Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterthesis

Embedded software for DC/DC converter

Tim Grieten

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

PROMOTOR :
Prof. dr. ir. Wilmar MARTINEZ

BEGELEIDER :
dr. ing. Kristof ENGELEN

Gezamenlijke opleiding UHasselt en KU Leuven

UHASSELT KU LEUVEN

Universiteit Hasselt | Campus Diepenbeek | Faculteit Industriële Ingenieurswetenschappen | Agoralaan Gebouw H - Gebouw B | BE 3590 Diepenbeek

Universiteit Hasselt | Campus Diepenbeek | Agoralaan Gebouw D | BE 3590 Diepenbeek
Universiteit Hasselt | Campus Hasselt | Martelarenlaan 42 | BE 3500 Hasselt

UHASSELT KU LEUVEN

2022
2023

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterthesis

Embedded software for DC/DC converter

Tim Grieten

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

PROMOTOR :
Prof. dr. ir. Wilmar MARTINEZ

BEGELEIDER :
dr. ing. Kristof ENGELEN

UHASSELT KU LEUVEN

Foreword

As the author of this master's thesis, it is with great excitement that I introduce this work, which represents the culmination of my research efforts in the field of embedded programming. Throughout this journey, I have explored the difficulties novice users face when learning embedded programming and developed a template that aims to alleviate these challenges. This template will be specifically employed by novice PhD students at a research centre specialising in power converters.

Embedded programming, with its blend of hardware and software intricacies, plays a pivotal role in numerous technological advancements. However, the learning curve can be formidable, especially for beginners. This realisation ignited my passion to explore the challenges faced by apprentice users and to seek solutions that could empower and accelerate their learning process.

Through extensive analysis, I identified the key pain points experienced by novice users within the context of power converters. I delved into the intricacies of this domain, striving to uncover the specific hurdles that hindered their progress. Armed with this knowledge, I set out to develop a template that would serve as a guiding light for novice PhD students at the research centre.

The template I have created is the culmination of countless hours of investigation, experimentation, and refinement. It provides a structured and intuitive framework that simplifies the creation of new embedded projects in power converters. By streamlining project setup, code structure, and component integration, the template allows novice users to focus their energy on understanding the core principles of power converters without being overwhelmed by the complexities of embedded programming.

As I reflect on this journey, I am filled with gratitude for the guidance and support I have received along the way. I extend my heartfelt appreciation to my thesis advisor, mentors, and faculty members who believed in my research and provided invaluable insights. Additionally, I am grateful to the research centre specializing in power converters for their collaboration and support in validating the effectiveness of the developed template.

To the novice PhD students who will utilise this template, I offer my sincerest congratulations on embarking on your research journey. I have crafted this template with your needs in mind, aiming to simplify your transition into the world of embedded programming. I hope it will serve as a valuable tool, empowering you to focus on power converter research's exciting challenges and possibilities.

Finally, I would like to express my appreciation to the readers who have shown interest in this master's thesis. It is my sincere hope that the insights and findings presented within these pages will inspire future research and innovation in the field of embedded programming. May this work contribute to a more accessible and impactful approach to education, opening doors for aspiring researchers and fostering advancements in power converter technology.

Tim Grieten

Table of Content

List of Tables

List of Figures

Abstract

There are several challenges in developing embedded controllers for power electronics. The learning curve is very steep and, although there is plenty of information online, it is entrenched in lengthy manuals and complicated datasheets. Furthermore, there is no predefined starting point or template for new projects.

Since creating a control system is rather time-consuming and often not relevant in the research of power electronics converters, it is desirable to reduce control software development time as much as possible. This thesis aims to facilitate this in the best manner possible.

Several steps were taken to solve this problem. To start with, a suitable controller was chosen. This was done using a comparative study of market-available controllers. Then, a framework was developed. The purpose of this framework was to reduce the effort required on the part of the user for programming the controller. Finally, the available information was made more easily accessible to researchers and PhD students starting with embedded programming.

The result of the comparative study is that no controller yields a decisive advantage over another. Therefore, the Texas Instruments Delfino was chosen due to availability and prior experience. The framework resulted in a prefab project and a library containing utility functions. Lastly, a quick start guide that explains the framework and refers to the most important sources was written.

Abstract

Het ontwikkelen van embedded controllers voor vermogenelektronica kent de nodige uitdaging. Zo is de leercurve stijl en is de online beschikbare informatie vaak verpakt in lange handleidingen en complexe datasheets. Daarnaast is er geen vastliggend startpunt of template om aan nieuwe projecten te beginnen.

Aangezien het ontwikkelen van controlesystemen tijdsintensief is en vaak ook niet relevant is in het onderzoek naar vermogenelektronische convertoren, is het aangewezen om de ontwikkelingstijd van deze controlesystemen zo veel mogelijk te beperken. Dit is dan ook het doel van deze thesis.

Hiervoor zijn enkele stappen nodig. Ten eerste werd er een geschikte controller geselecteerd. Dit werd gedaan door middel van een vergelijkende studie van beschikbare controllers. Daarna werd een framework ontwikkeld. Het doel hiervan was om de programmeertijd aan de kant van de gebruiker te reduceren. Ten slotte was het aangewezen om de beschikbare informatie toegankelijk te maken voor beginnende onderzoekers en doctoraatsstudenten.

Het resultaat van de vergelijkende studie is dat geen controller een significant voordeel kan leveren. Daarom werd er gekozen voor de Texas Instruments Delfino omwille van beschikbaarheid en ervaring. Het framework resulteerde in een startproject met bijhorende bibliotheek met nutsfuncties. Tot slot werd een handleiding die het framework uitlegt en naar de belangrijkste infobronnen verwijst geschreven.

1 Introduction

1.1 Context

EnergyVille is an energy research institution located in Genk, Belgium. EnergyVille is the result of a cooperation between KU Leuven, UHasselt, VITO, and imec. The ambition of EnergyVille is to design durable and intelligent energy solutions and use this knowledge to support private stakeholders during the energy transition. EnergyVille is able to design these integrated solutions because it has expertise in solar energy, batteries, power electronics, energy management systems, and much more. This master's thesis will be situated in the field of power electronics.

Power electronics can broadly be defined as the application of electronics to manage and convert electrical power flows. Power electronics can be found in many day-to-day and industrial applications, such as phone chargers or variable-speed drives. This master's thesis will be centred around DC/DC converters.

DC/DC converters are used to convert DC power flows between different voltage levels. The output voltage can be lower than the input voltage (buck converter), or higher than the input voltage (boost converter). Designs that allow both an increase and decrease depending on circumstances also exist.

The general operating principle of such converters revolves around combining electromagnetic components with one or multiple switches. Usually, these switches come in the form of insulated-gate bipolar transistor (IGBT) or silicon carbide (SiC) transistors and are operated at high switching frequencies. A computer or controller is needed to measure the converter's operating conditions and adjust and generate the control signals for the transistors accordingly.

Power electronics are becoming increasingly important given the daily advances in the electrification of energy systems. As such, this master's thesis will play a small part in ensuring that all energy transition stakeholders receive solutions to their specific needs regarding the conditioning and supply of power.

1.2 Problem Statement

EnergyVille is interested in using its existing DC/DC converters supply to conduct experiments or offer services to industrial customers. A customer usually has specific needs and requirements regarding when the converter needs to change its operating parameters. A higher-level controlling system usually measures the current environmental operating conditions and translates these into specific voltage – and current requirements for the converter. This higher-level system needs to be specifically designed to suit every stakeholder's needs. Especially in industrial or experimental applications, this results in different software logic for every different setup. Therefore, such software must be easy and quick to develop.

A second control level is usually needed to make this software design process quick and easy. This second, lower-level logic translates the voltage – and current requirements from the higher level into specific control (pulse width modulation, PWM) signals that control the converter elements. Because programming the converter directly requires specific knowledge of how the converter works in terms of hardware, it can get complicated quickly. A good design approach is to design this lower-level logic once and then use it repetitively in all further applications.

However, if this lower-level software is designed by an external party, then periodically there will be updates that require the developers at EnergyVille to rethink certain higher-level solutions. This makes the design process overly complicated and labour-intensive. It is also practically impossible to make custom changes to third-party solutions.

It is unacceptable that a great deal of work is required to implement the converters into an application. Furthermore, it is highly impractical that these converters require constant supervision and software maintenance once deployed in the field or in an experimental setup. Therefore, the need for an in-house low-level controller with accessible and easily controllable firmware arises.

1.3 Objectives

The primary objective of this master's thesis is to lay the foundation to allow EnergyVille to be able to rapidly prototype power converters in the future. A controller will have to be chosen and programmed to achieve this. There are several additional requirements for this controller.

Firstly, the controller will need to be able to control voltages that are relevant to industrial applications. In other words, the performance of the controller needs to be up-to-date. Achieving ease in programming may not be achieved by selecting a simplistic platform. A literature study on market-available controllers will be performed to compare different platforms.

Secondly, the controller and software have to be user-friendly and easy to maintain. This is especially useful for PhD students, as they only have limited time to get to know embedded programming. The goal is to choose the hardware and write the software in such a way that EnergyVille engineers can focus their time on practically implementing the converters in industrial applications or test setups. It is paramount that they do not waste their time on fixing controller-level bugs and problems. Time and energy can be focused on valuable research if existing technology does not have to be reinvented.

Thirdly, a test setup will be built to test the software designed and evaluated. It is advisable to use off-the-shelf parts as much as possible for the setup to expedite the development process and to ensure time is not wasted on assembling parts. This approach also makes it possible to adapt and expand the setup quickly.

1.4 Structure

This thesis will start by presenting a literature review of several controllers available on the market. It will continue by choosing a controller in chapter 3, and creating an architecture around this controller. This will include information on chosen software and hardware. Later, in chapter 4, all implemented software will be discussed. Finally, all hardware testing will be presented in chapter 5.

2 Literature study

2.1 Platform Comparison

This section aims to objectively compare possible controllers for a DC/DC prototyping setup. Several possible solutions are presented, being:

1. Imperix B-Board PRO [1],
2. Taraz RCP Box [2],
3. Taraz PEController [3],
4. Taraz PE-RCP [4],
5. Compact RIO system [5],
6. TI Delfino [6].

First, this chapter will provide general information on the possible systems. Then, the technical details of the competitors will be summarised over several chapters in tables so that an objective comparison can be made. After that, various system-specific details will be discussed in a separate chapter. A brief run-down will be presented in 3.1.

2.1.1 Imperix B-Board PRO

The B-Board PRO is a converter control module developed by Imperix. It combines high I/O functionality with a small footprint and expanded access to communication standards. The B-Board can be fully programmed using SIMULINK, PLECS, or C/C++ [1]. Figure 1 depicts the B-Board PRO.

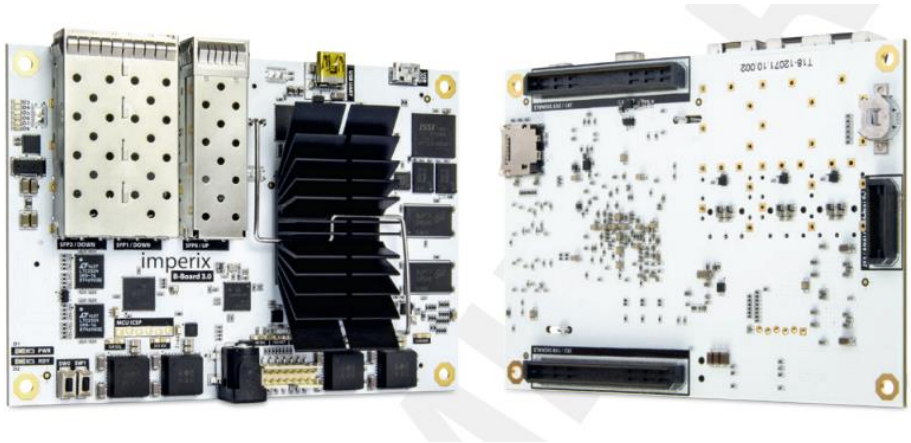

Figure 1: B-Board PRO converter control module [1]

Being one of the contenders for a possible solution, detailed information on the board's capabilities will be provided in the following chapters.

2.1.2 Taraz RCP Box

The Rapid Control Prototyping Box, or RCP Box, is a converter control prototyping solution developed and maintained by Taraz Technologies [2]. As it is intended for the same functionality as the B-Board PRO, it shares many similarities. One difference is that the RCP Box is presented in a more sturdy housing, as shown in Figure 2.

Figure 2: Taraz Technologies RCP Box [2]

Just like the B-Board PRO, the RCP Box provides access to expanded I/O capabilities & communication standards. The RCP Box can be programmed using MATLAB, PSIM, PLECS or directly via C [2].

2.1.3 Taraz PEController

The Taraz PEController is an industrial-grade controller module based on the ST Microelectronics ARM Cortex M7 and M4 dual-core microcontrollers. The PEController is specially designed for power electronics applications. The PEController features an integrated touchscreen to enable HMI functionality [3]. Figure 3 shows the Taraz PEController.

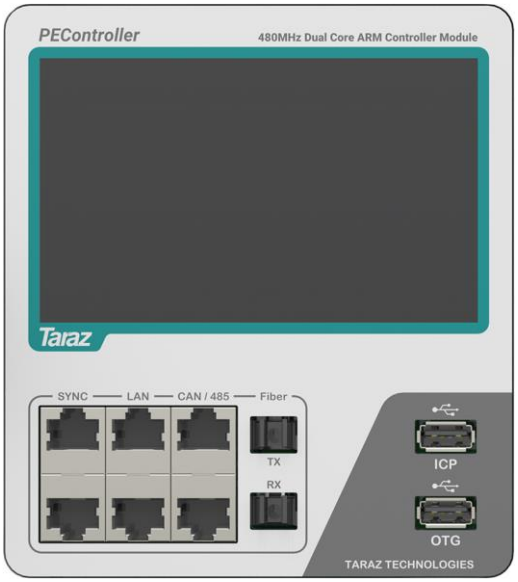

Figure 3: Taraz PEController [3]

The PEController is programmed using C. It also provides several communication protocols and certain I/O capabilities [3].

2.1.4 Taraz PE-RCP

The PE-RCP is a controller module designed for power electronics applications. Just like the RCP-Box, it is based on the Texas Instruments C2000 series microcontroller. Similarly to its competitors, the PE-RCP sports numerous I/O capabilities [4]. Figure 4 shows the PE-RCP.

Figure 4: Taraz PE-RCP [4]

The PE-RCP can be programmed using MATLAB Simulink, PSIM, as well as embedded C [4].

2.1.5 Compact RIO System

The Compact RIO, or CRIO for short, is developed & maintained by National Instruments. On the surface level, the CRIO shares many similarities with PLC systems for industrial automation. For instance, the CRIO system is made up of a CPU, to which I/O- and communication cards can be attached to form a custom system [5]. A notable difference with classical PLC systems is that the CRIO sports an FPGA. This allows the CRIO to generate high-frequency PWM signals. This is generally not possible with standard PLCs. Figure 5 shows a demonstration of a CRIO setup.

Figure 5: Compact RIO [5]

NI offers several software solutions to aid in programming & developing CRIO systems [5].

2.1.6 TI Delfino

The Texas Instruments Delfino, or TMS320F28379D more specifically, is a converter control solution developed by Texas Instruments. The CPU and all peripherals are placed on a small PCB [6]. To access the I/Os, it is recommended to mate the PCB with a docking station. The control card is represented in Figure 7, while the docking station is shown in Figure 6.

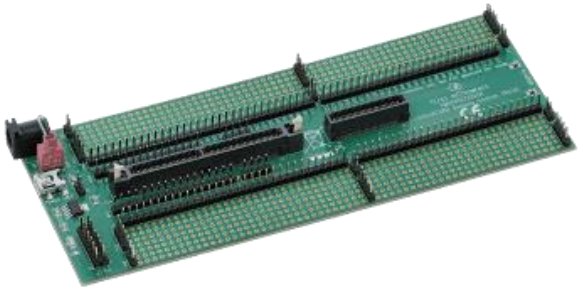

Figure 6: TMDSHSECDOCK docking station for TI Delfino [7]

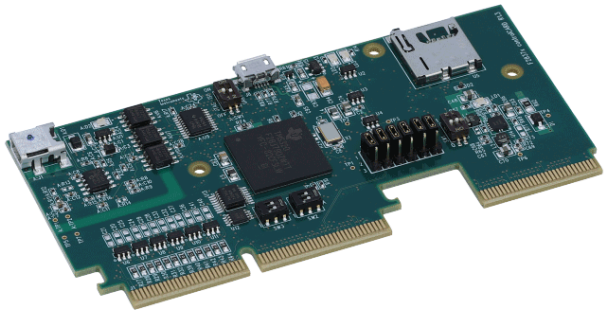

Figure 7: Example of a TMDSDOCK28379D TI Delfino [8]

The TI Delfino can be programmed directly in C using the Code Composer IDE from national instruments [6], it is also possible to create programs in MATLAB using Simulink.

2.2 PWM Outputs

Table 19 contains information about the PWM outputs of the different controllers. Information on the number of outputs, voltage level, resolution and remarks are provided.

Both the B-Board and the Delfino come out on top regarding the number of PWM outputs. The RCP Box has a lower amount of PWM outputs. The PEController and the PE-RCP have an average number of PWM outputs, sitting in between the Delfino and the RCP Box. It should also be noted that the PWM outputs are the same pins as the Digital Outputs for the Taraz controllers. When other logical devices are being controlled, even fewer PWM outputs will be available. On the other hand, the CRIO system offers a configurable amount of outputs. This comes in two ways. Firstly, the number of output cards linked to the CPU can be changed depending on the requirements. Secondly, many different output cards exist. The configuration of these various output cards differs in the number of outputs and their general speed & precision.

Another aspect connected with PWM outputs is their resolution. Here, the B-Board scores well overall, combining a relatively high number of outputs with decent resolution. While the PWM outputs of the TI Delfino are similar in number compared to the B-Board, they are divided into two groups. One group has a very good resolution (0.150 ns), whilst the other has a lower resolution (25 ns). That way, the Delfino provides a better resolution for some outputs traded for a lower resolution for the other outputs. The CRIO cards offer a slightly lower precision, though their values are still acceptable. Information on precision could not be found for the Taraz systems.

An additional aspect of the PWM outputs is the voltage levels. They are also provided in the table. However, these values are not as important as they can be changed easily using a level changer.

2.3 Analogue Inputs

Table 20 is a summary of the analogue input capabilities of the different contenders. The number of channels, bits, voltage range and sample rate are presented.

The Delfino has the highest amount of inputs, sitting at 24. The voltage range on the Delfino's AIs can be set using 2 reference pins on the device. It should be noted that the Delfino has 24 AIs, multiplexed over 4 channels. Each of these channels can be configured to be 12-bit or 16-bit. The bit level and amount of AIs per channel will greatly affect the maximally achievable sample rate.

Next in line are the controllers from Taraz Technologies. The PE-RCP Box and PE-RCP controllers sport 8 12-bit and another 8 16-bit AIs for a total of 16. All 16 AIs on the PEController are 16-bit. Taraz deviates from its competitors when reporting its controllers' sample rate (SR). Rather than reporting the SR of the channel, it reports the SR of each input. The SR of every input drops significantly when processing multiple inputs on a single channel. The SR of each channel for the Taraz modules is reported in the remarks section.

Following Taraz is the Imperix B-Board. With 8 16-bit AIs, it scores slightly less in this category. It still has a good sample rate, between 1 and 2 Ms/s.

The CRIO system, like in the previous segment, has got multiple options available. The first card has 2 24-bit AIs. The second has 8 12-bit AIs. Both cards have a notably lower sample rate when compared with the other contenders.

2.4 General Purpose Inputs/Outputs

Table 21 reviews the GPIO availabilities of the different controllers.

The table is divided between pins that can only serve as an input, ones that can only serve as an output and pins that can act like both. Voltage levels are also provided.

The Imperix B-Board and the TI Delfino both have a large amount of GPIO capability. The controllers from Taraz have significantly fewer GPIOs. Moreover, some DOs and DIOs have to double as PWM outputs, reducing the amount available to the user. Lastly, the CRIO appears to have very few GPIO pins available. However, multiple cards can be added to a CPU to customise the number of GPIOs available.

2.5 Communication

The availability of communication protocols is enumerated in Table 22.

As the availability of communication options is considered to be a "nice to have", a further discussion will not be provided here.

2.6 Various Information

2.6.1 Fault Inputs

Fault inputs are a specific type of digital input designed to quickly interrupt the controller. One example of fault signals can be found in converters. If, for instance, a part of the converter malfunctions, a fault signal can be generated. To guarantee the safe operation of the converter, such fault signals must be responded to swiftly [1].

In this line-up, the B-Board PRO from Imperix is the only contender to offer such fault inputs. The board has 16 of these inputs, operating at a voltage level of 1.8V. According to [1], the controller can interrupt its PWM signals within 50 ns after receiving a fault input.

2.6.2 B-Board Networking Capabilities

An advanced feature of the B-Board PRO is its ability to work in a networked configuration. The I/O capabilities can be vastly expanded using this configuration, as shown in Table 1.

Table 1: B-Board PRO: Comparison between single-unit and networked operating modes I/O capabilities

	Single (1 unit)	**Networked (64 units)**
Analogue inputs	8	512
PWM outputs	32	2048
Digital outputs	16	1024
Digital inputs	16	1024

In other words, the B-Board PRO is capable of working in a networked configuration alongside 63 other B-Boards. This capability can provide a workaround should the amount of I/O capability on one board be unsatisfactory [1].

2.6.3 Taraz Sync

The Taraz systems include a communication protocol to facilitate networking between controllers. This protocol is called Sync. It allows a master to transmit reference signals to up to 10 slaves, whilst the slaves can give feedback to the master [2]–[4]. Additionally, a network of interconnected controllers can be established using the fibre optic ports on the controllers. Fibre optics are better suited for rugged operating environments [2]–[4]. Figure 8 shows how different controllers have to be linked up for Sync to work.

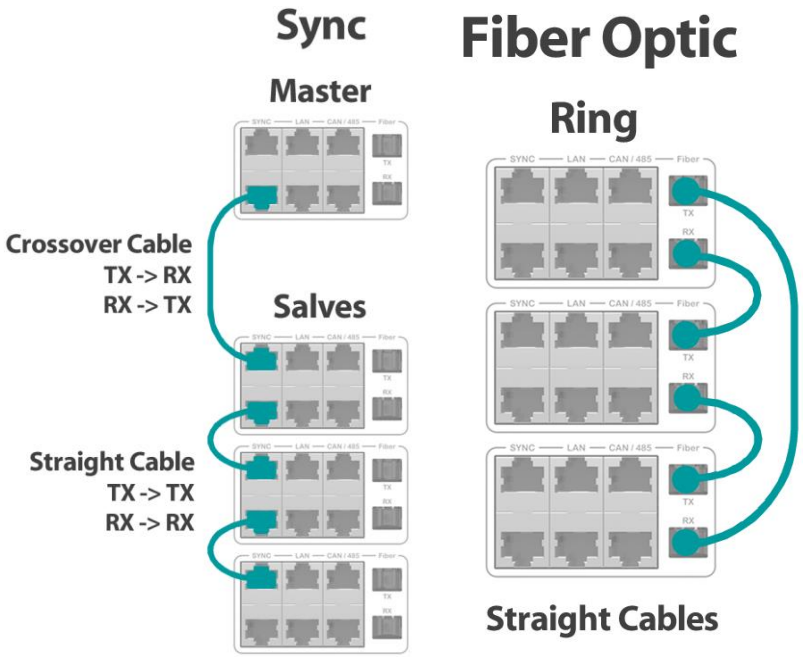

Figure 8: Sync protocol master-slave communication & fibre optic daisy chain for Taraz controllers [2]–[4]

A Sync-based network (left) can be created by connecting all the controllers in series using the Sync ports. The connection between the master and the first slave has to be made using a crossover cable.

Figure 8 also shows the setup for fibre optic-based communications (right). In this scenario, a ring network is used.

3 System Architecture

3.1 Controller

The TI Delfino (TMS320f28379d) is selected as the platform on which to build the framework. There are several reasons for this. Firstly, the literature study has presented an overall tight race between the competitors. The B-Board, PE-RCP Box, PE Controller, PE-RCP and Delfino all have similar performance. The Compact RIO is more limited in hardware capabilities but might be able to make up for this in increased flexibility. All things considered, none of the competitors are able to provide a decisive hardware advantage.

Secondly, the in-house availability of the Delfino is a major plus for preferring it over its competitors. Around the time of working on this part of the thesis, chip shortages impacted several suppliers and led to longer delivery times. Therefore, the physical shipping and delivery times were of concern. It was decided to play it safe and work with the already available controller, rather than risking a long waiting time for a platform with similar characteristics.

Lastly, the Delfino is already in use by many engineers and students at EnergyVille. Introducing a different platform that provides no significant advantage would cause unnecessary problems. Instead, new PhD students can be guided into the existing company ecosystem more easily by providing a framework for the Delfino.

3.2 Power Stack

A small hardware setup is required to test several aspects of the framework. A switching DC/DC converter is ideal for this purpose due to the possibility of choosing a simple topology. Any switching converter setup will require an electrical switch that can handle the main power flow of the converter. The Semikron Semiteach was chosen for this purpose.

The Semiteach is a 3-phase IGBT inverter with a built-in brake chopper, cooling, isolated driver and IGBT protection equipment, designed mainly for educational purposes [9] Whilst it is configured as a 3-phase inverter, it is also possible to use the individual IGBT switches in another converter topology. Figure 9 shows the Semiteach.

Figure 9: Semikron Semiteach IGBT module stack [9]

The reasons for choosing the Semiteach are threefold. Firstly, the Semiteach is available at EnergyVille, avoiding any trouble regarding delivery issues and avoiding the need to select a replacement. Secondly, the Semiteach's six switches can be employed flexibly to create the desired DC/DC topology. Lastly, the combination of an isolated driver, built-in IGBT protection and a hard case, protecting the user against direct contact with power wires, creates a safe working environment.

More information on the specifications and implementation will be provided on page 53.

3.3 User Interface

3.3.1 Current State of Affairs

To understand how any proposed solution can influence the development process of Delfino software, one must first understand how things are handled now. When a new PhD student starts using a Delfino, they have to gain considerable knowledge before being able to perform any practical programming. This knowledge has to be gained mostly through online information, guides and programming manuals. These manuals are often lengthy, and complex, and usually try to give a complete system-level description rather than providing an accessible initiation for novice users

For instance, the TMS320f2837xD Dual-Core Microcontrollers Technical Reference Manual [10] is 2871 pages long and is the primary source of information for programming the Delfino. Also available are a 227-page datasheet [6], a table containing the docking station pin layout, a compiler optimization guide and many other documents. This is a considerable amount of information. Of course, only a minority of this information is useful for getting started. Nevertheless, a newcomer has to sift through these documents to find the relevant information.

Once sufficiently knowledgeable about the Delfino's systems, the PhD student can proceed to the actual programming. Example projects developed by Texas Instruments usually serve as the starting point for a new project. However, these example projects were created by professional embedded developers and this is reflected in their complexity. It is difficult and time-consuming to figure out why certain code fragments were used, what it does, and how to manipulate them to suit the application at hand.

3.3.2 Proposed Solution

The solution proposed by this thesis is twofold. Firstly, the aim is to develop a prefab Delfino project from which future projects can be started. The prefab project will be comprised of two major parts. On the one hand, a framework containing the backbone of the project will have to be developed. This backbone will provide the main structure of the program and will ensure that PhD students do not have to create their own. Instead, they will simply have to write their own configuration and control code and paste it into the correct place in the program structure.

On the other hand, the framework or backbone will be accompanied by a small library containing an initial set of basic functions that can prove useful in many different circumstances. More detailed information on this framework and library is presented in section 4.

Secondly, PhD students will be initiated into the Delfino and the prefab project by a quick start guide. This aims to guide the PhD student through all of the available information. This will be achieved by pointing out what documents to use, which pages are useful and how to approach learning the Delfino and its systems. Also, the quick start guide will provide a practical explanation of how to utilise the framework and library as developed by this thesis.

In summary, the quick start guide will address issues related to the gathering and utilization of information, whilst the framework and library will assist in the practical usage of the Delfino. By addressing these two problem areas, this thesis aims to alleviate many of the problems presented in section 3.3.1.

3.4 Programming Environment

A programming environment is needed to develop programs for the Delfino. This section will briefly discuss the chosen environment and programming aids.

3.4.1 Code Composer Studio

Code Composer Studio is an integrated development environment created by Texas Instruments [11]. It was created specifically to design software for Texas Instruments embedded processors. Figure 10 is a screenshot depicting the Code Composer Studio programming environment.

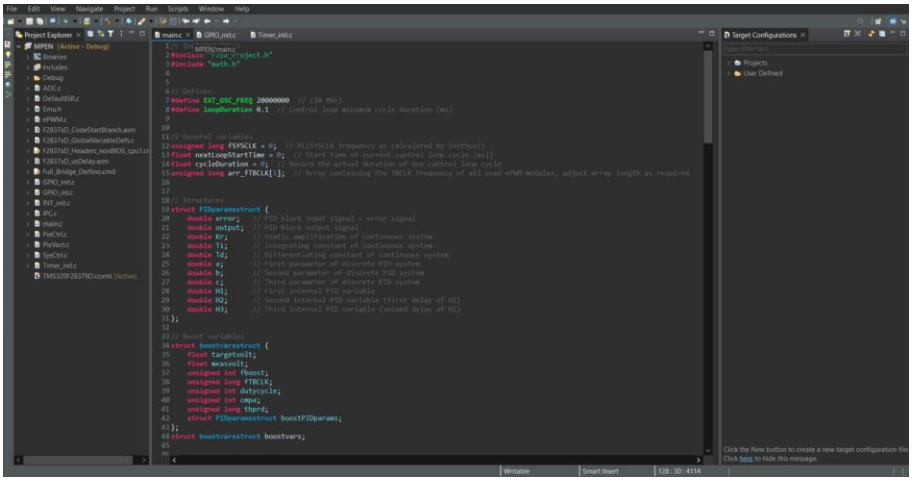

Figure 10: Code Composer Studio

As is to be expected from an IDE, CCS provides several functions to assist the programmer. For instance, code is formatted with different colours depending on the functionality of said code. CCS also gives a project overview and allows the user to quickly create or access source or header files in the project. The most important benefits, however, lie in the integrated debug interface and the C2000Ware software development kit.

3.4.2 CCS Debug Interface

CCS comes with a built-in debug interface to test the performance of the software. To use the debug interface, the user must first connect a control card (a Delfino in this case) to the engineering PC via a USB cable. Then, the debug interface can be launched. This will automatically build the program and download it to the Delfino. Once ready, the control card will start executing the instructions. The info stored in variables and registers is uploaded to the engineering PC by the onboard JTAG interface. This way, the developer can follow the actions of the control card live and manipulate variables as desired. Figure 11 shows a screenshot of the CCS debug interface.

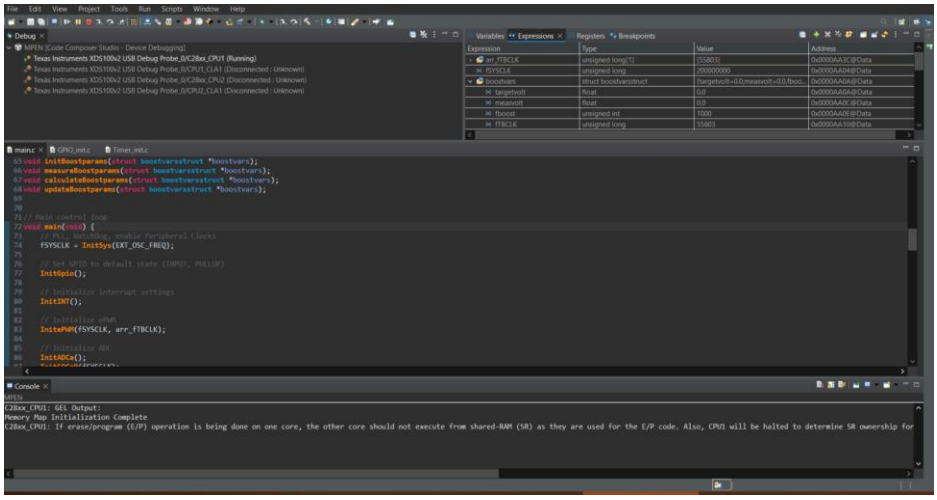

Figure 11: Code Composer Studio debug interface

Two elements in the debug interface are of primary interest. Firstly, the code is visible in the middle section. Secondly, the information stored in variables and registers is displayed in the top-right section of the interface. Figure 12 shows the variable and register inspector.

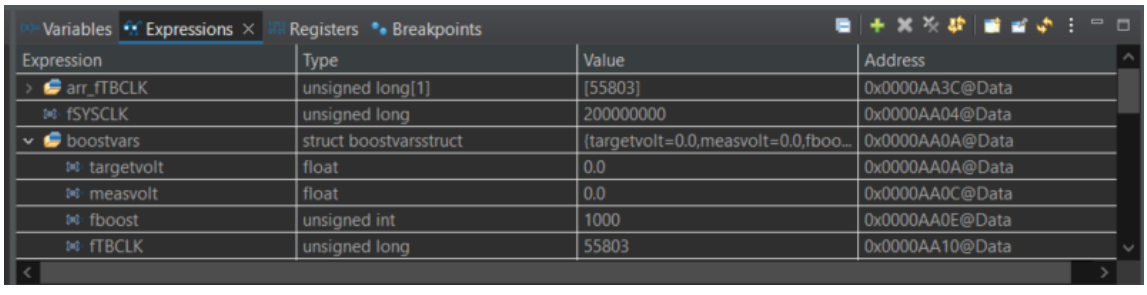

Figure 12: Variable and register inspector

The inspector provides a table containing the name of the variable or register, the data type, the value stored in this variable and the memory address. The values can be edited directly through the inspector.

3.4.3 C2000Ware Software Development Kit

C2000Ware [12] is a software development kit (SDK) specifically tailored to assist in the software development process of TI microcontrollers of the C2000 series. It includes pre-built libraries and drivers specific to each device. The SDK download also contains documents detailing the control cards' hardware layout and the docking stations' pin layout.

The SDK also contains further libraries for signal generation, floating point mathematics, digital control and much more [12]. These libraries may have useful functions in them. However, due to time constraints, these possibilities were not investigated.

3.4.4 Simulink Coder & Embedded Coder

Simulink has apps that can automatically generate embedded code for a microcontroller in C for a Simulink model [13], [14]. It is possible that such code generated by Simulink can be used in the framework presented by this thesis. If true, this could prove to be a very useful addition to the framework. It would allow developers to generate code using Simulink and paste it into the framework. However, this possibility was not researched in depth due to time constraints.

3.5 Safety

Due to the nature of power converters, some safety considerations are necessary. The main safety hazard lies in direct or indirect contact with the process voltage of a hardware setup. Because the hardware setup built for this thesis (see section 4.8) is primarily needed for the testing of controller behaviour, it is possible to keep the voltage levels low. This way, no harmful consequences are associated with direct or indirect contact with live wires.

A secondary safety hazard lies in the utilised equipment itself. Excessive voltages or currents could lead to damage or safety incidents. There are several ways to mitigate this. Firstly, power sources with voltage and current limits should always be used. Secondly, the Semiteach has a driver with built-in protection. This protection includes short-circuit protection, drive interlocking, galvanic driver isolation and supply undervoltage lock-out [15].

However, additional safety measures should be considered if future projects ramp up the voltages and currents to dangerous levels. The work delivered by this thesis in no way guarantees safety. It is the responsibility of any future user to think wisely about necessary steps to reduce risk to an acceptable level.

4 Software Design

4.1 Inspiration

An intriguing question that might arise when thinking about a possible framework to ease embedded software development is the following: How come engineers and PhD students struggle to develop software for their setups, whilst teenagers with little to no programming experience can successfully build and control small setups using an Arduino?

On the one hand, this is arguably partly the case because an Arduino is simpler in capabilities compared to platforms destined for the research sector. On the other hand, Arduino offers a set of predefined functions for use in projects. Certain tasks, like changing a pin from input to output or performing timing-related actions, are simplified to a single command [16]. Also, a blank Arduino project offers a standard framework that provides the user with space to initiate the device [17] followed by a loop [18] in which the actual logic is implemented.

The availability of these utility functions alleviates a lot of work for the developer as they can now focus their attention towards controlling the actual hardware setup and implementing the necessary logic. This approach has proven its worth and will serve as inspiration for much of the software design section.

4.2 Standardised Clock Frequency Calculations

Clock signals are digital signals, usually comprised of a 50% duty cycle square wave and form the beating heart of most digital devices. The clock signal communicates the speed at which a CPU or peripheral module should execute instructions. The Delfino's base clock signal is generated by an onboard oscillator. Because the Delfino makes use of decentralised, autonomous peripheral modules, multiple different clock signal frequencies are required on the board. Section 4.2.1 introduces the clocking scheme on a TI Delfino.

4.2.1 TI Delfino Clocking

Figure 13 presents the clocking scheme on the Delfino graphically. A discussion is presented below.

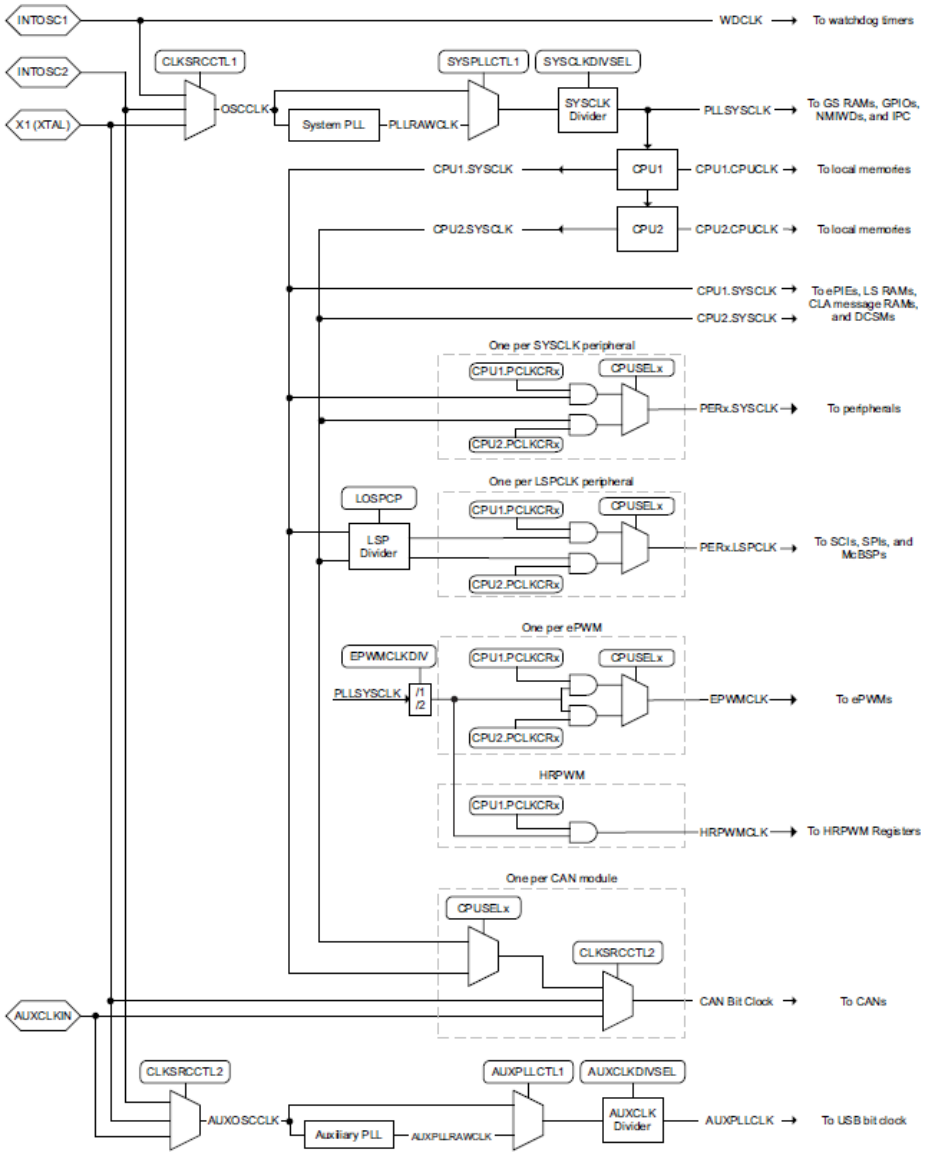

Figure 13: TI Delfino clocking scheme [10]

The onboard oscillators generate the clocking signals, displayed in the image's top-left corner. Three sources are available: Internal oscillators 1 and 2 (INTOSC1/2) and external oscillator 1 (X1/XTAL). The framework is configured to use the external oscillator. This slot is equipped with a 10 MHz or 20 MHz oscillator depending on the control card used.

Since the CPU is rated at 200 MHz, the frequency of the external oscillator has to be increased. This is done by the PLL module, visible in the top-middle of the image. The PLL module multiplies the oscillator frequency by a value which can be set using its multiplier registers. The PLL output signal, named PLLSYSCLK, is then sent to the CPUs and several peripheral modules.

Next, the CPU passes this signal onto several more peripheral modules. These peripheral modules are each equipped with clock dividers and prescalers. This makes it possible for the user to again change the frequency of the clock signal to a desired level within each peripheral module, depending on the localised needs.

The relationship between the clock signal and the performance of a module should not be underestimated. Slowing down the clock signal in an ePWM module will also slow down the ePWM modules' output proportionally. Similar situations are possible in timer modules or communication

peripherals. Also, when the developer changes to a control card with a different external oscillator, all frequencies will change proportionately.

Whilst this architecture provides many benefits and flexibility, it also creates considerable challenges for the user. Knowledge of the clock signal frequency is important in many applications. However, calculating these frequencies requires intricate knowledge of the clocking scheme, register settings and oscillator frequency. Such a calculation is time-consuming. Section 4.2.2 introduces a solution to this problem.

4.2.2 Frequency Calculation Implementation

A straightforward solution to this issue is to automate the frequency calculation by providing a function that can do so. Such a function is implemented for the ePWM modules, for the PLL and for the timer submodules. This way, the most frequently used submodules have already been automated in terms of clock frequency calculation.

Table 2: Summary of frequency calculation functions

Function	Input	Return
PLL frequency calculation	1. System clock frequency 2. PLL settings	PLL output frequency
ePWM frequency calculation	1. System clock frequency 2. ePWM prescaler settings	ePWM base frequency
Timer period calculation	1. System clock frequency 2. Timer divider settings 3. Timer counter settings	The period between timer interrupts

To perform the calculation, the functions first need to read the relevant settings in the registers. Then they can be used to calculate the output frequency based on the input frequency given to the function. Reading the dividers or multipliers from the registers has to be done with care as the value stored in the register usually does not match its actual effect on the frequency. For instance, a divider might contain the value ‘2’ but in reality, it only divides the frequency by 1. The function needs to convert the stored value to its logic value first. Several outlying values are addressed separately.

4.3 Control Loop

The main function and the accompanying control loop are the central backbones of the framework. They provide the structure in which future applications will be programmed. By providing this structure, future users will no longer need to set this up themselves. Figure 14 represents the main function schematically.

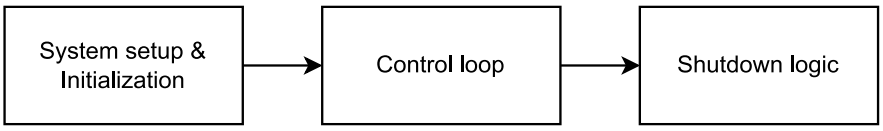

Figure 14: Schematic representation of the main function

Broadly speaking, the main can be divided into 3 distinct parts. To begin, the Delfino's settings and subsystems are configured and readied in the system setup & initialization block. In practice, this block will contain many function calls that each decide how a specific submodule should be set up. Once the Delfino has been configured, the main function continues to the control loop. This is a repeating section of code that supervises and controls the hardware. Figure 15 breaks the control loop down into its subcomponents. To end, once the system has completed its task or needs to be shut down in a controlled fashion due to safety reasons, the control loop stops and the main function passes on to the shutdown block.

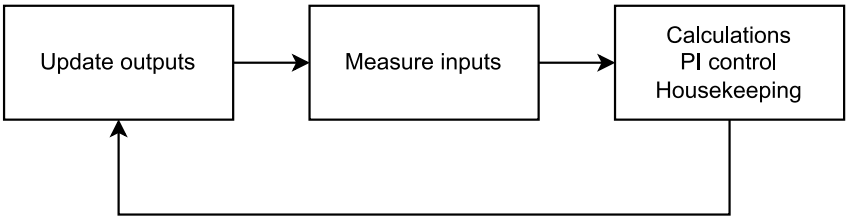

Figure 15: Schematic representation of the control loop

The control loop itself can be subdivided into 3 separate blocks as well. The first action on each iteration of the control loop is to update the output settings. This means that the new output settings, which have been calculated in the last block, are written to the hardware. Next, new information on the state of the system is measured through the inputs. Finally, these measurements are passed on to the final block, which takes care of the calculations, PI control and general logic.

4.3.1 Fixed-Period Control Loop

The control loop has been implemented in such a way that it can be executed at fixed intervals. This interval period can be easily changed in the program. The goal of this fixed-rate interval is to gain control over the logic and to yield predictable PI results. This implementation is achieved practically by keeping the CPU stuck in an empty loop at the start of each cycle until the set amount of time has passed. The timing-related functions used to achieve this are described in section 0.

Whilst this approach works well for simple applications, it can create some challenges when working on more complex converters. When the control loop updates the PWM output settings, some of these changes are not updated immediately. This is to prevent the PWM modules from updating their settings in the middle of a PWM period. Instead, the PWM module operates in shadowing mode where the updated values are only copied to the active registers when the PWM module has finished

its current cycle. This creates another level of unpredictability in the control loop. A solution to this problem is to fix the control loop frequency to the frequency of the PWM module.

4.3.2 Fixed-Frequency Control Loop

By setting the control loop period equal to the PWM module period (or an integer multiplier of the PWM module period), the duration between the updated parameters being written to the shadow register and the active registers being updated becomes predictable. This increases the reliability of the control loop as a whole.

However, the implementation of this proposal creates an additional challenge, as explained by the following example. Figure 16 depicts the typical shape of the inductor current in a boost converter:

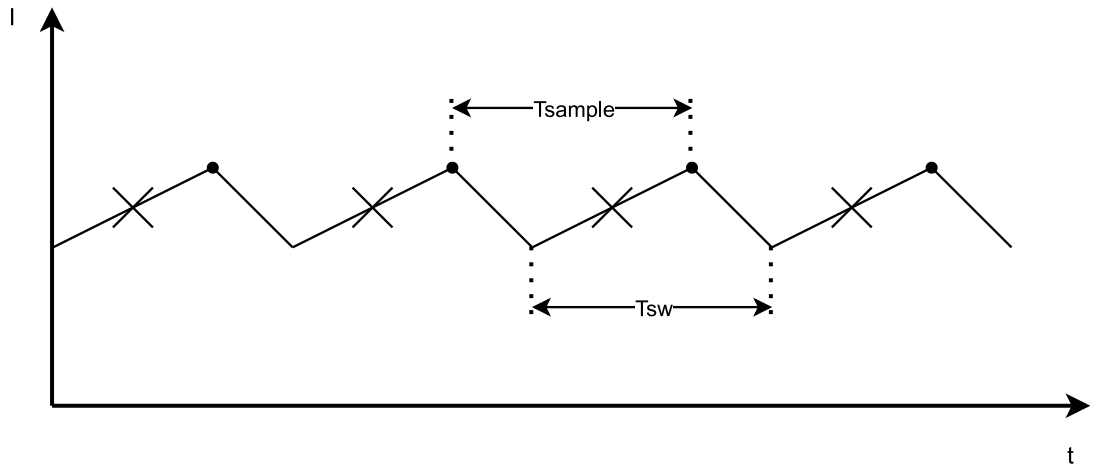

Figure 16: Sampling aliasing applied to an example boost converter inductor

The consequence of fixing the control loop period to the PWM period is that now the measurement gets taken at a similar point on the signal every loop. These measurement points could for instance be the points indicated with a dot. These measurements would indicate to the controller a higher current than actually present in the boost converter.

4.3.3 Future Work

A way to fix the problems described above is by executing the measurement at a known point in the PWM cycle. For instance, if the measurement is performed as indicated by the crosses in Figure 16, the measurement result would be the effective average current. Implementing this, however, requires precise synchronization between the control loop and the PWM modules. Another challenge in this approach stems from the fact that the Delfino uses successive approximation ADCs. The sample and hold window on such devices makes it extra challenging to precisely time the moment of measurement.

Alternatively, multiple samples could be taken per switching period. These samples could then be used to calculate the average current in said switching period. Whilst this approach might alleviate some issues regarding the timing of the sample and hold window, it still requires precise synchronization between the control loop and PWM submodule.

Lastly, to decrease the CPU overload in the measurement process, the CLA could be put into use. The CLA, or Control Law Accelerator, is a submodule on the Delfino that can be used to perform post-measurement calculations on ADC results. It is a 32-bit floating point floating-point math unit that can read the ADC results as they are generated and process them into appropriate results. By shifting this floating-point calculation from the CPU to an independent submodule, the CPU becomes free to perform other time-critical tasks.

4.4 Timing

Many applications for power electronics require that timing functions are available to the controller and the developer. A digital PID controller might have to be run at fixed intervals, or a delay during the startup sequence of an external device might be desired. If a certain process state cannot be attained after a specified period, a device might have to be shut down due to safety concerns.

DSPs, much like most computers, have no standard built-in sense of time. The only option available to the developer is to use the internal clock signals to build time-related mechanisms. Creating such timing functionalities can be labour-intensive. To speed up the development of power electronics applications, standardised timing functionality was developed for the TI Delfino.

4.4.1 TI Delfino Timer Peripherals

The TI Delfino is equipped with three timer submodules, out of which two are available to the user. The timers work autonomously from the CPUs. This means that they do not require constant oversight from the program to work correctly. They contain several registers through which they can be configured by the user Figure 17 depicts the important parts of the timer architecture graphically.

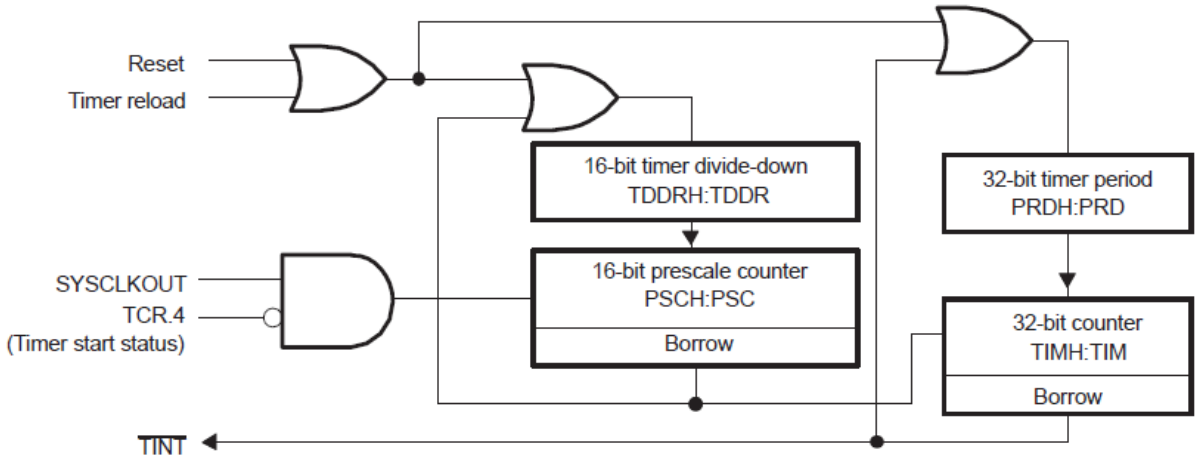

Figure 17: Schematic representing timer peripheral architecture [10]

The inputs of interest are the system clock signal and the counter and prescaler settings. The timer uses a 16-bit prescaler and 32-bit counter. Note that the 16-bit prescaler is divided into an 8-bit high register (PSCH) and an 8-bit low register (PSC). The 32-bit counter is similarly divided into a 16-bit high counter register (TIMH) and a 16-bit low counter register (TIM).

Every system clock signal pulse the low prescaler register (PSC) gets decremented by 1. Once the low prescaler register reaches 0, the high prescaler register is decremented by one and the low prescaler register is reloaded with the value stored in TDDR. Once both prescaler registers reach 0, the prescaler is reset with the value stored in TDDRH:TDDR and a pulse is generated towards the counter.

The counter itself works in the same way as the prescaler, though instead of the system clock, it receives the output signal of the prescaler as a clock source. This means that the input clock frequency can be controlled by changing the values stored in the TDDRH:TDDR registers, effectively generating a slowed-down version of the system clock frequency. By controlling the counter period through the PRDH:PRD, the time it takes for the counter to count to 0 can be precisely controlled. Once the counter reaches 0, an interrupt signal can be sent to the CPU.

By controlling the period of the prescaler and counter, a wide variety of interrupt periods can be achieved. Broadly speaking, this interrupt period can vary between several nanoseconds to slightly over 16 days.

4.4.2 Timing Functions

Several functionalities were developed using the timer submodules on the Delfino. Table 3 provides a summary. More information is provided below.

Table 3: Summary of timing functions

Number	Function	Input	Return
1	Millis	1. Timer interrupt period 2. Timer interrupt count	Time since the program start in milliseconds
2	Timer period calculation	1. TDDRH 2. TDDR 3. PRDH 4. PRD 5. System clock frequency	The time between timer interrupt signals, in milliseconds
3	Delay	1. Desired delay time	/
4	Delay-µs	1. Desired delay time	/
5	Timer ISR	1. Timer interrupt count	Incremented timer interrupt count

1. Millis: This function calculates the time that has passed since the program started. It achieves this by multiplying the period between timer interrupts by the number of times the timer has generated an interrupt. This function can be used in a variety of timing-related problems or in generating trigonometric carrier waves.

2. Timer period Calculation: This function is responsible for determining the period between timer interrupt signals. It requires all 4 timer register settings and the system clock frequency as inputs. It returns the period between timer interrupt signals. By providing this functionality in a library, future users no longer need to perform this calculation by hand. They will simply be able to read out the return value from this function and adjust their settings as needed. This prevents the need to study the functionality of the timer submodules in-depth.

3. Delay: This function is based on the Millis function. Its purpose is to hold the program in an empty loop until the desired delay time has passed. This can prove useful if a delay during the startup of external hardware is required or in similar situations.

4. Delay-µs: The standard delay function is based on the Millis function, which in turn is based on timer interrupts. However, during the startup and configuration sequence of the Delfino, the CPU ignores all interrupts. Therefore, the Delay and Millis functions will only work once the control program is running. The Delay-µs functions can generate an approximate delay by executing an instruction with a known length several times. Therefore, it is independent of system interrupts and will always work, albeit with the compromise of an estimated delay

time. This function was developed and copyrighted by Texas Instruments.

5. Timer ISR: A custom timer interrupt service routine (ISR) had to be implemented to count the number of timer interrupts. The custom ISR simply increments the timer interrupt count every time the ISR is handled. To keep the ISR as lightweight as possible, the variables are stored globally in the program rather than passing them on through the function call.

4.4.3 Future work

Currently, the time is calculated based on the timer interrupt period. This creates a direct link between time precision and interrupt period. To maximally increase precision, the interrupt period should be reduced to a minimum. The side effect of this is that the CPU will have to handle considerably more interrupts, reducing its effective performance. These 2 factors have to be in balance to achieve a correctly performing system.

An alternative implementation could be achieved by directly reading the timer registers (PSCH, PSC, TIMH and TIM) to calculate the time. Setting the timer to its maximum possible period would yield a period of slightly more than 16 days. By using such an implementation, the interrupts are no longer required and could be turned off altogether. Keeping the timer overflow count would make it possible to extend the maximum allowable running time to several millennia. The downside of this methodology is that the function to calculate the current time will have more work and hence will require more time to be completed.

4.5 PI(D) Control

The need for PID control in modern technology cannot be understated. This is also true in power electronics applications. The goal of a PID controller is to manage the output of a system based on what it should be and what it currently is. PID controllers can also be designed to compensate for an external disturbance.

For instance, in a DC/DC converter, a PID controller could be used to regulate the output voltage or the current. The PID controller would continuously receive measurements and adjust the control parameters so that the desired voltages and currents are achieved. The same PID controller could also handle external disturbances inflicted on the system. For instance, the load connected to the converter might suddenly change as a new device is attached to the output. Usually, no information is known about the new load. The PID controller will have to compensate the current so that the output voltage is maintained.

However, implementing PID control is not easy. In theory, the formulas associated with PID are straightforward. On the other hand, these formulas are derived for analogue systems. Using these analogue formulas in digital systems is less than ideal. A digital PID scheme is implemented to address these issues.

This PID scheme aims to reduce the effects inflicted by discretization and quantization. The ambition is to create a PID schema that will yield repeatable results, meaning that performing the same experiment multiple times will lead to similar results every time. Preferably, the schema also makes the results predictable. This way, the amplification and time constants can be scientifically guessed through system modelling and simulation.

4.5.1 Digital PID Scheme

The implemented PID schema is derived from [19] and is shown in Figure 18.

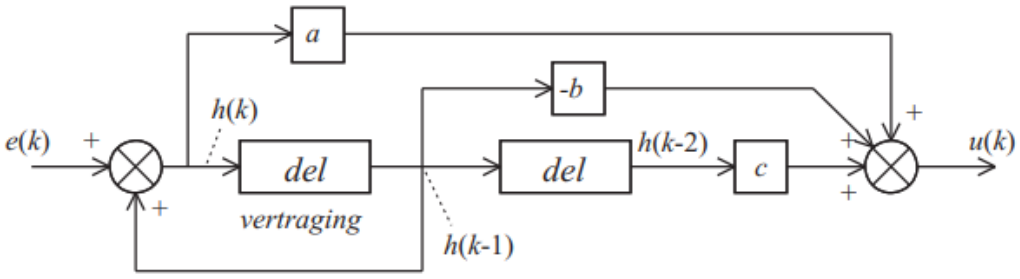

Figure 18: Discrete PID regulator diagram [19]

The error signal, $e(k)$, is at the left of the image. The output signal, $u(k)$, is at the right. First, the error signal is added to the first delay variable, $h(k-1)$. The result of this addition, $h(k)$, is the starting point for the PID control.

The PID calculation starts by multiplying $h(k)$ and a. This product is the first of three terms that get added later to yield the output. Then, the first delay variable ($h(k-1)$) gets multiplied with $-b$ to obtain the second term for the output. Finally, the second delay variable ($h(k-2)$) is multiplied by c to get the final term. The terms are then added together to form the output.

Once the calculation for this control cycle is over, the delay action has to be performed. The first delay variable, $h(k-1)$, is shifted to the second delay variable memory slot, $h(k-2)$. Then, the first variable ($h(k)$) gets copied to the first delay slot. Now the controller is ready to start the next cycle.

In the above calculation, three constants are used. They are not the K, τ_i and τ_d from the equivalent analogue PID controller. Instead, they are calculated using the formulas in Figure 19.

$$a = K_r(1 + \frac{T}{2\tau_i} + \frac{\tau_d}{T}), \quad b = K_r(1 - \frac{T}{2\tau_i} + \frac{2\tau_d}{T}), \quad c = K_r\frac{\tau_d}{T}$$

Figure 19: Formulas to convert PID parameters from analogue to digital equivalents [19]

In these formulas, K, τ_i and τ_d are the parameters defined for the analogue PID equivalent. T is the control loop execution period. a, b and c are compensated for the digitalization effects by using these formulas.

4.5.2 Implementation

Several things are required to implement the PID schema practically. Table 4 summarises the functions related to PID control.

Table 4: Summary of PID functions

Number	Function	Input	Return
1	ConvPIDParams		
2	PerformPIDControl	A pointer to the structure containing all PID variables	/
3	PerformPIDDelay		

1. ConvPIDParams: This function should be called to convert the entered analogue PID parameters and convert them to their digital equivalents. This function only requires a pointer to the PID parameters structure as input. It writes the results directly into the provided variables in this structure. This function is normally only used during the configuration phase of the controller.
2. PerformPIDControl: The formulas of the digital PID scheme are implemented in this function. This function reads the PID constants and the current error signal from the structure and performs the calculation. Then, it will check the result of this calculation to ensure that the output lies within a configurable zone. This is implemented to prevent excessively large output signals. Once completed, it will write the output to the memory location provided in the structure.
3. PerformPIDDelay: This function is responsible for correctly performing the delay action and is called by PerformPIDControl. This function moves the delay variables to their correct slot in the parameters structure.

As mentioned above, a variable structure was implemented alongside the PID functions. This structure contains all variables that are needed for the PID control. This keeps all the related variables together and also makes it possible to use the same PID functions for multiple different control loops. Future users would achieve this by calling an instance of the PID variables structure for each system they want to perform PID control on.

Another advantage of using a structure is that it defines the way the variables are stored. This way, only the pointer to the structure has to be passed to any function for the control to work. The functions know what variables are in the structure and how they should use them. All the user has to do is initiate the variables and update the error signal every control cycle.

4.5.3 Limitations

A limitation of the used PID schema is that it is only an approximation. While it prevents systems that are stable in the analogue domain from becoming unstable in the digital domain, there will be changes to the system's frequency response [19]. This is something that should be taken into consideration when using it to design power electronics converters.

4.5.4 Future Work

Two relatively simple things can be improved on the current design. To begin with, the possibility of changing the PID constants during testing and reconverting them can be implemented. Currently, these constants are converted during the configuration phase of the program and cannot be updated during operation. The possibility of changing these parameters during testing will make it easier to tune the values.

Secondly, anti-windup protection can be implemented. Anti-windup prevents stops the integrator once the PID output has reached a hard limit. This prevents the integrator from winding up in one direction, leading to a prolonged period of inaction. This is not only recommended for practicality but also for safety.

A more profound improvement can be achieved by studying the alternatives to this solution. Alternative schemes or even completely novel approaches can be implemented to improve the behaviour of the PID controller. Due to time constraints, it was not possible to examine this in the scope of this thesis

4.6 Analog to Digital Conversion

Another important aspect of control engineering is the ability to perform accurate measurements and thus close the PID control loop. This section will first explain the working principle behind the successive-approximation converters present on the Delfino. It will continue by highlighting some of the things to consider when using an ADC. It will end by introducing the reader to a function that can help in configuring the ADC submodules.

4.6.1 ADC Submodule Introduction

There are several important things to review when discussing the layout of an ADC module. Figure 20 shows a schematical representation of an ADC submodule on the Delfino.

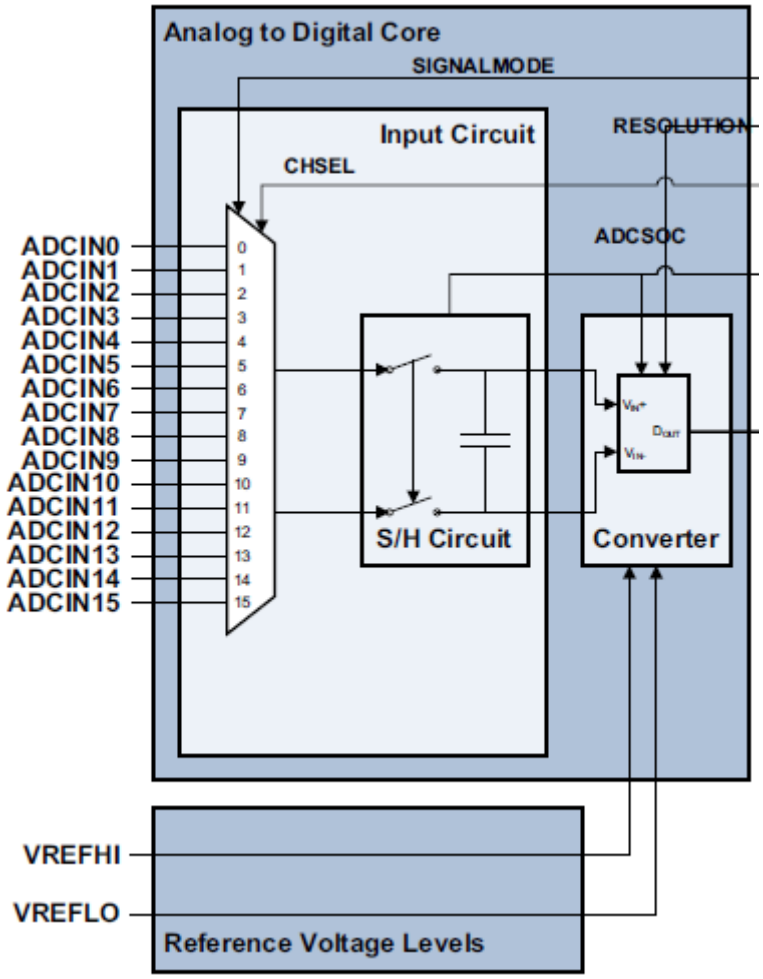

Figure 20: Graphical representation of an ADC submodule in the TI Delfino [10]

The input pins are located on the left side of the image. Up to 16 pins are available per ADC channel. These pins are multiplexed into the sample and hold circuit (S/H circuit). The S/H circuit is placed centrally in the image. The purpose of the S/H circuit is to charge a capacitor with the voltage present on the input pin, and then isolate and hold that voltage level for conversion. This is to ensure that the conversion is performed on a stable voltage source, leading to more accurate results.

The converting itself is performed by the actual converter channel, located at the right of the image. It is a successive-approximation style ADC. It converts the analogue voltage to a digital signal by successively approximating the sampled voltage and adjusting until the smallest possible error is

obtained. Each converter channel is clocked by an internal clock signal, which is obtained by dividing the system clock signal coming from the CPU.

Each converter channel can be operated in 12-bit or in 16-bit mode. In 12-bit mode, the converter will operate in single-ended input mode. An electrical representation of this mode is presented in Figure 21.

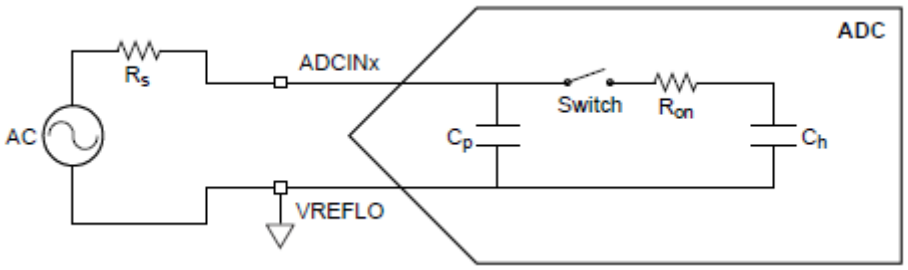

Figure 21: Single-ended input mode [10]

In single-ended input mode, the input voltage is compared referred to the ground of the Delfino. It is important to design the measuring circuit so that the Delfino ground and external system ground are at equal potential. The conversion process then references the input voltage to the internal 3.3V reference. Alternatively, the 16-bit mode together with differential input mode can be used. This is presented in Figure 22.

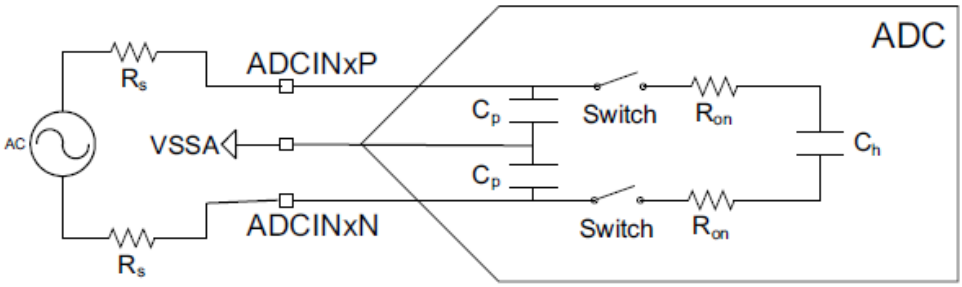

Figure 22: Differential input mode [10]

In differential mode, the input voltage will be compared to an external reference ground. 2 Input pins are now required for the measurement. Besides allowing for greater precision, it is now also possible to measure negative voltages. The input voltage is still referenced to the internal 3.3V reference.

4.6.2 ADC Configuration

The ADC must be configurable and controllable by the user program. Therefore, a start of conversion (SOC) is needed. A SOC is a collection of registers that determine the parameters and conditions needed to execute a conversion. For instance, a SOC contains information on the pin it should convert and the desired S/H time. There are up to 16 SOCs available per ADC channel. Each SOC contains information on which pin to convert, the desired S/H time, and when the SOC should be triggered. The results are stored in 16 end of conversion blocks (EOC), corresponding to each SOC. If desired, simple fixed-point calculations can be performed on the result automatically using the post-processing blocks (PPBs).

An important thing to consider is the S/H time to use. There are two major requirements for this signal:

1. The S/H time must respect a minimum value defined by the datasheet,
2. The S/H time can be no shorter than one ADC clock cycle.

To comply with the first requirement, the developer has to browse through the datasheet of the control card used to find the minimum required S/H period. Then this number has to be converted to the number of clock cycles it would take for this time to pass.

The second requirement might seem odd at first. After all, how can the S/H circuit be configured to run faster than at least one clock cycle? The answer lies in the fact that the S/H circuit is clocked via the system clock, rather than the ADC clock. This means that the S/H clock signal is faster than the ADC clock signal by a factor defined by the ADC clock divide-down register. This is another aspect to be considered and another calculation to be made.

Besides the hardware-imposed requirements, there are other factors to consider. Shortening the S/H time will allow for more conversions to occur in a shorter period of time. This will allow the PI controller to work with more recent data, improving its accuracy.

On the other hand, a very fast S/H frequency might not allow for sufficient time for the S/H capacitor to charge. This means that the conversion is performed on a voltage different to the one present on the input pin. This can be especially problematic if one ADC channel is used to measure multiple voltages with greatly varying values.

4.6.3 SOC Sanity Check

To make it easier to configure a SOC, a function that checks the validity of the proposed S/H period was implemented. A flow chart of this function is presented in Figure 23.

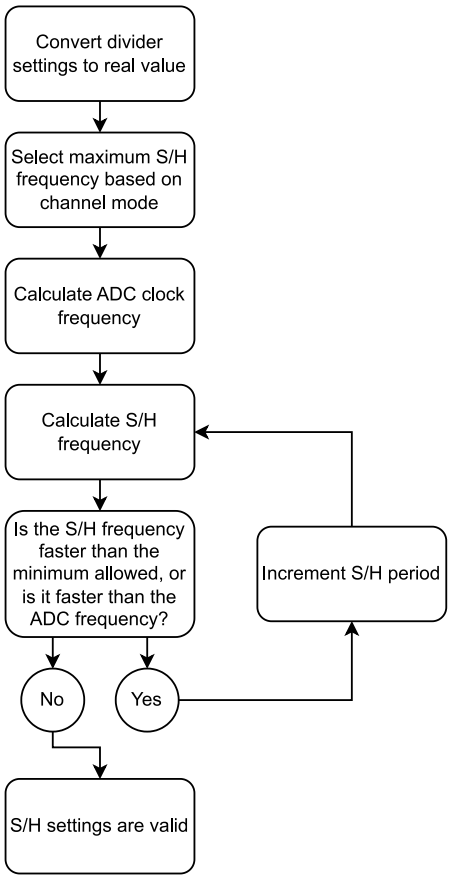

Figure 23: SOC sanity check flowchart

The sanity check requires information about the system clock frequency, ADC clock divider setting, ADC operating mode and the proposed S/H period, stored in the acquisition period selector or ACQPS register.

The sanity check starts by converting the ADC clock divider from its stored value to its real value. For instance, when the ADC clock divider is programmed with 6, it will actually divide the clock signal by 4. Since the actual factor is required for the calculation, this conversion needs to be made first.

Next, the sanity check will have to select the minimum S/H period based on the operating mode of the ADC. The minimum period is different in 12-bit than in 16-bit mode.

After initializing all required numbers, the real calculation can now start. First, the ADC clock frequency is calculated based on the system clock frequency and the converted divider setting. The sanity check continues by calculating the current proposed S/H period. This is done using the system clock frequency and the proposed ACQPS setting.

The sanity check now has to perform the actual check: Are the proposed values valid? In other words, is the S/H period not shorter than the minimum allowed and at least one ADC clock cycle long? If not, then the proposed ACQPS will be incremented and the calculation will be repeated. If yes, the ACQPS value will be returned.

This sanity check can be used in two different ways. The first option is to calculate a desired ACQPS setting beforehand, and then using the sanity check to ensure its validity. Alternatively, the sanity check can be used to calculate the fastest possible S/H frequency for the user by suggesting an ACQPS of zero.

4.7 GPIO functions

Several functions related to the general-purpose input/output system were also implemented. These functions are for controlling the digital input/output pins. The core of these functions was developed by Texas Instruments. However, their formatting was rather complex, especially for newcomers to embedded programming. That's why their formatting and presentation were modified to make it more approachable. Table 5 presents a summary of the GPIO functions available in the framework. More explanation is provided below the table.

Table 5: Summary of GPIO functions

Number	Name	Purpose
1	GPIO_ReadPin	Reading the state of a pin
2	GPIO_WritePin	Changing the state of a pin
3	GPIO_SetupPinMux	Changing the functionality of a pin
4	GPIO_SetupPinOptions	Changing the mode of a pin
5	GPIO_SetupXINTGpio	Linking a pin to an external interrupt channel
6	Config_XINT1	Configure external interrupt channel 1
7	XINT1_isr	Custom external interrupt 1 service routine

1. GPIO_ReadPin: This function simply reads the state of a pin in the appropriate register and returns its digital value as a 0 or 1. The desired pin to be read has to be provided in its function call. Provided by TI.
2. GPIO_WritePin: This function changes the state of an output pin. The pin number and the desired output state have to be provided. Provided by TI.
3. GPIO_SetupPinMux: This function can change the connectivity of a pin. The Delfino is equipped with a crossbar system that can link each input pin to multiple different functionalities on the board. For instance, a pin can be changed from serving as a digital input to the CPU to serving as an output pin for a communication module. Provided by TI.
4. GPIO_SetupPinOptions: This function should be used to change a pin from input to output mode. This function also takes care of extra flags, such as the desired state of a pull-up resistor. Provided by TI.
5. GPIO_SetupXINTGpio: Using this function, a pin can be configured as a source for one of five external interrupt channels. This is useful for processing external fault signals that need to be handled quickly.
6. Config_XINT1: An example for configuring the external interrupt system is provided in this function. It contains all the steps that should be taken to properly configure the external interrupt channel. This code can be copy-pasted to easily configure the other external interrupts as well.
7. XINT1_isr: This is a custom interrupt service routine that contains the code to be executed when external interrupt 1 is triggered.

4.8 Future work

A potentially useful feature which was planned but not completed due to time constraints is logging. The idea behind a logging function is to create the ability to export data to an engineering PC without using the debug interface. Logging would make use of the onboard communication modules of the Delfino. For instance, the onboard USB controller could be used to easily transfer data to an engineering PC via serial communication protocols. The data could be read using a serial monitor or a custom program in Python. This would allow the user to more accurately track the status of the Delfino before a certain event occurs, making it easier to debug setups.

This could be taken further by preconfiguring more onboard communication modules. This way, future projects could use these protocols to establish communication between different devices in a network. The importance of communications in modern electrical appliances with the advent of smart grids cannot be overstated. However, these are more advanced features and were thus not appropriate for the scope of this project.

While an initial control loop was implemented in this thesis, several issues and ways of improving the design were identified. These were related to the synchronization of the control loop to the external system. Either the placement of the measurement needs to be precisely controlled to obtain the real value of the measured voltage or current, or many measurements need to be taken per cycle to obtain an average.

The section that leaves the most room for improvement is the PI(D) control. An initial scheme was implemented to make the framework usable in a practical setup, but the implementation has certain limitations. For one, the used digital control scheme is not perfect. Future research into better schemes and alternative approaches are useful. Furthermore, it was identified that integrator anti-windup and live PID parameter conversion can be easily implemented to improve the quality of the PID implementation.

Besides things directly related to the PID control itself, a way of improving the calculating speed of the controller was proposed using the control law accelerator. The CLA can perform automatic calculations on the ADC results using 32-bit floating point math. This would reduce the CPU overload on the PID calculations, allowing for faster control loop execution.

A submodule that was mostly completed was the timer. While the timers are completely configured in the current framework, it was noted that alternative ways of implementing the time-keeping functions are available. The choice between these two methods is a balance between short calculation latency and higher interrupt frequency, or longer calculations but no interrupts. Ultimately, it is up to the user to decide whether it is better to switch to the alternative method. As the configuration is already completed, this switch can be made relatively easily.

Finally, a section that requires significant improvement is the measurements. While a configuration and a utility function were provided, no practical work on measuring equipment was performed. Getting a precise and accurate measurement is key to designing a fast PI(D) controller. If there are too many erratic measurements, the controller will have to be slowed down to ensure it does not react to these erratic measurements, and it may become impractical to use differential action in the controller. This will become apparent during the practical tests in the next chapter.

5 Hardware Setup & Testing

A small hardware setup was built to test the solutions presented in Chapter 4. This chapter will first continue by introducing the reader to the different hardware elements. Then, several tests will be presented in the following sections. These include static and dynamic tests and will primarily be oriented towards testing the PID control loop. This chapter will end by presenting a conclusion on the test results.

5.1 Bidirectional Buck/Boost Converter

A bidirectional buck/boost converter was chosen for the testing. There are multiple advantages to using this topology. Firstly, it is a relatively simple topology that can be built using off-the-shelf parts at EnergyVille. This was an important factor in speeding up development. Furthermore, the required control signal is a complementary PWM signal. The configuration to generate such a signal is comparatively easy. This is another factor in getting the testing done in a shorter period. Lastly, the bidirectional nature of the topology allows for more flexible testing, allowing for a multitude of cases to be tested.

Two iterations of this converter will be used for testing: One using a constant voltage load and the other using a resistive load at the output.

Figure 24 shows the setup using a constant voltage load.

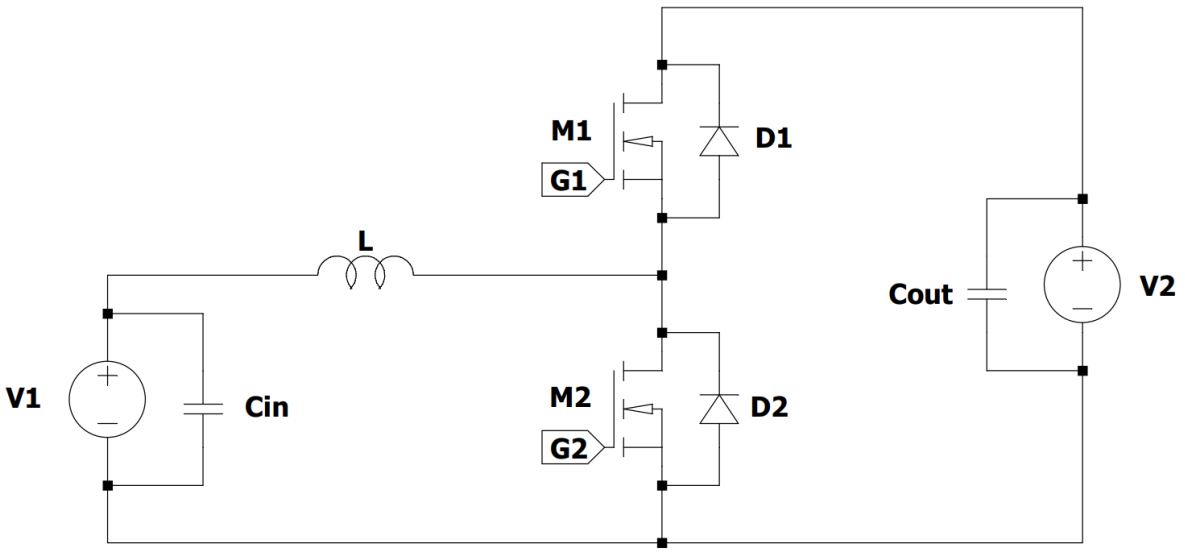

Figure 24: Bidirectional buck/boost converter with constant voltage load schematic

The switching converter itself is located centrally in the image. M1 and M2 are the IGTB switches. They are located in the Semiteach converter stack. D1 and D2 are the freewheeling diodes and are also included in the Semiteach. The switch and diode are the SKM50GB12T4 fast IGBT4 modules [20] and form an integrated package. The IGBTs receive their control signal from the Delfino via a level shifter. This signal is represented via G1 and G2. More information about the level shifter will be presented in section 5.4.

The voltage input for this setup, V1, is located on the left. The voltage output, V2, is located on the right. Two bidirectional DC power supplies are used. These power supplies are decoupled from the

switching device using a custom capacitor bank. Each capacitor bank contains several ceramic capacitors to cancel out high-frequency noise and several electrolytic capacitors to buffer the voltage during the switching periods. The output capacitor bank is in addition to the capacitors already present in the output of the Semiteach.

The inductor is located on the input side of the converter. An inductor that was already available at EnergyVille was chosen for the setup. The inductance was in the magnitude of several mH. The advantage of having such a large inductance is that the discontinuous conduction mode (DCM) band is reduced to a minimum. DCM occurs when the inductor current becomes 0 during at least part of the switching cycle. The effect of operation in DCM will be visible in some of the tests.

The inductor's purpose is to keep the current flowing through it constant. This is the core principle behind the operation of this converter. When switch M2 is closed and M1 opened, the entire input voltage is available for the inductor to charge its magnetic field. This causes the current in the inductor to rise. The load is fed from the output capacitor bank during this process.

Once the charging period is over, switch M2 is opened and M1 is closed. Theoretically, the output voltage, which is factor 2 higher than the input voltage in this test, will force the current to run from output to input. However, the inductor will resist this change in current. It will use the energy stored in its magnetic field to boost the voltage from the input source to the voltage level of the output source. The inductor current decreases during this process. Table 6 summarises these switching states.

Table 6: Bidirectional buck/boost converter switching states

M_1	M_2	V_1	V_2	V_L	I_L
Opened	Closed	$20V$	$40V$	$= -V_1 = -20V$	Increasing
Closed	Opened	$20V$	$40V$	$= V_2 - V_1 = 20V$	Decreasing

The inductor voltage can be deduced using Kirchhoff's voltage law. The switching states are achieved in practice by generating complementary PWM signals with a small dead time using the Delfino. A switching frequency of 20 kHz will be used.

An alternative setup uses a resistive load instead of a voltage source as output. This is represented in Figure 25.

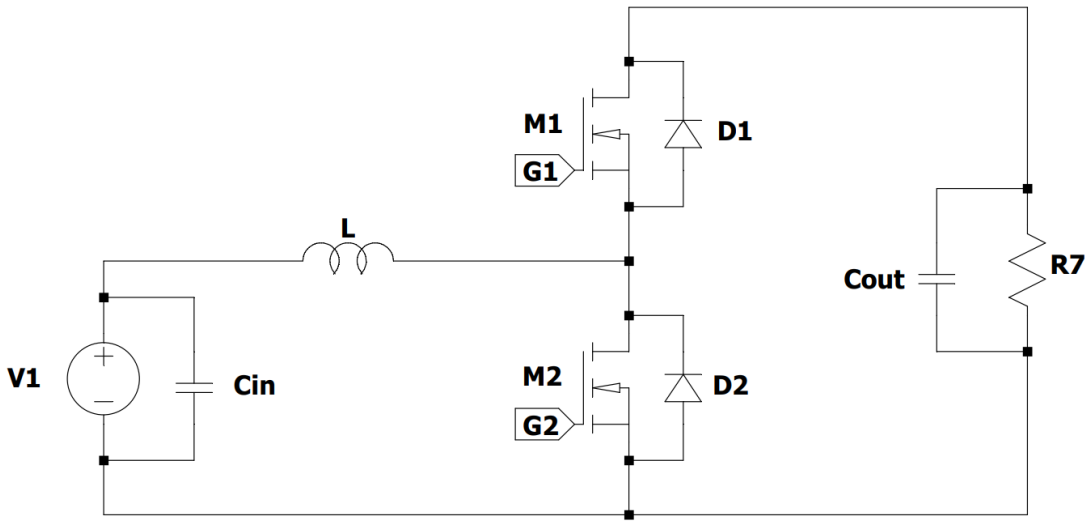

Figure 25: Bidirectional buck/boost converter with resistive load schematic

The only difference is that the output voltage source, V2, has been replaced by a resistor (R7). Whilst seemingly insignificant at first, this has an important consequence in the requirements for PI control. This will be explained in detail in section 5.2.

5.2 Control Loop Design

5.2.1 Current Control

To understand the working principle behind the implemented control loops, it is important to understand the role of the output capacitors in the system. The voltage over the legs of a capacitor is determined by the charge held by the capacitor. The charge in the capacitor is effectively the result of the capacitor current and the duration of said current. By regulating the current through the capacitor, the charge in the capacitor and thus the voltage over the capacitor can be controlled.

In the setup using two voltage sources, the capacitor charge is controlled directly by the voltage sources themselves. They will regulate their current so that the voltage setpoint is achieved at all times. This means that the Delfino only has to change the PWM duty cycle to transfer power from one source to another. The control loop to achieve this is presented in Figure 26.

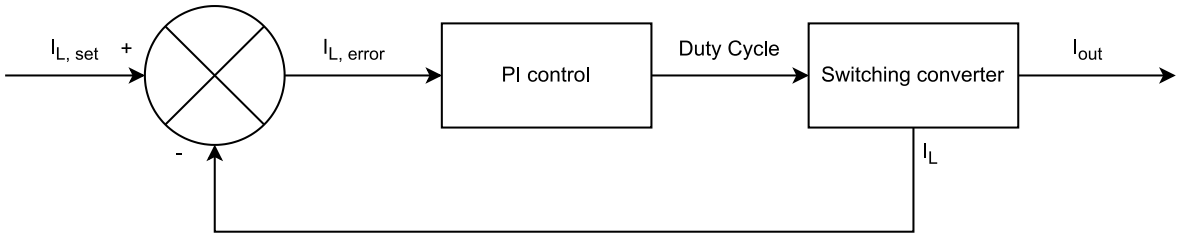

Figure 26: PI control loop for current control

The desired inductor current is on the left side of the image. The measured inductor current, coming out the bottom of the converter on the right side of the image, gets subtracted from the desired inductor current to form the inductor current error. This gets passed to the PI control logic in the Delfino. The PI control logic regulates the PWM duty cycle to bring the measured current towards the desired current. In this setup, the Delfino does not consider the voltages as they are defined and regulated by the power supplies. The parameters of the PI control are presented in Table 7.

Table 7: PI parameters used for current control

Symbol	Value	Description
K	1	Static amplification
τ_i	0.1 s	Integrator time constant
τ_d	0	Differentiator time constant
$T_{control\ loop}$	130 µs	Control loop period

The amplification and integrating values were obtained experimentally. As the design of a control system is not the core of this thesis, it was not attempted to maximise the response speed. These values were chosen because the response time obtained using these values proved satisfactory. The loop control period was deliberately chosen to not equal an integer multiple of the PWM period ($50\ \mu s$) to avoid any of the issues described in section 4.3.2. Unless explicitly specified otherwise, these parameters are used for the current control loop.

5.2.2 Voltage Control

An important consequence of replacing the voltage source at the output with the resistor in Figure 25 is that the voltage at the output is no longer kept in balance. A current increase in the converter will be fully absorbed by the output capacitors. The output voltage will continuously increase because of this. As the output voltage increases, the resistor current will increase proportionally. Once the resistor current is again equal to the converter current, the system has regained a steady state.

In practice, such fluctuations in output voltage are undesirable. That is the reason why a second control loop is needed. The purpose of this loop is to keep the output voltage constant under varying load conditions. This control loop is presented schematically in Figure 27.

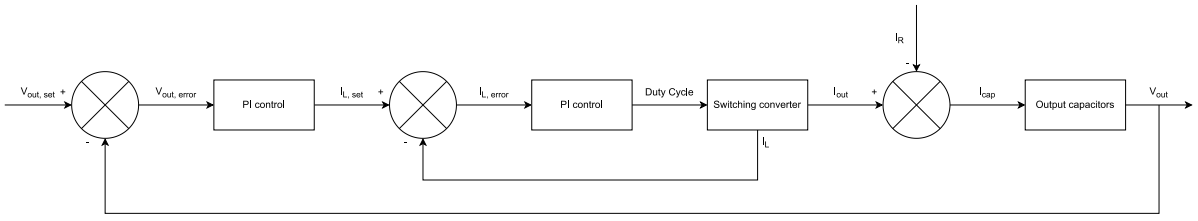

Figure 27: PI control loop for voltage control

The voltage control starts from the setpoint for the desired output voltage. The measured output voltage is subtracted from this value to achieve the voltage error. This error is passed to the voltage PI control. The output of the voltage PI control is the setpoint for the inductor current, which is passed on to the current control loop (same as in Figure 26). The output current of the converter is sent to the output capacitors and the load. The current flowing through the load is subtracted from the converter output current to get the capacitor current. The capacitors convert this current into the output voltage.

The resistor current is an unknown disturbance in this system. It is impossible to calculate or know this disturbance since the value of the output load is not known and is variable. From this perspective, it becomes clear that the practical purpose of the voltage control is to compensate the current control setpoint for the unknown disturbance.

The parameters of the voltage PI control are presented in Table 8.

Table 8: PI parameters used for voltage control

Symbol	Value	Description
K	1	Static amplification
τ_i	0.05 s	Integrator time constant
τ_d	0	Differentiator time constant
$T_{control\ loop}$	130 µs	Control loop period

Just like the PI parameters for the current control, these values were obtained experimentally. The control loop period is the same as the one presented in Table 7 (they are the same variable). It is reiterated that the design of the fastest possible controller is not the purpose of this thesis. These values were deemed satisfactory because they allowed for the controller behaviour to be visualised using an oscilloscope. These values were used for the voltage control loop during the testing unless explicitly stated otherwise.

5.3 Semikron Semiteach

The Semiteach contains the actual power switches used in the test setup. A general discussion on the Semiteach was provided in section 3.2. This section will provide the technical details relevant to this application. Table 9 presents a summary.

Table 9: Technical details for the Semiteach power converter

Symbol	Value	Description
I_{max}	$30A$	Maximum permanent current
$V_{DC,max}$	$750V$	Maximum DC bus voltage
$f_{sw,max}$	$50\ kHz$	Maximum switching frequency
C_{out}	$0.88 – 1.32\ mF$	Output capacitor bank capacity
V_{driver}	$15V$	Driver supply voltage
$V_{i,tr,h}$	$12.5V$	High input threshold
$V_{i,tr,l}$	$4.5V$	Low input threshold
R_i	$10\ k\Omega$	Input resistance

In practice, only the maximum current will introduce a limit. All other maximum ratings are significantly higher than the values intended for use in the setup. The input resistance needs to be accounted for when designing the level shifter.

5.4 Level Shifter

Because the voltage levels of the Delfino and the Semiteach do not match, a level shifter is needed. A level shifter changes the voltage level of a signal. The level shifter design is presented in Figure 28.

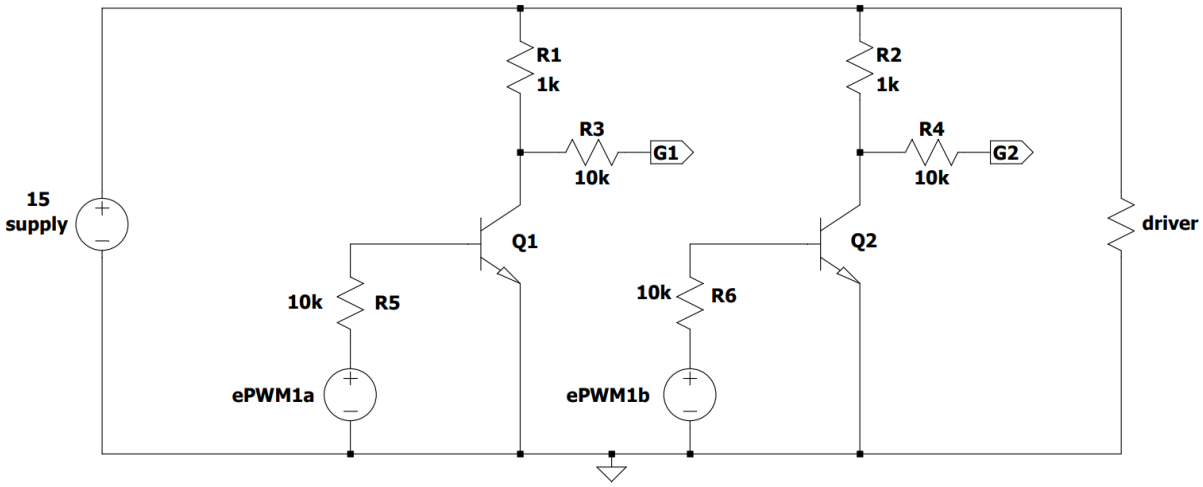

Figure 28: Level shifter schematic

There are several key components visible in this image. Firstly, the Delfino output signals, ePWM1 and ePWM2, are located at the bottom of the image. Their voltage is 3.3 volts. Secondly, The supply voltage is located at the left of the image. This is the source used to change the voltage level of the Delfino output signals, as well as to supply the Semiteach gate driver (visible on the right). The supply operates at 15 volts. Furthermore, two NPN transistors (Q1 and Q2) are present in the image. They, together with the resistors do the actual conversion of voltage levels.

The process starts when the Delfino generates a signal. This signal goes to the transistor base through R5 or R6. The purpose of these resistors is to limit the current flowing through the base of the transistor. As the transistor greatly amplifies the base current to obtain the collector current, it is necessary to limit the base current. As a significant collector-emitter current now flows through the transistor and R1 or R2, the voltage available to R3 or R4 reduces to almost 0. Note that R3 and R4 resemble the Semiteach driver input resistance. This effectively brings the Semiteach input voltage down to 0, causing the associated IGBT to stop conducting.

When the Delfino pulls its output voltage down to 0, no current will flow through the transistor base. This means that no collector-emitter current will flow through the transistor. The Semiteach input voltage is now equal to $R_3 \cdot V_{supply} \,/\, (R_1 + R_3)$. The Semiteach will interpret this as a high.

Notice how the usage of the level shifter has effectively inverted the Delfino output signals. This needs to be accounted for in the software. This can also lead to dangerous situations. Namely, when the Delfino is not turned on and no output signals are generated, the level shifter will interpret the absence of signals as two high signals. Switching both IGBTs on at the same time would theoretically cause a short circuit. However, the Semiteach's gate driver has built-in interlocking protection which prevents this from happening.

It would be better to route the level shifter output signals through an exclusive or gate before passing them to the Semiteach. This is something that needs to be considered when using this level shifter design for a power converter that has no interlocking or short-circuit protection.

5.5 Measurements

Several measurements are required to achieve closed-loop control. Moreover, these measurements need to be performed by the controller itself. There are measurement devices for current and voltage specifically designed for use in power electronics setups available on the market. However, implementing these would cost additional time. Instead, readily available laboratory-grade measurement equipment was chosen for this purpose.

As indicated by the control loop diagrams in Figure 26 and Figure 27, two separate measurements are required to control the system. The first measurement is the inductor current. This current is measured using a current clamp. Figure 29 shows an example current clamp.

Figure 29: Example of a current probe [21]

The second measurement, the output voltage, is measured using a differential probe. Figure 30 shows an example of a differential probe.

Figure 30: Example of a differential probe [22]

To evaluate the performance of the controller as scientifically as possible, an additional measurement is created. The setpoint for the current control loop will be visualised on an oscilloscope using an onboard DAC module. This makes it possible to more precisely track the status of the PI control system.

5.6 Current Control Testing in Steady State Conditions

The first test is designed to evaluate the ability of the current control loop to keep the converter in a steady state. First, the control loop will attempt to send 20A in boost converter mode, and then it will attempt to receive 20A in buck converter mode. Table 10 shows the test results in boost converter mode.

Table 10: System parameters during boost converter steady state operation

Symbol	Result	Description
V_{in}	$20.0V$	Input voltage
V_{out}	$40.0V$	Output voltage
U_{sw}	$40.8V$	Switch voltage amplitude
I_{in}	$20.1A$	Input current
I_{out}	$-8.0A$	Output current
$I_{L,avg}$	$20.1A$	Inductor current, mean
DC	40.5%	Duty cycle (top switch)

The input voltage is kept at 20V, whilst the output voltage is 40V. This will be kept constant throughout the testing. The input current settled at 20.1A, whilst the output current reached -8.0A.

At first, this imbalance might seem odd. Using a boost factor of 2, it is to be expected that the output current is half of the input current. The difference can be explained by the voltage loss over the IGBTs. IGBTs are designed for applications with relatively high voltages. At a higher voltage level, the voltage drops experienced over the IGBTs are relatively small. However, since this setup uses the Semiteach at much lower than nominal voltages, the voltage drop over the IGBTs becomes significant. The converter has to work at a boost factor slightly higher than 2 to compensate for the voltage drop. This leads to a slightly lower-than-expected output current.

Besides logging key performance parameters, several signals were visualised using an oscilloscope. A screenshot of these measurements is presented in Figure 31.

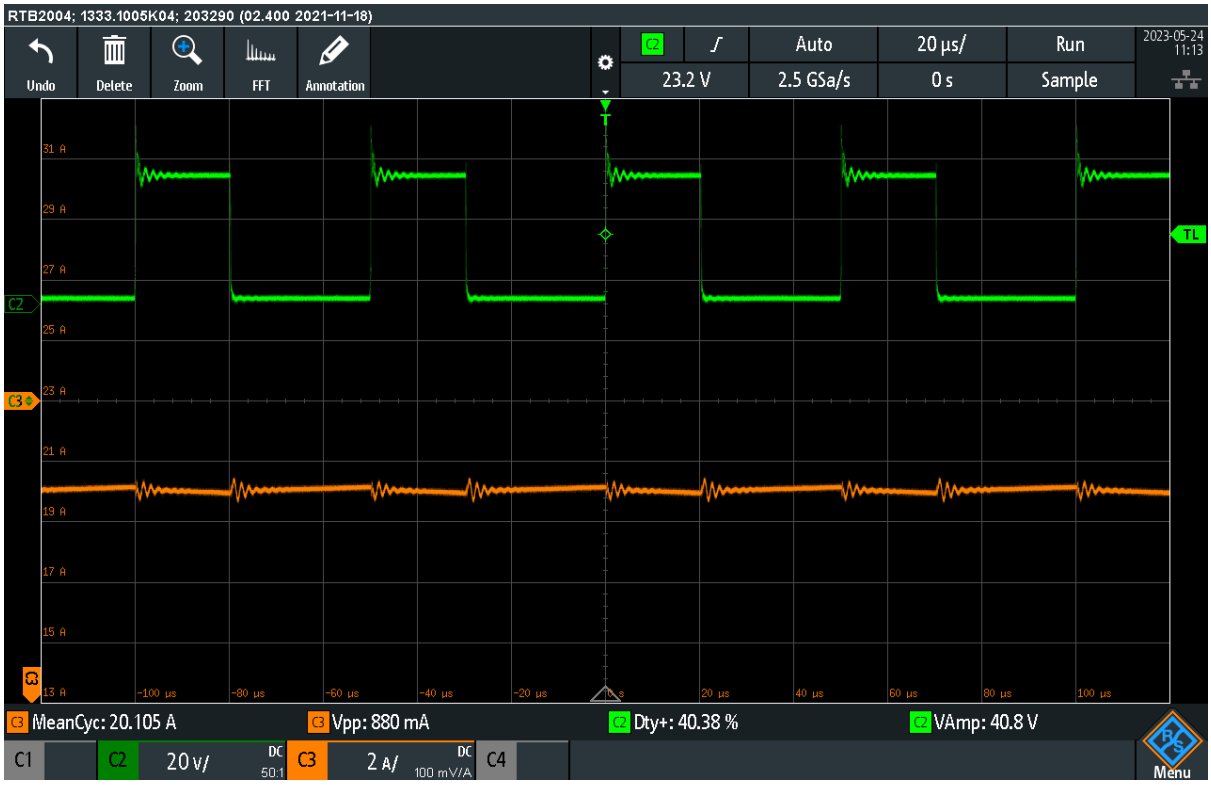

Figure 31: Oscilloscope screenshot during boost converter steady state operation

The voltage over the bottom IGBT is presented in green. The inductor current is presented in orange. When the IGBT is conducting, the voltage over it becomes 0. When the bottom IGBT is not conducting, the voltage over it is equal to the output voltage. This means that the voltage signal on the oscilloscope is the inverse of the control signal going into the bottom switch input. In other words, the voltage signal in this image corresponds to the PWM signal for the top IGBT. Therefore, the duty cycle presented at the bottom of the screen is the duty cycle of the top switch.

The inductor current appears to be mostly flat, bar from some oscillations when the switches operate. The small size of the ripple in the current can be explained by the fact that the inductor used in the setup has a significant inductance. When the inductor is charging its magnetic field, the current through it increases slightly. When the inductor is using its magnetic field to boost the voltage, the current decreases slightly.

The above logic can also be matched with the lower IGBT voltage pattern. When the lower switch is opened (high voltage), the converter is boosting current towards the output and thus the current in the inductor will be decreasing. When the lower switch is closed (low voltage), the inductor is being charged from the low side voltage source and thus the current will increase slightly.

Next, this exercise can be repeated but in buck mode. Table 11 shows the results.

Table 11: System parameters during buck converter steady state operation

Symbol	Result	Description
V_{in}	$20.0V$	Input voltage
V_{out}	$40.0V$	Output voltage
U_{sw}	$39.6V$	Switch voltage amplitude
I_{in}	$-20.0A$	Input current
I_{out}	$12.1A$	Output current
$I_{L,avg}$	$-19.8A$	Inductor current, mean
DC	59.3%	Duty cycle (top switch)

The voltage levels remain unchanged. Whilst the input current has only been negated, the output current has changed significantly. This can be explained easily. When the Delfino regulates current, it is trying to regulate the inductor current. Since the inductor is located on the input side of the power stack, the input current and inductor current are practically the same. In the first test, the Delfino regulated the input current to 20A. After accounting for the voltage drop over the IGBTs, only 8A out of the expected 10A arrived at the output. In the second experiment, the output has to provide the extra current to compensate for the losses so that the input receives exactly 20A.

An oscilloscope screenshot was also taken from this test. It is presented in Figure 32.

Figure 32: Oscilloscope screenshot during buck converter steady state operation

The signals visible in the image are the same as in the previous screenshot. The signals remain largely unchanged, with the only major difference being the duty cycle. Now, the bottom switch is open the majority of the time. In practice, this means that the current flow will reverse.

5.7 Current Control Step Response Testing

Next, the controller's ability to rapidly bring the system to a new setpoint was tested. This was done by first bringing the system to a stable operating state, and then manually changing the current setpoint. This test is repeated three times with different values. Table 12 presents a summary of the key system parameters during the first test.

Table 12: System parameters during the first step response test

Symbol	Result	Description
V_{in}	$20.0V$	Input voltage
V_{out}	$40.0V$	Output voltage
$I_{in,1}$	$10.2A$	Input current, start
$I_{in,2}$	$20.2A$	Input current, end
$I_{out,1}$	$-4.4A$	Output current, start
$I_{out,2}$	$-8.1A$	Output current, end
$I_{63\%}$	$16.3A$	63% level
τ	$13.5\ ms$	Time constant

This test started at a stable level of 10A, followed by an immediate step to 20A. At a current of 16.3A, 63% of the step is completed. The time it takes to reach this level is an important indicator of speed. This time was recorded at 13.5 milliseconds using an oscilloscope. Figure 33 is a screenshot of this measurement.

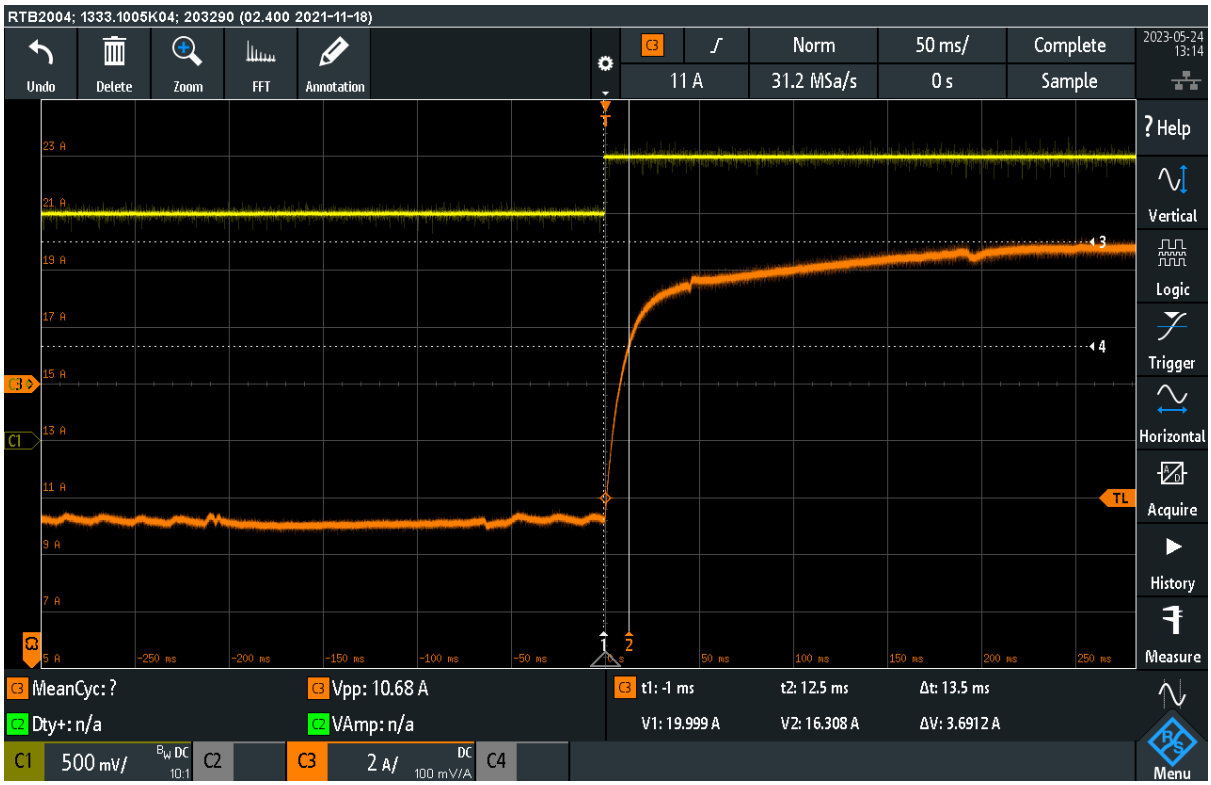

Figure 33: Oscilloscope screenshot during the first step response test

The measured inductor current is plotted in orange. The setpoint for the current is indicated in yellow. Note that the current setpoint is not drawn to scale.

The inductor current is stable at 10A until the setpoint is adjusted. The adjustment is visible through the step in the yellow line. This moment is also indicated by the position of vertical cursor 1. Once the step is initiated, the current increases rapidly until it settles at 20A, indicated by horizontal cursor 3. The 63% level is marked by horizontal cursor 4. The moment the current reaches this level is marked by vertical cursor 2. The time difference between the two vertical cursors is recorded at 13.5 milliseconds.

Several transient effects are visible in the inductor current. One is visible at 50 milliseconds, while another is present at 200 milliseconds. These transients were not studied as they are not the essence of this test.

Next, the test was repeated in buck converter mode. Table 13 provides a summary of the results.

Table 13: System parameters during the second step response test

Symbol	Result	Description
V_{in}	$20.0V$	Input voltage
V_{out}	$40.0V$	Output voltage
$I_{in,1}$	$-9.9A$	Input current, start
$I_{in,2}$	$-20A$	Input current, end
$I_{out,1}$	$5.6A$	Output current, start
$I_{out,2}$	$12.0A$	Output current, end
$I_{63\%}$	$-16.3A$	63% level
τ	$16.5\ ms$	Time constant

This time around, it took the system 16.5 milliseconds to reach the 63% level. This step response was also recorded and is visible in Figure 34

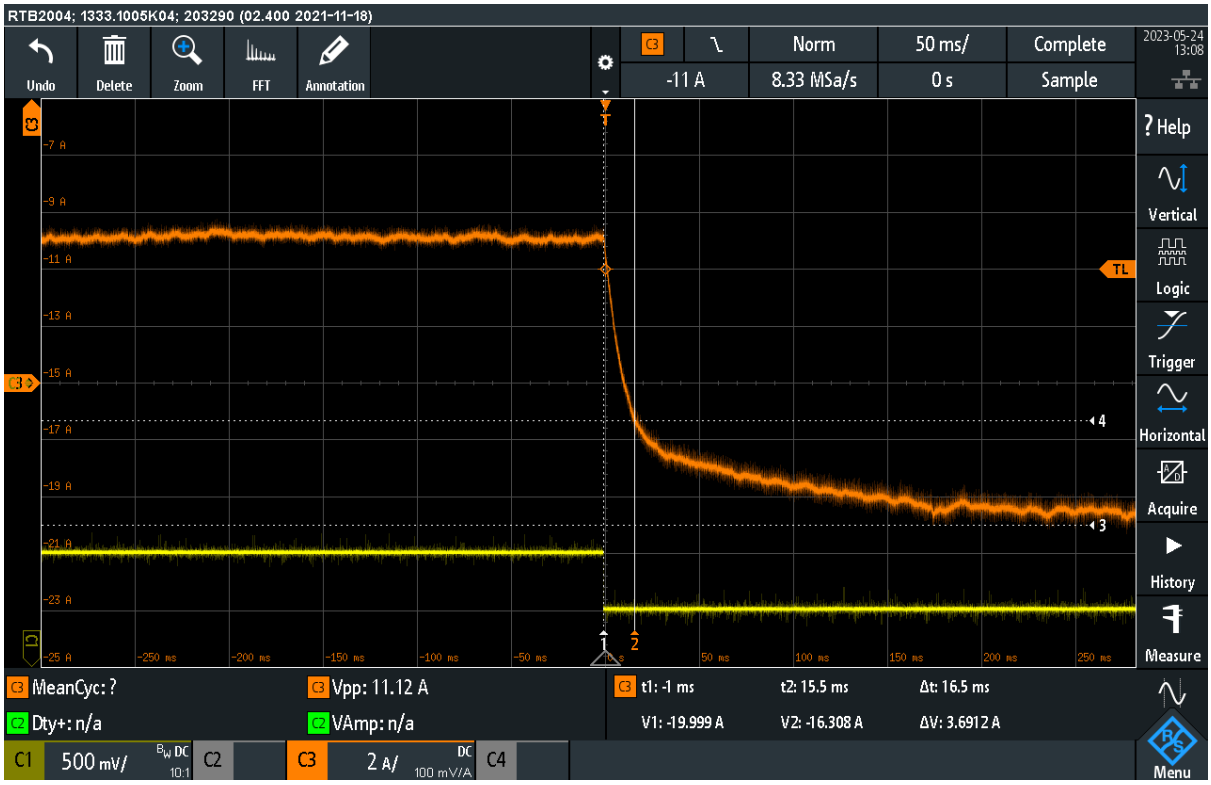

Figure 34: Oscilloscope screenshot during the second step response test

The composition of this screenshot is the same as in the previous step response test. This test provided similar results compared to the previous test, with only a 3 millisecond increase in time constant.

Table 14 presents the key results from the last step response test.

Table 14: System parameters during the third step response test

Symbol	Result	Description
V_{in}	$20.0V$	Input voltage
V_{out}	$40.0V$	Output voltage
$I_{in,1}$	$-20.0A$	Input current, start
$I_{in,2}$	$20.2A$	Input current, end
$I_{out,1}$	$12.0A$	Output current, start
$I_{out,2}$	$-8.1A$	Output current, end
$I_{63\%}$	$5.2A$	63% level
τ	$101\ ms$	Time constant

This test yielded a significantly larger time constant, at 101 milliseconds. The fourfold increase in step size should not influence the time constant of the system. To understand this better, the step response must be examined in detail using an oscilloscope. Figure 35 shows this screenshot.

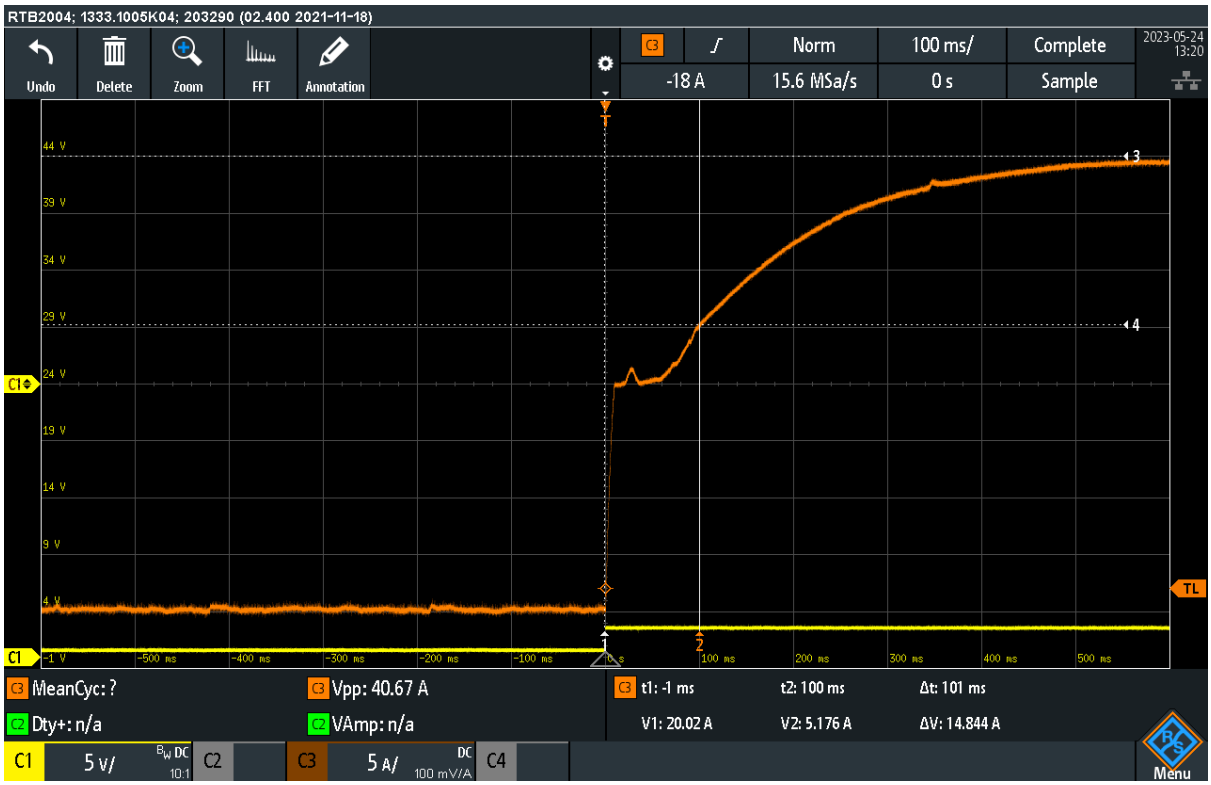

Figure 35: Oscilloscope screenshot during the third step response test

The composition of this screenshot is again identical to the previous ones. However, the shape of the current has changed significantly. The step response appears normal until the current comes close to 0. Only after a short pause at 0 does the current restart the normal response pattern. It is visible from the image that this phenomenon causes a significant delay in the step response time. if the current had completed the step response with the same swiftness as the initial rise, it might have achieved a time constant of similar proportion as the previous two tests.

While this effect is not researched in depth, a possible explanation can be provided as follows. Normally this converter operates in continuous conduction mode, or CCM. This means that the

inductor current, whilst it does oscillate as in Figure 16, does not ever becomes 0 during a switching cycle. In other words, the inductor has enough magnetic energy stored to supply the output throughout the entire cycle.

However, at low operating currents, the inductor might not have enough magnetic energy stored. This results in the inductor current becoming equal to zero for part of the switching cycle. While the power converter can still fulfil its role in this operating mode, the system dynamics can be influenced significantly by this phenomenon. Since the delay in the step response originates from a short pause when the current is close to 0, it is proposed that the effects of DCM at least contribute significantly to this delay.

5.8 Current Control Step Response Testing with Varying Control Loop Periods

The primary reason for choosing the digital PI control scheme presented in 4.5 is that it is designed to mitigate effects caused by differences in controller speed. The scheme compensates the PI parameters to decouple the controller response time from the control loop period. Several step responses with differing control loop periods were recorded to test this statement. Table 15 summarises the results from the four tests, while Table 16 explains the used symbols.

Table 15: System parameters during step response testing with varying control loop periods

Symbol	Result 1	Result 2	Result 3	Result 4
V_{in}	$20.0V$	$20.0V$	$20.0V$	$20.0V$
V_{out}	$40.0V$	$40.0V$	$40.0V$	$40.0V$
$T_{control\ loop}$	$1300\ \mu s$	$130\ \mu s$	$13\ \mu s$	$1.3\ \mu s$
$I_{in,1}$	$5.2A$	$5.2A$	$5.2A$	$5.2A$
$I_{in,2}$	$25.1A$	$25.1A$	$25.1A$	$25.1A$
$I_{out,1}$	$-2.3A$	$-2.3A$	$-2.3A$	$-2.3A$
$I_{out,2}$	$-9.6A$	$-9.6A$	$-9.6A$	$-9.6A$
$I_{63\%}$	$17.6A$	$17.6A$	$17.6A$	$17.6A$
τ	$17\ ms$	$15\ ms$	$15\ ms$	$16\ ms$

Table 16: System parameters during step response testing with varying control loop periods explained

Symbol	Description
V_{in}	Input voltage
V_{out}	Output voltage
$T_{control\ loop}$	Control loop period
$I_{in,1}$	Input current, start
$I_{in,2}$	Input current, end
$I_{out,1}$	Output current, start
$I_{out,2}$	Output current, end
$I_{63\%}$	63% level
τ	Time constant

The step response size and starting point were kept constant during all four tests. The control loop period was varied between 1300 microseconds and 1.3 microseconds with factor 10 step sizes. The

resulting time constants varied by no more than 2 milliseconds. They were recorded the same way using an oscilloscope. Figure 36 through Figure 39 show these results.

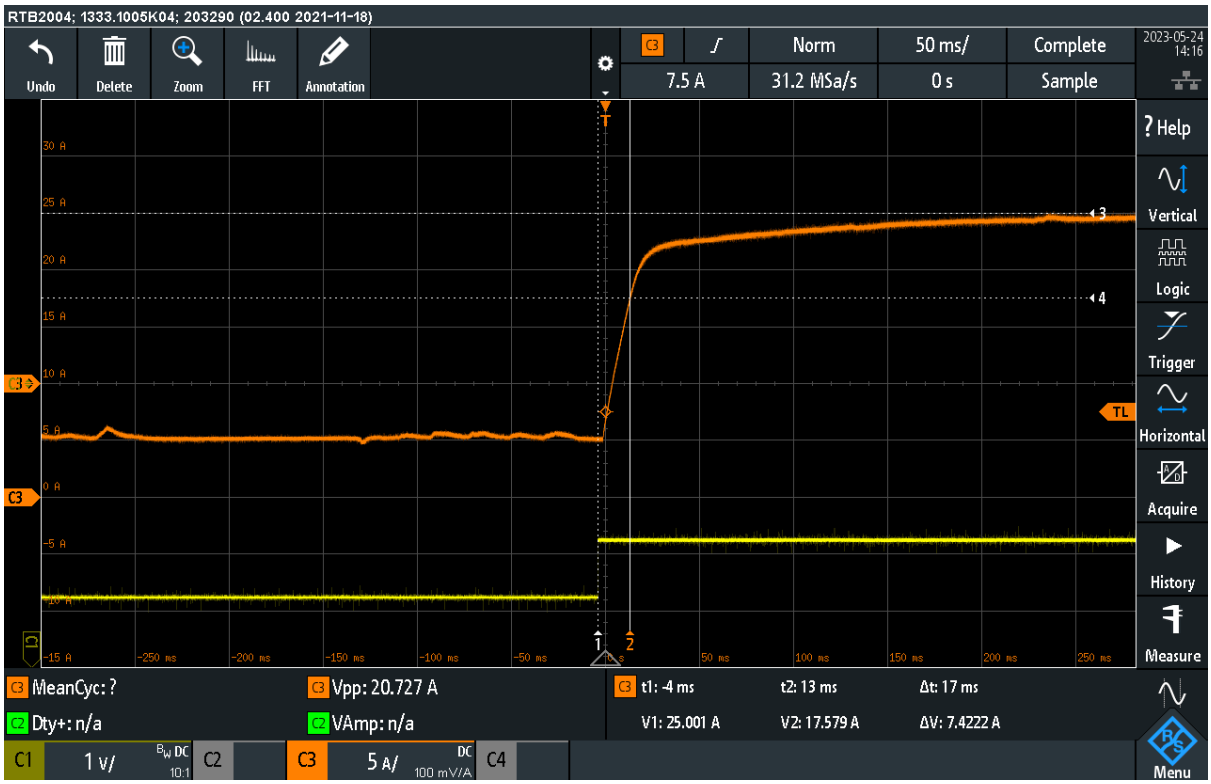

Figure 36: Oscilloscope screenshot during step response testing with the first control loop period

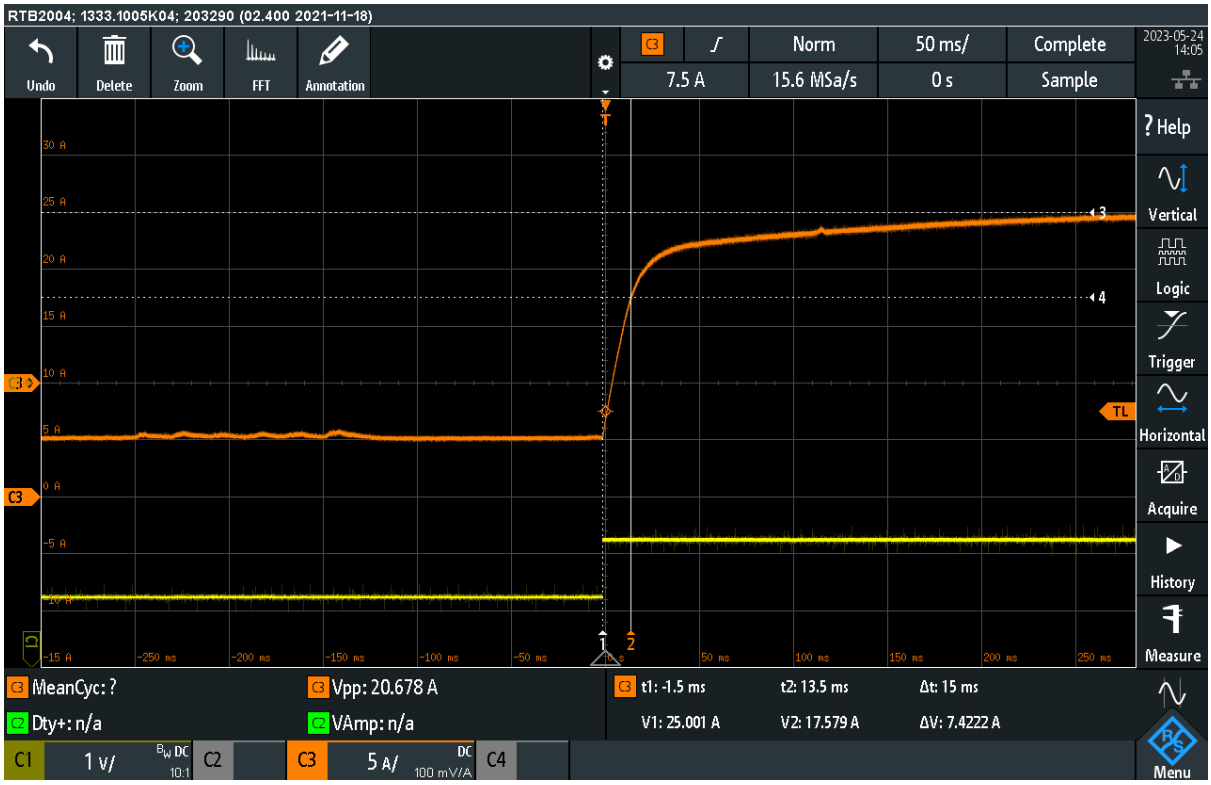

Figure 37: Oscilloscope screenshot during step response testing with the second control loop period

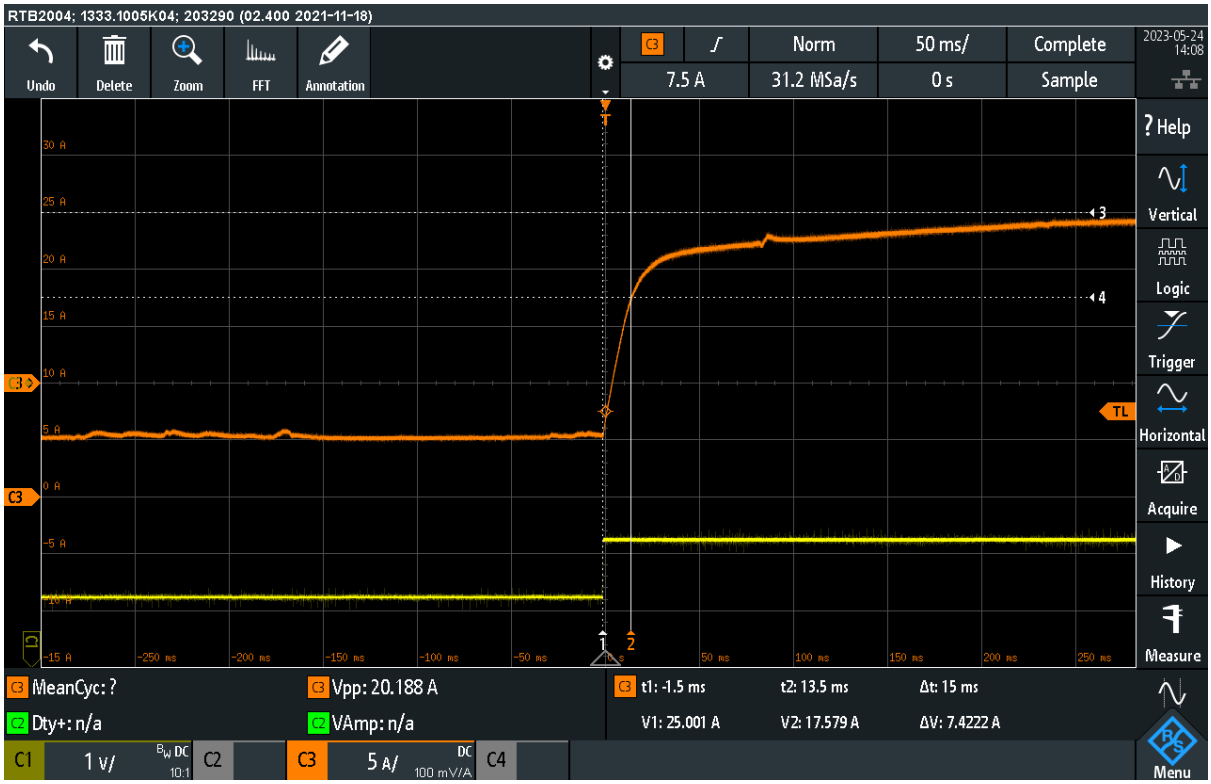

Figure 38: Oscilloscope screenshot during step response testing with the third control loop period

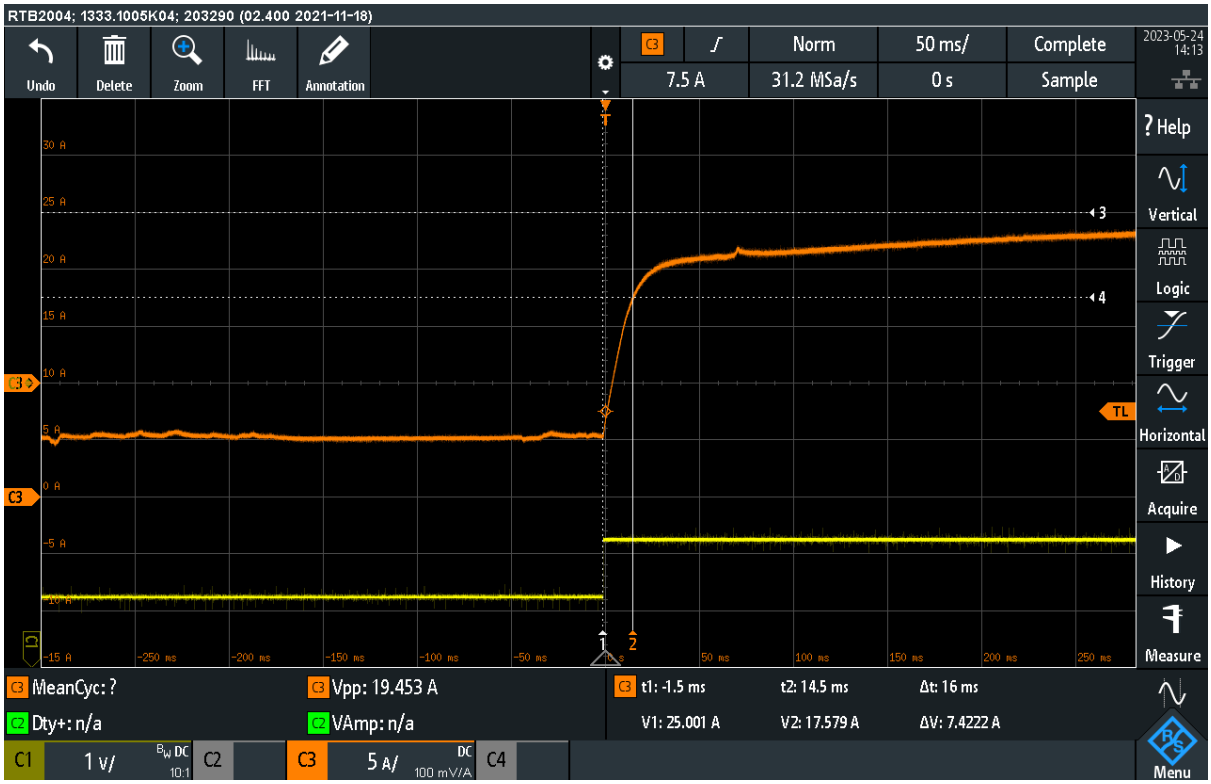

Figure 39: Oscilloscope screenshot during step response testing with the fourth control loop period

These images are composed similarly to previous tests. The images show the current initially idle at 5A. The initial section of the step response does not differ largely between the four images, hence the time constants are similar in all cases.

The only larger difference present between the responses is the settling time. This is visible in the tail of the response, more at the right on the screenshots. In Figure 36, the inductor current manages to reach the target towards the end of the oscilloscope screenshot. In Figure 39 however, there still is a gap between the inductor current and its target. Generally speaking, the shorter the control loop period, the longer it takes for the integrator to settle the current towards the intended target.

5.9 Current Control Ramp Response Testing

The final test to evaluate the current control performance is a ramp response test. In this test, the current will be changed gradually between two levels. The purpose of this test is to evaluate the lag between the current setpoint and the actual current. Table 17 shows the test conditions and results.

Table 17: System parameters during ramp response testing

Symbol	Result	Description
V_{in}	$20.0V$	Input voltage
V_{out}	$40.0V$	Output voltage
I_1	$2A$	Ramp current, min
I_2	$28A$	Ramp current, max
m	$49.6\ A/s$	Ramp current, angle
e_{dyn}	$1.66A$	Dynamic error, vertical
e_{dyn}	$33\ ms$	Dynamic error, horizontal

The test will change the current setpoint from 2A to 28A at a predetermined rate of 50A/s. In reality, a rate of 49.6A/s was recorded. The resulting dynamic error, visible in Figure 40, was recorded at 1.66A vertically and 33 milliseconds horizontally.

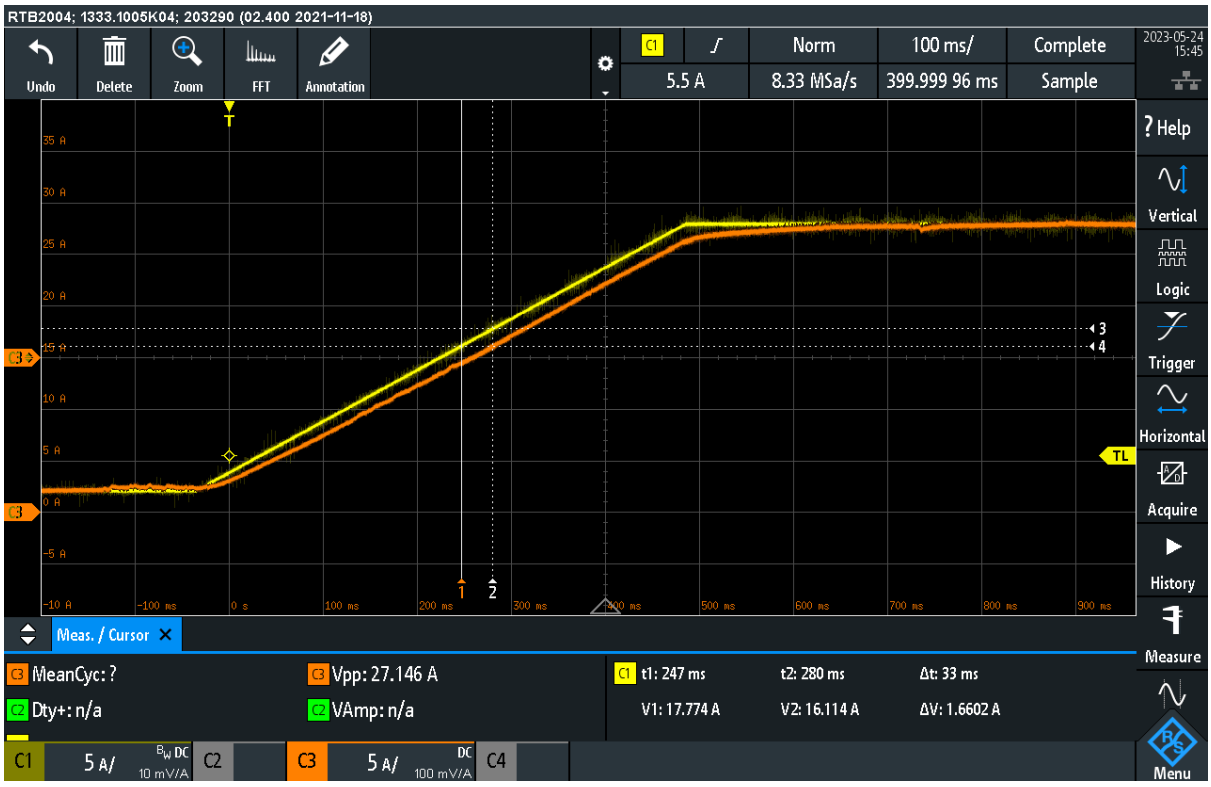

Figure 40: Oscilloscope screenshot during ramp response testing

The signals presented in this screenshot along with their colours remain unchanged. However, this time the recorded setpoint was scaled to match the measured current.

The measured current and desired current are initially both steady at 2A. Once the manual trigger for the ramp had been given, the setpoint rose to 28A at a rate of 49.6A/s. After an initial delay, the measured current follows the ramp and eventually settles at a fixed lag. Once the setpoint current flattens, the measured current quickly settles at the desired output current.

The intersection of cursor 2 and the setpoint current and measured current are indicated by horizontal cursors 3 and 4 respectively. The difference in height between cursors 3 and 4 indicates the vertical dynamic error. This exercise can be repeated horizontally, leading to the horizontal dynamic error.

5.10 Voltage Control Testing

In the final round of testing, the voltage source at the output of the converter is replaced by a resistor. Since the output voltage is no longer managed by the voltage source, an extra control loop needs to be added. This test will evaluate the voltage control loop's ability to compensate for unknown external disturbances. This test will be conducted by starting the system with no load attached and then suddenly connecting the resistor to resemble a rapid and unexpected external disturbance. Table 18 summarises the results.

Table 18: System parameters during voltage step response testing

Symbol	Amount	Description
V_{in}	$20.0V$	Input voltage
V_{out}	$40.0V$	Output voltage
$I_{in,1}$	$0A$	Input current, start
$I_{in,2}$	$11.4A$	Input current, end
$I_{out,1}$	$0A$	Output current, start
$I_{out,2}$	$4.6A$	Output current, end
$\Delta V_{out,max}$	$20V$	Max output voltage drop
$I_{out,max}$	$11.4A$	Max current used
$I_{out,set,max}$	$\sim 14A$	Max current demanded

The current flowing in the system was initially zero. The load drew a current of 4.6A after settling, needing an input current of 11.4A. The maximum output voltage drop was recorded at 20V. The maximum current drawn from the input was no greater than the end input current, indicating that the actual current did not have an overshoot. The current setpoint did record an overshoot, maximising at approximately 14A. Figure 41 and Figure 42 show the oscilloscope screenshots of this measurement.

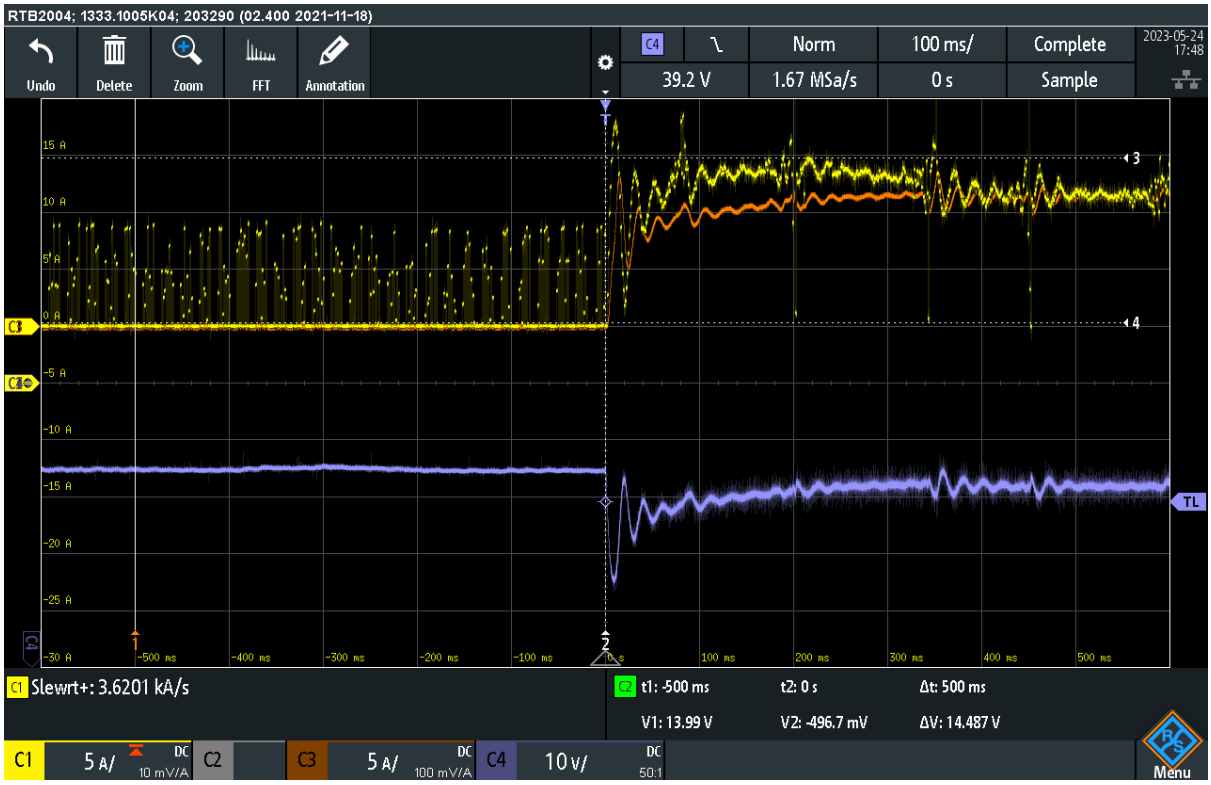

Figure 41: Oscilloscope screenshot during voltage step response testing (1)

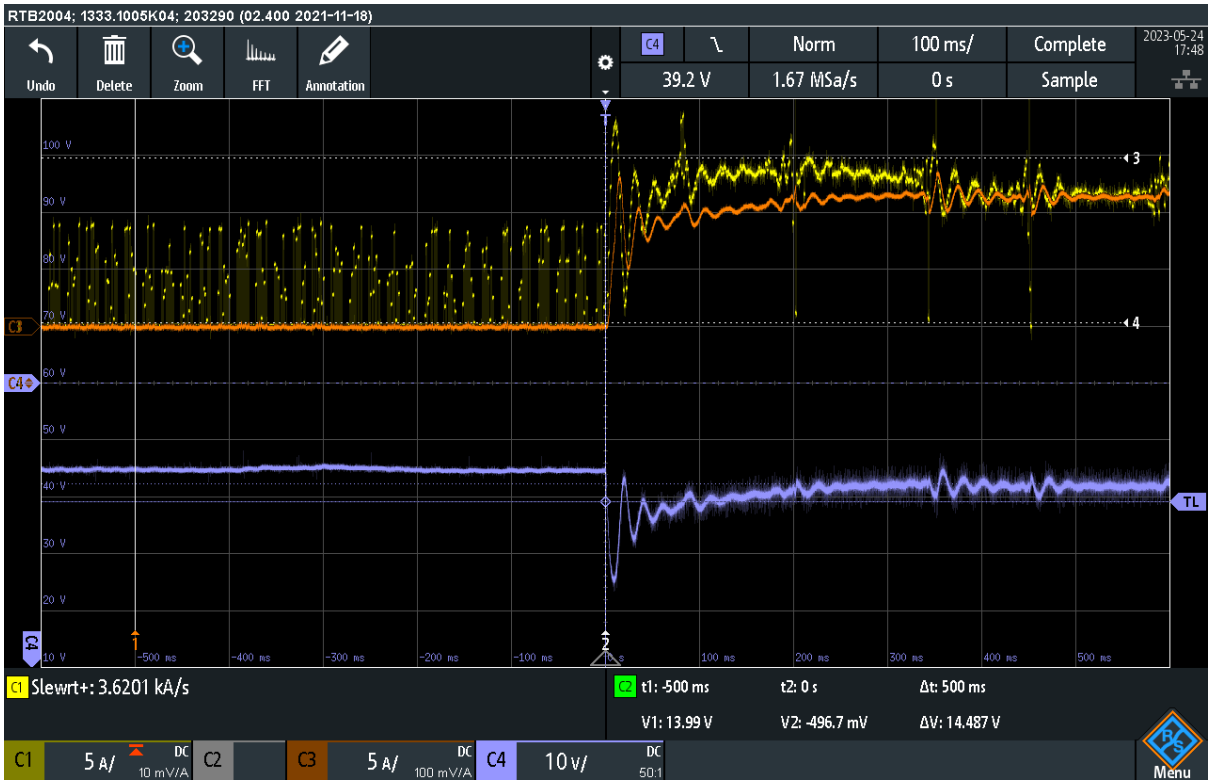

Figure 42: Oscilloscope screenshot during voltage step response testing (2)

The measured current and the setpoint current are still indicated in orange and yellow respectively. The output voltage has now been added to the image in blue. Figure 41 and Figure 42 are completely identical apart from the labels on the vertical axis.

Initially, the currents and voltage are stable. The currents were both 0, while the voltage was stable slightly above the 40V setpoint. It is not known why the system failed to achieve its target voltage with no load attached. Testing on this was not performed due to time constraints.

While the measured current stays neatly around the zero mark, the setpoint current indicates large oscillations. It is reiterated that this is not a direct digital export of the setpoint. Rather, a reference voltage is generated using a DAC that is measured by an oscilloscope. It is proposed that this process is at least significantly responsible for generating this noise.

Before the attachment of the load, the current setpoint is hovering around the zero mark. This means that the current setpoint periodically becomes negative. However, the DAC used was unable to generate negative voltages. It is possible that loading a negative 16-bit value into the unsigned, 12-bit register of the DAC leads to unexpected results.

The attachment of the load is visible by the sudden rise in current and corresponding drop in voltage in the middle of the image. The voltage drops steeply initially but begins to slowly recover thereafter. Several oscillations are visible on the voltage.

The voltage drop is paired with a sudden increase in the current setpoint. However, the actual current does not follow immediately, allowing for the voltage to drop significantly. While the setpoint current clearly shows an overshoot, the actual current fails to follow this and instead behaves like a first-order system.

A voltage drop of 50% is extremely large and unacceptable in most applications. However, the purpose of this test was to demonstrate proof of concept rather than to present the most optimally tuned regulator. To reduce the voltage drop, the response time of the controller has to be reduced, or the inertia of the system has to be increased. The latter can be achieved easily by adding more capacitors to the output capacitor bank. The former, however, is harder to achieve. The challenge lies in obtaining an accurate and precise measurement in the controller. The measurement in this setup was not fine-tuned due to time constraints, leading to significant noise on the Delfino's measurements. The PI parameters need to be slowed down to ensure that the controller does not excessively react to incorrect measurements.

If the stability of the measurements were to be improved, it would be possible to tune the parameters to make the controller quicker. Also, some differentiating action could be used. In its current form, the PI scheme supports the usage of a differentiator. This was briefly tested, but ultimately not included in this thesis because the noisy measurements caused the differentiator to make the system unstable.

6 Conclusion

This thesis was situated in the field of power electronics research. Embedded control is required to test novel power electronics topologies or applications. Developing an embedded program to test a hardware setup is arduous and time-consuming due to the steep learning curve associated with embedded programming. This thesis aimed to mitigate some of these issues.

Firstly, a short study comparing different market-available embedded platforms was performed. This study compared several key performance indicators of each platform. This included aspects like the availability of GPIO, the capability of ADC peripherals et cetera. This comparison concluded that no platform could yield a significant advantage over another. Therefore, the choice was made to use the Texas Instruments Delfino, a platform that was already available in-house.

After choosing an appropriate platform, it was then necessary to evaluate the challenges posed by this specific platform. It was determined that the mitigation of project and hardware configuration aspects, combined with the creation of commonly-needed utility functions would help reduce programming time and ease the learning for novice users. A project template was created to eliminate most of the general configuration work. Instead, users can now start directly with writing software for their setup. The template provides the user with a predetermined project structure. The coding process was accelerated further by eliminating repetitive work through the creation of utility functions. A quick start guide was written to explain the purpose and usage of the template and utility functions.

Lastly, a thorough evaluation of the work was needed. This was achieved by building a hardware setup. This provided a realistic scenario for which an embedded project was needed. Several additional improvements could be made to the template and utility functions while developing a program for the hardware test setup. Furthermore, the hardware setup allowed for a performance evaluation of the utility functions related to PI(D) control.

A bidirectional buck-boost converter was used for the test setup. This topology was chosen for its relative simplicity and bidirectional nature. The Semiteach, a power stack developed by Semikron, was used as the cornerstone of the setup. This choice was made because the Semiteach was available in-house. A level shifter was needed, Due to the difference in logic voltage levels between the Delfino and the Semiteach. Several tests were performed using two voltage sources in current control mode, and one additional test using one voltage source and a resistive load was performed in voltage control mode.

The current control testing can be divided into four categories. First, several experiments in steady-state conditions were conducted. These tests evaluated the PI controller's ability to keep the converter stable. Secondly, the system's time response behaviour was evaluated by means of several step responses. The system behaved like a first-order system and was able to attain a new balance state autonomously. The influence of operation in discontinuous conduction mode was visible in these tests. Thirdly, the system's response to a ramping input signal was recorded. As expected, the system followed the input ramp with a constant following error. Lastly, the step response tests in current control mode were repeated with varying digital control loop periods. This was done to investigate the influence of the control loop period on the controller's speed and accuracy. It was concluded that the influence was minimal.

Finally, a voltage control loop was added. This allowed for one of the voltage sources to be replaced with a linear resistor. This provided a realistic test case, serving as a proof of concept. One step response, whereby a certain load was suddenly connected to the converter, was recorded. The system was able to regain its set output voltage after a small voltage dip.

Several suggestions for future work are scattered throughout this thesis. On the one hand, these include suggestions for improving the utility functions. For instance, a different design approach to some of the timing functions or improvements to the PID control scheme are possible. On the other hand, improvements related to the operation and control of hardware setups are possible. This mainly includes ways to increase the accuracy and stability of measurements.

References

[1] Imperix, ‘B-Board PRO Embeddable Controller’, Apr. 21, 2020. https://imperix.com/wp-content/uploads/document/B-Board_Datasheet.pdf (accessed Dec. 07, 2022).

[2] Taraz Technologies, ‘PE-RCP Box’, Apr. 10, 2022. https://www.taraztechnologies.com/Downloads/Datasheets/PE-RCP_Box.pdf (accessed Dec. 07, 2022).

[3] Taraz Technologies, ‘PEController’, Apr. 10, 2022. https://imperix.com/wp-content/uploads/document/B-Board_Datasheet.pdf (accessed Dec. 07, 2022).

[4] Taraz Technologies, ‘PE-RCP’, Apr. 10, 2022. https://www.taraztechnologies.com/Downloads/Datasheets/PE-RCP.pdf (accessed Dec. 07, 2022).

[5] National Instruments, ‘CompactRIO Systems’. https://www.ni.com/nl-be/shop/compactrio.html (accessed Dec. 07, 2022).

[6] Texas Instruments, ‘TMS320F2837xD Dual-Core Microcontrollers’, Feb. 2021. https://www.ti.com/lit/ds/symlink/tms320f28379d.pdf?ts=1667880276437&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTMS320F28379D (accessed Dec. 07, 2022).

[7] Texas Instruments, ‘TMDSHSECDOCK HSEC180 controlCARD baseboard docking station’. https://www.ti.com/tool/TMDSHSECDOCK (accessed Dec. 07, 2022).

[8] Texas Instruments, ‘TMDSDOCK28379D F28379D Delfino Experimenter Kit’. https://www.ti.com/product/TMDSDOCK28379D/part-details/TMDSDOCK28379D (accessed Dec. 07, 2022).

[9] SEMIKRON, ‘Semiteach IGBT’, Sep. 01, 2015. https://www.thierry-lequeu.fr/data/SEMITEACH-IGBT-datasheet-2019.pdf (accessed Apr. 20, 2023).

[10] Texas Instruments, ‘TMS320F2837xD Dual-Core Microcontrollers Technical Reference Manual’, Sep. 2019. https://www.ti.com/lit/ug/spruhm8i/spruhm8i.pdf (accessed Apr. 20, 2023).

[11] Texas Instruments, ‘Code Composer Studio Integrated Development Environment’. https://www.ti.com/tool/CCSTUDIO (accessed Apr. 20, 2023).

[12] Texas Instruments, ‘C2000WARE’. https://www.ti.com/tool/C2000WARE (accessed May 10, 2023).

[13] The Mathworks, ‘Embedded Coder’. https://nl.mathworks.com/products/embedded-coder.html (accessed May 10, 2023).

[14] The Mathworks, ‘Simulink Coder’. https://nl.mathworks.com/products/simulink-coder.html (accessed May 10, 2023).

[15] SEMIKRON, ‘SKHI 21A (R) Hybrid Dual MOSFET Driver’. https://www.semikron-danfoss.com/dl/service-support/downloads/download/semikron-datasheet-skhi-22-a-b-h4-r-l5012522.pdf (accessed May 15, 2023).

[16] Arduino, ‘Language Reference’. https://www.arduino.cc/reference/en/ (accessed May 02, 2023).

[17] Arduino, ‘Setup()’. https://www.arduino.cc/reference/en/language/structure/sketch/setup/ (accessed May 02, 2023).

[18] Arduino, ‘Loop()’. https://www.arduino.cc/reference/en/language/structure/sketch/loop/ (accessed May 02, 2023).

[19] J. Baeten, ‘Labview en Digitale Controle - Een korte inleiding tot digitale controle’. Faculty of Engineering Technology KU Leuven - UHasselt.

[20] SEMIKRON, ‘SKM50GB12T4 Fast IGBT4 modules’. https://www.semikron-danfoss.com/dl/service-support/downloads/download/semikron-datasheet-skm50gb12t4-22892000.pdf (accessed May 15, 2023).

[21] Testec, ‘Current probes’. https://www.testec.de/en/products/product-categories/current-probes/ (accessed Jun. 09, 2023).

[22] Testec, ‘Differential probes’. https://www.testec.de/en/products/product-categories/differrential-probes/ (accessed Jun. 09, 2023).

[23] National Instruments, ‘NI-9401 Datasheet’, Jun. 10, 2022. https://www.ni.com/docs/en-US/bundle/ni-9401-specs/page/specs.html (accessed Dec. 07, 2022).

[24] National Instruments, ‘NI 9402 Specifications’, Dec. 18, 2021. https://www.ni.com/docs/en-US/bundle/ni-9402-specs/page/specifications.html (accessed Dec. 07, 2022).

[25] National Instruments, ‘NI-9403 Datasheet’, Jul. 14, 2022. https://www.ni.com/docs/en-US/bundle/ni-9403-specs/page/specs.html (accessed Dec. 07, 2022).

[26] National Instruments, ‘NI 9218 Specifications’, Dec. 19, 2021. https://www.ni.com/docs/en-US/bundle/ni-9218-specs/page/specs.html (accessed Dec. 07, 2022).

[27] National Instruments, ‘NI-9381 Specifications’, Dec. 19, 2021. https://www.ni.com/docs/en-US/bundle/ni-9381-specs/page/specs.html (accessed Dec. 07, 2022).

[28] National Instruments, ‘C Series LIN Interface Module’. https://www.ni.com/nl-be/shop/hardware/products/c-series-lin-interface-module.html (accessed Dec. 08, 2022).

[29] National Instruments, ‘C Series Serial Interface Module’. https://www.ni.com/nl-be/shop/hardware/products/c-series-serial-interface-module.html (accessed Dec. 08, 2022).

[30] National Instruments, ‘C Series CAN Interface Module’. https://www.ni.com/nl-be/shop/hardware/products/c-series-can-interface-module.html (accessed Dec. 08, 2022).

Attachments

Table 19: Information summary on PWM outputs

	Imperix B-Board [1]	**Taraz PE-RCP Box** [2]	**Taraz PEController** [3]	**Taraz PE-RCP** [4]	**Compact RIO (NI 9401)** [23]	**Compact RIO (NI 9402)** [24]	**Compact RIO (NI 9403)** [25]	**TMS320F2837xD** [6]
Amount	32	16	24	24	8 (Configurable)	4 (Configurable)	32 (Configurable)	40
Voltage [V]	3.3 (PWM 0-15) 1.8 (PWM 16-31)	5	3.3/5	3.3/5.0	5	5	5	3.3
Resolution [ns]	1.6 (PWM 0-15) 2 (PWM 16-31)				10	25		25 (24) 0.150 (16)
Remark		Same pins as DOs	Pins shared with DO/DIO	Pins shared with DO/DIO				

Table 20: Information summary on analogue inputs (AIs)

	Imperix B-Board [1]	**Taraz PE-RCP Box** [2]	**Taraz PEController** [3]	**Taraz PE-RCP** [4]	**Compact RIO (NI 9218)** [26]	**Compact RIO (NI 9381)** [27]	**TMS320F2837xD** [6]
Amount	8	16	16	16	2	8	24 (4 channels)
Bits	16	16 (x8) 12 (x8)	16	16 (x8) 12 (x8)	24	12	12 or 16
Voltage Range [V]	5.0	10.0	10.0	10.0	16.0 or 60.0	5.0	Configurable
Sample Rate [Ms/s]	Between 1.0 and 2.0	0.36 (16-bit) 0.43 (12-bit)	0.250	0.36 (16-bit) 0.43 (12-bit)	0.051	0.02	1.1 (16-bit) 3.5 (12-bit)
Remark		Higher SR when fewer inputs used simultaneously: 1.1 Ms/s (16-bit) 3.5 Ms/s (12-bit)	Higher SR when fewer used inputs simultaneously: 1.0 Ms/s (16-bit)	Higher SR when fewer used inputs simultaneously: 1.1 Ms/s (16-bit) 3.5 Ms/s (12-bit)			

Table 21: Information summary on GPIOs

	Imperix B-Board [1]	**Taraz PE-RCP Box** [2]	**Taraz PEController** [3]	**Taraz PE-RCP** [4]	**Compact RIO (NI 9381)** [27]	**TMS320F2837xD** [6]
Amount in	16	16	17	18		
Voltage [V]	3.3	3.3/5.0	3.3/5.0	3.3/5.0		
Remarks						
Amount out	16	16	16	18		
Voltage [V]	3.3	5.0	3.3/5.0	3.3/5.0		
Remarks		Pins shared with PWM	Pins shared with PWM	Pins shared with ePWM		
Amount in/out	36		8	8	4	169
Voltage [V]	3.3		3.3/5.0	3.3	5.0	3.3
Remarks			Pins shared with PWM	Pins shared with ePWM		

Table 22: Enumeration of communication interfaces & protocols supported by the controllers

	Imperix B-Board [1]	**Taraz PE-RCP Box** [2]	**Taraz PEController** [3]	**Taraz PE-RCP** [4]	**Compact RIO** [28]–[30]	**TMS320F2837xD** [6]	
SPI	■		■	■		■	SPI
Fibre Optics		■	■	■			Fibre Optics
USB	■	■	■	■		■	USB
Ethernet	■		■				Ethernet
SFP+	■						SFP+
Sync		■	■	■			Sync
SD card		■	■	■			SD card
CAN		■	■	■	■	■	CAN
Serial					■		Serial
LIN					■		LIN
I2C			■	■		■	I2C
UART			■			■	UART
McBSP						■	McBSP
uPP						■	uPP