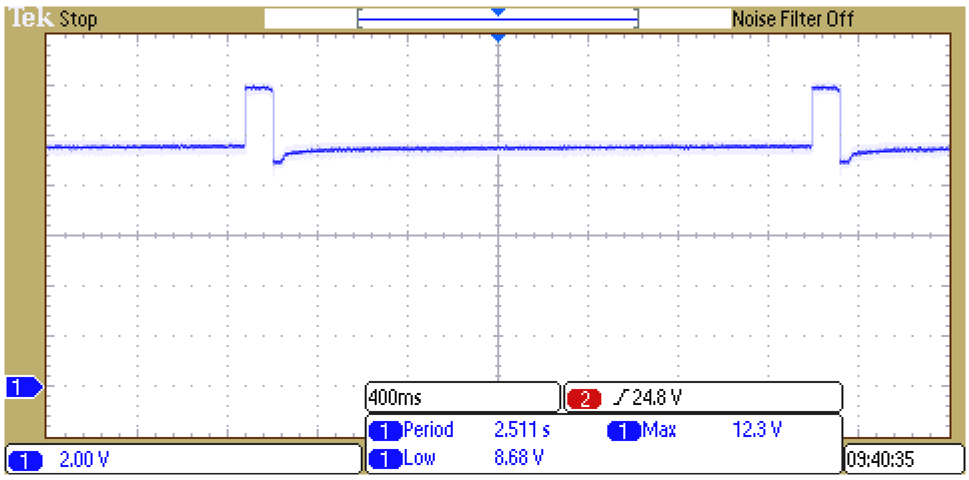


Figure 4.2 – BCR input voltage during MPPT
Source: [19]

The EPS has two operational modes, which influence how much power is drawn from the solar arrays. If the battery voltage is below a given threshold (the end-of-charge or EoC voltage), then MPPT mode is used. In this mode the EPS tracks the maximum power point of the PV arrays to charge as much energy into the battery as possible. If the battery voltage exceeds the EoC voltage, the EPS switches to end-of-charge mode, in which the solar array operating points are moved away from the MPP to draw only the power required to top up the battery until it is fully charged [6][19].

4.3 The MOVE-II Solar Arrays

MOVE-II uses a total of 16 multi-junction solar cells manufactured by Azur Space. A multi-junction cell is a type of PV cell consisting of multiple p-n junctions on top of each other, made from different semiconductor materials, each of which absorbs different wavelengths of light. Consequently, multi-junction cells make use of a broader spectrum of sunlight and have greater efficiency than single-junction ones.

Eight of the solar cells are made from the type 3G30 GaAs-based 3-junction wafer of Azur Space and will be referred to as 3J cells. These are mounted on four sides (also called Sidepanels) of the cubical satellite body. The other eight cells are 4-junction types and reside on the Flappanels of the satellite, which are 4 deployable surfaces that are in the same plane as the top of the spacecraft when they are unfolded. These will be referred to as 4J cells. The 4J cells are twice as large as the 3J ones each and they are the key power source of the satellite.

The cells, following the architecture of the EPS, are arranged into 3 arrays. Two arrays (SA1 and SA2) consist of four serially connected 4J cells each and are connected to BCRs buck 1 and buck 2. The third array (SA3) comprises the eight 3J cells in a mixed series-parallel topology and is connected to the SEPIC BCR. The exact connections of the cells and their location on the satellite facets are shown in Fig. 4.3. The two Sidepanels not visible in the image (SP x- and SP y+) also host two 3J cells each. The cells seen on the top face (Toppanel) of the satellite are part of the Payload and do not contribute to power generation for the spacecraft components.

The connection scheme of SA3 is realized so that the cells on the same Sidepanel are connected in series with each other, and all four Sidepanels are connected in parallel. The diodes shown in this array are so-called blocking diodes which prevent reverse currents when the array is partially shaded. Their role will be explained further in Section 6.2.2.

In Fig. 4.3, it is visible that the cells within SA1 and SA2 are all in the same plane and thus are always illuminated equally. However, the cells of SA3 are on four different planes and even within the same plane, one of the cells may be in sunlight while the other one is shaded by a Flappanel. This means that the output of this array is strongly influenced by partial shading effects.

To properly dimension the SAS, we must know what maximum voltages and currents the arrays can produce. Information about the 3J cells is provided in the manufacturer's datasheet [20]. The document specifies certain values for different ages of the cell, since some attributes degrade with time. For these values, the highest of the given quantities must be considered when specifying the requirements against the SAS (such as maximum output voltage or current, etc.).

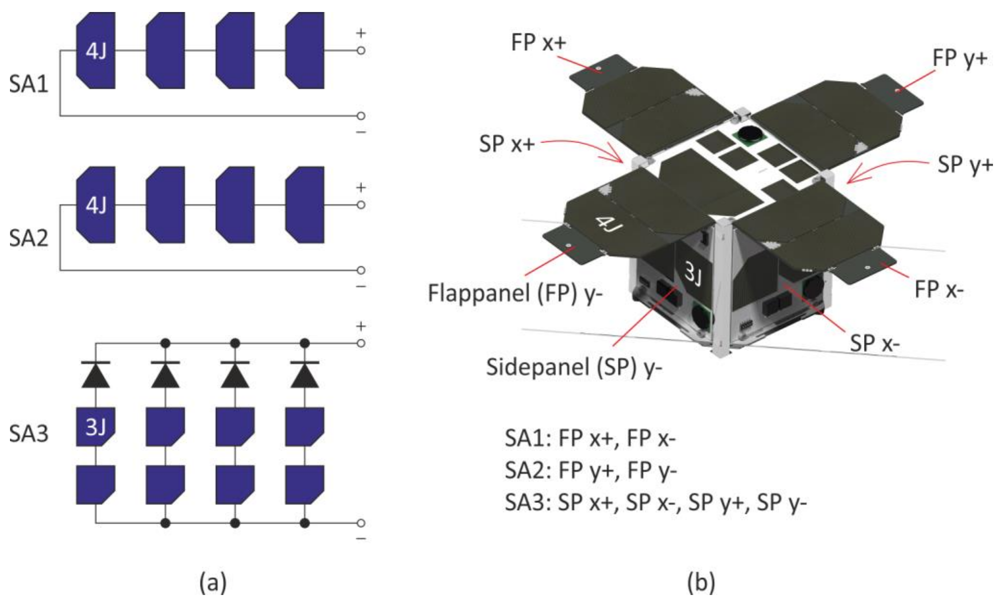


Figure 4.3 – Solar array topologies (a) and location of solar cells (b) on MOVE-II

Since the 4J cells are not commercially available yet, they have no published datasheet. The necessary information about them was obtained from measurements previously conducted by Lucas Krempel and Martin Rutzinger [21].

Quantities about the cells relevant for the work presented here are the following:

- $I_{\mathrm{sc,max}}$: maximum short-circuit current
- $V_{\mathrm{oc,max}}$: maximum open-circuit voltage
- $V_{\mathrm{MPP,max}}$: maximum voltage at MPP
- $I_{\mathrm{MPP,max}}$: maximum current at MPP
- P_{max}: maximum output power
- η: efficiency
- A: surface area
- K_{I}: temperature coefficient of short-circuit current
- K_{V}: temperature coefficient of open-circuit voltage

The highest available illumination in space on an orbit around Earth is $G_{\mathrm{max}} = 1367\,\mathrm{W/m^2}$. Items in the above list with the label “maximum” in their description are measured at this irradiation, at a given temperature. E.g. the short-circuit current of a cell will equal $I_{\mathrm{sc,max}}$ at a given temperature T_0 if $G = G_{\mathrm{max}}$. Therefore, the following equation holds:

$$P_{\mathrm{max}} = V_{\mathrm{MPP,max}} \cdot I_{\mathrm{MPP,max}} = G_{\mathrm{max}} \cdot A \cdot \eta. \tag{4.1}$$

Lower irradiance than G_{max} will be experienced by a cell if it is hit by the sunrays at an incident angle different from $0°$, since only the perpendicular component of the Poynting vector of the illuminating light generates photo current. Moreover, if a cell is in shadow, then its irradiance drops almost to zero in space.

The temperature coefficients K_I and K_V assume a linear relationship between their described quantity and the temperature, which is an accurate and widely used approximation [35]. The values for the MOVE-II cells are summarized in Table 4.2.

Table 4.2 – Properties of the MOVE-II solar cells

Parameter	3J	4J
$I_\mathrm{sc,max}$	$260.1\ \mathrm{mA}$	$449.4\ \mathrm{mA}$
$V_\mathrm{oc,max}$	$2700\ \mathrm{mV}$	$3363.8\ \mathrm{mV}$
$I_\mathrm{MPP,max}$	$252.2\ \mathrm{mA}$	$429\ \mathrm{mA}$
$V_\mathrm{MPP,max}$	$2411\ \mathrm{mV}$	$2900.5\ \mathrm{mV}$
P_max	$0.608\ \mathrm{W}$	$1.244\ \mathrm{W}$
η	$29.48\ \%$	$30.16\ \%$
A	$15.09\ \mathrm{cm}^2$	$30.18\ \mathrm{cm}^2$
K_I	$0.18\ \mathrm{mA/°C}$	$0.49\ \mathrm{mA/°C}$
K_V	$-6.2\ \mathrm{mV/°C}$	$-7.89\ \mathrm{mV/°C}$
Conditions	$G = 1367\ \mathrm{W/m}^2$ $T_0 = 28\ \mathrm{°C}$ Spectrum: AM0	$G = 1367\ \mathrm{W/m}^2$ $T_0 = 25\ \mathrm{°C}$ Spectrum: AM0

The last row in the table specifies under what conditions the given values were measured. T_0 marks the cell temperature. AM0 refers to the spectrum of sunlight not filtered by the atmosphere, i.e. as it is in space.

In order to make use of the given temperature coefficients, the lowest and highest temperatures (T_min, T_max) that the cells can reach during the orbit are also needed. These are known from thermal simulations previously carried out by the MOVE-II team [22]. Since these results have some level of uncertainty and in order to guarantee a broader usability of the SAS, a substantial margin will be added to the simulated temperature values. These are shown in Table 4.3.

Table 4.3 – Minimum and maximum temperatures of the MOVE-II solar cells

Cell	T_min (simulation)	T_min (with margin)	T_max (simulation)	T_max (with margin)
3J	$-5.2\ \mathrm{°C}$	$-10\ \mathrm{°C}$	$20.5\ \mathrm{°C}$	$30\ \mathrm{°C}$
4J	$-5.8\ \mathrm{°C}$	$-10\ \mathrm{°C}$	$47.8\ \mathrm{°C}$	$60\ \mathrm{°C}$

The maximum open-circuit voltage of SA1 and SA2 is given by 4 times that of an individual 4J cell due to their series connection. Also because of the series topology, the current of the array will equal to that of a single cell. When accounting for temperature effects, T_max should be considered for I_sc, since it increases with rising temperature ($K_\mathrm{I} > 0$), whereas T_min should be considered for V_oc, since it increases with falling temperature ($K_\mathrm{V} < 0$). Thus, the highest possible open-circuit voltage and short-circuit current of SA1 and SA2 ($V_\mathrm{oc,max}^\mathrm{SA1,2}$, $I_\mathrm{sc,max}^\mathrm{SA1,2}$) will be

$$V_\mathrm{oc,max}^\mathrm{SA1,2} = 4 \cdot \left[V_\mathrm{oc,max}^\mathrm{4j} + K_\mathrm{V}^\mathrm{4j} \cdot \left(T_\mathrm{min}^\mathrm{4j} - T_0^\mathrm{4j}\right)\right] = 14.56\ \mathrm{V}, \tag{4.2}$$

$$I_\mathrm{sc,max}^\mathrm{SA1,2} = I_\mathrm{sc,max}^\mathrm{4j} + K_\mathrm{I}^\mathrm{4j} \cdot \left(T_\mathrm{max}^\mathrm{4j} - T_0^\mathrm{4j}\right) = 466.5\ \mathrm{mA}. \tag{4.3}$$

In the equations above, the upper index “4j” refers to values associated with 4J cells.

Let us now do the same calculations for the third solar array. Since in SA3 each parallel string (each Sidepanel) consists of 2 series cells, the maximum open-circuit voltage is 2 times that of a 3J cell. When computing the maximum short-circuit current, one should consider that although there are 4 strings of cells connected in parallel, these cannot be all fully illuminated at the same time due to the cubical shape of the satellite. The two Sidepanels not facing the Sun are completely shadowed and the total illumination of the other two Sidepanels will be maximal if the incident angle of the sunrays is $45°$ on both of them (see Fig. 4.4). Then the summed magnitude (G_s) of the perpendicular components of the illuminating Poynting vector will be

$$G_\mathrm{s} = 2 \cdot G \cos 45° = \sqrt{2} G. \tag{4.4}$$

Since the photo-generated current is directly proportional to illumination, the maximum current of a single Sidepanel, which itself equals to that of a single cell, must be multiplied only by $\sqrt{2}$ to yield the maximum array current (instead of 4, which would be the case for 4 fully illuminated Sidepanels.). This gives

$$V_{\mathrm{oc,max}}^{\mathrm{SA3}} = 2 \cdot \left[V_{\mathrm{oc,max}}^{\mathrm{3j}} + K_\mathrm{V}^{\mathrm{3j}} \cdot \left(T_{\mathrm{min}}^{\mathrm{3j}} - T_0^{\mathrm{3j}}\right)\right] = 5.87 \text{ V}, \tag{4.5}$$

$$I_{\mathrm{sc,max}}^{\mathrm{SA1,2}} = \sqrt{2} \cdot \left[I_{\mathrm{sc,max}}^{\mathrm{3j}} + K_\mathrm{I}^{\mathrm{3j}} \cdot \left(T_{\mathrm{max}}^{\mathrm{3j}} - T_0^{\mathrm{3j}}\right)\right] = 368.3 \text{ mA}. \tag{4.6}$$

Again, the index “3j” refers to 3J cells. For both cell types, the minimum and maximum temperature values are the ones with the added margin from Table 4.3.

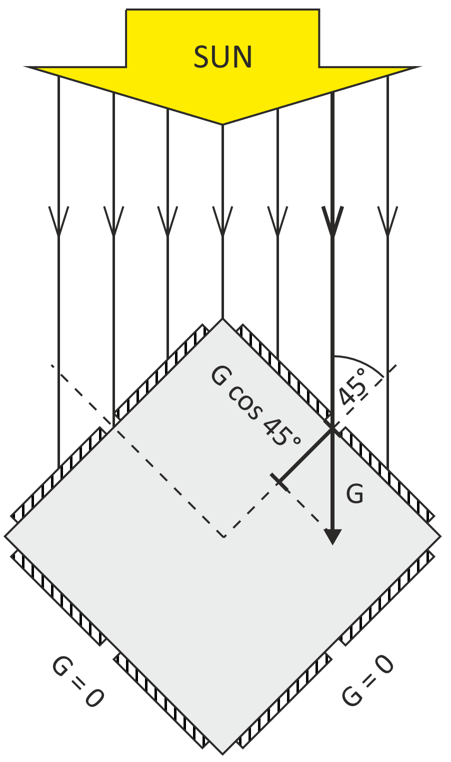


Figure 4.4 – Optimal Sidepanel illumination

The data provided above suffices to specify the maximum ratings of the solar array simulator. Further data provided about the cells in Table 4.2 will be used in Section 6.2 to generate their VI curves for different solar intensities and temperatures.

5 Solar Array Simulator Design

5.1 Introduction

This chapter presents the design process of the SaS V2. After reviewing the structure of the MOVE-II HiL environment in detail, all information necessary to provide a quantitative specification of the simulator will have been collected. Following the specification and an overview of possible solutions, design decisions and steps will be explained. Finally, the function of the device and its conformity with the requirements will be demonstrated.

5.2 Choice of Solution Approach

First, let us review why we have chosen to design a custom SAS for MOVE-II instead of using one of the many existing solutions mentioned in Chapter 2.

The current MOVE-II solar array simulator, the SaS V1, could be integrated into the HiL system with software changes only. However, this device does not have circuitry to simulate an arbitrary VI curve and hence it is not useful for accurate power budget verification.

The commercial solar array simulators presented in Section 2.4 are primarily intended for testing high-power devices, such as voltage inverters connected to solar power plants and so they have much higher voltage and current ratings than what is necessary for our purposes. Consequently, they are expensive, physically large, and their enormous dynamic range necessitates trade-offs to their resolution and accuracy. Moreover, since they usually have only one output, three units would have to be purchased to substitute the three solar arrays of MOVE-II.

The scientific papers reviewed in Section 2.5 do not present practical, general-purpose solutions. Instead, they point out how a specific attribute of an SAS can be improved compared with devices that exist at the time of publishing. Moreover, the information they contain is usually not detailed enough to reproduce the simulators without effort comparable to what would be needed for a design from scratch.

The considerations above motivate the development of a custom SAS rather than opting for an existing solution. In the following sections, the design process will be explained in detail. All files relevant to the design, including schematic diagrams, PCB layouts and source codes are accessible in the project's Git repository. A link to the repository and a list of the files can be found in Appendix E.

5.3 Test Environment Concept

5.3.1 Present HiL Setup

The SAS will function as a part of the HiL simulation environment. The setup in its current state can only be used to test the ADCS subsystem. The ADCS consists of a central unit, called the Mainpanel and five peripheral units which are the Toppanel and the four Sidepanels of the satellite. While the peripheral panels contain the sensors and the actuators required for attitude determination and control functionality, the task of the Mainpanel is to run a central control algorithm and command the peripheral panels.

The main building blocks of the current HiL environment and their roles are summarized below.

- *HiL PC*: The space environment is simulated in Matlab and Simulink running on a personal computer, referred to as the HiL PC. It keeps track of the state of the satellite, such as its position in the orbit, its orientation and its angular velocity around all three axes, in case the satellite is rotating. Based on models of the ADCS sensors, it determines what they would measure and output in the current state and communicates this information to the ADCS Mainpanel (via the Panel Emulator).
- *Mainpanel*: The Mainpanel is utilized as the controller of the ADCS in the MOVE-II HiL setup. It receives simulated sensor outputs from the HiL PC and generates commands for the actuators, trying to force the satellite to the desired orientation. In fact, the Mainpanel functions in exactly the same way as if it was in space.
- *Panel Emulator*: The communication bridge between the HiL PC and the Mainpanel is the Panel Emulator, which mimics the ADCS peripheral panels. On the one hand, it receives simulated sensor data from the HiL PC and generates the corresponding digital output signals of the real sensors. On the other hand, it receives actuation commands from the Mainpanel and forwards them to the HiL PC. Since the Mainpanel communicates to the sensors and actuators via a serial peripheral interface (SPI) bus, and the Panel Emulator is connected to the HiL PC via Ethernet, the Panel Emulator can be considered as a converter interface between SPI and Ethernet.

The HiL PC uses the magnetorquer actuation commands generated by the Mainpanel and received through the Panel Emulator to continuously update the state of the satellite. The simulation runs in real time (one second in the satellite's life is simulated in one second) and the HiL PC sends out data (such as sensor outputs) every $200\ \mathrm{ms}$, i.e. at $5\ \mathrm{Hz}$. The ADCS HiL setup is visualized by Fig. 5.1.

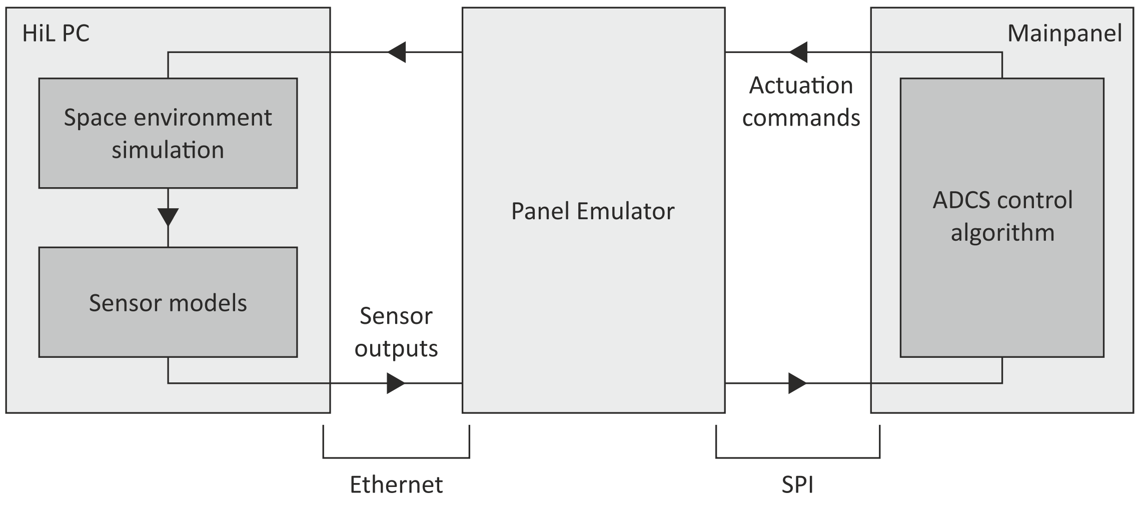


Figure 5.1 – The ADCS HiL concept

5.3.2 Extended HiL Setup

The HiL environment must be extended in order to be useful for power budget verification. Once a solar array simulator which is able to communicate to the HiL PC is at hand, this can be done easily.

When integrating the SAS into the setup, the following points should be kept in mind. Firstly, the ADCS part of the HiL setup should be kept in place so that the satellite orientation is simulated as accurately as possible, since this information is decisive when computing the illuminations of the solar cells. Secondly, all of the MOVE-II subsystems should be part of the loop so that the power consumption of the satellite is realistic. This includes the insertion of the ADCS peripheral panels which consume notable power when applying currents to the magnetorquer coils. This also means that the actuation commands of the Mainpanel now will have to be routed to the actual peripheral panels as well as to the Panel Emulator. This can be done with some extra cabling and does not require additional electronics or software changes. A block diagram of the extended HiL setup is shown in Fig. 5.2.

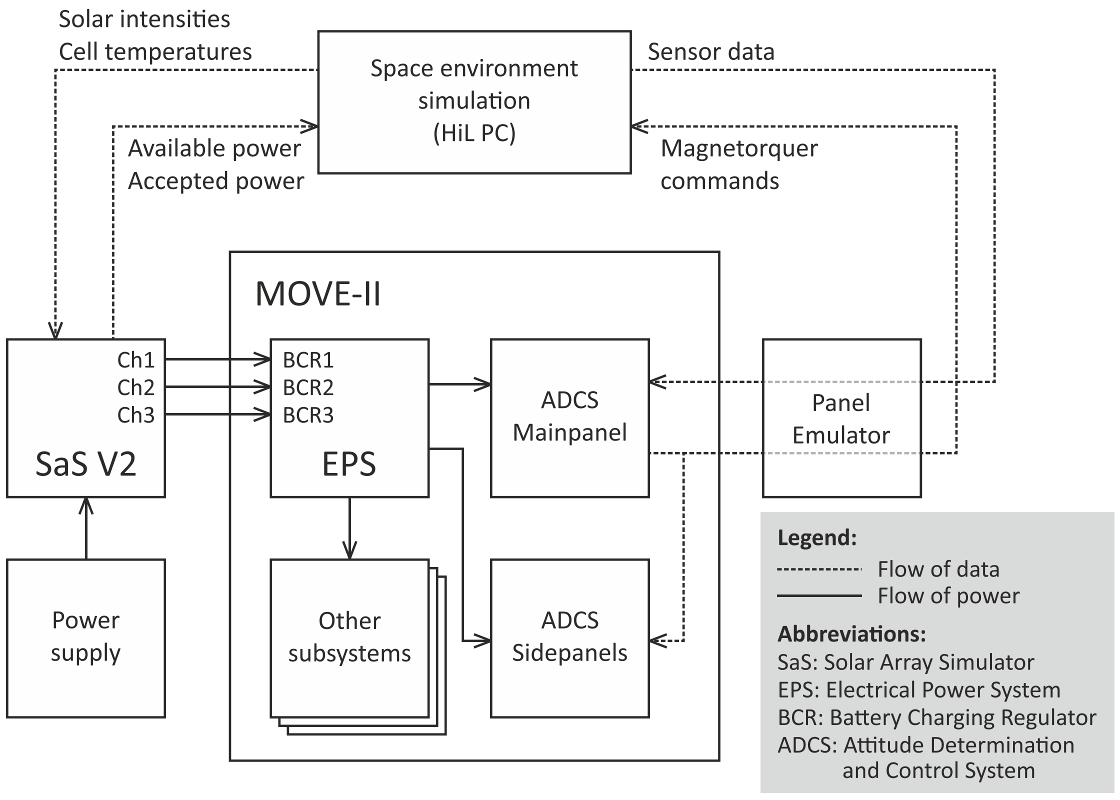


Figure 5.2 – Extended HiL setup block diagram

As seen from the diagram above, the SAS will take the solar intensities and the temperatures of the cells as input from the HiL PC. In the present ADCS-only HiL setup, the Matlab simulation already keeps track of the illumination of the satellite facets so that it can simulate the sun sensors. The ADCS team has carried out some modifications to convert this data to the illumination of the solar cells.

The solar intensities of the cells are determined based on the incident angle of the illuminating light. Additionally, the cells on the Sidepanels may be shaded by the Flappanels. This effect is modeled with a simple method based on a ray-tracing algorithm which determines whether the geometric midpoint of a given cell falls into the shadow of a Flappanel. If the point is shaded, the illumination of the cell is considered as zero. Otherwise, the illumination is calculated as if the cell was not shaded at all. Consequently, the illumination level sent by the HiL PC may be zero even if the cell is party in the Sun. This is to some extent compensated for by the fact that if the cell is partly shaded, but its midpoint is in the Sun, then its whole surface is considered to be illuminated. The cell temperatures are simulated based on a thermal model kindly provided by the ADCS team especially for this purpose. From now on, the SAS input data coming from the HiL PC will be considered as given and to be outside of the scope of the thesis work.

An arrow pointing from the SAS to the HiL PC in Fig. 5.2 shows that the simulator will be telling the computer how much power is available on its outputs (based on the MPPs of the simulated VI curves) and how much of that power is being accepted by the EPS (based on the actual operating points on the curves). The figure also shows that the SAS will be attached to the EPS directly through its 3 battery charge regulators, just like the real solar arrays would be.

With the extended HiL setup, a simulation will take place as described below.

1. The battery is charged to a desired starting level and initial conditions of the simulation are set. These are the following:
    a. The date and time of the beginning of the simulation, which defines the starting position of the satellite in the orbit and the position of the Sun.
    b. The initial orientation and initial angular velocities of the satellite around all three axes.
    c. The parameters of the satellite's orbit affecting the frequency and length of the sun phases and eclipses.
2. The simulation is launched. The HiL PC starts sending information to the SAS and the Mainpanel. Based on the answers of the Mainpanel, it updates the satellite state.
3. Meanwhile, the SAS outputs the VI curves of the solar arrays according to the received illumination and temperature values. The EPS uses the offered power to charge the battery. The SAS sends back data about the available and accepted power to the HiL PC. This is not necessary for the simulation, but can be used to build statistics about the efficiency of the EPS MPPT mechanism.
4. After a pre-defined time, the simulation is stopped.
5. The central computer (CDH) of the satellite, as well as the ground station keeps track of and logs the battery level throughout the mission. After the simulation, this data is acquired and analyzed. If an overall trend that the battery level was increasing can be observed, the test case is considered power positive. Otherwise, it is logged as power negative.
6. As a final step, the temporal evolution of the battery level can be correlated to other data generated during the simulation, such as the power values sent by the SAS or other data logged by the satellite (power consumption of subsystems, etc.).

5.4 Specifications

Now we are ready to define the specification of the SaS V2 precisely. The specification is presented by Table 5.1, which also describes the conditions which have led to each listed point. Column 4 provides a reference to the sections where these conditions were discussed.

Table 5.1 – SaS V2 specifications

Attribute	Specification	Governing Condition	Reference (Section)
Types of simulated VI curves	Arbitrary within output range	Different types of cells at different solar intensities and temperatures must be simulated. Simulating partial shading should also be possible because of SA3.	4.3
Number of output channels	3	MOVE-II has 3 solar arrays.	4.3
Maximum open-circuit voltage ($V_{\mathrm{oc,max}}$)	$16\ \mathrm{V}$	The open-circuit voltage of SA1 and SA2 may go as high as $14.56\ \mathrm{V}$. A margin of around $1.5\ \mathrm{V}$ is added.	4.3
Maximum short-circuit current ($I_{\mathrm{sc,max}}$)	$550\ \mathrm{mA}$	The short-circuit current of SA1 and SA2 may go as high as $466.5\ \mathrm{mA}$. A margin of around $80\ \mathrm{mA}$ is added.	4.3
Settling time (τ_{s})	$< 100\ \mathrm{ms}$	The EPS maximum power point tracker measures the open-circuit voltage of the arrays for $100\ \mathrm{ms}$. The SAS output should stabilize within this time.	4.2
VI curve update frequency ($f_{\mathrm{u,VI}}$)	$> 5\ \mathrm{Hz}$	The HiL PC sends new input at a rate of $5\ \mathrm{Hz}$. For every new input, the VI curves have to be updated.	5.3.1
Interfaces	Ethernet, USB	The HiL PC communicates with the ADCS Panel Emulator via Ethernet. To keep the system consistent, the same type of interface will be used with the SAS. USB is added for increased flexibility.	5.3.1

Further notes:

- Since the negative poles of the BCRs in MOVE-II are connected through common ground, the channels of the SAS need not be isolated from each other.
- To keep the design simple, the SAS will be controllable only via its communication interfaces and will not have physical controls on its front panel.

5.5 Overview of Possible Solutions

5.5.1 Simple Methods

One of the simplest ways to simulate an approximated VI curve is by means of a bench power supply with constant current capability in series with a variable resistor (R_s) as shown in Fig. 5.3. The selected output voltage (V_{out}) and current limit (I_{lim}) of the PSU directly determine V_{oc} and I_{sc} of the resulting voltage-current characteristics, which is a two-line approximation of a VI curve. The position of the knee-point of the line can be shifted by adjusting R_s.

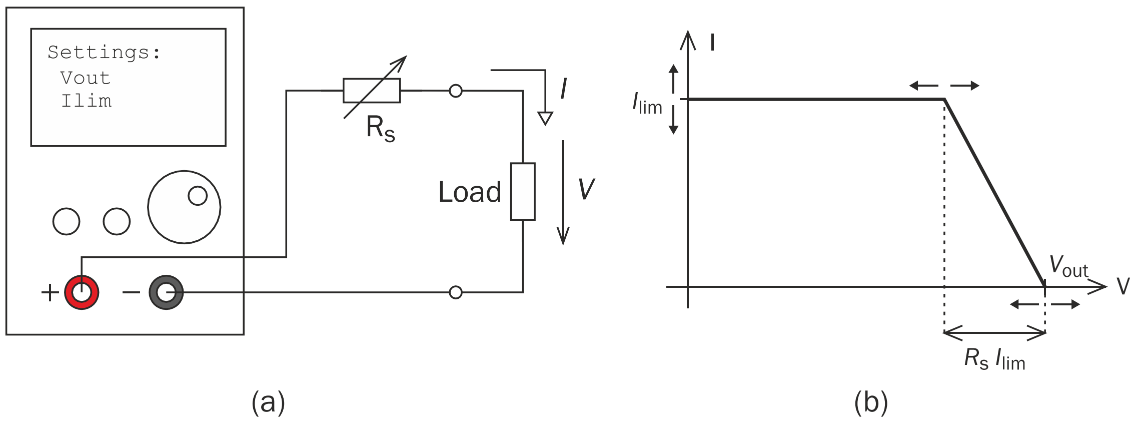


Figure 5.3 – Simple solar array simulator with lab power supply (a) Structure, (b) output curve.

The method is often used to test MPPT algorithms if a more sophisticated SAS is not available, however, it is not flexible and accurate enough for our purposes.

A solar cell can be modeled by an equivalent circuit which consists only of a current source, a diode and two resistors (see Fig. 6.1 in Section 6.2.1). Another easy way to construct an SAS is to simply build up the equivalent circuit. However, this method is rarely used nowadays due to its inflexibility and high internal power dissipation when the SAS is unloaded [23][17].

5.5.2 Amplified Reference Method

The most common way to simulate a VI curve is the amplified reference method. Its first variant is the current source (CS) based reference amplifier whose operating principle is shown in Fig. 5.4 (a).

The key component of the solution is the VI curve reference circuit, a nonlinear two-port, which always outputs a voltage proportional to the current that corresponds to its input voltage on the VI curve being simulated. Putting it another way, the VI curve reference circuit contains information about the simulated PV device and is used to query the current that corresponds to any voltage from 0 to V_{oc} on the VI curve. The input of this two-port is simply connected to the load so that it measures its voltage at all times. Its output controls an amplifier stage (in this case a current source) which forces the queried current onto the load. This closed-loop configuration ensures that in the steady state, the operating point given by the voltage and the current of the load can only lie on the VI curve defined by the reference.

The second variant of the method, the voltage source (VS) based reference amplifier is the counterpart of the circuit discussed above and is depicted in Fig. 5.4 (b). This time, the VI curve reference must implement the inverse relation, that is, it must deliver the voltage that corresponds to a given current on the VI curve. Instead of measuring the load voltage, this time its current is measured with some form of current sensing which feeds a proportional voltage to the input of the reference circuit. The output stage is a voltage source controlled by the output of the reference two-port.

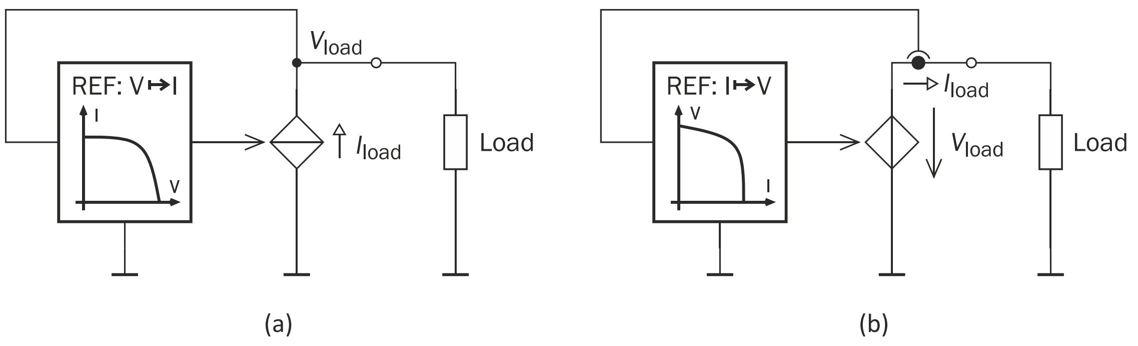


Figure 5.4 – Operating principle of amplified reference methods (a) CS based and (b) VS based version.

It should be noted that with these methods there is no need to measure the impedance of the load (i.e. both its current and voltage at the same time) in order to find where the load line intersects the VI curve. The closed-loop nature of the amplified reference method guarantees that the steady-state operating point will be on the intersection of the VI curve and the load line, as long as the control loop is stable. Nevertheless, some implementations do measure the load impedance, find the intersection point mathematically, and set it as the output in a single step, using either a voltage or a current source as the output stage. In this case, the reference should implement the relation between load impedance and output voltage, or current, depending on the type of output stage used [11][24].

The reference two-port can be implemented either digitally (with a lookup-table) or as an analog nonlinear circuit. As mentioned in Section 2.5, an analog solution results in faster stabilization, but is less flexible than a digital LUT-based solution. The two-port must have high input impedance so that the entire current generated by the output stage flows through the load.

The output stage is essentially a voltage or current regulator and it can be implemented either as switch-mode or linear. Switch mode regulators are more efficient (dissipate less heat) than linear ones but are more complex in structure, have higher noise and generally react slower to changes in the load.

While both of the reference amplifier methods discussed above work in theory, neither of them can cover the entire VI curve alone. In particular, a VS based reference amplifier has trouble near the short-circuit point, while a CS based one has problems near open circuit. For example, if a CS based reference amplifier is open circuited, the load current will be 0, but the current source has no way to set the output voltage to the desired V_{oc}. This is because it can only control the output current and 0 current through an open circuit does not define any voltage, i.e. the output voltage can be arbitrary. In practice, it will usually saturate to the supply voltage that the circuit is powered from, which may be higher than the desired V_{oc}.

Theoretically, if the load has an arbitrarily large finite impedance, the current source will still be able to set the output voltage by forcing a very small current through the load. In practice however, in the case of such very small signals, the noise superimposed on them will spoil the stability of the control loop, which means that a CS based reference amplifier will get unstable when it gets close to the open-circuit point. Likewise, a VS based reference amplifier will become unstable near the short-circuit point, because a voltage source does not have control over the current through its load when that is a short circuit.

Additionally, purely current or voltage source based reference amplifiers suffer from reduced accuracy in different regions of the VI curve. A typical VI curve has 3 regions, as shown in Fig. 5.5: the current source region, the maximum power region and the voltage source region. For example, the current source region can be reliably emulated with a CS based reference amplifier because the output voltage defines the output current very clearly in this region. However, in the voltage source region, where the slope of the curve gets very steep ($|\mathrm{d}I/\mathrm{d}V| \gg 1\ \mathrm{S}$)[5], a small error in voltage measurement may cause large errors in output current. Moreover, for a digital VI curve reference, the employed LUT must have a very high resolution to differentiate between the points in this region of the curve. A VS based reference amplifier will have similar problems in the current source region (where $|\mathrm{d}I/\mathrm{d}V| \ll 1\ \mathrm{S}$).

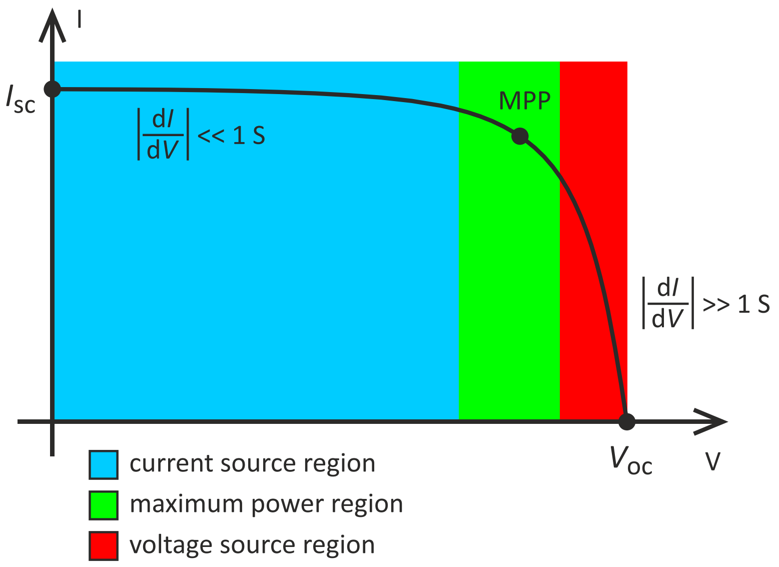


Figure 5.5 – VI curve regions

The SaS V2 must work in all three regions to fulfill the specification of simulating an arbitrary VI curve. This can be achieved by combining the two methods and building a hybrid reference amplifier, which acts like a voltage source in the voltage source region and like a current source in the current source region of the VI curve (in the maximum power region, either solution can work) [24]. Such a method will be presented in the next section.

5.6 Design Process

5.6.1 General Guidelines

Based on previous considerations, some basic design decisions about the SAS can be made. These are summarized in Table 5.2. In the following sections, the final design based on these decisions will be presented using a top-down

[5] The symbol S stands for siemens, the unit of electrical admittance. $1\ \mathrm{S} = 1\ 1/\Omega$.

approach. First, an overview of the system will be given and then each system component will be explained in detail.

Table 5.2 – SaS V2 design concept

Design Aspect	Decision	Reason
Solution basis	Hybrid reference amplifier	To reliably cover the entire VI curve from short to open circuit.
Type of VI curve reference circuit	Digital (LUT-based)	To be able to simulate arbitrary curves and change them on the fly.
Type of output stage	Linear	To reduce design complexity, output noise and settling time.

5.6.2 System Overview

The core of the SaS V2 design is an electrical circuit which consists of multiple logical entities. These are shown in Fig. 5.6 along with all electrical signals that connect them. The ground symbol under the blocks in the figure points out that the signals are referenced to a common ground.

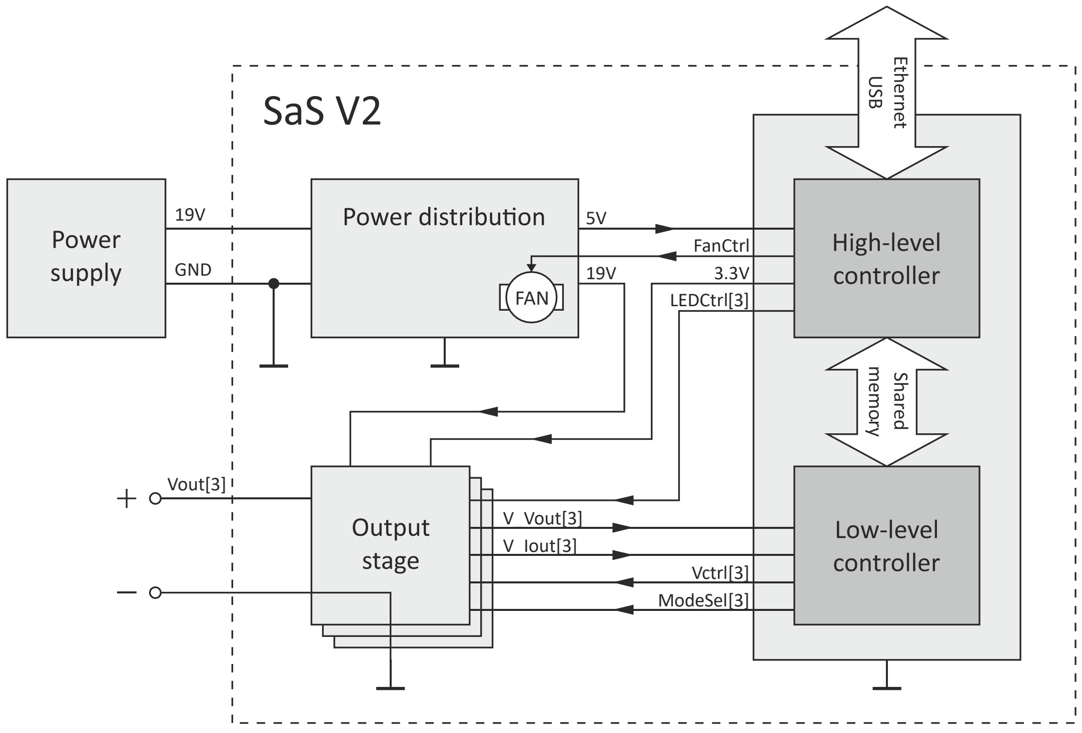


Figure 5.6 – SaS V2 block diagram

The task of each block is detailed below.

- *Output stages*: There are 3 output stages for the 3 channels which can function either as a voltage-controlled voltage source or a voltage-controlled current source to set the operating point on the output. Each output stage connects to the controller unit via its own set of signal wires (*V_Vout, V_Iout, Vctrl, ModeSel, LEDCtrl*). Since the output stages directly correspond to the SAS channels, they will also be referred to simply as "channels".

- *Controller unit*: This block consists of two submodules:
  - The *low-level controller* implements the VI curve references of the 3 channels and controls the output stages.
  - The *high-level controller* implements the I/O functions towards the HiL PC and the user. It exchanges information with the low-level controller via shared memory and controls some peripherals whose functions are not time-critical. It also generates the 3.3 V power bus.
- *Power distribution*: This module accepts power from an external power supply and distributes it among the rest of the units through regulated voltage buses. It also powers a fan for cooling the circuits.

Table 5.3 lists the interconnecting signals and their roles. The notation *“[3]”* behind a signal name means that 3 independent instances of that signal exist in the system (one for each channel). Choices of voltage levels will be explained in the following subsections.

Table 5.3 – List of interconnecting signals in the SaS V2

Signal Name	Signal Type	Value/Range	Description
V_Vout[3]	Analog	0 ... 2.5 V	A voltage proportional to the instantaneous output voltage of a channel.
V_Iout[3]	Analog	0 ... 2.5 V	A voltage proportional to the instantaneous output current of a channel.
Vctrl[3]	Analog	0 ... 2.5 V	A voltage proportional to the desired output voltage or current of a channel.
Vout[3]	Analog	0 ... 16 V	The output of a channel. Load is connected between this point and ground.
ModeSel[3]	Digital	0 or 3.3 V	Selects whether an output stage functions as a voltage source or as a current source.
LEDCtrl[3]	Digital	0 or 3.3 V	Each channel has an LED indicating whether it is on or off. This signal controls the LED of a channel.
FanCtrl	Digital	0 or 3.3 V	Turns the cooling fan on or off.
19V	Power	19 V	19 V power bus for the output stages.
5V	Power	5 V	5 V power bus for the controller unit.
3.3V	Power	3.3 V	3.3 V power bus for digital logic.
GND	Power	0 V	Common ground.

Next, each submodule will be explained. The schematic diagrams shown aim to depict the operating principle of the circuits only and some wires whose connections are obvious, as well as most bypass capacitors are omitted for clarity. Generally, the bypassing recommendations in the datasheets were followed and most integrated circuits are bypassed with a $100\,\mathrm{nF}$ ceramic capacitor. Full circuit diagrams can be found in Appendix A.

5.6.3 Output Stage

5.6.3.1 Operation

The output stage of the SaS V2 is a simple linear regulator, that, based on which signal is fed back to its error amplifier, can either regulate the output voltage or the output current. The circuit diagram is shown in Fig. 5.7 and component values are given in Table 5.4.

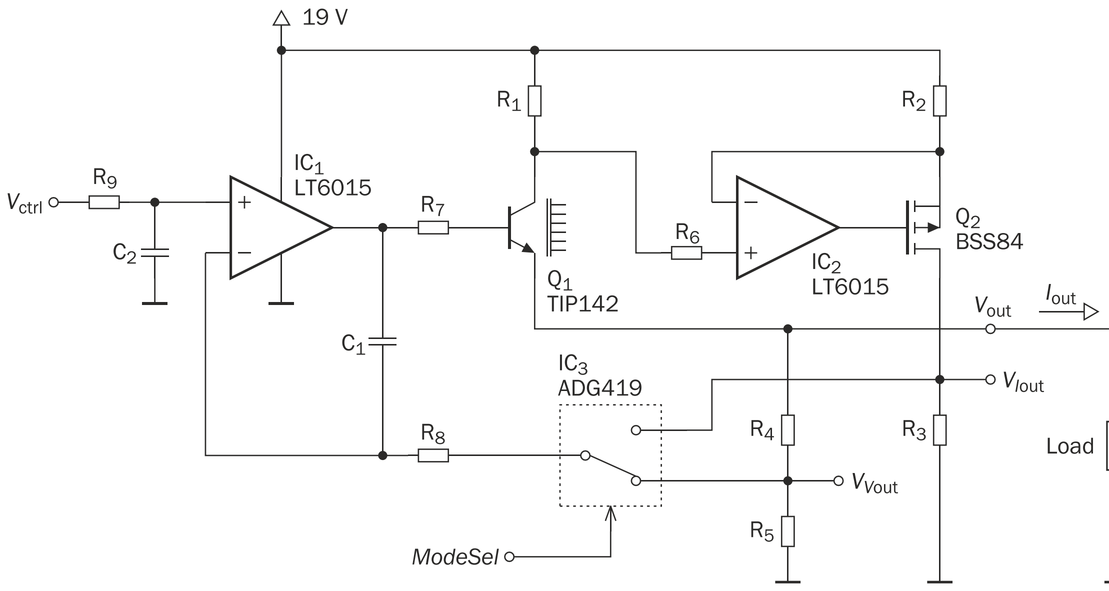


Figure 5.7 – Output stage circuit diagram

Component	R1	R2	R3	R4	R5	R6
Value	1 Ω (1%)	2.4 kΩ (1%)	10 kΩ (1%)	75 kΩ (1%)	13 kΩ (1%)	2.4 kΩ
Component	R7	R8	R9	C1	C2	
Value	100 Ω	3.3 kΩ	2.7 kΩ	10 nF	47 nF	

Table 5.4 – Output stage component values

The component IC3 is a digitally controlled analog switch that determines whether the output stage operates as a voltage source or as a current source. It is supplied from 3.3 V (not shown in the figure). If the switch is in the lower position, as shown in the circuit diagram, then IC1, Q1, R4 and R5 form a conventional linear voltage regulator, where Q1 is the pass transistor, IC1 is the error amplifier and V_{ctrl} is the reference signal determining the output. Due to op-amp action, $V_{V_{\text{out}}}$ (the potential of the inverting input) will equal to V_{ctrl} (the potential of the non-inverting input). If $V_{V_{\text{out}}}$ changes for any reason, e.g. it becomes lower, then the output of the error amplifier rises causing the base-emitter voltage, and hence the collector current of Q1 to increase, which in turn raises $V_{V_{\text{out}}}$ until it equals V_{ctrl} again. This causes the output of the regulator to be

$$V_{\text{out}} = \left(1 + \frac{R_4}{R_5}\right) \cdot V_{V_{\text{out}}} = \left(1 + \frac{R_4}{R_5}\right) \cdot V_{\text{ctrl}} \cong 6.77 \cdot V_{\text{ctrl}}. \qquad (5.1)$$

The reason for providing the feedback signal to the error amplifier through a voltage divider instead of feeding back V_{out} directly, is that V_{ctrl} can only go up to 2.5 V. The values of R4 and R5 are selected so that for $V_{\text{ctrl}} = 2.5\ \text{V}$ the output voltage reaches at least the required $16\ \mathrm{V}$. In fact, for the resistance values shown in Table 5.4, the full-scale value of V_{out} is

$$V_{\mathrm{out,FS}} = 6.77 \cdot 2.5\ \mathrm{V} \cong 16.92\ \mathrm{V}, \tag{5.2}$$

which allows a desirable headroom for V_{ctrl} of around $137\ \mathrm{mV}$. The term "headroom" refers to the fact that V_{ctrl} does not have to be driven to its full-scale value of $2.5\ \mathrm{V}$ to reach the highest desired output voltage of $16\ \mathrm{V}$. Instead, $V_{\mathrm{out}} = 16\ \mathrm{V}$ is reached when $V_{\mathrm{ctrl}} = 16\ \mathrm{V}/6.77 \cong 2.363\ \mathrm{V}$, which is $137\ \mathrm{mV}$ below $2.5\ \mathrm{V}$.

Resistor R_7 provides overcurrent protection to the base-emitter diode of Q_1 in case the output is short circuited. The components R_8 and C_1 ensure the stability of the regulator's control loop by means of dominant-pole compensation (i.e. by reducing the open-loop gain so that it rolls off with frequency to below $0\ \mathrm{dB}$ before the open-loop phase shift reaches 180°) [25]. Their values were chosen experimentally.

The circuit described above will regulate the output current if instead of $V_{V_{\mathrm{out}}}$, a voltage proportional to the output current is fed back to the error amplifier. This voltage ($V_{I_{\mathrm{out}}}$) is generated by the components R_1, IC_2, Q_2, R_2 and R_3, which form a high-side current measurement circuit. Feeding back $V_{I_{\mathrm{out}}}$ is achieved by simply switching IC_3 to its upper position.

The high-side current measurement works as follows. Since Q_1 operates in its linear region, $I_{\mathrm{C}} \approx I_{\mathrm{E}}$ will apply, where I_{C} and I_{E} are the transistor's collector and emitter current. Hence, assuming that the feedback voltage divider draws negligible current, the current through R_1 will practically equal to I_{out} and due to Ohm's law, the voltage across R_1 will be directly proportional to I_{out}. IC_2 and Q_2 operate in a similar control loop as IC_1 and Q_1, which ensures that the voltage across R_2 equals to that across R_1. Finally, since the drain current of P-channel MOSFET Q_2 equals to its source current, the voltage across R_3 ($V_{I_{\mathrm{out}}}$) will be proportional to that across R_2 (because they share the same current), which is in turn proportional to I_{out}. Based on these considerations it is easy to show that

$$V_{I_{\mathrm{out}}} = \frac{R_1 R_3}{R_2} I_{\mathrm{out}}. \tag{5.3}$$

Since $V_{I_{\mathrm{out}}}$ equals to V_{ctrl} thanks to IC_1, the output current in current source mode (i.e. if the switch is in its upper position) can be expressed as

$$I_{\mathrm{out}} = \frac{R_2}{R_1 R_3} V_{I_{\mathrm{out}}} = \frac{R_2}{R_1 R_3} V_{\mathrm{ctrl}} = 240\ \frac{\mathrm{mA}}{\mathrm{V}} \cdot V_{\mathrm{ctrl}}. \tag{5.4}$$

The role of R_6 is to compensate for the input bias current of IC_2. The resistor R_1 is effectively a current shunt and its value was chosen relatively small to minimize the voltage drop across it, but high enough to provide good signal fidelity. The values of R_2 and R_3 were chosen so that the full-scale value of I_{out} (i.e. its value when $V_{\mathrm{ctrl}} = 2.5\ \mathrm{V}$) is at least the required $550\ \mathrm{mA}$. With the resistances shown in Table 5.4, this value will be

$$I_{\mathrm{out,FS}} = 240\ \frac{\mathrm{mA}}{\mathrm{V}} \cdot 2.5\ \mathrm{V} = 600\ \mathrm{mA}, \tag{5.5}$$

giving a headroom for V_{ctrl} of $2.5\ V - 550\ \mathrm{mA}/(240\ \mathrm{mA/V}) \cong 208\ \mathrm{mV}$.

The voltages $V_{V_{\text{out}}}$ and $V_{I_{\text{out}}}$ are not only used for feedback within the output stage, but are also routed out to the low-level control unit so that it can measure the output voltage and current of the channel (see signals *V_Vout* and *V_Iout* in Table 5.3). The points providing these voltages have nonzero output resistances (as in their Thevenin equivalent circuits), which are the following:

$$\begin{aligned} R_{\text{outV}} &= R_4 \parallel R_5 = \frac{R_4 R_5}{R_4 + R_5} \\ R_{\text{outI}} &= R_3, \end{aligned} \tag{5.6}$$

where R_{outV} is the output resistance of the point providing $V_{V_{\text{out}}}$ and R_{outI} is the output resistance of the point providing $V_{I_{\text{out}}}$.

The purpose of the RC lowpass filter formed by R_9 and C_2 is to limit the slew rate of V_{ctrl}. As described in Section 5.5.2, the output stage functions as part of a control loop where the VI curve reference circuit measures the channel output and changes V_{ctrl} to set a new output. To point out the importance of the lowpass filter, let us analyze the transient behavior of this control loop when V_{ctrl} is connected directly to the non-inverting input of IC_1 without lowpass filtering.

Let us suppose that the load on the channel output has just changed and the operating point has not stabilized yet on the VI curve, but, instead it is at point 1 as shown in part (a) of Fig. 5.8 (the operating point will always lie on the load line due to Ohm's law). Also, let us suppose that the output stage is in current source mode.

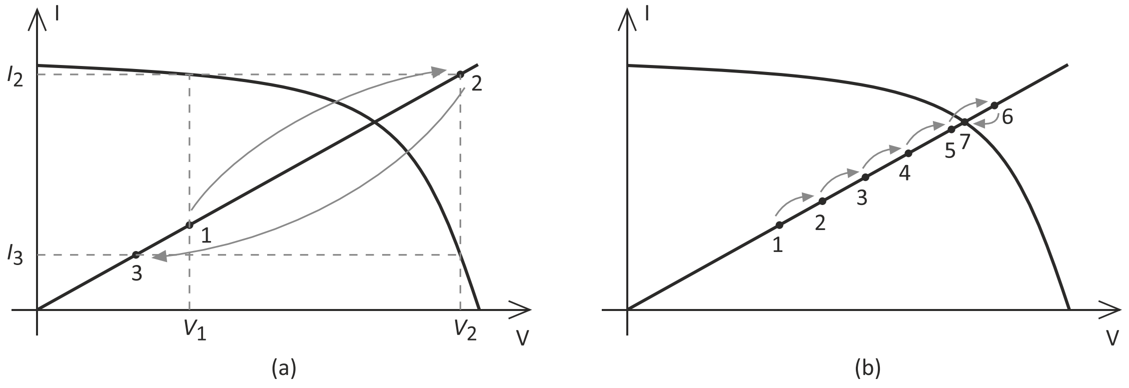


Figure 5.8 – Movement of the operating point
(a) With unfiltered and (b) with filtered control voltage.

When the VI curve reference inspects the output, it will measure voltage V_1 and hence will immediately start regulating the output current to I_2. Since the reference circuit is digital, it can only sample the output voltage at a finite rate. This means that when the next sample is taken, the output will already have stabilized at point 2, because the settling time of the output stage is likely to be much shorter than the sampling period. Hence, the reference circuit will now measure V_2 and instruct the output stage to set the output current to I_3, which shifts the operating point to position 3 before the next sample is taken. This process continues and depending on its starting position, the operating point will either oscillate around the VI curve or stabilize itself on it only after several over- and undershoots.

By limiting the slew rate of V_{ctrl}, the output current will not jump to the new value immediately, but will gradually approach it. This gives the reference circuit enough time to re-measure the output voltage as it changes, and continuously readjust the output current so that it finally lands on the VI curve. This situation is illustrated by Fig. 5.8 (b).

The time constant of the RC filter is a tradeoff between stability and speed. It should be selected so that the control loop is always stable but should not be too high as that increases the overall settling time of the SAS. The time constant given by the component values shown in Table 5.4 has been found to eliminate oscillation completely but to still keep the SAS sufficiently quick.

5.6.3.2 Component Selection

To minimize the current measurement error introduced by the fact that in reality $I_{\mathrm{C}} < I_{\mathrm{E}}$ for Q_1, the Darlington transistor TIP142 is used, whose high current gain of $\beta \approx 1000$ at $I_{\mathrm{C}} = 500\ \mathrm{mA}$ makes the error negligible [26]. An N-channel MOSFET would be an even better choice in that regard due to its drain and source current being exactly equal, but a MOSFET usually needs several volts of gate to source voltage at high drain currents. This means that the output voltage of the error amplifier may need to go higher than $19\ \mathrm{V}$ at full load, necessitating the use of a higher supply voltage. Using a P-channel MOSFET would mitigate this problem and would reduce the overall voltage dropout of the regulator (as would a PNP bipolar transistor), but regulators in such a low-dropout (LDO) configuration are much harder to stabilize under loads with different input capacitances. Since we generally do not assume anything about the load connected to the SAS, the LDO configuration has been rejected. The heatsink symbol next to Q_1 represents the fact that the device needs appropriate cooling (see Section 5.6.7).

The supply voltage was chosen based on the dropout voltage of the regulator, i.e. the maximum voltage drop across the components which are in series with the load (that is, R_1 and Q_1). The minimum required supply voltage ($V_{\mathrm{s,min}}$) can be expressed as

$$\begin{aligned} V_{\mathrm{s,min}} &= V_{\mathrm{out,max}} + V_{\mathrm{DO}} \\ &= V_{\mathrm{out,max}} + R_1 \cdot I_{\mathrm{out,max}} + V_{\mathrm{CE,sat}} \\ &= 16\ \mathrm{V} + 1\ \Omega \cdot 550\ \mathrm{mA} + 2\ \mathrm{V} \\ &= 18.55\ \mathrm{V}, \end{aligned} \tag{5.7}$$

where $V_{\mathrm{out,max}}$ and $I_{\mathrm{out,max}}$ are the maximum load voltage and current, V_{DO} is the regulator's dropout voltage and $V_{\mathrm{CE,sat}}$ is the collector-emitter saturation voltage of Q_1. To have some extra margin compared to the value above, the supply voltage was chosen to be $V_{\mathrm{s}} = 19\ \mathrm{V}$.

The input offset voltage of IC_2 has a direct influence on the accuracy of the high-side current measurement. It can be shown that for a nonzero offset voltage the output of the current measurement will deviate from the value expressed in Equation (5.3) in the following way:

$$V'_{I_{\mathrm{out}}} = \frac{R_1 R_3}{R_2} I_{\mathrm{out}} - \frac{V_{\mathrm{os}}}{R_1}, \tag{5.8}$$

where V_{os} is the input offset voltage of IC2. To minimize this effect, the precision op-amp LT6015 with a typical offset voltage of $\pm 45\ \mu V$ is used. Even though for IC1 no such high precision is needed, the same device is used there for the sake of simplicity.

Since resistors R_1, R_2, R_3, R_4 and R_5 determine the output voltage and current, any error in their values influences the overall accuracy of the output stage. Therefore, low tolerance (1%) resistors were used to realize them.

Since there are no specific requirements against the analog switch apart from the fact that it should be controllable with $3.3\ V$ logic signals, a generic MOSFET-based switch, the ADG419 has been selected. For Q_2 a general small signal device suffices, so the widely available BSS84 was chosen.

5.6.4 Low-Level Control Unit

5.6.4.1 Implementation Platform

The task of the digital control unit of the SaS V2 is two-fold. On the one hand, it has to communicate with the HiL PC and the user, and on the other hand it has to implement the VI curve reference for the 3 channels. The latter task is time-critical, meaning that the reference should react to changes at its input as fast as possible to ensure that the operating point trips back onto the VI curve quickly following a load variation. This drove the decision to split the controller into a low and a high-level unit, so that the VI curve reference can fulfill its duty uninterrupted.

As the implementation platform, the BeagleBone Black single-board computer was chosen whose simplified block diagram is shown in Fig. 5.9 (only parts relevant to this design are displayed).

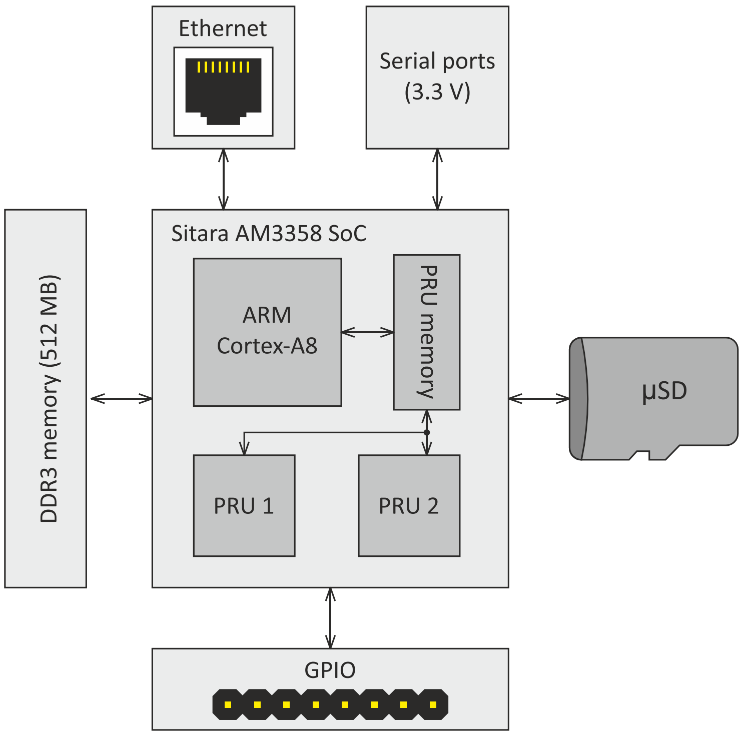


Figure 5.9 – Simplified block diagram of the BeagleBone Black single-board computer

The device is essentially a Linux-based embedded computer built around a Sitara AM3358 system-on-chip (SoC) with an ARM Cortex-A8 processor in it, running at $1\ \mathrm{GHz}$. The BeagleBone Black has Ethernet connectivity, which makes it suitable as the high-level controller. Apart from the main processor, the SoC found on the board also incorporates two microcontrollers, or programmable real-time units (PRUs) which are based on 32-bit cores clocked at $200\ \mathrm{MHz}$. These are ideal for implementing the low-level controller. The system also hosts $512\ \mathrm{Mbytes}$ of DDR3 RAM, four serial ports and a microSD card slot for the operating system and backup storage. The size of the microSD card used in this project is $16\ \mathrm{Gbytes}$.

The BeagleBone Black has several general-purpose input/output (GPIO) pins which can be controlled by both the PRUs (at high speeds and with predictable timing) and by the Linux system running on the main processor (at lower speeds and with non-deterministic timing). Since the main processor and the PRUs share part of their memory address space, communication between them is fast and easy to implement [28].

5.6.4.2 Data Acquisition Circuit

The VI curve reference for all 3 channels is implemented with one of the PRUs. To fulfill this functionality, the PRU needs extra circuitry around it, which is composed of the following parts:

- An analog-to-digital converter (ADC) to sample output voltages and currents of the channels.
- Digital-to-analog converters (DACs) to generate the reference signals for the output regulators.
- Signal conditioning circuitry for the ADC.

The analog-to-digital conversion of the various signals can be multiplexed in time, so one ADC suffices. However, 3 different reference voltages must be provided for the 3 channels at the same time, so one DAC per channel is needed. The PRU itself acts as a lookup table between the ADC and the DACs and it is also responsible for controlling these devices.

The schematic diagram of the data acquisition circuit is shown in Fig. 5.10 and component values are given in Table 5.5.

All components shown in the diagram are powered from the 3.3 V power bus generated by the BeagleBone board. The PRU communicates with the ADC and with the DACs via an SPI bus which consists of one MISO (master in slave out), one MOSI (master out slave in), one SCK (serial clock) and four SS (slave select) signals, one for each slave. The SPI bus is implemented with GPIO pins in software as the PRU does not have a hardware SPI port.

The converters are 12-bit devices, which provides the following resolution in reading and setting the channel output voltage and current:

$$\begin{aligned} V_{\mathrm{LSB}} &= \frac{V_{\mathrm{out,FS}}}{2^{12}} \cong 4.13\ \mathrm{mV} \\ I_{\mathrm{LSB}} &= \frac{I_{\mathrm{out,FS}}}{2^{12}} \cong 0.15\ \mathrm{mA}. \end{aligned} \tag{5.9}$$

In the above expressions V_{LSB} and I_{LSB} are the voltage and current corresponding to the least significant bit (LSB) of the ADC measurement or the binary DAC value, i.e. the achievable resolution in measuring and setting the channel output. $V_{\mathrm{out,FS}}$ and $I_{\mathrm{out,FS}}$ are the full-scale output voltage and current from Equations (5.2) and (5.5). Since there is no requirement to set the output with an extremely high precision, the resolutions calculated above are sufficient.

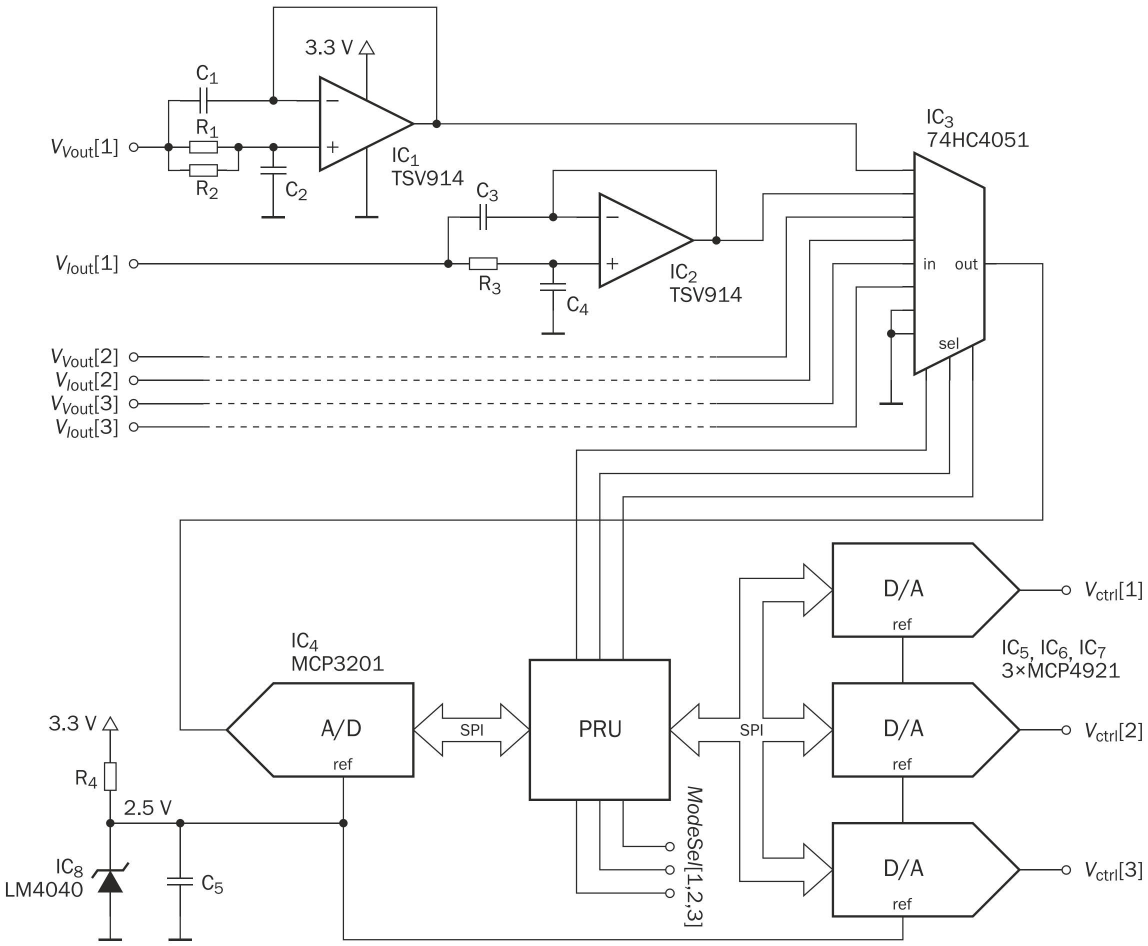


Figure 5.10 – Data acquisition module circuit diagram

Component	R1	R2	C1	C2	
Value	13 kΩ	75 kΩ	1 nF	1 nF	
Component	R3	C3	C4	R4	C5
Value	10 kΩ	1 nF	1 nF	1.6 kΩ	100 nF

Table 5.5 – Data acquisition module component values

The converters accept an external reference voltage, which must be below their supply voltage (3.3 V). In this design 2.5 V is used because voltage references for this value are commonly available and it is also high enough to provide good signal fidelity. The voltage reference is implemented with an LM4040 shunt reference IC (also called integrated Zener). The resistor R_4 sets the operating point of the Zener, while C_5 provides noise decoupling.

To route the different signals to the ADC input, an 8-to-1 analog multiplexer, the 74HC4051 is used. Three GPIO pins of the PRU are used to select between the multiplexer inputs. Another three GPIO pins (*ModeSel*[1,2,3]) are routed out to the output stages to control the mode selection switches in their feedback loop.

Successive approximation A/D converters such as the MCP3201 used in this design need to be driven from a low-impedance source on their inputs to charge their sampling capacitor quickly and achieve high sample rates. Hence, every input signal ($V_{V_{\text{out}}}[1]$ to $V_{I_{\text{out}}}[3]$) is buffered with an op-amp. These op-amps are also used to filter noise from the input signals. The passive components on the amplifier inputs (e.g. R_3, C_3 and C_4 for IC_2), together with the op-amps and the output resistances of the signal sources as seen in Equation (5.6), form a unity-gain second order Sallen and Key lowpass filter. The filters have a corner frequency of

$$f_{\text{c}} = \frac{1}{2\pi RC}, \tag{5.10}$$

where R and C are the resistor and capacitor values in the filters. (Note that all capacitances are the same and the resistance values used equal to the output resistances of the signal sources.) This gives a corner frequency of $f_{\text{c1}} = 14.36\ \text{kHz}$ for IC_1 and $f_{\text{c2}} = 15.92\ \text{kHz}$ for IC_2. These values were measured to reduce the noise of the ADC measurements to $\pm 1\ \text{LSB}$ from around $\pm 10\ \text{LSB}$ without the filters, while not slowing down the response time of the SAS significantly.

The rest of the input signals are filtered with the same kind of circuitry as $V_{V_{\text{out}}}[1]$ and $V_{I_{\text{out}}}[1]$. The filtering of $V_{V_{\text{out}}}[2]$ and $V_{V_{\text{out}}}[3]$ is identical to that of $V_{V_{\text{out}}}[1]$, while the filtering of $V_{I_{\text{out}}}[2]$ and $V_{I_{\text{out}}}[3]$ is identical to that of $V_{I_{\text{out}}}[1]$. The dashed lines in Fig. 5.10 symbolize that these filters were omitted from the circuit diagram for clarity.

The op-amps used must have rail-to-rail input-output (RRIO) capability so that they do not saturate at the maximum input signal value of $2.5\ \text{V}$ when powered from $3.3\ \text{V}$. Other parameters such as input offset voltage or bias current are not critical for these devices, so a general-purpose RRIO amplifier, the TSV914 was selected. Since four op-amps are integrated in a single IC, two chips are needed and two amplifiers in the second one are unused.

5.6.4.3 Software Design

The PRU needs to be programmed so that together with the external components presented in the previous section, it acts as a VI curve reference. The PRU must control all three output channels in parallel, which is achieved by multiplexing between them, i.e. controlling them one after another in a fast loop. For a given channel, the PRU is programmed to perform the following steps.

1. Set the ADC input multiplexer to measure the channel's output voltage, take measurement and save result to memory.
2. Set the ADC input multiplexer to measure the channel's output current, take measurement and save result to memory.
3. If channel is in voltage source mode, load measured current from memory. If channel is in current source mode, load measured voltage from memory.
4. Based on measured value, check whether the operating point resides in the voltage source or current source region of the VI curve. If necessary, change the operating mode of the channel.
5. Look up value from LUT based on ADC reading.
6. Set channel's DAC based on value read from LUT.

These steps are repeated for every channel in a loop. After addressing channel 3, the program loops back to handle channel 1, and so on.

Steps 1 and 2 involve communicating with the converter ICs via SPI. Even though for a specific operational mode of the output stage only either the voltage or the current is needed, both are measured so that the high-level controller will be able to display both values on screen to the user, and so that switching between modes is quicker.

For each channel, the program stores in memory whether it is currently in voltage or current source mode, i.e. the state of the channel's mode selection switch. In step 3, the program decides based on this information which measurement result to load back from the memory. In step 4 the program compares the mode of the active channel with the mode that should be used in the region of the VI curve where the operating point resides. If they are different, the channel mode is switched, which is done in two steps. First, the measurement result which is relevant for the new mode is loaded from memory (current in voltage source mode and vice versa). Then, the mode switch of the channel is toggled.

Which mode should be used at a specific point on the VI curve is decided by the program automatically. The program looks up the value in the LUT which corresponds with the measured output quantity (output voltage or current). Then, the program also looks up the entry right after it and computes the difference of the two values to determine the slope of the curve in that point. If the magnitude of the slope is higher than a given threshold, the mode will be changed. Otherwise, the currently active mode is kept.

The actual LUT functionality is fulfilled in steps 5 and 6, when the controller looks up the corresponding current to the measured voltage (in current source mode) or the corresponding voltage to the measured current (in voltage source mode) and the queried value is programmed into the DAC.

To make the above mechanism work, a LUT for each channel has to be loaded in the PRU memory. Since the output channels need to operate both in voltage and current source mode, two LUTs per channel are necessary at any time: one for the current versus voltage relation and one for the voltage versus current relation. The former is referred to as voltage to current or V-I LUT while the latter is referred to as current to voltage or I-V LUT. The two describe the same VI curve.

A single LUT is a series of integer numbers in memory where the position of the number within the LUT corresponds to the x-value (e.g. the voltage, in current source mode) of a point on the VI curve while the number itself corresponds to the y-value (e.g. the current, in current source mode). The first LUT entry corresponds to an x-value of 0, while the last one corresponds to (nearly) $V_{\text{out,FS}}$ or $I_{\text{out,FS}}$.[6] To make use of the full resolution of the DACs, a LUT entry must be stored on 12 bits, which takes up 2 bytes (16 bits) in the PRU memory. Since two LUTs are needed for each channel, a total of 6 LUTs must be stored. Generating and writing the tables into the PRU memory is the task of the high-level controller and will be discussed in Sections 5.6.5 and 6.2.4.

[6] Since a LUT has 256 entries (see later why), the last one, i.e. Nr. 255 actually corresponds to $255/256 \cdot \alpha$, where α is either $V_{\text{out,FS}}$ or $I_{\text{out,FS}}$.

The high-level controller should also be able to update the LUTs during program execution, but since the LUTs are constantly being read by the PRU, it is not good practice to update them in-place, as that could result in that the PRU reads inconsistent data. To avoid this problem, an additional buffer space is reserved. First, the updated LUT is written to this buffer space and then a pointer is directed to it. The memory area with the old LUT becomes the new buffer space. Each output channel has its own pointer which stores where its LUT can currently be found in memory. Together with the buffer, room for a total of 8 LUTs must be reserved. To use the full 12-bit resolution of the ADC, each of these LUTs should contain 2^{12} entries, resulting in a total size of

$$8\ \text{tables} \cdot 2^{12}\ \text{entries/table} \cdot 2\ \text{bytes/entry} = 64\ \text{kbyte}. \qquad (5.11)$$

The PRUs have only got a total of $28\ \text{kbyte}$ of data RAM. Although they can also read and write the main DDR memory, that process is significantly slower, as access to the main memory is routed through a switch fabric outside of the PRU [29]. However, the size of the LUTs can be reduced significantly without a noticeable performance penalty. In fact, it is enough to store the LUTs with 8 bits resolution, that is, 256 entries per table. This is because the LUT corresponding each operational mode of the output stage is only used in the corresponding VI curve region, where the rate of change of the value being looked up is low (refer to Fig. 5.5). Thus, even with the reduced resolution the neighboring values will be close to each other. With the reduced resolution, the LUTs will only take up

$$8\ \text{tables}\ \cdot 2^{8}\ \text{entries/table} \cdot 2\ \text{bytes/entry} = 4\ \text{kbyte} \qquad (5.12)$$

of memory space.

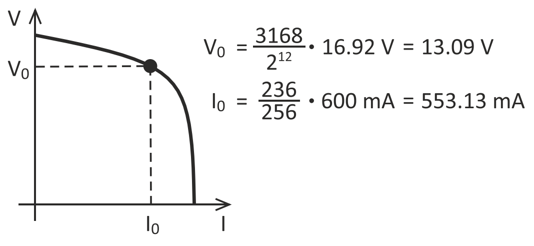
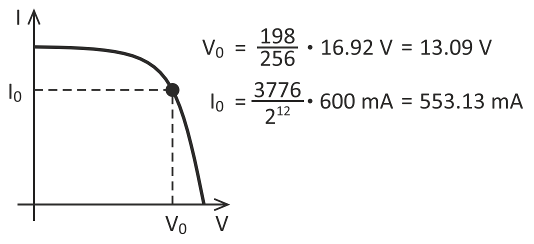
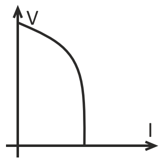
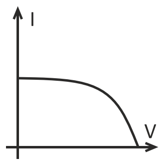
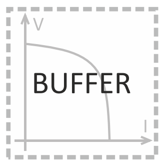
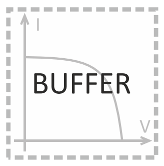
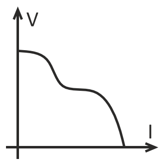
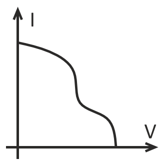
To still utilize the available precision of the measured ADC values, the PRU always looks up the two entries whose x-values are the closest to the measurement, one being lower and the other one being higher than that. Then, the program interpolates between the two LUT entries linearly.

The LUTs and the measured output quantities are placed at well-defined positions in the PRU memory so that both the PRU and the high-level controller know where to find them. To illustrate how the LUTs work, let us look at a simple example based on Table 5.6, which shows a part of the PRU memory map.

The table presents how LUTs are placed in the memory and illustrates the interpretation of their entries based on a selected point of the VI curve stored in the first table. The same point is highlighted on the curve depicting the I-V and the V-I LUT. Next to the highlighted points, the conversion between the binary and the natural values is shown. Note that in the lookup tables, each entry is stored on two addresses (two bytes). From memory address 0x1000 to 0x1002, the meaning of the LUT pointers is illustrated. At the moment that the figure captures, LUT Nr. 3 is the buffer area, while the rest of the LUTs are used by the three output channels. The full PRU memory map can be found in Appendix B.

The functionality described above along with an SPI library for the controller chips were written in the PRU assembly language and translated to machine code with the PASM assembler [30]. The main loop runs freely, i.e. it does not contain explicit timing elements such as software delays, apart from delays in the SPI library which keep the communication speed below the maximum allowed values defined in the MCP3201 and MCP4921 datasheets. The resulting sample rate of the output quantities was measured by probing one of the DAC slave select signals with an oscilloscope and was found to be $5.31\ \text{kHz}$.

Table 5.6 – Structure of LUTs in the PRU memory

Nr.	ADDRESS (HEX)	CONTENT (DEC)	NAME	MEANING
0. 236. 255.	0x0000 0x0001 ⋮ 0x01D8 0x01D9 ⋮ 0x01FE 0x01FF	⋮ 3168 ⋮	LUT_1_I-V	 V V_0 I_0 I $V_0 = \frac{3168}{2^{12}} \cdot 16.92\text{ V} = 13.09\text{ V}$ $I_0 = \frac{236}{256} \cdot 600\text{ mA} = 553.13\text{ mA}$
0. 198. 255.	0x0200 0x0201 ⋮ 0x028C 0x028D ⋮ 0x03FE 0x03FF	⋮ 3776 ⋮	LUT_1_V-I	 I I_0 V_0 V $V_0 = \frac{198}{256} \cdot 16.92\text{ V} = 13.09\text{ V}$ $I_0 = \frac{3776}{2^{12}} \cdot 600\text{ mA} = 553.13\text{ mA}$
0. 255.	0x0400 0x0401 ⋮ 0x05FE 0x05FF	⋮	LUT_2_I-V	
0. 255.	0x0600 0x0601 ⋮ 0x07FE 0x07FF	⋮	LUT_2_V-I	
0. 255.	0x0800 0x0801 ⋮ 0x09FE 0x09FF	⋮	LUT_3_I-V	
0. 255.	0x0A00 0x0A01 ⋮ 0x0BFE 0x0BFF	⋮	LUT_3_V-I	
0. 255.	0x0C00 0x0C01 ⋮ 0x0DFE 0x0DFF	⋮	LUT_4_I-V	
0. 255.	0x0E00 0x0E01 ⋮ 0x0FFE 0x0FFF	⋮	LUT_4_V-I	
	0x1000	4	CH1_LUTPTR	Channel 1 uses LUT_4_I-V and LUT_4_V-I
–	0x1001	1	CH2_LUTPTR	Channel 2 uses LUT_1_I-V and LUT_1_V-I
	0x1002	2	CH3_LUTPTR	Channel 3 uses LUT_2_I-V and LUT_2_V-I

5.6.5 High-Level Control Unit

The analog hardware and the low-level control unit create a convenient platform to simulate arbitrary VI curves. Provided that the program on the PRU is running, the high-level control unit needs to do nothing else but write the desired lookup tables into the PRU memory and update the LUT pointers.

The software of the high-level control unit runs on the main processor under the Linux operating system. In order to launch the program, provide inputs and monitor the outputs, the user has to connect to the SAS with a terminal program running on a PC. Connection is possible with secure shell (SSH) via the Ethernet port, or with serial communication via USB. The USB port of the SAS is realized with an Adafruit CP2104 UART to USB converter module connected to one of the four serial ports of the BeagleBone.

For basic VI curve simulation tasks, a simple program for the high-level controller was written in C. It takes three text files as input which define the VI curves to be simulated on the three channels. The files describe the VI curves as a set of voltage-current pairs. Each line contains a voltage and a current value separated by a comma, which together define a point of the VI curve. The list can contain an arbitrary number of points, and these can appear in any order within the file. Comments in the files are preceded by a '#' character. An example is shown in Fig. 5.11, along with the graph of the VI curve it describes.

```
# VI Curve File
# V [V], I [A]

0,  0.4  # SC point
8,  0.4
12, 0.3
14, 0.1
14, 0    # OC point
```

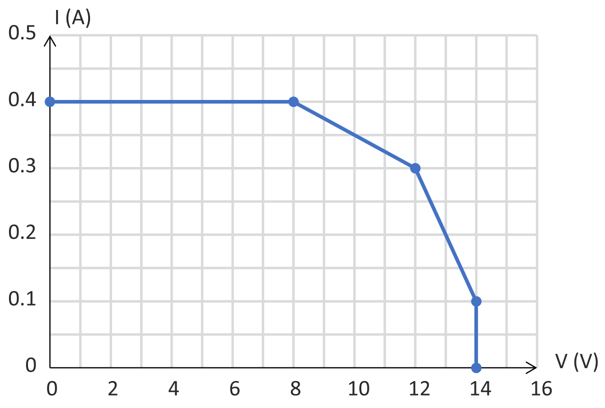


Figure 5.11 – Example VI curve file and corresponding VI curve

After the program reads in the files, it converts them to the corresponding lookup tables according to the logic illustrated earlier in Table 5.6. Between the points defined in the files, the program interpolates linearly. Finally, it places the LUTs into the PRU memory and launches the PRU program. While the program is running, the momentary output voltage and current of each channel is displayed in the terminal window.

The high-level unit also controls the output LEDs of the channels and the cooling fan. The output LEDs indicate whether a channel is on (simulating a VI curve) or off (its output current is regulated to 0). For controlling these peripherals, the GPIO library presented in [31] was used.

The high-level controller can be programmed to realize any behavior on the channels, e.g. to go through a given sequence of VI curves. In the HiL environment, the controller needs to communicate with the HiL PC and constantly update the LUTs. This will be explained in Chapter 6.

5.6.6 Power Supply and Distribution

The SaS V2 can be powered from any power supply providing a stable voltage of $19\,\mathrm{V}$ DC at a maximum of $1.85\,\mathrm{A}$, which is the highest possible current consumption of the device when all channels are fully loaded. For practical purposes, a $19\,\mathrm{V}/4.74\,\mathrm{A}$ power brick, the Meanwell GST90A19 was purchased and used with the SAS for all tests presented in this text.

The input voltage must be forwarded to the output stages, to the fan and after conversion down to $5\,\mathrm{V}$, to the BeagleBone board. This is done with a separate power distribution PCB which also hosts the MOSFET switching the fan. The circuit diagram of the power distribution module is shown in Fig. 5.12. Component values are given in Table 5.7.

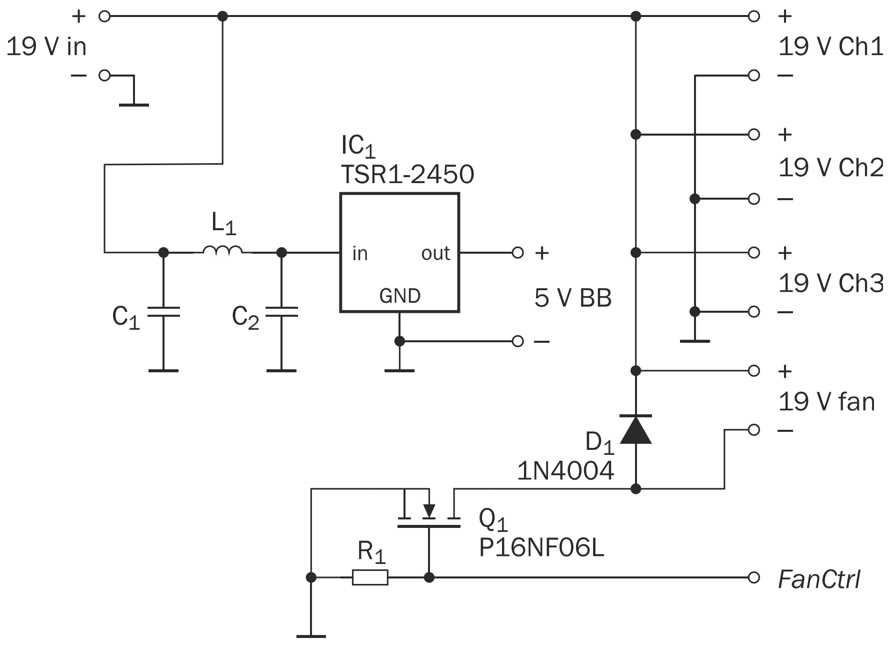


Figure 5.12 – Power distribution module circuit diagram

Component	R_1	C_1	C_2	L_1
Value	$10\,\mathrm{k\Omega}$	$10\,\mu\mathrm{F}$	$10\,\mu\mathrm{F}$	$4.7\,\mu\mathrm{H}$

Table 5.7 – Power distribution module component values

The N-channel MOSFET switching the fan on and off must have a gate-source threshold voltage ($V_{\mathrm{GS(th)}}$) lower than $3.3\,\mathrm{V}$, hence the type P16NF06L was selected with a maximum $V_{\mathrm{GS(th)}}$ of $2.5\,\mathrm{V}$. Resistor R_1 pulls the gate to ground in case the input *FanCtrl* is not driven. To protect the MOSFET against inductive kickback, a flyback diode (D_1) is connected in parallel with the fan.

The TSR1-2450 integrated switching voltage regulator is used to generate the $5\,\mathrm{V}$ needed to power the BeagleBone. The filter constructed from L_1, C_1 and C_2 is a recommendation of the regulator's datasheet to reduce EMI emissions generated by the switching action of the part [32].

No bypass capacitors are shown in the circuit diagram, because all bypassing takes place on the boards receiving the indicated voltages.

5.6.7 Thermal Design

The series pass transistors of the output stages (Q_1 in Fig. 5.7) may need to dissipate substantial power depending on the connected load and thus need appropriate cooling. For this purpose, the transistors are mounted on heatsinks. To dimension the heatsinks, the data presented in Table 5.8 is needed.

Table 5.8 – Thermal data for heatsink dimensioning

Symbol	Meaning	Value
$T_{\mathrm{J,max}}$	Maximum allowed junction temperature of Q_1	$150\ ℃$
$T_{\mathrm{A,max}}$	Maximum ambient temperature	$40\ ℃$
P_{max}	Maximum power dissipated by Q_1	$11.04\ \mathrm{W}$
$R_{\Theta\mathrm{JC}}$	Junction to case thermal resistance of Q_1	$1.92\ ℃/\mathrm{W}$
$R_{\Theta\mathrm{CS}}$	Case to sink thermal resistance of Q_1	$0.5\ ℃/\mathrm{W}$

The values $T_{\mathrm{J,max}}$ and $R_{\Theta\mathrm{JC}}$ were taken from the TIP142 datasheet [27]. The value $R_{\Theta\mathrm{CS}}$ is the typical case to sink thermal resistance of the TO-220 package that the transistor is built with [33]. The value given for $T_{\mathrm{A,max}}$ is a realistic estimate for the temperature of the air around the transistor when it gets hot. The maximum dissipated power (P_{max}) is calculated as follows:

$$\begin{aligned} P_{\mathrm{max}} &= I_{\mathrm{out,FS}} \cdot \left(V_{\mathrm{s}} - I_{\mathrm{out,FS}} \cdot R_1\right) \\ &= 600\ \mathrm{mA} \cdot (19\ \mathrm{V} - 600\ \mathrm{mA} \cdot 1\ \Omega) \\ &= 11.04\ \mathrm{W}, \end{aligned} \tag{5.13}$$

where V_s is the supply voltage of the output stage, R_1 is the current shunt resistor (see Table 5.4) and $I_{\mathrm{out,FS}}$ is the highest current that can flow through Q_1.

The junction temperature (T_{J}) of a semiconductor device that dissipates power P is calculated as follows:

$$T_{\mathrm{J}} = T_{\mathrm{A}} + (R_{\Theta\mathrm{JC}} + R_{\Theta\mathrm{CS}} + R_{\Theta\mathrm{SA}}) \cdot P, \tag{5.14}$$

where T_{A} is the ambient temperature and $R_{\Theta\mathrm{SA}}$ is the sink to ambient thermal resistance, which is an attribute of the used heatsink. The smaller this value, the better thermal conductivity it provides. By rearranging Equation (5.14), we can compute the maximum allowed thermal resistance ($R_{\Theta\mathrm{SA,max}}$) of a heatsink that keeps the junction temperature below $T_{\mathrm{J,max}}$.

$$R_{\Theta\mathrm{SA,max}} = \frac{T_{\mathrm{J,max}} - T_{\mathrm{A,max}}}{P_{\mathrm{max}}} - R_{\Theta\mathrm{JC}} - R_{\Theta\mathrm{CS}} = 7.54\ ℃/\mathrm{W}. \tag{5.15}$$

This thermal resistance is achievable with a medium-sized heatsink. In the SaS V2 the type MC33279 is used with a thermal resistance of $R_{\Theta\mathrm{SA}} = 7.1\ ℃/\mathrm{W}$.

The formula used in (5.14) assumes natural convection of the air which is not granted if the power device is in a closed case, such as the one that the SaS V2 is built into. Hence, a small fan (the type KD2405PHS2 from Sunon) at the back of the enclosure and vents at the front ensure that air constantly flows around the transistors.

It should be noted that the above numbers were calculated for the worst-case scenario when the output stage is effectively shorted and all available power is dissipated inside the SAS. During normal operation this is almost never the case and a substantial portion of the power is dissipated outside the device, i.e. on the load.

With the cooling configuration explained above, the worst-case temperature of the transistor cases was measured by setting a maximum output current of $600\ \mathrm{mA}$, shorting all outputs and waiting until thermal balance is reached. The test was carried out at room temperature (around $25\ ℃$). The maximum case temperature was measured to be $T_\mathrm{C} = 95\ ℃$, which corresponds to a junction temperature of

$$\begin{aligned} T_\mathrm{J} &= T_\mathrm{C} + R_{\Theta\mathrm{JC}} \cdot P_\mathrm{max} \\ &= 95\ ℃ + 1.92\ ℃/\mathrm{W} \cdot 11.04\ \mathrm{W} \\ &= 116.2\ ℃, \end{aligned} \tag{5.16}$$

which is below the maximum allowed junction temperature. Even though the margin between the computed $R_{\Theta\mathrm{SA,max}}$ and the chosen $R_{\Theta\mathrm{SA}}$ is small, the maximum junction temperature is safely below its upper limit thanks to constant heat convection caused by the fan.

5.6.8 Mechanical Design

The SaS V2 consists of the following physical units:

1. 3 output stages
2. Data acquisition board
3. Power distribution board
4. BeagleBone Black embedded computer
5. Adafruit CP2104 UART to USB converter module
6. Miscellaneous components: sockets, switches, fuse, cooling fan

Items 1 to 3 are the circuits shown in Figures 5.7, 5.10 and 5.12 and were realized as custom PCBs. The data acquisition board was designed as a BeagleBone cape, which is a PCB that plugs directly into the GPIO headers of the BeagleBone Black. This provides robust connections between the data acquisition circuit and the PRU. Items 4 and 5 are commercial off-the-shelf modules.

The PCBs were mounted onto an acrylic sheet and connected together with wires of $0.5\ \mathrm{mm}$ diameter for power signals and with $50\ \mathrm{mil}$ spacing ribbon cables for data signals. Finally, the whole assembly was placed in a Hammond 1598DBK plastic enclosure with dimensions of $180 \times 206 \times 64\ \mathrm{mm}$. Fig. 5.13 shows the inner construction of the device.

The front and back panels of the box were made from plastic sheets and the openings for the various connectors as well as the vents were created with a laser cutter. To label the device's connectors, stickers that match the size and the layout of the front and back panel were printed and applied onto the panels. The enclosed device is displayed in Figures 5.14 and 5.15.

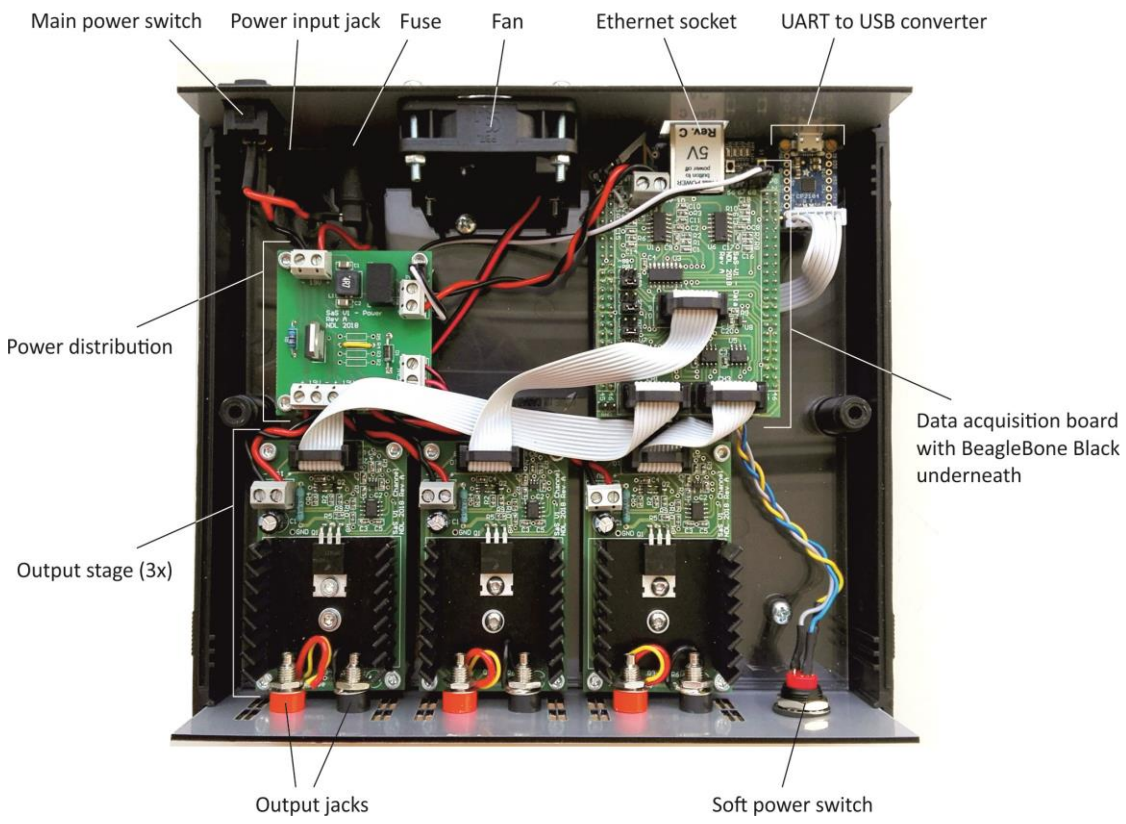


Figure 5.13 – Inner construction of the SaS V2

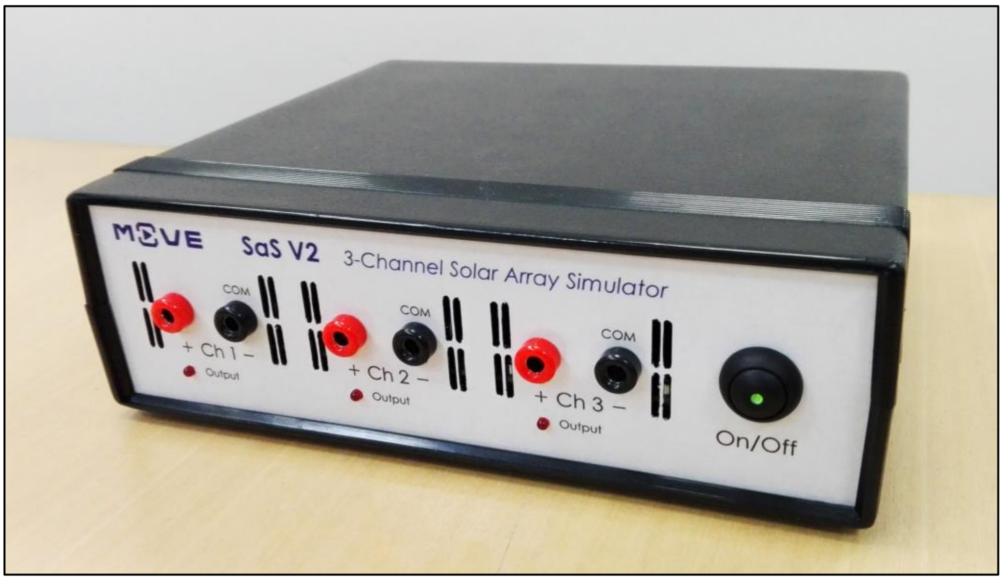


Figure 5.14 – Front panel of the SaS V2

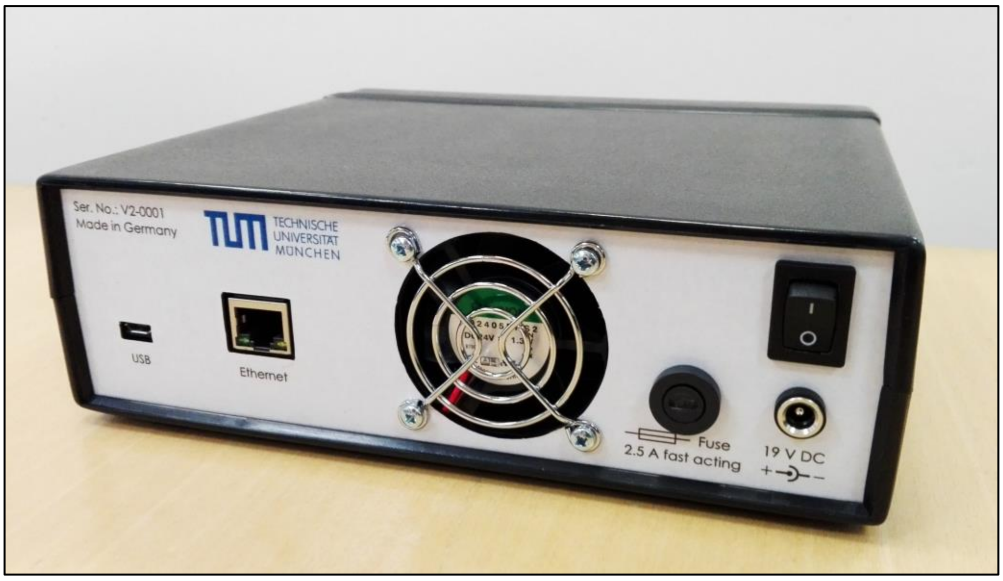


Figure 5.15 – Back panel of the SaS V2

The “COM” labels above the negative output jacks of the channels remind the user that these are connected to common ground (the channels are not isolated). The switch on the back panel disconnects the power supply plugged into the DC jack from the power distribution board when turned off. The push button at the front is a soft on/off switch for the BeagleBone Black. The fuse is a fast acting $2.5\ \mathrm{A}$ high rupturing capacity (HRC) device. It is connected between the back-panel switch and the power distribution board and protects the power supply and the SAS in case of an internal short circuit.

5.7 Verification of Functionality

5.7.1 Overview of Tests

This section demonstrates the basic operation of the SaS V2. First, its general ability to output a VI curve will be shown and then its accuracy will be measured. After that, the device’s transient response to abrupt load changes will be investigated. Since the three output channels are constructed identically, the measurements were all carried out using channel 1. To conduct the measurements, the following instruments were used:

- Uni-T 139C multimeters to measure voltage and current. This meter has an accuracy of $\pm(0.7\% + 3)$ for DC voltage and $\pm(0.7\% + 2)$ for DC current in the measurement ranges used during the tests.[7]
- Tektronix DPO 4104 digital storage oscilloscope to visualize VI curves and analyze transient response.

Data about the VI curves used for the measurements is provided in Table 5.9.

[7] The notation $\pm(a\% + b)$ means a measurement tolerance of $a\%$ of the displayed vale plus b least significant digits.

Table 5.9 – Remarkable points of the VI curves used for testing

	V_{oc}	I_{sc}	V_{MPP}	I_{MPP}
Curve 1	16 V	550 mA	13.25 V	502 mA
Curve 2	15 V	500 mA	12.29 V	228 mA
Curve 3	13.4 V	450 mA	11.5 V	430 mA

5.7.2 General Functionality

A simple and effective way to visualize the output curve of the SaS V2 is to probe the signals *V_Vout* and *V_Iout* with an oscilloscope in XY mode. Since these two voltages are proportional to the output voltage and current, the position of the point that appears on the oscilloscope screen will directly correspond to the operating point. If the load of the SAS is varied, e.g. by using a potentiometer as the load, the point will travel along the simulated VI curve. Then, by activating the infinite persistence mode of the oscilloscope and turning the potentiometer, the simulated curve will be drawn on the screen.

To verify the device's basic functionality, Curves 1 and 2 from Table 5.9 were programmed into the SaS V2 and measured with the method described above. Screen captures from the oscilloscope are shown in Fig. 5.16.

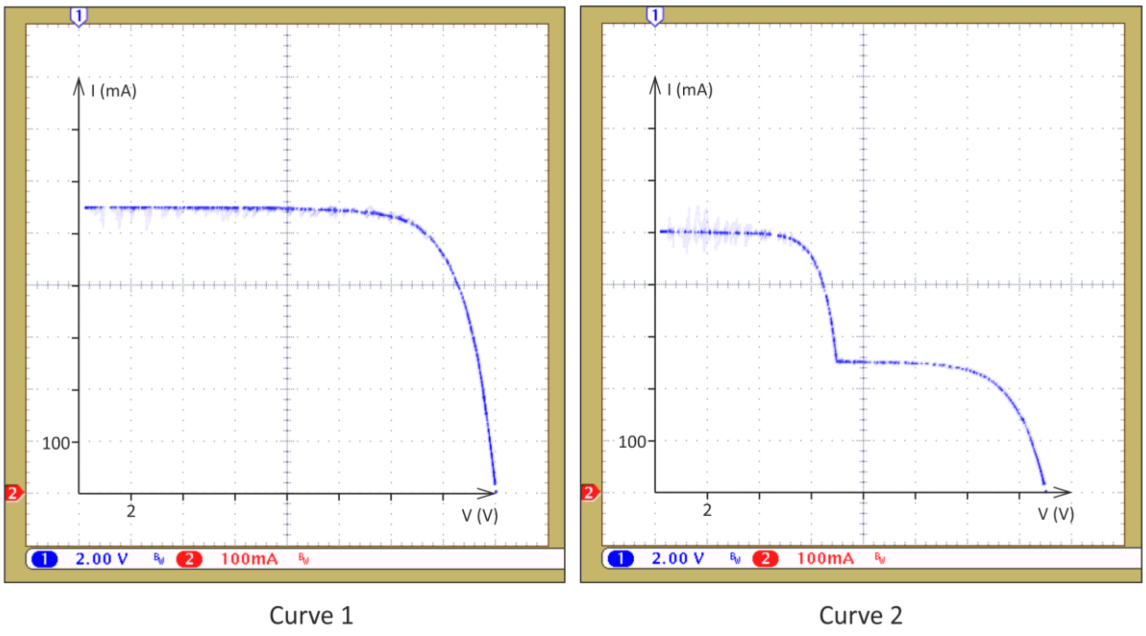


Figure 5.16 – Simulated VI curves captured with an oscilloscope

To aid visualization, the imaginary V-I coordinate system was drawn onto the screen capture images. Curve 1 demonstrates that the simulator can output the maximum open-circuit voltage and short-circuit current specified in Table 5.1. Curve 2 proves the ability to simulate a staircase-shaped curve, which can occur with a partially shaded solar array.

5.7.3 Accuracy

The accuracy of the simulator is a measure of similarity between the anticipated and the actual output curve. The anticipated VI curve is defined by the used VI curve file, while the actual curve was measured with two multimeters and a variable load in the setup shown in Fig. 5.17.

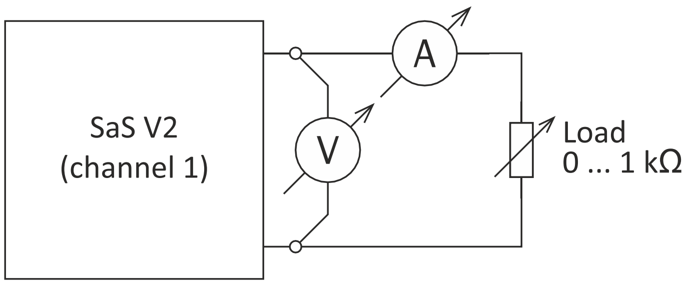


Figure 5.17 – Test setup for accuracy measurement

The VI curve used for this test was Curve 1 from Table 5.9. The load was varied from $0\ \Omega$ to $1\ \mathrm{k}\Omega$ and additionally, the open-circuit point was measured.[8]

The error, i.e. the distance between the programmed and the measured curve can be expressed either in voltage or in current. Relative errors can be computed as shown below.

$$\begin{aligned} \epsilon_{\mathrm{V,rel}}(I_0) &= \frac{\left|V_{\mathrm{p}}(I_0) - V_{\mathrm{m}}(I_0)\right|}{V_{\mathrm{p}}(I_0)}, \\ \epsilon_{\mathrm{I,rel}}(V_0) &= \frac{\left|I_{\mathrm{p}}(V_0) - I_{\mathrm{m}}(V_0)\right|}{I_{\mathrm{p}}(V_0)}, \end{aligned} \tag{5.17}$$

where the symbols have the following meaning:

- $\epsilon_{V,\mathrm{rel}}(I_0)$: relative error of output voltage at output current I_0
- $V_{\mathrm{p}}(I_0)$: voltage of the programmed curve at current I_0 (expected voltage)
- $V_{\mathrm{m}}(I_0)$: voltage of the measured curve at current I_0
- $\epsilon_{I,\mathrm{rel}}(V_0)$: relative error of output current at output voltage V_0
- $I_{\mathrm{p}}(V_0)$: current of the programmed curve at voltage V_0 (expected current)
- $I_{\mathrm{m}}(V_0)$: current of the measured curve at voltage V_0

The voltage error is meaningful in the voltage source region of the VI curve, while the current error is meaningful in the current source region. Observing for example the current error in the voltage source region may give very high errors due to the steepness of the V-I function even if the two curves deviate from each other only by a little amount. Thus, the accuracy of the simulator was determined by measuring $\epsilon_{I,\mathrm{rel}}$ in the region $V = 0 \dots V_{\mathrm{MPP}}$ and $\epsilon_{V,\mathrm{rel}}$ in the region $I = I_{\mathrm{MPP}} \dots 0$ and selecting the highest error from each region.

The programmed and the measured curves are plotted in Fig. 5.18. The MPP, which marks the border between current and voltage error measurement, is highlighted. The relative current and voltage errors are plotted in Figures 5.19

[8] The true short-circuit point cannot be measured with this setup due to the internal resistance of the current meter (around $2\ \Omega$).

and 5.20, respectively. Based on the highest error values, the accuracy of the device for this curve is the following:

- $\epsilon_{I,\mathrm{rel,max}} = 0.39\%$
- $\epsilon_{V,\mathrm{rel,max}} = 0.39\%$

The accuracy is better than 0.5% in both cases, which will be sufficient for completing the power budget verification task.

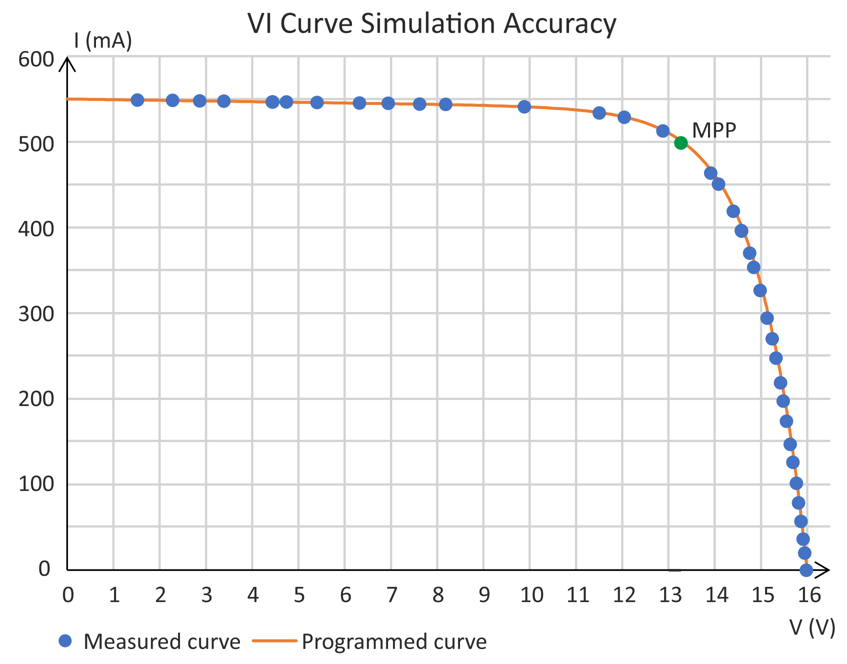


Figure 5.18 – Plot of programmed and measured VI curve

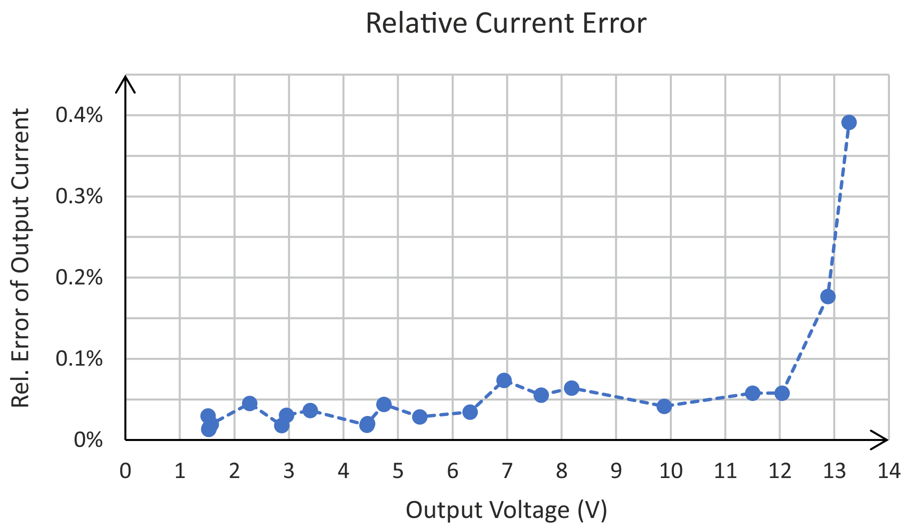


Figure 5.19 – Relative error of output current over output voltage

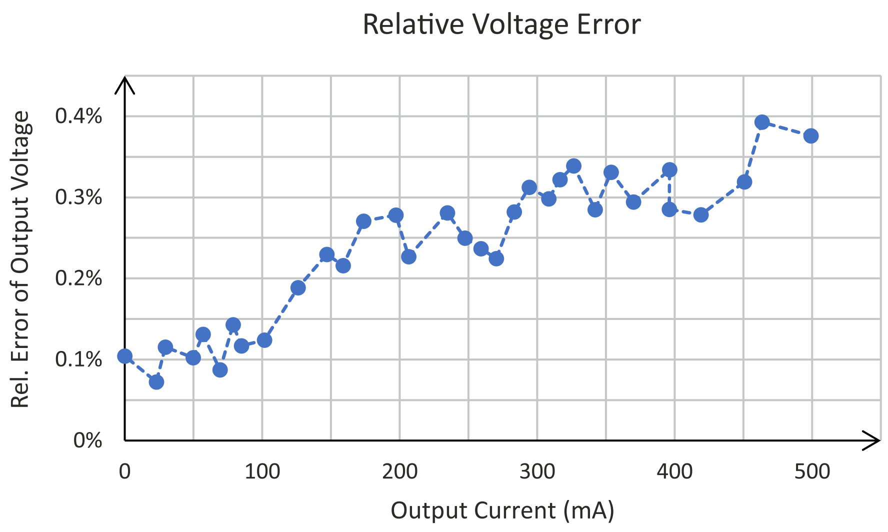


Figure 5.20 – Relative error of output voltage over output current

5.7.4 Transient Response

After the load on the output of the SAS changes, the device needs time until the operating point is driven back onto the VI curve. The transient response of the simulator describes how the output quantities change until the new operating point stabilizes and how long this process takes.

The transient response of the SaS V2 was observed in two scenarios. First, Curve 1 from Table 5.9 was programmed into the device and its response to a full-scale step was measured. A full-scale step means that the load is changed from short to open circuit as well as from open to short circuit. In the second scenario, Curve 3 was programmed into the device and it was connected to BCR buck 1 of the MOVE-II engineering model. (Curve 3 was used instead of Curve 1 in order not to exceed the maximum input current specification of the BCR stated in Table 4.1.) During maximum power point tracking, the EPS is switching the operating point between the open-circuit point and the MPP. The transient behavior of the SaS V2 was measured in this case too. In both tests, the response was captured by probing the signals *V_Vout* and *V_Iout* with an oscilloscope in single-shot mode.

The results of the first test can be seen in Fig. 5.21. Part (a) of the figure shows a load change from short to open, while part (b) shows the opposite direction. Trace 1 (in blue) corresponds to the output voltage, while Trace 2 (in red) corresponds to the output current. The vertical position of zero voltage and current is marked by an arrow at the left of the screen. The load change happens at the instant of time marked by an arrow with the letter T (for “trigger”) at the top of the screen. The scale of the axes can be seen in the bottom part of the images (e.g. $2.00\ \mathrm{V}$ means $2\ \mathrm{V}$ per vertical division for Trace 1, $1.00\ \mathrm{ms}$ means $1\ \mathrm{ms}$ per horizontal division, etc.). Note that the time scale is different in the two screen captures.

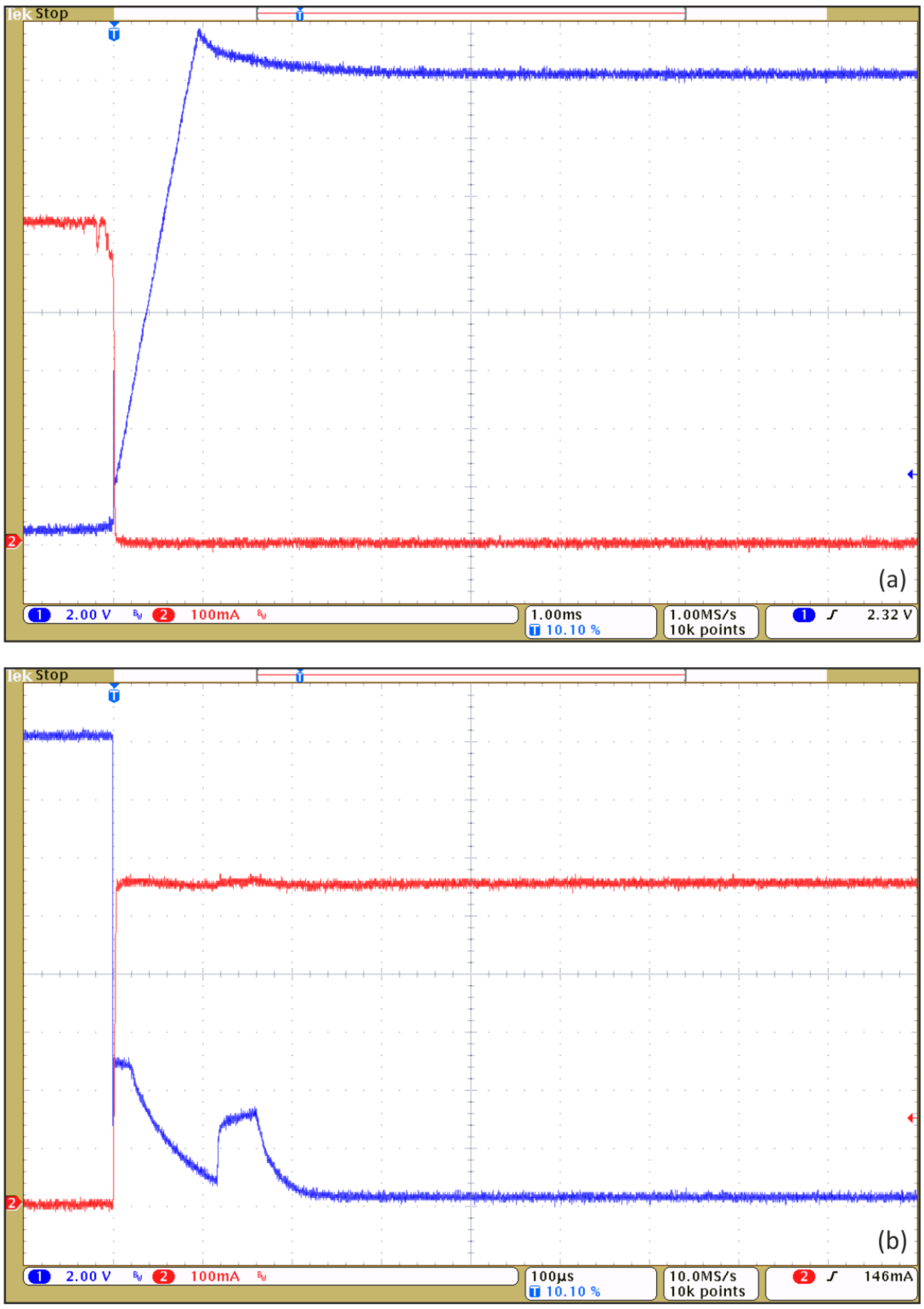
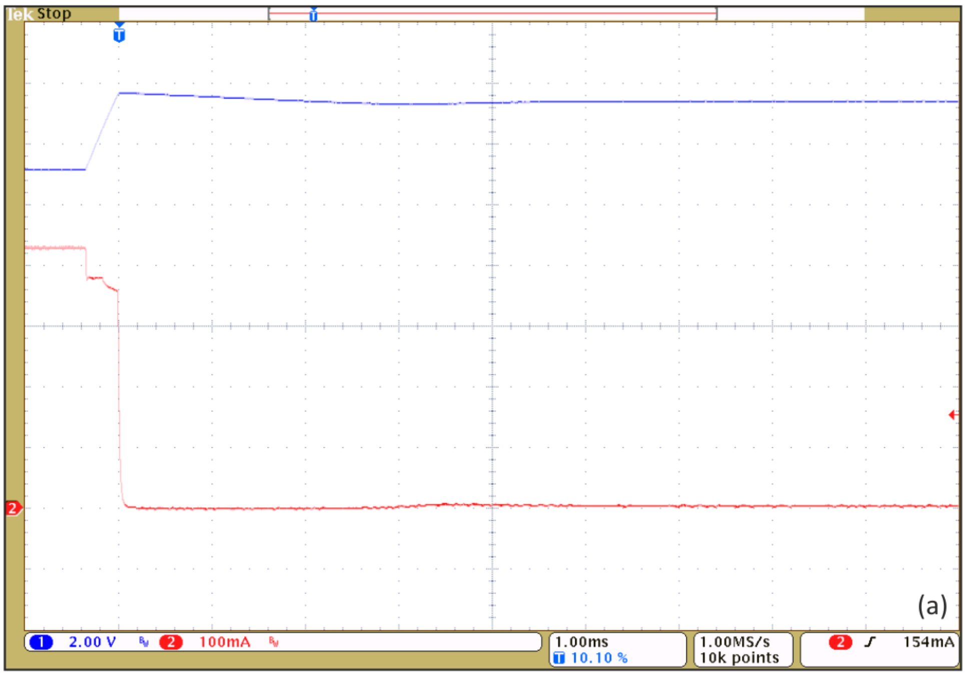


Figure 5.21 – Transient response of SaS V2 for full-scale load steps
(a) Load changes from short to open circuit,
(b) load changes from open to short circuit.
Trace 1 (blue): output voltage.
Trace 2 (red): output current.

The step response of the device in the short to open direction shows a voltage overshoot of around 10%. In the open to short direction, no significant overshoot can be seen. The operating point stabilizes within under $5\ \mathrm{ms}$ in both cases.

The results of the second test are visualized by Fig. 5.22. In part (a) the transition from the MPP to open circuit can be seen, while part (b) again shows the opposite direction. Note the different vertical scaling of Trace 2 in part (b).



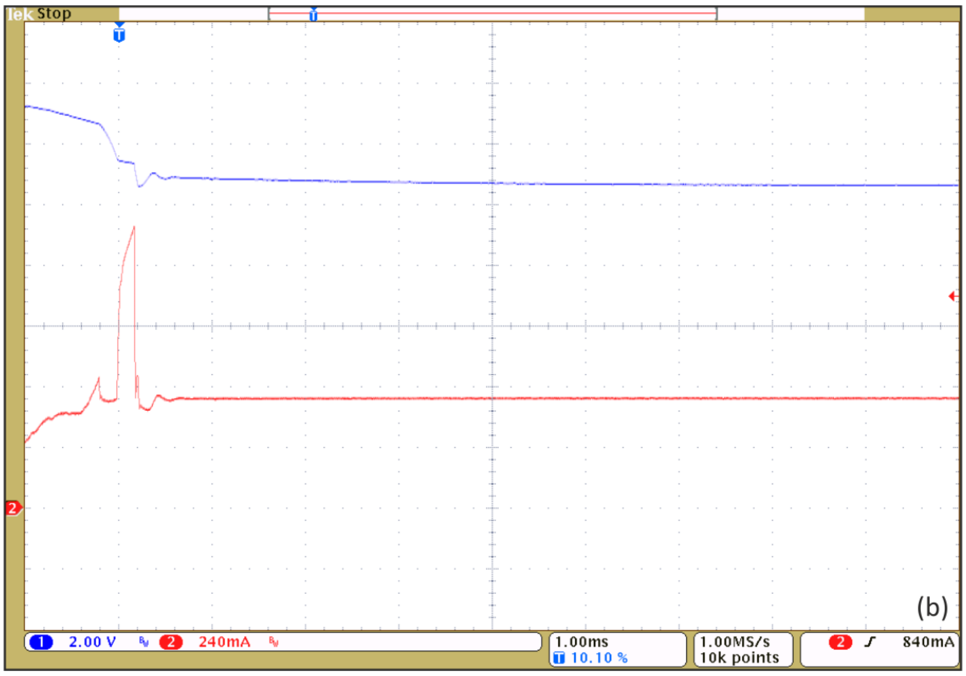


Figure 5.22 – Transient response of SaS V2 during MPPT
(a) Operating point moves from MPP to open-circuit point,
(b) operating point moves from open-circuit point to MPP.
Trace 1 (blue): output voltage.
Trace 2 (red): output current.

Settling times in both cases are again under $5\ \mathrm{ms}$. While the first step response only shows a small voltage overshoot, the second one captures a moment when the current rises to almost $1.3\ \mathrm{A}$ before it settles to its final value. The duration of the current spike is around $200\ \mu\mathrm{s}$. Even though the maximum current value exceeds the limit of $0.5\ \mathrm{A}$ shown in Table 4.1, the numbers listed there are stated for the DC value of the input currents and voltages. In fact, such current overshoots are common with traditional power supplies as well, when, due to a load change, they have to perform mode crossover, i.e. switch from constant voltage mode to constant current mode. Since such a short current spike does not carry enough energy to damage the EPS, the SAS can be safely used to power the satellite. During the spike, the SAS may technically deliver more power to the satellite than the maximum available power on the simulated curve. However, the amount of time that this happens is negligibly small, since it only occurs for $200\ \mu s$ every MPPT cycle, i.e. every $2.5\ s$ (0.008% of the time). Therefore, this phenomenon will not falsify the power budget verification results.

The transient response measurements shown above demonstrate that the settling time of the SaS V2 is below $5\ \mathrm{ms}$ ($\tau_s < 5\ \mathrm{ms}$), and hence it meets the specification defined in Table 5.1. Moreover, we can conclude that the transient response is not always free from overshoot, but this does not pose a problem for the power budget verification task.

6 Interfacing to the HiL Setup

6.1 Introduction

Now that the basic ability of the SaS V2 to simulate VI curves has been verified, the next step is to integrate the device into the HiL environment and bring the system to the extended state depicted in Fig. 5.2. To reach this goal, the high-level controller has to implement the following tasks.

1. Receive data about the state of the solar cells from the HiL PC.
2. Use the received data to generate the VI curves of the solar arrays and convert these curves to lookup tables for the low-level controller.
3. Determine the available power and measure the accepted power on each channel.
4. Send the data about available and accepted power to the HiL PC.

The available power on a channel is the power of the MPP on the VI curve currently simulated on that channel. The accepted power is the power that is being consumed by the connected BCR, i.e. the power of the actual operating point.

The communication with the HiL PC (points 1 and 4), the VI curve computation (point 2) and the channel power determination (point 3) are three tasks that need to run in parallel and have to communicate to each other. They are implemented as three threads of a single C program that runs on the high-level controller. This chapter explains how each of these tasks have been realized.

6.2 Computation of VI Curves

6.2.1 Mathematical Model of a Solar Cell

During the HiL simulation, the simulation PC only provides the illumination and the temperature of the individual solar cells and it is the task of the SAS to translate this information into full VI curves of the three solar arrays. One way to do this would be to measure and store the VI curves of the solar cells for every combination of illumination and temperature that may occur during the simulation. Since this would be an extremely tedious task, it is better to employ a PV cell model which lets us compute the VI curves for various temperatures and illuminations mathematically, based on information about the cells we want to model.

A widely used model of a solar cell is the single-diode model shown in Fig. 6.1 [36]. In the circuit diagram, the current source models the photo-generated current, the diode models the PN junction of the solar cell, while R_p and R_s model parasitic resistances of the cell. Symbols seen in Fig. 6.1 have the following meanings:

- I_{pv} is the photo-generated (photovoltaic) current.
- I_0, a and V_{T} are parameters of the diode which appear in the Shockley-equation describing its voltage-current characteristics (see later).
- R_{p} is the equivalent parallel output resistance. Typically large ($R_{\mathrm{p}} > 1\ \mathrm{k\Omega}$).

- R_s is the equivalent series output resistance. Typically small ($R_\mathrm{s} < 1\,\Omega$).
- I is the output current.
- V is the output voltage.

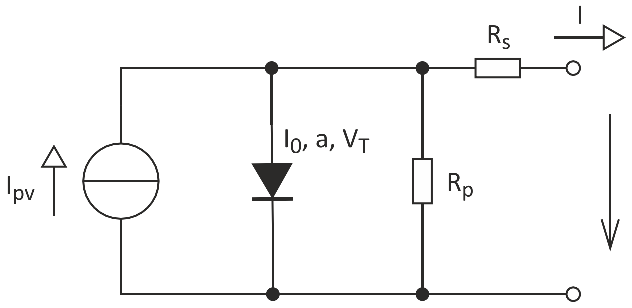


Figure 6.1 – Single-diode model of a solar cell

Based on the equivalent circuit, the relationship between the output voltage and current can be summarized in a single equation, which is easily obtained via the following steps.

I_pv can be expressed as

$$I_\mathrm{pv} = I_\mathrm{D} + \frac{V_\mathrm{D}}{R_\mathrm{p}} + I, \tag{6.1}$$

where V_D and I_D are the diode forward voltage and current, respectively. Since R$_\mathrm{p}$ is in parallel with the diode, V_D is also the voltage across R$_\mathrm{p}$. Moreover, it can be expressed as

$$V_\mathrm{D} = R_\mathrm{s}I + V. \tag{6.2}$$

The relationship between V_D and I_D is given by the Shockley diode equation [37]:

$$I_\mathrm{D} = I_0\left(e^{\frac{V_\mathrm{D}}{a\cdot V_\mathrm{T}}} - 1\right), \tag{6.3}$$

where I_0 is the diode's reverse saturation current, V_T is the thermal voltage and a is the diode ideality factor, a dimensionless quantity typically ranging from 1 to 2. The thermal voltage is a function of temperature and is expressed as

$$V_\mathrm{T} = \frac{kT}{q}, \tag{6.4}$$

where k is the Boltzmann constant, q is the unit charge and T is the absolute temperature in kelvins. By merging Equations (6.1) to (6.3) we obtain

$$I_\mathrm{pv} - I_0\left(e^{\frac{R_\mathrm{s}I+V}{a\cdot V_\mathrm{T}}} - 1\right) - \frac{R_\mathrm{s}I + V}{R_\mathrm{p}} - I = 0, \tag{6.5}$$

which is an implicit nonlinear equation describing the V-I relationship of a single solar cell.

The model can be extended so that it describes an assembly of cells connected in series if they are illuminated equally and are at the same temperature. This can be assumed for small solar panels and for multi-junction solar cells (multiple PN junctions stacked on each other), since the individual cells within these devices are all in the same plane and within a relatively small area, compared with, for instance a solar array composed of multiple PV panels. In the extended model, the diode of the equivalent circuit in Fig. 6.1 is replaced by n serially connected diodes, n being the number of PN junctions within the PV panel or the multi-junction cell. This model is still called a single-diode model and should not be confused with multi-diode models where the extra diodes stand for additional physical effects within a single cell.

The current through the string of series diodes is equal to the current of an individual diode within the string and can be expressed as

$$I_\mathrm{D} = I_0\left(e^{\frac{V_\mathrm{D}}{n\cdot a\cdot V_\mathrm{T}}} - 1\right). \tag{6.6}$$

Equation (6.6) is obtained from (6.3) by substituting V_D with V_D/n, since the total forward voltage, V_D, is now distributed equally amongst n diodes (or almost equally, in the case of a multi-junction cell composed of different PN junctions). Hence, the V-I relationship of n equally illuminated series cells at equal temperature is given by

$$I_\mathrm{pv} - I_0\left(e^{\frac{R_\mathrm{s}I+V}{n\cdot a\cdot V_\mathrm{T}}} - 1\right) - \frac{R_\mathrm{s}I + V}{R_\mathrm{p}} - I = 0. \tag{6.7}$$

For the sake of completeness we mention that a similar extension of the single-diode model is possible for n cells in parallel connection. In this case, I_pv and I_0 have to be substituted in (6.5) by nI_pv and nI_0, respectively.

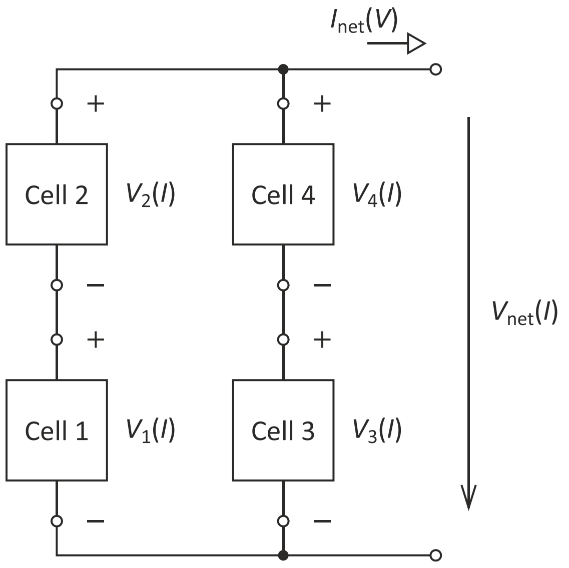
Equation (6.7) can be used to compute the VI curve of a single cell (with $n = 1$) or of multiple serially connected cells at the same illumination and temperature (with $n > 1$). Since neither V or I can be expressed from (6.7) explicitly, the equation needs to be solved numerically to find the output current for a given output voltage, or vice versa. This does not pose a problem in practice because the V-I relationship is monotonous and hence a single solution exists for any voltage or current value on the VI curve. Therefore, the computation of the VI curve is straightforward, as long as the model parameters (I_pv, I_0, n, a, V_T, R_s and R_p) are known. Using (6.7) to plot I versus V yields the familiar solar cell VI curve seen for example in Fig. 1.3.

Some other models for solar cells also exist. For example, the double-diode model [34] uses an extra parallel diode in the equivalent circuit to represent the effect of recombination carriers. The no-resistor model is equivalent to the single-diode model but without parasitic resistances, i.e. with $R_\mathrm{s} = 0$ and $R_\mathrm{p} \to \infty$. The benefit of this model is that it allows V and I to be expressed explicitly [18]. In the following discussion the single-diode model will be used because it is sufficiently accurate for our purposes and it is relatively simple.

6.2.2 From Solar Cells to Solar Arrays

6.2.2.1 Arrays without Protection Diodes

If PV cells are combined into a solar array, then the uniform illumination and temperature of the cells is generally not granted and hence the assembly can no longer be modeled with just one current source and diodes with identical parameters. However, the VI curve of the array can still be obtained by computing the curves of the cells individually and then summing the currents of the curves for parallel cells and the voltages of the curves for series cells. As an example, let us look at Fig. 6.2 which shows a solar array and how its net VI curve can be found.



$$
\begin{aligned}
&1.)\quad V_{12}(I) = V_1(I) + V_2(I)\\
&2.)\quad V_{34}(I) = V_3(I) + V_4(I)\\
&3.)\quad V_{12}(I) \rightarrow I_{12}(V)\\
&4.)\quad V_{34}(I) \rightarrow I_{34}(V)\\
&5.)\; I_{\text{net}}(V) = I_{12}(V) + I_{34}(V)\\
&6.)\; I_{\text{net}}(V) \rightarrow V_{\text{net}}(I)
\end{aligned}
$$

Figure 6.2 – Example for the computation of a solar array's net VI curve

The notation $V_i(I)$ marks the function that yields the output voltage of cell i for any output current, while $I_i(V)$ marks the inverse relationship. E.g. $V_1(I)$ is the $V(I)$ function of Cell 1. Multi-digit indices mark functions related to multiple connected cells, e.g. $V_{12}(I)$ is the net $V(I)$ function of the string composed of Cell 1 and Cell 2. Since these functions do not exist analytically, they should be thought of as dense tables of pre-computed numerical values.

In step 1, the net VI curve of the string of Cells 1 and 2 is computed by summing the voltages of the two cells corresponding to each current value. This operation is also referred to as summing the two VI curves along the current (I) axis. In step 2, the same operation is done for the string of Cell 3 and Cell 4.

In steps 3 and 4, the computed functions are inverted from the $V(I)$ form to an $I(V)$ form. Since the functions are effectively tables of values, inversion is just the question of which of the two variables in the table is considered as the independent one.

In step 5, the net VI curve of the array is computed by summing the currents of the VI curves of the two strings for each voltage value. This operation is also referred to as summing the VI curves along the voltage (V) axis.

Finally, in step 6, the net $I(V)$ function is inverted to yield the net $V(I)$ function, if this is needed. Again, this is just a change in how a table of numbers is looked at, rather than an operation.

The VI curves of the individual cells potentially have different open-circuit voltages and short-circuit currents due to their different illumination and temperature. In order to sum curves with different V_{oc} and I_{sc}, the curves also have to be known beyond these limits. Fig. 6.3 shows two VI curves with segments beyond their open-circuit voltage and short-circuit current[9] and shows their sums along both the voltage and the current axis, i.e. the net VI curves when the cells are connected in parallel and in series, respectively.

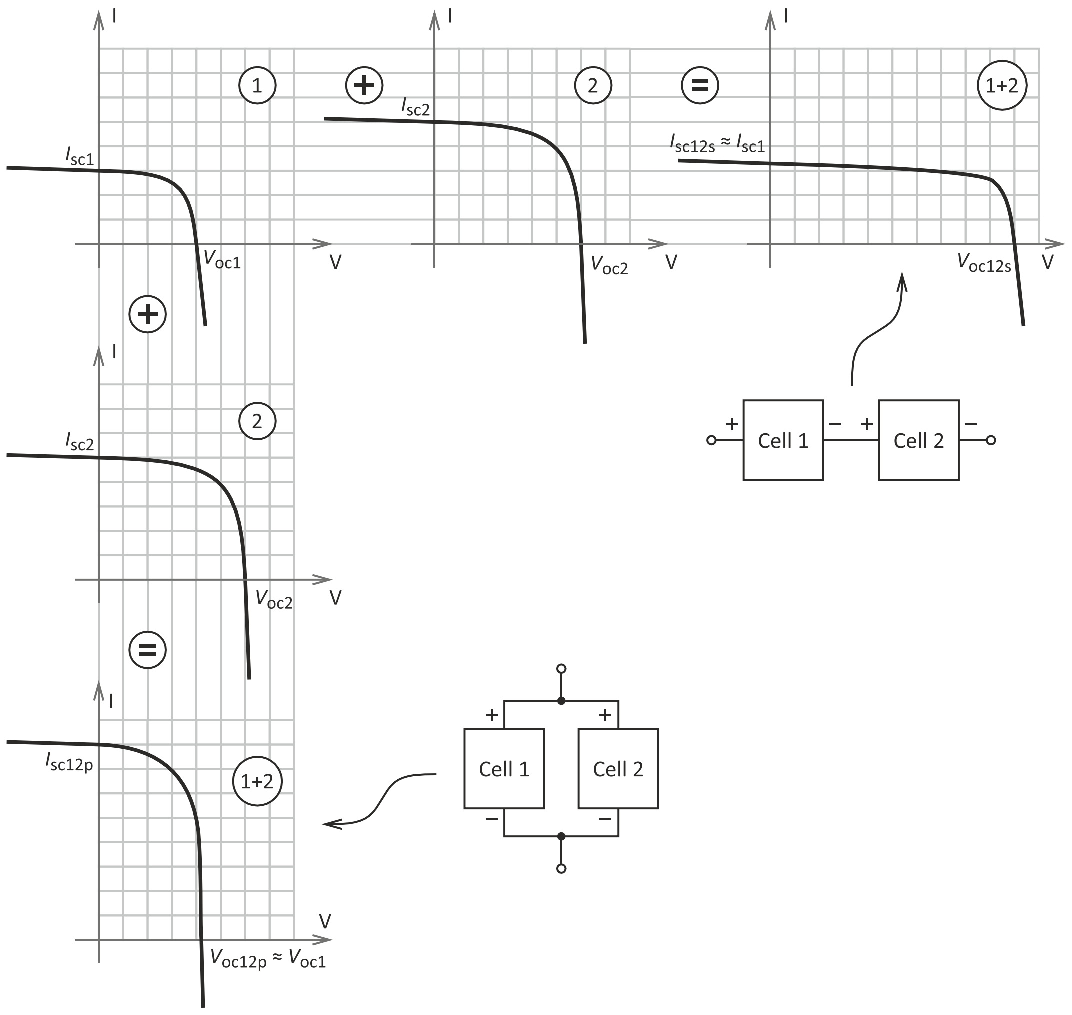


Figure 6.3 – Sums of VI curves of bare solar cells

When summing the curves along the V axis, the open-circuit voltage of the sum curve will almost equal to V_{oc1}, just slightly exceeding it. This is because Curve 1 starts yielding high negative currents beyond its open-circuit voltage rapidly and hence the sum of the currents will reach zero shortly after V_{oc1}. Physically, this means that the two cells are connected in parallel, and if the operating point exceeds V_{oc1}, then Cell 2 causes a reverse current to flow into Cell 1.

When the curves are summed along the I axis, the net short-circuit current will almost equal to I_{sc1} (slightly exceeding it) for a similar reason: beyond I_{sc1}, the voltage of Curve 1 reaches high negative values rapidly, hence the sum of the voltages will quickly become zero. What physically happens is that the cells are connected in series and the one with the lower short-circuit current, i.e. the less illuminated cell (Cell 1) blocks the current generated inside the better illuminated cell (Cell 2). If the operating point falls above I_{sc1}, most of the excess photo-current of Cell 2 flows through its internal PN-junction while some of it flows

[9] Of course, the operating point can only shift to these segments if an external voltage or current with appropriate value is applied to the cell terminals.

through the parallel resistance of Cell 1 generating a reverse voltage between its terminals.

The effects described above (reverse currents for parallel cells and current blocking for series cells) are common with partially shaded solar arrays since partial shading of the array causes the VI curves of the individual cells to be different. These effects are usually undesired because they decrease the maximum power of the array and can damage the cells experiencing the reverse currents or voltages. Fortunately, the problem can be solved as described in the next section.

6.2.2.2 Arrays with Protection Diodes

The issue with partially shaded arrays discussed above can be mitigated by adding protection diodes to the array. For cells in series, parallel bypass diodes are connected to each cell, while for parallel cells, series blocking diodes are added to each of them. In the case of parallel strings of series cells, one blocking diode per string is used [38]. The technique is depicted in Fig. 6.4. Next to each circuit, the way how the protection diodes change the net VI curve of the assembly is shown.

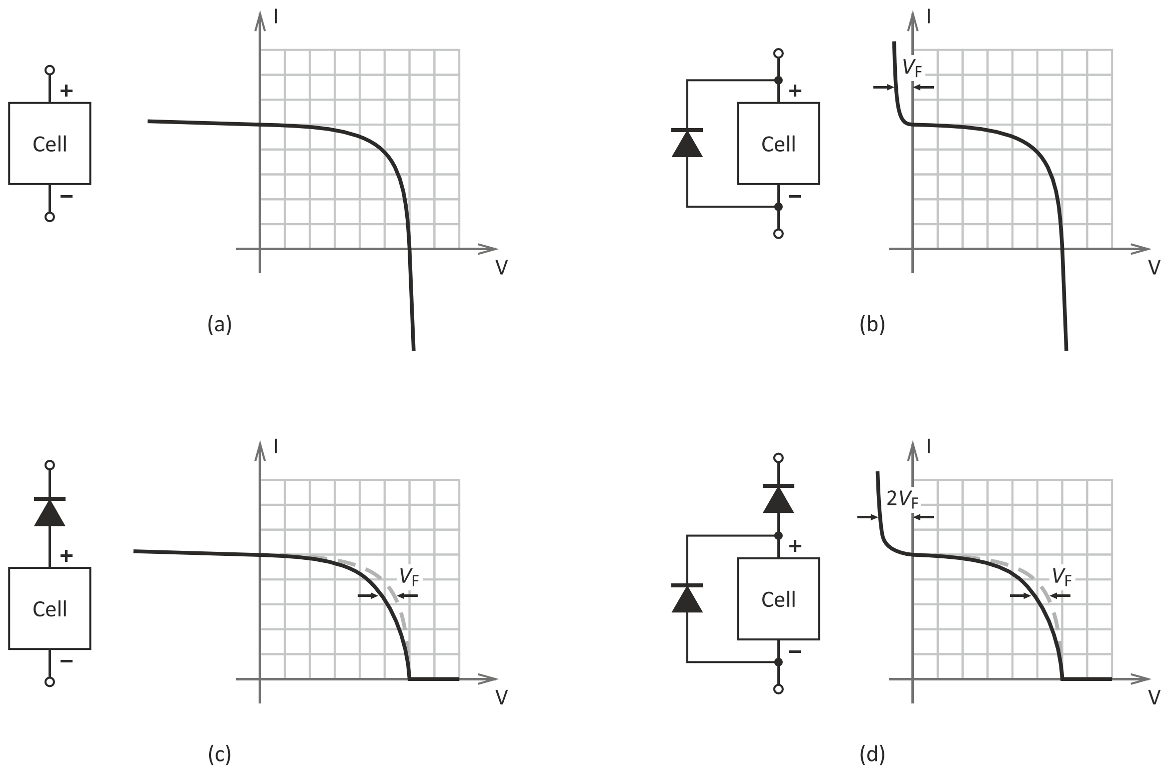


Figure 6.4 – VI curves of solar cells with and without protection diodes
(a) Without protection diodes, (b) with parallel bypass diode (c) with series blocking diode, (d) with both types of diodes.

The distances marked by arrows indicate the effect of the voltage drop across the protection diodes, and V_{F} is the diode forward voltage (around $0.6\ \mathrm{V}$). Mathematically, the method flattens out the VI curves beyond V_{oc} and I_{sc}. Physically, a series blocking diode blocks the reverse current coming from a parallel cell and a parallel bypass diode allows the excess current of a series cell to bypass the protected cell. Effectively, the extra output voltage and current that was lost in an array without protection diodes is now preserved. This typically results in a staircase-shaped net VI curve of the array. As an example, Fig. 6.5 shows the sum of the VI curves from Fig. 6.3 along both axes in the case that they have protection diodes. The voltage drop across the protection diodes is neglected in the figure.

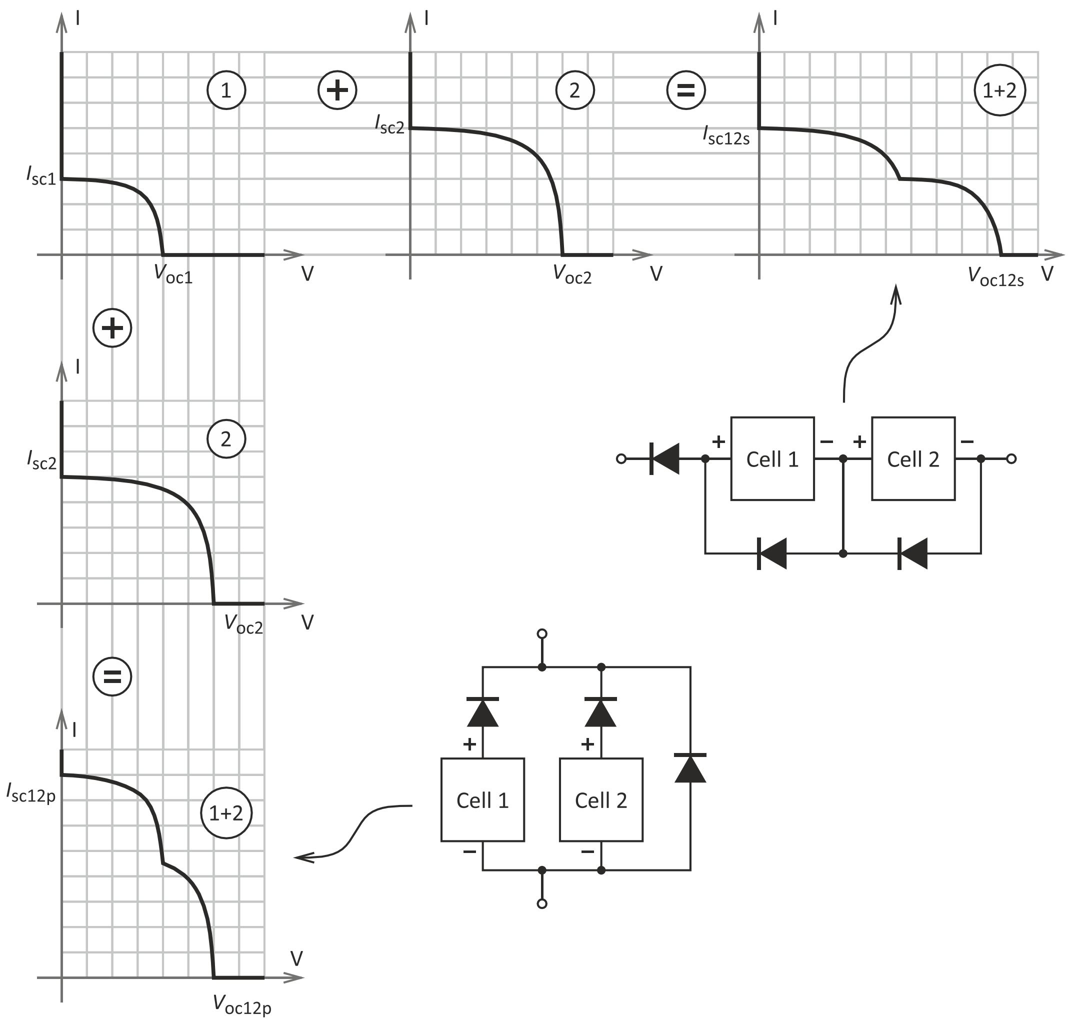


Figure 6.5 – Sums of VI curves of protected solar cells

6.2.3 Parameter Determination

In the previous two sections it was shown how to model a solar cell and a solar array. We have seen that if the VI curves of the cells within an array are known, then the net VI curve of the array can be obtained by summing the curves of the individual cells. However, in order to generate the curves of the individual cells, the parameters of the model shown in Fig. 6.1 have to be known.

These parameters are I_{pv}, I_0, n, a, V_{T}, R_{s} and R_{p}, and they may depend on the physical properties of the cell as well as on the current environmental parameters, i.e. illumination (G) and temperature (T). The parameters R_{s}, R_{p}, a and n are attributes of the PV cell and do not depend on G and T (or their dependence on these variables is negligible). Also, n is trivial, e.g. it is 3 for a 3-junction cell. The thermal voltage (V_{T}) depends only on the temperature and can easily be calculated using (6.4). Finally, I_{pv} and I_0 depend on the cell properties as well as on both of the environmental variables.

A possible method to compute these parameters was introduced in [35], which has the following benefits:

- Most of its input values that are related to cell attributes are usually provided in the datasheet of PV cells.
- It guarantees that the remarkable points of the mathematically generated VI curve will coincide with those of the real (measured) VI curve.
- It considers the temperature dependence of both the output voltage and the output current.

Due to these benefits, the method was employed with the SaS V2. Next, the steps of the method that yield the model parameters will be briefly introduced. We start by providing input parameters to the model, which are the following:

- The desired environmental variables: G, T.
- The remarkable points of the VI curve of the cell we want to model at arbitrary known environmental conditions G_0 and T_0. These are:
  - I_{sc0}: short-circuit current at G_0, T_0,
  - V_{oc0}: open-circuit voltage at G_0, T_0,
  - I_{MPP0}: maximum power current at G_0, T_0,
  - V_{MPP0}: maximum power voltage at G_0, T_0.
- G_0 and T_0.
- K_{I}: temperature coefficient of the short-circuit current.
- K_{V}: temperature coefficient of the open-circuit voltage.
- a: diode ideality factor.
- R_{s}: series output resistance of the cell.

The values of G_0, T_0, I_{sc0}, V_{oc0}, I_{MPP0}, V_{MPP0}, K_{I} and K_{V} are provided in the solar cell's datasheet or otherwise they have to be obtained by measurement. For the MOVE-II PV cells, they were listed in Table 4.2.[10] The values of a and R_{s} can usually not be found in the datasheet. They effectively have to be guessed in order to adjust the model later.

The method will output the missing model parameters: R_{p}, I_{pv} and I_0. Then, considering that n and V_{T} are found trivially, all parameters of the equation (6.7) are known and the VI curve can be generated.

The photovoltaic current is calculated as

$$I_{\mathrm{pv}} = [I_{\mathrm{sc0}} + K_{\mathrm{I}}(T - T_0)] \cdot \frac{G}{G_0}. \tag{6.8}$$

This equation expresses that the photovoltaic current equals to the short-circuit current. This is a very good approximation, since if a cell is shorted, then the diode and R_{p} in its equivalent circuit are effectively short circuited, because R_{s} is usually very small. Consequently, nearly all of the photovoltaic current will flow out of the cell. Equation (6.8) also expresses the linear dependence of the short-circuit current on G.

[10] For example, I_{sc0} is listed as $I_{\mathrm{sc,max}}$ in Table 4.2, because for the data provided there, the test conditions are such that $G_0 = G_{\mathrm{max}} = 1367\ \mathrm{W/m^2}$.

The diode reverse saturation current is found as

$$I_0 = \frac{I_{\mathrm{sc0}} + K_{\mathrm{I}}(T - T_0)}{e^{\frac{V_{\mathrm{oc0}}+K_{\mathrm{V}}(T-T_0)}{anV_{\mathrm{T}}}} - 1}. \tag{6.9}$$

Finally, the parallel output resistance is obtained through

$$R_{\mathrm{p}} = \frac{V_{\mathrm{MPP0}} + I_{\mathrm{MPP0}}R_{\mathrm{s}}}{I_{\mathrm{sc0}} - I_{\mathrm{MPP0}} + I_{00}\left(1 - e^{\frac{V_{\mathrm{MPP0}}+I_{\mathrm{MPP0}}R_{\mathrm{s}}}{anV_{\mathrm{T0}}}}\right)}, \tag{6.10}$$

where V_{T0} is the thermal voltage at temperature T_0, i.e.

$$V_{\mathrm{T0}} = \frac{kT_0}{q}, \tag{6.11}$$

and I_{00} is given by

$$I_{00} = \frac{I_{\mathrm{sc0}}}{e^{\frac{V_{\mathrm{oc0}}}{anV_{\mathrm{T0}}}} - 1}. \tag{6.12}$$

Equations (6.8), (6.9) and (6.10) warranty that if $G = G_0$ and $T = T_0$ then the computed (or mathematical) curve will cross the remarkable points of the measured (or experimental) curve. This means that the mathematical short-circuit current will equal I_{sc0}, the mathematical open-circuit voltage will equal V_{oc0} and that the computed curve will cross the $(V_{\mathrm{MPP0}}, I_{\mathrm{MPP0}})$ point. However, it is not automatically ensured that the point of the computed curve which crosses $(V_{\mathrm{MPP0}}, I_{\mathrm{MPP0}})$ is also the actual MPP of the computed curve. Putting it another way, $V_{\mathrm{MPP,m}} = V_{\mathrm{MPP0}}$ and $I_{\mathrm{MPP,m}} = I_{\mathrm{MPP0}}$ are not guaranteed, where $V_{\mathrm{MPP,m}}$ and $I_{\mathrm{MPP,m}}$ are the mathematical maximum power voltage and current.

In order to fulfill this requirement too, the two remaining parameters, namely R_{s} and a have to be selected properly. Usually, tweaking these two parameters by hand until the MPP of the mathematical and the experimental curve coincide is feasible, since we know that in most practical cases $1 \leq a \leq 2$ and $R_{\mathrm{s}} < 1\ \Omega$. Alternatively, an optimization algorithm can be used to find a and R_{s}. Moreover, if the entire experimental curve is available (not just its remarkable points), for example from the datasheet or from own measurements, then it can help in finding the appropriate values of a and R_{s}. In this case, the two parameters are tweaked so that the mathematical and the experimental curve match as closely as possible. Of course, a good match between the curves also grants that their MPPs will coincide (or be very close to each other).

For the MOVE-II solar cells, the determination of a and R_{s} was done by hand-tweaking. Experimental VI curves for both cells were available to aid this process: one for the 3J cells from its datasheet [20], and one for the 4J cells from previous measurements [21]. A second experimental curve for the 4J cells for a higher temperature was used to verify the results.

Table 6.1 summarizes all the cell attributes that are relevant for the VI curve computation, including the values of a and R_{s} that were found to give the best match between the mathematical and the experimental curves. The table gives a reminder about the source of the numbers.

It should be noted that some of the listed parameters degrade with time due to radiation in space. In Table 6.1, beginning-of-life (BoL) values are given. Since the MOVE-II mission is relatively short (about half a year), it can be assumed that the cell parameters will remain close to their BoL values during the whole time. If the simulation of aged cells is desired, the corresponding degraded values have to be used to readjust the tweaked parameters and to generate the VI curves. Some solar cell datasheets, including [20], contain the parameter values for a number of different degradation levels.

Table 6.1 – Summary of solar cell parameters for VI curve generation

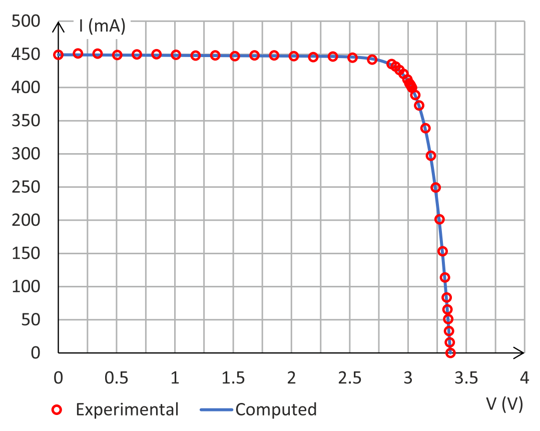
Parameter	Source	3J	4J
G_0	datasheet/measurement	$1367\ \mathrm{W/m^2}$	$1367\ \mathrm{W/m^2}$
T_0	datasheet/measurement	$28\ \mathrm{°C}$	$25\ \mathrm{°C}$
I_{sc0}	datasheet/measurement	$260.1\ \mathrm{mA}$	$449.4\ \mathrm{mA}$
V_{oc0}	datasheet/measurement	$2700\ \mathrm{mV}$	$3363.8\ \mathrm{mV}$
I_{MPP0}	datasheet/measurement	$252.2\ \mathrm{mA}$	$429\ \mathrm{mA}$
V_{MPP0}	datasheet/measurement	$2411\ \mathrm{mV}$	$2900.5\ \mathrm{mV}$
K_{I}	datasheet/measurement	$0.18\ \mathrm{mA/°C}$	$0.49\ \mathrm{mA/°C}$
K_{V}	datasheet/measurement	$-6.2\ \mathrm{mV/°C}$	$-7.89\ \mathrm{mV/°C}$
n	trivial	3	4
a	tweaked	1.02	1.275
R_{s}	tweaked	$0.02\ \Omega$	$0.09\ \Omega$
R_{p}	computed with (6.10)	$3894\ \Omega$	$1041\ \Omega$

The quality of the solar cell model was evaluated using the three available experimental curves. The match between the mathematical and the experimental curves can be assessed in the same way as what was described in Section 5.7.3 to express the accuracy of the SAS. This time the reference is the measured (experimental) curve and hence, the relative current error ($\epsilon_{I,\mathrm{rel}}$) and the relative voltage error ($\epsilon_{V,\mathrm{rel}}$) are calculated as follows:

$$\begin{aligned}\epsilon_{\mathrm{V,rel}}(I_0) &= \frac{|V_{\mathrm{e}}(I_0) - V_{\mathrm{c}}(I_0)|}{V_{\mathrm{e}}(I_0)},\\ \epsilon_{\mathrm{I,rel}}(V_0) &= \frac{|I_{\mathrm{e}}(V_0) - I_{\mathrm{c}}(V_0)|}{I_{\mathrm{e}}(V_0)},\end{aligned} \tag{6.13}$$

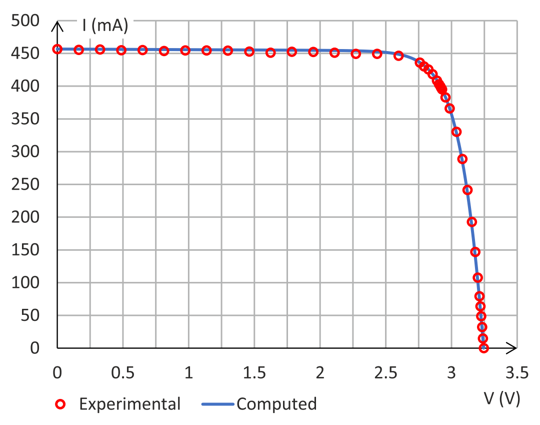
where V_{e} and I_{e} are the voltage and the current of the experimental curve and V_{c} and I_{c} are the voltage and the current of the computed curve. Again, $\epsilon_{I,\mathrm{rel}}$ is considered from the short-circuit point to the MPP and $\epsilon_{V,\mathrm{rel}}$ is considered form the MPP to the open-circuit point.

In Fig. 6.6, three pairs of curves are shown: the three experimental curves and their computed counterparts for the same environmental variables. The relative voltage and current errors are given under the diagrams.



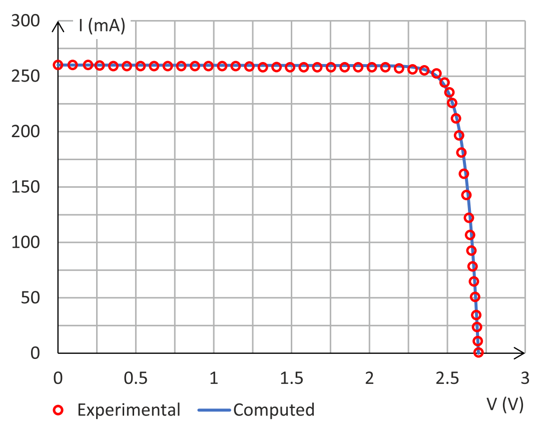
Cell: 4J

$T = 25\ ^{\circ}\mathrm{C}$ $\epsilon_{I,\mathrm{rel,max}} = 0.5\%$
$G = 1367\ \mathrm{W/m^2}$ $\epsilon_{V,\mathrm{rel,max}} = 0.5\%$



Cell: 4J

$T = 40\ ^{\circ}\mathrm{C}$ $\epsilon_{I,\mathrm{rel,max}} = 1\%$
$G = 1367\ \mathrm{W/m^2}$ $\epsilon_{V,\mathrm{rel,max}} = 0.12\%$



Cell: 3J

$T = 28\ ^{\circ}\mathrm{C}$ $\epsilon_{I,\mathrm{rel,max}} = 0.94\%$
$G = 1367\ \mathrm{W/m^2}$ $\epsilon_{V,\mathrm{rel,max}} = 0.98\%$

Figure 6.6 – Comparison of experimental and computed VI curves

6.2.4 Computation of VI Curves in the SaS V2

Using the methods described in the previous sections, the VI curves of the MOVE-II solar arrays can be generated under any environmental conditions. The information necessary to do this are the desired environmental variables (G, T) and the solar cell properties from Table 6.1. The computation process for a single solar array is summarized below.

1. For each cell in the array:
    a. Temperature and illumination-dependent model parameters (I_{pv}, I_0) are computed based on (6.8) and (6.9).
    b. The VI curve of the cell is generated based on (6.7).
2. The VI curve of the array is generated by summation of VI curves as described in Section 6.2.2.

The generation of the arrays' VI curves is the task of the high-level controller of the SaS V2, and it has to be done for all three arrays every time a new set of environmental variables is received from the HiL PC, that is, every $200\ \mathrm{ms}$.

The computation of the VI curve of a single cell could be done, for example, by first determining the V_{oc} of the curve and then going through the voltage axis from 0 to V_{oc} with a fixed resolution. For every voltage value, the current would be computed by numerically solving (6.7). However, this method would require a very high resolution (many steps along the voltage axis) to get a dense curve in the voltage-source region. At the same time, this would result in unnecessarily many points computed in the current-source region.

To avoid this problem, an alternative method was employed, which ensures that the computed points along the curve are spaced almost uniformly in every VI curve region. The method was implemented as a recursive algorithm and works as follows. The desired density, i.e. the maximum distance of neighboring points along the VI curve is defined as a percentage of the open-circuit voltage and the short-circuit current of the curve. The algorithm will insert new points into the curve until there are no neighboring points left whose distance is greater than the defined threshold, either in voltage or in current. First, the open-circuit and the short-circuit points are computed. This is done by solving (6.7) with the substitution $I = 0$ to get V_{oc} and then with the substitution $V = 0$ to get I_{sc}. If the two points are further away from each other than the defined threshold, then a new point is inserted at $V_{\mathrm{oc}}/2$. Then the same check is repeated for any two neighboring points and insertions are continued until the desired density condition is met. The benefit of this method is that it generates just as many points of the VI curve as necessary, hence it is faster and the resulting set of (V, I) points is more compact. This also accelerates the summing of the curves later, since a smaller set of points has to be worked from.

Even this method is not fast enough for real-time computation though. It has been found that around 250 points per curve have to be computed to generate smooth VI curves, and for each point, an implicit nonlinear equation has to be solved, which in turn involves several evaluations of the equation to solve. The method described above was implemented in Matlab and tested on a personal computer with an Intel Core i3-5010U processor running at $2.1\ \mathrm{GHz}$. This resulted in an average computational time of approximately $190\ \mathrm{ms}$ per curve. It is therefore unlikely that the ARM Cortex-A8 processor in the BeagleBone Black could perform this task several times within $200\ \mathrm{ms}$, even if the same method was implemented in a compiled language such as C.

As a solution, the VI curves of the cells are pre-computed for all possible combinations of temperature and illumination within a certain range and with a certain resolution. The chosen ranges are $G = 0 \dots 1367\ \mathrm{W/m^2}$ and $T = -20\ \mathrm{°C} \dots 80\ \mathrm{°C}$ and the resolutions are $1\ \mathrm{W/m^2}$ and $1\ \mathrm{°C}$. The total number of resulting VI curves is therefore

$$1368\ \text{illuminations}\ \times 101\ \text{temperatures} \times 2\ \text{cell types} = 276336\ \text{curves}. \quad (6.14)$$

The curves were generated with a Matlab script using the method described above and stored in separate files each. For solving (6.7), the built-in *fzero* Matlab function was utilized, which uses a combination of bisection, secant, and inverse quadratic interpolation methods to find the root of a nonlinear function. The resulting data size is $546\ \mathrm{Mbyte}$, but $1.07\ \mathrm{Gbyte}$ disk space is required due to file system overhead. This still fits easily onto the $16\ \mathrm{Gbyte}$ micro SD card used in the SAS. When a new set of environmental variables is received from the HiL PC, the high-level controller just reads in the relevant VI curve files, and it can proceed to the next step instantly.

After the VI curves of the cells have been loaded, they have to be summed according to the logic described in Section 6.2.2 and based on the solar array topologies shown in Fig. 4.3. The VI curves are stored as voltage-current pairs and it can generally not be expected that two VI curves contain the currents for the same voltage values (or vice versa). Hence, the curves that should be summed, effectively need to be merged. During a merge along the current axis for example, all current values that appear in any of the source curves are considered. If a current appears in one curve, but not in the other one, the missing point is determined by linear interpolation.

The VI curves are only stored from their short-circuit to open-circuit points, but in general, they may be needed beyond these points too, since the MOVE-II solar arrays are only partially diode-protected (see Fig. 4.3). In this case the missing points are found by linear extrapolation based on the slope of the curves at their short-circuit and open-circuit points.

The merging algorithm is fast enough to run on the high-level controller in real time. The core of the algorithm consists of two functions: one of them sums two curves along the voltage axis, while the other one sums them along the current axis. An additional parameter in the functions determines if protection diodes are present. The forward voltage drop (V_F) across the protection diodes is not considered by the functions, but this is desirable, since MOVE-II uses active ideal diodes of the type SM74611 with close to zero forward voltage.

In general, the VI curves of all three solar arrays have to be generated by successive merges of the curves of individual cells and sub-arrays. Yet due to the specific array topologies of MOVE-II, some shortcuts are possible.

The two solar arrays composed of the 4J cells on the Flappanels (SA1 and SA2) are identical in structure and also in illumination and temperature, since they are in the same plane. Therefore, once the curve of SA1 is determined, it can be reused for SA2. Also, the individual cells within these arrays will always have the same illumination and temperature at any point in time. Since four cells are in series in one array, the array VI curve can be computed simply by scaling up the VI curve of a single 4J cell along the voltage axis by a factor of 4. In practice, this is done by multiplying the voltage of every point of the curve by 4. In fact, the VI curve files describing the 4J cells already include this scaling (they effectively describe four 4J cells in series), so that it does not have to be done in runtime.

For SA3, no such shortcuts are possible, since all the cells in this array may have different illumination and temperature at the same time. Hence, the array VI curves are determined through successive merges of curves of cells and sub-arrays with the method shown in Fig. 6.2, but with four parallel strings instead of two. During the process, the presence of the blocking diodes and the absence of bypass diodes is considered.

When the net VI curves of the arrays are computed, they are sets of voltage-current pairs stored as floating-point numbers. As the final step, the high-level controller converts these lists to lookup tables, writes them into the PRU memory according to the logic explained in Section 5.6.4.3 and updates the LUT pointers.

The VI curves will then be automatically simulated on the channel outputs. Channel 1 simulates SA1, channel 2 simulates SA2 and channel 3 simulates SA3.

6.3 Channel Power Determination

Determining the available and the accepted power on the three output channels is an easy task compared with the VI curve computation. Once the net curve that should be simulated on a specific channel has been generated, the program goes through each point in the curve and finds the one with the largest V-I product, i.e. power. This is the MPP of the curve and its power equals to the maximum available power on that channel.

The measurement of the accepted power, i.e. the actual output power of the channels is done by the low-level controller, since it continuously needs to measure the output voltage and current on each channel to fulfill its own task. It writes the measured quantities to a table at a specific address in the PRU memory that the high-level controller can access. After reading these values, the high-level controller computes their product and converts it to natural units (watts).

The low-level controller measures the output quantities at a frequency of $5.31\ \mathrm{kHz}$, but the high-level controller only reads them and computes their product at a frequency of $1\ \mathrm{kHz}$. Yet again, the power values are sent to the HiL PC only every $200\ \mathrm{ms}$ (at $5\ \mathrm{Hz}$, which is the same frequency at which the HiL PC sends data to the SAS). Hence, the approximately 200 power values collected during a communication cycle are averaged and a total of 6 values are sent to the simulation PC. These are the available power on each channel, and the average accepted power on each channel since the reception of the last packet from the simulation PC.

6.4 Communication Interface

The space environment simulation on the HiL PC runs in Simulink. The data between the Simulink model and the SaS V2 is exchanged via Ethernet using the WebSocket protocol, which is a full-duplex communication scheme in the application network layer, based on the transmission control protocol (TCP). Most importantly, the WebSocket protocol lets the server (the HiL PC) send data to the client (the SAS) without a previous request from the client. Data between Simulink and the SaS V2 is transferred through an intermediate WebSocket server. The block diagram in Fig. 6.7 illustrates the communication scheme.

The WebSocket server runs on the HiL PC and is implemented in Python using the *Websockets* library. On the SAS side, a WebSocket client is implemented in C++ using the *Websocketpp* library. The compiled WebSocket server is called from the main C program of the SAS and is run on a separate thread, as mentioned earlier. The program on the SAS knows the IP address of the simulation PC and connects to the WebSocket server when it starts.

When the simulation is launched, the WebSocket server on the HiL PC starts sending data to the client, i.e. the SAS. A single packet of data contains 9 illumination levels and 9 temperatures: one for each of the 8 Sidepanel (3J) cells, and one for the Toppanel (the 4J cells). Additionally, a timestamp is included in the message, which identifies the packet.

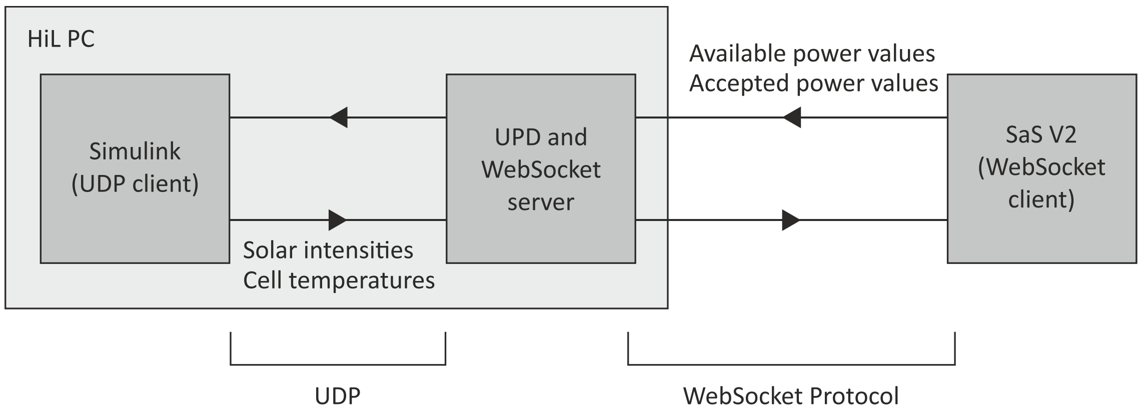


Figure 6.7 – Communication scheme between Simulink and the SaS V2

As soon as the SAS receives a packet, it starts generating the new VI curves, and when that is completed, it determines the new available power on each channel. Then, it loads the updated lookup tables into the PRU memory and immediately starts collecting data about the accepted power on the channels. When the next packet is received, it computes the average accepted power for each channel and sends this data, along with the available power values, to the WebSocket server. The timestamp that identifies the packet previously sent by the HiL PC is included in the answer of the SAS. When the packet is sent, the SAS starts over by processing the next input values. Since the SAS sends a packet to the simulation PC every time it receives a packet from it (except for the very first received packet), the frequency of outgoing data is the same as that of incoming data, that is, $5\ \mathrm{Hz}$. The exact structure of the packets sent by both parties can be found in Appendix C.

Since the input of the SAS (illumination and temperature values) is generated in Simulink, and also the SAS outputs (power values) are needed by the Simulink model for further processing and visualization, a communication link must exist between the Simulink model and the WebSocket server. This is implemented using user datagram protocol (UDP) based messages between the two programs. The UDP packets are transferred within the HiL PC rather than over the network. On the Simulink side, a *UDP Send* and a *UDP Receive* block is used to implement a UDP client. On the WebSocket server's side, the *Socket* Python library is used to implement a UDP server.

The communication scheme described above is the same as what is used with the ADCS Panel Emulator. The SAS communication was implemented based on program code originally written for the Panel Emulator, which is a work of Jonis Kiesbye and Jan van Brügge.

The Simulink model does some post-processing to the data it receives from the SAS. First, it sums the power values of the separate channels to get the total available and the total accepted power. The available and accepted power values are plotted using an *Oscilloscope* Simulink block. This is done for the separate channels and for the summed values too, so that a total of 8 plots are drawn during the simulation: the available and accepted power for channel 1, 2 and 3, and the total available and total accepted power. Example plots can be seen in the next section (Fig. 6.8 and 6.9).

The instantaneous maximum power point tracking (MPPT) efficiency ($\eta_{\mathrm{MPPT,i}}$) shows what portion of the available power is being accepted at a given time t. It is expressed as

$$\eta_{\mathrm{MPPT,i}}(t) = \frac{P_{\mathrm{acc}}(t)}{P_{\mathrm{avail}}(t)}, \tag{6.15}$$

where $P_{\mathrm{acc}}(t)$ and $P_{\mathrm{avail}}(t)$ are the instantaneous accepted and available power values at time t, respectively. The accumulated MPPT efficiency ($\eta_{\mathrm{MPPT,a}}$) shows how much of the total available energy was accepted in a time frame defined by t_1 and t_2. It is expressed as

$$\eta_{\mathrm{MPPT,a}}(t_1, t_2) = \frac{E_{\mathrm{acc}}(t_1, t_2)}{E_{\mathrm{avail}}(t_1, t_2)}, \tag{6.16}$$

where $E_{\mathrm{acc}}(t_1, t_2)$ is the total accepted energy and $E_{\mathrm{avail}}(t_1, t_2)$ is the total available energy between time instances t_1 and t_2. They can be calculated as integrals of the corresponding power values over time:

$$\begin{aligned} E_{\mathrm{acc}}(t_1, t_2) &= \int_{t_1}^{t_2} P_{\mathrm{acc}}(\tau)\,\mathrm{d}\tau, \\ E_{\mathrm{avail}}(t_1, t_2) &= \int_{t_1}^{t_2} P_{\mathrm{avail}}(\tau)\,\mathrm{d}\tau. \end{aligned} \tag{6.17}$$

The Simulink model uses an integrator block to keep track of the accumulated MPPT efficiency since the start of the simulation until the current time t, i.e. of $\eta_{\mathrm{MPPT,a}}(0, t)$. The efficiency is calculated based on the total available and accepted power values.

6.5 Verification of Functionality

With all the hardware and software components in place, the design of SaS V2 is finished and the device can be inserted into the extended HiL environment. In this section it will be shown that the HiL system with the SaS V2 in it is functional. For the following tests, the complete extended HiL setup depicted in Fig. 5.2 with the MOVE-II engineering model was used.

Before starting the first simulation, it was checked whether the SAS performs its tasks fast enough to keep up with the simulation pace. Effectively, this means that its maximum achievable VI curve update frequency ($f_{\mathrm{u,VI}}$) was measured. This was done by asserting one of the GPIO pins of the BeagleBone Black during VI curve computation and de-asserting it again afterwards, when the SAS is idle waiting for the next input and only doing background tasks such as measuring the output power and updating the console screen. The duration of the computation was measured with an oscilloscope several times and was always found to be under $10\ \mathrm{ms}$, which corresponds to an achievable VI curve update frequency of at least $100\ \mathrm{Hz}$. Hence, the SAS meets the corresponding requirement listed in Table 5.1.

Next, the output of the SAS from a typical simulation will be shown. The simulation was run using the extended HiL setup, which means that the behavior of the ADCS also affects the results. The graphs in Fig. 6.8 show the available and accepted power levels over time. The first three graphs correspond to the three output channels, and hence to the solar arrays SA1, SA2 and SA3, in this order. The fourth graph shows the total power values, i.e. the sum of the first three graphs. The blue curves represent the available power, while the red ones represent the accepted power.

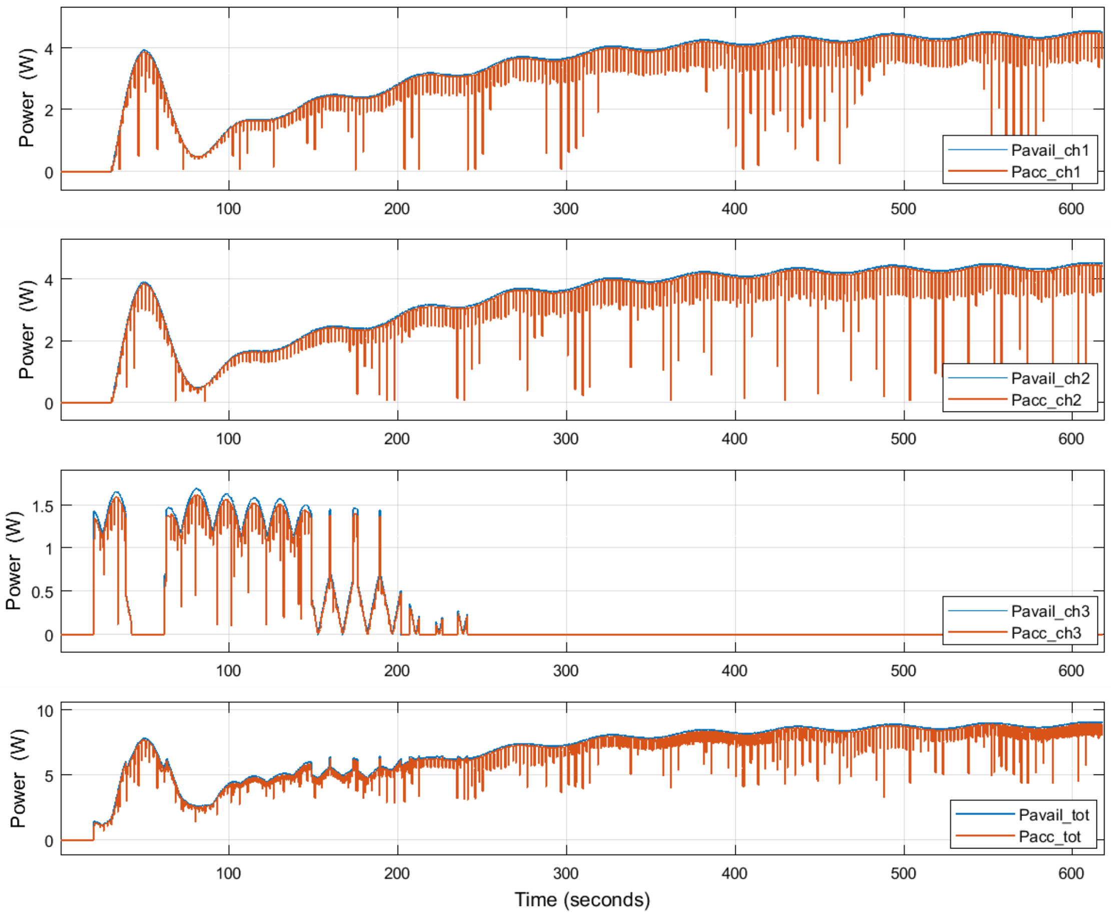


Figure 6.8 – Output of the SaS V2 during a Sunpointing simulation

The effect of the ADCS Sunpointing mechanism can be observed on the graphs. In Sunpointing mode, the ADCS rotates the satellite so that the Flappanels are pointed towards the Sun for maximum solar power input. At the beginning of the simulation, the satellite is at an angle where the Flappanels are not illuminated and channels 1 and 2 do not output any power. At the same time, some of the Sidepanels receive sunlight and channel 3 delivers power. As the satellite rotates, the Flappanels become illuminated and the power output on channels 1 and 2 increases, while the power on channel 3 decreases, since the Sidepanels now move into shadow. After around 80 seconds, Sunpointing starts to take effect and the power on channels 1 and 2 increases again as the satellite points the Flappanels to the Sun. After around 400 seconds, Sunpointing is completed. Channels 1 and 2 reach their maximum output power and the power on channel 3 drops to zero. In this state, the sunrays are perpendicular to the Flappanels and parallel to the Sidepanels. Small periodic variations in the power output can still be observed due to the precessional rotation of the satellite. In the graphs, the red curves closely follow the blue ones. To observe the relationship of the blue and red curves, i.e. the available and accepted power levels, Fig. 6.9 shows a closeup of the simulation output from $t = 40\,\mathrm{s}$ to $t = 100\,\mathrm{s}$ (where t is the simulation time).

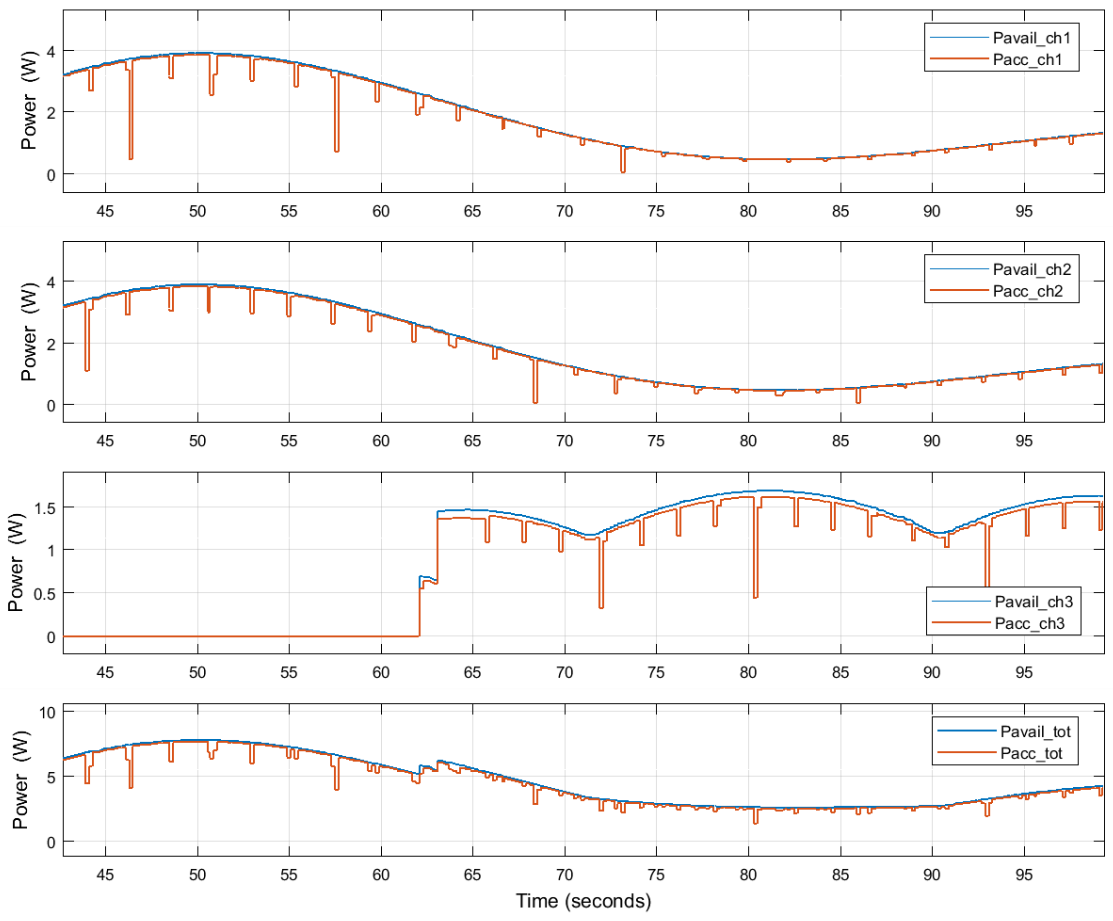


Figure 6.9 – Closeup on the output of the SaS V2 during a Sunpointing simulation

On channels 1 and 2, the accepted power almost equals to the available power, which means that $\eta_{\mathrm{MPPT,i}}$ is close to 100%. Approximately every 2.5 seconds the accepted power drops momentarily. This is the result of the MPPT mechanism of the EPS, in which the solar arrays are open-circuited for around $100\,\mathrm{ms}$ every $2.5\,\mathrm{s}$. During these periods, the accepted power is zero. The red curves do not drop completely to zero at these points, because the open-circuit is maintained only for $100\,\mathrm{ms}$ and every data point in the SAS output corresponds to an average computed over $200\,\mathrm{ms}$.

On channel 3, the same effect can be observed. Here, the MPP tracking efficiency is lower, which is apparent from the fact that the gap between the red and the blue curves is larger. However, $\eta_{\mathrm{MPPT,i}}$ can still be estimated to be over 90% at all times except for the MPPT periods.

From the results shown above we can conclude that the SaS V2 works effectively in the HiL setup. Based on the input received from the HiL PC, it simulates the VI curves of the MOVE-II solar arrays and delivers power to the satellite. The inputs generated by the HiL PC are in turn influenced by the ADCS-related part of the HiL setup. Finally, the SaS V2 outputs data about the available and accepted power levels, which is logged by the HiL PC and can be used to estimate the MPP tracking efficiency of the satellite's EPS.

7 Power Budget Analysis

7.1 Introduction

In this chapter, the results of two power budget tests will be presented. The tests were performed using the extended HiL setup with the SaS V2 and the MOVE-II engineering model (EM) as shown in Fig. 5.2.

The selected test cases represent the two possible operational modes of MOVE-II, which are safe mode and nominal mode. The safe mode is a power-saving mode in which several subsystems of the satellite are switched off and many of the processes running on its central computer are stopped. This includes any subsystem and process that is not crucial for the survival of the spacecraft and for maintaining the radio link with the ground station. Since the ADCS is also turned off in this mode, the satellite may tumble, i.e. rotate in an uncontrolled manner.

In nominal mode, the satellite operates normally and all subsystems are active, including the Payload, so that MOVE-II can fulfill its scientific mission. The power consumption in this mode is higher than in safe mode, but since the ADCS performs Sunpointing, the incoming power is also expected to be higher.

These two modes are representative to the entire power budget of the satellite. However, the results that follow should not be considered as a final power budget verification, as that would require testing many other mission scenarios as well. With these tests, three goals are pursued: to perform a first HiL-based power budget verification, to deliver examples for more complete investigations in the future, and to demonstrate that the SaS V2 is indeed a suitable tool for power budget verification.

Some limitations about the validity of the results should be considered. Firstly, the shadow dropped by the Flappanels onto the Sidepanel cells is modeled by the simplified ray-tracing method described in Section 5.3.2, and hence it is not entirely accurate. Secondly, the thermal model of the solar cells, which is developed by the ADCS team, is still a work in progress at the time of writing. Therefore, it could not be utilized in the tests and the HiL PC was set up to send a constant temperature of $300\,\mathrm{K}$ (about $27\,°\mathrm{C}$) to the SAS. Both of these limitations may be eliminated in the future.

In Sections 7.3 and 7.4, the test results will be presented in two graphs each. The first one shows the temporal development of the battery charge level and the trend of this graph shows whether the power budget is positive or negative. It should be noted that the battery level is an estimate made by the CDH subsystem based on the battery terminal voltage (under load). The second graph shows the total available and total accepted power from the SAS on the same time axis as used for the battery level. The amount of accepted power should correlate well with increasing and decreasing segments of the battery level diagram. While the power values are logged by the HiL system, the battery level is logged by the satellite ground station, based on the beacon messages that the EM sends out over the radio link every minute. The two datasets were merged manually as an additional step after the simulation. As the sample period of the battery level is one minute, the SAS power levels were also sampled down to this value. Since the original sample period of the power levels is $200\,\mathrm{ms}$, this was done by averaging every 300 neighboring data points, which corresponds to a time frame of $200\,\mathrm{ms} \cdot 300 = 60\,\mathrm{s}$. The simulation results can be found in tabulated format in the project's Git repository (see Appendix E).

7.2 Overview of Test Cases

The properties and initial conditions of the test cases are presented in Table 7.1.

Table 7.1 – Overview of power budget verification test cases

Attribute	Safe mode test	Nominal mode test
Satellite mode	Safe mode	Nominal mode
ADCS mode	Off	Sunpointing
Initial Julian date	2458139.515035 days	2458139.493035 days
Initial orbital position	At beginning of sun phase	At beginning of eclipse
Initial angular velocity	$0.1\,\mathrm{rad}/s$	$0.1\,\mathrm{rad}/s$
Initial battery level	60%	45%
Duration	4 orbits	4 orbits

Every orbit consists of a sun phase, when the satellite is illuminated by the Sun, and an eclipse, when it is shaded by the Earth. During eclipse, there is no incoming energy, and for a positive power budget, the satellite's battery has to be charged to a sufficient level during every sun phase to survive the next eclipse.

The initial position of the satellite in its orbit is determined by the date and time that corresponds to the start of the simulation. This is defined by the initial Julian date, which is given in Table 7.1 for both test cases. The Julian date defines time as the number of days elapsed since 12:00 1 Jan. 4713 BC. Units smaller than a day (hours, seconds, etc.) are expressed in the fractional part of the Julian date.

The duration of the sun phases, and the eclipses is determined by the orbital parameters, which were selected according to the actual future orbit of the MOVE-II flight model. These parameters are the same for the two test cases and are given in Appendix D for reference.

The orbital parameters define a $575\,\mathrm{km}$ high sun-synchronous low-earth orbit. The duration of one orbit is approximately $5770\,s$, which is 1 hour, 36 minutes and 10 seconds. Within this time, about 33 minutes are spent in eclipse and the remaining 63 minutes make up the sun phase. Both tests were run for 4 orbits which corresponds to a simulation time of 6 hours, 24 minutes and 40 seconds. This is long enough to recognize certain patterns in the simulation output and a positive or negative trend of the battery charge level.

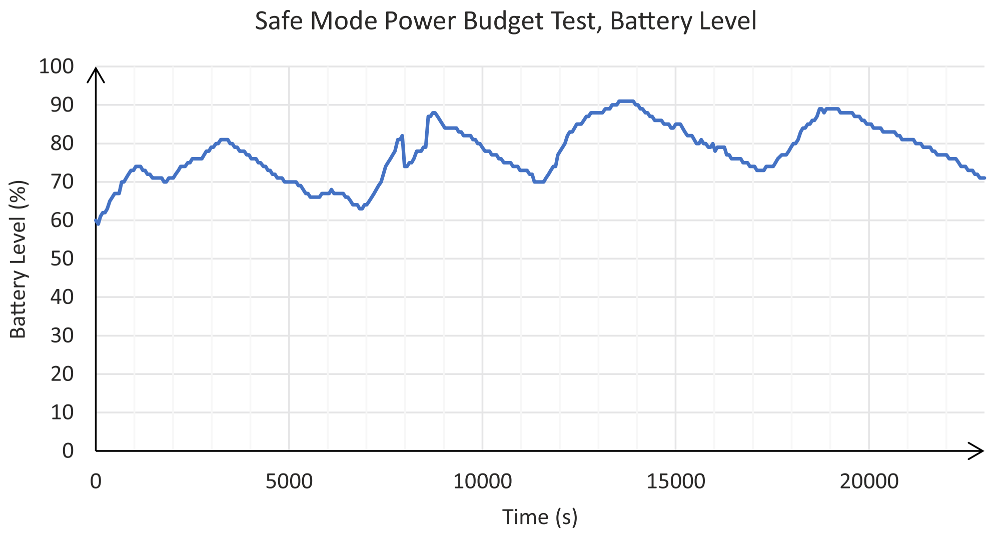
Both test cases start with a tumbling satellite. The initial angular velocity is $0.06\,\mathrm{rad}/s$ around all three axes, which corresponds to the net angular velocity shown in Table 7.1, since $0.06\,\mathrm{rad}/s \cdot \sqrt{3} \cong 0.1\,\mathrm{rad}/s$. In the safe mode test, tumbling is maintained during the whole simulation due to the lack of ADCS action. In the nominal mode test, the ADCS first stops the tumbling motion, and then performs Sunpointing.

As shown in Table 7.1, the battery level was brought to around half of the full capacity before launching the simulations to make room both for battery charge and discharge.

Next, the simulation results will be shown. The nominal mode is expected to be power positive due to Sunpointing. However, previous software-based attempts estimated the safe mode to be power negative, hence the safe mode test is of special interest [2].

7.3 Safe Mode Test

The results of the safe mode test are shown in Fig. 7.1.



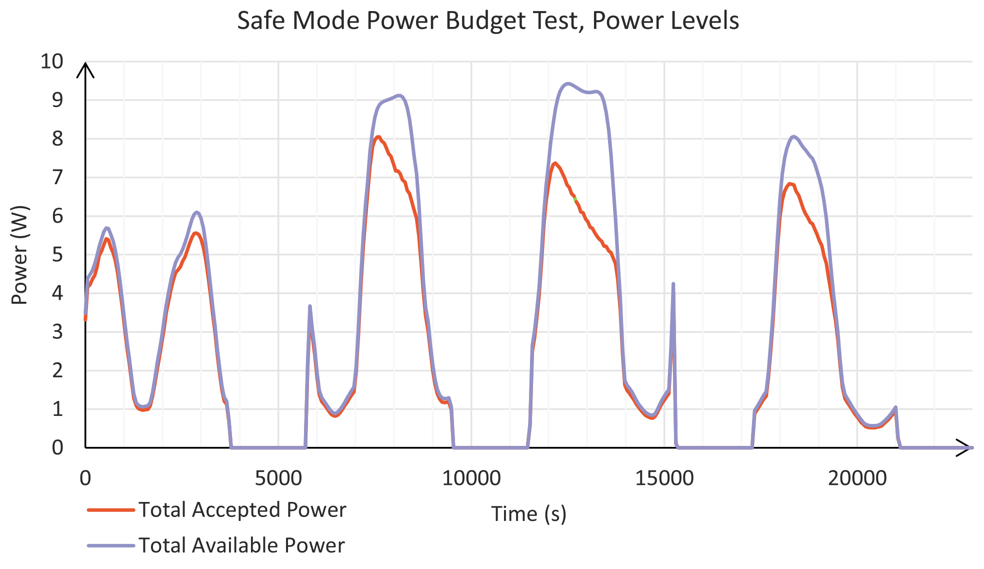


Figure 7.1 – Results of safe mode test

The periodicity of sun phases and eclipses can be observed well on the bottom graph. In eclipses, no power is delivered to the satellite and the battery level decreases. In sun phases, there is power delivery, but the available power changes within a large range between approximately 1 and $9\,\mathrm{W}$ during a single phase due to tumbling. Since the angular velocity is $0.1\,\mathrm{rad/s}$, the power levels change more rapidly than seen in Fig. 7.1. However, due to a sampling period of 1 minute, this kind of detail is lost, while making the graph easier to read. The gaps in power delivery caused by the MPPT mechanism are not visible either for the same reason. The correlation between the accepted power level and the shape of the battery level curve is apparent when the two graphs are compared.

At the beginning of the simulation, the accepted power is close to the available power and, based on the simulation data, the accumulated MPPT efficiency during the first sun phase is 93.6%. As the battery level reaches around 80% (at about $t = 7500\ s$), the MPPT efficiency drops significantly. This is because at this point the satellite does not need all the available power and the EPS switches to EoC mode where only the power necessary to serve the subsystems and finish the battery charging cycle is accepted.

The most important conclusion about the test case is the slightly positive trend of the battery level. It starts at 60% and reaches 70% at the end of the simulation. According to this test, the power budget of the satellite in safe mode is positive under the conditions given in Table 7.1. However, the battery level only increases with around 5% on average during an entire orbit (from the beginning of one sun phase to the beginning of the next one), which is a relatively small margin.

7.4 Nominal Mode Test

The results of the nominal mode test are shown in Fig. 7.2.

During this test, the Sunpointing was active. The positive effect this has on the available power is clearly visible when the bottom graphs of the two test cases are compared. After the initial eclipse, the satellite starts detumbling and approximately 32 minutes after the start of the sun phase ($t \approx 3500\ s$), it completes Sunpointing. At this point, the total available power is $9.9\ \mathrm{W}$. This agrees with the data given about the 4J cells in Table 4.2. Since the peak output power of a 4J cell is $1.244\ \mathrm{W}$, the maximum available power from SA1 and SA2, which contain a total of 8 cells, is $8 \cdot 1.244\ \mathrm{W} \cong 9.95\ \mathrm{W}$.

During the first sun phase the EPS is in MPPT mode and the accumulated MPPT efficiency for this period is 95.3%. MPPT continues in the second sun phase until approximately $t = 9600\ s$, when the battery level reaches about 80% and the EPS switches to EoC mode. In EoC mode, the MPPT efficiency rolls off as the satellite needs less and less power to top up the battery. Since the battery level decreases again in eclipse, MPPT mode is activated again at the beginning of every sun phase. It can be observed though that the timeframe in which the MPPT mode is active can get shorter with every sun phase.

Looking at the top graph, we can observe that the battery level increased from 45% to over 90% by the end of the simulation. Moreover, the range within which the battery level changes is much larger than it was in safe mode, because both the power collection and the power consumption of the satellite is higher. A positive trend in the battery level is clearly visible and we can conclude that according to this test, the satellite is power positive in nominal mode under the conditions given in Table 7.1. This time, the increase of battery charge during an orbit is around 15%, which is a relatively large margin. In Fig. 7.2, from the third orbit, this figure decreases as the battery level starts to saturate.

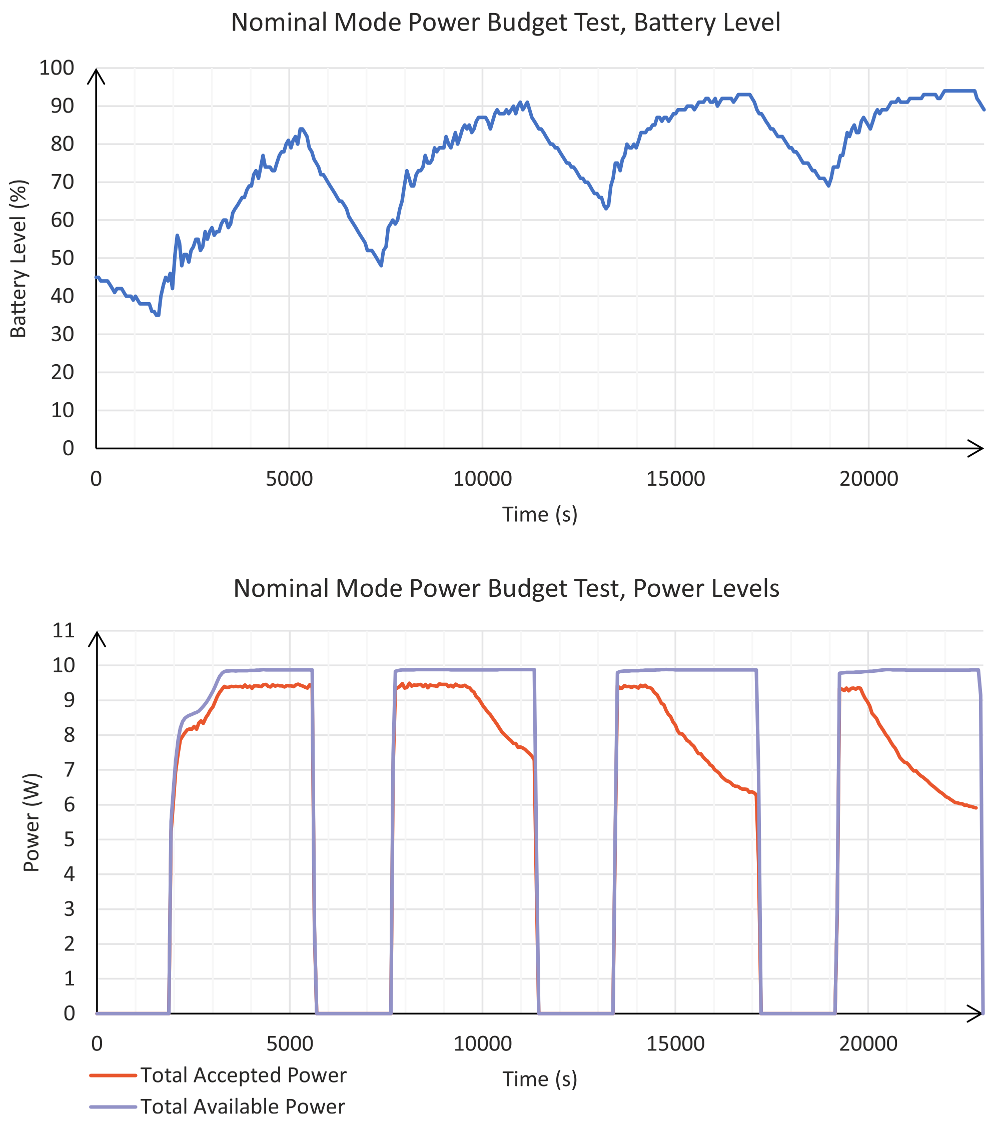


Figure 7.2 – Results of nominal mode test

8 Discussion

The power budget is a crucial aspect of every satellite. Running out of battery power carries the risk that the spacecraft will not be able to recover to its nominal mode and an otherwise promising mission may fail permanently. Therefore, it is important to have a reliable way to verify a power budget. For the MOVE-II satellite, previous power budget related estimates were based on software simulation, which inherently suffers from modelling inaccuracies. To eliminate these inaccuracies, tests based on the hardware-in-the-loop approach were desired, which necessitated the creation of a new solar array simulator for MOVE-II, the SaS V2.

The SaS V2 was designed to fit the requirements of MOVE-II and the existing HiL environment originally created for testing the satellite's ADCS subsystem. Compared with commercially available solar array simulators, the SaS V2 offers relatively little output power (just over $25\ \mathrm{W}$ in total), but can update its simulated VI curve quickly (around every $10\ \mathrm{ms}$) and has three output channels to substitute the three solar arrays of MOVE-II. The device is realized in a compact form-factor and its bill-of-material cost is a fraction of the price of a typical commercial SAS.

The SaS V2 is capable of simulating arbitrarily shaped VI curves within its maximum output voltage and current range. This is due to the fact that the simulated curves are stored as lookup tables in the device's memory and the contents of the tables may be changed freely. A linear output stage that can act either as a voltage or a current source ensures that the output remains stable at any operating point on the VI curve. Tests have verified that the device can simulate the programmed VI curves reliably and accurately, with a maximum deviation between the specified and measured curve of under 1%.

It has been demonstrated that the SaS V2 meets its design specifications, which were laid out based on the properties of MOVE-II and the HiL system. The SaS V2 can be used as a standalone device controlled from a terminal program running on a personal computer connected to the simulator via Ethernet or USB. However, the full potential of the SAS is utilized when integrated into the HiL system. Through the execution of two test cases, it was shown that the SaS V2 serves as the basis of a useful power budget verification tool and generates meaningful outputs that also help learning more about the behavior of the satellite's EPS.

The performed tests delivered power positive results for the two most important operational modes of the satellite. Moreover, the HiL approach provides much higher confidence in the test results than previous software-based estimation methods. It should be mentioned however, that the validity of the results is still limited to some extent, mainly due to the lack of a thermal model on the HiL PC for calculating the temperatures of the solar cells.

To generate the lookup tables that describe the VI curves of the MOVE-II solar cells under various environmental conditions, a simple but sufficiently accurate solar cell model, the single-diode model was used. It is supplemented by a method that computes the model parameters based on basic measurement results about the cells that are often provided in the datasheets of PV devices.

The use of the SaS V2 is not restricted to MOVE-II. It can also be used to test other devices with similar power requirements, if the software of the SAS is adopted accordingly. The output ratings of the SaS V2 may be a limiting factor here. Depending on the used cells and topology, solar arrays of other CubeSats might have higher output voltages and currents than the maximum specified output of the SaS V2 ($16\ \mathrm{V}$, $550\ \mathrm{mA}$).

The device is a first prototype and has some imperfections. Firstly, it exhibits a short but relatively large current overshoot when suddenly loaded by the satellite during MPPT. Secondly, due to the topology of the output stage, it can only regulate the output voltage or the output current at a time. When an output channel simulates a completely shaded array and must not output any power, its output current is regulated to zero. However, zero output voltage is not guaranteed in this state under some load conditions, e.g. when the channel is open-circuited. Nevertheless, the output power does remain zero and the output voltage also drops to zero if a load tries to draw current from the channel.

After powering up the SaS V2, the output stages are in an unknown state until the high- and low-level controller programs are started. Although it was checked that the channels do not output any harmful power in this state, a mechanism to physically disconnect the load until the device boots up is certainly missing.

Another weakness of the simulator is its high internal power dissipation due to the linear output stage used. Nevertheless, the device functions reliably even when short-circuited, which is the worst-case condition for a linear power stage. The other imperfections mentioned above do not affect the device's ability to fulfill its intended task either, and it was successfully used to perform the first HiL-based power budget verification of MOVE-II, which was one of the primary goals of this thesis.

9 Conclusion

The SaS V2 was built with the purpose of verifying the MOVE-II power budget and getting more insight to the behavior of the EPS and the power consumption profile of the entire satellite. From the tests that have been performed so far, some basic conclusions can be drawn, which will be summarized here.

The net peak MPPT efficiency of the EPS was found to be higher than previously expected – approximately 95%. This level of MPPT efficiency is maintained even if the illumination is constantly changing at moderate speeds: tumbling was found to reduce the MPPT efficiency compared with Sunpointing mode only by a marginal amount (around 2%). It has been found that the EPS attempts to accept as much power as it can until the battery charge reaches around 80%, according to the battery charge estimation of the CDH. After that, the MPPT efficiency is decreased deliberately, since not all of the available power can be utilized anymore.

The two test cases performed provide information about the satellite's power budget in safe mode and nominal mode. In safe mode, the available solar power is constantly fluctuating which decreases the average available power. Even so, the case was found to be power positive, with the battery level increasing with approximately 5% per orbit.

In nominal mode, the ADCS rotates the satellite to point the Flappanel cells (and the Payload) towards the Sun, which increases the total available power significantly. The power budget was found to be clearly positive in this case with an average gain in battery level of about 15% per orbit.

The performed tests prove that the extended HiL setup with the integrated SAS is capable of providing insights into the satellite's power budget and to the behavior of the EPS. These would be very difficult to obtain and would be less accurate through mathematical analysis.

10 Future Work

There is room for improvement and future tasks related to multiple aspects of the solar array simulator and the HiL environment.

One of these is the execution of a detailed power budget verification. The test cases presented in this thesis may serve as an example for that, but data not considered in those tests may also be analyzed, such as the power consumption of individual subsystems. A comprehensive power budget verification should also perform the same tests multiple times to check if the results are repeatable.

Before simulating the planned test cases, the thermal model for the solar cells should be implemented and integrated into the Simulink model on the HiL PC. Optionally, the shading model of the Sidepanel cells could be improved so that partial shading of the cells scales the solar intensity proportionally to the illuminated cell area.

Another task related to the credibility of the HiL simulation results is to check the correctness of the generated VI curves of the MOVE-II cells. The computed and experimental VI curves have been compared in this thesis, but only three experimental curves were available to support this process. By carrying out measurements for both cell types under different illuminations and temperatures, the utilized model and the tweaked parameters could be verified with a higher confidence. This process needs special equipment, since the VI curves must be measured under the AM0 spectrum, which, in a laboratory environment, can only be generated with a sun simulator. Alternatively, for the 4J cells, the future measurement results of the MOVE-II Payload can be used, since the task of the Payload happens to be the measurement of the VI curves of a 4J cell identical to the ones on the Flappanels. The Payload will carry out the measurements at different solar intensities and temperatures and will transmit the results to the ground station. If the current PV cell model shows unacceptably high inaccuracies for relevant illumination and temperature ranges, then first the tweaked parameters have to be changed. If that brings no improvement, a different kind of solar cell model may have to be utilized.

When the SaS V2 is used independently of the HiL system, the software offers only basic functionality. The user can define the VI curves of the three channels in three text files, but there is for example no way to change the curves while the program is running. The development of a feature-rich, preferably graphical user interface would make the SaS V2 easier to use and more generally applicable. Most importantly, the user interface should offer a feature for VI curve computation and an ability to cycle through a given sequence of curves with pre-defined timings. These are features often found in the software written for commercial solar array simulators.

Should a next version of the SaS V2 be made, the weaknesses that must be solved in hardware should be addressed. The output stage can be redesigned to minimize voltage and current overshoots and to be able to regulate both zero voltage and zero current at the same time even if short-circuited or open-circuited. Relays can be added before the output jacks to disconnect the load when necessary, e.g. during the boot process. The analog-to-digital conversion of the measured output quantities may be moved onto the output channel PCBs, using three ADCs instead of one. This would make it possible to measure the output voltage directly across the output terminals and could improve the accuracy of the device.

Theoretically, the maximum output voltage and current of the simulator can be scaled up significantly by using a higher supply voltage and changing the responsible resistor values in the output stages. However, this would necessitate stronger cooling or paralleling multiple power transistors to share the output current and hence the power dissipation. Therefore, in a future version of the SAS, a more efficient solution should be considered for the output stage, such as a switch mode converter or a linear output stage with a switching pre-regulator. This would open the opportunity to test devices with much higher power requirements, including multi-unit CubeSats.

Finally, the separate PCBs that the device currently consists of could be merged onto one or two larger boards. This would reduce the cost of the device as well as simplify its assembly.

References

[1] MOVE-II website. Available: *www.move2space.de/MOVE-II*. Accessed: 25 Sept. 2018

[2] J. Kiesbye, “Hardware-in-the-Loop Verification of the Distributed, Magnetorquer-Based Attitude Determination & Control System of the CubeSat MOVE-II,” Master’s Thesis, Technical University of Munich, Munich, 2017

[3] Sunpower Corporation, “Sunpower E20/435 Solar Panel”. Available: *us.sunpower.com/sites/sunpower/files/media-library/data-sheets/ds-e20-series-435-solar-panel-datasheet.pdf*. Accessed: 25 Sept. 2018

[4] P. Horowitz and W. Hill, “Solar Cells,” in *The Art of Electronics*, 2nd ed. Cambridge, Cambridge University Press, 1989, pp. 932-933.

[5] C. Riordan and R. Hulstron, “What is an air mass 1.5 spectrum? (solar cell performance calculations),” in *IEEE Conference on Photovoltaic Specialists*, Kissimmee, FL, USA, 1990, pp. 1085-1088 vol.2.

[6] R. Amann, “Electrical Power System,” in *MOVE-II System Documentation*, Technical University of Munich, Munich, 2018. Available: *gitlab.lrz.de/move-ii/move-ii_system-documentation*. Accessed: 25 Sept. 2018

[7] R. Amann, SaS V1 repository on GitLab. Available: *gitlab.lrz.de/move-ii/eps_solar-array-simulator*. Accessed: 25 Sept. 2018

[8] Keysight Technologies, Inc., “Keysight E4360 Modular Solar Array Simulators”. Available: *literature.cdn.keysight.com/litweb/pdf/5989-8485EN.pdf*. Accessed: 25 Sept. 2018

[9] Chroma Ate, Inc., “Programmable DC Power Supply, Solar Array Simulation, Model 62000H-S Series”. Available: *www.chromaate.com/File/DownLoad/42169*. Accessed: 25 Sept. 2018

[10] Aplab Ltd., “SAS12010 PV Panel/Solar Array Simulator”. Available: *www.aplab.com/images/ stories/pdfs/SAS12010%20Solar%20Array%20Simulator.pdf*. Accessed: 25 Sept. 2018

[11] A. Xenophontos, J. Rarey, A. Trombetta and A. M. Bazzi, “A flexible low-cost photovoltaic solar panel emulation platform,” in *2014 Power and Energy Conference at Illinois (PECI)*, Champaign, IL, 2014, pp. 1-6.

[12] S. Gadelovits, M. Sitbon and A. Kuperman, “Rapid Prototyping of a Low-Cost Solar Array Simulator Using an Off-the-Shelf DC Power Supply,” in *IEEE Transactions on Power Electronics*, vol. 29, no. 10, pp. 5278-5284, October 2014.

[13] H. Nagayoshi, S. Orio, Y. Kono and H. Nakajima, “Novel PV array/module I-V curve simulator circuit,” in *Conference Record of the Twenty-Ninth IEEE Photovoltaic Specialists Conference, 2002.*, New Orleans, LA, USA, 2002, pp. 1535-1538.

[14] O. Midtgard, “A simple photovoltaic simulator for testing of power electronics,” in *2007 European Conference on Power Electronics and Applications*, Aalborg, 2007, pp. 1-10.

[15] A. Koran, T. LaBella and J. Lai, “High Efficiency Photovoltaic Source Simulator with Fast Response Time for Solar Power Conditioning Systems Evaluation,” in *IEEE Transactions on Power Electronics*, vol. 29, no. 3, pp. 1285-1297, March 2014.

[16] Z. Zhou and J. Macaulay, “An Emulated PV Source Based on an Unilluminated Solar Panel and DC Power Supply,” College of Engineering, Swansea University, Dec. 2017. Available: *www.mdpi.com/1996-1073/10/12/2075*. Accessed: 25 Sept. 2018

[17] H. Votzi, F. A. Himmelstoss and H. Ertl, “Basic linear-mode solar-cell simulators,” *2009 35th Annual Conference of IEEE Industrial Electronics*, Porto, 2009, pp. 261-265.

[18] D. M. K. Schofield, M. P. Foster and D. A. Stone, "Low-cost solar emulator for evaluation of maximum power point tracking methods," in *Electronics Letters*, vol. 47, no. 3, pp. 208-209, 3 February 2011.

[19] Clyde Space Ltd., “User Manual: 3rd Generation EPS Range,” document No. USM-1335, issue D, October 2017

[20] Azur Space Solar Power GmbH, “30% Triple Junction GaAs Epitaxial Wafer Type: TJ GaAs Epitaxial Wafer 3G30W-Advanced”. Available: *www.azurspace.com/images/products/ 0004301-00-01_DB_3G30W-Advanced.pdf*. Accessed: 25 Sept. 2018

[21] L. Krempel and M. Rutzinger, “Test Data of Solar Cells by Airbus,” February 2017. Available: *redmine.move2space.de/projects/move2/wiki/Test_data_of_solar_cells_by_Airbus*. Accessed: 25 Sept. 2018

[22] K. Janzer, “Thermal Analysis & Design,” in *MOVE-II System Documentation*, Technical University of Munich, Munich, 2018. Available: *gitlab.lrz.de/move-ii/move-ii_system-documentation*. Accessed: 25 Sept. 2018

[23] J. Paulkovich, “Dynamic Solar Cell Power System Simulator,” Technical Report, NASA Goddard Space Flight Center, Greenbelt, MD, United States, June 1965. Available: *ntrs.nasa.gov/search.jsp?R=19660001915.* Accessed: 25 Sept. 2018

[24] Y. Li, T. Lee, F. Z. Peng and D. Liu, “A Hybrid Control Strategy for Photovoltaic Simulator,” in *2009 Twenty-Fourth Annual IEEE Applied Power Electronics Conference and Exposition*, Washington, DC, 2009, pp. 899-903.

[25] P. Horowitz and W. Hill, “Feedback Amplifier Frequency Compensation,” in *The Art of Electronics*, 3rd ed. Cambridge, Cambridge University Press, 1989, pp. 280-287.

[26] On Semiconductor LLC, “TIP142T NPN Epitaxial Silicon Darlington Transistor”. Available: *www.mouser.com/ds/2/149/TIP142T-890084.pdf*, Accessed: 25 Sept. 2018

[27] STMicroelectronics NV, “TIP142T TIP147T Complementary Power Darlington Transistors”. Available: *www.st.com/resource/en/datasheet/tip142t.pdf*. Accessed: 25 Sept. 2018

[28] G. Coley, J. Kridner and R. P. J. Day, “BeagleBone Black System Reference Manual,” Revision C, October 2017. Available: *github.com/beagleboard/beaglebone-black/wiki/System-Reference-Manual*. Accessed: 25 Sept. 2018

[29] Texas Instruments, Inc., “AM335x PRU-ICSS Reference Guide,” lit. Nr. SPRUHF8A, June 2013. Available: *elinux.org/images/d/da/Am335xPruReferenceGuide.pdf*. Accessed: 25 Sept. 2018

[30] D. Molloy, “Real-Time BeagleBone Interfacing,” in *Exploring BeagleBone*. John Wiley & Sons, 2014.

[31] D. Molloy, “Interfacing to the BeagleBone Input/Outputs,” in *Exploring BeagleBone*. John Wiley & Sons, 2014

[32] Traco Power, Inc., “DC/DC Step Down Converter, TSR 1 Series”. Available: *www.tracopower.com/products/tsr1.pdf*. Accessed: 25 Sept. 2018

[33] P. Horowitz and W. Hill, “Heat and Power Design,” in *The Art of Electronics*, 3rd ed. Cambridge, Cambridge University Press, 1989, pp. 623-628.

[34] M. Hejri, H. Mokhtari, M. R. Azizian, M. Ghandhari and L. Söder, “On the Parameter Extraction of a Five-Parameter Double-Diode Model of Photovoltaic Cells and Modules,” in *IEEE Journal of Photovoltaics*, vol. 4, no. 3, pp. 915-923, May 2014.

[35] M. G. Villalva, J. R. Gazoli and E. R. Filho, “Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays,” in *IEEE Transactions on Power Electronics*, vol. 24, no. 5, pp. 1198-1208, May 2009.

[36] M. Reisch, “Solarzellen,” in *Halbleiterbauelemente*. Berlin, Springer-Verlag, 2005, pp. 319-326.

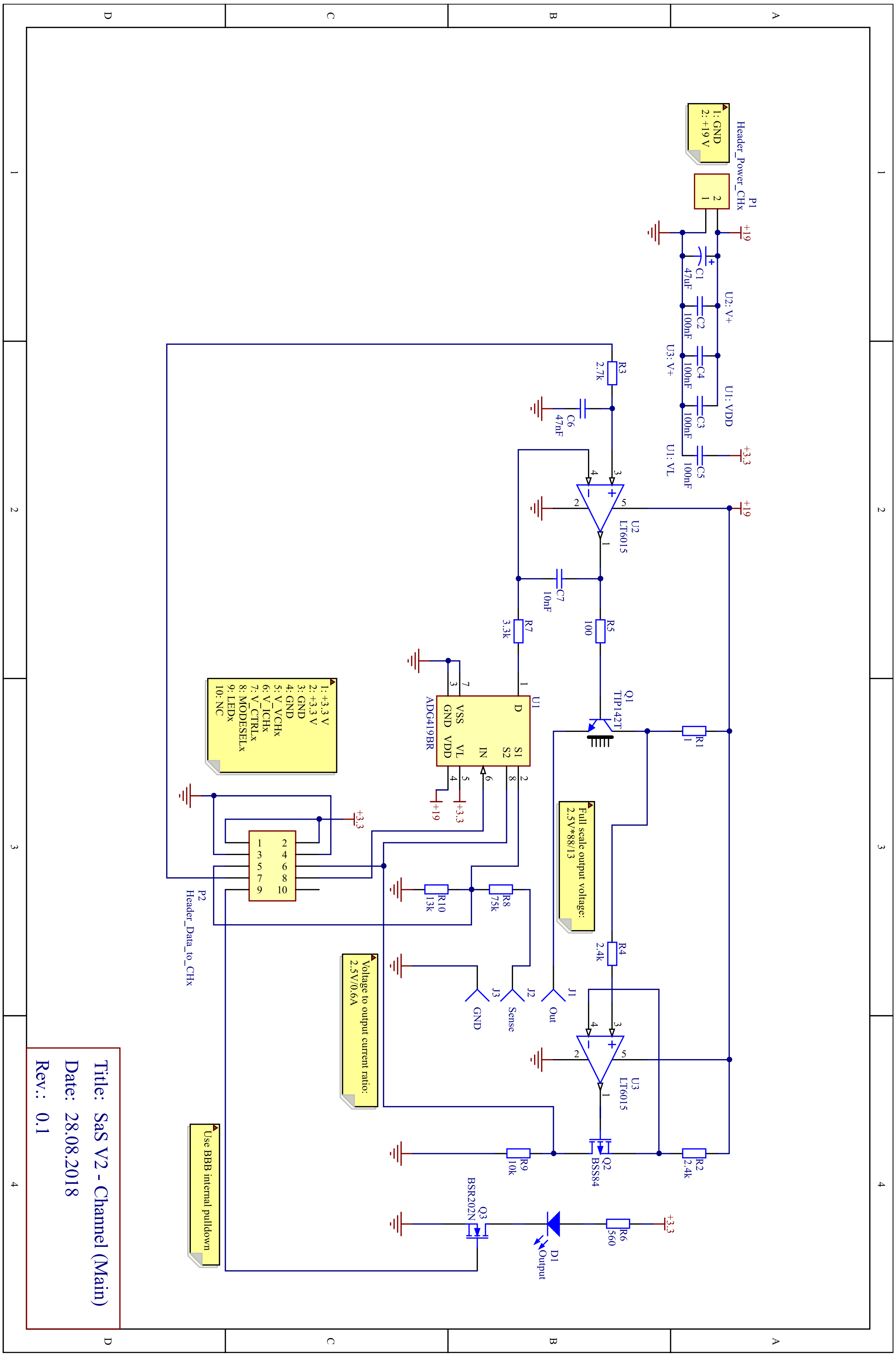
[37] M. Reisch, “Diodenkennlinie, Parameterbestimmung,” in *Halbleiterbauelemente*. Berlin, Springer-Verlag, 2005, pp. 65-74.

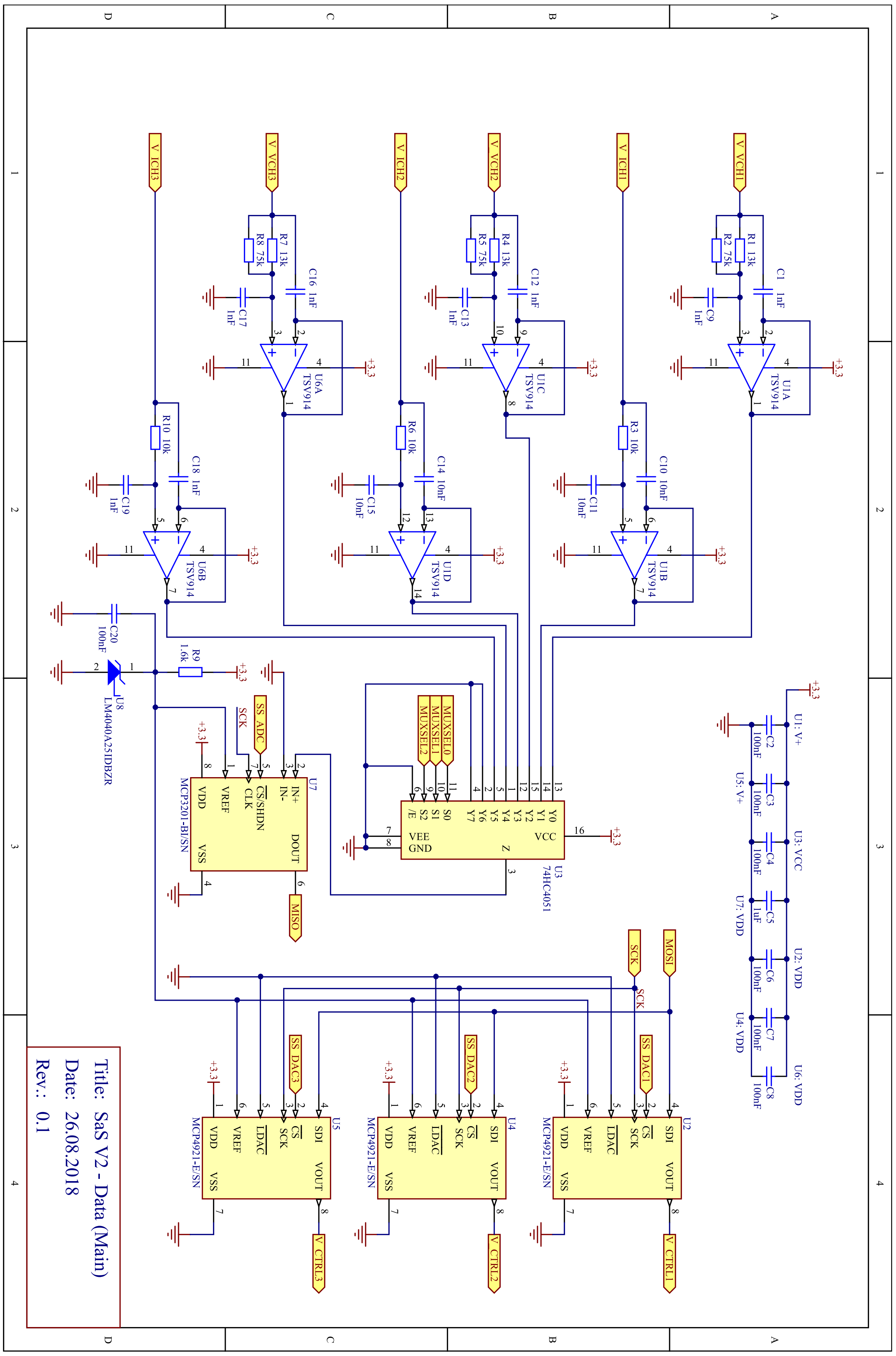
[38] H. Patel and V. Agarwal, “MATLAB-Based Modeling to Study the Effects of Partial Shading on PV Array Characteristics,” in *IEEE Transactions on Energy Conversion*, vol. 23, no. 1, pp. 302-310, March 2008.

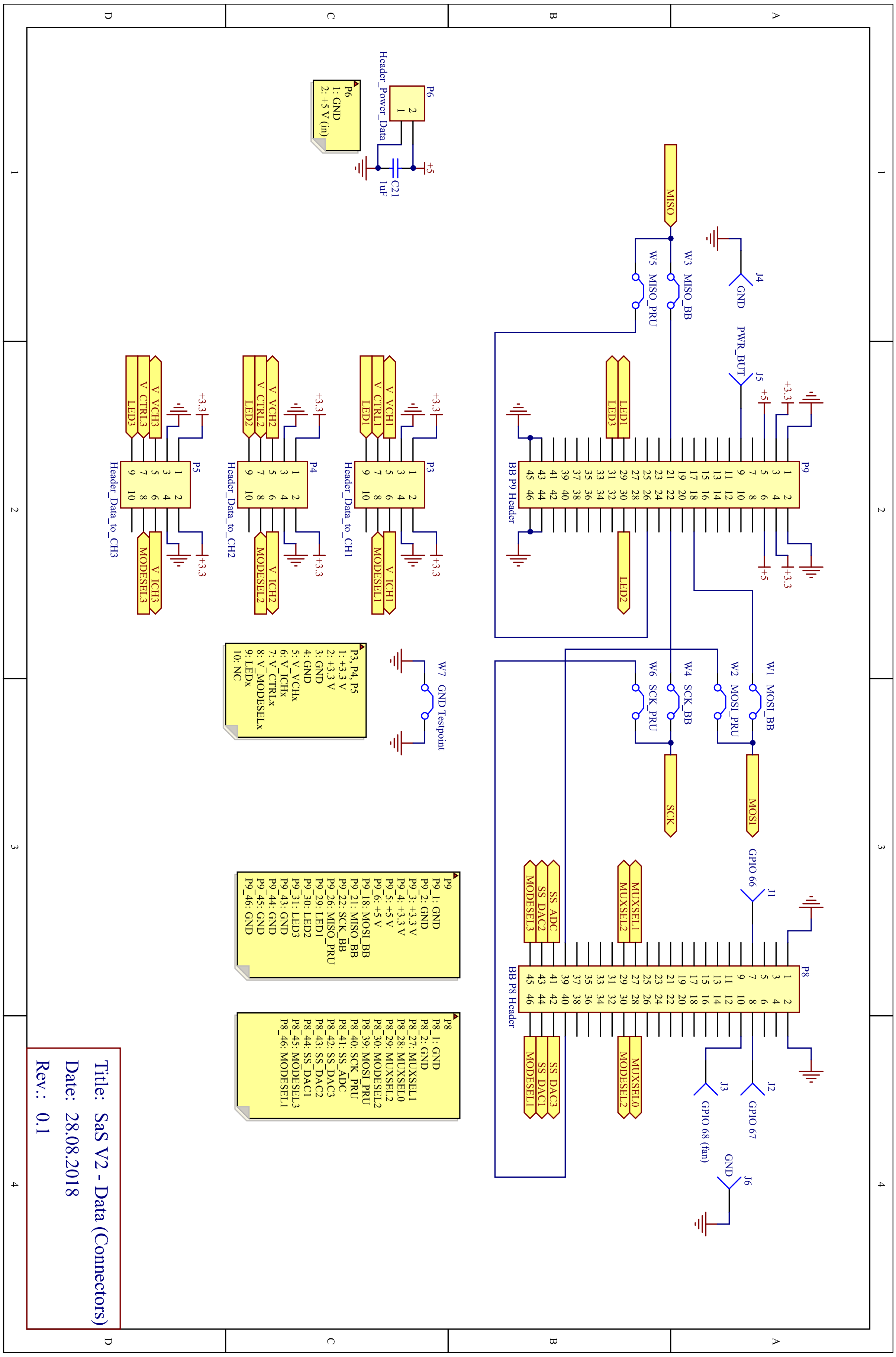
Appendices

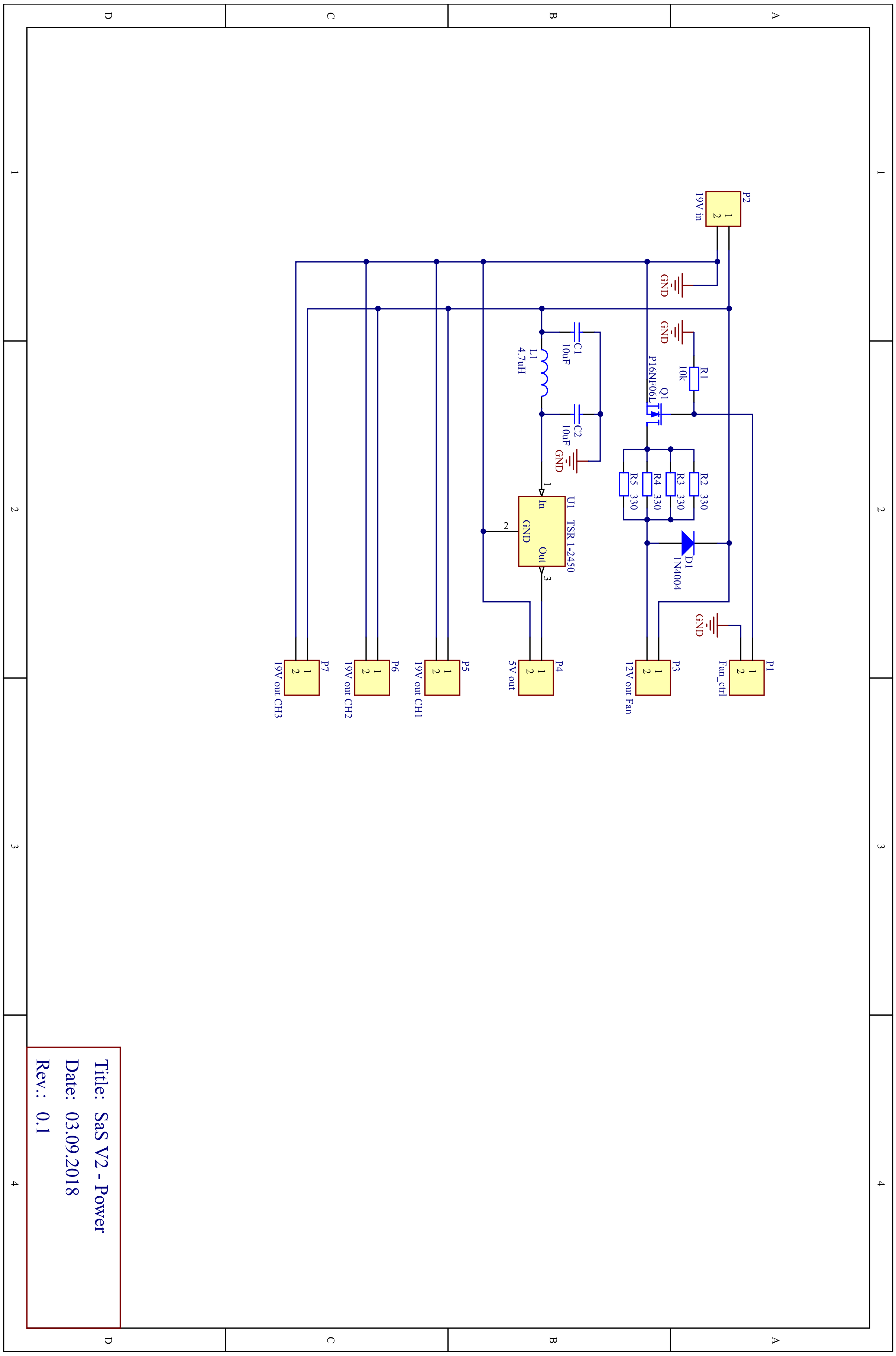
Appendix A: SaS V2 Circuit Diagrams

The following pages contain the schematic diagrams of the electrical circuits in the SaS V2. The diagrams were exported directly from Altium Designer, the CAD tool used for designing the PCBs.


Title: SaS V2 - Channel (Main)
Date: 28.08.2018
Rev.: 0.1
1: GND
2: +19 V
P1
Header_Power_CHx
+19
C1 47uF
C2 100nF
C4 100nF
C3 100nF
C5 100nF
U2: V+
U1: VDD
U3: V+
U1: VL
+3.3
R3 2.7k
C6 47nF
U2 LT6015
C7 10nF
R5 100
R7 3.3k
Q1 TIP142T
R1 1
U1 ADG419BR
D S1 S2 IN VL VDD GND VSS
Full scale output voltage: 2.5V*88/13
1: +3.3 V
2: +3.3 V
3: GND
4: GND
5: V_VCHx
6: V_ICHx
7: V_CTRLx
8: MODESELx
9: LEDx
10: NC
P2
Header_Data_to_CHx
R10 13k
R8 75k
R4 2.4k
J1 Out
J2 Sense
J3 GND
Voltage to output current ratio: 2.5V/0.6A
U3 LT6015
R9 10k
Q2 BSS84
R2 2.4k
Q3 BSR202N
D1 Output
R6 560
Use BBB internal pulldown


Title: SaS V2 - Data (Main)
Date: 26.08.2018
Rev.: 0.1
V_VCH1
V_ICH1
V_VCH2
V_ICH2
V_VCH3
V_ICH3
U1A TSV914
U1B TSV914
U1C TSV914
U1D TSV914
U6A TSV914
U6B TSV914
U3 74HC4051
U7 MCP3201-BI/SN
U8 LM4040A25IDBZR
U2 MCP4921-E/SN
U4 MCP4921-E/SN
U5 MCP4921-E/SN
MUXSEL0
MUXSEL1
MUXSEL2
SS_ADC
SCK
MISO
MOSI
SS_DAC1
SS_DAC2
SS_DAC3
V_CTRL1
V_CTRL2
V_CTRL3
+3.3
U1: V+
U5: V+
U3: VCC
U7: VDD
U2: VDD
U4: VDD
U6: VDD


Title: SaS V2 - Data (Connectors)
Date: 28.08.2018
Rev.: 0.1
P6
1: GND
2: +5 V (in)
Header_Power_Data
C21
1uF
P3, P4, P5
1: +3.3 V
2: +3.3 V
3: GND
4: GND
5: V_VCHx
6: V_ICHx
7: V_CTRLx
8: V_MODESELx
9: LEDx
10: NC
P9
P9_1: GND
P9_2: GND
P9_3: +3.3 V
P9_4: +3.3 V
P9_5: +5 V
P9_6: +5 V
P9_18: MOSI_BB
P9_21: MISO_BB
P9_22: SCK_BB
P9_26: MISO_PRU
P9_29: LED1
P9_30: LED2
P9_31: LED3
P9_43: GND
P9_44: GND
P9_45: GND
P9_46: GND
P8
P8_1: GND
P8_2: GND
P8_27: MUXSEL1
P8_28: MUXSEL0
P8_29: MODESEL2
P8_30: MODESEL2
P8_39: MOSI_PRU
P8_40: SCK_PRU
P8_41: SS_ADC
P8_42: SS_DAC3
P8_43: SS_DAC2
P8_44: SS_DAC1
P8_45: MODESEL3
P8_46: MODESEL1
BB P9 Header
BB P8 Header
Header_Data_to_CH1
Header_Data_to_CH2
Header_Data_to_CH3
W1 MOSI_BB
W2 MOSI_PRU
W3 MISO_BB
W4 SCK_BB
W5 MISO_PRU
W6 SCK_PRU
W7 GND Testpoint
J1 GPIO 66
J2 GPIO 67
J3 GPIO 68 (fan)
J4 GND
J5 PWR_BUT
J6 GND
MISO
MOSI
SCK


P2
19V in
GND
GND
R1
10k
Q1
P16NF06L
C1
10uF
L1
4.7uH
C2
10uF
GND
R2 330
R3 330
R4 330
R5 330
D1
1N4004
U1 TSR 1-2450
In
GND
Out
GND
P1
Fan_ctrl
P3
12V out Fan
P4
5V out
P5
19V out CH1
P6
19V out CH2
P7
19V out CH3
Title: SaS V2 - Power
Date: 03.09.2018
Rev.: 0.1

Appendix B: Low-Level Controller Memory Map

The following table contains the memory map of the low-level controller (PRU). On a single address, one byte is stored. The PRU uses the little-endian byte order, which means that for multi-byte values, the byte with the lowest decimal places is stored at the lowest address.

ADDRESS (HEX)	NAME	LENGTH PER ENTRY (bits)	TOTAL LENGTH (bytes)	DESCRIPTION
0x0000 ⋮ 0x01FF	LUT_1_I-V	16	512	Lookup table slot Nr. 1, I to V direction.
0x0200 ⋮ 0x03FF	LUT_1_V-I	16	512	Lookup table slot Nr. 1, V to I direction.
0x0400 ⋮ 0x05FF	LUT_2_I-V	16	512	Lookup table slot Nr. 2, I to V direction.
0x0600 ⋮ 0x07FF	LUT_2_V-I	16	512	Lookup table slot Nr. 2, V to I direction.
0x0800 ⋮ 0x09FF	LUT_3_I-V	16	512	Lookup table slot Nr. 3, I to V direction.
0x0A00 ⋮ 0x0BFF	LUT_3_V-I	16	512	Lookup table slot Nr. 3, V to I direction.
0x0C00 ⋮ 0x0DFF	LUT_4_I-V	16	512	Lookup table slot Nr. 4, I to V direction.
0x0E00 ⋮ 0x0FFF	LUT_4_V-I	16	512	Lookup table slot Nr. 4, V to I direction.
0x1000	CH1_LUTPTR	8	1	Shows which LUT slot is being used by channel 1.
0x1001	CH2_LUTPTR	8	1	Shows which LUT slot is being used by channel 2.
0x1002	CH3_LUTPTR	8	1	Shows which LUT slot is being used by channel 3.
0x1003 0x1004	CH1_IMEAS	16	2	Measured output current on channel 1. $(1\text{ count} = 0.6/2^{12}\text{ A})$
0x1005 0x1006	CH1_VMEAS	16	2	Measured output voltage on channel 1. $(1\text{ count} = 16.92/2^{12}\text{ V})$
0x1007 0x1008	CH2_IMEAS	16	2	Measured output current on channel 2. $(1\text{ count} = 0.6/2^{12}\text{ A})$
0x1009 0x100A	CH2_VMEAS	16	2	Measured output voltage on channel 2. $(1\text{ count} = 16.92/2^{12}\text{ V})$
0x100B 0x100C	CH3_IMEAS	16	2	Measured output current on channel 3. $(1\text{ count} = 0.6/2^{12}\text{ A})$

ADDRESS (HEX)	NAME	LENGTH PER ENTRY (bits)	TOTAL LENGTH (bytes)	DESCRIPTION
0x100D 0x100E	CH3_VMEAS	16	2	Measured output voltage on channel 3. $(1\ \text{count} = 16.92/2^{12}\ \text{V})$
0x100F	CH1_MODE	8	1	Regulation mode of channel 1. (0x0: VS, 0x1: CS)
0x1010	CH2_MODE	8	1	Regulation mode of channel 2. (0x0: VS, 0x1: CS)
0x1011	CH3_MODE	8	1	Regulation mode of channel 3. (0x0: VS, 0x1: CS)
0x1012	EXIT_FLAG	8	1	High-level contr. sets this byte to 0x1 to stop PRU.
0x1013 ⋮ 0x1FEF	UNUSED	–	4061	Unused area.
0x1FF0 ⋮ 0x1FFF	CALL_STACK	16	16	Area to store return addresses for subroutines.

Appendix C: Structure of Data Packets

The structure of the data packets sent by the HiL PC and received by the SAS is the following.

Name	Data Type	Description
temp[Z-]	64-bit double	Temperature of the cells on the Flappanels [K]
temp[X+r]	64-bit double	Temperature of the right-hand side cell on the X+ Sidepanel [K]
temp[X+l]	64-bit double	Temperature of the left-hand side cell on the X+ Sidepanel [K]
temp[X-r]	64-bit double	Temperature of the right-hand side cell on the X- Sidepanel [K]
temp[X-l]	64-bit double	Temperature of the left-hand side cell on the X- Sidepanel [K]
temp[Y+r]	64-bit double	Temperature of the right-hand side cell on the Y+ Sidepanel [K]
temp[Y+l]	64-bit double	Temperature of the left-hand side cell on the Y+ Sidepanel [K]
temp[Y-r]	64-bit double	Temperature of the right-hand side cell on the Y- Sidepanel [K]
temp[Y-l]	64-bit double	Temperature of the left-hand side cell on the Y- Sidepanel [K]
sun_intensity[Z-]	64-bit double	Illumination level of the cells on the Flappanels [W/m^2]
sun_intensity[X+r]	64-bit double	Illumination level of the right-hand side cell on the X+ Sidepanel [W/m^2]
sun_intensity[X+l]	64-bit double	Illumination level of the left-hand side cell on the X+ Sidepanel [W/m^2]
sun_intensity[X-r]	64-bit double	Illumination level of the right-hand side cell on the X- Sidepanel [W/m^2]
sun_intensity[X-l]	64-bit double	Illumination level of the left-hand side cell on the X- Sidepanel [W/m^2]
sun_intensity[Y+r]	64-bit double	Illumination level of the right-hand side cell on the Y+ Sidepanel [W/m^2]
sun_intensity[Y+l]	64-bit double	Illumination level of the left-hand side cell on the Y+ Sidepanel [W/m^2]
sun_intensity[Y-r]	64-bit double	Illumination level of the right-hand side cell on the Y- Sidepanel [W/m^2]
sun_intensity[Y-l]	64-bit double	Illumination level of the left-hand side cell on the Y- Sidepanel [W/m^2]
timestamp	64-bit double	Time of transmission since simulation start [sec]

The structure of the data packets sent by the SAS and received by the HiL PC is the following.

Name	Data Type	Description
Pavail[1]	64-bit double	Available power on channel 1 [W]
Pavail[2]	64-bit double	Available power on channel 2 [W]
Pavail[3]	64-bit double	Available power on channel 3 [W]
Pacc[1]	64-bit double	Accepted power on channel 1 [W]
Pacc[2]	64-bit double	Accepted power on channel 2 [W]
Pacc[3]	64-bit double	Accepted power on channel 3 [W]
timestamp	64-bit double	Timestamp of received packet that this packet answers

Appendix D: Orbital Parameters Used for Power Budget Tests

The parameters of the orbit for both test cases in Chapter 7 are given in the table below. These values were set up in the Simulink model before launching the simulation.

Symbol	Value	Description
μ	$398600\,\mathrm{km}^3/\mathrm{s}^2$	Gravitational parameter
a	$6953.14\,\mathrm{km}$	Semi-major axis of circular 575 km orbit
T	$5770.1\,\mathrm{s}$	Orbit time
Epoch	18021	Two-line element set (TLE) epoch
n	$14.9738\,1/\mathrm{day}$	TLE revolutions per day
$\dot{n}$	$4.56 \cdot 10^{-6}\,1/\mathrm{day}^2$	TLE first derivative of mean motion
$\ddot{n}$	0	TLE second derivative of mean motion
BSTAR	$0.67675 \cdot 10^{-4}$	TLE BSTAR drag term
i	$97.788°$	TLE inclination
RAAN	$97.788°$	TLE right ascension of the ascending node
e	$6.9162 \cdot 10^{-5}$	TLE eccentricity
ω	0	TLE argument of perigee
M	0.016	TLE mean anomaly

Appendix E: List of Files in the Project’s Git Repository

A repository stored on the GitLab server of the Leibniz Supercomputing Centre contains all source code and design files relevant to this project. The name of the repository is *eps_hil-sas-v2* and it can be accessed via the following link:

gitlab.lrz.de/move-ii/eps_hil-sas-v1.

The Simulink model for the HiL simulation is maintained by the HiL team and is stored in the *adcs_hil* repository, which is accessible via the following link:

gitlab.lrz.de/move-ii/adcs_hil/tree/master.

The table below lists the files in the *eps_hil-sas-v2* repository. Directories are marked by a ‘/’ symbol after their names. The indentation of the lines represents the folder hierarchy.

File or folder name	Description
ethernet_control/	Folder of the Ethernet-based software of the high-level controller (for HiL).
src/	Folder of source files.
communication.cpp	C++ source file of network communications.
communication.h	Header file of communication.cpp.
debug.c	C source file of debugging functions.
debug.h	Header file of debug.c.
ivcurve.c	C source file implementing VI curve related functions.
ivcurve.h	Header file of ivcurve.c.
peripherals.cpp	C++ source file for controlling non-time-critical peripherals.
peripherals.h	Header file of peripherals.cpp.
pru_memmap.h	Header file with macros defining PRU memory locations.
SaS_control.c	Main C source file.
SaS_control.h	Header file of SaS_control.c
scrmessage.c	C source file for displaying messages on console screen.
scrmessage.h	Header file of scrmessage.c.
websocketpp/	Folder containing the code of the *Websocketpp* library.
CMakeLists.txt	Build instructions for cmake.
files_control/	Folder of the file-based (standalone) software of the high-level controller.
iv_curves/	Folder where the simulated VI curves have to be placed.
src/	Folder of source files.
debug.c	C source file of debugging functions.
debug.c	Header file of debug.c.
ivcurve.c	C source file implementing VI curve related functions.
ivcurve.h	Header file of ivcurve.c.
peripherals.cpp	C++ source file for controlling non-time-critical peripherals.
peripherals.h	Header file of peripherals.cpp.
pru_memmap.h	Header file with macros defining PRU memory locations.
SaS_control.c	Main C source file.
SaS_control.h	Header file of SaS_control.c
scrmessage.c	C source file for displaying messages on console screen.
scrmessage.h	Header file of scrmessage.c.
CMakeLists.txt	Build instructions for cmake.

File or folder name	Description
misc/	Folder containing files other than the controller program files.
altium/	Folder of Altium design files such as schematics and PCB layouts.
SaS_V2_Channel.zip	Archive of design files of output channel.
SaS_V2_Data.zip	Archive of design files of data acquisition board.
SaS_V2_Power.zip	Archive of design files of power distribution board.
dto/	Folder of device tree overlay files. For reference only.
readme.txt	Description of files in this folder.
SaS_V1_overlay.dts	Source code of the device tree overlay for SaS V2.
SaS_V1_overlay-00A0.dtbo	Compiled device tree overlay for SaS V2.
libraries/	Folder containing useful libraries. For reference only.
bus/	Folder of libraries for high-level bus communication.
BusDevice.cpp	C++ source file of library for abstract bus device.
BusDevice.h	Header file of BusDevice.cpp.
SPIDevice.cpp	C++ source file for SPI device.
SPIDevice.h	Header file of SPIDevice.cpp.
gpio/	Folder of high-level GPIO library.
GPIO.cpp	C++ source file of GPIO library.
GPIO.h	Header file of GPIO.cpp.
readme.txt	Description of files in this folder.
matlab_scripts/	Folder containing Matlab scripts for VI curve generation.
curvegen.m	Script that generates families of VI curves.
curvegen_single.m	Script that generates a single VI curve.
sortStruct.m	Function for sorting Matlab struct objects.
mechanical/	Folder of mechanical design files.
back_panel.dxf	DXF file for laser-cutting the SaS V2 back panel.
enclosure.cdr	Editable file containing all mechanical designs and panel labels.
front_panel.dxf	DXF file for laser-cutting the SaS V2 front panel.
panel_lables.pdf	PDF export of the panel labels for printing.
plexi.dxf	DXF file for laser-cutting the acrylic layer which holds the SaS V2 PCBs.
pbv/	Folder of power budget verification results.
nominalmode_results.xlsx	Results of the nominal mode test case.
safemode_results.xlsx	Results of the safe mode test case.
useful_vi_curves/	Folder containing definitions of some useful VI curves.
charge_buck.txt	VI curve to charge the satellite through the buck BCRs.
charge_sepic.txt	VI curve to charge the satellite through the SEPIC BCR.
full_scale.txt	VI curve that spans to the specified output range of the SaS V2.
limits.txt	VI curve that spans to absolute maximum output range of the SaS V2.
partial_sh.txt	Typical VI curve of a partially shaded solar array with protection diodes.
readme.txt	Description of files in this folder.
pru/	Folder of the program files of the low-level controller (PRU).
src/	Folder of source files.
delay.ph	Assembly source file of library of software delays.
gosub.ph	Assembly source file of subroutine calls.
memstore.ph	Assembly source file of quick memory stores and loads.
SaS_control.p	Main assembly source file of low-level control program.

File or folder name	Description
spi16.ph	Assembly source file of 16-bit SPI library.
make_pru_bin.sh	Script to assemble the PRU binary.
Readme.md	Readme of the Git repository.