

# **Microgrid Operation With Load Shedding and Battery Control**

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Electrical Engineering

in the

College of Graduate Studies

University of Idaho

by

Jordan Benjamin Scott

Major Professor: Herbert L. Hess, Ph.D.

Committee Members: Brian K. Johnson, Ph.D., John Crepeau, Ph.D.

Department Administrator: Mohsen Guizani, Ph.D.

December 2017

**Authorization to Submit Thesis**

This thesis of Jordan Scott, submitted for the degree of Master of Science with a major in Electrical Engineering and titled “Microgrid Operation With Load Shedding and Battery Control,” has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies for approval to the College of Graduate Studies for approval.

Major Professor: \_\_\_\_\_ Date: \_\_\_\_\_

Herbert L. Hess, Ph.D.

Committee Members: \_\_\_\_\_ Date: \_\_\_\_\_

Brian K. Johnson Ph.D.

\_\_\_\_\_ Date: \_\_\_\_\_

John Crepeau, Ph.D.

Department Administrator:

\_\_\_\_\_ Date: \_\_\_\_\_

Mohsen Guizani, Ph.D

## **Abstract**

One of the goals of this thesis is to implement a load shedding scheme within a microgrid to drop the lowest priority load when demand is anticipated to become greater than the supply. Another goal is to implement a battery controller that communicates to the battery when to charge, when to discharge, and how much to charge or discharge. By doing this, critical loads could be kept online and will not have to be shed.

This research project successfully designed and tested a load shedding scheme in a real time digital simulator to keep critical loads online, as well as adding a battery controller to help keep loads from being shed when possible. This controller includes commands such as when to charge, when to discharge, and how much to discharge. By doing these things a significant portion of the microgrid studied has been shown to continue to have power in the event of a blackout that could last anywhere from minutes to days, no matter what time of year the blackout occurs.

## Acknowledgements

While working on my thesis I have had the great honor of working with some very amazing people whom I would like to thank.

I would like to thank both my advisor Dr. Herbert Hess and Dr. Brian Johnson for helping me with this project. Both of whom went above and beyond what was required. They spent many hours providing valuable knowledge and making sure I had the resources needed to make this project a success. They made working on this project enjoyable and for that I thank them.

I had the honor of working with several undergraduate students during the course of my research, and without them I would not have been able to accomplish as much work as I did. Those students include: Jacquelyn England, Lexi Turkerburg, Christine Page, Maximilian Schnitker, Nathan Bliesner, and Keegan Miley-Hunter.

Each of those students helped me accomplish various aspects of the project. Jacquelyn and Lexi helped me get the project going and did whatever task was asked of them. Both of them helped work on the load shedding and the HMI. Christine and Keegan ended up taking over the HMI and spent many hours getting it set up. Maximilian worked on the load shedding and helped develop a model for testing. Nathan took on a new task designing a controller for the batteries.

Last of all I would like to thank my high school math teacher Mr. Keith Price. He taught engineering is more than math. He taught me learning math is just a stepping stone for building and designing various things, and to look for the fun in it. As a result he is one of the main reasons I am an engineer today.

### Dedication

*I would like to dedicate this work to my incredible family,  
my parents Greg and Andrea as well as my sisters Holly  
and Hailie.*

## Table of Contents

<b>Authorization to Submit Thesis.....</b>	<b>ii</b>
<b>Abstract .....</b>	<b>iii</b>
<b>Acknowledgements .....</b>	<b>iv</b>
<b>Dedication.....</b>	<b>v</b>
<b>Table of Contents.....</b>	<b>vi</b>
<b>List of Figures .....</b>	<b>x</b>
<b>List of Tables.....</b>	<b>xiii</b>
<b>Chapter 1 Load Shedding Introduction .....</b>	<b>1</b>
1.1 Background.....	1
1.2 Problem Statement.....	1
1.3 Proposed Solution.....	3
<b>Chapter 2 Literature Review.....</b>	<b>5</b>
2.0 Micogrid Overview.....	5
2.1 Benefits of a Microgrid.....	5
2.2 Load Shedding.....	5
2.3 Battery Control .....	7
2.4 Diesel to Natural Gas Converters .....	7
2.5 Chapter 2 Summary .....	8
<b>Chapter 3 Design and Model Development.....</b>	<b>9</b>

3.0 Load Shedding Definition.....	9
3.1 Undervoltage Parameters.....	9
3.2 Initial Design .....	10
3.3 Chapter 3 Summary .....	12
<b>Chapter 4 Software Development for Load Shedding Scheme .....</b>	<b>13</b>
4.0 Software.....	13
4.1 RSCAD Implementation.....	14
4.2 Load Shedding Scheme in RSCAD.....	14
4.3 Cbuilder in RSCAD.....	14
4.4 Chapter 4 Summary .....	16
<b>Chapter 5 Controller Implementation in RSCAD.....</b>	<b>17</b>
5.0 I/O Ports of the Component.....	17
5.1 Parameters of Component.....	18
5.2 Chapter 5 Summary .....	20
<b>Chapter 6 Simulink and RTDS Simulation Results of Load Shedding Scheme.....</b>	<b>21</b>
6.0 Simulation in Simulink .....	21
6.1 Simulating the Load Shedding Scheme in RSCAD .....	22
6.2 Chapter 6 Summary .....	27
<b>Chapter 7: Battery Controller .....</b>	<b>29</b>
7.1 Introduction.....	29

7.2 Battery Model Introduction .....	31
7.3 Battery Controller in RSCAD.....	32
7.4 Battery Controller Implementation in RSCAD .....	35
7.5 Battery Power System Model Configuration in RSCAD .....	36
7.7 Implementation in Simulink .....	41
7.8 Simulation of Battery Controller in Simulink .....	42
7.9 Chapter 7 Summary .....	43
<b>Chapter 8: Diesel to Natural Gas Conversion Kits.....</b>	<b>45</b>
8.0 Introduction to Diesel to Natural Gas Conversion .....	45
8.1 Diesel to Natural Gas Conversion Kit.....	45
8.2 Conversion Kit Specifications .....	46
8.3 Chapter 8 Summary .....	46
<b>Chapter 9: Automation Controller .....</b>	<b>48</b>
9.0 Automation Controller Introduction .....	48
9.1 Hardware Development.....	48
9.2 Software Configuration .....	49
9.3 Protection Relay.....	49
9.4 GOOSE Protocol Communications .....	51
9.5 The Human Machine Interface .....	52
9.6 Chapter 9 Summary .....	54



<b>Chapter 10 Summary of Accomplished Work and Future Work.....</b>	<b>55</b>
10.0 Summary of Accomplished Work .....	55
10.1 Future Work: Transfer of Automation Logic to RTAC.....	56
10.2 Future Work Time Based Load Shedding .....	56
10.3 Future Work Fully Implementing Load Shedding in RTDS Hardware in Loop Simulation.....	57
10.4 Battery Control Future Work.....	57
10.5 Future Work on Diesel to Natural Gas Conversion.....	58
10.6 Future Work Communication Between an RTDS and Automation Controller.....	58
10.7 Future Work Establishing GOOSE Protocol Communications for the RTAC.....	60
<b>References.....</b>	<b>62</b>
Appendix A: Truth Tables for Finite State Machine .....	66
Appendix B: Next State and Output Logic Diagrams .....	68
Appendix C: RSCAD Files.....	77
.h and .c files .....	77
Appendix D: Battery Controller Coding.....	94
Appendix E: Battery Power System Coding .....	99
Appendix F: Software and Hardware Used in Automation controller Development.....	103

## List of Figures

Figure 1.1: Microgrid Area.....	2
Figure 1.2: Monthly Average Hydro Electric Generator Flow Rates [1].....	3
Figure 1.3: Monthly Solar Cell Power Generation Potential [1].....	3
Figure 3.1: Load Shedding Scheme CFD.....	10
Figure 3.2: Data Flow Diagram for Main Undervoltage System.....	11
Figure 4.1: Graphical Representation of Load Shedding Controller.....	15
Figure 5.1: List of Inputs.....	18
Figure 5.2: List of Outputs.....	18
Figure 5.3: List of Parameters.....	19
Figure 5.4: Overview of Available System Configurations.....	20
Figure 6.1: Simulink Simulations All Loads Triggering.....	21
Figure 6.2: Simulink Simulations Load Logic Resetting.....	22
Figure 6.3: Load Shedding Interface.....	24
Figure 6.4: Load Shedding Simulation 2 Bus Voltages.....	25
Figure 6.5: Load Shedding Simulation 3 2 Loads Tripped.....	26
Figure 6.6: Load Shedding Simulation 4 All but Highest Trip.....	27
Figure 7.1: Battery Control Flow Diagram.....	30
Figure 7.2: Battery Controller.....	32
Figure 7.3: Controller Inputs.....	33
Figure 7.4: Controller Outputs.....	34
Figure 7.5: Controller Parameters.....	35
Figure 7.6: 2 Controller System.....	35

Figure 7.7: Battery Model.....	36
Figure 7.8: Battery Inputs.....	37
Figure 7.9: Battery Nodes.....	37
Figure 7.10: Battery Parameters .....	38
Figure 7.11: Battery Currents .....	38
Figure 7.12: Battery Monitored Values .....	39
Figure 7.13: State Transition Diagram .....	40
Figure 7.14: Battery Controller in Simulink.....	42
Figure 9.1: Hardware Implementation Scheme.....	48
Figure 9.2: Software Implementation Scheme .....	49
Figure 9.3: Example of SEL Relay LED Alarm Light Protection.....	50
Figure 9.4: HMI for One Breaker .....	53
Figure 9.5: Bus Voltages and Line Currents for One Relay.....	53
Figure 10.1: Combined Average Generation Output Profile Over a 24Hr Period in 4 Seasons [1].....	56
Figure 10.2: Editing Outputs in the SCD file .....	59
Figure 10.3: GTNET-GSE Output Parameters.....	60
Figure 10.4: Establishing GOOSE Protocol for the Relay .....	61
Figure B.1: S2 Logic.....	68
Figure B.2: S1 Logic.....	69
Figure B.3: S0 Logic.....	70
Figure B.4: Output Logic Load 1 .....	71
Figure B.5: Output Logic Load 2 .....	72

Figure B.6: Output Logic Load 3 .....	73
Figure B.7: Output Logic Load 4 .....	74
Figure B.8: Output Logic Load 5 .....	75
Figure C.1: Cbuilder .def File.....	77

**List of Tables**

Table 7.1: Output Transition Table .....	41
Table 7.2: Battery Controller Simulation Low Frequency Input.....	42
Table 7.3: Battery Controller Simulation High Frequency Input.....	43
Table 9.1: Example of SEL Relay Breaker Trip Tag .....	50
Table 9.2: Example of Establishing Alias Relay Tag.....	51
Table 9. 3: Example of HMI Relay Trigger Tag .....	51
Table A. 1: State Transition Table With Encoding .....	66
Table A. 2: Output Table With Encoding.....	67

## **Chapter 1 Load Shedding Introduction**

### **1.1 Background**

In the instant that a local or regional blackout occurs, power is lost. The duration of this loss can range from minutes to several days. To mitigate the effects of such a blackout on a regional utility, a microgrid has been designed to provide power and stability to critical facilities such as hospitals, jails, and government buildings. In an ideal microgrid, available generation, would always be greater than demand. In a situation where load exceeds generation load would have to be shed. As a result of this, in order to keep the microgrid operational, a scheme is needed to implement prioritized load shedding.

### **1.2 Problem Statement**

The proposed microgrid shown in Figure 1.1 will utilize two hydro-electric generation sites and solar power from within the micogrid. The available generation is often less than the total load in the microgrid footprint. These power sources are influenced by season variations and daily weather patterns and could fluctuate greatly in their generated power, voltage magnitude, and frequency (Figure 1.2 and 1.3) [1]. These figures only show the seasonal variation, not daily. Due to this potential for instability and due to mismatch between generation and loads, it is necessary to have a plan for immediate reductions in power consumption to maintain operational voltage to critical loads.

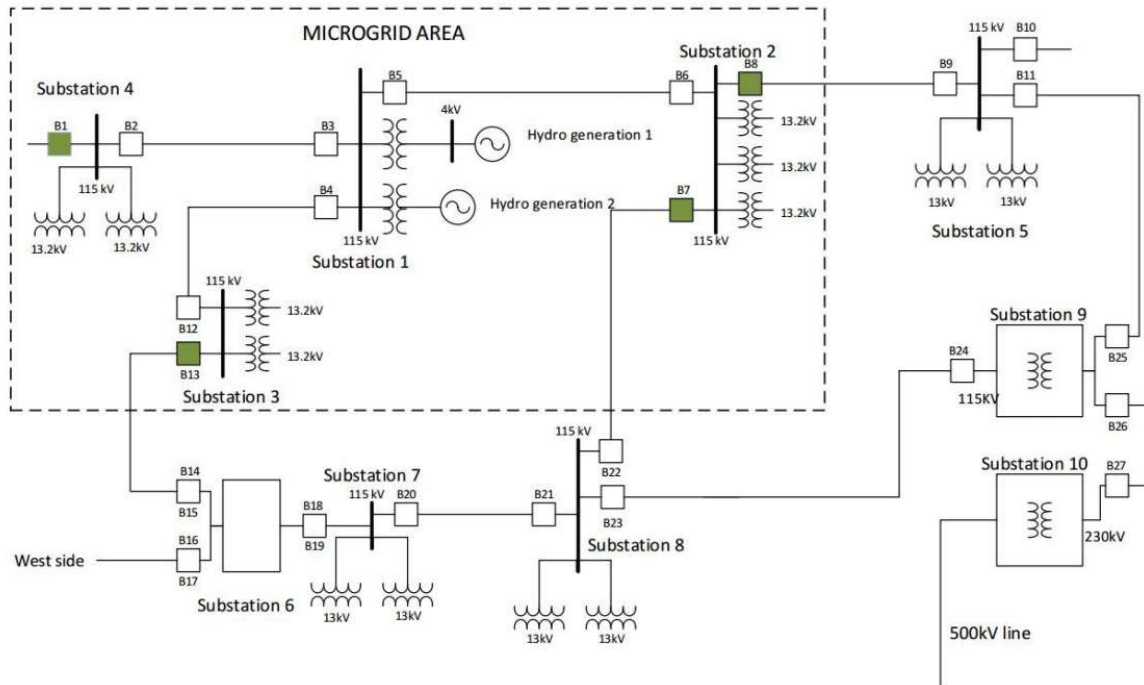


Figure 1.1: Microgrid Area

Another problem can occur if load reduction is not enough to maintain operational voltage and frequency to the critical loads. If the load is too small, generation can be reduced. If load is too large, generation can't supply all the loads. The next step from there is to shed load to keep the microgrid from becoming unstable and having a voltage collapse. The first load to be shed is the lowest priority load in the system. Once this first load has been shed, the load shedding scheme will determine if another load needs to be shed in turn in order to maintain stability.

One of the other questions that arises from shedding the load is how long should the system wait before it sheds another load? If the load shedding scheme is initiated too soon, frequency and voltage oscillations could cause a false shedding command to be sent to one of the loads. The solution to this is to set sufficient time delay between load shedding commands so there is negligible chance of transients causing an error. A backup to this would be to set the load shedding controller such that a load has to be under a set voltage

threshold for a set amount of time before it sheds. The timing has to be set so that a load isn't shed if any of the bus voltages in the system merely fall below the set voltage momentarily, but also has to be set fast enough so that the system doesn't become unstable and suffer a blackout.

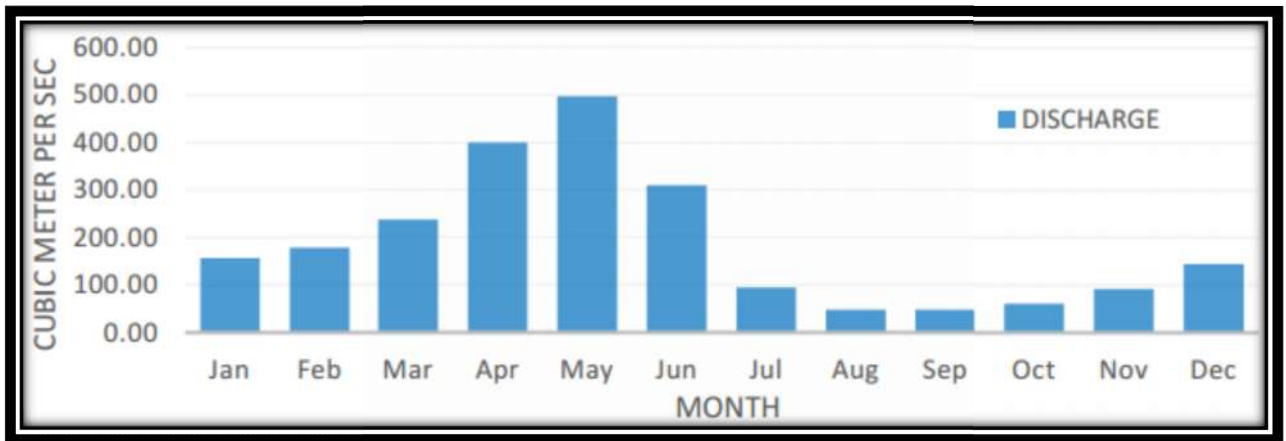


Figure 1.2: Monthly Average Hydro Electric Generator Flow Rates [1]

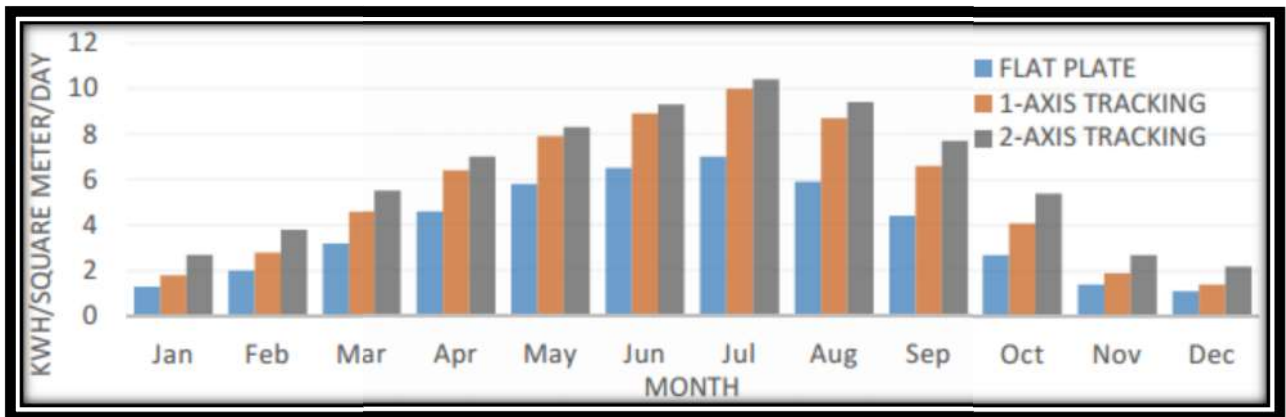


Figure 1.3: Monthly Solar Cell Power Generation Potential [1]

### 1.3 Proposed Solution

When the microgrid is in an island state, it will be responsible for providing power to the critical loads in the local area. The goal of this work is to develop a load shedding scheme when it isn't possible to supply all the loads in the microgrid. The load shedding



scheme will act to disconnect the lowest level priority loads to maintain the voltage magnitude. This load shedding scheme constantly measures voltage, and will trigger when a critical load's voltage magnitude drops below a preselected level for a preselected time, such as 0.90 per-unit for 5 seconds. Once an undervoltage event occurs, the system will identify the prioritized critical load based on the information found from a past research project [1]. Once this identification has been made, the lowest priority load will be shed. Shedding load provides voltage, and frequency stability to the microgrid. As a result of the load shedding scheme constantly monitoring voltage, if shedding the first load is not enough to maintain acceptable voltage levels across the system, the load shedding scheme will initiate another load shed. This process will continue until voltages across the system are within an acceptable range. Once voltage is in this range, the load shedding scheme reverts to only monitoring voltages.

This thesis covers a few main points. In detail, it defines what load shedding is and why it is needed. Following this is the development of a load shedding scheme which includes logic diagrams and software development. Finally it shows the results of the load shedding scheme. In this microgrid load shedding is the last resort. One of the options to be checked before load is shed utilize the batteries that can supply power to the microgrid, and keep load from being shed.

The objective of the energy storage model is to add the batteries and their energy storage control in the present microgrid model. The purpose of the battery controller is to evaluate the frequency of the system and send a signal to the battery to correct the frequency, as well as help with real power management.

## **Chapter 2 Literature Review**

### **2.0 Microgrid Overview**

Microgrids are power system configurations that give both economic and environmental benefits when compared to expanding legacy modern power systems. [15]. In other words, microgrids are power systems with power consumption located near generation such that the microgrid can become an independent entity, even controlled and dispatched separately [7]. Another useful aspect of a microgrid is that it can generate in both islanded and grid connected mode. In order to be able to do this, concepts such as load shedding and battery control need to be looked at in order to keep the system stable.

### **2.1 Benefits of a Microgrid**

Microgrids offer several potential benefits power utilities:

1. Reduce the carbon footprint by reduction in the use of conventional sources of energy if they are based on renewable energy generation [16].
2. Lower operational costs by reducing the number and length of transmission and distribution lines [11].
3. Encouraging utilization of renewable energy sources [3].
4. Supplemental power for peak loading, which in turn postpones upgrading the system [31].
5. Improve system reliability [17].

### **2.2 Load Shedding**

An advantage of local control is that it relies on local measurements, thus avoiding certain communication dependencies leading to jitter or delay. The communication architecture within the microgrid needs to be well established, so that it can be utilized

within the demand control applications. It is advisable to combine the central and local control schemes where fast and coordinated response is required [1]. Load shedding is performed in order to keep the frequency and voltage of the power system stable. The most economical way of improving system stability is to equalize the generation to load (via load shedding or generator control), thereby minimizing the disturbance impact to the power system [12].

A simplified model of the microgrid in Figure 1.1 is implemented in Real Time Digital Simulator (RTDS) [1]. RTDS provides opportunity to perform hardware in loop simulations. It provides low level output signals, which resemble the real system data such as current transformer outputs or voltage transformer outputs. These outputs can be directed to intelligent electronic devices to perform protection or automation operations [1]. One of those operations is load shedding, which will shed load based on undervoltage conditions.

A historical method for detecting power unbalances is to detect a decrease in power system voltages [13]. Once this detection has been made, and load shedding has been decided, a load shedding scheme will determine which load to shed. By doing this, voltage levels will be regulated to improve local reliability and stability [14].

One of the popular load shedding methods for a microgrid is implementing an undervoltage scheme. The Western Region of the Entergy System is located within Entergy Texas service area, utilizes this type of scheme [22]. Through design and testing it was found that this method would be sufficient. This load shedding scheme was put into service in June 2014 and as of mid 2015 no operational issues have been reported [22].

### **2.3 Battery Control**

By installing a battery energy storage system within a microgrid, power can be supplied to critical loads. There are also other pros of using batteries in a power system. A few of the advantages are savings in terms of reduced power bought during the peak hours, improving power quality in cases where variable renewable power sources are connected to the grid, and providing emergency power to critical loads [1]. In order to be efficient, battery sizing needs to be looked at.

As stated in the previous paragraph, it is important to size the battery appropriately. Batteries can improve the balance between generation and demand, and thus, the battery size will have significant impact on the grid's economic operation [1]. When sizing batteries for this system, the following rules were followed. Those rules include: operating considerations, design and aging margins, voltage window design, and charging and discharging limitations [18]. Once the size of the batteries has been calculated, the next thing to look at is placement.

Optimally placed batteries can regulate the voltage throughout the power system, thus reducing the need to expand the current power system or to shed load. [19]. This thesis uses what was described in the previous paragraphs and implement a battery model and battery controller in Matlab Simulink.

### **2.4 Diesel to Natural Gas Converters**

Throughout the microgrid area, there are several diesel generators that can produce approximately 14.75 MW of power. Due to emission limits, these aren't a viable source of power for this microgrid. A dual fuel diesel engine is a diesel engine fitted with a dual fuel conversion kit to enable use of cleaner burning alternative fuel like compressed natural gas

in addition to oil [20]. Performance tests have shown that the emissions are significantly less when compared to the standard diesel generator [21]. This thesis will look into applying diesel to natural gas conversions within the microgrid area.

One minor issue that can be encountered with diesel to natural gas converters are diesel is still needed to ignition of the engine [23]. Another issue is that once the engine is modified the warranty can be voided [23]. These issues were overcome by keeping a supply of diesel ready, and by using a United States law that says warranties are not voided after aftermarket accessories have been added [23].

## **2.5 Chapter 2 Summary**

Chapter 2 is a very brief literature review on work pertaining to microgrids. This chapter discusses what a microgrid is and the benefits that come from it. This is followed by how load shedding is an option to keep the system up and running when generation can't meet demand. There are several methods to achieve load shedding with one of the more common ways to accomplish this is to use an undervoltage scheme. Load shedding is considered the last resort option. Before load is shed within the microgrid batteries can be added to keep the system stable. To best utilize the batteries a battery model and a battery controller will be implemented to best serve the microgrid.

## **Chapter 3 Design and Model Development**

### **3.0 Load Shedding Definition**

Load shedding is the intentional reduction of load on a power system, normally to increase, but possibly to decrease the voltage magnitude and frequency to appropriate levels. In other words, load shedding is the act of balancing available generation and load when load exceeds generation. Controlling the voltage magnitude of the microgrid will help maintain the overall stability of the system. By shedding load the microgrid controller will be able to maintain power to the highest priority loads without interruption.

Load shedding helps voltage stability by balancing reactive power to help keep the microgrid operational. Frequency stability is also important factor when looking at a power system. When applying reactive power, voltage stability is the indicator of success, while working with real power, frequency stability is the indicator of success. For this project, the load shedding scheme maintains both voltage and frequency stability. It is being done this way because the method has been demonstrated in literature review. In later chapters of this thesis, frequency measurements determine how the batteries operate.

### **3.1 Undervoltage Parameters**

There are many methods for detecting conditions to trigger load shedding in power systems, yet one of the most reliable is the monitoring of the system voltage [1]. Voltage monitoring will give a clear snapshot of the system's stability with the capacity to react quickly. This project will be monitoring the microgrid's voltage magnitudes to detect a pre-selected voltage value of below 0.9 per-unit. These parameters were chosen to allow the control systems of the hydro and solar power generation to maintain power to the critical loads for as long as possible before another load needs to be shed.

### 3.2 Initial Design

For developing a load shedding scheme, there were a few problems to keep in mind. The first problem is that the load shedding scheme would shed loads based of a priority list. The second problem is that it would shed load only after bus voltages had fallen below a certain tolerance level. The third problem is that once a load has been shed, a delay needs to be implemented so a false load shed signal is not sent and a load is taken offline. The control flow and data flow diagrams, shown in Figures 3.1 and 3.2, were created to address the problems according to these points. Figure 3.1 and Figure 3.2 are the control flow and data flow diagrams, respectively. . Figure 3.2 is a more general control flow diagram, which shows how the load shedding scheme will operate in conjunction with the battery controller.

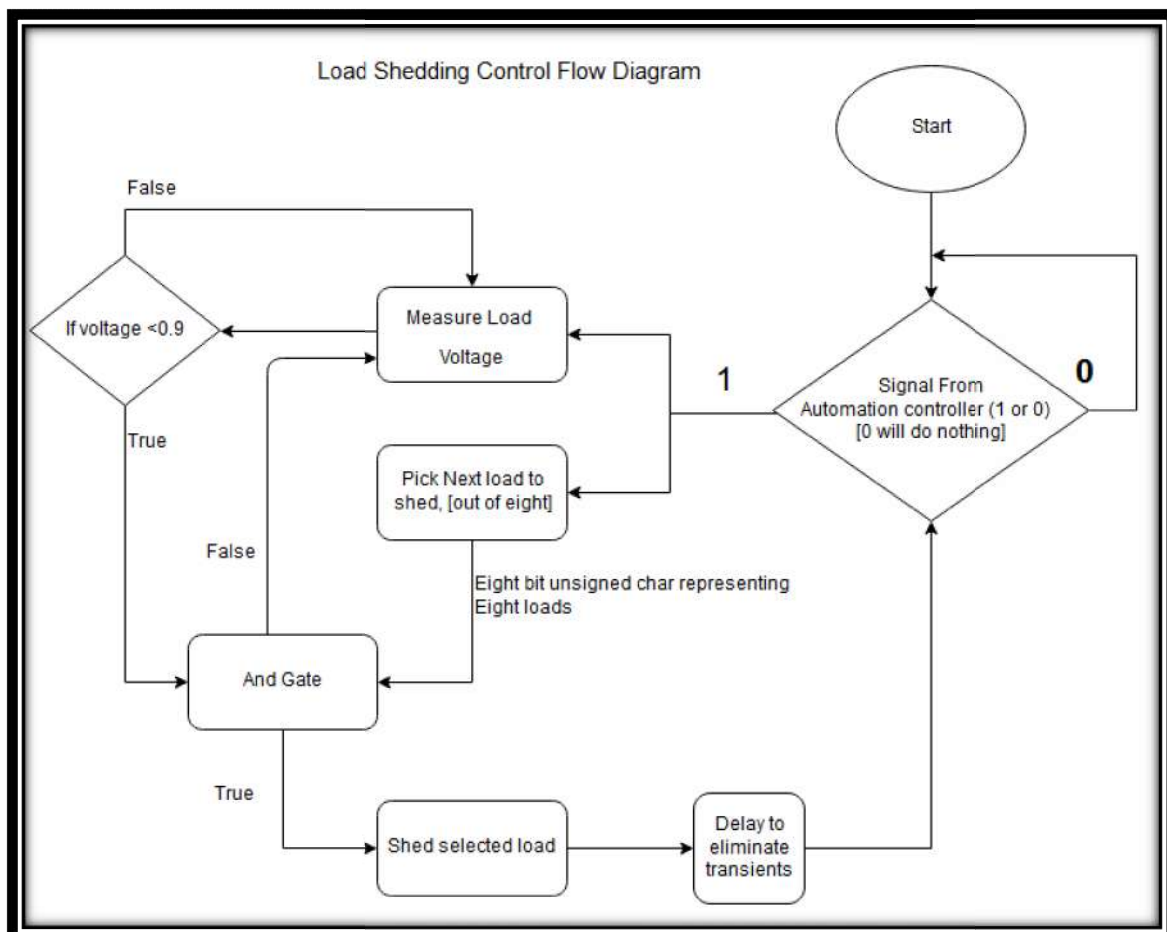


Figure 3.1: Load Shedding Scheme Control Flow Diagram

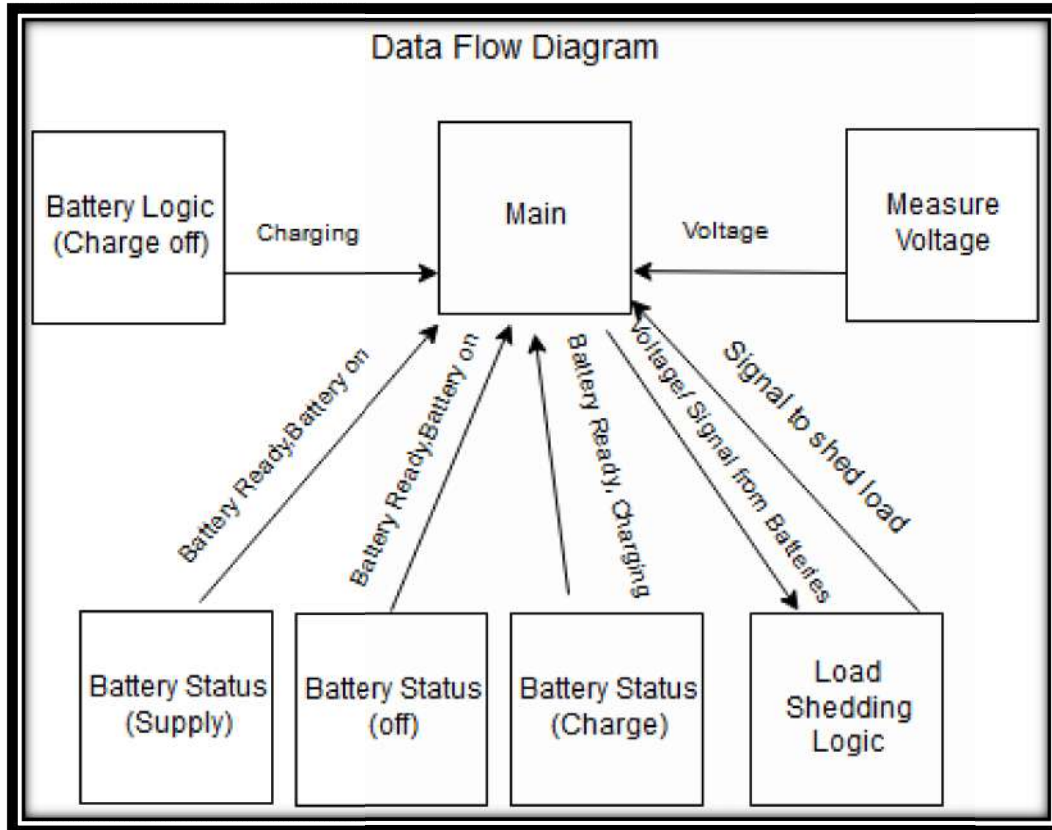


Figure 3.2: Data Flow Diagram for Main Undervoltage System

The first stage was creating the logic inside Simulink that would wait until the program sends a shed-load signal. This will occur if the batteries cannot aid the microgrid's voltage levels further. Once the load shedding controller receives this signal it will measure the incoming voltage from the microgrid loads and compare the magnitudes with the preselected value of 0.9. The load shedding scheme will check if an undervoltage event has occurred and output a Boolean value. This Boolean value will meet the load-shed signal at a logic block that will assure this conditions are met before a load is shed. If the signal to shed a load and an undervoltage event have occurred, the load-shedding controller will output the selected load that should be shed to the main program as an eight bit, unsigned character. The purpose of Figure 3.2 is to show a high level overview of how the battery controller will work with the load shedding scheme. The results in Chapter 6 will demonstrate successful



solution to problems discussed in the first paragraph of this section come from Figures 3.1 and 3.2 respectively.

### **3.3 Chapter 3 Summary**

The main points in the chapter are describing what a load shedding scheme is and why it is needed. Load shedding is the dropping a load from the power system in order to keep a power system stable. This chapter also discusses how the load shedding scheme will be created, as well as how it will work with the battery controller. Chapters 4-6 go into detail on the loading shedding scheme, and chapters 7-9 discuss the battery controller.

## **Chapter 4 Software Development for Load Shedding Scheme**

### **4.0 Software**

As stated in Chapter 3, the initial modeling and testing of the load shedding scheme was performed in the MATLAB graphical control flow simulation software, Simulink. This platform was chosen due to the vast control system libraries and the dynamic simulation capabilities the software provides. Simulink has the capacity to graphically model and test/remove errors from a control system inside the Integrated Development Environment (IDE), without using external resources. This process allows for a quick error removal and boosts the progress towards the final goal. Simulink code also has the potential to be converted to RSCAD code and implemented into the microgrid model to test the load shedding scheme with the RTDS hardware [24].

The load shedding model working in Simulink was used as a reference while the same load shedding scheme was built in RSCAD. This was done due to the fact RSCAD doesn't have a lot of tutorials to help build a control system, so Simulink was used as a stepping stone to ensure the load shedding would react correctly in RSCAD. A Finite State Machine, (FSM), was chosen over using unsigned characters and bit-masking due to the compiler requirements for real time operation of the RSCAD software when transferring c-code between the two programs. A Moore FSM was selected and designed with two inputs and five outputs. The first input comes from either the main system as an undervoltage signal or from the energy storage system when the backup batteries are unable to meet the demand of the microgrid. The second input is a reset signal that will set all the outputs to their default position. The software design were created based on the truth table and can be found in Appendix A. Following this, the FSM was created inside a program Simulink. Figures B1-B9 in Appendix B are the next state and output logic respectively. The current

state and next state logic connect via a resettable flip-flop that was designed for this system. The flip-flop operates on an external clock/pulse-generator to ease the portability into future systems. Figures B1-B9 in Appendix B also show the equations used for each logic section for the loads.

#### **4.1 RSCAD Implementation**

The final test model for the microgrid was built in RSCAD. This software uses a special purpose computing system to simulate in real time. Inside RSCAD, a new component was built specifically for the load shedding scheme using c-code. The component was then implemented into the microgrid model and setup to measure the per-unit voltage of the load buses to ensure power system stability.

#### **4.2 Load Shedding Scheme in RSCAD**

The RSCAD simulation software uses premade functions as well as a component builder to design and program the user's control or power blocks to use in a circuit or power system. For this project the Cbuilder was used to draw and code a block that will be added into the RSCAD draft circuit of the Microgrid built by a previous graduate student [1]. The use of this software should give accurate feedback on the microgrid's ability to operate with a load-shedding scheme.

#### **4.3 Cbuilder in RSCAD**

Cbuilder allows users to design, draw, code, and parameterize control and power components for use in a system models. The block design of the load shedding controller was created with the appropriate inputs and outputs needed entered into the header and c-file. Figure 4.1 illustrates the design of the load shedding controller based off Figure 3.1.

The colors of the I/O ports signify the value type; Green are real numbers (doubles data type 64 bits long) and Blue are integers (32 bits long). A more in-depth view on how the controller was designed and the C code implemented is found in Appendices A-C.

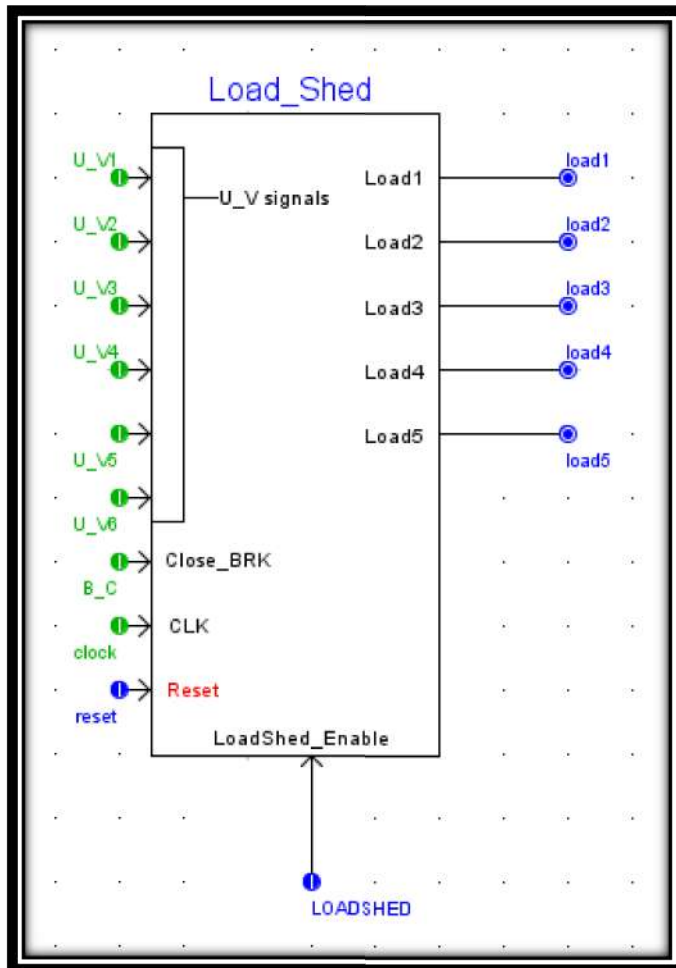


Figure 4.1: Graphical Representation of Load Shedding Controller

The tools in Cbuilder described above were used to address the following problems introduced in Chapter 3. The first problem is that the load shedding scheme would shed loads based of a priority list. The second problem is that it would shed load only after bus voltages had fallen below a certain tolerance level. The third problem is that once a load has been shed, a delay needs to be implemented so a false load shed signal is not sent and a load

is taken offline. Once a load has been shed a delay is asserted before bus voltages are measured again.

#### **4.4 Chapter 4 Summary**

The main point of this chapter was to discuss the software development for the load shedding scheme. It was first built and tested in Simulink to use the available resources available in the program. Once the controller was behaving correctly in Simulink it was then built in RSCAD using Cbuilder. Chapter 5 will further discuss the development of the load shedding scheme, and Chapter 6 will go over the results.

## Chapter 5 Controller Implementation in RSCAD

### 5.0 I/O Ports of the Component

. This chapter discusses how the load shedding controller in Figure 4.1 was developed and implemented. We want this pseudo hardware to measure voltage, and if the bus voltages fall below a set tolerance level then shed the lowest priority load on the list of loads is shed. Once a load has been shed a delay is implemented before measures bus voltages are processed so a false trip signal is not sent.

The six U\_V inputs, Figure 5.1, are the per unit values from bus nodes in the power system that the user desires to monitor for undervoltage protection. The “Close\_BRK”, (B\_C as seen below in Figure 5.2), input is the signal that tells the load-shedding controller to compare the bus voltages with the value that was chosen as the limit at which the loads should be brought back on. The “CLK” input is the clock signal that determines when the load-shedding controller will process the programmed code. When the “CLK” signal is asserted the load-shedding controller will execute the program and will determine whether a load will be shed or reconnected to the power system. The reset input has been put into the design as a precaution for the unlikely event that the FSM goes into a state that it was not designed for. When a reset is asserted the load-shedding controller software state will be forced into state zero, (idle state). The “LOADSHED” input is the signal that tells the load-shedding controller to shed load when the undervoltage conditions are met.

▼ INPUTS			
reset	INT	▼	IO Point: reset (INT) ▼
clock	REAL	▼	IO Point: clock (REAL) ▼
U_V2	REAL	▼	IO Point: U V2 (REAL) ▼
U_V3	REAL	▼	IO Point: U V3 (REAL) ▼
U_V4	REAL	▼	IO Point: U V4 (REAL) ▼
U_V5	REAL	▼	IO Point: U V5 (REAL) ▼
U_V6	REAL	▼	IO Point: U V6 (REAL) ▼
U_V1	REAL	▼	IO Point: U V1 (REAL) ▼
B_C	REAL	▼	IO Point: B C (REAL) ▼
LOADSHED	INT	▼	IO Point: LOADSHED (INT) ▼

Figure 5.1: List of Inputs For the Load Shedding Scheme

The outputs are the signals that go to the breaker controllers inside the power system. As seen from Figure 5.4, each output is an integer and can be changed inside the parameter window in the draft file to whatever value is needed for a specific breaker controller. The number of loads can be readily adjusted in the C code. An operating scenario with these parameters is tested in chapter 6.

▼ OUTPUTS			
load1	INT	▼	IO Point: load1 (INT) ▼
load2	INT	▼	IO Point: load2 (INT) ▼
load3	INT	▼	IO Point: load3 (INT) ▼
load4	INT	▼	IO Point: load4 (INT) ▼
load5	INT	▼	IO Point: load5 (INT) ▼

Figure 5.2: List of Outputs for the Load Shedding Scheme

### 5.1 Parameters of Component

The parameters of the load shedding component were designed to make the coding of the component intuitive. Each of the parameters can be seen in both Figure 5.3 and Figure 5.4. The value of the per-unit voltage that will trigger an undervoltage event is controlled by the “U\_Vrange” parameter. The value at which the breakers will be allowed is controlled by

the “Close\_BRK\_Rg” parameter. Both the “U\_Vrange” and “Close\_BRK\_Rg” are both real numbers that are represented by 64 bits. In the instance the system that runs the load-shedding controller has memory registers, both values for load action can be altered to an integer (32 bits), or a char (8 bits), to meet the conditions of the system. The controller will delay load shedding as well as the reclosing of breakers with the “FaultDelay” and the “Close\_BRK\_t”, both of which are a time delay, with the code shown in Appendix C. Prtyp is the solve model card type used in RSCAD. Proc is the assigned controls processor. Pri is the priority level of the loads. These parameters control the number of iterations the program goes through, effectively delaying the program. The last of the parameters are “OpenLOAD1-5” and “CloseLOAD1-5” and are used to set the values that output to the breaker controllers.

▼ PARAMETERS/COMPUTATIONS			
U_Vrange	REAL	▼	Parameter: U Vrange
prtyp	INT	▼	Parameter: prtyp
FaultDelay	INT	▼	Parameter: FaultDelay
Close_BRK_t	INT	▼	Parameter: Close BRK t
Close_BRK_Rg	REAL	▼	Parameter: Close BRK Rg
CloseLOAD3	INT	▼	Parameter: CloseLOAD3
CloseLOAD4	INT	▼	Parameter: CloseLOAD4
CloseLOAD5	INT	▼	Parameter: CloseLOAD5
OpenLOAD1	INT	▼	Parameter: OpenLOAD1
OpenLOAD2	INT	▼	Parameter: OpenLOAD2
OpenLOAD3	INT	▼	Parameter: OpenLOAD3
OpenLOAD4	INT	▼	Parameter: OpenLOAD4
OpenLOAD5	INT	▼	Parameter: OpenLOAD5
CloseLOAD1	INT	▼	Parameter: CloseLOAD1
CloseLOAD2	INT	▼	Parameter: CloseLOAD2

Figure 5.3: List of Parameters for the Load Shedding Scheme



PARAMETERS							
SECTION: "CONFIGURATION"							
Proc	Assigned Controls Processor	INTEGER		1	1	36	
Pri	Priority Level	INTEGER		1	1		
U_Vrange	Range of undervoltage detection	REALVAR					
prtyp	Solve Model on card type:	TOGGLE	;RPC;GPC/PB5	GP...	1	2	
Close_BRK_Rg	Range of voltage to close breakers	REAL	pu	0.92	0		
SECTION: "OUTPUTS"							
CloseLOAD1	Value of output int to close Breaker	INTEGER		1	0		
CloseLOAD2	Value of output int to close Breaker	INTEGER		1	0		
CloseLOAD3	Value of output int to close Breaker	INTEGER		1	0		
CloseLOAD4	Value of output int to close Breaker	INTEGER		1	0		
CloseLOAD5	Value of output int to close Breaker	INTEGER		1	0		
OpenLOAD1	Value of output int to OPEN Breaker	INTEGER		8	0		
OpenLOAD2	Value of output int to OPEN Breaker	INTEGER		8	0		
OpenLOAD3	Value of output int to OPEN Breaker	INTEGER		8	0		
OpenLOAD4	Value of output int to OPEN Breaker	INTEGER		8	0		
OpenLOAD5	Value of output int to OPEN Breaker	INTEGER		8	0		
SECTION: "DELAY"							
FaultDelay	Delay to rule out faults	INTVAR	sec	5	0		
Close_BRK_t	Delay until breaker(s) close	INTVAR		1			

Figure 5.4: Overview of Available System Configurations for the Load Shedding Scheme

## 5.2 Chapter 5 Summary

Chapter 5 describes how the load shedding controller in Figure 4.1 was developed. We want this controller program to measure voltage and if the bus voltages fall below a set tolerance level then shed the lowest priority load on the list of loads. It should then delay before it measures bus voltages again so a false trip signal is not sent. In short this chapter discusses how the system operates. An example of using this software is shown in Chapter 6.

## Chapter 6 Simulink and RTDS Simulation Results of Load Shedding Scheme

### 6.0 Simulation in Simulink

The Simulink simulation software was first to be used to test the load-shedding scheme. During the simulation a fixed step size was used to observe the potential time frame of the decision to shed load. Figure 6.1 shows the results of a simulation using a pulse generator with a period of 50 seconds to enable signal to shed load, which was initiated at 32 seconds. These signals are triggered to overlap each other. Once the first load has been shed and 50 seconds has passed load 2 will shed, which will then overlap the first load shed.

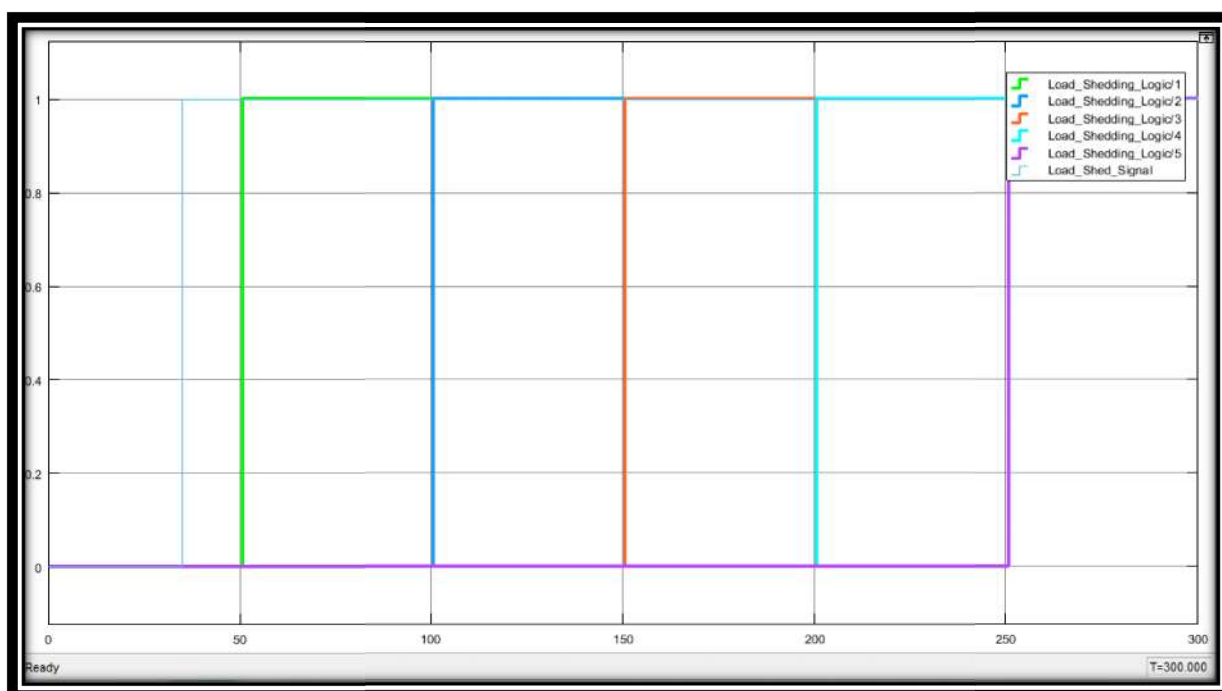


Figure 6.1: Simulink Simulations All Loads Shed in Succession

The decision to shed load shown in Figure 6.1 was based on the undervoltage logic scheme and the timing for shedding loads was based off the period of the pulse generator. The period of the pulses can then be adjusted to shed loads slower or faster based off of what works best for the system. The program may also be reset by asserting the reset input

at any time. Upon asserting the reset, at the rising edge of the pulse generator all five outputs will be pulled to zero. This can be observed in Figure 6.2.

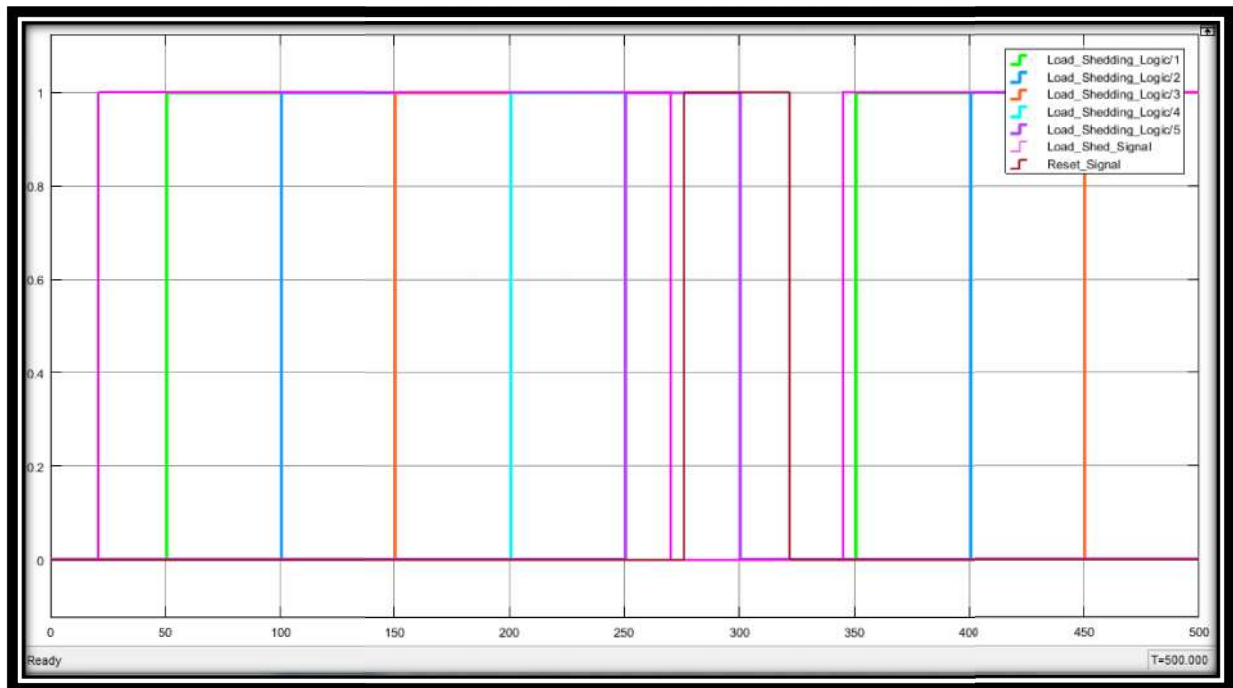


Figure 6.2: Simulink Simulation All Loads Shed in Succession Then Reset

In Figure 6.2 five outputs have sent the signal to shed load (refer to the legend on the top right). The reset input is then asserted at 285 seconds on Figure 6.2 and all load outputs are pulled to zero at 300 seconds. This 15 second delay is to ensure all the loads will be reset before the load shedding process starts again. After two pulse generator periods the load shed signal is once again asserted to show that the first load will be triggered after a reset has occurred.

### 6.1 Simulating the Load Shedding Scheme in RSCAD

The Real Time Digital Simulator (RTDS), was used to simulate the microgrid model with the load-shedding controller implemented into the model with the needed per-unit measurements attached at the correct positions. Each of the bus voltage measurements are

represented meters and green lights when each load is connected (breaker closed), and red lights when the load is disconnected from the microgrid (breaker open). This is opposite to industry practice where red represents a live circuit and green an open circuit. It should be noted that these meters are on the source side of the breaker, and as a result when load is shed the voltage meter values will increase. The meter readings for the loads that are shed in Figures 6.3, 6.5, and 6.6 will go to zero because those meters are on the load side of the breaker. The most critical load, Meter U\_V1, does not have an indicator light because it will never be shed from the microgrid. This will allow the most critical loads to stay active for the longest time with little reduction in voltage. The load-shedding controller was set to trigger an undervoltage event at a per-unit voltage of below 0.90 and reconnect a load to the microgrid at 0.93 per-unit voltage. These settings should allow the microgrid to remain stable, but further study is required to verify this. The goal of simulating the load-shedding scheme is to make sure the loads are shed when there is a threat of undervoltage and yet delayed enough to keep any transients from triggering unnecessary load shedding.

The RTDS Runtime simulation window can be seen in Figures 6.3 through 6.6 in multiple stages of the simulation. Figure 6.3 is the setup of the Runtime program. The top left meters and the lights directly below them correspond with each other as most critical load on the far left (“U\_V1”), and least critical load in the middle, (“U\_V6” and “Load1”). The switches to the right of these meters are the integer inputs that tell the load-shedding controller to shed or reconnect loads if the proper voltage condition is present. To the right of the switches are the sliders that control the excitation of the hydroelectric generators one and two, along with the “Clock” of the load-shedding controller between them. The reset switch is the last control component and is used to set the FSM back to an idle state in case

of program failure. Below the control components are the meters and graphs that monitor the bus voltages and power of specific loads.

In order to run this simulation and test the load shedding scheme as shown in the figures below the excitation of the hydroelectric dam generators was lowered in small increments. As the simulation continues one will be able to see the load voltages drop to nearly zero as each of the loads are shed. The two graphs from the middle to the right-hand side are the three-phase RMS voltage and the power of the most critical load, respectively.

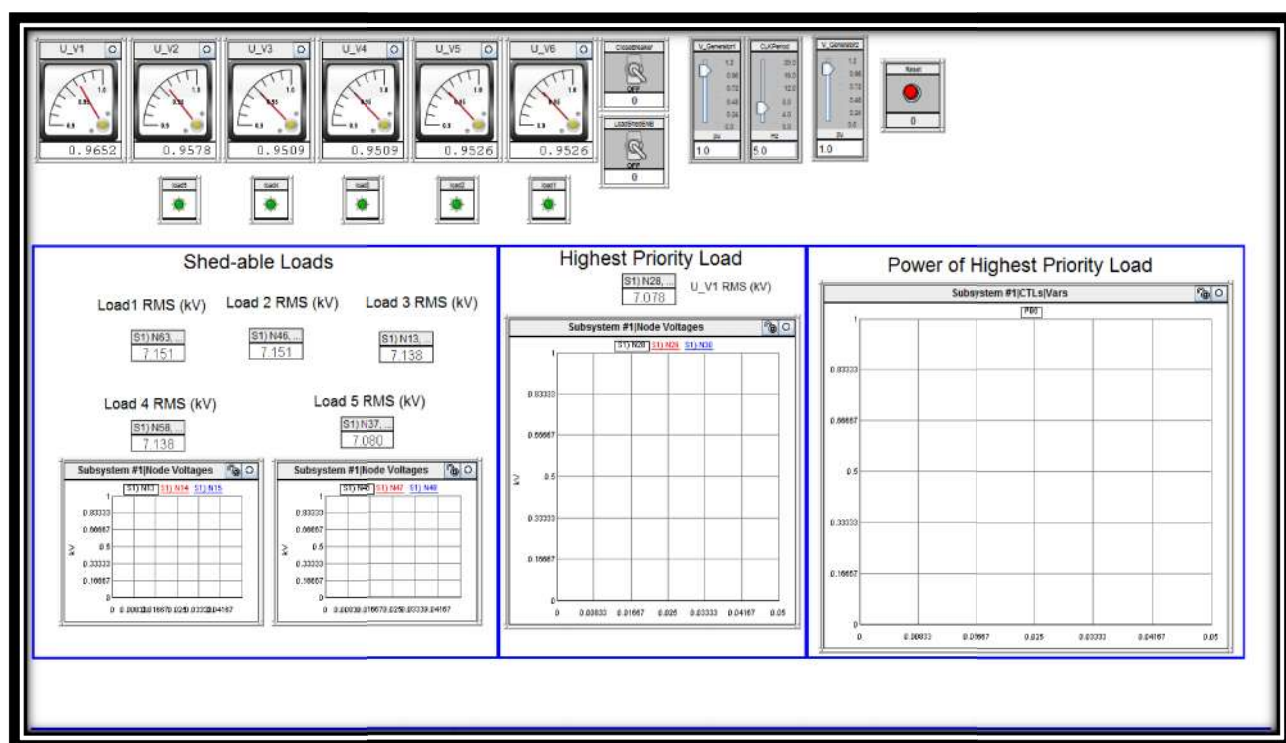


Figure 6.3: Load Shedding Interface in RTDS Runtime

Once the simulation was running to demonstrate the load shedding scheme the next step in the test was turning down the output voltage of hydroelectric generator one to simulate excess load on the system. Transitioning between Figure 6.3 and Figure 6.4 one can observe that as the voltage of the system drops the monitors will light up blue to indicate

the voltage the bus voltage has dropped below 0.9 per unit. Once the controller detects this undervoltage event a delay counter starts and waits for any transients to clear to keep loads on as long as possible.

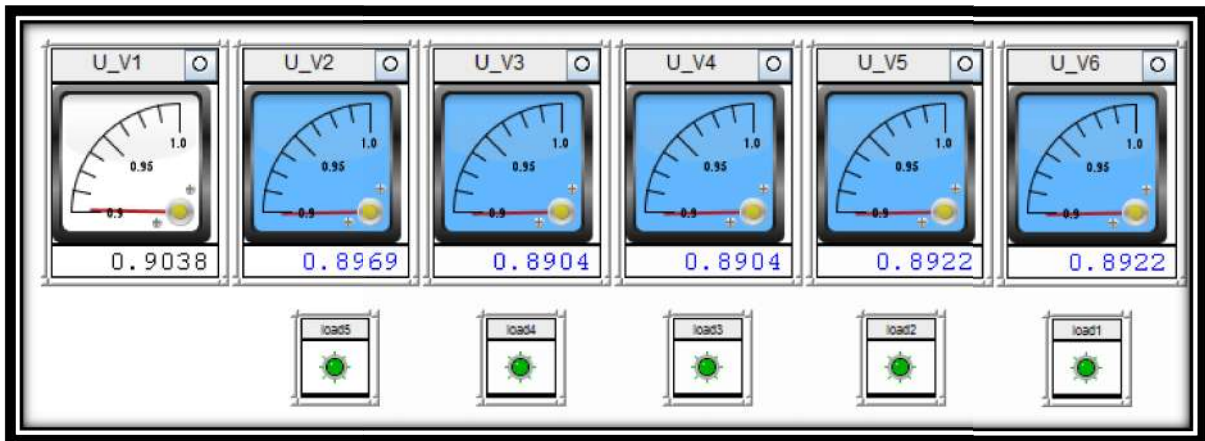


Figure 6.4: Load Shedding Simulation 2 Bus Voltages

As the delay period was completed the first load was shed. Then, a second delay period passed while the voltage was under the designed limit. Figure 6.5 shows the lights monitoring U\_V5 and U\_V6 have turned red and the bus voltages of the first two loads have dropped to almost zero. The reduction in power and voltage of the highest priority load is also visible in Figure 6.5 as well.

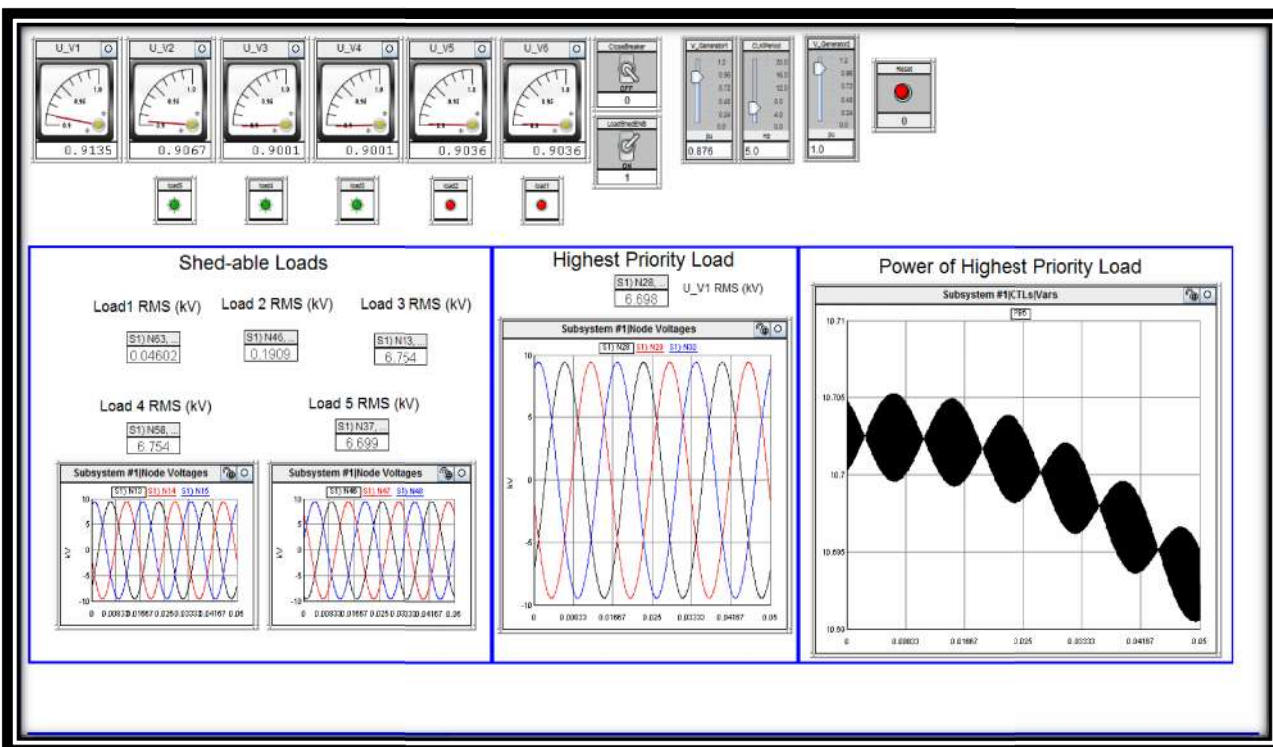


Figure 6.5: Load Shedding Simulation 3 2 Loads Tripped

The last step in testing the load-shedding was lowering the per-unit voltage of generator low enough so that the only load left online is U\_V1 which is the highest priority load. This test was performed to keep the most critical load at or above 0.9 volts per-unit. With a voltage level at or above 0.9 per-unit the systems connected to that bus would still be able to operate. Noted in Figure 6.5 is the power of the highest priority load, this value appears to vary a lot because the vertical scale is small, when in reality it varies by approximately .1MW. Figure 6.6 shows all the loads except the last load as shed. The most critical load, (U\_V1), has a per-unit value of above 0.9.

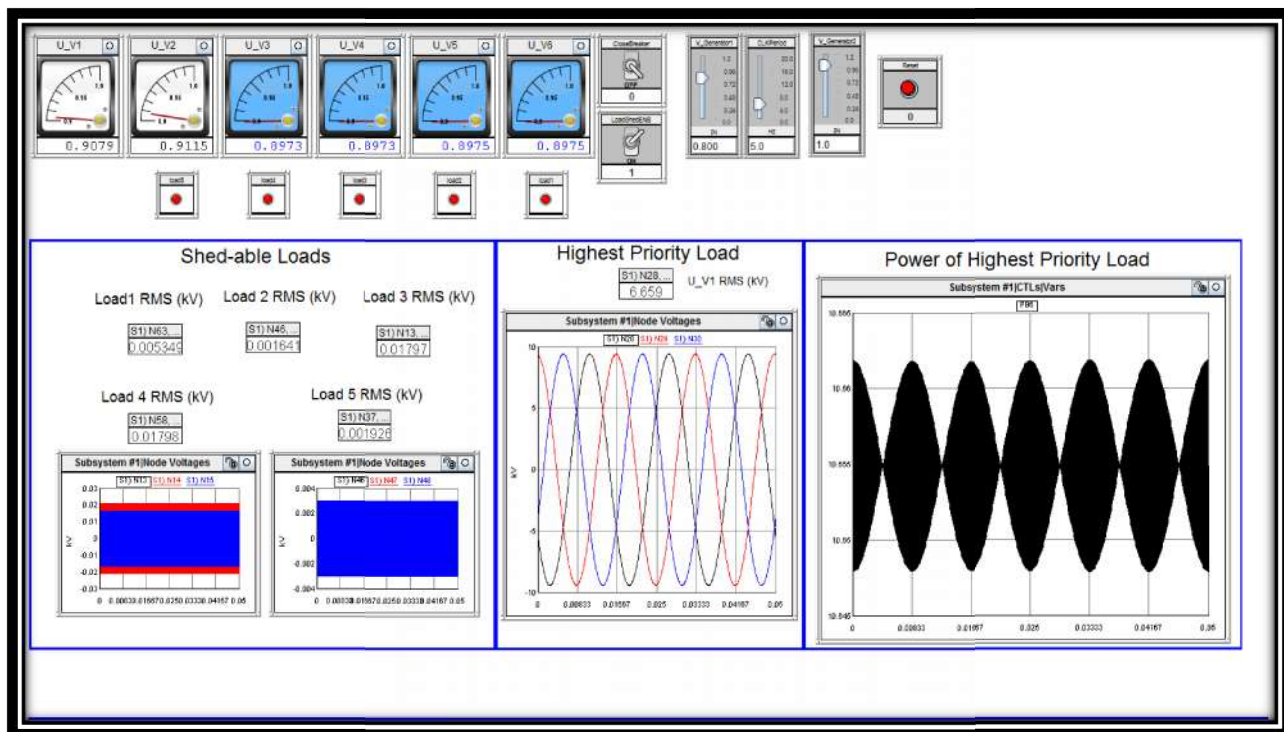


Figure 6.6: Load Shedding Simulation 4 All but Highest Trip

In Figure 6.6 above, the power and voltage is stable at 0.9079 volts per-unit. Although there are some buses that are undervoltage the most critical remains connected within voltage limits.

## 6.2 Chapter 6 Summary

Chapter 6 of this thesis centered around the implementation of a load shedding scheme. One of the issues of the microgrid discussed in chapter was actions to take in certain instances when not being able to supply power to every load in the system. This chapter takes the list of prioritized loads from phase 1 of this project. It then discussed how the load shedding scheme was implemented in Simulink, which was in turn used as a base to implement in RSCAD. This was done for ease of development and to ensure accuracy. The tests results from simulations for Simulink and RSCAD prove that the load shedding scheme



worked in simulation. Figure 4.1 shows a block diagram of just the load shedding controller. A more in depth view on how the controller was designed and the C code implemented is found in Appendices A-C. Specifically the content of the appendices is as follows: Appendix A shows the truth tables for the finite state machine, Appendix B show the next state and output logic diagrams and equations, and Appendix C shows the RSCAD Cbuilder files and the Ccode that goes along with them.

In this microgrid, the load shedding is not the first option to take action. Before any load is shed the system checks to see if there are any batteries that can supply power to the microgrid. In order to use these batteries efficiently there needs to be a battery controller to tell the batteries whether they need to supply power to the microgrid or to charge. To implement this a battery model built and tested. This is discussed more in greater detail in the next few chapters.

## Chapter 7: Battery Controller

### 7.1 Introduction

The purpose of the battery controller is to evaluate the frequency of the system and send a signal to the battery to correct the frequency, as well as help with real power management. If the batteries can't maintain all of the bus voltages at .9 per-unit, then the battery controller sends a signal to the load shedding scheme. The controller is a stackable unit that will communicate to other controllers to ensure they have done everything necessary before changing the state of other batteries [3]. Appendix D shows the coding for the battery controller shown in Figure 7.1. The battery controller needs to be able to perform several tasks. Those tasks include supplying power to the system at the appropriate time or using power from the system to charge. A flow diagram is shown in Figure 7.1.

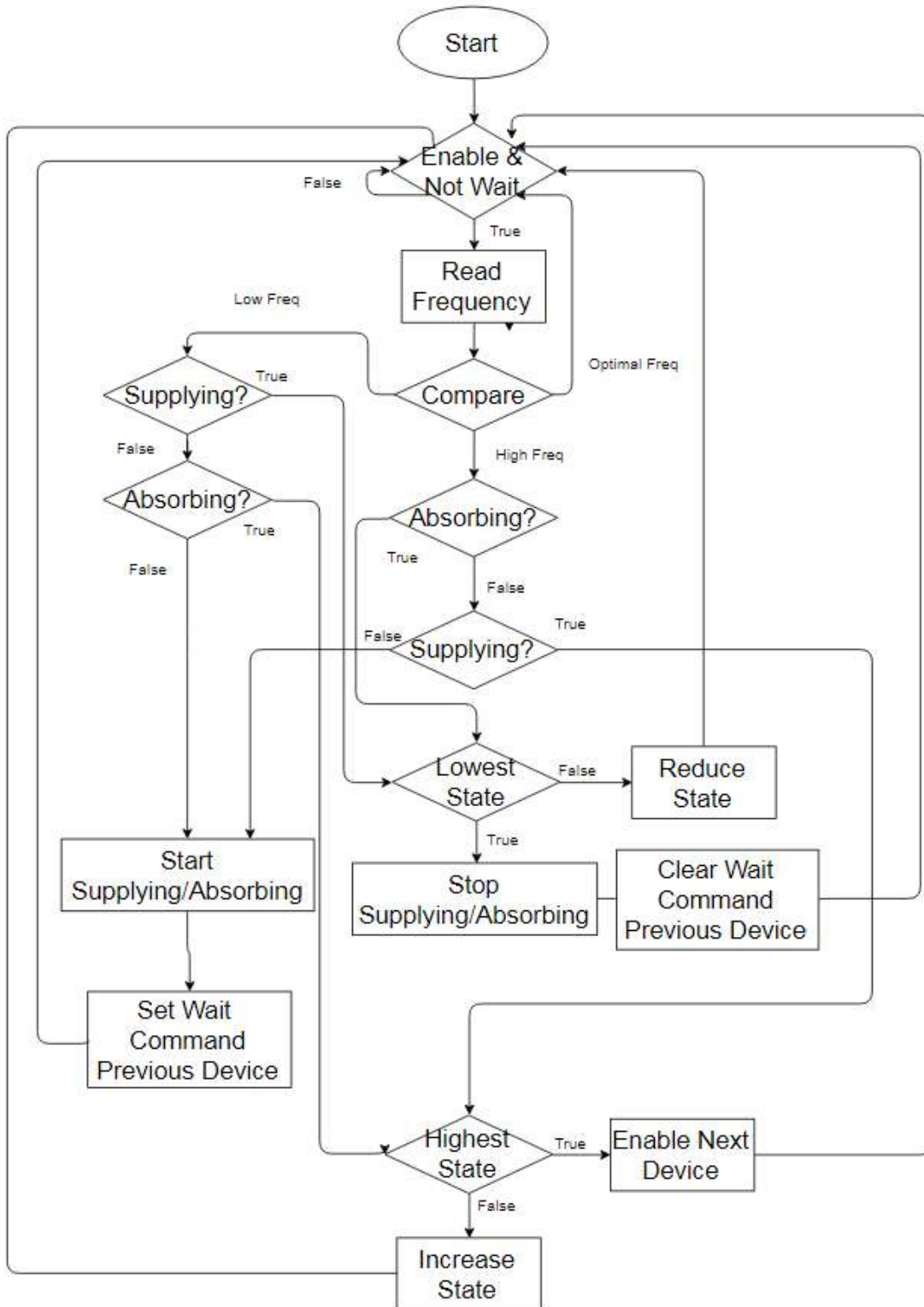


Figure 7.1: Battery Control Flow Diagram

The batteries will react to changes in frequency of the system make a decision based off that. The controller is set up such that it can increase or decrease the amount of power transfer to or from the batteries. The first step with the controller is that it takes frequency measurements from across the system and compares those frequencies to a reference. Based off of those comparisons the battery controller will make one of a few decisions. The first is the frequency higher or lower than the system normal. Then the controller will decide to either start charging the batteries or using the batteries to supply power. Once a battery is in supply mode, the controller will measure voltages across the system to check if one battery is adequate to maintain a stable system. If one battery isn't enough a signal called "Enable" is sent from the battery controller to the second battery to turn on. This model should help to correct frequency drops in the system during peak load times.

## **7.2 Battery Model Introduction**

The main purpose in creating a power system model of the battery is so the battery controller can be tested to see if it operates correctly with the other controls in the microgrid. The battery model used in the project works as an ideal current source with three different settings. A generic ideal current source was used and is shown in Figure 7.7. These settings correspond to the values described later in this chapter. There are three levels of supplying or charging for the batteries, which are low, medium and high. A value of 1 means supply or absorb the least amount of current, a value of 3 means supply or absorb the highest amount of current possible, and a value of 2 is medium level.

The battery controller, which is described in more detail later in this chapter will measure and compare the frequency, and then determine if the batteries should be supplying or absorbing power and at what rate. Once this decision has been made, the battery

controller will send commands to the batteries themselves. The controller will update the commands to the batteries in real time. For example, if the batteries are in supply mode and the controller determines it needs to start absorbing power a new command will be sent to the batteries to change modes.

### 7.3 Battery Controller in RSCAD

The design of the controller with the present microgrid model was done in RSCAD using the CBuilder functionality described in Chapter 6. By using CBuilder a user can create control blocks, like the one shown in Figure 7.2. The control block consists of inputs and outputs that are transmitted to and from code programmed in a C-based coding environment.

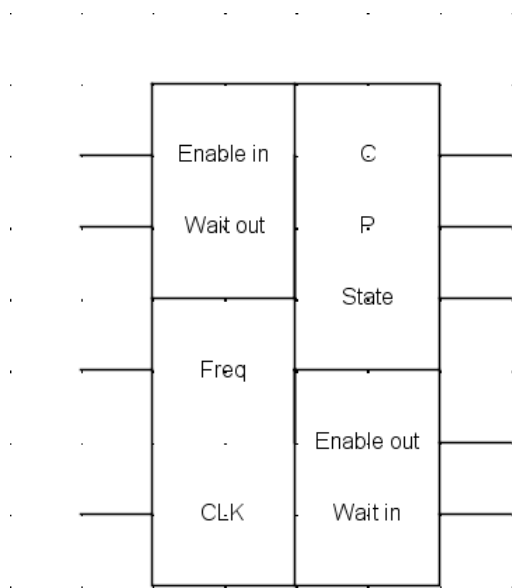


Figure 7.2: Battery Controller

The inputs of the system are shown in Figure 7.3. The “Freq” refers to the frequency input used for comparison. “CLK” is a pulse function with a period set to the user’s preference. The period of the pulse function will designate how often the controller can

make adjustments. “L\_in”, which is “Enable in” in Figure 7.3, is the signal from the previous controller in a predetermined sequence. If “L\_in” is a zero the controller will not change state because the previous controller is working to resolve the problem. This input will not allow multiple batteries to be set to charge or discharge at the same time. If more than one battery is required the batteries are set to go into a charge or discharge state based off of a predetermined sequence. The last input is “W\_in”, which is “Wait in” in Figure 7.3, is used to communicate with the next controller. This input allows the batteries to be turned off sequentially and not simultaneously.

▼ INPUTS			
Freq	REAL	▼	IO Point: Freq (REAL) ▼
CLK	REAL	▼	IO Point: CLK (REAL) ▼
L_in	INT	▼	IO Point: L in (INT) ▼
W_in	INT	▼	IO Point: W in (INT) ▼

Figure 7.3: Controller Inputs

Each of these inputs are points on the controller symbol and are declared by RSCAD to be used in the coding environment. Like the inputs of the system, the outputs are set, as shown in Figure 7.4, to be used in the C-code of the program. “L\_out”, which is “Enable Out” in Figure 7.3, is the output to the next controller and will output zero until the battery is operating at full capacity for charge or discharge. When releasing energy into the system “L\_out” will be ‘1’. If the battery is absorbing energy from the system the output will be ‘-1’. The output ‘State’ of the battery will be an integer value of 1, 2, or 3. This value communicates to the battery how much power the battery can either supply or absorb. A value of 1 means the state is at the lowest power setting, this correlates to the battery charging at the slowest rate possible, or being able to supply a smaller amount of power for the longest amount of time possible. A value of 3 is opposite of this in that the battery will

charge at the fastest rate possible, or supply the largest amount of power for the shortest period of time. Having value of 2 means the batteries supply power or charge at a medium rate, which isn't the highest or lowest amount possible. "P\_out" and "C\_out" are binary values where "P\_out" is a value of '1' when the battery is supplying power to the system. "C\_out" is a value of '1' when the batteries are charging, or absorbing power, from the system. If "P\_out" is a value of 1 then "C\_out" has the value of 0. This means that the batteries can't be supplying and absorbing power at the same time. "W\_out" is the output to the previous controller and is '1' when the previous controller needs to wait. The wait command will be sent whenever the battery is supplying or absorbing the power allowing only one battery to change conditions at a time.











▼ OUTPUTS					
L_out	INT	▼	IO Point: L out (INT)	▼	 
State	INT	▼	IO Point: State (INT)	▼	 
P_out	INT	▼	IO Point: P out (INT)	▼	 
C_out	INT	▼	IO Point: C out (INT)	▼	 
W_out	INT	▼	IO Point: W out (INT)	▼	 

Figure 7.4: Controller Outputs as Assigned in RSCAD

The last of the items used in the coding of the controller, shown in Figure 7.5, are the parameters. These parameters are set when the controller is introduced in the draft portion of the model and can be changed to fit the user's needs. "High\_Freq" is the value at which the controller considers the frequency to be high enough to start charging the batteries. "Low\_Freq" parameter determines the value for the frequency at which the batteries will supply power to the microgrid. The values in Figure 7.6 are the frequency input values of Figure 7.3.





PARAMETERS/COMPUTATIONS			
High_Freq	REAL	Parameter: High Freq	 
Low_Freq	REAL	Parameter: Low Freq	 

Figure 7.5: Controller Parameters

### 7.4 Battery Controller Implementation in RSCAD

The completed controller will be connected to the battery component and the other battery controllers in the system. The stacking of the controller will allow there to be no limitation in the number or rating of batteries that can be added to the system. A two-battery system controller is shown in Figure 7.6. Here “CLK” has a frequency of 1 Hertz so the controller can change states every second as needed. Enable in, or “L\_in”, is set on the first controller to ‘1’ to tell the controller there is no previous controller. “Enable out” and “Wait in” is connected to the second controller. The second controller has “Wait in” set to ‘0’ which indicates, to the first controller, that it is the last controller in the stack. C, P, and State are connected to their corresponding battery in the system.

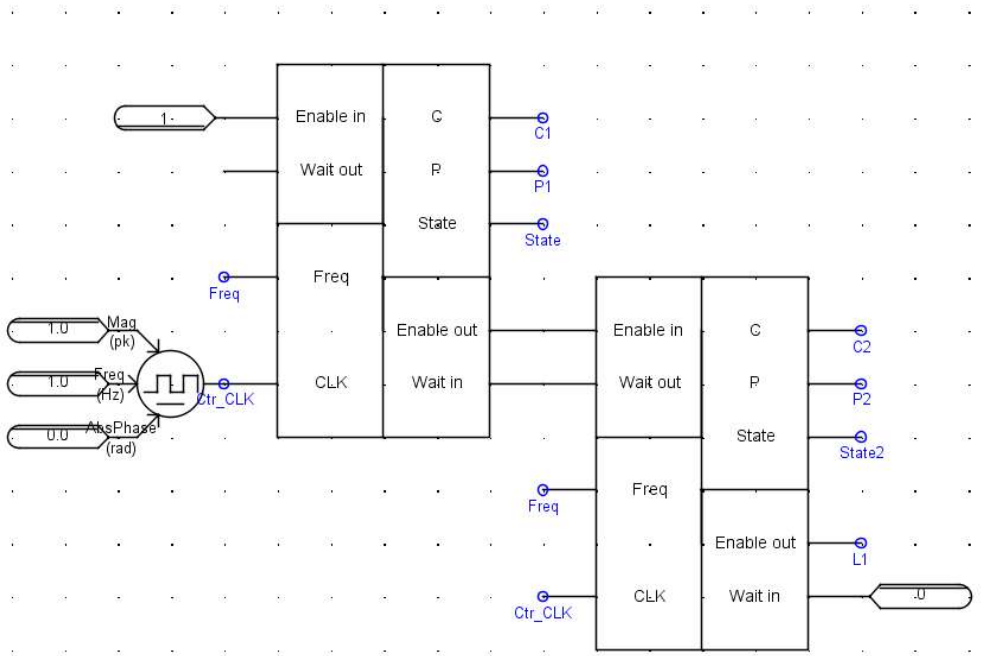


Figure 7.6: 2 Controller System



The last input is “Freq” which is the frequency measurement input from the system. The controller will use the value for “Freq” to make decisions for the batteries. Those decisions are if the battery should be supplying or absorbing power, and at what rate of power transfer.

### 7.5 Battery Power System Model Configuration in RSCAD

The power system model was developed in RSCAD CBuilder using a C-based coding environment. The inputs to the model are shown in Figure 7.7 and Figure 7.8 for the power system model. The input values described in this paragraph are the same values described earlier and in Appendix D. “C\_in” is a binary value that indicates to the battery to absorb energy, or charge the battery. “P\_in” is a binary value which indicates to the battery to supply power. “State” is an integer value from the controller which indicates how much power to absorb or supply to the system. “CLK” is an input of a pulse function which has a frequency of 1 Hz for real time calculations. Figure 7.7 depicted below shows this model.

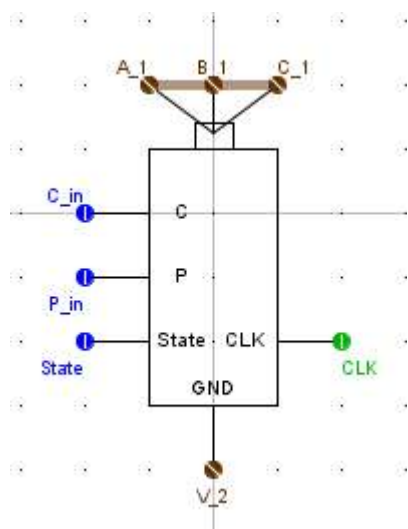


Figure 7.7: Battery Model in RSCAD

▼ INPUTS			
C_in	INT	▼	IO Point: C in (INT) ▼
P_in	INT	▼	IO Point: P in (INT) ▼
State	INT	▼	IO Point: State (INT) ▼
CLK	REAL	▼	IO Point: CLK (REAL) ▼

Figure 7.8: Battery Inputs as Assigned in RSCAD

The nodes for the battery are shown in Figure 7.9. The nodes connect to the rest of the system. The nodes are also used to measure the voltage values from which to calculate the response of the system. “A\_1”, “B\_1”, and “C\_1” are the positive terminals of the battery. “V\_2” is the negative terminal of all three phases and will be ungrounded in the system.

▼ NODES			
V_2	REAL	▼	Node: V 2 ▼
A_1	REAL	▼	Node: A 1 ▼
B_1	REAL	▼	Node: B 1 ▼
C_1	REAL	▼	Node: C 1 ▼

Figure 7.9: Battery Nodes

The parameters shown in Figure 7.10 are set by the user to be employed in the calculations in the code. “Power1”, “Power2”, and “Power3” are power settings of the battery. Depending on the situation the system may require a higher or lower level of power. The controller will determine which power setting the battery will use. “Power1” is the low power setting and “Power3” is the high power setting of the system. “Initial\_Energy” is set by the user and is the starting stored energy of the battery in Joules. “MonI” is a binary value controlled by a switch used to trigger whether internal values will be monitored in runtime or not. When activated, “Current\_Batt”, “Energy\_Stored”, and “Power” can be monitored in Runtime. “Current\_Batt” is the current through one of the phases of the battery. “Energy\_Stored” is the amount of energy in joules remaining in the battery. “Power” is the output power of the battery at any given time in watts. “Max\_Charge” is in units of joules

and refers to the limit of energy that the battery can absorb. “Max\_Discharge” is in units of joules and refers to the limit of energy discharge for the battery. “Max\_Charge” and “Max\_Discharge” is defined by the user for each of the battery units.

























▼ PARAMETERS/COMPUTATIONS					
Name	CHAR	▼	Parameter: Name	▼	 
Power1	REAL	▼	Parameter: Power1	▼	 
Power2	REAL	▼	Parameter: Power2	▼	 
Power3	REAL	▼	Parameter: Power3	▼	 
Initial_Energy	REAL	▼	Parameter: Initial Energy	▼	 
MonI	INT	▼	Parameter: Monitor	▼	 
Current_Batt	CHAR	▼	Parameter: Current Batt	▼	 
Energy_Stored	CHAR	▼	Parameter: Energy Stored	▼	 
Max_Charge	REAL	▼	Parameter: Max Charge	▼	 
Max_Discharge	REAL	▼	Parameter: Max Discharge	▼	 
Power	CHAR	▼	Parameter: Power	▼	 
Base_Power	REAL	▼	Parameter: Base Power	▼	 

Figure 7.10: Battery Parameters Entered by User

When read, the nodes for the power system device will give the voltage at that point. In order to control and measure the current through the nodes, the system uses injections for each of the nodes. IA, IB, and IC are the current injections on the positive terminal of the battery system. IG is the negative terminal connection current for the power system. The sum of IA, IB, IC, and IG should equal to 0.









▼ INJECTIONS					
IA	REAL	▼	Node: A 1	▼	 
IB	REAL	▼	Node: B 1	▼	 
IC	REAL	▼	Node: C 1	▼	 
IG	REAL	▼	Node: V 2	▼	 

Figure 7.11: Battery Current Labels

The variables shown in Figure 7.12 are used in the code and are connected to the monitored values mentioned in Figure 7.10. Setting one of these variables equal to a value will output that value and be monitored when simulating the model. “IMON” outputs the current from phase A to the “Current\_Batt” variable. “En” outputs the energy stored in the

battery at that time and outputs that value to “Energy\_Stored”. “P\_out” outputs the power that the battery is generating to the variable “Power”.

▼ create: GENERIC OUTPUTS			
IMON	REAL	▼	(Current_Batt, "Branch Currents", -10, 110, "A", ...)
En	REAL	▼	(Energy_Stored, "Energy Storage", -10, 110, "J", ...)
P_out	REAL	▼	(Power, "Energy Storage", -10, 110, "W", MonI==1);

Figure 7.12: Monitored Battery Values

## 7.6 Software Development in Simulink

The energy storage in the microgrid system will help to offset load peaks in the system. To achieve this a controller will evaluate the current state of the system and react appropriately. Under conditions where generation exceeds load in the microgrid the system will store energy and at peak times that system will release energy from the batteries to balance the system and avoid load shedding. The energy storage devices, or batteries, will activate one at a time to correct the system. The software used to develop the controller is Matlab Simulink. A finite state machine was chosen to develop the battery controller as shown in Figure 7.13. FH indicates a high frequency condition input, while FL indicates a low frequency condition input. S0 is the initial idle state the batteries are in, which means the batteries are not charging or discharging. This system will constantly be monitoring frequency. If FH, a high frequency condition, is measured then the system will move to state one, which is charging battery 1. Once in S1 the system will compare the frequency again, if it stays high it will move to S2 and start charging battery 2. This process is followed throughout Figure 7.13.

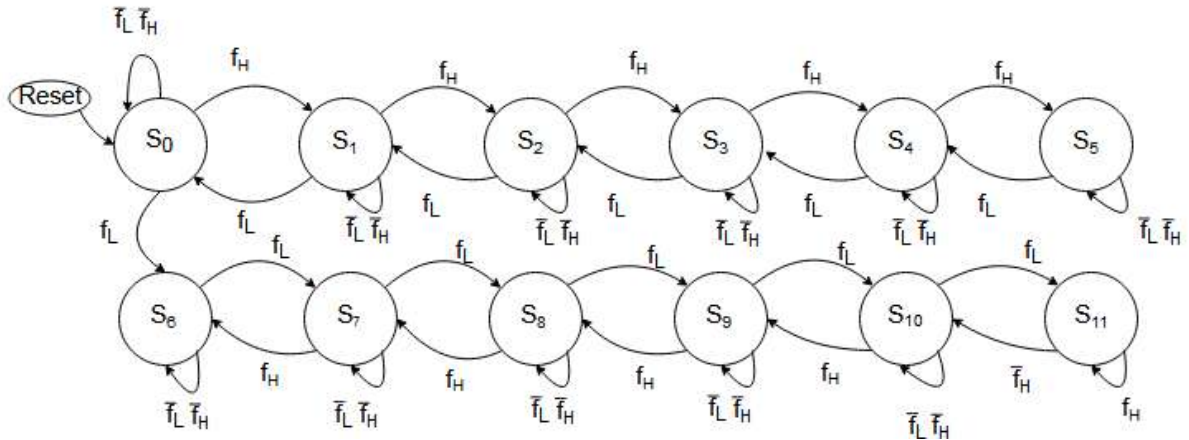


Figure 7.13: State Transition Diagram

The State Transition Diagram shown in Figure 7.13 shows the transition logic of the controller for the energy storage devices. The controller in this example uses five batteries and each battery has three states; on, off, and charging. S<sub>0</sub> indicates all batteries are off, S<sub>1</sub>-S<sub>5</sub> indicates the respective batteries are charging, states S<sub>6</sub>-S<sub>10</sub> indicate the batteries are supplying power, and S<sub>11</sub> is when the batteries have reached their power limit so load needs to be shed. The Boolean output for each of the five batteries in the system are shown in Table 7.1. In the table L is the command to shed load and P is the command for each of the batteries to supply power to the system. The last variable is C and that is to charge the battery when everything is set to a zero value, the batteries will be disconnected. Table 7.1 shows the exact same information as Figure 7.13, but in a tabular format.

State	Outputs										
	$L$	$P_5$	$P_4$	$P_3$	$P_2$	$P_1$	$C_5$	$C_4$	$C_3$	$C_2$	$C_1$
$S_0$	0	0	0	0	0	0	0	0	0	0	0
$S_1$	0	0	0	0	0	0	0	0	0	0	1
$S_2$	0	0	0	0	0	0	0	0	0	1	1
$S_3$	0	0	0	0	0	0	0	0	1	1	1
$S_4$	0	0	0	0	0	0	0	1	1	1	1
$S_5$	0	0	0	0	0	0	1	1	1	1	1
$S_6$	0	0	0	0	0	1	0	0	0	0	0
$S_7$	0	0	0	0	1	1	0	0	0	0	0
$S_8$	0	0	0	1	1	1	0	0	0	0	0
$S_9$	0	0	1	1	1	1	0	0	0	0	0
$S_{10}$	0	1	1	1	1	1	0	0	0	0	0
$S_{11}$	1	1	1	1	1	1	0	0	0	0	0

Table 7.1: Output Transition Table

### 7.7 Implementation in Simulink

Figure 7.14 shows a block diagram of the battery controller built in Matlab Simulink. This figure was built from the data in Figure 7.13 and Table 7.1. The inputs  $f_L$  and  $f_H$  are constants and simulate frequency in the system being low or high. Display  $C_1$ - $C_5$  and  $P_1$ - $P_5$  will show a value of 0 or 1. For  $C_1$ - $C_5$  a value of 1 means the battery is in a charging state while a 0 means it is not charging. For  $P_1$ - $P_5$  a value of 1 indicates the batteries are discharging (supplying power) and 0 indicates no power is being supplied. The battery controller was implemented to be used in conjugation with a load shedding scheme, but in the case of looking strictly at battery control it is not needed.

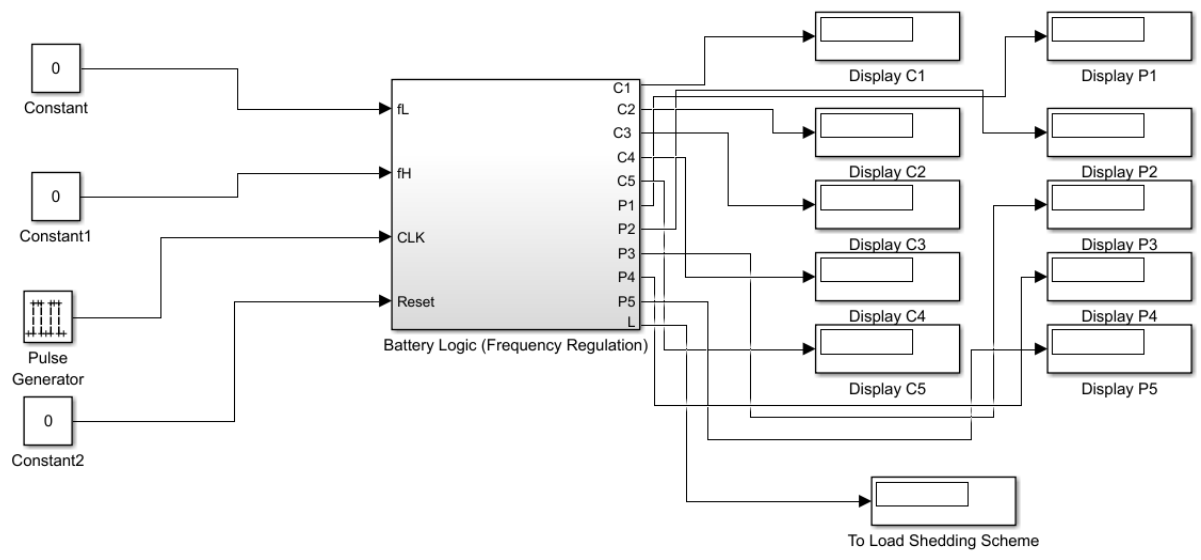


Figure 7.14: Battery Controller in Simulink

**7.8 Simulation of Battery Controller in Simulink**

Table 7.2 and Table 7.3 describe the battery controller response if there is a sustained low frequency and high frequency input. Charging or discharging a battery will reduce or increase the frequency level of the system respectively. For the cases shown in Tables 7.2 and 7.3 the batteries were not programmed to affect system frequency, but rather to show that the battery converters can be stopped or started sequentially. In the Tables below we see that at each time step of .2 ms the next battery will either start charging or discharging.

time (ms)	Ouput:C		Output:P	
0	C1	0	P1	1
0.2	C2	0	P2	1
0.4	C3	0	P3	1
0.6	C4	0	P4	1
0.8	C5	0	P5	1

Table 7.2: Battery Controller Simulation Low Frequency Input

time (ms)	Ouput:C		Output:P	
0	C1	1	P1	0
0.2	C2	1	P2	0
0.4	C3	1	P3	0
0.6	C4	1	P4	0
0.8	C5	1	P5	0

Table 7.3: Battery Controller Simulation High Frequency Input

## 7.9 Chapter 7 Summary

The main point of this chapter was to design a battery controller that could sense a low or high frequency condition and then start charging or discharging batteries in a set sequence over time. This task was accomplished and can be seen in Figures 7.2 and 7.3. As discussed in the results section charging or discharging the batteries didn't affect the frequency since it wasn't part of the full system.

Another key point discusses the design of the controller within the microgrid model. This was done in RSCAD using the CBuilder functionality and in Matlab Simulink. Some of this issues addressed in this chapter include, "Should the battery be charging or supplying power?" Then once in this charging or supplying state; "What should be the rate of charge?" The design of the controller was based around those questions.

This chapter also gives a very simple overview of an energy storage model. The objective of this model is to add the batteries and their energy storage control in the present microgrid model. This model will use an ideal current source as the subject of application for this model. These batteries have three power settings listed which are used to develop the characteristics of the supplying and absorbing power abilities of the batteries.



The results shown in Figures 7.2 and 7.3 are programmed with a constant low or high frequency input changed by the operator. The simulations show the converter can be stopped or started. The next step would be to program the batteries so a more realistic simulation could be performed to show how they affect system frequency, and actually include in the microgrid simulation. This is discussed more in the future work section of this thesis.

## **Chapter 8: Diesel to Natural Gas Conversion Kits**

### **8.0 Introduction to Diesel to Natural Gas Conversion**

There are currently 18 backup generators distributed in the microgrid area, mostly at the hospitals. Together all the generators can provide approximately 14.75 MW. This generation would add a significant amount of power to the current generation available [1]. By using the power from the generators, it would be possible to keep critical loads from shedding. Currently, all of the hospital backup generators are setup to feed only their respective facilities. There is no existing connection or method, for them to synchronize with the microgrid and transfer their output power to the local distribution system.

The hospital generators are currently diesel-fueled. Because of environmental restrictions, the use of diesel generators is limited to short durations during emergencies. To allow more use of the generators, environmental standards require a more eco-friendly fuel source, such as natural gas. After the conversion, the generators can be re-commissioned and approved for longer run times. It is anticipated that the conversion will allow the generators to offset the daily peak load and, therefore, reduce peak load rise of energy prices.

Being able to use the excess power from the hospital generators would affect the microgrid in two ways. The first is the power supplied has the potential to keep a load from having to be shed, assuming the natural gas pumping stations have power in a regional blackout. The second is the power could be used to charge the batteries.

### **8.1 Diesel to Natural Gas Conversion Kit**

A company called GFS Corp is experienced in converting commercial grade diesel generators to a hybrid mix of diesel and natural gas. Most of its projects provide the

customer's existing generators with peak shaving capabilities. GFS Corp is able to perform this conversion for any type of diesel generator.

## **8.2 Conversion Kit Specifications**

According to a phone interview performed with a representative of the company, the conversion process will provide the following benefits to the newly converted engines:

- 1) Non-invasive retrofit. The existing engines will not need to be disassembled for any part of the conversion process. [3]
- 2) The engines will start up on diesel, then convert to a hybrid operation of diesel and natural gas. [3]
- 3) Maximum operation of 70% natural gas to 30% diesel. [3]
- 4) Conversion for one unit usually takes two days (one day to install and another day to re-commission). [3]
- 5) No added power loss after the conversion process. [4]
- 6) Extends generator run time using pipelined natural gas [4]

The information described in above would help the microgrid in several ways. A few of them include providing more power which in turn could keep load from being shed, charging the batteries, and a cleaner burning fuel source. Another perk of these kits is they are installed in a minimal amount of time which lessens impact on the area. A key point for these kits is that there can be benefits beyond the microgrid.

## **8.3 Chapter 8 Summary**

This chapter has discussed some basic research done on diesel to natural gas conversion kits. In detail it goes over the conversion kit specifications and some of the benefits it could have for the microgrid. Not only is it cleaner burning, which result in longer

run times during normal operations, but there is also no power loss after the conversion. The next step would be the implementation of a model of these converted generators in PowerWorld and RSCAD. The future work section goes over this in more detail.

## Chapter 9: Automation Controller

### 9.0 Automation Controller Introduction

Automation control is the use of various control systems for operating equipment such as machinery, processes in factories, and other applications and vehicles with minimal or reduced human intervention [1]. To introduce more control the microgrid, an automation controller was constructed by combining hardware shown in Figure 9.1, automation logic, such as the load-shedding and battery control logic used in this project, and a Human Machine Interface (HMI). The automation logic in the protective relays decides when action is necessary to improve the state of the system. The sensors are placed throughout the microgrid to collect and transmit data to the real-time automation controller (RTAC) communications processor for control action and for the information to be displayed through an HMI. The operator uses the HMI to analyze and interact with the system.

### 9.1 Hardware Development

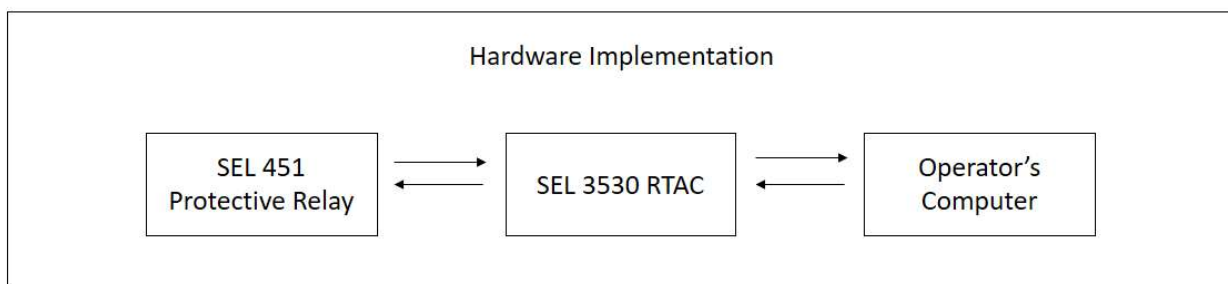


Figure 9.1: Hardware Implementation Scheme

As shown in Figure 9.1, the data flow within the hardware is bidirectional. The currents, voltages, and frequencies from the microgrid are measured by the protective relays and sent to the RTAC to make informed commands, if action is necessary. The operator's computer, far right on Figure 9.1, monitors the process through the HMI and has the ability to intervene and reconfigure device settings.

## 9.2 Software Configuration

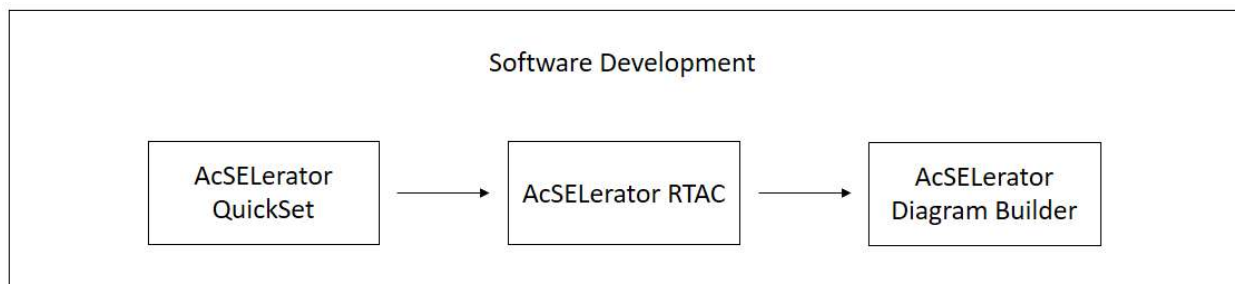


Figure 9.2: Software Implementation Scheme

In Figure 9.2, vendor settings software programs were used to configure and commission the RTAC and implement load-shedding in relays that are already present for system protection. From the settings configured in the RTAC, the HMI can be customized for the system. Appendix F shows the software and hardware used in automation controller development. Once the software is configured and the HMI is developed it can be used to monitor values from with the hardware described in Figure 9.1, and control things such as breakers in the system.

## 9.3 Protection Relay

When setting the automation controller, reference tags are used to transmit and receive desired outcomes in the relay. Tags act as pointers to program commands within the relay. Based on assigned tag values and automation logic, the automation controller either performs actions on the microgrid as shown in Table 9.1 or turns on corresponding LED alarm lights as shown in Figure 9.3.

Setting	Description	Range	Value
BK1MTR	Breaker 1 Manual Trip (SELogic)	Valid range = The legal operators: AND OR NOT R_TRIG F_TRIG	OC1 OR PB8_PUL

Table 9.1: Example of SEL Relay Breaker Trip Tag

In Figure 9.4, *value* is the tag monitored by the relay to trigger the assigned tag in *settings*. In this case, the setting is the breaker manual trip action. OC1 OR PB8\_PUL is a pushbutton that triggers the manual trip when pressed.

```
#ALARM ELEMENTS
PSV34 := HALARM OR SALARM
```

Figure 9.3: Example of SEL Relay LED Alarm Light Protection

Figure 9.3 shows an example of protection logic in the relay written in SELogic. PSV34 is a protection variable tag that the relay monitors for HALARM OR SALARM which are alarms for hardware failure or settings error. PSV34 points to an LED alarm light tag so the relay will turn on an LED light in response to hardware or settings failure.

The relays also have alias tags to transmit information to the RTAC that will modify the state of the visuals in the HMI. All of this is being done so that the automation controller can be used to interact with the microgrid in terms of battery control and load shedding.

Setting	Description	Range	Value
SITM1	SER Points and Aliases, Point 1	Device Word Element, Alias Name, Asserted Text, DeAsserted Text, HMI Alarm	50P1, 50P1, ASSERTED, DEASSERTED, Y

Table 9.2: Example of Establishing Alias Relay Tag

Table 9.3 shows a case where an alias tag is used as a value for an LED alarm light, and if asserted, the HMI will display a triggered LED light.

Setting	Description	Range	Value
PB6_LED	Pushbutton LED 6 (SELogic)	Valid range = The legal operators: AND OR NOT R_TRIG F_TRIG	50P1

Table 9. 3: Example of HMI Relay Trigger Tag

In Table 9.3, the defined alias overcurrent element tag 50P1 will trigger the PB6\_LED assigned LED light on the HMI.

#### 9.4 GOOSE Protocol Communications

The communications protocol used in this project is called General Object-Oriented Substation Events (GOOSE) protocol which was chosen because it was the best option available at the time to work with the RTDS. This protocol is described in the International



Electrotechnical Commission (IEC) 61850 standard for substations which enables the integration of all protection, control, measurement and monitoring functions [6]. GOOSE communication is a publisher-subscriber protocol that contains publisher and subscriber Intelligent Electronic Devices (IEDs). Publisher IEDs send data packets to the system while subscriber IEDs read data packets published by a given IED.

To configure the publications and subscriptions, every IED must have a corresponding IED Capability Description (ICD) file. The ICD file provides a description of the items supported by an IED. ICD files for the devices that are to be interconnected on the same network are merged into the Substation Configuration Description (SCD) file. The SCD file is an XML based text file that describes the IEC-61850 compliant devices comprising a substation. An SCD file is often generated by an editor program which takes component capability files (.icd files) and merges them into the .scd file. IEC standard 61850-6 defines the configuration description language for communication in electrical substations related to IEDs [6]. This is all done so that the automation controller can successfully communicate with the microgrid model developed in RSCAD.

## **9.5 The Human Machine Interface**

The Human Machine Interface (HMI) is the platform for cognition and communication between human and machine, and is the approach for information transmission [8]. The HMI facilitates this communication through the use of visuals designed to effectively convey the state of the running system.

The HMI also allows the operator to control a system through the use of interactive visual controls, such as buttons, switches, and dials. In order for the operator to make efficient use of these controls, it is necessary for the operator to quickly and accurately

understand the state of the system. The status of the substation can be conveyed to the operator in different ways. Colors, shapes, values and labels can all be used to transfer data to the operator. Figure 9.4 shows an example of how an HMI could be designed for a single breaker. When the breaker is open, the box turns red, and after an alarm is triggered, it flashes. The buttons below the box allow the operator to trip or close the breaker.



Figure 9.4: HMI for One Breaker

Bus voltages and line currents may also be added in the HMI to allow the user to monitor the transmission lines individually. Figure 9.5 is an example of recently labeled values giving information to the operator, for the operator to interpret.

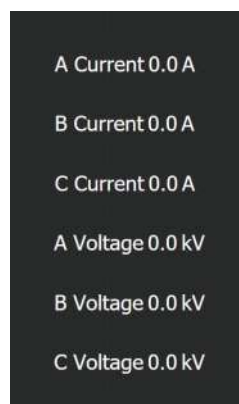


Figure 9.5: Bus Voltages and Line Currents for One Relay

## **9.6 Chapter 9 Summary**

Chapter 9 has discussed the purpose of an automation controller, which is to automate various control systems for operating equipment such as machinery, processes in factories, and other applications and vehicles with minimal or reduced human intervention [1]. Chapter discusses the steps needed in setting up an automation controller. Those steps include: hardware development, software development, HMI, and communication protocols. Chapter 10 goes over in detail future work for the automation controller.

## **Chapter 10 Summary of Accomplished Work and Future Work**

### **10.0 Summary of Accomplished Work**

As this research project ended several valuable milestones were achieved in the pursuit of improving the microgrid. A major problem that often occurs in a microgrid is not being able to meet the demand of the system. This research project successfully designed and tested a load shedding scheme in RSCAD to deal with this issue. The load shedding scheme works by following a few simple steps. The first is to measure the voltage at all the busses throughout the microgrid and check those values to see if any of them fall below the tripping value. If any of these values fall below the tripping value of 0.9 per unit the load shedding scheme will select the lowest priority load and shed it. Currently load is only being shed. When more power is available an operator will manually have to bring the loads back online. Batteries as well as a battery controller have been added to the system to help keep loads from being shed when possible, but no combined simulations of load shedding and battery control were performed. This controller included commands such as when to charge, when to discharge, and how much to discharge. By doing those things a large portion of the microgrid area will continue to have power in the event of a blackout. This would be able to power critical loads the user selects which could include: hospitals, police stations, and government buildings. In conclusion the benefit gained from implementing a load shedding scheme and battery controller into the microgrid is the ability to keep the power on for critical loads during outages.

## 10.1 Future Work: Transfer of Automation Logic to RTAC

The load shedding controller is designed to function independently and would be fully realized if the control logic was taken from the RSCAD software and ported into an automation controller development environment.

## 10.2 Future Work Time Based Load Shedding

The load-shedding controller will trigger the opening of breakers in a sequential manner. As implemented in simulation the only way to change the sequence is to alter the draft file's outputs from the controller. This would allow dynamic bus monitoring throughout the year. With the change of seasons and temperature the hydro generation plus varies greatly as seen in Figure 10.1 [1]. There are various values of needed power during the course of the year and each day. The ability to change which loads are shed and what is critical to the stability of the microgrid. It should be noted that the system rarely be in island mode. In the event that happens the duration will be unknown.

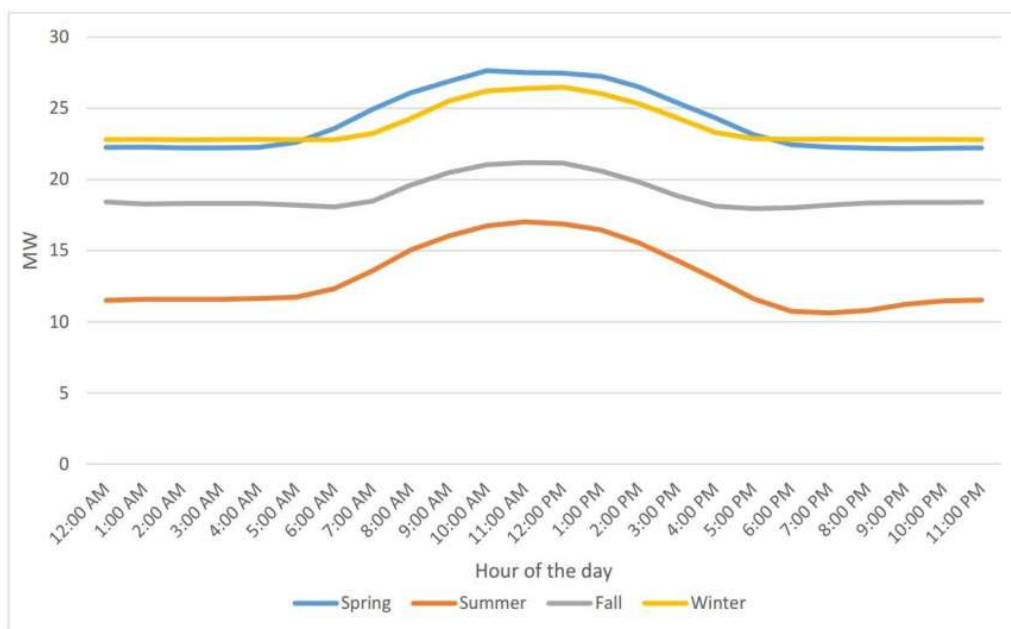


Figure 10.1: Combined Solar and Hydro Average Generation Output Profile Over a 24Hr Period in 4 Seasons [1]

### **10.3 Future Work Fully Implementing Load Shedding in RTDS Hardware in Loop Simulation**

Moving the loads in RSCAD can be done by using the GTNET-SKT communication protocol functions to pass a date and time into the draft file. The GTNET-SKT protocol uses a LAN/WAN connection with the TCP or UDP sockets on the processor cards. With the proper configuration of the GTNET card(s) it should be possible to pass a date and time to the load-shedding controller. A user would then be able to write additional code to take the information from the GTNET card(s) and assign different load priorities during specific times of the day and/or or year.

### **10.4 Battery Control Future Work**

A preliminary battery controller was designed and presented in Chapter 7. It was attempted to go beyond the scope of the project and create a battery controller in RSCAD, but the project was over before it could be finished.

The work that was completed is shown in Appendices D-F. In the RSCAD model when the battery controller and battery model were tested the battery controller measured the frequency performed a comparison, determined how much power should be supplied and absorbed, and then sent the command to the battery. The battery in turn supplied the correct amount of power, but introduced a transient into the system that never dampened out. The next step for this would be to determine if the battery or the controller is introducing the transient into the system and fix it. Once this is completed the battery control could be implemented in the microgrid with the load shedding scheme.

### **10.5 Future Work on Diesel to Natural Gas Conversion**

The next step has two parts. The first would be to perform a cost-benefit analysis of converting the diesel generators to the hybrid diesel/natural gas generator. This would be done to see if the benefits are great enough to justify the expense to install them. Step 2 would be performed if the kits were installed, which is to model the converted generators in RSCAD. Once modeled in RSCAD, a stability and transient study could be performed to see how the generators interact with the rest of the system and if they improve the quality of the microgrid.

### **10.6 Future Work Communication Between an RTDS and Automation Controller**

An RTDS can simulate complex networks that have been developed in the simulation software RSCAD. A GTNET card was used to interface the RTDS to the relay over a Local Area Network (LAN) connection using GOOSE protocol [9]. The work in Chapter 9 discussed creating an automation controller, but not the communication between an RTDS model and the automation controller.

The SCD editor program within the GTNET-GSE function block facilitates the creation of the SCD file. To configure communication from the RTDS to the protective relay, outputs were added to the SCD editor. In Figure 10.2, both integer values and binary statuses were outputted to monitor and control the breakers, bus voltages, and breaker currents.

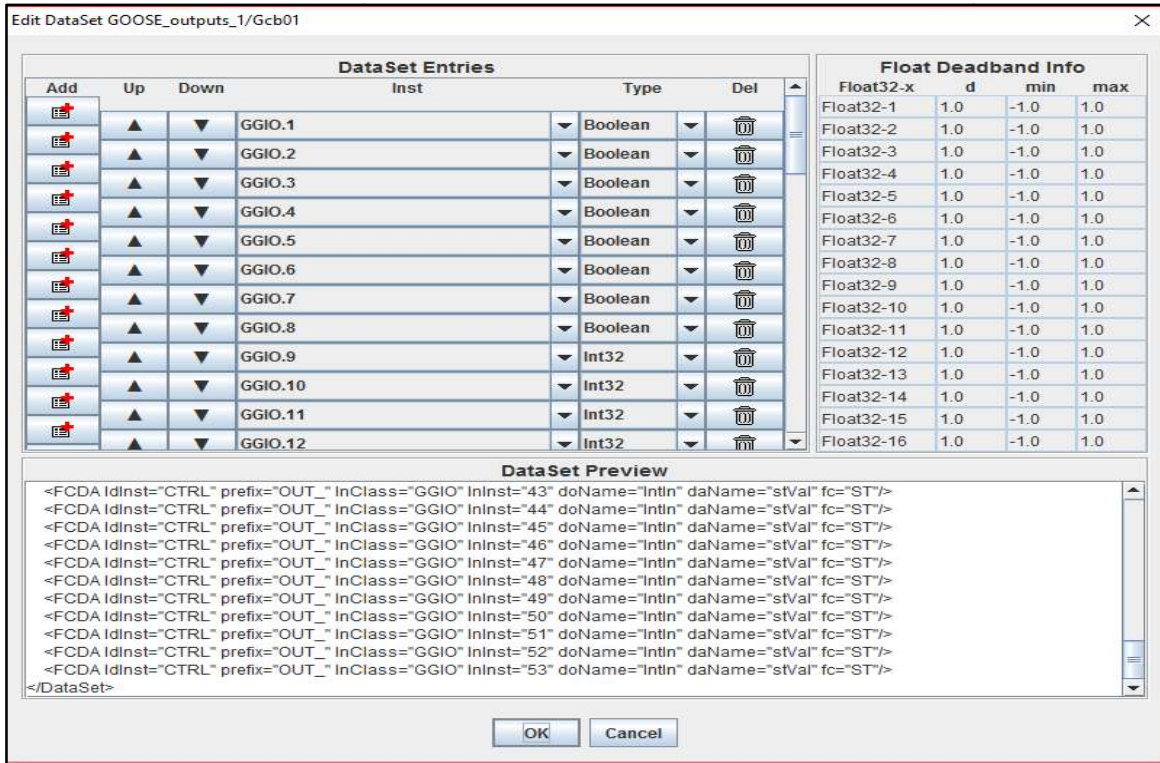


Figure 10.2: Editing Outputs in the SCD file

It is necessary that the parameters within the GTNET-GSE function block are edited to link information within the simulation to those outputs, as shown in Figure 10.3.



_rtds_GTNET_GSE_v5.def						
RX/TX 1 Output Signal Names/Types			RX/TX 1 Input Signal Names/Types			
Output Deadband Parameters			RX/TX 1 Output Retransmit Curve			
CONFIGURATION			GOOSE Configuration			
Name	Description	Value	Unit	Min	Max	
IED101T	Output 1 Type	BOOL		0	12	
nIED101	Output 1 Signal Name	BRK1		0	0	
IED101B	Output 1 Boolean bitmap bit num (32..1)	1		1	32	
IED102T	Output 2 Type	BOOL		0	12	
nIED102	Output 2 Signal Name	BRK2		0	0	
IED102B	Output 2 Boolean bitmap bit num (32..1)	1		1	32	
IED103T	Output 3 Type	BOOL		0	12	
nIED103	Output 3 Signal Name	BRK3		0	0	
IED103B	Output 3 Boolean bitmap bit num (32..1)	1		1	32	
IED104T	Output 4 Type	BOOL		0	12	

Figure 10.3: GTNET-GSE Output Parameters

The work described in this section is to determine how the automation controller communicates with the microgrid model built in RSCAD.

### 10.7 Future Work Establishing GOOSE Protocol Communications for the RTAC

Designing and configuring SEL devices in IEC 61850 installations are possible by using AcSELeRator Architect Software. This software provides a means of configuring and documenting the IEC 61850 communications settings between vendor-agnostic devices [10].

Upon importing the SCD file extracted from RSCAD, GOOSE communication can be made accessible to the RTAC by packaging the data as directed by the IEC-61850 standard as shown in Figure 10.4.

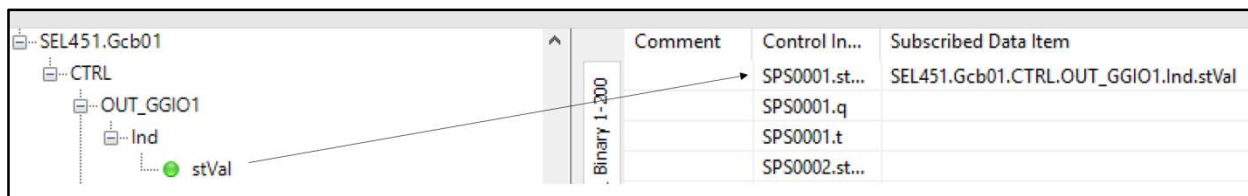


Figure 10.4: Establishing GOOSE Protocol for the Relay

The AcSELERator RTAC software configures the automation controller by uploading project files through a connected ethernet cable. GOOSE protocol is established in the RTAC once an architect file is imported into the project file. A virtual data map must then be created using tags to point to command settings to transmit and receive data using GOOSE protocol.

## References

- [1] Penkey, P. K. (2016). Critical Load Serving Capability by Microgrid Operation (Master's thesis, University of Idaho) (p. 32 to 49). ProQuest Dissertations Publishing.
- [2] UET.(n.d). Uni.System Grid-Scale Energy Storage Solution. Retrieved June 20<sup>th</sup>, 2017 from [http://www.uettechnologies.com/images/product/UET\\_UniSystem\\_Product\\_Sheet\\_reduced.pdf](http://www.uettechnologies.com/images/product/UET_UniSystem_Product_Sheet_reduced.pdf)
- [3] P. (2012, May 4). Bi-Fuel Enters the Digital Age... Introducing the EVO-SP® System. Retrieved October 02, 2017, from <http://www.gfs-corp.com/industry.php/bifuel/>
- [4] GFS Corp. (n.d.). Natural Gas Conversions for High Horsepower Diesel Engines Offering Substantial Fuel Cost Savings and Cleaner Emissions. Retrieved February 09, 2017, from <http://www.gfs-corp.com>
- [5] Bennett, S. (1993). A History of Control Engineering 1930-1955. London: Peter Peregrinus Ltd.
- [6] Fernandes, Chilton, et al. "Testing of Goose Protocol of IEC61850 Standard in Protection IED." *International Journal of Computer Applications*, vol. 93, no. 16, 2014, pp. 30–35., doi:10.5120/16301-6112.

[7] Xuesong, Z., Liyang, Y., & Youjie, M. (2014). An Overview of Micro-Grid. *Applied Mechanics and Materials*, 552, 1-6.

[8] Gong, Chao. "Human-Machine Interface: Design Principles of Visual Information in Human-Machine Interface Design." *Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC '09. International Conference On*, vol. 2, 2009, pp. 262–265.

[9] RTDS Technologies. Real Time Digital Simulation for the Power Industry: Manual Set. Winnipeg, MB

[10] AcSELERator Architect Software. (n.d.). Retrieved May 18, 2017, from <https://selinc.com/products/5032/>

[11] Mahmoud, M., Hussain, S. A., & Abido, M. (2014). Modeling and control of microgrid: An overview. *Journal of the Franklin Institute*, 351(5), 2822-2859.

doi:10.1016/j.jfranklin.2014.01.016

[12] Bounghook, C., Heechul, K., Mussab, A. M., & Seeley, N. C. (2008). The Application of a Redundant Load-Shedding System for Islanded Power Plants. *35th Annual Western Protective Relay Conference*, 1-10.

- [13] Manson, S., Zweigle, G., & Yedidi, V. (2014). Case Study: An Adaptive Underfrequency Load-Shedding System. *Industry Applications, IEEE Transactions on*, 50(3), 1659-1667.
- [14] Kroposki, Lasseter, Ise, Morozumi, Papatlianassiou, & Hatziargyriou. (2008). Making microgrids work. *Power and Energy Magazine, IEEE*, 6(3), IEEE Power and Energy Magazine, May-June 2008, Vol.6(3).
- [15] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, 'Microgrids,' *IEEE Power Energy Mag.*, vol. 5, no. 4, pp. 78–94, 2007.
- [16] S. Parhizi, H. Lotfi, A. Khodaei and S. Bahramirad, 'State of the Art in Research on Microgrids: A Review,' in *IEEE Access*, vol. 3, no. , pp. 890-925, 2015. doi: 10.1109/ACCESS.2015.2443119
- [17] B. Kroposki, R. Lasseter, T. Ise, S. Morozumi, S. Papathanassiou, and N. Hatziargyriou, 'Making microgrids work,' *IEEE Power Energy Mag.*, vol. 6, no. 3, pp. 40–53, 2008.
- [18] *IEEE Guide for the Selection and Sizing of Batteries for Uninterruptible Power Systems*. Place of Publication Not Identified, IEEE, 1995.
- [19] Nazaripouya, H., Wang, Y., Chu, P., Pota, H. R., & Gadh, R. (2015). Optimal sizing and placement of battery energy storage in distribution system based on solar size for

voltage regulation. *2015 IEEE Power & Energy Society General Meeting*.

doi:10.1109/pesgm.2015.7286059

[20] Thipse, S., Kavathekar, K., Rairikar, S., Tyagi, A. et al., "*Development of Environment Friendly Diesel-CNG Dual Fuel Engine for Heavy Duty Vehicle Application in India*," SAE Technical Paper 2013-26-0015, 2013, <https://doi.org/10.4271/2013-26-0015>.

[21] Studies of performance and emission characteristics of compressed natural gas fuelled S.I. engine and developing CNG conversion kit. (2013). *IOSR Journal of Mechanical and Civil Engineering*, 9(4), 23-29. doi:10.9790/1684-0942329

[22] Kolluri, S. V., Ramamurthy, J. R., Wong, S. M., Peterson, M., Yu, P., & Chander, M. R. (2015). Relay-based undervoltage load shedding scheme for Entergys Western Region. *2015 IEEE Power & Energy Society General Meeting*. doi:10.1109/pesgm.2015.7285651

[23] Jensen, S. (2006, December 1). Converting Diesel Engines to Dual Fuel The Pros and Cons of Common Engine Types. Retrieved from

<https://www.energyconversions.com/whitepaperdualfuelengines>

[24] Ieee Pes Task Force on Real-Time Simulation of Power Energy Systems. "Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis." *Power and Energy Technology Systems Journal, IEEE*, vol. 2, 2015, pp. 63–73.

### Appendix A: Truth Tables for Finite State Machine

The truth table is a part of the proper design of a Finite State Machine. The table is used to map the State, Next State and Output logic. The input U\_V signifies a signal from the RSCAD simulation model that will initiate the load shedding process. The input B\_C is a signal from an external source that will send a signal to close a relay of the latest open load.

Current State	Inputs		Next State
S	U_V	B_C	S*
S0	0	0	S0
S0	0	1	S0
S0	1	0	S1
S0	1	1	S1
S1	0	0	S1
S1	0	1	S0
S1	1	0	S2
S1	1	1	S2
S2	0	0	S2
S2	0	1	S1
S2	1	0	S3
S2	1	1	S3
S3	0	0	S3
S3	0	1	S2
S3	1	0	S4
S3	1	1	S4
S4	0	0	S4
S4	0	1	S3
S4	1	0	S5
S4	1	1	S5
S5	0	0	S5
S5	0	1	S4
S5	1	0	S5
S5	1	1	S5

Table A. 1: State Transition Table With Encoding

<b>Current State</b>	<b>Outputs</b>				
S	L5	L4	L3	L2	L1
S0	0	0	0	0	0
S1	0	0	0	0	1
S2	0	0	0	1	1
S3	0	0	1	1	1
S4	0	1	1	1	1
S5	1	1	1	1	1

Table A. 2: Output Table With Encoding

The outputs are integer that values will be passed to the breaker control relays in the RSCAD system.



## Appendix B: Next State and Output Logic Diagrams

Appendix B is a collection of the FSM logic diagrams of the Boolean logic used in implementing the load shedding scheme.

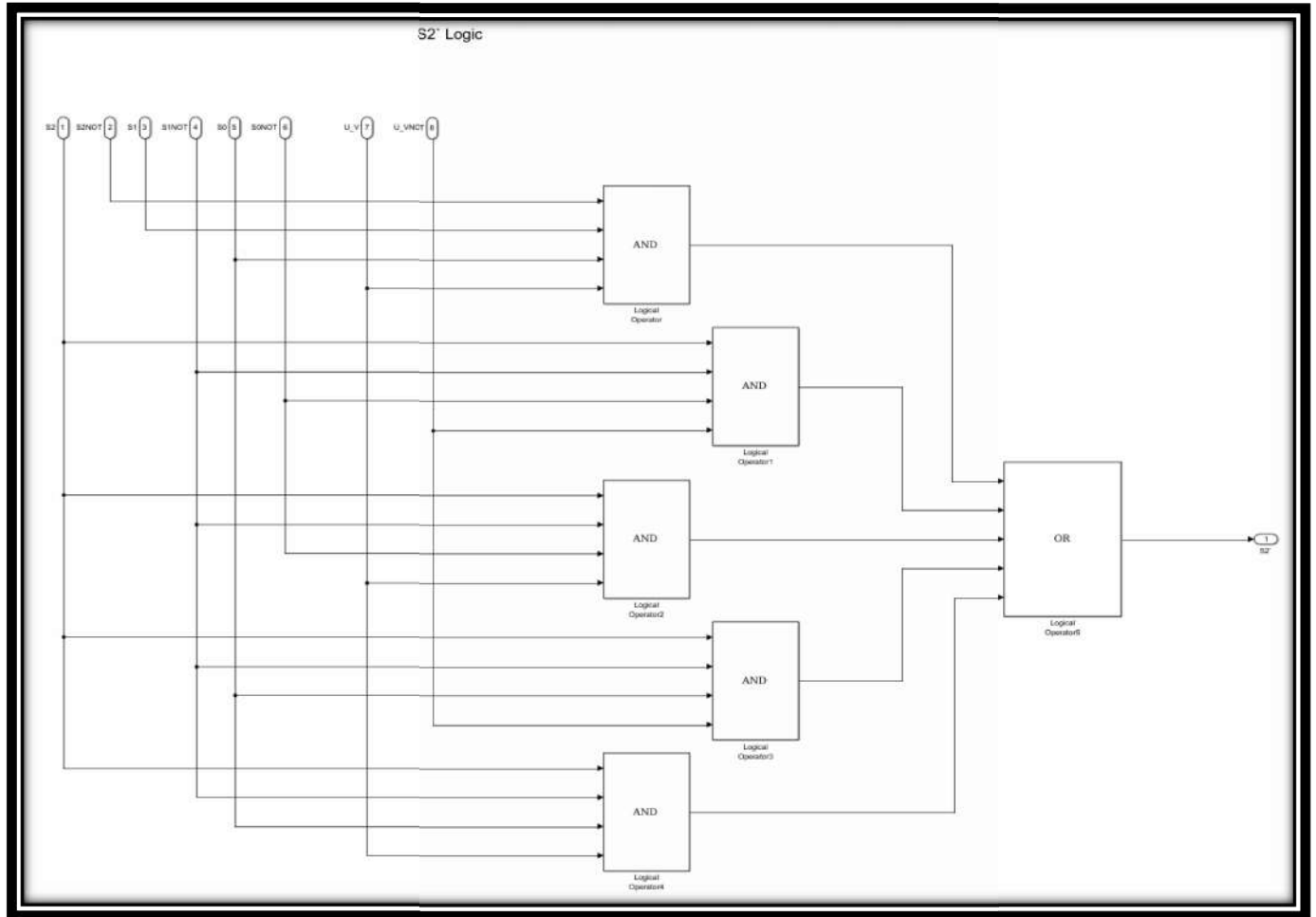


Figure B.1: S2 Logic

$$S_2' = \overline{S_2}S_1S_0U\_V + S_2\overline{S_1}\overline{S_0}\overline{U\_V} + S_2\overline{S_1}\overline{S_0}U\_V + S_2\overline{S_1}S_0\overline{U\_V} + S_2\overline{S_1}S_0U\_V$$

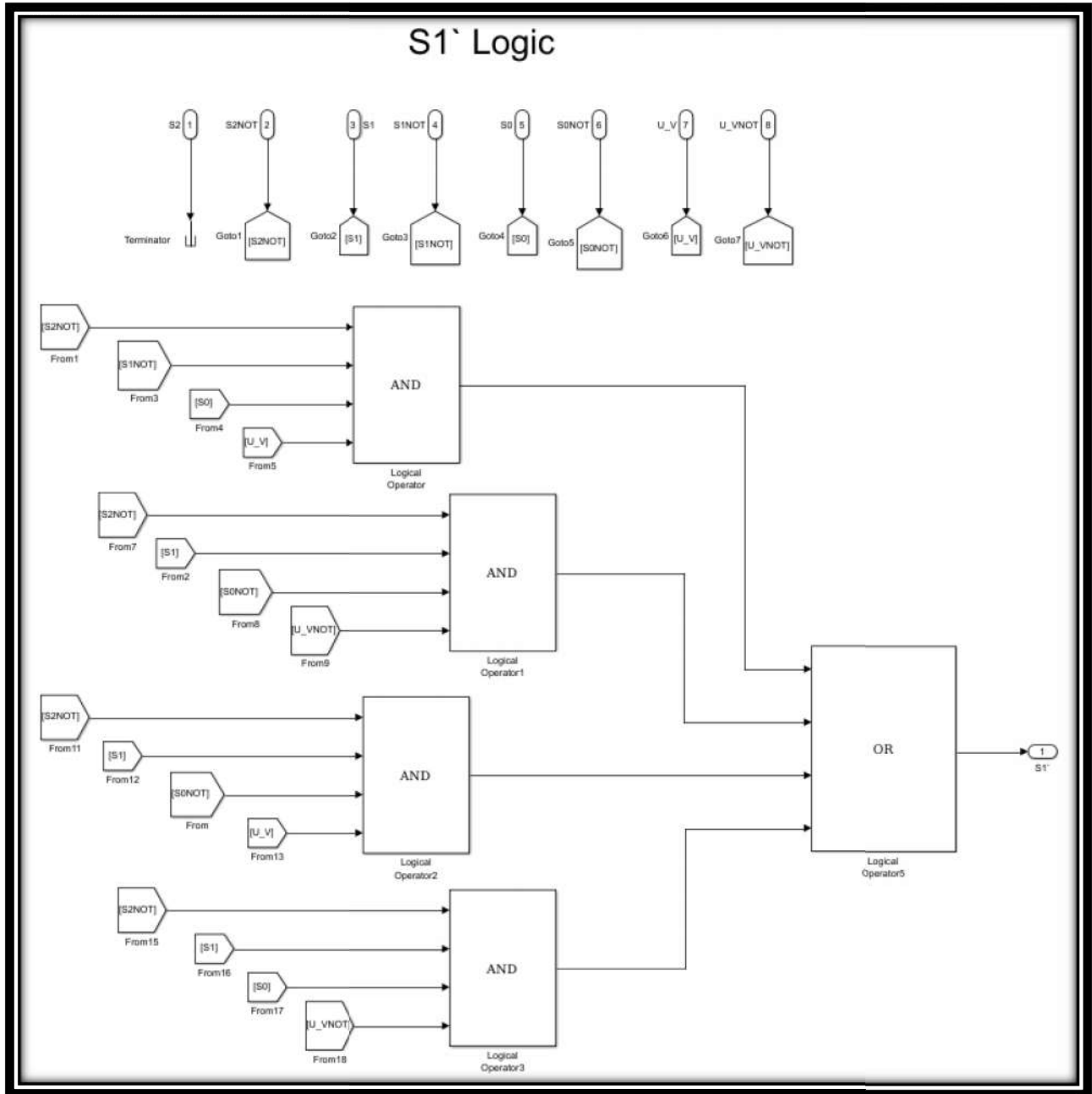


Figure B.2: S1 Logic

$$S'_1 = \overline{S_2} \overline{S_1} S_0 U_V + \overline{S_2} S_1 \overline{S_0} \overline{U_V} + \overline{S_2} S_1 \overline{S_0} U_V + \overline{S_2} S_1 S_0 \overline{U_V}$$

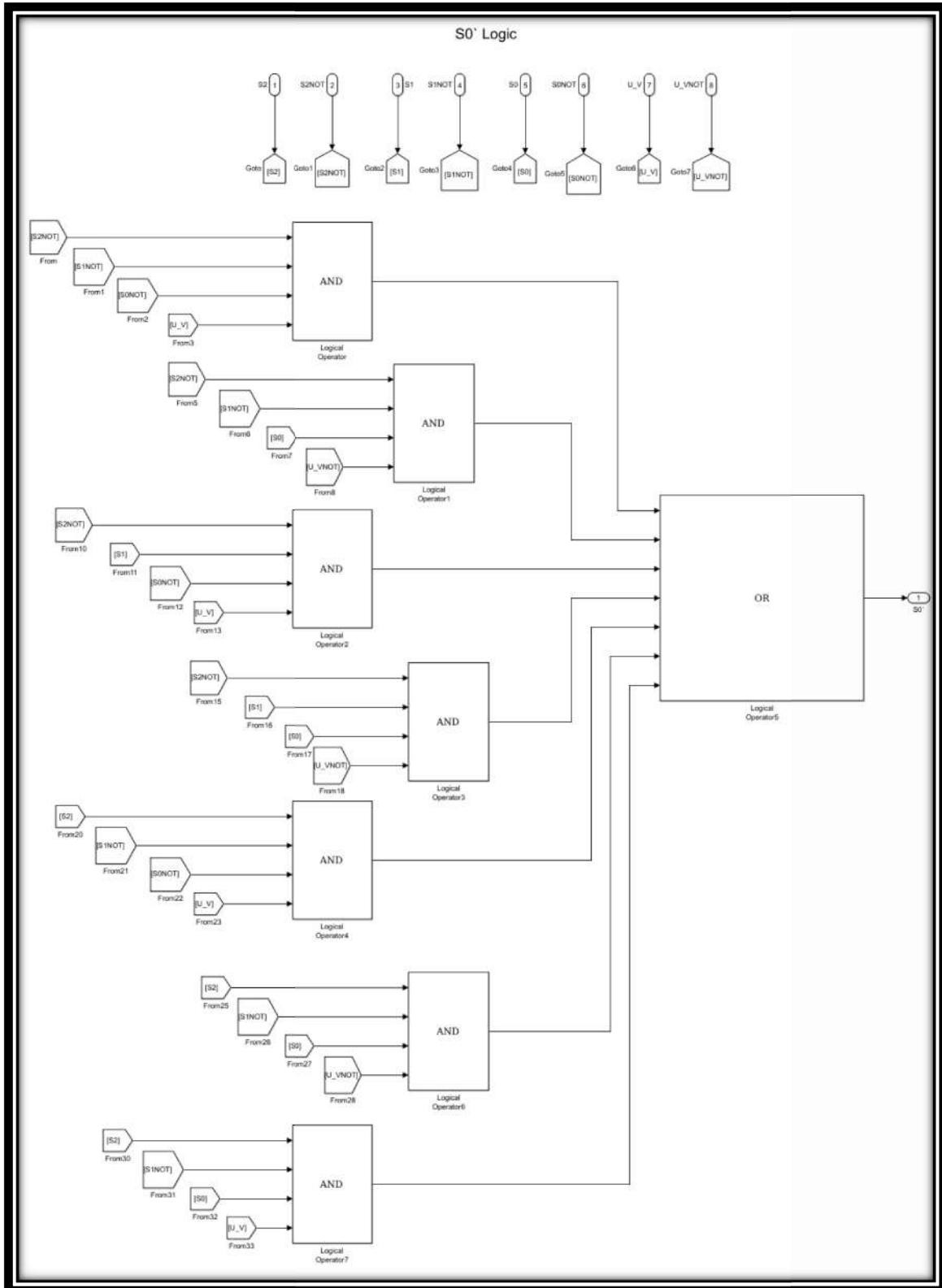


Figure B.3: S0 Logic

$$S'_0 = \overline{S_2} \overline{S_1} \overline{S_0} U \_V + \overline{S_2} \overline{S_1} S_0 \overline{U} \_V + \overline{S_2} S_1 \overline{S_0} U \_V + \overline{S_2} S_1 S_0 \overline{U} \_V + S_2 \overline{S_1} \overline{S_0} U \_V + S_2 \overline{S_1} S_0 \overline{U} \_V + S_2 S_1 \overline{S_0} U \_V + S_2 S_1 S_0 \overline{U} \_V$$

$$L_1 = \overline{S_2} \overline{S_1} S_0 + \overline{S_2} S_1 \overline{S_0} + \overline{S_2} S_1 S_0 + S_2 \overline{S_1} \overline{S_0} + S_2 \overline{S_1} S_0$$

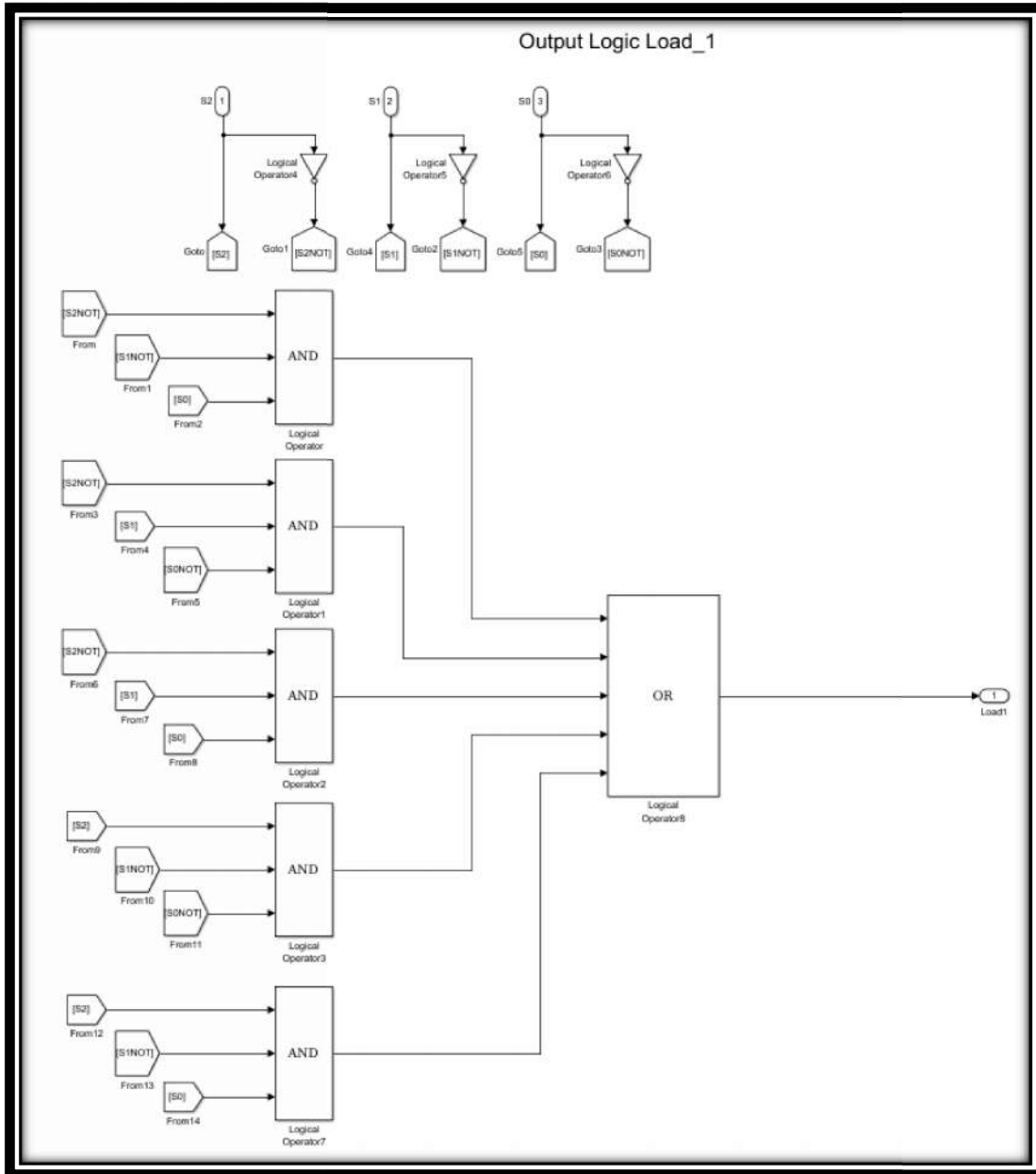


Figure B.4: Output Logic Load 1

$$L_2 = \overline{S_2}S_1\overline{S_0} + \overline{S_2}S_1S_0 + S_2\overline{S_1}\overline{S_0} + S_2\overline{S_1}S_0$$

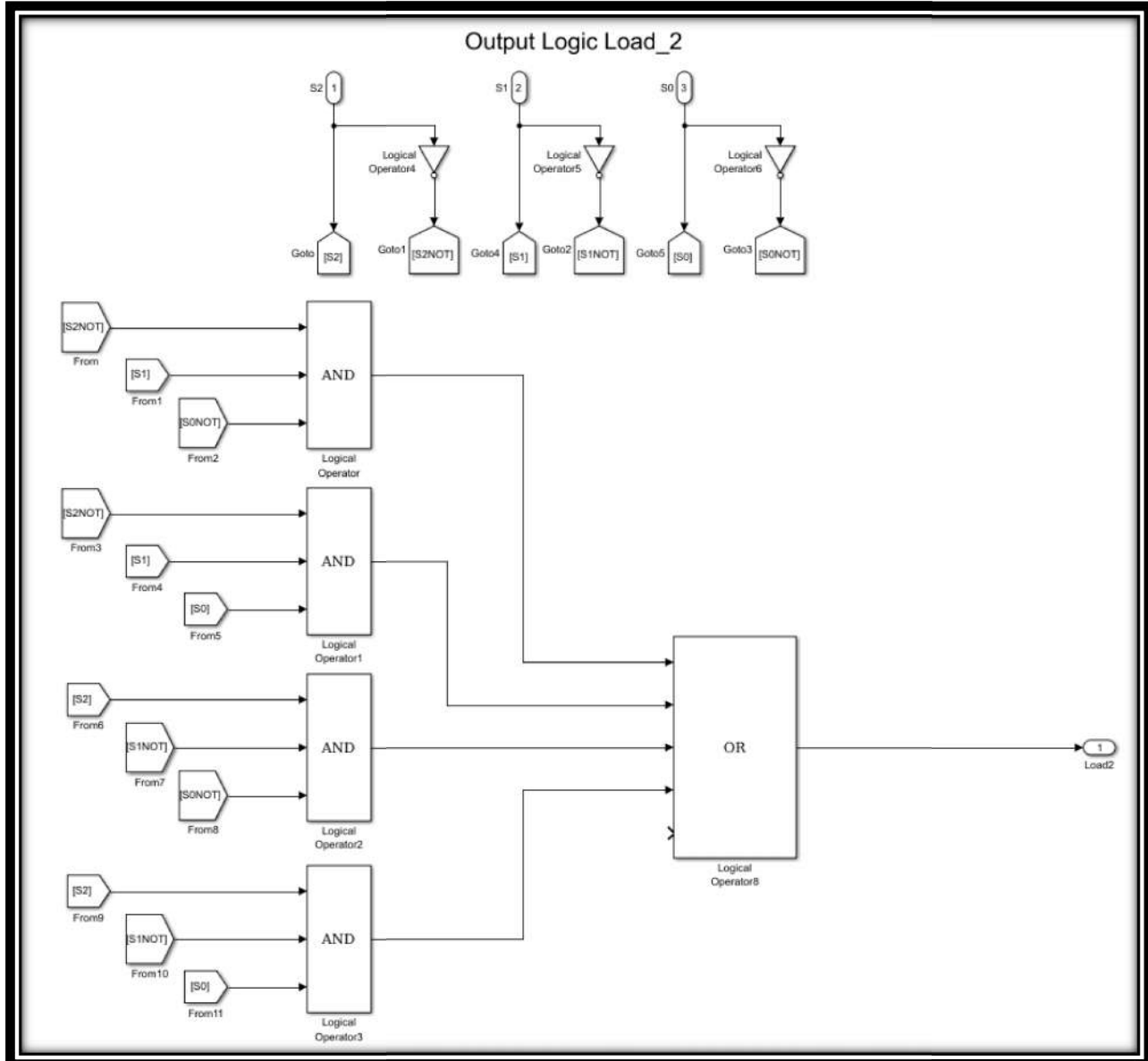


Figure B.5: Output Logic Load 2

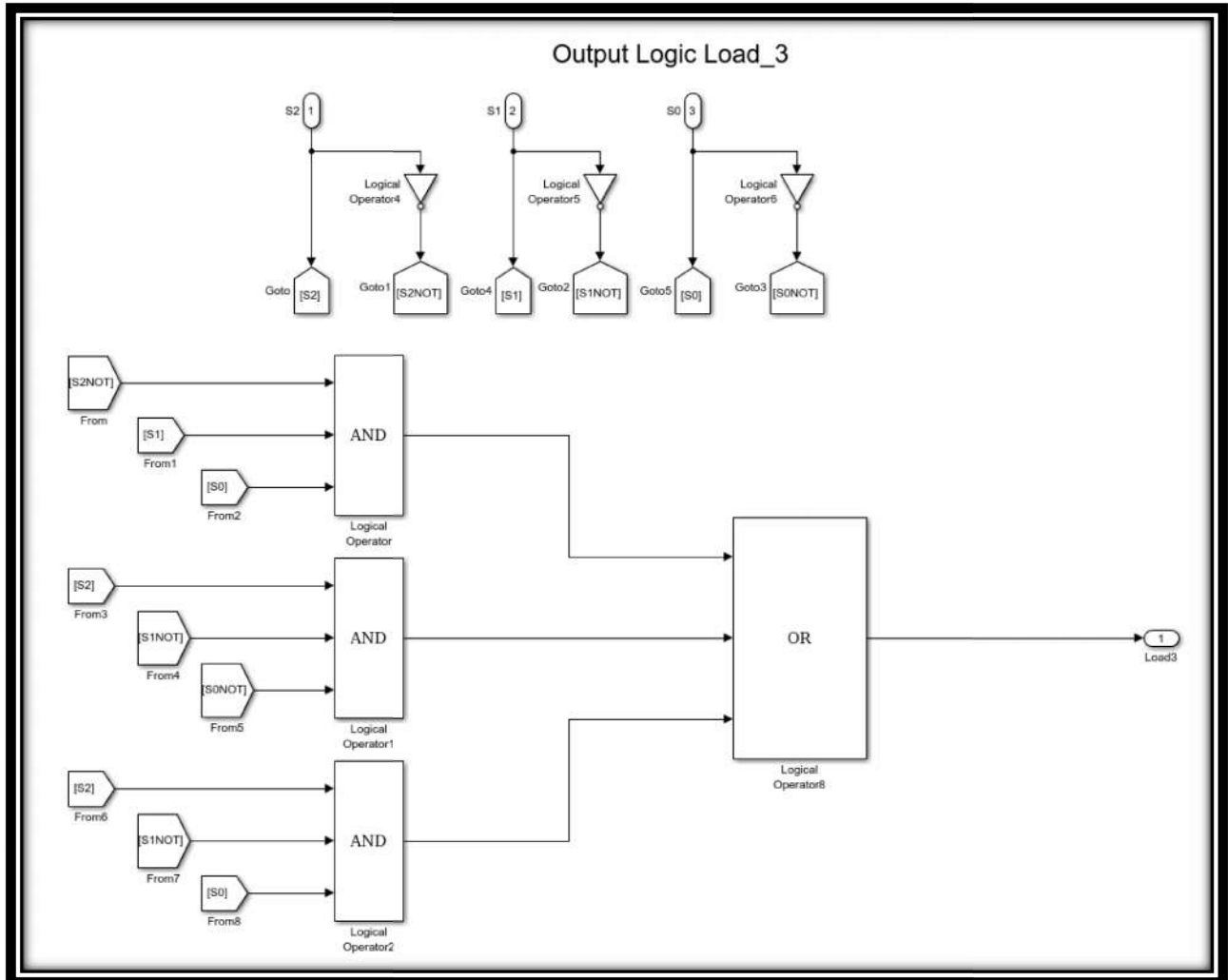


Figure B.6: Output Logic Load 3

$$L_3 = \overline{S_2}S_1S_0 + S_2\overline{S_1}\overline{S_0} + S_2\overline{S_1}S_0$$

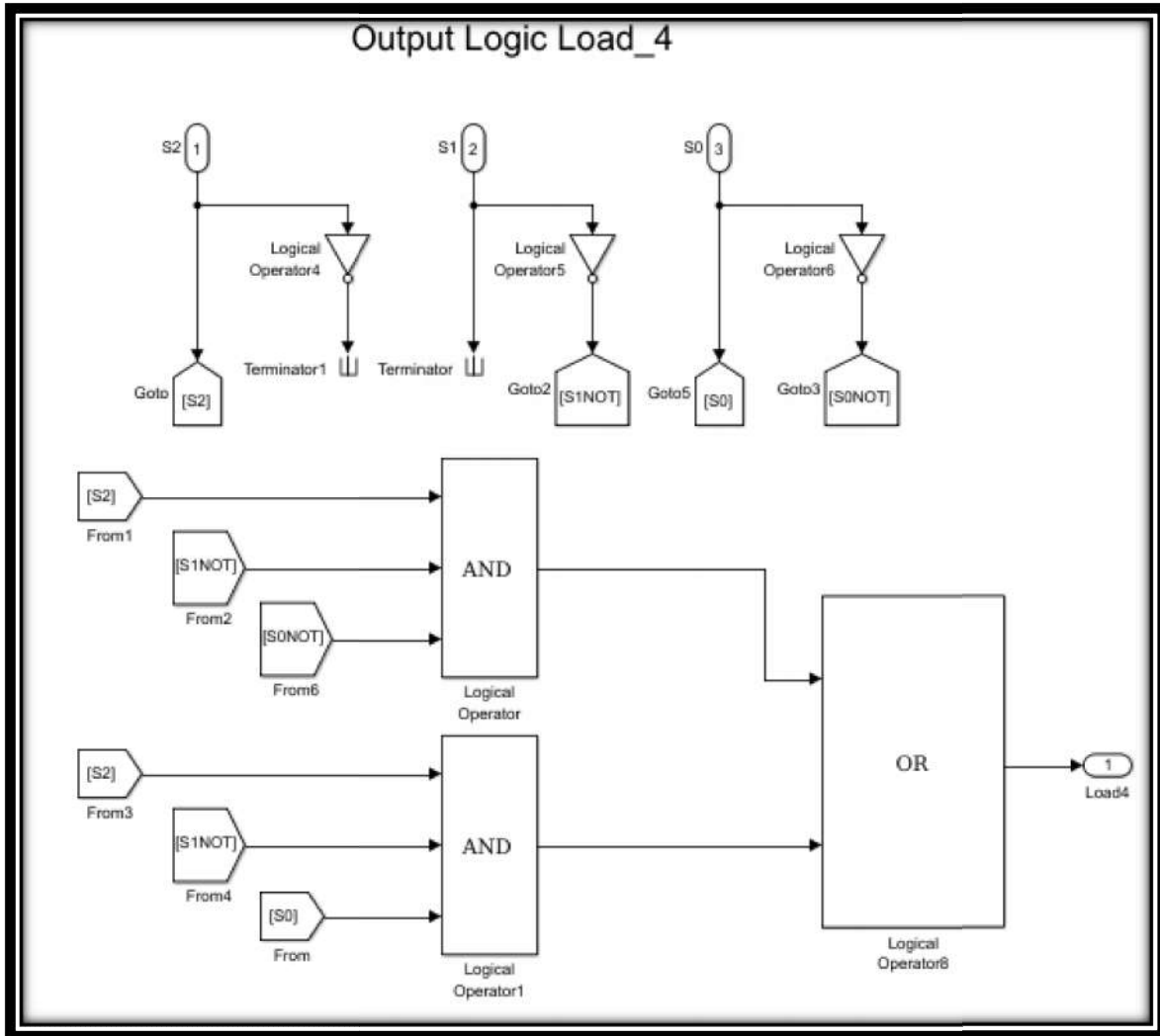


Figure B.7: Output Logic Load 4

$$L_4 = S_2 \overline{S_1} \overline{S_0} + S_2 \overline{S_1} S_0$$

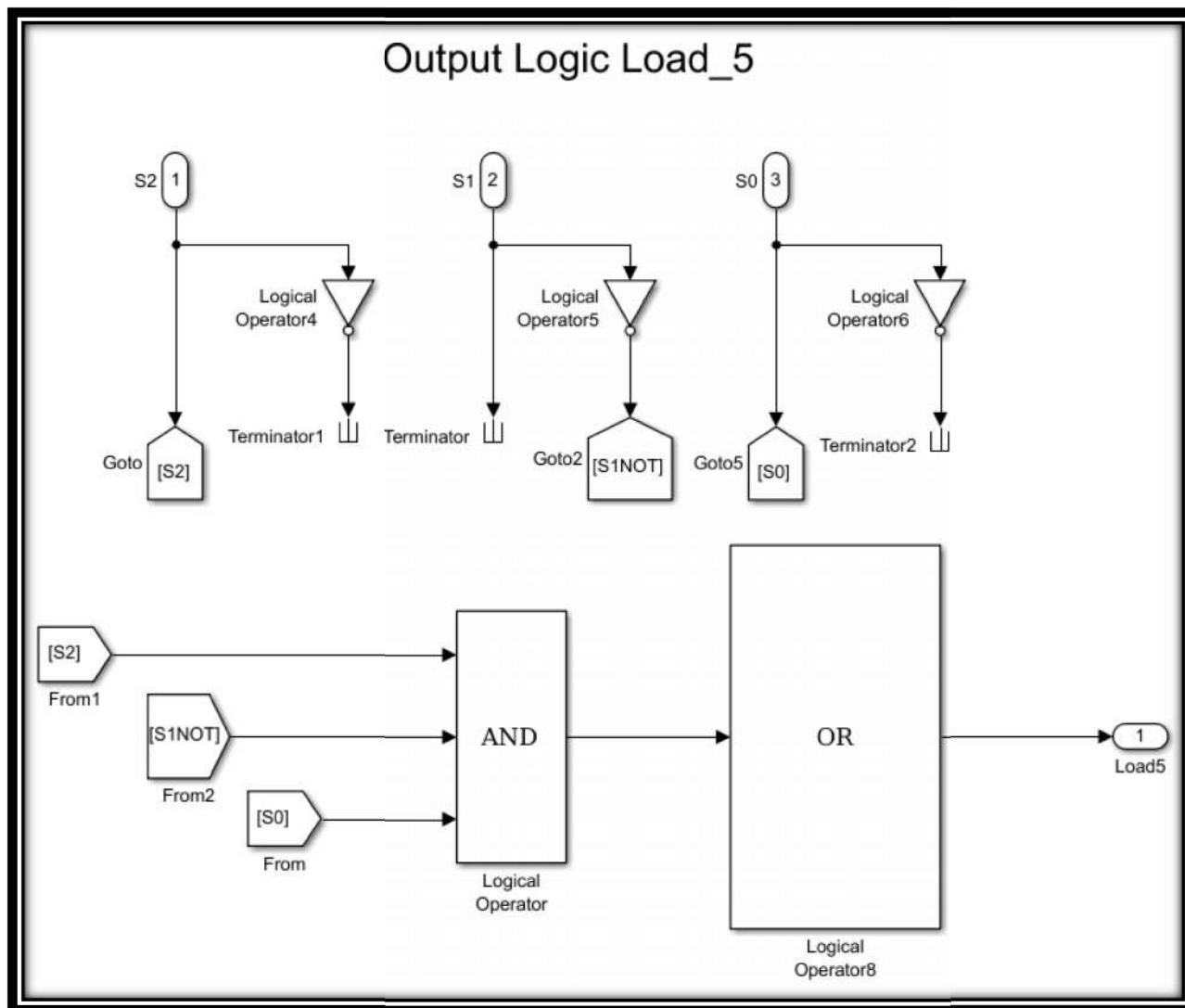


Figure B.8: Output Logic Load 5

$$L_5 = S_2 \bar{S}_1 S_0$$





## Appendix C: RSCAD Files

The figure below is the RSCAD CBuilder .def file of the load-shedding controller.

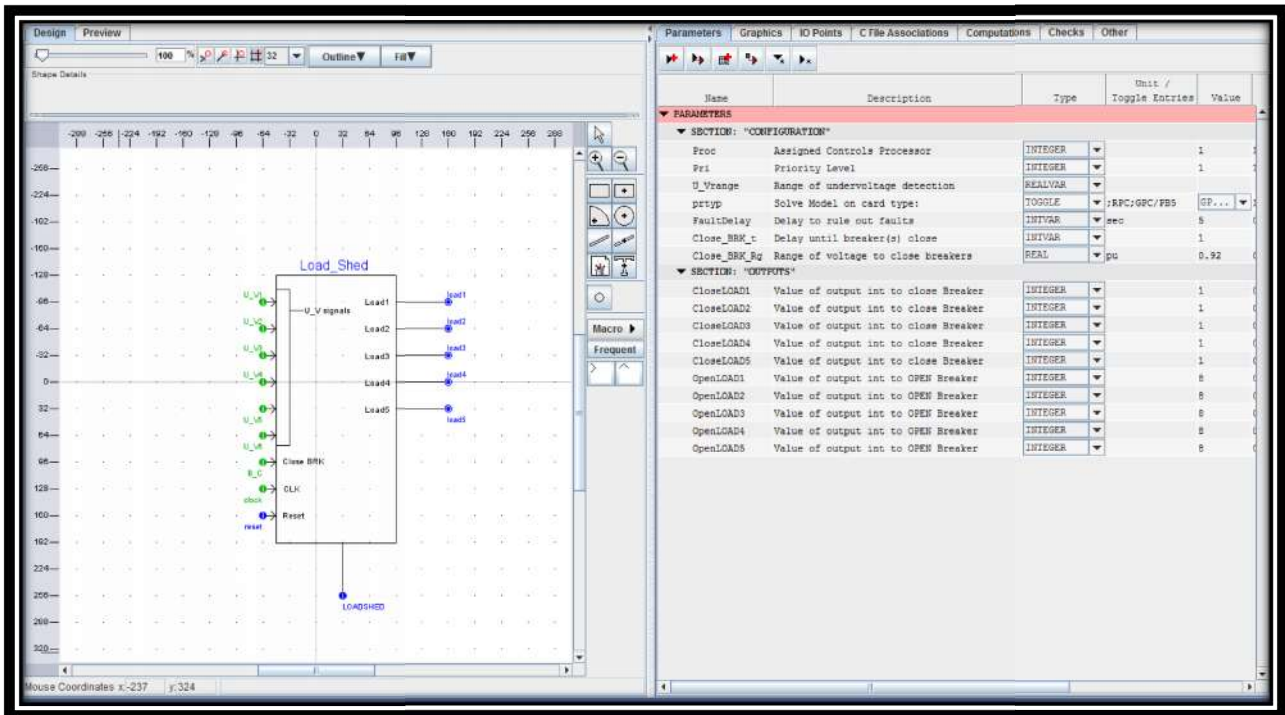


Figure C.1: Cbuilder .def File

### .h and .c files

Header file:

MODEL\_TYPE: CTL

```
#define PI      3.1415926535897932384626433832795 // definition of PI
#define TWOPI  6.283185307179586476925286766559 // definition of 2.0*PI
#define E      2.71828182845904523536028747135266 // definition of E
#define EINV   0.36787944117144232159552377016147 // definition of E Inverse (1/E)
#define RT2    1.4142135623730950488016887242097 // definition of square root 2.0
#define RT3    1.7320508075688772935274463415059 // definition of square root 3.0
#define INV_ROOT2 0.70710678118654752440084436210485
```

INPUTS:

```
int reset;
double clock;
double U_V2;
double U_V3;
double U_V4;
```

```

double U_V5;
double U_V6;
double U_V1;
double B_C;
int LOADSHED;

```

OUTPUTS:

```

int load1;
int load2;
int load3;
int load4;
int load5;

```

PARAMETERS:

```

double U_Vrange; //Range of undervoltage detection
int prtyp; //Solve Model on card type:
int FaultDelay; //Delay to rule out faults
int Close_BRK_t; //Delay until breaker(s) close
double Close_BRK_Rg; //Range of voltage to close breakers
int CloseLOAD3; //Value of output int to close Breaker
int CloseLOAD4; //Value of output int to close Breaker
int CloseLOAD5; //Value of output int to close Breaker
int OpenLOAD1; //Value of output int to OPEN Breaker
int OpenLOAD2; //Value of output int to OPEN Breaker
int OpenLOAD3; //Value of output int to OPEN Breaker
int OpenLOAD4; //Value of output int to OPEN Breaker
int OpenLOAD5; //Value of output int to OPEN Breaker
int CloseLOAD1; //Value of output int to close Breaker
int CloseLOAD2; //Value of output int to close Breaker

```

C file:

VERSION:

3.001

```
#include "loadshed.h"
```

STATIC:

```

// -----
// double dt;
// i through n are counters for load shed delay
int i = 0;

```

```

int j = 0;
int k = 0;
int l = 0;
int m = 0;
int n = 0;
// v through z are counters for closing breakers
int v = 0;
int w = 0;
int x = 0;
int y = 0;
int z = 0;
// These are the ints for initializing each state's counters
int S0_init = 1;
int S1_init = 1;
int S2_init = 1;
int S3_init = 1;
int S4_init = 1;
int S5_init = 1;

int state = 0;
int prev_clock = 1;

// STATES DEFINED
#define S0 0
#define S1 1
#define S2 2
#define S3 3
#define S4 4
#define S5 5

// - End of STATIC: Section -

RAM_FUNCTIONS:

// -----

RAM_PASS0:

//-----

// ----- End of RAM_PASS0: Section -----

RAM_PASS2:

// -----

```

```

// ----- End of RAM_PASS2: Section -----

CODE:

// -----

//FINITE STATE MACHINE LOGIC

if(clock){          // If statement that will use the clock to dictate when the code will be
run
    if (!prev_clock){
        prev_clock=1;

        if(LOADSHED == 1){          // Controller Will only shed load if this value is true

            switch(state){
                case S0:

                    if(S0_init == 1){
                        i=0; j=0; k=0; l=0; m=0; n=0;
                        v=0; w=0; x=0; y=0; z=0;
                    }

                    if((B_C &&((U_V1+U_V2+U_V3+U_V4+U_V5+U_V6)/6))
> Close_BRK_Rg){ // This statement checks the bus voltages to reconnect a load

                        state=S0;}

                    if(U_V1 < U_Vrange){

                        if(i> FaultDelay){
                            state = S1;
                        }
                        else{
                            i++;
                        }
                    }
                    else{
                        i = 0;
                    }

                    if(U_V2 < U_Vrange){

                        if(j> FaultDelay){

```

```
        state = S1;
    }
    else{
        j++;
    }
}
else{
    j = 0;
}
```

```
if(U_V3 < U_Vrange){
    if(k > FaultDelay){
        state = S1;
    }
    else{
        k++;
    }
}
else{
    k = 0;
}
```

```
if(U_V4 < U_Vrange){
    if(l > FaultDelay){
        state = S1;
    }
    else{
        l++;
    }
}
else{
    l = 0;
}
```

```
if(U_V5 < U_Vrange){
    if(m > FaultDelay){
        state = S1;
    }
    else{
        m++;
    }
}
```

```

    }
}
else{
    m = 0;
}

if(U_V6 < U_Vrange){

    if(n> FaultDelay){
        state = S1;
    }
    else{
        n++;
    }
}
else{
    n = 0;
}

S0_init = 0;

break;

case S1:

if(S1_init == 1){// Initialize the state counters

    i=0; j=0; k=0; l=0; m=0; n=0;
    v=0; w=0; x=0; y=0; z=0;
}

if((B_C &&((U_V1+U_V2+U_V3+U_V4+U_V5+U_V6)/6))>
    Close_BRK_Rg){
    if(v > Close_BRK_t){
        S0_init = 1;
        state=S0;
    }
    else{
        v++;
    }
}
else{
    v = 0;
}

```

```
if(U_V1 < U_Vrange){  
    if(i> FaultDelay){  
        state = S2;  
    }  
    else{  
        i++;  
    }  
}  
else{  
    i = 0;  
}
```

```
if(U_V2 < U_Vrange){  
    if(j> FaultDelay){  
        state = S2;  
    }  
    else{  
        j++;  
    }  
}  
else{  
    j = 0;  
}
```

```
if(U_V3 < U_Vrange){  
    if(k> FaultDelay){  
        state = S2;  
    }  
    else{  
        k++;  
    }  
}  
else{  
    k = 0;  
}
```

```
if(U_V4 < U_Vrange){
```



```

        if(l> FaultDelay){
            state = S2;
        }
        else{
            l++;
        }
    }
else{
    l = 0;
}

```

```

if(U_V5 < U_Vrange){

    if(m> FaultDelay){
        state = S2;
    }
    else{
        m++;
    }
}
else{
    m = 0;
}

```

```

if(U_V6 < U_Vrange){

    if(n> FaultDelay){
        state = S2;
    }
    else{
        n++;
    }
}
else{
    n = 0;
}
S1_init = 0;
break;

```

case S2:

```

if(S2_init == 1){

    i=0; j=0; k=0; l=0; m=0; n=0;

```

```

        v=0; w=0; x=0; y=0; z=0;
    }
    if((B_C &&((U_V1+U_V2+U_V3+U_V4+U_V5+U_V6)/6)) >
        Close_BRK_Rg){
        if(w > Close_BRK_t){
            S1_init = 1;
            state=S1;
        }
        else{
            w++;
        }
    }
    else{
        w = 0;
    }
}

```

```

if(U_V1 < U_Vrange){

    if(i> FaultDelay){
        state = S3;
    }
    else{
        i++;
    }
}
else{
    i = 0;
}
}

```

```

if(U_V2 < U_Vrange){

    if(j> FaultDelay){
        state = S3;
    }
    else{
        j++;
    }
}
else{
    j = 0;
}
}

```

```
if(U_V3 < U_Vrange){  
    if(k > FaultDelay){  
        state = S3;  
    }  
    else{  
        k++;  
    }  
}  
else{  
    k = 0;  
}
```

```
if(U_V4 < U_Vrange){  
    if(l > FaultDelay){  
        state = S3;  
    }  
    else{  
        l++;  
    }  
}  
else{  
    l = 0;  
}
```

```
if(U_V5 < U_Vrange){  
    if(m > FaultDelay){  
        state = S3;  
    }  
    else{  
        m++;  
    }  
}  
else{  
    m = 0;  
}
```

```
if(U_V6 < U_Vrange){  
    if(n > FaultDelay){
```

```

        state = S3;
    }
    else{
        n++;
    }
}
else{
    n = 0;
}
S2_init = 0;
break;

```

case S3:

```

if(S3_init == 1){

    i=0; j=0; k=0; l=0; m=0; n=0;
    v=0; w=0; x=0; y=0; z=0;
}

if((B_C && ((U_V1+U_V2+U_V3+U_V4+U_V5+U_V6)/6))>
    Close_BRK_Rg){
    if(x > Close_BRK_t){
        S2_init = 1;
        state=S2;
    }
    else{
        x++;
    }
}
else{
    x = 0;
}

if(U_V1 < U_Vrange){

    if(i> FaultDelay){
        state = S4;
    }
    else{
        i++;
    }
}
else{

```

```
        i = 0;
    }

    if(U_V2 < U_Vrange){

        if(j> FaultDelay){
            state = S4;
        }
        else{
            j++;
        }
    }
    else{
        j = 0;
    }

    if(U_V3 < U_Vrange){

        if(k> FaultDelay){
            state = S4;
        }
        else{
            k++;
        }
    }
    else{
        k = 0;
    }

    if(U_V4 < U_Vrange){

        if(l> FaultDelay){
            state = S4;
        }
        else{
            l++;
        }
    }
    else{
        l = 0;
    }
}
```

```

if(U_V5 < U_Vrange){
    if(m> FaultDelay){
        state = S4;
    }
    else{
        m++;
    }
}
else{
    m = 0;
}

```

```

if(U_V6 < U_Vrange){
    if(n> FaultDelay){
        state = S4;
    }
    else{
        n++;
    }
}
else{
    n = 0;
}
S3_init = 0;
break;

```

case S4:

```

if(S4_init == 1){
    i=0; j=0; k=0; l=0; m=0; n=0;
    v=0; w=0; x=0; y=0; z=0;
}

if((B_C && ((U_V1+U_V2+U_V3+U_V4+U_V5+U_V6)/6))>
    Close_BRK_Rg){
    if(y > Close_BRK_t){
        S3_init = 1;
        state=S3;
    }
    else{
        y++;
    }
}

```

```
}  
else{  
    y = 0;  
}
```

```
if(U_V1 < U_Vrange){  
    if(i > FaultDelay){  
        state = S5;  
    }  
    else{  
        i++;  
    }  
}  
else{  
    i = 0;  
}
```

```
if(U_V2 < U_Vrange){  
    if(j > FaultDelay){  
        state = S5;  
    }  
    else{  
        j++;  
    }  
}  
else{  
    j = 0;  
}
```

```
if(U_V3 < U_Vrange){  
    if(k > FaultDelay){  
        state = S5;  
    }  
    else{  
        k++;  
    }  
}  
else{  
    k = 0;  
}
```

```
}  
  
if(U_V4 < U_Vrange){  
    if(l> FaultDelay){  
        state = S5;  
    }  
    else{  
        l++;  
    }  
}  
else{  
    l = 0;  
}  
  
if(U_V5 < U_Vrange){  
    if(m> FaultDelay){  
        state = S5;  
    }  
    else{  
        m++;  
    }  
}  
else{  
    m = 0;  
}  
  
if(U_V6 < U_Vrange){  
    if(n> FaultDelay){  
        state = S5;  
    }  
    else{  
        n++;  
    }  
}  
else{  
    n = 0;  
}  
S4_init = 0;  
break;
```

case S5:



```

    if(S5_init == 1){
        i=0; j=0; k=0; l=0; m=0; n=0;
        v=0; w=0; x=0; y=0; z=0;
    }

    if((B_C && ((U_V1+U_V2+U_V3+U_V4+U_V5+U_V6)/6)) >
    Close_BRK_Rg){
        if(z > Close_BRK_t){
            S4_init = 1;
            state=S4;
        }
        else{
            z++;
        }
    }
    else{
        z = 0;
    }

    S5_init = 0;
    break;

}

}

}

if(!clock){
    prev_clock=0;}
if(reset){
    state = S0;}

// OUTPUT LOGIC  CloseLOAD# AND OpenLOAD# ARE PARAMETERS

if(state==S0){
    load1 = CloseLOAD1;           //SO ALL OF THESE WOULD BE 1'S
    load2 = CloseLOAD2;
    load3 = CloseLOAD3;
    load4 = CloseLOAD4;
    load5 = CloseLOAD5;
}
if(state==S1){
    load1 = OpenLOAD1;           // THIS OUTPUT WOULD BE 8. AND SO ON.

```

```
        load2 = CloseLOAD2;
        load3 = CloseLOAD3;
        load4 = CloseLOAD4;
        load5 = CloseLOAD5;
    }
    if(state==S2){
        load1 = OpenLOAD1;
        load2 = OpenLOAD2;
        load3 = CloseLOAD3;
        load4 = CloseLOAD4;
        load5 = CloseLOAD5;
    }
    if(state==S3){
        load1 = OpenLOAD1;
        load2 = OpenLOAD2;
        load3 = OpenLOAD3;
        load4 = CloseLOAD4;
        load5 = CloseLOAD5;
    }
    if(state==S4){
        load1 = OpenLOAD1;
        load2 = OpenLOAD2;
        load3 = OpenLOAD3;
        load4 = OpenLOAD4;
        load5 = CloseLOAD5;
    }
    if(state==S5){
        load1 = OpenLOAD1;
        load2 = OpenLOAD2;
        load3 = OpenLOAD3;
        load4 = OpenLOAD4;
        load5 = OpenLOAD5;
    }
}
```

```
// ----- End of CODE: Section -----
```

## Appendix D: Battery Controller Coding

```

#include "Ctr_Batt.h"
STATIC:
int prev_CLK,state;

// - End of STATIC: Section -

RAM_FUNCTIONS:
RAM_PASS0:
    setStackingLoad( this_, 10, ""); //This may need to be removed depending on the version
RSCAD
// ----- End of RAM_PASS0: Section -----

RAM_PASS2:
prev_CLK=0;
state=0;

// ----- End of RAM_PASS2: Section -----

CODE:
if(CLK){
    if(!prev_CLK){ //Rising Edge of Clock
        prev_CLK=1;
        if(!L_in){
            state=0;
        }
        else if(!W_in){
//-----Next State Logic-----
            switch(state){
                case (0):
                    if(High_Freq<=Freq){
                        state=1;
                    }
                    else if(Low_Freq>=Freq){
                        state=5;
                    }
                }
                else{
                    state=0;
                }
                break;
            case (1):
                if(High_Freq<=Freq){
                    state=2;
                }
                else if(Low_Freq>=Freq){

```

```
        state=0;
    }
    else{
        state=1;
    }
    break;
case (2):
    if(High_Freq<=Freq){
        state=3;
    }
    else if(Low_Freq>=Freq){
        state=1;
    }
    else{
        state=2;
    }
    break;
case (3):
    if(High_Freq<=Freq){
        state=4;
    }
    else if(Low_Freq>=Freq){
        state=2;
    }
    else{
        state=3;
    }
    break;
case (4):
    if(High_Freq<=Freq){
        state=4;
    }
    else if(Low_Freq>=Freq){
        state=2;
    }
    else{
        state=3;
    }
    break;
case (5):
    if(High_Freq<=Freq){
        state=0;
    }
    else if(Low_Freq>=Freq){
        state=6;
    }
}
```

```

        else{
            state=5;
        }
        break;
case (6):
    if(High_Freq<=Freq){
        state=5;
    }
    else if(Low_Freq>=Freq){
        state=7;
    }
    else{
        state=6;
    }
    break;
case (7):
    if(High_Freq<=Freq){
        state=6;
    }
    else if(Low_Freq>=Freq){
        state=8;
    }
    else{
        state=7;
    }
    break;
case (8):
    if(High_Freq<=Freq){
        state=6;
    }
    else if(Low_Freq>=Freq){
        state=8;
    }
    else{
        state=7;
    }
    break;
    }
}
}
else{
    prev_CLK=0;
}
//-----Output Logic-----
switch(state){

```

```
case 0:
    C_out=0;
    P_out=0;
    State=1;
    L_out=0;
    W_out=0;
    break;
case 1:
    C_out=1;
    P_out=0;
    State=1;
    L_out=0;
    W_out=1;
    break;
case 2:
    C_out=1;
    P_out=0;
    State=2;
    L_out=0;
    W_out=1;
    break;
case 3:
    C_out=1;
    P_out=0;
    State=3;
    L_out=0;
    W_out=1;
    break;
case 4:
    C_out=1;
    P_out=0;
    State=3;
    L_out=-1;
    W_out=1;
    break;
case 5:
    C_out=0;
    P_out=1;
    State=1;
    L_out=0;
    W_out=1;
    break;
case 6:
    C_out=0;
    P_out=1;
    State=2;
```

```
        L_out=0;
        W_out=1;
        break;
case 7:
    C_out=0;
    P_out=1;
    State=3;
    L_out=0;
    W_out=1;
    break;
case 8:
    C_out=0;
    P_out=1;
    State=3;
    L_out=1;
    W_out=1;
    break;
}
```

## Appendix E: Battery Power System Coding

```

#include "Battery3.h"

STATIC:

// Static Variables to be used throughout the function

int prev_CLK,start;
double Energy,VA,prev_IA,power;
double Loss2=1.09091;
double Loss3=1.45455;

// - E n d   o f   S T A T I C :   S e c t i o n -

RAM_FUNCTIONS:
RAM_PASS0:
    setStackingLoad( this_, 10, "");

// ----- End of RAM_PASS0: Section -----

RAM_PASS2:

// Initial values in the system this section of
// of code is only ran once at the beginning of
// the simulation
// -----
Energy = Inital_Energy;
prev_IA=0;
prev_CLK=0;
start=1;
power=0;

// ----- End of RAM_PASS2: Section -----

CODE:

    BEGIN_T0:
// -----
// RSCAD uses the current injections to calculate
// the voltage of the system in BEGIN_T0 currents for
// the battery is considered to be a balanced system
// using the previous value determined at the end
// of the code initialized at zero the current is set
// -----

```



```

IA=prev_IA;
IB=IA;
IC=IA;
IG=-1*(IA+IB+IC);

T0_T2:
VA=A_1-V_2;

if(CLK){    //Clock High

if(!prev_CLK){    // Rising Edge of Clock
    prev_CLK=1;
    switch(State){ //State defined by controller
        case (1):
            if(!VA){
                power=0;
            }
            else if(P_in){
                if(Max_Discharge<(Energy-Power1)){
                    power=-Power1;
                    Energy=(Energy-Power1);
                }
                else{
                    power=0;
                }
            }
            else if(C_in){
                if(Max_Charge>(Energy+Power1)){
                    power=Power1;
                    Energy=Energy+Power1;
                }
                else{
                    power=0;
                }
            }
            else{
                power=0;
            }
            break;
        case (2):
            if(!VA){
                power=0;
            }
            else if(P_in){
                if(Max_Discharge<(Energy-Power2*Loss2)){
                    power=-Power2;

```

```

        Energy=(Energy-Power2*Loss2);
    }
    else{
        power=0;
    }
}
else if(C_in){
    if(Max_Charge>(Energy+Power2*Loss2)){
        power=Power2;
        Energy=Energy+Power2*Loss2;
    }
    else{
        power=0;
    }
}
else{
    power=0;
}
break;
case (3):
    if(!VA){
        power=0;
    }
    else if(P_in){
        if(Max_Discharge<(Energy-Power3*Loss3)){
            power=-Power3;
            Energy=(Energy-Power3*Loss3);
        }
        else{
            power=0;
        }
    }
    else if(C_in){
        if(Max_Charge>(Energy+Power3*Loss3)){
            power=Power3;
            Energy=Energy+Power3*Loss3;
        }
        else{
            power=0;
        }
    }
    else{
        power=0;
    }
    break;
}
}

```

```
        }
    }
    else{
        prev_CLK=0;

    }
    if(VA){
        prev_IA=power/(VA*Base_Power);
    }
    else{
        prev_IA=0;
    }
//-----Monitored Variables-----
    En=Energy;//Stored energy in Battery
    P_out=power; //Power out of Battery
    IMON=prev_IA; //Current of a single phase out of Battery

// ----- End of CODE: Section -----
```

## **Appendix F: Software and Hardware Used in Automation controller Development**

- SEL-5030 AcSELeRator Quickset software: used to configure and commission the SEL-451 Protection, Automation, and Bay Control System.
- SEL-5032 AcSELeRator Architect software: used to configure Generic Object-Oriented Substation Events (GOOSE) Communications.
- SEL-5033 AcSELeRator RTAC software: used to configure the SEL-3530 Real-Time Automation Controller (RTAC).
- SEL-5035 AcSELeRator Diagram Builder software: enabled the creation of a Human-Machine Interface (HMI).
- SEL-3530 Real-Time Automation Controller (RTAC): contains automation logic and sends commands to the SEL-451 to act upon it. It also supplies data to the HMI.
- SEL-451 Protection, Automation, and Bay Control System: measures values off of the transmission lines and controls the connected breaker according to commands sent from the RTAC.