Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

GIJÓN POLYTECHNIC SCHOOL OF ENGINEERING

BACHELOR'S DEGREE IN INDUSTRIAL AND AUTOMATIC ELECTRONICS ENGINEERING

SYSTEMS AND AUTOMATION ENGINEERING AREA

AND

ELECTRONIC TECHNOLOGY AREA

CONTROL OF POWER ELECTRONIC CONVERTERS FOR THE INTEGRATION OF DISTRIBUTED GENERATION AND ENERGY STORAGE IN DC AND AC MICROGRIDS

Marta Pérez Molinero

TUTOR: Ángel Navarro Rodríguez

COTUTOR: Ramy Georgious Zaher Georgious

FECHA: June 2023

ABSTRACT

The work done throughout this project consists of the development and simulation of several control systems for Power Electronic Converters (PEC) that allow the integration into the grid of Distributed Generation (DG) as well as Energy Storage Systems (ESS).

This is a very relevant issue today as the use of distributed energy generation systems, instead of the centralized generation used so far, is becoming increasingly widespread. The change is mainly due to the search for greater integration of clean energy sources and has been aggravated by the energy crisis we have been living in for the last few years. Nevertheless, the proper control of power converters combined with the introduction of EES has additional applications, like supporting the main grid or allowing the user to have his own self-consumption network.

Specifically, the control of a bidirectional DC/DC converter, known in the literature as a Synchronous Boost Converter (SBC) or Bidirectional Boost Converter (BBC), and a three-phase DC/AC converter, known as an inverter, is analyzed. Different simulations are carried out to study the behavior of different control loops and, in addition, the models developed are taken to a Hardware In The Loop (HIL) system with the intention of testing the designed controllers in an environment closer to reality.

RESUMEN

El trabajo realizado a lo largo de este proyecto consiste en el desarrollo y simulación de varios sistemas de control para Convertidores Electrónicos de Potencia (PEC) que permitan la integración en la red de Generación Distribuida (GD) así como de Sistemas de Almacenamiento de Energía (SSE).

Este es un tema muy relevante en la actualidad, ya que cada vez está más extendido el uso de sistemas de generación de energía distribuida, en lugar de la generación centralizada que se ha venido utilizando hasta ahora. El cambio se debe principalmente a la búsqueda de una mayor integración de fuentes de energía limpias y se ha visto agravado por la crisis energética que vivimos desde hace unos años. No obstante, el control adecuado de los convertidores de potencia combinado con la introducción de EES tiene aplicaciones adicionales, como dar soporte a la red principal o permitir al usuario disponer de su propia red de autoconsumo.

En concreto, se analiza el control de un convertidor DC/DC bidireccional, conocido en la literatura como Synchronous Boost Converter (SBC) o Bidirectional Boost Converter (BBC), y de un convertidor DC/AC trifásico, conocido como inversor. Se realizan diferentes simulaciones para estudiar el comportamiento de diferentes lazos de control y, además, los modelos desarrollados se llevan a un sistema Hardware In The Loop (HIL) con la intención de probar los controladores en un entorno más cercano a la realidad.

ACRONYMS

- RES: Renewable Energy Sources.
- ESS: Energy Storage System.
- PEC: Power Electronic Converter.
- DG: Distributed Generation.
- SBC: Synchronous Boost Converter.
- BBC: Bidirectional Boost Converter.
- DC: Direct Current.
- AC: Alternating current.
- MG: Microgrid.
- SG: Synchronous Generator.
- VSG: Virtual Synchronous Generator.
- HIL: Hardware-in-the-Loop.
- PI: Proportional Integral (controller)
- PWM: Pulse Width Modulation.
- Duty: Also known as ”duty cycle”. Relationship between the time during which a signal is in its active state and its period.
- PCC: Point of Common Coupling. Point in an electric system where a microgrid connects with the main grid.
- SRF: Synchronous Reference Frame.
- PLL: Phase Lock Loop.

- SRF-PLL: Phase Lock Loop method implemented in the Synchronous Reference Frame.
- DSRF-PLL: Double Phase Lock Loop.
- I/O: Input/Output (channels).
- MCU: Microcontroller Unit.
- TI: Texas Instruments.
- CCS: Code Composer Studio.
- ADC: Analog to Digital Converter.
- DI/DO: Digital Input/Output.
- RoCoF: Rate Of Change Of Frequency.
- IE: Inertia-Emulation.
- SV: Synchronverter.
- CCVSM: Current-controlled virtual synchronous machine.
- VCVSM: Voltage-controlled virtual synchronous machine.

Index

List of Figures

List of Tables

1. Introduction

1.1.- BACKGROUND AND MOTIVATION

In the energy and climate crisis environment in which we find ourselves nowadays, all technologies that facilitate reducing energy consumption are becoming increasingly relevant. In the case of energy production, distribution, and storage, which is the subject of this paper, microgrids (MG) with autonomous operation capabilities have been investigated intensively.

The importance of microgrids lies in the fact that they allow for the transition from a centralized power system to a distributed generation system. In distributed systems, unlike in traditional ones, it is possible to introduce renewable energy sources. This possibility of greater integration of clean energy has become even more necessary in recent years. Figure 1.1 shows the percentage of renewable energy production in the European Union in 2020. It can be seen that RES made up 37% of gross electricity consumption, but this percentage exceeds 50% for countries such as Austria, Sweden, or Denmark [1].

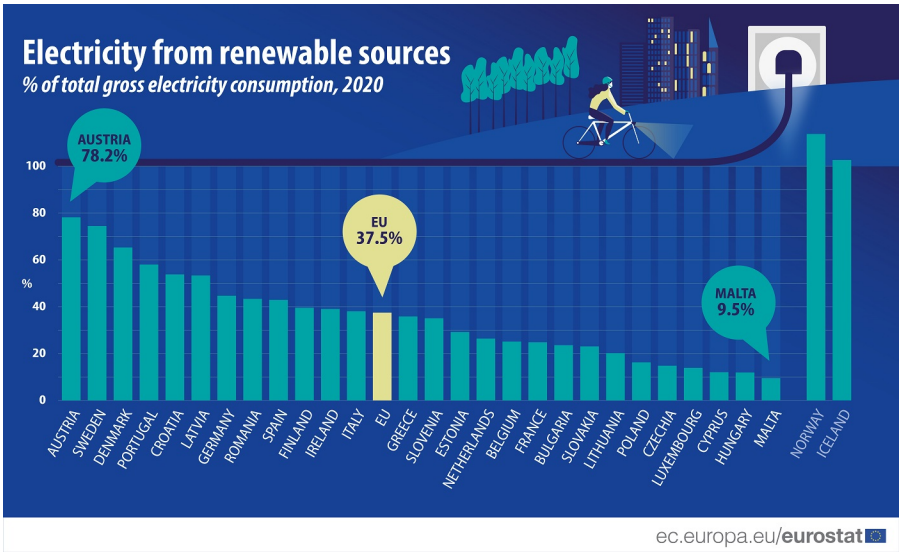

Figure 1.1.- Percentage of the Total Gross Electricity Consumption obtained from RES in 2020 [1].

Inside an MG, the elements that allow for the proper operation of the system are the PECs. Without power converters, it would be impossible to inject the energy obtained from any renewable source into the grid, as different voltage and current adaptations are needed to adapt the electrical signal received to the parameters of the distribution network. Power electronic converters, along with energy storage systems, also allow for the stabilization of the electricity supply so that there are no abrupt changes, which is a typical problem of renewable sources, as they depend on many environmental factors.

1.2.- OBJECTIVES

The main objective of the work is the development of a real-time control system, for different PEC, that allows the integration into the grid of distributed generation and energy storage systems, considering the use of batteries.

Although the applications of this type of systems are very varied and will be discussed in later sections, the work has been developed to study the possible types of control applicable to the microgrid of a housing development with a renewable energy source and a battery for energy storage. This housing development will be able to operate autonomously or grid-connected and, in addition to using energy storage as a power source, the possibility of supporting the grid is also considered.

Firstly, the integration of ESS will be carried out in a Direct Current (DC) microgrid using DC/DC converters. Afterwards, the options for the integration of distributed resources in an Alternating Current (AC) grid will be analyzed, using three-phase DC/AC converters.

One of the most remarkable parts of the project is the implementation of the control system in an embedded system using C programming language and model-based programming, and its evaluation in real-time simulators and Hardware In the Loop (HIL) platforms.

Summing up, the objectives of the project are:

- Proper design of a control system for a DC/DC converter.
  - Voltage control of a DC bus.
- Proper design of several control systems for a three-phase inverter.
  - Voltage control of a DC bus.
  - Power control.
  - Voltage control of an AC bus.
  - Droop-control (frequency and voltage).
  - Virtual Synchronous Generator (VSG).
- Control of both converters simultaneously, within the same model.
- Implementation of all the above cases in a HIL system.

1.3.- ORGANIZATION OF CONTENTS

This document is organized as follows:

- In Chapter 2, the state of the art is presented in order to help the reader understand the developed work.
- In Chapter 3, an overview of the order in which the work has been carried out is given.
- Chapter 4 deals with the design of the different control systems implemented.
- In Chapter 5, the description of the operations necessary to be able to perform HIL simulations is explained.
- All results are presented and analyzed in Chapter 6.

- In Chapter 7 some conclusions are extracted from these results, and different ideas are given for future extensions of the project.

2. State of Art

In this chapter, the main aspects of distributed generation systems are introduced and so are the types of microgrids, and their characteristics, together with a brief description of the different types of converters found in them. Also several applications of the introduction of ESS inside a MG are presented. Finally, a short analysis of the HIL concept is given.

2.1.- DISTRIBUTED GENERATION SYSTEM

The traditional electric system is based on centralized generation, on which power is produced far away from the consumers. However, as represented by Figure 2.1, the current trend is moving towards distributed generation systems. On this new type of system, which consists of microgrids, power generation takes place closer to the consumer.

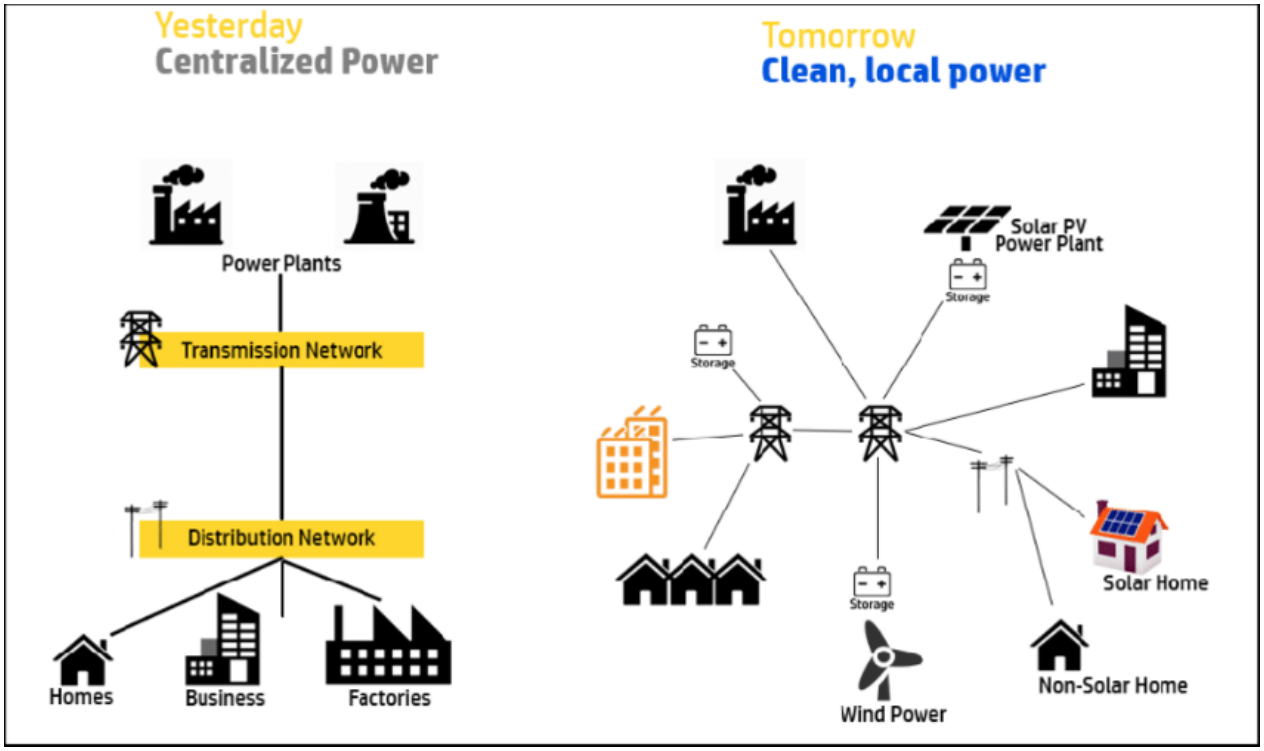

Figure 2.1.- Centralized Vs distributed generation [2].

The main benefits of distributed generation compared to the traditional one derive from the decentralization of the system [4]. Transmission losses are significantly reduced as the energy is generated closer to where it is demanded. The decentralization also allows for better optimization of the grid's operation by balancing power flow distribution. Distributed generation improves the grid's reliability, ensuring power supply even if part of the grid fails. Moreover, the microgrids that make up the system do not need to be entirely dependent on the main grid, allowing for the possibility of obtaining energy from other sources, mainly ESS. This enables, for instance, access to electricity in areas where originally it would be difficult to obtain (for example rural areas).

However, this decentralization also has its drawbacks, being the main problem an increase in the system's complexity. The incorporation of several generators and different circulation lines is a challenge when it comes to implementing a control system.

Another important factor that explains the growing popularity of distributed generation systems is that they allow for the introduction of Renewable Energy Sources (RES), thus reducing the carbon footprint. The main RES one can think of are solar and wind power, both of them depend on climatic conditions, which establishes a need for ESS in order to ensure that the power supply is maintained. ESS also improves the stability and power quality of the grid.

2.2.- DESIGN STRATEGIES FOR A MICROGRID

An MG is a cluster of generators, loads, and energy storage systems that can be analyzed as a small-scale electric system [5–7]. Its main objectives are, besides incentivizing the use of renewable energy, to add reliability to the grid and to reduce the losses during the transmission and distribution of electricity. The MGs can operate in two modes: connected to the main grid or in "island mode", isolated from the grid. ESS as well as PECs, with an adequate control system, play a key role in ensuring the proper operation of a microgrid.

In addition to the distinction according to the type of connection of the microgrid with the electric grid, microgrids can also be classified according to the bus (DC or AC). This will have an

effect on the technology used, the structure of the system, and the control strategies implemented. With this approach, three types of microgrids can be distinguished: DC, AC, and Hybrid; whose outline is shown in Figures 2.2, 2.3, and 2.4 respectively

Research on transmission efficiency is currently underway, both in DC and AC, to try to conclude which methodology is the most appropriate. The main reason for the use of AC is that it allows to raise the voltage, decreasing losses in transport. On the other hand, DC systems offer greater reliability against certain events and their control is easier [8, 9].

While energy generated from renewable sources is obtained as a direct current signal, most loads used to work with alternating current. However, many of these loads are shifting towards a mode of continuous operation, leading to a growing popularity of MGs in DC, as the double energy conversion is avoided, therefore eliminating some of the complexity of the system [10].

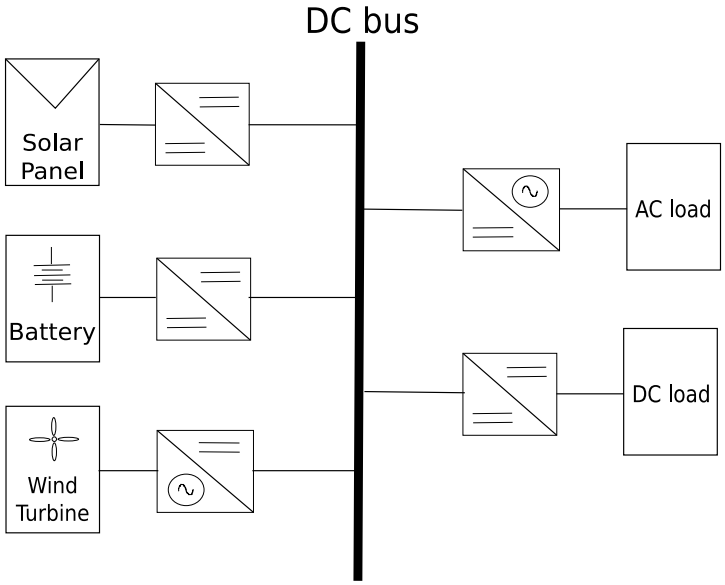

Figure 2.2.- DC Microgrid Scheme.

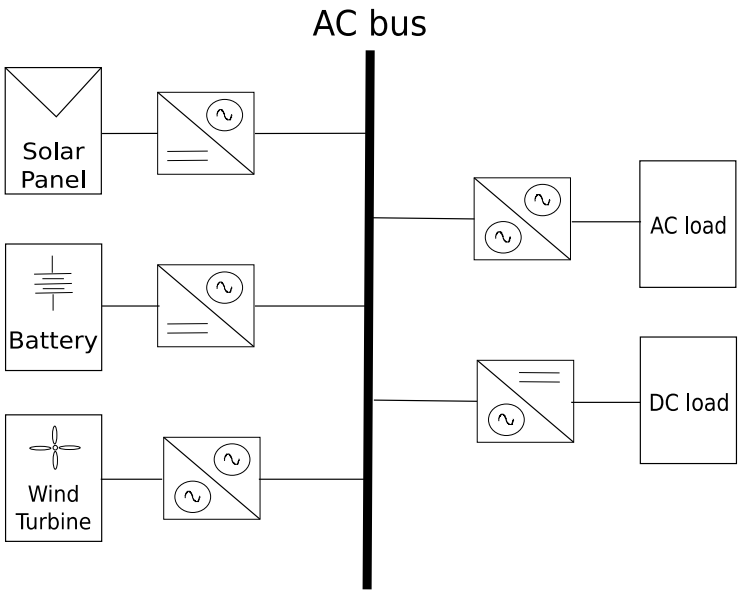

Figure 2.3.- AC Microgrid Scheme.

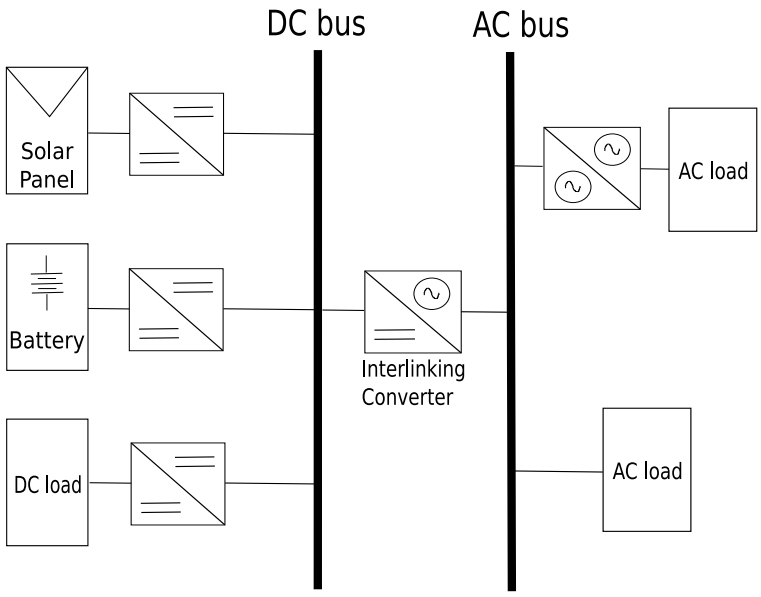

Figure 2.4.- Hybrid Microgrid Scheme.

2.3.- POWER ELECTRONIC CONVERTERS

Power converters are electronic devices capable of transforming an input voltage into a different output voltage with the desired characteristics. As mentioned in the section above, PECs

play a fundamental role in distributed generation systems, allowing for the integration of ESS and RES. Power converters are also required to integrate elements that operate in DC within AC networks, to integrate generators of an AC nature with variable frequency, or to work at the point of maximum power in renewable generation. The main goal of PECs used in DC microgrids is to adapt voltage levels and allow the connection of various energy resources, which generally work at low voltages, with a distribution bus, or with different loads [11]. Besides, the use of storage systems, typically batteries, makes bidirectional converters a necessity. Meaning that the PEC must be capable of handling power flow in both directions [12].

For the DC/DC converter, different topologies could be implemented. As no galvanic isolation is needed, the most basic topologies are considered. Among them, the Buck Converter, the Boost Converter, and the Buck-Boost Converter are found. For the first one, the voltage at the output is always lower than the voltage at the input, while for the second topology, this goes the other way around. The last topology mentioned is a combination of the previous two, allowing for the voltage at the output to be either lower or higher than the input voltage. However, none of those converters meets the bidirectionality condition. Therefore, a Synchronous Boost Converter is chosen, as it is a simple topology that allows for the sought bidirectionality and it allows for the battery voltage level to be raised, which is what is needed for the studied scenario. Moreover, it is the main topology used in battery integration for the voltage levels established in this project, as it improves power conversion efficiency when compared with the asynchronous topology [13].

As far as the DC/AC converter goes, a three-phase inverter is analyzed. Again, the condition of bidirectionality is met, as the converter is also able to work as a rectifier.

In power converters we always find switched elements, these are power transistors. Perhaps, the most common power transistors in power electronics systems are MOSFETs and IGBTs. Generally speaking, MOSFETs are more suitable for low-voltage, fast-switching applications, while IGBTSs are more suitable for high-voltage, slower-switching applications, as shown in Figure 2.5. For the application developed in this project, taking into account the magnitude of the currents that these elements must be able to withstand, and knowing that the switching frequency to be set is 10

kHz, it is most appropriate to use IGBTs.

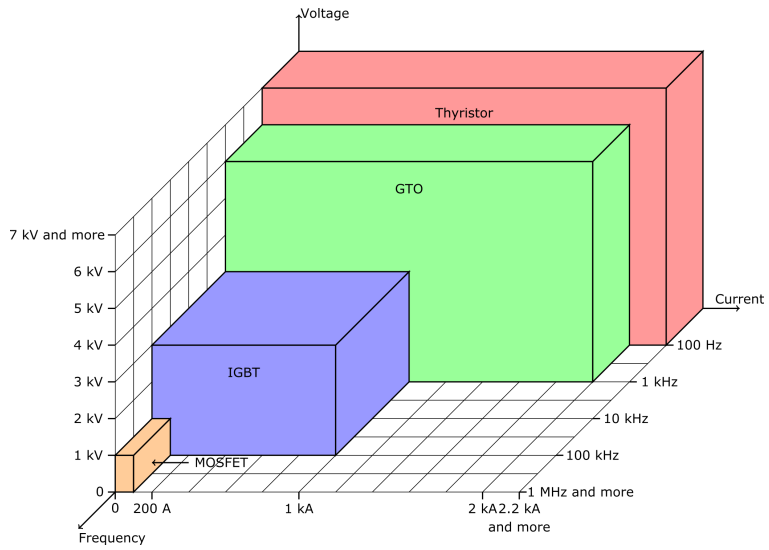

Figure 2.5.- Power Transistors Range of Application [3].

2.3.1.- Three-phase inverters' role in a MG

The roles of the inverter within an MG in a microgrid include the management of power flow from one resource to another, as well as ensuring the stability of the grid. Four different roles that an inverter may play in an MG can be distinguished depending on the operation mode of the MG with regard to the main grid.

2.3.1.1.- Grid Feeding-Inverter

Grid-feeding inverters appear on grid-connected MGs, as voltage and frequency are established by the main grid. Grid-feeding inverters manage power flow, they can be seen as ideal current sources and several grid-feeding inverters may appear connected in parallel inside the same MG. Their main function is to inject power to support the grid in supplying the different loads connected. Therefore, a power control is applied to this type of converter.

2.3.1.1.1 Reactive Power Compensation Generally, the power references are set so that the inverter provides a large percentage of the active power demanded by the loads, whereas the reactive

power reference is typically set to zero. However, electric loads are mainly inductive and therefore consume reactive power, which has to be fully provided by the grid.

Reactive power compensation is one of the ways in which grid quality can be improved, reducing power losses and contributing to voltage stability. Moreover, this method helps reduce the need, that many companies face, to include large capacitor banks for power factor compensation.

With an inverter, the reactive power compensation can be conducted following different control strategies. All of these consist mainly of controlling the amount of reactive power injected, or monitoring the grid's power factor [14, 15].

2.3.1.2.- Grid Forming Inverter

Grid-forming inverters are in charge of generating the electric grid to which the different loads connect when the MG is operating in island mode. Therefore, only one inverter of this type can be present inside an MG [16].

Grid-forming inverters operate as ideal voltage sources, establishing the voltage and frequency of the grid, so a voltage control needs to be implemented. They will either inject or consume power in order to maintain stability, but no power references as such are established.

2.3.1.3.- Grid Supporting Inverter

These are converters that operate in a way intended to support the grid by regulating both voltage and frequency. The type of control applied on grid-supporting inverters is known as droop control, which consists of adjusting the grid's voltage and frequency according to reactive and active power droops respectively [16–18].

2.3.1.4.- Virtual Synchronous Generator

This type of operation will be discussed in more detail below, in Section 2.4.2.2. As a general feature, the VSG is a novel solution based on the concepts of Grid Forming and Grid Supporting, whose main objective is to provide inertia to the grid [19].

2.4.- EES AND PECS CONTROL APPLICATIONS

On this section, different auxiliary services that power converters can provide to the grid are analyzed with the intention of highlighting the importance of developing appropriate control systems that allow for the integration of ESS.

2.4.1.- Ramping Events in RES

As discussed in the previous sections, the integration of RES is one of the means by which both the climate crisis and the energy crisis are to be tackled. Government policies have focused mainly on wind and solar energy, an interest that has been accompanied by a large number of academic studies. However, those resources are overly dependent on climatic conditions, making their energy production uncertain and unreliable.

One type of event that is typical in both solar and wind power plants, which is a major concern and becomes more relevant as the presence of RES within the electricity grid increases, is the one known as ”ramping event” [20, 21]. Ramping events can be caused by different factors. In wind power generation, they can appear due to thunderstorms or cyclones, while in solar power generation, they can be caused simply by the fact that a cloud passes over the photovoltaic facility.

This ramping behavior also affects the grid voltage regulation. For photovoltaic power generation, when the sun’s rays suddenly stop reaching the photovoltaic panels, the output power of the inverters decreases rapidly. Such abrupt variations produce severe fluctuations in the grid voltage. An electric grid that gets its energy from renewable sources does not have the capacity to quickly react to these changes (there is no grid inertia). Therefore, ramping events can lead to power quality issues and voltage instability.

In an attempt to solve these problems, different ramp-rate control strategies based on smoothing techniques are under study. They mainly consist of injecting power, obtained from ESS, into the grid when a ramping event is taking place.

2.4.2.- Inertia Support

In this section, the previously mentioned problem of lack of inertia is analyzed.

2.4.2.1.- Synchronous Generator

In traditional power stations, energy generation consists of the movement of a turbine, which is connected to a synchronous generator, responsible for the generation on electricity. A synchronous generator is an electrical machine able to transform mechanical rotational energy into electrical energy, basing its operation on Farady-Lenz's Law.

Synchronous generators are capable of continuously modifying and controlling both active and reactive power. The active power can be regulated by acting on the main source driving the rotation of the turbine coupled to the generator (water, steam, ...). This suggests that the active power is related to the rotational frequency, which in turn is directly proportional to the frequency of the generated electrical signal. On the other hand, reactive power can be controlled by changing the current in the rotor's winding, which is equivalent to controlling the stator's terminal voltage. Therefore reactive power is related to the voltage magnitude.

When a synchronous generator is rotating and suddenly a new load is connected, more power is demanded and the rotor decelerates, which implies a reduction in the electric frequency. However, as energy is stored in the rotor in the form of inertia, this transition is produced smoothly. In general terms, inertia can be said to benefit the grid's stability.

2.4.2.2.- RES and Grid's Inertia: Virtual Synchronous Generator

Inertia is an attribute that restricts frequency variations in the event of abrupt changes in generation or load. However, distributed generators and energy storage systems are connected to the grid through at least one power conversion stage. As opposed to synchronous generators, PECs do not have a real moving mass providing inertia and do not contribute to the inertia of the power system. Therefore, the high penetration of RES decreases this intrinsic inertia [22–25].

This lack of inertia, together with the intermittency inherent to RESs, causes frequency instability in the power system. This means that when there is any variation in the demanded power, an abrupt change in the frequency will take place. Therefore, the grid is more susceptible to becoming unstable. In fact, if for any reason there is a part of the grid that needs to be disconnected, the rest of the system might not be able to cope with it.

To allow the incorporation of renewable energy generation plants, while maintaining grid inertia, it is possible to control the PECs operation so that it emulates the effect of the synchronous generator. When this type of control is applied, the power converter is known as Virtual Synchronous Generator (VSG).

Droop control is also used for mimicking the behavior of SGs. Both VSG and droop-controlled converters can be considered to fall into the category of Grid Supporting Inverters. However, the droop control does not provide virtual rotational inertia, which means that large values of Rate Of Change Of Frequency (RoCoF) appear (changes in frequency are still abrupt). Large RoCoF values may endanger secure system operation because of mechanical limitations of individual synchronous machines or timing issues related to load connection schemes [26].

Several control strategies for synchronous machine emulation have emerged, some of them are mentioned below.

- Inertia-emulation (IE).
- Synchronverter (SV).
- Current-controlled virtual synchronous machine (CCVSM).
- Voltage-controlled virtual synchronous machine (VCVSM).

The inertial power response can be calculated from an estimation of the grid's frequency. However, such control will depend on a Phase Lock Loop (PLL) to obtain the grid synchronization, meaning that this technique can not be applied in islanded systems. Another method through which

virtual inertia can be provided is by simulating the electromechanical swing equation, (2.1), of a synchronous machine. Where H is the inertia of the machine, ω is the angular speed, τ_m is the mechanical torque generated, τ_e is the electric resisting torque and D is the damping factor. To include this equation inside the control loop, variations of the angular speed are considered to be very small so that the torque balance can be expressed as a power balance. Converters in which this technique is applied operate as grid-forming converters while also providing inertia support.

For the correct operation of clean energy systems incorporating VSGs, the introduction of ESS is once again essential, as elements capable of storing energy are needed to simulate the inertia of the generators.

$$2H \cdot \frac{dw}{dt} = \tau_m - \tau_e - D \cdot \Delta\omega \tag{2.1}$$

2.4.3.- RES For Self-Consumption

The installation of small-scale renewable energy generation facilities, designed for self-consumption, is becoming increasingly common. These are typically small photovoltaic plants on individual properties or buildings, although wind turbines designed for self-consumption are also available.

In this scenario, citizens are no longer mere consumers. Their role within the electricity system is to be both a producer and a consumer. Hence, the term ”prosumer” is born. Prosumers can produce their own energy, saving money on their energy bills as they are less dependent on fluctuating energy prices. This type of installation also allows energy management to reduce consumption peaks that may occur throughout the day. Moreover, they can sell any surplus energy they are able to generate, injecting it into the grid.

Prosumers are therefore forming their own microgrid, with a scheme similar to the one shown in Figure 2.6. On those systems, when enough power is obtained from the photovoltaic installation, no connection to the grid is needed and, if a power surplus is generated, the energy can be stored in the ESS.

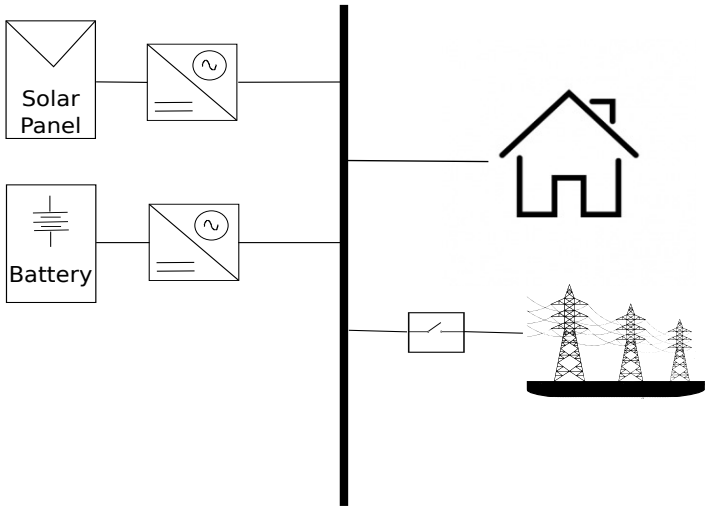

Figure 2.6.- Self-Consumption Microgrid: Photovoltaic Generation & Energy Storage.

When no photovoltaic power is obtained, the energy needed to feed the loads can be obtained from the ESS. Also, if at any time there is no energy available in the storage systems and not enough power is being obtained from the solar panels, the installation can feed from the grid. On the other hand, if there is a surplus in the energy produced and the batteries are already charged, the connection with the grid can be made to inject the excess power.

2.5.- HARDWARE IN THE LOOP

Hardware in the loop (HIL) simulation is a type of real-time simulation [27], which means that the computer model runs at the same rate as the actual physical system. This technique is used in the development and mainly in the testing of control algorithms. It can also be used to analyze the plant, which is the physical system, and establish whether the designed model is valid.

The main advantage of HIL is that the behavior of the system can be analyzed without the need for building physical prototypes. Instead, a real-time computer, known as the HIL test system, is used to emulate the plant. This HIL simulator can accurately reproduce the dynamics of the physical system, including an electrical simulation of all sensors and actuators. Meanwhile, the control algorithm runs on a real-time embedded controller; both elements interact with each other

through Input/Output (I/O) channels. A scheme of how a HIL system works is shown in Figure 2.7.

Therefore, HIL simulations are especially useful when the system incorporates components that are difficult to model or when testing the algorithm on the physical system is either costly or dangerous. The first applications of HIL were in the aerospace industry, and the automotive industry has also benefited from such simulations. Nowadays, HIL has found its way into the field of power systems, as they need to be tested under realistic but controlled conditions.

Other important benefits of HIL are the possibility to run through a high number of scenarios with no additional costs and the repeatability of the tests performed (HIL testing is always non-destructive).

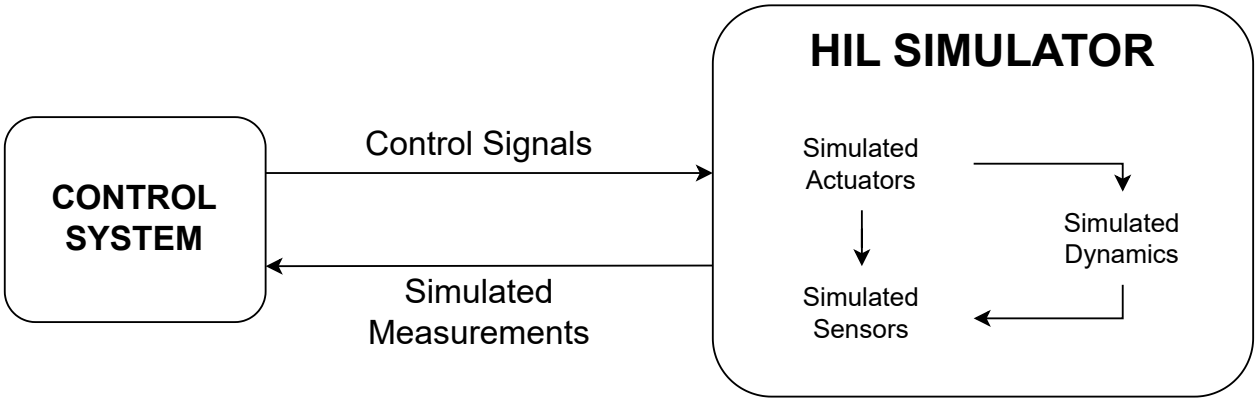

Figure 2.7.- HIL System.

3. Working Methodology

This project is divided into two main parts. The first one consists of the design and simulation of the control systems for both the DC/DC converter and the three-phase inverter, as well as the simulation of a microgrid model which incorporates both converters. In the second part, HIL simulations are carried out to check the control systems designed in a scenario closer to reality.

For the first section, the DC/DC converter control system aims for energy storage integration on a DC bus. On the other hand, different control strategies are analyzed for the inverter, which connects a DC bus to the mains (or to an AC bus). In both cases, a cascade control is implemented, with an internal loop for current control and an external loop for controlling another variable of interest, usually the voltage or the power.

The simulation software used throughout the whole project is PLECS (Piecewise Linear Electrical Circuit Simulation), a tool for the simulation of electrical circuits, developed by Plexim. PLECS is specially designed for power electronics, but it can be used for any electrical network. Moreover, it includes the possibility of modeling controls and different physical domains (thermal, magnetic ...) in addition to the electrical system.

For the real-time simulation of the power converters, Plexim's RTbox 1 platform accessible from the LEMUR power laboratory server is used. While for the implementation of the digital controller, the Texas Instruments TMS320F28335 Microcontroller Unit (MCU) is used. PLECS environment is again needed to design both the plant and the control systems. To do so, the RTbox libraries have to be installed, as well as the TI-C2000 microcontroller code generation toolbox, being both resources are available on Plexim's website [28, 29].

The whole HIL system can be seen in Figure 3.1, where the MCU is connected to the RTbox through a control card interface.

Figure 3.1.- RTbox and MCU connected through the control card interface.

The PLECS RT Box is a real-time simulator that contains a dual-core ARM CPU and an FPGA. The real-time simulation is performed on the CPU core, while the FPGA acts as an interface to the RT Box inputs and outputs. The same code used for a PLECS simulation on the PC can also be used to simulate the model on the RT Box.

The F28335 belongs to the Texas Instruments (TI) C2000 processor family, which is purpose-built for real-time power electronics control. PLECS Coder allows for two ways to deploy generated embedded code, the first one is to directly build and program the MCU from PLECS, and the second one is to generate code into a Code Composer Studio (CCS) project to build, debug, and program from there. During the development of the project, both approaches have been taken.

On the following page, the evolution of the developed work is shown graphically through the Gantt chart in Figure 3.2.

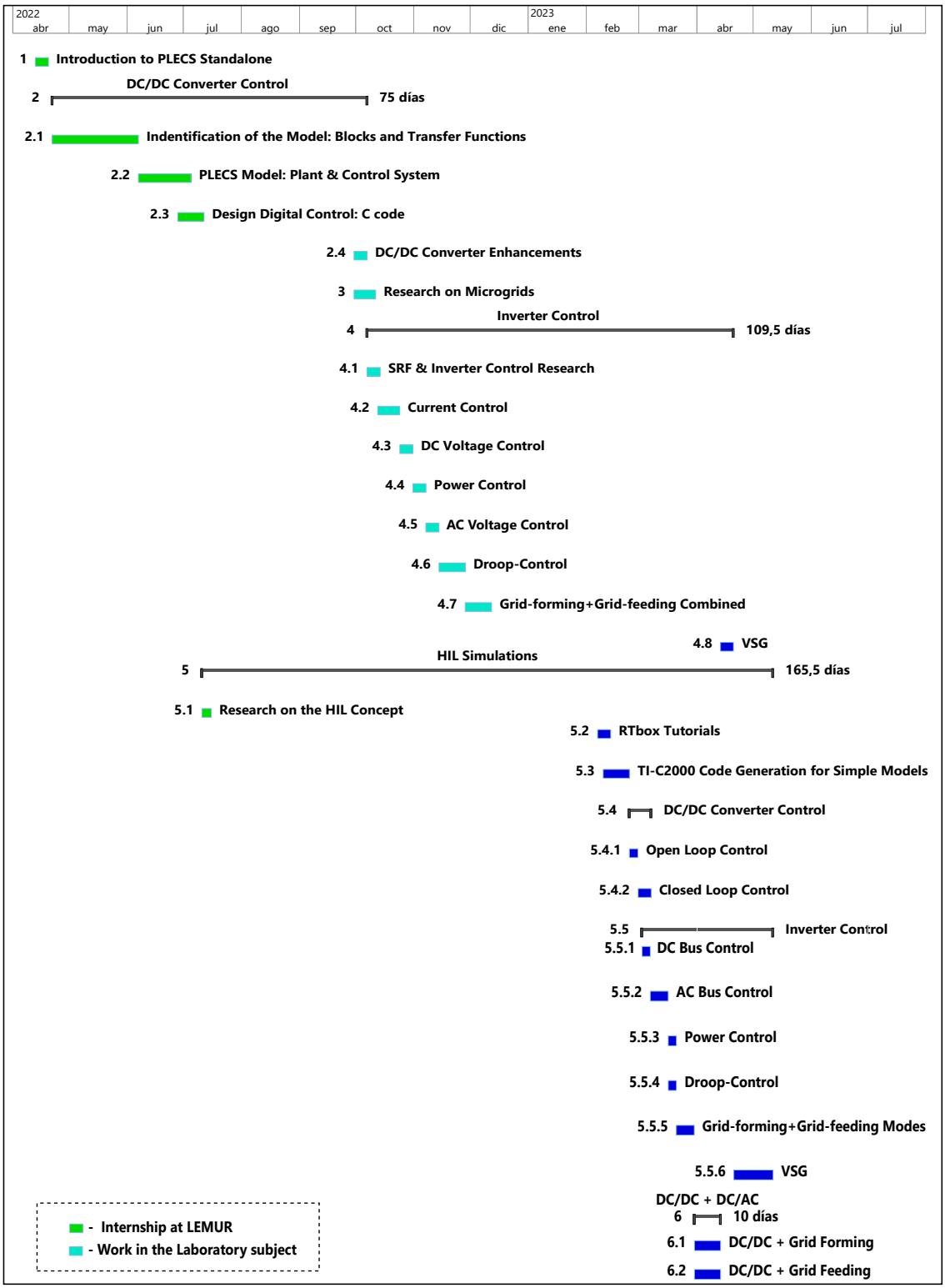

Figure 3.2.- Gantt Diagram.

4. Design of Control Systems for PEC

Before starting with the description of the design of the different control loops, the general parameters considered for the microgrid are presented below in Table 4.1.

Table 4.1.- System parameters.

Parameter	Value	Unit
Nominal Power	25	KVA
Three-phase: line-to-line Voltage	400	V
Three-phase: phase Voltage	230	V
DC Bus Voltage	800	V
Nominal Frequency	50	Hz
Switching Frequency	10	kHz

The nominal active power demand for the housing stock is considered to be 25 kW. However, as this demand will not be constant and peaks in consumption may occur, the converters will be able to handle up to 50 kW. In addition, this oversizing will allow for contingencies and flexibility in the event of network growth.

4.1.- SYNCHRONOUS BOOST CONVERTER

This DC/DC converter acts as the interface between the MG bus and a battery, as can be seen in Figure 4.1. Behaving like a Buck converter during the charging state of the battery and as a Boost during its discharge. In our case of interest, it is considered that energy needs to be extracted from the battery to inject it into the DC bus, from which the loads are fed. Therefore, the variable which is aimed to be controlled is precisely the voltage of this bus. However, in case the generation exceeds the load on the DC grid, this storage system will be able to absorb surplus power.

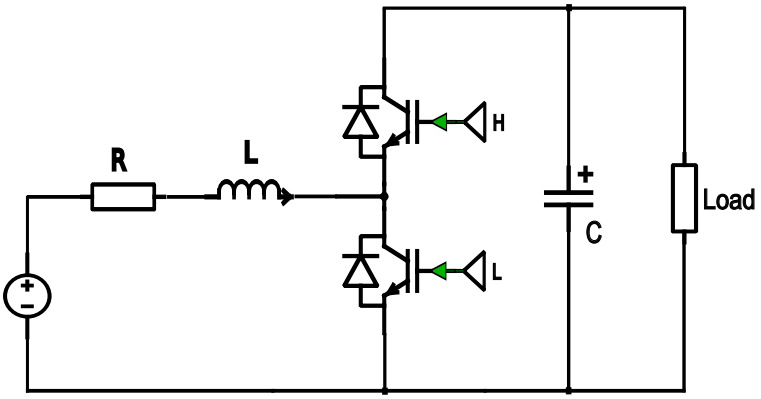

Figure 4.1.- Synchronous boost converter.

The battery is connected to the low voltage port of the converter and, to the other port, a capacitor that models the DC bus is connected. Loads connected in parallel with the bus will behave as disturbances in the voltage control system.

4.1.1.- Control System Design

A controlled voltage source is used to simulate the battery, this allows us to introduce a step input in the voltage to verify the rejection of this type of disturbance by the control loop. Moreover, a step is also introduced in the load to ensure a correct rejection of this kind of disturbance. The different currents and voltages needed to be measured for implementing the control system are indicated in Figure 4.2.

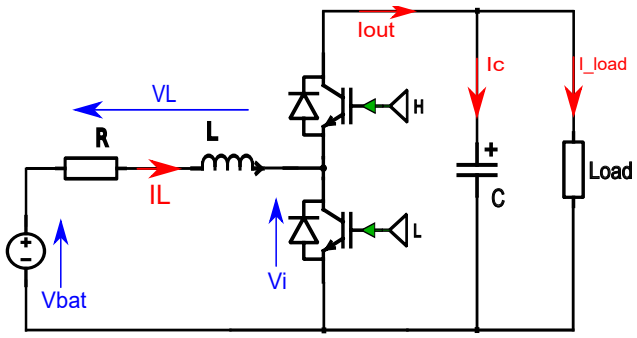

Figure 4.2.- Voltages and currents across the SBC.

As an inverter is to be connected to the DC bus at a later stage, an important factor to take into account is that the inverter can only generate a three-phase voltage whose maximum amplitude is half of the DC voltage at its input. As the inverter will be either connected to the grid or generating the grid itself, the peak voltage at the output will be $230 \cdot \sqrt{2}V$. For this reason, a voltage of 800V is set as the reference value for the DC link. The value of the battery's voltage used so that the DC/DC converter can reach this reference is 400V.

Low-pass type filters, implemented using a second-order transfer function, are incorporated into the signal measurements. They work as anti-aliasing filters with such a cutoff frequency that the Nyquist–Shannon theorem for signal sampling is guaranteed. In this simulation, the sampling frequency is the same as the switching frequency: 10kHz.

The objective of the control loop is to obtain the PWM modulating signals that govern the transistor switching. The inner loop looks as presented in Figure 4.3 and is intended to control the current through the coil. The control action obtained at the output of the inner loop regulator is the voltage across the R-L filter. But, to obtain the modulating signal, it is necessary to compute the voltage between the terminals of the transistors from the voltage at the filter. This can be done using (4.1) and then normalizing the obtained value with the voltage measurement from the DC bus, as shown by (4.2).

$$V_i = V_{bat} - V_{RL} \tag{4.1}$$

$$d = \frac{V_i}{V_{out}} \tag{4.2}$$

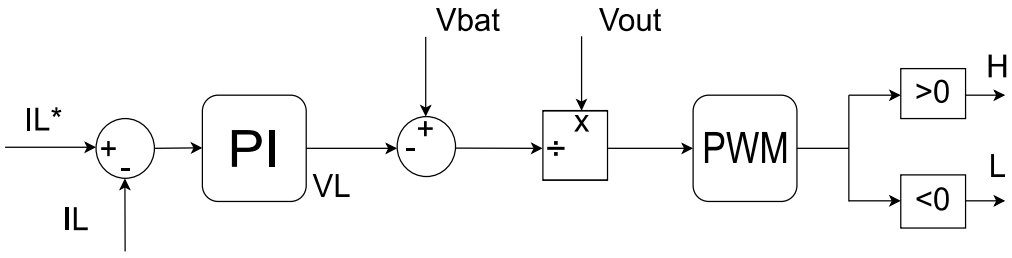

Figure 4.3.- Inner-loop of the SBC control.

In the outer loop, shown in Figure 4.4, the voltage control is performed and the reference current applied to the inner loop is obtained. The control action after the regulator is now the current flowing through the capacitor. Considering that the input power is the same as the output power and (4.3), taking into account that the output current is divided when a load is introduced (4.4), the current through the inductor follows equation (4.5).

$$P_{in} = V_{bat} \cdot I_L = P_{out} = V_{out} \cdot I_{out} \tag{4.3}$$

$$I_{out} = I_c + I_{load} \tag{4.4}$$

$$I_L = \frac{(I_c + I_{load}) \cdot V_{out}}{V_{bat}} \tag{4.5}$$

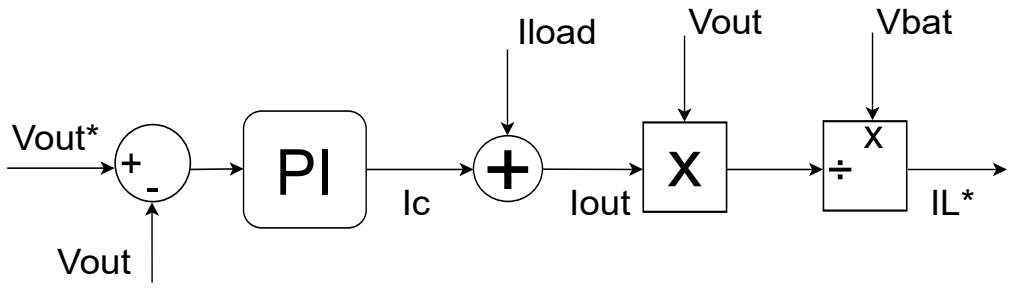

Figure 4.4.- Outer-loop of the SBC control.

4.1.2.- Controller Tunning

Both regulators are of parallel PI type (4.6), and their tuning is carried out following the pole-zero cancellation method.

$$D(s) = k_p \cdot \frac{s + \frac{K_i}{K_p}}{s} \tag{4.6}$$

In order to perform a correct tuning, it is necessary to take into account the bandwidth required for each loop. Knowing that the inner loop has to be fast enough compared to the outer loop, so that, from the point of view of the outer loop, the inner one is seen as a unit block with zero dynamics. Taking this into account, a bandwidth of 500Hz is chosen for the inner loop, so that the outer loop is tuned at 50Hz. The plant on which the current control is applied is identified by the transfer function described by (4.7).

$$G_i(s) = \frac{1}{R_{SBC} + L_{SBC} \cdot s} \tag{4.7}$$

With this plant, the cancellation method leads to expressions (4.8) and (4.9) to obtain the proportional and integral gains for the PI controller.

$$k_p = L_{SBC} \cdot w_B \cdot 2\pi \tag{4.8}$$

$$k_i = R_{SBC} \cdot w_B \cdot 2\pi \tag{4.9}$$

To tune the external loop, the resistive load connected in parallel with the capacitor is included as part of the plant, which is therefore equivalent to (4.10). Now the gains for the PI controller are (4.11) and (4.12).

$$G_v(s) = \frac{R_{load}}{1 + R_{load} \cdot C_{bus} \cdot s} \tag{4.10}$$

$$k_p = C_{bus} \cdot w_B \cdot 2\pi \tag{4.11}$$

$$k_i = \frac{w_B \cdot 2\pi}{R_{load}} \tag{4.12}$$

The final parameters chosen for this converter are presented in Table 4.2.

Table 4.2.- SBC parameters.

Control Loop	Parameter	Value	Unit
	R_{SBC}	0.01	Ω
	L_{SBC}	1	mH
	C_{bus}	700	µC
Current	Bandwidth	500	$\frac{rad}{s}$
Current	K_p	3.14	
Current	K_i	31.41	
Voltage	Bandwidth	50	$\frac{rad}{s}$
Voltage	K_p	0.22	
Voltage	K_i	12.27	

4.1.3.- Precharge Circuit

In order to avoid any problem that may occur if the control is applied to the plant when the DC bus voltage is null, it is necessary to design a precharge circuit. This circuit is made by

introducing a precharge resistance between the power supply and the bus capacitor. As can be seen in Figure 4.5, a switch or relay must be connected in parallel with the resistor. This switch is controlled so that it remains open during the charging period and switches once the bus is charged, thereby shortcircuiting the limiting resistor.

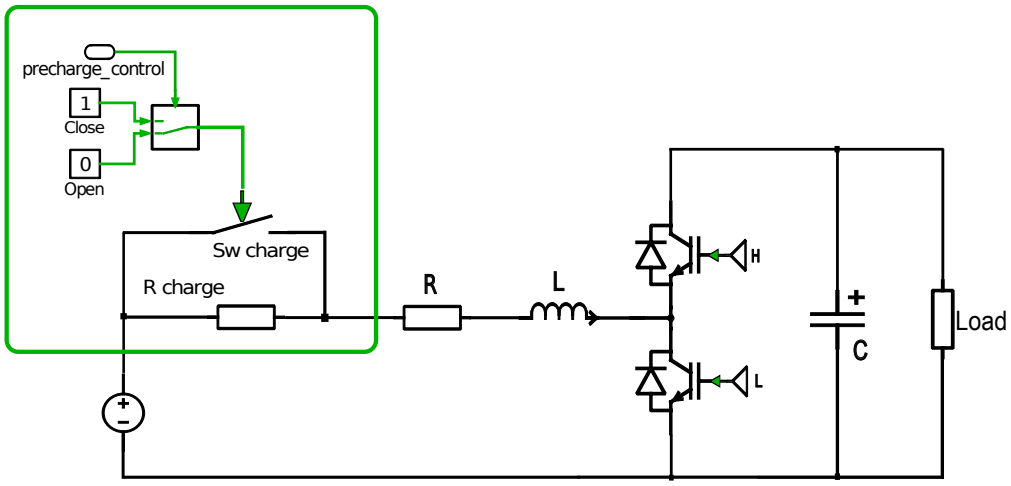

Figure 4.5.- Precharge circuit included on the SBC scheme.

During the loading stage, the signals obtained by the control loop used to rule the transistor branch are not used. Instead, the upper-side transistor must remain open while the low-side one is kept closed. However, the system is still working and the controllers are integrating. This means that the error will accumulate and the control action will increase while this stage lasts. Therefore, it must be ensured that, once the bus is loaded, the input error signal of the controllers has been reset to a null value.

4.1.4.- Control Discretization Using the C-Script

If the control loop is to be implemented in a real-time embedded system, such as a micro-controller, the continuous controller must be discretized. This discretization has been carried out following Tustin's method with (4.13), where Ts represents the sampling period. The expression for the discrete controllers is given by (4.14). And, from there, (4.15) used to compute the control action (u) from the measured error (e) is obtained. In this expression the subscript "k" represents the current value of the variables; while the subscript "k-1" refers to the value they had in the last

execution cycle.

Moreover, not only the controllers have been discretized, but all the blocks that made up the control system have been exchanged for a ”C-Script” block, as can be observed in Figure 4.6. This element allows the implementation of the control in the C programming language, being the inputs of the C-Script block all the signal measurements. It can be observed in Figure 4.6 that there are two outputs, the duty cycle needed to generate the appropriate PWM and the logic signal used to command the precharge of the DC bus.

When using the C-Script block, an important parameter that needs to be defined is the Sample Time, as it controls when the block is called and therefore when the update and output functions are executed. The Sample Time used is the same as the Sampling Period mentioned before, which is the inverse value of the switching frequency $T_s = \frac{1}{f_s}$.

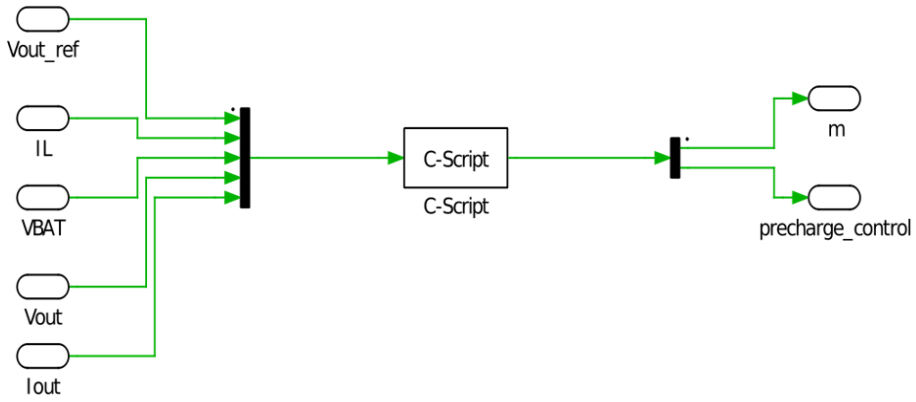

Figure 4.6.- Control loops replaced by a C-script block.

$$s = \frac{2}{T_s} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{4.13}$$

$$D(z) = k_p \cdot \frac{(\frac{2}{T_s} \cdot k_i + 1) + (1 - \ \cdot k_i) \cdot z^{-1}}{\frac{2}{T_s} - \frac{2}{T_s} \cdot z^{-1}} = \frac{a0 + a1 \cdot z^{-1}}{b0 - b1 \cdot z^{-1}} \tag{4.14}$$

$$b0 \cdot u_k = b1 \cdot u_{k-1} + a0 \cdot e_k + a1 \cdot e_{k-1} \tag{4.15}$$

The C-Script block has several predefined function calls. For this simulation, only the four of them are presented below and in Figure 4.7 are used:

- code declarations(): It works as a global header file, where all variables and functions are defined. It is only executed once, at the beginning of the simulation.
- start(): The start function is used to assign the initial values to all internal state variables. It is only executed once, at the beginning of the simulation.
- output()
- update(): The update function is called directly after executing the output function. Therefore, it is used to ensure that state variables are updated only once for each simulation cycle.

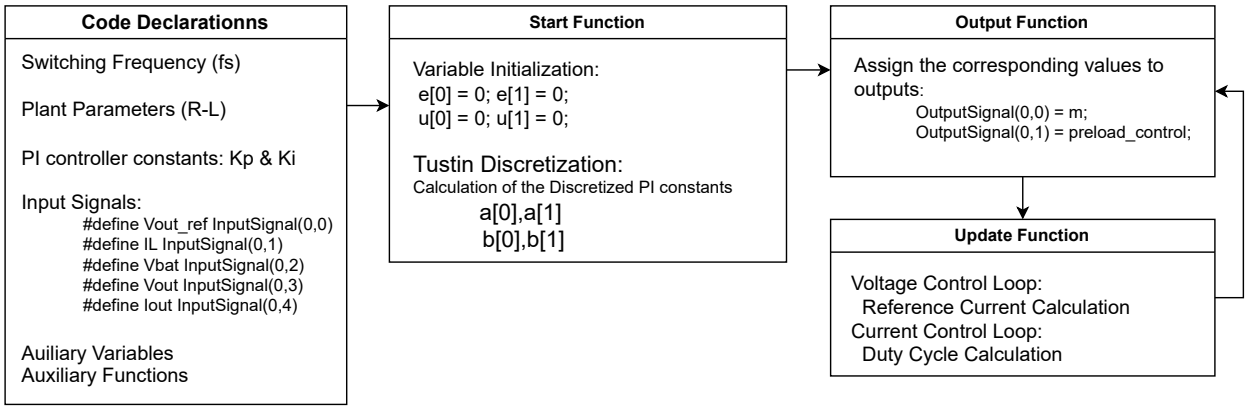

Figure 4.7.- Overview of the C-Scrip block functions used.

The process that takes place every simulation cycle is described in Figure 4.8.

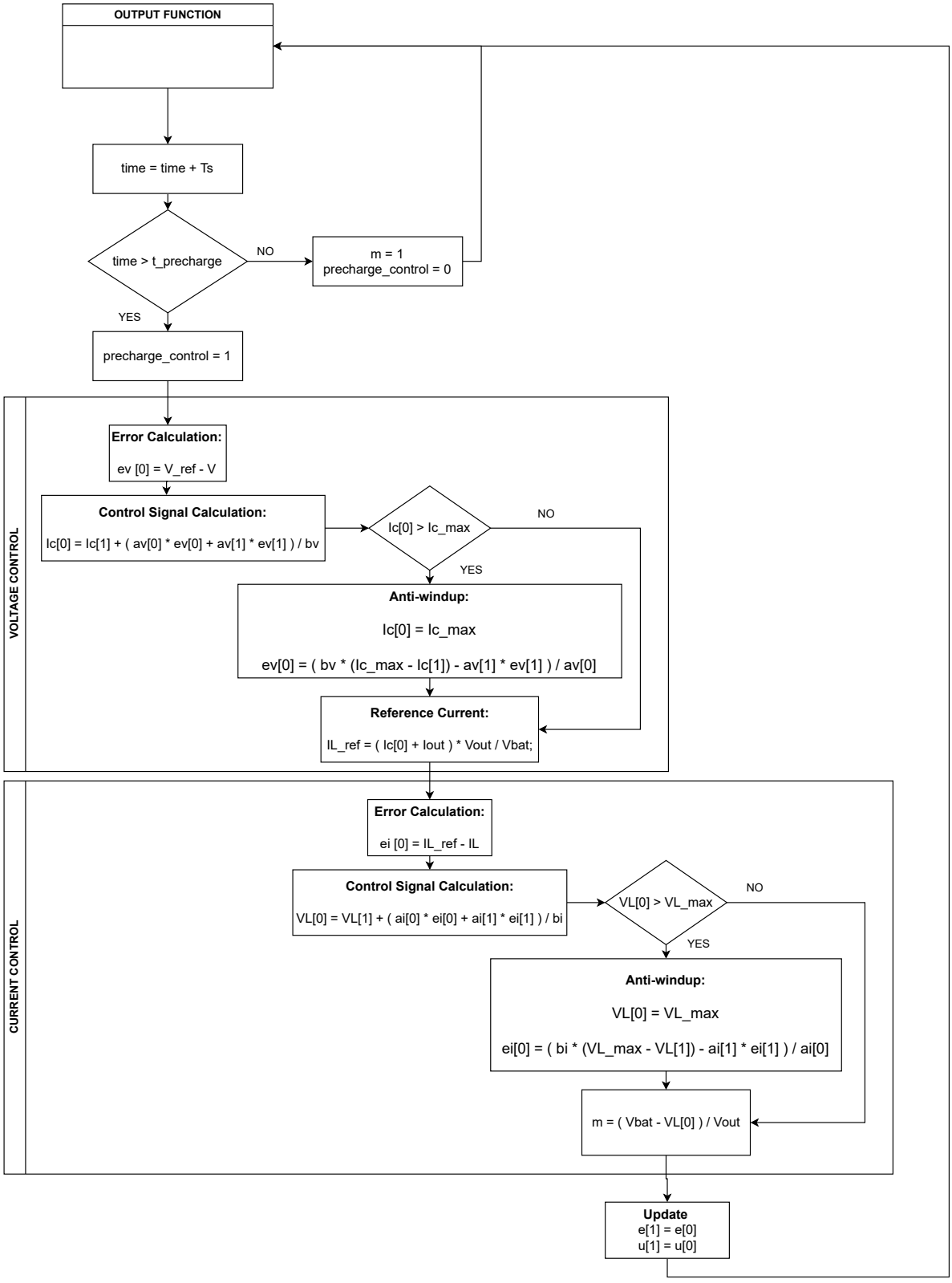

Figure 4.8.- Update Function's Flow Diagram.

4.2.- THREE-PHASE INVERTER

The type of control applied to the inverter will depend on whether it is connected to the main grid or not. In either case, a frequency control can be applied, becoming the converter a ”Grid Supporting” inverter. If the MG works in island mode, it will be the inverter the element in charge of creating the grid to which the different loads connect, and the inverter would be in this case a ”Grid Forming” one. In this case, a voltage control should be applied at the output.

On the other hand, when the MG is connected to the main grid, different control modes can be implemented. Firstly, a voltage control of the DC bus can be performed. However, if the bus is already being controlled by, for instance, a DC/DC, we can have a ”Grid Feeding” inverter in which a power control is established.

Moreover, it is common for the same microgrid to be able to work in both modes through the switching of the Point of Common Coupling (PCC). In this case, the voltage signal generated by the inverter when the PCC is open must be synchronized with the one of the grid, so that the connection can be made whenever it is necessary.

The model on which the different types of control are to be analyzed is shown in Figure 4.9, which also shows the labels used for the current and voltage measurements necessary for the implementation of the control systems. The values of the inverter parameters are presented in Table 4.3.

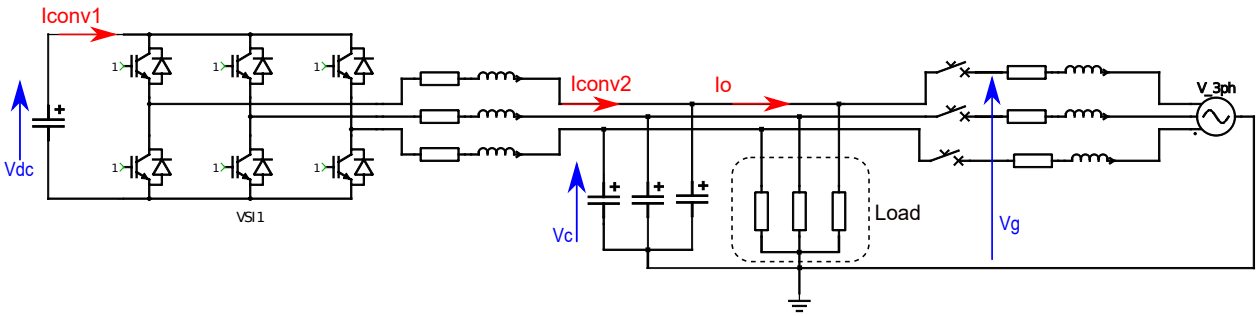

Figure 4.9.- Simulated AC microgrid model.

Table 4.3.- Inverter parameters

Parameter	Value	Unit
R_{inv}	0.01	Ω
L_{inv}	1	mH
C_f	10	μC
R_g	1	mΩ
L_g	0.1	mH

4.2.1.- Introduction to inverter control

The control strategy is the same as for the DC/DC converter. However, the main difference is that now the system works with sinusoidal signals. This means that it is not possible to implement PI regulators in the same way that it was done in DC. In addition, the control becomes more complex, as now it has to simultaneously monitor 3 variables for each three-phase signal.

Those are the reasons why a reference frame change is made into the Synchronous Reference Frame (SRF) as shown in Figure 4.10 and represented by the matrices of 4.16 and 4.17. In order to do so, Clarke and Park's transformations are used. Clarke's transform allows us to go from the three-phase system in the time domain to a two-phase system in an orthogonal reference frame. Park's transform is applied to obtain an orthogonal rotational reference frame (dq), known as SRF.

$$\begin{bmatrix} X_d \\ X_q \\ X_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\theta & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta - \frac{4\pi}{3}) \\ -\sin\theta & -\sin(\theta - \frac{2\pi}{3}) & -\sin(\theta - \frac{4\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} X_a \\ X_b \\ X_c \end{bmatrix} \tag{4.16}$$

$$\begin{bmatrix} X_a \\ X_b \\ X_c \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\theta & -\sin\theta & \frac{1}{2} \\ \cos(\theta - \frac{2\pi}{3}) & -\sin(\theta - \frac{2\pi}{3}) & \frac{1}{2} \\ \cos(\theta - \frac{4\pi}{3}) & -\sin(\theta - \frac{4\pi}{3}) & \frac{1}{2} \end{bmatrix} \begin{bmatrix} X_d \\ X_q \\ X_0 \end{bmatrix} \tag{4.17}$$

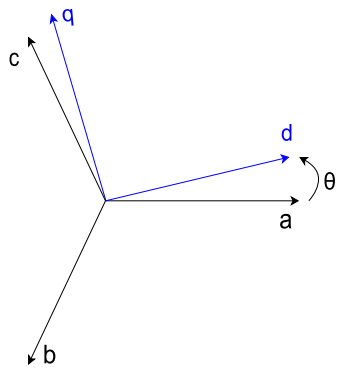

Figure 4.10.- Synchronous Reference Frame.

Notice that this change is not possible without a synchronization signal, which will set the rotational speed at which the SRF rotates. In this scenario, the reference signal is the phase angle of the three-phase signal; for which some phase estimation technique needs to be used. With the SRF-PLL [30] method good results are obtained under ideal conditions.

Nevertheless, when the voltages of the system undergo variations, for example, due to imbalances in the different phases, and when a good dynamic response is needed, this method is not the best approach. Therefore, for power converters control, a more appropriate method is the Double Synchronous Reference Frame PLL (DSRF-PLL) [31]. This is an ampliation of the SRF-PLL, in which two reference frames rotating in opposite directions are used to obtain an estimate for the positive and negative sequences in unbalanced systems.

The system analyzed in the SRF is simpler, as it uses two DC variables instead of three AC ones. Therefore, the same control scheme as in the DC/DC converter can be followed [32].

4.2.2.- Current Control

The current control system is divided into two identical loops, shown in Figure 4.11, one for each component of the new reference frame. Each of them includes a PI regulator tuned by the pole cancellation method, again following (4.8) and (4.9), the computed values are shown in Table

4.4.

Table 4.4.- Current control parameters

Parameter	Value	Unit
Bandwidth	500	$\frac{rad}{s}$
K_p	3.14	
K_i	31.41	

With the same analysis as in DC, the relationships between voltage and current in the R-L filter for both SRF components are obtained [33].

$$V_d(t) = R_{inv} \cdot I_d(t) + L_{inv} \cdot \frac{d(I_d(t))}{dt} + L_{inv} \cdot w \cdot I_q(t) \tag{4.18}$$

$$V_q(t) = R_{inv} \cdot I_q(t) + L_{inv} \cdot \frac{d(I_q(t))}{dt} - L_{inv} \cdot w \cdot I_d(t) \tag{4.19}$$

In equations (4.18) and (4.19) a term that establishes a coupling between both SRF components can be identified. In order to avoid the coupling, a feed-forward compensation is introduced in the control loop.

Another difference with respect to the DC control is that now the voltage obtained has to be normalized by $\frac{V_{DC}}{2}$. As the maximum voltage at the output of the inverter is precisely half the voltage at the input. The normalized value must be changed back to the three-phase system, in order to obtain the three modulators needed to generate the PWM signals for the three branches of the inverter.

4.2.3.- DC Bus Voltage Control

In this subsection, the case in which the inverter's role is to keep the voltage stable on a DC bus is analyzed. To implement this type of control, the power balance between the inverter's input and output ports has to be taken into account. As we are still working on the SRF, new expressions for active and reactive power should be obtained. These are well known in the literature and match

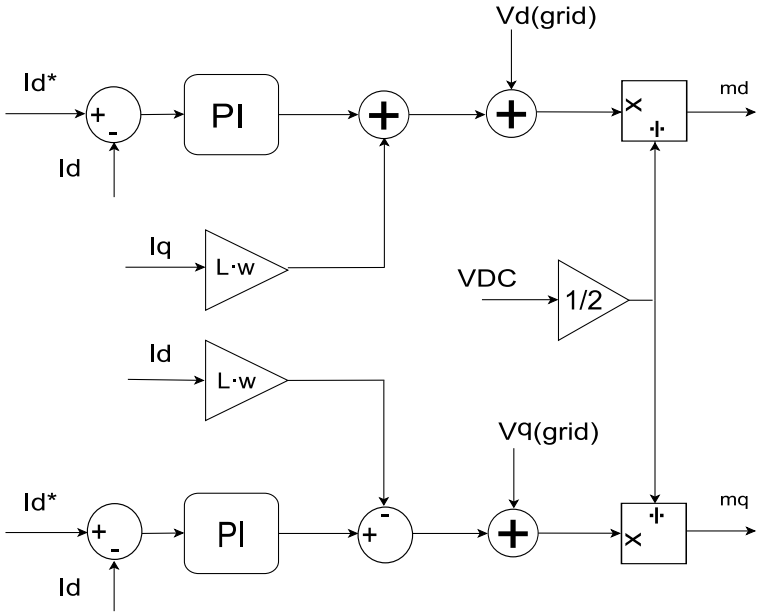

Figure 4.11.- Current control loop for the DC/AC converter.

(4.20) and (4.21).

$$P_{dq} = \frac{3}{2}(V_d \cdot I_d + V_q \cdot I_q) \tag{4.20}$$

$$Q_{dq} = \frac{3}{2}(V_q \cdot I_d - V_d \cdot I_q) \tag{4.21}$$

The most usual approach is to work in a way that the value for V_q is null, as it is aligned with one of the three-phase axes, thereby simplifying both active and reactive power equations.

The control loop is shown in Figure 4.12. The PI regulator is tuned once again following the same approach as for the SBC, the values obtained for the constants can be found in Table 4.5. The input of the PI regulator is the error between the voltage reference for the DC bus and the measured voltage. The control action obtained is then the current I_c that flows through the capacitor. However, the variable we are interested in is the value of the current flowing through the input port of the inverter, I_{conv2}, having again the need to use a feed-forward approach following (4.22).

$$I_{conv2} = I_{conv1} - I_c \tag{4.22}$$

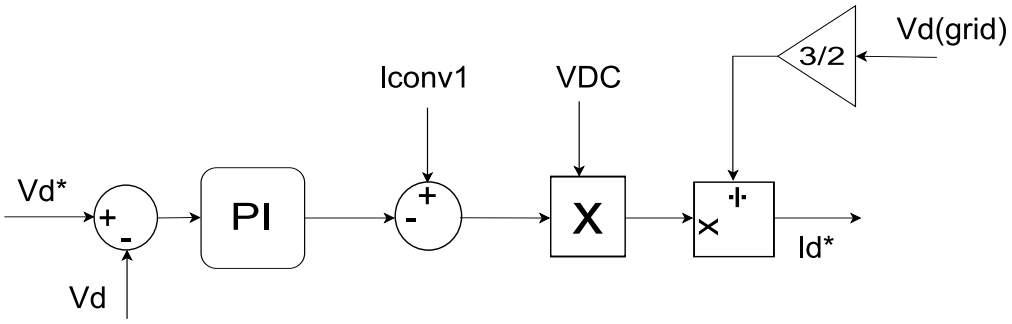

Figure 4.12.- DC voltage control loop for the DC/AC converter.

Once this value has been computed, a reference for the active power can be obtained by multiplying it times the bus' voltage. Then, (4.20) is used to obtain a reference for I_d that is applied to the inner control loop. On the other hand, with the aim to obtain null reactive power, the reference for I_q is directly established to zero.

Table 4.5.- DC Voltage control parameters

Parameter	Value	Unit
Bandwidth	50	$\frac{rad}{s}$
K_p	0.22	
K_i	12.27	

4.2.4.- Power Control

In the case analyzed in the previous section, power references were obtained from the DC bus voltage reference. Now those power references are known, thus being the control loop significantly simplified, as can be seen in Figure 4.13.

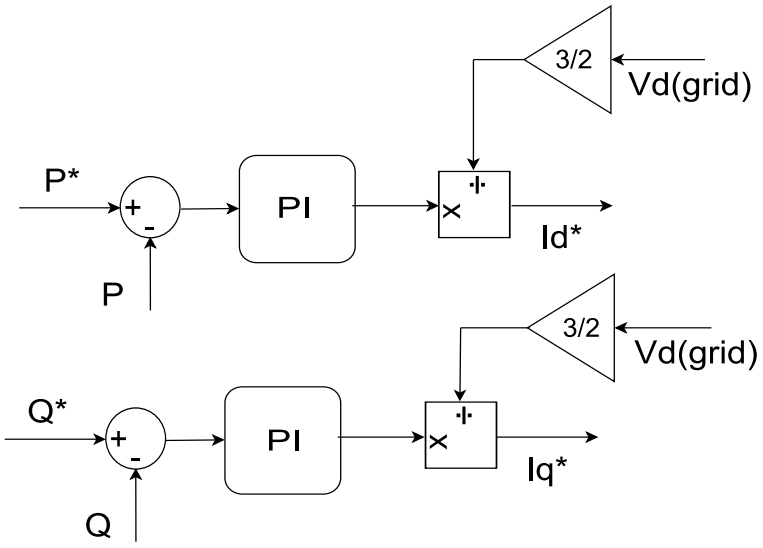

Figure 4.13.- Power control loop for the DC/AC converter.

An important characteristic of this system is that there is no plant that can be used to tune the PI regulators. The input of those regulators is, as always, the error obtained as the difference between the reference and the measured power, while the control action is directly a value expressed in power magnitudes (Watts). The tuning is performed taking into account that the loop's bandwidth has to be at least ten times smaller than the current loop's. As the bandwidth depends mainly on the proportional gain of the controller, this gain is chosen so that it is ten times smaller than the proportional gain of the current controller; while the integral is not modified, as can be seen in table 4.6.

Table 4.6.- Power control parameters

Parameter	Value
K_p	0.31
K_i	31.41

4.2.5.- AC Bus Control

When working with a grid forming inverter, a control over the AC Bus Voltage must be performed following the scheme shown in Figure 3.4. The approach taken is the same as the one implemented when controlling the output voltage on the DC/DC Converter. Also, just like in the current control, two identical loops are implemented; one for each SRF voltage signal component. The control loop for one of the components is shown in Figure 4.14.

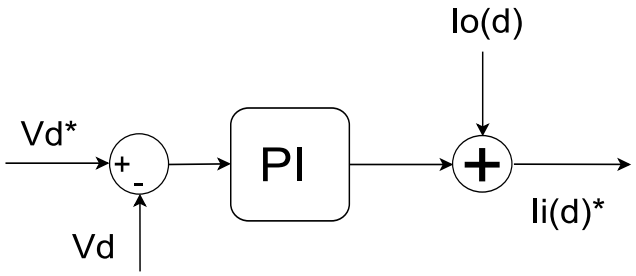

Figure 4.14.- Control loop for the "d"-SRF component of the AC bus.

In this model, a three-phase generator is used to simulate the grid and as a reference signal, since it is considered that at any moment the connection between the MG and the main grid could be made by switching the PCC. However, in applications where the MG is completely isolated, those references would be virtually generated.

4.2.5.1.- Controller Tuning

The Direct Synthesis Controller tuning is used to obtain the parameters for the PI regulators [34]. This consists of making the transfer function of the feedback system match the model transfer function of a second-order system with an additional zero (4.23). Following this approach, expressions (4.24) and (4.25) are obtained, whose numerical equivalent is shown in Table 4.7.

$$J(s) = \frac{V_c}{V_c*} = \frac{\frac{k_p}{C_f} \cdot s + \frac{k_i}{C_f}}{s^2 + \frac{k_p}{C_f} \cdot s + \frac{k_i}{C_f}} = \frac{2 \cdot w_n \xi \cdot s + w_n^2}{s^2 + 2 \cdot w_n \xi \cdot s + w_n^2} \tag{4.23}$$

$$k_p = 2 \cdot w_n \cdot \xi \cdot C_f \tag{4.24}$$

$$k_i = w_n^2 \cdot C_f \tag{4.25}$$

$$w_b = w_n \cdot 0.64 \tag{4.26}$$

To perform the tuning, the relationship between the bandwidth and the natural frequency of the system, shown by (4.26), is also taken into account. Also, the damping factor could be set at its typical value, $\frac{2}{\sqrt{2}}$. However, this is a typical value for ordinary second-order systems, but the particular system being analyzed has an additional zero. By introducing a damping factor greater than 1, the response will still be that of a second-order system with a significantly smaller overshoot.

Table 4.7.- AC Voltage control parameters

Parameter	Value	Unit
Bandwidth	50	$\frac{rad}{s}$
χ	10	
K_p	0.1	
K_i	2.37	

4.2.6.- Frequency Control: Droop Control

There are two alternatives that can be followed in order to implement the frequency control. The first one would be to establish a reference for the active power and to measure the frequency; while the second one would be the opposite, to establish a reference for the frequency and to obtain the value of the power that is applied to the feedback loop. In this project, the second option is chosen, as it gives more stable systems, and its control loop looks like the scheme in Figure 4.15.

Nevertheless, before this control can be applied, different parameters have to be defined. Firstly, the nominal power and frequency of the converter have to be established, as well as the maximum admissible power and the maximum frequency deviation allowed. Once those values are known, the droop-control ramp is obtained. The droop-control can also be applied to recreate the relationship between voltage and reactive power. The approach is the same: obtaining a droop-

control ramp based on the nominal values for the voltage and the reactive power of the converter, and the maximum variations allowed for them. Both ramps are shown in Figure 4.16, and all the parameters are presented in Table 4.8.

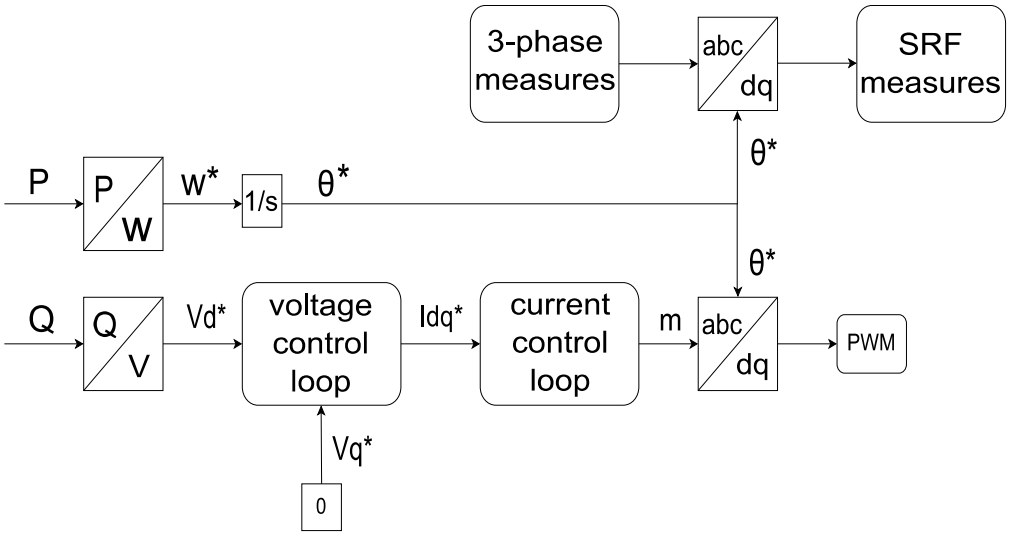

Figure 4.15.- Block diagram for the droop-control.

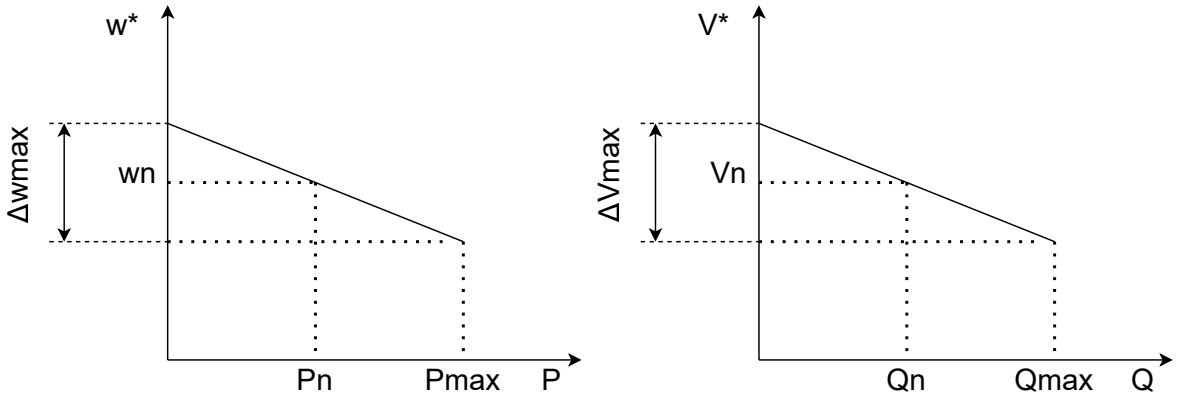

Figure 4.16.- Droop-control ramps: relationships between P-ω and Q-V.

4.2.7.- Virtual Synchronous Generator

Between the different control strategies mentioned in Section 2.4.2.2, through which a VSG can be implemented, the one corresponding to the CCVSM is analyzed. This system performs an active power control based on a frequency droop and a virtual swing equation. The damping term

Table 4.8.- Droop-control parameters

Parameter	Value	Unit
Active Power Reference	1	pu
Reactive Power Reference	0	pu
ΔV	5	%
$\Delta\omega$	2.5	%
K_{droop}	-0.025	pu

is based on an estimated frequency, eliminating the need for a PLL.

The control loop for this system is represented by Figure 4.17. Where H is the inertia constant of the machine, ω represents the angular speed ($f[Hz] = \frac{\omega}{2\pi}$), and D is the damping factor. The values of these parameters are set according to [22], they are both normalized values, as the control loop is implemented on a per-unit system basis. The parameter H, in this per-unit system, represents the time during which the system can operate at its nominal power without any source of energy.

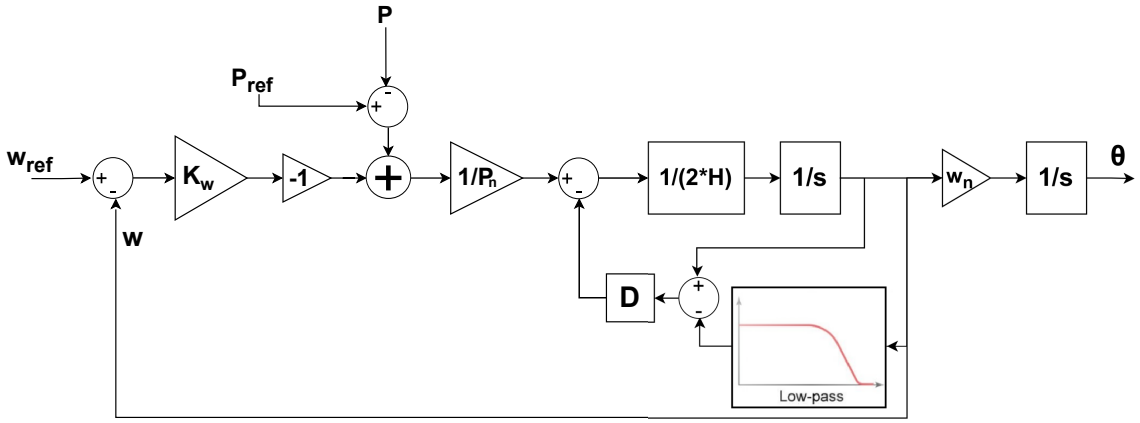

Figure 4.17.- CCVSM control loop.

The gain K_w is the inverse of the droop constant used in the frequency droop control. That is the slope of the P-f droop ramp (4.27). Signals ω_{ref} and P_{ref} could be commands coming from a

secondary control. However, for this scenario, ω_{ref} takes the value of the nominal grid frequency and P_{ref} is the power at which the converter is supposed to operate at rated conditions. All the different parameters are presented in Table 4.9.

$$K_w = \frac{1}{K_{droop}} = \frac{Pn - Pmax}{wn - wmin} \tag{4.27}$$

Table 4.9.- VSG control parameters

Parameter	Value	Unit
K_w	-40	pu
H	1	s
D	5	pu

4.2.8.- Inverter's Operating Mode Change

So far, the case of a grid forming and a grid feeding inverter have been independently analyzed. But the reality is that, on many occasions, the inverter should be able to operate in any mode. This is what it is intended to be carried out throughout this section.

Initially, the inverter is disconnected from the main grid and a voltage control on the AC bus is applied. When this voltage signal is correctly synchronized with the grid's voltage, the PCC can be switched and a power control can then be implemented.

4.2.8.1.- Control System

For this system to work properly, two different outer control loops have to be used, one for each mode; while the inner loop for current control is the same in both cases. From each control loop a reference value for the current is obtained. When the inverter works as a grid forming, the reference obtained from the voltage control loop is applied to the current control system; while when working as a grid feeding, the reference obtained from the power control loop is used. This

process can be seen graphically in Figure 4.18

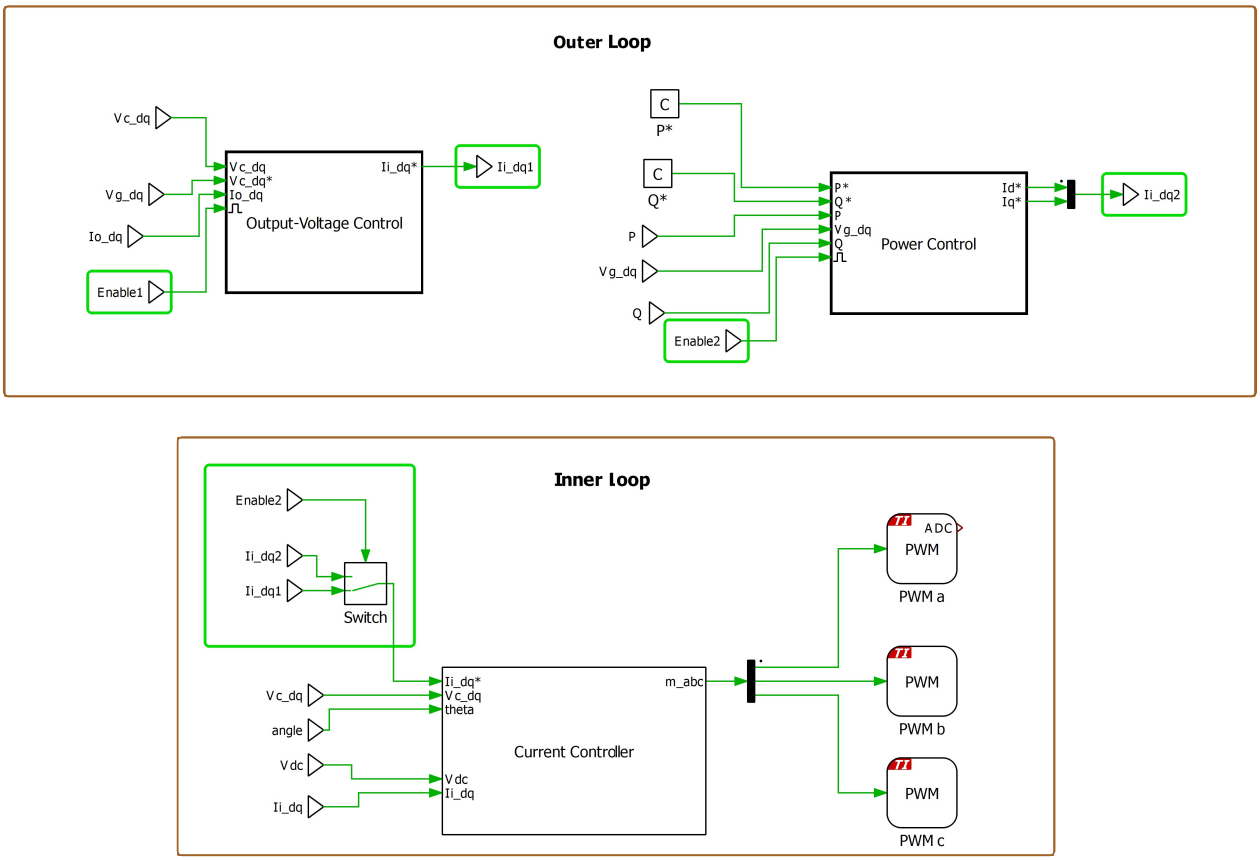

Figure 4.18.- Control system for both power and voltage control.

It is also important to take into account that, when the reference current obtained by one of the outer loops is not being applied to the inner loop, if the outer controller is integrating the error will accumulate. To avoid this problem, the whole outer loop is disabled while not in use, and both loops will never be active simultaneously. To control the enabling of the systems, a state machine is used.

The state machine structure is shown in Figure 4.19. It consists only of two states, connected by a transition. The first state is the one corresponding to the grid-forming operating mode. In this state, the enabling signal for the AC voltage control loop is set to 1, while the one for the power control loop is set to 0. In the second state, which corresponds to the grid feeding mode, the opposite values are established. The transition from one state to another takes place when the

following condition is satisfied: $Sync == 1$ && $CurrentTime > time$. On that condition, ”Current-Time” is an internal variable that takes the value of the actual simulation time, while ”time” is the instant chosen for the PCC connection. The variable ”Sync” is a boolean variable that establishes whether the synchronization has been successful. It is obtained by comparing the absolute value of the difference between the reference voltage, which is the grid’s voltage, and the one measured at the AC bus, with a tolerance value. If the error is smaller than the tolerance, the synchronization is considered adequate and the variable takes the value ”1”, otherwise its value would be ”0”.

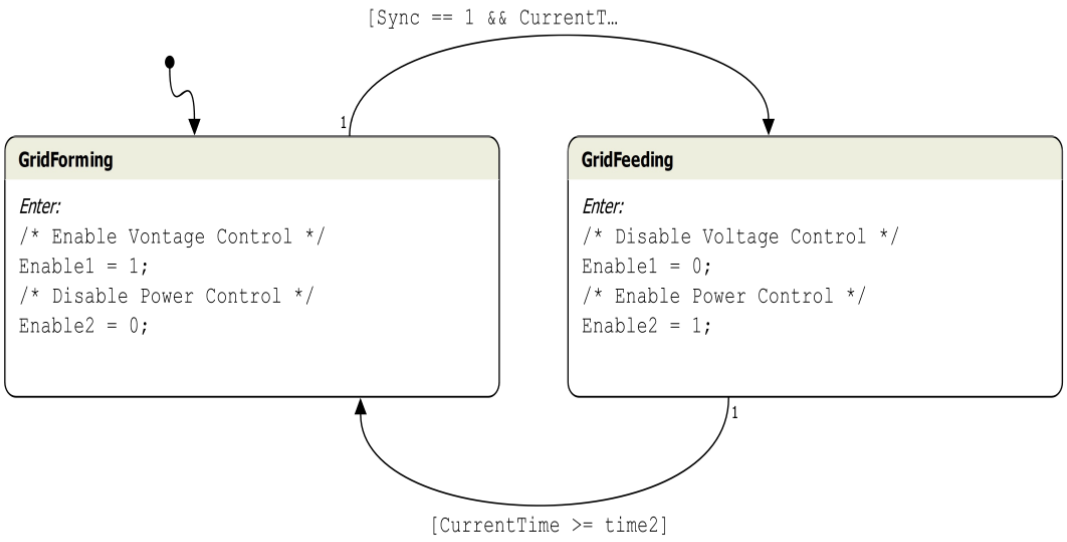

Figure 4.19.- State machine to perform the change of the inverter control mode.

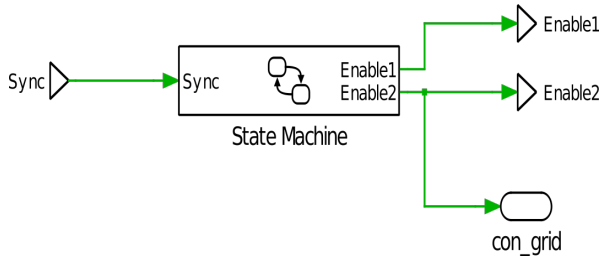

Figure 4.20.- State machine’s inputs and outputs.

The state machine also allows the transition from grid feeding mode to grid forming. In this case, the transition takes place when the time chosen for the disconnection of the PCC arrives. Notice also in Figure 4.20 that the switching of the PCC is commanded by the same signal used to enable the grid feeding control.

The command for the change from one mode to another could have also been done through manual switches controlled by the operator, instead of directly setting the time instants as conditions for the transition. This approach will be followed in the models analyzed later on.

4.3.- MICROGRID MODEL

Once the control of both converters has been analyzed, the next step is to combine them to create a simple microgrid model, which looks like the one in Figure 4.21. The DC/DC converter is still connected to a battery and a DC bus; and to the same DC bus, a three-phase inverter is connected.

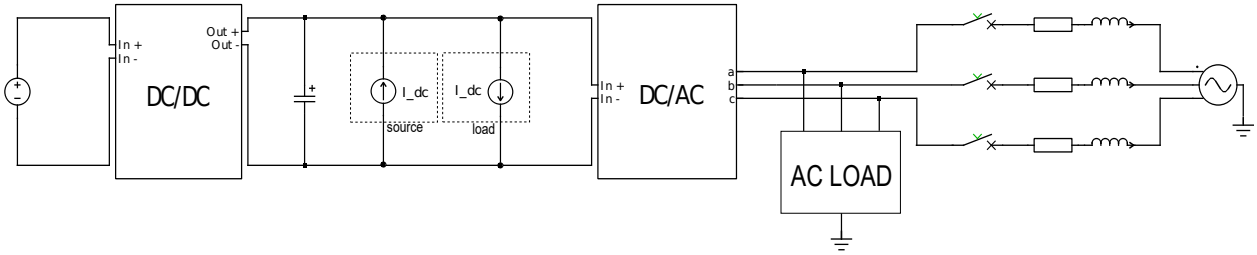

Figure 4.21.- Simulated microgrid model.

In this system, the control is designed so that the inverter does not start working until the DC bus has been stabilized by the DC/DC converter. For that to happen, the state machine of Figure 4.22 is again used.

This machine works as follows: Initially, both the DC/DC converter and the inverter are disabled. For the system to start working, the state machine must receive a logic ”1” through its input channel labeled ”ON”. At that moment, the DC bus will be precharged, and, afterwards, the DC/DC converter control will start working following the voltage reference of $800V$. When the

DC bus is perfectly stable, if a logic "1" is received through the input "Sw_AC", the inverter is to be connected. To ensure that the transition only takes place once the DC bus is stable, a minimum time of $0.5s$ must elapse from the time the reference value reaches 800V until the inverter starts working. Both mentioned inputs are controlled by manual switches.

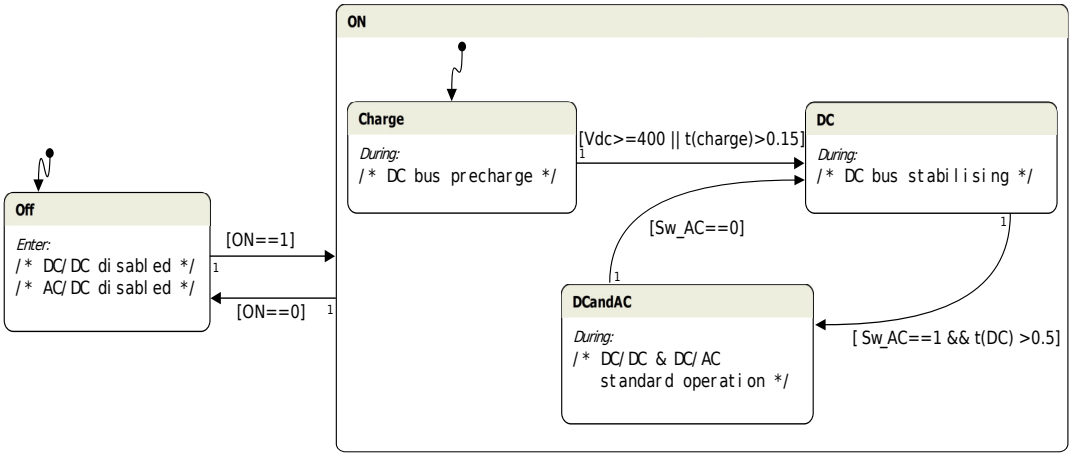

Figure 4.22.- State machine to control the microgrid operation.

5. Hardware-In-the-Loop Implementation

All the designed models are implemented using HIL simulations. In this chapter, the transition from a standard simulation to a real-time HIL simulation is explained.

The workflow followed is presented in Figure 5.1. It begins with the design of a PLECS model, which should perform as expected before configuring the model for real-time simulation. This first step has already been described in the previous chapter. For the standard simulation, both the plant and the controller could be located on the same subsystem. However, to be able to upload the plant to the RT Box and the controller to the MCU, two different subsystems are established. These subsystems will communicate with each other through I/O signals, just as the physical elements do. But these signals need to go through an adaptation process, which is analyzed in the following section.

For all models, a discretization step-size of $\frac{1}{fs} = 0.1ms$ is selected for the control system. The step size chosen for the plant model however will vary. Ideally, 1µs step size should be established, but as the complexity of the model increases the RT Box is not able to execute the model in such a short period of time. Sometimes, if the step size is too small, an overrun error may occur; other times the model can run, but the results are unreliable. Therefore, the step size will vary between 2µs and 5µs, depending on the complexity of the model. The sample step size of the model running on the RT Box can be seen, as well as the current cycle time, on the RT Box Web Interface. The current cycle time shows how long the RT Box takes to execute a model step.

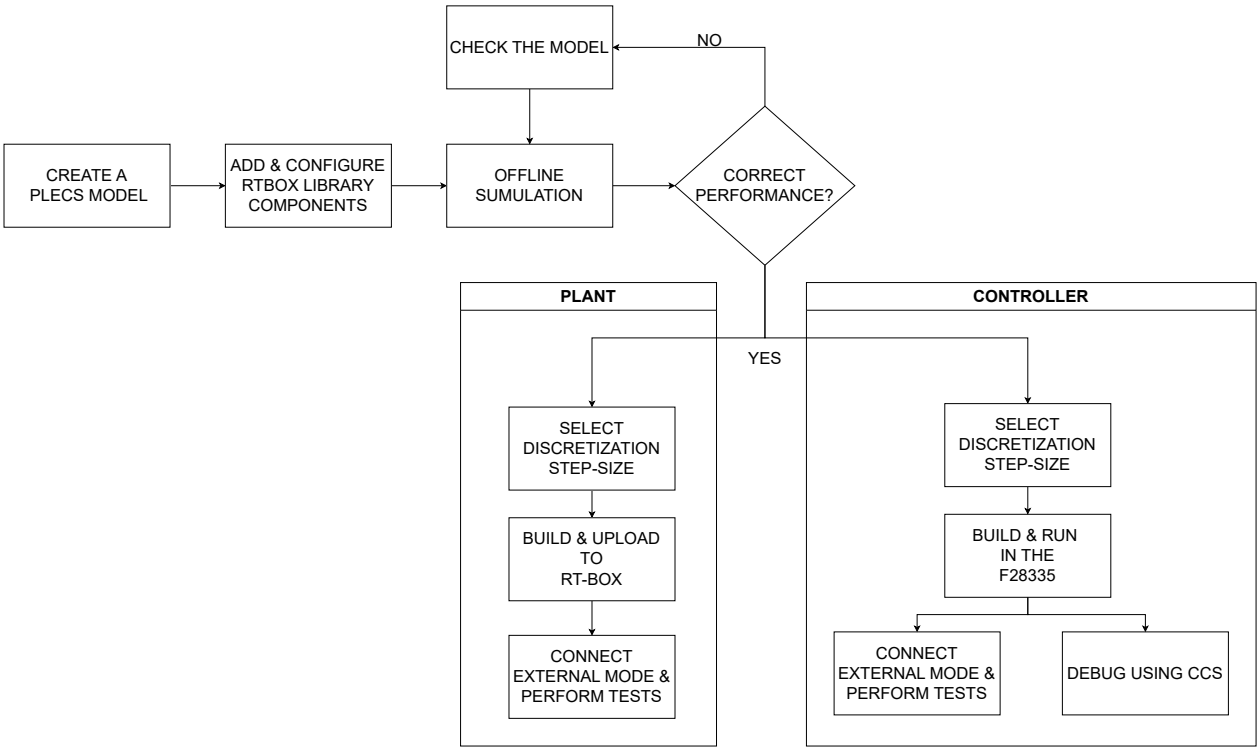

Figure 5.1.- Real-time simulation workflow.

5.1.- SIGNAL ADAPTATION

To allow communication between the plant and the control system, RT Box and TI-C2000 specific components are added to map simulation signals to I/O.

Firstly, the blocks used to generate the PWM signals need to be replaced by ”PWM Out” blocks. These elements can generate two opposing pwm signals, so one transistor branch can be controlled by just one ”PWM Out” block. In the same way, for the plant to capture the pwm signals, ”PWM capture” blocks are used. It is important to notice that both elements work with values between 0 and 1.

The next adaptation is that of the voltage and current signals that need to be measured to carry out the control. These signals are measured in the plant by voltmeters and ammeters, and afterward, they are sent to an ”Analog Out” block. The analog outputs of the RTbox have the internal structure shown in Figure 5.2. This circuit allows the scaling of the voltages to 0V and

3.3V, but it also introduces a gain of 0.65 in between the analog output pins of the RT Box and the analog input pins of the Control Card. Also, notice that no anti-aliasing filter is included. This is because, to implement an anti-aliasing filter, the maximum frequency of the signals must be known. Therefore, to allow the RT Box to work with different models operating at different frequencies, no anti-aliasing filters are introduced. However, those filters are still necessary to obtain the correct measurements, so they must be included inside the model of the plant, between the sensors and the "Analog Out" blocks. On the other hand, the signals are captured at the controller subsystem by an "ADC IN" block.

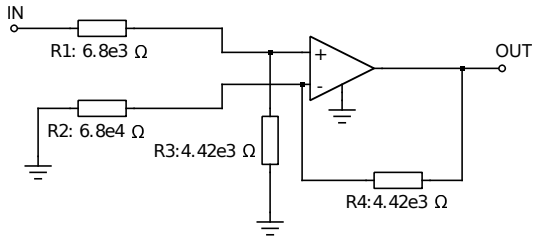

Figure 5.2.- RTbox analog output.

The RT Box analog outputs are already mapped to the Control Card [35]. This implies that, when configuring the input and output blocks, these connections need to be taken into account. For the designed models, the channels shown in Figure 5.3 are chosen for the different signals. The analog output conditioning circuit of the RT Box can scale the voltages at levels appropriate for the microcontroller. However, the scaling parameters and any offset needed have to be defined, both for the plant and the controller parts. To do so, the maximum value for all signals is calculated taking into account the maximum power established as a limit for the power converter (the same for DC/DC and DC/AC). The scale factor is therefore the one obtained when applying equation (5.1).

$$scale = \frac{3.3}{Y_{max}} \tag{5.1}$$

If the signal can take both positive and negative values, an offset has to be introduced as shown in Figure 5.4. This means that an offset of 1.65 is established on the output, and the scale factor looks now like equation (5.2).

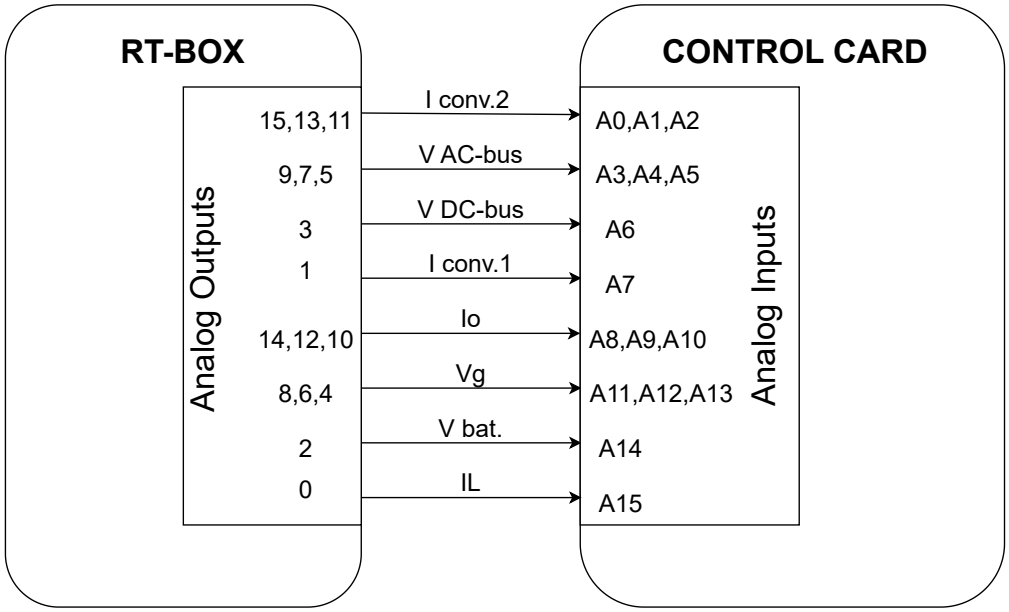

Figure 5.3.- Analog signals exchange between RTbox and the MCU.

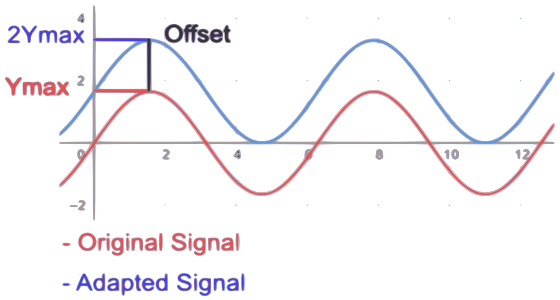

Figure 5.4.- Signal adapted by introducing a DC offset.

$$scale_{RTbox} = \frac{3.3}{2 \cdot Y_{max}} \tag{5.2}$$

All the adaptations seen so far are made on the plant model. However, they need to be undone when received by the microcontroller so that the original measurement is the value to be operated within the control loop. Therefore, on the "ADC IN" block, the scale is the inverse of that applied in the output. It is now important to remember the gain introduced by the RT Box as shown in (5.3).

$$scale_{MCU} = \frac{1}{Gain_{RTbox} \cdot scale_{RTbox}} \tag{5.3}$$

Moreover, to restore the original value the offset, when introduced, has to be eliminated. To do so, an offset of $-Y_{max}$ is introduced on the "ADC IN" block for the corresponding signals. The operation that the ADC block performs equation 5.4.

$$y = input \cdot scale_{MCU} + offset \tag{5.4}$$

5.2.- ADC SAMPLING

The TI C2000 Target Support Package allows the synchronization of the ADC operation with the PWM signal generation. This is done through the ADC triggers and task trigger signals as shown in Figure 5.5. The ADC works through two interrupts, the start-of-conversion, and the end-of-conversion interrupts. When the start-of-conversion interrupt is triggered, all ADC channels associated with the unit are converted sequentially. This interrupt can be triggered by the PWM so that the conversion begins once the PWM carrier reaches an underflow or overflow. On the other hand, by connecting the control task trigger to the ADC unit task signal, the ADC end-of-conversion interrupt triggers the control task, ensuring that the ADC registers are updated before executing the following control loop.

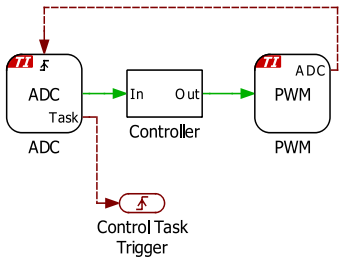

Figure 5.5.- ADC sampling and PWM synchronization.

5.3.- EXTERNAL MODE AND PARAMETER INLINING

The External Mode allows the visualization of real-time signals on the Scopes included in the model, and it can be activated for both the plant and the controller. While the PLECS model is connected via the External Mode, no modifications can be made to the model. However, if any

modification to a certain element or parameter needs to be made in real time, the specific block can be added to the parameter inlining exceptions list.

5.4.- SYNCHRONOUS BOOST CONVERTER

5.4.1.- Open Loop Operation

Firstly, a simple model just containing only the SBC and a PWM generator as the controller is implemented. The duty for the pwm signal is obtained directly from a constant block, which is included in the parameter inlining exceptions list so that its value can be modified in real time. For different values of the duty cycle, different voltages should be obtained at the DC bus. With $d = 0$, the lower-side switch should be closed while the high-side one stays open. The opposite situation should be obtained for $d = 1$. For other values between 0 and 1, it is expected that, the lower the duty the higher the voltage.

Once the performance of this initial model has been verified, the next step is to include the designed control system.

5.4.2.- Closed Loop Operation

In Figure 5.6 it can be seen how the measurements go through anti-aliasing filters (implemented by second-order transfer functions) before going to their corresponding analog outputs. Also, the precharge system is included, as well as a resistive load connected to the DC bus through a breaker. The switching command for the two breakers included is obtained from digital inputs of the RT Box, which are connected to the digital outputs of the Control Card.

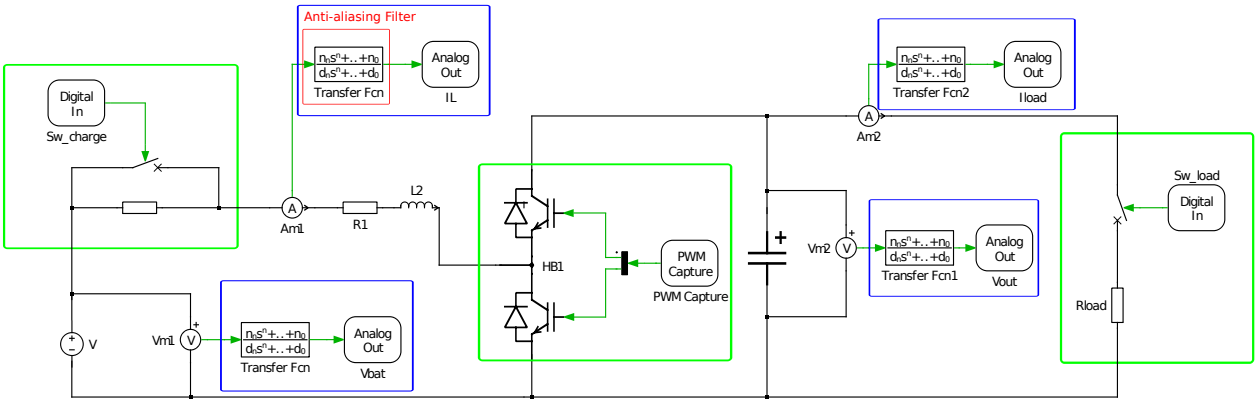

Figure 5.6.- Synchronous Boost Converter Model Adapted for the HIL simulation.

Inside the controller subsystem, the state machine of Figure 5.7 is introduced. It is used to control the charging state of the DC bus and to ensure that, once the bus is charged and the control is going to be applied, all PI signals are reset. The state machine also controls the connection of the load to the DC bus.

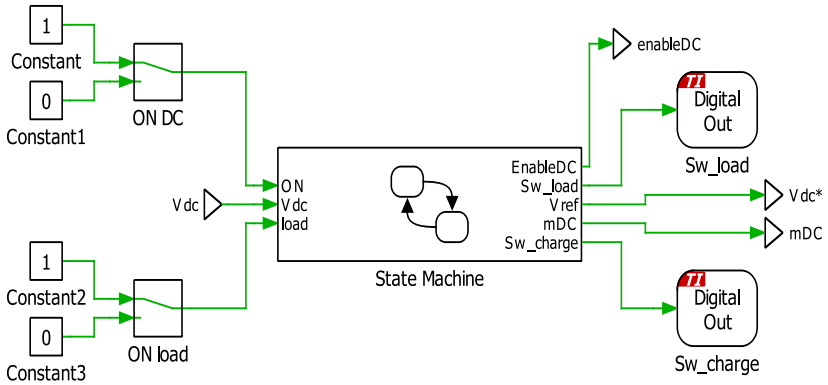

Figure 5.7.- State Machine for the SBC Control.

In Figure 5.8, the different states in which the system can be found are described. The default transition goes to the ”OFF” state, during which no pwm signal is controlling the switching of the IGBTs. If the state of the manual switch labeled ”ON” is changed from ”0” to ”1”, then the converter starts working. On this second state, two sub-states are present, one of them corresponding to the precharge of the DC bus and the other to the normal operation of the converter. During the state of normal operation, the load can be connected to the DC bus by flipping the switch with

the "LOAD" label. The transition from the charging state to the normal operation mode takes place when a time $tc = 0.15s$ has elapsed inside the charging state. This is so because at that time the voltage at the capacitor, which follows (5.5) while charging, reaches its final value.

$$V(t) = V_f \cdot (1 - \exp \frac{-tc}{R_{pc} \cdot C}) \tag{5.5}$$

Both switches connected to the inputs of the state machine are included in the parameter inlining exceptions list so that they can be acted upon during the real-time simulation with the external mode activated.

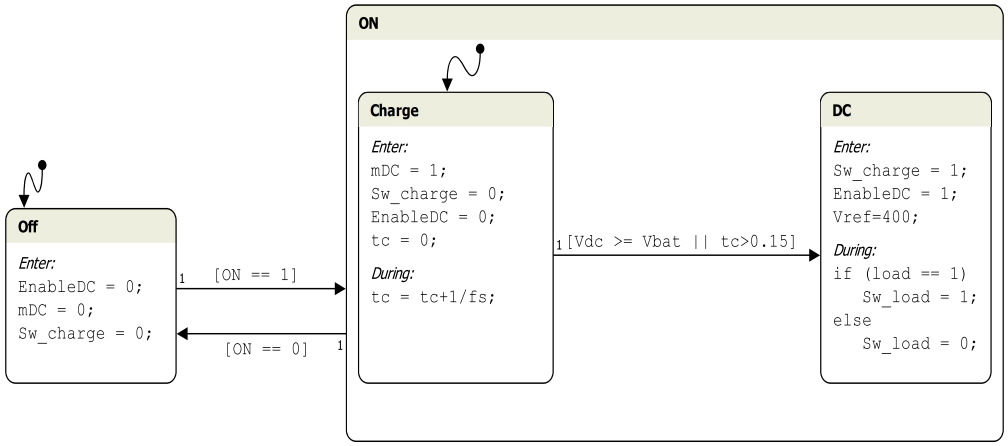

Figure 5.8.- Description of the State Machine for the SBC Control.

5.4.3.- CCS code

The designed model can be directly programmed in the MCU device from PLECS (model-based programming). However, a second approach that can be taken is to generate code into a CCS project to then build and program from there. The CCS project will have a source and header file for defining the control system, as well as an additional source file for the state machine functions.

On the control system source file, two main functions can be distinguished. These are the initialization function and the step function. The initialization function will only be executed once at the beginning of the simulation, while the step function is executed periodically. This structure is presented in Figure 5.9, and there it can be seen that the process taking place inside the step function is practically the same as the one defined in Section 4.1.4 (page 31) for the update function.

On the state machine's source file the different states, transitions, and actions are defined. Afterwards, the main function can be found. This is the one referred to inside the step function in 5.9. With Code Composer Studio it is easy to debug the program and make any needed modifications. In the CCS environment, the real-time ADC readings performed by the MCU can be seen, as well as the value of any other variables like, for instance, the inputs of the state machine. In Figure 5.10, on the left picture, the different ADC channels are shown, while the right-hand side variables correspond to the state machine's inputs.

The "Manual Switches" can also be operated from CCS, by manually changing the values of the corresponding internal variables, which are the ones shown in Figure 5.11

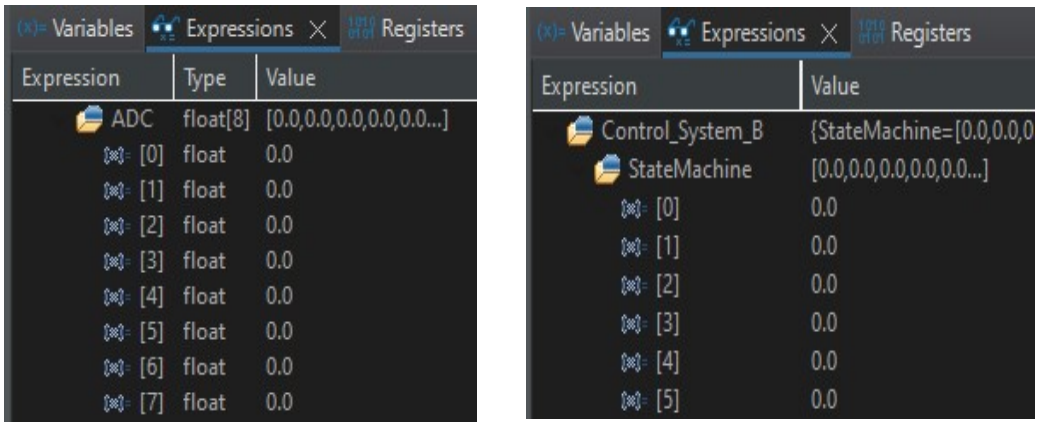

Figure 5.10.- CCS Variable Visualization.

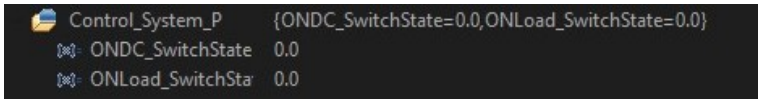

Figure 5.11.- Variables Associated With the Manual Switches' State.

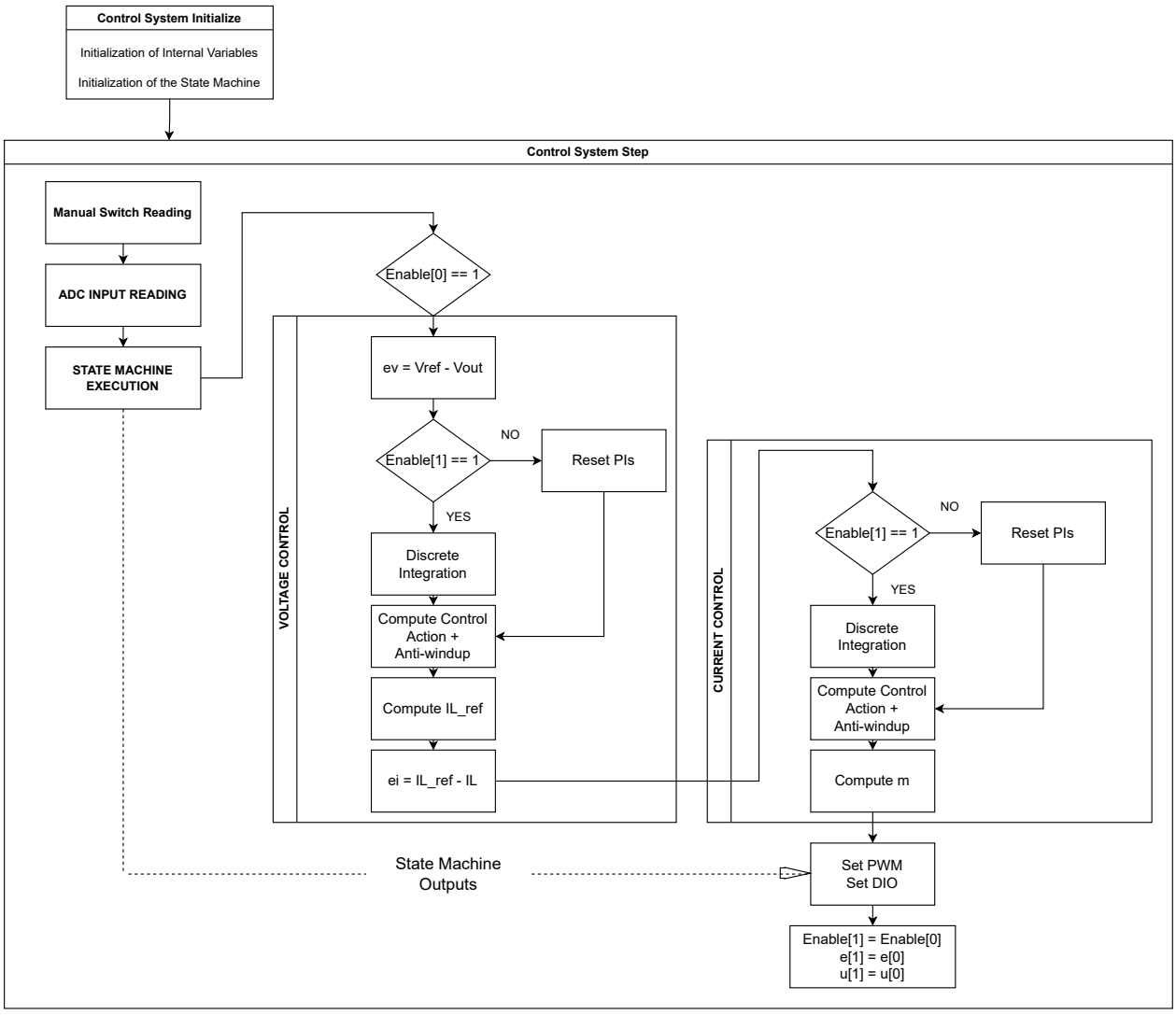

Figure 5.9.- CCS Code: Flow Diagram.

5.5.- THREE-PHASE INVERTER

For the three-phase inverter, all the different control modes are tested in HIL simulations. For all of the models, in addition to the adaptations needed for transmission of the measured analog signals, how the PWM signals are obtained needs to be modified.

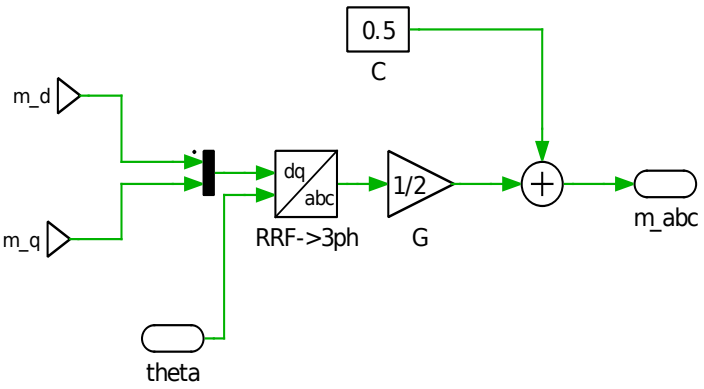

Figure 5.12.- Level Adaptation for PWM Generation.

In the standard simulation models, the modulating signals for the PWM generation, obtained as a result of the current control, were sinusoidal signals with values ranging from -1 to 1. However, as was previously discussed, the TI-C2000 PWM generator blocks can manage signals whose values vary from 0 to 1. The adaptation shown in 5.12 is therefore made.

Other relevant modifications introduced to the models to perform the HIL simulations are explained in the following subsections.

5.5.1.- Power Control

In the power control system, power references are introduced with constant blocks included in the parameter inlining exceptions list. This allows the modification of the references while the simulation is running. Also, the real-time changes in the signals can be observed in the scopes. It is relevant to mention that power references can be positive or negative, the system will be able to follow them in both cases.

5.5.2.- Inverter's Operating Mode Change

The same approach as for the SBC controller's state machine is taken. However, now the transition from one state to another is made when a manual switch is acted upon. Also, the signal which commands the connection of the PCC goes from a digital output of the control card to a digital input of the RT Box.

The signal coming from the manual switch is initially ”0” and the inverter will be forming the grid, but during the real-time simulation, the switch can be flipped indicating that the connection with the main grid should be made. At that moment, if the condition of synchronization is satisfied, a logic ”1” is set at the corresponding digital output of the control card. While the inverter is working as a grid feeding, meaning that the connection has been established, the power references can be modified as desired. Moreover, the switch could be flipped again to go back to a grid-forming operation.

6. Simulations and HIL Results

In this section, some of the results obtained after performing simulations for every type of control analyzed in previous chapters are presented:

- The DC bus' voltage response when being controlled by the SBC and by the inverter is given. The disturbance rejection is also analyzed in both cases.
- The inverter operation acting both as a grid-forming and a grid feeding is presented, as well as the results obtained when both models are combined and the state machine is introduced.
- The grid-supporting operation of the inverter is also analyzed. The results obtained with a droop-control and with the VSG are compared.
- The operation of both converters is analyzed for the complete microgrid model. The effect of introducing DC loads and DC sources is shown.
- Finally, some real-time simulation results are presented.

6.1.- SYNCHRONOUS BOOST CONVERTER

On the simulation results, shown in Figure 6.1 two main stages are clearly differentiated. Firstly, the output voltage increases following the exponential function typical of the charging of a capacitor, as was expected. Afterwards, the control system is applied and the voltage rises, reaching the reference value (800 V) with a 95% settling time of 6.6ms. In Figure 6.1, it can be noticed that the response has a small overshoot, this appears because no DC load has been connected at the beginning of the simulation and, for the tunning of the controllers, it was assumed that a load would be connected. In that scenario, the response is the one presented in Figure 6.2.

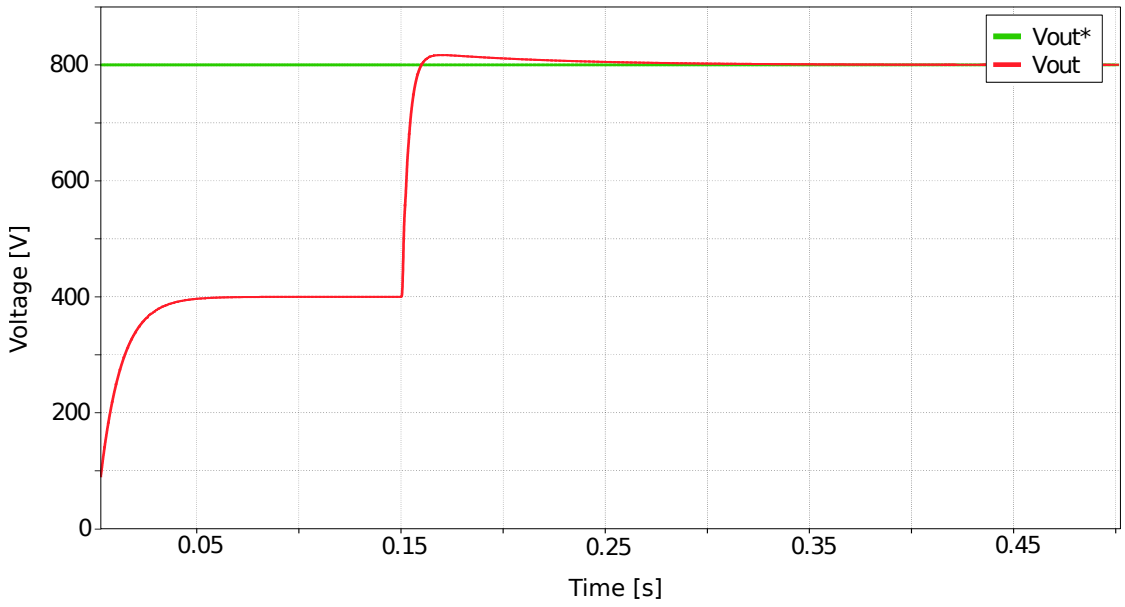

Figure 6.1.- DC voltage control simulation output (I) with no load.

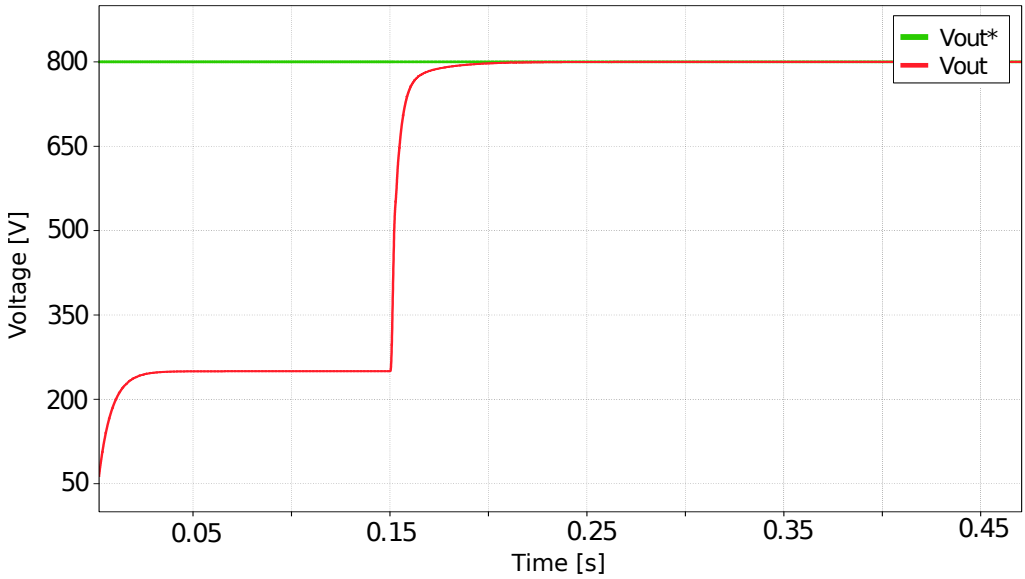

Figure 6.2.- DC voltage control simulation output (II) with load.

Going into more detail, the system's response to disturbances is analyzed. On one hand, changes in the input voltage (the battery) have almost no effect on the response, as can be seen in Figure 6.3. When a step is introduced, making the voltage at the battery drop from 400 V to 350 V, there is only a 1.6V voltage drop at the capacitor, which is equivalent to 0.2%.

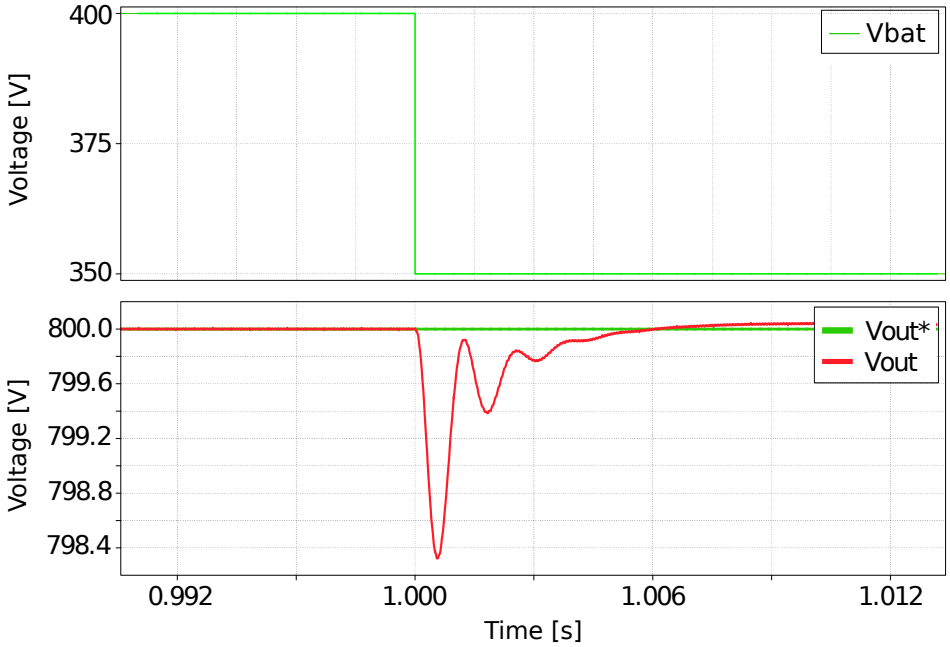

Figure 6.3.- System's response to changes in the input voltage.

On the other hand, in Figure 6.4, the effect of a load disturbance is shown. When a new load is connected in parallel, the current needed at the output so that the voltage is kept constant increases. This implies that at the instant the load is connected, a voltage drop takes place of 3% takes place.

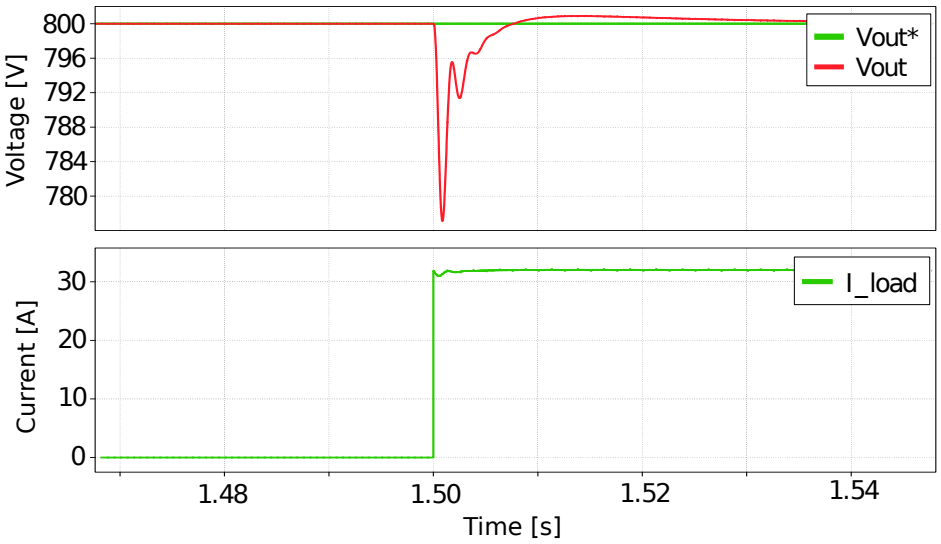

Figure 6.4.- System's response to a load disturbance.

In both cases, the measured voltage reaches again its nominal value at, approximately, 6.6ms. Taking all these factors into account, the control system's disturbance rejection can be considered adequate.

6.2.- INVERTER CONTROL

6.2.1.- DC Bus Control

In Figure 6.5, the evolution of the DC bus voltage, controlled only by the inverter, can be seen. The initial transient appears due to the time it takes for the PLL to measure the phase accurately. If this transient is disregarded, the response can be analyzed as that of an underdamped system.

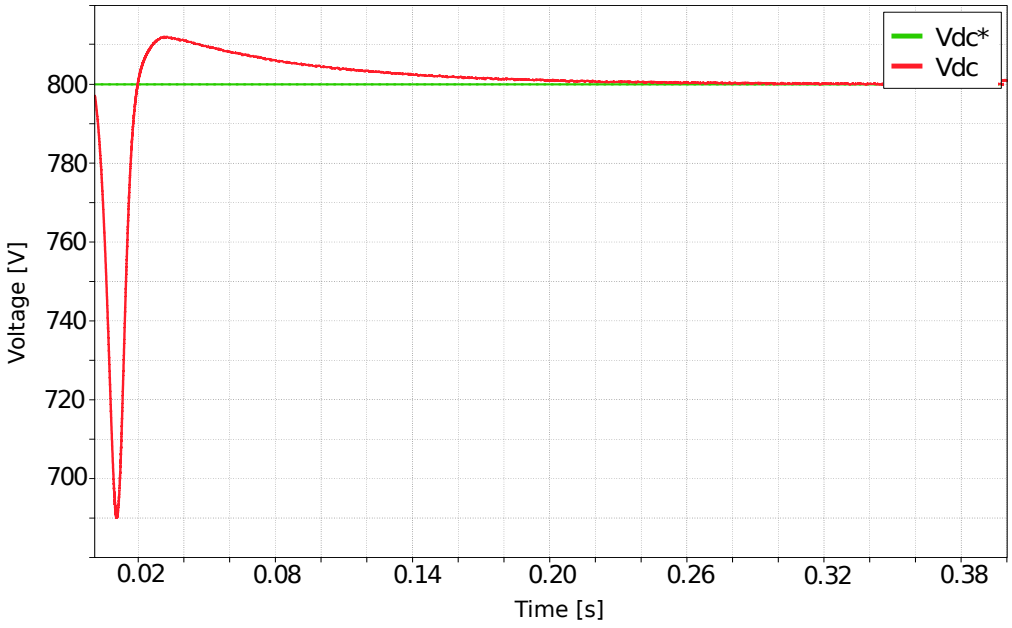

Figure 6.5.- DC bus voltage initial response.

When the step on the reference is introduced, the settling time is 0.9ms and a 0.3% overshoot is obtained, as shown in Figure 6.6.

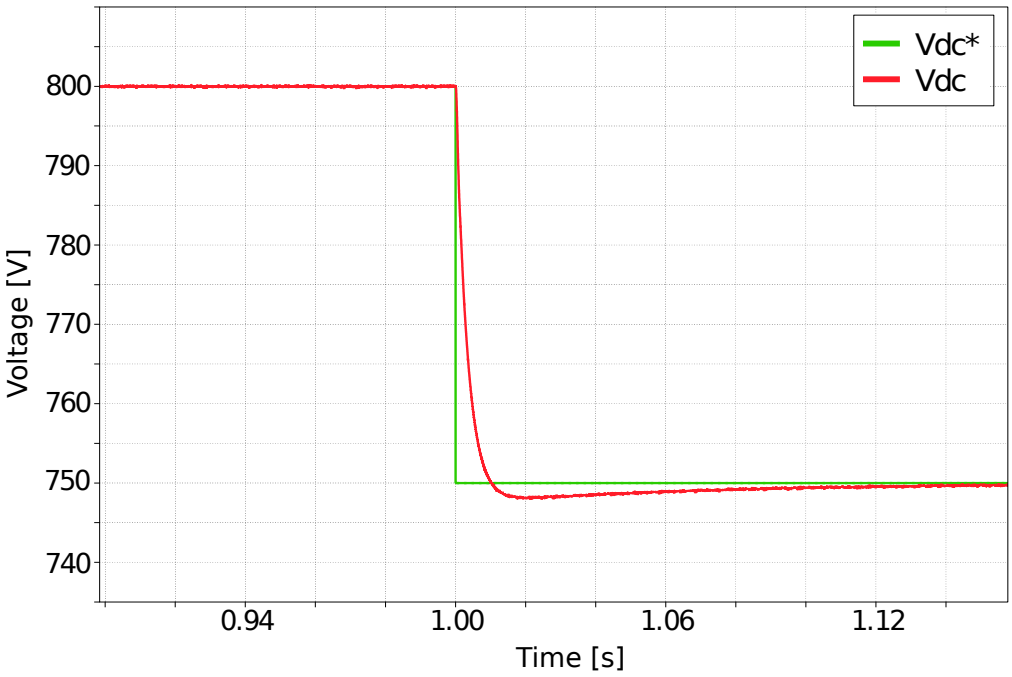

Figure 6.6.- DC bus step response.

6.2.2.- Grid Forming Inverter

Analyzing the response of the grid forming inverter control system, shown in Figure 6.7 a similar transient state as in the previous case can be seen. It is even more noticeable in this model that the transient is due to the synchronization of the PLL, as it is also present on the reference signals. If the PLL were ideal, these signals would be constant from the beginning, as they are directly obtained from a three-phase generator. Nevertheless, once the PLL synchronizes, the outputs follow the references without any major problems. The three-phase voltage obtained can be seen in Figure 6.8.

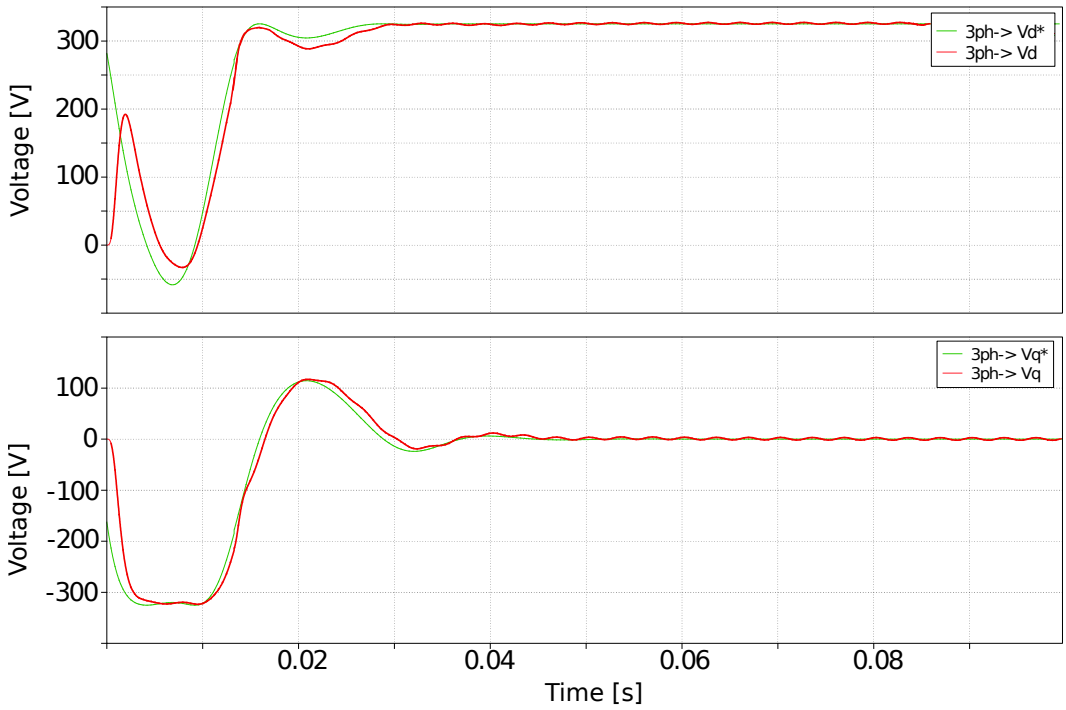

Figure 6.7.- AC bus voltage response (SRF).

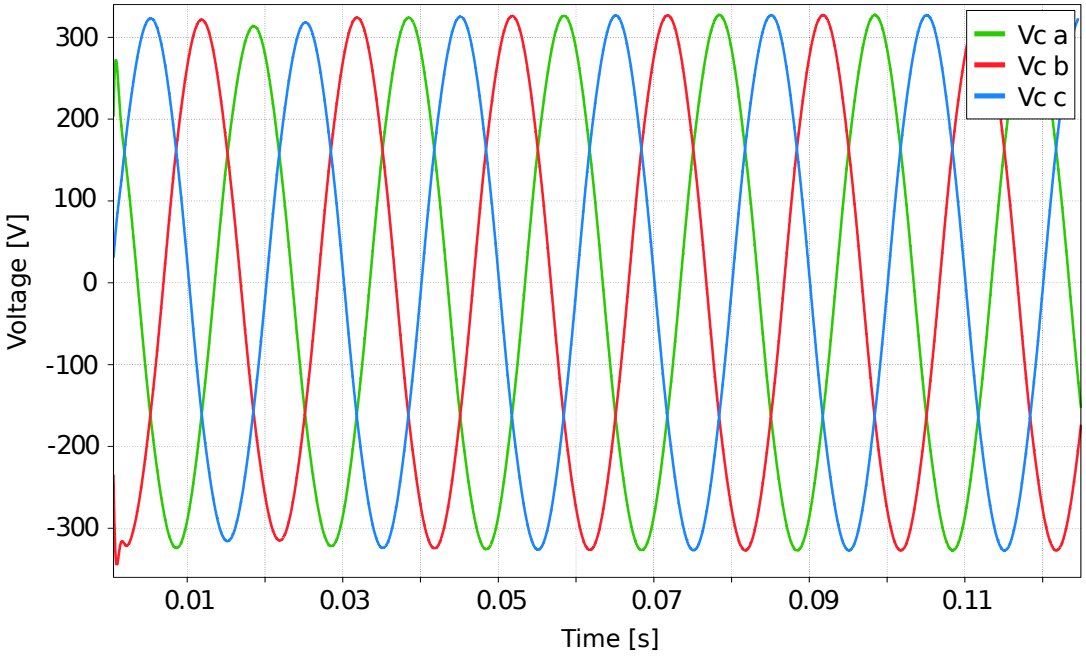

Figure 6.8.- AC bus voltage response (Three-Phase).

As always, the response to disturbances is analyzed. In this case, the usual disturbance is

the connection of a new load to be fed by the grid-forming inverter. In the model under study, a resistive load, whose value is the same as that of the original one, is connected. This means that the current needed at the output must double its value, as the total resistive load is reduced in half. The effect of the perturbance introduced at 0.15s can be appreciated in Figure 6.9 for different values of damping factor (ξ).

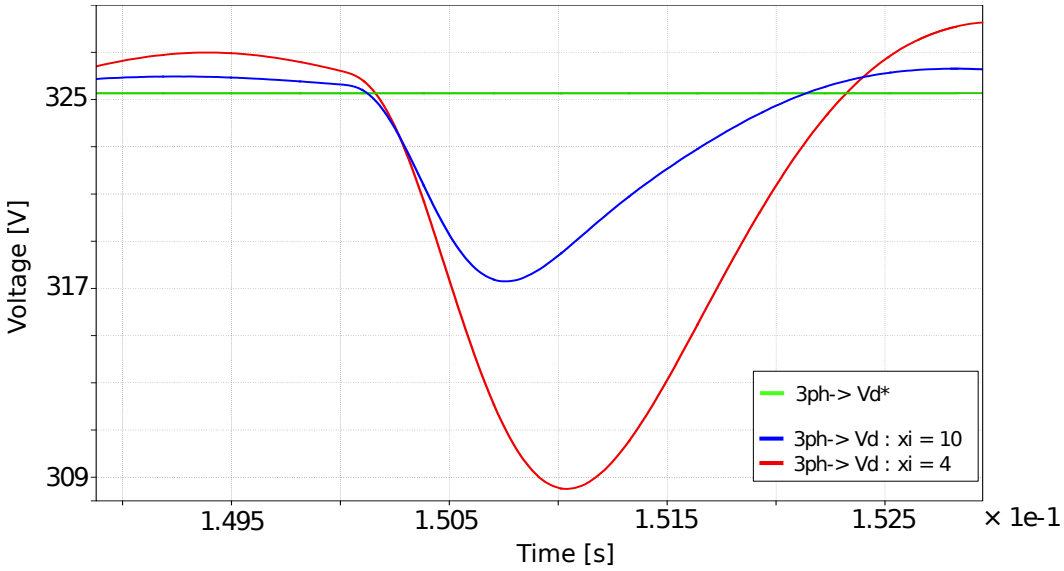

Figure 6.9.- Disturbance rejection depending on the damping factor.

For $\xi = 4$, the "d" component of the voltage suffers a reduction of approximately 25%, but it can recover its nominal value in less than 0.01s. For a higher value, $\xi = 10$, the reduction is only a 16% and the recovery takes place even quicker. On the other hand, as the load introduced is purely resistive, the "q" component is not affected by the perturbance.

6.2.3.- Grid Feeding Inverter

In order to set proper references, great knowledge of the grid to be fed by the inverter is necessary. In this model, the reference for the active power is set at 0.6pu (remember that the rated power (1pu) is 25kW), and that of the reactive power at 0.2pu.

The response shown in Figure 6.10 corresponds to what was expected. The value of the measured power increases slowly until it reaches the reference, with a null error in steady state.

When a resistive load is connected at 0.25s, which corresponds to a disturbance, the control loop's output does not undergo any changes.

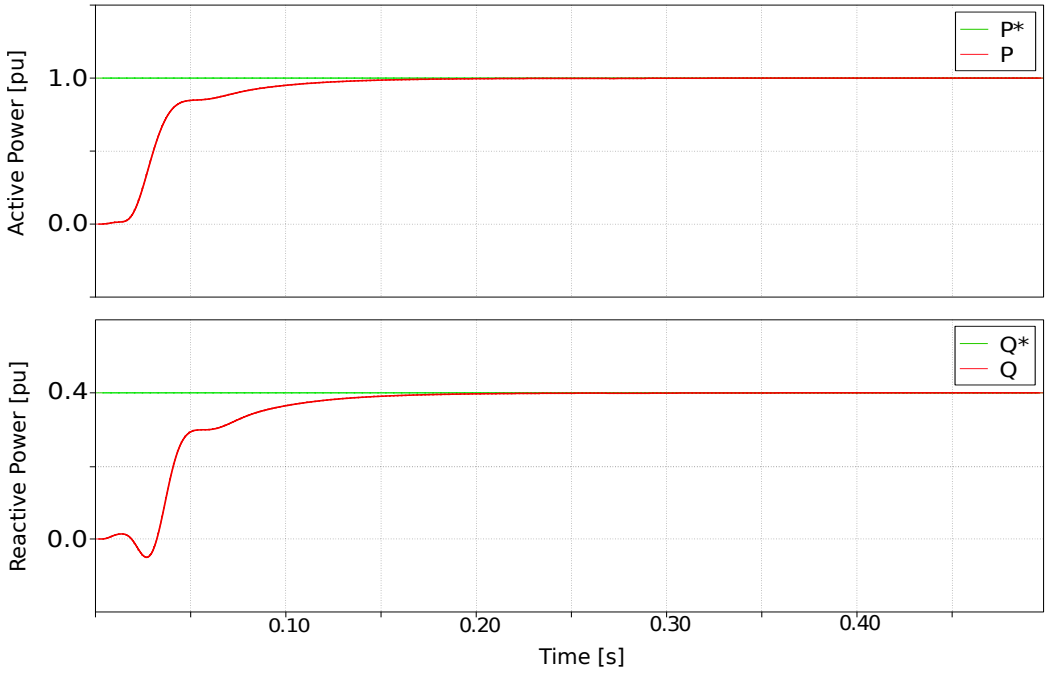

Figure 6.10.- Grid feeding inverter control response.

6.2.4.- Droop Control

For the simulation model, a frequency variation of ±2.5% is set as a limit, and the power rating is established accordingly to the connected load. The load initially connected is purely resistive and demands approximately the nominal power of the converter (1 pu). At 0.5s simulation time a new resistive load is connected in parallel, this brings the inverter to a new operating point where the power it injects almost doubles the rated power (2 pu), which is also the value set as maximum power.

The results of this simulation are presented in Figure 6.11. It can be observed that after an initial transient, the frequency settles around its nominal value, 50Hz, decreasing towards 48.75 Hz when the step in the load is introduced. These results are in line with the expectations.

For the Q-V control, the nominal voltage is 230 V (RMS) and the nominal reactive power

is set at 0var, as the load initially is purely resistive. This means that no reactive power will be injected unless it is demanded by the load. A deviation of ±5% is established for the voltage.

Now the system follows changes in reactive power, which means that an inductive (or capacitive) load needs to be connected to do the test. It is essential to notice that if a purely inductive load were connected in parallel, a resonant (and unstable) system would be obtained. To avoid this problem, resistance is added in series with the inductances. The result of the simulation can be seen in Figure 6.12. The magnitude of the voltage is measured by the ”d” component in the SRF, and it can be observed that, as more reactive power is demanded by the load, the voltage's magnitude decreases.

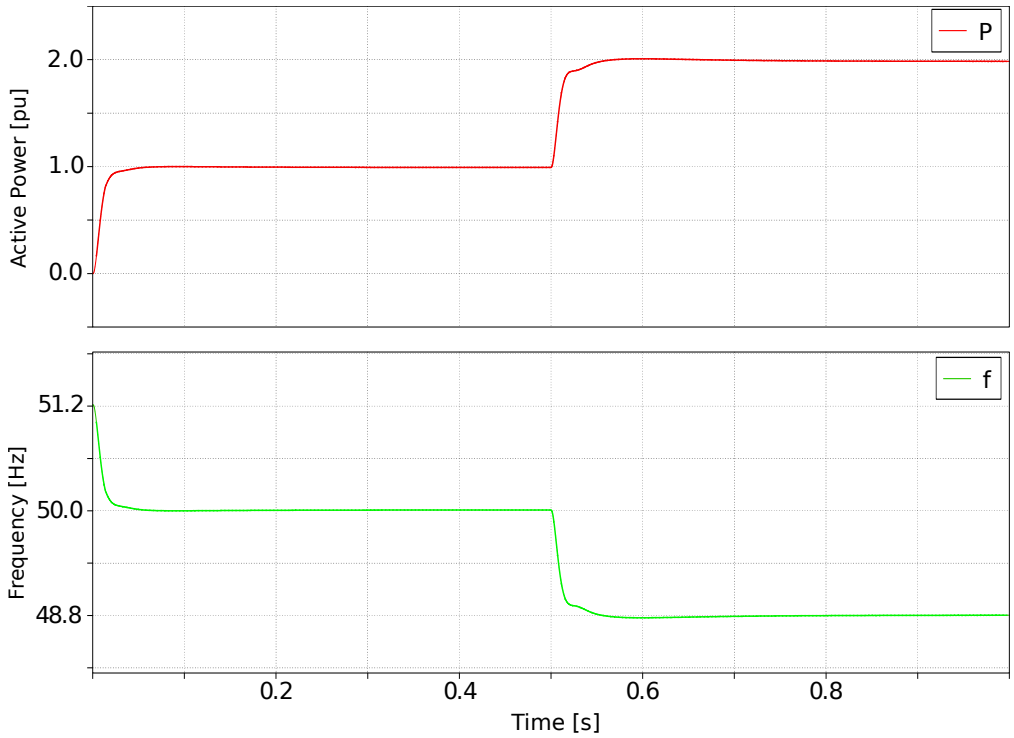

Figure 6.11.- P-F control system results.

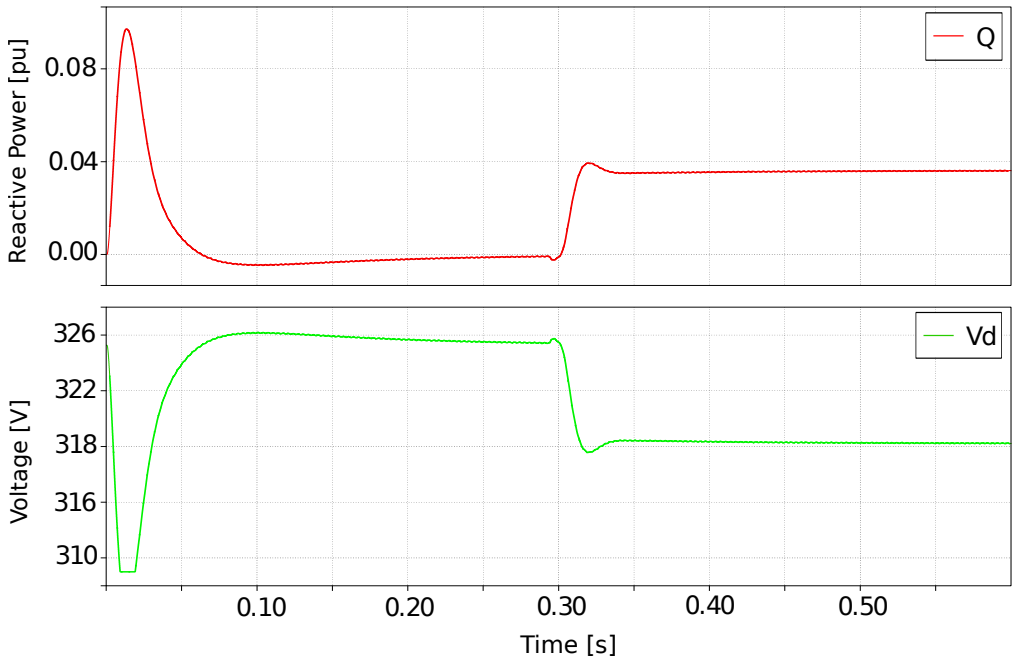

Figure 6.12.- Q-V control system results.

6.2.5.- Virtual Synchronous Generator

Firstly, the control loop from Figure 4.17 is tested without introducing the inverter model as a plant. In other words, the power measurement that would be computed from the voltage and current at the inverter output is now obtained directly from a ”Step” block, which allows the power to change from one constant value to another at a given instant in the simulation.

Under these conditions, the results obtained for different values of H and D can be evaluated in Figure 6.13. Also, the VSG frequency response is compared to the one obtained when a droop control is applied. It can be seen that the change in frequency is smoother for the VSG responses, as one would expect from the theoretical basis. Also, it can be appreciated that the parameter that has a higher influence on the speed of the response is the parameter H, which corresponds to the value of the inertia.

Once it has been verified that the system is working as intended, the modeled inverter is included as the plant to which this control loop is applied. The comparison between the droop control and the VSG control for this scenario is presented in the graph of Figure 6.14.

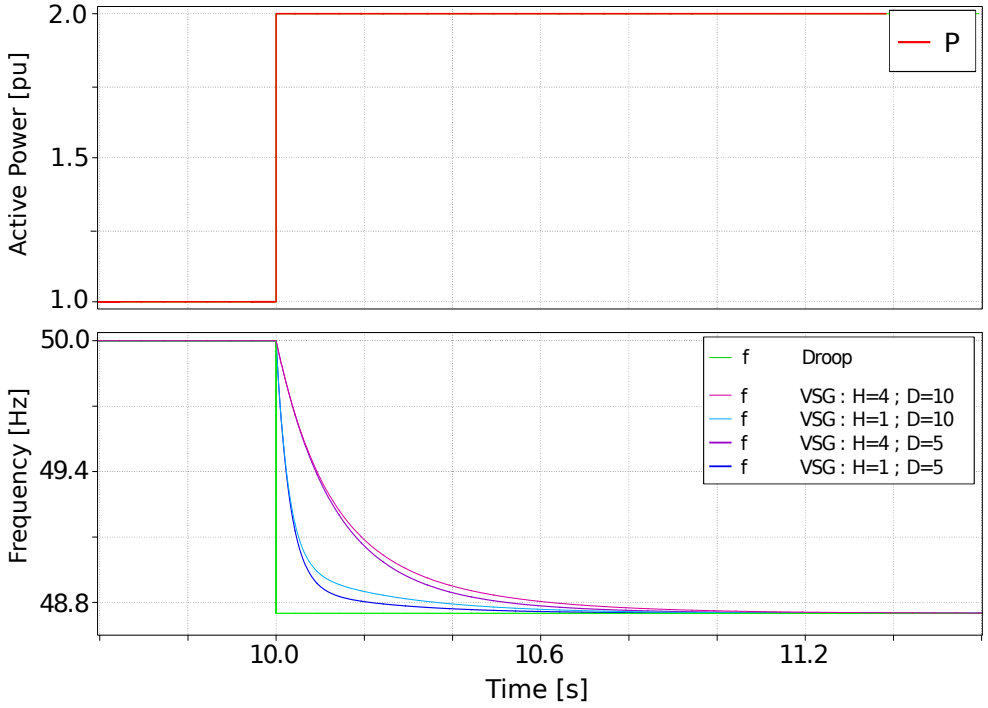

Figure 6.13.- VSG control: frequency response to a power step input for different inertia constants.

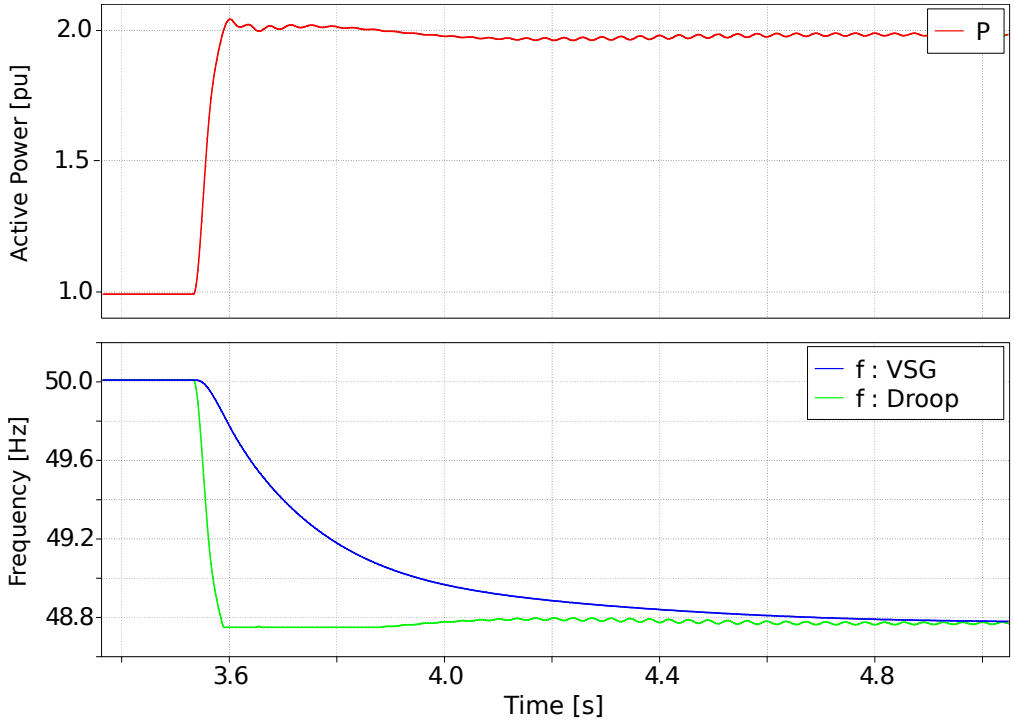

Figure 6.14.- VSG inverter frequency response.

6.2.6.- Inverter's Operating Mode Change

This section is a combination of the two above. Therefore, no surprises are expected in the results. As was previously established in the equivalent section of Chapter 3, the first operating mode is the grid forming one, and, after the PCC's connection, the grid feeding mode is established. This connection is intended to take place at a simulation time of 0.25s if the synchronization is adequate.

It can be seen in Figure 6.15 that, the voltage signals disappear from the graph exactly at 0.25s. This means that the voltage control subsystem has been disabled, and therefore the transition between states has taken place. At that same time, the signals appear on the power chart, meaning that this subsystem has been enabled.

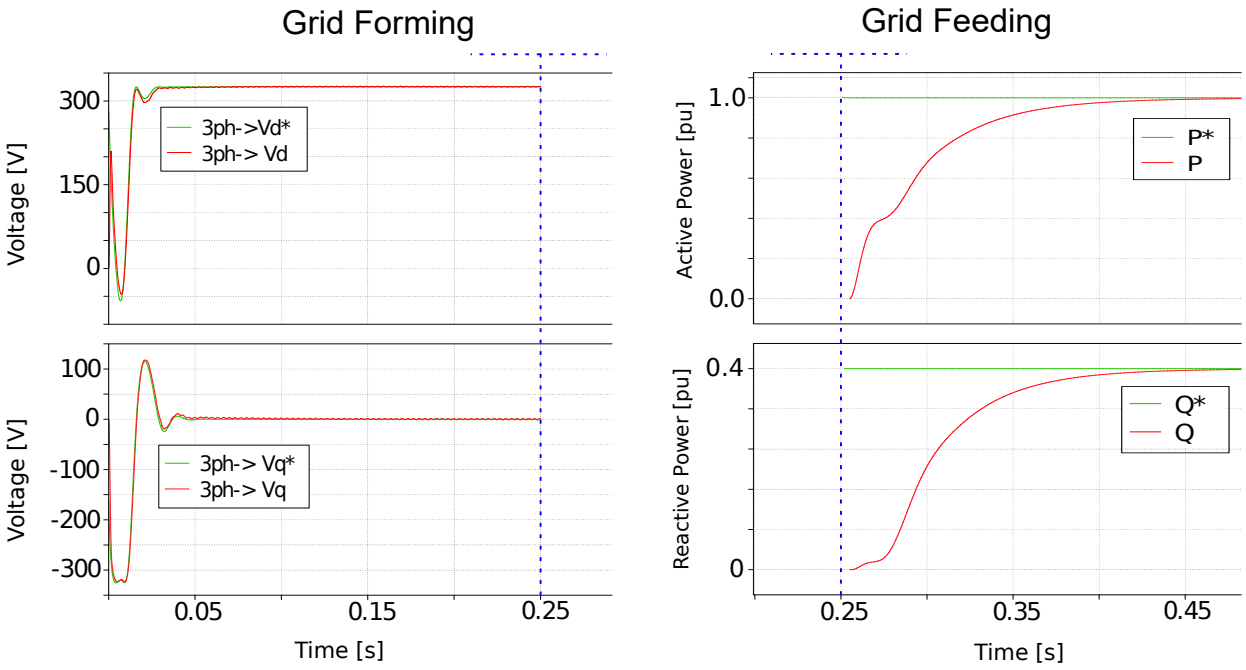

Figure 6.15.- Grid forming & Grid feeding response.

As all PIs are reset when the change in control mode takes place, the inverter can go back and forth from one mode to another without making the system unstable and always following the proper reference.

6.3.- MICROGID MODEL

To analyze the system under different conditions and transitions, the following states are defined according to the state machine structure of Figure 4.22. State 0 corresponds to the state in which no converter is working. In state 1 the DC bus is being precharged and in state 2 the DC/DC converter is working following the corresponding voltage reference. Finally, in state 3, the inverter is also switched on.

6.3.0.1.- Grid Forming

For this model, the corresponding DC and AC bus voltage evolution are shown in Figures 6.16 and 6.17 respectively.

In Figure 6.16, the precharge state is clearly defined, starting the inverter control at t=0.2s. The connection (t=0.65s) and disconnection (t=1.7s) of the inverter and different perturbances (t=1s, t=1.3s, and t=1.55s) on the DC bus can be appreciated as expected. From the point of view of the DC/DC converter, the connection of the inverter has the same effect as the connection of a load.

On the other hand, in Figure 6.17 it can be seen that, when the inverter operation is enabled, the three-phase voltage sought is established.

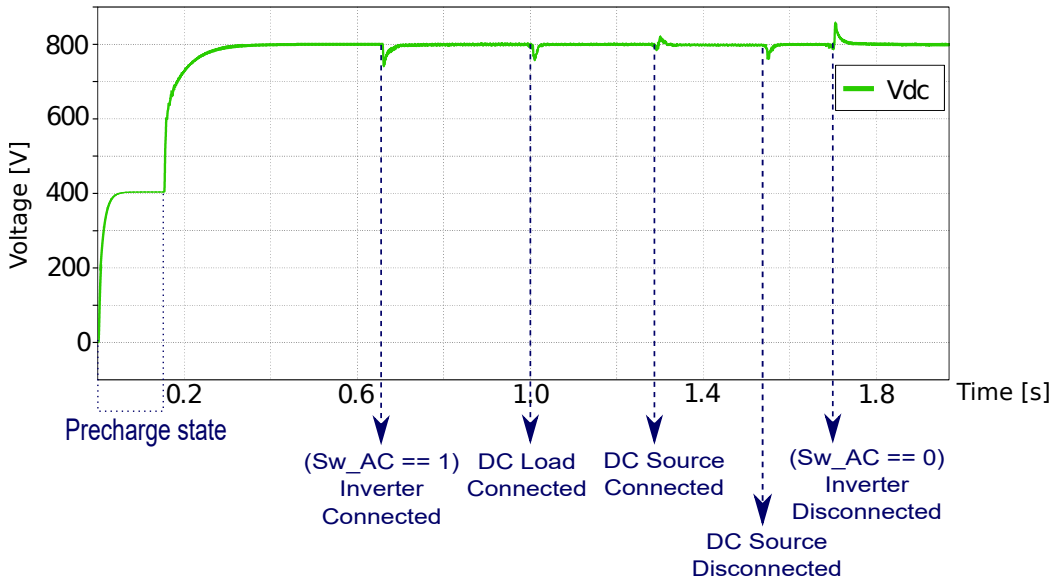

Figure 6.16.- DC bus voltage in the MG model with grid forming inverter.

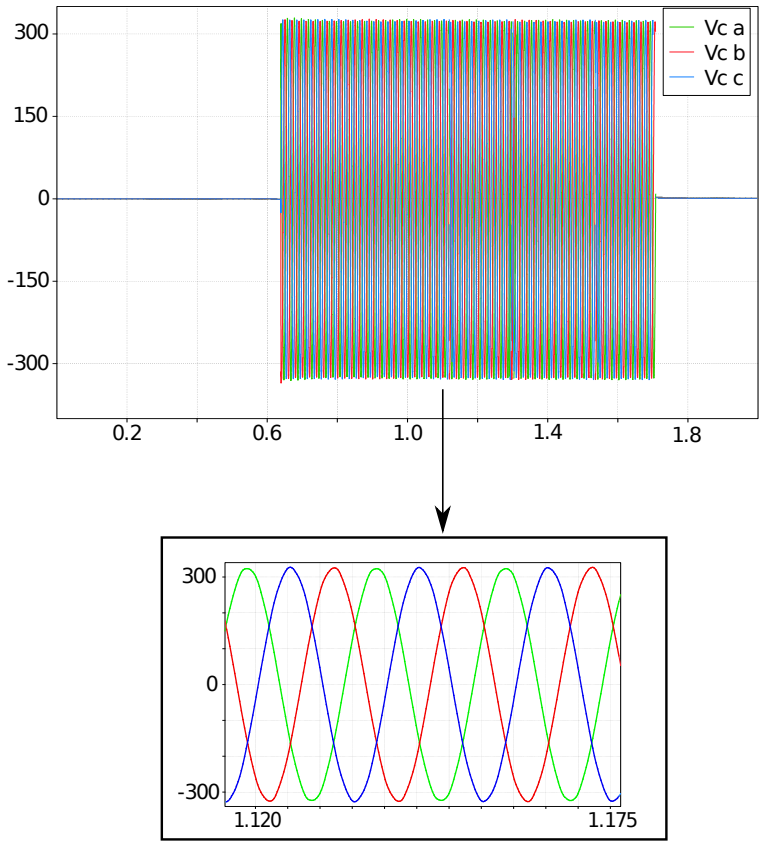

Figure 6.17.- AC bus voltage in the MG model with grid forming inverter.

6.3.0.2.- Grid Feeding

Similar results as the ones for the MG with the grid-forming inverter are obtained. However, now the AC signal of interest is the output current, as the value to be controlled is that of the power injected.

In this MG model, some DC elements are also included. More precisely, a DC load that consumes 10A is now connected to the DC bus, as well as a new source that injects also 10A. To test the operation of the overall system, the following simulation, whose output can be observed in Figure 6.18 is carried out:

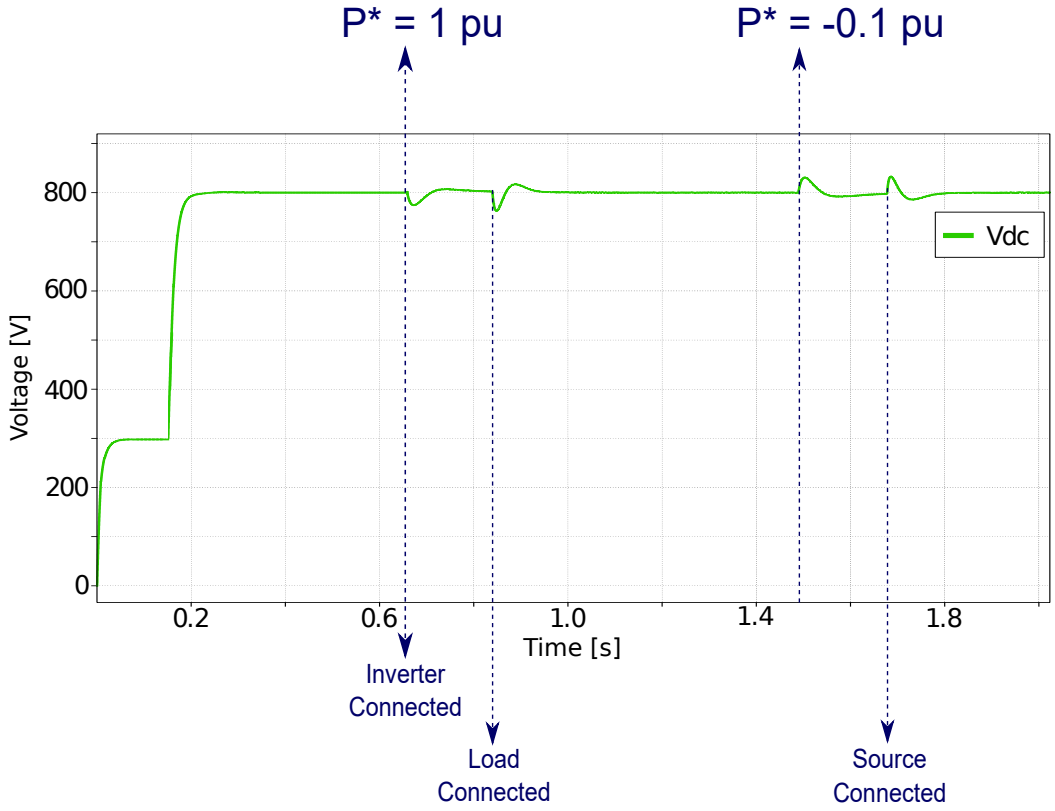

Figure 6.18.- DC signals response to the connection of loads and sources to the DC bus.

1. Initially, only the Synchronous Boost is working. Once the DC bus voltage has been stabilized, the inverter is connected.

2. The inverter is commanded to inject active power into the grid ($P* = 0.5pu$), and no DC

element is connected. The evolution of the power injected by the inverter can be seen in Figure 6.19.

3. After 0.85s, a DC load is also connected.

4. At 1.5s, the inverter is commanded to absorb active power from the grid ($P* = -4pu$). This means that the converter works as a rectifier and feeds the DC load. This implies that less power is consumed from the battery.

5. At 1.7s, the DC source is connected. As the load consumes the same power that the source is injected and the converter is still working as a rectifier, there is a power surplus in the DC bus that is absorbed by the ESS with the help of the DC/DC converter.

6. Finally, at 1.6s, the DC/AC converter is commanded to neither absorb nor inject power. The DC bus is therefore stable and no power is absorbed nor injected into the battery or the grid.

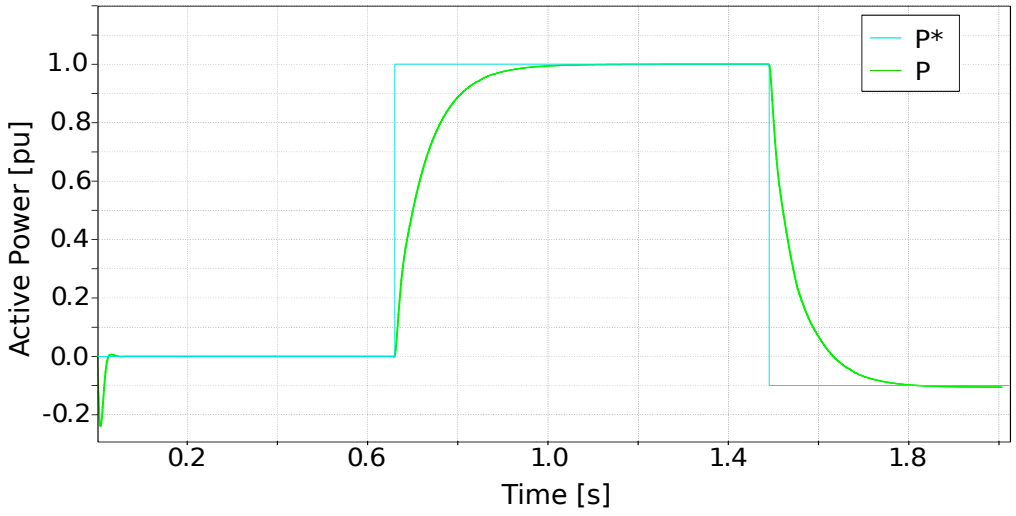

Figure 6.19.- Power injected by the inverter.

6.4.- HIL SIMULATIONS

It is difficult to show in images everything that can be done in real-time simulations (connecting and disconnecting loads, connecting the inverter to the grid, and changing its control mode...) as the system responses are quite fast and it is not possible to capture the changes.

As a general conclusion of all the HIL simulations performed, it can be said that the results are in line with what was expected after completing the standard simulations. The external mode also worked as expected, allowing different commands to be sent to the microcontroller in real-time. Real-time commands from the CCS environment have also been tested.

In the following sections, some of the results obtained are highlighted.

6.4.1.- Synchronous Boost Converter

The results obtained, which are displayed in Figure 6.20, coincide more or less with what was expected. The DC bus does not stabilize at the 800V set as a reference, there is an error of approximately 3.38%.

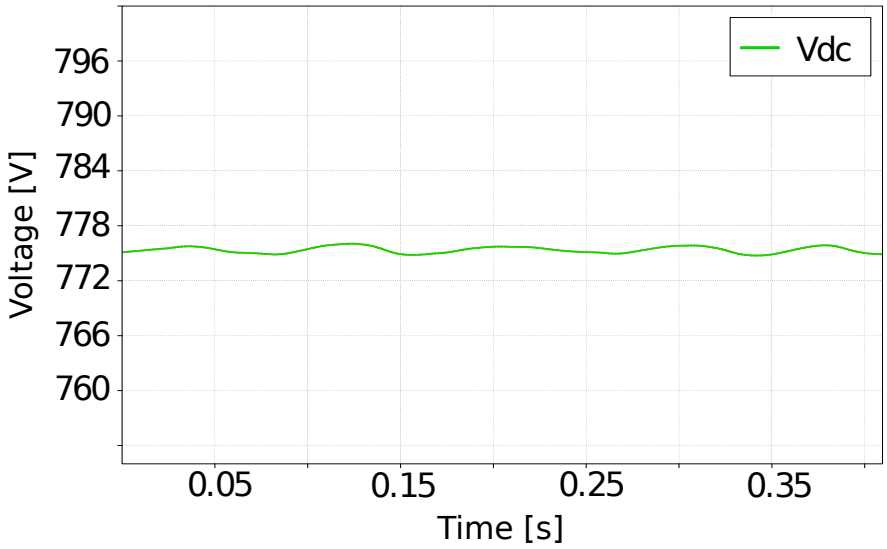

Figure 6.20.- DC bus voltage measured in real-time inside the RT Box.

It has been concluded that this error is not due to failures in the control loop but to a calibration issue in the analog channels. If the values that the MCU is reading are analyzed, one can see that from, the MCU's point of view, the voltage does oscillate around 800V. This can be seen more clearly in Figure 6.21.

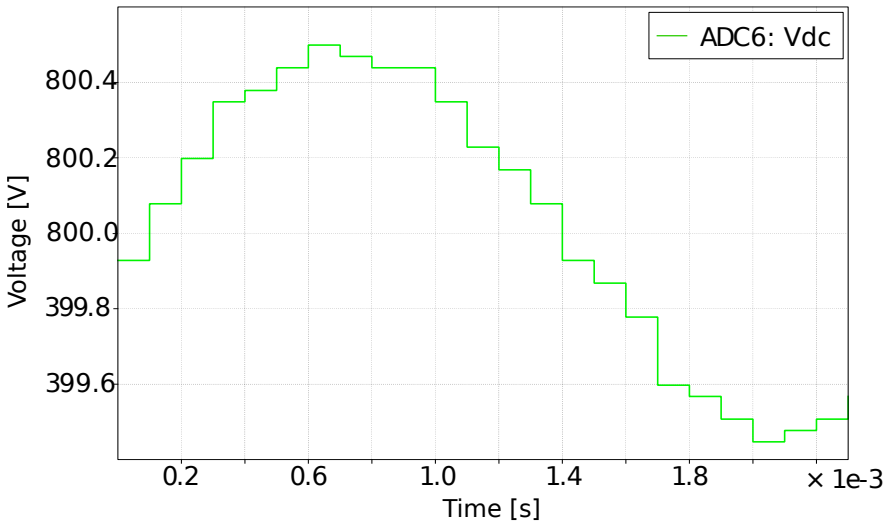

Figure 6.21.- Real-time analog input readout corresponding to the DC bus voltage.

6.4.2.- Three-phase Inverter

6.4.2.1.- DC Bus Control

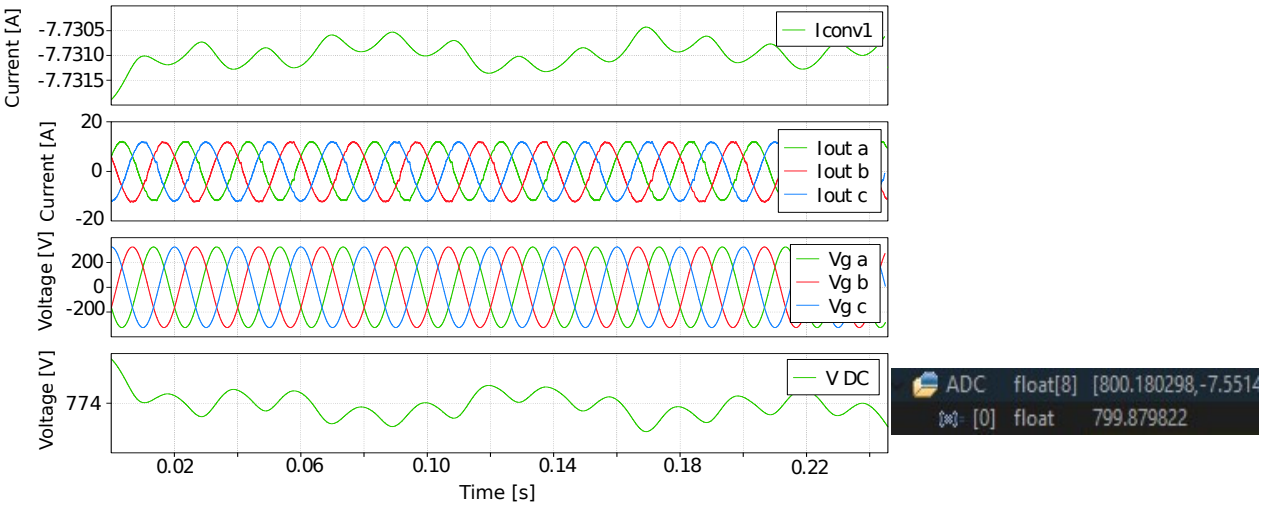

Figure 6.22.- AC & DC real-time measurements during the DC bus control.

The results obtained for this model are again in line with the expectations. The different signals can be seen in Figure 6.22. Once again, the DC bus does not reach its reference value due to calibrating issues.

The verification that the reason is indeed a faulty calibration has now been carried out from the CCS environment. The value read by the MCU is presented also in Figure 6.22, at the bottom-righ side.

6.4.2.2.- Droop Control & VSG

The frequency control has also been tested. The results shown in Figure 6.23 show the frequency obtained as a result of the droop control when a load consuming the nominal power of the inverter, is connected.

The inverter operating as a VSG has also been implemented in a HIL simulation, obtaining similar results as those of the Droop Control. This is so because the main difference between both controls is the speed of response, which is much smoother in the VSG. But this transient is not so easily observed in the real-time simulation as it was in section 6.2.5. These results can be seen in Figure 6.24.

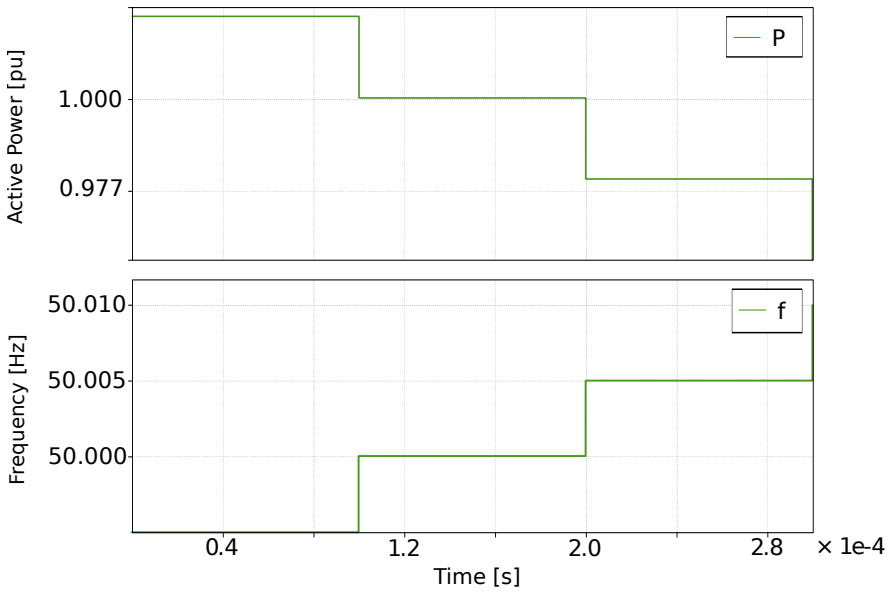

Figure 6.23.- Real-time droop control measurements.

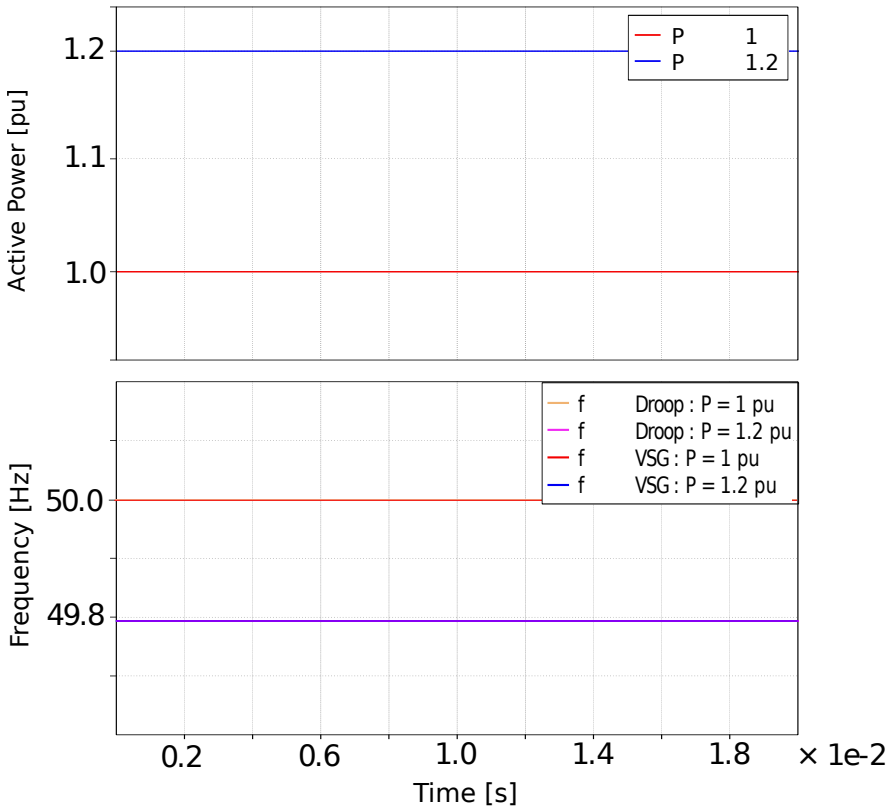

Figure 6.24.- Real-Time VSG measurements

6.4.2.3.- Grid Feeding & Grid Forming Operating Modes

For the grid-feeding inverter, the main test carried out in real-time consisted of changing the reference power commands. The effect it had on the real-time output current measured in the RT Box can be seen in Figure 6.25.

The change between operating modes has also been tested with HIL. In Figure 6.26, the output current and voltages for both modes in the same model can be seen. It can be appreciated that the output current has a greater amplitude when following the power reference, as the reference value is higher than the active power demanded by the load.

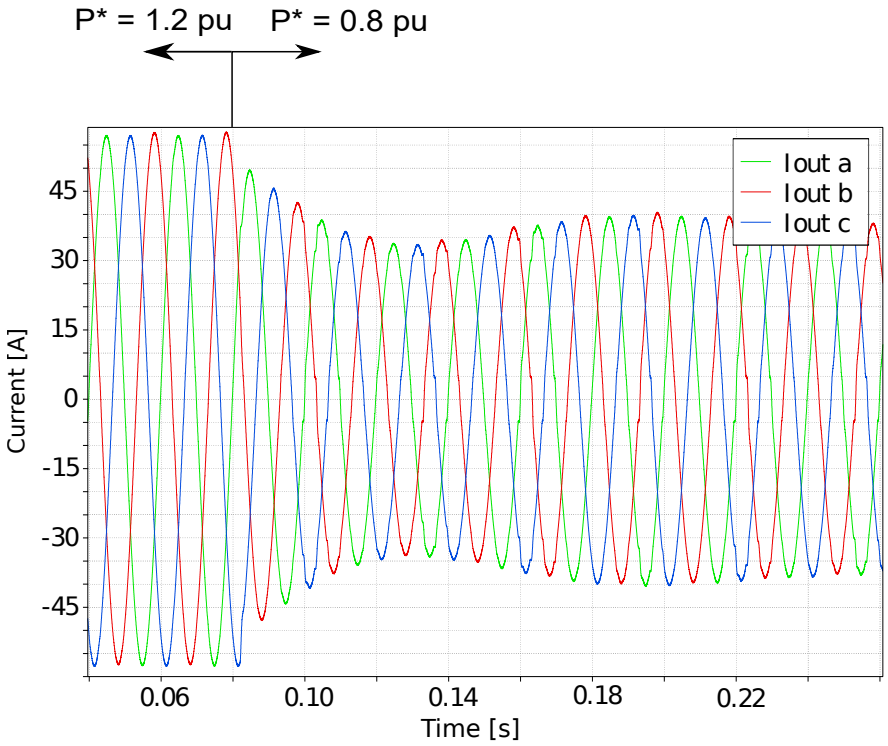

Figure 6.25.- Output current depending on the given reference power.

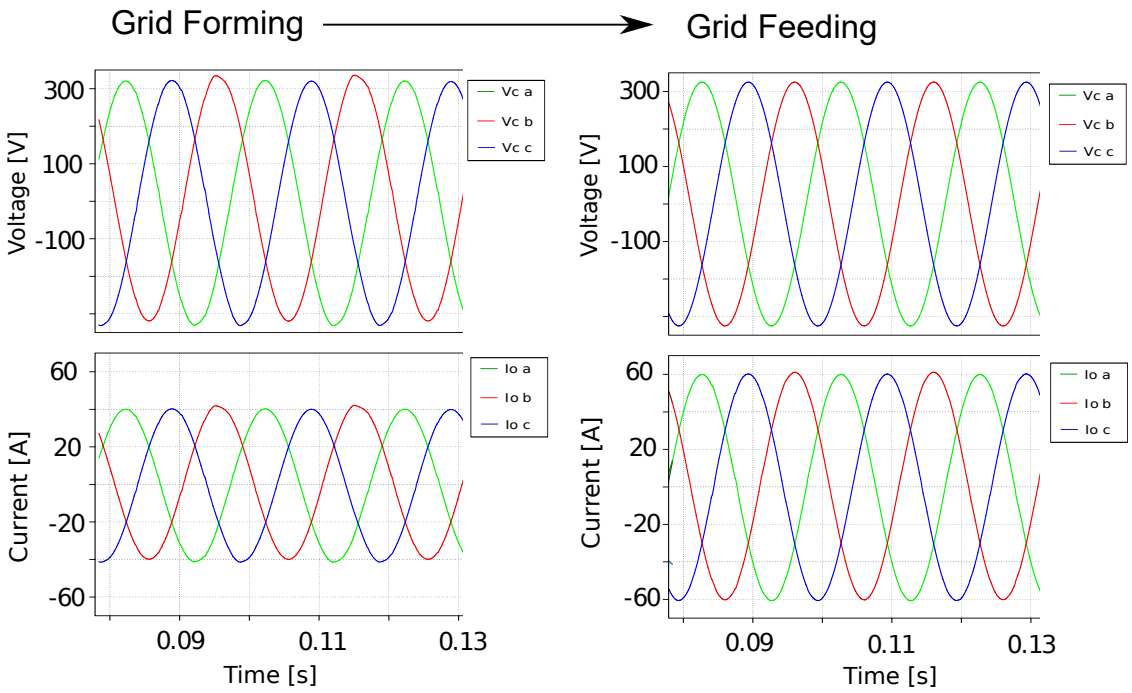

Figure 6.26.- Change in the operation mode: grid forming (AC load consuming P=0.4pu) & grid feeding (P*=0.6pu).

The bus' voltage while working as a grid feeding is a pure sinusoidal waveform, as it is directly established from a three-phase generator. Meanwhile, although during the grid forming operation, this perfect sinusoidal signal is not achieved, it is accurate enough to achieve a valid synchronization with the grid.

The change between operating modes has been tested in both directions. This means going from grid forming to grid feeding and afterwards back to grid forming. Concluding that the control loop has been implemented correctly.

6.4.3.- Microgrid Model

For the MG model, with both the DC/DC and the DC/AC converters, the same situations as in Section 6.3, obtaining similar results.

1. With the grid-forming inverter, both the DC bus voltage and the three-phase voltage are maintained as expected after simulating both converters individually.
2. With the grid-feeding inverter, both positive and negative power references can be commanded.
3. The DC/DC converter can manage the surplus power in the DC bus, both when a DC source is connected and when a negative power reference is given to the inverter.
4. Loads connected to the DC bus are seen as disturbances, as happened in Section 6.3. However, it is important to mention that the decoupling of disturbances inside the control loops is only possible for one of the converters. As another ADC channel would be needed to include a feed-forward compensation in both of them and the 15 channels are already been used.

In the following graphs, some results obtained from the MG model are presented.

Firstly, for the MG with a grid-forming inverter, the effect of connecting a DC load can be seen in Figure 6.27. For this model, an AC load consuming approximately P = 1pu is connected, so the inverter will be injecting that much power into the AC bus.

Then, for the MG with a grid-feeding inverter, the power measurements with the signals inside the RT Box are obtained at the moment when a new power command is given. The active power reference was set at 1.2pu initially, and it has been changed in real-time to 1pu. The MG response can be seen in Figure 6.28. There, it can be appreciated that the power given by the inverter does not reach the reference, it settles at a lower value. However, as represented in Graph 6.29, the value calculated inside the MCU from RT Box outputs does follow the reference. The error is given once again by the calibration problem, the control system works as expected.

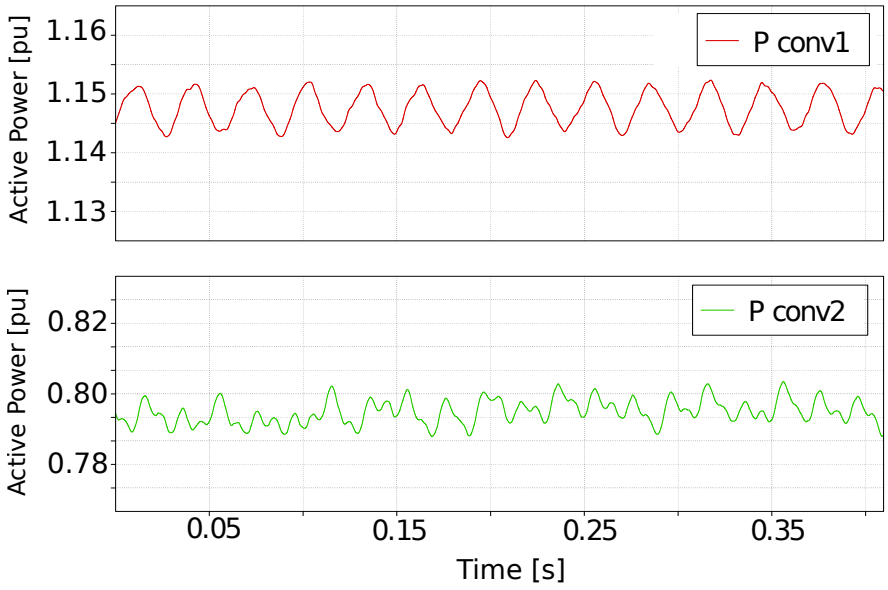

Figure 6.27.- Power measurements with a DC load

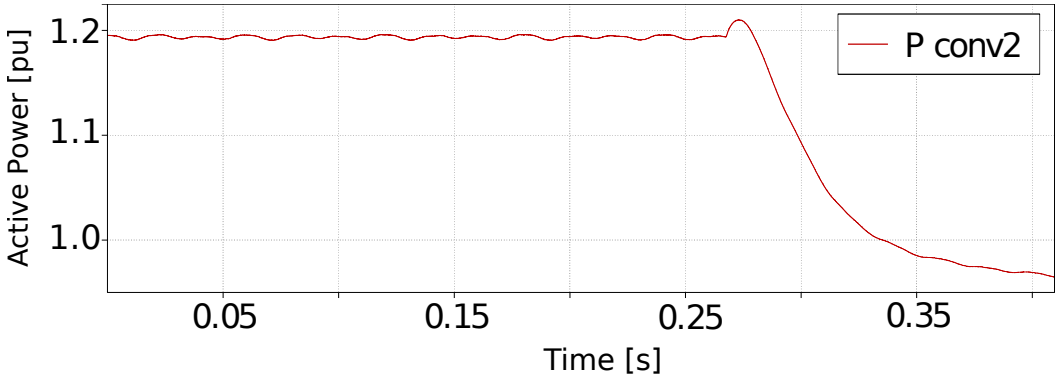

Figure 6.28.- System's response to a change in the power command.

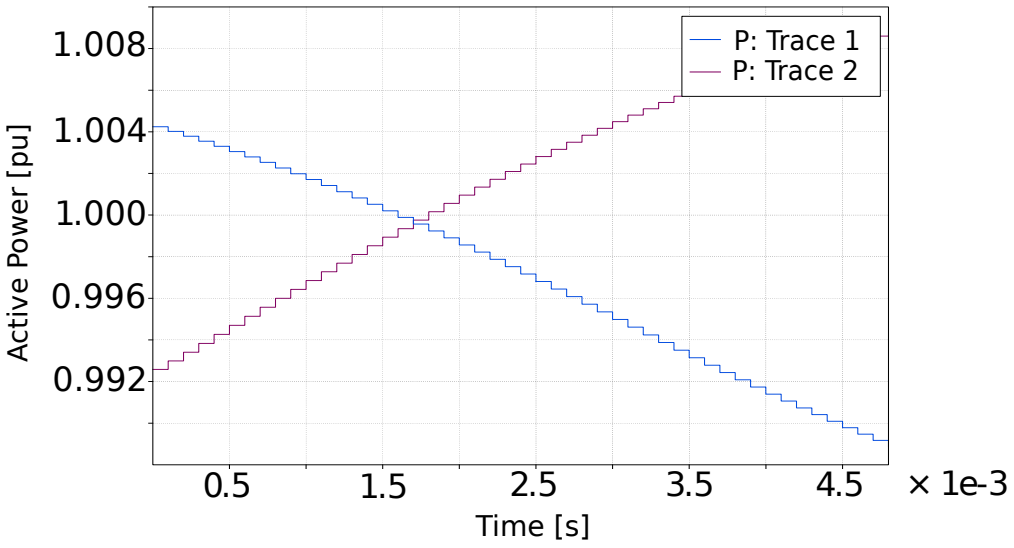

Figure 6.29.- Power measurements inside the MCU captured in two different instants (P* = 1pu).

Moreover, the results shown in Figure 6.30 correspond to a moment in the simulation when a DC load is also connected. The inverter is given a negative power command so that the load is fed from the grid instead of from the battery. However, the load does not consume all the power injected by the inverter. The surplus power goes through the DC/DC converter, feeding the battery.

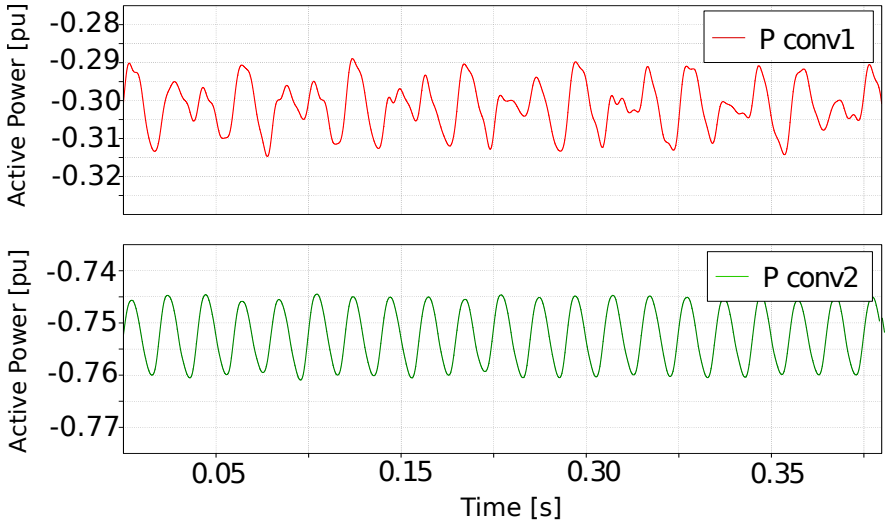

Figure 6.30.- System's response to a negative power command.

7. Conclusions and Future Work

7.1.- CONCLUSIONS

As a general conclusion, it can be said that it has been possible to successfully design and simulate the control of a DC/DC converter to stabilize the voltage of a DC bus from a battery. Both in a continuous and in a discrete control, programmed in C-code. Moreover, the discrete control has not only been implemented with the Discrete PI controllers available in PLECS but also by developing the whole control loop by programming in C language.

Regarding inverter control, all the following control systems have been implemented satisfactorily and with the expected results:

- DC Bus Control.
- Power Control: Grid-Feeding Inverter.
- AC Bus Control: Grid-Forming Inverter.
- Droop-Control: Grid-Supporting Inverter.
- VSG: Grid-Supporting Inverter with inertia.

It has also been possible to design a control loop capable of allowing the inverter to work as grid forming and grid feeding, changing the operating mode depending on the needs of the microgrid.

On the other hand, when comparing the results obtained from the droop-control and the VSG, it can be concluded that the frequency response to an active power step is smoother for the VSG, which corresponds to theoretical expectations.

Furthermore, HIL simulations have been successfully carried out, which was one of the main objectives of the project, by the two possible approaches: model-based programming and

generating code and debugging with CCS. These simulations have made it possible to approximate the conditions of the designed control systems to a real scenario. Therefore allowing to test their validity under more demanding conditions.

7.2.- FUTURE WORK

As future work, the MG model could also be analyzed taking into account the charging and discharging periods of the battery. In this scenario, when the battery needs to be charged, a power control would be applied to the inverter with a negative power command. This means that the MG needs to be connected to the grid. However, when the battery has enough charge for the correct operation of the system, the MG could also work in islanded mode.

In addition to the real battery behaviour, the power semiconductors and the electronic components of the converters could also be analysed in detail, sizing them appropriately and calculating the losses that would occur in the system.

On the other hand, if the project evolves with the design of control loops and HIL testing, the monitoring needed to cope with the ramping events could be studied in detail. Another approach following this line of work could be to implement a larger MG, with different converters connected in parallel. This would require a more powerful HIL platform to connect several MCUs, as it would not be possible to implement all the controls with just one MCU. Also, to make the results of the simulations even more accurate, a calibration of the connections between the RTbox and the MCU should be done.

To sum up, the proposals for future work are:

- Analyse the real battery behavior.
- Specific DC/DC and DC/AC converters design (selection of power electronics components)
- Design of a control system to face ramping events in renewable energy generation.

- Control of a more complete microgrid, comprised of several converters.
- Proper calibration of the RTbox and TMS320F28335 microcontroller I/O.

Bibliography

[1] Renewable energy on the rise: 37news - eurostat.

[2] Arpita De. Optimal sizing and positioning of grid integrated distributed generator: a review. *International Journal of Engineering & Technology*, 9:299, 03 2020.

[3] Wikimedia Commons. File:switches domain.svg — wikimedia commons, the free media repository, 2023. [Online; accessed 12-June-2023].

[4] Yanbin Mao, Feng Liu, and Shengwei Mei. On the topological characteristics of power grids with distributed generation. In *Proceedings of the 29th Chinese Control Conference*, pages 4714–4720, 2010.

[5] Muhammad Hammad Saeed, Wang Fangzong, Basheer Ahmed Kalwar, and Sajid Iqbal. A review on microgrids' challenges and perspectives. *IEEE Access*, 9:166502–166517, 2021.

[6] Luis E. Zubieta and Peter W. Lehn. A high efficiency unidirectional dc/dc converter for integrating distributed resources into dc microgrids. pages 280–284. IEEE, 6 2015.

[7] Daming Zhang and John Fletcher. Operation of autonomous ac microgrid at constant frequency and with reactive power generation from grid-forming, grid-supporting and grid-feeding generators. *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2018-October:1560–1565, 2 2019.

[8] Rohan Sirsi, Shashikant Prasad, Abhishek Sonawane, and Atul Lokhande. Efficiency comparison of ac distribution system and dc distribution system in microgrid. pages 325–329. Institute of Electrical and Electronics Engineers Inc., 10 2016.

[9] Rohan Sirsi and Yadnyesh Ambekar. Efficiency of dc microgrid on dc distribution system. Institute of Electrical and Electronics Engineers Inc., 1 2016.

[10] Diego A. Aponte-Roa, Gerardo David Guerrero Cabarcas, and Wayne W. Weaver. Ac vs dc power efficiency comparison of a hybrid wind/solar microgrid. Institute of Electrical and Electronics Engineers Inc., 4 2020.

[11] Zhuang Fusheng and R. T. Naayagi. Power converters for dc microgrids - modelling and simulation. *International Conference on Innovative Smart Grid Technologies, ISGT Asia 2018*, pages 994–999, 9 2018.

[12] Nisha Kondrath. Bidirectional dc-dc converter topologies and control strategies for interfacing energy storage systems in microgrids: An overview. *2017 5th IEEE International Conference on Smart Energy Grid Engineering, SEGE 2017*, pages 341–345, 9 2017.

[13] Gregor Hoogers. Portable applications. *Fuel Cell Technology Handbook*, pages 75–161, 1 2009.

[14] Khalil Sinjari, Jaeeun Kim, and Joydeep Mitra. Single-phase and three-phase reactive power support for microgrids by grid connected electric vehicle. *2020 52nd North American Power Symposium, NAPS 2020*, 4 2021.

[15] Asheesh Dhaneria. Grid connected pv system with reactive power compensation for the grid. *2020 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference, ISGT 2020*, 2 2020.

[16] Stefan Reichert, Gerd Griepentrog, and Benjamin Stickan. Comparison between grid-feeding and grid-supporting inverters regarding power quality. In *2017 IEEE 8th International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, pages 1–4, April 2017.

[17] Njabulo Mlilo, Tony Ahfock, and Jason Brown. Grid frequency support from inverter connected generation. In *2021 31st Australasian Universities Power Engineering Conference (AUPEC)*, pages 1–5, Sep. 2021.

[18] Hongyang Lai, Kang Xiong, Zhenyuan Zhang, and Zhe Chen. Droop control strategy for microgrid inverters: A deep reinforcement learning enhanced approach. *Energy Reports*, 9:567–575, 2023. 2022 The 3rd International Conference on Power Engineering.

[19] Zakaria Afshar, Mahmoud Mollayousefi Zadeh, S.M.T. Bathaee, Ali Mehrizi-Sani, and Josep M. Guerrero. A novel frequency and voltage controller for parallel voltage source converters and synchronous generators coexisting in islanded microgrids. In *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, pages 1–7, 2020.

[20] Mingjian Cui, Jie Zhang, Cong Feng, Anthony R. Florita, Yuanzhang Sun, and Bri-Mathias Hodge. Characterizing and analyzing ramping events in wind power, solar power, load, and netload. *Renewable Energy*, 111:227–244, 2017.

[21] Dhivya Sampath Kumar, Salish Maharjan, Albert, and Dipti Srinivasan. Ramp-rate limiting strategies to alleviate the impact of pv power ramping on voltage fluctuations using energy storage systems. *Solar Energy*, 234:377–386, 2022.

[22] Eneko Unamuno, Jon Are Suul, Marta Molinas, and Jon Andoni Barrena. Comparative eigenvalue analysis of synchronous machine emulations and synchronous machines. 1:3863–3870, 2019.

[23] Haseeb Ur Rehman, Xiangwu Yan, Mohamed Abdelkarim Abdelbaky, Mishkat Ullah Jan, and Sheeraz Iqbal. An advanced virtual synchronous generator control technique for frequency regulation of grid-connected pv system. *International Journal of Electrical Power & Energy Systems*, 125:106440, 2 2021.

[24] Li Yan, Tian Xinshou, Hu Juan, Zhang Zhankui, Yu Hui, Xiong Xiong, Wu Ming, and Ma Wenyuan. Study on distributed generation to improve stability of distribution network based on virtual synchronous generator technology. In *2019 IEEE 8th International Conference on Advanced Power System Automation and Protection (APAP)*, pages 98–103, 2019.

[25] Xin Meng, Zeng Liu, Jinjun Liu, Shike Wang, Baojin Liu, and Ronghui An. Comparison between inverters based on virtual synchronous generator and droop control. In *2017 IEEE Energy Conversion Congress and Exposition (ECCE)*, pages 4077–4084, 2017.

[26] European Network of Transmission System Operators for Electricity. Rate of change of frequency (rocof) withstand capability entso-e guidance document for national implementation for network codes on grid connection. 2017.

[27] MathWorks España. Conceptos básicos de la simulación hardware-in-the-loop - matlab & simulink.

[28] The simulation platform for power electronic systems rt box user manual november 2022, 2022.

[29] PLECS. The simulation platform for power electronic systems ti c2000 target support user manual version 1.5, 2022.

[30] Abad Lorduy, Antonio Lázaro, Andrés Barrado, Cristina Fernández, Isabel Quesada, and Carlos Lucena. Simplified synchronous reference frame control of the three phase grid connected inverter. *Conference Proceedings - IEEE Applied Power Electronics Conference and Exposition - APEC*, pages 1026–1033, 2010.

[31] P. Rodríguez, J. Pou, J. Bergas, I. Candela, R. Burgos, and D. Boroyevich. Double synchronous reference frame pll for power converters control. *PESC Record - IEEE Annual Power Electronics Specialists Conference*, 2005:1415–1421, 2005.

[32] Shantanu Chatterjee and Saibal Chatterjee. Simulation of synchronous reference frame pll based grid connected inverter for photovoltaic application. *2015 1st Conference on Power, Dielectric and Energy Management at NERIST, ICPDEN 2015*, 4 2015.

[33] Ángel Navarro Rodríguez. Transient frequency drift compensation on weak 3-phase ac microgrids using an energy storage system, 2014.

[34] Weiwei Li, Xinbo Ruan, Chenlei Bao, Donghua Pan, and Xuehua Wang. Grid synchronization systems of three-phase grid-connected power converters: A complex-vector-filter perspective. *IEEE Transactions on Industrial Electronics*, 61:1855–1870, 4 2014.

[35] Rt box controlcard interface user manual march 2021, 2021.

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

GIJÓN POLYTECHNIC SCHOOL OF ENGINEERING

BACHELOR'S DEGREE IN INDUSTRIAL AND AUTOMATIC ELECTRONICS ENGINEERING

SYSTEMS AND AUTOMATION ENGINEERING AREA

AND

ELECTRONIC TECHNOLOGY AREA

CONTROL OF POWER ELECTRONIC CONVERTERS FOR THE INTEGRATION OF DISTRIBUTED GENERATION AND ENERGY STORAGE IN DC AND AC MICROGRIDS

Marta Pérez Molinero

TUTOR: Ángel Navarro Rodríguez

COTUTOR: Ramy Georgious Zaher Georgious

FECHA: June 2023

Appendix

Index

A. Code Generated

A.1.- DC/DC CONVERTER

A.1.1.- C-Block Code

CODE DECLARATIONS

```
//PARAMETERS
#define Fs 10e+3 //switching frequency
#define Pmax 50e+3 //maximum power
#define Vbat_nom 400 //nominal battery voltage
#define R 0.01
#define L 0.001
#define C 700e-6
#define Rload 25
#define precharge_time 0.15
#define pi 3.1416
#define N 2

//INPUT SIGNALS
#define Vout_ref InputSignal(0,0)
#define IL InputSignal(0,1)
#define Vbat InputSignal(0,2)
#define Vout InputSignal(0,3)
#define Iout InputSignal(0,4)

//VARIABLES
double Vrl[N],Ic[N],e_Vdc[N], e_Il[N];
static double a_i[N],a_v[N],b;
static double m;
static double IL_ref,IL_max, Ic_max, Vrl_max;
static double time;
static double kp_i, ki_i, kp_v, ki_v;
static int precharge_control;

//AUXILIARY FUNCTIONS
//variable initialization function
void Init(double v[], int n)
{
	for (int i=0; i<n; i++)
	v[i]=0;
}
//updating the control action and the error vectors
void ActualizaVectores (double u[], double e[])
{
	u[1]=u[0];
	e[1]=e[0];
}
//Tustin discretization parameters
void CoefDiscretizacion(double a[],double kp, double ki)
{
	a[0] = kp*(2*Fs + ki/kp);
	a[1]=kp*(ki/kp - 2*Fs);
}
//PI + Anti-windup
void PI_AntiWindup(double u[],double e[], double a[], double b, double umax)
{
	u[0]=u[1]+(a[0]*e[0]+a[1]*e[1])/b;
	if (u[0]>umax)
	{
		u[0] = umax;
		e[0]=(b*(umax-u[1])-a[1]*e[1])/a[0];
	}
	ActualizaVectores(u,e);
}
```

START FUNCTION

```
//Initialization

// PI constants
kp_i=L*500*2*pi;
ki_i=2*pi*500*R;
kp_v=C*50*2*pi;
ki_v = 2*pi*50/Rload;

//Variable Initialization
Init (Vrl, 2);
Init (Ic, 2);
Init (e_Vdc,2);
Init (e_Il,2);
IL_max = Pmax/Vbat_nom;
time = 0;
precharge_control = 0;

//Tustin Discretization Constants
b=2*Fs;
CoefDiscretizacion(a_i,kp_i,ki_i);
CoefDiscretizacion(a_v,kp_v,ki_v);
```

OUTPUT FUNCTION

```
//OUTPUT SIGNALS
OutputSignal(0,0) = m;
OutputSignal(0,1) = precharge_control;
```

UPDATE FUNCTION

```
time = time + 1/Fs;

//Precharge Control
if (time < precharge_time)
{
        m = 1;
}

else
//DC Bus Control
{
        precharge_control = 1;

        //PI control Vdc - Outer Loop
        e_Vdc[0]=Vout_ref-Vout;
        Ic_max=Pmax/Vout;
        PI_AntiWindup(Ic,e_Vdc,a_v,b,Ic_max);

        IL_ref = (Ic[0]+Iout)*Vout/Vbat; //Feed-Forward

        //PI control IL - Inner Loop
        Vrl_max = R*IL_max+L*Fs*(IL_max-IL);
        if (IL_ref > IL_max)
        IL_ref = IL_max;
        e_Il[0] = IL_ref-IL;
        PI_AntiWindup(Vrl,e_Il,a_i,b,Vrl_max);

        m=(Vbat-Vrl[0])/Vout;
}
```

A.1.2.- CCS Code

```
/*
 * Implementation file for: DCDC/Control System
 * Generated with: PLECS 4.7.2
 *                          TI2833x 1.5.2
 * Generated on: 8 Jun 2023 14:26:50
 */
#include "Control_System.h"
#ifndef PLECS_HEADER_Control_System_h_
#error The wrong header file "Control_System.h" was included. Please check
#error your include path to see whether this file name conflicts with the
#error name of another header file.
#endif /* PLECS_HEADER_Control_System_h_ */
#if defined(__GNUC__) && (__GNUC__ > 4)
#   define _ALIGNMENT 16
#   define _RESTRICT __restrict
#   define _ALIGN __attribute__((aligned(_ALIGNMENT)))
#   if defined(__clang__)
#      if __has_builtin(__builtin_assume_aligned)
#         define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#      else
#         define _ASSUME_ALIGNED(a) a
#      endif
#   else
#      define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#   endif
#else
#   ifndef _RESTRICT
#      define _RESTRICT
#   endif
#   ifndef _ALIGN
#      define _ALIGN
#   endif
#   ifndef _ASSUME_ALIGNED
#      define _ASSUME_ALIGNED(a) a
#   endif
#endif
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>
#include "plx_hal.h"
#define PLECSRunTimeError(msg) Control_System_errorStatus = msg
struct FSM_Struct
{
   int fsm_isMajorTimeStep;
   float fsm_currentTime;
   const float *fsm_internalConstants;
   const float ***fsm_inputs;
   float ***fsm_outputs;
   float *fsm_discStates;
   float *fsm_zCSignals;
   int *fsm_takenTransitions;
   float *fsm_nextSampleHit;
   float fsm_samplingFrequency;
   const char **fsm_errorStatus;
   const char **fsm_warningStatus;
};
static struct FSM_Struct Control_System_fsm_struct[1];
static const float Control_System_UNCONNECTED = 0;
static bool Control_System_D_bool[1];
static float Control_System_deriv[6] _ALIGN;
void Control_System_0_fsm_start(const struct FSM_Struct *fsm_struct);
void Control_System_0_fsm_output(const struct FSM_Struct *fsm_struct);
static uint32_t Control_System_tickLo;
static int32_t Control_System_tickHi;
Control_System_BlockOutputs Control_System_B;
#if defined(EXTERNAL_MODE) && EXTERNAL_MODE
const float * const Control_System_ExtModeSignals[] = {
```

```
   &Control_System_B.StateMachine[1],
   &Control_System_B.ADC[6]
};
#endif /* defined(EXTERNAL_MODE) */
Control_System_Parameters Control_System_P = {
   /* Parameter 'SwitchState' of
    *  Manual Signal Switch : 'Control System/ON DC'
    */
   0.f,
   /* Parameter 'SwitchState' of
    *  Manual Signal Switch : 'Control System/ON load'
    */
   0.f
};
Control_System_ModelStates Control_System_X _ALIGN;
const char * Control_System_errorStatus;
const float Control_System_sampleTime = 0.0001f;
const char * const Control_System_checksum =
   "e22f47c70bbbd9dc8995a8f2fab0ac91fd7582ff";
/* Target declarations */
extern void Control_System_initHal();

void Control_System_initialize(float time)
{
   float remainder;
   Control_System_errorStatus = NULL;
   Control_System_tickHi =
      floor(time/(4294967296.0*Control_System_sampleTime));
   remainder = time - Control_System_tickHi*4294967296.0*
               Control_System_sampleTime;
   Control_System_tickLo = floor(remainder/Control_System_sampleTime + .5);
   remainder -= Control_System_tickLo*Control_System_sampleTime;
   if (fabsf(remainder) > 1e-6*fabsf(time))
   {
      Control_System_errorStatus =
         "Start time must be an integer multiple of the base sample time.";
   }

   /* Target pre-initialization */
   Control_System_initHal();


   /* Initialization for State Machine: 'Control System/State Machine' */
   {
      static const float* fsm_inputPtrs[] = {
         &Control_System_B.ONDC, &Control_System_B.ADC[6],
         &Control_System_B.ONLoad
      };
      static const float** fsm_inputs[] = {
         &fsm_inputPtrs[0], &fsm_inputPtrs[1], &fsm_inputPtrs[2]
      };
      static float* fsm_outputPtrs[] = {
         &Control_System_B.StateMachine[0], &Control_System_B.StateMachine[1],
         &Control_System_B.StateMachine[2], &Control_System_B.StateMachine[3],
         &Control_System_B.StateMachine[4], &Control_System_B.StateMachine[5]
      };
      static float** fsm_outputs[] = {
         &fsm_outputPtrs[0], &fsm_outputPtrs[1], &fsm_outputPtrs[2],
         &fsm_outputPtrs[3], &fsm_outputPtrs[4], &fsm_outputPtrs[5]
      };
      static int fsm_takenTransitions[2];
      static float fsm_nextSampleHit;
      static const float fsm_internalConstants[] = {
         10000.f,400.f
      };
      static const char* fsm_errorStatus = NULL;
      static const char* fsm_warningStatus = NULL;
      Control_System_fsm_struct[0].fsm_isMajorTimeStep = 1;
```

```
      Control_System_fsm_struct[0].fsm_internalConstants =
         fsm_internalConstants;
      Control_System_fsm_struct[0].fsm_inputs = fsm_inputs;
      Control_System_fsm_struct[0].fsm_outputs = fsm_outputs;
      Control_System_fsm_struct[0].fsm_discStates =
         &Control_System_X.StateMachine[0];
      Control_System_fsm_struct[0].fsm_discStates[1] = 0.f;
      Control_System_fsm_struct[0].fsm_discStates[2] = 48.f;
      Control_System_fsm_struct[0].fsm_discStates[3] = 0.f;
      Control_System_fsm_struct[0].fsm_zCSignals = NULL;
      Control_System_fsm_struct[0].fsm_takenTransitions =
         fsm_takenTransitions;
      Control_System_fsm_struct[0].fsm_nextSampleHit = &fsm_nextSampleHit;
      Control_System_fsm_struct[0].fsm_samplingFrequency = 10000.f;
      Control_System_fsm_struct[0].fsm_errorStatus = &fsm_errorStatus;
      Control_System_fsm_struct[0].fsm_warningStatus = &fsm_warningStatus;
      Control_System_0_fsm_start(&Control_System_fsm_struct[0]);
      if (*Control_System_fsm_struct[0].fsm_errorStatus)
         Control_System_errorStatus =
            *Control_System_fsm_struct[0].fsm_errorStatus;
   }

   /* Initialization for Subsystem: 'Control System/DC bus Control' */
   Control_System_X.Delay = 0.f;
   Control_System_X.DiscreteIntegrator_first = -1;
   Control_System_X.DiscreteIntegrator_i1_x = 0;
   Control_System_X.Delay_1 = 0.f;
   Control_System_X.DiscreteIntegrator_1_first = -1;
   Control_System_X.DiscreteIntegrator_1_i1_x = 0;
   Control_System_B.DCBusControl = 0.f;
}

void Control_System_step(void)
{
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Manual Signal Switch: 'Control System/ON DC'
    * incorporates
    *  Constant: 'Control System/Constant'
    *  Constant: 'Control System/Constant1'
    */
   Control_System_B.ONDC = Control_System_P.ONDC_SwitchState ? 1.f : 0.f;

   /* ADC: 'Control System/ADC' */
   Control_System_B.ADC[0] = PLXHAL_ADC_getIn(0, 0);
   Control_System_B.ADC[1] = PLXHAL_ADC_getIn(0, 1);
   Control_System_B.ADC[2] = PLXHAL_ADC_getIn(0, 2);
   Control_System_B.ADC[3] = PLXHAL_ADC_getIn(0, 3);
   Control_System_B.ADC[4] = PLXHAL_ADC_getIn(0, 4);
   Control_System_B.ADC[5] = PLXHAL_ADC_getIn(0, 5);
   Control_System_B.ADC[6] = PLXHAL_ADC_getIn(0, 6);
   Control_System_B.ADC[7] = PLXHAL_ADC_getIn(0, 7);
   Control_System_B.ADC[8] = PLXHAL_ADC_getIn(0, 8);
   Control_System_B.ADC[9] = PLXHAL_ADC_getIn(0, 9);
   Control_System_B.ADC[10] = PLXHAL_ADC_getIn(0, 10);
   Control_System_B.ADC[11] = PLXHAL_ADC_getIn(0, 11);
   Control_System_B.ADC[12] = PLXHAL_ADC_getIn(0, 12);
   Control_System_B.ADC[13] = PLXHAL_ADC_getIn(0, 13);
   Control_System_B.ADC[14] = PLXHAL_ADC_getIn(0, 14);
   Control_System_B.ADC[15] = PLXHAL_ADC_getIn(0, 15);
```

```
/* Manual Signal Switch : 'Control System/ON load'
    * incorporates
    *  Constant: 'Control System/Constant2'
    *  Constant: 'Control System/Constant3'
    */
   Control_System_B.ONLoad = Control_System_P.ONLoad_SwitchState ? 1.f : 0.f;

   /* State Machine : 'Control System/State Machine' */
   Control_System_0_fsm_output(&Control_System_fsm_struct[0]);
   if (*Control_System_fsm_struct[0].fsm_errorStatus)
      Control_System_errorStatus =
         *Control_System_fsm_struct[0].fsm_errorStatus;

   /* Subsystem : 'Control System/DC bus Control' */
   Control_System_D_bool[0] = 0;
   Control_System_D_bool[0] = Control_System_D_bool[0] ||
                              Control_System_B.StateMachine[0];
   if (!Control_System_D_bool[0] && Control_System_X.DCBusControl[0])
   {
      Control_System_X.DiscreteIntegrator_first = 1;
      Control_System_X.DiscreteIntegrator_1_first = 1;
   }
   if (Control_System_D_bool[0])
   {

      /* Signal Inport : 'Control System/DC bus Control/Vout' */
      Control_System_B.Vout = Control_System_B.ADC[6];

      /* Signal Inport : 'Control System/DC bus Control/Vbat' */
      Control_System_B.Vbat = Control_System_B.ADC[14];

      /* Signal Inport : 'Control System/DC bus Control/Iload' */
      Control_System_B.Iload = Control_System_B.ADC[7];

      /* Signal Inport : 'Control System/DC bus Control/Vdc*' */
      Control_System_B.Vdc_ = Control_System_B.StateMachine[3];

      /* Sum : 'Control System/DC bus Control/Sum' */
      Control_System_B.Sum = Control_System_B.Vdc_ - Control_System_B.Vout;

      /* Zero-Order Hold : 'Control System/DC bus Control/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold' */
      Control_System_B.Zero_OrderHold = Control_System_B.Sum;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController1/Source
Select/internal/Constant' */
      Control_System_B.Constant = 0.219911486f;

      /* Product : 'Control System/DC bus Control/Discrete PID\nController1/Discrete
Time/Product' */
      Control_System_B.Product_1 = Control_System_B.Zero_OrderHold *
                                   Control_System_B.Constant;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController1/Source
Select/internal/Constant1' */
      Control_System_B.Constant1 = 6.28318531f;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController1/Anti-
windup\nmethod/Back-Calculation/Constant' */
      Control_System_B.Constant_1 = 1.f;

      /* Product : 'Control System/DC bus Control/Discrete PID\nController1/Discrete
Time/Product4' */
      Control_System_B.Product4 = Control_System_B.Zero_OrderHold *
                                  Control_System_B.Constant1 *
                                  Control_System_B.Constant_1;
```

```
      /* Delay : 'Control System/DC bus Control/Discrete PID\nController1/Anti-
windup\nmethod/Back-Calculation/Unit Delay/Trapezoidal/Delay' */
      Control_System_B.Delay = Control_System_X.Delay;

      /* Sum : 'Control System/DC bus Control/Discrete PID\nController1/Discrete
Time/Sum2' */
      Control_System_B.Sum2 = Control_System_B.Product4 +
                              Control_System_B.Delay;

      /* Enable : 'Control System/DC bus Control/Enable' */
      Control_System_B.Enable = Control_System_B.StateMachine[0];

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController1/Init
select/internal/Constant' */
      Control_System_B.Constant_2 = 0.f;

      /* Discrete Integrator : 'Control System/DC bus Control/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
      if (Control_System_X.DiscreteIntegrator_first < 0 ||
          (!Control_System_X.DiscreteIntegrator_i3_prevReset &&
           Control_System_B.Enable))
      {
         Control_System_B.DiscreteIntegrator = Control_System_B.Constant_2;
      }
      else if (Control_System_X.DiscreteIntegrator_first)
      {
         Control_System_B.DiscreteIntegrator =
            Control_System_X.DiscreteIntegrator_i1_x;
      }
      else
      {
         Control_System_B.DiscreteIntegrator =
            Control_System_X.DiscreteIntegrator_i1_x + 5e-05f*
            (Control_System_X.DiscreteIntegrator_i2_prevU +
             Control_System_B.Sum2);
      }

      /* Sum: 'Control System/DC bus Control/Discrete PID\nController1/Discrete
Time/Sum'
       * incorporates
       *  Subsystem: 'Control System/DC bus Control'
       */
      Control_System_B.Sum_1 = Control_System_B.Product_1 +
                               Control_System_UNCONNECTED +
                               Control_System_B.DiscreteIntegrator;

      /* Saturation: 'Control System/DC bus Control/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation' */
      Control_System_B.Saturation = Control_System_B.Sum_1;
      if (Control_System_B.Saturation > 62.5f)
      {
         Control_System_B.Saturation = 62.5f;
      }
      else if (Control_System_B.Saturation < -62.5f)
      {
         Control_System_B.Saturation = -62.5f;
      }

      /* Sum: 'Control System/DC bus Control/Sum3' */
      Control_System_B.Sum3 = Control_System_B.Iload +
                              Control_System_B.Saturation;

      /* Product: 'Control System/DC bus Control/Product1' */
      Control_System_B.Product1 = Control_System_B.Vout *
                                  Control_System_B.Sum3;

      /* Product: 'Control System/DC bus Control/Product2' */
```

```
      Control_System_B.Product2 = Control_System_B.Product1 /
                                  Control_System_B.Vbat;

      /* Signal Inport : 'Control System/DC bus Control/IL' */
      Control_System_B.IL = Control_System_B.ADC[15];

      /* Sum : 'Control System/DC bus Control/Sum1' */
      Control_System_B.Sum1 = Control_System_B.Product2 - Control_System_B.IL;

      /* Zero-Order Hold : 'Control System/DC bus Control/Discrete
PID\nController/Discrete Time/Zero-Order\nHold' */
      Control_System_B.Zero_OrderHold_1 = Control_System_B.Sum1;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController/Source
Select/internal/Constant' */
      Control_System_B.Constant_3 = 3.14159265f;

      /* Product : 'Control System/DC bus Control/Discrete PID\nController/Discrete
Time/Product' */
      Control_System_B.Product_3 = Control_System_B.Zero_OrderHold_1 *
                                   Control_System_B.Constant_3;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController/Source
Select/internal/Constant1' */
      Control_System_B.Constant1_1 = 31.4159265f;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController/Anti-
windup\nmethod/Back-Calculation/Constant' */
      Control_System_B.Constant_4 = 1.f;

      /* Product : 'Control System/DC bus Control/Discrete PID\nController/Discrete
Time/Product4' */
      Control_System_B.Product4_1 = Control_System_B.Zero_OrderHold_1 *
                                    Control_System_B.Constant1_1 *
                                    Control_System_B.Constant_4;

      /* Delay : 'Control System/DC bus Control/Discrete PID\nController/Anti-
windup\nmethod/Back-Calculation/Unit Delay/Trapezoidal/Delay' */
      Control_System_B.Delay_1 = Control_System_X.Delay_1;

      /* Sum : 'Control System/DC bus Control/Discrete PID\nController/Discrete
Time/Sum2' */
      Control_System_B.Sum2_1 = Control_System_B.Product4_1 +
                                Control_System_B.Delay_1;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController/Init
select/internal/Constant' */
      Control_System_B.Constant_5 = 0.f;

      /* Discrete Integrator : 'Control System/DC bus Control/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
      if (Control_System_X.DiscreteIntegrator_1_first < 0 ||
          (!Control_System_X.DiscreteIntegrator_1_i3_prevReset &&
           Control_System_B.Enable))
      {
         Control_System_B.DiscreteIntegrator_1 = Control_System_B.Constant_5;
      }
      else if (Control_System_X.DiscreteIntegrator_1_first)
      {
         Control_System_B.DiscreteIntegrator_1 =
            Control_System_X.DiscreteIntegrator_1_i1_x;
      }
      else
      {
         Control_System_B.DiscreteIntegrator_1 =
            Control_System_X.DiscreteIntegrator_1_i1_x + 5e-05f*
            (Control_System_X.DiscreteIntegrator_1_i2_prevU +
```

```
            Control_System_B.Sum2_1);
      }

      /* Sum : 'Control System/DC bus Control/Discrete PID\nController/Discrete
Time/Sum'
       * incorporates
       *  Subsystem : 'Control System/DC bus Control'
       */
      Control_System_B.Sum_2 = Control_System_B.Product_3 +
                               Control_System_UNCONNECTED +
                               Control_System_B.DiscreteIntegrator_1;

      /* Saturation : 'Control System/DC bus Control/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation' */
      Control_System_B.Saturation_1 = Control_System_B.Sum_2;
      if (Control_System_B.Saturation_1 > 400.f)
      {
         Control_System_B.Saturation_1 = 400.f;
      }
      else if (Control_System_B.Saturation_1 < -400.f)
      {
         Control_System_B.Saturation_1 = -400.f;
      }

      /* Sum : 'Control System/DC bus Control/Sum2' */
      Control_System_B.Sum2_2 = Control_System_B.Vbat -
                                Control_System_B.Saturation_1;

      /* Sum : 'Control System/DC bus Control/Discrete PID\nController1/Anti-
windup\nmethod/Back-Calculation/Sum' */
      Control_System_B.Sum_3 = -Control_System_B.Sum_1 +
                               Control_System_B.Saturation;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController1/Source
Select/internal/Anti-windup\nmethod/Back-Calculation/Constant' */
      Control_System_B.Constant_6 = 1.f;

      /* Product : 'Control System/DC bus Control/Discrete PID\nController1/Anti-
windup\nmethod/Back-Calculation/Product' */
      Control_System_B.Product = Control_System_B.Sum_3 *
                                 Control_System_B.Constant_6;

      /* Sum : 'Control System/DC bus Control/Discrete PID\nController/Anti-
windup\nmethod/Back-Calculation/Sum' */
      Control_System_B.Sum_4 = -Control_System_B.Sum_2 +
                               Control_System_B.Saturation_1;

      /* Constant : 'Control System/DC bus Control/Discrete PID\nController/Source
Select/internal/Anti-windup\nmethod/Back-Calculation/Constant' */
      Control_System_B.Constant_7 = 1.f;

      /* Product : 'Control System/DC bus Control/Discrete PID\nController/Anti-
windup\nmethod/Back-Calculation/Product' */
      Control_System_B.Product_2 = Control_System_B.Sum_4 *
                                   Control_System_B.Constant_7;
      Control_System_B.DCBusControl = 1.f / Control_System_B.Vout *
                                      Control_System_B.Sum2_2;
   }

   /* PWM  : 'Control System/PWM' */
   {
      PLXHAL_PWM_setDuty(0,
                         (Control_System_B.StateMachine[0] !=
                          0.f) ? Control_System_B.DCBusControl :
Control_System_B.StateMachine[
                           4]);
   }
```

```
   /* Digital Out : 'Control System/Sw_load' */
   PLXHAL_DIO_set(0, Control_System_B.StateMachine[2]);
   /* Digital Out : 'Control System/Sw_charge' */
   PLXHAL_DIO_set(1, Control_System_B.StateMachine[5]);
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Update for Subsystem : 'Control System/DC bus Control' */
   Control_System_X.DCBusControl[0] = Control_System_D_bool[0];
   if (Control_System_D_bool[0])
   {

      /* Update for Delay : 'Control System/DC bus Control/Discrete
PID\nController1/Anti-windup\nmethod/Back-Calculation/Unit Delay/Trapezoidal/Delay'
*/
      Control_System_X.Delay = Control_System_B.Product;

      /* Update for Discrete Integrator : 'Control System/DC bus Control/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
      Control_System_X.DiscreteIntegrator_first = 0;
      Control_System_X.DiscreteIntegrator_i1_x =
         Control_System_B.DiscreteIntegrator;
      Control_System_X.DiscreteIntegrator_i2_prevU = Control_System_B.Sum2;
      Control_System_X.DiscreteIntegrator_i3_prevReset =
         !!(Control_System_B.Enable);

      /* Update for Delay : 'Control System/DC bus Control/Discrete
PID\nController/Anti-windup\nmethod/Back-Calculation/Unit Delay/Trapezoidal/Delay' */
      Control_System_X.Delay_1 = Control_System_B.Product_2;

      /* Update for Discrete Integrator : 'Control System/DC bus Control/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
      Control_System_X.DiscreteIntegrator_1_first = 0;
      Control_System_X.DiscreteIntegrator_1_i1_x =
         Control_System_B.DiscreteIntegrator_1;
      Control_System_X.DiscreteIntegrator_1_i2_prevU =
         Control_System_B.Sum2_1;
      Control_System_X.DiscreteIntegrator_1_i3_prevReset =
         !!(Control_System_B.Enable);
   }

   /* Update for PWM  : 'Control System/PWM' */
   PLXHAL_PWM_enableAllOutputs();
}

void Control_System_terminate()
{
}
```

A.1.3.- State Machine

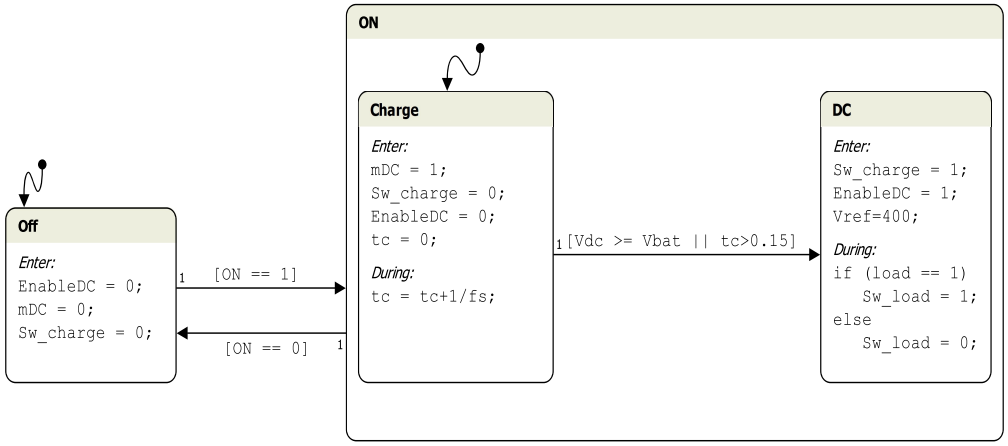

Figure A.1.- Description of the State Machine for the SBC Control.

```
/*
 * State machine file for: Control System/State Machine
 * Generated with: PLECS 4.7.2
 * Generated on: 8 Jun 2023 14:26:50
 */

typedef float real_t;
#define REAL_MAX FLT_MAX
#define REAL_MIN FLT_MIN
#define REAL_EPSILON FLT_EPSILON
struct FSM_Struct
{
   int fsm_isMajorTimeStep;
   float fsm_currentTime;
   const float *fsm_internalConstants;
   const float ***fsm_inputs;
   float ***fsm_outputs;
   float *fsm_discStates;
   float *fsm_zCSignals;
   int *fsm_takenTransitions;
   float *fsm_nextSampleHit;
   float fsm_samplingFrequency;
   const char **fsm_errorStatus;
   const char **fsm_warningStatus;
};

enum FSM_State
{
   FSM_STATE_NONE,
   FSM_STATE_OFF,
   FSM_STATE_ON_CHARGE,
   FSM_STATE_ON_DC
};

enum FSM_Transition
{
   FSM_TRANSITION_NONE,
   FSM_TRANSITION_OFF_1,
   FSM_TRANSITION_ON_1,
   FSM_TRANSITION_ON_CHARGE_1,
   FSM_INITIAL_TRANSITION_ON,
   FSM_INITIAL_TRANSITION
};

#define FSM_MAX_NUM_TAKEN_TRANSITIONS 2

#define FSM_INTERNAL_VARIABLES_OFFSET 1
#define FSM_NUM_INTERNAL_VARIABLES 3

#define CurrentState fsm_struct->fsm_discStates[0]
#define TakenTransition(i) fsm_struct->fsm_takenTransitions[i]
#define IsMajorStep fsm_struct->fsm_isMajorTimeStep
#define CurrentTime fsm_struct->fsm_currentTime
#define SetErrorMessage(string) { *fsm_struct->fsm_errorStatus = (string); }
#define SetWarningMessage(string)

/* input variables */
#define ON (*fsm_struct->fsm_inputs[0][0])
#define Vdc (*fsm_struct->fsm_inputs[1][0])
#define load (*fsm_struct->fsm_inputs[2][0])

/* internal constants */
#define fs fsm_struct->fsm_internalConstants[0]
#define Vbat fsm_struct->fsm_internalConstants[1]

/* internal variables */
#define tac fsm_struct->fsm_discStates[FSM_INTERNAL_VARIABLES_OFFSET + 0]
#define V fsm_struct->fsm_discStates[FSM_INTERNAL_VARIABLES_OFFSET + 1]
```

```
#define tc fsm_struct->fsm_discStates[FSM_INTERNAL_VARIABLES_OFFSET + 2]

/* output variables */
#define EnableDC (*fsm_struct->fsm_outputs[0][0])
#define state (*fsm_struct->fsm_outputs[1][0])
#define Sw_load (*fsm_struct->fsm_outputs[2][0])
#define Vref (*fsm_struct->fsm_outputs[3][0])
#define mDC (*fsm_struct->fsm_outputs[4][0])
#define Sw_charge (*fsm_struct->fsm_outputs[5][0])



static void fsm_state_Off_EnterAction(const struct FSM_Struct* fsm_struct)
{
   EnableDC = 0;
   mDC = 0;
   Sw_charge = 0;
}

static void fsm_state_ON_Charge_EnterAction(
                                              const struct FSM_Struct* fsm_struct)
{
   mDC = 1;
   Sw_charge = 0;
   EnableDC = 0;
   tc = 0;
}

static void fsm_state_ON_DC_EnterAction(const struct FSM_Struct* fsm_struct)
{
   Sw_charge = 1;
   EnableDC = 1;
   Vref=800;
}


static void fsm_state_ON_Charge_DuringAction(
                                               const struct FSM_Struct* fsm_struct)
{
   tc = tc+1/fs;
}

static void fsm_state_ON_DC_DuringAction(const struct FSM_Struct* fsm_struct)
{
   if (load == 1)
      Sw_load = 1;
   else
      Sw_load = 0;
}



void Control_System_0_fsm_start(const struct FSM_Struct *fsm_struct)
{
   int fsm_i;
   CurrentState = FSM_STATE_NONE;
   for (fsm_i = 0; fsm_i < FSM_MAX_NUM_TAKEN_TRANSITIONS; fsm_i++)
   {
      TakenTransition(fsm_i) = FSM_TRANSITION_NONE;
   }
}

void Control_System_0_fsm_output(const struct FSM_Struct *fsm_struct)
{
   if (IsMajorStep)
   {
      int fsm_i;
      for (fsm_i = 0; fsm_i < FSM_MAX_NUM_TAKEN_TRANSITIONS; fsm_i++)
```

```
      {
         TakenTransition(fsm_i) = FSM_TRANSITION_NONE;
      }
      switch ((int)CurrentState)
      {
      case FSM_STATE_OFF:
         if (ON == 1)
         {
            TakenTransition(0) = FSM_TRANSITION_OFF_1;
            TakenTransition(1) = FSM_INITIAL_TRANSITION_ON;
            fsm_state_ON_Charge_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_ON_CHARGE;
         }
         break;

      case FSM_STATE_ON_CHARGE:
         if (ON == 0)
         {
            TakenTransition(0) = FSM_TRANSITION_ON_1;
            fsm_state_Off_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_OFF;
         }
         else if (Vdc >= Vbat || tc > 0.15)
         {
            TakenTransition(0) = FSM_TRANSITION_ON_CHARGE_1;
            fsm_state_ON_DC_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_ON_DC;
         }
         else
         {
            fsm_state_ON_Charge_DuringAction(fsm_struct);
         }
         break;

      case FSM_STATE_ON_DC:
         if (ON == 0)
         {
            TakenTransition(0) = FSM_TRANSITION_ON_1;
            fsm_state_Off_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_OFF;
         }
         else
         {
            fsm_state_ON_DC_DuringAction(fsm_struct);
         }
         break;

      default:
         TakenTransition(0) = FSM_INITIAL_TRANSITION;
         fsm_state_Off_EnterAction(fsm_struct);
         CurrentState = FSM_STATE_OFF;
         break;
      }
   }
}
```

A.2.- DC/AC CONVERTER

A.2.1.- DC Voltage Control

```
/*
 * Implementation file for: DC Voltage Control/Control System
 * Generated with: PLECS 4.7.2
 *                          TI2833x 1.5.2
 * Generated on: 8 Jun 2023 15:59:10
 */
#include "Control_System.h"
#ifndef PLECS_HEADER_Control_System_h_
#error The wrong header file "Control_System.h" was included. Please check
#error your include path to see whether this file name conflicts with the
#error name of another header file.
#endif /* PLECS_HEADER_Control_System_h_ */
#if defined(__GNUC__) && (__GNUC__ > 4)
#   define _ALIGNMENT 16
#   define _RESTRICT __restrict
#   define _ALIGN __attribute__((aligned(_ALIGNMENT)))
#   if defined(__clang__)
#      if __has_builtin(__builtin_assume_aligned)
#         define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#      else
#         define _ASSUME_ALIGNED(a) a
#      endif
#   else
#      define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#   endif
#else
#   ifndef _RESTRICT
#      define _RESTRICT
#   endif
#   ifndef _ALIGN
#      define _ALIGN
#   endif
#   ifndef _ASSUME_ALIGNED
#      define _ASSUME_ALIGNED(a) a
#   endif
#endif
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>
#include "plx_hal.h"
#define PLECSRunTimeError(msg) Control_System_errorStatus = msg
static const float Control_System_UNCONNECTED = 0;
static float Control_System_deriv[6] _ALIGN;
static uint32_t Control_System_tickLo;
static int32_t Control_System_tickHi;
Control_System_BlockOutputs Control_System_B;
Control_System_Parameters Control_System_P = {
   /* Parameter 'Value' of
    *  Constant : 'Control System/VDC*'
    */
   800.f
};
Control_System_ModelStates Control_System_X _ALIGN;
const char * Control_System_errorStatus;
const float Control_System_sampleTime = 0.0001f;
const char * const Control_System_checksum =
   "d65c64705d90210444a048190edf831a70d7cff8";
/* Target declarations */
extern void Control_System_initHal();

void Control_System_initialize(float time)
{
   float remainder;
   Control_System_errorStatus = NULL;
   Control_System_tickHi =
      floor(time/(4294967296.0*Control_System_sampleTime));
   remainder = time - Control_System_tickHi*4294967296.0*
```

```
            Control_System_sampleTime;
   Control_System_tickLo = floor(remainder/Control_System_sampleTime + .5);
   remainder -= Control_System_tickLo*Control_System_sampleTime;
   if (fabsf(remainder) > 1e-6*fabsf(time))
   {
      Control_System_errorStatus =
         "Start time must be an integer multiple of the base sample time.";
   }

   /* Target pre-initialization */
   Control_System_initHal();


   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Integrator' */
   Control_System_X.Integrator_x = 0.f;

   /* Initialization for Discrete Integrator : 'Control System/DC-Voltage
Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_first = -1;
   Control_System_X.DiscreteIntegrator_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controlller/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_1_first = -1;
   Control_System_X.DiscreteIntegrator_1_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controlller/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_2_first = -1;
   Control_System_X.DiscreteIntegrator_2_i1_x = 0;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn[0] = 0.f;
   Control_System_X.TransferFcn[1] = 0.f;

   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Continuous PID\nController/Continous Time/Ki
Integrator/Ki ~= 0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_x = 0;
   Control_System_X.Integrator1_i1_first = 1;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_1 = 0.f;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_2 = -0.00450158158f;
}

void Control_System_step(void)
{
   if (Control_System_errorStatus)
   {
      return;
   }

   /* ADC : 'Control System/ADC' */
   Control_System_B.ADC[0] = PLXHAL_ADC_getIn(0, 0);
   Control_System_B.ADC[1] = PLXHAL_ADC_getIn(0, 1);
```

```
   Control_System_B.ADC[2] = PLXHAL_ADC_getIn(0, 2);
   Control_System_B.ADC[3] = PLXHAL_ADC_getIn(0, 3);
   Control_System_B.ADC[4] = PLXHAL_ADC_getIn(0, 4);
   Control_System_B.ADC[5] = PLXHAL_ADC_getIn(0, 5);
   Control_System_B.ADC[6] = PLXHAL_ADC_getIn(0, 6);
   Control_System_B.ADC[7] = PLXHAL_ADC_getIn(0, 7);

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Integrator' */
   if (Control_System_X.Integrator_x > 6.28318531f ||
       Control_System_X.Integrator_x < 0.f)
   {
      float span = 6.28318531f - (0.f);
      Control_System_X.Integrator_x -= 0.f;
      Control_System_X.Integrator_x = Control_System_X.Integrator_x - span*
                                      floorf(
                                             Control_System_X.Integrator_x/
                                             span) + (0.f);
   }
   Control_System_B.Integrator = Control_System_X.Integrator_x;

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Cos' */
   Control_System_B.Cos = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Sin' */
   Control_System_B.Sin = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->d' */
   Control_System_B.abc_d = 0.666666687f *
                            (((Control_System_B.ADC[5] *
                               Control_System_B.Cos) +
                              (Control_System_B.ADC[6] *
                               (((-0.5f) *
                                 Control_System_B.Cos) +
     (0.866025388f *
     Control_System_B.Sin)))) +
                             (Control_System_B.ADC[7] *
     (((-0.5f) *
       Control_System_B.Cos) - (0.866025388f * Control_System_B.Sin))));

   /* Zero-Order Hold : 'Control System/DC-Voltage Controll/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/DC-Voltage Controll/Sum'
    *  Constant : 'Control System/VDC*'
    */
   Control_System_B.Zero_OrderHold = Control_System_P.VDC__Value -
                                     Control_System_B.ADC[0];

   /* Discrete Integrator : 'Control System/DC-Voltage Controll/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Reset
select/None/Constant'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Init
select/internal/Constant'
    *  Sum : 'Control System/DC-Voltage Controll/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/DC-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
```

```
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_first)
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_i2_prevU +
          (Control_System_B.Zero_OrderHold * 3.14159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/DC-Voltage Controll/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/DC-Voltage Controll/Discrete PID\nController/Discrete
Time/Sum'
    *  Product : 'Control System/DC-Voltage Controll/Discrete
PID\nController/Discrete Time/Product'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation =
      (Control_System_B.Zero_OrderHold *
       0.219911486f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator;
   /* Saturation : 'Control System/DC-Voltage Controll/Saturation' */
   Control_System_B.Saturation_1 = Control_System_B.abc_d;
   if (Control_System_B.Saturation_1 < 300.f)
   {
      Control_System_B.Saturation_1 = 300.f;
   }

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF1/Cos' */
   Control_System_B.Cos_1 = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF1/Sin' */
   Control_System_B.Sin_1 = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF1/abc->d' */
   Control_System_B.abc_d_1 = 0.666666687f *
                              (((Control_System_B.ADC[2] *
                                 Control_System_B.Cos_1) +
                                (Control_System_B.ADC[3] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_1) +
     (0.866025388f *
     Control_System_B.Sin_1)))) +
                               (Control_System_B.ADC[4] *
     (((-0.5f) *
       Control_System_B.Cos_1) - (0.866025388f * Control_System_B.Sin_1))));

   /* Zero-Order Hold : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Sum1'
```

```
    *  Product : 'Control System/DC-Voltage Controll/Divide'
    *  Product : 'Control System/DC-Voltage Controll/Product1'
    *  Sum : 'Control System/DC-Voltage Controll/Sum1'
    *  Gain : 'Control System/DC-Voltage Controll/Gain'
    */
   Control_System_B.Zero_OrderHold_1 =
      (((-Control_System_B.Saturation +
         Control_System_B.ADC[1]) *
        Control_System_B.ADC[0]) /
       (1.5f*Control_System_B.Saturation_1)) - Control_System_B.abc_d_1;

   /* Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_1_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_1_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_1 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_1_first)
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_1_i2_prevU +
          (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controlller/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Product'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_2 =
      (Control_System_B.Zero_OrderHold_1 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_1;
   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF1/abc->q' */
   Control_System_B.abc_q = 0.666666687f *
                            ((((-Control_System_B.ADC[2]) *
                               Control_System_B.Sin_1) +
                              (Control_System_B.ADC[3] *
```

```
                              ((0.5f *
                                Control_System_B.Sin_1) +
     (0.866025388f *
     Control_System_B.Cos_1)))) +
                             (Control_System_B.ADC[4] *
     ((0.5f *
       Control_System_B.Sin_1) - (0.866025388f * Control_System_B.Cos_1))));

   /* Gain : 'Control System/Current Controlller/Gain' */
   Control_System_B.Gain = 0.5f*Control_System_B.ADC[0];

   /* Product : 'Control System/Current Controlller/Product2'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Sum2'
    *  Sum : 'Control System/Current Controlller/Sum7'
    *  Gain : 'Control System/Current Controlller/Gain2'
    */
   Control_System_B.Product2 =
      (Control_System_B.abc_d +
       (Control_System_B.Saturation_2 +
     (0.314159265f*Control_System_B.abc_q))) / Control_System_B.Gain;

   /* Trigonometric Function : 'Control System/Current Controlller/RRF->3ph/Cos' */
   Control_System_B.Cos_2 = cosf(Control_System_B.Integrator);

   /* Zero-Order Hold : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Sum3'
    *  Constant : 'Control System/Constant'
    */
   Control_System_B.Zero_OrderHold_2 = -Control_System_B.abc_q + 0.f;

   /* Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_2_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_2_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_2 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_2_first)
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_2_i2_prevU +
          (Control_System_B.Zero_OrderHold_2 * 31.4159265f * 1.f) + 0.f);
```

```
   }

   /* Saturation : 'Control System/Current Controlller/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController1/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/Current Controlller/Discrete
PID\nController1/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_3 =
      (Control_System_B.Zero_OrderHold_2 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_2;
   /* Product : 'Control System/Current Controlller/Product3'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Sum4'
    *  Sum : 'Control System/Current Controlller/Sum6'
    *  Gain : 'Control System/Current Controlller/Gain1'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->q'
    */
   Control_System_B.Product3 = 1.f / Control_System_B.Gain *
                               ((-(0.314159265f*
                                   Control_System_B.abc_d_1) +
                                 Control_System_B.Saturation_3) +
                                (0.666666687f *
     ((((-Control_System_B.ADC[5]) *
        Control_System_B.Sin) +
       (Control_System_B.ADC[6] *
       ((0.5f *
         Control_System_B.Sin) +
     (0.866025388f *
     Control_System_B.Cos)))) +
     (Control_System_B.ADC[7] *
     ((0.5f *
       Control_System_B.Sin) - (0.866025388f * Control_System_B.Cos))))));

   /* Trigonometric Function : 'Control System/Current Controlller/RRF->3ph/Sin' */
   Control_System_B.Sin_2 = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Current Controlller/RRF->3ph/dq->a' */
   Control_System_B.dq_a =
      (Control_System_B.Product2 *
       Control_System_B.Cos_2) -
      (Control_System_B.Product3 * Control_System_B.Sin_2);

   /* Function : 'Control System/Current Controlller/RRF->3ph/dq->b' */
   Control_System_B.dq_b =
      (Control_System_B.Product2 *
       (((-0.5f) *
         Control_System_B.Cos_2) +
     (0.866025388f *
     Control_System_B.Sin_2))) +
      (Control_System_B.Product3 *
       ((0.5f *
         Control_System_B.Sin_2) + (0.866025388f * Control_System_B.Cos_2)));

   /* PWM  : 'Control System/PWM a' */
   {
      PLXHAL_PWM_setDuty(0, 0.5f + (0.5f*Control_System_B.dq_a));
   }
   /* PWM  : 'Control System/PWM b' */
   {
      PLXHAL_PWM_setDuty(1, 0.5f + (0.5f*Control_System_B.dq_b));
   }
```

```
   /* PWM  : 'Control System/PWM c' */
   {
      PLXHAL_PWM_setDuty(2, 0.5f +
                         (0.5f*
                          (-Control_System_B.dq_b - Control_System_B.dq_a)));
   }
   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/beta' */
   Control_System_B.beta = 0.577350259f *
                           (Control_System_B.ADC[6] -
                            Control_System_B.ADC[7]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction1' */
   Control_System_B.TrigonometricFunction1 =
      cosf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/alpha' */
   Control_System_B.alpha = 0.333333343f *
                            (((2.f *
                               Control_System_B.ADC[5]) -
                              Control_System_B.ADC[6]) -
                             Control_System_B.ADC[7]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction' */
   Control_System_B.TrigonometricFunction = sinf(Control_System_B.Integrator);

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn[0] = 222.144147f*
                                     Control_System_X.TransferFcn[0];
   Control_System_B.TransferFcn[1] = 222.144147f*
                                     Control_System_X.TransferFcn[1];

   /* Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain1'
    * incorporates
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Product'
    */
   Control_System_B.Gain1 = 2.f*
                            (Control_System_B.TrigonometricFunction *
                             Control_System_B.TrigonometricFunction1);

   /* Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Sum'
    * incorporates
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Constant'
    *  Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain'
    *  Math Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Math'
    */
   Control_System_B.Sum = (-1.f) +
                          (2.f*
                           (Control_System_B.TrigonometricFunction1 *
     Control_System_B.TrigonometricFunction1));

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn1'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->q'
    */
   Control_System_B.Fcn1 =
```

```
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) +
       (Control_System_B.TransferFcn[0] *
     Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn[1] * Control_System_B.Sum);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->d'
    */
   Control_System_B.Fcn =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn[0] *
     Control_System_B.Sum)) -
      (Control_System_B.TransferFcn[1] * Control_System_B.Gain1);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   {
      float u, y;
      y =
         sqrtf(powf(Control_System_B.Fcn,2.f) + powf(Control_System_B.Fcn1,
                                                     2.f));
      u = 0.0001f;
      if (u > y)
         y = u;
      Control_System_B.Normalize = y;
   }

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   Control_System_B.Normalize_i1 = Control_System_B.Fcn1 /
                                   Control_System_B.Normalize;

   /* Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Reset select/None/Constant' */
   Control_System_B.Constant = 0.f;

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   if (Control_System_X.Integrator1_i1_first ||
       (!Control_System_X.Integrator1_i2_prevReset &&
        Control_System_B.Constant))
   {
      Control_System_X.Integrator1_x = 314.159265f;
   }
   Control_System_B.Integrator1 = Control_System_X.Integrator1_x;

   /* Saturation : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Saturation/internal/Saturation
Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product3'
```

```
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum1'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Kd Integrator/Kd = 0/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant3'
    */
   Control_System_B.Saturation_4 =
      (Control_System_B.Normalize_i1 *
       314.159265f) +
      (((Control_System_B.Normalize_i1 *
         0.f) - 0.f) * 0.f) + Control_System_B.Integrator1;
   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_1 = 222.144147f*
                                    Control_System_X.TransferFcn_1;

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_2 = 222.144147f*
                                    Control_System_X.TransferFcn_2;
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Update for Discrete Integrator : 'Control System/DC-Voltage Controll/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/DC-Voltage Controll/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/DC-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/DC-Voltage Controll/Discrete PID\nController/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_first = 0;
   Control_System_X.DiscreteIntegrator_i1_x =
      Control_System_B.DiscreteIntegrator;
   Control_System_X.DiscreteIntegrator_i2_prevU =
      (Control_System_B.Zero_OrderHold * 3.14159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
```

```
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_1_first = 0;
   Control_System_X.DiscreteIntegrator_1_i1_x =
      Control_System_B.DiscreteIntegrator_1;
   Control_System_X.DiscreteIntegrator_1_i2_prevU =
      (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_1_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_2_first = 0;
   Control_System_X.DiscreteIntegrator_2_i1_x =
      Control_System_B.DiscreteIntegrator_2;
   Control_System_X.DiscreteIntegrator_2_i2_prevU =
      (Control_System_B.Zero_OrderHold_2 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_2_i3_prevReset = !!(0.f);

   /* Update for PWM  : 'Control System/PWM a' */
   PLXHAL_PWM_enableAllOutputs();

   /* Update for Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_i1_first = 0;
   Control_System_X.Integrator1_i2_prevReset = !!(Control_System_B.Constant);

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Integrator' */
   Control_System_deriv[5] = Control_System_B.Saturation_4;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->d'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn1'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->q'
    */
   Control_System_deriv[2] =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1 *
     Control_System_B.Sum)) +
      (Control_System_B.TransferFcn_2 *
```

```
       Control_System_B.Gain1)-222.144147f*Control_System_X.TransferFcn[0];
   Control_System_deriv[3] =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1 *
     Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn_2 *
       Control_System_B.Sum)-222.144147f*Control_System_X.TransferFcn[1];

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum3'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   Control_System_deriv[0] =
      (Control_System_B.Normalize_i1 * 56982.1876f * 1.f) + 0.f;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_deriv[1] = Control_System_B.Fcn-222.144147f*
                             Control_System_X.TransferFcn_1;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_deriv[4] = Control_System_B.Fcn1-222.144147f*
                             Control_System_X.TransferFcn_2;

   /* Update continuous states */
   Control_System_X.Integrator_x += 0.0001f*Control_System_deriv[5];
   Control_System_X.TransferFcn[0] += 0.0001f*Control_System_deriv[2];
   Control_System_X.TransferFcn[1] += 0.0001f*Control_System_deriv[3];
   Control_System_X.Integrator1_x += 0.0001f*Control_System_deriv[0];
   Control_System_X.TransferFcn_1 += 0.0001f*Control_System_deriv[1];
   Control_System_X.TransferFcn_2 += 0.0001f*Control_System_deriv[4];
}

void Control_System_terminate()
{
}
```

A.2.2.- Power Control

```
/*
 * Implementation file for: Power Control/Control System
 * Generated with         : PLECS 4.7.2
 *                          TI2833x 1.5.2
 * Generated on           : 8 Jun 2023 16:24:08
 */
#include "Control_System.h"
#ifndef PLECS_HEADER_Control_System_h_
#error The wrong header file "Control_System.h" was included. Please check
#error your include path to see whether this file name conflicts with the
#error name of another header file.
#endif /* PLECS_HEADER_Control_System_h_ */
#if defined(__GNUC__) && (__GNUC__ > 4)
#   define _ALIGNMENT 16
#   define _RESTRICT __restrict
#   define _ALIGN __attribute__((aligned(_ALIGNMENT)))
#   if defined(__clang__)
#      if __has_builtin(__builtin_assume_aligned)
#         define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#      else
#         define _ASSUME_ALIGNED(a) a
#      endif
#   else
#      define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#   endif
#else
#   ifndef _RESTRICT
#      define _RESTRICT
#   endif
#   ifndef _ALIGN
#      define _ALIGN
#   endif
#   ifndef _ASSUME_ALIGNED
#      define _ASSUME_ALIGNED(a) a
#   endif
#endif
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>
#include "plx_hal.h"
#define PLECSRunTimeError(msg) Control_System_errorStatus = msg
static const float Control_System_UNCONNECTED = 0;
static float Control_System_deriv[18] _ALIGN;
static uint32_t Control_System_tickLo;
static int32_t Control_System_tickHi;
Control_System_BlockOutputs Control_System_B;
#if defined(EXTERNAL_MODE) && EXTERNAL_MODE
const float * const Control_System_ExtModeSignals[] = {
   &Control_System_B.TransferFcn,
   &Control_System_P.P__Value
};
#endif /* defined(EXTERNAL_MODE) */
Control_System_Parameters Control_System_P = {
   /* Parameter 'Value' of
    *  Constant : 'Control System/P*'
    */
   30000.f,
   /* Parameter 'Value' of
    *  Constant : 'Control System/Q*'
    */
   1000.f
};
Control_System_ModelStates Control_System_X _ALIGN;
const char * Control_System_errorStatus;
const float Control_System_sampleTime = 0.0001f;
const char * const Control_System_checksum =
   "03f267b33ca9f3f227f3802bb6af5ea0fd97abc6";
```

```
/* Target declarations */
extern void Control_System_initHal();

void Control_System_initialize(float time)
{
   float remainder;
   Control_System_errorStatus = NULL;
   Control_System_tickHi =
      floor(time/(4294967296.0*Control_System_sampleTime));
   remainder = time - Control_System_tickHi*4294967296.0*
               Control_System_sampleTime;
   Control_System_tickLo = floor(remainder/Control_System_sampleTime + .5);
   remainder -= Control_System_tickLo*Control_System_sampleTime;
   if (fabsf(remainder) > 1e-6*fabsf(time))
   {
      Control_System_errorStatus =
         "Start time must be an integer multiple of the base sample time.";
   }

   /* Target pre-initialization */
   Control_System_initHal();


   /* Initialization for Transfer Function : 'Control System/Transfer Fcn' */
   Control_System_X.TransferFcn[0] = 0.f;
   Control_System_X.TransferFcn[1] = 0.f;

   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Integrator' */
   Control_System_X.Integrator_x = 0.f;

   /* Initialization for Discrete Integrator : 'Control System/Power Control/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_first = -1;
   Control_System_X.DiscreteIntegrator_i1_x = 0;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Transfer Fcn2' */
   Control_System_X.TransferFcn2[0] = 0.f;
   Control_System_X.TransferFcn2[1] = 0.f;
   Control_System_X.TransferFcn2[2] = 0.f;
   Control_System_X.TransferFcn2[3] = 0.f;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controlller/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_1_first = -1;
   Control_System_X.DiscreteIntegrator_1_i1_x = 0;

   /* Initialization for Transfer Function : 'Control System/Transfer Fcn1' */
   Control_System_X.TransferFcn1[0] = 0.f;
   Control_System_X.TransferFcn1[1] = 0.f;

   /* Initialization for Discrete Integrator : 'Control System/Power Control/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_2_first = -1;
   Control_System_X.DiscreteIntegrator_2_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controlller/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_3_first = -1;
   Control_System_X.DiscreteIntegrator_3_i1_x = 0;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Transfer Fcn1' */
```

```
   Control_System_X.TransferFcn1_1[0] = 0.f;
   Control_System_X.TransferFcn1_1[1] = 0.f;
   Control_System_X.TransferFcn1_1[2] = 0.f;
   Control_System_X.TransferFcn1_1[3] = 0.f;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_1[0] = 0.f;
   Control_System_X.TransferFcn_1[1] = 0.f;

   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Continuous PID\nController/Continous Time/Ki
Integrator/Ki ~= 0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_x = 0;
   Control_System_X.Integrator1_i1_first = 1;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_2 = 0.f;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_3 = -0.00450158158f;
}

void Control_System_step(void)
{
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Transfer Function : 'Control System/Transfer Fcn' */
   Control_System_B.TransferFcn = 35530.5758f*Control_System_X.TransferFcn[0];

   /* ADC : 'Control System/ADC' */
   Control_System_B.ADC[0] = PLXHAL_ADC_getIn(0, 0);
   Control_System_B.ADC[1] = PLXHAL_ADC_getIn(0, 1);
   Control_System_B.ADC[2] = PLXHAL_ADC_getIn(0, 2);
   Control_System_B.ADC[3] = PLXHAL_ADC_getIn(0, 3);
   Control_System_B.ADC[4] = PLXHAL_ADC_getIn(0, 4);
   Control_System_B.ADC[5] = PLXHAL_ADC_getIn(0, 5);
   Control_System_B.ADC[6] = PLXHAL_ADC_getIn(0, 6);
   Control_System_B.ADC[7] = PLXHAL_ADC_getIn(0, 7);
   Control_System_B.ADC[8] = PLXHAL_ADC_getIn(0, 8);
   Control_System_B.ADC[9] = PLXHAL_ADC_getIn(0, 9);
   Control_System_B.ADC[10] = PLXHAL_ADC_getIn(0, 10);
   Control_System_B.ADC[11] = PLXHAL_ADC_getIn(0, 11);
   Control_System_B.ADC[12] = PLXHAL_ADC_getIn(0, 12);
   Control_System_B.ADC[13] = PLXHAL_ADC_getIn(0, 13);

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Integrator' */
   if (Control_System_X.Integrator_x > 6.28318531f ||
       Control_System_X.Integrator_x < 0.f)
   {
      float span = 6.28318531f - (0.f);
      Control_System_X.Integrator_x -= 0.f;
      Control_System_X.Integrator_x = Control_System_X.Integrator_x - span*
                                      floorf(
                                             Control_System_X.Integrator_x/
                                             span) + (0.f);
   }
   Control_System_B.Integrator = Control_System_X.Integrator_x;
```

```
   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Cos' */
   Control_System_B.Cos = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Sin' */
   Control_System_B.Sin = sinf(Control_System_B.Integrator);

   /* Zero-Order Hold : 'Control System/Power Control/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Power Control/Sum'
    *  Constant : 'Control System/P*'
    */
   Control_System_B.Zero_OrderHold = Control_System_P.P__Value -
                                     Control_System_B.TransferFcn;

   /* Discrete Integrator : 'Control System/Power Control/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Reset
select/None/Constant'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Init
select/internal/Constant'
    *  Sum : 'Control System/Power Control/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Power Control/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_first)
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_i2_prevU +
          (Control_System_B.Zero_OrderHold * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Power Control/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Power Control/Discrete PID\nController/Discrete
Time/Sum'
    *  Product : 'Control System/Power Control/Discrete PID\nController/Discrete
Time/Product'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation =
      (Control_System_B.Zero_OrderHold *
       0.314159265f) + Control_System_UNCONNECTED +
```

```
      Control_System_B.DiscreteIntegrator;
   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Transfer Fcn2'
*/
   Control_System_B.TransferFcn2[0] = 3947.84176f*
                                      Control_System_X.TransferFcn2[0];
   Control_System_B.TransferFcn2[1] = 3947.84176f*
                                      Control_System_X.TransferFcn2[2];

   /* Saturation : 'Control System/Power Control/Saturation' */
   Control_System_B.Saturation_1 = Control_System_B.TransferFcn2[0];
   if (Control_System_B.Saturation_1 < 300.f)
   {
      Control_System_B.Saturation_1 = 300.f;
   }

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Cos' */
   Control_System_B.Cos_1 = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Sin' */
   Control_System_B.Sin_1 = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->d' */
   Control_System_B.abc_d = 0.666666687f *
                            (((Control_System_B.ADC[0] *
                               Control_System_B.Cos_1) +
                              (Control_System_B.ADC[1] *
                               (((-0.5f) *
                                 Control_System_B.Cos_1) +
                                (0.866025388f *
      Control_System_B.Sin_1)))) +
                             (Control_System_B.ADC[2] *
                              (((-0.5f) *
                                Control_System_B.Cos_1) -
   (0.866025388f * Control_System_B.Sin_1))));

   /* Zero-Order Hold : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Sum1'
    *  Product : 'Control System/Power Control/Divide'
    *  Gain : 'Control System/Power Control/Gain'
    */
   Control_System_B.Zero_OrderHold_1 =
      (Control_System_B.Saturation /
       (1.5f*Control_System_B.Saturation_1)) - Control_System_B.abc_d;

   /* Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_1_first < 0 ||
```

```
       (!Control_System_X.DiscreteIntegrator_1_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_1 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_1_first)
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_1_i2_prevU +
          (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controlller/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Product'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_2 =
      (Control_System_B.Zero_OrderHold_1 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_1;
   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->q' */
   Control_System_B.abc_q = 0.666666687f *
                            ((((-Control_System_B.ADC[0]) *
                               Control_System_B.Sin_1) +
                              (Control_System_B.ADC[1] *
                               ((0.5f *
                                 Control_System_B.Sin_1) +
                                (0.866025388f *
      Control_System_B.Cos_1)))) +
                             (Control_System_B.ADC[2] *
                              ((0.5f *
                                Control_System_B.Sin_1) -
   (0.866025388f * Control_System_B.Cos_1))));

   /* Gain : 'Control System/Current Controlller/Gain' */
   Control_System_B.Gain = 0.5f*Control_System_B.ADC[6];

   /* Product : 'Control System/Current Controlller/Product2'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Sum2'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->d'
    *  Sum : 'Control System/Current Controlller/Sum7'
    *  Gain : 'Control System/Current Controlller/Gain2'
    */
   Control_System_B.Product2 =
      ((0.666666687f *
        (((Control_System_B.ADC[3] *
           Control_System_B.Cos) +
          (Control_System_B.ADC[4] *
           (((-0.5f) *
             Control_System_B.Cos) +
            (0.866025388f *
      Control_System_B.Sin)))) +
         (Control_System_B.ADC[5] *
          (((-0.5f) *
            Control_System_B.Cos) -
```

```
      (0.866025388f *
      Control_System_B.Sin))))) +
       (Control_System_B.Saturation_2 +
        (0.314159265f*Control_System_B.abc_q))) / Control_System_B.Gain;

   /* Trigonometric Function : 'Control System/Current Controlller/RRF->3ph/Cos' */
   Control_System_B.Cos_2 = cosf(Control_System_B.Integrator);

   /* Zero-Order Hold : 'Control System/Power Control/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Power Control/Sum1'
    *  Constant : 'Control System/Q*'
    *  Transfer Function : 'Control System/Transfer Fcn1'
    */
   Control_System_B.Zero_OrderHold_2 = Control_System_P.Q__Value -
                                       (35530.5758f*
                                        Control_System_X.TransferFcn1[0]);

   /* Discrete Integrator : 'Control System/Power Control/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Reset
select/None/Constant'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Init
select/internal/Constant'
    *  Sum : 'Control System/Power Control/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Power Control/Discrete PID\nController1/Discrete
Time/Product4'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_2_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_2_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_2 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_2_first)
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_2_i2_prevU +
          (Control_System_B.Zero_OrderHold_2 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Power Control/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Power Control/Discrete PID\nController1/Discrete
Time/Sum'
    *  Product : 'Control System/Power Control/Discrete PID\nController1/Discrete
Time/Product'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_3 =
```

```
      (Control_System_B.Zero_OrderHold_2 *
       0.314159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_2;
   /* Zero-Order Hold : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Sum3'
    *  Product : 'Control System/Power Control/Divide1'
    *  Gain : 'Control System/Power Control/Gain1'
    */
   Control_System_B.Zero_OrderHold_3 = -Control_System_B.abc_q +
                                       (Control_System_B.Saturation_3 /
                                        (-1.5f*
                                         Control_System_B.Saturation_1));

   /* Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_3_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_3_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_3 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_3_first)
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_3_i2_prevU +
          (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controlller/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController1/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/Current Controlller/Discrete
PID\nController1/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_4 =
      (Control_System_B.Zero_OrderHold_3 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_3;
   /* Product : 'Control System/Current Controlller/Product3'
```

```
     * incorporates
     *  Sum : 'Control System/Current Controlller/Sum4'
     *  Sum : 'Control System/Current Controlller/Sum6'
     *  Gain : 'Control System/Current Controlller/Gain1'
     *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->q'
     */
    Control_System_B.Product3 = 1.f / Control_System_B.Gain *
                                ((-(0.314159265f*
                                    Control_System_B.abc_d) +
                                  Control_System_B.Saturation_4) +
                                 (0.666666687f *
                                  ((((-Control_System_B.ADC[3]) *
                                     Control_System_B.Sin) +
                                    (Control_System_B.ADC[4] *
         ((0.5f *
           Control_System_B.Sin) +
         (0.866025388f *
        Control_System_B.Cos)))) +
        (Control_System_B.ADC[5] *
        ((0.5f *
          Control_System_B.Sin) - (0.866025388f * Control_System_B.Cos))))));

    /* Trigonometric Function : 'Control System/Current Controlller/RRF->3ph/Sin' */
    Control_System_B.Sin_2 = sinf(Control_System_B.Integrator);

    /* Function : 'Control System/Current Controlller/RRF->3ph/dq->a' */
    Control_System_B.dq_a =
        (Control_System_B.Product2 *
         Control_System_B.Cos_2) -
        (Control_System_B.Product3 * Control_System_B.Sin_2);

    /* Function : 'Control System/Current Controlller/RRF->3ph/dq->b' */
    Control_System_B.dq_b =
        (Control_System_B.Product2 *
         (((-0.5f) *
           Control_System_B.Cos_2) +
          (0.866025388f *
        Control_System_B.Sin_2))) +
        (Control_System_B.Product3 *
         ((0.5f *
           Control_System_B.Sin_2) + (0.866025388f * Control_System_B.Cos_2)));

    /* PWM  : 'Control System/PWM a' */
    {
        PLXHAL_PWM_setDuty(0, 0.5f + (0.5f*Control_System_B.dq_a));
    }
    /* PWM  : 'Control System/PWM b' */
    {
        PLXHAL_PWM_setDuty(1, 0.5f + (0.5f*Control_System_B.dq_b));
    }
    /* PWM  : 'Control System/PWM c' */
    {
        PLXHAL_PWM_setDuty(2, 0.5f +
                           (0.5f*
                            (-Control_System_B.dq_b - Control_System_B.dq_a)));
    }
    /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF1/Cos' */
    Control_System_B.Cos_3 = cosf(Control_System_B.Integrator);

    /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF1/Sin' */
    Control_System_B.Sin_3 = sinf(Control_System_B.Integrator);

    /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/beta' */
    Control_System_B.beta = 0.577350259f *
                            (Control_System_B.ADC[12] -
```

```
                          Control_System_B.ADC[13]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction1' */
   Control_System_B.TrigonometricFunction1 =
      cosf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/alpha' */
   Control_System_B.alpha = 0.333333343f *
                            (((2.f *
                               Control_System_B.ADC[11]) -
                              Control_System_B.ADC[12]) -
                             Control_System_B.ADC[13]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction' */
   Control_System_B.TrigonometricFunction = sinf(Control_System_B.Integrator);

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_1[0] = 222.144147f*
                                       Control_System_X.TransferFcn_1[0];
   Control_System_B.TransferFcn_1[1] = 222.144147f*
                                       Control_System_X.TransferFcn_1[1];

   /* Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain1'
    * incorporates
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Product'
    */
   Control_System_B.Gain1 = 2.f*
                            (Control_System_B.TrigonometricFunction *
                             Control_System_B.TrigonometricFunction1);

   /* Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Sum'
    * incorporates
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Constant'
    *  Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain'
    *  Math Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Math'
    */
   Control_System_B.Sum = (-1.f) +
                          (2.f*
                           (Control_System_B.TrigonometricFunction1 *
                            Control_System_B.TrigonometricFunction1));

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn1'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->q'
    */
   Control_System_B.Fcn1 =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) +
       (Control_System_B.TransferFcn_1[0] *
        Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn_1[1] * Control_System_B.Sum);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn'
```

```
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->d'
    */
   Control_System_B.Fcn =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1[0] *
        Control_System_B.Sum)) -
      (Control_System_B.TransferFcn_1[1] * Control_System_B.Gain1);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   {
      float u, y;
      y =
         sqrtf(powf(Control_System_B.Fcn,2.f) + powf(Control_System_B.Fcn1,
                                                     2.f));
      u = 0.0001f;
      if (u > y)
         y = u;
      Control_System_B.Normalize = y;
   }

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   Control_System_B.Normalize_i1 = Control_System_B.Fcn1 /
                                   Control_System_B.Normalize;

   /* Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Reset select/None/Constant' */
   Control_System_B.Constant = 0.f;

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   if (Control_System_X.Integrator1_i1_first ||
       (!Control_System_X.Integrator1_i2_prevReset &&
        Control_System_B.Constant))
   {
      Control_System_X.Integrator1_x = 314.159265f;
   }
   Control_System_B.Integrator1 = Control_System_X.Integrator1_x;

   /* Saturation : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Saturation/internal/Saturation
Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product3'
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum1'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Kd Integrator/Kd = 0/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant3'
```

```
    */
   Control_System_B.Saturation_5 =
      (Control_System_B.Normalize_i1 *
       314.159265f) +
      (((Control_System_B.Normalize_i1 *
         0.f) - 0.f) * 0.f) + Control_System_B.Integrator1;
   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_2 = 222.144147f*
                                    Control_System_X.TransferFcn_2;

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_3 = 222.144147f*
                                    Control_System_X.TransferFcn_3;

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Cos' */
   Control_System_B.Cos_4 = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Sin' */
   Control_System_B.Sin_4 = sinf(Control_System_B.Integrator);
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Update for Discrete Integrator : 'Control System/Power Control/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Power Control/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Power Control/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Power Control/Discrete PID\nController/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_first = 0;
   Control_System_X.DiscreteIntegrator_i1_x =
      Control_System_B.DiscreteIntegrator;
   Control_System_X.DiscreteIntegrator_i2_prevU =
      (Control_System_B.Zero_OrderHold * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController/Reset
select/None/Constant'
```

```
    */
   Control_System_X.DiscreteIntegrator_1_first = 0;
   Control_System_X.DiscreteIntegrator_1_i1_x =
      Control_System_B.DiscreteIntegrator_1;
   Control_System_X.DiscreteIntegrator_1_i2_prevU =
      (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_1_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Power Control/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Power Control/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Power Control/Discrete PID\nController1/Discrete
Time/Product4'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Power Control/Discrete PID\nController1/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_2_first = 0;
   Control_System_X.DiscreteIntegrator_2_i1_x =
      Control_System_B.DiscreteIntegrator_2;
   Control_System_X.DiscreteIntegrator_2_i2_prevU =
      (Control_System_B.Zero_OrderHold_2 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_2_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controlller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controlller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controlller/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controlller/Discrete PID\nController1/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_3_first = 0;
   Control_System_X.DiscreteIntegrator_3_i1_x =
      Control_System_B.DiscreteIntegrator_3;
   Control_System_X.DiscreteIntegrator_3_i2_prevU =
      (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_3_i3_prevReset = !!(0.f);

   /* Update for PWM  : 'Control System/PWM a' */
   PLXHAL_PWM_enableAllOutputs();

   /* Update for Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_i1_first = 0;
   Control_System_X.Integrator1_i2_prevReset = !!(Control_System_B.Constant);

   /* Derivatives for Transfer Function : 'Control System/Transfer Fcn'
    * incorporates
    *  Product : 'Control System/Power \nCalcutation/Product2'
```

```
    *  Transfer Function : 'Control System/Bloque de\nTransformaciones/Transfer Fcn1'
    *  Constant : 'Control System/Power \nCalcutation/Constant1'
    */
   Control_System_deriv[14] = Control_System_X.TransferFcn[1];
   Control_System_deriv[15] =
      (3947.84176f*
       Control_System_X.TransferFcn1_1[0]) * 1.5f *
      Control_System_B.TransferFcn2[0]-35530.5758f*
      Control_System_X.TransferFcn[0]-266.532721f*
      Control_System_X.TransferFcn[1];

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Integrator' */
   Control_System_deriv[5] = Control_System_B.Saturation_5;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Transfer Fcn2'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->d'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->q'
    */
   Control_System_deriv[10] = Control_System_X.TransferFcn2[1];
   Control_System_deriv[11] = 0.666666687f *
                              (((Control_System_B.ADC[11] *
                                 Control_System_B.Cos_4) +
                                (Control_System_B.ADC[12] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_4) +
                                  (0.866025388f *
      Control_System_B.Sin_4)))) +
                               (Control_System_B.ADC[13] *
                                (((-0.5f) *
                                  Control_System_B.Cos_4) -
      (0.866025388f *
      Control_System_B.Sin_4))))-3947.84176f*Control_System_X.TransferFcn2[0]-
                              88.8442402f*Control_System_X.TransferFcn2[1];
   Control_System_deriv[12] = Control_System_X.TransferFcn2[3];
   Control_System_deriv[13] = 0.666666687f *
                              ((((-Control_System_B.ADC[11]) *
                                 Control_System_B.Sin_4) +
                                (Control_System_B.ADC[12] *
                                 ((0.5f *
                                   Control_System_B.Sin_4) +
                                  (0.866025388f *
      Control_System_B.Cos_4)))) +
                               (Control_System_B.ADC[13] *
                                ((0.5f *
                                  Control_System_B.Sin_4) -
      (0.866025388f *
      Control_System_B.Cos_4))))-3947.84176f*Control_System_X.TransferFcn2[2]-
                              88.8442402f*Control_System_X.TransferFcn2[3];

   /* Derivatives for Transfer Function : 'Control System/Transfer Fcn1'
    * incorporates
    *  Product : 'Control System/Power \nCalcutation/Product3'
    *  Transfer Function : 'Control System/Bloque de\nTransformaciones/Transfer Fcn1'
    *  Gain : 'Control System/Power \nCalcutation/Gain'
    *  Constant : 'Control System/Power \nCalcutation/Constant1'
    */
   Control_System_deriv[16] = Control_System_X.TransferFcn1[1];
   Control_System_deriv[17] =
      (3947.84176f*
       Control_System_X.TransferFcn1_1[2]) *
      (-1.f*
       Control_System_B.TransferFcn2[0]) * 1.5f-35530.5758f*
      Control_System_X.TransferFcn1[0]-266.532721f*
      Control_System_X.TransferFcn1[1];
```

```
   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Transfer Fcn1'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF1/abc->d'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF1/abc->q'
    */
   Control_System_deriv[6] = Control_System_X.TransferFcn1_1[1];
   Control_System_deriv[7] = 0.666666687f *
                             (((Control_System_B.ADC[8] *
                                Control_System_B.Cos_3) +
                               (Control_System_B.ADC[9] *
                                ((-0.5f) *
                                  Control_System_B.Cos_3) +
                                 (0.866025388f *
      Control_System_B.Sin_3)))) +
                              (Control_System_B.ADC[10] *
                               ((-0.5f) *
                                 Control_System_B.Cos_3) -
      (0.866025388f *
      Control_System_B.Sin_3))))-3947.84176f*
                             Control_System_X.TransferFcn1_1[0]-
                             88.8442402f*Control_System_X.TransferFcn1_1[1];
   Control_System_deriv[8] = Control_System_X.TransferFcn1_1[3];
   Control_System_deriv[9] = 0.666666687f *
                             ((((-Control_System_B.ADC[8]) *
                                Control_System_B.Sin_3) +
                               (Control_System_B.ADC[9] *
                                ((0.5f *
                                  Control_System_B.Sin_3) +
                                 (0.866025388f *
      Control_System_B.Cos_3)))) +
                              (Control_System_B.ADC[10] *
                               ((0.5f *
                                 Control_System_B.Sin_3) -
      (0.866025388f *
      Control_System_B.Cos_3))))-3947.84176f*
                             Control_System_X.TransferFcn1_1[2]-
                             88.8442402f*Control_System_X.TransferFcn1_1[3];

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->d'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn1'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->q'
    */
   Control_System_deriv[2] =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_2 *
        Control_System_B.Sum)) +
      (Control_System_B.TransferFcn_3 *
       Control_System_B.Gain1)-222.144147f*Control_System_X.TransferFcn_1[0];
   Control_System_deriv[3] =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_2 *
        Control_System_B.Gain1)) -
```

```
      (Control_System_B.TransferFcn_3 *
       Control_System_B.Sum)-222.144147f*Control_System_X.TransferFcn_1[1];

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum3'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   Control_System_deriv[0] =
      (Control_System_B.Normalize_i1 * 56982.1876f * 1.f) + 0.f;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_deriv[1] = Control_System_B.Fcn-222.144147f*
                             Control_System_X.TransferFcn_2;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_deriv[4] = Control_System_B.Fcn1-222.144147f*
                             Control_System_X.TransferFcn_3;

   /* Update continuous states */
   Control_System_X.TransferFcn[0] += 0.0001f*Control_System_deriv[14];
   Control_System_X.TransferFcn[1] += 0.0001f*Control_System_deriv[15];
   Control_System_X.Integrator_x += 0.0001f*Control_System_deriv[5];
   Control_System_X.TransferFcn2[0] += 0.0001f*Control_System_deriv[10];
   Control_System_X.TransferFcn2[1] += 0.0001f*Control_System_deriv[11];
   Control_System_X.TransferFcn2[2] += 0.0001f*Control_System_deriv[12];
   Control_System_X.TransferFcn2[3] += 0.0001f*Control_System_deriv[13];
   Control_System_X.TransferFcn1[0] += 0.0001f*Control_System_deriv[16];
   Control_System_X.TransferFcn1[1] += 0.0001f*Control_System_deriv[17];
   Control_System_X.TransferFcn1_1[0] += 0.0001f*Control_System_deriv[6];
   Control_System_X.TransferFcn1_1[1] += 0.0001f*Control_System_deriv[7];
   Control_System_X.TransferFcn1_1[2] += 0.0001f*Control_System_deriv[8];
   Control_System_X.TransferFcn1_1[3] += 0.0001f*Control_System_deriv[9];
   Control_System_X.TransferFcn_1[0] += 0.0001f*Control_System_deriv[2];
   Control_System_X.TransferFcn_1[1] += 0.0001f*Control_System_deriv[3];
   Control_System_X.Integrator1_x += 0.0001f*Control_System_deriv[0];
   Control_System_X.TransferFcn_2 += 0.0001f*Control_System_deriv[1];
   Control_System_X.TransferFcn_3 += 0.0001f*Control_System_deriv[4];
}

void Control_System_terminate()
{
}
```

A.2.3.- AC Voltage Control

```
/*
 * Implementation file for: AC Voltage Control/Control System
 * Generated with         : PLECS 4.7.2
 *                          TI2833x 1.5.2
 * Generated on           : 8 Jun 2023 16:13:45
 */
#include "Control_System.h"
#ifndef PLECS_HEADER_Control_System_h_
#error The wrong header file "Control_System.h" was included. Please check
#error your include path to see whether this file name conflicts with the
#error name of another header file.
#endif /* PLECS_HEADER_Control_System_h_ */
#if defined(__GNUC__) && (__GNUC__ > 4)
#   define _ALIGNMENT 16
#   define _RESTRICT __restrict
#   define _ALIGN __attribute__((aligned(_ALIGNMENT)))
#   if defined(__clang__)
#      if __has_builtin(__builtin_assume_aligned)
#         define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#      else
#         define _ASSUME_ALIGNED(a) a
#      endif
#   else
#      define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#   endif
#else
#   ifndef _RESTRICT
#      define _RESTRICT
#   endif
#   ifndef _ALIGN
#      define _ALIGN
#   endif
#   ifndef _ASSUME_ALIGNED
#      define _ASSUME_ALIGNED(a) a
#   endif
#endif
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>
#include "plx_hal.h"
#define PLECSRunTimeError(msg) Control_System_errorStatus = msg
static const float Control_System_UNCONNECTED = 0;
static float Control_System_deriv[7] _ALIGN;
static uint32_t Control_System_tickLo;
static int32_t Control_System_tickHi;
Control_System_BlockOutputs Control_System_B;
Control_System_ModelStates Control_System_X _ALIGN;
const char * Control_System_errorStatus;
const float Control_System_sampleTime = 0.0001f;
const char * const Control_System_checksum =
   "14b4a77c4ae6a6d1daf13bcaf629a2487b9eb7f9";
/* Target declarations */
extern void Control_System_initHal();

void Control_System_initialize(float time)
{
   float remainder;
   Control_System_errorStatus = NULL;
   Control_System_tickHi =
      floor(time/(4294967296.0*Control_System_sampleTime));
   remainder = time - Control_System_tickHi*4294967296.0*
               Control_System_sampleTime;
   Control_System_tickLo = floor(remainder/Control_System_sampleTime + .5);
   remainder -= Control_System_tickLo*Control_System_sampleTime;
   if (fabsf(remainder) > 1e-6*fabsf(time))
   {
      Control_System_errorStatus =
```

```
          "Start time must be an integer multiple of the base sample time.";
   }

   /* Target pre-initialization */
   Control_System_initHal();


   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Integrator' */
   Control_System_X.Integrator_x = 0.f;

   /* Initialization for Discrete Integrator : 'Control System/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_first = -1;
   Control_System_X.DiscreteIntegrator_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controller/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_1_first = -1;
   Control_System_X.DiscreteIntegrator_1_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_2_first = -1;
   Control_System_X.DiscreteIntegrator_2_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controller/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_3_first = -1;
   Control_System_X.DiscreteIntegrator_3_i1_x = 0;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn[0] = 0.f;
   Control_System_X.TransferFcn[1] = 0.f;

   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Continuous PID\nController/Continous Time/Ki
Integrator/Ki ~= 0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_x = 0;
   Control_System_X.Integrator1_i1_first = 1;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_1 = 0.f;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_2 = -0.00450158158f;
}

void Control_System_step(void)
{
   if (Control_System_errorStatus)
   {
      return;
   }

   /* ADC : 'Control System/ADC' */
   Control_System_B.ADC[0] = PLXHAL_ADC_getIn(0, 0);
   Control_System_B.ADC[1] = PLXHAL_ADC_getIn(0, 1);
```

```
   Control_System_B.ADC[2] = PLXHAL_ADC_getIn(0, 2);
   Control_System_B.ADC[3] = PLXHAL_ADC_getIn(0, 3);
   Control_System_B.ADC[4] = PLXHAL_ADC_getIn(0, 4);
   Control_System_B.ADC[5] = PLXHAL_ADC_getIn(0, 5);
   Control_System_B.ADC[6] = PLXHAL_ADC_getIn(0, 6);
   Control_System_B.ADC[7] = PLXHAL_ADC_getIn(0, 7);
   Control_System_B.ADC[8] = PLXHAL_ADC_getIn(0, 8);
   Control_System_B.ADC[9] = PLXHAL_ADC_getIn(0, 9);
   Control_System_B.ADC[10] = PLXHAL_ADC_getIn(0, 10);
   Control_System_B.ADC[11] = PLXHAL_ADC_getIn(0, 11);
   Control_System_B.ADC[12] = PLXHAL_ADC_getIn(0, 12);
   Control_System_B.ADC[13] = PLXHAL_ADC_getIn(0, 13);

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Integrator' */
   if (Control_System_X.Integrator_x > 6.28318531f ||
       Control_System_X.Integrator_x < 0.f)
   {
      float span = 6.28318531f - (0.f);
      Control_System_X.Integrator_x -= 0.f;
      Control_System_X.Integrator_x = Control_System_X.Integrator_x - span*
                                      floorf(
                                             Control_System_X.Integrator_x/
                                             span) + (0.f);
   }
   Control_System_B.Integrator = Control_System_X.Integrator_x;

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF4/Cos' */
   Control_System_B.Cos = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF4/Sin' */
   Control_System_B.Sin = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF4/abc->d' */
   Control_System_B.abc_d = 0.666666687f *
                            (((Control_System_B.ADC[3] *
                               Control_System_B.Cos) +
                              (Control_System_B.ADC[4] *
                               (((-0.5f) *
                                 Control_System_B.Cos) +
                                (0.866025388f *
      Control_System_B.Sin)))) +
                             (Control_System_B.ADC[5] *
                              (((-0.5f) *
                                Control_System_B.Cos) -
   (0.866025388f * Control_System_B.Sin))));

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Cos' */
   Control_System_B.Cos_1 = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Sin' */
   Control_System_B.Sin_1 = sinf(Control_System_B.Integrator);

   /* Zero-Order Hold : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Output-Voltage Controll/Sum'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->d'
    */
   Control_System_B.Zero_OrderHold =
      (0.666666687f *
       (((Control_System_B.ADC[11] *
          Control_System_B.Cos_1) +
         (Control_System_B.ADC[12] *
```

```
          (((-0.5f) *
            Control_System_B.Cos_1) +
          (0.866025388f *
        Control_System_B.Sin_1)))) +
          (Control_System_B.ADC[13] *
        (((-0.5f) *
          Control_System_B.Cos_1) -
        (0.866025388f * Control_System_B.Sin_1))))) - Control_System_B.abc_d;

   /* Discrete Integrator : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Reset select/None/Constant'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Init select/internal/Constant'
    *  Sum : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum2'
    *  Product : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_first)
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_i2_prevU +
          (Control_System_B.Zero_OrderHold * 2.37117246f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum'
    *  Product : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation =
      (Control_System_B.Zero_OrderHold *
       0.0486946861f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator;
   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Cos' */
   Control_System_B.Cos_2 = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Sin' */
   Control_System_B.Sin_2 = sinf(Control_System_B.Integrator);
```

```
   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Cos' */
   Control_System_B.Cos_3 = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Sin' */
   Control_System_B.Sin_3 = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->d' */
   Control_System_B.abc_d_1 = 0.666666687f *
                              (((Control_System_B.ADC[0] *
                                 Control_System_B.Cos_3) +
                                (Control_System_B.ADC[1] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_3) +
                                  (0.866025388f *
      Control_System_B.Sin_3)))) +
                               (Control_System_B.ADC[2] *
                                (((-0.5f) *
                                  Control_System_B.Cos_3) -
   (0.866025388f * Control_System_B.Sin_3))));

   /* Zero-Order Hold : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum1'
    *  Sum : 'Control System/Output-Voltage Controll/Sum1'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->d'
    */
   Control_System_B.Zero_OrderHold_1 =
      (Control_System_B.Saturation +
       (0.666666687f *
        (((Control_System_B.ADC[8] *
           Control_System_B.Cos_2) +
          (Control_System_B.ADC[9] *
        (((-0.5f) *
          Control_System_B.Cos_2) +
        (0.866025388f *
      Control_System_B.Sin_2)))) +
      (Control_System_B.ADC[10] *
      (((-0.5f) *
        Control_System_B.Cos_2) -
      (0.866025388f * Control_System_B.Sin_2)))))) - Control_System_B.abc_d_1;

   /* Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_1_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_1_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_1 = 0.f;
```

```
   }
   else if (Control_System_X.DiscreteIntegrator_1_first)
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_1_i2_prevU +
          (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controller/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_1 =
      (Control_System_B.Zero_OrderHold_1 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_1;
   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->q' */
   Control_System_B.abc_q = 0.666666687f *
                            ((((-Control_System_B.ADC[0]) *
                               Control_System_B.Sin_3) +
                              (Control_System_B.ADC[1] *
                               ((0.5f *
                                 Control_System_B.Sin_3) +
                                (0.866025388f *
      Control_System_B.Cos_3)))) +
                             (Control_System_B.ADC[2] *
                              ((0.5f *
                                Control_System_B.Sin_3) -
   (0.866025388f * Control_System_B.Cos_3))));

   /* Gain : 'Control System/Current Controller/Gain' */
   Control_System_B.Gain = 0.5f*Control_System_B.ADC[6];

   /* Product : 'Control System/Current Controller/Product2'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum2'
    *  Sum : 'Control System/Current Controller/Sum7'
    *  Gain : 'Control System/Current Controller/Gain2'
    */
   Control_System_B.Product2 =
      (Control_System_B.abc_d +
       (Control_System_B.Saturation_1 -
        (0.314159265f*Control_System_B.abc_q))) / Control_System_B.Gain;

   /* Trigonometric Function : 'Control System/Current Controller/RRF->3ph/Cos' */
   Control_System_B.Cos_4 = cosf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF4/abc->q' */
   Control_System_B.abc_q_1 = 0.666666687f *
                              ((((-Control_System_B.ADC[3]) *
                                 Control_System_B.Sin) +
                                (Control_System_B.ADC[4] *
                                 ((0.5f *
                                   Control_System_B.Sin) +
                                  (0.866025388f *
```

```
      Control_System_B.Cos)))) +
                               (Control_System_B.ADC[5] *
                                ((0.5f *
                                  Control_System_B.Sin) -
   (0.866025388f * Control_System_B.Cos))));

   /* Zero-Order Hold : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Output-Voltage Controll/Sum2'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->q'
    */
   Control_System_B.Zero_OrderHold_2 =
      (0.666666687f *
       ((((-Control_System_B.ADC[11]) *
          Control_System_B.Sin_1) +
         (Control_System_B.ADC[12] *
          ((0.5f *
            Control_System_B.Sin_1) +
           (0.866025388f *
      Control_System_B.Cos_1)))) +
        (Control_System_B.ADC[13] *
      ((0.5f *
        Control_System_B.Sin_1) -
      (0.866025388f * Control_System_B.Cos_1))))) - Control_System_B.abc_q_1;

   /* Discrete Integrator : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Reset select/None/Constant'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Init select/internal/Constant'
    *  Sum : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum2'
    *  Product : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_2_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_2_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_2 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_2_first)
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_2_i2_prevU +
          (Control_System_B.Zero_OrderHold_2 * 2.37117246f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum'
```

```
    *  Product : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_2 =
      (Control_System_B.Zero_OrderHold_2 *
       0.0486946861f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_2;
   /* Zero-Order Hold : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum3'
    *  Sum : 'Control System/Output-Voltage Controll/Sum3'
    *  Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->q'
    */
   Control_System_B.Zero_OrderHold_3 = -Control_System_B.abc_q +
                                       ((0.666666687f *
                                         (((((-Control_System_B.ADC[8]) *
                                            Control_System_B.Sin_2) +
                                           (Control_System_B.ADC[9] *
                                            ((0.5f *
                                              Control_System_B.Sin_2) +
                                             (0.866025388f *
      Control_System_B.Cos_2)))) +
                                          (Control_System_B.ADC[10] *
                                           ((0.5f *
                                             Control_System_B.Sin_2) -
      (0.866025388f *
      Control_System_B.Cos_2))))) + Control_System_B.Saturation_2);

   /* Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_3_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_3_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_3 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_3_first)
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_3_i2_prevU +
          (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f);
   }
```

```
   /* Saturation : 'Control System/Current Controller/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_3 =
      (Control_System_B.Zero_OrderHold_3 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_3;
   /* Product : 'Control System/Current Controller/Product3'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum4'
    *  Sum : 'Control System/Current Controller/Sum6'
    *  Gain : 'Control System/Current Controller/Gain1'
    */
   Control_System_B.Product3 = 1.f / Control_System_B.Gain *
                               (((0.314159265f*
                                  Control_System_B.abc_d_1) +
                                 Control_System_B.Saturation_3) +
                                Control_System_B.abc_q_1);

   /* Trigonometric Function : 'Control System/Current Controller/RRF->3ph/Sin' */
   Control_System_B.Sin_4 = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Current Controller/RRF->3ph/dq->a' */
   Control_System_B.dq_a =
      (Control_System_B.Product2 *
       Control_System_B.Cos_4) -
      (Control_System_B.Product3 * Control_System_B.Sin_4);

   /* Function : 'Control System/Current Controller/RRF->3ph/dq->b' */
   Control_System_B.dq_b =
      (Control_System_B.Product2 *
       (((-0.5f) *
         Control_System_B.Cos_4) +
        (0.866025388f *
      Control_System_B.Sin_4))) +
      (Control_System_B.Product3 *
       ((0.5f *
         Control_System_B.Sin_4) + (0.866025388f * Control_System_B.Cos_4)));

   /* PWM  : 'Control System/PWM a' */
   {
      PLXHAL_PWM_setDuty(0, 0.5f + (0.5f*Control_System_B.dq_a));
   }
   /* PWM  : 'Control System/PWM b' */
   {
      PLXHAL_PWM_setDuty(1, 0.5f + (0.5f*Control_System_B.dq_b));
   }
   /* PWM  : 'Control System/PWM c' */
   {
      PLXHAL_PWM_setDuty(2, 0.5f +
                         (0.5f*
                          (-Control_System_B.dq_b - Control_System_B.dq_a)));
   }
   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/beta' */
   Control_System_B.beta = 0.577350259f *
                           (Control_System_B.ADC[12] -
                            Control_System_B.ADC[13]);
```

```
   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction1' */
   Control_System_B.TrigonometricFunction1 =
      cosf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/alpha' */
   Control_System_B.alpha = 0.333333343f *
                            (((2.f *
                               Control_System_B.ADC[11]) -
                              Control_System_B.ADC[12]) -
                             Control_System_B.ADC[13]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction' */
   Control_System_B.TrigonometricFunction = sinf(Control_System_B.Integrator);

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn[0] = 222.144147f*
                                     Control_System_X.TransferFcn[0];
   Control_System_B.TransferFcn[1] = 222.144147f*
                                     Control_System_X.TransferFcn[1];

   /* Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain1'
    * incorporates
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Product'
    */
   Control_System_B.Gain1 = 2.f*
                            (Control_System_B.TrigonometricFunction *
                             Control_System_B.TrigonometricFunction1);

   /* Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Sum'
    * incorporates
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Constant'
    *  Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain'
    *  Math Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Math'
    */
   Control_System_B.Sum = (-1.f) +
                          (2.f*
                           (Control_System_B.TrigonometricFunction1 *
                            Control_System_B.TrigonometricFunction1));

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn1'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->q'
    */
   Control_System_B.Fcn1 =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) +
       (Control_System_B.TransferFcn[0] *
        Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn[1] * Control_System_B.Sum);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn'
    * incorporates
```

```
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->d'
    */
   Control_System_B.Fcn =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn[0] *
        Control_System_B.Sum)) -
      (Control_System_B.TransferFcn[1] * Control_System_B.Gain1);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   {
      float u, y;
      y =
         sqrtf(powf(Control_System_B.Fcn,2.f) + powf(Control_System_B.Fcn1,
                                                     2.f));
      u = 0.0001f;
      if (u > y)
         y = u;
      Control_System_B.Normalize = y;
   }

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   Control_System_B.Normalize_i1 = Control_System_B.Fcn1 /
                                   Control_System_B.Normalize;

   /* Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Reset select/None/Constant' */
   Control_System_B.Constant = 0.f;

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   if (Control_System_X.Integrator1_i1_first ||
       (!Control_System_X.Integrator1_i2_prevReset &&
        Control_System_B.Constant))
   {
      Control_System_X.Integrator1_x = 314.159265f;
   }
   Control_System_B.Integrator1 = Control_System_X.Integrator1_x;

   /* Saturation : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Saturation/internal/Saturation
Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product3'
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum1'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Kd Integrator/Kd = 0/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant3'
    */
```

```
   Control_System_B.Saturation_4 =
      (Control_System_B.Normalize_i1 *
       314.159265f) +
      (((Control_System_B.Normalize_i1 *
         0.f) - 0.f) * 0.f) + Control_System_B.Integrator1;
   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_1 = 222.144147f*
                                    Control_System_X.TransferFcn_1;

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_2 = 222.144147f*
                                    Control_System_X.TransferFcn_2;
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Update for Discrete Integrator : 'Control System/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum2'
    *  Product : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController/Reset select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_first = 0;
   Control_System_X.DiscreteIntegrator_i1_x =
      Control_System_B.DiscreteIntegrator;
   Control_System_X.DiscreteIntegrator_i2_prevU =
      (Control_System_B.Zero_OrderHold * 2.37117246f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_1_first = 0;
   Control_System_X.DiscreteIntegrator_1_i1_x =
      Control_System_B.DiscreteIntegrator_1;
   Control_System_X.DiscreteIntegrator_1_i2_prevU =
      (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_1_i3_prevReset = !!(0.f);
```

```
   /* Update for Discrete Integrator : 'Control System/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum2'
    *  Product : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Output-Voltage Controll/Discrete
PID\nController1/Reset select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_2_first = 0;
   Control_System_X.DiscreteIntegrator_2_i1_x =
      Control_System_B.DiscreteIntegrator_2;
   Control_System_X.DiscreteIntegrator_2_i2_prevU =
      (Control_System_B.Zero_OrderHold_2 * 2.37117246f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_2_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_3_first = 0;
   Control_System_X.DiscreteIntegrator_3_i1_x =
      Control_System_B.DiscreteIntegrator_3;
   Control_System_X.DiscreteIntegrator_3_i2_prevU =
      (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_3_i3_prevReset = !!(0.f);

   /* Update for PWM  : 'Control System/PWM a' */
   PLXHAL_PWM_enableAllOutputs();

   /* Update for Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_i1_first = 0;
   Control_System_X.Integrator1_i2_prevReset = !!(Control_System_B.Constant);

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Integrator' */
   Control_System_deriv[5] = Control_System_B.Saturation_4;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn'
```

```
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->d'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn1'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->q'
    */
   Control_System_deriv[2] =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1 *
        Control_System_B.Sum)) +
      (Control_System_B.TransferFcn_2 *
       Control_System_B.Gain1)-222.144147f*Control_System_X.TransferFcn[0];
   Control_System_deriv[3] =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1 *
        Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn_2 *
       Control_System_B.Sum)-222.144147f*Control_System_X.TransferFcn[1];

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum3'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   Control_System_deriv[0] =
      (Control_System_B.Normalize_i1 * 56982.1876f * 1.f) + 0.f;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_deriv[1] = Control_System_B.Fcn-222.144147f*
                             Control_System_X.TransferFcn_1;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_deriv[4] = Control_System_B.Fcn1-222.144147f*
                             Control_System_X.TransferFcn_2;

   /* Update continuous states */
   Control_System_X.Integrator_x += 0.0001f*Control_System_deriv[5];
   Control_System_X.TransferFcn[0] += 0.0001f*Control_System_deriv[2];
   Control_System_X.TransferFcn[1] += 0.0001f*Control_System_deriv[3];
   Control_System_X.Integrator1_x += 0.0001f*Control_System_deriv[0];
   Control_System_X.TransferFcn_1 += 0.0001f*Control_System_deriv[1];
   Control_System_X.TransferFcn_2 += 0.0001f*Control_System_deriv[4];
}

void Control_System_terminate()
{
}
```

A.2.4.- Droop-Control

```
/*
 * Implementation file for: Droop-Control/Control System
 * Generated with         : PLECS 4.7.2
 *                          TI2833x 1.5.2
 * Generated on           : 8 Jun 2023 16:47:50
 */
#include "Control_System.h"
#ifndef PLECS_HEADER_Control_System_h_
#error The wrong header file "Control_System.h" was included. Please check
#error your include path to see whether this file name conflicts with the
#error name of another header file.
#endif /* PLECS_HEADER_Control_System_h_ */
#if defined(__GNUC__) && (__GNUC__ > 4)
#   define _ALIGNMENT 16
#   define _RESTRICT __restrict
#   define _ALIGN __attribute__((aligned(_ALIGNMENT)))
#   if defined(__clang__)
#      if __has_builtin(__builtin_assume_aligned)
#         define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#      else
#         define _ASSUME_ALIGNED(a) a
#      endif
#   else
#      define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#   endif
#else
#   ifndef _RESTRICT
#      define _RESTRICT
#   endif
#   ifndef _ALIGN
#      define _ALIGN
#   endif
#   ifndef _ASSUME_ALIGNED
#      define _ASSUME_ALIGNED(a) a
#   endif
#endif
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>
#include "plx_hal.h"
#define PLECSRunTimeError(msg) Control_System_errorStatus = msg
typedef struct
{
   const size_t numX, numY, numZ;
   const float x0, y0, z0;
   const float dXInv, dYInv, dZInv;
   const float *xVec, *yVec, *zVec, *data;
} plecsLUT;
#define DATA2D(x,y) lut->data[(xIdx+(x))*lut->numY+yIdx+(y)]
#define DATA3D(x,y, \
               z) lut->data[((xIdx+ \
                             (x))*lut->numY+(yIdx+(y)))*lut->numZ+zIdx+(z)]
static size_t calcIndex(float x0, float dXInv, size_t numX, float x,
                        float* fraction)
{
   size_t idx = 0;
   float div = (x-x0)*dXInv;
   if (div > 0)
   {
      idx = div;
      if (idx > numX-2)
         idx = numX-2;
   }
   *fraction = div-idx;
   return idx;
}
static float plecsinterp(float frac, float val0, float val1)
```

```
{
   return (val0 == val1) ? val0 : (1-frac)*val0 + frac*val1;
}
static float plecslu1deven(plecsLUT *lut, float x)
{
   float xFrac = 0.f;
   size_t xIdx = calcIndex(lut->x0, lut->dXInv, lut->numX, x, &xFrac);
   return plecsinterp(xFrac, lut->data[xIdx], lut->data[xIdx+1]);
}
static const float Control_System_UNCONNECTED = 0;
static float Control_System_deriv[12] _ALIGN;
static const float Control_System_lut_0_data[] = {
   314.159,306.305
};
static const float Control_System_lut_1_data[] = {
   325.269,292.742
};
static plecsLUT Control_System_lut[2] = {
   {2, 0, 0, 20000.f, 0.f, 0.f, 3.33333333e-05f, 0.f, 0.f, 0, 0, 0,
    Control_System_lut_0_data},
   {2, 0, 0, 0.f, 0.f, 0.f, 0.01f, 0.f, 0.f, 0, 0, 0,
    Control_System_lut_1_data}
};
static uint32_t Control_System_tickLo;
static int32_t Control_System_tickHi;
Control_System_BlockOutputs Control_System_B;
#if defined(EXTERNAL_MODE) && EXTERNAL_MODE
const float * const Control_System_ExtModeSignals[] = {
   &Control_System_B.abc_d,
   &Control_System_B.abc_q,
   &Control_System_B.Integrator,
   &Control_System_UNCONNECTED,
   &Control_System_UNCONNECTED,
   &Control_System_B.TransferFcn,
   &Control_System_B.Gain,
   &Control_System_B.TransferFcn1,
   &Control_System_B.Saturation
};
#endif /* defined(EXTERNAL_MODE) */
Control_System_ModelStates Control_System_X _ALIGN;
const char * Control_System_errorStatus;
const float Control_System_sampleTime = 0.0001f;
const char * const Control_System_checksum =
   "7f2f802952fd42d032ff7ada723cd76cf7340ea6";
/* Target declarations */
extern void Control_System_initHal();

void Control_System_initialize(float time)
{
   float remainder;
   Control_System_errorStatus = NULL;
   Control_System_tickHi =
      floor(time/(4294967296.0*Control_System_sampleTime));
   remainder = time - Control_System_tickHi*4294967296.0*
               Control_System_sampleTime;
   Control_System_tickLo = floor(remainder/Control_System_sampleTime + .5);
   remainder -= Control_System_tickLo*Control_System_sampleTime;
   if (fabsf(remainder) > 1e-6*fabsf(time))
   {
      Control_System_errorStatus =
         "Start time must be an integer multiple of the base sample time.";
   }

   /* Target pre-initialization */
   Control_System_initHal();
```

```
   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Integrator' */
   Control_System_X.Integrator_x = 0.f;

   /* Initialization for Transfer Function : 'Control System/Droop Control/Transfer
Fcn' */
   Control_System_X.TransferFcn[0] = 0.f;
   Control_System_X.TransferFcn[1] = 0.f;

   /* Initialization for Transfer Function : 'Control System/Droop Control/Transfer
Fcn1' */
   Control_System_X.TransferFcn1[0] = 0.f;
   Control_System_X.TransferFcn1[1] = 0.f;

   /* Initialization for Integrator : 'Control System/Droop Control/Integrator' */
   Control_System_X.Integrator_1_x = 0.f;

   /* Initialization for Discrete Integrator : 'Control System/Droop Control/Output-
Voltage Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_first = -1;
   Control_System_X.DiscreteIntegrator_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controller/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_1_first = -1;
   Control_System_X.DiscreteIntegrator_1_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Droop Control/Output-
Voltage Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_2_first = -1;
   Control_System_X.DiscreteIntegrator_2_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controller/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_3_first = -1;
   Control_System_X.DiscreteIntegrator_3_i1_x = 0;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_1[0] = 0.f;
   Control_System_X.TransferFcn_1[1] = 0.f;

   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Continuous PID\nController/Continous Time/Ki
Integrator/Ki ~= 0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_x = 0;
   Control_System_X.Integrator1_i1_first = 1;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_2 = 0.f;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_3 = -0.00450158158f;
}

void Control_System_step(void)
{
   if (Control_System_errorStatus)
   {
```

```
      return;
   }

   /* ADC : 'Control System/ADC' */
   Control_System_B.ADC[0] = PLXHAL_ADC_getIn(0, 0);
   Control_System_B.ADC[1] = PLXHAL_ADC_getIn(0, 1);
   Control_System_B.ADC[2] = PLXHAL_ADC_getIn(0, 2);
   Control_System_B.ADC[3] = PLXHAL_ADC_getIn(0, 3);
   Control_System_B.ADC[4] = PLXHAL_ADC_getIn(0, 4);
   Control_System_B.ADC[5] = PLXHAL_ADC_getIn(0, 5);
   Control_System_B.ADC[6] = PLXHAL_ADC_getIn(0, 6);
   Control_System_B.ADC[7] = PLXHAL_ADC_getIn(0, 7);
   Control_System_B.ADC[8] = PLXHAL_ADC_getIn(0, 8);
   Control_System_B.ADC[9] = PLXHAL_ADC_getIn(0, 9);
   Control_System_B.ADC[10] = PLXHAL_ADC_getIn(0, 10);
   Control_System_B.ADC[11] = PLXHAL_ADC_getIn(0, 11);
   Control_System_B.ADC[12] = PLXHAL_ADC_getIn(0, 12);
   Control_System_B.ADC[13] = PLXHAL_ADC_getIn(0, 13);

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Integrator' */
   if (Control_System_X.Integrator_x > 6.28318531f ||
       Control_System_X.Integrator_x < 0.f)
   {
      float span = 6.28318531f - (0.f);
      Control_System_X.Integrator_x -= 0.f;
      Control_System_X.Integrator_x = Control_System_X.Integrator_x - span*
                                      floorf(
                                             Control_System_X.Integrator_x/
                                             span) + (0.f);
   }
   Control_System_B.Integrator = Control_System_X.Integrator_x;

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Cos' */
   Control_System_B.Cos = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Sin' */
   Control_System_B.Sin = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->d' */
   Control_System_B.abc_d = 0.666666687f *
                            (((Control_System_B.ADC[11] *
                               Control_System_B.Cos) +
                              (Control_System_B.ADC[12] *
                               (((-0.5f) *
                                 Control_System_B.Cos) +
                                (0.866025388f *
      Control_System_B.Sin)))) +
                             (Control_System_B.ADC[13] *
                              (((-0.5f) *
                                Control_System_B.Cos) -
   (0.866025388f * Control_System_B.Sin))));

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->q' */
   Control_System_B.abc_q = 0.666666687f *
                            ((((-Control_System_B.ADC[11]) *
                               Control_System_B.Sin) +
                              (Control_System_B.ADC[12] *
                               ((0.5f *
                                 Control_System_B.Sin) +
                                (0.866025388f *
      Control_System_B.Cos)))) +
                             (Control_System_B.ADC[13] *
                              ((0.5f *
                                Control_System_B.Sin) -
   (0.866025388f * Control_System_B.Cos))));
```

```
   /* Transfer Function : 'Control System/Droop Control/Transfer Fcn' */
   Control_System_B.TransferFcn = 98696.044f*Control_System_X.TransferFcn[0];

   /* Saturation : 'Control System/Droop Control/Saturation1'
    * incorporates
    *  1D Look-Up Table : 'Control System/Droop Control/1D-Table1'
    */
   Control_System_B.Saturation1 = plecslu1deven(&Control_System_lut[0],
                                                Control_System_B.TransferFcn);
   if (Control_System_B.Saturation1 > 322.013247f)
   {
      Control_System_B.Saturation1 = 322.013247f;
   }
   else if (Control_System_B.Saturation1 < 306.305284f)
   {
      Control_System_B.Saturation1 = 306.305284f;
   }

   /* Gain : 'Control System/Droop Control/Gain' */
   Control_System_B.Gain = 0.159154943f*Control_System_B.Saturation1;

   /* Transfer Function : 'Control System/Droop Control/Transfer Fcn1' */
   Control_System_B.TransferFcn1 = 98696.044f*
                                   Control_System_X.TransferFcn1[0];

   /* Saturation : 'Control System/Droop Control/Saturation'
    * incorporates
    *  1D Look-Up Table : 'Control System/Droop Control/1D-Table'
    */
   Control_System_B.Saturation = plecslu1deven(&Control_System_lut[1],
                                               Control_System_B.TransferFcn1);
   if (Control_System_B.Saturation > 357.796031f)
   {
      Control_System_B.Saturation = 357.796031f;
   }
   else if (Control_System_B.Saturation < 292.742207f)
   {
      Control_System_B.Saturation = 292.742207f;
   }

   /* Integrator : 'Control System/Droop Control/Integrator' */
   Control_System_B.Integrator_1 = Control_System_X.Integrator_1_x;

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF4/Cos' */
   Control_System_B.Cos_1 = cosf(Control_System_B.Integrator_1);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF4/Sin' */
   Control_System_B.Sin_1 = sinf(Control_System_B.Integrator_1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF4/abc->d' */
   Control_System_B.abc_d_1 = 0.666666687f *
                              (((Control_System_B.ADC[3] *
                                 Control_System_B.Cos_1) +
                                (Control_System_B.ADC[4] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_1) +
                                  (0.866025388f *
      Control_System_B.Sin_1)))) +
                               (Control_System_B.ADC[5] *
                                (((-0.5f) *
                                  Control_System_B.Cos_1) -
   (0.866025388f * Control_System_B.Sin_1))));

   /* Zero-Order Hold : 'Control System/Droop Control/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Zero-Order\nHold'
```

```
    * incorporates
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Sum'
    */
   Control_System_B.Zero_OrderHold = Control_System_B.Saturation -
                                     Control_System_B.abc_d_1;

   /* Discrete Integrator : 'Control System/Droop Control/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Reset select/None/Constant'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Init select/internal/Constant'
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum2'
    *  Product : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_first)
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_i2_prevU +
          (Control_System_B.Zero_OrderHold * 2.37117246f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum'
    *  Product : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_1 =
      (Control_System_B.Zero_OrderHold *
       0.0486946861f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator;
   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Cos' */
   Control_System_B.Cos_2 = cosf(Control_System_B.Integrator_1);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Sin' */
   Control_System_B.Sin_2 = sinf(Control_System_B.Integrator_1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->d' */
   Control_System_B.abc_d_2 = 0.666666687f *
```

```
                              (((Control_System_B.ADC[8] *
                                 Control_System_B.Cos_2) +
                                (Control_System_B.ADC[9] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_2) +
                                  (0.866025388f *
      Control_System_B.Sin_2)))) +
                               (Control_System_B.ADC[10] *
                                (((-0.5f) *
                                  Control_System_B.Cos_2) -
   (0.866025388f * Control_System_B.Sin_2))));

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Cos' */
   Control_System_B.Cos_3 = cosf(Control_System_B.Integrator_1);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Sin' */
   Control_System_B.Sin_3 = sinf(Control_System_B.Integrator_1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->d' */
   Control_System_B.abc_d_3 = 0.666666687f *
                              (((Control_System_B.ADC[0] *
                                 Control_System_B.Cos_3) +
                                (Control_System_B.ADC[1] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_3) +
                                  (0.866025388f *
      Control_System_B.Sin_3)))) +
                               (Control_System_B.ADC[2] *
                                (((-0.5f) *
                                  Control_System_B.Cos_3) -
   (0.866025388f * Control_System_B.Sin_3))));

   /* Zero-Order Hold : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum1'
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Sum1'
    */
   Control_System_B.Zero_OrderHold_1 =
      (Control_System_B.Saturation_1 +
       Control_System_B.abc_d_2) - Control_System_B.abc_d_3;

   /* Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_1_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_1_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_1 = 0.f;
   }
```

```
   else if (Control_System_X.DiscreteIntegrator_1_first)
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_1_i2_prevU +
          (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controller/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_2 =
      (Control_System_B.Zero_OrderHold_1 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_1;
   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->q' */
   Control_System_B.abc_q_1 = 0.666666687f *
                              ((((-Control_System_B.ADC[0]) *
                                 Control_System_B.Sin_3) +
                                (Control_System_B.ADC[1] *
                                 ((0.5f *
                                   Control_System_B.Sin_3) +
                                  (0.866025388f *
      Control_System_B.Cos_3)))) +
                               (Control_System_B.ADC[2] *
                                ((0.5f *
                                  Control_System_B.Sin_3) -
   (0.866025388f * Control_System_B.Cos_3))));

   /* Gain : 'Control System/Current Controller/Gain' */
   Control_System_B.Gain_1 = 0.5f*Control_System_B.ADC[6];

   /* Product : 'Control System/Current Controller/Product2'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum2'
    *  Sum : 'Control System/Current Controller/Sum7'
    *  Gain : 'Control System/Current Controller/Gain2'
    */
   Control_System_B.Product2 =
      (Control_System_B.abc_d_1 +
       (Control_System_B.Saturation_2 -
        (0.314159265f*Control_System_B.abc_q_1))) / Control_System_B.Gain_1;

   /* Trigonometric Function : 'Control System/Current Controller/RRF->3ph/Cos' */
   Control_System_B.Cos_4 = cosf(Control_System_B.Integrator_1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->q' */
   Control_System_B.abc_q_2 = 0.666666687f *
                              ((((-Control_System_B.ADC[8]) *
                                 Control_System_B.Sin_2) +
                                (Control_System_B.ADC[9] *
                                 ((0.5f *
                                   Control_System_B.Sin_2) +
                                  (0.866025388f *
      Control_System_B.Cos_2)))) +
```

```
                              (Control_System_B.ADC[10] *
                               ((0.5f *
                                 Control_System_B.Sin_2) -
   (0.866025388f * Control_System_B.Cos_2))));

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF4/abc->q' */
   Control_System_B.abc_q_3 = 0.666666687f *
                             ((((-Control_System_B.ADC[3]) *
                                Control_System_B.Sin_1) +
                               (Control_System_B.ADC[4] *
                                ((0.5f *
                                  Control_System_B.Sin_1) +
                                 (0.866025388f *
      Control_System_B.Cos_1)))) +
                              (Control_System_B.ADC[5] *
                               ((0.5f *
                                 Control_System_B.Sin_1) -
   (0.866025388f * Control_System_B.Cos_1))));

   /* Zero-Order Hold : 'Control System/Droop Control/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Sum2'
    *  Constant : 'Control System/Droop Control/Constant'
    */
   Control_System_B.Zero_OrderHold_2 = 0.f - Control_System_B.abc_q_3;

   /* Discrete Integrator : 'Control System/Droop Control/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Reset select/None/Constant'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Init select/internal/Constant'
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum2'
    *  Product : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_2_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_2_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_2 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_2_first)
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_2_i2_prevU +
          (Control_System_B.Zero_OrderHold_2 * 2.37117246f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
```

```
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum'
    *  Product : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_3 =
      (Control_System_B.Zero_OrderHold_2 *
       0.0486946861f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_2;
   /* Zero-Order Hold : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum3'
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Sum3'
    */
   Control_System_B.Zero_OrderHold_3 = -Control_System_B.abc_q_1 +
                                       (Control_System_B.abc_q_2 +
                                        Control_System_B.Saturation_3);

   /* Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_3_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_3_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_3 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_3_first)
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_3_i2_prevU +
          (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controller/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant'
```

```
     *  Subsystem : 'Control System'
     */
    Control_System_B.Saturation_4 =
       (Control_System_B.Zero_OrderHold_3 *
        3.14159265f) + Control_System_UNCONNECTED +
       Control_System_B.DiscreteIntegrator_3;
    /* Product : 'Control System/Current Controller/Product3'
     * incorporates
     *  Sum : 'Control System/Current Controller/Sum4'
     *  Sum : 'Control System/Current Controller/Sum6'
     *  Gain : 'Control System/Current Controller/Gain1'
     */
    Control_System_B.Product3 = 1.f / Control_System_B.Gain_1 *
                                (((0.314159265f*
                                   Control_System_B.abc_d_3) +
                                  Control_System_B.Saturation_4) +
                                 Control_System_B.abc_q_3);

    /* Trigonometric Function : 'Control System/Current Controller/RRF->3ph/Sin' */
    Control_System_B.Sin_4 = sinf(Control_System_B.Integrator_1);

    /* Function : 'Control System/Current Controller/RRF->3ph/dq->a' */
    Control_System_B.dq_a =
       (Control_System_B.Product2 *
        Control_System_B.Cos_4) -
       (Control_System_B.Product3 * Control_System_B.Sin_4);

    /* Function : 'Control System/Current Controller/RRF->3ph/dq->b' */
    Control_System_B.dq_b =
       (Control_System_B.Product2 *
        (((-0.5f) *
          Control_System_B.Cos_4) +
         (0.866025388f *
       Control_System_B.Sin_4))) +
       (Control_System_B.Product3 *
        ((0.5f *
          Control_System_B.Sin_4) + (0.866025388f * Control_System_B.Cos_4)));

    /* PWM  : 'Control System/PWM a' */
    {
       PLXHAL_PWM_setDuty(0, 0.5f + (0.5f*Control_System_B.dq_a));
    }
    /* PWM  : 'Control System/PWM b' */
    {
       PLXHAL_PWM_setDuty(1, 0.5f + (0.5f*Control_System_B.dq_b));
    }
    /* PWM  : 'Control System/PWM c' */
    {
       PLXHAL_PWM_setDuty(2, 0.5f +
                          (0.5f*
                           (-Control_System_B.dq_b - Control_System_B.dq_a)));
    }
    /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/beta' */
    Control_System_B.beta = 0.577350259f *
                            (Control_System_B.ADC[12] -
                             Control_System_B.ADC[13]);

    /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction1' */
    Control_System_B.TrigonometricFunction1 =
       cosf(Control_System_B.Integrator);

    /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/alpha' */
    Control_System_B.alpha = 0.333333343f *
                             (((2.f *
                                Control_System_B.ADC[11]) -
```

```
                              Control_System_B.ADC[12]) -
                             Control_System_B.ADC[13]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction' */
   Control_System_B.TrigonometricFunction = sinf(Control_System_B.Integrator);

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_1[0] = 222.144147f*
                                       Control_System_X.TransferFcn_1[0];
   Control_System_B.TransferFcn_1[1] = 222.144147f*
                                       Control_System_X.TransferFcn_1[1];

   /* Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain1'
    * incorporates
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Product'
    */
   Control_System_B.Gain1 = 2.f*
                            (Control_System_B.TrigonometricFunction *
                             Control_System_B.TrigonometricFunction1);

   /* Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Sum'
    * incorporates
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Constant'
    *  Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain'
    *  Math Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Math'
    */
   Control_System_B.Sum = (-1.f) +
                          (2.f*
                           (Control_System_B.TrigonometricFunction1 *
                            Control_System_B.TrigonometricFunction1));

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn1'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->q'
    */
   Control_System_B.Fcn1 =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) +
       (Control_System_B.TransferFcn_1[0] *
        Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn_1[1] * Control_System_B.Sum);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->d'
    */
   Control_System_B.Fcn =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1[0] *
        Control_System_B.Sum)) -
      (Control_System_B.TransferFcn_1[1] * Control_System_B.Gain1);
```

```
   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   {
      float u, y;
      y =
         sqrtf(powf(Control_System_B.Fcn,2.f) + powf(Control_System_B.Fcn1,
                                                     2.f));
      u = 0.0001f;
      if (u > y)
         y = u;
      Control_System_B.Normalize = y;
   }

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   Control_System_B.Normalize_i1 = Control_System_B.Fcn1 /
                                   Control_System_B.Normalize;

   /* Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Reset select/None/Constant' */
   Control_System_B.Constant = 0.f;

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   if (Control_System_X.Integrator1_i1_first ||
       (!Control_System_X.Integrator1_i2_prevReset &&
        Control_System_B.Constant))
   {
      Control_System_X.Integrator1_x = 314.159265f;
   }
   Control_System_B.Integrator1 = Control_System_X.Integrator1_x;

   /* Saturation : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Saturation/internal/Saturation
Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product3'
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum1'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Kd Integrator/Kd = 0/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant3'
    */
   Control_System_B.Saturation_5 =
      (Control_System_B.Normalize_i1 *
       314.159265f) +
      (((Control_System_B.Normalize_i1 *
         0.f) - 0.f) * 0.f) + Control_System_B.Integrator1;
   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_2 = 222.144147f*
                                    Control_System_X.TransferFcn_2;
```

```
   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_3 = 222.144147f*
                                    Control_System_X.TransferFcn_3;
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Update for Discrete Integrator : 'Control System/Droop Control/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum2'
    *  Product : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController/Reset select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_first = 0;
   Control_System_X.DiscreteIntegrator_i1_x =
      Control_System_B.DiscreteIntegrator;
   Control_System_X.DiscreteIntegrator_i2_prevU =
      (Control_System_B.Zero_OrderHold * 2.37117246f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_1_first = 0;
   Control_System_X.DiscreteIntegrator_1_i1_x =
      Control_System_B.DiscreteIntegrator_1;
   Control_System_X.DiscreteIntegrator_1_i2_prevU =
      (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_1_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Droop Control/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum2'
    *  Product : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant1'
```

```
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Droop Control/Output-Voltage Controll/Discrete
PID\nController1/Reset select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_2_first = 0;
   Control_System_X.DiscreteIntegrator_2_i1_x =
      Control_System_B.DiscreteIntegrator_2;
   Control_System_X.DiscreteIntegrator_2_i2_prevU =
      (Control_System_B.Zero_OrderHold_2 * 2.37117246f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_2_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_3_first = 0;
   Control_System_X.DiscreteIntegrator_3_i1_x =
      Control_System_B.DiscreteIntegrator_3;
   Control_System_X.DiscreteIntegrator_3_i2_prevU =
      (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_3_i3_prevReset = !!(0.f);

   /* Update for PWM  : 'Control System/PWM a' */
   PLXHAL_PWM_enableAllOutputs();

   /* Update for Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_i1_first = 0;
   Control_System_X.Integrator1_i2_prevReset = !!(Control_System_B.Constant);

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Integrator' */
   Control_System_deriv[5] = Control_System_B.Saturation_5;

   /* Derivatives for Transfer Function : 'Control System/Droop Control/Transfer Fcn'
    * incorporates
    *  Product : 'Control System/Droop Control/Power \nCalcutation/Product2'
    *  Constant : 'Control System/Droop Control/Power \nCalcutation/Constant1'
    */
   Control_System_deriv[7] = Control_System_X.TransferFcn[1];
   Control_System_deriv[8] = Control_System_B.abc_d_2 * 1.5f *
                             Control_System_B.abc_d_1-98696.044f*
                             Control_System_X.TransferFcn[0]-
                             444.221201f*Control_System_X.TransferFcn[1];

   /* Derivatives for Transfer Function : 'Control System/Droop Control/Transfer
Fcn1'
    * incorporates
    *  Product : 'Control System/Droop Control/Power \nCalcutation/Product3'
    *  Gain : 'Control System/Droop Control/Power \nCalcutation/Gain'
    *  Constant : 'Control System/Droop Control/Power \nCalcutation/Constant1'
```

```
    */
   Control_System_deriv[9] = Control_System_X.TransferFcn1[1];
   Control_System_deriv[10] = Control_System_B.abc_q_2 *
                              (-1.f*
                               Control_System_B.abc_d_1) * 1.5f-98696.044f*
                              Control_System_X.TransferFcn1[0]-444.221201f*
                              Control_System_X.TransferFcn1[1];

   /* Derivatives for Integrator : 'Control System/Droop Control/Integrator' */
   Control_System_deriv[11] = Control_System_B.Saturation1;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->d'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn1'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->q'
    */
   Control_System_deriv[2] =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_2 *
        Control_System_B.Sum)) +
      (Control_System_B.TransferFcn_3 *
       Control_System_B.Gain1)-222.144147f*Control_System_X.TransferFcn_1[0];
   Control_System_deriv[3] =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_2 *
        Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn_3 *
       Control_System_B.Sum)-222.144147f*Control_System_X.TransferFcn_1[1];

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum3'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   Control_System_deriv[0] =
      (Control_System_B.Normalize_i1 * 56982.1876f * 1.f) + 0.f;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_deriv[1] = Control_System_B.Fcn-222.144147f*
                             Control_System_X.TransferFcn_2;
```

```
   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_deriv[4] = Control_System_B.Fcn1-222.144147f*
                             Control_System_X.TransferFcn_3;

   /* Update continuous states */
   Control_System_X.Integrator_x += 0.0001f*Control_System_deriv[5];
   Control_System_X.TransferFcn[0] += 0.0001f*Control_System_deriv[7];
   Control_System_X.TransferFcn[1] += 0.0001f*Control_System_deriv[8];
   Control_System_X.TransferFcn1[0] += 0.0001f*Control_System_deriv[9];
   Control_System_X.TransferFcn1[1] += 0.0001f*Control_System_deriv[10];
   Control_System_X.Integrator_1_x += 0.0001f*Control_System_deriv[11];
   Control_System_X.TransferFcn_1[0] += 0.0001f*Control_System_deriv[2];
   Control_System_X.TransferFcn_1[1] += 0.0001f*Control_System_deriv[3];
   Control_System_X.Integrator1_x += 0.0001f*Control_System_deriv[0];
   Control_System_X.TransferFcn_2 += 0.0001f*Control_System_deriv[1];
   Control_System_X.TransferFcn_3 += 0.0001f*Control_System_deriv[4];
}

void Control_System_terminate(){}
```

A.2.5.- VSG

```
/*
 * Implementation file for: VSG_HIL/Control System
 * Generated with         : PLECS 4.7.2
 *                          TI2833x 1.5.2
 * Generated on           : 8 Jun 2023 17:01:22
 */
#include "Control_System.h"
#ifndef PLECS_HEADER_Control_System_h_
#error The wrong header file "Control_System.h" was included. Please check
#error your include path to see whether this file name conflicts with the
#error name of another header file.
#endif /* PLECS_HEADER_Control_System_h_ */
#if defined(__GNUC__) && (__GNUC__ > 4)
#   define _ALIGNMENT 16
#   define _RESTRICT __restrict
#   define _ALIGN __attribute__((aligned(_ALIGNMENT)))
#   if defined(__clang__)
#      if __has_builtin(__builtin_assume_aligned)
#         define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#      else
#         define _ASSUME_ALIGNED(a) a
#      endif
#   else
#      define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#   endif
#else
#   ifndef _RESTRICT
#      define _RESTRICT
#   endif
#   ifndef _ALIGN
#      define _ALIGN
#   endif
#   ifndef _ASSUME_ALIGNED
#      define _ASSUME_ALIGNED(a) a
#   endif
#endif
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>
#include "plx_hal.h"
#define PLECSRunTimeError(msg) Control_System_errorStatus = msg
typedef struct
{
   const size_t numX, numY, numZ;
   const float x0, y0, z0;
   const float dXInv, dYInv, dZInv;
   const float *xVec, *yVec, *zVec, *data;
} plecsLUT;
#define DATA2D(x,y) lut->data[(xIdx+(x))*lut->numY+yIdx+(y)]
#define DATA3D(x,y, \
               z) lut->data[((xIdx+ \
                              (x))*lut->numY+(yIdx+(y)))*lut->numZ+zIdx+(z)]
static size_t calcIndex(float x0, float dXInv, size_t numX, float x,
                        float* fraction)
{
   size_t idx = 0;
   float div = (x-x0)*dXInv;
   if (div > 0)
   {
      idx = div;
      if (idx > numX-2)
         idx = numX-2;
   }
   *fraction = div-idx;
   return idx;
}
static float plecsinterp(float frac, float val0, float val1)
```

```
{
   return (val0 == val1) ? val0 : (1-frac)*val0 + frac*val1;
}
static float plecslu1deven(plecsLUT *lut, float x)
{
   float xFrac = 0.f;
   size_t xIdx = calcIndex(lut->x0, lut->dXInv, lut->numX, x, &xFrac);
   return plecsinterp(xFrac, lut->data[xIdx], lut->data[xIdx+1]);
}
static const float Control_System_UNCONNECTED = 0;
static float Control_System_deriv[14] _ALIGN;
static const float Control_System_lut_0_data[] = {
   325.269,292.742
};
static plecsLUT Control_System_lut[1] = {
   {2, 0, 0, 0.f, 0.f, 0.f, 0.01f, 0.f, 0.f, 0, 0, 0,
    Control_System_lut_0_data}
};
static uint32_t Control_System_tickLo;
static int32_t Control_System_tickHi;
Control_System_BlockOutputs Control_System_B;
#if defined(EXTERNAL_MODE) && EXTERNAL_MODE
const float * const Control_System_ExtModeSignals[] = {
   &Control_System_B.abc_d,
   &Control_System_B.abc_q,
   &Control_System_B.Integrator,
   &Control_System_UNCONNECTED,
   &Control_System_UNCONNECTED,
   &Control_System_B.TransferFcn1,
   &Control_System_B.Saturation,
   &Control_System_B.Sum2
};
#endif /* defined(EXTERNAL_MODE) */
Control_System_ModelStates Control_System_X _ALIGN;
const char * Control_System_errorStatus;
const float Control_System_sampleTime = 0.0001f;
const char * const Control_System_checksum =
   "298f885a6253c7120887c6564fa05bd70be77954";
/* Target declarations */
extern void Control_System_initHal();

void Control_System_initialize(float time)
{
   float remainder;
   Control_System_errorStatus = NULL;
   Control_System_tickHi =
      floor(time/(4294967296.0*Control_System_sampleTime));
   remainder = time - Control_System_tickHi*4294967296.0*
               Control_System_sampleTime;
   Control_System_tickLo = floor(remainder/Control_System_sampleTime + .5);
   remainder -= Control_System_tickLo*Control_System_sampleTime;
   if (fabsf(remainder) > 1e-6*fabsf(time))
   {
      Control_System_errorStatus =
         "Start time must be an integer multiple of the base sample time.";
   }

   /* Target pre-initialization */
   Control_System_initHal();


   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Integrator' */
   Control_System_X.Integrator_x = 0.f;

   /* Initialization for Transfer Function : 'Control System/VSG Control/Transfer
Fcn1' */
   Control_System_X.TransferFcn1[0] = 0.f;
```

```
   Control_System_X.TransferFcn1[1] = 0.f;

   /* Initialization for Integrator : 'Control System/VSG Control/Integrator2' */
   Control_System_X.Integrator2_x = 0.f;

   /* Initialization for Integrator : 'Control System/VSG Control/Integrator1' */
   Control_System_X.Integrator1_x = 0.f;

   /* Initialization for Discrete Integrator : 'Control System/VSG Control/Q-
V/Output-Voltage Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_first = -1;
   Control_System_X.DiscreteIntegrator_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controller/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_1_first = -1;
   Control_System_X.DiscreteIntegrator_1_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/VSG Control/Q-
V/Output-Voltage Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_2_first = -1;
   Control_System_X.DiscreteIntegrator_2_i1_x = 0;

   /* Initialization for Discrete Integrator : 'Control System/Current
Controller/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator' */
   Control_System_X.DiscreteIntegrator_3_first = -1;
   Control_System_X.DiscreteIntegrator_3_i1_x = 0;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn[0] = 0.f;
   Control_System_X.TransferFcn[1] = 0.f;

   /* Initialization for Integrator : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Continuous PID\nController/Continous Time/Ki
Integrator/Ki ~= 0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_1_x = 0;
   Control_System_X.Integrator1_1_i1_first = 1;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_1 = 0.f;

   /* Initialization for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_X.TransferFcn_2 = -0.00450158158f;

   /* Initialization for Transfer Function : 'Control System/VSG Control/Transfer
Fcn' */
   Control_System_X.TransferFcn_3[0] = 0.f;
   Control_System_X.TransferFcn_3[1] = 0.f;

   /* Initialization for Transfer Function : 'Control System/VSG Control/F' */
   Control_System_X.F = 0.f;
}

void Control_System_step(void)
{
   if (Control_System_errorStatus)
   {
      return;
```

```
   }

   /* ADC : 'Control System/ADC' */
   Control_System_B.ADC[0] = PLXHAL_ADC_getIn(0, 0);
   Control_System_B.ADC[1] = PLXHAL_ADC_getIn(0, 1);
   Control_System_B.ADC[2] = PLXHAL_ADC_getIn(0, 2);
   Control_System_B.ADC[3] = PLXHAL_ADC_getIn(0, 3);
   Control_System_B.ADC[4] = PLXHAL_ADC_getIn(0, 4);
   Control_System_B.ADC[5] = PLXHAL_ADC_getIn(0, 5);
   Control_System_B.ADC[6] = PLXHAL_ADC_getIn(0, 6);
   Control_System_B.ADC[7] = PLXHAL_ADC_getIn(0, 7);
   Control_System_B.ADC[8] = PLXHAL_ADC_getIn(0, 8);
   Control_System_B.ADC[9] = PLXHAL_ADC_getIn(0, 9);
   Control_System_B.ADC[10] = PLXHAL_ADC_getIn(0, 10);
   Control_System_B.ADC[11] = PLXHAL_ADC_getIn(0, 11);
   Control_System_B.ADC[12] = PLXHAL_ADC_getIn(0, 12);
   Control_System_B.ADC[13] = PLXHAL_ADC_getIn(0, 13);

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Integrator' */
   if (Control_System_X.Integrator_x > 6.28318531f ||
       Control_System_X.Integrator_x < 0.f)
   {
      float span = 6.28318531f - (0.f);
      Control_System_X.Integrator_x -= 0.f;
      Control_System_X.Integrator_x = Control_System_X.Integrator_x - span*
                                      floorf(
                                             Control_System_X.Integrator_x/
                                             span) + (0.f);
   }
   Control_System_B.Integrator = Control_System_X.Integrator_x;

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Cos' */
   Control_System_B.Cos = cosf(Control_System_B.Integrator);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF/Sin' */
   Control_System_B.Sin = sinf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->d' */
   Control_System_B.abc_d = 0.666666687f *
                            (((Control_System_B.ADC[11] *
                               Control_System_B.Cos) +
                              (Control_System_B.ADC[12] *
                               (((-0.5f) *
                                 Control_System_B.Cos) +
                                (0.866025388f *
      Control_System_B.Sin)))) +
                             (Control_System_B.ADC[13] *
                              (((-0.5f) *
                                Control_System_B.Cos) -
   (0.866025388f * Control_System_B.Sin))));

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF/abc->q' */
   Control_System_B.abc_q = 0.666666687f *
                            ((((-Control_System_B.ADC[11]) *
                               Control_System_B.Sin) +
                              (Control_System_B.ADC[12] *
                               ((0.5f *
                                 Control_System_B.Sin) +
                                (0.866025388f *
      Control_System_B.Cos)))) +
                             (Control_System_B.ADC[13] *
                              ((0.5f *
                                Control_System_B.Sin) -
   (0.866025388f * Control_System_B.Cos))));
```

```
   /* Transfer Function : 'Control System/VSG Control/Transfer Fcn1' */
   Control_System_B.TransferFcn1 = 3947.84176f*
                                   Control_System_X.TransferFcn1[0];

   /* Saturation : 'Control System/VSG Control/Q-V/Saturation'
    * incorporates
    *  1D Look-Up Table : 'Control System/VSG Control/Q-V/1D-Table'
    */
   Control_System_B.Saturation = plecslu1deven(&Control_System_lut[0],
                                               Control_System_B.TransferFcn1);
   if (Control_System_B.Saturation > 357.796031f)
   {
      Control_System_B.Saturation = 357.796031f;
   }
   else if (Control_System_B.Saturation < 292.742207f)
   {
      Control_System_B.Saturation = 292.742207f;
   }

   /* Integrator : 'Control System/VSG Control/Integrator2' */
   Control_System_B.Integrator2 = Control_System_X.Integrator2_x;

   /* Gain : 'Control System/VSG Control/Gain9' */
   Control_System_B.Gain9 = 314.159265f*Control_System_B.Integrator2;

   /* Sum : 'Control System/VSG Control/Sum2'
    * incorporates
    *  Constant : 'Control System/VSG Control/wn2'
    */
   Control_System_B.Sum2 = -Control_System_B.Gain9 + 314.159265f;

   /* Integrator : 'Control System/VSG Control/Integrator1' */
   Control_System_B.Integrator1 = Control_System_X.Integrator1_x;

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF4/Cos' */
   Control_System_B.Cos_1 = cosf(Control_System_B.Integrator1);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF4/Sin' */
   Control_System_B.Sin_1 = sinf(Control_System_B.Integrator1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF4/abc->d' */
   Control_System_B.abc_d_1 = 0.666666687f *
                              (((Control_System_B.ADC[3] *
                                 Control_System_B.Cos_1) +
                                (Control_System_B.ADC[4] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_1) +
                                  (0.866025388f *
      Control_System_B.Sin_1)))) +
                               (Control_System_B.ADC[5] *
                                (((-0.5f) *
                                  Control_System_B.Cos_1) -
   (0.866025388f * Control_System_B.Sin_1))));

   /* Zero-Order Hold : 'Control System/VSG Control/Q-V/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Sum'
    */
   Control_System_B.Zero_OrderHold = Control_System_B.Saturation -
                                     Control_System_B.abc_d_1;

   /* Discrete Integrator : 'Control System/VSG Control/Q-V/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
```

```
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Reset select/None/Constant'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Init select/internal/Constant'
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum2'
    *  Product : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_first)
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator =
         Control_System_X.DiscreteIntegrator_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_i2_prevU +
          (Control_System_B.Zero_OrderHold * 2.37117246f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum'
    *  Product : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_1 =
      (Control_System_B.Zero_OrderHold *
       0.0389557489f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator;
   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Cos' */
   Control_System_B.Cos_2 = cosf(Control_System_B.Integrator1);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF3/Sin' */
   Control_System_B.Sin_2 = sinf(Control_System_B.Integrator1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->d' */
   Control_System_B.abc_d_2 = 0.666666687f *
                              (((Control_System_B.ADC[8] *
                                 Control_System_B.Cos_2) +
                                (Control_System_B.ADC[9] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_2) +
                                  (0.866025388f *
      Control_System_B.Sin_2)))) +
                               (Control_System_B.ADC[10] *
                                (((-0.5f) *
                                  Control_System_B.Cos_2) -
```

```
   (0.866025388f * Control_System_B.Sin_2))));

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Cos' */
   Control_System_B.Cos_3 = cosf(Control_System_B.Integrator1);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/3ph-
>RRF2/Sin' */
   Control_System_B.Sin_3 = sinf(Control_System_B.Integrator1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->d' */
   Control_System_B.abc_d_3 = 0.666666687f *
                              (((Control_System_B.ADC[0] *
                                 Control_System_B.Cos_3) +
                                (Control_System_B.ADC[1] *
                                 (((-0.5f) *
                                   Control_System_B.Cos_3) +
                                  (0.866025388f *
      Control_System_B.Sin_3)))) +
                               (Control_System_B.ADC[2] *
                                (((-0.5f) *
                                  Control_System_B.Cos_3) -
   (0.866025388f * Control_System_B.Sin_3))));

   /* Zero-Order Hold : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum1'
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Sum1'
    */
   Control_System_B.Zero_OrderHold_1 =
      (Control_System_B.Saturation_1 +
       Control_System_B.abc_d_2) - Control_System_B.abc_d_3;

   /* Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_1_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_1_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_1 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_1_first)
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_1 =
         Control_System_X.DiscreteIntegrator_1_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_1_i2_prevU +
```

```
          (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controller/Discrete
PID\nController/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_2 =
      (Control_System_B.Zero_OrderHold_1 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_1;
   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF2/abc->q' */
   Control_System_B.abc_q_1 = 0.666666687f *
                              ((((-Control_System_B.ADC[0]) *
                                 Control_System_B.Sin_3) +
                                (Control_System_B.ADC[1] *
                                 ((0.5f *
                                   Control_System_B.Sin_3) +
                                  (0.866025388f *
      Control_System_B.Cos_3)))) +
                               (Control_System_B.ADC[2] *
                                ((0.5f *
                                  Control_System_B.Sin_3) -
   (0.866025388f * Control_System_B.Cos_3))));

   /* Gain : 'Control System/Current Controller/Gain' */
   Control_System_B.Gain = 0.5f*Control_System_B.ADC[6];

   /* Product : 'Control System/Current Controller/Product2'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum2'
    *  Sum : 'Control System/Current Controller/Sum7'
    *  Gain : 'Control System/Current Controller/Gain2'
    */
   Control_System_B.Product2 =
      (Control_System_B.abc_d_1 +
       (Control_System_B.Saturation_2 -
        (0.314159265f*Control_System_B.abc_q_1))) / Control_System_B.Gain;

   /* Trigonometric Function : 'Control System/Current Controller/RRF->3ph/Cos' */
   Control_System_B.Cos_4 = cosf(Control_System_B.Integrator1);

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF3/abc->q' */
   Control_System_B.abc_q_2 = 0.666666687f *
                              ((((-Control_System_B.ADC[8]) *
                                 Control_System_B.Sin_2) +
                                (Control_System_B.ADC[9] *
                                 ((0.5f *
                                   Control_System_B.Sin_2) +
                                  (0.866025388f *
      Control_System_B.Cos_2)))) +
                               (Control_System_B.ADC[10] *
                                ((0.5f *
                                  Control_System_B.Sin_2) -
   (0.866025388f * Control_System_B.Cos_2))));

   /* Function : 'Control System/Bloque de\nTransformaciones/3ph->RRF4/abc->q' */
   Control_System_B.abc_q_3 = 0.666666687f *
                              ((((-Control_System_B.ADC[3]) *
                                 Control_System_B.Sin_1) +
                                (Control_System_B.ADC[4] *
```

```
                                ((0.5f *
                                  Control_System_B.Sin_1) +
                                 (0.866025388f *
      Control_System_B.Cos_1)))) +
                               (Control_System_B.ADC[5] *
                                ((0.5f *
                                  Control_System_B.Sin_1) -
   (0.866025388f * Control_System_B.Cos_1))));

   /* Zero-Order Hold : 'Control System/VSG Control/Q-V/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Sum2'
    *  Constant : 'Control System/VSG Control/Q-V/Constant'
    */
   Control_System_B.Zero_OrderHold_2 = 0.f - Control_System_B.abc_q_3;

   /* Discrete Integrator : 'Control System/VSG Control/Q-V/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Reset select/None/Constant'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Init select/internal/Constant'
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum2'
    *  Product : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_2_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_2_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_2 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_2_first)
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_2 =
         Control_System_X.DiscreteIntegrator_2_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_2_i2_prevU +
          (Control_System_B.Zero_OrderHold_2 * 2.37117246f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum'
    *  Product : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_3 =
      (Control_System_B.Zero_OrderHold_2 *
       0.0389557489f) + Control_System_UNCONNECTED +
```

```
      Control_System_B.DiscreteIntegrator_2;
   /* Zero-Order Hold : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Zero-Order\nHold'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum3'
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Sum3'
    */
   Control_System_B.Zero_OrderHold_3 = -Control_System_B.abc_q_1 +
                                       (Control_System_B.abc_q_2 +
                                        Control_System_B.Saturation_3);

   /* Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Reset
select/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Init
select/internal/Constant'
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    */
   if (Control_System_X.DiscreteIntegrator_3_first < 0 ||
       (!Control_System_X.DiscreteIntegrator_3_i3_prevReset && 0.f))
   {
      Control_System_B.DiscreteIntegrator_3 = 0.f;
   }
   else if (Control_System_X.DiscreteIntegrator_3_first)
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x;
   }
   else
   {
      Control_System_B.DiscreteIntegrator_3 =
         Control_System_X.DiscreteIntegrator_3_i1_x + 5e-05f*
         (Control_System_X.DiscreteIntegrator_3_i2_prevU +
          (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f);
   }

   /* Saturation : 'Control System/Current Controller/Discrete
PID\nController1/Saturation/internal/Saturation Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant'
    *  Subsystem : 'Control System'
    */
   Control_System_B.Saturation_4 =
      (Control_System_B.Zero_OrderHold_3 *
       3.14159265f) + Control_System_UNCONNECTED +
      Control_System_B.DiscreteIntegrator_3;
   /* Product : 'Control System/Current Controller/Product3'
    * incorporates
    *  Sum : 'Control System/Current Controller/Sum4'
    *  Sum : 'Control System/Current Controller/Sum6'
    *  Gain : 'Control System/Current Controller/Gain1'
```

```
    */
   Control_System_B.Product3 = 1.f / Control_System_B.Gain *
                               (((0.314159265f*
                                  Control_System_B.abc_d_3) +
                                 Control_System_B.Saturation_4) +
                                Control_System_B.abc_q_3);

   /* Trigonometric Function : 'Control System/Current Controller/RRF->3ph/Sin' */
   Control_System_B.Sin_4 = sinf(Control_System_B.Integrator1);

   /* Function : 'Control System/Current Controller/RRF->3ph/dq->a' */
   Control_System_B.dq_a =
      (Control_System_B.Product2 *
       Control_System_B.Cos_4) -
      (Control_System_B.Product3 * Control_System_B.Sin_4);

   /* Function : 'Control System/Current Controller/RRF->3ph/dq->b' */
   Control_System_B.dq_b =
      (Control_System_B.Product2 *
       (((-0.5f) *
         Control_System_B.Cos_4) +
        (0.866025388f *
      Control_System_B.Sin_4))) +
      (Control_System_B.Product3 *
       ((0.5f *
         Control_System_B.Sin_4) + (0.866025388f * Control_System_B.Cos_4)));

   /* PWM  : 'Control System/PWM a' */
   {
      PLXHAL_PWM_setDuty(0, 0.5f + (0.5f*Control_System_B.dq_a));
   }
   /* PWM  : 'Control System/PWM b' */
   {
      PLXHAL_PWM_setDuty(1, 0.5f + (0.5f*Control_System_B.dq_b));
   }
   /* PWM  : 'Control System/PWM c' */
   {
      PLXHAL_PWM_setDuty(2, 0.5f +
                         (0.5f*
                          (-Control_System_B.dq_b - Control_System_B.dq_a)));
   }
   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/beta' */
   Control_System_B.beta = 0.577350259f *
                           (Control_System_B.ADC[12] -
                            Control_System_B.ADC[13]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction1' */
   Control_System_B.TrigonometricFunction1 =
      cosf(Control_System_B.Integrator);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/3ph->SRF/alpha' */
   Control_System_B.alpha = 0.333333343f *
                            (((2.f *
                               Control_System_B.ADC[11]) -
                              Control_System_B.ADC[12]) -
                             Control_System_B.ADC[13]);

   /* Trigonometric Function : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Phase\ndetector/DSRF/sin\/cos/Trigonometric\nFunction' */
   Control_System_B.TrigonometricFunction = sinf(Control_System_B.Integrator);

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn[0] = 222.144147f*
                                     Control_System_X.TransferFcn[0];
```

```
   Control_System_B.TransferFcn[1] = 222.144147f*
                                     Control_System_X.TransferFcn[1];

   /* Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain1'
    * incorporates
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Product'
    */
   Control_System_B.Gain1 = 2.f*
                            (Control_System_B.TrigonometricFunction *
                             Control_System_B.TrigonometricFunction1);

   /* Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Sum'
    * incorporates
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Constant'
    *  Gain : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Gain'
    *  Math Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/sin\/cos/Math'
    */
   Control_System_B.Sum = (-1.f) +
                          (2.f*
                           (Control_System_B.TrigonometricFunction1 *
                            Control_System_B.TrigonometricFunction1));

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn1'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->q'
    */
   Control_System_B.Fcn1 =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) +
       (Control_System_B.TransferFcn[0] *
        Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn[1] * Control_System_B.Sum);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork+1/Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq+1/αβ->d'
    */
   Control_System_B.Fcn =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn[0] *
        Control_System_B.Sum)) -
      (Control_System_B.TransferFcn[1] * Control_System_B.Gain1);

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   {
      float u, y;
      y =
         sqrtf(powf(Control_System_B.Fcn,2.f) + powf(Control_System_B.Fcn1,
                                                     2.f));
      u = 0.0001f;
      if (u > y)
         y = u;
```

```
      Control_System_B.Normalize = y;
   }

   /* Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Normalize' */
   Control_System_B.Normalize_i1 = Control_System_B.Fcn1 /
                                   Control_System_B.Normalize;

   /* Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Reset select/None/Constant' */
   Control_System_B.Constant = 0.f;

   /* Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   if (Control_System_X.Integrator1_1_i1_first ||
       (!Control_System_X.Integrator1_1_i2_prevReset &&
        Control_System_B.Constant))
   {
      Control_System_X.Integrator1_1_x = 314.159265f;
   }
   Control_System_B.Integrator1_1 = Control_System_X.Integrator1_1_x;

   /* Saturation : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Saturation/internal/Saturation
Select/constant/Saturation'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product3'
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum1'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant2'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Kd Integrator/Kd = 0/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant3'
    */
   Control_System_B.Saturation_5 =
      (Control_System_B.Normalize_i1 *
       314.159265f) +
      (((Control_System_B.Normalize_i1 *
         0.f) - 0.f) * 0.f) + Control_System_B.Integrator1_1;
   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_1 = 222.144147f*
                                    Control_System_X.TransferFcn_1;

   /* Transfer Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer Fcn' */
   Control_System_B.TransferFcn_2 = 222.144147f*
                                    Control_System_X.TransferFcn_2;
   if (Control_System_errorStatus)
   {
      return;
   }

   /* Update for Discrete Integrator : 'Control System/VSG Control/Q-V/Output-Voltage
Controll/Discrete PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
```

```
    * incorporates
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Sum2'
    *  Product : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Discrete Time/Product4'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Anti-windup\nmethod/None/Constant1'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController/Reset select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_first = 0;
   Control_System_X.DiscreteIntegrator_i1_x =
      Control_System_B.DiscreteIntegrator;
   Control_System_X.DiscreteIntegrator_i2_prevU =
      (Control_System_B.Zero_OrderHold * 2.37117246f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete PID\nController/Discrete
Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_1_first = 0;
   Control_System_X.DiscreteIntegrator_1_i1_x =
      Control_System_B.DiscreteIntegrator_1;
   Control_System_X.DiscreteIntegrator_1_i2_prevU =
      (Control_System_B.Zero_OrderHold_1 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_1_i3_prevReset = !!(0.f);

   /* Update for Discrete Integrator : 'Control System/VSG Control/Q-V/Output-Voltage
Controll/Discrete PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Sum2'
    *  Product : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Source Select/internal/Constant1'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Anti-windup\nmethod/None/Constant1'
    *  Constant : 'Control System/VSG Control/Q-V/Output-Voltage Controll/Discrete
PID\nController1/Reset select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_2_first = 0;
   Control_System_X.DiscreteIntegrator_2_i1_x =
      Control_System_B.DiscreteIntegrator_2;
   Control_System_X.DiscreteIntegrator_2_i2_prevU =
      (Control_System_B.Zero_OrderHold_2 * 2.37117246f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_2_i3_prevReset = !!(0.f);
```

```
   /* Update for Discrete Integrator : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Ki Integrator/Ki ~=
0/Discrete\nIntegrator/edge_triggered/Discrete\nIntegrator'
    * incorporates
    *  Sum : 'Control System/Current Controller/Discrete PID\nController1/Discrete
Time/Sum2'
    *  Product : 'Control System/Current Controller/Discrete
PID\nController1/Discrete Time/Product4'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Source
Select/internal/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Anti-
windup\nmethod/None/Constant1'
    *  Constant : 'Control System/Current Controller/Discrete PID\nController1/Reset
select/None/Constant'
    */
   Control_System_X.DiscreteIntegrator_3_first = 0;
   Control_System_X.DiscreteIntegrator_3_i1_x =
      Control_System_B.DiscreteIntegrator_3;
   Control_System_X.DiscreteIntegrator_3_i2_prevU =
      (Control_System_B.Zero_OrderHold_3 * 31.4159265f * 1.f) + 0.f;
   Control_System_X.DiscreteIntegrator_3_i3_prevReset = !!(0.f);

   /* Update for PWM  : 'Control System/PWM a' */
   PLXHAL_PWM_enableAllOutputs();

   /* Update for Integrator : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1' */
   Control_System_X.Integrator1_1_i1_first = 0;
   Control_System_X.Integrator1_1_i2_prevReset =
      !!(Control_System_B.Constant);

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Integrator' */
   Control_System_deriv[5] = Control_System_B.Saturation_5;

   /* Derivatives for Transfer Function : 'Control System/VSG Control/Transfer Fcn1'
    * incorporates
    *  Product : 'Control System/VSG Control/Power \nCalcutation/Product3'
    *  Gain : 'Control System/VSG Control/Power \nCalcutation/Gain'
    *  Constant : 'Control System/VSG Control/Power \nCalcutation/Constant1'
    */
   Control_System_deriv[8] = Control_System_X.TransferFcn1[1];
   Control_System_deriv[9] = Control_System_B.abc_q_2 *
                             (-1.f*
                              Control_System_B.abc_d_1) * 1.5f-3947.84176f*
                             Control_System_X.TransferFcn1[0]-88.8442402f*
                             Control_System_X.TransferFcn1[1];

   /* Derivatives for Integrator : 'Control System/VSG Control/Integrator2'
    * incorporates
    *  Gain : 'Control System/VSG Control/Inertia'
    *  Sum : 'Control System/VSG Control/Sum6'
    *  Gain : 'Control System/VSG Control/1\/Pn5'
    *  Sum : 'Control System/VSG Control/Sum3'
    *  Sum : 'Control System/VSG Control/Sum7'
    *  Constant : 'Control System/VSG Control/Pn2'
    *  Transfer Function : 'Control System/VSG Control/Transfer Fcn'
    *  Gain : 'Control System/VSG Control/Kw'
    *  Gain : 'Control System/VSG Control/D'
    *  Sum : 'Control System/VSG Control/Sum5'
    *  Transfer Function : 'Control System/VSG Control/F'
    */
   Control_System_deriv[12] = 0.5f*
                              ((5e-05f*
```

```
                                ((20000.f -
                                  (3947.84176f*
                                   Control_System_X.TransferFcn_3[0])) -
                                 (-3819.71863f*
                                  Control_System_B.Sum2))) -
                                (5.f*
                                 (-(5.f*
   Control_System_X.F) + Control_System_B.Integrator2)));

   /* Derivatives for Integrator : 'Control System/VSG Control/Integrator1' */
   Control_System_deriv[11] = Control_System_B.Gain9;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF2/Continuous/Transfer
Fcn'
    * incorporates
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->d'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Decoupling\nnetwork-1/Fcn1'
    *  Function : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Phase\ndetector/DSRF/Tdq-1/αβ->q'
    */
   Control_System_deriv[2] =
      (((Control_System_B.alpha *
         Control_System_B.TrigonometricFunction1) -
        (Control_System_B.beta *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1 *
        Control_System_B.Sum)) +
      (Control_System_B.TransferFcn_2 *
       Control_System_B.Gain1)-222.144147f*Control_System_X.TransferFcn[0];
   Control_System_deriv[3] =
      (((Control_System_B.beta *
         Control_System_B.TrigonometricFunction1) +
        (Control_System_B.alpha *
         Control_System_B.TrigonometricFunction)) -
       (Control_System_B.TransferFcn_1 *
        Control_System_B.Gain1)) -
      (Control_System_B.TransferFcn_2 *
       Control_System_B.Sum)-222.144147f*Control_System_X.TransferFcn[1];

   /* Derivatives for Integrator : 'Control System/Bloque de\nTransformaciones/Three-
Phase PLL/Continuous PID\nController/Continous Time/Ki Integrator/Ki ~=
0/Integrator/edge_triggered/Integrator1'
    * incorporates
    *  Sum : 'Control System/Bloque de\nTransformaciones/Three-Phase PLL/Continuous
PID\nController/Continous Time/Sum3'
    *  Product : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Continous Time/Product1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Source Select/internal/Constant1'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant'
    *  Constant : 'Control System/Bloque de\nTransformaciones/Three-Phase
PLL/Continuous PID\nController/Anti-windup\nmethod/None/Constant1'
    */
   Control_System_deriv[0] =
      (Control_System_B.Normalize_i1 * 56982.1876f * 1.f) + 0.f;

   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF1/Continuous/Transfer
Fcn' */
   Control_System_deriv[1] = Control_System_B.Fcn-222.144147f*
                             Control_System_X.TransferFcn_1;
```

```
   /* Derivatives for Transfer Function : 'Control System/Bloque
de\nTransformaciones/Three-Phase PLL/Phase\ndetector/DSRF/LPF3/Continuous/Transfer
Fcn' */
   Control_System_deriv[4] = Control_System_B.Fcn1-222.144147f*
                             Control_System_X.TransferFcn_2;

   /* Derivatives for Transfer Function : 'Control System/VSG Control/Transfer Fcn'
    * incorporates
    *  Product : 'Control System/VSG Control/Power \nCalcutation/Product2'
    *  Constant : 'Control System/VSG Control/Power \nCalcutation/Constant1'
    */
   Control_System_deriv[6] = Control_System_X.TransferFcn_3[1];
   Control_System_deriv[7] = Control_System_B.abc_d_2 * 1.5f *
                             Control_System_B.abc_d_1-3947.84176f*
                             Control_System_X.TransferFcn_3[0]-
                             88.8442402f*Control_System_X.TransferFcn_3[1];

   /* Derivatives for Transfer Function : 'Control System/VSG Control/F' */
   Control_System_deriv[13] = Control_System_B.Integrator2-5.f*
                              Control_System_X.F;

   /* Update continuous states */
   Control_System_X.Integrator_x += 0.0001f*Control_System_deriv[5];
   Control_System_X.TransferFcn1[0] += 0.0001f*Control_System_deriv[8];
   Control_System_X.TransferFcn1[1] += 0.0001f*Control_System_deriv[9];
   Control_System_X.Integrator2_x += 0.0001f*Control_System_deriv[12];
   Control_System_X.Integrator1_x += 0.0001f*Control_System_deriv[11];
   Control_System_X.TransferFcn[0] += 0.0001f*Control_System_deriv[2];
   Control_System_X.TransferFcn[1] += 0.0001f*Control_System_deriv[3];
   Control_System_X.Integrator1_1_x += 0.0001f*Control_System_deriv[0];
   Control_System_X.TransferFcn_1 += 0.0001f*Control_System_deriv[1];
   Control_System_X.TransferFcn_2 += 0.0001f*Control_System_deriv[4];
   Control_System_X.TransferFcn_3[0] += 0.0001f*Control_System_deriv[6];
   Control_System_X.TransferFcn_3[1] += 0.0001f*Control_System_deriv[7];
   Control_System_X.F += 0.0001f*Control_System_deriv[13];
}

void Control_System_terminate()
{
}
```

A.2.6.- State Machine: Operating Mode Change

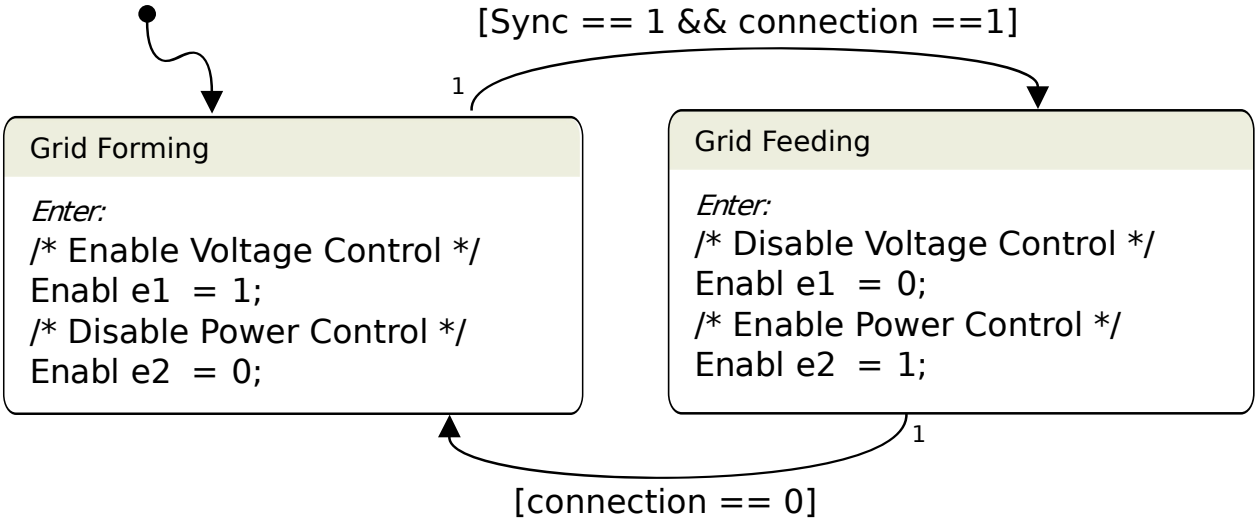

Figure A.2.- State machine to perform the change of the inverter control mode.

```
/*
 * State machine file for: Control System/State Machine
 * Generated with    : PLECS 4.7.2
 * Generated on      : 8 Jun 2023 17:52:45
 */

typedef float real_t;
#define REAL_MAX FLT_MAX
#define REAL_MIN FLT_MIN
#define REAL_EPSILON FLT_EPSILON
struct FSM_Struct
{
   int fsm_isMajorTimeStep;
   float fsm_currentTime;
   const float *fsm_internalConstants;
   const float ***fsm_inputs;
   float ***fsm_outputs;
   float *fsm_discStates;
   float *fsm_zCSignals;
   int *fsm_takenTransitions;
   float *fsm_nextSampleHit;
   float fsm_samplingFrequency;
   const char **fsm_errorStatus;
   const char **fsm_warningStatus;
};

enum FSM_State
{
   FSM_STATE_NONE,
   FSM_STATE_CONTROLTENSON,
   FSM_STATE_CONTROLPOTENCIA
};

enum FSM_Transition
{
   FSM_TRANSITION_NONE,
   FSM_TRANSITION_CONTROLTENSON_1,
   FSM_INITIAL_TRANSITION,
   FSM_TRANSITION_CONTROLPOTENCIA_1
};

#define FSM_MAX_NUM_TAKEN_TRANSITIONS 1

#define CurrentState fsm_struct->fsm_discStates[0]
#define TakenTransition(i) fsm_struct->fsm_takenTransitions[i]
#define IsMajorStep fsm_struct->fsm_isMajorTimeStep
#define CurrentTime fsm_struct->fsm_currentTime
#define SetErrorMessage(string) { *fsm_struct->fsm_errorStatus = (string); }
#define SetWarningMessage(string)

/* input variables */
#define Sync (*fsm_struct->fsm_inputs[0][0])
#define connection (*fsm_struct->fsm_inputs[1][0])

/* output variables */
#define Enable1 (*fsm_struct->fsm_outputs[0][0])
#define Enable2 (*fsm_struct->fsm_outputs[1][0])



static void fsm_state_ControlTenson_EnterAction(
                                                const struct FSM_Struct* fsm_struct)
{
   /* Habilitar el Control de Tension */
   Enable1 = 1;
   /* Deshabilitar el Control de Potencia */
   Enable2 = 0;
}
```

```
static void fsm_state_ControlPotencia_EnterAction(
                                                    const struct FSM_Struct*
fsm_struct)
{
   /* Habilitar el Control de Potencia */
   Enable1 = 0;
   /* Deshabilitar el Control de Tension */
   Enable2 = 1;
}



void Control_System_0_fsm_start(const struct FSM_Struct *fsm_struct)
{
   int fsm_i;
   CurrentState = FSM_STATE_NONE;
   for (fsm_i = 0; fsm_i < FSM_MAX_NUM_TAKEN_TRANSITIONS; fsm_i++)
   {
      TakenTransition(fsm_i) = FSM_TRANSITION_NONE;
   }
}

void Control_System_0_fsm_output(const struct FSM_Struct *fsm_struct)
{
   if (IsMajorStep)
   {
      int fsm_i;
      for (fsm_i = 0; fsm_i < FSM_MAX_NUM_TAKEN_TRANSITIONS; fsm_i++)
      {
         TakenTransition(fsm_i) = FSM_TRANSITION_NONE;
      }
      switch ((int)CurrentState)
      {
      case FSM_STATE_CONTROLTENSON:
         if (Sync == 1 && connection == 1)
         {
            TakenTransition(0) = FSM_TRANSITION_CONTROLTENSON_1;
            fsm_state_ControlPotencia_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_CONTROLPOTENCIA;
         }
         break;

      case FSM_STATE_CONTROLPOTENCIA:
         if (connection == 0)
         {
            TakenTransition(0) = FSM_TRANSITION_CONTROLPOTENCIA_1;
            fsm_state_ControlTenson_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_CONTROLTENSON;
         }
         break;

      default:
         TakenTransition(0) = FSM_INITIAL_TRANSITION;
         fsm_state_ControlTenson_EnterAction(fsm_struct);
         CurrentState = FSM_STATE_CONTROLTENSON;
         break;
      }
   }
}
```

A.3.- MICROGRID MODEL

A.3.1.- State Machine

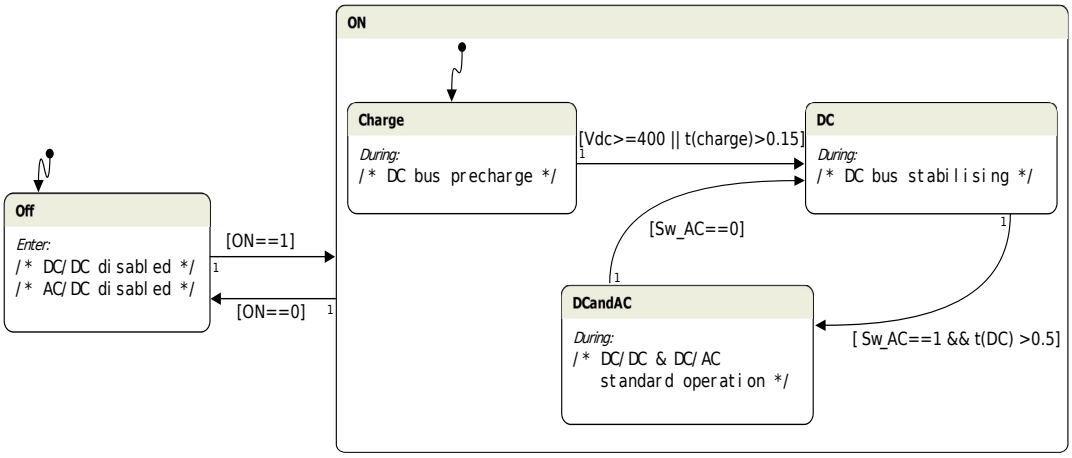

Figure A.3.- State machine to control the microgrid operation.

```
/*
 * State machine file for: Control System/State Machine
 * Generated with    : PLECS 4.7.2
 * Generated on      : 8 Jun 2023 18:17:51
 */

typedef float real_t;
#define REAL_MAX FLT_MAX
#define REAL_MIN FLT_MIN
#define REAL_EPSILON FLT_EPSILON
struct FSM_Struct
{
   int fsm_isMajorTimeStep;
   float fsm_currentTime;
   const float *fsm_internalConstants;
   const float ***fsm_inputs;
   float ***fsm_outputs;
   float *fsm_discStates;
   float *fsm_zCSignals;
   int *fsm_takenTransitions;
   float *fsm_nextSampleHit;
   float fsm_samplingFrequency;
   const char **fsm_errorStatus;
   const char **fsm_warningStatus;
};

enum FSM_State
{
   FSM_STATE_NONE,
   FSM_STATE_OFF,
   FSM_STATE_ON_CHARGE,
   FSM_STATE_ON_DC,
   FSM_STATE_ON_DCANDAC
};

enum FSM_Transition
{
   FSM_TRANSITION_NONE,
   FSM_TRANSITION_OFF_1,
   FSM_TRANSITION_ON_1,
   FSM_TRANSITION_ON_CHARGE_1,
   FSM_INITIAL_TRANSITION_ON,
   FSM_TRANSITION_ON_DC_1,
   FSM_TRANSITION_ON_DCANDAC_1,
   FSM_INITIAL_TRANSITION
};

#define FSM_MAX_NUM_TAKEN_TRANSITIONS 2

#define FSM_INTERNAL_VARIABLES_OFFSET 1
#define FSM_NUM_INTERNAL_VARIABLES 3

#define CurrentState fsm_struct->fsm_discStates[0]
#define TakenTransition(i) fsm_struct->fsm_takenTransitions[i]
#define IsMajorStep fsm_struct->fsm_isMajorTimeStep
#define CurrentTime fsm_struct->fsm_currentTime
#define SetErrorMessage(string) { *fsm_struct->fsm_errorStatus = (string); }
#define SetWarningMessage(string)

/* input variables */
#define ON (*fsm_struct->fsm_inputs[0][0])
#define Vdc (*fsm_struct->fsm_inputs[1][0])
#define Sw_AC (*fsm_struct->fsm_inputs[2][0])

/* internal constants */
#define fs fsm_struct->fsm_internalConstants[0]

/* internal variables */
```

```
#define tac fsm_struct->fsm_discStates[FSM_INTERNAL_VARIABLES_OFFSET + 0]
#define V fsm_struct->fsm_discStates[FSM_INTERNAL_VARIABLES_OFFSET + 1]
#define tc fsm_struct->fsm_discStates[FSM_INTERNAL_VARIABLES_OFFSET + 2]

/* output variables */
#define EnableDC (*fsm_struct->fsm_outputs[0][0])
#define state (*fsm_struct->fsm_outputs[1][0])
#define EnableAC (*fsm_struct->fsm_outputs[2][0])
#define Vref (*fsm_struct->fsm_outputs[3][0])
#define mDC (*fsm_struct->fsm_outputs[4][0])
#define Sw_charge (*fsm_struct->fsm_outputs[5][0])



static void fsm_state_Off_EnterAction(const struct FSM_Struct* fsm_struct)
{
   EnableDC = 0;
   mDC = 0;
   Sw_charge = 0;
   state = 0;
}

static void fsm_state_ON_Charge_EnterAction(
                                            const struct FSM_Struct* fsm_struct)
{
   mDC = 1;
   Sw_charge = 0;
   EnableDC = 0;
   tc = 0;
   state = 1;
}

static void fsm_state_ON_DC_EnterAction(const struct FSM_Struct* fsm_struct)
{
   Sw_charge = 1;
   EnableDC = 1;
   tc=0;
   Vref=800;
   state = 3;
}

static void fsm_state_ON_DCandAC_EnterAction(
                                             const struct FSM_Struct* fsm_struct)
{
   state=4;
   EnableAC=1;
}


static void fsm_state_ON_Charge_DuringAction(
                                             const struct FSM_Struct* fsm_struct)
{
   tc = tc+1/fs;
}

static void fsm_state_ON_DC_DuringAction(const struct FSM_Struct* fsm_struct)
{
   tc = tc +1/fs;
}


static void fsm_state_ON_DCandAC_ExitAction(
                                            const struct FSM_Struct* fsm_struct)
{
   EnableAC = 0;
}
```

```
void Control_System_0_fsm_start(const struct FSM_Struct *fsm_struct)
{
   int fsm_i;
   CurrentState = FSM_STATE_NONE;
   for (fsm_i = 0; fsm_i < FSM_MAX_NUM_TAKEN_TRANSITIONS; fsm_i++)
   {
      TakenTransition(fsm_i) = FSM_TRANSITION_NONE;
   }
}

void Control_System_0_fsm_output(const struct FSM_Struct *fsm_struct)
{
   if (IsMajorStep)
   {
      int fsm_i;
      for (fsm_i = 0; fsm_i < FSM_MAX_NUM_TAKEN_TRANSITIONS; fsm_i++)
      {
         TakenTransition(fsm_i) = FSM_TRANSITION_NONE;
      }
      switch ((int)CurrentState)
      {
      case FSM_STATE_OFF:
         if (ON == 1)
         {
            TakenTransition(0) = FSM_TRANSITION_OFF_1;
            TakenTransition(1) = FSM_INITIAL_TRANSITION_ON;
            fsm_state_ON_Charge_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_ON_CHARGE;
         }
         break;

      case FSM_STATE_ON_CHARGE:
         if (ON == 0)
         {
            TakenTransition(0) = FSM_TRANSITION_ON_1;
            fsm_state_Off_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_OFF;
         }
         else if (Vdc >= 400 || tc > 0.15)
         {
            TakenTransition(0) = FSM_TRANSITION_ON_CHARGE_1;
            fsm_state_ON_DC_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_ON_DC;
         }
         else
         {
            fsm_state_ON_Charge_DuringAction(fsm_struct);
         }
         break;

      case FSM_STATE_ON_DC:
         if (ON == 0)
         {
            TakenTransition(0) = FSM_TRANSITION_ON_1;
            fsm_state_Off_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_OFF;
         }
         else if (Sw_AC == 1 && tc >= 0.5)
         {
            TakenTransition(0) = FSM_TRANSITION_ON_DC_1;
            fsm_state_ON_DCandAC_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_ON_DCANDAC;
         }
         else
         {
            fsm_state_ON_DC_DuringAction(fsm_struct);
         }
```

```
            break;

         case FSM_STATE_ON_DCANDAC:
            if (ON == 0)
            {
               fsm_state_ON_DCandAC_ExitAction(fsm_struct);
               TakenTransition(0) = FSM_TRANSITION_ON_1;
               fsm_state_Off_EnterAction(fsm_struct);
               CurrentState = FSM_STATE_OFF;
            }
            else if (Sw_AC == 0)
            {
               fsm_state_ON_DCandAC_ExitAction(fsm_struct);
               TakenTransition(0) = FSM_TRANSITION_ON_DCANDAC_1;
               fsm_state_ON_DC_EnterAction(fsm_struct);
               CurrentState = FSM_STATE_ON_DC;
            }
            break;

         default:
            TakenTransition(0) = FSM_INITIAL_TRANSITION;
            fsm_state_Off_EnterAction(fsm_struct);
            CurrentState = FSM_STATE_OFF;
            break;
         }
      }
   }
```

B. Control Card Interface Pin Map

TI F28335 controlCARD Pin Map

Function	RT Box	100-pin		RT Box	Function
V33D-ISO		1	51		V33D-ISO
		2	52		
		3	53		
		4	54		
		5	55		
GND-ISO		6	56		GND-ISO
ADCIN-B0	AO14	7	57	AO15	ADCIN-A0
GND		8	58		GND
ADCIN-B1	AO12	9	59	AO13	ADCIN-A1
GND		10	60		GND
ADCIN-B2	AO10	11	61	AO11	ADCIN-A2
GND		12	62		GND
ADCIN-B3	AO8	13	63	AO9	ADCIN-A3
GND		14	64		GND
ADCIN-B4	AO6	15	65	AO7	ADCIN-A4
		16	66		
ADCIN-B5	AO4	17	67	AO5	ADCIN-A5
		18	68		
ADCIN-B6	AO2	19	69	AO3	ADCIN-A6
		20	70		
ADCIN-B7	AO0	21	71	AO1	ADCIN-A7
		22	72		
GPIO-00, EPWM-1A	DI17	23	73	DI16	GPIO-01, EPWM-1B
GPIO-02, EPWM-2A	DI19	24	74	DI18	GPIO-03, EPWM-2B

Function	RT Box	100-pin		RT Box	Function
GPIO-04, EPWM-3A	DI21	**25**	**75**	DI20	GPIO-05, EPWM-3B, ECAP-1
GPIO-06, EPWM-4A	DI23	**26**	**76**	DI22	GPIO-07, EPWM-4B, ECAP-2
GND		**27**	**77**		+5 V
GPIO-08, EPWM-5A, CANTX-B	DI25	**28**	**78**	DI24	GPIO-09, EPWM-5B, SCITX-B, ECAP-3
GPIO-10, EPWM-6A, CANRX-B	DI27	**29**	**79**	DI26	GPIO-11, EPWM-6B, SCIRX-B, ECAP-4
GPIO-48, ECAP5	DI29	**30**	**80**	DI28	GPIO-49, ECAP6
		31	**81**		
		32	**82**		+5 V
		33	**83**	DO0	GPIO-13, TZ-2, CANRX-B
		34	**84**	DO5	GPIO-14, TZ-3, SCITX-B
GPIO-24, ECAP-1, EQEPA-2	DO6	**35**	**85**	DO7	GPIO-25, ECAP-2, EQEPB-2
GPIO-26, ECAP-3, EQEPI-2	DO4	**36**	**86**		
GND		**37**	**87**		+5 V
		38	**88**		
		39	**89**		
GPIO-20, EQEPA-1, CANTX-B	DO2	**40**	**90**	DO3	GPIO-21, EQEPB-1, CANRX-B
		41	**91**	DO1	GPIO-23, EQEPI-1, SCIRX-B
		42	**92**		+5 V
GPIO-28 , SCIRX-A		**43**	**93**		GPIO-29, SCITX-A
GPIO-30, CANRX-A		**44**	**94**		GPIO-31, CANTX-A
GPIO-32	DI31	**45**	**95**	DI30	GPIO-33
		46	**96**		+5 V
GND		**47**	**97**		JTAG-TDI

Function	RT Box	100-pin		RT Box	Function
JTAG-TCK		**48**	**98**		JTAG-TDO
JTAG-TMS		**49**	**99**		JTAG-TRSTn
JTAG-EMU1		**50**	**100**		JTAG-EMU0

Table 3.4: TI F28335 controlCARD pin map