VOLTAGE SUPPORT CONTROL FOR DISTRIBUTED GENERATION SYSTEMS

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Roozbeh Kabiri

M. E. (2011) RMIT University
B.Sc. (2009) Tehran Polytechnic

School of Electrical and Computer Engineering
College of Science, Engineering and Health
RMIT University Australia

August 2015
Copyright Notice

In return for freely distributing this PhD thesis, I kindly request that each time another copy of this work (either in electronic or in printed form) gets passed to another entity, the name and email address of the new recipient be emailed to myself at:

kabiri.roozbeh@gmail.com

This thesis may not be placed electronically where public download access is available without prior authorisation from the author.

Kind regards,

Roozbeh Kabiri
Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other institution, and to the best of the author’s knowledge, contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

_____________________________
Roozbeh Kabiri
ABSTRACT

Distributed Generation (DG) - defined as small-scale electricity generation at or near the end user - is becoming more prevalent in recent years with an increasing penetration of renewable generation emerging in utility distribution grids. Flexible operation of these DG units is a major objective for such grids. In the operational control framework, DG units should be considered as a part of the system, so that system reliability can be enhanced by ancillary service provided by DG systems.

As conventional electrical networks were not built to accommodate large DG penetration levels, several new challenges have emerged. An important issue that appears in low voltage (LV) feeders is over-voltage and under-voltage, which can limit the level of DG penetration that can be tolerated and this also affects the LV protection systems. On the other hand, control of DG units enables a wide range of new support functions such as real/reactive power control under unbalanced voltage conditions, which can help to mitigate these unbalanced network voltages.

Considering these factors, the objectives of this thesis are detailed as follows:

The first objective is to investigate the application of DG systems to provide broader system level support functions within a grid. Initially, voltage regulation of an LV feeder is investigated using PV inverter reactive power injection. The inadequate effect of PV inverter reactive power injection on voltage regulation was identified, and it was shown that their range of performance can be limited because the higher resistance characteristics of a typical LV feeder constrains the effectiveness of voltage regulation by reactive power injection. To resolve this limitation a combined voltage regulation strategy, is proposed where an electronic tap-changer is incorporated into the feeder distribution transformer with the PV DG systems at each feeder bus then providing local reactive power support. The investigation accounted for a variety of issues such as feeder impedance, dynamic transformer tap changing, different load types
and levels of PV penetration to achieve voltage regulation of LV feeders incorporating large scale PV penetration.

The second objective is to propose a new DG current regulation method under unbalanced voltage conditions which can achieve desired positive and negative sequence current control objectives. The complexity and practical limitations of existing current control methods, which are associated with current sequence separation, inevitably compromises the current regulator’s performance, particularly during transient events. An improved current controller is required to offer a less complex structure, while also enhancing the overall performance of the current regulator. A novel current regulation method under unbalanced voltage conditions is presented in this thesis which eliminates the need for current sequence extraction and at the same time significantly improves the dynamic response of the current regulation strategy.

Finally, the third research objective is to investigate inverter power control methods which can contribute to voltage quality, in particular during unbalanced voltage conditions. A detailed theoretical analysis is developed for the output power control of DG inverters under such conditions, incorporating factors which can contribute to mitigation of abnormal voltage conditions such as real and reactive power oscillatory terms, and differing grid impedance characteristics ranging from resistive to inductive. The results are used to identify the most effective control approach which achieves better voltage quality at the point of common coupling. Furthermore, fault ride through capabilities can be added to the DG inverter functionality to support the grid voltage during such symmetrical/asymmetrical faults, which consequently allow increased penetration of Distributed Energy Resources (DER).

All theoretical developments presented in this work have been validated using detailed simulation and experimental results. The contributions contained in this thesis have been submitted in three journal papers and six conference papers.
ACKNOWLEDGEMENTS

Four years of adventure comes to a close, despite all the challenges, obstacles and dead ends along the road, in large part thanks to the emotional and intellectual support of my family, friends and colleagues who made this journey possible.

First and foremost, I wish to thank my supervisors Prof. Grahame Holmes and Dr. Brendan McGrath. Grahame has always been a personal inspiration for me to take this next step in my life and I feel particularly fortunate and privileged to have worked with him. I wish to also express my deep gratitude to Brendan for his invaluable guidance which made this journey possible for me.

Secondly I wish to thank everyone within the Power Electronic Group (PEG) at RMIT University for their invaluable assistance, help and above all friendship. In particular, I wish to thank Dr. Stewart Parker and Dr. Carlos Teixeira for their technical assistance with the experimental system. I wish to also thank Dr. Lasantha Meegahapola for his contributions and support during the past year and Dr. Babak Shoeiby for his help and for sharing his experiences.

Next I would like to thank all my friends in Australia, including Amin, Farzin, Neda, Tahmine and the ones I have forgotten, for helping me feel at home so far from home.

Finally, I wish to thank my family, particularly my parents Morteza and Esmat, and my siblings Negar and Nima. Their never-ending support and encouragement makes it possible to achieve my goals in life no matter how far away we are from each other.
Several parts of the work and ideas presented in this thesis have been published by the author during the course of the research. These publications are listed below.


# TABLE OF CONTENTS

Abstract ........................................................................................................................................... v
Acknowledgements .......................................................................................................................... vii
Publications ........................................................................................................................................ ix
Table of Contents ............................................................................................................................... xi
List of Figures ..................................................................................................................................... xvii
List of Tables ...................................................................................................................................... xxiii
Glossary of Terms .............................................................................................................................. xxv
List of Symbols ................................................................................................................................. xxvii

## Chapter 1 Introduction ..................................................................................................................... 1

1.1. Motivation .................................................................................................................................. 1
1.2. Objectives ................................................................................................................................. 2
1.3. Identification of Original Contributions ................................................................................... 4
1.4. Thesis Structure ........................................................................................................................ 5

## Chapter 2 Literature Review .......................................................................................................... 7

2.1. Electrical Network and Distributed Generation ......................................................................... 8
  2.1.1. New Era in System Control and Management .................................................................. 10
  2.1.2. The Challenge of Power Management ............................................................................. 11
  2.1.3. The Challenge of Voltage Regulation ............................................................................. 11
2.2. DER Integration into the Utility Network .................................................................................. 12
# Table of Contents

2.2.1. Voltage Control................................................................. 12
2.2.2. Reactive Power ................................................................. 13
2.2.3. Power Curtailment ............................................................. 15
2.2.4. DG sizing and location....................................................... 15
2.2.5. Unbalanced Voltage Mitigation.......................................... 16
2.2.6. Loss Reduction................................................................. 17

2.3. Inverter Control Modelling .................................................. 17
2.3.1. Current Regulation Method in Synchronous Reference Frame .... 18
2.3.2. Power Control Strategies .................................................. 24
2.3.3. Unbalanced Voltage Mitigation .......................................... 28

2.4. Summary ............................................................................. 29

Chapter 3 Influence of PV Inverter Reactive Power on Voltage Regulation ..... 31
3.1. PV Reactive Power Support for LV Feeder Voltage Regulation .......... 32
3.2. LV Feeder Model .................................................................. 32
3.3. Exemplar DG Unit Reactive Power Injection Strategies ................. 34
3.3.1. Reactive Power Management (RPM) [47] ............................. 34
3.3.2. Power Loss Minimization (PLM) [58] ................................. 36
3.3.3. Voltage Rise Minimization (VRM) [29] ............................... 37
3.3.4. Power Curtailment Strategy (PCS) [33] ............................... 39
3.4. Simulation Results .................................................................. 40
3.5. Summary ............................................................................. 44

Chapter 4 LV Feeder Voltage Regulation Using Electronic Tap Changer ....... 45
4.1. Integrated Feeder Voltage Regulation and Loss Minimisation Strategy ...... 46
4.1.1. Explains the Electronic Tap Changer Principles ..................... 47
4.1.2. Feeder Voltage Regulation using Transformer Tap Selection ........ 51
4.1.3. PV Reactive Injection for Loss Minimization .......................... 60
4.2. Comparative Voltage Regulation Performance of Strategies Investigated and Proposed Method ................................................................. 60
4.3. Summary .............................................................................................................. 66

Chapter 5 DG Unit Current Control for Unbalanced Voltage Conditions ........ 67

5.1. Fundamentals of Oscillating Power Control in The SRF ......................... 67
5.2. Double Synchronous Reference Frame Current Regulation ....................... 69
  5.2.1. Conventional DSRF Current Regulation with Sequence Extraction ...... 69
  5.2.2. Proposed New Strategy for Unbalanced Current Regulation .......... 75
  5.2.3. Performance Evaluation of Proposed DSRF Current Regulator ....... 80
5.3. Consolidated Power Control for a Three-Phase DG System .................... 84
  5.3.1. Active Power Ripple Control ................................................................. 85
  5.3.2. Reactive Power Ripple Control ............................................................. 86
  5.3.3. Balanced Current Control ................................................................. 86
  5.3.4. Combined Power Ripple Control ....................................................... 87
5.4. Summary ......................................................................................................... 90

Chapter 6 Distribution System Modelling for Unbalanced Voltage Mitigation . 91

6.1. Power Flow Ripple Control and Its Effect on PCC Grid Voltage .............. 92
  6.1.1. Case I: Grid with resistive feeder impedance ................................. 94
  6.1.2. Case II: Grid with inductive feeder impedance .............................. 96
  6.1.3. Case III: Grid with resistive/inductive impedance ......................... 98
6.2. Distribution Feeder Results ........................................................................... 100
6.3. Unbalanced Voltage Management with High Penetration of Inverter Interfaced Generators ............................................................................. 103
  6.3.1. System Topology- DG Unit/Microgrid Modelling ....................... 104
  6.3.2. Results and Discussion ................................................................. 105
6.4. Summary ..................................................................................................... 107
## Table of Contents

**Chapter 7 Description of the Simulation Systems** ................................................. 109

7.1. PSIM .................................................................................................................. 109

7.1.1. Power Stage and Sensors .............................................................................. 110

7.1.2. Controller Stage ............................................................................................. 111

7.2. DIgSILENT ......................................................................................................... 117

7.2.1. DIgSILENT Simulation Language (DSL) .................................................. 117

7.2.2. DIgSILENT Programing Language (DPL) ................................................ 126

7.3. DIgSILENT vs. PSIM Model of a Single DG Unit ........................................... 127

7.4. MATLAB ........................................................................................................... 131

7.5. Summary ........................................................................................................... 132

**Chapter 8 Description of the Experimental Systems** .......................................... 133

8.1. Hardware ........................................................................................................... 133

8.1.1. Power Stage ................................................................................................... 134

8.1.2. Controller Stage ............................................................................................ 136

8.2. Software ............................................................................................................ 139

8.2.1. Background Routine ..................................................................................... 139

8.2.2. Foreground Routine ...................................................................................... 139

8.2.3. User Interface ................................................................................................ 140

8.3. Summary ........................................................................................................... 140

**Chapter 9 Conclusions** ...................................................................................... 141

9.1. Summary of Research ...................................................................................... 142


9.1.2. Combined Strategy for LV Feeder Voltage Regulation .............................. 142

9.1.3. DSRF Current Regulation without Sequence Current Extraction ............ 143

9.1.4. Flexible Oscillating Power Control ............................................................... 143

9.2. Future Work ..................................................................................................... 144
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3. Thesis Closure</td>
<td>145</td>
</tr>
<tr>
<td><strong>Appendix A Current Regulator Gains Calculations</strong></td>
<td>147</td>
</tr>
<tr>
<td>A.1. Closed-Loop Current Control with DSRF</td>
<td>147</td>
</tr>
<tr>
<td>A.2. Maximum Current Regulator Gains</td>
<td>149</td>
</tr>
<tr>
<td><strong>Appendix B Selected DPL Scripts</strong></td>
<td>151</td>
</tr>
<tr>
<td><strong>Appendix C DSP Source Code for the Experimental System</strong></td>
<td>159</td>
</tr>
<tr>
<td>B.1. main.h</td>
<td>159</td>
</tr>
<tr>
<td>B.2. main.c</td>
<td>164</td>
</tr>
<tr>
<td>B.3. vsi.h</td>
<td>194</td>
</tr>
<tr>
<td>B.4. vsi.c</td>
<td>199</td>
</tr>
<tr>
<td><strong>References</strong></td>
<td>263</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1. DG system. .................................................................................................................. 9
Figure 2.2. Voltage source inverter connected to the grid. ......................................................... 18
Figure 2.3. Schematic diagram of the three-phase DG unit and its corresponding control architecture. .......................................................................................................................... 20
Figure 2.4. Three-phase $dq$-PLL .......................................................................................... 21
Figure 2.5. Stationary frame current regulation. ....................................................................... 22
Figure 2.6. Synchronous frame current regulation for positive sequence only. .............. 23
Figure 2.7. DSRF current regulation for current sequence control ........................................ 23
Figure 2.8. SRF current regulation reinforced with resonant controller ................................ 24
Figure 2.9. Oscillating powers during unbalanced voltage conditions. ................................... 25
Figure 3.1. LV Distribution Feeder Configuration. ................................................................. 33
Figure 3.2. PV generation and load profile of one bus during a typical summer day ....... 33
Figure 3.3. Volt-Var control mode .......................................................................................... 35
Figure 3.4. Characteristic power factor ..................................................................................... 35
Figure 3.5. Power Curtailment Strategy .................................................................................... 40
Figure 3.6. End Bus #20 voltage – Existing Q Injection Strategies ........................................ 43
Figure 3.7. Feeder losses – Existing Q Injection Strategies ....................................................... 43
Figure 3.8. Power curtailed by PCS for each consumer ............................................................ 43
Figure 4.1. Daily generation variation of a typical PV DG system with cloud cover changes [99]. ................................................................................................................................. 47
Figure 4.2. (a) PV inverter capability curve. (b) Relationship between inverter size and its reactive power capability .................................................................................................................. 47
List of Figures

Figure 4.3. Tap transitions (up/down) for different load characteristics (inductive/capacitive) ................................................................. 49
Figure 4.4. Tap transitions for inductive load, (top) tap down and (bottom) tap up...... 50
Figure 4.5. Tap transitions for capacitive load, (top) tap down and (bottom) tap up .... 50
Figure 4.6. Simple model of power transmission between two nodes ................... 51
Figure 4.7. Voltage variation at Bus R with power flow .................................... 54
Figure 4.8. Tap changer control algorithm .......................................................... 54
Figure 4.9. LV Distribution Feeder Configuration (repeated here for clarity) ........ 56
Figure 4.10. Case 1: Feeder voltage profile: without tap changing (Top); with centralized tap control (Middle); with local tap control (Bottom) ......................... 57
Figure 4.11. Case 2: Grid impedance = 1.5Zb . Feeder voltage profile: without tap changing (Top); with centralised tap control (Middle); with local tap control (Bottom). 58
Figure 4.12. Case 3: Distributed PV injection. Feeder voltage profile: without tap changing (Top); no tap changing with Grid impedance Z = 1.5Zb (Middle); local tap control with Grid impedance Z = 1.5Zb (Bottom) ........................................ 59
Figure 4.13. PV generation and load profile of one bus during a typical summer day (repeated here for clarity) .......................................................... 60
Figure 4.14. End Bus #20 voltage – Proposed Strategy ........................................ 61
Figure 4.15. Feeder losses – Proposed Strategy .................................................. 61
Figure 4.16. Reactive power load profile vs. reactive power capacity provided by PV inverter ............................................................................. 62
Figure 4.17. Feeder voltage profile when maximum reverse power is flowing in the feeder .............................................................................. 62
Figure 4.18. Total power transfer to the grid via transformer (positive/negative power means that power is exported/imported) .............................................. 63
Figure 4.19. End Bus #20 voltage with cloud cover changes – Proposed Strategy- Method 1 ................................................................................. 64
Figure 4.20. End Bus #20 voltage for different line characteristics and Kpos/Kneg .. 65
Figure 5.1. Three phase VSI structure for a DG system (repeated here for clarity) ..... 68
Figure 5.2. Current Control using Extracted Sequence Currents and Double Synchronous Reference Frame (DSRF) Regulators ................................................. 72
Figure 5.3. SOGI-QSG ......................................................................................... 72
Figure 5.4. Sequence current references in their corresponding frame of references using (4). (upper trace) Positive sequence \((id_p\_refPSRF\) and \(iq_p\_refPSRF\)). (lower trace) Negative sequence \((id_n\_refNSRF\) and \(iq_n\_refNSRF\)) .................................................. 74

Figure 5.5. Injected currents for PSRF and DSRF with sequence extraction. (upper trace) Positive sequence \((id_pPSRF\) and \(iq_pPSRF\)). (lower trace) Negative sequence \((id_nNSRF\) and \(iq_nNSRF\)) ................................................................. 74

Figure 5.6. Proposed unbalanced DG current control using DSRF without Sequence Current Extraction................................................................................................................................. 75

Figure 5.7. Phasor diagram of the sequence components. ................................................................................................................................. 75

Figure 5.8. Projection of sequence current references on the opposite frame of references. (upper trace) Projection of positive sequence current on the NSRF \((id_p\_refNSRF\) and \(iq_p\_refNSRF\)). (lower trace) Projection of negative sequence current on the PSRF \((id_n\_refPSRF\) and \(iq_n\_refPSRF\)) .................................................. 78

Figure 5.9. Sequence current references (dotted lines) and the injected currents for PSRF and DSRF without sequence extraction. (upper trace) Positive sequence \((idPSRF\) and \(iqPSRF\)). (lower trace) Negative sequence \((idNSRF\) and \(iqNSRF\)) .................................................. 78

Figure 5.10. Step-up transient response. (left) simulation. (right) experiment. (top) one-phase grid voltage sag condition. (bottom) corresponding grid-side current \(ig\) ............. 82

Figure 5.11. Step-up transient response. (left) simulation. (right) experiment. (top) two-phase grid voltage sag condition. (bottom) corresponding grid-side current \(ig\) ............. 82

Figure 5.12. Experimental results: Three-phase current (top) & PCC voltages (bottom). 83

Figure 5.13. Experimental results: Three-phase current (top) & PCC voltages (bottom). 83

Figure 5.14. Proposed DG converter system with double synchronous frame controllers and actively damped LCL filter system ................................................................. 88

Figure 5.15. Experimental faulty grid voltages, where one phase sags to 50% (created by hardware grid simulator). .................................................................................................................. 89

Figure 5.16. Oscillating Power Controller Performance (PSIM simulation). Three-phase currents (upper trace). Real and reactive powers (lower trace). ................................. 89

Figure 6.1. Three-phase grid-connected DG inverter. .................................................. 93

Figure 6.2. Experimental results for grid with resistive characteristic. (left) Real power injection. (middle) Reactive power injection. (right) combined PQ injection ................. 95

Figure 6.3. Voltage magnitude of sagging phase: resistive grid........................................ 95

Figure 6.4. Experimental results for grid with inductive characteristic. (left) Real power injection. (middle) Reactive power injection. (right) combined PQ injection ................. 97

Figure 6.5. Voltage magnitude of sagging phase: inductive grid........................................ 97
List of Figures

Figure 6.6. Experimental results for grid with mixed RL characteristic. (left) Real power injection. (middle) Reactive power injection. (right) combined PQ injection.......... 99

Figure 6.7. Voltage magnitude of sagging phase: mixed RL grid. ........................................... 99

Figure 6.8. 20-bus distribution feeder (Balanced). ................................................................. 100

Figure 6.9. Injected power of DG units to the feeder. ........................................................... 101

Figure 6.10. Synchronous generator powers................................................................. 101

Figure 6.11. Voltage profile along the feeder. ................................................................. 101

Figure 6.12. Distribution feeder configuration (Unbalanced)................................. 102

Figure 6.13. Simulation results for distribution feeder. Reactive power (left) and real power (right) injection to utility grid for each DG unit. .................. 103

Figure 6.14. Feeder configuration .................................................................................. 104

Figure 6.15. VUF at each DG LV Bus PCC. ........................................................................ 106

Figure 6.16. Real and reactive power injected by DG units. ................................. 106

Figure 7.1. PSIM schematic diagram of VSI switching converter. ......................... 110

Figure 7.2. PSIM schematic diagram of LCL filter.......................................................... 111

Figure 7.3. PSIM schematic diagram of electricity grid.................................................. 111

Figure 7.4. Transformations and sequence extraction..................................................... 113

Figure 7.5. PLL structure in PSIM.................................................................................. 114

Figure 7.6. Instantaneous power calculation and reference generating ($K$, $P_{ref}$ and $Q_{ref}$) diagrams.......................................................................................................................... 114

Figure 7.7. Reference current generation for PSRF and NSRF................................. 115

Figure 7.8. DSRF current regulation circuit diagrams............................................... 116

Figure 7.9. A three-level PWM modulator in PSIM...................................................... 116

Figure 7.10. Power stage of the three-phase DG unit (single line presentation). ...... 118

Figure 7.11. Scheme of the DG converter controller implemented in DIgSILENT (Balanced). .................................................................................. 119

Figure 7.12. Scheme of the DG converter controller implemented in DIgSILENT (Unbalanced)........................................................................... 120

Figure 7.13. PLL Frame. ................................................................................................. 121
Figure 7.14. *abc* to *dq* Frame (Unbalanced). ................................................................. 122
Figure 7.15. *PQ* Frame (Unbalanced). ............................................................................... 123
Figure 7.16. *PQ* Frame (Balanced). ................................................................................... 124
Figure 7.17. *Current* Frame (Unbalanced). ......................................................................... 125
Figure 7.18. *Current* Frame (Balanced). ............................................................................ 125
Figure 7.19. Sampling procedure. .......................................................................................... 126
Figure 7.20. Simulation Results. Left: DlgSILENT Results. Right: PSIM Results. .... 128
Figure 7.21. Phase A current of both simulations. ................................................................. 129
Figure 7.22. Simulation results. Top: Inverter-side and filter capacitor currents. Middle: Grid-side three-phase current. Bottom: Real and reactive powers injected to the grid. 130
Figure 7.23. DPL elements. .................................................................................................. 131
Figure 7.24. Transformer with electronic tap changer modeled in MATLAB. ........ 132
Figure 8.1. Block diagram of experimental system. ................................................................. 134
Figure 8.2. Three-phase H-bridge and controller boards. ...................................................... 135
Figure 8.3. LCL filter (converter side inductor). ................................................................. 135
Figure 8.4. LCL filter (grid side inductor). ............................................................................ 136
Figure 8.5. LCL filter (common mode inductor). .................................................................. 136
Figure 8.6. Grid simulator (California Instruments MX30 programmable AC source). 137
Figure 8.7. MegaPower DC supply .................................................................................... 137
Figure 8.8. Functional diagram of CPT-E13 [19]. ................................................................. 138
Figure 8.9. Functional diagram of CPT-DA2810 [19]........................................................... 138
Figure A.1. A reference frame current regulator with cross coupling. .............................. 147
Figure A.2. Dual loop regulator with capacitor current active damping. ....................... 149
LIST OF TABLES

Table 2.1. Comparison of power control strategies. ........................................... 28
Table 3.1. System Configuration. ........................................................................... 34
Table 4.1. System Configuration. ........................................................................... 56
Table 5.1. System Parameters. ................................................................................. 80
Table 5.2. Current References for Figure 5.10 and Figure 5.11. ................................. 81
Table 6.1. System Parameters. ................................................................................. 93
Table 6.2. DGs Parameters ....................................................................................... 102
Table 6.3. DGs Parameters ....................................................................................... 105
Table 7.1. DG Parameters. ...................................................................................... 128
# Glossary of Terms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Alternating Current</td>
</tr>
<tr>
<td>abc</td>
<td>Three-phase stationary reference frame</td>
</tr>
<tr>
<td>ADC</td>
<td>Analogue to Digital Converter</td>
</tr>
<tr>
<td>CPT</td>
<td>Creative Power Technologies</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>DDSRF</td>
<td>Decoupled Double Synchronous Reference Frame</td>
</tr>
<tr>
<td>DER</td>
<td>Distributed Energy Resource</td>
</tr>
<tr>
<td>DFIG</td>
<td>Doubly Fed Induction Generator</td>
</tr>
<tr>
<td>DG</td>
<td>Distributed Generation</td>
</tr>
<tr>
<td>DlgSILENT</td>
<td>DlgSILENT simulation package</td>
</tr>
<tr>
<td>DNO</td>
<td>Distribution Network Operators</td>
</tr>
<tr>
<td>DPL</td>
<td>DlgSILENT Programing Language</td>
</tr>
<tr>
<td>dq</td>
<td>Direct and Quadrature (rotating reference frame)</td>
</tr>
<tr>
<td>DSL</td>
<td>DlgSILENT Simulation Language</td>
</tr>
<tr>
<td>DSOGI</td>
<td>Dual Second Order Generalized Integrator</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>DSRF</td>
<td>Double Synchronous Reference Frame</td>
</tr>
<tr>
<td>EMT</td>
<td>ElectroMagnetic Transients</td>
</tr>
<tr>
<td>FLL</td>
<td>Frequency Locked Loop</td>
</tr>
<tr>
<td>FRT</td>
<td>Fault Ride-Through</td>
</tr>
<tr>
<td>HV</td>
<td>High Voltage</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>IGBT</td>
<td>Insulated Gate Bipolar Transistor</td>
</tr>
<tr>
<td>L</td>
<td>Inductive (filter)</td>
</tr>
<tr>
<td>LCL</td>
<td>Inductive-Capacitive-Inductive (filter)</td>
</tr>
<tr>
<td>LTC</td>
<td>Load Tap Changing</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>LV</td>
<td>Low Voltage</td>
</tr>
<tr>
<td>LVRT</td>
<td>Low Voltage Ride-Through</td>
</tr>
<tr>
<td>MATLAB</td>
<td>MATLAB mathematical package</td>
</tr>
<tr>
<td>MPPT</td>
<td>Maximum Peak Power Tracking</td>
</tr>
<tr>
<td>MSOGI</td>
<td>Multiple Double Synchronous Reference Frame</td>
</tr>
<tr>
<td>MV</td>
<td>Medium Voltage</td>
</tr>
<tr>
<td>NSRF</td>
<td>Negative Synchronous Reference Frame</td>
</tr>
<tr>
<td>OLTC</td>
<td>On-Load Tap Changer</td>
</tr>
<tr>
<td>PCC</td>
<td>Point of Common Coupling</td>
</tr>
<tr>
<td>PCS</td>
<td>Power Curtailment Strategy</td>
</tr>
<tr>
<td>PEG</td>
<td>Power and Energy Group</td>
</tr>
<tr>
<td>PI</td>
<td>Proportional Plus Integral</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Locked Loop</td>
</tr>
<tr>
<td>PLM</td>
<td>Power Loss Minimization</td>
</tr>
<tr>
<td>PR</td>
<td>Proportional Plus Resonance</td>
</tr>
<tr>
<td>PSIM</td>
<td>PowerSIM simulation package</td>
</tr>
<tr>
<td>PSRF</td>
<td>Positive Synchronous Reference Frame</td>
</tr>
<tr>
<td>PV</td>
<td>Photovoltaic</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>QSG</td>
<td>Quadrature-Signal Generator</td>
</tr>
<tr>
<td>RPM</td>
<td>Reactive Power Management</td>
</tr>
<tr>
<td>RS232</td>
<td>Serial Communications Standard</td>
</tr>
<tr>
<td>SCI</td>
<td>Serial Communication Interface</td>
</tr>
<tr>
<td>SCR</td>
<td>Silicon Controlled Rectifiers</td>
</tr>
<tr>
<td>SRF</td>
<td>Synchronous Reference Frame</td>
</tr>
<tr>
<td>SVC</td>
<td>Static var Compensator</td>
</tr>
<tr>
<td>TI</td>
<td>Texas Instruments</td>
</tr>
<tr>
<td>VRM</td>
<td>Voltage Rise Minimization</td>
</tr>
<tr>
<td>VSI</td>
<td>Voltage Source Inverter</td>
</tr>
<tr>
<td>αβ</td>
<td>Orthogonal stationary reference frame</td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS

$C_f$ Filter capacitance
$G_x, H_x$ Various transfer functions (plant and controller)
$K_{Tap}$ Transformer tap coefficient
$K_d$ Damping gain
$K_{droop}$ Droop coefficient
$K_i$ Controller integral gain
$K_p$ Controller proportional gain
$L_f$ Inverter-side inductance
$L_{fg}$ Grid-side inductance
$P_{Loss}$ Power loss
$P_{MPPT}$ maximum PV power input
$P_{Trans}$ Power transfer at transformer terminal
$P_{i}, i \in \{1, 2, 3 \ldots \}$ Real power flow in the lines
$P_{inv}$ Output power of inverter
$P_{ref}$ Reference average real power
$P_{x}, x \in \{c, s\}$ Magnitude of the double fundamental frequency oscillating real power
$Q_{i}, i \in \{1, 2, 3 \ldots \}$ Reactive power flow in the lines
$Q_{ref}$ Reference average reactive power
$Q_{x}, x \in \{c, s\}$ Magnitude of the double fundamental frequency oscillating reactive power
$R_{x}, x \in \{grid, l\}$ Grid/line resistance
$R_{x}, x \in \{grid, l\}$ Grid/line reactance
$Tap_{nominal}$ 1pu tap position
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ap}^{pos}$</td>
<td>Electronic tap changer position</td>
</tr>
<tr>
<td>$V_{DC}$</td>
<td>DC bus voltage</td>
</tr>
<tr>
<td>$V_{cri}$</td>
<td>Critical voltage</td>
</tr>
<tr>
<td>$V_{est.}$</td>
<td>Estimated voltage</td>
</tr>
<tr>
<td>$V_{inv}$</td>
<td>Inverter voltage</td>
</tr>
<tr>
<td>$V_n$</td>
<td>Nominal line to line voltage</td>
</tr>
<tr>
<td>$V_r$</td>
<td>Receiving end voltage</td>
</tr>
<tr>
<td>$V_s$</td>
<td>Sending end voltage</td>
</tr>
<tr>
<td>$Z_x, x \in {grid,l}$</td>
<td>Grid/line impedance</td>
</tr>
<tr>
<td>$f_{samp}$</td>
<td>Sampling frequency</td>
</tr>
<tr>
<td>$i_c$</td>
<td>Capacitor current</td>
</tr>
<tr>
<td>$i_g$</td>
<td>Grid current</td>
</tr>
<tr>
<td>$i_{ref}$</td>
<td>Reference current</td>
</tr>
<tr>
<td>$i_x, x \in {a, b, c, \alpha, \beta, d, q}$</td>
<td>Current</td>
</tr>
<tr>
<td>$i_x, x \in {p, n}$</td>
<td>Sequence current component</td>
</tr>
<tr>
<td>$v_x, x \in {a, b, c, \alpha, \beta, d, q}$</td>
<td>Voltage</td>
</tr>
<tr>
<td>$\gamma_{L_C}^2$</td>
<td>LCL filter parameter ($\frac{1}{LfgC_f}$)</td>
</tr>
<tr>
<td>$\tau_d$</td>
<td>PWM transport delay time</td>
</tr>
<tr>
<td>$\tau_i$</td>
<td>Controller integral/resonant time constant</td>
</tr>
<tr>
<td>$\omega'$</td>
<td>Resonant frequency (rad.s$^{-1}$)</td>
</tr>
<tr>
<td>$\omega_0$</td>
<td>Fundamental frequency (rad.s$^{-1}$)</td>
</tr>
<tr>
<td>$\omega_c$</td>
<td>Controller crossover frequency (rad.s$^{-1}$)</td>
</tr>
<tr>
<td>$\omega_{crit}$</td>
<td>Critical resonant frequency (rad.s$^{-1}$)</td>
</tr>
<tr>
<td>$\phi_m$</td>
<td>Phase margin</td>
</tr>
<tr>
<td>$i$</td>
<td>90° time domain shift at fundamental frequency</td>
</tr>
<tr>
<td>$K$</td>
<td>Oscillating power controller factor</td>
</tr>
<tr>
<td>$n$-factor</td>
<td>Unbalanced voltage factor</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of buses</td>
</tr>
<tr>
<td>$P$</td>
<td>Real power</td>
</tr>
<tr>
<td>$P_{2\omega}$</td>
<td>Real power oscillations with twice the fundamental frequency</td>
</tr>
<tr>
<td>$Q$</td>
<td>Reactive power</td>
</tr>
</tbody>
</table>
List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_{2\omega}$</td>
<td>Reactive power oscillations with twice the fundamental frequency</td>
</tr>
<tr>
<td>$s$</td>
<td>Laplace domain variable</td>
</tr>
<tr>
<td>$S$</td>
<td>Apparent power</td>
</tr>
<tr>
<td>$T$</td>
<td>Controller sampling period</td>
</tr>
<tr>
<td>$f$</td>
<td>Switching frequency</td>
</tr>
<tr>
<td>$\delta V$</td>
<td>Voltage deviation</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Instantaneous angular phase angle</td>
</tr>
</tbody>
</table>

The use of a superscript $p$ or $n$ (e.g. $v^p$) denotes positive or negative component in PSRF or NSRF respectively.

The use of a superscript PSRF/NSRF (e.g. $i_{d}^{PSRF}$) denotes a component in PSRF/NSRF.

The use of a superscript $l$ (e.g. $p^l$) denotes load.

The use of a superscript $g$ (e.g. $p^g$) denotes DG power generation.
Chapter 1

INTRODUCTION

Across the world, the paradigm of an electricity supply grid is changing as increasing levels of renewable generation are connected to the power grid at the distribution network level. Since these small-scale distributed generation (DG) sources (Photo Voltaic (PV) solar, wind turbines and fuel cells) are generally interfaced through power electronic converters, they can also assist with the operation and control of an electrical grid network by varying both their real and reactive power injection in response to network needs. However, as the penetration level of DG into electrical grid networks continues to increase, the conventional response of these systems to disconnect in the event of a grid disturbance becomes less and less acceptable. Consequently, an ability to mitigate disturbance events, is becoming at least desirable (if not mandatory), for modern DG systems.

The objective of this thesis is to explore voltage support of an electrical grid network using power electronic interfaced DGs. This chapter describes the fundamental motivation for these investigations and the primary objectives of the research. In addition, the original contributions achieved in the course of this research are presented and the structure of the thesis is outlined.

1.1. Motivation

Conventional electrical systems are changing by moving to incorporate distributed energy sources located at or near end users. Consequently, the nature of power transfer from central power plants through transmission and distribution networks to load centres is changing, and bidirectional power flow is becoming more commonplace in the electrical network. This integration of DG sources into the distribution system takes
advantage of the nature of renewable energy resources to electrify remote areas and provides a backup source of power in case of large scale deficiency of supply from the main grid network.

These changes create new challenges for voltage and power management and protection issues, all of which can affect the reliability of the system. By transitioning a distribution network from passive to active, and due to the stochastic nature of solar irradiance, control actions are dynamic in nature. Furthermore, since photovoltaic (PV) units are generally considered uncontrollable (non-dispatchable), DG interconnections need to offer high levels of robustness and flexibility. Otherwise, DG PV penetration levels can be limited or sometimes disrupted as a common solution by utility operators to overcome voltage regulation problems.

The driving force behind this study is to enhance the control functionality of DG inverters to provide voltage support under both balanced and unbalanced voltage conditions. The second focus for the work is to explore how to integrate DG sources into an LV feeder to maintain system voltages within their regulatory limits and thus ensure more secure power delivery to the consumers.

1.2. Objectives

This thesis has three primary objectives, as follows.

Voltage Regulation of LV Feeders with High Penetration of DG PV

Voltage regulation of LV feeders with a high penetration of DG PV is becoming a major challenge for utility network operators. The importance of this topic is illustrated by the increasing interest in the literature on the control of LV feeders, taking into account PV inverter capabilities to inject reactive power into the grid and considering strategies such as total power loss minimisation and voltage regulation, sag/swell and voltage balance factors. The most common approaches to manage the feeder voltage are to curtail the real power and to manage the reactive power provided by PV inverters. However, these approaches usually have the key assumptions that rapid communication capability is readily available, local changes in PV cloud cover do not substantially influence the performance of the DG generators, enough reserve PV reactive power is available when it is most needed and prediction of the hourly available generation from the DG units for each day in advance is possible. Determining the optimal solution in
this way can make the controller quite complicated and vulnerable to a poorer quality performance because of the underlying assumptions.

This thesis presents a comprehensive investigation on DG PV capabilities to contribute to voltage regulation in LV feeders, for balanced voltage conditions. The effectiveness of DG reactive power injection is explored and it is shown that it is in many cases insufficient for voltage support. The thesis then proposes a combined voltage regulation strategy incorporating an electronic tap-changer into the feeder distribution transformer with local reactive power support from the DG inverters. This proposed strategy also allows power loss minimization, achieved by local reactive power support from the DG PV inverters.

Sequence Current Regulation of DG Inverters under unbalanced voltage conditions

In order to provide unbalanced voltage support, it is required to add sequence current control to the overall system control framework, which allows investigating the effect of such control on the positive and negative sequence components of grid voltages. One of the major challenges with current regulation under unbalanced voltage conditions is sequence current separation. Sequence current extraction is a complex procedure which has significant practical limitations. Such controller complications can compromise the overall performance of the inverter, particularly during transient events.

This thesis explores current regulation methods under unbalanced voltage conditions and investigates their limitations with respect to such factors as implementation and performance. A novel current regulation method under unbalanced voltage conditions is then devised which is able to accurately control current sequence components without requiring the current sequences to be explicitly extracted. This achieves significantly improved inverter current regulation performance with a much simpler controller implementation.

Quantification of Oscillatory Power Set-points for Voltage Support under Unbalanced Voltage Conditions

Real and reactive power control of DG systems when the Point of Common Coupling (PCC) voltages are unbalanced has been investigated in the literature. However, the relationship between these oscillating powers and unbalanced voltages has not been sufficiently quantified to create a generic approach that can provide
flexible power control while mitigating unbalanced voltages. This has a significant impact on the controller design and the level of voltage support that the DG system can offer during such abnormal conditions.

This thesis provides an analytical quantification of real and reactive power control during asymmetrical voltage conditions. The result is used for the formulation of a generalized power ripple control strategy, its experimental realization and the assessment of the impact of different ripple control regimes and feeder characteristics on the voltage unbalance factor for a distribution feeder.

1.3. Identification of Original Contributions

This thesis presents a number of novel contributions to the voltage control of LV feeders, power control and closed-loop current control of DG systems under balanced and unbalanced voltage conditions, as follows.

1. Comparison of exemplar voltage control strategies using DG reactive power and identification of limitations on LV feeder voltage support

A platform is developed which allows the modelling of DG systems in detail to investigate the voltage profile of an LV feeder with large PV penetration. In addition, the integration of the four exemplar voltage control strategies from the literature into a coherent framework that allows them to be directly compared, taking into account total feeder losses and under/over voltage mitigation.

2. Development of a new method for voltage regulation of LV feeders using transformer electronic tap changing, with PV inverter reactive power injection

In the proposed control scheme, primary voltage regulation incorporates an electronic tap-changer into the feeder distribution transformer in presence of DG PV to better accommodate cloud effect and high PV penetration levels while minimizing feeder losses using PV inverter reactive power support.

3. Introduction of a novel Double Synchronous Reference Frame (DSRF) current controller

A new sequence current control strategy is introduced in the synchronous frame that eliminates the need for sequence current separation. This allows implementation of
a simpler controller while still achieving excellent inverter performance, in particular during transient events.

4. **Formulation of a generalized power ripple control strategy under asymmetrical voltage conditions**

   Comprehensive system modelling is presented that relates positive/negative sequence and dq frame quantities to six real and reactive power flow quantities (integrating a power control strategy with synchronous frame current regulators). The proposed formulation enables a generalized ripple control strategy to be developed with only one gain parameter, while simplifying the task of choosing between active power ripple elimination, reactive power ripple elimination and current unbalance elimination. Moreover, the optimal operational power control is explored to maximize unbalanced voltage support.

1.4. **Thesis Structure**

   The material presented in this thesis is organised as follows.

   Chapter 1 (this chapter) provides an introduction to this thesis, presents the motivation and objectives of this research work, and identifies the contributions of this thesis.

   Chapter 2 presents a literature review on sequence current regulation and ripple power control to provide voltage support during unbalanced voltage conditions with a particular focus on synchronous frame controllers. Furthermore, it describes several methods proposed for LV grid voltage regulation considering large penetration of Distributed Energy Resources (DERs) into the grid.

   Chapter 3 provides an overview of different methods proposed for LV grid voltage regulation with high DER penetration into the grid. The chapter then explores several exemplar reactive power control strategies that have been proposed for voltage regulation of distribution feeders with the presence of high DG PV penetration levels, identifying their limitations on the level of voltage support achieved and their impact on feeder losses.

   Chapter 4 identifies the importance of including a fast voltage regulation mechanism into the system control. A combined method is then developed using a
transformer electronic tap changer to regulate the feeder voltage, with the local reactive power reserve used to minimise losses. This method allows the accommodation of a high penetration of DG power into the grid while accommodating fast changes in PV output power caused by the cloud shear effect.

Chapter 5 firstly explores sequence current regulation methods to achieve full control of the negative sequence current as well as the positive sequence current, which consequently translates to oscillating power control. Utilising this understanding, the impact of sequence decomposition is identified, showing how it compromises the controller transient performance. A simpler sequence current regulation method in the \ is then proposed that does not require sequence current extraction while still achieving excellent dynamic response. Then, this developed sequence current regulation method is used to control the sequence components. A power ripple control strategy in the Synchronous Reference Frame (SRF) is developed that allows real or reactive power ripple control or balancing of the feeder current. Lastly, a combined method is proposed, which has been used to investigate the impact of each control strategy on unbalanced voltage mitigation.

Chapter 6 uses the unbalanced voltage support concept developed in previous chapters to investigate unbalanced voltage mitigation with a high penetration of DG systems while exploring the performance of multiple DG inverters connected to the local network. This chapter provides a detailed design procedure for grid connection of DG systems which can be used for large scale system modelling.

Chapter 7 provides a description of the simulation systems used in this study which were used to obtain the simulation results presented in the previous chapters.

Chapter 8 provides a description of the experimental systems developed to validate the simulated and analytical results presented throughout the thesis.

Chapter 9 provides a summary of the work presented. Important contributions are reviewed and proposals are made for future research work.
Chapter 2
LITERATURE REVIEW

The impact of DG systems on electrical networks has been a popular topic for investigations in recent years, with an abundance of research on high DG PV penetration into LV feeder, voltage quality enhancement under unbalanced voltage conditions, real and reactive power support of DG inverters for grids with asymmetrical voltages and current control of DG systems under abnormal conditions. The review material in this chapter is presented in three sections.

Section 2.1 explores how conventional electrical systems are changing from centralised power plants to decentralised systems. The key identifications from this section are that:

- Control of a distributed system and its operation is complex and although there are some guidelines in the literature there is no clear optimal approach,

- The operation of DERs coordinated to the rest of the distributed system elements requires more consideration to make it effective for voltage support.

Section 2.2 looks into the integration of DG PV units into the utility grid, with a special focus on voltage profile when coordination between DG systems is considered. The key identifications from this section are that:

- There are numerous approaches to control the real and reactive power of DG PV to regulate the voltage profile along the LV feeders; still there is no comparative tool which can investigate the contribution of each strategy to the main performance aspects of the feeder such as losses and voltage rise/fall,
• Voltage regulation of such systems with distribution transformers with an electronic tap changer has not been investigated, particularly taking into account the clouding effect which affects the PV profile.

Section 2.3 reviews various DG control methods that have been proposed for operation under unbalanced voltage conditions. A detailed literature review is provided in this section of real and reactive power control and closed loop current control under unbalanced voltage condition. The key identifications from this section are:

• There are many variations on real and reactive power management under unbalanced voltage conditions with subtle similarities and differences. However, a general approach for the synchronous frame of reference has not been readily identified,

• Unbalanced current regulation in the synchronous frame is not optimized to operate with reduced complexity and to achieve a better performance, in particular for transient response.

2.1. Electrical Network and Distributed Generation

The current model for an electricity network (generation and distribution) [9, 10] is dominated by centralized power plants that can be hundreds of kilometres away from the actual users of the primary power source. Therefore, centralized power generation models require power distribution from the upstream (generation sources) to the typically distant downstream locations. This requires long distance transmission lines and a complex network of load distribution feeders.

This system of centralized power plants has significant disadvantages [11]. In addition to the transmission distance issues, it contributes to greenhouse gas emission [12], inefficiencies and power loss over the lengthy transmission lines, environmental issues where the power lines are constructed, and security related issues [13].

By locating the source near or at the end-user location, these issues can be mitigated using distributed generation sources [14, 15]. Distributed generation is an approach that employs small-scale technologies to produce electricity close to the end users of the power. Typically, DG technologies consist of renewable energy generators, and in many cases, lower-cost electricity can be provided utilising distributed
generators. In addition, higher power reliability and security is possible compared to traditional power plants [15].

A DG system can also provide electricity to replace the primary source if it fails, thus allowing the customer’s local facility to continue to operate satisfactorily during the outage [15]. Furthermore, DG units can help delay the purchase of new transmission or distribution infrastructures, including equipment such as distribution lines and substations [16].

Figure 2.1 shows a typical DG system including roof-top PV panels, wind and solar farms that are connected to the primary source of power via transmission lines [17]. A DG unit consists of a first level of energy conversion which can be any type of DERs such as photovoltaic, wind or microturbines. The next level is power conversion or conditioning performed by static power converters or generators which are connected to the utility systems via filters, isolation transformers and protection devices. Power electronic systems are one of the main parts of DG units, particularly when renewables such as solar or wind power are utilized [17]. DC/AC power electronic converters are commonly used as the power electronic interface for connection to the electrical grids (PCC) [17]. Typically, these incorporate a voltage source inverter formation of switches.

It is not only the requirements related to renewable energy sources that determine the specification of a power electronic interface but also the impact on the overall performance of the power system, especially where the DERs constitute a significant

![Figure 2.1. DG system.](image-url)
part of system capacity [17].

2.1.1. New Era in System Control and Management

Conventional distribution systems are generally designed to operate without any power generation at the distribution level, whether they are network type systems as in urban areas, or radial type systems as in rural areas. Integration of DG systems via power electronic converters changes this passive characteristic of the electrical network.

A network with distributed generation is considered to be an active network, within which bi-directional power flow occurs in the system. Consequently, in such systems, management and control of power flows and the voltage conditions at the end user connections are significantly affected [14]. With high levels of penetration, managing such a dynamic set of resources opens a new era where the basic distribution paradigm changes. One approach to manage such a system is to break it down into small clusters (microgrids) [16]. Consequently, in the case of disturbances, DGs and their corresponding loads can separate from the distribution system to isolate the microgrid. Therefore, the service to the loads is maintained without any loss of supply. Furthermore, the ability of DG systems to cope with disturbances can help to mitigate the propagation of the disturbance into the rest of the system [16].

On the one hand, centralised controllers are proposed to manage the system including the distributed resources. By monitoring the entire system, the optimal operation point for each DG is determined and consequently high system reliability is achieved [18]. However, peer-to-peer concepts that require no single critical component for operation of the microgrid are often preferred, such as a master controller [15]. This implies that with the loss of any microgrid component, the system continues to operate. Therefore, each DG must be able to have an effective response to system changes without requiring data from the rest of the system. The plug-and-play model is the other aspect of the DG systems which implies that, without requiring any re-engineering in the control of other microgrid components, a new unit can be added to any place in the microgrid [19]. Many studies assume that all the key functions to control the microgrid are available, which currently does not exist.

Based on the DG characteristics and the distribution system operational characteristics, the impacts of DG on a distribution system can manifest either
positively or negatively [14]. Among the positive impacts, voltage support, improved power management and loss reduction can be identified [20, 21]. However, achieving these benefits is much more challenging in practice. These challenges can be at a system level or they can be the operating criteria that DG Units should meet [22].

2.1.2. The Challenge of Power Management

High level power quality is essential for industrial, commercial and residential customers. One of the proposals to create a reliable power system is to have two-way control using smart meters to meet customer demands [16]. The complex control structure that is required to implement such load/generation control scheme, makes it less attractive for system operators. It is the DER integration to the distribution systems that needs to be robust to increase the system reliability and power quality as a consequence. Therefore, traditional grid control approaches no longer apply to DG systems. They must provide flexibility for bi-directional flow of power, and be able to take advantage of all reserve real and reactive power to enhance the voltage quality when it is most needed.

The inverter control strategy has a significant impact for providing high power quality. Its role is more even critical when power management is required for voltage support. To compensate for over/under- voltage conditions, real and reactive power management contribute either positively or negatively based on the control approach. Even under unbalanced voltage conditions, it is the operation of the DG system that determines whether it is possible to mitigate such conditions or not.

2.1.3. The Challenge of Voltage Regulation

Radial feeders are designed for radial power flows from the substation to the end consumers, with the primary voltage regulation mechanisms are due to is load-tap-changing (LTC) transformers, switched capacitors and supplementary line regulators. But DG changes the situation as it can interfere with the primary voltage regulators [14, 20]. Consider a DG just after the LTC transformer that incorporates considerable line drop compensation. Consequently, the voltage is set to a lower value which is not enough to maintain the voltage levels at the end of the feeder. On the other hand, a DG
can result in reverse power flows in the system which can raise the voltage levels and may even exceed the upper voltage limits [23].

With regard to the future integration of large scale DG PV systems, fluctuations in the solar irradiance which translate into feeder voltage fluctuations become an important issue [24]. Due to the stochastic nature of photovoltaic power, a dynamic control approach is required to respond quickly to PV power changes.

It is important for the DG unit control to be able to have robust performance during the events created by the distribution system such as voltage sag and unbalanced voltage conditions. Their reliability should also be improved by adding fault ride through capabilities to their control structure to not only maintain the system services during faults, but also to help to mitigate such conditions [25].

This section has shown that future grids with high penetration of DG systems change the conventional paradigm of the electrical system and how reshaping this concept requires rethinking about how to design and control the DG units. Furthermore, it requires consideration of how to integrate all these DG systems together to be able to interact with each other safely within the microgrid. The main focus of this thesis is on power and voltage management with presence of DER systems from both the DG unit and the system point of view, particularly during unbalanced voltage conditions.

2.2. DER Integration into the Utility Network

It is not only the design of the DG unit that matters, but also it is important to investigate how a large number of DG units behave and interact in a distribution system [23, 26-28]. It is a major concern that the presence of a DER in the distribution level can interfere with conventional grid control algorithms.

2.2.1. Voltage Control

With rising levels of penetration of distributed photovoltaic generation into LV feeders, an emerging concern for electrical utility companies is how to maintain the voltage profile along the feeder within regulatory limits throughout the day [29-31]. Conventionally, voltage regulation along an LV feeder is achieved primarily by design. In this approach the voltage drop along the feeder is calculated for the connected load profile, the feeder conductors are sized to limit this voltage drop to acceptable limits,
and the tap setting of the feed-in distribution transformer from the Medium Voltage (MV) network is set sufficiently high to ensure the end point feeder voltage is still above the minimum regulatory limit. Typically, this will result in a feed-in voltage from the distribution transformer of between 1.02 and 1.05 pu, and an end point feeder voltage of between 0.95 and 0.98 pu, i.e. a voltage drop along the feeder of about 0.05-0.07 pu under full load conditions. Load fluctuations during the day will cause this end point voltage to rise and fall, but always between a maximum limit of the feed-in transformer set-point voltage, and a minimum (designed) voltage limit under full load conditions.

As DG PV penetration levels increase into a feeder, this voltage regulation approach becomes less effective [32, 33]. During the day, when the residential load demand is reduced, PV generation is at its peak, and power flows in the reverse direction from the end of the feeder to the supply point. This causes the voltage at the end of the feeder to rise with respect to the distribution transformer connection point, potentially causing over-voltages if the transformer tap setting is too high [23]. On the other hand, as the evening load peak increases and PV generation reduces, power flows along the feeder towards the feeder end point, and the end point voltage falls with respect to the distribution transformer connection point. This can cause an under voltage if the transformer tap setting is too low. Typically, one single tap setting will not satisfy this wide range of load conditions. Furthermore, a rapid capability to correct voltage profile variations is required, because moving clouds over the PV arrays of the DG systems can cause rapid changes in the PV injected power within only a few seconds [24, 34], with corresponding feeder voltage fluctuations.

2.2.2. Reactive Power

A variety of studies have reported how to control a feeder voltage profile in the presence of substantial DG penetration. The typical approach is to command reactive power generation from the DG inverter units, to prevent significant voltage rises (or under-voltages) [29, 35-37]. The study in [32] shows how the use of PV inverters with a 10% increase in rating can benefit the system by leveraging the reactive power capabilities of the inverters, allowing a minimum reactive power capability of approximately 46% during full PV production. For the reactive power control method,
Chapter 2  Literature Review

sensitivity analysis is used in [38] showing that the same reactive power is more effective for voltage support when located at the end of a feeder.

Optimization methods are very popular for voltage regulation purposes and dealing with reactive power [39, 40]. Reactive power optimization control method is proposed in [40] to minimize the voltage deviations and also minimize the losses, hence showing that there is a trade-off between these two control objectives as they fight against each other.

Refs [41] and [26] further investigated the impact of high residential PV penetration on voltage profiles, considering the effects of feeder impedances, level of penetration, transformer short circuit resistance and protection and operation of the feeder. However, the common approach to reduce the slope and variation of a feeder voltage profile under high DG penetration conditions is to inject compensating reactive power. Two reactive power injection methods (constant power factor and voltage rise minimization methods) combined with an On-Load Tap Changer (OLTC) [29], are shown to be effective under different load and generation conditions, but increase the feeder losses. Furthermore, it is shown that voltage rise minimization increases the stress on tap changing when compared with the constant power factor method.

Various centralized and distributed control strategies have been proposed to address these voltage rise problems by varying the reactive power injected by the PV inverters themselves [38, 42]. However, centralised control strategies generally need a significant investment in sensors and communication systems, while localised distributed control strategies typically require relatively complex control concepts within the PV inverters to ensure that the DG systems continue to work effectively without adverse interactions [18, 43].

The use of DG systems to regulate the voltage of the grid network introduces operational constraints from the utility point of view, and the complexity of developing either a centralized or a localized distributed control scheme currently makes the Distribution Network Operators (DNOs) less inclined to use DG inverters even as secondary voltage regulators [44]. Furthermore, while a number of centralised control strategies have been proposed [40, 42, 45], they usually have the key assumptions that rapid communication capability is readily available and that local changes in PV cloud cover do not substantially influence the performance of the DG sources. Alternatively, various distributed/decentralized control approaches have been proposed to minimize
reliance on communication and achieve a faster response to environmental changes [29]. However, such local schemes cannot guarantee optimal control because they do not have access to the overall network feeder voltage profile and power flows. In addition, most reactive power injection schemes can achieve only a relatively limited range of voltage control, since the effect of reactive power injection on voltage regulation for (typically) higher resistance LV feeders is not as significant as it is for High Voltage (HV) transmission and sub transmission lines [36].

2.2.3. Power Curtailment

To manage the voltage in a distributed system with large DG PV penetration, power curtailment is proposed mainly when overvoltage may occur due to high DG penetration [33, 36, 46].

In [36] real power curtailment is proposed to prevent power quality problems with high voltage during light load situations. However, it is shown that a large amount of power might be curtailed and its distribution can be unfair between the single inverters. To resolve the issue regarding unfair power curtailment between inverters, a droop based real power curtailment is proposed in [33] with two methods. In the first method the same droop coefficient is considered for all inverters, which means the inverters further along the feeder have to curtail more real power. The second method presented results in approximately equal sharing between inverters but at the expense of higher output power losses. The risk of cascading is shown in [31, 46] where [46] proposes power curtailment to mitigate voltage rise and avoid such risks considering both the local and global control approaches.

2.2.4. DG sizing and location

In Australia, grid network operators presently manage the feeder voltage rise issue by limiting the installed PV generation capacity on individual feeders and requiring PV inverters to operate at unity power factor. The DG inverters are simply not allowed to provide any primary voltage regulatory support function to avoid adverse interactions between conventional voltage control technologies, such as upstream tap changers, and a possible DG inverter reactive power response [47]. However, this conservative strategy is not based on any substantive technical assessment and is more a matter of
expediency to avoid voltage regulation problems. Alternatively, different methods have been developed for system planning to determine the operating limits for real and reactive power instalment on each bus in a distribution system, and then selecting the candidate buses for DG placement and sizing based on clustering system buses [48].

2.2.5. Unbalanced Voltage Mitigation

The impact of distributed generation on voltage unbalance has been investigated in the literature as a common issue at the distribution level [49-51]. In [52] a two-stage coordination between OLTC and Static var Compensator (SVC) is presented for an unbalanced distribution system taking into account the system power loss, the voltage profile and the number of OLTC and SVC switching, proposing that uniform operation of OLTC and non-uniform operation of SVC is most beneficial to the system. An energy storage system is the solution proposed by [50] in order to mitigate unbalanced voltage conditions while improving the efficiency of the network. To handle unbalanced voltage conditions, [53] proposes an optimization method based on calculating the sensitivity of bus voltages to inject real and reactive power. The proposed online optimization method for voltage control considers coordination between reactive power sources.

Ref [54] considers both the real power curtailment and reactive power control of a DG inverter to improve the performance of unbalanced LV distribution networks. The proposed operational optimization strategy uses multi-objective optimal power flow to improve the voltage quality and balance the voltages while minimizing the losses and the generation costs. Alternatively, the work in [55] describes a methodology for system optimization with reactive power, proposing a three-phase optimal power flow calculation suitable for unbalanced systems. It is demonstrated that unbalanced voltages can be mitigated with unbalanced reactive power compensation along the feeder.

Distribution transformers are the other alternative to improve the voltage quality when voltages are unbalanced [51, 56]. The main control objective is individual tap changing of the transformer. In [51] Scott transformers are used for voltage balancing in LV feeders.
2.2.6. Loss Reduction

Feeder loss reduction is another factor that is commonly taken into account when considering how best to regulate reactive power injection at a local Distributed Generation (DG) level [48, 57-59].

In [58] a local adaptive control algorithm is proposed to balance the need for power quality (voltage regulation) against improving the performance of distributed reactive power control in a radial feeder. However, the main objective is to minimize losses while keep its voltages within the regulatory limits. Ref [48] generates the cost function of the optimization problem based on costs of power and energy losses and total reactive power; then determining the candidate buses for real and reactive power injection considering clustering system buses. The loss reduction is explored in [59] with an optimization algorithm as well. However, one major assumption is that hourly DG generation can be predicted at least a day in advance and also the impact on the voltage quality of the feeder has not been investigated.

2.3. Inverter Control Modelling

Inverter control has a direct impact on system operating conditions. This issue becomes more significant when inverter control structures can have either an adverse impact on the system, or improve conditions such as unbalanced voltages. Furthermore, special functionalities should be considered in a control structure design that enable the inverter to continue operation even under abnormal conditions such as asymmetrical voltages.

The control structures of DG unit systems need to be considered within the overall framework of their associated power system network [60, 61]. Typically, this is done by incorporating a simplified “averaged” DG inverter model into a standard power system analysis package. Many of these studies model the DG inverter system as an average equivalent voltage source behind reactance, similar to a conventional rotating machine. Some include a simplified closed loop regulation system for the inverter, but almost all ignore what are known from inverter studies to be important second order effects, such as modulation harmonics, sampling delays, and closed loop gain settings [62]. Hence the validity of their modelling in terms of representing the response of a real physical inverter system is at best uncertain. Thus, detailed models are required to have a clear
look at the performance of the overall system especially when system voltages are unbalanced.

Two of the main controlling aspects for a typical DG inverter working under unbalanced voltage conditions are power and current control structures which require special considerations to enable the inverter to have satisfactory performance.

2.3.1. Current Regulation Method in Synchronous Reference Frame

Current regulation in the synchronous reference frame is one of the most common approaches to control current. One of the main reasons is the direct relation between the $d$ and $q$ axes and real and reactive powers which makes it preferable for industry applications.

Figure 2.2 shows the structure of a typical three phase voltage source inverter used to connect a DG system to the electrical network. It consists of a centre tapped DC voltage source which supplies a switched three phase bridge converter. The converter outputs connect to the utility network via an LCL filter at the PCC. The converter is controlled by a high level PQ control algorithm that calculates the required currents to

![Figure 2.2. Voltage source inverter connected to the grid.](image-url)
inject commanded levels of real and reactive power for given measured grid voltages. These commanded currents are then processed by a lower level current regulator, to create commanded average output voltages for each phase leg that feed through the Pulse Width Modulator to generate the phase leg switching commands. To adequately simulate the dynamic response of such a DG inverter system, it is important to properly represent all three of these converter control systems.

For control purposes it is convenient to transform the converter variables into a synchronous rotating $dq$ reference frame using the following $abc$ to $dq$ transformation [63].

\[
\begin{bmatrix}
    i_d \\
    i_q \\
    i_0
\end{bmatrix} = \begin{bmatrix}
    \cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\
    -\sin(\theta) & -\sin(\theta - \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3}) \\
    \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}
\end{bmatrix} \begin{bmatrix}
    i_a \\
    i_b \\
    i_c
\end{bmatrix}
\]

where $i_{dq0}$ denotes the current elements in $d$, $q$ and $\theta$ axis.

This transformation allows simple PI structures to be used for the power and current controllers, since they operate on “DC” converter variables in the synchronous frame and hence can achieve zero steady-state error because of the infinite DC gain of a PI regulator [64]. Furthermore, only two regulator structures are required, in the $d$ and $q$ axes only, because the floating neutral connection of the utility grid means that the three phase currents must always sum to zero, and hence there is no zero-sequence current flowing through the inverter system.

**Control Scheme for Balanced Three-Phase DG System**

The main objective of the controller is to regulate the real and reactive power that is injected into the grid system, using a closed loop current regulator that drives a high frequency PWM switching controller. The detailed structure of the primary control system that achieves this objective is shown in Figure 2.3.

The target real and reactive power commands $P_{ref}$ and $Q_{ref}$ that feed into the system are either commanded from a remote controller, or are set to constant values for a desired level of PQ control. As indicated in Figure 2.3, $i_{abc}$ and $u_{abc}$ are measured in
Chapter 2  Literature Review

the $abc$ frame, and then converted to the $dq$ frame using (2.1). These commands are then compared against the measured real and reactive power, calculated using

$$\begin{align*}
P &= u_{d}i_{d} + u_{q}i_{q} \\
Q &= u_{q}i_{d} + u_{d}i_{q}
\end{align*}$$

(2.2)

where $u_{dq}$ and $i_{dq}$ are the measured grid voltages and currents in the $dq$ frame. The power errors are then processed by standard PI regulators to generate commanded values for the current regulators in the next level down control system.

The current controllers are conventional synchronous $dq$ frame PI current regulators, with their PI gains set to the maximum possible stable values [65]. The regulator outputs command the required modulation indices for the three phase inverter system, and are converted back to the stationary $abc$ frame and fed into an asymmetrical regular sampled PWM modulator system to control the switching of each phase leg. Simulating the control systems in this way is well established in the literature as a technique that achieves a highly accurate simulation representation of real physical converter operation, including in particular their high frequency dynamic and filter responses [65].

The Inductive-Capacitive-Inductive (LCL) filters are one of the most common filters that are being used for grid connected applications as they offer significantly improved attenuation of the PWM switching harmonics with less size and weight when compared with conventional inductive (L) filters [66-69]. These filters introduce a resonance peak into the plant frequency response which can cause resonant stability problems. Typically, either passive or active damping within the current regulator is required to avoid this instability [68, 69]. However, since passive damping increases the

Figure 2.3. Schematic diagram of the three-phase DG unit and its corresponding control architecture.
system losses, active damping by introducing a compensation term that is proportional to the capacitor current is the preferred approach.

The physical voltages and currents must always be measured in the stationary $abc$ reference frame. Since the control structure is based in the synchronous $dq$ frame, these quantities must be transformed before being used in the control loop calculations which requires continuous knowledge of the synchronous frame reference angle. This is measured using a phase locked loop as shown in Figure 2.4. Essentially, the loop compares the angle of the quadrature axis voltage $u_q$ against zero, and forces the synchronous frame angle to the value that achieves this result using a standard PI regulator [70].

**Control Scheme for Unbalanced Three-Phase DG System**

Under unbalanced voltage conditions where negative sequence components are no longer zero, it is important to regulate current set-points properly to achieve desired level of power injection. Current regulation for grid connected inverters is well-established, in particular for balanced systems where no negative sequence is present. These investigations consider different frames of references [64, 65, 71] and also many studies have taken into account the type of filtering required to enhance the performance of the current regulator.

On the other hand, current regulation of distributed generation sources in an unbalanced three phase system is a challenging task due to the appearance of negative sequence components. In order to regulate the sequence current components, the frame of reference is a key factor.

Sequence currents can be controlled in the stationary frame ($a\beta/abc$ frames) by applying the P+Resonant (PR) controller as shown in Figure 2.5 [72-77]. The negative sequence effect can be seen in the magnitude and the phase of $\alpha$ and $\beta$ axis values when

![Figure 2.4. Three-phase $dq$-PLL.](image)
Chapter 2  Literature Review

compared with balanced voltages where the currents have the same phase and magnitude in each axis. In addition to PR controllers, the proposed method in [78] uses a deadbeat current controller in $\alpha\beta$ frame to work with unbalanced voltage disturbances.

An alternative to stationary frame current regulators is Synchronous Reference Frame (SRF) current controllers which enable the controller to work with DC components rather than AC component. Hence infinite PI gains can be used to consequently achieve zero steady state error. But this fact no longer applies to systems that are working under unbalanced voltage conditions. When the voltages are unbalanced, the negative sequence component appears as a double fundamental frequency. As a result, the signal is no longer DC and applying conventional Proportional-Integral (PI) controllers would have steady state error. So sequence component decoupling is proposed to solve this issue [79]. But, under unbalanced voltage conditions the SRF-PLL becomes ineffective for sequence detection [70]. However this can be solved by using a Decoupled Double Synchronous Reference Frame based PLL (DDSRF-PLL), while [80] proposes a new frequency adaptive sequence detection technique based on a Dual Second Order Generalized Integrator PLL (DSOGI-PLL). In [81] a multiresonant Frequency Locked Loop (FLL) is proposed for grid synchronization. It consists of multiple SOGIs and a FLL to make it frequency adaptive (MSOGI-FLL).

Once the sequence components are decoupled, the positive sequence can be controlled using a single Proportional-Integral (PI) regulator which can balance the three-phase currents injected to the utility grid [72] as shown in Figure 2.6. But this strategy does not have control of the negative sequence components which are critical to control oscillating powers. So control of the negative components should be considered as well to achieve complete control of the current sequence components. Therefore, one

Figure 2.5. Stationary frame current regulation.
SRF is required to control the positive sequence component (PSRF) and a second SRF is needed to control the negative sequence component (NSRF). This is known as Double Synchronous Reference Frame (DSRF), as shown in Figure 2.7 [82-88]. An enhanced DSRF controller is proposed in [84] improving its structure by adding a decoupling network for estimating and compensating the undesirable current oscillations.

As stated above the negative sequence components appear as a double fundamental frequency on the positive frame of reference. Therefore, rather than using

![Figure 2.6. Synchronous frame current regulation for positive sequence only.](image1)

![Figure 2.7. DSRF current regulation for current sequence control.](image2)
DSRF with PI controller, a single synchronous reference frame can be used by reinforcing the conventional PI with a resonant controller tuned to the twice the fundamental frequency to compensate for the negative sequence (see Figure 2.8) [89]. When comparing [89] with [88], similarities can be identified where a reinforced strategy using resonant controllers is used in both references, except that [88] uses a dual structure instead of a single SRF as used in [89].

2.3.2. Power Control Strategies

DG systems are commonly interfaced through a power electronic Voltage Source Inverter (VSI), operating in closed loop current regulation [90]. Under balanced network voltage conditions, the real and reactive power injection from such systems is controlled by commanding from the current regulator, a current phasor that has the appropriate magnitude and phase angle with respect to the measured grid voltage (i.e. PCC). Synchronous frame or stationary frame current regulation is typically used for convenience and ease of implementation.

When the network voltages become unbalanced (because of unbalanced loads or short term disturbances [91]), power control becomes more challenging, since the negative-sequence component of the unbalanced voltage causes double fundamental frequency oscillations in both the real and reactive power injections as shown in Figure 2.9.

The real/reactive power can be decoupled to average real/reactive power ($P/Q$) and the oscillating part ($P_{2o}/Q_{2o}$) as asymmetrical voltages appear. Generally, by decoupling the positive and negative sequence components of the measured currents

![Diagram of SRF current regulation reinforced with resonant controller.](image-url)

Figure 2.8. SRF current regulation reinforced with resonant controller.
and voltages, each of these power components can be calculated. On the other hand, based on the desired level of average powers and their oscillating parts and unbalanced voltage sequence components, the resulting current references can be calculated, achieving the target level of power injection.

Various strategies have been proposed to address this issue, looking to either balance the three phase AC currents, or to minimise the real and/or reactive power injection oscillation, depending on the identified target objective [75, 78, 82, 89, 92, 93].

One of the major approaches to deal with unbalanced voltage conditions is to regulate the real output power oscillations injected into the grid [82, 83, 85, 87, 88]. This approach is highly beneficial for DC bus voltage regulation and three-phase Pulse-Width Modulation (PWM) rectifiers. In [85] a power control scheme is proposed where oscillating parts in real power ($P_{2\omega}$) and average reactive power ($Q$) vanish, while the reactive power oscillating part ($Q_{2\omega}$) remains. This eliminates dc-link voltage ripples. It is shown that the proposed method has limited degrees of freedom which prevent it eliminating real power oscillation ($P_{2\omega}$). The same control scheme is applied in [82] to deal with dc-link ripples. A similar power control strategy was also reported in [88] using four control laws to calculate current references for desired power control. The first two laws determine the input average real and reactive powers while the third and fourth control laws try to nullify the oscillating components of the output instantaneous real power. The proposed method in [83] enhances the method in [88] as it tries to

![Figure 2.9. Oscillating powers during unbalanced voltage conditions.](image-url)
determine the current references to achieve DC voltage without ripple and sinusoidal line currents. It is shown that this control scheme has nearly unity vector power factor since the power factor is not directly controlled.

Power control is a main concern in wind turbine applications, with special consideration when they are working under unbalanced grid voltage conditions [72, 94, 95]. The behaviour of a doubly fed induction generator (DFIG) for wind turbine applications connected to an unbalanced grid has been investigated in [72]. This study considers the electromagnetic torque oscillation cancellation strategy without sequence cancellation which determines the power exchange between the DFIG stator and the grid-side converter. The impact on the three-phase current and the torque is then analysed based on the power control strategy. The method proposed in [95] presents five different control approaches for power injection. Instantaneous active/reactive control regulates injected powers in a way that the oscillating parts are eliminated at the expense of injecting non-sinusoidal three-phase currents. In the other four approaches the currents are more sinusoidal but add the expense of ripple in the injected powers. The study reported in [95] does not present any strategy to inject sinusoidal current into the grid but at least eliminates ripple in real or reactive power. Consequently, there have been studies to report this issue by proposing flexible power control strategies that offer sinusoidal current injection and eliminate real or reactive power oscillations [78]. In [78] flexible active power control is proposed during grid faults using the same methodology as in [95]. The five different approaches include regulating both the real and the reactive powers (non-sinusoidal currents), real power regulation (ripples on reactive power, non-sinusoidal currents), real power regulation (higher ripples on reactive power, sinusoidal currents), reactive power regulation (ripples on real power, sinusoidal currents) and balancing the currents (ripples on real and reactive power, sinusoidal currents).

In general, to inject sinusoidal current into the grid, power ripple elimination can be applied to either the real power or the reactive power (injecting unbalanced currents). Alternatively, balanced currents can be injected with the expense of having oscillations on both real and reactive power. These general approaches can be achieved using combined control schemes as reported in [73, 75-77, 86, 89, 96].

The approach presented in [75] develops independent control strategies for the ripple content of the real and reactive power based on two simple gain parameters, and
then combines them into a joint stationary frame PQ controller. While two joint strategies have been presented, only one of these methods can offer constant real/reactive power injection. The proposed strategy is implemented in the stationary frame to enable the stationary frame current regulator. A more clear presentation on calculating the real current component and the reactive power component is addressed in [73], which is implemented in the stationary frame as well. In the proposed control scheme no harmonic distortion appears in the injected currents as the cross terms in positive and negative sequences have been eliminated. In [77] the stationary frame current references are calculated to achieve a combined control strategy proposing that the network impedance is considered. This results in a more flexible and effective power control strategy compared to the conventional methods that assume the network impedance is more inductive. It is shown that when the network impedance is mainly inductive, the same results as in [73] will be expected. An alternative to these methods is proposed in [76] which requires no Phase-Locked Loop (PLL) by introducing a notch filter. With previous methods, a coefficient is introduced to achieve combined control. But in this case, the flexibility is just between injecting regulated real and reactive powers (highly distorted currents) and injecting sinusoidal current (ripple in the real and reactive powers).

The power control strategy developed in [89] is implemented in the synchronous frame. The two main characteristics for the described method are to minimize the dc-link oscillations (regulating real power) maximizing the exchange power (unbalanced currents and oscillating real and reactive powers). Another alternative for power control in the synchronous frame is presented in [86]. Three separate control algorithms are proposed to balance the current by controlling the positive sequence component only, regulating real power and forcing real power oscillations to reduce the reactive power ripples.

Table 2.1 summarises the characteristics of each power control strategy as described above based on their six main control objectives. Note that study presented in [78] considers most of the control objectives except that it does not propose a combined method. It should be taken into account that if P and Q regulation is provided, it means that distorted non-sinusoidal currents are injected into the grid.
2.3.3. Unbalanced Voltage Mitigation

DG systems can influence their feeder PCC voltage by virtue of the control strategy that they employ, and in fact several studies have shown how to integrate Fault Ride-Through (FRT) and voltage support functions into a DG controller architecture [73, 74, 77].

For positive sequence voltage recovery and negative sequence voltage reduction, [77] proposes a FRT control strategy considering the network impedance, unlike the conventional methods which are not effective for LV grid as the network impedance is mainly resistive. However, the proposed method considers a high amount of real and reactive power ripple injected into the grid which might cause the inverter to work with over rated currents. In [73] a flexible voltage support approach is presented. A control parameter is used to modify the voltage support strategy according to the voltage sag. For three-phase voltage sag, it is recommended that the best solution is to raise the voltage in all phases, and for asymmetrical voltages, voltage equalization is preferred, as conventional strategies can lead to overvoltage. Ref [74] investigates reactive power injection for voltage support. Three different methods are compared considering their advantages and limitations. Mitigation of voltage dips (balanced and unbalanced) has been studied in [30] proposing series compensation and microgrid operation to prevent disruption of sensitive electronic equipment.

Table 2.1. Comparison of power control strategies.

<table>
<thead>
<tr>
<th></th>
<th>P Regulation</th>
<th>Q Regulation</th>
<th>P and Q Regulation</th>
<th>Balancing Currents</th>
<th>Sinusoidal Currents</th>
<th>Combined Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>[82, 83, 85, 87-89]</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>[72]</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>[78]</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>[75]</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>[73, 77]</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>[76]</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>[86]</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>[96]</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
</tbody>
</table>
2.4. Summary

This chapter has been presented in three sections, focusing on the voltage support contribution of DG systems. The first section reviewed the distributed generation introduction to the electrical system and new control concerns associated with it. The second section reviewed voltage support of aggregated DG inverters taking into account reactive power support of DG inverters, power losses, transformer tap changing and power curtailment. The third section reviewed the DG unit control strategies in particular during the unbalanced voltage conditions, including sequence current regulation and oscillating power control options.

The control approaches in the literature proposed for voltage regulation with DG penetration, show that most common ways to regulate voltage are DG PV reactive power support, power curtailment and distribution transformer tap changing. However, there are still doubts about DG PV capabilities for voltage support. Moreover, the impact of fast changes in PV power fluctuations on the voltage stability is still unclear.

Some methods have been identified in the literature dealing with positive and negative sequence current regulation. As the control design remains the same in stationary frame current regulators, there has been no further study related to stationary frame control. However, as the current regulation in synchronous reference frame changes when negative sequence component appears, there are some studies dealing with current control strategy in the SRF when the voltages are unbalanced. Most of them rely on a complex procedure of sequence element extraction.

Several major methods of power control have been presented in this literature review for unbalanced voltage conditions, with respect to real and reactive power ripple regulation. There is no particular comprehensive control strategy to provide flexible and combined power control in the SRF when the voltages are unbalanced. Furthermore, there are no coherent comparisons or identification of the optimal strategies. The contribution of different power control strategies on unbalanced voltage mitigation further complicates this lack of clarity.

There is scope to resolve many of these issues associated with DG unit control design for unbalanced voltage support and coordination of DG units for voltage regulation. Of particular focus in this thesis are:
• Investigation of different control approaches for LV feeder voltage regulation with DG penetration, considering their effectiveness for different conditions and proposing a new control algorithm which supports the fast dynamics of the grid system caused by PV power fluctuations.

• The DSRF current regulation of grid connected inverters, by eliminating the need for sequence current extraction that ensures robust performance and better transient response.

• The implications of oscillating power control strategies on the SRF that ensure the controller meets the requirements of providing flexible power control.

By addressing the issues mentioned above, the DG contribution to unbalanced voltages and voltage regulation of DG systems with high DG penetration can then be more fully explored.
Chapter 3

INFLUENCE OF PV INVERTER REACTIVE POWER ON VOLTAGE REGULATION

With increasing levels of small-scale distributed generation (DG) systems connecting into the electrical grid, there is a growing awareness of potentially adverse interactions between these systems and the grid because of their differing responses to steady state and transient network events. It is also now being recognized that the dynamics of the electrical system can depend on new factors such as the level of DG penetration.

This chapter\(^1\) explores the voltage regulation of an LV distribution network with a high penetration of DERs. Because of the bi-directional variations in real power flow that are caused by these PV systems, traditional strategies for managing the voltage profile along these feeders are becoming less effective. Hence PV inverter reactive power injection is being seen as an opportunity to improve voltage regulation without requiring upgrades in the distribution feeder.

In this chapter, some common strategies that exist in the literature (which are mainly utilizing the PV inverter reactive power) are analysed and compared to find the opportunities with each method. In particular, two important system characteristics (power loss and voltage regulation) are investigated and compared.

\(^1\) Many of the contributions contained in this chapter were first published by the author in:
3.1. PV Reactive Power Support for LV Feeder Voltage Regulation

Despite increasing levels of solar-PV penetration in electrical distribution networks, to date the inverters of these PV systems have not been significantly utilized for distribution network control. One particular area of interest for these inverters is their potential to inject/absorb reactive power to/from grid to help manage the voltage profile of their distribution feeder. Various reactive power management strategies have been proposed to address this issue using different voltage-reactive power relationships, but their effectiveness can be limited by competing performance objectives, or simply because the higher resistance characteristics of a typical LV feeder constrain the range of voltage regulation that can be achieved by reactive power injection.

Despite all the work done on investigating reactive power influence on voltage control and loss minimization, there is no comprehensive comparison tool to highlight the limitations and weaknesses of each strategy. In this chapter the performances of four exemplar PV reactive power injection control strategies are compared for a typical Australian LV feeder that are subjected to high PV penetration levels, looking at both the feeder daily voltage profile and losses. The investigation uses a detailed grid network simulation model to consider all significant factors that can influence the system voltage regulation and active power losses, such as DG power factor, line impedances, load type and level of PV penetration.

3.2. LV Feeder Model

Figure 3.1 shows the scale model of the LV network used for this investigation. It consists of a three phase radial feeder supplying 20 LV buses, 600 m long fed by a step-down distribution transformer with a tap changing capability from a typical 22kV MV grid supply. Each LV bus supplies a three phase real and reactive power load according to a predefined daily load profile (Figure 3.2), and also has a PV DG system connected with a specified real power injection rating and additional reactive power injection capability. The simulation is arranged so that different load and PV generation conditions can be readily implemented for each bus along the feeder. The injected real power for each DG unit is varied as shown in Figure 3.2 based on a standard daily solar power irradiation profile [27], and the injected reactive power is varied according to the various algorithms considered in this thesis. The DG inverters are modelled as an
average power equivalent to a switched inverter connected to its electrical bus through a conventional LCL filter, since previous work has shown this approach to be quite sufficient for this type of investigation. Standard DlgsILENT library models are used to model the distribution lines, transformer, loads and the remainder of network elements. Table 3.1 summarises the system parameters.

The operating limits for LV network voltages in Australia for steady state operation are 230/400V at 50 Hz for nominal voltage, 253/440V for the upper voltage limit and 205/356V for the lower voltage limit [97]. Hence an acceptable voltage range is between 0.89 pu and 1.1 pu, which has therefore been used for this investigation.
Chapter 3  Influence of PV Inverter Reactive Power on Voltage Regulation

3.3. Exemplar DG Unit Reactive Power Injection Strategies

The general principle of feeder voltage control using DG systems is to adjust the DG reactive power injection in response to either local or global variations of the feeder bus voltages. From the variety of approaches that have been reported to achieve this objective, the following four strategies have been selected as representative of the present state-of-the-art. Note that to allow a comparative evaluation to be conducted in this thesis, there is a need to reformulate these strategies into a common framework.

3.3.1. Reactive Power Management (RPM) [47]

RPM sets the reactive power injection levels of the PV inverters according to a predefined relationship between the inverter power factor and the LV bus voltage. One variation is to specify the required power factor according to the feeder bus voltage, as shown in Figure 3.3, and then calculate the required reactive power injection as a
function of the actual real power injection [36]. Another alternative is to calculate the “characteristic power factor” and hence the required reactive power injection, as a function of the inverter real power output, as shown in Figure 3.4. In either case, the inverter reactive power output is subject to the constraint that the inverter has the kVA capacity to provide the commanded reactive power. Otherwise the maximum possible injected reactive power is limited by the inverter kVA rating. Note that lagging power factor is defined as when reactive power flows from the grid to the inverter; that is, when the inverter acts as an inductive load from the grid perspective. However, while RPM is designed to enable PV inverters to inject or absorb reactive power, its primary objective is not to control bus voltages but only to try and reduce voltage rises along the LV feeder caused by PV real power injection.

Figure 3.3. Volt-Var control mode.

Figure 3.4. Characteristic power factor.
3.3.2. Power Loss Minimization (PLM) [58]

PLM controls the DG reactive power injection to minimise the total power dissipation in the network [42, 48, 58, 59]. For this strategy, the real and reactive power flow \((P_i \text{ and } Q_i)\) from bus \(i\) to bus \(i+1\) is approximately given by:

\[
P_i = P_{i-1} - (p_i^l - p_i^\theta)
\]

\[
Q_i = Q_{i-1} - (q_i^l - q_i^\theta)
\]

where \(p^\theta, q^\theta\) are the DG active and reactive power generations and \(p^l, q^l\) are the local active and reactive load powers. The feeder power loss is then given by ([58])

\[
P_{\text{Loss}} = \sum_{i=1}^{N-1} r_i \frac{P_i^2 + Q_i^2}{V_o^2}, N: \text{number of buses}
\]

where \(r_i\) is the resistance between node \(i\) and \(i+1\) and \(V_o\) is the voltage at the feeder transformer.

The reactive power set-points \((q_i^\theta)\) to minimize the feeder power loss \((P_{\text{Loss}})\) can be found using standard optimisation strategies. However, this requires knowledge of all power flows along the feeder, and hence needs a centralised controller and data transfer of all locally measured data to this controller. Alternatively, this control mode can be implemented locally by just supplying the local reactive loads. In this case, total power losses are reduced by limiting the power transfer in the lines, but it is not an optimal solution for minimizing the total power losses in the feeder. Also, to fully implement this strategy, it may be required to predict the hourly available generation from the DG units for each day in advance [42, 59] so that the optimal set-points can be calculated. Predicting these conditions and determining the optimal solution in this way can make the controller quite complicated, and vulnerable to a poorer quality performance because of prediction errors.

Generating the cost function of the optimization problem can include more than one factor which increases the complexity of the calculation. The cost of real power losses and the cost of total reactive power are among the elements that can change the cost function, i.e.

\[
\min_{q_i^\theta} P_{\text{Loss}} = \sum_{i=1}^{N-1} r_i \frac{P_i^2 + Q_i^2}{V_o^2} \text{ subject to }
\]
where \( V_{\text{min}} \) [\( V_{\text{max}} \)] is the minimum [maximum] allowed voltage and \( q_{DG,i|\text{min}} \) [\( q_{DG,i|\text{max}} \)] is the minimum [maximum] reactive power output of each DG unit \( (q_i^q = \text{Constr}[q_i]) \). The term \( q_i^q \) acts as the control variable which appears in \( Q_i \) and in order to minimize the power losses, \( q_i^q \) supplies the local load.

\[
\min_{q_i^q} k_p P_{\text{Loss}} + k_q \sum_{i=1}^{N} q_i^q \quad (3.5)
\]

In (3.4) it is simply attempted to minimize the amount of power loss in the network regardless of cost associated with that loss minimization. In contrast, (3.5) can be used to also consider costs associated with each variable. Coefficients \( k_p \) and \( k_q \) show the cost of power loss and cost of reactive power sources, respectively.

There is ongoing debate in the renewable generation arena about whether DNO’s should be able to define active power operating set-points for DG inverters (i.e. operating away from their MPPT for network operational reasons) \([18, 29, 42, 45]\). On the other hand, defining the reactive power set-point of the DG inverter clearly falls within the responsibility of the DNO, since it is primarily used to adjust the voltage profile of the network feeder. However, the real and reactive power operating set-points of the DG resources that are connected to a feeder bus are clearly the primary control variables for feeder voltage regulation, together with other variables such as tap position for the feeder supply transformer (assuming an on-line tap changing capability).

3.3.3. Voltage Rise Minimization (VRM) [29]

VRM directly controls the injected reactive power to prevent bus voltage rise in the presence of distributed generation \([37, 39]\). Hence this methodology can be used to attempt to achieve a flat voltage profile along the LV feeder.

For the system shown in Figure 3.1, the voltage drop between any two nodes is given by:

\[
\Delta V = V_i - V_{i+1} = \frac{R P_i + X Q_i}{V_i} + \frac{X P_i - R Q_i}{V_i} \quad (3.6)
\]
where \( R \) and \( X \) represent the line impedance between two nodes \( i \) and \( i+1 \). Since the LV feeder impedance is essentially resistive, this voltage drop is approximately given by the real part of (3.6) as

\[
\Delta V \approx \frac{R P_i + X Q_i}{V_i}
\]  

(3.7)

Hence to minimize \( \Delta V \), the reactive power injection at bus \( i \) should be

\[
Q_i = -(R/X)P_i
\]  

(3.8)

In general, of course this level of reactive power injection is usually beyond the PV inverter’s rating, and so the best it can do is to inject the maximum reactive power it has available. Another alternative to inject reactive power according to

\[
q_i^q = -(R/X)p_i^q
\]  

(3.9)

which minimizes the feeder voltage rise caused by \( p_i^q \).

VRM can be implemented with local control approaches, but since line impedance data is required for the calculation, it can be more difficult to calculate when access to the feeder line impedances is not readily available to the PV inverter. When implemented using a centralized controller, this approach becomes a similar optimization problem as for the PLM strategy.

From a mathematical point of view, this suggested cost function proposes to minimize voltage deviations without considering losses. Assume a maximum voltage deviation of

\[
\delta V = \max \left| \frac{V_i - V_0}{V_0} \right| < \varepsilon
\]  

(3.10)

where \( \varepsilon \) is a very small value as the voltage bound. With respect to (3.10), to ensure minimum voltage deviation, \( p_i^l - p_i^q + (X/R)(q_i^l - q_i^q) \) should be zero \( (q_i^q = F_i^{(v)} = Constr[q_i^l + (R/X)(p_i^l - p_i^q)]) \). Thus, the objective function can be defined as follows

\[
\min_{q_i^q} \delta V \quad \text{subject to}
\]

\[
V_{\min} \leq V_i \leq V_{\max} \quad \text{and} \quad q_{DG,i}|_{\min} \leq q_i^q \leq q_{DG,i}|_{\max}
\]

(3.11)
Chapter 3 Influence of PV Inverter Reactive Power on Voltage Regulation

The control variables for the VRM method are the same as the control variables for PLM. So the level of reactive power generation is the main control variable, and hence has been used in this study.

Note that PLM and VRM are in conflict with each other, since PLM is achieved when \( Q_l = 0 \), while VRM needs \( Q_l \) to be non-zero. This issue can be resolved to some extent by using an optimization strategy to find the optimal point for loss reduction and voltage regulation when both strategies are implemented. In general, the combination of these two objectives generates a multi-objective optimization problem as follows

\[
\min_{q^g} [P_{Loss}, \delta V]
\]

To solve such a multi-objective optimization problem, it is very common to use ad-hoc weights for the objectives [35, 45].

\[
F_i = \text{Constr} \left[ \alpha F_i^{(Ploss)} + \beta F_i^{(V)} \right]
\]

where \( \alpha \) and \( \beta \) denote weighting coefficients. However, the difficulty of using this approach is finding an appropriate definition for the coefficients. In many cases \( \beta = 1 - \alpha \) which implies that if \( \alpha = 1 \) the loss reduction (PLM) is dominant and if \( \alpha = 0 \) (\( \beta = 1 \)) then VRM is the primary objective [35, 40, 45].

3.3.4. Power Curtailment Strategy (PCS) [33]

Power curtailment (PCS) is an alternative to reactive power support, where the active power generation is constrained to prevent voltage rise along the feeder as shown in Figure 3.5. PCS is mainly used for systems where the voltage is more related to active power rather than to reactive power, such as LV systems with high resistive line characteristics. In such systems, the inverter output voltage defines how power curtailment should be implemented. When the voltage reaches its maximum allowable value, the inverter is asked to stop tracking towards maximum PV peak power and move to produce less power. If the feeder continues to face overvoltage problems, the inverter then totally stops transferring power to the grid.

Local controllers can use PCS by using a predefined characteristic based on the inverter voltage and output power as shown in Figure 3.5. Another alternative is droop-based power curtailment, where the output power of inverter is a function of inverter voltage, i.e.
where $P_{\text{inv}}$ is the output power of the inverter, based on the inverter voltage ($V_{\text{inv}}$). If $V_{\text{inv}}$ is less than a critical voltage ($V_{\text{cri}}$) the inverter will operate with maximum PV power input ($P_{\text{MPPT}}$). Otherwise, the inverter output power is defined based on the droop coefficient ($K_{\text{droop}}$) as in (3.14).

Unfortunately, PCS directly opposes a fundamental paradigm of distributed generation, which is to take advantage of distributed renewable energy sources to inject as much active power as possible, rather than limiting it. Furthermore, one of the main drawbacks of this method is how to achieve a fair power curtailment among all consumers, since consumers furthest from substation are required to produce less power compared to those who are closer to substation. However, various strategies have been proposed to offer fair power curtailment between consumers such as droop-based active power curtailment [33].

The performances of these four exemplar state-of-the-art reactive power injection strategies are now compared, looking at voltage rise across the feeder and feeder losses over a daily load and PV energy injection profile.

3.4. Simulation Results

The control strategies presented in Section 3.3 have been investigated and compared as the feeder operating conditions vary over a daily cycle. To identify the impact of each control strategy, two primary characteristics of the electrical system (voltage profile and power losses) are considered in the studies. For a worst case scenario, a constant PQ load type has been used, where the load is independent from
other factors such as bus voltage. A loading condition with power factor of 0.85 has been used, with a real power load profile as shown in Figure 3.2.

Figure 3.6 and Figure 3.7 show the performance of the previously proposed reactive power injection strategies under the PV generation and load profiles presented in Figure 3.2. Figure 3.6 shows the variation in voltage at the last bus (where the worst case scenario for the voltage profile can be seen) for the different strategies, while Figure 3.7 presents their corresponding LV feeder losses. From these results, it can be seen that VRM achieves the best feeder voltage regulation, but at the expense of a substantial increase in feeder losses. PCS is able to also adequately maintain the feeder voltage within regulatory limits, but only at the expense of significantly curtailing the DG power flow back into the grid.

The other strategies of RPM, PLM and Base (no Q injection) are unable to maintain the feeder voltage within the required limits during the peak PV generation period. This is partially because reactive power injection is not a particularly effective voltage regulation strategy for a highly resistive LV feeder, and also because the capability of the PV inverters to inject reactive power is quite limited during the day when the PV power peaks, because of inverter rating limitations. Unfortunately, this is exactly when the maximum reactive power injection capability is required.

As could be expected, the best performance for minimizing feeder losses is achieved by PCS. However, this is not a preferable approach for integrating distributed energy resources into the grid since it constrains the allowable distributed generation that can be injected. In contrast, VRM is the worst strategy for power loss minimization since it tries to maintain the voltage profile within the regulatory limits by injecting reactive power and this must increase the feeder losses. The RPM strategy performance highly depends on predefined operational power factors for the PV inverters. As a consequence, if a local load power factor deviates from unity, a better voltage profile could be expected, but with increased power losses in the feeder.

Figure 3.8 shows how significantly the PCS strategy reduces the DG PV penetration. In mid-day, when the power generation is at its peak, PCS strategy commands to curtail approximately 30% of PV power to prevent over voltages in the feeder. While PCS reduces the penetration level, there are other factors to be considered to implement this strategy as well. Many distribution feeder voltage regulation investigations assume a constant DC bus voltage behind the inverter, and ignore the
issue of how much energy is being extracted from a PV panel by the solar MPPT algorithm. The point of this comment is to recognise that any power curtailment strategy driven from a grid perspective must require the PC MPPT algorithm to “detune”, unless local energy storage is provided to store the displaced energy from the PV panel during the grid power curtailment period.
Figure 3.6. End Bus #20 voltage – Existing Q Injection Strategies.

Figure 3.7. Feeder losses – Existing Q Injection Strategies.

Figure 3.8. Power curtailed by PCS for each consumer.
Chapter 3 Influence of PV Inverter Reactive Power on Voltage Regulation

3.5. Summary

This chapter has explored the voltage regulation performance of four exemplar reported reactive power support strategies for PV distributed generation systems, identifying their weaknesses and limitations as their feeder load and PV generation vary over a daily profile. The investigation has shown that the basic approach of RPM (as proposed in standards), despite being designed to maintain the voltage within the regulatory limits, is not able to deal very well with DG penetration and low load profile conditions. Then power loss minimization (PLM) and voltage rise minimization (VRM) strategies were then explored, showing that VRM regulates the voltage but significantly increases the losses, while PLM minimizes the feeder losses but cannot maintain voltages within the regulatory limits. The last approach explored is PCS, which is an alternative to reactive power injection which curtails real power to prevent over voltages. However, it has been shown here that this strategy significantly decreases the level of DG power penetration which is not very attractive in the context of DG systems since it acts in contradiction to MPPT. Essentially, while approaches such as VRM and PCS can achieve proper voltage regulation, this is only achieved by either compromising the level of real power injection or significantly increasing the power losses in the LV feeder.
Chapter 4

LV FEEDER VOLTAGE REGULATION USING ELECTRONIC TAP CHANGER

Chapter 3 has investigated four exemplar reactive power control strategies for voltage regulation of LV feeders. It was shown that using DG PV inverter reactive power, it is possible to regulate the voltage along an LV feeder. However, it was also shown that it can usually only be achieved by curtailing a significant part of PV power or significantly increasing the power losses.

This chapter\(^1\) proposes an alternative approach by incorporating an electronic tap changer into the distribution transformer that supplies the LV feeder so that the LV supply voltage magnitude can step change on a 50Hz cycle by cycle basis. A simple algorithm then varies the transformer tap position as the power flow through the transformer changes, to keep the voltages along the feeder well within regulatory limits irrespective of the magnitude and direction of power flow. Finally, feeder loss minimization is achieved by providing local load reactive power support from the PV inverters. The strategy has been confirmed by detailed simulation investigations on feeders with a variety of characteristics, under a wide range of load and PV power injection conditions.

\(^1\) Many of the contributions contained in this chapter were first published by the author in:
4.1. Integrated Feeder Voltage Regulation and Loss Minimisation Strategy

Conventional tap changers for power system distribution transformers are mechanical devices that can switch between tapped outputs of the transformer winding to change the effective transformer turns ratio and hence its output voltage. They can be either “off-line” where they need only change tap while the transformer is de-energised, or “on-line”, where they are designed to change tap while conducting load current. Online tap changers have to maintain a continuous current path during tap changing without creating a transient short circuit between the taps, and so are more complex and expensive. They also require some seconds to change taps, and must always sequentially transition from tap to tap. These limitations make them unsuitable for rapid voltage control.

The integrated voltage regulation strategy proposed in this thesis is to incorporate an electronic tap changer [98] into the primary distribution transformer of the feeder, and use its functionality as the primary voltage control mechanism. There are several advantages for electronic tap changers. They are very low maintenance systems because there are no mechanical parts and no switching arc as a consequence. They are also high speed systems because of the nature of their solid state switches, which allow the tap to be changed once every fundamental half cycle. This means the transformer can quickly respond to rapid feeder power flow and voltage fluctuations as the PV injected generation varies rapidly as shown in Figure 4.1.

The reactive power injection of the PV inverters at each bus is then locally controlled to supply the load reactive power at that bus, within the overall kVA rating capability of the inverter as shown in Figure 4.2a. This control strategy approximately minimises the VA power flow along the feeder, and hence substantially reduces the feeder power losses. Note of course that the reactive power capacity of a PV inverter is limited by its kVA rating. But if slightly overrated inverters are installed, these inverters can provide substantial reactive power support even when operating with full active power generation [32]. For example, with a 10\% increase in inverter size, i.e. $S_{\text{rated}} = 1.1 P_{\text{max}}$, the available reactive power support can be increased from zero to about 46\% even at maximum PV power generation. This results in a power factor capability of unity to 0.91 leading/lagging as shown in Figure 4.2b. (Alternatively, it is commented
that typical inverter interfaced DGs usually operate below 90% capacity even under maximum real power generation conditions [27], so they can usually still provide substantial reactive power injection capability without requiring an overrated capacity.)

4.1.1. Explains the Electronic Tap Changer Principles

Electronic tap changing systems are a relatively newly established technology that replaces the switches of mechanical transformer tap changers with electronic switches using either back-to-back Silicon Controlled Rectifiers (SCR), or more recently IGBT’s
While these systems require more design effort to ensure their robustness in a power system application, they do offer the very significant advantages of no moving parts, and the ability to jump from one tap to any other tap repeatedly at half-cycle intervals (transitioning at the zero crossing intervals of the load current to minimise losses and switching noise). This capability makes them very attractive for high speed voltage regulation applications, particularly managing the very fast voltage fluctuations in distribution feeders caused by the rapid changes in PV power injection from DG systems that can occur as cloud cover varies (Figure 4.1).

As discussed in [98], the SCR switching sequence for an electronic tap changer requires careful timing to avoid excessive circulating current. There are four different sequences required to change a tap as shown in Figure 4.3. In general, when tapping down, the product of voltage and current at the transformer terminal (“\(v_iI\)” product) should be negative, while when tapping up, the “\(v_iI\)” product should be positive. This allows the incoming SCR pair to always pick up the load current without creating any circulating current.

For example, to tap down an inductive load or to tap up a capacitive load, the outgoing (conducting) SCRs should be switched OFF before the load current crosses zero. This immediately turns OFF the non-conducting SCR of the outgoing pair, while load current continues to flow through the forward conducting SCR for the remainder of the fundamental current half cycle. The incoming SCR pair is then turned ON just before the end of the half cycle, and smoothly picks up the load current as it transitions through the zero crossing without a significant voltage transient or excessive circulating current between the taps. On the other hand, to tap up an inductive load or tap down a capacitive load the incoming SCR switches must be turned ON significantly after the load current zero crossing. The conducting SCRs are turned OFF just after the current zero crossing so that the load current continues to flow through the forward conducting SCR. When the incoming SCR pair is turned ON, this outgoing SCR is reverse biased, and the new SCR pair smoothly picks up load current.

When the load power factor is close to unity, some transient circulating current is unavoidable, but it is small because of the minimal voltage across the taps that exists at the zero crossing point [98]. It is also important to recognise that for a physical system, current commutation between the SCR switches is not instantaneous because of winding
leakage impedances, and this needs to be taken into account when considering the detailed switching timing design for the electrical tap changer.

Figure 4.4 and Figure 4.5 show simulated tap transitions for different load characteristics. As shown in this figure, the actual voltage transition always occurs when the load current reaches zero. The result is essentially no distortion in the transformer output current, and only a small step in the transformer output voltage, as the tap position changes. Finally, since individual tap changers are used for each phase of a three phase system, separate regulation of each phase voltage of the LV feeder is of course quite straightforward, unlike many mechanical tap changing systems.
Figure 4.4. Tap transitions for inductive load, (top) tap down and (bottom) tap up.

Figure 4.5. Tap transitions for capacitive load, (top) tap down and (bottom) tap up.
4.1.2. Feeder Voltage Regulation using Transformer Tap Selection

Figure 4.6 shows a simple representation of an LV feeder, fed from a distribution transformer with online electronic tap changing capability. With this type of tap changer, the feeder voltage profile can be readily raised or lowered without requiring any consideration of speed of response (since electronic tap changers can switch essentially continuously at half fundamental cycle intervals) or restriction to a limited number of lifetime tapping cycles. Two methods are proposed here to determine the tap position.

**Method 1: Using power flow at the transformer terminals**

The tap changer position can be determined using a simple algorithm based on the active power transferred through the distribution transformer, varying the tap from a maximum allowable value (say between 1.05 and 1.1pu) when the transformer is supplying rated power along the feeder to the loads, to a minimum allowable value (say between 0.95 and 0.9pu) when the DG units are feeding back the maximum PV generated power. Hence a linear relationship can be used to change the electronic tap changer position as the power flow through the distribution transformer varies according to:

\[
\begin{align*}
Tap_{pos} &= Tap_{nominal} + K_{pos}P_{trans} \\
Tap_{pos} &= Tap_{nominal} + K_{neg}P_{trans}
\end{align*}
\]

where \( Tap_{pos} \) is the electronic tap changer position, \( Tap_{nominal} \) is the 1pu tap position, \( P_{trans} \) is the power transfer at the distribution transformer terminal and \( K_{pos}/K_{neg} \) is a coefficient based on the LV grid impedance and the grid capacity that matches the transformer power transfer to the required voltage regulation sensitivity along the
feeder. The appropriate value of $K_{\text{pos}}/K_{\text{neg}}$ for a feeder network is determined by calculating the voltage rise/fall that is required to maintain the feeder voltage profile within voltage limits for maximum designed feeder power flow in either direction (sending or receiving). Hence $K_{\text{pos}}/K_{\text{neg}}$ in fact simply defines the level of tap range required for the transformer for any given feeder impedance and load/PV generation profile.

**Method 2: End bus voltage approximation**

For a worst case condition, it is assumed for this analysis that the entire feeder load is concentrated at the second (end) bus, although of course in practice both the loads and the DG sources will be distributed along the length of the feeder.

Representing the feeder line impedance $Z$ as $Z = R + jX = Ze^{j\theta}$, the real and reactive power injected into the line from the sending end is given by:

$$
S = V_s \left( \frac{V_s - V_r}{R + jX} \right) = \frac{V_s V_r}{Z} e^{-j(\phi_2 - \phi_1 - \theta)} - \frac{V_r^2}{Z} e^{j\theta}
$$

(4.2)

Separating (4.2) into real and reactive power components gives:

$$
\begin{align*}
P &= \frac{V_s V_r}{Z} \cos(\delta - \theta) - \frac{V_r^2}{Z} \cos(\theta) \\
Q &= -\frac{V_s V_r}{Z} \sin(\delta - \theta) - \frac{V_r^2}{Z} \sin(\theta)
\end{align*}
$$

(4.3)

where $\delta = \phi_2 - \phi_1$ is the power angle across the feeder line.

For any given real power injection conditions, the voltage at the end “Bus R” can be calculated by rearranging (4.3) to give

$$
V_{r,\text{est.}} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \text{ where}
$$

$$
a = \frac{-\cos(\theta)}{Z}, \quad b = \frac{V_s}{Z} \cos(\delta - \theta) \quad \text{and} \quad c = -P
$$

(4.4)

Figure 4.7 shows how the distribution feeder voltage varies as the power sent through the feeder changes. As expected, voltage droop occurs when positive power is flowing from the grid, while reverse power flow (DG injection) causes the end voltage to increase. From Figure 4.7 it can be seen that changing the transformer tap position
changes the absolute magnitude of the “Bus R” voltage, but not its voltage droop as a function of power flow through the feeder.

From this understanding, a simple voltage regulation strategy can be proposed. At the start of a control cycle, the feeder end voltage is estimated using (4.4) and the measured injected power flowing from the feeder distribution transformer. If this voltage is below 0.89 pu, the transformer tap is increased by a sufficient step to raise it above 0.89 pu. If the end bus voltage is above 1.1pu, the transformer tap is decreased by a sufficient step to lower it below 1.1pu. The required size of the tap step can be determined using (4.4), as illustrated in Figure 4.7, which shows the tap setting required to achieve a given change in feeder end voltage for a given power injection. Note for the simple example shown in Figure 4.7, there are only three possible tap positions, and hence the decision of which tap to use reduces to a “low”, “normal” and “high” alternative. Figure 4.8 shows the overall concept of the tap change voltage control algorithm.

The effectiveness of the rapid acting tap changer has been investigated over a wide range of conditions for a 20 bus feeder as shown in Figure 4.9 with the parameters presented in Table 4.1. Three cases are presented to illustrate its performance. Case #1 presents a scenario in which all of the PV injection is concentrated at Bus #20, while at the same time there is negligible residential load demand along the feeder. This represents an absolute worst case midday scenario with maximum PV generation and minimal local demand. Case #2 explores this same loading/PV supply condition but for a feeder with an impedance that is 50% larger. This scenario is intended to explore a weak grid situation which is expected to exhibit a much greater level of susceptibility to PV penetration. Case #3 explores a more typical PV penetration scenario, whereby the PV sources are evenly distributed along to the feeder, again with negligible residential load conditions.

For all of the above described cases, three different control options are investigated:

- No dynamic tap action with the feeder transformer tap set to 1.0 pu;
- Dynamic tap setting using a centralised calculation to select the tap value that optimises the feeder voltage profile (requires measurement of all node voltages);
Dynamic tap setting using the proposed local control algorithm (Method 2, Figure 4.8) that determines the required tap position based on the measured transformer real power flow.

Cases #1 and #2 (Figure 4.10 and Figure 4.11) show a steadily rising feeder voltage profile from the distribution transformer to the feeder end bus, with an increasing end bus voltage as the level of PV injection increases. For Case #1, without transformer tap action, the end bus voltage exceeds regulatory limits when the DG

![Graph](image)

**Figure 4.7.** Voltage variation at Bus R with power flow.

![Flowchart](image)

**Figure 4.8.** Tap changer control algorithm.
injection exceeds 20%. With transformer tap control, up to 40% DG penetration can be achieved without overvoltage by lowering the tap setting as the DG generation levels increase. A similar result is achieved for Case #2, although with a reduced level of maximum allowable DG penetration because of the increased line impedance.

NOTE: The result for Case #2 illustrates operation with a weak feeder system where large amounts of PV generation are requested by customers, but there is not enough economic justification to increase the feeder rating. Rapid adaptive tap changing offers an effective and lower cost alternative voltage control mechanism that can achieve a substantial increase in the allowable PV penetration without voltage regulation problems, and without requiring the feeder rating to increase.

For the more typical distributed PV source scenario of Case #3, Figure 4.12 shows that for the base feeder impedance condition, all feeder node voltages remain within the regulatory limits even with 40% DG penetration. Overvoltages occur when the feeder impedance is increased by 50%, again corresponding to the weak grid scenario. However, these over-voltages are readily controlled to lie within the regulatory limits by changing the distribution transformer supply voltage using the proposed transformer tap changing algorithm.
Figure 4.9. LV Distribution Feeder Configuration (repeated here for clarity).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_n$</td>
<td>Nominal Voltage (L-L)</td>
<td>400 V</td>
</tr>
<tr>
<td>$n_{bus}$</td>
<td>Number of Buses</td>
<td>20</td>
</tr>
</tbody>
</table>

**Loading per Node**

| $P$ | Real Power | 4 %–40 % DG Penetration |
| $Q$ | Reactive Power | Section 3.3 |
| $V_{range}$ | Voltage Range | $0.89-0.95 \leq V \leq 1.05-1.1 \text{ pu}$ |

**DG Unit**

| $S$ | Rating | 4 kVA |
| $L_f$ | Filter | 0.1-0.25 pu |
| $L_{fg}$ | Grid-side Filter | 0.01-0.025 pu |
| $C_f$ | Shunt Filter | 7.5-15 µF |
| $L_k-k+1$ | |
| $R_l$ | Resistance | 0.27 Ω/km |
| $L_l$ | Inductance | 0.24 mH/km |

**Transformer**

| $S_{trans}$ | Rating | 200 kVA |
| $f$ | Frequency | 50 Hz |
| $n_{Tap}$ | Max/Min Tap Position | ±10 |
| $V_{tap}$ | Voltage per Tap | 1.5% |
| $P_{loss}$ | Copper Losses | 3 kW |
| $x_0$ | Leakage Reactance | 0.5 pu |
| $V_{sc}$ | Short-Circuit Voltage | 3% |
Figure 4.10. Case 1: Feeder voltage profile: without tap changing (Top); with centralized tap control (Middle); with local tap control (Bottom).
Figure 4.11. Case 2: Grid impedance = 1.5Z₀. Feeder voltage profile: without tap changing (Top); with centralised tap control (Middle); with local tap control (Bottom).
Figure 4.12. Case 3: Distributed PV injection. Feeder voltage profile: without tap changing (Top); no tap changing with Grid impedance $Z = 1.5Z_b$ (Middle); local tap control with Grid impedance $Z = 1.5Z_b$ (Bottom).
4.1.3. PV Reactive Injection for Loss Minimization

With the primary voltage control along the feeder accomplished with the electronic tap changer, the reactive power capability of the DG PV inverters can now be used to minimize the feeder power loss. The control approach used is based on PLM strategy as described in Chapter 3 Section 3.3, with the local real and reactive powers of the load measured at the DG PCC by technologies such as local smart meters. The PV inverter is then simply commanded to inject reactive power equal to the measured local load reactive power, up to the limits determined by its current level of real power injection. Note that with this strategy, there is no interaction between the primary feeder voltage control mechanism (the electronic tap changer) and PV inverter, and so the system will always be stable.

4.2. Comparative Voltage Regulation Performance of Strategies Investigated and Proposed Method

The exemplar strategies presented in Chapter 3 and the integrated strategy presented in this chapter have been investigated as the feeder operating conditions vary over a daily cycle. The end bus #20 voltage profile and feeder power losses are used as a comparative measure.

The LV feeder model under study is considered to be the same as the system presented in Chapter 3 Section 3.2 for comparison purposes. The LV feeder, PV and load profile characteristics and system parameters are shown in Figure 4.9, Figure 4.13, and Table 4.1 respectively.

![Figure 4.13. PV generation and load profile of one bus during a typical summer day (repeated here for clarity).](image_url)
The same loading condition is considered as well where a PQ load with a power factor of 0.85 and a real power magnitude profile as shown in Figure 4.13, has been used for each load bus.

Figure 4.14 and Figure 4.15 show the effectiveness of the new combined voltage regulation strategy with tap changer and local reactive power injection. The end bus #20 voltage profile is now kept well within voltage regulatory limits throughout the daily load profile, and the feeder losses are still acceptably low at 6 kW peak (which is 7.5% of the total feeder rated capacity of 80 kW). Note however that the feeder losses are somewhat higher than PLM because the overall reduced feeder voltages create slightly higher currents for the constant PQ loads that are used.

![Figure 4.14. End Bus #20 voltage – Proposed Strategy.](image1)

![Figure 4.15. Feeder losses – Proposed Strategy.](image2)
Note also in Figure 4.14 how the electronic tap changer position varies in response to changes in the level of PV penetration to keep the voltage profile of the feeder within the acceptable range. For this study, a calculation period of 10 minutes was used. In particular, it can be seen from Figure 4.14 and Figure 4.15 that compared to PLM strategy where loss is minimized but the voltage is out of limits, the proposed method is well able to maintain the voltage within the regulatory limits while still minimizing feeder losses. Figure 4.16 shows the reactive power load profile and the reactive power capacity of the PV inverter, where it can be seen that for a typical 3.5 kW inverter there is enough reactive power capacity to supply the local load even during the peak PV generation period just before noon.

![Figure 4.16. Reactive power load profile vs. reactive power capacity provided by PV inverter.](image)

![Figure 4.17. Feeder voltage profile when maximum reverse power is flowing in the feeder.](image)
Figure 4.17 shows the overall voltage profile of the LV feeder for the four exemplar strategies and the new combined strategy, for a worst case scenario when the PV generation peaks while at the lowest level of loading for the LV feeder. This condition is a worst case for the feeder voltage profile. The VRM and PCS strategies can only just maintain the voltage profile along the feeder, while the Base, RPM and PLM strategies cannot maintain the feeder voltage profile, and significant consumer overvoltage can be observed after approximately the 9th bus. In contrast the new combined strategy comfortably maintains the feeder voltage within regulatory limits by reducing the source end voltage to below 0.95 pu, without any feeder power flow operating constraints.

Figure 4.18 shows the total power transferred to the grid via the feeder transformer. The new combined strategy and the Base, RPM and PLM strategies have the best performance in terms of minimum power for the case of maximum power delivery to the grid. VRM is the next best to taking advantage of distributed generation while PCS shows the worst performance by commanding the DG units to inject less power to the grid to maintain the voltage profile along the feeder. Finally, it can have noted that when there is no DG PV penetration, the power transfer from the utility grid to the feeder is slightly higher for the new strategy than for the other strategies, because of the smoother voltage profile along the LV feeder achieved by the electronic tap changer (Note that if the voltage profile is kept more constant, power transfer may well be slightly higher than if the load voltage sags without PV support, depending on the
load type.

When cloud in an area occurs and the PV injected power varies, the tap positions may need to be switched in seconds or less to maintain the feeder voltage within regulatory limits. The electronic tap changer fills this need very effectively, since it can easily and rapidly switch between tap positions to maintain the feeder voltage profile as the PV injection levels vary. Figure 4.1 shows the rapid level of real power variation that can occur when significant cloud variation occurs over a PV panel.

Figure 4.19 shows the performance of the new combined system when all PV systems are subject to this rapid variation of injected power (this is obviously a very worst case scenario, and would be very unlikely to occur in a real system because of geographical separation between the DG PV units). Without rapid voltage control, the feeder end bus voltage will fluctuate as the cloud cover varies, often exceeding the regulatory limit of 1.1 pu. In contrast, with rapid dynamic tap changing and the new combined voltage regulation strategy, the feeder end bus voltage is easily maintained within regulatory limits throughout the day. Furthermore, unlike a centrally co-ordinated reactive power or voltage regulation strategy such as PLM, the tap changer regulation approach does not require high speed and extensive communication along the feeder, responding rapidly instead to only the real power flow through the transformer that can be measured directly at the transformer terminals.

Finally, Figure 4.20 shows the influence of the $K_{Tap}$ coefficient on the end bus

![Figure 4.19. End Bus #20 voltage with cloud cover changes – Proposed Strategy-Method 1.](image)
#20 voltage as the R/X ratio of the feeder line changes. $K_{pos}$ is used when power is transferring from the grid to the feeder, and $K_{neg}$ is used when power is transferring from the grid to the feeder. As could expected, the tap changer co-efficient must be increased as the feeder R/X ratio increases, to provide an increased range of transformer tapped voltage output that can satisfactorily control the feeder voltage profile.

The resulting system readily maintains the feeder voltage profile well within regulatory limits throughout the entire day despite substantial load/generation fluctuations throughout this period. The results show a substantial performance improvement for the new strategy compared to existing voltage control strategies, and provide an effective simulation platform to explore other reactive power control strategies.

![Figure 4.20. End Bus #20 voltage for different line characteristics and $K_{pos}/K_{neg}$.](image)
4.3. Summary

This chapter has presented a new combined voltage regulation strategy for an LV feeder under high PV penetration levels. The strategy incorporates an electronic tap changer for the feeder distribution transformer from the MV network to rapidly adjust the absolute magnitude of the feeder voltage profile in response to changes in PV Distributed Generation power injection. The strategy measures the power flowing through the transformer to calculate the tap position that will best maintain the feeder node voltages within regulatory limits, and then uses local reactive power injection from each DG inverter system to minimise feeder currents and hence minimize feeder losses.

The new system achieves excellent voltage regulation with very high levels of PV penetration and even with rapidly fluctuating PV injection levels, using only a very simple compensation algorithm based on the measured power flowing through the distribution transformer. The approach not only increases the level of DG penetration that can be achieved, but can also react to fast changes in PV injected power caused by intermittent cloud shading. This ensures a secure power delivery to the consumers without violating regulatory voltage constraints.
Chapter 5
DG Unit Current Control for Unbalanced Voltage Conditions

The previous chapter has presented voltage support control for a high penetration of DG PV systems in LV feeders where a balanced system was under study. Phasor simulation techniques were used since the system is operating under sinusoidal steady state conditions. However, the case is different for the study of systems working under unbalanced voltage conditions since such investigations require taking into account time-domain analysis, in particular for transient studies. Consequently, detailed control modelling of the DG inverter system (current regulation and power control) is required to manage unbalanced voltage conditions to enable investigations into unbalanced voltage mitigation in LV feeders.

This chapter¹ presents a consolidated control strategy that uses a double sequence frame current regulator to continuously adjust between eliminating real power oscillations, balancing the inverter three phase currents, and eliminating reactive power oscillations, when connecting to an unbalanced grid voltage.

5.1. Fundamentals of Oscillating Power Control in The SRF

The DG converter arrangement used in this thesis is a standard three phase VSI, connected to the grid network at its PCC through an LCL filter, as shown in Figure 5.1. Typically, such systems are controlled by a three level control structure – a high level

¹ Many of the contributions contained in this chapter were first published by the author in:
power control scheme, which feeds into a mid-level current regulation system, and which then in turn calculates voltage commands for the lowest level PWM controller.

For this work, only a three-wire system has been investigated. However since [75] has shown that results from a three-wire scenario are consistent with a four-wire system when a current regulator is used that eliminates zero sequence currents, the conclusions from this work can also be readily applied to a four-wire system.

For an unbalanced system, there are three possible reference strategies for the higher level controllers. The first approach is to operate in the $\alpha\beta$ stationary frame, using resonant regulators to control the inverter currents since all quantities are sinusoidal [64]. The second approach is to operate in the positive synchronous rotating $dq$ reference frame, using simple DC regulators to control the positive sequence currents. However, in this frame the negative sequence currents oscillate at twice the fundamental frequency, and hence two resonant regulators are still required in this rotating frame [82]. The third alternative is to use the positive rotating synchronous frame to control the positive sequence currents only, and to introduce a negative rotating sequence frame to control the negative sequence currents only. This arrangement requires only two simple PI control structures in each rotating frame of reference, which is straightforward to design and conceptually easy to implement [84]. However, it does require the measurement of the $\alpha\beta$ frame voltages and currents to be separated into positive and negative sequence components before the rotating frame transformation.
5.2. Double Synchronous Reference Frame Current Regulation

In the literature review in Chapter 2 a lack of fundamental knowledge on how to design and implement a current regulator for a system with unbalanced voltages in the synchronous frame was identified. In particular, it was shown how the implications of sequence current separation for current regulation are not well understood for the SRF. This section now focuses on low level control of a grid connected inverter under unbalanced voltage conditions to demonstrate its complexity when sequence extraction is involved, and then proposes a new controller which eliminates this need for sequence separation. This is particularly beneficial for riding through transient events.

Firstly, the current controller performance of a grid connected inverter under unbalanced voltage conditions is reviewed when it is implemented in a DSRF. It explores how sequence current components need to be extracted into DC components for each sequence in their respective positive or negative SRF, which can then be regulated by a standard PI controller. Then, one of the major contributions of this thesis is presented, which is an improved approach for unbalanced current regulation that requires neither sequence separation of the measured currents nor a resonant controller stage in the SRFs. The approach essentially allows simple PI positive and negative SRF current controllers to simultaneously regulate both their target sequence DC component, and the opposite sequence double frequency component. When the PI gains are then maximised according to established theory [65, 66], the two controllers smoothly coordinate to achieve precise current regulation with excellent transient performance.

5.2.1. Conventional DSRF Current Regulation with Sequence Extraction

Unbalanced grid voltages can be caused either by unbalanced loading conditions or fault disturbance influences. When the inverter of a DG source is connected to such an unbalanced grid, both the positive and negative sequence currents injected from the DG must be accurately regulated to control real and reactive power flow oscillations into the grid and to help mitigate the unbalanced voltages at the PCC [73, 75, 77, 78]. The usual strategy to achieve this outcome is to implement separate PI current regulators in the positive and negative synchronously rotating frames of reference (DSRF), to independently control the DG injected positive and negative sequence currents. However, conventional practice is that because each sequence current appears
as a double fundamental frequency component in its opposite polarity frame of reference, the measured currents must be separated into positive and negative sequence components before their transformation into their respective synchronous frames. The separated components then individually feed into their respective current regulator in their SRF, which produces a commanded voltage for the DG inverter to track the required positive or negative sequence current reference (as appropriate). Finally, these voltages are transformed back to the stationary frame and summed, to create the total commanded VSI output voltage for the inverter PWM modulator [83, 84].

The central challenge with this strategy is that the positive and negative sequence current components must be extracted from the measured currents before their transformation into their SRF. This is a relatively complex computational process that typically uses either a notch filter or a delayed phase shift transformation [83]. Both of these approaches constrain the transient response of the measurement, and create signal path delays that can significantly reduce the maximum possible PI regulator gains [84].

An alternative approach is to combine the positive and negative commanded sequence currents into hybrid reference currents in each of the positive and negative SRF current regulators. This strategy was first presented for active rectifiers in [88], but still requires relatively complex PI plus resonant current regulators in each of the positive and negative SRFs.

To operate in a DSRF, the stationary abc frame measured voltages and currents must be appropriately transformed. This can be done by transforming firstly to the stationary αβ frame using:

\[
\begin{bmatrix}
    v_{\alpha} \\
    v_{\beta} \\
    v_{\gamma}
\end{bmatrix} =
\begin{bmatrix}
    1 & -0.5 & -0.5 \\
    0 & \sqrt{3}/2 & -\sqrt{3}/2
\end{bmatrix}
\begin{bmatrix}
    v_a \\
    v_b \\
    v_c
\end{bmatrix}
\]

(5.1)

\[
\begin{bmatrix}
    i_{\alpha} \\
    i_{\beta} \\
    i_{\gamma}
\end{bmatrix} =
\begin{bmatrix}
    1 & -0.5 & -0.5 \\
    0 & \sqrt{3}/2 & -\sqrt{3}/2
\end{bmatrix}
\begin{bmatrix}
    i_a \\
    i_b \\
    i_c
\end{bmatrix}
\text{and} \quad i_c = -(i_a + i_b)
\]

(5.2)

and then extracting the positive and negative sequence components using

\[
\begin{align*}
[v_{\alpha\beta}]^p &= \frac{1}{2} \begin{bmatrix} 1 & -i \\ i & 1 \end{bmatrix} [v_{\alpha\beta}] \\
[v_{\alpha\beta}]^n &= \frac{1}{2} \begin{bmatrix} 1 & i \\ -i & 1 \end{bmatrix} [v_{\alpha\beta}]
\end{align*}
\]

(5.3)
where “\( i \)” denotes a 90° phase shift operator in the time domain. While various methods have been proposed to generate this phase shifted signal, the standard approaches are to use a time delay operator, which has dynamic response limitations, or a PLL based on a second order generalized integrator [79-81].

It is well known that accurate current regulation into an unbalanced grid voltage requires control of both the positive and the negative sequence currents. Typically, two synchronous frame current regulators (DSRF) are required to achieve this objective, one rotating in the positive sequence direction and the other rotating in the negative sequence direction, as shown in Figure 5.2.

The DSRF regulator system works by initially transforming the measured phase currents to the \( \alpha \beta \) stationary reference frame. The measured current sequence components are then extracted using (5.4) which requires implementation of the 90° (fundamental 50 Hz) time domain shift. Various strategies have been proposed to implement this time domain shift, such as a notch filter to explicitly extract each sequence component element [83], or a Dual Second Order Generalized Integrator-based Quadrature-Signal Generator (DSOGI_QSG) [80], as shown in Figure 5.2 and Figure 5.3. The DSOGI_QSG transfer functions are given by

\[
\frac{i_x'}{i_x}(s) = \frac{k\omega's}{s^2 + k\omega's + \omega'^2} \text{ for } x = \{\alpha, \beta\}
\]

\[
\frac{qi_x'}{i_x}(s) = \frac{k\omega'^2}{s^2 + k\omega's + \omega'^2} \text{ for } x = \{\alpha, \beta\}
\]

where the resonant frequency is set by \( \omega' \) and the damping factor is set by \( k \). Note that \( qi' \) always lags \( i' \) by 90° regardless of the \( k \) and \( \omega' \) values.

The extracted \( \alpha \beta \) stationary reference frame sequence components are transformed into their respective \( dq \) SRF and compared against their target sequence references to create the current regulator input error signals. The PI controllers then calculate their commanded voltage outputs, which are transformed back into the stationary frame, summed, and passed to the PWM modulator to create switching signals. This control system introduces a number of undesirable properties including

\[
[i_{\alpha\beta}^p] = \frac{1}{2} \begin{bmatrix} 1 & -i \end{bmatrix} [i_{\alpha\beta}]
\]

\[
[i_{\alpha\beta}^n] = \frac{1}{2} \begin{bmatrix} 1 & i \end{bmatrix} [i_{\alpha\beta}]
\]

(5.4)
that it firstly requires decoupling of the sequence components, which affects the maximum possible current regulator gains that can be achieved and this consequently compromises the transient response of the system.

Reference [84] proposes an alternative decoupling method to separate the positive and negative sequences, by transforming the measured currents to the synchronous reference frames before separation. Cross coupling terms are then added to the signals to eliminate oscillations caused by the negative [positive] sequence on the PSRF [NSRF] using

\[
i_d^p = i_d^p + e^{-j(\theta^p-\theta^n)} \cdot i_d^q - e^{-j(\theta^p-\theta^n)} \cdot \left( i_{dq,ref} - \Delta i_d^q \right) \tag{5.7}
\]
\[
\begin{align*}
\dot{i}_{dq}^n &= i_{dq}^n + e^{-j(\theta^n - \theta^p)} \cdot i_{dq}^p - e^{-j(\theta^n - \theta^p)} \cdot (i_{dq,ref}^p - \Delta i_{dq}^p) \\
\end{align*}
\] (5.8)

where J matrix is \( J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \) and \( e^{-j\theta} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \).

Note that a low pass filter is still required for (5.7) and (5.8) to obtain the current error mean value (\( \Delta i_{dq}^n \) and \( \Delta i_{dq}^p \)), which compromises the transient response of the controller. Furthermore, this approach still introduces substantial complexity into the computational process.

The essential difficulty with the system of Figure 5.2 is that the commanded reference sequence currents are fully decoupled DC quantities as shown in Figure 5.4. Hence the measured currents must also be fully separated into their sequence components before they can be used as regulator feedback variables. The delays and complexity of the sequence extraction processes of (5.5)-(5.8) inevitably cause incomplete separation (incomplete “cross coupling” between the \( dq \) variables), particularly during transient changes in the commanded current, as shown in Figure 5.5.

The identification of delays relates more to the measurement and separation processes that must be used for the calculation, these delays are typically inherent to the sequence of computation required, and may not be resolved by just executing each computational stage faster.

Consequently, each sequence current regulator’s performance is compromised during such transitions, as it attempts to regulate both its own measured sequence current plus a disturbance feedback injection because of incomplete sequence separation from the primary measured input currents.
Figure 5.4. Sequence current references in their corresponding frame of references using (4). (upper trace) Positive sequence ($i_{d_p}^{PSRF}$ and $i_{q_p}^{PSRF}$). (lower trace) Negative sequence ($i_{d_n}^{NSRF}$ and $i_{q_n}^{NSRF}$).

Figure 5.5. Injected currents for PSRF and DSRF with sequence extraction. (upper trace) Positive sequence ($i_{d_p}^{PSRF}$ and $i_{q_p}^{PSRF}$). (lower trace) Negative sequence ($i_{d_n}^{NSRF}$ and $i_{q_n}^{NSRF}$).
5.2.2. Proposed New Strategy for Unbalanced Current Regulation

For the approach presented in this thesis, rather than extracting the sequence components from the measured currents, the current references are reshaped to include both their own DC sequence component, and an AC component caused by the projection of the PSRF [NSRF] onto the NSRF [PSRF]. Figure 5.6 shows this new strategy, where combined SRF target sequence current references have been created using direct DC components, and a double frequency component from the opposite SRF frame. Figure 5.7 presents a phasor diagram of the sequence components in stationary frame.

Figure 5.6. Proposed unbalanced DG current control using DSRF without Sequence Current Extraction.

Figure 5.7. Phasor diagram of the sequence components.
Each controller now works to regulate both positive and negative sequence currents simultaneously, and hence does not require these sequence currents to be separately extracted from the measured feedback signal.

The main differences between the conventional DSRF and the proposed method are where the positive and negative sequence transformations are processed. The conventional DSRF applies the transformations on the actual currents, while the proposed method performs the transformations on the current commands. Theoretically, these methods should accomplish the same outcome at steady state. Differences mostly occur in the implementation, like the amount of required computation, design of low-pass or notch filter in the signal processing stage, the resolution of fixed-point math processing and so on.

**DSRF without Sequence Extraction**

The combined current references are created by transforming the commanded DC sequence references, from the positive SRF to the negative SRF as defined in (5.9) and the negative SRF to the positive SRF as defined in (5.10).

\[
I_{dq,p,ref}^{NSRF} = [T_{PSRF2aβ}]^{TNSRF}_{aβ}[I_{dq,p,ref}^{PSRF}]
\]

\[
I_{dq,n,ref}^{NSRF} = [T_{NSRF2aβ}]^{TNSRF}_{aβ}[I_{dq,n,ref}^{NSRF}]
\]

\[
T_{aβ2PSRF} \begin{bmatrix} q \\ d \end{bmatrix} = \begin{bmatrix} \cos(ωt) & -\sin(ωt) \\ \sin(ωt) & \cos(ωt) \end{bmatrix} \begin{bmatrix} β \\ α \end{bmatrix}
\]

\[
T_{aβ2NSRF} \begin{bmatrix} q \\ d \end{bmatrix} = \begin{bmatrix} \cos(-ωt) & -\sin(-ωt) \\ \sin(-ωt) & \cos(-ωt) \end{bmatrix} \begin{bmatrix} β \\ α \end{bmatrix}
\]

\[
T_{PSRF2aβ} \begin{bmatrix} β \\ α \end{bmatrix} = \begin{bmatrix} \cos(ωt) & \sin(ωt) \\ -\sin(ωt) & \cos(ωt) \end{bmatrix} \begin{bmatrix} q \\ d \end{bmatrix}
\]

\[
T_{NSRF2aβ} \begin{bmatrix} β \\ α \end{bmatrix} = \begin{bmatrix} \cos(-ωt) & \sin(-ωt) \\ -\sin(-ωt) & \cos(-ωt) \end{bmatrix} \begin{bmatrix} q \\ d \end{bmatrix}
\]

where \(T_{PSRF2aβ}/T_{NSRF2aβ}\) is positive/negative SRF frame (dq) to \(αβ\) frame transformation and \(T_{aβ2PSRF}/T_{aβ2NSRF}\) is \(αβ\) frame to positive/negative SRF (dq) transformation.

The results are then summed as shown in (5.15)-(5.16). Note that each resulting reference retains its previous DC component, and also includes its opposite commanded sequence component as a double fundamental frequency AC term.
The PSRF and NSRF rotate counter clockwise and clockwise respectively as shown in Figure 5.6. The difference between the angular frequencies of these SRFs makes the projection of their DC components appear as AC terms of a double frequency on the opposite SRF, i.e.

\[
\theta^{\text{PSRF}} - \theta^{\text{NSRF}} = \omega t + \phi_p^{\text{PSRF}} - (-\omega t + \phi_n^{\text{NSRF}}) = 2\omega t
\]  

(5.17)

under the condition where \(\phi_p^{\text{PSRF}} = \phi_n^{\text{NSRF}} = 0\) and \(\theta^{\text{PSRF}} [\theta^{\text{NSRF}}]\) is the instantaneous angular phase angle of the positive [negative] sequence frame. Eqns. (5.9) and (5.10) can then be applied to (5.15) and (5.16) to make the DC components appear as oscillating variables on the opposite sequence frame, viz:

\[
\begin{align*}
\dot{i}_{d,ref}^{\text{PSRF}} &= i_{d,ref}^{\text{PSRF}} + \cos(2\omega t) \cdot i_{n,ref}^{\text{NSRF}} + \sin(2\omega t) \cdot i_{q,ref}^{\text{NSRF}} \\
\dot{i}_{q,ref}^{\text{PSRF}} &= \sin(2\omega t) \cdot i_{d,ref}^{\text{NSRF}} + \cos(2\omega t) \cdot i_{q,ref}^{\text{NSRF}} \\
\dot{i}_{d,ref}^{\text{NSRF}} &= \cos(2\omega t) \cdot i_{d,ref}^{\text{PSRF}} - \sin(2\omega t) \cdot i_{q,ref}^{\text{PSRF}} \\
\dot{i}_{q,ref}^{\text{NSRF}} &= \sin(2\omega t) \cdot i_{d,ref}^{\text{PSRF}} + \cos(2\omega t) \cdot i_{q,ref}^{\text{PSRF}} 
\end{align*}
\]  

(5.18)

(5.19)

Figure 5.8 shows the resulting current reference components projected into their opposite rotating frame. As expected, the amplitude of each oscillating component is the same as the DC value of the sequence current in the original reference frame.

The total current reference for each reference frame then becomes its DC component in that frame (Figure 5.4) summed with the projection of the DC component
of the opposite frame into the destination frame (Figure 5.8). Figure 5.9 shows this resultant, which is the total reference current that the current regulator now has to track.

Figure 5.8. Projection of sequence current references on the opposite frame of references. (upper trace) Projection of positive sequence current on the NSRF ($i_{d,p,ref}^{NSRF}$ and $i_{q,p,ref}^{NSRF}$). (lower trace) Projection of negative sequence current on the PSRF ($i_{d,n,ref}^{PSRF}$ and $i_{q,n,ref}^{PSRF}$).

Figure 5.9. Sequence current references (dotted lines) and the injected currents for PSRF and DSRF without sequence extraction. (upper trace) Positive sequence ($i_{d,PSRF}^{PSRF}$ and $i_{q,PSRF}^{PSRF}$). (lower trace) Negative sequence ($i_{d,NSRF}^{NSRF}$ and $i_{q,NSRF}^{NSRF}$).
**Closed-Loop Current Control with DSRF**

Once combined in this way the sequence current components can then be controlled without separation using conventional PI controllers. This means that the PI controllers in the positive and negative SRFs deal with both DC and AC terms by working in parallel (it is known that PI controllers are not sufficient for AC systems as they have steady state error). From Appendix A it is known that PI controllers in the positive and negative SRFs are the analytical equivalent of a P+Resonant controller in the stationary frame (which is quite able to regulate positive and negative sequence currents since they appear as AC terms with a fundamental frequency in the stationary frame). Hence transforming both the PSRF and NSRF PI controllers back into the stationary frame and combining them gives

$$
\begin{bmatrix}
G_{(dq)P1} \\
G_{(dq)N1}
\end{bmatrix}_{\alpha\beta} = \begin{bmatrix}
G_{(dq)P1}^P \\
G_{(dq)N1}^N
\end{bmatrix}_{\alpha\beta}
$$

$$
= \begin{bmatrix}
2(K_p + \frac{K_i s}{s^2 + \omega_0^2}) & 0 \\
0 & 2(K_p + \frac{K_i s}{s^2 + \omega_0^2})
\end{bmatrix}
$$

(5.20)

which is immediately recognizable as a conventional stationary frame PR controller but with double magnitude diagonal terms. Note also from (5.20) that adding these two regulators together has resulted in the cancellation of the cross coupling terms since they have opposite signs in the stationary frame. Hence the two resultant stationary frame $\alpha\beta$ controllers are completely independent of each other and so the approach presented in [65]-[66] can be used here to again determine the maximum possible gains for the two DSRF controllers.

**Maximum Current Regulator Gains**

Now, based on the equivalent current controller developed in the previous section and taking into account that two SRF controllers are involved, the maximum current regulator gains can be calculated using the same approach as Appendix A, the maximum possible proportional gain becomes

$$
K_p \approx \frac{\omega_c(L_f + L_{fg})}{2V_{dc}}
$$

(5.21)

while the integral reset time can be set to:
The maximum and minimum possible damping gains can also be calculated using the method presented in [66] for an LCL resonant frequency below critical frequency, which gives damping limits of $K_{d,\text{min}} = 0.036$ and $K_{d,\text{max}} = 0.109$. The damping gain used in this thesis was set to approximately midway between these two values. Note that this damping gain also has a factor of 2 compared to [66] – the same as for the proportional gain.

### 5.2.3. Performance Evaluation of Proposed DSRF Current Regulator

For this part of the work, the effect of the proposed strategy has been explored with experimental investigations, using a 5 kVA three-phase inverter that is connected to the grid via an LCL filter and an isolation transformer. The parameters of the system and the controller gains are listed in Table 5.1.

A fixed point DSP (TMS320F2810) was used as the system controller for all measurement, control and PWM functions. DC bus voltage compensation was included into the PWM processes to ensure that variations in the DC bus voltages caused by real power ripple did not affect the quality of the regulated AC currents.

The three-phase unbalanced grid voltages were created by a programmable

<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Description</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>DG rating</td>
<td>5 kVA</td>
</tr>
<tr>
<td>$L_f$</td>
<td>Inverter-side filter inductance</td>
<td>3 mH</td>
</tr>
<tr>
<td>$L_{fg}$</td>
<td>Grid-side filter inductance</td>
<td>1 mH</td>
</tr>
<tr>
<td>$C_f$</td>
<td>Filter Capacitance</td>
<td>7.5 µF</td>
</tr>
<tr>
<td>$2V_{DC}$</td>
<td>DC bus voltage</td>
<td>400 V</td>
</tr>
<tr>
<td>$f_s$</td>
<td>Switching frequency</td>
<td>10 kHz</td>
</tr>
<tr>
<td>$f_{samp}$</td>
<td>Sampling frequency</td>
<td>20 kHz</td>
</tr>
<tr>
<td>$f$</td>
<td>Grid frequency</td>
<td>50 Hz</td>
</tr>
<tr>
<td>$V_g$</td>
<td>Nominal grid phase voltage</td>
<td>110 Vrms</td>
</tr>
<tr>
<td>$K_p$</td>
<td>Proportional gain</td>
<td>0.048 $A^{-1}$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>Integral reset time</td>
<td>0.0021 s</td>
</tr>
<tr>
<td>$K_d$</td>
<td>Damping gain</td>
<td>0.099</td>
</tr>
<tr>
<td>$R_{grid}$</td>
<td>Line resistance</td>
<td>1.5/0.3142 Ω</td>
</tr>
<tr>
<td>$L_{grid}$</td>
<td>Line reactance</td>
<td>1 mH (0.314 Ω)</td>
</tr>
</tbody>
</table>
Chapter 5  DG Unit Current Control for Unbalanced Voltage

California Instrument MX30/45 grid simulator. The grid-side impedances ($Z_{grid}$) were made up from a series combination of discrete inductors, resistors.

The performance of the proposed current regulator incorporating SDRF without current sequence extraction has been evaluated using detailed time domain simulations and then verified with experimental investigations.

Figure 5.10 and Figure 5.11 show matching simulation and experimental time-domain system responses of the system injecting arbitrary sequence current commands into a stiff unbalanced voltage grid. The sequence current references both before and after a step change at 0.05 s are listed in Table 5.2.

Figure 5.10 presents the results for a grid with a single phase voltage sag of 80%. From this figure, it can be seen that the proposed current regulator can readily follow the sequence current references without the need for sequence extraction. The dynamic performance of the current regulator is also very good, with a rise time of ~ 1 msec.

Figure 5.11 shows the performance of the controller with a two-phase voltage sag of 80%. Once again, the sequence current control performance matches simulation regardless of the changed grid voltage conditions, injecting the same sequence currents with a similar level of transient response.

Finally, Figure 5.12 and Figure 5.13 show experimental results for an unbalanced set of grid voltages with a weak grid (a large grid impedance) connection through a higher value resistor. A one-phase voltage sag condition is shown in Figure 5.12, while Figure 5.13 illustrates a two-phase voltage sag condition (indicative of an asymmetrical grid fault away from the feeder). While balanced three-phase currents (i.e. no negative sequence current) are injected to the grid, the PCC voltages remain unbalanced as expected. However, at 0.25 sec, a compensating negative sequence current is injected, and the PCC voltages essentially balance. Hence it is clear that unbalanced PCC voltages can be corrected by injecting appropriate compensating unbalanced currents. In general, mitigation of such an unbalanced PCC voltage condition depends on the grid

<table>
<thead>
<tr>
<th>Time</th>
<th>$i_{d_p_ref}^{PSRF}$</th>
<th>$i_{d_p_ref}^{PSRF}$</th>
<th>$i_{d_n_ref}^{PSRF}$</th>
<th>$i_{q_n_ref}^{PSRF}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 &lt; t &lt; 0.05s$</td>
<td>$2A_{rms}$</td>
<td>$0A_{rms}$</td>
<td>$0A_{rms}$</td>
<td>$0A_{rms}$</td>
</tr>
<tr>
<td>$t &gt; 0.05s$</td>
<td>$4A_{rms}$</td>
<td>$0A_{rms}$</td>
<td>$-0.1A_{rms}$</td>
<td>$-0.3A_{rms}$</td>
</tr>
</tbody>
</table>
connection impedance and the strategies used to regulate real power injection, minimizing reactive power oscillation or injecting balanced three phase currents into the grid [73, 77], as appropriate.

Figure 5.10. Step-up transient response. (left) simulation. (right) experiment. (top) one-phase grid voltage sag condition. (bottom) corresponding grid-side current $i_g$.

Figure 5.11. Step-up transient response. (left) simulation. (right) experiment. (top) two-phase grid voltage sag condition. (bottom) corresponding grid-side current $i_g$. 
Chapter 5  DG Unit Current Control for Unbalanced Voltage

Figure 5.12. Experimental results: Three-phase current (top) & PCC voltages (bottom).

Figure 5.13. Experimental results: Three-phase current (top) & PCC voltages (bottom).
5.3. Consolidated Power Control for a Three-Phase DG System

The primary objective of the top level power control system is to regulate the average real and reactive power that is injected into the grid system. The secondary objective under unbalanced voltage conditions is to either 1) eliminate the active power ripple, or 2) eliminate the reactive power ripple, or 3) simply balance the grid injected currents, as required.

The power control loop aims to generate commanded values for the current regulators that form the next level down control system. For this purpose, these commanded values are calculated based on the target real and reactive power commands \( P_{\text{ref}} \) and \( Q_{\text{ref}} \) which are either commanded from a remote controller, or set to constant values for a desired level of PQ control.

The power injected into the grid is measured using

\[
\begin{align*}
    P &= v_d i_d + v_q i_q \\
    Q &= v_q i_d - v_d i_q
\end{align*}
\]  

(5.23)

where \( v_{dq} \) and \( i_{dq} \) are the measured grid voltages and currents in the \( dq \) frame. For unbalanced PCC voltage conditions, the real and reactive power flows can be calculated from the positive and negative sequence voltage and current components [85] using

\[
\begin{bmatrix}
P \\
Q \\
P_c \\
Q_c
\end{bmatrix}
= 
\begin{bmatrix}
+v_d^p & +v_q^p & +v_d^n & +v_q^n \\
+v_q^p & -v_d^p & +v_d^n & -v_q^n \\
+v_d^p & +v_q^p & +v_d^n & +v_q^n \\
+v_q^p & -v_d^p & -v_d^n & +v_q^n \\
+v_d^p & +v_q^p & +v_d^n & +v_q^n \\
+v_q^p & -v_d^p & -v_d^n & +v_q^n \\
-v_d^p & -v_q^p & +v_d^n & +v_q^n \\
-v_q^p & +v_d^p & +v_d^n & +v_q^n
\end{bmatrix}
\times
\begin{bmatrix}
i_d^p \\
i_q^p \\
i_d^n \\
i_q^n
\end{bmatrix}
\]  

(5.24)

where the \( P \) and \( Q \) terms in (5.24) are the average real and reactive powers injected into the grid, and the \( P_c, P_n, Q_c, Q_n \) terms define the magnitude of the double fundamental frequency oscillating real and reactive powers as quadrature components referenced to the synchronous rotating frame. Eqn. (5.24) can then be used to create commanded references for the current regulators from the required grid power injections, viz:
By deleting unnecessary rows, inverting the remaining reduced matrix and replacing the calculated powers with known reference values, commanded currents for the three different ripple control strategies can be determined as described in the following sections.

5.3.1. Active Power Ripple Control

For control of active power ripple only, the last two rows of (5.25) can be deleted, since the reactive power ripple is uncontrolled. The reduced matrix is then inverted, the average P and Q injections are set to known values of $P_{\text{ref}}$ and $Q_{\text{ref}}$, and the target quadrature active power ripple magnitudes are set to zero, i.e. $P_c = P_s = 0$, to give:

\[
\begin{bmatrix}
    i_{d}^p \\
    i_{q}^p \\
    i_{d}^n \\
    i_{q}^n
\end{bmatrix}_{\text{ref}} = \begin{bmatrix}
    +v_{d}^p & +v_{q}^p & +v_{d}^n & +v_{q}^n \\
    +v_{q}^p & -v_{d}^p & +v_{q}^n & -v_{d}^n \\
    +v_{d}^n & +v_{q}^n & +v_{d}^p & +v_{q}^p \\
    +v_{q}^n & -v_{d}^n & +v_{q}^p & -v_{d}^p \\
    -v_{d}^n & -v_{q}^n & +v_{d}^p & +v_{q}^p
\end{bmatrix}^{-1} \times \begin{bmatrix}
    P_{\text{ref}} \\
    Q_{\text{ref}} \\
    0 \\
    0
\end{bmatrix}
\]  
\( (5.26) \)

\[
\begin{bmatrix}
    i_{d}^p \\
    i_{q}^p \\
    i_{d}^n \\
    i_{q}^n
\end{bmatrix}_{\text{ref}} = \begin{bmatrix}
    +v_{d}^p / D & +v_{q}^p / E \\
    +v_{q}^p / D & -v_{d}^p / E \\
    -v_{d}^n / D & +v_{q}^n / E \\
    -v_{q}^n / D & -v_{d}^n / E
\end{bmatrix} \times \begin{bmatrix}
    P_{\text{ref}} \\
    Q_{\text{ref}}
\end{bmatrix}
\]  
\( (5.27) \)

\[
\begin{align*}
D &= (v_{d}^p)^2 + (v_{q}^p)^2 - (v_{d}^n)^2 - (v_{q}^n)^2 \\
E &= (v_{d}^p)^2 + (v_{q}^p)^2 + (v_{d}^n)^2 + (v_{q}^n)^2
\end{align*}
\]  
\( (5.28) \)

With the reference currents set by commanded average power injections and zero real power oscillations only, reactive power oscillations will now occur, since their magnitude is uncontrolled. The quadrature component magnitudes of these oscillations can be obtained by combining (5.27) with the last two lines of (5.24), to give
5.3.2. Reactive Power Ripple Control

For control of reactive power ripple only, the middle two rows of (5.25) can be deleted, since active power ripple is uncontrolled. The reduced matrix is then inverted, the average \( P \) and \( Q \) injections are set to known values of \( P_{\text{ref}} \) and \( Q_{\text{ref}} \), and the quadrature reactive power ripple magnitudes are set to zero, i.e. \( Q_c = Q_s = 0 \), to give:

\[
\begin{bmatrix}
\frac{v_d^n v_q^n - v_d^n v_d^n}{D} & \frac{v_d^n v_q^n + v_q^n v_d^n}{E} \\
\frac{v_d^n v_d^n + v_q^n v_q^n}{D} & \frac{v_d^n v_q^n - v_d^n v_q^n}{E}
\end{bmatrix} \times \begin{bmatrix}
P \\ Q_{\text{ref}}
\end{bmatrix}
\]

\[(5.29)\]

\[
\begin{bmatrix}
\frac{v_d^n v_q^n - v_d^n v_d^n}{D} & \frac{v_d^n v_q^n + v_q^n v_d^n}{E} \\
\frac{v_d^n v_d^n + v_q^n v_q^n}{D} & \frac{v_d^n v_q^n - v_d^n v_q^n}{E}
\end{bmatrix}^{-1}
\times \begin{bmatrix}
P \\ Q_{\text{ref}}
\end{bmatrix}
\]

\[(5.30)\]

\[
\begin{bmatrix}
v_d^p \\
v_q^p \\
v_d^n \\
v_q^n
\end{bmatrix}_{\text{ref}} = \begin{bmatrix}
+v_d^p & +v_q^p & +v_d^n & +v_q^n \\
+v_q^p & -v_d^p & +v_d^n & -v_q^n \\
+v_q^n & -v_d^n & +v_q^p & -v_d^n \\
-v_d^n & -v_q^n & +v_d^p & +v_q^n
\end{bmatrix}^{-1} \times \begin{bmatrix}
P_{\text{ref}} \\ Q_{\text{ref}}
\end{bmatrix}
\]

\[(5.31)\]

Similarly to (5.29), the quadrature magnitudes of the oscillating real power can be obtained from (5.31) and (5.24) as

\[
\begin{bmatrix}
P \\ P_s
\end{bmatrix} = 2 \begin{bmatrix}
\frac{v_d^n v_q^n + v_q^n v_d^n}{E} & \frac{v_d^n v_q^n - v_q^n v_d^n}{D} \\
\frac{v_d^n v_d^n + v_q^n v_q^n}{E} & \frac{v_d^n v_q^n + v_q^n v_d^n}{D}
\end{bmatrix} \times \begin{bmatrix}
P \\ Q_{\text{ref}}
\end{bmatrix}
\]

\[(5.32)\]

5.3.3. Balanced Current Control

For balanced grid currents, only the top left hand four elements of (5.25) are required, since both the active and reactive power ripple are uncontrolled, and the negative sequence current references are forced to zero, so that:

\[
\begin{bmatrix}
v_d^p \\
v_q^p \\
v_d^n \\
v_q^n
\end{bmatrix}_{\text{ref}} = \begin{bmatrix}
+v_d^p & +v_q^p \\
+v_q^p & -v_d^p
\end{bmatrix}^{-1} \times \begin{bmatrix}
P_{\text{ref}} \\ Q_{\text{ref}}
\end{bmatrix}, \quad i_d^n_{\text{ref}} = i_q^n_{\text{ref}} = 0
\]

\[(5.33)\]
Chapter 5  DG Unit Current Control for Unbalanced Voltage

\[
\begin{bmatrix}
  i_d^p \\
  i_q^p \\
\end{bmatrix}_{\text{ref}} = \begin{bmatrix}
  \frac{+v_d^p}{(v_d^p)^2 + (v_q^p)^2} \\
  \frac{+v_q^p}{(v_d^p)^2 + (v_q^p)^2} \\
  \frac{+v_q^p}{(v_d^p)^2 + (v_q^p)^2} \\
  \frac{-v_d^p}{(v_d^p)^2 + (v_q^p)^2} \\
\end{bmatrix} \times \begin{bmatrix}
  P_{\text{ref}} \\
  Q_{\text{ref}} \\
\end{bmatrix}
\]

(5.34)

Balanced currents are obtained at the expense of both active and reactive power oscillations. The oscillating power magnitudes can be calculated by substituting (5.34) into the last four lines of (5.24), to give

\[
\begin{bmatrix}
  P_c \\
  P_s \\
  Q_c \\
  Q_s \\
\end{bmatrix} \begin{bmatrix}
  (v_d^n v_d^p + v_q^n v_q^p) \\
  (v_d^n v_d^p - v_q^n v_q^p) \\
  (v_d^n v_d^p + v_q^n v_q^p) \\
  (v_d^n v_d^p - v_q^n v_q^p) \\
\end{bmatrix} \begin{bmatrix}
  (v_d^p v_d^p - v_q^p v_d^p) \\
  (v_d^p v_d^p + v_q^p v_d^p) \\
  (v_d^p v_d^p + v_q^p v_d^p) \\
  (v_d^p v_d^p - v_q^p v_d^p) \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  P_{\text{ref}} \\
  Q_{\text{ref}} \\
\end{bmatrix}
\]

(5.35)

Comparing the oscillating power magnitudes of (5.29), (5.32) and (5.35), it can be seen that the factor of “2” has disappeared when balanced currents are generated. This is because the power oscillations are shared between both the real and reactive powers for this third case.

5.3.4. Combined Power Ripple Control

Equations (5.27), (5.31) and (5.34) can be combined as follows:

\[
\begin{bmatrix}
  i_d^p \\
  i_q^p \\
  i_d^n \\
  i_q^n \\
\end{bmatrix}_{\text{ref}} = \begin{bmatrix}
  +v_d^p/D' & +v_q^p/E' \\
  +v_q^p/D' & -v_d^p/E' \\
  -Kv_d^n/D' & +Kv_q^n/E' \\
  -Kv_q^n/D' & -Kv_d^n/E' \\
\end{bmatrix} \times \begin{bmatrix}
  P_{\text{ref}} \\
  Q_{\text{ref}} \\
\end{bmatrix}
\]

(5.36)

where

\[
D' = (v_d^n)^2 + (v_q^n)^2 - K((v_d^n)^2 + (v_q^n)^2)
\]
\[
E' = (v_d^n)^2 + (v_q^n)^2 + K((v_d^n)^2 + (v_q^n)^2)
\]

(5.37)

and \(K\) is a real number in the range of \([-1,1]\) that defines the required power ripple control strategy. When \(K = +1\), (5.36) and (5.37) revert to (5.27) and (5.28) to control active power ripple only at the expense of unbalanced grid currents and oscillatory reactive power. Similarly, when \(K = -1\), (5.36) and (5.37) revert to (5.30) and (5.31) to
control reactive power ripple only at the expense of unbalanced grid currents and oscillatory active power. Finally, when $K = 0$, (5.36) and (5.37) revert to (5.33) and (5.34), and the injected grid currents only are balanced.

The consolidated oscillating power control concept has been verified initially using a detailed PSIM simulation representation of Figure 5.14, feeding into an unbalanced grid supply where one phase voltage sags to 50%. The filter and current regulator gain parameters of the simulated system have been matched to those of the experimental system presented in section 6.1.

In this case study, a fixed amount of average real power injection is considered without reactive power injection. To achieve desired ripple power control, the PCC voltage is measured and the sequence components are decoupled to generate the current references based on equations (5.36) and (5.37). It is now possible to see how these power ripples shape the three-phase currents injected to the grid.

For the 50% sag unbalanced grid voltage condition shown in Figure 5.15, the controller performance is shown in Figure 5.16. For the period $0.0 \leq t < 0.1$ s, the “Active Power Control” strategy ($K = +1$) is implemented and no oscillation is present in the real power. As expected, oscillations appear in the reactive power, and the three-phase currents are also not balanced. For the period $0.1 \leq t < 0.2$ s, the “Current Control” strategy is used with $K = 0$. Balanced currents are then generated to feed into

---

![Figure 5.14. Proposed DG converter system with double synchronous frame controllers and actively damped LCL filter system](image)

---
the grid, but with both real and reactive power oscillation as a consequence. Finally, for
the period $0.2 \leq t < 0.3\, \text{s}$, $K$ is changed to $K = -1$ to enable the “Reactive Power
Control” strategy. In contrast to active power control, no reactive power oscillations are
now present, but there is now real power oscillation, and once again the grid currents
are unbalanced.
It is useful to mention that real power regulation increases the peak current in the sagged phase voltages, while reactive power regulations results in injection of higher currents in the normal phases.

5.4. Summary

In this chapter in order to investigate sequence current impact on unbalanced voltages, a sequence current regulation strategy for DG systems in the SRF when the grid voltages are unbalanced was presented. It was shown that in order to regulate the positive and negative current sequences, these components should be extracted and then decoupled to allow PI controllers to be used to regulate the DC values of decoupled sequence components. It was then shown how positive and negative sequence three phase currents can be injected into an unbalanced three phase grid without requiring sequence current separation of the measured feedback current. The proposed controller architecture considerably reduces the complexity of the current controller, and, when maximum possible PI controller gains are used, significantly improves the dynamic response of the current regulation strategy.

The chapter then presented a strategy to vary between eliminating the real power ripple, the reactive power ripple, or to just balance the grid currents, for a DG inverter operating into a grid network with unbalanced voltages at the PCC. The strategy calculates positive and negative sequence current references from the required average power injection and power ripple objectives, and feeds these references to synchronous frame closed loop current regulators. The system achieves the required power ripple elimination objective under all load conditions. Detailed experimental results are presented to confirm the viability of the strategy, and to explore its influence on sagging phase voltages for a variety of injection conditions.
Chapter 6

**DISTRIBUTION SYSTEM MODELLING FOR UNBALANCED VOLTAGE MITIGATION**

In Chapter 5 a DSRF current regulation without sequence separation and a combined oscillating power control strategy was proposed for a DG unit to investigate the potential of DG control strategies on voltage support, in particular when the three phase voltages are unbalanced.

In this chapter\(^1\) the effect of combined oscillating power control strategy on unbalanced PCC voltages will be investigated under different levels of P and Q injection. In particular, it allows the voltage response of a feeder with either an inductive or a resistive line characteristic to be explored. The analysis identifies three major factors of significance which affect the unbalanced voltage mitigation, viz:

- Grid characteristics (resistive and/or inductive),
- Voltage sag type,

---

\(^1\) Many of the contributions contained in this chapter were first published by the author in:


• Oscillating power control strategy,

The results provide a new understanding as to the influence of real/reactive power ripple mitigation on unbalanced network voltages for grid connected DG systems.

Furthermore, in order to take into account the second order effects of inverter systems, simulation models of both the grid network and the DG inverter systems that are sufficiently detailed to realistically represent their real world physical system behaviours are being used. Inverter systems are usually simulated in detail using specialist packages, which are not particularly suited to modelling larger scale power systems. Similarly, power system simulation packages typically represent inverter systems using simpler averaged models, which do not adequately reflect the inverter’s real dynamic response to transient events. This chapter addresses this issue, by presenting a detailed power inverter model developed within the DIgSILENT power network simulation package (presented in Section 7.2), which is sufficiently detailed to represent the inverter operation up to the PWM switching frequency, but is still computationally viable for use with larger scale system studies.

Finally, to investigate unbalanced voltage mitigation, the developed detailed model of the DG switched mode power converter is used to accurately represent the dynamic AC response of grid connected inverter interfaced DGs. Voltage support under unbalanced voltage conditions is investigated using the proposed power control strategy in Chapter 5. The results show that such detailed converter representations can accurately characterise the impact of DGs on an electrical grid, and in particular can identify their dynamic behaviour under unbalanced voltage conditions. The results also show that the proposed new inverter control strategies can mitigate voltage unbalance and improve the feeder voltage profile.

6.1. Power Flow Ripple Control and Its Effect on PCC Grid Voltage

The combined power ripple control strategy is now used to investigate the effect on voltage of eliminating the active or the reactive power ripple, or just balancing the grid injected currents, when the DG feeds into a grid distribution system with a significant voltage unbalance at the PCC. For this investigation, the DG system is connected to the main grid through a feeder with different line characteristics ($Z_{grid}$) as shown in Figure 6.1 (the system parameters are listed in Table 6.1). The effect of the
three different power ripple control strategies on the voltage quality at the point of common coupling is then explored as the grid feeder impedance varies from resistive to reactive.

For this study, the voltage unbalanced $n$-factor [91], which is the ratio between negative and positive sequence voltage amplitudes, is used as the performance parameter to quantify the system voltage unbalance at the PCC. The $n$-factor is defined by

$$n - factor = \frac{V^n}{V^p} = \frac{(v^n_d)^2 + (v^n_q)^2}{(v^p_d)^2 + (v^p_q)^2}$$  \hspace{1cm} (6.1)$$

and is calculated from measurements at the PCC. The influence of the various oscillating power control strategies is then evaluated by comparing the $n$-factor of the unbalanced grid supply voltages against the $n$-factor of the DG connected system at its

![Figure 6.1. Three-phase grid-connected DG inverter.]

### Table 6.1. System Parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>DG rating</td>
<td>5 kVA</td>
</tr>
<tr>
<td>$L_f$</td>
<td>Inverter-side filter inductance</td>
<td>3 mH</td>
</tr>
<tr>
<td>$L_{f_g}$</td>
<td>Grid-side filter inductance</td>
<td>1 mH</td>
</tr>
<tr>
<td>$C_f$</td>
<td>Filter Capacitance</td>
<td>7.5 $\mu$F</td>
</tr>
<tr>
<td>$2V_{DC}$</td>
<td>DC bus voltage</td>
<td>400 V</td>
</tr>
<tr>
<td>$f_s$</td>
<td>Switching frequency</td>
<td>10 kHz</td>
</tr>
<tr>
<td>$f_{samp}$</td>
<td>Sampling frequency</td>
<td>20 kHz</td>
</tr>
<tr>
<td>$f$</td>
<td>Grid frequency</td>
<td>50 Hz</td>
</tr>
<tr>
<td>$V_g$</td>
<td>Nominal grid phase voltage</td>
<td>110 Vrms</td>
</tr>
<tr>
<td>$K_p$</td>
<td>Proportional gain</td>
<td>0.048 $A^{-1}$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>Integral reset time</td>
<td>0.0021 s</td>
</tr>
<tr>
<td>$K_d$</td>
<td>Damping gain</td>
<td>0.099</td>
</tr>
<tr>
<td>$R_{grid}$</td>
<td>Line resistance</td>
<td>1.5/0.3142 $\Omega$</td>
</tr>
<tr>
<td>$L_{grid}$</td>
<td>Line reactance</td>
<td>1 mH (0.314 $\Omega$)</td>
</tr>
</tbody>
</table>
PCC. In addition, the voltage magnitude of the sagging phase is compared against the main grid phase voltage sag, to see what improvement has been achieved by the DG power compensation injection.

For each investigation, the voltage sag at the PCC was measured with no active power ripple, balanced grid current, and no reactive power ripple, respectively, for commanded average power injections of P only, Q only, and a combined PQ command. The power ripple objective was set by varying $K$ over the range $K = +1, 0, -1$ at 0.1s steps, for each average PQ test condition.

Experimental results are presented here for the three cases of resistive, inductive and combined RL grid network impedances, for the three operating conditions of:

- 800 W average P only injection
- 800 var Q only injection
- combined 566 W P and 566 var Q injection (apparent power equal to 800 VA).

6.1.1. Case I: Grid with resistive feeder impedance

Figure 6.2 shows the results for a grid with a resistive line impedance, where for each average PQ condition the ripple power control strategy is changed at 0.1 second intervals during the experiments, from “no active power ripple”, $K = +1$, to “balanced currents”, $K = 0$, to “no reactive power ripple”, $K = -1$. From this figure, it can be seen that irrespective of the average PQ injection levels, the power ripple control scheme always achieves the commanded ripple objective, which confirms the capability of the strategy. Furthermore, it is clear that for this resistive grid network, eliminating real power ripple achieves the lowest $n$-factor outcome, and hence this ripple objective provides the best grid voltage negative sequence mitigation irrespective of the average power injection conditions. It should be noted also for the PQ injection case that the $n$-factor has only been reduced to 0.185 (compared to 0.177 for the P only injection case). This is because the injected real power is only 566W for this case, which further confirms that only real power injection achieves mitigation of the PCC voltage unbalance for a resistive only feeder impedance.

Figure 6.3 shows how the different ripple objectives influence the absolute voltage magnitude of the sagging phase voltage at the PCC. From this figure it can been
seen that while all power ripple injection alternatives improve the sagging phase voltage, real power ripple elimination always achieves the best outcome.

Figure 6.2. Experimental results for grid with resistive characteristic. (left) Real power injection. (middle) Reactive power injection. (right) combined PQ injection.

Figure 6.3. Voltage magnitude of sagging phase: resistive grid.
6.1.2. Case II: Grid with inductive feeder impedance

Figure 6.4 shows the results for a grid with an inductive line impedance, once again with the same change in ripple power objective from “no active power ripple” to “balanced currents” to “no reactive power ripple” at 0.1 second intervals, for each average PQ injected condition.

These results show that injecting active power into the system has no significant impact on the voltage unbalance n-factor, as shown in Figure 6.4 (left). Furthermore, whenever average reactive power is injected, the n-factor increases unless the reactive power oscillation is eliminated ($K = -1$). So for a grid with an inductive characteristic, not only is the best unbalanced voltage mitigation always achieved when reactive power ripple is eliminated, but it can only essentially be restored to the uncompensated grid n-factor. This is in sharp contrast to the improvement in voltage unbalance n-factor that can be achieved with a resistive line impedance by eliminating real power ripple.

Figure 6.5 shows that for an inductive grid impedance, real power injection still achieves the best absolute sagged phase voltage increase. Hence if real power is to be injected into the grid, the best DG performance is achieved with $K = +1$, since this will maximize the increase in the sagged phase voltage without increasing the overall voltage n-factor. However, with average reactive power injection, the “no reactive power ripple” strategy is the best alternative to avoid increasing the n-factor.
Figure 6.4. Experimental results for grid with inductive characteristic. (left) Real power injection. (middle) Reactive power injection. (right) combined PQ injection.

Figure 6.5. Voltage magnitude of sagging phase: inductive grid.
6.1.3. Case III: Grid with resistive/inductive impedance

Figure 6.6 presents results for a grid with a mixed resistive/inductive characteristic, where the grid impedance has an impedance angle of 45 degrees ($Z_{\text{grid}} = 0.314 + 0.314j$). As could be expected, for active average power injection and the three different control strategies, the system performance is similar to the case of a grid with resistive characteristics, Figure 6.2 (left). However, the reduction in the $n$-factor is less than for a purely resistive grid impedance.

When average reactive power is injected into the grid the results are the same as for reactive power injection into an inductive grid, as shown in Figure 6.4 (middle). Hence the “no reactive power ripple” strategy is best for this average power injection condition. Finally, the performance with mixed average power injection is essentially the same irrespective of the power ripple objective, i.e. no significant change in the voltage $n$-factor compared to a grid without DG injection.

Figure 6.7 shows that once more, real power injection achieves the best increase in the absolute sagged phase voltage magnitude for the mixed impedance feeder irrespective of the ripple power objective.

The conclusion from these studies is that regardless of the type of grid impedance, active power injection always achieves the best absolute unbalanced voltage mitigation. Furthermore, a “no active power ripple” strategy ($K = +1$) is always the best approach unless reactive power is injected into the grid. In this case it is better to proceed to a “no reactive power ripple” strategy ($K = -1$) to keep the voltage unbalance $n$-factor at least no worse than that of the unbalanced grid supply.
Figure 6.6. Experimental results for grid with mixed RL characteristic. (left) Real power injection. (middle) Reactive power injection. (right) combined PQ injection.

Figure 6.7. Voltage magnitude of sagging phase: mixed RL grid.
6.2. Distribution Feeder Results

To study a balanced multi-bus multi-DG system, a 20-bus feeder was constructed incorporating distributed generation as shown in Figure 6.8. For this network, a DG unit modelled using the approach just presented was connected to each bus, together with a local PQ load on the bus. The total penetration of the DG units was limited to not exceed 40% of total installed load, with the remainder of the feeder power being supplied by the synchronous generator at the “Station” bus, as shown in Figure 6.8. To avoid any artificial simulated frequency coherency between the inverters, the switching frequency of each inverter was randomly set with different values between 5 kHz to 6 kHz. For this study example, while the DG units are injecting real power into the grid, a step change in reactive power reference is applied to each DG unit. The active power injection of all DG units is 4 kW and at 0.05 s the reactive power injection of DG units increases to 2 kvar.

The power output of DG units is shown in Figure 6.9. The dynamics of the system ensure a secure transition to the new operation point. The synchronous generator output powers are presented in Figure 6.10. It is shown that as could be anticipated, at 0.05 s, when reactive power generation of DG units changes, the reactive power injected by the synchronous generator decreases while its active power slightly increases because of the changes in the losses in the network.

To illustrate the impact of this increase in reactive power generation along the distribution feeder, the voltage magnitude is shown in Figure 6.11 for two buses along the feeder, one at the synchronous generator terminal (upper trace), and one for bus #20 at the end of the feeder (lower trace). As the reactive power injection increases from the distributed generation system, the voltage profile along the feeder increases, with minimal overshoot and no sign of dynamic instability during the transient event.

![Figure 6.8. 20-bus distribution feeder (Balanced).](image-url)
Figure 6.9. Injected power of DG units to the feeder.

Figure 6.10. Synchronous generator powers.

Figure 6.11. Voltage profile along the feeder.
Chapter 6  Distribution System Modelling for Unbalanced Voltage Mitigation

Figure 6.12 shows a distribution feeder under unbalanced voltage conditions consists of 10 DG units where \( P_i \) and \( Q_i \) represent the real and reactive power flowing down the feeder from node \( i \), \( p^d \) and \( q^d \) represent the DG active and reactive power generation and \( p^l \) \( [q^l] \) represents the local load active \( [\text{reactive}] \) power. The DG parameters are provided in Table 6.2.

In this section the overall performance of a feeder with 10 detailed DG units is provided. Figure 6.13 presents the power injection to the grid for each DG unit. The power set-points and the step time for each DG unit are different to prevent any coherency. Furthermore, three different operating scenarios can be seen from this figure.

For the first scenario three balanced currents \( (K=0) \) are injected into the utility

![Diagram of a distribution feeder configuration](image)

**Figure 6.12. Distribution feeder configuration (Unbalanced).**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DG1</th>
<th>DG2</th>
<th>DG3</th>
<th>DG4</th>
<th>DG5</th>
<th>DG6</th>
<th>DG7</th>
<th>DG8</th>
<th>DG9</th>
<th>DG10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S ) kVA</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>( L_f ) mH</td>
<td>6</td>
<td>12.7</td>
<td>12.7</td>
<td>12.7</td>
<td>3</td>
<td>6</td>
<td>12.7</td>
<td>12.7</td>
<td>12.7</td>
<td>12.7</td>
</tr>
<tr>
<td>( L_{fg} ) mH</td>
<td>2</td>
<td>1.27</td>
<td>1.27</td>
<td>1.27</td>
<td>1</td>
<td>2</td>
<td>1.27</td>
<td>1.27</td>
<td>1.27</td>
<td>1.27</td>
</tr>
<tr>
<td>( C_f ) µF</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>7.5</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>( 2V_{dc} ) kV</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( f_s ) kHz</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5.2</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>( f_{samp} ) kHz</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10.4</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>( V_g ) V</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>( K_p )</td>
<td>0.0192</td>
<td>0.0382</td>
<td>0.0382</td>
<td>0.0540</td>
<td>0.0192</td>
<td>0.0192</td>
<td>0.0382</td>
<td>0.0382</td>
<td>0.0382</td>
<td>0.0382</td>
</tr>
<tr>
<td>( T_i )</td>
<td>0.0042</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0026</td>
<td>0.0021</td>
<td>0.0042</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0037</td>
</tr>
</tbody>
</table>
Chapter 6  Distribution System Modelling for Unbalanced Voltage Mitigation

6.3. Unbalanced Voltage Management with High Penetration of Inverter Interfaced Generators

Conventionally, DG systems are not significantly utilised for voltage regulation of a network and mostly just inject real power into the grid. However, they are known to contribute to overvoltage and voltage quality issues in a grid system, particularly under unbalanced load/generation conditions. On the other hand, the power management ancillary capability of the inverter of the DG sources can also significantly improve network voltage quality when smart reactive power injection is considered.

The purpose of this section is to show that the previous voltage unbalance mitigation ideas for a LV feeder would readily translate to a MV system of 22kV feeders supplied from a 66 kV subtransmission system. The DG systems are represented as a single lumped PQ LV injection source at each LV distribution transformer, and the source voltage unbalance was assumed to come from an unbalanced 66kV grid. Next, real power ripple control ($K=1$) is applied to all DG units and finally reactive power ripple control ($K=-1$) is the main objective.
subtransmission supply. This scenario was selected as a representative case study, just to illustrate the same response achieved in principle as for the previous LV feeder.

### 6.3.1. System Topology- DG Unit/Microgrid Modelling

Figure 6.14 illustrates the local grid network topology used in this study, which is based on a physical distribution network. The main 33MVA transformer connects a local 22kV feeder to the 66kV external grid supply. This feeder then supplies 16 off 400V LV lines via 22/0.4kV distribution transformers. For high DG penetration levels, it can be anticipated that LV lines that have DG systems connected would deliver more than 10 kVA back into the 22kV feeder. Hence, since Australian standards [44] require DG units rated higher than 10 kVA to be three-phase systems, three-phase DG systems with capability to inject 10 kVA or more were connected at the LV side of ten of the three-phase distribution transformers to represent a high penetration level. Furthermore, to avoid any simulation coherency issues caused by the multiple DG systems, different characteristics were used for the various DG units in the local network (i.e. using

![Diagram of Feeder Configuration](image)

Figure 6.14. Feeder configuration.
different values for the LCL filters, switching frequencies and sampling rates). The system parameters which have been used for DG systems are provided in Table 6.3.

### 6.3.2. Results and Discussion

Substantial simulation studies have been carried out using the local grid model shown in Figure 6.14, with a Voltage Unbalance Factor \( n\)-factor level of 5.26% caused by an upstream unbalanced 66kV source. Figure 6.15 shows the variation in \( n\)-factor for the 10 off DG injection buses, for the three alternative ripple alternatives described in Chapter 5. As shown in this figure, \( K=+1 \) (i.e. no real power oscillation) best mitigates the unbalanced voltage condition while reactive power regulation, \( K=-1 \), is the worst case scenario from an unbalanced voltage mitigation perspective. Such a finding is an important consideration when reactive power injection is proposed for grid voltage support in unbalanced conditions.

Figure 6.16 shows the real and reactive power injections corresponding to Figure 6.15. As can be seen, the unbalanced supply voltage creates oscillating real and reactive power when a balanced current is injected, or oscillating real/ reactive power only if the DG injected current is controlled to achieve this result. Figure 6.16 also shows how rapidly the DG inverters can transition between these operating conditions. Note also that other factors such as the connection of DG units along each LV line, filtering, line impedances, R/X ratios and the DG location in the local grid can affect system dynamic response.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DG1</th>
<th>DG2</th>
<th>DG3</th>
<th>DG4</th>
<th>DG5</th>
<th>DG6</th>
<th>DG7</th>
<th>DG8</th>
<th>DG9</th>
<th>DG10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S ) kVA</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>( L_f ) mH</td>
<td>6</td>
<td>12.7</td>
<td>12.7</td>
<td>12.7</td>
<td>3</td>
<td>6</td>
<td>12.7</td>
<td>12.7</td>
<td>12.7</td>
<td>12.7</td>
</tr>
<tr>
<td>( L_{fg} ) mH</td>
<td>2</td>
<td>1.27</td>
<td>1.27</td>
<td>1.27</td>
<td>1</td>
<td>2</td>
<td>1.27</td>
<td>1.27</td>
<td>1.27</td>
<td>1.27</td>
</tr>
<tr>
<td>( C_f ) ( \mu F )</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>7.5</td>
<td>7.5</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>( 2V_{dc} ) kV</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( f_r ) kHz</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5.2</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>( f_{samp} ) kHz</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10.4</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>( f ) Hz</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>( V_g ) V</td>
<td>400400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>( K_p )</td>
<td>0.0192</td>
<td>0.0382</td>
<td>0.0382</td>
<td>0.0540</td>
<td>0.0192</td>
<td>0.0192</td>
<td>0.0382</td>
<td>0.0382</td>
<td>0.0382</td>
<td>0.0382</td>
</tr>
<tr>
<td>( T_i )</td>
<td>0.0042</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0026</td>
<td>0.0021</td>
<td>0.0042</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0037</td>
<td>0.0037</td>
</tr>
</tbody>
</table>
The result in Figure 6.16 clearly shows the same capability to control the injected PQ fundamental magnitude and double frequency ripple, and its effect on Main Bus n-factor. The results show that while the PQ control worked well, the effect on voltage unbalance was more limited than for the LV system. This is entirely to be expected given the relative size of the injected PQ and the MV feeder impedances.

Figure 6.15. VUF at each DG LV Bus PCC.

Figure 6.16. Real and reactive power injected by DG units.
6.4. Summary

In this chapter it was presented that the DG system achieves the required power ripple elimination objective under all load conditions with any $R/X$ ratio grid impedance, and allows the mitigation effect of this control strategy on the unbalanced grid voltages at the PCC to be readily investigated.

Furthermore, as the DG penetration continues to grow, such small DGs may soon be required to meet some LVRT requirement where these LVRT grid codes are imposed mostly on large scale wind farms. These LVRT grid codes requires DGs to provide certain active power or reactive power to support grid stability when grid fault occurs. The proposed technique in this chapter lays a good foundation in this regard.

Next, this chapter presented a modelling strategy for DG converters that is suitable for studying the large scale integration of DG systems into the electrical grid. The model combines the fundamental inverter switching modulation process with a leading edge current regulator and higher level real and reactive power controllers, to create a detailed representation that fully describes the inverter steady state and transient responses to grid connection events.

For unbalanced voltage mitigation studies, a precise simulation result was presented to evaluate the power ripple control strategies, exploring the performance of multiple DG inverters connected to the local network. The results show that detailed DG models allow the grid dynamics to be accurately determined during unbalanced voltage conditions, and that DG systems can either contribute to unbalanced voltages or mitigate grid voltage unbalance depending on how they are controlled.
Chapter 7

DESCRIPTION OF THE SIMULATION SYSTEMS

Simulation is the primary analysis strategy that was used to generate the results of the large scale system investigations that have been presented throughout this thesis. In order to explore such new concepts prior to their implementation in experimental structures, simulations are used as they provide time efficient solutions using numerical solutions without the need to analytically solve the problems. In this work the PSIM [100], DigiSILENT [101] and MATLAB software packages have been used extensively. The PSIM package allows detailed simulation of switched mode power systems and discrete close loop control modelling. For larger power system investigations and power flow studies, the DigiSILENT software was used, applying time domain simulations to investigate ElectroMagnetic Transients (EMT). The MATLAB software package was used for transfer function calculations and unbalanced voltage optimization studies.

This chapter provides an overview of the environment used for each of the three software packages used, presented in three sections. In each environment a specific system was developed to validate a specific concept presented in this thesis.

7.1. PSIM

This package supports discrete time analysis of electronic/electrical circuits with their corresponding controllers. The implementation of relatively simple component models and consequently achieving a better convergence rate makes the PSIM tool particularly advantageous for the analysis of such circuits.
There are four major element categories in PSIM: power circuit, control circuit, sensors and switch controller. The power circuit represents the electrical components. The sensors are used to provide control circuits with required measurements from power stage and finally the switch controllers are the drivers for the switch devices as commanded from the control stage. Using these categories, the simulation processes used in PSIM are presented in two subsections.

7.1.1. Power Stage and Sensors

Figure 7.1 shows the model of a VSI-based DG unit. A DC voltage source is connected to a VSI configuration of IGBT switches. The phase legs of the VSI are then connected to an LCL filter (as shown in Figure 7.2) which then connects to the main electricity grid through the distribution line impedance (Figure 7.3).

Sensors are placed at various positions for acquiring measurements. For the purpose of feedback current control and active damping the grid-side current and the filter capacitor current are measured and fed into the controller. The PCC voltage (between the grid-side filter inductance and the line impedance) is measured to synchronize the system to the grid voltage and also for power control purposes. Other circuit measurements are taken for exploring the circuit behaviour.

Figure 7.1. PSIM schematic diagram of VSI switching converter.
7.1.2. Controller Stage

The control circuit includes all the building blocks and calculations to generate the sequence components, calculate the current references, create the modulation commands for the inverter switches, generate the PLL, and perform the power calculations.

Figure 7.4 shows the transformations and sequence component extraction procedure for the PCC voltage. First the three phase voltages are transformed from $abc$ frame to $a\beta$ frame and then a simple time delay is applied to calculate the positive and
negative sequence components in $\alpha\beta$ frame. Then a Park transformation is used to convert these parameters to the PSRF and NSRF.

For synchronization purposes a PLL circuit was designed as shown in Figure 7.5. The positive sequence component of $v_d$ passes through a loop filter to calculate the frequency error, which is then added to the nominal frequency to achieve the estimated frequency. This frequency is integrated to yield the DG output voltage phase angle. Before calculating the current references, it is required to generate the reference real and reactive powers as well as the $K$ for ripple power control. As presented in Figure 7.6, these parameters are generated and step signal generators are used to investigate the transient response of the system to step changes of these parameters. This figure also shows the instantaneous power calculation used to verify the injected powers into the grid.

Figure 7.7 shows the current reference generation for both the PSRF and the NSRF as discussed in Chapter 5. The decoupled sequence current references are transferred to the opposite frame of reference and then the overall current reference is generated by summing the current set-points in the PSRF/NSRF and the projection of the components from the opposite frame of reference.

Figure 7.8 illustrates the implementation of PI current regulator in DSRF. The current tracking error is then passed through a combination of gains and integrators arranged as a PI compensator. A delay block represents the switching and transport delays due to digital implementation. The output is the modulation command which is then passed to the PWM modulator to control the switch timing of the IGBT switches by translating it into on-off commands for each of the switches using three-level PWM modulation, as shown in Figure 7.9. The carrier waveform is a periodic triangle oscillating at the switching frequency from -1 to +1. Comparing the modulation command with this carrier waveform creates the on-off commands for the switches.
Figure 7.4. Transformations and sequence extraction.
Chapter 7  Description of the Simulation Systems

Figure 7.5. PLL structure in PSIM.

Figure 7.6. Instantaneous power calculation and reference generating ($K$, $P_{ref}$ and $Q_{ref}$) diagrams.
Figure 7.7. Reference current generation for PSRF and NSRF.
Chapter 7 Description of the Simulation Systems

**DSRF Current Regulator**

![DSRF Current Regulator Diagram](image)

Figure 7.8. DSRF current regulation circuit diagrams.

**PWM and Gate Signals**

![PWM and Gate Signals Diagram](image)

Figure 7.9. A three-level PWM modulator in PSIM.
Chapter 7  Description of the Simulation Systems

7.2. DlgSILENT

DlgSILENT is power analysis tool which supports all standard power system analysis needs. It can handle very large power systems, including new technologies such as distributed generation. It also allows for modelling of power electronic components and control system, although not as conventionally as PSIM due to some implicit assumptions underpinning the internal simulation structure.

There are two main functions in DlgSILENT that have been used in this thesis. The first one is the DlgSILENT Simulation Language for detailed modelling and time domain simulations. The second one is the DlgSILENT Programming Language which is used for large system analysis and optimization purposes.

7.2.1. DlgSILENT Simulation Language (DSL)

An EMT simulation platform is provided in DlgSILENT to solve the power system electromagnetic transient problems. A graphical object wiring diagram platform called Composite Model Frames supports the detailed modelling of the power systems.

Distributed Generation Unit

The modelling concepts presented above have been incorporated into the DlgSILENT power system analysis package, starting with the basic power stage shown in Figure 7.10, and then building the required control system structures to create the full model DG system. The inverter modelling is based on the “PWM converter” model with two DC connections. The DC connections are connected to a DC source which represents the primary source of power. Then, to add the LCL filter, two “Common Impedances” are used for inverter-side and grid-side inductors, and a “Shunt/Filter C” is located in between these inductances for filter capacitance.

Figure 7.11 and Figure 7.12 shows the overall control structure of the DG unit as implemented in DlgSILENT for balanced and unbalanced systems respectively. Unbalanced system control modelling is based on the current regulation method and power quality management as described in Chapter 5. The composite frame consists of voltage and current measurement blocks, sample and hold blocks, clock generator, \( dq \) transformation block, \( PLL \) system, power control frame \((PQ)\), current control frame
and finally the modulation indexes which are being fed into the inverter modulator and power stage.

Figure 7.10. Power stage of the three-phase DG unit (single line presentation).
Figure 7.11. Scheme of the DG converter controller implemented in DlgsILENT (Balanced).
Figure 7.12. Scheme of the DG converter controller implemented in DIgSILENT (Unbalanced).
Phase-Locked Loop (PLL Block)

Considering the positive and negative sequence components, it is required to develop a synchronising system that can precisely make sure that the DG unit is synchronised with the grid even under unbalanced voltage conditions. Thus, in order to be synchronized with the positive sequence, a zero crossing strategy can be implemented. Alternatively, as used in this research, a voltage sequence extraction method is used to decouple the positive and negative voltage components. Then, by applying a conventional PI regulator, grid synchronisation can be achieved (Figure 7.13.)

Synchronous Reference Frame Transformations (ab2dq Block)

As shown in Figure 7.14, this block is used for transformation purposes where all the measured signals in the stationary frame are transformed to the corresponding positive/negative synchronous reference frame (PSRF/NSRF) for unbalanced systems. The outputs of this block are fed into the PQ block and Current block. In case of balanced system modelling this block is used for transformation to the positive synchronous reference frame.

Power Controller (PQ Block)

The current reference set-points are calculated in this block. As presented in Chapter 5, the current reference calculation is based on the sequence components. Therefore, flexibility is added to the controller to either minimize real or reactive power.
ripple, or to achieve balanced phase current when DG is connected to an unbalanced grid. Figure 7.15 shows the designed current reference generator frame ($PQ$) in DlgSILENT.
Figure 7.15. PQ Frame (Unbalanced).
As shown in Figure 7.16 for balanced systems, this block begins by calculating the actual real and reactive power for the converter, using the measured voltages and currents. A PI regulator then compares the power set-points (Note there is a higher level of control which generates these power set-points) against these actual powers, and uses the errors to command the required currents to achieve the commanded real and reactive power. A particular benefit of this approach is that commanded step changes in power flow translate to ramp changes in the commanded current because of the action of the PI controller, which reduces transients injected into the distribution system from the converter.

**Current Controller (Current Block)**

Figure 7.17 and Figure 7.18 show how current regulation is implemented in DIgSILENT. The reference currents are provided by the PQ block. The mathematics of the modelling of the current regulator for unbalanced conditions have already been explained in Chapter 5. In order to apply active damping, the measured capacitor currents are fed into this block as well. These currents are added as additional feedback signals after the PI regulators, with a damping gain of $K_d$.

The outputs of the current controllers are the commanded dq frame voltages that

![Figure 7.16. PQ Frame (Balanced).](image-url)
are required to be produced by the converter power stage. They are converted back into stationary frame $\alpha\beta$ commands as the output of the control block, to suit the DIgSILENT power stage representation of the converter system.

Figure 7.17. Current Frame (Unbalanced).

Figure 7.18. Current Frame (Balanced).
Sampling and Clock Generation

To account for digitalization, sampling blocks are used with the same clock sequence for all of the sampling blocks. One important issue was to ensure that the power stage voltages and currents were measured exactly at the transition point of each half carrier period (a synchronous sampling) as shown in Figure 7.19, to avoid sampling switching ripple effects. This required the inclusion of the sample and hold model (ElmSamp) into the variable measurement blocks, to keep the measured quantities constant over each half carrier period. The sampling frequency is set to twice the switching frequency of the inverter. The sampling is done at the rising edge of the clock. Therefore, the clock is defined in a fashion that the rising edge of the clock is located exactly at the minimum and maximum points of the carrier. This eliminates measurement of switching ripple current.

7.2.2. DlgsILENT Programming Language (DPL)

The DPL-Programming Language offers a flexible interface for parametric sweep calculations (load flows), data pre-processing including input/output handling and report generation. The DPL object-oriented scripting language provides a basic set of commands including C++-like object-oriented syntax, flow commands (such as "if-then-else", "do-while"), input/import, output/export and reporting routines, mathematical expressions and access to the objects and the parameters including

![Figure 7.19. Sampling procedure.](image-url)
The DPL platform was used for large power system modelling incorporating distributed generation. To investigate the PV reactive power influence on voltage profile and the optimization process, DPL codes are used.

Figure 7.23 shows the system parameters to create a DPL script and solve the optimization problem. In order to execute multiple number of load flows while accessing the power system element parameters (such as real and reactive power set-points, load profile and transformer tap setting), a DPL command is developed.

It includes the load and PV profile (refer to Figure 3.2 and Figure 4.1) using data matrices (LoadMat and PVMat), access to all loads to set the loading condition for each time period (LoadSet_a, _b and _c) and access to transformer tap setting (TransSet_a, _b and _c). DPL codes are responsible to properly apply the initial conditions to the system parameters, run the load flow with corresponding optimization objectives and finally generate the report (the DPL code can be found in Appendix B).

### 7.3. DlgSILENT vs. PSIM Model of a Single DG Unit

DIgSILENT is a power system analysis tool that is primarily used for electrical network planning and operation optimization. However, its use in the context of this investigation requires it to implement a more detailed simulation of the DG inverter system, including a quite precise representation of the inverter switching processes. Hence to validate the accuracy of its simulation, it was compared against the more established power converter simulation package – PSIM. This strategy has the further advantage that previous work [2] had already verified matching PSIM models against experimental test converters, and so there is a high level of confidence that the PSIM simulation accurately reflects the physical performance of a real world converter.

Hence an identical DG system was developed in PSIM and used as a direct reference verification for the DIgSILENT model. The measurements were taken in the same stationary frame context, and then transformed to the synchronous frame in the same way as with the DIgSILENT approach. The controller gain settings were also kept identical for both simulation implementations. The parameters for the DG unit under investigation are given in Table 7.1
Chapter 7  Description of the Simulation Systems

Table 7.1. DG Parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>DG rating</td>
<td>10 kVA</td>
</tr>
<tr>
<td>$L_f$</td>
<td>Inverter-side filter inductance</td>
<td>12.7 mH</td>
</tr>
<tr>
<td>$L_{fg}$</td>
<td>Grid-side filter inductance</td>
<td>1.27 mH</td>
</tr>
<tr>
<td>$C_f$</td>
<td>Filter Capacitance</td>
<td>15 μF</td>
</tr>
<tr>
<td>$2V_{DC}$</td>
<td>DC bus voltage</td>
<td>1000 V</td>
</tr>
<tr>
<td>$f_s$</td>
<td>Switching frequency</td>
<td>10 kHz</td>
</tr>
<tr>
<td>$f_{samp}$</td>
<td>Sampling frequency</td>
<td>20 kHz</td>
</tr>
<tr>
<td>$f$</td>
<td>Grid frequency</td>
<td>50 Hz</td>
</tr>
<tr>
<td>$V_g$</td>
<td>Nominal grid voltage</td>
<td>400 Vrms</td>
</tr>
<tr>
<td>$K_p$</td>
<td>Proportional gain</td>
<td>0.0382 A$^{-1}$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>Integral reset time</td>
<td>0.0036 s</td>
</tr>
</tbody>
</table>

Figure 7.20 shows results for both packages, with the left and right hand columns in this figure providing results for DlgsILENT and PSIM respectively. For these results, the DG unit is controlled in an active/reactive power control mode. The reactive power set-point is set to zero, while the active power has a non-zero value which has a commanded step change from 4 kW to 9 kW at 0.04 s.

The first row of Figure 7.20 shows the three-phase grid currents for both simulations, where it can be seen that the results are essentially identical to each other (including in particular the transient responses).

The next two rows of Figure 7.20 show the dynamic response of the power

Figure 7.20. Simulation Results. Left: DlgsILENT Results. Right: PSIM Results.
controller and the current controller respectively in the $dq$ frame for the step change in the commanded real power at 0.04 seconds. For the power controllers, shown in the second row of Figure 7.20, the inverter output real power quickly follows its set-point from 4 kW to 9 kW within a fraction of a fundamental cycle without any overshoot or stability oscillation. At the same time, the reactive power is kept at 0 Var, again without any significant transient overshoot. In fact, there is some evidence from detailed investigation that the small fluctuation in reactive power that occurs during the real power step change is actually a measurement artefact, as the PLL tracks the change in network load angle of the DG system as the real power increases. Matching results obtained with the PSIM model on the right hand side of Figure 7.20 confirm the validity of the DIgSILENT simulation response.

For the current regulator, the commanded active power corresponds to the direct axis current, while the commanded reactive power corresponds to the quadrature axis current. Hence the real power change causes a step in the $d$-axis current ($i_d$), while the $q$-axis current component ($i_q$) remains at zero because there is no reactive power injection into the grid. The last row of Figure 7.20 shows how the current regulator very accurately tracks the currents commanded by the higher level power regulators. Once again, the essentially identical response of the DIgSILENT and PSIM models confirms the validity and accuracy of the DIgSILENT model. Note also how the current references do not actually change instantly, because of the somewhat delayed response.

![Phase A Currents](image)

Figure 7.21. Phase A current of both simulations.
of the PI power regulators to the actual commanded power step.

As final confirmation of the performance of the DIgSILENT model, the phase A currents for the two simulation packages are plotted in detail against each other in Figure 7.21. The steady state results match perfectly, while the transient current responses during the commanded power step are extremely close. The slight differences in transient response are essentially caused by the differing approach used for the numerical calculations in each package, and are not significant.

To investigate the performance of a DG under unbalanced voltage conditions, a control system is designed in DIgSILENT based on the control algorithms developed in Chapter 5. The DG parameters are given in Table 7.1. Four different cases have been considered for simulation purposes:

1) 4-9 ms: $P_{ref}=3$ kW, $Q_{ref}=0$ Var and $K=0$
2) 9-14 ms: $P_{ref}=6$ kW, $Q_{ref}=0$ Var, $K=0$
3) 14-19 ms: $P_{ref}=6$ kW, $Q_{ref}=0$ Var, $K=1$
4) 19-20 ms: $P_{ref}=6$ kW, $Q_{ref}=0$ Var, $K=-1$
Simulation results in Figure 7.22 validate the controller effectiveness that has been designed in this study achieving the desired power ripple control with smooth transient response.

### 7.4. MATLAB

The main use of the MATLAB software is to model electronic transformer tap changing and calculate bode plots of various transfer functions. This is done using Simulink and m files in MATLAB. Figure 7.24 shows a transformer with an electronic tap changer as modelled in MATLAB. To investigate the transient response to tap down/up with different load characteristics, a transformer with 4 tap positions is used. Then thyristors are placed on each tap to open and close for tap changing. The gate signals are generated using step functions to create the proper switching sequence when a tap is changing.
Chapter 7 Description of the Simulation Systems

7.5. Summary

This chapter has presented an overview of the PSIM, MATLAB, DIgSILENT-based simulation systems used throughout this thesis. The built in power electronics and control blocks in PSIM are used to model the power stage, PWM modulator and the control algorithm for individual DG inverters. The simulation was developed to validate the proposed algorithms prior to match the experimental systems.

The DIgSILENT package was used for large scale power system modelling while including detailed and average models of power electronic converters. The models are used to investigate load flow analysis, time domain analysis (transient response) and also for optimization when reactive power is managed throughout a feeder. Coordinated and local control approaches are developed in DPL scripts while detail modelling is implemented in DSL.

Finally, MATLAB m-files and Simulink are used for electronic tap changing studies and calculating bode plots.
Chapter 8

DESCRIPTION OF THE EXPERIMENTAL SYSTEMS

To validate the performance of the theoretical developments and simulation results presented in this thesis, a 5 kVA grid connected inverter was established using the experimental system and equipment in the RMIT power electronics laboratory.

This chapter presents the experimental systems in two sections. First, the hardware is presented for both the power stage of the converter and the controller stage. Secondly, the software for this experimental setup is presented including the foreground and background tasks and the user interfaces which are used to communicate with the operating system.

8.1. Hardware

Figure 8.1 presents a block diagram of the experimental system. The three phase inverter is supplied from a DC source and then it is connected to the grid via an LCL filter. The current measurements are fed into the Controller motherboard (CPT-E13). Then, the control algorithm in the DSP controller card (CPT-DA2810) processes the data and generates the modulation indices. Finally, the gate drivers generate the switching commands to be applied to the IGBT switches. The user interface is provided by RS-232 connection, enabling the user to apply a step change to the reference values, grab the parameter values and perform debugging.
8.1.1. Power Stage

For the power stage, the main components are the three-phase VSI (voltage source converter), LCL filter and grid simulator.

Three-Phase Inverter: The experimental system is based on a modified NY-VSI converter supplied by Creative Power Technologies (CPT) [102]. The main components of this system include three pairs of 1200V 50A IGBT (three-phase VSI), protective devices, DC link and four bulk capacitors as shown in Figure 8.2.

LCL Filter: The LCL consists of inverter-side inductor (Figure 8.3), grid-side inductor (Figure 8.4), filter capacitor and common mode inductor (Figure 8.5). Three matching configurable inductors are used for the inverter-side inductors ranging from 200 µH to 20 mH. The grid side inductors are three commercial laminated steel core inductors with a value of approximately 1 mH. A custom made inductor by wrapping insulated wire was used for the common mode inductor.
Figure 8.2. Three-phase H-bridge and controller boards.

Figure 8.3. LCL filter (converter side inductor).
Grid Simulator: For grid connected experiments and in order to create unbalanced voltages a California Instruments MX30-3Pi 30kVA programmable bidirectional AC source is used [103], as shown in Figure 8.6. This grid simulator can produce harmonic free arbitrary AC waveforms.

DC Power Supply: A current limited DC supply (MagnaPower TS series) is used capable of producing regulated 1000 V DC. The front panel of DC source is shown in Figure 8.7.

8.1.2. Controller Stage

The DSP board and measurement devices are located all within the controller stage.
Chapter 8  Description of the Experimental Systems

DSP Controller: The controller boards include a CPT-E13 board and CPT-DA2810 DSP sub-board which contains the TI TMS320F2810 DSP [104] and its circuitry as shown in Figure 8.2. Functional diagrams of the CPT-E13 and CPT-DA2810 are shown in Figure 8.8 and Figure 8.9 respectively.

The communication with the inverter is provided by a Serial Communication Interface (SCI) for chip programming, to read measurements and variables, or to command new reference values. An isolated RS-232 serial communication board provides the connection to the user’s controlling laptop.

Figure 8.6. Grid simulator (California Instruments MX30 programmable AC source).

Figure 8.7. MegaPower DC supply
Figure 8.8. Functional diagram of CPT-E13 [19].

Figure 8.9. Functional diagram of CPT-DA2810 [19].
8.2. Software

For operation of the DSP, a software was developed which is based on the previously written code structures available from CPT engineers and the Power and Energy Group (PEG) of RMIT University. It contains serial communication, analogue to digital converter (ADC) readings, closed-loop modulation and a user interface.

There are two levels of DSP operation. First is the continuous-loop background routine which is responsible for low priority tasks such as the user interface. Second is the interrupt-driven foreground tasks which are responsible for closed-loop regulation, PWM modulation and protection. The DSP code can be found in Appendix C.

8.2.1. Background Routine

In general, a state machine controls the operation of the system. There are several states involved in the closed loop operation of a grid connected inverter. Some of them have initialization roles such as initializing the ADCs, DC voltage pre-charge, grid voltage phase sequence detection and phase synchronization. Other states are user triggered such as Stop or Start commands. There is also a fault state which provides protection for fault events by disabling the PWM switches and opening the main contactor.

8.2.2. Foreground Routine

The functionality of the interrupt routine can be expressed by the following sequential tasks. First the analogue measurements are recorded, then a zero crossing detection method is used to check the synchronization to the grid. After that over-voltage and over-current protections check the system’s status (for in a fault event the machine state changes to State Fault). Both software and hardware fault detection help to provide system protection. Next, the start-up control applies the initial conditions to all internal controllers. Before current regulation, the reference generation is responsible for generating the references for negative sequence current components as well as for the positive sequence current components in DSRF. This section also allows the step changes to be applied to the current reference commands. It is followed by current regulation using PI controllers and then active damping of the capacitor current.
final stage is the modulator calculation where the voltage command is translated to an appropriate clock count for the hardware PWM comparators of the TMS320F2810.

8.2.3. User Interface

Interface to Grid Simulator

For the user interface the MXGUI software is used to communicate with the grid simulator. Using this USB based communication channel the three phase voltages can be set while also defining the phases, magnitudes and current limit values. The grid simulator contactors can open and close using the same software.

Communication with DSP board

To communicate with the DSP controller board, Tera Term software is used providing the text based terminal. This bidirectional interface allows using the host computer keyboard to set parameters such as current references, step change values and start or run the system.

8.3. Summary

This chapter has provided details of the experimental systems used to validate the work presented in this thesis. A programmable California Instruments MX30-3Pi 30kVA is used to simulate the unbalanced grid voltages. A CPT-NY converter is used then which is connected to the grid simulator through an LCL filter. A CPT-E13 and CPT-DA2810 controller boards were used for DSP based control system using a TI TMS320F2810 DSP. Finally, an overview of the foreground and background tasks implemented in these systems has been presented.
Chapter 9

CONCLUSIONS

Increasing levels of distributed generation penetration in the utility grid are changing conventional electrical network characteristics. Such distributed sources can be placed anywhere in the system including near or at the end user, with multiple small-scale technologies being used to produce electricity. In addition, these technologies can take advantage of renewable energy sources such as wind and solar. Consequently, this can cause bidirectional flow of power in the utility network, which complicates the control of real and reactive power flows. Therefore, control of a DG unit becomes very important as its contribution to the overall system performance can be positive, or it can worsen the situation. The control of such DG units has been the focus of significant research effort for more than a decade, with the control under abnormal conditions (such as unbalanced voltages) being given more focus in more recent years.

Knowledge and understanding from current regulation and power control have been applied in this thesis to the control of a DG system, specifically with the aim of determining the factors required for unbalanced voltage support. In addition, voltage support in an LV feeder with high DG penetration has been explored in terms of implementation, dynamic performance and the use of DG inverters to contribute using reactive power injection.

The work carried out in this thesis makes a contribution to the field of distributed generation system voltage support. The contributions are summarised in the following sections, including both the use of DSRF current regulation without requiring extraction of the sequence current components and the development of an oscillation power control method to best support the unbalanced voltages. Furthermore, the roles of DG PV reactive power and electronic tap changer in voltage regulation of distribution
feeders are presented. There is also scope to extend the real and reactive power control to regulate the voltage of distribution feeders, or to apply the knowledge of transformers with electronic tap changers to compensate for unbalanced conditions. This chapter also provides suggestions for future work.

9.1. Summary of Research


This thesis explores the voltage regulation performance of four previously reported reactive power support strategies for PV distributed generation systems, identifying their weaknesses and limitations as their feeder load and PV generation vary over a daily profile.

To investigate the effectiveness of these approaches on LV feeder voltage regulation, advanced analysis and simulation results have been included in the thesis. It is shown that reactive power management (RPM) and power loss minimization (PLM) methods are not capable of maintaining the feeder voltages in particular with peak PV penetration. On the other hand, the voltage rise minimization (VRM) method keeps the voltages within the regulatory limits at the expense of significantly increasing the feeder losses. Lastly, power curtailment (PCS) was presented as an alternative to reactive power to regulate the voltage but it was shown that this strategy curtails a significant amount of real power which is in direct contrast to the DG concept that tries to increase the level of DG penetration.

9.1.2. Combined Strategy for LV Feeder Voltage Regulation

This thesis presents a new combined voltage regulation strategy for an LV feeder under high PV penetration levels. The strategy proposes an electronic tap changer for the feeder distribution transformer from the MV network, and then uses local reactive power injection from each DG inverter system to minimise feeder currents and hence minimize feeder losses. The performance of the new strategy is compared against four exemplar state-of-the-art reactive power injection strategies, looking at voltage rise across the feeder and feeder losses over a daily load and PV energy injection profile. The new system achieves excellent voltage regulation with very high levels of PV
penetration and even with rapidly fluctuating PV injection levels, using only a very simple compensation algorithm based on the measured power flowing through the distribution transformer.

The tap changing transformer is the primary voltage control mechanism. The tap position is selected based on a simple characteristic between the real power flow and its direction and the tap position of the electronic tap changer. Then the remaining capacity of the PV inverters is used to inject reactive power to compensate for local reactive loads to minimize the losses in the feeder.

9.1.3. DSRF Current Regulation without Sequence Current Extraction

This thesis has demonstrated how positive and negative sequence three phase currents can be injected into an unbalanced three phase grid without requiring sequence current separation of the measured feedback current. The approach considerably reduces the complexity of the current controller, and, when maximum possible PI controller gains are used, improves the dynamic response of the current regulation strategy.

The need for current sequence extraction was eliminated by reshaping the current references by finding the projection of each sequence component on the opposite reference frame. So both the reference and the feedback currents have both DC and double frequency AC terms. It is now just required to find the \( d \) and \( q \) components of the measured currents in PSRF and DSRF using simple Park Transformation.

The parallel PI controllers for PSRF and NSRF were shown to be able to regulate the AC term as well as the DC term while working simultaneously and together they can compensate for the AC term as it is present on the opposite frame of reference. Furthermore, it was analytically proven that established theory can be used to find the maximized gains for the PI controller to achieve the optimal performance out of the controller. Detailed simulation and experimental results were presented to confirm the proposed strategy while offering much simpler control structure and better transient response.

9.1.4. Flexible Oscillating Power Control

This thesis has presented a strategy to vary between eliminating the real power ripple, the reactive power ripple, or to just balance the grid currents, for a DG inverter
operating into a grid network with unbalanced voltages at the PCC. The strategy calculates positive and negative sequence current references from the required average power injection and power ripple objectives, and feeds these references into synchronous frame closed loop current regulators. The system achieves the required power ripple elimination objective under all load conditions with any R/X ratio grid impedance, and allows the mitigation of the unbalanced grid voltages at the PCC to be readily investigated.

Detailed experimental results were presented to confirm the viability of the strategy, and to explore its influence on sagging phase voltages for a variety of injection conditions. It was shown that real power injection when real power oscillations are regulated is the most contributing strategy for unbalanced voltage mitigation. On the other hand, if the real power is limited and reactive power must be used, in grids with resistive characteristics no significant unbalanced voltage support is achieved and in inductive grids reactive power oscillation regulation is the most beneficial approach. One of the main reasons that the reactive power injection is not significantly affecting the unbalanced conditions is that it increases the un-sagged phase voltages as well as the sagged phase voltages. Consequently, overvoltage can be expected in the normal phases while trying to support the sagged phase voltage.

9.2. Future Work

While this thesis has presented the unbalanced voltage support of DG units and the voltage regulation of distribution feeders with a combined strategy, there is still scope for further research in this area.

This thesis has shown the impact of real and reactive power oscillation on the PCC voltages and also different operating conditions have been taken into account to achieve the best unbalanced voltage support by regulating these power ripples. Despite the fact that the analysis in this thesis shows the significant impact of the control approach on the PCC voltages, finding the optimal solution to maximize the unbalanced voltage support still needs to be explored. In particular, the impact of the $K$ parameter outside of the +1 and -1 limit have yet to be fully appreciated.

The work presented in this thesis investigates locally controlled DG systems to regulate feeder voltage. The coordinated control approaches can be explored in this case
as well to fully understand the performance and effectiveness of each control approach while understanding their limitations. Coordinated control of the load and the DG power is yet to be explored fully to identify the stability limitations of the control scheme.

Voltage regulation of distribution systems has been presented in this thesis by using electronic tap changers to provide fast voltage regulation. While the voltage regulation performance under balanced conditions has been validated through extensive studies, in principle their extension to unbalanced systems is straightforward. However, this requires validation. Furthermore, the proposed power ripple control can also be applied to the three-phase distribution networks to further investigate their contribution to unbalanced voltage mitigation.

9.3. Thesis Closure

The increasing levels of DG systems in the distribution network present a challenge for achieving stable performance. From a DG unit control point of view, unbalanced voltage support is an opportunity to provide ride-fault-through capability using synchronous frame controllers as they offer a direct relationship between \( d/q \) axis and real/reactive powers. On the other hand, from power system point of view, the integration of numerous sources in the distribution level has made conventional voltage regulatory mechanisms unsatisfactory. These needs have driven substantial research into DG current and power control design and LV feeder voltage regulation with high DG penetration.

This thesis has presented an analysis of current and power control of grid-connected inverters under unbalanced voltage conditions using the synchronous reference frame, leading to the development of a DSRF current regulator without requiring sequence decomposition and offering better performance. With this controller in place, the impact of positive and negative sequence currents on the oscillating powers and consequently on the unbalanced voltage mitigation was explored. In addition, LV feeder voltage regulation was investigated, proposing a combined strategy using an electronic tap changer as the primary voltage regulation method and using PV reactive power capability to minimize the feeder losses.

The control and design ideas presented in this thesis make a significant contribution towards the use of DSRF current regulators for unbalanced voltage support,
promoting simpler design and mitigating unbalanced conditions while integrating an electronic tap changer into the main distribution transformer to achieve voltage regulation over wide range of operating conditions.
Appendix A

CURRENT REGULATOR GAINS CALCULATIONS

This appendix provides the current regulation gain calculations required for DSRF current controllers.

A.1. Closed-Loop Current Control with DSRF

For unbalanced voltage conditions where a DSRF current regulation is applied, the maximum possible gains for the PI current regulators in each SRF can now be set by recognising that each synchronous frame regulator can be transformed back to an equivalent stationary frame controller according to [71]. In [71] the frequency-domain analysis of three-phase linear current regulators is addressed and their relationship in terms of moving from synchronous frame to stationary frame or vice versa has been shown. The same principle here has been used to analytically design the current regulator. Figure A.1 shows a reference frame current regulator with cross coupling (the

![Figure A.1. A reference frame current regulator with cross coupling.](image-url)
cross coupling terms \((H_{12}(s)\) and \(H_{21}(s)\)) in the proposed controller are zero and in a general format \(H_{12}(s) = H_{21}(s)\).

\[
\begin{align*}
V_1 &= H_{11}(s)E_1(s) + H_{12}(s)E_2(s) \\
V_2 &= H_{21}(s)E_1(s) + H_{22}(s)E_2(s)
\end{align*}
\]  

(A.1)

Applying \(dq\) frame to \(\alpha\beta\) frame transformation to equation (A.1), results in transforming any current regulator with cross coupling terms in synchronous frame to an equivalent current regulator in stationary frame.

\[
\begin{bmatrix}
V_p(s) \\
V_a(s)
\end{bmatrix} = \frac{1}{2} \begin{bmatrix}
A(s) & B(s) \\
-B(s) & A(s)
\end{bmatrix} \begin{bmatrix}
E_p(s) \\
E_a(s)
\end{bmatrix}
\]  

where

\[
A(s) = \begin{bmatrix}
H_{11}^p(s + j\omega_o) + jH_{12}^p(s - j\omega_o) \\
H_{11}^p(s - j\omega_o) - jH_{12}^p(s + j\omega_o)
\end{bmatrix} 
\]  

and

\[
B(s) = \begin{bmatrix}
H_{12}^p(s + j\omega_o) + jH_{11}^p(s - j\omega_o) \\
H_{12}^p(s - j\omega_o) + jH_{11}^p(s + j\omega_o)
\end{bmatrix}
\]  

(A.2)

Note that superscript \(\phi^s\) denotes a stationary frame quantity and the superscript \(\phi^e\) denotes a synchronous frame quantity. Assuming that this is a PI regulator in PSRF, the projection of this regulator to the stationary reference frame using (A.2) is

\[
\begin{bmatrix}
G_{(dq)PI}^{PSRF} \\
\end{bmatrix}_{dq} = \begin{bmatrix}
K_p + \frac{K_i}{s} & 0 \\
0 & K_p + \frac{K_i}{s}
\end{bmatrix}
\]  

(A.3)

\[
\rightarrow G_{(dq)PI}^{PSRF} = \begin{bmatrix}
K_p + \frac{K_i}{s^2 + \omega_o^2} & \frac{K_i\omega_o}{s^2 + \omega_o^2} \\
\frac{K_i\omega_o}{s^2 + \omega_o^2} & K_p + \frac{K_i}{s^2 + \omega_o^2}
\end{bmatrix}
\]

Applying the same approach for the PI regulator in the NSRF gives a transformation to the stationary frame of

\[
\begin{bmatrix}
G_{(dq)PI}^{NSRF} \\
\end{bmatrix}_{dq} = \begin{bmatrix}
K_p + \frac{K_i}{s} & 0 \\
0 & K_p + \frac{K_i}{s}
\end{bmatrix}
\]  

(A.4)

\[
\rightarrow G_{(dq)PI}^{NSRF} = \begin{bmatrix}
K_p + \frac{K_i}{s^2 + \omega_o^2} & \frac{K_i\omega_o}{s^2 + \omega_o^2} \\
\frac{K_i\omega_o}{s^2 + \omega_o^2} & K_p + \frac{K_i}{s^2 + \omega_o^2}
\end{bmatrix}
\]

Summing the two transformed reference controllers now gives
Appendix A  Current Regulator Gains Calculations

\[
\begin{bmatrix}
G_{(dq)pl}\end{bmatrix}_{\alpha\beta}^P = \begin{bmatrix}
G_{(dq)pl}\end{bmatrix}_{\alpha\beta}^N
\end{bmatrix}_{\alpha\beta} = \begin{bmatrix}
2(K_p + \frac{K_i s}{s^2 + \omega_0^2}) & 0 \\
0 & 2(K_p + \frac{K_i s}{s^2 + \omega_0^2})
\end{bmatrix} \tag{A.5}
\]

which is a conventional stationary frame PR controller with no cross coupling terms.

A.2. Maximum Current Regulator Gains

Figure A.2 shows a s-domain representation of a dual-loop single phase current regulator which is appropriate for this application [65]. A PR controller transfer function for this system is given by

\[
G_c(s) = 2\left(K_p + \frac{K_i s}{s^2 + \omega_0^2}\right) = 2K_p\left(1 + \frac{1}{T_i}\frac{s}{s^2 + \omega_0^2}\right) \tag{A.6}
\]

where \(K_p\) and \(T_i\) are the proportional gain and the integral reset time of the controller. Note also that in this s-domain representation, the inverter has been modelled as a DC gain \(V_{DC}\), while an exponential delay block \(e^{-sT_d}\) has been included in the forward path of the feedback loops to represent the effects of controller PWM sampling and transport delay \(T_d\). As shown in Figure A.2, active damping is introduced to the system, with feedback of the capacitor current \(i_c(s)\) via a damping gain \(K_d\). Therefore, the LCL filter transfer function is now split into two parts to model this arrangement, as

\[
G_{ic}(s) = \frac{1}{s L_f (s^2 + \omega_{res}^2)} \tag{A.7}
\]

\[
G_i(s) = \frac{i_g(s)}{i_c(s)} = \frac{\gamma L_c}{s^2} \tag{A.8}
\]

where \(\omega_{res} = \frac{L_f + L_fg}{\sqrt{L_f L_fg}}\) is the resonance frequency of the LCL filter, \(L_f\) and \(L_fg\) are the

![Figure A.2. Dual loop regulator with capacitor current active damping.](image-url)

\[i_{g\_reg}(s)\]
Appendix A  Current Regulator Gains Calculations

inverter-side and grid-side filter inductances, $C_f$ is the filter capacitance, $V_{DC}$ is half of the DC bus voltage and $\gamma_{LC} = \frac{1}{\sqrt{L_f g C_f}}$

Hence for a given design phase margin $\phi_m$ accounting for the inherent physical implementation delay $T_d$ in the feedback loops, the maximum possible gains of the current controller can be calculated. To analyse the frequency response of the regulator, its open loop forward path expressions have been developed using (A.6)-(A.8), to achieve

$$H(s) = \frac{G_c(s)V_{DC}G_{ic}(s)G_i(s)}{e^{-sT_d} + K_dV_{DC}G_{ic}(s)}$$  \hspace{1cm} (A.9)

Using [66], for a given design phase margin $\phi_m$ and accounting for the inherent physical implementation delay $T_d$ in the feedback loops, the maximum possible gains of the current controller can be calculated by transforming the model of Figure A.2 into the $z$-domain. The analysis then proceeds by determining whether the resonant frequency of the LCL filter, $\omega_{res}$, is below the critical frequency $\omega_{crit}$, where stabilisation via active damping is required [66] (which it is for this example), where:

$$\omega_{res} = \frac{L_f + L_{fg}}{\sqrt{L_f L_{fg} C_f}} \hspace{1cm} \omega_{crit} = \frac{\pi}{3T}$$  \hspace{1cm} (A.10)

where $T$ is the controller sampling period ($T = 1/f_{samp}$).

When the filter resonant frequency is below the controller critical frequency, to achieve an adequate stability phase margin the controller crossover frequency should be set to $\omega_c = 0.3\omega_{res}$. The controller forward path gain is then set to unity at the crossover frequency in order to calculate the maximum possible proportional gain, viz.:

$$K_p \approx \frac{\omega_c (L_f + L_{fg})}{2V_{DC}}$$  \hspace{1cm} (A.11)

Finally, the integral reset time is set to:

$$T_i \approx \frac{10}{\omega_c} = \frac{1}{2K_i}$$  \hspace{1cm} (A.12)

so that the controller resonator phase contribution is small at the controller crossover frequency.
Appendix B

SELECTED DPL SCRIPTS

This appendix provides the developed code in the DPL for the simulation systems investigated for this research.

! Definitions
int ierr, i, LFC, LFT, n, inc, dec, rows, cols, Lag, PLC;
double
K_Trans_Neg, Tap_No, K_Trans, maxVol, count, threshold, pf_step, minVol, x, y, z, Q_Cap, S_Rating, load_factor, Ptot, Pstep, Load_pfactor, DistLoad_real, DistLoad_reactive, DG_real, DG_reactive, PVPower, PVPower_real, PVPower reactive, PVPowerpu, PVRating, pfactor;
string s, s1, s2, s3;
object O, OT;
set S, ST;
threshold = 0.001; ! for ploss minimization
pf_step = 0.1; ! rate of change in PV pf
maxVol = 1.1; ! pu
minVol = 0.89; ! pu
PVRating = 4; ! kW
S_Rating = 4; ! kVA inverter rating

! for calculating the transformer tap position based on the proposed controller
K_Trans = 0.15;
K_Trans_Neg = 0.25;

Load_pfactor = 0.85; ! for loads
Lag = 1; ! lag=1 from grid to inverter...lead=0 from inverter to the grid
load_factor = 1; ! load will be multiplied into this number

! Excel setup

ierr = ddeOpen('', 'Excel', 'System');
if (.not.ierr)
{
! excel can be opened. Now close and connect to a specific sheet
ddeClose();
ierr = ddeOpen('', 'Excel', 'Sheet1');
if (ierr)
{
Info('No Sheet1 yet, creating one...');
ierr = ddeOpen('', 'Excel', 'Sheet1');
if (ierr)
{
printf('Cannot open DDE to Excel');

151
} 
else 
{ 
! create a new sheet 
  ddeExe('[New(1)]'); 
  ddeClose(); 
  ierr = ddeOpen('', 'Excel', 'Sheet1'); 
} 
} 
} 
if (ierr) 
{ 
  Error('Could not open DDE connection'); 
  exit(); 
} 
! write bus names 
S = BusSet_a.AllBars(); 
O = S.First(); 
x = 1; 
y = 1; 
while (O) 
{ 
  ! Bus Names Feeder a 
  s1 = sprintf('R%dC%d',1,y); 
  s2 = O:loc_name; 
  ddePoke(s1,s2); 
  y = y + 1; 
  O = S.Next(); 
} 
S = BusSet_b.AllBars(); 
O = S.First(); 
while (O) 
{ 
  ! Bus Names Feeder b 
  s1 = sprintf('R%dC%d',1,y); 
  s2 = O:loc_name; 
  ddePoke(s1,s2); 
  y = y + 1; 
  O = S.Next(); 
} 
S = BusSet_c.AllBars(); 
O = S.First(); 
while (O) 
{ 
  ! Bus Names Feeder c 
  s1 = sprintf('R%dC%d',1,y); 
  s2 = O:loc_name; 
  ddePoke(s1,s2); 
  y = y + 1; 
  O = S.Next(); 
} 
! Feeder a transformer tap 
  s1 = sprintf('R%dC%d',1,y); 
  s2 = 'Trans a Tap'; 
  ddePoke(s1,s2); 
  y = y + 1; 
! Feeder a transformer active power 
  s1 = sprintf('R%dC%d',1,y); 
  s2 = 'P Transfer a'; 
  ddePoke(s1,s2); 
  y = y + 1; 
! Feeder a transformer reactive power 
  s1 = sprintf('R%dC%d',1,y); 
  s2 = 'Q Transfer a'; 
  ddePoke(s1,s2); 
  y = y + 1; 
! Feeder a Losses P
Appendix B
Selected DPL Scripts

s1 = sprintf('R%dC%d',1,y);
s2 = 'P Losses a';
ddePoke(s1,s2);
y = y + 1;

!Feeder a Losses Q
s1 = sprintf('R%dC%d',1,y);
s2 = 'Q Losses a';
ddePoke(s1,s2);
y = y + 1;

!Feeder b transformer tap
s1 = sprintf('R%dC%d',1,y);
s2 = 'Trans b Tap';
ddePoke(s1,s2);
y = y + 1;

!Feeder b transformer active power
s1 = sprintf('R%dC%d',1,y);
s2 = 'P Transfer b';
ddePoke(s1,s2);
y = y + 1;

!Feeder b transformer reactive power
s1 = sprintf('R%dC%d',1,y);
s2 = 'Q Transfer b';
ddePoke(s1,s2);
y = y + 1;

!Feeder b Losses P
s1 = sprintf('R%dC%d',1,y);
s2 = 'P Losses b';
ddePoke(s1,s2);
y = y + 1;

!Feeder b Losses Q
s1 = sprintf('R%dC%d',1,y);
s2 = 'Q Losses b';
ddePoke(s1,s2);
y = y + 1;

!Feeder c transformer tap
s1 = sprintf('R%dC%d',1,y);
s2 = 'Trans c Tap';
ddePoke(s1,s2);
y = y + 1;

!Feeder c transformer active power
s1 = sprintf('R%dC%d',1,y);
s2 = 'P Transfer c';
ddePoke(s1,s2);
y = y + 1;

!Feeder c transformer reactive power
s1 = sprintf('R%dC%d',1,y);
s2 = 'Q Transfer c';
ddePoke(s1,s2);
y = y + 1;

!Feeder c Losses P
s1 = sprintf('R%dC%d',1,y);
s2 = 'P Losses c';
ddePoke(s1,s2);
y = y + 1;

!Feeder c Losses Q
s1 = sprintf('R%dC%d',1,y);
s2 = 'Q Losses c';
ddePoke(s1,s2);
y = y + 1;

! Load Flow Analysis Begins

rows = LoadMat.SizeX();
for (i = 1; i < rows+1; i=i+1)
Appendix B  Selected DPL Scripts

!!!!!!CREATING INITIAL CONDITIONS!!!!!!!
nfactor = 1; ! for PVs
DistLoad_real = LoadMat.Get(i,1);
DistLoad_reactive = DistLoad_real * sin(acos(Load_pfactor)) / Load_pfactor;
PVPowerpu = PVMat.Get(i,1);
PVPower = PVPowerpu * PVRating;
PVPower_real = PVPower;
Q_Cap = sqrt(sqr(S_Rating) - sqr(PVPower_real));
PVPower_reactive = DistLoad_reactive * load_factor;
if (PVPower_reactive > Q_Cap)
{
PVPower_reactive = Q_Cap ;
}
DG_real = DistLoad_real * load_factor - PVPower_real;
! DG_reactive = DistLoad_reactive * load_factor - Q_Cap;
DG_reactive = DistLoad_reactive * load_factor - PVPower_reactive;

!!!!Feedr a loads setting
S = LoadSet_a.All();
O = S.First();
while (O)
{
  O:plini = DG_real ; ! KW
  O:qlini = DG_reactive; ! kVar
  O = S.Next();
}

!!!!Feedr b loads setting
S = LoadSet_b.All();
O = S.First();
while (O)
{
  O:plini = DG_real ; ! KW
  O:qlini = DG_reactive; ! kVar
  O = S.Next();
}

!!!!Feedr c loads setting
S = LoadSet_c.All();
O = S.First();
while (O)
{
  O:plini = DG_real ; ! KW
  O:qlini = DG_reactive; ! kVar
  O = S.Next();
}

!!!!Feedr a Transformer setting
S = TransSet_a.All();
O = S.First();
while (O)
{
  O:nntap=0;
  O = S.Next();
}

!!!!Feedr b Transformer setting
S = TransSet_b.All();
O = S.First();
while (O)
{
  O:nntap=0;
  O = S.Next();
}

!!!!Feedr c Transformer setting
S = TransSet_c.All();
O = S.First();
while (O)
{
    O:nntap=0;
    O = S.Next();
}

!!!!Load flow
LDF.Execute();

!!!! Trans C
ST = TransSet_c.All();
OT = ST.First();
z = OT:m:Psum:buslv;
if (z > 0)
{
    Tap_No = round(z * K_Trans);
}
else
{
    Tap_No = round(z * K_Trans_Neg);
}
if (Tap_No < OT:t:ntpnm)
{
    if (Tap_No > OT:t:ntpmx)
    {
        OT:nntap = Tap_No ;
    }
}
if (Tap_No > OT:t:ntpmm)
{
    if (Tap_No < OT:t:ntpnm)
    {
        OT:nntap = OT:t:ntpmx;  
    }
}
if (Tap_No < OT:t:ntpnm)
{
    if (Tap_No > OT:t:ntpmm)
    {
        OT:nntap = OT:t:ntpnm;  
    }
}

!!!! Trans B
ST = TransSet_b.All();
OT = ST.First();
z = OT:m:Psum:buslv;
if (z > 0)
{
    Tap_No = round(z * K_Trans);
}
else
{
    Tap_No = round(z * K_Trans_Neg);
}
if (Tap_No < OT:t:ntpnm)
{
    if (Tap_No > OT:t:ntpmx)
    {
        OT:nntap = Tap_No ;
    }
}
if (Tap_No > OT:t:ntpmm)
{
    if (Tap_No < OT:t:ntpnm)
    {
        OT:nntap = OT:t:ntpmx;  
    }
}
if (Tap_No < OT:t:ntpnm)
{
    if (Tap_No > OT:t:ntpmm)
    {
        OT:nntap = OT:t:ntpmm;  
    }
}

!Trans A
ST = TransSet_a.All();
OT = ST.First();
z = OT:m:Psum:buslv;
if (z > 0)
{
    Tap_No = round(z * K_Trans);
}
else
{
    Tap_No = round(z * K_Trans_Neg);
}
if (Tap_No < OT:t:ntpmn)
{
    if (Tap_No > OT:t:ntpmx)
    {
        OT:nntap = Tap_No;
    }
}
if (Tap_No > OT:t:ntpmx)
{
    OT:nntap = OT:t:ntpmx;
}
if (Tap_No < OT:t:ntpmn)  
{
    OT:nntap = OT:t:ntpmn;
}

!!!Load flow
LDF.Execute();

!=================================================================
! Creating the report in Excel
!=================================================================

! write bus names a
S = BusSet_a.AllBars();
O = S.First();
y = 1;
while (O)
{
    z = O:m:u;
    s1 = sprintf('%d%d',i+1,y);
    s3 = sprintf('%f',z);
    ddePoke(s1,s3);
    y = y + 1;
    O = S.Next();
}

! write bus names b
S = BusSet_b.AllBars();
O = S.First();
while (O)
{
    z = O:m:u;
    s1 = sprintf('%d%d',i+1,y);
    s3 = sprintf('%f',z);
    ddePoke(s1,s3);
    y = y + 1;
    O = S.Next();
}

! write bus names c
S = BusSet_c.AllBars();
O = S.First();
while (O)
{
    z = O:m:u;
    s1 = sprintf('%d%d',i+1,y);
    s3 = sprintf('%f',z);
ddePoke(s1,s3);
y = y + 1;
O = S.Next();
}

!Feeder a transformer tap/P/Q
S = TransSet_a.All();
O = S.First();
while (O)
{
   s1 = sprintf('R%dC%d',i+1,y);
   s2 = sprintf('%f',O:nntap);
   ddePoke(s1,s2);
   y = y + 1;
   s1 = sprintf('R%dC%d',i+1,y);
   s2 = sprintf('%f',O:m:Psum:buslv);
   ddePoke(s1,s2);
   y = y + 1;
   s1 = sprintf('R%dC%d',i+1,y);
   s2 = sprintf('%f',O:m:Qsum:buslv);
   ddePoke(s1,s2);
   y = y + 1;
   O = S.Next();
}

!Feeder b transformer tap/P/Q
S = TransSet_b.All();
O = S.First();
while (O)
{
   s1 = sprintf('R%dC%d',i+1,y);
   s2 = sprintf('%f',O:nntap);
   ddePoke(s1,s2);
   y = y + 1;
   s1 = sprintf('R%dC%d',i+1,y);
   s2 = sprintf('%f',O:m:Psum:buslv);
   ddePoke(s1,s2);
   y = y + 1;
   s1 = sprintf('R%dC%d',i+1,y);
   s2 = sprintf('%f',O:m:Qsum:buslv);
   ddePoke(s1,s2);
   y = y + 1;
   O = S.Next();
}

!Feeder c transformer tap/P/Q
S = TransSet_c.All();
O = S.First();
while (O)
{
    s1 = sprintf('R%dC%d',i+1,y);
    s2 = sprintf('%f',O:rtap);
    ddePoke(s1,s2);
    y = y + 1;
    s1 = sprintf('R%dC%d',i+1,y);
    s2 = sprintf('%f',O:m:sum:buslv);
    ddePoke(s1,s2);
    y = y + 1;
    s1 = sprintf('R%dC%d',i+1,y);
    s2 = sprintf('%f',O:m:Qsum:buslv);
    ddePoke(s1,s2);
    y = y + 1;
    O = S.Next();
}

!Feeder c P/Q Losses
s1 = sprintf('R%dC%d',i+1,y);
    s2 = sprintf('%f',GridSum_c:c:LossP);
    ddePoke(s1,s2);
    y = y + 1;
    s1 = sprintf('R%dC%d',i+1,y);
    s2 = sprintf('%f',GridSum_c:c:LossQ);
    ddePoke(s1,s2);
    y = y + 1;
}
Appendix C

DSP SOURCE CODE FOR THE EXPERIMENTAL SYSTEM

This appendix provides the TMS320F2810 DSP C source code based on that developed Creative Power Technologies [103]. The background operation code is within the main.c and main.h files; while the state machine and the main control interrupt routine is included in vsi.c and vsi.h.

B.1. main.h

/**
 * file
 * brief NY5kW_Active Rectifier software definitions
 *
 * Developed By:
 * Creative Power Technologies, (C) Copyright 2010
 * author A.McIver
 * History:
 * 1 01/05/07 DGH - derived from ele2.5kva\code\latest\main.h
 * 1 25/07/07 AM - added fault definitions
 * 1 09/10/07 AM - updated documentation
 * 1 08/04/08 PM - 30kW battery Charger modification - ported from 25kVA Boost
 * 1 16/04/09 PM - Release as V1.05
 * 1 21/01/10 PM - Added AUTO_START_DELAY definition
 * 1 02/02/10 PM - Started Port to new version of 30kW BC - 30kW2
 * 1 24/08/10 PM - System release for internal testing V1.00
 * 1 03/11/10 PM - Production version released V1.01
 * 1 17/03/11 PM - Ported from 30kW BC2 for use in NY 5kW Active Rectifier
 * 1 15/05/12 PM - Cleanned up comments
 * 1 26/05/12 SP - Begin conversion to LCL active rectifier
 */

/* The following text is for the documentation */
/**
 * mainpage Nanyang 5kW Active Rectifier Introduction
 */

The Nanyang 5kW Active Rectifier takes a nominal 400V 3ph AC input and produces a nominal 700V DC bus. The input stage is an active rectifier and the DC output connects to a separately DSP controlled a bi-directional phase shifted square wave DC-DC converter. Both stages have the same switching frequency. This section describes the DSP

159
Appendix C  DSP Source Code for the Experimental System

Controller and Power Stage Hardware for the Active Rectifier.

\section{Hardware Hardware}

The Power Stage synchronizes to the incoming AC supply of nominally 400Vac and rectifies it to produce 700V. In the bidirectional Battery Charger the DC bus is then fed through an H bridge using phase shifted square waves to a high frequency transformer. The output of the transformer is rectified and filtered to produce a nominal DC output (340 - 440VDC).

The Fuel Cell Inverter is uni-directional and steps up a nominal 55-90VDC input to produce a nominal 600-700VDC bus that is connected to the grid via the active rectifier/inverter stage.

\section{Communications Communications}

The VSI DSP controller has a 485 link to the DC/DC DSP controller.

\section{Software Modules}

- PWM
- Parameter Database
- Comms from VSI to DC/DC Converter

\section{Project Level Definitions}

The following definitions are needed in the project file.

\begin{verbatim}
# define SYSCLK_OUT (150e6)
# define HSPCLK (SYSCLK_OUT)
# define LSPCLK (SYSCLK_OUT/4)

/// boot ROM sine table size
# define TABLE_SIZE 512
/// boot ROM sine table magnitude
# define MAX_SINE_TABLE 16384

# define SW_FREQ 10000.0/5000.0 // switching freq in Hz
# define F_FREQ 10000.0

# define VHI_NOM 650.0 // target hv dc bus Vdc
# define IAC_NOM 7.5 // rated AC current in Arms

// Loop Gains - Cf=15uF, or Cf=7.5810kHz
# define KDM_INNER 14 //27 // [V/A]
# define KCM_INNER  20//39.6 // [V/A]
# define KP_OUTER   0.92 //0.36//0.711 // [A/A]
# define TR_OUTER 0.0021 //0.0021 //0.0042 // [s] or 0.0021@10kHz?
\end{verbatim}
Appendix C   DSP Source Code for the Experimental System

// Loop Gains - Cf=1.5uF
//#define K_INNER  41.9  // [V/A]
//#define KP_OUTER 1  // [A/A]
//#define TR_OUTER 0.0019  // [s]

// DC Bus control gains
#define KP_VHI 1
#define TINT_VHI (0.05)

/* =========================================================================
__Fault_Definitions()
========================================================================= */
/** @name Inverter fault codes */
//@{
// (mirrored in para.h:Active Rectifier fault word bits)
#define FAULT_PDPINT 0x00000001 ///< hardware gate fault trip - act rect
#define FAULT_HW_AC_OC 0x00000004 ///< hardware AC over current trip
#define FAULT_A_PH_FUSE 0x00000008 ///< Input A Phase Fuse Failure
#define FAULT_B_PH_FUSE 0x00000010 ///< Input B Phase Fuse Failure
#define FAULT_C_PH_FUSE 0x00000020 ///< Input C Phase Fuse Failure
#define FAULT_OT 0x00000040 ///< hardware over temperature trip
#define FAULT_SW_VHI_OV 0x00000080 ///< unused
#define FAULT_HW_VHI_OV 0x00000200 ///< hardware Vdc over voltage trip
#define FAULT_SW_UVIN 0x00000400 ///< software Vac under voltage input trip
#define FAULT_CHARGE 0x00001000 ///< charging fault
#define FAULT_EMERG 0x00002000 ///< emergency stop button pressed
#define FAULT_VDC_BUS 0x00004000 ///< voltage imbalance across DC bus
#define FAULT_CONT 0x00008000 ///< contactor failure
#define FAULT_COMMS 0x00010000 ///< NOT SET used by display board
//@}
#define FAULT_MAX 16 // includes all faults to FAULT_COMMS
#define FAULT_SPARES (0)

// faults that prevent converter operation
#define FAULT_FATAL  (0xFFFF & ~(FAULT_COMMS|FAULT_SPARES))
// faults that are checked in particular states
#define FAULT_ST_VIN_UV  (0xFFFF & ~(FAULT_SW_UVIN|FAULT_COMMS|FAULT_SPARES))

extern Uint16 detected_faults;
extern Uint16 fault_gate_flag;

/// Individual fault setting function
void main_fault_set(Uint16 new_fault);

/// individual fault clearing function
void main_fault_clear(Uint16 cleared_fault);

/// overall fault clearing function
void main_fault_clear_all(void);

/// checks whether faults have cleared
Uint16 main_fault_get_reported(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** fn inline Uint16 main_fault_get(void)
 \brief Returns the fault word. This function returns the fault word. It is inline for efficient use at the interrupt level.*/

161
Appendix C   DSP Source Code for the Experimental System

\author A.McIver
\par History:
\li 19/07/07 AM - initial creation
\li 24/02/10 PM - Modified to be 32 bit
\returns the fault word */
inline Uint16 main_fault_get(void) {
    return detected_faults;
} /* end main_fault_get */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
* inline void main_fault_set_int(Uint16 new_fault)
* brief Sets bits in the fault word.
* This function sets one or more bits in the fault word. It is inline for
* efficient use at the interrupt level and must only be used at the interrupt
* level.
*
\author A.McIver
\par History:
\li 25/07/07 AM - initial creation
\li 24/02/10 PM - Modified to be 32 bit
\returns the fault word */
inline void main_fault_set_int(Uint16 new_fault) {
    detected_faults |= new_fault;
} /* end main_fault_set_int */

/* =========================================================================
__State_Machine_Definitions()
========================================================================= */
/** @name State Machine Definitions */
//@{
/// State function type
typedef void (* funcPtr)(void);

/// Simple state machine type
typedef struct {
    funcPtr f; ///< Pointer to present state function
    int first; ///< Flag set for first execution of present state function
} State_Type;

/* State Handling Macros */
#define NEXT_STATE(_s_,_f_) { _s_.f = (funcPtr)&_f_; 
                          _s_.first = 1; }
#define IS_FIRST_STATE(_s_) (_s_.first == 1)
#define DONE_FIRST_STATE(_s_) (IS_CURRENT_STATE(_s_, (funcPtr)&_f_))
#define IS_CURRENT_STATE(_s_, _f_) (_s_.f == (funcPtr)&_f_)
//@}

/* =========================================================================
__Watchdog_Timer_definitions()
============================================================================ */
/** @name Watchdog Timer Definitions */
//@{
// fast 1 msec watchdogs
#define WD_TIMER_MAX 10
#define WD_CHARGE 0

162
Appendix C  DSP Source Code for the Experimental System

#define WD_RAMP 1
#define WD_C485 2
#define WD_COMMs_Dv 3
#define WD_BIOs0 4
#define WD_BIOs1 5
#define WD_RESET 6
#define WD_COMMs_Vd 7
#define WD_Vd_Heart 8
#define WD_C485_0 9

extern Uint16
wd_timer[WD_TIMER_MAX]; // watchdog countdown timers
//@}

/**
 _Exported_Variables()
 */

/**
 _Exported_Prototypes()
 */

#include "grab.h"

// Sets and clears bits in a parameter word
void main_para_bits(Uint16 ref, Uint16 set, Uint16 clear);

/**
 _Debug_Test_Pins()
 */

// uses pins on digital I/O connector X13
/*
#define DIGIO5_SET()
GpioDataRegs.GPBSET.all = DIGIO5
#define DIGIO5_CLEAR()
GpioDataRegs.GPBCLEAR.all = DIGIO5
#define DIGIO6_SET()
GpioDataRegs.GPBSET.all = DIGIO6
#define DIGIO6_CLEAR()
GpioDataRegs.GPBCLEAR.all = DIGIO6
*/

/**
 _Grab_Code_Definitions()
 */

//grab type
#define GRAB_SHORT 1
#define GRAB_LONG 1
#define GRAB_DOUBLE 1

// grab array size
#define GRAB_LENGTH 500
#define GRAB_WIDTH 6
#define GRAB_DEC 0

// modes
#define GRAB_GO 0 // logging data and waiting for trigger to stop
#define GRAB_IDLE 1 // waiting for a start signal
#define GRAB_TRIG 2 // waiting for a trigger point
#define GRAB_STOPPED 3 // finished logging data
#define GRAB_SHOW 4 // showing logged data

// macros
// starts waiting for a trigger
#define GrabStart() { grab_index = 0; grab_mode = GRAB_TRIG; }

// goes from triggered to running
#define GrabRun() { grab_index = 0; grab_mode = GRAB_GO; }

// forces a running grab to stop
#define GrabStop() grab_mode = GRAB_STOPPED;

163
Appendix C  DSP Source Code for the Experimental System

```c
#define GrabClear() { grab_index = 0; grab_mode = GRAB_IDLE; }

// status tests
#define GrabRunning() (grab_mode == GRAB_GO)
#define GrabStopped() (grab_mode == GRAB_STOPPED)
#define GrabAvail() (grab_mode >= GRAB_STOPPED)
#define GrabTriggered() (grab_mode == GRAB_TRIG)

// log data macro
#define GrabStore(_loc_,_data_) \
    grab_array[grab_index][_loc_] = _data_; 

// wrap around storage
#define GrabStep() { grab_index++; \
    if (grab_index >= GRAB_LENGTH) \
        grab_mode = GRAB_STOPPED; \
}

// variables
extern int16 grab_mode,
    grab_index;
extern Uint16 grab_dec;

#ifdefined GRAB_SHORT
    extern short grab_array[GRAB_LENGTH][GRAB_WIDTH];
#endif

#ifdefined GRAB_LONG
    extern long grab_array[GRAB_LENGTH][GRAB_WIDTH];
#endif

#ifdefined GRAB_DOUBLE
    extern double grab_array[GRAB_LENGTH][GRAB_WIDTH];
#endif

// functions
void GrabDisplay(void);
void GrabInit(void);

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

B.2. main.c

/**
\file
\brief ny-5kW Active Rectifier software using the CPT-DA2810 and CPT-E13
fs=10kHz

The 30kW battery charger system uses the CPT-E01 and CPT-E10 PCBs. It uses a
three phase VSI that feeds a 400Hz output into a rectifier to produce a
controlled dc output voltage and current.

The communications to the display board are via X27 (COM1 RS485). COM0 is used
as a debug RS232 TTL port.

164
Appendix C  DSP Source Code for the Experimental System

\par Developed By:
Creative Power Technologies, (C) Copyright 2009
\author A.McIver
\author G.Holmes

\par History:
\li 01/05/07 DGH - derived from ele2.5kva\code\latest\main.c
\li 28/03/07 AM - added version numbering
\li 29/06/07 AM - added comms
\li 13/09/07 AM - converted version numbering to system wide version
\li 09/10/07 AM - updated documentation
\li 14/11/07 PM - V1.01 software generated - due to VSI current scaling change
\li 07/12/07 PM - tweaked fault restart times
\li 07/12/07 AM - V1.02 candidate software generated - not passed
\li 07/12/07 PM - Modified software to include reverse polarity fault detection
\li 08/12/07 PM/DGH - Modified fault order for LED reporting - V1.02 software
\li 07/04/08 PM - Ported from 25kVA boost code
\li 11/04/08 PM - Modified for the 30kW battery charger
\li 16/04/09 PM - Release as V1.05
\li 10/11/09 PM - Changed P_VERSION to P_VERSION_E01
\li 02/02/10 PM - Started Port to new version of 30kW BC - 30kW2
\li 25/08/10 PM - changed include file for lib_e01 to new structure
\li 03/11/10 PM - Added Creative as manufacturer in display - V1.01 software
\li 17/03/11 PM - Ported from 30kW BC2 for use in NY 5kW Active Rectifier
\li 09/12/11 DS - Commissioning NY 5kW Active Rectifier
\li 15/05/12 PM - Cleaned up Comments
\li 26/05/12 SP - Begin conversion to LCL active rectifier
\li 27/06/14 RK - negative sequence added
*/

const char flash_id[] = "\n\nRoozbeh's 5kVA LCL VSI @10kHz\n";

// include compiler standard include files
#include <math.h>
//#include <stdio.h>

// processor standard include files
#include <DSP281x_Device.h>
#include <DSP281x_Examples.h>
#include <bios0.h>
#include "bios1.h"

// board standard include files
//#include <lib_e13.h>
#include "lib_e13_ny_5kW_vsi.h"
#include <sf_at45.h>

// common project include files

// local include files
#include "conio.h"
#include "main.h"
#include "mb_para.h"
#include "modbus.h"
#include "para.h"
#include "para_v.h"
#include "panel_led.h"
#include "vsi.h"

/*
  \_Definitions()
* /
#define RESET_TIME  2000 //msec

extern const char
*build_timestamp;

165
/* __Typedefs() */

/// Time related flag type
/** This structure holds flags used in background timing. */
typedef struct {
  Uint16 msec:1, ///< millisecond flag
  sec0_1:1, //@/ tenth of a second flag
  sec:1; //@/ second flag
} type_flag;

/// Time type
/** This structure holds the variables used in background timing. */
typedef struct {
  type_flag flag; //@/ bitwise flag structure
  Uint16 count_msec; //@/ count of milliseconds since last second event
} type_time;

#ifndef BUILD_RAM
// These are defined by the linker (see build_flash.cmd)
extern Uint16 RamfuncsLoadStart;
extern Uint16 RamfuncsLoadEnd;
extern Uint16 RamfuncsRunStart;
#endif

// Background variables
Uint16 quit = 0, //@/ exit flag
i = 0,
disp_pi_data = 0; //@/ flag set to display the PI loop data

/// timing variables
type_time time = {
  0,0,0,
  0
};

// fault variables
Uint16
  detected_faults = 0, //@/ bits set for faults detected
  restarting_faults = 0, //@/ bits set for faults that are in restart zone
  known_faults = 0; //@/ bits set for faults already being processed

Uint16
  fault_gate_flag = 0, //@/ set when the fault source is a gate fault
  fault_lockout_flag = 0, //@/ flag set to lockout inverter
Appendix C  DSP Source Code for the Experimental System

/// Restarting faults have times that are counting down to zero. When they
/// reach zero they are removed from restarting_faults.
fault_restart_timer[FAULT_MAX], // system restart countdown timers
fault_count[FAULT_MAX], // count of fault occurrences for lockout
fault_count_timer[FAULT_MAX]; // lockout count clear countdown timers

const Uint16
fault_count_time[FAULT_MAX] = // time before the fault count is reset
{ 300, // FAULT_PDPINT
  150, // FAULT_SW_AC_OC
  150, // FAULT_HW_AC_OC
  300, // FAULT_A_PH_LOSS
    300, // FAULT_B_PH_LOSS
    300, // FAULT_C_PH_LOSS
   0,   // FAULT_OT
  60,  // FAULT_SW_VHI_OV
  60,  // FAULT_HW_VHI_OV
  60,  // FAULT_SW_OVIN
  0,   // FAULT_SW_UVIN
 300, // FAULT_CHARGE
 300, // FAULT_EMERG
 300, // FAULT_VDC_BUS
 300, // FAULT_CONT
  0,   // FAULT_COMMS
},
fault_count_lockout[FAULT_MAX] = // number of faults to cause a lockout
{ 2, // FAULT_PDPINT
  3, // FAULT_SW_AC_OC
  3, // FAULT_HW_AC_OC
  3, // FAULT_A_PH_LOSS
    3, // FAULT_B_PH_LOSS
    3, // FAULT_C_PH_LOSS
  3, // FAULT_OT
  3, // FAULT_SW_VHI_OV
  2, // FAULT_HW_VHI_OV
  2, // FAULT_SW_OVIN
  0, // FAULT_SW_UVIN
  2, // FAULT_CHARGE
  0, // FAULT_EMERG
  3, // FAULT_VDC_BUS
  3, // FAULT_CONT
  0, // FAULT_COMMS
},
fault_restart_time[FAULT_MAX] = // time to restart after the fault
{ 30, // FAULT_PDPINT
  30, // FAULT_SW_AC_OC
  30, // FAULT_HW_AC_OC
  30, // FAULT_A_PH_LOSS
    30, // FAULT_B_PH_LOSS
    30, // FAULT_C_PH_LOSS
  30, // FAULT_OT
  20, // FAULT_SW_VHI_OV
  30, // FAULT_HW_VHI_OV
  2,  // FAULT_SW_OVIN
  2, // FAULT_SW_UVIN
 120, // FAULT_CHARGE
 10, // FAULT_EMERG
 30, // FAULT_VDC_BUS
 30, // FAULT_CONT
  1, // FAULT_COMMS
};

Uint16
wd_timer[WD_TIMER_MAX] =
{
Appendix C  DSP Source Code for the Experimental System

0,0,0,0,0,0,0,0,0
}; /// watchdog countdown timers

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// grab code variables
int16
  grab_mode = GRAB_IDLE,
  grab_index;

Uint16
  grab_dec; // grab decimation counter

#ifdef GRAB_SHORT
  short
    grab_array[GRAB_LENGTH][GRAB_WIDTH];
#else
  double
    grab_array[GRAB_LENGTH][GRAB_WIDTH];
#endif

#ifdef GRAB_LONG
  long
    grab_array[GRAB_LENGTH][GRAB_WIDTH];
#endif

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// test code variables
Uint16
  ph Ref = 0; // internal reference for parameter to hack
int16
  ph_val = 0; // parameter value to hack

/*********
serial OL modulation command
**********/
double
  mod_serial=0.0;

/*********
serial CL Current command
**********/
double
  iq mag=0.0;

/* _Local_Function_Prototypes()

  * ****************************************************** */

// Initialises the fault handling system
void main_fault_init(void);

// Processes the fault event counting and clearing process
void main_fault_process(void);

// Resets the fault handling after a fault lockout
void main_fault_reset(void);

// process command from display
void process_command(Uint16 com);

// Processes data written from the display
void process_comms_data(void);

// perform a factory reset on the parameter database
void reset_para(void);

// store latest parameter values in the parameter database for display
void store_para(void);

// display operating info
void com_display(Uint16 mode);

// process keyboard input
void com_keyboard(void);

// 1 second interrupt for display
interrupt void isr_cpu_timer0(void);

// Function defined in bios0i_485.c
// Supervisory protection to force TX CTRL off.
void bios_watchdog_COM0(void);

// Function defined in bios1i_485.c
// Supervisory protection to force TX CTRL off.
void bios_watchdog_COM1(void);

// Hack into the parameter flash for testing ************
void para_hack(Uint16 type, Uint16 ref, int16 val);

/* =========================================================================
 * __Main
 * =========================================================================

This is the main function. It:
\l\1 Initialise the DSP and its peripherals
\l\1 Initialise the E01 PCB into a safe condition
\l\1 Copies the RAM based functions to RAM
\l\1 Starts the 1ms timers for background timing
\l\1 Sets up the two com ports
\l\1 Initialises the various software modules
\l\1 Runs the background loop

\author A.McIver
\par History:
\li 09/10/07 AM - initial creation
\li 29/04/08 AM - added fault system initialisation
\li 27/01/10 PM - Added watchdog handling
\li 09/07/14 PM - Added negative sequence
*/

void main(void)
{
    DINT;

    // Initialise DSP for PCB
    lib_e13_init(150/*MHz*/,30000/*kHz*/,150000/*kHz*/);

    // Initialize the PIE control registers to their default state.
    InitPieCtrl();

    // Disable CPU interrupts and clear all CPU interrupt flags:
    IER = 0x0000;
    IFR = 0x0000;

    // Initialize the PIE vector table with pointers to the shell Interrupt
    // Service Routines (ISR).
    // This will populate the entire table, even if the interrupt
    // is not used in this example. This is useful for debug purposes.
    // The shell ISR routines are found in DSP281x_DefaultIsr.c.
    // This function is found in DSP281x_PieVect.c.
    InitPieVectTable();

#ifndef BUILD_RAM
    // Copy time critical code and Flash setup code to RAM
    // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
    // symbols are created by the linker. Refer to the F2810.cmd file.
    MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
#endif
Appendix C  DSP Source Code for the Experimental System

// Call Flash Initialization to setup flash waitstates
// This function must reside in RAM
InitFlash();
#endif
InitAdc();
InitCpuTimers();
bios_init_COM0(38400L);
bios_init_COM1(38400L,8,PARITY_NONE,STOP_TWO);

// Configure CPU-Timer 0 to interrupt every millisecond:
// 150MHz CPU Freq, 0.001 second Period (in uSeconds)
ConfigCpuTimer(&CpuTimer0, 150.0/*MHz*/, 1000.0/*us*/);
StartCpuTimer0();

// Map interrupt to interrupt service routine.
EALLOW;
PieVectTable.TINT0 = &isr_cpu_timer0;
EDIS;

// Enable TINT0 in the PIE: Group 1 interrupt 7
PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
IER |= M_INT1; // Enable CPU Interrupt 1
EnableInterrupts();

// Used for test pins
CLEAR_LED0_E13();
CLEAR_LED1_E13();
CLEAR_LED2_E13();
CLEAR_LED3_E13();
puts_COM0(flash_id);
/* Display Version Number information */
puts_COM0(build_timestamp);
puts_COM0("n");
pled_init();
// Initialise flash RAM
sf_init();
sf_set_protect(SF_UNPROTECT);
GrabInit(); /* ***** Grab code should be commented out ******/
if (para_init(PARA_INIT_NODEFAULT) != 0) // bad settings
{
    Uint16 status;
    status = PARA_READ(P_STATUS);
    status |= ST_BAD_SETTINGS;
    para_write_int(P_STATUS,status);
}
puts_COM0("\nMB Addr:");
putd_COM0(PARA_READ(P_ADDR));
puts_COM0("n");
modbus_init(PARA_READ(P_ADDR));
vs_set_mode(VSI_CI);
vs_set_id(0.0);
vs_set_id_n(0.0);
vs_set_iq(0.0);
vs_set_iq_n(0.0);
main_fault_init();
mb_para_init();
// Enable watchdog
EALLOW;
void main_loop(void)
 */
 while(quit == 0)
 {  
 if (time.flag.msec != 0) // millisecond events
 {  
     time.flag.msec = 0;
     vsi_state_machine();
     bios_watchdog_COM0();
     bios_watchdog_COM1();
     modbus_process();
     KickDog();
     reset_para();
 }
 else if (time.flag.sec0_1 != 0) // tenth of second events
 {  
     time.flag.sec0_1 = 0;
     store_para();
     com_keyboard();
     com_display(0); // continue one second display
 }
 else if (time.flag.sec != 0) // one second events
 {  
     time.flag.sec = 0;
     main_fault_process();
     com_display(1); // trigger new one second display
 }
 else // low priority events
 {  
     if (mb_para_length() != 0)
     {  
         process_comms_data();
     }
 }
 } /* end while quit == 0 */

SOFT_CHARGE_RELAY_OFF();
MAIN_CONTACTOR_OFF();
EvaRegs.T1CON.bit.TENABLE = 0;
EvaRegs.ACTRA.all = 0x0000;
EvbRegs.T3CON.bit.TENABLE = 0;
EvbRegs.ACTFB.all = 0x0000;
DINT;
} /* end main */

/**  
 _Exported_Functions()  
============================================================================

Some parameter words are bits fields. This function encapsulates the process  
of setting and clearing bits in these parameters. There is no error checking  
on the validity of the function arguments.

The clear bits are cleared before the set bits are set.

author A.McIver
par History:
li 04/07/07 AM - initial creation
*/
Appendix C  DSP Source Code for the Experimental System

\param[in] ref The parameter reference
\param[in] set The bits to set in the word
\param[in] clear The bits to clear in the word
*/
void main_para_bits(Uint16 ref, Uint16 set, Uint16 clear)
{
    int16
    word;
    para_read(ref, &word);
    word &= ~clear;
    word |= set;
    para_write_int(ref, word);
} /* end main_para_bits */

/**
This function is called when a fault is detected to set its bit in the
fault word. Since this function is called from both background and interrupt
is has to be an atomic operation.

\author A. McIver
\par History:
\li 19/07/07 AM - initial creation
\li 24/04/10 PM - Modified to be 32 bit

\param[in] new_fault The new fault to be added to the fault word.
*/
void main_fault_set(Uint16 new_fault)
{
    DINT;
    detected_faults |= new_fault;
    EINT;
} /* end main_fault_set */

/**
This function is used to remove an individual fault from the fault word. Since
the fault word can be altered at the interrupt level, this function needs to
be atomic.

\author A. McIver
\par History:
\li 19/07/07 AM - initial creation
\li 24/02/10 PM - Modified to be 32 bit

\param[in] cleared_fault The fault to be cleared from the fault word.
*/
void main_fault_clear(Uint16 cleared_fault)
{
    DINT;
    detected_faults &= ~cleared_fault;
    EINT;
} /* end main_fault_clear */

/**
This function tries to clear all the faults. It clears the fault word, then
checks whether the faults are still present. Since the fault word can be
altered at the interrupt level, this function uses the atomic main_fault_set
call to set bits.

\author A. McIver
\par History:
\li 19/07/07 AM - initial creation
\li 01/05/08 AM - reduced time that interrupts are disabled
void main_fault_clear_all(void) {
    Uint16 i = 0;
    DINT;
    if (fault_gate_flag != 0) {
        RESET_GATES();
        detected_faults &= ~FAULT_PDPINT;
        EvaRegs.COMCONA.bit.FCOMPOE = 1; // full compare enable
        for (fault_gate_flag=0; fault_gate_flag<1000; fault_gate_flag++)
        {
            i += fault_gate_flag;
        }
        fault_gate_flag = 0;
        ENABLE_GATES();
        for (i=0; i<10; i++)
        {
            ENABLE_GATES();
        }
    }
    EINT;
    if (EvaRegs.COMCONA.bit.PDPINTASTATUS == 0)
    {
        main_fault_set(FAULT_PDPINT);
    }
    if (GET_I_OV_TRIP())
    {
        main_fault_set(FAULT_HW_AC_OC);
    }
} /* end main_fault_clear_all */

Uint16 main_fault_get_reported(void)
{
    return (known_faults|detected_faults);
} /* end main_fault_get_reported */

Local_Functions()

This function is used to check whether the faults have cleared so that the inverter can restart.

author A.McIver
par History:
li 27/07/07 AM - initial creation
li 24/02/10 PM - Modified to be 32 bit

returns Known and detected faults */

This function performs a factory reset on the parameter database. It uses the reset watchdog timer to control the timing of resetting all the parameters to their default values. At the end of the process the DSP is reset.
void reset_para(void)
{
    if (wd_timer[WD_RESET] == 0)
    {
        return;
    }
    switch (wd_timer[WD_RESET])
    {
        case RESET_TIME - 100:
            para_write_int(P_ADDR, P_ADDR_DEF);
            break;
        case RESET_TIME - 200:
            para_write_int(P_VHV_SET, P_VHV_SET_DEF);
            break;
        case RESET_TIME - 300:
            para_write_int(P_IQ_SET, P_IQ_SET_DEF);
            break;
        case 1:
            EALLOW;
            SysCtrlRegs.WDKEY = 0x0000; // resets DSP
            EDIS;
            break;
        default:
            // do nothing
            }
} /* end reset_para */

void process_comms_data(void)
{
    Uint16
    res,
    ref;

    res = mb_para_get_write(&ref);
    if (res != 0) // queue is empty
    {
        return;
    }
    switch (ref)
    {
        case P_COMMAND:
            process_command(PARA_READ(P_COMMAND));
            break;
        case P_ADDR:
            modbus_init(PARA_READ(P_ADDR));
            break;
        case P_VHV_SET:
            vsi_set_vhi(PARA_READ(P_VHV_SET));
            break;
        case P_ID_SET:
            {
                vsi_set_id(PARA_READ(P_ID_SET));
            }
            break;
    } /* end process_comms_data */
default:
    puts_COM0("\nBad comms write:");
    putu(ref);
    puts_COM0(" ignoring\n");
}
} /* end process_comms_data */


Any commands received from the system controller are handled here. The available commands are defined in para.h.

\author A.McIver
\par History:
\li 06/07/07 AM - derived from vsi:main:process_command
\li 27/01/10 PM - Added DB_ACK_FLASH and DB_FACTORY_RESET commands
\par
\param[in] com Command from display
*/
void process_command(Uint16 com)
{
    switch (com)
    {
    case MB_CLEAR_FAULTS:
        main_fault_reset();
        break;
    case MB_RUN:
        vsi_enable();
        break;
    case MB_STOP:
        vsi_disable();
        break;
    case MB_ACK_FLASH:
        // acknowledgement of bad settings received
        para_write_int(P_STATUS, PARA_READ(P_STATUS)&~ST_BAD_SETTINGS);
        break;
    case MB_FACTORY_RESET:
        vsi_disable();
        // reset DSP requested at completion of factory reset
        // turn off outputs - already done at start of factory reset
        // Open contactors
        MAIN_CONTACTOR_OFF();
        SOFT_CHARGE_RELAY_OFF(); // disconnect from supply
        EvaRegs.T1CON.all = 0; // stop timer
        EvaRegs.T2CON.all = 0; // stop timer
        EvbRegs.T3CON.all = 0; // stop timer
        EvbRegs.T4CON.all = 0; // stop timer
        EvaRegs.COMCONA.all = 0x0000; // disable compare outputs
        EvbRegs.COMCONB.all = 0x0000; // disable compare outputs
        // Trigger Reset Timer - checked at msec timer level
        wd_timer[WD_RESET] = RESET_TIME; // msec
        break;
    case MB_SET_CV:
        if (vsi_get_state() == ST_VSI_STOP)
        {
            vsi_set_mode(VSI_CV);
            puts_COM0("Voltage Regulated ACREC\n");
        }
        break;
    case MB_SET_CI:
        if (vsi_get_state() == ST_VSI_STOP)
        {
            vsi_set_mode(VSI_CI);
            puts_COM0("Current Regulated VSI\n");
        }
    }
Appendix C  DSP Source Code for the Experimental System

break;
}
} /* end process_command */

/**
* Initialises the fault variables. Called at power up.
*/

void main_fault_init(void)
{
    Uint16
    i;
    for (i=0; i<FAULT_MAX; i++)
    {
        fault_count[i] = 0;
        fault_count_timer[i] = 0;
        fault_restart_timer[i] = 0;
    }
} /* end main_fault_init */

/**
* This function is called once per second. It detects any new faults and starts
a clearing timer. If the fault count for a new fault has reached the lockout
threshold, then the lockout flag is set to prevent the inverter restarting. It
also counts down the timers for any recent faults to clear the fault count.

author A.McIver
par History:
\li 27/07/07 AM - initial creation
*/
void main_fault_process(void)
{
    Uint16
    i,
    new_faults,
    cleared_faults,
    fault_mask;
    Uint16
    i,
    restart_clear_flag = 0;
    // new faults - bits set in detected faults but not in known faults
    new_faults = detected_faults&(~known_faults);
    // cleared faults - bit set in known faults, but
    cleared_faults = known_faults&(~detected_faults|restarting_faults);
    known_faults |= detected_faults;
    restarting_faults |= cleared_faults;
    // check for faults clearing
    detected_faults = vsi_check_fault();
    // process new faults
    if (new_faults != 0)
    {
        fault_mask = 0x0001;
    }
}
// scan each new fault
for (i=0; i<FAULT_MAX; i++)
{
    if (new_faults&fault_mask)
    {
        // only for lockout faults
        if (fault_count_lockout[i] != 0)
        {
            fault_count[i]++;
            // check for lockout
            if (fault_count[i] >= fault_count_lockout[i])
            {
                fault_lockout_flag = 1;
                puts_COM0("nlockout
");
            }
        }
    }
    fault_mask <<= 1;
}

// process cleared faults - start restart timer and count timer
if (cleared_faults != 0)
{
    fault_mask = 0x0001;
    // scan each new fault
    for (i=0; i<FAULT_MAX; i++)
    {
        if (cleared_faults&fault_mask)
        {
            fault_count_timer[i] = fault_count_time[i];
            fault_restart_timer[i] = fault_restart_time[i];
        }
        fault_mask <<= 1;
    }
}

// process existing faults
for (i=0; i<FAULT_MAX; i++)
{
    if (fault_count_timer[i] > 0)
    {
        fault_count_timer[i]--;
        // check for clear timer finish
        if (fault_count_timer[i] == 0)
        {
            // dec fault count and restart count timer
            if (fault_count[i] > 0)
            {
                fault_count[i]--;
                fault_count_timer[i] = fault_count_time[i];
            }
        }
    }
}

fault_mask = 0x0001;
restart_clear_flag = 0;
for (i=0; i<FAULT_MAX; i++)
{
    if (fault_restart_timer[i] > 0)
    {
        fault_restart_timer[i]--;
        // check for clear timer finish unless locked out
        if ( (fault_restart_timer[i] == 0)&&(fault_lockout_flag == 0) )
        {
            // remove from known faults and restarting faults
            known_faults &= ~fault_mask;
            restarting_faults &= ~fault_mask;
            restart_clear_flag = 1;
        }
    }
}
fault_mask <<= 1;
}
// check for restart timer finish
if ( (detected_faults == 0) && (restarting_faults == 0)
    && (restart_clear_flag == 1) )
{
    // clear faults
    main_fault_clear_all();
}
/* end main_fault_process */

/ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** This function is called to restart the inverter after a lockout event. It clears the lockout flag, resets all the fault counts and timers and then clears all the faults. */

\author A.McIver
\par History:
\li 27/07/07 AM - initial creation
\li 29/04/08 AM - reset restarting_faults as well */
void main_fault_reset(void)
{
    Uint16
    i;
    fault_lockout_flag = 0;
    for (i=0; i<FAULT_MAX; i++)
    {
        fault_count[i] = 0;
        fault_count_timer[i] = 0;
        fault_restart_timer[i] = 0;
    }
    restarting_faults = 0x0000;
    known_faults = 0x0000;
    main_fault_clear_all();
} /* end main_fault_reset */

/ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/** The display accesses system parameters by reading the data from the parameter database. This function updates the database with the latest measurements. */

\author A.McIver
\par History:
\li 25/05/07 AM - initial creation
\li 08/08/07 AM - changed power calculation to tenths of kW
\li 14/09/07 AM - fixed setting of LED voltage status bits for normal case
\li 01/12/08 AM - added calibration offset to Vout */
Uint16
    fault_new;
Uint16
    status, p_real, p_react;
int16
    vsi_status, vac_grid, volts_in, amps;

void store_para(void)
{
    Uint16
    led_bits = 0;
    volts_in = vsi_get_vhi();
    para_write_int(P_VHV, volts_in);
    amps = (int16)vsi_get_iac(1);
para_write_int(P_IPH1,amps);
if (amps > (int16)(IAC_NOM*10.0))
{
    led_bits |= PLED_OLOAD;
}
amps = (int16)vsi_get_iac(2);
para_write_int(P_IPH2,amps);
if (amps > (int16)(IAC_NOM*10.0))
{
    led_bits |= PLED_OLOAD;
}
amps = (int16)vsi_get_iac(3);
para_write_int(P_IPH3,amps);
if (amps > (int16)(IAC_NOM*10.0))
{
    led_bits |= PLED_OLOAD;
}
vac_grid = (int16)vsi_get_vac(0);
para_write_int(P_VIN_AVG,vac_grid);
vac_grid = (int16)vsi_get_vac(1);
para_write_int(P_VIN_BA,vac_grid);
vac_grid = (int16)vsi_get_vac(2);
para_write_int(P_VIN_CA,vac_grid);
p_real = vsi_get_p();
para_write_int(P_POUT,(int16)p_real);
p_react = vsi_get_q();
para_write_int(P_QOUT,(int16)p_react);

// update battery charger fault word
fault_new = main_fault_get_reported();
para_write_int(P_FAULT,(int16)fault_new);

// update status word
status = PARA_READ(P_STATUS);
// battery charger status bits
if (fault_new == 0)
{
    led_bits |= PLED_AC_VOLTS;
}
else if (vsi_status > 0)
{
    status = ~ST_RUN|ST_FAULT;
    led_bits |= PLED_STOP;
} else if (vsi_status == 0)
{
    status |= ST_RUN;
    status &= ~ST_FAULT;
    led_bits |= PLED_RUN;
} else
{
    status |= ST_FAULT;
    status &= ~ST_RUN;
    led_bits |= PLED_FAULT;
}
else
{
    status |= ST_FAULT;
    status &= ~ST_RUN;
    led_bits |= PLED_FAULT;
}}
led_bits |= PLED_AC_UV;
}
else if (fault_new & FAULT_SW_OVIN)
{
    led_bits |= PLED_AC_OV;
}
}

if (modbus_comms_ok() == 0)
{
    led_bits |= PLED_COMMS;
}
para_write_int(P_STATUS, status);
pled_load(led_bits);
} /* end store_para */

void com_display(Uint16 mode)
{
    static Uint16 display_state = 0xFFFF;
    int16 fault_state = 0;
    // don't trigger until existing information has printed
    if ((mode == 1) && (display_state > HELP_FINISH))
        display_state = 0;
    else if (mode == 2) // display help parameters
        display_state = HELP_START;
    else
        display_state++;

    if (GrabAvail())
    {
        GrabDisplay();
    }
display_state = 0;
return;
}

switch (display_state)
{
    case 0:
        fault_state = vsi_get_status();
        puts_COM0("\n");
        if (fault_state == -1)
        {
            puts_COM0("F");
            putxx(main_fault_get_reported());
            puts_COM0(" ");
        }
        puts_COM0(vsi_state_str[vsi_get_state()]);
        puts_COM0(" Sync:");
        putu(in_sync);
        break;
    case 1:
        if (vsi_get_mode() == VSI_CI)
            puts_COM0(" ci");
        else if (vsi_get_mode() == VSI_OL)
            puts_COM0(" ol");
        else
            puts_COM0(" cv");
        puts_COM0(" Vh:");
        putd(vsi_get_vhi());
        puts_COM0(" Vac:");
        putd(vsi_get_vac(0));
        break;
    case 2:
        if (vsi_get_mode() == VSI_CI)
        {
            puts_COM0(" Id_r:");
            putdbl((double)PARA_READ(P_ID_SET)/10.0,2);
            puts_COM0(" Iq_r:");
            putdbl((double)PARA_READ(P_IQ_SET)/10.0,2);
            puts_COM0(" Id_N_r:");
            putdbl((double)PARA_READ(P_ID_N_SET)/10.0,2);
            puts_COM0(" Iq_N_r:");
            putdbl((double)PARA_READ(P_IQ_N_SET)/10.0,2);
        }
        else if (vsi_get_mode() == VSI_OL)
        {
            puts_COM0(" Mod_dep:");
            putdbl(mod_serial,2);
        }
        else
        {
            puts_COM0(" Vh_r:");
            putu(PARA_READ(P_VHV_SET));
            puts_COM0(" Iq_r:");
            putdbl((double)PARA_READ(P_IQ_SET)/10.0,2);
        }
        break;
    case 3:
        if (vsi_get_mode() == VSI_CI)
        {
            puts_COM0(" Id_st:");
            putdbl((double)PARA_READ(P_IDSTEP_SET)/10.0,2);
            puts_COM0(" Iq_st:");
            putdbl((double)PARA_READ(P_IQSTEP_SET)/10.0,2);
            puts.COM0(" Id_N_st:");
            putdbl((double)PARA_READ(P_IDSTEP_N_SET)/10.0,2);
            puts.COM0(" Iq_N_st:");
            putdbl((double)PARA.READ(P_IQSTEP_N_SET)/10.0,2);
else if (vsi_get_mode() == VSI_CV) {
    puts_COM0(" Iq_st:");
    putdb1((double)PARA_READ(P_IQSTEP_SET)/10.0,2);
} break;
case 4:
    if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV)) {
        puts_COM0(" St:");
        putu(vsi_get_step());
        puts_COM0(" AD:");
        putu(vsi_get_active_damping());
        puts_COM0(" HC:");
        putu(vsi_get_harmcomp());
    } if(vsi_get_mode() == VSI_OL) {
        puts_COM0(" SVC:");
        putu(vsi_get_svc());
    } break;

// Print Help Screen
    case HELP_START:
        puts_COM0("\n\ne/E - enable active rectifier switching\n");
        break;
    case HELP_START+1:
        puts_COM0("d/D - disable active rectifier switching\n");
        break;
    case HELP_START+2:
        puts_COM0("F - clear faults\n");
        break;
    case HELP_START+3:
        break;
    case HELP_START+4:
        break;
    case HELP_START+5:
        puts_COM0("\nMode Selection: \n");
        break;
    case HELP_START+6:
        puts_COM0("V - set rectifier mode to constant voltage mode\n");
        break;
    case HELP_START+7:
        puts_COM0("C - set vsi mode to constant current mode\n");
        break;
    case HELP_START+8:
        puts_COM0("X - set vsi mode to open loop mode\n");
        break;
    case HELP_START+9:
        break;
    case HELP_START+10:
        break;
    case HELP_START+11:
        puts_COM0("\nIn constant voltage mode: \n");
        break;
    case HELP_START+12:
        puts_COM0("y/b - Increment/Decrement vhi\n");
        break;
    case HELP_START+13:
        puts_COM0("y/B - Fast Increment/Decrement vhi\n");
        break;
    case HELP_START+14:
        puts_COM0("u/n - increment/decrement iq\n");
        break;
    case HELP_START+15:
        puts_COM0("U/N - fast increment/decrement iq\n");
case HELP_START+16:
    break;
case HELP_START+17:
    break;
case HELP_START+18:
    puts_COM0("nIn constant current mode:\n");
    break;
case HELP_START+19:
    puts_COM0("i/m - Increment/Decrement id\n");
    break;
case HELP_START+20:
    puts_COM0("I/M - Fast Increment/Decrement id\n");
    break;
case HELP_START+21:
    puts_COM0("u/n - increment/decrement iq\n");
    break;
case HELP_START+22:
    puts_COM0("U/N - fast increment/decrement iq\n");
    break;
case HELP_START+23:
    break;
case HELP_START+24:
    puts_COM0("nIn Open Loop Mode:\n");
    break;
case HELP_START+25:
    puts_COM0("i/m - increment/decrement modulation depth\n");
    break;
case HELP_START+26:
    puts_COM0("I/M - fast increment/decrement modulation depth\n");
    break;
case HELP_START+27:
    puts_COM0("u/n - increment/decrement Q_OUT - iq\n");
    break;
case HELP_START+28:
    puts_COM0("U/N - fast increment/decrement Q_OUT - iq\n");
    break;
case HELP_START+29:
    break;
case HELP_START+30:
    puts_COM0("n6/7 - inc/dec Vdc out Kp\n");
    break;
case HELP_START+31:
    puts_COM0("t - Main contactor Close\n");
    break;
case HELP_START+32:
    break;
case HELP_START+33:
    puts_COM0("n6/7 - inc/dec Vdc out Kp\n");
    break;
case HELP_START+34:
    puts_COM0("s - fast inc/dec Vdc out Kp\n");
    break;
case HELP_START+35:
    puts_COM0("8/9 - inc/dec Vdc out Ki\n");
    break;
case HELP_START+36:
    puts_COM0("*/ - fast inc/dec Vdc out Ki\n");
    break;
case HELP_START+37:
    puts_COM0("0 - display Vdc out PI loop parameters\n");
    break;
case HELP_START+38:
    break;
case HELP_START+39:
    puts_COM0("ng - start grab\n");
    break;
case HELP_START+40:
    puts_COM0("G - Grab Clear\n");
break;
case HELP_START+41:
    puts_COM0("a/A - dec/inc para hack ref\n");
break;
case HELP_START+42:
    puts_COM0("z/Z - dec/inc para hack val\n");
break;
case HELP_START+43:
break;
case HELP_START+44:
    puts_COM0("h/H - corrupt/rewrite parameter ph_ref\n");
b Serialization: none
Returns: nothing
Description: Process characters from COM0.
Notes:
e/E - enable active rectifier switching
d/D - disable active rectifier switching
F - clear faults
X - set vsi mode to open loop mode
V - set rectifier mode to constant voltage mode
C - set vsi mode to constant current mode

t - main contactor Open
T - Main contactor Close

In open loop mode
i/m - increment/decrement modulation depth
I/M - fast increment/decrement modulation depth

In Constant Current Mode
i/m - Increment/Decrement id
I/M - Fast Increment/Decrement id
q/w - Increment/Decrement id_N
k/l - Increment/Decrement iq_N

In Constant Voltage Mode
i/m - Increment/Decrement vhi
I/M - Fast Increment/Decrement vhi
u/n - increment/decrement _Q_OUT - iq
U/N - fast increment/decrement _Q_OUT - iq
6/7 - inc/dec Vdc out Kp
^/& - fast inc/dec Vdc out Kp
8/9 - inc/dec Vdc out Ki
* / - fast inc/dec Vdc out Ki
0 - display Vdc out PI loop parameters
g - start grab
G - stop Grab Interrupt Data
a/A - dec/inc para hack ref
z/Z - dec/inc para hack val
h - corrupt parameter ph_ref
H - rewrite parameter ph_ref with ph_val
?
- Help screen print

History:
22/06/05 AM - initial creation
29/02/12 PM - Modified for VSI NY
27/06/14 RK - Modified for negative sequence

*/
void com_keyboard(void)
{
    char c;
    if (kbhit_COM0())
    {
        c = getc_COM0();
        switch (c)
        {
        case 'E':
            case 'e':
                //mc_enable();
                vvi_set_vhi(PARA_READ(P_VHV_SET));
                vvi_set_id(PARA_READ(P_ID_SET));
                vvi_set_id_n(0.0);
                vvi_set_iq(iq_mag);
                vvi_enable();
                puts_COM0("e");
                break;
        case 'D':
            case 'd':
                vvi_disable();
                puts_COM0("d");
                break;
        case 'F':
            main_fault_reset();
            break;
        case 'X':
            vvi_set_mode(VSI_OL);
            puts_COM0("nOpen Loop VSI
\n");
            break;
        case 'C':
            vvi_set_mode(VSI_CI);
            puts_COM0("nCurrent Regulated VSI
\n");
            break;
        case 'V':
            vvi_set_mode(VSI_CV);
            puts_COM0("nVoltage Regulated ACREC\n\n");
            break;
        case 'i': // small increase
            if (vsi_get_mode() == VSI_OL)
            {
                if (mod_serial<=2.0)
                {
                    mod_serial+=0.01;
                }
                else
                {
                    mod_serial=2.0;
                }
                vvi_set_mod(mod_serial);
            }
            if (vsi_get_mode() == VSI_CI)
            {
                para_write_int(P_ID_SET, PARA_READ(P_ID_SET) + 1);
            }
vsi_set_id(PARA_READ(P_ID_SET));

if (vsi_get_mode() == VSI_CV)
{
    para_write_int(P_VHV_SET, PARA_READ(P_VHV_SET) + 1);
    vsi_set_vhi(PARA_READ(P_VHV_SET));
}
break;

case 'I': // large increase
if (vsi_get_mode() == VSI_OL)
{
    if (mod_serial<=2.0)
    {
        mod_serial+=0.05;
    }
    else
    {
        mod_serial=2.0;
    }
    vsi_set_mod(mod_serial);
}
if (vsi_get_mode() == VSI_CI)
{
    para_write_int(P_ID_SET, PARA_READ(P_ID_SET) + 10);
    vsi_set_id(PARA_READ(P_ID_SET));
}
if (vsi_get_mode() == VSI_CV)
{
    para_write_int(P_VHV_SET, PARA_READ(P_VHV_SET) + 10);
    vsi_set_vhi(PARA_READ(P_VHV_SET));
}
break;

// negative sequence added

case 'q': // small increase id_N
if (vsi_get_mode() == VSI_CI)
{
    para_write_int(P_ID_N_SET, PARA_READ(P_ID_N_SET) + 1);
    vsi_set_id_n(PARA_READ(P_ID_N_SET));
}
break;

case 'm': // small decrease
if (vsi_get_mode() == VSI_OL)
{
    if (mod_serial>=0.0)
    {
        mod_serial-=0.01;
    }
    else
    {
        mod_serial=0.0;
    }
    vsi_set_mod(mod_serial);
}
if (vsi_get_mode() == VSI_CI)
{
    para_write_int(P_ID_SET, PARA_READ(P_ID_SET) - 1);
    vsi_set_id(PARA_READ(P_ID_SET));
}
if (vsi_get_mode() == VSI_CV)
{
    para_write_int(P_VHV_SET, PARA_READ(P_VHV_SET) - 1);
    vsi_set_vhi(PARA_READ(P_VHV_SET));
}
break;
case 'M': //large decrease
    if (vsi_get_mode() == VSI_OL)
    {
        if (mod_serial>=0.0)
        {
            mod_serial-=0.05;
        }
        else
        {
            mod_serial=0.0;
        }
        vsi_set_mod(mod_serial);
    }
    if (vsi_get_mode() == VSI_CI)
    {
        if (vsi_get_mode() == VSI_CI)
        {
            para_write_int(P_ID_SET, PARA_READ(P_ID_SET) - 10);
            vsi_set_id(PARA_READ(P_ID_SET));
        }
        if (vsi_get_mode() == VSI_CV)
        {
            para_write_int(P_VHV_SET, PARA_READ(P_VHV_SET) - 10);
            vsi_set_vhi(PARA_READ(P_VHV_SET));
        }
    }
    break;

    //negative sequence added
    case 'w': //small decrease id_N
    if (vsi_get_mode() == VSI_CI)
    {
        para_write_int(P_ID_N_SET, PARA_READ(P_ID_N_SET) - 1);
        vsi_set_id_n(PARA_READ(P_ID_N_SET));
    }
    break;

    //reactive power
    case 'u': //small iq increase
    if ((vsi_get_mode() == VSI_CI)|| (vsi_get_mode() == VSI_CV))
    {
        para_write_int(P_IQ_SET, PARA_READ(P_IQ_SET) + 1);
        vsi_set_iq(PARA_READ(P_IQ_SET));
    }
    break;
    case 'U': //large iq increase
    if ((vsi_get_mode() == VSI_CI)|| (vsi_get_mode() == VSI_CV))
    {
        para_write_int(P_IQ_SET, PARA_READ(P_IQ_SET) + 10);
        vsi_set_iq(PARA_READ(P_IQ_SET));
    }
    break;

    //negative sequence added
    case 'k': //small iq_N increase
    if (vsi_get_mode() == VSI_CI)
    {
        para_write_int(P_IQ_N_SET, PARA_READ(P_IQ_N_SET) + 1);
        vsi_set_iq_n(PARA_READ(P_IQ_N_SET));
    }
    break;

    case 'n': //small iq decrease
    if ((vsi_get_mode() == VSI_CI)|| (vsi_get_mode() == VSI_CV))
    {
        para_write_int(P_IQ_SET, PARA_READ(P_IQ_SET) - 1);
        vsi_set_iq(PARA_READ(P_IQ_SET));
    }
    break;
    case 'N': //large iq decrease
if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
{
    para_write_int(P_IQ_SET, PARA_READ(P_IQ_SET) - 10);
    vsi_set_iq(PARA_READ(P_IQ_SET));
}
break;

// negative sequence added

if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
{
    para_write_int(P_IQ_N_SET, PARA_READ(P_IQ_N_SET) - 1);
    vsi_set_iq_n(PARA_READ(P_IQ_N_SET));
}
break;

// negative sequence added

if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
{
    para_write_int(P_IQSTEP_SET, PARA_READ(P_IQSTEP_SET) + 1);
    vsi_set_iqstep(PARA_READ(P_IQSTEP_SET));
}
break;

if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
{
    para_write_int(P_IQSTEP_SET, PARA_READ(P_IQSTEP_SET) + 10);
    vsi_set_iqstep(PARA_READ(P_IQSTEP_SET));
}
break;

if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
{
    para_write_int(P_IQSTEP_SET, PARA_READ(P_IQSTEP_SET) - 1);
    vsi_set_iqstep(PARA_READ(P_IQSTEP_SET));
}
break;

if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
{
    para_write_int(P_IQSTEP_SET, PARA_READ(P_IQSTEP_SET) - 10);
    vsi_set_iqstep(PARA_READ(P_IQSTEP_SET));
}
break;

// negative sequence added

if (vsi_get_mode() == VSI_CI)
{
    para_write_int(P_IQSTEP_N_SET, PARA_READ(P_IQSTEP_N_SET) + 1);
    vsi_set_iqstep_n(PARA_READ(P_IQSTEP_N_SET));
}
break;

if (vsi_get_mode() == VSI_CI)
{
    para_write_int(P_IQSTEP_N_SET, PARA_READ(P_IQSTEP_N_SET) - 1);
    vsi_set_iqstep_n(PARA_READ(P_IQSTEP_N_SET));
}
break;

// real current step

if (vsi_get_mode() == VSI_CI)
{
    para_write_int(P_IDSTEP_SET, PARA_READ(P_IDSTEP_SET) + 1);
    vsi_set_idstep(PARA_READ(P_IDSTEP_SET));
}
break;
case 'P': // large id step increase
    if (vsi_get_mode() == VSI_CI)
    {
        para_write_int(P_IDSTEP_SET, PARA_READ(P_IDSTEP_SET) + 10);
        vsi_set_idstep(PARA_READ(P_IDSTEP_SET));
    }
    break;
case ',':// small id step decrease
    if (vsi_get_mode() == VSI_CI)
    {
        para_write_int(P_IDSTEP_SET, PARA_READ(P_IDSTEP_SET) - 1);
        vsi_set_idstep(PARA_READ(P_IDSTEP_SET));
    }
    break;
case '>': // large id step decrease
    if (vsi_get_mode() == VSI_CI)
    {
        para_write_int(P_IDSTEP_SET, PARA_READ(P_IDSTEP_SET) - 10);
        vsi_set_idstep(PARA_READ(P_IDSTEP_SET));
    }
    break;
    // negative sequence added
    case 'j': // small id_N step increase
        if (vsi_get_mode() == VSI_CI)
    {
        para_write_int(P_IDSTEP_N_SET, PARA_READ(P_IDSTEP_N_SET) + 1);
        vsi_set_idstep_n(PARA_READ(P_IDSTEP_N_SET));
    }
    break;
case 's': // small id_N step decrease
    if (vsi_get_mode() == VSI_CI)
    {
        para_write_int(P_IDSTEP_N_SET, PARA_READ(P_IDSTEP_N_SET) - 1);
        vsi_set_idstep_n(PARA_READ(P_IDSTEP_N_SET));
    }
    break;
    // Enable/Disable stepping
    case '+':
        if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
            vsi_set_step(1);
        break;
    case '-':
        if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
            vsi_set_step(0);
        break;
    // Enable/Disable active damping
    case '9':
        if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
            vsi_set_activedamping(1);
        break;
    case '(': 
        if ((vsi_get_mode() == VSI_CI) || (vsi_get_mode() == VSI_CV))
            vsi_set_activedamping(0);
        break;
    // Enable/Disable space vector centering (open loop only)
    case '8':
        if (vsi_get_mode() == VSI_OL)
            vsi_set_svc(1);
        break;
    case '*':
        if (vsi_get_mode() == VSI_OL)
            vsi_set_svc(0);
        break;
    // Enable/Disable harmonic compensators
    // case '!':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vsi_set_harmcomp(HARMCOMP5);
// break;
// case '0':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vset_harmcomp(HARMCOMP7);
// break;
// case '#':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vset_harmcomp(HARMCOMP11);
// break;
// case '8':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vset_harmcomp(HARMCOMP13);
// break;
// case '1':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vset_clear_harmcomp(HARMCOMP5);
// break;
// case '2':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vset_clear_harmcomp(HARMCOMP7);
// break;
// case '3':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vset_clear_harmcomp(HARMCOMP11);
// break;
// case '4':
// if ((vsi_get_mode() == VSI_CI)||(vsi_get_mode() == VSI_CV))
// vset_clear_harmcomp(HARMCOMP13);
// break;
// Main contactor
// case 't':
// MAIN_CONTACTOR_OFF();
// break;
// case 'T':
// MAIN_CONTACTOR_ON();
// break;
// case 'g': // grab interrupt data
GrabRun();
break;
// case 'G': // stop grab interrupt data
GrabStop();
GrabClear();
break;
// case 'a': // dec para hack ref
// if (ph_ref > 0)
// ph_ref--;
// puts_COM0("\nPH ref = ");
// putu(ph_ref);
// puts_COM0("\n");
// break;
// case 'A': // inc para hack ref
// if (ph_ref < P_MAX - 1)
// ph_ref++;
// puts_COM0("\nPH ref = ");
// putu(ph_ref);
// puts_COM0("\n");
// break;
// case 'z': // dec para hack val
// ph_val--;
// puts_COM0("\nPH val = ");
// putd(ph_val);
// puts_COM0("\n");
// break;
// case 'Z': // inc para hack val
// ph_val++;
// puts_COM0("\nPH val = ");
Appendix C  DSP Source Code for the Experimental System

```c
// putd(ph_val);
// puts_COM0("n");
// break;
// case 'h': // corrupt parameter ph_ref
// para_hack(1,ph_ref,ph_val);
// puts_COM0("nPH ref = ");
// putd(ph_ref);
// puts_COM0(" corrupted
");
// break;
// case 'H': // rewrite parameter ph_val
// para_hack(0,ph_ref,ph_val);
// puts_COM0("nPH ref = ");
// putd(ph_ref);
// puts_COM0(" rewritten
");
// break;
// case '?': // print help information
// com_display(2);
// break;

// case '[':
// zx_offset += 1000000;
// break;
// case ']':
// zx_offset -= 1000000;
// break;
}
} /* end com_keyboard */

/* _Interrupts() */

/**
 * This interrupt occurs every millisecond. It sets flags at different intervals for the background loop to detect for event triggering. It also processes the watchdog timers.
 */

#ifndef BUILD_RAM
#pragma CODE_SECTION(isr_cpu_timer0, "ramfuncs");
#endif
interrupt void isr_cpu_timer0(void)
{
    static Uint16
    i_count = 0,
    ii;

    for (ii=0; ii<WD_TIMER_MAX; ii++)
    {
        if (wd_timer[ii] > 0)
            wd_timer[ii]--;
    }
    i_count++;
    if (i_count >= 100)
    {
        i_count = 0;
        time.flag.sec0_1 = 1;
        time.flag.msec = 1;
        time.count_msec++;
```
if (time.count_msec >= 1000)
{
    time.count_msec = 0;
    time.flag.sec = 1;
}

// Acknowledge this interrupt to receive more interrupts from group 1
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
} /* end isr_cpu_timer0 */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* Display the grabbed data for file storage mode == 1 prints the header, else prints a line of grabbed data starts from grab_index and continues on. */
void GrabDisplay(void)
{
    int j;
    static int skip = 0;
    if (grab_mode == GRAB_STOPPED)
    {
        puts_COM0("\n\ni\nt");
        for (j=0; j<GRAB_WIDTH; j++)
        {
            putd(j);
            puts_COM0("\t");
        }
        puts_COM0("\n");
        grab_mode = GRAB_SHOW;
        grab_index = 0;
    }
    else if (grab_mode == GRAB_SHOW)
    {
        if (skip == 0)
        {
            skip = 1;
            putd(grab_index);
            puts_COM0("\t");
            for (j=0; j<GRAB_WIDTH; j++)
            {
                #ifdef GRAB_SHORT
                putd(grab_array[grab_index][j]);
                #endif
            }
        }
    }
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* Initialise the grab data storage array */
void GrabInit(void)
{
    int i,j;
    grab_index = 0;
    grab_mode = GRAB_STOPPED;
    /* clear the grab array */
    for (i=0; i<GRAB_LENGTH; i++)
        for (j=0; j<GRAB_WIDTH; j++)
            grab_array[i][j] = 0;
    /* end for i */
    /* end for j */
    grab_mode = GRAB_IDLE;
    grab_dec = 0;
} /* end GrabInit */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* __Grab_Code() */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
ifdef GRAB_LONG
putl(grab_array[grab_index][j]);
endif
ifdef GRAB_DOUBLE
putdbl(grab_array[grab_index][j],4);
endif
puts_COM0("
"n"n"");
grab_index++;
if (grab_index >= GRAB_LENGTH)
{
    grab_mode = GRAB_IDLE;
    puts_COM0("
"n"");
}
else
{
    skip = 0;
}
} /* end GrabDisplay */

/* =========================================================================
__Test_Code()
============================================================================ */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This function writes data to the flash to corrupt or alter parameters. It is
a hack for testing the parameter storage code.

\author A.McIver
\param History:
\li 09/01/09 AM - initial creation
\param[in] type Sets the type of write: 0 for a new value, 1 for a corrupt value
\param[in] ref The internal reference for the parameter to write to
\param[in] val The new value to write
*/
void para_hack(Uint16 type, Uint16 ref, int16 val)
{
    Uint16
    data[6];
    data[0] = (unsigned char)(val&0x00FF);
    data[2] = data[0];
    data[4] = data[0];
    data[1] = (unsigned char)(val>>8);
    data[3] = data[1];
    data[5] = data[1];

    if (type != 0)
    {
        data[2] = (~data[2])&0x00FF;
        data[4] = data[0] + 1;
    }
    sf_write_buffer(256, data, 6);
    sf_write_page(ref);
} /* end para_hack */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
### Appendix C  DSP Source Code for the Experimental System

#### B.3. vsi.h

```c
/**
 * file vsi.h
 * brief ny5KW (inverter configuration stage) using the CPT-E13
 *
 * Developed By: Creative Power Technologies, (C) Copyright 2011
 * author A. McIver
 * History:
 * 01/05/07 DGH - derived from ele2.5kva/code/latest/cfpp.h
 * 25/07/07 AM  - added fault definitions
 * 04/10/07 AM  - updated documentation
 * 11/04/08 PM  - Ported to 30kW battery charger
 * 16/04/09 PM  - Release as V1.05
 * 02/02/10 PM  - Started Port to new version of 30kW BC - 30kW2
 * 17/03/11 PM  - Started Port to NY 5kW Active Rectifier
 * 15/05/12 PM  - Updated comments to preen and correct
 * 26/05/12 SP  - Begin conversion to LCL active rectifier
 * 27/06/14 RK  - negative sequence added
 */

#define ST_VSI_UV 0 ///< AC under voltage wait
#define ST_VSI_CHARGE 1 ///< Vhi DC bus is charging
#define ST_VSI_STOP 2 ///< rectifier is waiting for start signal
#define ST_VSI_RAMP 3 ///< rectifier is ramping up to target output
#define ST_VSI_RUN 4  ///< rectifier is operating normally
#define ST_VSI_FAULT 5  ///< rectifier is in a fault condition
#define ST_VSI_ERROR 6  ///< rectifier is in an unknown condition
#define ST_VSI_INIT 7  ///< rectifier is in initialization state
#define ST_VSI_CAL 8  ///< rectifier is in calibration state
#define ST_VSI_CAL_IAC 9  ///< rectifier is in calibrate Iac state
#define ST_VSI_SEQ 10  ///< rectifier is in check phase sequence state
#define ST_VSI_CNTCR 11  ///< rectifier is in contactor close state
#define ST_VSI_START 12  ///< rectifier is in VSI start State
#define ST_VSI_MAX 13  ///< array bound

#define VSI_CV 0 ///< constant voltage mode
#define VSI_OL 1  ///< open loop mode
#define VSI_CI 2  ///< Current Regulated

#define HARMCOMP3 0x01
#define HARMCOMP5 0x02
#define HARMCOMP7 0x04
#define HARMCOMP11 0x08
#define HARMCOMP13 0x10

#include "__Function_Prototypes()"

void vsi_state_machine(void);```

---

`VSI state machine`
/// Enable vsi switching (assuming no faults).
void vsi_enable(void);

/// Disable active rectifier switching.
void vsi_disable(void);

/// Set the operating mode for the vsi
void vsi_set_mode(Uint16 mode);

/// Returns the operating mode for the active rectifier
Uint16 vsi_get_mode(void);

/// Check for faults. Returns 0 for stopped, 1 for running, -1 for faulted.
int16 vsi_get_status(void);

/// Checks the slow speed faults
Uint32 vsi_check_fault(void);

/// Returns the operating state of the vsi state machine
Uint16 vsi_get_state(void);

/// Set the target d & q axis AC current for open loop operation
void vsi_set_id(int16 id);
void vsi_set_id_n(int16 id_n);
void vsi_set_iq(int16 iq);
void vsi_set_iq_n(int16 iq_n);
void vsi_set_idstep(int16 idst);
void vsi_set_idstep_n(int16 idst_n);
void vsi_set_iqstep(int16 igst);
void vsi_set_iqstep_n(int16 igst_n);

/// Set the target Vhi DC bus voltage
void vsi_set_vhi(Uint16 v);

/// Set the target modulation depth
void vsi_set_mod(double mod);

/// Set the step change flag for testing
void vsi_set_step(Uint16 stepp);

/// Enable/Disable active damping
void vsi_set_activedamping(Uint16 addi);

/// Enable/Disable space vector centering (open loop)
void vsi_set_svc(Uint16 svc);

/// Enable/Disable harmonic compensators
void vsi_set_harmcomp(Uint16 hc);
void vsi_clear_harmcomp(Uint16 hc);
Uint16 vsi_get_harmcomp(void);

/// Returns the target AC current
double vsi_get_iac_ref(void);

/** @name Measurement retrieval functions */
//@{
Uint16 vsi_get_vhi(void);  ///< returns Vdc in Volts
Uint16 vsi_get_vhi_mid(void);  ///< returns bus mid point in Volts
Uint16 vsi_get_vac(Uint16 phase);  ///< returns Vrms in Volts
Uint16 vsi_get_iac(Uint16 phase);  ///< returns Arms in tenths of an Amp
int16 vsi_get_p(void);  ///< returns output real power in W
int16 vsi_get_q(void);  ///< returns output reactive power in VAr
Uint16 vsi_get_va(void);  ///< returns output VA in VA
Uint16 vsi_get_pf(void);  ///< returns output power factor in hundredths
Uint16 vsi_get_step(void);  ///< returns if stepping or not
Uint16 vsi_get_activedamping(void);
Uint16 vsi_get_svc(void);
}
Appendix C  DSP Source Code for the Experimental System

//

/* __Typedefs() */

/// ADC channel type
/** This structure hold variables relating to a single ADC channel. These
variables are used for filtering, averaging, and scaling of this analog
quantity. */
typedef struct
{
    int16 raw, ///< raw ADC result from last sampling
    filt,  ///< decaying average fast filter of raw data
    fixed;

    int32 rms_sum, ///< interrupt level sum of data
    rms_sum_bak,  ///< background copy of sum for averaging
    dc_sum,      ///< interrupt level sum
    dc_sum_bak;  ///< background copy of sum for processing

    double real;  ///< background averaged and scaled measurement
} type_adc_ch;

typedef struct
{
    int16 raw_hi,  ///< raw ADC result from last hi sampling
    raw_lo,      ///< raw ADC result from last lo sampling
    filt,        ///< decaying average fast filter of raw data
    filt_fixed;

    int32 rms_sum, ///< interrupt level sum of data
    rms_sum_bak,  ///< background copy of sum for averaging
    dc_sum,      ///< interrupt level sum
    dc_sum_bak;  ///< background copy of sum for processing

    double real;  ///< background averaged and scaled measurement
} type_adc_ch_hl;

/* __ADC_Storage_Type() */

/// ADC storage type
/** This structure holds all the analog channels and some related variables
for the averaging and other processing of the analog inputs. There are also
virtual channels for quantities directly calculated from the analog inputs.

There are two separate RMS calculations. The output AC currents are calculated
every fundamental cycle based on the VSI phase variable. The input AC voltages
are calculated every 0.2 seconds (~10 fundamental cycles). This is because the
input AC is not synchronous with the VSI. The maximum error over 10 cycles is
+/-2.5%. The DC bus voltage and output DC voltage and current and power are
also calculated at this rate. */
typedef struct
{
    Uint16 count_cal,  ///< counter for low speed calibration summation
    count_rms,       ///< counter for full fund. period for RMS calculations
    count_rms_bak,   ///< background copy of RMS counter
    count_rms_in;    ///< counter for input RMS calculations

    flag_cal,        ///< flag set to trigger background calibration averaging
    flag_rms,       ///< flag set to trigger background RMS averaging
} type_adc_storage;
flag_rms_in;///< flag set to trigger background RMS averaging
int16
iac2_a_dc, ///< Iac2 A phase dc offset in ADC counts
iac2_b_dc, ///< Iac2 B phase dc offset in ADC counts
iac1_a_dc, ///< Iac1 A phase dc offset in ADC counts
iac1_b_dc, ///< Iac1 B phase dc offset in ADC counts
iac1_c_dc;
type_adc_ch_hl
vhi,  ///< DC intermediate (input) voltage
vhi2,
vhi_mid, ///< DC intermediate (input) midpoint voltage
vac_ac, ///< meas line to line AC input voltage
vac_bc; ///< meas line to line AC input voltage
type_adc_ch
vac_ab, ///< calc line to line AC input voltage
iac2_a, ///< A phase AC grid current
iac2_b, ///< B phase AC grid current
p_total, ///< total real power calculation
q_total, ///< total reactive power calculation
iac1_a, ///< A phase AC vsi current
iac1_b, ///< B phase AC vsi current
iac1_c, ///< C phase AC vsi current
yHA, ///< bank A high reference
yLA, ///< bank A low reference
yHB, ///< bank B high reference
yLB; ///< bank B low reference
double
p_va; ///< total apparent power calculation

/// Control loop type
/** This structure holds variables relating to a PI control loop. */
typedef struct
{
  int16
  ref_adc, ///< reference quantity set by background in ADC counts
targ_adc, ///< target set by ramp or other control in ADC counts
targ_fixed,
err, ///< error in ADC counts
Kp, ///< proportional gain
Ki; ///< integral gain
Uint16
overflow, ///< flag set if output overflows
underflow; ///< flag set if output underflows
int32
err_prop, ///< proportional error term
er_1nt, ///< integral error term
er_int_sum;///< summation of integral error term
} type_pi_control;

/// Control loop type
/** This structure holds variables relating to a PR control loop. */
typedef struct
{
  int32
  alpha0,
  alpha1,
  alpha2,
beta0,
beta1,
beta2,
s0,
s1,
s2,
y0;
} type_pr_coeff;
typedef struct {
    int16    ref_adc,
    targ_adc,
    targ_fixed,
    Kp,
    shift;
    int32    err,
    err_res,
    prop;
    type_pr_coeff
    fund,
    thir,
    fif,
    sev,
    elev,
    teen3;
} type_pr_control;

typedef struct {
    int32    targ_fixed,
    err,
    K;
} type_p_control;

/* =========================================================================
__Exported_Variables()
============================================================================ */
extern type_pi_control
    vhi;
extern type_pr_control
    iac2_a, iac2_b;
extern type_p_control
    iacc_a, iacc_b;

extern Uint16
    ZX_seen,
    in_sync,
    ZX_in_sync,
    ZX_state,
    ZX_count,
    ZX_cycles,
    ZX_sum;

extern Uint32
    phase_step;

extern int16
    ZX_time;

extern int32
    zx_offset,
    ZX_time_phase,
    ZX_phase_scale,
    ZX_phase_err,
    ZX_err_sum;

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * */
B.4. vsi.c

/**
brief ele30kW2 (battery charger) using the CPT-E01

The battery charger operation is split into two separate parts; the active rectifier and the phase shifted square wave DC-DC converter (PSSW). During development, the two sub-systems were run independently, but in normal operation, the PSSW only runs once the rectifier is running.

The active rectifier can run in current mode, following an AC current reference, or voltage mode, running an outer DC bus (Vhi) voltage control loop that produces a demanded AC current. The AC current regulator is a P+resonant controller.

The PSSW can run in one of three modes; open loop, constant current, and constant voltage. There is an inner current loop with an outer voltage loop so that the demanded current is always limited.

Open loop mode runs a basic PSSW with the reference setting the PSSW phase shift. There is no closed loop control. This is provided for testing the hardware and low level software.

In the constant current mode the current reference is set by the background code. A PI loop provides closed loop regulation of the output DC current to follow the reference by setting the PSSW phase shift. On start up, the output current target ramps up from zero to the reference set by the background.

In the constant voltage mode the voltage reference is set by the background code. A PI loop provides closed loop regulation of the output DC voltage by setting the target current for the inner control loop. On start up, the output voltage target ramps up from the present output voltage due to the battery to the reference set by the background.

\par Developed By:
Creative Power Technologies, (C) Copyright 2010
\author G. Holmes
\author A. McIver
\par History:
\li 01/05/07 DGH - derived from ele2.5kva/code/latest/cfpp.c
\li 04/10/07 AM - updated documentation
\li 14/11/07 PM - Modified ac trip current to 130A - this included changing circuit board resistors R29,R39,R40 to 3k9
\li 09/04/08 PM - Ported to 30kW Battery Charger
\li 16/04/09 PM - Release as V1.05
\li 01/05/09 PM - Changed deadtime to 2.56us
\li 01/05/09 PM - Changed deadtime to 1.92us
\li 04/05/09 PM - Changed deadtime to 1.49us
\li 25/05/09 PM - Changed deadtime back to 2.56us after failure in testing
\li 10/11/09 AM - Added DIP switch 1 disable check to earth fault detection
\li 11/11/09 AM - Added Iac O/C event restart rather than inst. trip
\li 22/12/09 AM - Removed Iac O/C event restart
\li 02/02/10 PM - Started Port to new version of 30kW BC - 30kW2
\li 25/08/10 PM - changed include file for lib_e01 to new structure
\li 17/03/11 PM - Started Port to NY 5kW Active Rectifier
\li 15/05/12 PM - Lint clean to remove unreferenced bits
\li 26/05/12 SP - Begin conversion to LCL active rectifier
\li 27/06/14 RK - Negative sequence added
*/

// compiler standard include files
#include <math.h>

// processor standard include files
#include <DSP281x_Device.h>
#include <bios0.h>

// common project include files
#include "main.h"
#include "vsi.h"
#include "conio.h"
#include "para.h"
#include "para_v.h"

// board standard include files
#include <lib_da2810.h>
#include "lib_el3_ny_5kW_vsi.h"

/*
 * Definitions()
 */
#define __SQRT2 1.414213562
#define __SQRT3 1.732050808
#define __PI 3.141592653
#define INV_SQRT3 37837 // 65536/sqrt(3)
#define INV_SQRT2 46341 // 65536/sqrt(2)
#define SQRT3_ON2 56756 // 65536*sqrt(3)/2
#define TWO29 536870912 // 2^29
#define TWO28 268435456 // 2^28

#define PERIOD_2 ((Uint16)(HSPCLK/SW_FREQ/4.0))
#define PERIOD (2*PERIOD_2)

#define MAX_TIME (signed int)(PERIOD_2-6)
#define FSW_VSI (HSPCLK/PERIOD/2)
#define FINT_VSI (2.0*FSW_VSI)

// Other modulation definitions
#define POS_SAT 1
#define NEG_SAT -1
#define NOT_SAT 0

#define ADC_CAL_TIME 0.5 // seconds
#define ADC_COUNT_CAL (Uint16)(ADC_CAL_TIME * FINT_VSI)
Appendix C   DSP Source Code for the Experimental System

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/// input RMS scaling
#define ADC_RMS_PS 1
#define ADC_DC_IN_PS 4
#define DC_IN_TIME 0.2 // seconds
#define COUNT_DC_IN (Uint16)(DC_IN_TIME * FINT_VSI)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/// Calibration modes
#define CAL_INIT 0
#define CAL_AVG 1
#define CAL_DONE 2

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/// Fault thresholds
/// instantaneous input AC overcurrent trip point
#define TRIP_IAC_OC 20.0 // Amps
#define VSI_TRIP_IAC_OC (int16)(TRIP_IAC_OC/ADC_IPH_SC) // counts

/// input AC voltage trip limits
#if _DEBUG==1
#define TRIP_VAC_MIN 0.0 // Volts
#else
#define TRIP_VAC_MIN 338.0 // Volts
#endif
#define TRIP_VAC_MAX 482.0 // Volts

/// input AC hysteresis
#define TRIP_VAC_HYST 5 // Volts

/// Input voltage loss of phase
#define AC_DIFF_LIM 0.7
#define AC_DIFF_MIN 50.0 // Vl

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/// minimum starting voltage
#if _DEBUG==1
#define VDC_START 10//(TRIP_VAC_MIN*1.35 - 20) // Volts
#else
#define VDC_START (TRIP_VAC_MIN*1.35 - 20) // Volts
#endif
#define VSI_VDC_START (int16)(VDC_START/ADC_VDC_SC) // counts
#define VAC_MIN_CLOSE TRIP_VAC_MIN // volts

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
///Start up Delays
/// Delay in stop state before starting
#define START_DELAY 1000 // msec
/// Delay in cal state before starting
#define CAL_DELAY 3000 // msec
/// Time for DC bus to soft charge
#define SOFT_CHARGE_TIME 10000 // msec

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//VSI active rectifier definitions
/* the phase is scaled so that one fundamental is 65536 counts. */
#define PHASE_STEP_SC (4294967296.0/SW_FREQ/2.0)
#define PHASE_STEP (Uint32)(PHASE_STEP_SC*F_FREQ)
#define PHASE_120_POS (Uint32)(4294967296.0/3.0 + 0.5)  // 32 bit( 2^32 for one fundamental(360 deg) and devide by 3 to make it for 120 degree)
#define PHASE_120_NEG (Uint32)(4294967296.0/6.0 + 0.5)  // 32 bit( 2^32 for one fundamental(360 deg) and devide by 6 to make it for 60 degree)
Appendix C  DSP Source Code for the Experimental System

#define PHASE_120_NEG (Uint32)(4294967296.0/3.0*2.0 + 0.5)
// 30 degree offset for line-line measurement
#define PHASE_30 (Uint32)(4294967296.0/12.0 + 0.5)
// 90 degree offset for reactive power
#define PHASE_90 (Uint32)(4294967296.0/4.0 + 0.5)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* Outer loop current control definitions */
// AC current ramp step size (in Arms) must be >= 0.1
#define STEP_IAC 0.1
#define STEP_IAC_ADC ((int16)(STEP_IAC / ADC_IPH_SC))

// Fixed point scalling definitions
#define LARGE_Q 5
#define LARGE_Q_SCALE 32.0
#define LARGE_Q_ROUND 16
#define FIXED_Q 10
#define FIXED_Q_SCALE 1024.0
#define FIXED_Q_ROUND 512
#define SMALL_Q 13
#define SMALL_Q_SCALE 8192.0
#define SMALL_Q_ROUND 4096

#define ADC_IAC_SC_FIXED ((int32)(ADC_IAC_SC*SMALL_Q_SCALE))
#define ADC_VDC_SC_FIXED ((int32)(ADC_VDC_SC*SMALL_Q_SCALE))

#define PROP_SHIFT_PR 1
#define INT_DISCARD 3

// Delta operator definitions
#define DELTA_Q 5
#define DELTA_ROUND 16
#define ALPHA0_Q 19
#define ALPHA0_ROUND 262144

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Inner loop current control definitions
#define STEP_TIMER 20000

/// Initial fundamentel resonant controller states when startup occurs at backEMF phase A peak
/// Positive sequence
#define S1A_POS -94935
#define S2A_POS 4068
#define S1B_POS 48650
#define S2B_POS 161664

/// Negative sequence
#define S1A_NEG -94983
#define S2A_NEG 1095
#define S1B_NEG 47629
#define S2B_NEG -164334

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Inner loop DC bus compensation definitions
#define MIN_VDC_COMP 60.0 // V
#define MIN_VDC_COMP_FIXED (int16)(MIN_VDC_COMP*LARGE_Q_SCALE)

#define VHI_NOM_FIXED (int32)(VHI_NOM*LARGE_Q_SCALE*FIXED_Q_SCALE)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// DC bus voltage ramp step size (in V)
#define STEP_VHI 0.1
#define STEP_VHI_ADC ((int16)(STEP_VHI / ADC_VDC_SC + 1))
/// Maximum demanded AC current
#define IAC_MAX_AMPS (1.3*IAC_NOM)
#define IAC_MAX (((Int16)(__SQRRT2*IAC_MAX_AMPS / ADC_IPH_SC))
#define IAC_MAX_FIXED (((Int32)(__SQRRT2*IAC_MAX_AMPS*FIXED_Q_SCALE)
#define P_SHIFT_VHI 8
#define I_SHIFT_VHI 18

/// Vhi over shoot limit where current clamp activates
#define VHI_OS_LIM_SET (((Int16)(20.0 / ADC_VDC_SC + 1))
/// Vhi over shoot limit where current clamp de-activates
#define VHI_OS_LIM_CLEAR (((Int16)(10.0 / ADC_VDC_SC + 1))

/* * * * * * * * * * * * * * * * * * * */
/* Zero crossing states */
#define ZX_LOST 0 ///< No idea of anything
#define ZX_EST 1 ///< Initial fundamental frequency estimation
#define ZX_SYNC 2 ///< nudges the phase to stay synchronised
#define ZX_FREQ 3 ///< nudges the freq (phase_step) for persistent err
#define ZX_LOCK 4 ///< tests to see if system is locked into sync
#define ZX_MISC 5 ///< load levelling calculation state

/* Zero crossing constants */
/* Sync lost if no ZX in ~3.5 cycles */
#define ZX_MAX_COUNT ((Uint16)(3.5*FINT_VSI/F_FREQ)) // 1050
#define ZX_SYNC_LIMIT 10 /* Number of cycles in sync */
#define ZX_BIG_ERR (400*65536) /* ~2.2 degrees */
#define ZX_PHASE_ERR (3600*65536) /* 20 degrees - maximum sync phase error
#define ZX_FREQ_ERR (100*65536) /* Persistent phase error for freq change
#define ZX_FREQ_ERR_BIG (200*65536) /* Persistent phase error for freq change

/* ZX Offset for the following conditions
Sample Rate: 10kHz
Voltage: 415Vll
0deg at: rising edge zero crossing of phase A (sine) */
#define ZX_OFFSET_POS -1708019968 // trim phase for +ve phase seq
#define ZX_OFFSET_NEG 1865869376 // trim phase for -ve phase seq

/* ZX Offset for the following conditions
Sample Rate: 20kHz
Voltage: 415Vll
0deg at: rising edge zero crossing of phase A (sine) */
#define ZX_OFFSET_POS -1708019968 // trim phase for +ve phase seq
#define ZX_OFFSET_NEG 1865869376 // trim phase for -ve phase seq

/* * * * * * * * * * * * * * * * * * * */
/// delay between phase sequence tests
#define SEQ_DELAY 100 // ms

/// phase sequence states
#define SEQ_INIT 0
#define SEQ_WAIT_NEG 1
#define SEQ_WAIT_ZX  2
#define SEQ_WAIT  3
#define SEQ_DONE  4

/// actual phase sequence
#define POS_SEQ  1
#define NEG_SEQ  0

/* * * * * * * * * * * * * * * * * * * */
/// initial condition startup states
#define IC_WAIT 0
#define IC_PEAK 1
#define IC_RUN 2
Appendix C  DSP Source Code for the Experimental System

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/// Heat sink over temperature debounce threshold
#define OT_COUNT_LIM 100
/// Emergency stop debounce threshold
#define ES_COUNT_LIM 10
/// Contactor fail count limit
#define CF_COUNT_LIM 200
/* ============================================================================
__Macros() 
============================================================================ */
#define VSI_FAST_STOP() {
EvaRegs.ACTRA.all = 0x00000000;
EvbRegs.ACTRB.all = 0x00000000;
is_vsi_switching = 0;
ic_state = IC_WAIT;
}
#define VSI_ENABLE() {
EvaRegs.ACTRA.all = 0x00060000;
EvbRegs.ACTRB.all = 0x00660000;
}
// extracts the low 16 bits from a 32 bit number for grabbing
#define LOW16(_val_) ((int16) (_val_ & 0x0000FFFF))
// extracts the high 16 bits from a 32 bit number for grabbing
#define HIGH16(_val_) ((int16) (_val_ >> 16))
/* ============================================================================
__Variables() 
============================================================================ */
/// state machine level variables
Uint16 flag_step = 0, ///< flag set to allow step changes in the reference
is_vsi_switching = 0, // flag set if VSI switching is active
op_mode_vsi = VSI_CI; // operating mode of the VSI
/** @name PWM interrupt variables */
//@{
Uint16 timer1_dir,
timer3_dir;
// Boot ROM sine table starts at 0x003ff000 and has 641 entries of 32 bit sine
// values making up one and a quarter periods (plus one entry). For 16 bit
// values, use just the high word of the 32 bit entry. Peak value is 0x40000000
int16 *sin_table = (int16 *) 0x003FF000, // pointer to sine table in boot ROM
*cos_table = (int16 *) 0x003FF100, // pointer to cos table in boot ROM
phase_offset; ///< round off amount from sine lookup
int16 val_diff, ///< interpolation temp variable
val_lo, ///< interpolation temp variable
sin_val_a, sin_val_b, ///< interpolated sine table value
cos_val_a, cos_val_b;
Uint16 index, ///< index into sine look-up table (phase >> 22)
cos_peak; ///< peak of quadrature/cosine current only
Uint16 vsi_en_outputs = 0, ///< trigger to turn on vsi outputs
active_damping = 1, ///< state of active damping
harm_comp = 0, /// state of harmonic compensators
svc_enable=1;

Uint32 phase_a = 0, phase_b = 0, phase_a_zx = 0, ///< running phase angle (2^32 == 360degrees)
phase_120 = PHASE_120_POS; ///< angle between A and B phases

204
Appendix C  DSP Source Code for the Experimental System

```c
int16
fixed_vdc_comp, ///< clamped measured dc voltage for DC bus comp
vdc_comp,     ///< DC bus comp term
t_a, t_b, t_c, t_cm, ///< switching times
V_Asat = 0, V_Bsat = 0, V_Csat = 0, V_Dsat = 0,
V_Asat_prev = 0, V_Bsat_prev = 0, V_Csat_prev = 0,
t_off,       ///< 3rd harmonic offset
adc_vac_a,   ///< inst. demanded A phase voltage from PI loop
adc_vac_b,   ///< inst. demanded B phase voltage from PI loop
adc_vac_d,   ///< inst. demanded d axis voltage from PI loop
adc_vac_q,   ///< inst. demanded q axis voltage from PI loop
adc_vac_d_ss, ///< inst. demanded d axis voltage from PI loop NSRF
adc_vac_q_ss, ///< inst. demanded q axis voltage from PI loop NSRF
adc_vac_a_test,  ///< inst. demanded d axis voltage from PI loop
adc_vac_b_test,  ///< inst. demanded q axis voltage from PI loop
adc_vac_cm,

iac_lim_adc = IAC_MAX, ///< upper demanded current limit in iac peak ADC
id_ref_adc = 0,       ///< background real reference in iac ADC counts
id_n_ref_adc = 0,      ///< background real reference negative sequence in iac ADC counts
iq_ref_adc = 0,        ///< background reactive reference in iac ADC counts
iq_n_ref_adc = 0,      ///< background reactive reference negative sequence in iac ADC counts
id_step_ref_adc = 0,   ///< background real reference in iac ADC counts
id_n_step_ref_adc = 0, ///< background real reference negative sequence in iac ADC counts
iq_step_ref_adc = 0,   ///< background reactive reference in iac ADC counts
iq_n_step_ref_adc = 0, ///< background reactive reference negative sequence in iac ADC counts
id_targ_adc,          ///< demanded real current in iac peak ADC
id_n_targ_adc,        ///< demanded real current negative sequence in iac peak ADC
iq_targ_adc,          ///< demanded reactive current in iac peak ADC
iq_targ_fixed,        // beta
iq_n_targ_fixed,
id_step_targ_adc = 0,
id_n_step_targ_adc = 0,

iq_step_targ_adc = 0,

iq_n_step_targ_adc = 0,
id_total_targ_fixed = 0,
id_n_total_targ_fixed = 0,

id_ss_total_targ_fixed = 0,
id_n_ss_total_targ_fixed = 0,

iq_total_targ_fixed = 0,

iq_n_total_targ_fixed = 0,
iacc_a_fixed, iacc_b_fixed,
alpha,
beta,

iac2_alpha_test,

iac2_beta_test,

ia_test1,

ib_test1,

ia_test2,

ia_test2,

iac2_d,

iac2_q,

iac2_d_ss,

iac2_q_ss,

iac2_d_err,

iac2_q_err,

iac2_d_ss_err,

iac2_q_ss_err,

iac1_alpha,

iac1_beta,

iac1_d,

iac1_q,

iac1_d_ss,
```

Appendix C  DSP Source Code for the Experimental System

\[ \text{Kp}_{\text{PI}} = (\text{int16})\left(\frac{\text{SMALL}_Q\text{SCALE} \cdot (\text{double})\text{KP}_{\text{OUTER}} \cdot (\text{double})(1<<\text{PROP}_\text{SHIFT}_\text{PR})}{(\text{double})(1<<\text{PROP}_\text{SHIFT}_\text{PR})}\right), \]
\[ \text{Ki}_{\text{PI}} = \left(\text{int16}\right)\left(\frac{32768.0\cdot (\text{double})\text{KP}_{\text{OUTER}}}{(\text{double})(\text{TR}_{\text{OUTER}} \cdot (\text{double})\text{FSW}_{\text{VSI}} \cdot 2)\cdot (\text{double})(1<<\text{INT}_{\text{DISCARD}})}\right), \]
\[ \text{vd}, \]
\[ \text{vq}, \]
\[ \text{vd}_{\text{ss}}, \]
\[ \text{vq}_{\text{ss}}, \]
\[ \text{va}, \]
\[ \text{vb}, \]
\[ \text{va}_{\text{1}}, \]
\[ \text{vb}_{\text{1}}, \]
\[ \text{v}_{\text{alpha}}, \]
\[ \text{v}_{\text{beta}}, \]
\[ \text{adc}_{\text{vac}_{\text{alpha}}}, \]
\[ \text{adc}_{\text{vac}_{\text{beta}}}, \]
\[ \text{iacc}_{\text{d}}, \]
\[ \text{iacc}_{\text{q}}, \]
\[ \text{iacc}_{\text{d}_{\text{ss}}}, \]
\[ \text{iacc}_{\text{q}_{\text{ss}}}, \]
\[ \text{icm}_{\text{fixed}} = 0; \]

\text{int32}
\text{iac2}_{\text{d}_{\text{prop}}},
\text{iac2}_{\text{d}_{\text{err}_{\text{int}}}},
\text{iac2}_{\text{d}_{\text{int}}},
\text{iac2}_{\text{q}_{\text{prop}}},
\text{iac2}_{\text{q}_{\text{err}_{\text{int}}}},
\text{iac2}_{\text{q}_{\text{int}}},
\text{iac2}_{\text{d}_{\text{ss}_{\text{prop}}}},
\text{iac2}_{\text{d}_{\text{ss}_{\text{err}_{\text{int}}}}},
\text{iac2}_{\text{d}_{\text{ss}_{\text{int}}}},
\text{iac2}_{\text{q}_{\text{ss}_{\text{prop}}}},
\text{iac2}_{\text{q}_{\text{ss}_{\text{err}_{\text{int}}}}},
\text{iac2}_{\text{q}_{\text{ss}_{\text{int}}}},
\text{iacc}_{\text{d}_{\text{err}}},
\text{iacc}_{\text{q}_{\text{err}}},
\text{iacc}_{\text{d}_{\text{ss}_{\text{err}}}},
\text{iacc}_{\text{q}_{\text{ss}_{\text{err}}}},
\text{id}_{\text{targ}_{\text{fixed}}};

\text{Uint16}
\text{step}_{\text{count}} = 0, //< minimum time between reference steps.
\text{step}_{\text{current}} = 0;

\text{int32}
\text{vhi}_{\text{comp}_{\text{fixed}}};
//@)

/** @name Control Loop Variables */
//@{

/// Open loop control variables

\text{int32}
\text{mod}_{\text{vsi}_{\text{period}} = 0; //for OL modulation}

/// Outer bus voltage loop variables
\text{type}_{\text{pi}_{\text{control}}}
\text{vhi} =
Appendix C  DSP Source Code for the Experimental System

```c
{ 0, // ref_adc  
 0, // tarq_adc  
 0, // tarq_fixed  
 0, // err  
  (int16) (KP_VHI* (FIXED_Q_SCALE/LARGE_Q_SCALE)*(1L<<P_SHIFT_VHI)), // Kp  
  (int16) (1.0/TINT_VSI/TIME_VHI*(1L<<I_SHIFT_VHI)), // Ki  
 0, // overflow  
 0, // underflow  
 0L, // err_int  
 0L // err_int_sum  
}; // Vhi DC bus control loop
//@}
```
// Inner active damping current loop variables
type_p_control
iacc_a =
{ 0, // targ_fixed
  0, // err
  (int32)(double)SMALL_Q_SCALE*(double)KDM_INNER/(double)VHI_NOM), // K
},
iacc_b =
{ 0, // targ_fixed
  0, // err
  (int32)(double)SMALL_Q_SCALE*(double)KDM_INNER/(double)VHI_NOM), // K
},
icm =
{ 0, // targ_fixed
  0, // err
  (int32)(double)SMALL_Q_SCALE*(double)KCM_INNER/(double)VHI_NOM) // Kcm
};

/** @name ADC variables */
//@{
type_adc
adc =
{ 0, // count_cal
  0, // count_rms
  0, // count_rms_bak
  0, // count_rms_in
  0, // flag_cal,
  0, // flag_rms
  0, // flag_rms_in
  0, // iac2_a_dc
  0, // iac2_b_dc
  0, // iac1_a_dc
  0, // iac1_b_dc
  0, // iac1_c_dc
  0, // raw_hi
  0, // raw_lo
  0, // filt
  0, // filt_fixed
  0L, // rms_sum
  0L, // rms_sum_bak
  0L, // dc_sum
  0L, // dc_sum_bak
  0.0 // real
 }, // vhi
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // vhi2
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // vac_ac
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // vac_bc
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // vac_ab
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // iac2_a
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // iac2_b
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // p_total
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // q_total
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // iacl_a
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // iacl_b
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // iacl_c
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // yHA
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // yLA
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // yHB
 { 0, 0, 0, 0, OL, OL, OL, OL, 0.0 }, // yLB
 0.0, // p_va
};
Appendix C  DSP Source Code for the Experimental System

```c
//@
/** @name ADC calibration variables */
//@
int16
cal_gainA = 1<<14,///< calibration gain factor for A channel
cal_gainB = 1<<14,///< calibration gain factor for B channel
cal_offsetA = 0,///< calibration offset for A channel
cal_offsetB = 0;///< calibration offset for B channel
double
cal_gain_A, cal_gain_B,
cal_offset_A, cal_offset_B;
Uint16
cal_mode = 0,///< calibration mode
cal_complete = 0;///< flag set once initial calibration cycle completed
//@
/** @name Zero Crossing Synch Variables */
//@
Uint16
ZX_seen = 0,///< flag set when a zx event is detected
in_sync = 0,///< Flag to indicate that sync is achieved
ZX_in_sync = 0,///< > ZX_SYNC_LIMIT means that sync has been achieved
ZX_state = ZX_LOST, ///< State of the zero crossing synch process
ZX_count = 0,///< The number of switching cycles between ZX interrupts
ZX_count_grab,///< for grab code only
ZX_cycles = 0,///< Count of number of ZXs during averaging
ZX_sum = 0;///< Running sum for average
Uint32
phase_step = PHASE_STEP, ///< Change in phase angle in half a switching cycle
phase_step_zx = PHASE_STEP;
int16
ZX_time = 0;///< Time of captured ZX in timer units
int32
ZX_time_phase = 0L,///< Time of captured ZX in phase units
zx_offset = ZX_OFFSET_POS, ///< variable offset for tuning (30 deg offset for line-line measurement
ZX_phase_scale = 0L,///< Scale factor between timer and phase units
ZX_phase_err = 0L,///< Difference in phase units (2^16 == 360deg)
ZX_err_sum = 0L;///< Integral for frequency control
Uint16
seq_state = SEQ_INIT,///< phase sequence state variable
seq_count = 0,///< count of passes through the phase detection loop
seq_pos_count = 0,///< count of positive phase sequences detected
phase_seq = 2;
//@
/** @name Initial condition startup */
//@
int16
ic_state = IC_WAIT;
//@
/* _Local_Function_Prototypes() */

// ADC and PWM interrupt
interrupt void isr_pwm(void);

// PDPINT interrupt
interrupt void isr_pdpint(void);

// XINT1 Iac over current interrupt
```
interrupt void isr_over_current(void);

/// XINT2 Vdc over voltage interrupt
interrupt void isr_over_voltage(void);

/// Checks for slow thresholds on inputs and outputs
void check_voltage_limits(void);

/// initialize variables
void variable_init(void);

/// Sets up and starts the PWM outputs (VSI)
void pwm_init(void);

/// Sets up the ADC for sampling triggered by PWM timer
void adc_init(void);

/// Calibrates the adc for gain and offset using the reference inputs.
void adc_calibrate(void);

/// Scales the RMS ADC quantities for use in background.
void adc_scale_rms(void);

/// Scales the slow filtered ADC quantities for use in background.
void adc_scale_slow(void);

/* ============================================================================
__State_Machine_Definitions()
=====
======================================================================= */
/** @name Active Rectifier State Machine Definitions */
//@{
void st_vsi_init(void), ///< The state initialisation function
st_vsi_cal(void), ///< Idc calibration state
st_vsi_cal_iac(void), ///< Iac calibration state
st_vsi_seq(void), ///< Phase sequence detection state
st_vsi_charging(void), ///< Charging the input bus
st_vsi_close_contactor(void), ///< Closing the main contactor
st_vsi_stop(void), ///< Waiting for start trigger
st_vsi_start(void), ///< Starts VSI switching before ramping
st_vsi_ramp_ol(void), ///< Ramps up the AC current
st_vsi_ramp_cl(void), ///< Ramps up the Vhi voltage or ref current
st_vsi_run(void), ///< Maintaining target output voltage
st_vsi_fault(void); ///< Wait for faults to clear

State_Type vsi_state =
{
 &st_vsi_init, // state function ptr
 1 // first state flag
};
//@

/* ============================================================================
__Exported_Functions() */
/* */

/** This function is called from the main background loop once every millisecond.
It calls the individual VSI states and performs other millisecond event actions
including:
  \li Over temperature fault detection
  \li Earth Leakage fault detection

210
Appendix C  DSP Source Code for the Experimental System

\li  RMS calculations every 20ms
\li  Overload fault detection
\li  Input RMS and Slow DC averages every 0.2 seconds
\li  Input and Output voltage limit checking
\li  ADC calibration calculations every 0.5 seconds

There was an external shutdown on DIGIN2 opening. This was removed since no-one seemed to need it. It should not be implemented at this level anyway. It should operate more like an off signal at the high level.

\author A.McIver
\par History:
\li  01/05/06 DGH - derived from ele2.5kva/code/latest/cfpp.c
\li  06/07/07 AM - added over temperature fault detection
\li  26/07/07 AM - added earth leakage fault detection
\li  11/04/08 PM - modified to reflect bc_variables
\li  23/12/08 AM - added low speed trip detection
\li  28/05/09 AM - removed external shutdown on DIGIN2
\li  24/01/12 SG - added emergency stop debounce timer reset
*/

void vsi_state_machine(void)
{
    static Uint16
        ot_count = 0,
        es_count = 0,
        cf_count = 0;
    Uint16
        cf_flag = 0;

    DO_STATE(vsi_state);

    if (adc.flag_rms != 0)
    {
        adc.flag_rms = 0;
        adc_scale_rms();
    }
    else if (adc.flag_rms_in != 0)
    {
        adc.flag_rms_in = 0;
        adc_scale_slow();
        if (cal_complete)
        {
            check_voltage_limits();
        }
    }
    else if (adc.flag_cal != 0)
    {
        adc.flag_cal = 0;
        adc_calibrate();
    }
    // check for emergency stop button press
    if (IS_EMERG_STOP())
    {
        if (es_count < ES_COUNT_LIM)
        {
            es_count++;
            if (es_count >= ES_COUNT_LIM)
            {  
                VSI_FAST_STOP();
                main_fault_set(FAULT_EMERG);
                es_count = 0;
            }
        }
        else if (es_count > 0)
        {
            es_count--;
        }
    }
// check for over temperature fault
if (IS_HEATSINK_OT())
{
if (ot_count < OT_COUNT_LIM)
{
  ot_count++;
  if (ot_count >= OT_COUNT_LIM)
  {
    VSI_FAST_STOP();
    main_fault_set(FAULT_OT);
  }
}
else if (ot_count > 0)
{
  ot_count--;
}
// check for contactor aux input mismatch
if (IS_MAIN_CONTACTOR_ON())
{
if (!IS_CONTACTOR_AUX())
{
  cf_flag = 1;
}
else // !(IS_MAIN_CONTACTOR_ON())
{
if (IS_CONTACTOR_AUX())
{
  cf_flag = 1;
}
if (cf_flag != 0)
{
if (cf_count < CF_COUNT_LIM)
{
  cf_count++;
  if (cf_count >= CF_COUNT_LIM)
  {
    VSI_FAST_STOP();
    main_fault_set(FAULT_CONT);
  }
}
else if (cf_count > 0)
{
  cf_count--;
}
} /* end vsi_state_machine */

/ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This function can be used to switch the VSI from the stopped state to a
running state. It is useful in a manually controlled system, but in the
standalone production version, the VSI leaves the stop state automatically.

\author A.McIver
\par History:
\li 17/02/10 AM - derived from bc_disable
/*/ 
void vsi_enable(void)
{
if (main_fault_get_reported() == 0)
  is_vsi_switching = 1;
} /* end vsi_enable */
This function can be used to switch the VSI from the running state to a stop state. It is useful in a manually controlled system, but in the standalone production version, there is no need to stop except in fault conditions.

```c
void vsi_disable(void)
{
    is_vsi_switching = 0;
} /* end vsi_disable */
```

Set the operating mode for the active rectifier.

```c
void vsi_set_mode(Uint16 mode)
{
    Uint16 status;
    if (is_vsi_switching == 0)
    {
        op_mode_vsi = mode;
        status = PARA_READ(P_STATUS);
        if (op_mode_vsi == VSI_CV)
        {
            status |= ST_VSI_CV;
            status &= ~(ST_VSI_OL|ST_VSI_CI);
        }
        else if (op_mode_vsi == VSI_CI)
        {
            status |= ST_VSI_CI;
            status &= ~(ST_VSI_CV|ST_VSI_OL);
        }
        else if (op_mode_vsi == VSI_OL)
        {
            status |= VSI_OL;
            status &= ~(ST_VSI_CV|ST_VSI_CI);
        }
        para_write_int(P_STATUS,status);
    }
} /* end vsi_set_mode */
```

Get the operating mode for the active rectifier.

```c
Uint16 vsi_get_mode(void)
```
```c
{  
  return op_mode_vsi;  
} /* end vsi_get_mode */

/**
 * Set the step change flag for testing
 */
void vsi_set_step(Uint16 stepp)  
{  
  flag_step = stepp;  
} /* end vsi_set_step */

/**
 * Get the step flag.
 */
Uint16 vsi_get_step(void)  
{  
  return flag_step;  
} /* end vsi_get_mode */

/**
 * Set the target real AC RMS current for open loop operation. Converts the
 * reference from Amps to iac ADC counts.
 */
void vsi_set_id(int16 id)  
{  
  if (op_mode_vsi == VSI_CI)  
  {  
    id_ref_adc = (int16)(((double)id*__SQRT2/10.0)/ADC_IPH_SC);  
  }  
} /* end vsi_set_id */

/**
 * Set the target real AC RMS current negative sequence for open loop operation. Converts the
 * reference from Amps to iac ADC counts.
 */
void vsi_set_id_n(int16 id)  
{  
  if (op_mode_vsi == VSI_CI)  
  {  
    id_ref_adc = (int16)(((double)id*__SQRT2/10.0)/ADC_IPH_SC);  
  }  
} /* end vsi_set_id */
```
void vsi_set_id_n(int16 id_n)
{
    if (op_mode_vsi == VSI_CI)
    {
        id_n_ref_adc = (int16)((double)id_n*SQRT2/10.0)/ADC_IPH_SC;
    }
} /* end vsi_set_id_n */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Set the target reactive AC RMS current for reactive power control. Converts the 
reference from Amps to iac ADC counts.

\author A.McIver
\par History:
\li 11/01/11 DS - initial creation
\param[in] iq Target AC current in Arms
*/
void vsi_set_iq(int16 iq)
{
    iq_ref_adc = (int16)((double)iq*SQRT2/10.0)/ADC_IPH_SC; 
} /* end vsi_set_iq */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Set the target reactive AC RMS current negative sequence for reactive power control. 
Converts the 
reference from Amps to iac ADC counts.

\author R.Kabiri
\par History:
\li 27/06/14 RK - initial creation
\param[in] iq_n Target AC current in Arms
*/
void vsi_set_iq_n(int16 iq_n)
{
    iq_n_ref_adc = (int16)((double)iq_n*SQRT2/10.0)/ADC_IPH_SC; 
} /* end vsi_set_iq_n */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Set the target real AC RMS current for open loop 
operation. Converts the 
\param[in] id Target AC current in Arms
*/
void vsi_set_idstep(int16 idst)
{
    if (op_mode_vsi == VSI_CI)
    {
        id_step_ref_adc = (int16)((double)idst*SQRT2/10.0)/ADC_IPH_SC;
    }
} /* end vsi_set_id */
/**
Set the target real AC RMS current negative sequence for open loop operation. Converts the reference from Amps to iac ADC counts.

\author R.Kabiri
\param[in] id_n Target AC current in Amps
*/
void vsi_set_idstep_n(int16 idst_n)
{
    if (op_mode_vsi == VSI_CI)
    {
        id_n_step_ref_adc = (int16)((double)idst_n*\_SQRT2/10.0)/ADC_IPH_SC);
    }
} /* end vsi_set_idstep_n */

/**
Set the target reactive AC RMS current for reactive power control. Converts the reference from Amps to iac ADC counts.

\author A.McIver
\param[in] iq Target AC current in Amps
*/
void vsi_set_iqstep(int16 iqst)
{
    iq_step_ref_adc = (int16)((double)iqst*\_SQRT2/10.0)/ADC_IPH_SC);
} /* end vsi_set_iq */

/**
Set the target reactive AC RMS current negative sequence for reactive power control. Converts the reference from Amps to iac ADC counts.

\author R.Kabiri
\param[in] iq_n Target AC current in Amps
*/
void vsi_set_iqstep_n(int16 iqst_n)
{
    iq_n_step_ref_adc = (int16)((double)iqst_n*\_SQRT2/10.0)/ADC_IPH_SC);
} /* end vsi_set_iqstep_n */

/**
This function sets the target Vhi DC bus voltage for the active rectifier voltage control loop.

\author A.McIver
\param[in] vhi The target Vhi DC bus voltage
*/
void vsi_set_vhi(Uint16 v)
Appendix C  DSP Source Code for the Experimental System

```c
{ vhi.ref_adc = (int16)((double)v / ADC_VDC_SC); /* end vsi_set_vhi */
/**
Set the target modulation depth for open loop operation.

\author A.McIver
\par History:
\li 01/05/07 DGH - derived from ele2.5kva/code/latest/cfpp.c
\li 09/12/11 DS - modified from phase shift to mod depth
\param[in] phase Target phase shift in degrees */
void vsi_set_mod(double mod)
{ if (op_mode_vsi == VSI_OL)
  if (fabs(mod) < 2.0)
    { mod_vsi_period = (int16)(mod * (double)PERIOD_2);
    }
  else
    { mod_vsi_period = (int16)(2.0 * (double)PERIOD_2);
    }
  }
} /* end vsi_set_mod */
/**
Allows disabling/enabling of active damping.

\author S.Parker
\par History:
\li 31/05/12 SP - creation */
void vsi_set_activedamping(Uint16 add)
{ active_damping = add;
} /* end vsi_set_mod */
Uint16 vsi_get_activedamping(void)
{ return active_damping;
} /* end vsi_set_mod */
/**
Allows disabling/enabling of space vector centering.

\author S.Parker
\par History:
\li 10/07/12 SP - creation */
void vsi_set_svc(Uint16 svce)
{ if (op_mode_vsi == VSI_OL)
   svc_enable = svce;
} /* end vsi_set_mod */
Uint16 vsi_get_svc(void)
{ return svc_enable;
} /* end vsi_set_mod */
/**
```

217
Appendix C  DSP Source Code for the Experimental System

Allows disabling/enabling of harmonic compensators.

\author S.Parker
\par History:
\li 36/06/12 SP - creation
*/
void vsi_set_harmcomp(Uint16 hc)
{
    harm_comp |= hc;
} /* end vsi_set_mod */

void vsi_clear_harmcomp(Uint16 hhc)
{
    harm_comp &= ~hhc;
} /* end vsi_set_mod */

Uint16 vsi_get_harmcomp(void)
{
    return harm_comp;
} /* end vsi_set_mod */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* * Check system status. */
\author A.McIver
\par History:
\li 01/05/07 DGH - derived from ele2.5kva\code\latest\cfpp.c

\texttt{\textbackslash ret\texttt{val 1} vsi system running}
\texttt{\textbackslash ret\texttt{val 0} system stopped}
\texttt{\textbackslash ret\texttt{val -1} system faul\texttt{ted}}
*/
int16 vsi_get_status(void)
{
    if (main_fault_get_reported() != 0)
    {
        return -1;
    }
    else
    {
        return is_vsi_switching;
    }
} /* end vsi_get_status */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* * Retrieves filtered and scaled Vdc measurements */
\author A.McIver
\par History:
\li 01/05/07 DGH - derived from ele2.5kva\code\latest\cfpp.c

\returns DC bus voltage in DC Volts
*/
Uint16 vsi_get_vhi(void)
{
    return (Uint16)(adc.vhi.real + 0.5);
} /* end vsi_get_vhi */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* * Retrieves filtered and scaled bus capacitor mid point measurement */
\author A.McIver
\par History:
Appendix C  DSP Source Code for the Experimental System

\li 17/01/12 AM - derived from vsi_get_vhi

\returns DC bus mid point voltage in DC Volts
*/
Uint16 vsi_get_vhi_mid(void)
{
  return (Uint16)(adc.vhi_mid.real + 0.5);
} /* end vsi_get_vhi_mid */

Retrieves filtered and scaled Vac measurements.

This function returns the AC voltage in RMS volts. The input parameter selects which voltage is returned. It can be one of Vab, Vcb, or the average of the two.

\author A.McIver
\par History:
\li 01/05/07 DGH - derived from ele2.5kva/code/latest/cfpp.c

\returns AC output voltage in RMS Volts

\param[in] phase selects which voltage is provided */
Uint16 vsi_get_vac(Uint16 phase)
{
  if (phase == 0)
  {
    return (Uint16)( (adc.vac_ac.real + adc.vac_bc.real)/2.0 + 0.5 );
  } else if (phase == 1)
  {
    return (Uint16)( adc.vac_ac.real + 0.5 );
  } else
  {
    return (Uint16)( adc.vac_bc.real + 0.5 );
  }
} /* end vsi_get_vac */

This function returns the AC current in tenths of an RMS Amp. The input parameter selects which current is returned. It can be one of Ia, Ib, Ic, or the average of the three.

\author A.McIver
\par History:
\li 01/05/07 DGH - derived from ele2.5kva/code/latest/cfpp.c

\returns AC intermediate current in tenths of an RMS Amp

\param[in] phase selects which current is provided */
Uint16 vsi_get_iac(Uint16 phase)
{
  if (phase == 0)
  {
    return (Uint16)( 10.0/2.0*(adc.iac2_a.real + adc.iac2_b.real) + 0.5 );
  } else if (phase == 1)
  {
    return (Uint16)( 10.0*adc.iac2_a.real + 0.5);
  } else //if (phase == 2)
Appendix C  DSP Source Code for the Experimental System

{   return (Uint16)(10.0*adc.iac2_b.real + 0.5); }
} /* end vsi_get_iac */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This function returns the Iac reference either directly for the open loop
mode or the output of the DC voltage control loop for the constant voltage
mode.

\author A.McIver
\par History:
\li 15/04/08 AM - initial creation
\returns The target AC current in Arms
*/
double vsi_get_iac_ref(void)
{
    return ((double)id_targ_adc/__SQRT2*ADC_IPH_SC);
} /* end vsi_get_iac_ref */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Returns output real power

\author A.McIver
\par History:
\li 03/07/07 AM - initial creation
\li 14/05/12 AM - removed scaling
\returns output real power in W
*/
int16 vsi_get_p(void)
{
    return (int16)(adc.p_total.real + 0.5);
} /* end vsi_get_p */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Returns output apparent power

\author A.McIver
\par History:
\li 03/07/07 AM - initial creation
\li 14/05/12 AM - removed scaling
\returns output apparent power in VA
*/
Uint16 vsi_get_va(void)
{
    return (Uint16)(adc.p_va + 0.5);
} /* end vsi_get_va */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Returns output reactive power

\author A.McIver
\par History:
\li 03/07/07 AM - initial creation
\li 14/05/12 AM - removed scaling
\returns output reactive power in VAr

220
Appendix C   DSP Source Code for the Experimental System

*/
int16 vsi_get_q(void)
{
  return (int16)(adc.q_total.real + 0.5);
} /* end vsi_get_q */

/**
 * brief Returns output power factor
 * Calculates the power factor of the output and scales it to hundredths. So a
 * power factor of 0.8 is returned as 80. This function makes no attempt to
 * correct for the effect of the output LC filter on the measured power factor.
 */
Uint16 vsi_get_pf(void)
{
  if (adc.p_va > adc.p_total.real)
  {
    if (adc.p_va != 0)
      return (Uint16)(100.0 * adc.p_total.real / adc.p_va + 0.5);
    else
      return 0;
  }
  else
    return 100;
} /* end vsi_get_pf */

/**
 * This function tests for the presence of the software tested fault. It is in
 * bc.c because it accesses the low level analog values. Tested faults are:
 * FAULT_PDPINT
 * FAULT_HW_AC_OC
 * FAULT_SW_VOUT_OV
 * FAULT_OT
 * FAULT_VDC_SHORT - clears when tripped
 * FAULT_EARTH
 * FAULT_SW_AC_OC
 * FAULT_HW_VHI_OV
 * FAULT_SW_OVIN
 * FAULT_SW_UVIN
 * FAULT_CHARGE
 * FAULT_SW_VHI_OV
 */
Uint32 vsi_check_fault(void)
{
  Uint32 faults = 0;

  // check for hardware AC over current fault
  if (GET_I_OV_TRIP() || GET_IDC_OV_TRIP())
  {
    faults = faults | (1 << 0);
  }
  /*
  \returns Any faults detected.
  */
  return faults;
} /* end vsi_check_fault */
Appendix C  DSP Source Code for the Experimental System

 faults |= FAULT_HW_AC_OC;

 // check for output over voltage fault

 // check for software AC over current fault
 if ( (adc.iac2_a.filt > VSI_TRIP_IAC_OC)
   || (adc.iac2_a.filt < -VSI_TRIP_IAC_OC)
   || (adc.iac2_b.filt > VSI_TRIP_IAC_OC)
   || (adc.iac2_b.filt < -VSI_TRIP_IAC_OC)
   )
   { faults |= FAULT_SW_AC_OC;
   }

 if ( (adc.iac1_a.filt > VSI_TRIP_IAC_OC)
   || (adc.iac1_a.filt < -VSI_TRIP_IAC_OC)
   || (adc.iac1_b.filt > VSI_TRIP_IAC_OC)
   || (adc.iac1_b.filt < -VSI_TRIP_IAC_OC)
   || (adc.iac1_c.filt > VSI_TRIP_IAC_OC)
   || (adc.iac1_c.filt < -VSI_TRIP_IAC_OC)
   )
   { faults |= FAULT_SW_AC_OC;
   }

 // check for hardware DC bus over voltage fault
 if (GET_VDC1_OV_TRIP())
   { faults |= FAULT_HW_VHI_OV;
   }

 // check for input AC over voltage fault
 if ( (adc.vac_ac.real > TRIP_VAC_MAX)
   || (adc.vac_bc.real > TRIP_VAC_MAX) )
   { faults |= FAULT_SW_OVIN;
   }

 else if (main_fault_get_reported()&FAULT_SW_OVIN)
   { if ( (adc.vac_ac.real > (TRIP_VAC_MAX-TRIP_VAC_HYST))
     || (adc.vac_bc.real > (TRIP_VAC_MAX-TRIP_VAC_HYST)) )
     { faults |= FAULT_SW_OVIN;
     }
   }

 // check for input AC under voltage fault
 if ( (adc.vac_ac.real < TRIP_VAC_MIN)
   || (adc.vac_bc.real < TRIP_VAC_MIN) )
   { faults |= FAULT_SW_UVIN;
   }

 else if (main_fault_get_reported()&FAULT_SW_UVIN)
   { if ( (adc.vac_ac.real < (TRIP_VAC_MIN+TRIP_VAC_HYST))
     || (adc.vac_bc.real < (TRIP_VAC_MIN+TRIP_VAC_HYST)) )
     { faults |= FAULT_SW_UVIN;
     }
   }

 // charge fault is not checked
 return faults;
 } /* end vsi_check_fault */
Appendix C  DSP Source Code for the Experimental System

\par History:
\li 01/05/07 DGH - derived from ele2.5kva\code\latest\cfpp.c

\returns operating state of active rectifier state machine
*/
Uint16 vsi_get_state(void)
{
    Uint16 state;

    if (IS_CURRENT_STATE(vsi_state, st_vsi_run))
        state = ST_VSI_RUN;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_init))
        state = ST_VSI_INIT;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_cal))
        state = ST_VSI_CAL;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_cal_iac))
        state = ST_VSI_CAL_IAC;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_cal_iac))
        state = ST_VSI_SEQ;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_cal_iac))
        state = ST_VSI_SEQ;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_cal_iac))
        state = ST_VSI_SEQ;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_close_contactor))
        state = ST_VSI_SEQ;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_stop))
        state = ST_VSI_STOP;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_ramp_cl))
        state = ST_VSI_RAMP;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_ramp_cl))
        state = ST_VSI_RAMP;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_ramp_cl))
        state = ST_VSI_RAMP;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_start))
        state = ST_VSI_START;
    else if (IS_CURRENT_STATE(vsi_state, st_vsi_fault))
        state = ST_VSI_FAULT;
    else
        state = ST_VSI_ERROR;

    return state;
} /* end vsi_get_state */

/*===========================================================================
\_Interruts()
============================================================================
*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
\brief Updates VSI and stores ADC results

This interrupt is triggered by the completion of the ADC conversions. It then:
\li stores the ADC results
\li applies the ADC calibration factors
\li sums the calibration measurements
\li applies a fast decaying average filter to the analog signals
\li checks for fault conditions
\li performs low speed averaging and rms calculations
\li DC bus compensation
\li updates phase angle
\li calculates switching times
\li centers pulses in switching period
\li loads compares registers with switching times
\li sets up analogs for next interrupt

\author A.McIver
\author G.Holmes
\par History:
\li 01/05/07 DGH - derived from ele2.5kva\code\latest\cfpp.c

223
Appendix C   DSP Source Code for the Experimental System

\li 11/11/09 AM - added Iac O/C event restart rather than trip
\li 11/22/09 AM = removed Iac O/C event restart
\li 11/29/110 AM = added Idc out o/c counter for slower trip time
*/
#ifdef BUILD_RAM
#pragma CODE_SECTION(isr_pwm, "ramfuncs")
#endif
interrupt void isr_pwm(void)
{
    // Timing bit
    SET_LED0_E13();

    /* Find out the direction which the timers are going */
    //1 = PWM carrier is currently rising (underflow interrupt)
    //0 = PWM carrier is currently falling (period interrupt)
timer1_dir = EvaRegs.GPTCONA.bit.T1STAT;
timer3_dir = EvbRegs.GPTCONB.bit.T3STAT;
    
    /* ========================================================================= 
    isr_pwm_read_analogs() 
    ============================================================================= */
    adc.iac2_a.raw = (AdcRegs.ADCRESULT0>>4);
    adc.iac2_a.raw = (int16)( ((int32)adc.iac2_a.raw*(int32)cal_gainA) >> 14) - cal_offsetA - ADC_IPH_OFFSET - adc.iac2_a_dc;
    adc.iac2_b.raw = (AdcRegs.ADCRESULT2>>4);
    adc.iac2_b.raw = (int16)( ((int32)adc.iac2_b.raw*(int32)cal_gainA) >> 14) - cal_offsetA - ADC_IPH_OFFSET - adc.iac2_b_dc;
    adc.iac1_a.raw = (AdcRegs.ADCRESULT6>>4);
    adc.iac1_a.raw = (int16)( ((int32)adc.iac1_a.raw*(int32)cal_gainA) >> 14) - cal_offsetA - ADC_IPH_OFFSET - adc.iac1_a_dc;
    adc.iac1_b.raw = (AdcRegs.ADCRESULT8>>4);
    adc.iac1_b.raw = (int16)( ((int32)adc.iac1_b.raw*(int32)cal_gainA) >> 14) - cal_offsetA - ADC_IPH_OFFSET - adc.iac1_b_dc;
    adc.iac1_c.raw = (AdcRegs.ADCRESULT4>>4);
    adc.iac1_c.raw = (int16)( ((int32)adc.iac1_c.raw*(int32)cal_gainA) >> 14) - cal_offsetA - ADC_IPH_OFFSET - adc.iac1_c_dc;
    if (timer1_dir == 1) // Underflow interrupt
    {
        adc.vhi_raw_lo = (AdcRegs.ADCRESULT9>>4);
        adc.vhi_raw_lo = (int16)( ((int32)adc.vhi_raw_lo*(int32)cal_gainB) >> 14) - cal_offsetB;
        adc.vhi2_raw_lo = (AdcRegs.ADCRESULT1>>4);
        adc.vhi2_raw_lo = (int16)( ((int32)adc.vhi2_raw_lo*(int32)cal_gainB) >> 14) - cal_offsetB;
        adc.vhi_mid_raw_lo = (AdcRegs.ADCRESULT7>>4);
        adc.vhi_mid_raw_lo = (int16)( ((int32)adc.vhi_mid_raw_lo*(int32)cal_gainB) >> 14) - cal_offsetB;
        adc.vac_bc_raw_lo = (AdcRegs.ADCRESULT3>>4);
        adc.vac_bc_raw_lo = (int16)( ((int32)adc.vac_bc_raw_lo*(int32)cal_gainB) >> 14) - cal_offsetB - ADC_VAC_OFFSET;
        adc.vac_ac_raw_lo = (AdcRegs.ADCRESULT5>>4);
        adc.vac_ac_raw_lo = (int16)( ((int32)adc.vac_ac_raw_lo*(int32)cal_gainB) >> 14) - cal_offsetB - ADC_VAC_OFFSET;
    }
    else // Period interrupt
    {
        adc_vhi.raw_hi = (AdcRegs.ADCRESULT9>>4);
    
    
    224
Appendix C  DSP Source Code for the Experimental System

```c
adc.vhi.raw_hi = (int16)((int32)adc.vhi.raw_hi*(int32)cal_gainB) >> 14)
    - cal_offsetB;
adc.vhi2.raw_hi = (int16)((int32)adc.vhi2.raw_hi*(int32)cal_gainB) >> 14)
    - cal_offsetB;
adc.vhi_mid.raw_hi = (int16)((int32)adc.vhi_mid.raw_hi*(int32)cal_gainB) >> 14)
    - cal_offsetB;
adc.vac_bc.raw_hi = (int16)((int32)adc.vac_bc.raw_hi*(int32)cal_gainB) >> 14)
    - cal_offsetB - ADC_VAC_OFFSET;
adc.vac_ac.raw_hi = (int16)((int32)adc.vac_ac.raw_hi*(int32)cal_gainB) >> 14)
    - cal_offsetB - ADC_VAC_OFFSET;
}
// calibration from references
adc.yHA.dc_sum += (Uint32)(AdcRegs.ADCRESULT12>>4);
adc.yLA.dc_sum += (Uint32)(AdcRegs.ADCRESULT14>>4);
adc.yHB.dc_sum += (Uint32)(AdcRegs.ADCRESULT13>>4);
adc.yLB.dc_sum += (Uint32)(AdcRegs.ADCRESULT15>>4);
adc.count_cal++; if (adc.count_cal > ADC_COUNT_CAL) {
adc.count_cal = 0;
adc.yHA.dc_sum_bak = adc.yHA.dc_sum;
adc.yLA.dc_sum_bak = adc.yLA.dc_sum;
adc.yHB.dc_sum_bak = adc.yHB.dc_sum;
adc.yLB.dc_sum_bak = adc.yLB.dc_sum;
adc.yHA.dc_sum = 0;
adc.yLA.dc_sum = 0;
adc.yHB.dc_sum = 0;
adc.yLB.dc_sum = 0;
adc.flag_cal = 1;
}
// fast filter ADC results
adc.vhi.filt = (3*adc.vhi.filt + ((adc.vhi.raw_hi + adc.vhi.raw_lo)>>1) + 2)>>2;
adc.vhi2.filt = (3*adc.vhi2.filt + ((adc.vhi2.raw_hi + adc.vhi2.raw_lo)>>1) + 2)>>2;
adc.vhi_mid.filt = (3*adc.vhi_mid.filt + ((adc.vhi_mid.raw_hi + adc.vhi_mid.raw_lo)>>1) + 2)>>2;
adc.vac_ac.filt = (3*adc.vac_ac.filt + ((adc.vac_ac.raw_hi + adc.vac_ac.raw_lo)>>1) + 2)>>2;
adc.vac_bc.filt = (3*adc.vac_bc.filt + ((adc.vac_bc.raw_hi + adc.vac_bc.raw_lo)>>1) + 2)>>2;
adc.vac_ab.filt = adc.vac_ac.filt - adc.vac_bc.filt;
adc.iac2_a.filt = adc.iac2_a.raw;
adc.iac2_b.filt = adc.iac2_b.raw;
adc.iac1_a.filt = adc.iac1_a.raw;
adc.iac1_b.filt = adc.iac1_b.raw;
adc.iac1_c.filt = adc.iac1_c.raw;
// Fixed-point scaled variables
adc.iac2_a.fixed = (int16)(((int32)adc.iac2_a.raw*ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q_FIXED_Q); // scale is FIXED_Q
adc.iac2_b.fixed = (int16)(((int32)adc.iac2_b.raw*ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q_FIXED_Q); // scale is FIXED_Q
adc.iac1_a.fixed = (int16)(((int32)adc.iac1_a.raw*ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q_FIXED_Q); // scale is FIXED_Q
adc.iac1_b.fixed = (int16)(((int32)adc.iac1_b.raw*ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q_FIXED_Q); // scale is FIXED_Q
adc.iac1_c.fixed = (int16)(((int32)adc.iac1_c.raw*ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q_FIXED_Q); // scale is FIXED_Q
```
Appendix C  DSP Source Code for the Experimental System

```c
// adc.vhi.filt_fixed = (int16)(((int32)adc.vhi.filt*(int32)ADC_VDC_SC_FIXED +
(int32)64)<<(SMALL_Q-LARGE_Q)); // scale is LARGE_Q
// vvvvvvvvvv RESCALE RESISTORS AND CONNECT VDC3 BEFORE UNCOMMENTING vvvvvvvvv
adc.vhi.filt_fixed = (int16)(((int32)(adc.vhi.filt+adc.vhi2.filt)*ADC_VDC_SC_FIXED
 + (int32)128)<<(SMALL_Q-LARGE_Q+1)); // scale is LARGE_Q

/* =========================================================================

isr_pwm_zx()
===============
============================================================= */

if (EvRegs.CAPFIFOA.bit.CAP1FIFO != 0)
{
  ZX_time = PERIOD - EvRegs.CAP1FIFO;
  ZX_seen = 1;
  EvRegs.CAPFIFOA.all = 0x0000; // dump any other captured values
}

ZX_count++;
if ((ZX_count > ZX_MAX_COUNT) // Zero crossing signal lost
{
  VSI_FAST_STOP(); // Halt modulation
  in_sync = 0;
  ZX_state = ZX_LOST; // Restart searching for sync
  ZX_in_sync = 0;
  ZX_count = 0;
}

if (ZX_state == ZX_LOST) // No idea of anything: start freq est.
{
  in_sync = 0;
  if (ZX_seen != 0)
  {
    ZX_seen = 0;
    ZX_cycles = 0;
    ZX_sum = 0;
    ZX_count = 0;
    ZX_state = ZX_EST;
  }
}
else if (ZX_state == ZX_EST) // Roughly measure period and average
{
  if (ZX_seen != 0)
  {
    ZX_seen = 0;
    ZX_cycles++;
    ZX_sum += ZX_count;
    ZX_count = 0; // Reset counter
  }
  if (ZX_cycles >= ZX_CYCLE_AVG)
  {
    ZX_sum = ZX_sum/ZX_CYCLE_AVG;
    phase_step_zx = ((Uint32)(0xFFFF/ZX_sum))<<16; // Approximate frequency
    ZX_sum = ZX_sum/8; // Also use for glitch filter
    phase_a_zx = phase_step_zx + zx_offset; // Within phase_step
    ZX_state = ZX_MISC; // Calculate ZX_phase_scale first
  }
} else if (ZX_state == ZX_SYNC) // Accurately measure phase error
{
  if (ZX_seen != 0)
  {
    ZX_seen = 0;
    if (ZX_count > ZX_sum) // Ignore glitches
    {
      ZX_count_grab = ZX_count;
      ZX_count = 0;
      // Rescale to phase units
      ZX_time_phase = zx_offset + (((int32)ZX_time*ZX_phase_scale)>>5));
```
// Calculate phase error captured time
ZX_phase_err = phase_a_zx - ZX_time_phase;
// Limit size of phase change
if (ZX_phase_err > ZX_BIG_ERR)
{
    phase_a_zx -= ZX_BIG_ERR;
    // Integrate phase errors
    ZX_err_sum = (ZX_err_sum+ZX_BIG_ERR)>>1;
}
else if (ZX_phase_err < -ZX_BIG_ERR)
{
    phase_a_zx += ZX_BIG_ERR;
    ZX_err_sum = (ZX_err_sum-ZX_BIG_ERR)>>1;
}
else
{
    phase_a_zx = ZX_phase_err;
    ZX_err_sum = (ZX_err_sum+ZX_phase_err)>>1;
}
ZX_state = ZX_FREQ;

else if (ZX_state == ZX_FREQ) // Nudge frequency if needed
{
    // If too large, nudge freq (phase_step)
    if (ZX_err_sum > ZX_FREQ_ERR)
    {
        phase_step_zx = 100L;
        if (ZX_err_sum > ZX_FREQ_ERR_BIG)
        {
            phase_step_zx = 1000L;
        }
    }
    else if (ZX_err_sum < -ZX_FREQ_ERR)
    {
        phase_step_zx += 100L;
        if (ZX_err_sum < -ZX_FREQ_ERR_BIG)
        {
            phase_step_zx += 1000L;
        }
    }
    ZX_state = ZX_LOCK;
}
else if (ZX_state == ZX_LOCK) // Test to see if still in sync
{
    if (ZX_in_sync >= ZX_SYNC_LIMIT)
    {
        // Gone out of sync
        VSI_FAST_STOP();
        ZX_in_sync = 0;
        in_sync = 0;
    }
    else
    {
        in_sync = 1;
    }
}
else if ((ZX_phase_err>ZX_PHASE_ERR)||(ZX_phase_err<-ZX_PHASE_ERR))
{
    // In sync this cycle
    ZX_in_sync++;
}
else
{
    ZX_in_sync = 0;
}
ZX_state = ZX_MISC;
else if (ZX_state == ZX_MISC)
{
    ZX_phase_scale = (phase_step_zx<<5)/PERIOD;
    ZX_state = ZX_SYNC;
}

// Update phase angle
phase_a_zx += phase_step_zx;

if(op_mode_vsi == VSI_OL)
{
    // force 50Hz open loop
    phase_a += PHASE_STEP;
    phase_step = PHASE_STEP;
} else
{
    // sync in closed loop
    phase_a = phase_a_zx;
    phase_step = phase_step_zx;
}

// Zero crossings of phase A
if(phase_a < 0x80000000)
    SET_LED1_E13();
else
    CLEAR_LED1_E13();

// Quadrature current (cosine) peak detect
// if(phase_a < phase_step)
if( (phase_a - PHASE_90) < phase_step)
    cos_peak = 1;
else
    cos_peak = 0;

/* =========================================================================
isr_pwm_analog_faults()
============================================================================ */

// check for analog faults
if ( (adc.iac2_a.filt > VSI_TRIP_IAC_OC) |
     (adc.iac2_a.filt < -VSI_TRIP_IAC_OC) |
     (adc.iac2_b.filt > VSI_TRIP_IAC_OC) |
     (adc.iac2_b.filt < -VSI_TRIP_IAC_OC) )
{
    VSI_FAST_STOP(); // fast shutdown
    main_fault_set_int(FAULT_SW_AC_OC);
}
if ( (adc.iac1_a.filt > VSI_TRIP_IAC_OC) |
     (adc.iac1_a.filt < -VSI_TRIP_IAC_OC) |
     (adc.iac1_b.filt > VSI_TRIP_IAC_OC) |
     (adc.iac1_b.filt < -VSI_TRIP_IAC_OC) |
     (adc.iac1_c.filt > VSI_TRIP_IAC_OC) |
     (adc.iac1_c.filt < -VSI_TRIP_IAC_OC) )
{
    VSI_FAST_STOP(); // fast shutdown
    main_fault_set_int(FAULT_SW_AC_OC);
}

// poll over current hardware trip since interrupt is level triggered
if (GET_I_OV_TRIP()||GET_IDC_OV_TRIP()) //I=iac2, IDC=iac1
{
    VSI_FAST_STOP(); // fast shutdown
    main_fault_set_int(FAULT_HW_AC_OC);
}
// poll over voltage hardware trip since interrupt is level triggered
if (GET_VDC1_OV_TRIP())
{ VSI_FAST_STOP(); // fast shutdown
    main_fault_set_int(FAULT_HW_VHI_OV);
}

/*========================================================================
** isr_pwm_low_speed_average()**
========================================================================*/
adc.count_rms_in++;
adc.vhi.dc_sum += (int32)adc.vhi.filt;
adc.vhi_mid.dc_sum += (int32)adc.vhi_mid.filt;
if (adc.count_rms_in >= COUNT_DC_IN)
{
    adc.flag_rms_in = 1;
    adc.vhi.dc_sum_bak = adc.vhi.dc_sum;
    adc.vhi.dc_sum = 0L;
    adc.vhi_mid.dc_sum_bak = adc.vhi_mid.dc_sum;
    adc.vhi_mid.dc_sum = 0L;
    adc.count_rms_in = 0;
}
adc.count_rms++;
adc.iac2_a.rms_sum += (int32)(((int32)adc.iac2_a.filt*(int32)adc.iac2_a.filt) >>ADC_RMS_PS);
adc.iac2_a.dc_sum += (int32)adc.iac2_a.filt;
adc.iac2_b.rms_sum += (int32)(((int32)adc.iac2_b.filt*(int32)adc.iac2_b.filt) >>ADC_RMS_PS);
adc.iac2_b.dc_sum += (int32)adc.iac2_b.filt;
adc.iacl_a.rms_sum += (int32)(((int32)adc.iacl_a.filt*(int32)adc.iacl_a.filt) >>ADC_RMS_PS);
adc.iacl_a.dc_sum += (int32)adc.iacl_a.filt;
adc.iacl_b.rms_sum += (int32)(((int32)adc.iacl_b.filt*(int32)adc.iacl_b.filt) >>ADC_RMS_PS);
adc.iacl_b.dc_sum += (int32)adc.iacl_b.filt;
adc.iacl_c.rms_sum += (int32)(((int32)adc.iacl_c.filt*(int32)adc.iacl_c.filt) >>ADC_RMS_PS);
adc.iacl_c.dc_sum += (int32)adc.iacl_c.filt;
adc.vac_ac.rms_sum += (int32)(((int32)adc.vac_ac.filt*(int32)adc.vac_ac.filt) >>ADC_RMS_PS);
adc.vac_ac.dc_sum += (int32)adc.vac_ac.filt;
adc.vac_bc.rms_sum += (int32)(((int32)adc.vac_bc.filt*(int32)adc.vac_bc.filt) >>ADC_RMS_PS);
adc.vac_bc.dc_sum += (int32)adc.vac_bc.filt;
adc.vac_ab.rms_sum += (int32)(((int32)adc.vac_ab.filt*(int32)adc.vac_ab.filt) >>ADC_RMS_PS);
adc.vac_ab.dc_sum += (int32)adc.vac_ab.filt;
adc.p_total.rms_sum += (int32)(((int32)adc.iac2_a.filt*(int32)adc.vac_ac.filt)>>ADC_RMS_PS) + (((int32)adc.iac2_b.filt*(int32)adc.vac_bc.filt)>>ADC_RMS_PS);
adc.q_total.rms_sum += (int32)(((int32)adc.iac2_a.filt*(2*((int32)adc.vac_bc.filt-(int32)adc.vac_ac.filt)))>>ADC_RMS_PS) - (((int32)adc.iac2_b.filt*(2*((int32)adc.vac_ac.filt-(int32)adc.vac_bc.filt)))>>ADC_RMS_PS); //Q=(((vbc-0.5*vac)*ia-(vac-0.5*vbc)*ib)*2/sqrt(3)
// only update rms sum over full cycle
if (phase_a < phase_step)
Appendix C  DSP Source Code for the Experimental System

```c
adc.flag_rms = 1;
adc.iac2_a.rms_sum_bak = adc.iac2_a.rms_sum;
adc.iac2_a.rms_sum = 0L;
adc.iac2_a.dc_sum_bak = adc.iac2_a.dc_sum;
adc.iac2_a.dc_sum = 0L;
adc.iac2_b.rms_sum_bak = adc.iac2_b.rms_sum;
adc.iac2_b.rms_sum = 0L;
adc.iac2_b.dc_sum_bak = adc.iac2_b.dc_sum;
adc.iac2_b.dc_sum = 0L;
adc.iac1_a.rms_sum_bak = adc.iac1_a.rms_sum;
adc.iac1_a.rms_sum = 0L;
adc.iac1_a.dc_sum_bak = adc.iac1_a.dc_sum;
adc.iac1_a.dc_sum = 0L;
adc.iac1_b.rms_sum_bak = adc.iac1_b.rms_sum;
adc.iac1_b.rms_sum = 0L;
adc.iac1_b.dc_sum_bak = adc.iac1_b.dc_sum;
adc.iac1_b.dc_sum = 0L;
adc.iac1_c.rms_sum_bak = adc.iac1_c.rms_sum;
adc.iac1_c.rms_sum = 0L;
adc.iac1_c.dc_sum_bak = adc.iac1_c.dc_sum;
adc.iac1_c.dc_sum = 0L;
adc.vac_ac.rms_sum_bak = adc.vac_ac.rms_sum;
adc.vac_ac.rms_sum = 0L;
adc.vac_ac.dc_sum_bak = adc.vac_ac.dc_sum;
adc.vac_ac.dc_sum = 0L;
adc.vac_bc.rms_sum_bak = adc.vac_bc.rms_sum;
adc.vac_bc.rms_sum = 0L;
adc.vac_bc.dc_sum_bak = adc.vac_bc.dc_sum;
adc.vac_bc.dc_sum = 0L;
adc.vac_ab.rms_sum_bak = adc.vac_ab.rms_sum;
adc.vac_ab.rms_sum = 0L;
adc.vac_ab.dc_sum_bak = adc.vac_ab.dc_sum;
adc.vac_ab.dc_sum = 0L;
adc.p_total.rms_sum_bak = adc.p_total.rms_sum;
adc.p_total.rms_sum = 0L;
adc.q_total.rms_sum_bak = adc.q_total.rms_sum;
adc.q_total.rms_sum = 0L;
adc.count_rms_bak = adc.count_rms;
adc.count_rms = 0;
}

/*
 * is_rpm_startup()
 */

// Waiting to startup
if(ic_state == IC_WAIT)
{
    if(op_mode_vsi == VSI_CV)
    {
        if(phase_seq == POS_SEQ)
        {
            iac2_a.fund.s1 = S1A_POS;
            iac2_a.fund.s2 = S2A_POS;
            iac2_b.fund.s1 = S1B_POS;
            iac2_b.fund.s2 = S2B_POS;
        }
        else if(phase_seq == NEG_SEQ)
        {
            iac2_a.fund.s1 = S1A_NEG;
            iac2_a.fund.s2 = S2A_NEG;
            iac2_b.fund.s1 = S1B_NEG;
            iac2_b.fund.s2 = S2B_NEG;
        }
    }
    else if(op_mode_vsi == VSI_CI)
    {
    }
```

230
iac2_a.fund.s1 = 0;
iac2_a.fund.s2 = 0;
iac2_b.fund.s1 = 0;
iac2_b.fund.s2 = 0;
}
if(vsi_en_outputs==1)
{
ic_state = IC_PEAK;
vsi_en_outputs = 0;
//GrabRun();
}

// Looking to start on peak
if(ic_state == IC_PEAK)
{
if(op_mode_vsi == VSI_CV)
{
if(phase_seq == POS_SEQ)
{
iac2_a.fund.s1 = S1A_POS;
iac2_a.fund.s2 = S2A_POS;
iac2_b.fund.s1 = S1B_POS;
iac2_b.fund.s2 = S2B_POS;
}
else if(phase_seq == NEG_SEQ)
{
iac2_a.fund.s1 = S1A_NEG;
iac2_a.fund.s2 = S2A_NEG;
iac2_b.fund.s1 = S1B_NEG;
iac2_b.fund.s2 = S2B_NEG;
}
else if(op_mode_vsi == VSI_CI)
{
iac2_a.fund.s1 = 0;
iac2_a.fund.s2 = 0;
iac2_b.fund.s1 = 0;
iac2_b.fund.s2 = 0;
}

// Peak found
if(phase_a-PHASE_90 < phase_step)
{
VSI_ENABLE();
ic_state = IC_RUN;
}

/*
============================================================================
void isr_pwm_pi_rect()
============================================================================*/

/* Outer bus voltage loop */
vh1i.targ_fixed = (int16)(((int32)vh1i.targ_adc*(int32)ADC_VDC_SC_FIXED +
(int32)128)>> (SMALL_Q - LARGE_Q)); // Scale is LARGE_Q

// Vhi PI current loop
if (op_mode_vsi == VSI_CV)
{
vh1i.err = vh1i.targ_fixed - adc.vhi.filt_fixed;
/* vh1i.err_int = (long)vh1i.err * (long)vh1i.Ki;*/
if (vh1i.underflow)
{
if (vhi.err_int > 0)
{
    vhi.err_int_sum += vhi.err_int;
}
else if (vhi.overflow)
{
    if (vhi.err_int < 0)
    {
        vhi.err_int_sum += vhi.err_int;
    }
    else
    {
        vhi.err_int_sum += vhi.err_int;
    }
}
/*
 id_targ_fixed = ((int32)vhi.err*(int32)vhi.Kp)>>P_SHIFT_VHI;

 // check for saturation
 if (id_targ_fixed > IAC_MAX_FIXED)
 {
     id_targ_fixed = IAC_MAX_FIXED;
     vhi.overflow = 1;
     vhi.underflow = 0;
 }
 else if (id_targ_fixed < -IAC_MAX_FIXED)
 {
     id_targ_fixed = -IAC_MAX_FIXED;
     vhi.overflow = 0;
     vhi.underflow = 1;
 }
 else
 {
     vhi.overflow = 0;
     vhi.underflow = 0;
 }
*/

id_targ_fixed = ((int32)vhi.err*(int32)vhi.Kp)>>P_SHIFT_VHI;

// check for saturation
if (id_targ_fixed > IAC_MAX_FIXED)
{
    id_targ_fixed = IAC_MAX_FIXED;
    vhi.overflow = 1;
    vhi.underflow = 0;
}
else if (id_targ_fixed < -IAC_MAX_FIXED)
{
    id_targ_fixed = -IAC_MAX_FIXED;
    vhi.overflow = 0;
    vhi.underflow = 1;
}
else
{
    vhi.overflow = 0;
    vhi.underflow = 0;
}

/* void isr_pwm_waveform_calculations()
============================================================================ */

// Current reference stepping
if(flag_step)
{
    // Stepping enabled
    if((step_count >= STEP_TIMER)&&(cos_peak==1))
    {
        if(op_mode_vsi == VSI_CI)
        {
            if(step_current)
            {
                id_total_targ_fixed = (int16)(((int32)id_targ_adc*(int32)ADC_IAC_SC_FIXED +
(int32)4)>>SMALL_Q_FIXED_Q));
                id_n_total_targ_fixed = (int16)(((int32)id_n_targ_adc*(int32)ADC_IAC_SC_FIXED +
(int32)4)>>SMALL_Q_FIXED_Q));
                iq_total_targ_fixed = (int16)(((int32)iq_targ_adc*(int32)ADC_IAC_SC_FIXED +
(int32)4)>>SMALL_Q_FIXED_Q));
                iq_n_total_targ_fixed = (int16)(((int32)iq_n_targ_adc*(int32)ADC_IAC_SC_FIXED +
(int32)4)>>SMALL_Q_FIXED_Q));
                step_current = !step_current;
                step_count = 0;
                CLEAR_LED2_E13();
            }
            else
            {
        */
id_total_targ_fixed = (int16)(((int32)(id_targ_adc +
id_step_targ_adc)*(int32)ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q FIXED_Q));
id_n_total_targ_fixed = (int16)(((int32)(id_n_targ_adc +
id_n_step_targ_adc)* (int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));

iq_total_targ_fixed = (int16)(((int32)(iq_targ_adc +

iq_step_targ_adc)*(int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));

iq_n_total_targ_fixed = (int16)(((int32)(iq_n_targ_adc +

iq_n_step_targ_adc) * (int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));

step_current = !step_current;
step_count = 0;
SET_LED2_E13();
}
}
else if(op_mode_vsi == VSI CV)
{
if(step_current)
{

iq_total_targ_fixed = (int16)(((int32)iq_targ_adc*(int32)ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q FIXED_Q));

step_current = !step_current;

step_count = 0;

CLEAR_LED2_E13();
}
else
{

iq_total_targ_fixed = (int16)(((int32)iq_targ_adc +
iq_step_targ_adc) * (int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));

step_current = !step_current;

step_count = 0;

SET_LED2_E13();
}
}
}

if(op_mode_vsi == VSI CV)
{

// Always track id_targ from DC bus control for active rectifier

id_total_targ_fixed = id_targ_fixed;

// Increment step timer

step_count++;
}
else
{

// Stepping disabled

if(op_mode_vsi == VSI CI)
{

id_total_targ_fixed = (int16)(((int32)id_targ_adc*(int32)ADC_IAC_SC_FIXED + (int32)4)>>SMALL_Q FIXED_Q));

id_n_total_targ_fixed = (int16)(((int32)id_n_targ_adc*(int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));

iq_total_targ_fixed = (int16)(((int32)iq_targ_adc*(int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));

iq_n_total_targ_fixed = (int16)(((int32)iq_n_targ_adc*(int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));
}
else if(op_mode_vsi == VSI CV)
{

id_total_targ_fixed = id_targ_fixed;

iq_total_targ_fixed = (int16)(((int32)iq_targ_adc*(int32)ADC_IAC_SC_FIXED + (int32)4) >> (SMALL_Q FIXED_Q));
}

CLEAR_LED2_E13();

step_count = 0;

step_current = 0;
}
/* Phase A waveform */
index = (phase_a>>22) | 0x0001; // to access high word of 32 bit sine table
phase_offset = (phase_a&0x0007F0000L+32768)>>16;
// interpolate more accurate sin value
val_lo = sin_table[index];
val_diff = sin_table[index+2] - val_lo;
sin_val_a = val_lo + (int16)((int32)phase_offset*(int32)val_diff+(int32)64)>>7);
// interpolate more accurate cos value
val_lo = cos_table[index];
val_diff = cos_table[index+2] - val_lo;
cos_val_a = val_lo + (int16)((int32)phase_offset*(int32)val_diff+(int32)64)>>7);

// Calculate instantaneous current references
iacc_a.targ_fixed = (((int32)sin_val_a*(int32)id_total_targ_fixed+(int32)8192)>>14) +
((int32)cos_val_a*(int32)iq_total_targ_fixed+(int32)8192)>>14);

/* Phase B waveform */
phase_b = phase_a - phase_120;
index = (phase_b>>22) | 0x0001; // to access high word of 32 bit sine table
phase_offset = (phase_b&0x0007F0000L+32768)>>16;
// interpolate more accurate sin value
val_lo = sin_table[index];
val_diff = sin_table[index+2] - val_lo;
sin_val_b = val_lo + (int16)((int32)phase_offset*(int32)val_diff+(int32)64)>>7);
// interpolate more accurate cos value
val_lo = cos_table[index];
val_diff = cos_table[index+2] - val_lo;
cos_val_b = val_lo + (int16)((int32)phase_offset*(int32)val_diff+(int32)64)>>7);

// Calculate instantaneous current references
iacc_b.targ_fixed = (((int32)sin_val_b*(int32)id_total_targ_fixed+(int32)8192)>>14) +
((int32)cos_val_b*(int32)iq_total_targ_fixed+(int32)8192)>>14);

* Two stage transform to DQ frame iacc2*/
// Clarke Transform
alpha = (int32)adc.iacc_a.fixed;
beta = (((int32)INV_SQRT3*(int32)adc.iacc_a.fixed + (int32)32768))>>16) +
((int32)INV_SQRT3*(int32)adc.iacc_b.fixed + (int32)16384))>>15);

// Park Transform
iacc2_d = (((int32)alpha*(int32)cos_val_a+(int32)8192)>>14) +
((int32)beta*(int32)sin_val_a+(int32)8192)>>14);

// Park Transform for second synchronous frame of reference
iacc2_d_ss = (((int32)alpha*(int32)cos_val_a+(int32)8192)>>14) -
((int32)beta*(int32)sin_val_a+(int32)8192)>>14);

/* Two stage transform to DQ frame iacc*/
// Clarke Transform
alpha = (int32)adc.iacc_a.fixed;
beta = (((int32)INV_SQRT3*(int32)adc.iacc_a.fixed + (int32)32768))>>16) +
((int32)INV_SQRT3*(int32)adc.iacc_b.fixed + (int32)16384))>>15);

// Park Transform
iacc_d = (((int32)alpha*(int32)cos_val_a+(int32)8192)>>14) +
((int32)beta*(int32)sin_val_a+(int32)8192)>>14);

// Park Transform for second synchronous frame of reference
iacc_d_ss = (((int32)alpha*(int32)cos_val_a+(int32)8192)>>14) -
((int32)beta*(int32)sin_val_a+(int32)8192)>>14);
Appendix C  DSP Source Code for the Experimental System

iac1_q_ss = (((int32)alpha*(int32)cos_val_a+(int32)8192)>>14) +
    (((int32)beta*(int32)sin_val_a+(int32)8192)>>14);
iac1_d_ss = (((int32)beta*(int32)cos_val_a+(int32)8192)>>14) -
    (((int32)alpha*(int32)sin_val_a+(int32)8192)>>14);

    // Inv Park Transform
    alpha = (((int32)id_total_targ_fixed*(int32)sin_val_a+(int32)8192)>>14) +
            (((int32)iq_total_targ_fixed*(int32)cos_val_a+(int32)8192)>>14);
    beta = (((int32)id_total_targ_fixed*(int32)cos_val_a+(int32)8192)>>14) -
            (((int32)iq_total_targ_fixed*(int32)sin_val_a+(int32)8192)>>14);

    // Park Transform second of reference
    iq_ss_total_targ_fixed = (((int32)alpha*(int32)cos_val_a+(int32)8192)>>14) +
        (((int32)beta*(int32)sin_val_a+(int32)8192)>>14);
    id_ss_total_targ_fixed = (((int32)beta*(int32)cos_val_a+(int32)8192)>>14) -
        (((int32)alpha*(int32)sin_val_a+(int32)8192)>>14);

    // Inv Park Transform from second of reference
    alpha = -((int32)id_n_total_targ_fixed*(int32)sin_val_a+(int32)8192)>>14) +
        (((int32)iq_n_total_targ_fixed*(int32)cos_val_a+(int32)8192)>>14);
    beta = (((int32)id_n_total_targ_fixed*(int32)cos_val_a+(int32)8192)>>14) +
        (((int32)iq_n_total_targ_fixed*(int32)sin_val_a+(int32)8192)>>14);

    // Park Transform
    iq_n_ss_total_targ_fixed = (((int32)alpha*(int32)cos_val_a+(int32)8192)>>14) -
        (((int32)beta*(int32)sin_val_a+(int32)8192)>>14);
    id_n_ss_total_targ_fixed = (((int32)beta*(int32)cos_val_a+(int32)8192)>>14) +
        (((int32)alpha*(int32)sin_val_a+(int32)8192)>>14);

/* definitions
iac2_q: Positive synchronous frame, iac2 q axis
iac2_d: Positive synchronous frame, iac2 d axis
iac2_q_ss: Negative synchronous frame, iac2 q axis
iac2_d_ss: Negative synchronous frame, iac2 d axis
iac1_q: Positive synchronous frame, iac1 q axis
iac1_d: Positive synchronous frame, iac1 d axis
iac1_q_ss: Negative synchronous frame, iac1 q axis
iac1_d_ss: Negative synchronous frame, iac1 d axis

iq_total_targ_fixed: Positive synchronous frame, p reference current q axis
id_total_targ_fixed: Positive synchronous frame, p reference current d axis
iq_ss_total_targ_fixed: Negative synchronous frame, P reference current q axis
id_ss_total_targ_fixed: Negative synchronous frame, P reference current d axis

iq_n_total_targ_fixed: Negative synchronous frame, N reference current q axis
id_n_total_targ_fixed: Negative synchronous frame, N reference current d axis
iq_n_ss_total_targ_fixed: Positive synchronous frame, N reference current q axis
id_n_ss_total_targ_fixed: Positive synchronous frame, N reference current d axis*/

void isr_pwm_PI_current()
{ /* =========================================================================
    void isr_pwm_PI_current()
    // Positive synchronous frame
    // Calculate current errors
    iac2_d_err = (int32)id_total_targ_fixed + (int32)id_n_ss_total_targ_fixed - (int32)iac2_d;
    iac2_q_err = (int32)iq_total_targ_fixed + (int32)iq_n_ss_total_targ_fixed - (int32)iac2_q;

    // Proportional component
    iac2_d_prop = ((int32)iac2_d_err*(int32)Kp_PI + SMALL_Q_ROUND)>>SMALL_Q; //+1 because Kp
    iac2_q_prop = ((int32)iac2_q_err*(int32)Kp_PI + SMALL_Q_ROUND)>>SMALL_Q;

    // Anti-Windup for integrators
*/

235
if (V_Asat || V_Bsat || V_Csat)
{
    //dq frame
    iac2_d_err_int = 0;
    iac2_q_err_int = 0;
} else
{
    // dq frame
    iac2_d_err_int = iac2_d_err;
    iac2_q_err_int = iac2_q_err;
}

    // integrator
    // integrator dq frame
    iac2_d_int += ((int32)iac2_d_err_int * (int32)Ki_PI + 16384)<<(15+1);
    iac2_q_int += ((int32)iac2_q_err_int * (int32)Ki_PI + 16384)<<(15+1);

    vd = (iac2_d_prop>>PROP_SHIFT_PR) + (iac2_d_int>>INT_DISCARD);
    vq = (iac2_q_prop>>PROP_SHIFT_PR) + (iac2_q_int>>INT_DISCARD);

    // Negative synchronous frame
    // Calculate current errors
    iac2_d_ss_err = (int32)id_ss_total_targ_fixed + (int32)id_n_total_targ_fixed -
                    (int32)iac2_d_ss;
    iac2_q_ss_err = (int32)iq_ss_total_targ_fixed + (int32)iq_n_total_targ_fixed -
                    (int32)iac2_q_ss;
    // Proportional component
    iac2_d_ss_prop = ((int32)iac2_d_ss_err*(int32)Kp_PI + SMALL_Q_ROUND)>>SMALL_Q;  //scale is FIXED_Q_SCALE*2
    iac2_q_ss_prop = ((int32)iac2_q_ss_err*(int32)Kp_PI + SMALL_Q_ROUND)>>SMALL_Q;
    // Anti-Windup for integrators
    if (V_Asat || V_Bsat || V_Csat)
    {
        //dq frame
        iac2_d_ss_err_int = 0;
        iac2_q_ss_err_int = 0;
    } else
    {
        // dq frame
        iac2_d_ss_err_int = iac2_d_ss_err;
        iac2_q_ss_err_int = iac2_q_ss_err;
    }
    // integrator
    // integrator dq frame
    iac2_d_ss_int += ((int32)iac2_d_ss_err_int * (int32)Ki_PI + 16384)<<(15);
    iac2_q_ss_int += ((int32)iac2_q_ss_err_int * (int32)Ki_PI + 16384)<<(15);
    vd_ss = (iac2_d_ss_prop>>PROP_SHIFT_PR) + (iac2_d_ss_int>>INT_DISCARD);
    vq_ss = (iac2_q_ss_prop>>PROP_SHIFT_PR) + (iac2_q_ss_int>>INT_DISCARD);

    /* =========================================================================
    void isr_pwm_p_active_damping()
    ==============================================================
    // Positive synchronous frame
    */
    /* Common mode current component */
    icm_fixed = adc.iac1_a.fixed + adc.iac1_b.fixed + adc.iac1_c.fixed;
    icm.err = -icm_fixed;
    /* Inner capacitor current loop dq-frame */
    iacc_d = iac1_d - iac2_d;
    iacc_q = iac1_q - iac2_q;
    iacc_d_ss = iacl_d_ss - iac2_d_ss;
    iacc_q_ss = iacl_q_ss - iac2_q_ss;
    if (active_damping == 0)
    {
        // No AD - Don’t subtract capacitor current
        // dq-frame
        iacc_d_err = vd;
        iacc_q_err = vq;
        iacc_d_ss_err = vd_ss;
    } else
    {
Appendix C  DSP Source Code for the Experimental System

```c
iacc_q_ss_err = vq_ss;
}
else
{
    // // Run AD - Subtract capacitor current
    //dq_frame
    iacc_d_err = (int32)vd - (int32)iacc_d;
    iacc_q_err = (int32)vq - (int32)iacc_q;
    iacc_d_ss_err = (int32)vd_ss - (int32)iacc_d;
    iacc_q_ss_err = (int32)vq_ss - (int32)iacc_q;
}

////////////////////////////////////
// Positive synchronous frame
// Calculate target switching voltages dq-frame
adc_vac_d = (int16)(((int32)iacc_d_err*(int32)iacc_a.K + SMALL_Q_ROUND)>>SMALL_Q));// +1
because of devide K by 2
adc_vac_d = (int16)(((int32)adc_vac_d*(int32)PERIOD_2 + FIXED_Q_ROUND)>>FIXED_Q);

adc_vac_q = (int16)(((int32)iacc_q_err*(int32)iacc_a.K + SMALL_Q_ROUND)>>SMALL_Q));// +1
because of devide K by 2
adc_vac_q = (int16)(((int32)adc_vac_q*(int32)PERIOD_2 + FIXED_Q_ROUND)>>FIXED_Q);

// Alternative dq to abc transformation
adc_vac_a_test = (((((int32)sin_val_a*(int32)adc_vac_d+(int32)8192)>>14) +
((int32)cos_val_a*(int32)adc_vac_q+(int32)8192)>>14))+(int32)0)<<(0);
adc_vac_b_test = (((((int32)sin_val_b*(int32)adc_vac_d+(int32)8192)>>14) +
((int32)cos_val_b*(int32)adc_vac_q+(int32)8192)>>14))+(int32)0)<<(0);

////////////////////////////////////
// Negative synchronous frame
// Calculate target switching voltages dq-frame
adc_vac_d_ss = (int16)(((int32)iacc_d_err*(int32)iacc_a.K + SMALL_Q_ROUND)>>SMALL_Q));// +1
because of devide K by 2
adc_vac_d_ss = (int16)(((int32)adc_vac_d*(int32)PERIOD_2 + FIXED_Q_ROUND)>>FIXED_Q);

adc_vac_q_ss = (int16)(((int32)iacc_q_err*(int32)iacc_a.K + SMALL_Q_ROUND)>>SMALL_Q));// +1 because of devide K by 2
adc_vac_q_ss = (int16)(((int32)adc_vac_q*(int32)PERIOD_2 + FIXED_Q_ROUND)>>FIXED_Q);

/* Convert commands to ABC */
// Inv Park Transform
alpha = (((int32)adc_vac_d_ss*(int32)sin_val_a+(int32)8192)>>14) +
(((int32)adc_vac_q_ss*(int32)cos_val_a+(int32)8192)>>14);
beta = (((int32)adc_vac_d_ss*(int32)cos_val_a+(int32)8192)>>14) +
(((int32)adc_vac_q_ss*(int32)sin_val_a+(int32)8192)>>14);

// Inv Clarke Transform
adc_vac_a = alpha;
adc_vac_b = (((int32)SQRT3_ON2*(int32)beta+(int32)32768)>>16) -
(((int32)32768*(int32)alpha+(int32)32768)>>16 );

adc_vac_cm = (int16)(((int32)icm.err*(int32)icm.K + SMALL_Q_ROUND)>>SMALL_Q);
adc_vac_cm = (int16)(((int32)adc_vac_cm*(int32)PERIOD_2 + FIXED_Q_ROUND)>>FIXED_Q);
adc_vac_a_test += adc_vac_a;
adc_vac_b_test += adc_vac_b;

/* isr_pwm_modulator() */

//DC bus compensation
//compensation = mod_depth * nominal_DC_bus/real_DC_bus
if (adc.vhi.filt_fixed > MIN_VDC_COMP_FIXED)
fixed_vdc_comp = adc.vhi.filt_fixed;
```

237
Appendix C  DSP Source Code for the Experimental System

else
  fixed_vdc_comp = MIN_VDC_COMP_FIXED;

  vhi_comp_fixed = VHI_NOM_FIXED/fixed_vdc_comp;

  // Convert demanded inst. voltage to switching time
  if (op_mode_vsi != VSI_OL) // Closed loop
    { 
      t_a = -(int16)(((int32)adc_vac_a_test*(int32)vhi_comp_fixed + FIXED_Q_ROUND)>>FIXED_Q);
      t_b = -(int16)(((int32)adc_vac_b_test*(int32)vhi_comp_fixed + FIXED_Q_ROUND)>>FIXED_Q);
      t_cm = -(int16)(((int32)adc_vac_cm*(int32)vhi_comp_fixed + (int32)FIXED_Q_ROUND))>>FIXED_Q);
      t_c = -t_a - t_b;
    }
  else // Open loop
    { 
      //t_a = mod * sin * PERIOD
      t_a = ((int32)sin_val_a*(int32)mod_vsi_period+(int32)8192)>>14;
      t_b = ((int32)sin_val_b*(int32)mod_vsi_period+(int32)8192)>>14;
      t_c = -t_a - t_b;
    }

  /*
     determine offset for effective 3rd harmonic injection 
     t_off = -(max(Va,Vb,Vc)+min(Va,Vb,Vc))/2; */
  if (t_a > t_b)
    { 
      if (t_a > t_c)
        { 
          if (t_b > t_c)
            { 
              t_off = (t_b+1)>>1;
            }
          else
            
            { 
              t_off = (t_c+1)>>1;
            } 
        }
      else
        
        { 
          t_off = (t_a+1)>>1;
        } 
    }
  else
    { 
      if (t_b > t_c)
        { 
          if (t_a > t_c)
            { 
              t_off = (t_a+1)>>1;
            }
          else
            
            { 
              t_off = (t_c+1)>>1;
            } 
        }
      else
        
        { 
          t_off = (t_b+1)>>1;
        } 
    }

  // Allowance to disable space vector centering during open loop operation
  if ((op_mode_vsi != VSI_OL)||(svc_enable==1))
    { 
      // add offset into t_a, t_b and t_c
      t_a = t_a + t_off + t_cm;
      t_b = t_b + t_off + t_cm;
      t_c = t_c + t_off + t_cm;
    }

  //
// Setup initial turn on values for compare registers
if((ic_state==IC_WAIT)||(ic_state==IC_PEAK))
{
    if (op_mode_vsi == VSI_CV)
    {
        if(phase_seq == POS_SEQ)
        {
            t_a = TA_INIT_POS;
            t_b = TB_INIT_POS;
            t_c = TC_INIT_POS;
        }
        else if(phase_seq == NEG_SEQ)
        {
            t_a = TA_INIT_NEG;
            t_b = TB_INIT_NEG;
            t_c = TC_INIT_NEG;
        }
    }
    else if (op_mode_vsi == VSI_CI)
    {
        t_a = 0;
        t_b = 0;
        t_c = 0;
    }
}

// Phase A
V_Asat_prev = V_Asat;
if (t_a >= MAX_TIME)
{
    V_Asat = POS_SAT;
    EvbRegs.CMPR4 = 0;
}
else if (t_a <= -MAX_TIME)
{
    if((V_Asat==NEG_SAT)||(timer3_dir==0))
        EvbRegs.CMPR4 = PERIOD;
    else
        EvbRegs.CMPR4 = PERIOD - 1;
    V_Asat = NEG_SAT;
}
else
{
    V_Asat = NOT_SAT;
    EvbRegs.CMPR4 = PERIOD_2 - t_a;
}

// Phase B
V_Bsat_prev = V_Bsat;
if (t_b >= MAX_TIME)
{
    V_Bsat = POS_SAT;
    EvbRegs.CMPR5 = 0;
}
else if (t_b <= -MAX_TIME)
{
    if((V_Bsat==NEG_SAT)||(timer3_dir==0))
        EvbRegs.CMPR5 = PERIOD;
    else
        EvbRegs.CMPR5 = PERIOD - 1;
    V_Bsat = NEG_SAT;
}
else
{
    V_Bsat = NOT_SAT;
    EvbRegs.CMPR5 = PERIOD_2 - t_b;
}
Appendix C  DSP Source Code for the Experimental System

//Phase C
V_Csat_prev = V_Csat;
if (t_c >= MAX_TIME)
{
    V_Csat = POS_SAT;
    EvaRegs.CMPR1 = 0;
}
else if (t_c <= -MAX_TIME)
{
    if ((V_Csat==NEG_SAT)||(timer1_dir==0))
        EvaRegs.CMPR1 = PERIOD;
    else
        EvaRegs.CMPR1 = PERIOD - 1;
    V_Csat = NEG_SAT;
}
else
{
    V_Csat = NOT_SAT;
    EvaRegs.CMPR1 = PERIOD_2 - t_c;
}

/* =========================================================================
 * void isr_pwm_grab_data()
 * ------------------------------------------------------------------------ */

if (GrabRunning())
{
    grab_dec++;
    if (grab_dec > GRAB_DEC)
    {
        grab_dec = 0;
        GrabStore(0,id_n_total_targ_fixed);
        GrabStore(1,iq_n_total_targ_fixed);
        GrabStore(2,id_total_targ_fixed);
        GrabStore(3,id_total_targ_fixed);
        GrabStore(4,0);
        GrabStore(5,0);
        // GrabStore(6,sin_val_b);
        // GrabStore(7,HIGH16(zx_offset));
        // GrabStore(8,LOW16(zx_offset));
        // GrabStore(9,0);
        // GrabStore(10,0);
        GrabStep();
    }
}

// prepare for next interrupt
AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1; // clear interrupt flag
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE

// timing bit
CLEAR_LED0_E13();
} /* end isr_pwm */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
 * Handles the PDPINT interrupt caused by a gate fault.
 */

\author A.McIver
\par History:
\li 02/05/07 AM - initial creation
\*/*
 ifndef BUILD_RAM
 #pragma CODE_SECTION(isr_pdpint, "ramfuncs")
 endif
 interrupt void isr_pdpint(void)
{
VSI_FAST_STOP();
main_fault_set_int(FAULT_PDPINT);
fault_gate_flag = 1;
EvaRegs.EVAIFRA.all = BIT0;
// Acknowledge this interrupt to receive more interrupts from group 1
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
} /* end isr_pdpint */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Handles the rectifier or transformer AC over current hardware interrupt.
*
\author A.McIver
\par History:
\li 21/03/07 AM - initial creation
\li 11/11/09 AM - added Iac O/C event restart rather than trip
\li 22/12/09 AM - removed Iac O/C event restart
*/
#ifndef BUILD_RAM
#pragma CODE_SECTION(isr_over_current, "ramfuncs");
#endif
interrupt void isr_over_current(void)
{
    VSI_FAST_STOP();
    main_fault_set_int(FAULT_HW_AC_OC); // Includes Transformer overcurrent
    // Acknowledge this interrupt to receive more interrupts from group 1
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
} /* end isr_over_current */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
Handles the Vhi over voltage hardware interrupt.
*
\author A.McIver
\par History:
\li 02/05/07 AM - initial creation
*/
#ifndef BUILD_RAM
#pragma CODE_SECTION(isr_over_voltage, "ramfuncs");
#endif
interrupt void isr_over_voltage(void)
{
    VSI_FAST_STOP();
    main_fault_set_int(FAULT_HW_VHI_OV); // Acknowledge this interrupt to receive more interrupts from group 1
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
} /* end isr_over_voltage */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
 */

_VSI_State_Functions()
*/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This function initialises the ADC, the PMM, and LEDs. It resets the RMS target
target voltage to zero. It resets the target output voltage to zero and makes sure
that the soft charge relay and main contactor are open. It is followed by the
cal state.
*
\author A.McIver
\par History:
\li 22/06/05 AM - initial creation
\li 10/04/08 PM - ported from 25kVA Boost Code
\li 27/06/14 RK - negative sequence added
/*
// & st_vsi_init [style=bold]
void st_vsi_init(void)
{
    if (IS_FIRST_STATE(vsi_state))
    {
        DONE_FIRST_STATE(vsi_state);
        adc_init();
        pwm_init(); // initialises VSI stage
        variable_init();
        VSI_FAST_STOP();
        MAIN_CONTACTOR_OFF();
        SOFT_CHARGE_RELAY_OFF(); // disconnect from supply
        ENABLE_DIGOUT();
        id_targ_adc = 0;
        id_n_targ_adc = 0;
        iq_targ_adc = 0;
        iq_n_targ_adc = 0;
        id_ref_adc = 0;
        id_n_ref_adc = 0;
        iq_ref_adc = 0;
        iq_n_ref_adc = 0;
        active_damping = 1;
        svc_enable = 1;
        ic_state = IC_WAIT;
        wd_timer[WD_CHARGE] = START_DELAY;  // msec
    }
    if (wd_timer[WD_CHARGE] == 0)
    {
        cal_complete = 1;
        // & st_vsi_init -> st_vsi_cal
        NEXT_STATE(vsi_state,st_vsi_cal);
    }
} /* end st_vsi_init */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
The current inputs from the LEMs have significant offsets due to the tolerances between the LEMs and the op amp inputs. This function waits for the average measurements to take place and then calculates a base dc offset to correct for the offsets. This avoids a DC offset in the DC output current.

\author {A. McIver}
\par History:
\li 09/08/07 AM - initial creation
\li 10/04/08 PM - Ported from 25kVA Boost code
\li 28/05/09 AM - changed next state from charging to vin uv
/*
void st_vsi_cal(void)
{
    if (IS_FIRST_STATE(vsi_state))
    {
        DONE_FIRST_STATE(vsi_state);
        wd_timer[WD_CHARGE] = CAL_DELAY;
    }
    if (wd_timer[WD_CHARGE] == 0)
    {
        cal_complete = 1;
        // & st_vsi_cal -> st_vsi_vin_uv
        NEXT_STATE(vsi_state,st_vsi_vin_uv);
    }
} /* end st_vsi_cal */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
On start up and when recovering from a fault condition, the state machine waits in this state until the input voltage exceeds a minimum ac voltage. A state change will also be triggered by a fault code.

```c
void st_vsi_vin_uv(void)
{
    if (IS_FIRST_STATE(vsi_state))
    {
        DONE_FIRST_STATE(vsi_state);
        VSI_FAST_STOP();
        MAIN_CONTACTOR_OFF();
        SOFT_CHARGE_RELAY_OFF(); // disconnect from supply
        puts_COM0(" l ");
    }
    // check for faults
    if ( (main_fault_get_reported()&FAULT_ST_VIN_UV) != 0 )
    {
        //& st_vsi_vin_uv -> st_vsi_fault [style=dotted]
        NEXT_STATE(vsi_state,st_vsi_fault);
        return;
    }
    if ( (adc.vac_ac.real >= VAC_MIN_CLOSE)
        && (adc.vac_bc.real >= VAC_MIN_CLOSE) )
    {
        main_fault_clear(~FAULT_ST_VIN_UV); // clear ignored faults
        //& st_vsi_vin_uv -> st_vsi_cal_iac
        NEXT_STATE(vsi_state,st_vsi_cal_iac);
    }
} /* end st_rect_vin_uv */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
The current inputs from the LEMs have significant offsets due to the tolerance variations between the LEMs and the op amp inputs. This function waits for one second of rms measurements to take place while allows the dc offset in the AC currents to be measured.

```
void st_vsi_seq(void)
{
    if (IS_FIRST_STATE(vsi_state))
    {
        DONE_FIRST_STATE(vsi_state);
        seq_state = SEQ_INIT;
    }

    // check for faults
    if ((main_fault_get_reported()&FAULT_FATAL) != 0)
    {
        //& st_vsi_seq -> st_vsi_fault [style=dotted]
        NEXT_STATE(vsi_state, st_vsi_fault);
        return;
    }

    switch (seq_state)
    {
    case SEQ_INIT:
        seq_count = 0;
        seq_pos_count = 0;
        puts_COM0("seq ");
        seq_state = SEQ_WAIT_NEG;
        break;
    case SEQ_WAIT_NEG:
        if (adc.vac_bc.filt < 0)
        {
            seq_state = SEQ_WAIT_ZX;
        }
        break;
    case SEQ_WAIT_ZX:
        if (adc.vac_bc.filt > 0)
        {
            if (adc.vac_ac.filt < 0)
            {
                puts_COM0("-");
            }
        }
    }


```c
else
{
    seq_pos_count++;
    puts_COM0("+ ");
}
seq_count++;
if (seq_count >= 10)
{
    seq_state = SEQ_DONE;
    if (seq_pos_count >= 10)
    {
        phase_120 = PHASE_120_POS;
        zx_offset = ZX_OFFSET_POS;
        phase_seq = POS_SEQ;
    }
    else if (seq_pos_count == 0)
    {
        phase_120 = PHASE_120_NEG;
        zx_offset = ZX_OFFSET_NEG;
        phase_seq = NEG_SEQ;
    }
    else
    {
        seq_state = SEQ_INIT;
    }
}
else
{
    wd_timer[WD_CHARGE] = SEQ_DELAY;
    seq_state = SEQ_WAIT;
}
}
break;
case SEQ_WAIT:
if (wd_timer[WD_CHARGE] == 0)
{
    seq_state = SEQ_WAIT_NEG;
}
break;
case SEQ_DONE:
    //& st_vsi_seq -&gt; st_vsi_charging
    NEXT_STATE(vsi_state,st_vsi_charging);
    break;
}
} /* end st_vsi_seq */

/**
 * This state closes the soft charge relay to charge the input DC bus before the
 * main contactor is closed. The end of charge is defined as after 10 seconds of
 * charging or when the DC bus voltage is within 20V of the AC input line to line
 * voltage. If the bus isn't charged after 10 seconds there is a charging fault.
 */

void st_vsi_charging(void)
{
    double
        vac_max,
        vdc_lim;
    if (IS_FIRST_STATE(vsi_state))
    {
        DONE_FIRST_STATE(vsi_state);
    }
```
DSP Source Code for the Experimental System

MAIN_CONTACTOR_OFF();
SOFT_CHARGE_RELAY_ON(); // close soft charge contactor
wd_timer[WD_CHARGE] = SOFT_CHARGE_TIME; // msec
puts_COM0("c");
}
// check for faults
if ((main_fault_get_reported() & FAULT_FATAL) != 0)
{
  //& st_vsi_charging -> st_vsi_fault [style=dotted]
  NEXT_STATE(vsi_state, st_vsi_fault);
  return;
}

// find maximum line to line AC input
vac_max = adc.vac_bc.real;
if (adc.vac_bc.real > vac_max)
{
  vac_max = adc.vac_bc.real;
}
vdc_lim = vac_max*1.35 - 20.0;

// test for acceptable DC bus voltage after charging
if (adc.vhi.real > vdc_lim)
  // if (adc.vhi.real > 10)
  {
    //& st_vsi_charging -> st_vsi_close_contactor
    NEXT_STATE(vsi_state, st_vsi_close_contactor);
  }
  // end of charge delay without getting to an acceptable voltage
  if (wd_timer[WD_CHARGE] == 0)
  {
    main_fault_set(FAULT_CHARGE);
    //& st_vsi_charging -> st_vsi_fault [style=dotted]
    NEXT_STATE(vsi_state, st_vsi_fault);
    
  } /* end st_rect_charging */

This state closes the main contactor once the input bus is charged. It then waits for the contactor to close properly before continuing on. The soft charge relay is opened.

\author A.McIver
\par History:
\li 01/06/07 AM - initial creation
*/

void st_vsi_close_contactor(void)
{
  if (IS_FIRST_STATE(vsi_state))
  {
    DONE_FIRST_STATE(vsi_state);
    MAIN_CONTACTOR_ON(); // close main contactor
    SOFT_CHARGE_RELAY_ON();
    wd_timer[WD_CHARGE] = 500; // msec
    puts_COM0("m");
  }
  // check for faults
  if ((main_fault_get_reported() & FAULT_FATAL) != 0)
  {
    //& st_vsi_close_contactor -> st_vsi_fault [style=dotted]
    NEXT_STATE(vsi_state, st_vsi_fault);
    return;
  }
  // end of contactor closing delay

246
if (wd_timer[WD_CHARGE] == 0)
{
    SOFT_CHARGE_RELAY_OFF(); // open soft charge relay
    //& st_vsi_close_contactor -> st_vsi_stop
    NEXT_STATE(vsi_state, st_vsi_stop);
    return;
}
} /* end st_vsi_close_contactor */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This is the state where the VSI is stopped. There is no switching. In its automatic production configuration, the operation waits in this state for 1 second after the DC bus volts reach the starting value before moving to the ramp up state. During the wait, this state checks whether any faults have been detected and if so, moves to the fault state.

\author A.McIver
\par History:
\li 22/06/05 AM - initial creation
\li 27/06/14 RK - negative sequence added */

//& st_vsi_stop [peripheries=2]
void st_vsi_stop(void)
{
    if (IS_FIRST_STATE(vsi_state))
    {
        DONE_FIRST_STATE(vsi_state);
        VSI_FAST_STOP(); // turn off outputs
        id_targ_adc = 0;
        id_n_targ_adc = 0;
        iq_targ_adc = 0;
        iq_n_targ_adc = 0;
        puts_COM0("s");
    }

    // check for faults
    if ((main_fault_get_reported()&FAULT_FATAL) != 0)
    {
        //& st_vsi_stop -> st_vsi_fault [style=dotted]
        NEXT_STATE(vsi_state, st_vsi_fault);
        return;
    }
    if ( (adc.vac_bc.real < VAC_MIN_CLOSE) || (adc.vac_ac.real < VAC_MIN_CLOSE) )
    {
        //& st_vsi_stop -> st_vsi_vin_uv
        NEXT_STATE(vsi_state, st_vsi_vin_uv);
        return;
    }
    // check for stop signal or loss of ZK sync
    if ( in_sync == 0 )
    {
        //& st_vsi_ramp_vhi -> st_vsi_vin_uv
        NEXT_STATE(vsi_state, st_vsi_vin_uv);
        return;
    }
    // check voltage is above limits AND contactor is reporting as closed
    if ( (adc.vhi.filt > VSI_VDC_START) && IS_DIGIN1() )
    {
        if (is_vsi_switching != 0)
        {
            puts_COM0("1");
            //& st_vsi_stop -> st_vsi_start
            NEXT_STATE(vsi_state, st_vsi_start);
        }
    }
}
DSP Source Code for the Experimental System

// keep target following actual bus voltage
vhi.targ_adc = adc.vhi.filt;
} /* end st_vsi_stop */

/**
This function starts the VSI switching before initiating the ramp for the particular mode of operation.

The resonant integrator variables are initialised so that when the rectifier starts switching, the voltage produced matches the input voltage so there is no transient current. This also requires that the switching is enabled at the A phase peak and that the compare registers have the right values preloaded before the switching is enabled.

void st_vsi_start(void)
{
if (IS_FIRST_STATE(vsi_state))
{
    DONE_FIRST_STATE(vsi_state);
    puts_COM0("S");
}
if (in_sync != 0)
{
    // Reset variables
    step_current = 0;
    flag_step = 0;
    vhi.err_int_sum = 0L;
    iac2_a.fund.s1 = 0;
    iac2_a.fund.s2 = 0;
    iac2_b.fund.s1 = 0;
    iac2_b.fund.s2 = 0;
    id_total_targ_fixed = 0;
    id_n_total_targ_fixed = 0;
    iq_total_targ_fixed = 0;
    iq_n_total_targ_fixed = 0;
    iac2_d_int = 0;
    iac2_q_int = 0;
    iac2_d_ss_int = 0;
    iac2_q_ss_int = 0;
    
    if ( (op_mode_vsi == VSI_CV) || (op_mode_vsi == VSI_CI) )
    {
        vhi.targ_adc = adc.vhi.filt;
        //& st_vsi_start -> st_vsi_ramp_vhi
        NEXT_STATE(vsi_state,st_vsi_ramp_vhi);
    }
    else if (op_mode_vsi == VSI_OL)
    {
        //& st_vsi_start -> st_vsi_ramp_ol
        NEXT_STATE(vsi_state,st_vsi_ramp_ol);
    }
    // Can turn on outputs
    vsi_en_outputs = 1;
}
else
{
    //& st_vsi_ramp_ol -> st_vsi_vin_uv
```c
NEXT_STATE(vsi_state, st_vsi_vin_uv);
return;
}) /* end st_vsi_start */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This function starts the VSI switching and ramps the target AC current up to the reference AC current. This ramp rate is determined by the step size STEP_IAC and the calling frequency of this state machine which is assumed to be 1msec.

If a fault is detected the next state is the fault state. If the VSI is stopped or there is a loss of synchronisation then the next state is the Vin under voltage state. Otherwise, once the target reaches the reference, the next state is the run state.

//& st_vsi_ramp_ol -> st_vsi_fault [style=dotted]
NEXT_STATE(vsi_state, st_vsi_fault);
return;
}

// check for stop signal or loss of ZX sync
if ( (is_vsi_switching == 0) || (in_sync == 0) )
{
    //& st_vsi_ramp_ol -> st_vsi_vin_uv
    NEXT_STATE(vsi_state, st_vsi_vin_uv);
    return;
}

// check for target reached
else
{
    /* This section should ramp modulation depth */
    //& st_vsi_ramp_ol -> st_vsi_run
    NEXT_STATE(vsi_state, st_vsi_run);
}
}) /* end st_vsi_ramp_ol */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This function ramps the target output voltage from the initial bus voltage up to the reference bus voltage in voltage control mode.

This ramp rate is determined by the step size STEP_VHI and the calling frequency of this state machine which is assumed to be 1msec.

If a fault is detected the next state is the fault state. If the VSI is stopped or there is a loss of synchronisation then the next state is the Vin under voltage state. Otherwise, once the target reaches the reference, the next state is the run state.

//& st_vsi_ramp_ol -> st_vsiFault [style=dotted]
NEXT_STATE(vsi_state, st_vsiFault);
return;
}

// check for faults
if ( (mainFaultGetReported()&FAULT_FATAL) != 0 )
{
    //& st_vsi_ramp_ol -> st_vsiFault [style=dotted]
    NEXT_STATE(vsi_state, st_vsi_fault);
    return;
}

// check for stop signal or loss of ZX sync
if ( (is_vsi_switching == 0) || (in_sync == 0) )
{
    //& st_vsi_ramp_ol -> st_vsi_vin_uv
    NEXT_STATE(vsi_state, st_vsi_vin_uv);
    return;
}

// check for target reached
else
{
    /* This section should ramp modulation depth */
    //& st_vsi_ramp_ol -> st_vsi_run
    NEXT_STATE(vsi_state, st_vsi_run);
}
}) /* end st_vsi_ramp_ol */
```
Appendix C  DSP Source Code for the Experimental System

under voltage state. Otherwise, once the target reaches the reference, the next state is the run state.

\author A.McIver
\par History:
\li 15/02/10 AM - initial creation
\li 27/06/14 RK - negative sequence added

```c
void st_vsi_ramp_cl(void)
{
if (IS_FIRST_STATE(vsi_state))
{
    DONE_FIRST_STATE(vsi_state);
    puts_COM0("cl");
}

    // check for faults
    if ((main_fault_get_reported()&FAULT_FATAL) != 0)
    {
        //& st_vsi_ramp_vhi -> st_vsi_fault [style=dotted]
        NEXT_STATE(vsi_state,st_vsi_fault);
        return;
    }

    // check for stop signal or loss of ZX sync
    if ( (is_vsi_switching == 0)
    || (in_syn == 0) )
    {
        //& st_vsi_ramp_vhi -> st_vsi_vin_uv
        NEXT_STATE(vsi_state,st_vsi_vin_uv);
        return;
    }

    // check for target reached
    if (op_mode_vsi == VSI_CV)
    {
        if ( (vhi.targ_adc > vhi.ref_adc - STEP_VHI_ADC)
        && (vhi.targ_adc < vhi.ref_adc + STEP_VHI_ADC) )
        {
            vhi.targ_adc = vhi.ref_adc;
            //& st_vsi_ramp_vhi -> st_vsi_run
            NEXT_STATE(vsi_state,st_vsi_run);
        }

        // ramp reference towards target
        else if (vhi.targ_adc < vhi.ref_adc)
        {
            vhi.targ_adc += STEP_VHI_ADC;
        }
        else // vhi.targ_adc > vhi.ref_adc
        {
            vhi.targ_adc -= STEP_VHI_ADC;
        }
    }

    // don't bother ramping currents
    if (op_mode_vsi == VSI_CI)
    {
        id_targ_adc = id_ref_adc;
        id_n_targ_adc = id_n_ref_adc;
        iq_targ_adc = iq_ref_adc;
        iq_n_targ_adc = iq_n_ref_adc;
        NEXT_STATE(vsi_state,st_vsi_run);
    }
}
/* end st_vsi_ramp_vcl */
```

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
In this state, the VSI is running and following the reference. If the VSI is
stopped or the DC bus volts drop too low then the next state is the stop state. If a fault is detected the next state is the fault state.

/* & st_rect_run [peripheries=2]
void st_vsi_run(void)
{
    if (IS_FIRST_STATE(vsi_state))
    {
        DONE_FIRST_STATE(vsi_state);
        puts_COM0("r ");
    }

    // check for faults
    if ((main_fault_get_reported()&FAULT_FATAL) != 0)
    {
        //& st_vsi_run -> st_vsi fault [style=dotted]
        NEXT_STATE(vsi_state,st_vsi_fault);
        return;
    }
    // check for stop signal or loss of ZX sync
    if ( (is_vsi_switching == 0)
         || (in_sync == 0) )
    {
        //& st_vsi_run -> st_vsi_vin_uv
        NEXT_STATE(vsi_state,st_vsi_vin_uv);
        return;
    }

    // Pass references to interrupt as targets
    if (op_mode_vsi == VSI_CV)
    {
        vhi.targ_adc = vhi.ref_adc;
        iq_targ_adc = iq_ref_adc;
        iq_step_targ_adc = iq_step_ref_adc;
    }
    else if (op_mode_vsi == VSI_CI)
    {
        id_targ_adc = id_ref_adc;
        id_n_targ_adc = id_n_ref_adc;
        iq_targ_adc = iq_ref_adc;
        iq_n_targ_adc = iq_n_ref_adc;
        id_step_targ_adc = id_step_ref_adc;
        id_n_step_targ_adc = id_n_step_ref_adc;
        iq_step_targ_adc = iq_step_ref_adc;
        iq_n_step_targ_adc = iq_n_step_ref_adc;
    }
} /* end st_vsi_run */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This state is entered when a fault condition has been detected. It is not left until all the faults have been cleared by the background fault handling system.

/* author A.McIver */
/* History: */
/* \
| 03/11/05 AM - initial creation */
void st_vsi_fault(void)
Appendix C  DSP Source Code for the Experimental System

{ 
 if (IS_FIRST_STATE(vsi_state))
 { 
 DONE_FIRST_STATE(vsi_state);
 VSI_FAST_STOP(); // turn off outputs
 MAIN_CONTACTOR_OFF();
 SOFT_CHARGE_RELAY_OFF(); // disconnect from supply
 puts_COM0("f");
 }

if ((main_fault_get_reported()&FAULT_FATAL) == 0)
{
  //& st_vsi_fault
  NEXT_STATE(vsi_state,st_vsi_vin_uv);
}
} /* end st_vsi_fault */

/* =================================================================================
__Local_Functions()
============================================================================ */
/* * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * */
/**
This function checks for slow faults caused by inputs and outputs exceeding
thresholds.

For the fuse failure faults: When one of the phases is lost, two of the line
to line voltages collapse. An overly large difference between the maximum and
minimum line to line voltages causes a fuse failure fault. The maximum line to
line voltage is the two good phases while the other one is the bad phase.

Wiring on the Board is
#- VA to the neutral input,
#- VB to the A input and
#- VC to the B input.

\author A.McIver
\par History:
\li 01/05/08 AM - initial creation
\li 02/05/08 AM - added Iout trip
\li 06/05/10 AM - added fuse failure detection
*/
#define PH_BA 0
#define PH_CA 1
#define PH_BC 2
#define FUSE_FAIL_COUNT_MAX 5

void check_voltage_limits(void)
{
  double
  vac_min,
  vac_max,
  mid_ratio;
  Uint16
  max_phase = PH_BA;
  static Uint16
  fuse_fail_count = 0;

  // check for input voltage within limits
  if ( (adc.vac_ac.real > TRIP_VAC_MAX)
      || (adc.vac_bc.real > TRIP_VAC_MAX) )
  {
    VSI_FAST_STOP();
    main_fault_set(FAULT_SW_OVIN);
  }
  if ( (adc.vac_ac.real < TRIP_VAC_MIN)
|| (adc.vac_bc.real < TRIP_VAC_MIN) }
{
VSI_FAST_STOP();
main_fault_set(FAULT_SW_UVIN);
}
// check for loss of input phase
vac_min = adc.vac_ac.real;
if (adc.vac_bc.real < vac_min)
vac_min = adc.vac_bc.real;

vac_max = adc.vac_ac.real;
max_phase = PH_CA;
if (adc.vac_bc.real > vac_max)
{
vac_max = adc.vac_bc.real;
max_phase = PH_BC;
}

if ( (vac_max > AC_DIFF_MIN) 
&& (vac_min < AC_DIFF_LIM*vac_max) )
{
if (fuse_fail_count < FUSE_FAIL_COUNT_MAX)
{
fuse_fail_count++;
}
else
{
VSI_FAST_STOP();
GrabRun();
switch (max_phase)
{
    case PH_BA:
        main_fault_set(FAULT_C_PH_FUSE);
        break;
    case PH_CA:
        main_fault_set(FAULT_B_PH_FUSE);
        break;
    case PH_BC:
        main_fault_set(FAULT_A_PH_FUSE);
        break;
}
}
else if (fuse_fail_count > 0)
{
fuse_fail_count--;
}
if (adc.vhi.real > 50.0)
{
    mid_ratio = adc.vhi_mid.real / adc.vhi.real;
    if ( (mid_ratio < 0.3) || (mid_ratio > 0.7) )
    {
        main_fault_set(FAULT_VDC_BUS);
    }
}
} /* end check_voltage_limits */

/* Initialisation_Fuctions()

============================================================================ */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

This function initialises variables for use in the interrupt
author S.Parker
par History:
li 25/06/126 PM - initial creation
Appendix C  DSP Source Code for the Experimental System

*/
void variable_init(void)
{
    /* FOR 10kHz */
    // Initialise Resonant coefficients
    // Fundamental
    iac2_a.fund.beta0 = 2209;
    iac2_a.fund.beta1 = 141267;
    iac2_a.fund.beta2 = -6221;
    iac2_a.fund.alpha0 = 524288;
    iac2_a.fund.alpha1 = 4140;
    iac2_a.fund.alpha2 = 132465;
    iac2_b.fund.alpha0 = iac2_a.fund.alpha0;
    iac2_b.fund.alpha1 = iac2_a.fund.alpha1;
    iac2_b.fund.alpha2 = iac2_a.fund.alpha2;
    iac2_b.fund.beta0 = iac2_a.fund.beta0;
    iac2_b.fund.beta1 = iac2_a.fund.beta1;
    iac2_b.fund.beta2 = iac2_a.fund.beta2;

    // 5th
    iac2_a.fif.beta0 = 411;//2053;
    iac2_a.fif.beta1 = 25899;//129493;
    iac2_a.fif.beta2 = -24401;//-122005;
    iac2_a.fif.alpha0 = 524288;
    iac2_a.fif.alpha1 = 103437;
    iac2_a.fif.alpha2 = 3309988;
    iac2_b.fif.alpha0 = iac2_a.fif.alpha0;
    iac2_b.fif.alpha1 = iac2_a.fif.alpha1;
    iac2_b.fif.alpha2 = iac2_a.fif.alpha2;
    iac2_b.fif.beta0 = iac2_a.fif.beta0;
    iac2_b.fif.beta1 = iac2_a.fif.beta1;
    iac2_b.fif.beta2 = iac2_a.fif.beta2;

    // 7th
    iac2_a.sev.beta0 = 291;//2039;
    iac2_a.sev.beta1 = 18263;//127839;
    iac2_a.sev.beta2 = -24389;//-170723;
    iac2_a.sev.alpha0 = 524288;
    iac2_a.sev.alpha1 = 202637;
    iac2_a.sev.alpha2 = 6484375;
    iac2_b.sev.alpha0 = iac2_a.sev.alpha0;
    iac2_b.sev.alpha1 = iac2_a.sev.alpha1;
    iac2_b.sev.alpha2 = iac2_a.sev.alpha2;
    iac2_b.sev.beta0 = iac2_a.sev.beta0;
    iac2_b.sev.beta1 = iac2_a.sev.beta1;
    iac2_b.sev.beta2 = iac2_a.sev.beta2;

    // 11th
    iac2_a.elev.beta0 = 154;//1699;
    iac2_a.elev.beta1 = 9246;//101709;
    iac2_a.elev.beta2 = -40840;//-449245;
    iac2_a.elev.alpha0 = 524288;
    iac2_a.elev.alpha1 = 499648;
    iac2_a.elev.alpha2 = 13988739;
    iac2_b.elev.alpha0 = iac2_a.elev.alpha0;
    iac2_b.elev.alpha1 = iac2_a.elev.alpha1;
    iac2_b.elev.alpha2 = iac2_a.elev.alpha2;
    iac2_b.elev.beta0 = iac2_a.elev.beta0;
    iac2_b.elev.beta1 = iac2_a.elev.beta1;
    iac2_b.elev.beta2 = iac2_a.elev.beta2;

    // 13th
    iac2_a.teen3.beta0 = 120;//1565;
    iac2_a.teen3.beta1 = 7101;//92312;
Appendix C  DSP Source Code for the Experimental System

iac2_a.teen3.beta2 = -38689; // -502957;
iac2_a.teen3.alpha0 = 524288;
iac2_a.teen3.alpha1 = 697167;
iac2_a.teen3.alpha2 = 2230937;

iac2_b.teen3.alpha0 = iac2_a.teen3.alpha0;
iac2_b.teen3.alpha1 = iac2_a.teen3.alpha1;
iac2_b.teen3.alpha2 = iac2_a.teen3.alpha2;
iac2_b.teen3.beta0 = iac2_a.teen3.beta0;
iac2_b.teen3.beta1 = iac2_a.teen3.beta1;
iac2_b.teen3.beta2 = iac2_a.teen3.beta2;
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
 * This function initialises the VSI switching and hardware interrupts. It:
 * \li Sets the output pins on the DSP to PWM mode
 * \li Sets up timer 1 for the PWM carrier
 * \li Sets up timer 2 to trigger the ADC conversions
 * \li Sets up hardware over current interrupt XINT1
 * \li Sets up hardware over Vdc interrupt on XINT2
 * \li Maps the interrupt vectors to the appropriate ISRs
 * \li Enables the individual interrupts
 * \li Starts the timers
 * \li Enables the gate drivers
 */

void
pwm_init(
void
)
{
// Disable CPU interrupts
DINT;

RESET_GATES();

EvaRegs.GPTCONA.all = 0x0000;
EvaRegs.EVAIMRA.all = 0x0000;
EvaRegs.EVAIFRA.all = BIT0;
EvaRegs.COMCONA.all = 0x0000;
EvaRegs.ACTRA.all = 0x0000;
EvbRegs.GPTCONB.all = 0x0000;
EvbRegs.EVBIMRA.all = 0x0000;
EvbRegs.EVBIFRA.all = BIT0;
EvbRegs.COMCONB.all = 0x0000;
EvbRegs.ACTRB.all = 0x0000;

EvaRegs.T1CON.all = 0x0000;
EvaRegs.T2CON.all = 0x0000;
EvaRegs.T1CNT = 0x0000;
EvaRegs.T2CNT = 0x0000;

EvbRegs.T3CON.all = 0x0000;
EvbRegs.T4CON.all = 0x0000;
EvbRegs.T3CNT = 0x0000;
EvbRegs.T4CNT = 0x0000;

EvaRegs.CAPCONA.all = 0x0000;
EvaRegs.CAPFIFOA.all = 0x0000;
EALLOW;
// set up VSI outputs
GpioMuxRegs.GPAMUX.bit.PWM1_GPIOA0 = 1; // enable PWM1 pin - Phase A
GpioMuxRegs.GPAMUX.bit.PWM2_GPIOA1 = 1; // enable PWM2 pin - Phase A
GpioMuxRegs.GPAMUX.bit.CAP1_GPIOA8 = 1; // enable CAP1 pin
GpioMuxRegs.GPFQUAL.bit.QUALPRD = 6; // 500ns qualification period
GpioMuxRegs.GPAMUX.bit.PWM7_GPIOB0 = 1; // enable PWM7 pin - Phase C
GpioMuxRegs.GPAMUX.bit.PWM8_GPIOB1 = 1; // enable PWM8 pin - Phase C
GpioMuxRegs.GPAMUX.bit.PWM9_GPIOB2 = 1; // enable PWM9 pin
GpioMuxRegs.GPAMUX.bit.PWM10_GPIOB3 = 1; // enable PWM10 pin

EDIS;

// Timer 1 produces the PWM carrier for phase C
EvaRegs.T1PR = PERIOD;
// Timer 2 produces the ADC trigger
EvaRegs.T2PR = PERIOD - 1;
// Timer 3 produces the PWM carrier for phase A+B
EvbRegs.T3PR = PERIOD;

// Reset CMPRx
EvaRegs.CMPR1 = PERIOD_2; // 50% duty cycle for VSI
EvbRegs.CMPR4 = PERIOD_2;
EvbRegs.CMPR5 = PERIOD_2;

/* DBT DBTPS time
  9 2 0.24
  9 3 0.48
  9 4 0.96
  7 5 1.49
  9 5 1.92 <= used for Stewart's NY 5kVA
  12 3 0.64
  12 5 2.56
  15 5 3.2
  
  deadtime = 2^DBTPS * DBT / clock freq (150M)
*/

// Deadband Phase C
EvaRegs.DBTCONA.bit.DBT = 9;
EvaRegs.DBTCONA.bit.DBTPS = 4;
EvaRegs.DBTCONA.bit.EDBT1 = 1;

// Deadband Phase A and B
EvbRegs.DBTCONB.bit.DBT = 9;
EvbRegs.DBTCONB.bit.DBTPS = 4;
EvbRegs.DBTCONB.bit.EDBT1 = 1;
EvbRegs.DBTCONB.bit.EDBT2 = 1;

// Setup and load GPTCONA
EvaRegs.GPTCONA.bit.T2TOADC = 1; // underflow int flag starts ADC
EvaRegs.GPTCONA.bit.TCMPOE = 1; // Timer 1&2 compare output enable

// Setup and load COMCONA
EvaRegs.COMCONA.bit.ACTRLD = 2; // reload ACTR on immediately
EvaRegs.COMCONA.bit.SVENABLE = 0; // disable space vector PWM
EvaRegs.COMCONA.bit.CLD = 1; // reload on underflow or period match
EvaRegs.COMCONA.bit.FCOMPOE = 1; // full compare enable
EvaRegs.COMCONA.bit.CENABLE = 1; // enable compare operation

// Setup and load COMCONB
EvbRegs.COMCONB.bit.ACTRLD = 2; // reload ACTR immediately
EvbRegs.COMCONB.bit.SVENABLE = 0; // disable space vector PWM
EvbRegs.COMCONB.bit.CLD = 1; // reload on underflow or period match
EvbRegs.COMCONB.bit.FCOMPOE = 1; // full compare enable
EvbRegs.COMCONB.bit.CENABLE = 1; // enable compare operation

// Capture 1 gets Timer 2 on rising edge
EvaRegs.CAPCONA.bit.CAPRES = 1; // Release from reset
EvaRegs.CAPCONA.bit.CAP1EDGE = 1; // Rising edge on capture 1
EvaRegs.CAPCONA.bit.CAP12EN = 1; // Enable captures 1 and 2
Appendix C  DSP Source Code for the Experimental System

EvaRegs.CAPCONA.bit.CAP12TSEL = 0; // Based on Timer 2
EvaRegs.EVAIMRA.all = 0; // disable all interrupts
EvaRegs.EVAIMRA.bit.PDPINTA = 1; // enable interrupt on pdpinta
EvaRegs.EVAIFRA.bit.PDPINTA = 1; // clear interrupt flag

// Configure XINT1 for rising edge triggered interrupt (over current)
XIntrruptRegs.XINT1CR.bit.POLARITY = 1;
XIntrruptRegs.XINT1CR.bit.ENABLE = 1;

// Configure XINT2 for rising edge triggered interrupt (Vdc over voltage)
XIntrruptRegs.XINT2CR.bit.POLARITY = 1;
XIntrruptRegs.XINT2CR.bit.ENABLE = 1;

// Map interrupt vectors to ISR functions
EALLOW;
PieVectTable.PDPINTA = &isr_pdpinta;
PieVectTable.ADCINT = &isr_pwm;
PieVectTable.XINT1 = &isr_over_current;
PieVectTable.XINT2 = &isr_over_voltage;
EDIS;

// Enable PDPINTA in PIE: Group 1 interrupt 1
PieCtrlRegs.PIEIER1.bit.INTx1 = 1;

// Enable XINT1 in PIE: Group 1 interrupt 4
PieCtrlRegs.PIEIER1.bit.INTx4 = 1;

// Enable XINT2 in PIE: Group 1 interrupt 5
PieCtrlRegs.PIEIER1.bit.INTx5 = 1;

// Enable ADC interrupt in PIE: Group 1 interrupt 6
PieCtrlRegs.PIEIER1.bit.INTx6 = 1;

IER |= M_INT1; // Enable CPU Interrupts 1 (for int2 add |M_INT2)
EINT;

AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1; // clear interrupt flag
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE

/* Setup and load T3CON to start operation */
EvbRegs.T3CON.bit.TMODE = 1; // continous up/down count mode
EvbRegs.T3CON.bit.TPS = 0; // input clock prescaler
EvbRegs.T3CON.bit.TECMPR = 1; // enable time compare
EvbRegs.T3CNT = 0x0001; // preload to sync with timer 1

/* Setup and load T2CON to start with timer 1 */
EvaRegs.T2CON.bit.TMODE = 2; // continuous up count mode
EvaRegs.T2CON.bit.TPS = 0; // input clock prescaler
EvaRegs.T2CON.bit.T2SWT1 = 1; // enable timer from timer 1 enable

/* Setup and load T1CON to start operation */
EvaRegs.T1CON.bit.TMODE = 1; // continuous up/down count mode
EvaRegs.T1CON.bit.TPS = 0; // input clock prescaler
EvaRegs.T1CON.bit.TECMPR = 1; // enable time compare
EvaRegs.T1CON.bit.TENABLE = 1; // enable timer

// enable timer 3 as close as possible to timer 1
EvbRegs.T3CON.bit.TENABLE = 1; // enable timer3

ENABLE_GATES();

} /* end pwm_init */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/**
This functions initialises the ADC unit to:
\li Trigger a conversion sequence from timer 2 underflow
\li Convert the appropriate ADC channels
*/
Result registers as follows:
- E13 VSI Board Structure
  
<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCRESULT0</td>
<td>ADCINA0 lac2.a</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT1</td>
<td>ADCINB0 vhi2 (VDC3)</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT2</td>
<td>ADCINA1 lac2.b</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT3</td>
<td>ADCINB1 vac_bc</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT4</td>
<td>ADCINA2 lac1.c</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT5</td>
<td>ADCINB2 vac_ac</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT6</td>
<td>ADCINA3 lac1.a</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT7</td>
<td>ADCINB3 vhi_mid (VDC2)</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT8</td>
<td>ADCINA4 lac1_b</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT9</td>
<td>ADCINB4 vhi (VDC1)</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT10</td>
<td>ADCINA5 unused</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT11</td>
<td>ADCINB5 unused</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT12</td>
<td>ADCINA6 yHA</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT13</td>
<td>ADCINB6 yHB</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT14</td>
<td>ADCINA7 yLA</td>
<td>0x0007</td>
</tr>
<tr>
<td>ADCRESULT15</td>
<td>ADCINB7 yLB</td>
<td>0x0007</td>
</tr>
</tbody>
</table>

/* * * * * * * * * * * * * * * * * * * * *
See spra989a.pdf for calibration details
*/

void adc_init(void)
{
  AdcRegs.ADCMAXCONV.all = 0x0007; // Setup 8 conv's on SEQ1, SEQ2
  AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0; // Setup ADCINA/B0 as 1st conv.
  AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x1; // Setup ADCINA/B1 as 2nd conv.
  AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x2; // Setup ADCINA/B2 as 3rd conv.
  AdcRegs.ADCCHSELSEQ1.bit.CONV03 = 0x3; // Setup ADCINA/B3 as 4th conv.
  AdcRegs.ADCCHSELSEQ2.bit.CONV04 = 0x4; // Setup ADCINA/B4 as 5th conv.
  AdcRegs.ADCCHSELSEQ2.bit.CONV05 = 0x5; // Setup ADCINA/B5 as 6th conv.
  AdcRegs.ADCCHSELSEQ2.bit(CONV06 = 0x6; // Setup ADCINA/B6 as 7th conv.
  AdcRegs.ADCCHSELSEQ2.bit.CONV07 = 0x7; // Setup ADCINA/B7 as 8th conv.
  AdcRegs.ADCTRL1.bit.AQ_PS = 1; // lengthen acq window size
  AdcRegs.ADCTRL1.bit.SEQ_CASC = 1; // cascaded sequencer mode
  AdcRegs.ADCTRL2.bit.EVTSEQ_SEQ1 = 1; // EV manager start
  AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 1; // enable interrupt
  AdcRegs.ADCTRL2.bit.INTMOD_SEQ1 = 0; // int at end of every SEQ1
  AdcRegs.ADCTRL3.bit.SMODE_SEL = 1; // simultaneous sampling mode
  AdcRegs.ADCTRL3.bit.ADCCLKPS = 0x04; // ADCLK = HSPCLK/8 (9.375Mhz)
} /* end adc_init */
Appendix C  DSP Source Code for the Experimental System

```c
yHA = (double)adc.yHA.dc_sum_bak/(double)ADC_COUNT_CAL;
YLA = (double)adc.yLA.dc_sum_bak/(double)ADC_COUNT_CAL;
YHB = (double)adc.yHB.dc_sum_bak/(double)ADC_COUNT_CAL;
YLB = (double)adc.yLB.dc_sum_bak/(double)ADC_COUNT_CAL;

cal_gain_A = (xH - xL)/(yHA - yLA);
cal_offset_a = yLA * cal_gain_A - xL;

cal_gain_B = (xH - xL)/(yHB - yLB);
cal_offset_b = yLB * cal_gain_B - xL;

// sanity check on gains
if ( ( cal_gain_A > 0.95 ) && ( cal_gain_A < 1.05 ) )
  && ( cal_gain_B > 0.95 ) && ( cal_gain_B < 1.05 ) )
  && ( cal_offset_a > -80.0 ) && ( cal_offset_a < 80.0 )
  && ( cal_offset_b > -80.0 ) && ( cal_offset_b < 80.0 ) )
{
  cal_gainA = (int16)(cal_gain_A*(double)(1<<14));
  cal_gainB = (int16)(cal_gain_B*(double)(1<<14));
  cal_offsetA = (int16)cal_offset_a;
  cal_offsetB = (int16)cal_offset_b;
}

//sprintf(str,"cal:gA=%.3f,oA=%5.1f, gB=%.3f,oB=%5.1f\n",cal_gain_A,
  // cal_offset_a,cal_gain_B,cal_offset_b);
//puts_COM0(str);
/* end adc_calibrate */
```

// calculate Iac RMS quantity
ia2_dc = (double)adc.ia2_a.dc_sum_bak/(double)ADC_COUNT_CAL;
val = (double)adc.ia2_a.rms_sum_bak*(double)(1<<ADC_RMS_PS)
```
Appendix C  DSP Source Code for the Experimental System

```c
/ (double)adc.count_rms_bak = ia2_dc*ia2_dc;
if (val < 0.0) val = 0.0;
adc.iac2_a.real = ADC_IPH_SC * sqrt(val);

ib2_dc = (double)adc.iac2_b.dc_sum_bak/(double)adc.count_rms_bak;
val = (double)adc.iac2_b.rms_sum_bak*(double)(1<<ADC_RMS_PS)
/ (double)adc.count_rms_bak = ib2_dc*ib2_dc;
if (val < 0.0) val = 0.0;
adc.iac2_b.real = ADC_IPH_SC * sqrt(val);

// calculate Iac RMS quantity
ial_dc = (double)adc.iac1_a.dc_sum_bak/(double)adc.count_rms_bak;
val = (double)adc.iac1_a.rms_sum_bak*(double)(1<<ADC_RMS_PS)
/ (double)adc.count_rms_bak = ial_dc*ial_dc;
if (val < 0.0) val = 0.0;
adc.iac1_a.real = ADC_IPH_SC * sqrt(val);

ib1_dc = (double)adc.iac1_b.dc_sum_bak/(double)adc.count_rms_bak;
val = (double)adc.iac1_b.rms_sum_bak*(double)(1<<ADC_RMS_PS)
/ (double)adc.count_rms_bak = ib1_dc*ib1_dc;
if (val < 0.0) val = 0.0;
adc.iac1_b.real = ADC_IPH_SC * sqrt(val);

ic1_dc = (double)adc.iac1_c.dc_sum_bak/(double)adc.count_rms_bak;
val = (double)adc.iac1_c.rms_sum_bak*(double)(1<<ADC_RMS_PS)
/ (double)adc.count_rms_bak = ic1_dc*ic1_dc;
if (val < 0.0) val = 0.0;
adc.iac1_c.real = ADC_IPH_SC * sqrt(val);

// calculate Vac RMS quantity
val = (double)adc.vac_ac.dc_sum_bak/(double)adc.count_rms_bak;
val = (double)adc.vac_ac.rms_sum_bak*(double)(1<<ADC_RMS_PS)
/ (double)adc.count_rms_bak = val*val;
if (val < 0.0) val = 0.0;
vac_adc = sqrt(val);
adc.vac_ac.real = ADC_VAC_SC * vac_adc;

val = (double)adc.vac_bc.dc_sum_bak/(double)adc.count_rms_bak;
val = (double)adc.vac_bc.rms_sum_bak*(double)(1<<ADC_RMS_PS)
/ (double)adc.count_rms_bak = val*val;
if (val < 0.0) val = 0.0;
vbc_adc = sqrt(val);
adc.vac_bc.real = ADC_VAC_SC * vbc_adc;

val = (double)adc.vac_ab.dc_sum_bak/(double)adc.count_rms_bak;
val = (double)adc.vac_ab.rms_sum_bak*(double)(1<<ADC_RMS_PS)
/ (double)adc.count_rms_bak = val*val;
if (val < 0.0) val = 0.0;
adc.vac_ab.real = ADC_VAC_SC * sqrt(val);

// calculate real power quantity by averaging sum
val = (double)(1<<ADC_RMS_PS) * (double)adc.p_total.rms_sum_bak
/ (double)adc.count_rms_bak;
adc.p_total.real = ADC_VAC_SC * ADC_IPH_SC * val;

// calculate reactive power quantity by averaging sum
val = (double)(1<<ADC_RMS_PS) * (double)adc.q_total.rms_sum_bak
/ (double)adc.count_rms_bak;
adc.q_total.real = (ADC_VAC_SC * ADC_IPH_SC * val)/__SQRT3;

// apparent power - May need to put in a sign change for the vac_ac calculation
```

260
Appendix C  DSP Source Code for the Experimental System

```c
adc_p_va = adc.vac_ac.real * adc.iac2_a.real + adc.vac_bc.real * adc.iac2_b.real;
if (adc.p_va < 0.0)
    adc.p_va = -adc.p_va;
if (cal_mode == CAL_INIT)
    {
        adc.iac2_a_dc = 0;
        adc.iac2_b_dc = 0;
        adc.iac1_a_dc = 0;
        adc.iac1_b_dc = 0;
        adc.iac1_c_dc = 0;
        iac2_a_dc = 0.0;
        iac2_b_dc = 0.0;
        iac1_a_dc = 0.0;
        iac1_b_dc = 0.0;
        iac1_c_dc = 0.0;
        cal_counter = 0;
        cal_mode = CAL_AVG;
    }
else if (cal_mode == CAL_AVG)
    {
        iac2_a_dc = (1.0-IAC_DC_AVG)*iac2_a_dc + IAC_DC_AVG*iac2_a_dc;
        iac2_b_dc = (1.0-IAC_DC_AVG)*iac2_b_dc + IAC_DC_AVG*iac2_b_dc;
        iac1_a_dc = (1.0-IAC_DC_AVG)*iac1_a_dc + IAC_DC_AVG*iac1_a_dc;
        iac1_b_dc = (1.0-IAC_DC_AVG)*iac1_b_dc + IAC_DC_AVG*iac1_b_dc;
        iac1_c_dc = (1.0-IAC_DC_AVG)*iac1_c_dc + IAC_DC_AVG*iac1_c_dc;
        cal_counter++;
        if (cal_counter >= 150)
            {
                adc.iac2_a_dc = (int16)(iac2_a_dc + 0.5);
                adc.iac2_b_dc = (int16)(iac2_b_dc + 0.5);
                adc.iac1_a_dc = (int16)(iac1_a_dc + 0.5);
                adc.iac1_b_dc = (int16)(iac1_b_dc + 0.5);
                adc.iac1_c_dc = (int16)(iac1_c_dc + 0.5);
                cal_mode = CAL_DONE;
            }
    }
} /* end adc_scale_rms */

EQUATION C HAPTER (NEXT SECTION 1)

/* This function is called every 200ms to perform the RMS calculations on the input AC voltage signals and average the DC signals. The analog quantities are scaled to Volts and Amps for use in the background. */
void adc_scale_slow(void)
{
    // calculate filtered DC values
    adc.vhi.real = ADC_VDC_SC * (double)adc.vhi.dc_sum_bak / (double)COUNT_DC_IN;
    adc.vhi_mid.real = ADC_VDC_SC * (double)adc.vhi_mid.dc_sum_bak / (double)COUNT_DC_IN;
} /* end adc_scale_slow */

*/ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/
*/

261
REFERENCES


References


References


[100] *Powersim website*. Available: [www.powersimtech.com](http://www.powersimtech.com)


