

# Generic Energy Storage Controller for a Power Conditioner

J.P.F. Mostert

12873225

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science at the University of Stellenbosch.

*Pectora roboraunt cultus cecit*

**Study leaders:**

**Dr H.J. Beukes and Dr J.A. du Toit**

*March 2004*

---

## *Declaration*

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Ek, die ondergetekende verklaar hiermee dat die werk gedoen in hierdie tesis my eie oorspronklike werk is wat nog nie voorheen gedeeltelik of volledig by enige universiteit vir 'n graad aangebied is nie.

**Signature:**

**Date:**

---

## *Acknowledgements*

I would like to thank the following:

Firstly, my two study leaders, Dr. Jacques du Toit for his insight, help and knowledge and Dr. Johan Beukes for his guidance, direction and support.

Wernher Swiegers, Jaco Serdyn and Aniel le Roux, always there if help and information is needed.

All my friends and colleagues for their help and support.

God Almighty, in Who I put all my trust.

## *Opsomming*

Die tesis handel oor die ontwerp van 'n DSP gebaseerde beheerstelsel vir 'n lyn-interaktiewe on-onderbreekbare kragbron met drywings-kompensasie met 'n hulp omsetter vir algemene enegiestoor koppeling. The doel is om 'n wye verskeindheid energie store in die huidige drywings elektroniese stelsel te inkorporeer. 'n Tweede omstetter is by die stelsel gevoeg om die energie stelsel te beheer. 'n Nuwe beheerder is ontwikkel om die omsetter te beheer en daardeur die energie stoor. Loodsuur batterye, vloei batterye en vliegwiele is ondersoek om 'n basiese begrip te vorm en te identifiseer wat nodig is vir die beheer van sulke energie store. Die nuwe DSP beheerder en meetstelsel is ontwikkel gebaseer op hierde ondervindings. Om die nuwe beheerstelsel te toets is 'n gs na gs omsetter geïmplementeer vir die beheer van loodsuur en vloei batterye. Die omsetter is geherkonfigureer na 'n gs na ws opstelling en getoets. Die konfigurasie word gebruik vir die beheer van vliegwiele.

---

## *Synopsis*

This thesis presents the design of a DSP based controller system for an auxiliary converter for generic energy storage connection to a line-interactive power compensator. The aim is to utilize a wide range of energy storage systems and incorporate them into the existing power compensator. This was done by adding a second converter to the system. The new controller was developed to control this converter and thereby the energy storage. A study was done on lead acid batteries, flow batteries and flywheels in order to gain a basic understanding of these systems and identify the requirements for their control. Based on these findings, a new DSP based controller and measuring system was developed. To test the new system a dc to dc converter was implemented for the control of lead acid and flow batteries. A dc to ac converter was also tested for the control of flywheel energy storage.

<b>Table of Contents</b>
--------------------------

<b>Declaration</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>Opsomming</b>	<b>iii</b>
<b>Synopsis</b>	<b>iv</b>
<b>LIST OF FIGURES</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>Glossary</b>	<b>xi</b>
<b>1. Introduction</b>	<b>12</b>
1.1 Introduction	12
1.2 Power Conditioners	13
1.3 Digital Controller	13
1.4 Thesis Outline	14
<b>2. Power Compensator</b>	<b>16</b>
2.1 Introduction	16
2.2 Converter Topologies	17
2.2.1 Three Phase Converter	17
2.2.1.1 Pulse Width Modulated Switching	18
2.2.2 Four Phase Converter with Switched Neutral	20
2.3 250 kVA Line-Interactive Compensator	21
2.3.1 Controller	22
2.3.2 Single Converter Configuration	23
2.3.3 Dual Converter Configuration	24
2.3.3.1 DC to DC Configuration	26
2.3.3.2 AC to DC Converter Configuration	32
2.4 Conclusion	33
<b>3. Energy Storage Systems</b>	<b>34</b>

---

---

<b>3.1</b>	<b>Introduction</b>	<b>34</b>
<b>3.2</b>	<b>Flywheel Energy Storage</b>	<b>35</b>
3.2.1	Introduction	35
3.2.2	Energy Content	35
3.2.3	Flywheel shape factor	37
3.2.4	Energy Discharge	38
<b>3.3</b>	<b>Battery Energy Storage</b>	<b>39</b>
3.3.1	Introduction	39
3.3.2	Chemical Reactions	39
<b>3.4</b>	<b>Flow Batteries</b>	<b>40</b>
3.4.1	Introduction	40
3.4.2	Electrochemical Cell	41
3.4.3	Vanadium Redox Flow Battery	43
3.4.3.1	Principal of the Vanadium Battery	43
3.4.3.2	Features and Applications of VRB	44
<b>3.5</b>	<b>Conclusion</b>	<b>47</b>
<b>4.</b>	<b>Development of the Generic Energy Storage System</b>	<b>48</b>
<b>4.1</b>	<b>Introduction</b>	<b>48</b>
<b>4.2</b>	<b>System Description</b>	<b>48</b>
<b>4.3</b>	<b>Generic Energy Storage Controller</b>	<b>50</b>
4.3.1	Digital Signal Processor	50
4.3.1.1	DSP Configuration	52
4.3.1.2	DSP Software Setup	53
4.3.1.3	DSP JTAG Interface	54
4.3.1.4	Boot ROM Loader	55
4.3.2	GESC Memory Interface	57
4.3.3	Field Programmable Gate Array	58
4.3.4	Erasable Programmable Logic Device	59
4.3.5	PLD JTAG Interface	60
4.3.6	LVDS Communication Port	61
<b>4.4</b>	<b>Generic Energy Storage Measuring System</b>	<b>63</b>
4.4.1	Analog to Digital Conversion	64
<b>4.5</b>	<b>Input Output Board</b>	<b>68</b>
<b>5.</b>	<b>Development of Firmware for PLDs</b>	<b>70</b>
<b>5.1</b>	<b>Introduction</b>	<b>70</b>
<b>5.2</b>	<b>Measurement Data Flow from ADC to the DSP</b>	<b>70</b>
<b>5.3</b>	<b>Input Output Board Data Flow</b>	<b>76</b>
<b>5.4</b>	<b>Error Detection</b>	<b>77</b>

---

---

<b>5.5</b>	<b>The GESC and MC Connection</b>	<b>78</b>
<b>5.6</b>	<b>Summary</b>	<b>79</b>
<b>6.</b>	<b>Evaluation of the Energy Storage Controller</b>	<b>80</b>
6.1	Introduction	80
6.2	Practical Setup	80
6.3	Control Algorithm	81
6.3.1	Introduction	81
6.3.2	Derivation	81
6.3.3	Double Prediction	83
6.4	Simulations	85
6.4.1	Introduction	85
6.4.2	Simulation Results	86
6.5	Practical Results	89
6.5.1	Control Software	89
6.5.2	DC to DC Conversion Results	90
6.5.3	DC to AC Results	95
6.6	Summary	97
<b>7.</b>	<b>Conclusions</b>	<b>98</b>
7.1	Overview	98
7.2	Practical Evaluation	99
7.3	Future Work and Recommendations	99
	<b>References</b>	<b>101</b>
	<b>Appendix A GESC Schematics</b>	<b>I</b>
	<b>Appendix B GESMS Schematics</b>	<b>VIII</b>
	<b>Appendix C IOB Schematics</b>	<b>XV</b>
	<b>Appendix D DSP Software</b>	<b>XIX</b>
	<b>Appendix E FPGA Firmware</b>	<b>XXXV</b>
	<b>Appendix F GESMS Firmware</b>	<b>XLII</b>

---



## LIST OF FIGURES

FIGURE 1-1: CONTROLLER CONFIGURATION.....	12
FIGURE 1-2: COMPENSATOR TOPOLOGIES.....	13
FIGURE 2-1: POWER CONDITIONER.....	16
FIGURE 2-2: THREE-PHASE CONVERTER.....	17
FIGURE 2-3: GENERATION OF 3 $\Phi$ CONTROL WAVEFORMS.....	18
FIGURE 2-4: PHASE OUTPUT VOLTAGE OF THREE PHASE CONVERTER.....	19
FIGURE 2-5: LINE-TO-LINE VOLTAGE.....	20
FIGURE 2-6: FOUR-PHASE CONVERTER WITH SWITCHED NEUTRAL.....	20
FIGURE 2-7: SHUNT LINE-INTERACTIVE COMPENSATOR.....	21
FIGURE 2-8: MAIN CONTROLLER OVERVIEW.....	22
FIGURE 2-9: NORMAL LINE-INTERACTIVE CONFIGURATION.....	23
FIGURE 2-10: IGBT MODULE.....	23
FIGURE 2-11: LINE-INTERACTIVE CONFIGURATION WITH AUXILIARY CONVERTER.....	24
FIGURE 2-12: CONTROL AND MEASUREMENT CONNECTIONS.....	25
FIGURE 2-13: DC-BUS VOLTAGE WITH 250kW BATTERY LOAD TEST.....	27
FIGURE 2-14: LINE-TO-LINE OUTPUT VOLTAGE WITH 250kW LOAD TEST.....	27
FIGURE 2-15: SINGLE ENERGY STORAGE DC TO DC CONFIGURATION.....	28
FIGURE 2-16: DUAL ENERGY STORAGE DC TO DC CONFIGURATION.....	28
FIGURE 2-17: PWM CONTROL SIGNAL GENERATION.....	29
FIGURE 2-18: PHASE OUTPUT WAVEFORMS.....	29
FIGURE 2-19: PHASE CURRENT RIPPLE.....	30
FIGURE 2-20: CURRENT MEASUREMENT ERROR.....	31
FIGURE 2-21: AC TO DC CONFIGURATION.....	32
FIGURE 3-1: CONVERTER TOPOLOGIES.....	34
FIGURE 3-2: LEAD ACID BATTERY CELL.....	40
FIGURE 3-3: FLOW BATTERY.....	41
FIGURE 3-4: ELECTROCHEMICAL CELL.....	43
FIGURE 3-5: VANADIUM ELECTROLYTES.....	45
FIGURE 3-6: VANADIUM ELECTROLYTE CURRENT LEAKAGE.....	46
FIGURE 4-1: GENERIC ENERGY STORAGE SYSTEM OVERVIEW.....	49
FIGURE 4-2: CODE COMPOSER PROGRAMMING ENVIRONMENT.....	52
FIGURE 4-3: DSP SOFTWARE CONFIGURATION.....	53
FIGURE 4-4: INTERRUPT FLOW.....	54
FIGURE 4-5: JTAG CONNECTOR PINOUT.....	55
FIGURE 4-6: F2407A BOOT ROM CONFIGURATION.....	55
FIGURE 4-7: GESC BOOT ROM CONFIGURATION.....	56
FIGURE 4-8: GESC MEMORY MAP.....	57
FIGURE 4-9: CONFIGURATION DEVICE CONNECTION.....	59
FIGURE 4-10: CONNECTION BETWEEN THE GESC, MC AND LCD CONTROLLER.....	60
FIGURE 4-11: PLDs JTAG CHAIN.....	61
FIGURE 4-12: DATA RATE VS. CABLE LENGTH.....	62
FIGURE 4-13: LVDS COMMUNICATION LINK.....	62
FIGURE 4-14: PHOTOGRAPH OF THE GESC.....	63
FIGURE 4-15: VOLTAGE MEASUREMENT.....	67
FIGURE 4-16: CURRENT MEASUREMENT.....	67
FIGURE 4-17: PHOTOGRAPH OF THE GESMS.....	68
FIGURE 4-18: PHOTOGRAPH OF THE IOB.....	69
FIGURE 5-1: RECEIVING DATA FROM ADCS ON THE GESMS.....	71
FIGURE 5-2: SENDING MEASUREMENT DATA OVER LVDS PORT.....	72
FIGURE 5-3: CLOCK SIGNALS ON GESMS.....	72
FIGURE 5-4: CLOCK FREQUENCY COMPARISON.....	73
FIGURE 5-5: PROCESSING OF THE MEASUREMENT DATA IN FPGA.....	74

FIGURE 5-6: DPR IMPLEMENTED IN THE FPGA .....	75
FIGURE 5-7: INPUTS FLOW CHART .....	76
FIGURE 5-8: OUTPUTS FLOW CHART.....	76
FIGURE 5-9: ERROR DETECTION BLOCK DIAGRAM.....	77
FIGURE 5-10: ERROR DETECTION FLOW DIAGRAM.....	77
FIGURE 5-11: FLOW CHART OF GESC AND MC CONNECTION.....	78
FIGURE 6-1: DC TO DC CONVERTER EXPERIMENTAL SETUP.....	80
FIGURE 6-2: SWITCHING STATES FOR DC TO DC CONVERTER.....	82
FIGURE 6-3: DOUBLE PREDICTION .....	84
FIGURE 6-4: SIMPLORER SIMULATION PACKAGE .....	85
FIGURE 6-5: PHASE CURRENTS AND TOTAL CURRENT.....	86
FIGURE 6-6: PHASE CURRENTS AND TOTAL CURRENT WITH INTERLEAVED SWITCHING.....	87
FIGURE 6-7: TOTAL OUTPUT CURRENT WITH LOAD STEP .....	88
FIGURE 6-8: OUTPUT VOLTAGE WITH LOAD STEP .....	88
FIGURE 6-9: FLOW CHART OF CONTROL SOFTWARE.....	89
FIGURE 6-10: PHASE CURRENT WITH 4.5 A REFERENCE .....	90
FIGURE 6-11: AVERAGE MEASURED PHASE CURRENTS IN DSP WITH 4.5 A REFERENCE.....	90
FIGURE 6-12: PHASE CURRENT WITH 12 A REFERENCE .....	91
FIGURE 6-13: AVERAGE MEASURED PHASE CURRENT IN DSP WITH 12 A REFERENCE .....	91
FIGURE 6-14: CLOSE-LOOP PHASE AND OUTPUT CURRENTS .....	92
FIGURE 6-15: CLOSE-LOOP PHASE AND OUTPUT CURRENTS WITH INTERLEAVED SWITCHING .....	92
FIGURE 6-16: OPEN LOOP SINGLE SAMPLE PHASE CURRENTS.....	93
FIGURE 6-17: SINGLE SAMPLE PHASE CURRENTS IN DSP.....	93
FIGURE 6-18: OUTPUT CURRENT AND VOLTAGE WITH LOAD STEP.....	94
FIGURE 6-19: DC TO AC CONVERTER EXPERIMENTAL SETUP.....	95
FIGURE 6-20: TORQUE-SPEED CURVE WITH VOLT-HERTZ CONTROL.....	95
FIGURE 6-21: PHASE OUTPUT VOLTAGES WITH MODULATION INDEX 33% .....	96
FIGURE 6-22: PHASE OUTPUT VOLTAGES WITH MODULATION INDEX 50% .....	96
FIGURE 6-23: PHASE OUTPUT VOLTAGES WITH MODULATION INDEX 100% .....	96

## **LIST OF TABLES**

TABLE 1: FLYWHEEL SHAPE FACTOR.....	37
TABLE 2: FLYWHEEL ENERGY AND DESIGN STRESS .....	38
TABLE 3: ENERGY STORAGE CONTROL SYSTEM REQUIREMENTS .....	47
TABLE 4: ADDRESS DECODING .....	58
TABLE 5: BIT DEFINITIONS OF ADC CONTROL REGISTERS .....	65
TABLE 6: CONTROL REGISTER C0 BIT FUNCTIONS .....	65
TABLE 7: CONTROL REGISTER C1 BIT FUNCTIONS .....	65
TABLE 8: ENCODING MEASUREMENT DATA .....	73

## Glossary

GESS	Generic Energy Storage System
GESC	Generic Energy Storage Controller
GESMS	Generic Energy Storage Measuring System
MC	Main Controller
FESS	Flywheel Energy Storage
BESS	Battery Energy Storage
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
EPLD	Erasable Programmable Logic Device
ADC	Analog to Digital Converter
LVDS	Low Voltage Differential Signalling
JTAG	Joint Test Action Group
ROM	Read Only Memory
RAM	Random Access Memory
LCD	Liquid Crystal Display
MSPS	Million Samples Per Second
MIPS	Million Instructions Per Second
MFLOPS	Million Floating Point Operations Per Second
PWM	Pulse Width Modulation
I/O	Input Output
DC	Direct Current
AC	Alternating Current
3 $\phi$	Three Phase
UPS	Uninterrupted Power Supply
IGBT	Insulated Gate Bipolar Transistor
VRB	Vanadium Redox Battery
CT	Current Transformer
PCB	Printed Circuit Board

## 1. Introduction

### 1.1 Introduction

The focus of this thesis is the development of a DSP based controller system for a dual-converter line-interactive power compensator with generic energy storage. The compensator functions as a medium to high power Uninterrupted Power Supply (UPS) and shunt power line conditioner. The new controller system has to handle all the measurements, control calculations and associated inputs and outputs for the energy storage. The compensator was previously a single converter compensator and the existing controller does not have the resources to handle an additional converter. The new generic energy storage system (GESS) works in combination with a main controller (MC) that handles the power quality calculations and overall system control.

GESS is designed to work in combination with the MC, but can function on its own without connection to the MC. This enables the device to be configured for a wide range of tasks. The GESS is an extension of the main controller and is responsible for the control of the second converter connected to the energy storage system. This system is designed to handle a wide range of energy storage systems with the main focus on flywheels, lead acid batteries and flow batteries.

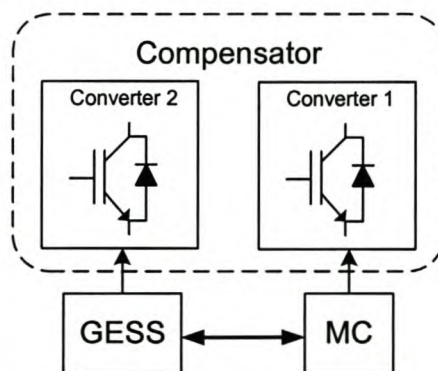


Figure 1-1: Controller Configuration

## 1.2 Power Conditioners

The converter is the main building block of a power compensator. The objective of the converter is to produce an output waveform of which the magnitude and frequency can be controlled.

Two of the most popular power compensators at medium to low power levels (<1MW) are generally classified as in-line or line-interactive compensators [3] [4]. The in-line configuration converts full load power to dc and back to ac at all times, while the line-interactive configuration interacts with the line only when required. Line-interaction has the benefit of lower losses and improved reliability and life expectancy.

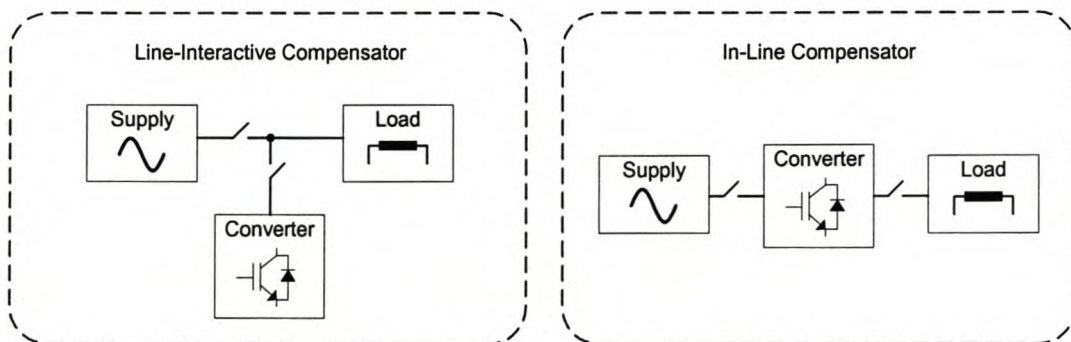


Figure 1-2: Compensator Topologies

## 1.3 Digital Controller

In the real world all signals are analog by nature. A digital controller operates on data represented by binary numbers that are composed of a restricted number of bits. A digital controller cannot use real world (analog) signals; the analog signal has to be converted to a digital representation of that signal by an analog to digital converter. The digital controller is then capable of using (manipulating) the data to control or perform a specific task. There are three main types of digital

processors; microprocessors, microcontrollers and digital signal processors. Each of the three is optimized to perform a specific task.

*Microprocessor:* This is a device mainly found in today's personal computers and laptops. It is optimized to move large amounts of data around. A microprocessor is not a stand-alone device but requires additional memory and chips to function.

*Microcontroller:* This is a much simpler device and is not as powerful as the microprocessor. Microcontrollers are used in a wide variety of applications from alarm systems to the electric toothbrush. They offer a single chip solution and do not require a number of external devices. They usually include internal memory.

*Digital Signal Processor:* There is a wide variety of digital signal processors (DSPs). DSPs can vary from the functionality of the microprocessor to that of the microcontroller depending on the application. The main difference is that the DSP is optimized to handle digital signal processing tasks. DSPs have specialized functions built in, such as ultra fast multipliers and accumulators. This allows the DSP to compute algorithms such as digital filtering and Fourier analysis in real time. DSPs will be used in applications where mathematical calculations dominate the controller function.

Some advantages of designing with DSPs over microprocessors or controllers are [36]:

- Single-cycle multiply-accumulate operations
- Real time performance, simulation and emulation
- Flexibility
- Increased system performance
- Reduced system cost

#### **1.4 Thesis Outline**

In Chapter 2 the basic working of a power compensator is discussed. The chapter starts with the basic converter topologies and builds on that until the 250

---

kVA line-interactive compensator is discussed. This system forms the basis of the discussion for the development of the GESS.

In Chapter 3 a study is made of lead acid batteries, flow batteries and flywheels and of how these energy storage systems can be incorporated into the existing line-interactive compensator. The aim of this chapter is to arrive at a basic understanding of these energy storage systems and what is needed to be able to control them.

The development of the GESS is dealt with in Chapter 4. This includes the hardware design of the controller, the measuring system and inputs and outputs. These are all separate components which are combined to form the complete controller system. All the different devices and components are discussed here.

In Chapter 5 the firmware development for all the PLD devices is described. This chapter links with Chapter 4 to form the complete hardware solution.

The evaluation of the GESS is done in Chapter 6. The system was tested by implementing a closed loop dc to dc converter which is used to control lead acid batteries and flow batteries. A basic dc to ac converter was also implemented for controlling flywheels.

The final conclusions are made in Chapter 7.

---



## 2. Power Compensator

### 2.1 Introduction

The power compensator is a device that enables the control of electrical power-flow to and from a load. Depending on the application, the output to the load may vary from constant or adjustable dc, to an ac load that requires adjustable frequency and magnitude. The control of power-flow is usually desired as a means to control one or more non-electrical parameters, eg the speed of a motor, the temperature of an oven or the intensity of lighting.

The power-flow through a compensator can be in both directions, from the utility to the load or from the load to the utility. The power compensator usually consists of more than one power conversion stage [4]. These stages are connected by energy storage elements such as capacitors. A power conversion stage is called a converter. The power electronic switches are the basic building blocks of a converter.

The output voltage and power can be controlled by controlling the switching of the semiconductor devices.

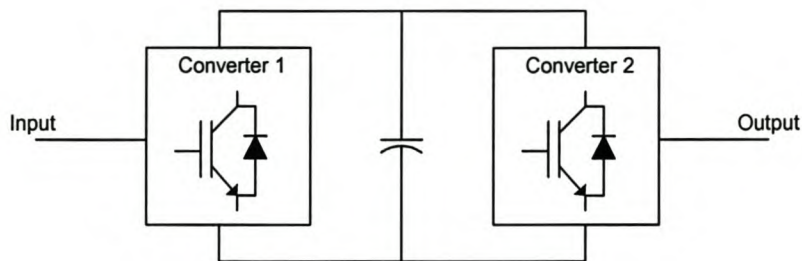


Figure 2-1: Power Conditioner

## 2.2 Converter Topologies

### 2.2.1 Three Phase Converter

The three-phase converter topology can be derived by using three half-bridge converters, with each half-bridge supporting one phase of the load. There are two switches in each half-bridge and six switches in total in the three-phase converter (Figure 2-2). In applications such as uninterruptible power supplies and motor drives, three-phase converters are commonly used to supply three-phase loads. The three-phase, three wire, converter can generate only two independent output voltages, as the phases are coupled.

In the three-wire configuration there is no path for neutral currents to flow, and for this reason only balanced three phase loads are connected. Load variations also affect the dc-bus capacitors. The coupling effect will cause the star-point to float, depending on the state of the switches.

The four-wire configuration enables neutral currents to flow back to the centre-point of the dc-bus capacitors, acting as a ground return. Unbalanced loads can now be connected to the converter. The disadvantage of this setup is that the neutral currents, containing high frequency components, must flow through the dc-bus capacitor, shortening the life expectancy considerably.

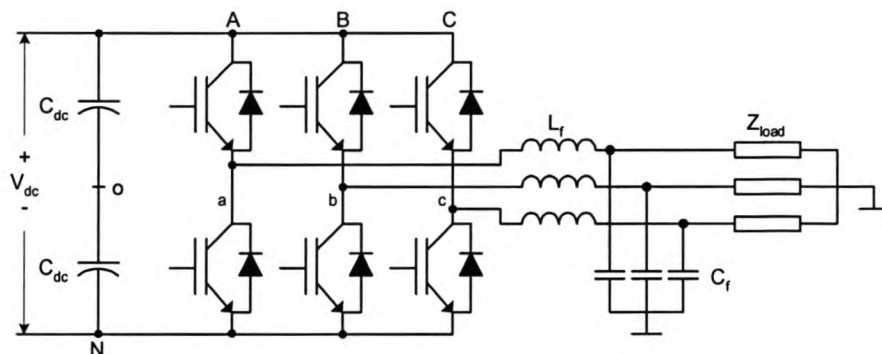
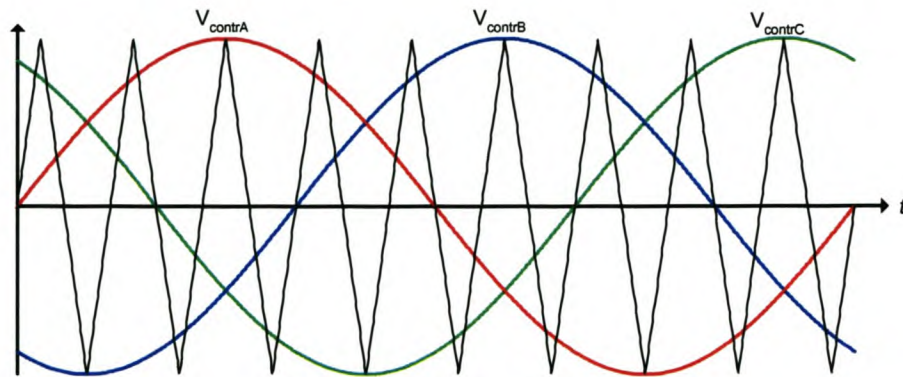


Figure 2-2: Three-Phase Converter

### 2.2.1.1 Pulse Width Modulated Switching

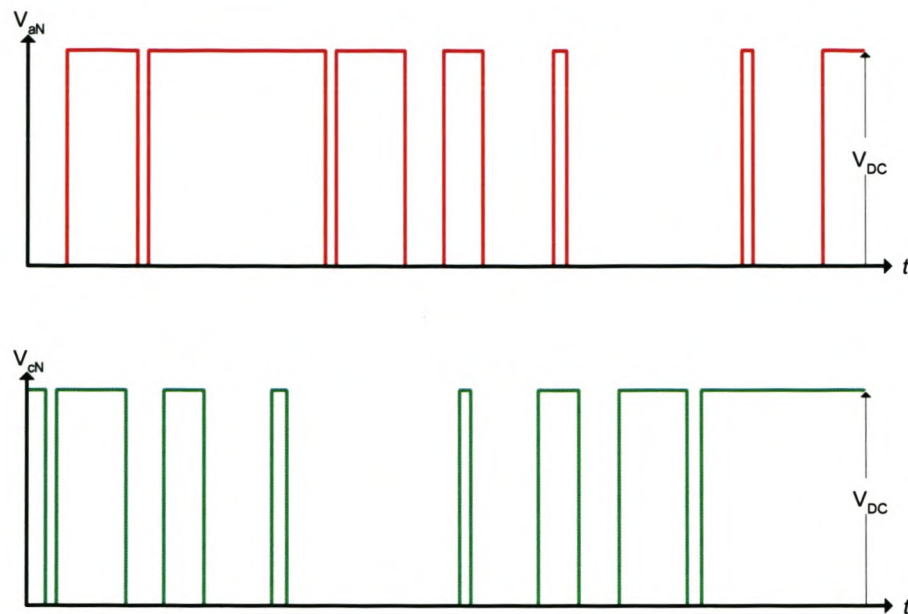
To obtain balanced three-phase output voltage waveform, three sinusoidal control signals,  $V_{\text{contrA}}$ ,  $V_{\text{contrB}}$  and  $V_{\text{contrC}}$ , that are 120 degrees out of phase, are compared to a triangular waveform in order to generate the switching signals. These waveforms can be seen in Figure 2-3.



**Figure 2-3: Generation of 3 $\Phi$  Control Waveforms**

The frequency of the triangular waveform establishes the converter switching frequency. The output voltages follow the changes in the amplitude and frequency of the control signals. The frequency of the control signal (e.g.  $V_{\text{contrA}}$ ) is the fundamental frequency of the output voltage (Figure 2-4) in each phase. The output of each phase depends only on  $V_{\text{dc}}$  and the switch status; the output voltage is independent of the output of the load current since one of the two switches in a phase is always on at any instant.

The average dc component of the output is identical in each phase which is measured with respect to the negative dc bus. These dc components are cancelled out in the line-to-line voltages.



**Figure 2-4: Phase Output Voltage of Three Phase Converter**

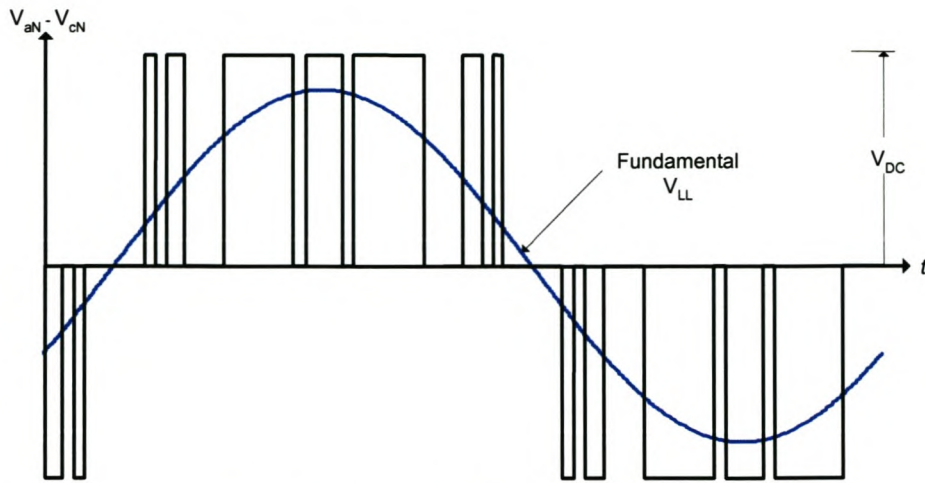
The peak value of the fundamental component in one of the converter phases is:

$$\hat{V}_{aN} = \frac{V_{DC}}{2} \quad (2-1)$$

In three-phase converters, only the harmonics in the line-to-line voltages are of concern. To calculate the line-to-line voltages the output of each phase is subtracted from the other phases. The result is a three-level voltage output with a higher frequency content. This high frequency component can be filtered out with a low pass filter, leaving the output voltage containing only the fundamental frequency (Figure 2-5). The maximum line-to-line rms voltage at the fundamental is:

$$\begin{aligned} V_{LL} &= \frac{\sqrt{3}}{\sqrt{2}} \hat{V}_{aN} \\ &= \frac{\sqrt{3}}{2\sqrt{2}} V_{DC} \\ &= 0.612V_{DC} \end{aligned} \quad (2-2)$$

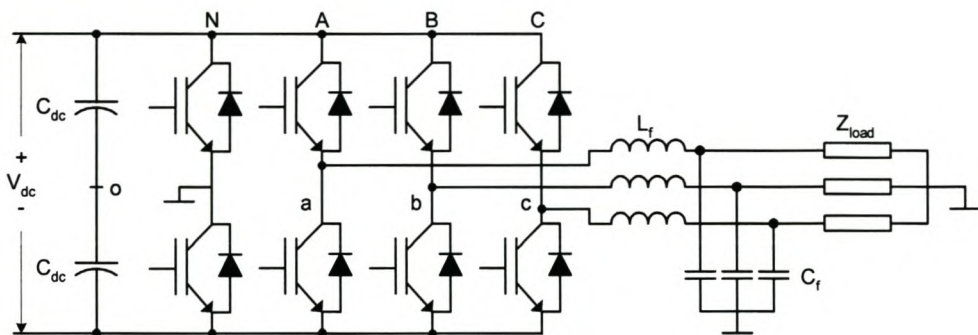
This equation is only valid for direct sinusoidal control



**Figure 2-5: Line-to-Line Voltage**

### 2.2.2 Four Phase Converter with Switched Neutral

The operation and control of a four-phase converter do not differ much from those of the three-phase converter. The star-point of the outputs of the load connects to the output of the fourth phase arm. The star point can therefore be forced to a specific voltage, making the outputs of the three main phase arms decoupled.



**Figure 2-6: Four-Phase Converter with Switched Neutral**

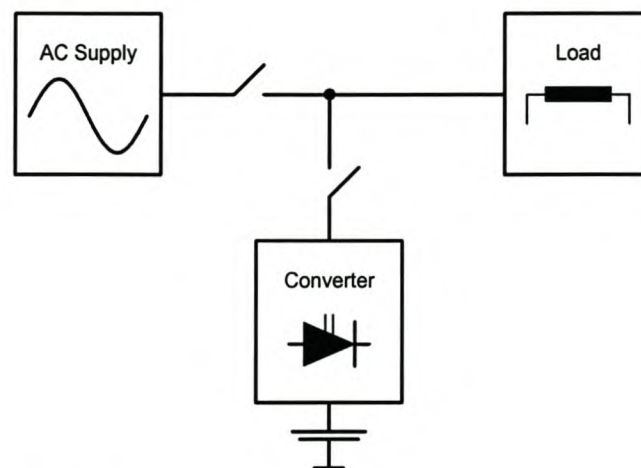
This enables the control of each phase independently. Neutral currents now flow to the switched neutral point, solving the problem of switching current through the dc-bus capacitors.

There are two basic approaches to controlling the output of the fourth phase arm. When using space vector control, all three dimensions can be utilized, due to the return path for the zero sequence (neutral) currents. The second method is to pulse width modulate the fourth phase arm to a fixed voltage, simulating the dc-bus centre-point.

### 2.3 250 kVA Line-Interactive Compensator

The 250 kVA line-interactive compensator discussed below is an existing industrial UPS system. It is capable of active power filtering, voltage regulation and dip compensation. It has been installed in various locations across the world, protecting sensitive loads, increasing the power quality and reliability on mini-grid networks. The device is interfaced with lead acid batteries as energy storage. The energy storage enables the device to compensate for dips and power outages on the supply.

This system forms the basis of the discussion for development of the GESS.



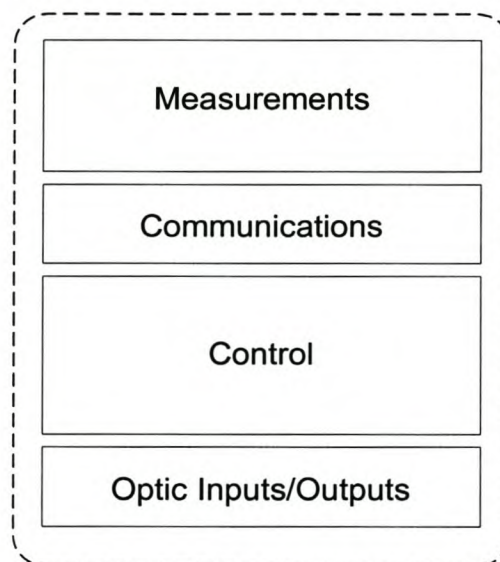
**Figure 2-7: Shunt Line-Interactive Compensator**

### 2.3.1 Controller

The existing MC is for a single converter shunt line-interactive compensator. The main components of the MC are the TMS320C31-50 floating point DSP from Texas Instruments [32] and an EPM81500 FPGA from Altera. A separate controller with a TMS320C50 DSP from Texas Instruments [33] is used to control the human interface and LCD display. All the control algorithms and overall system processing is done by the MC.

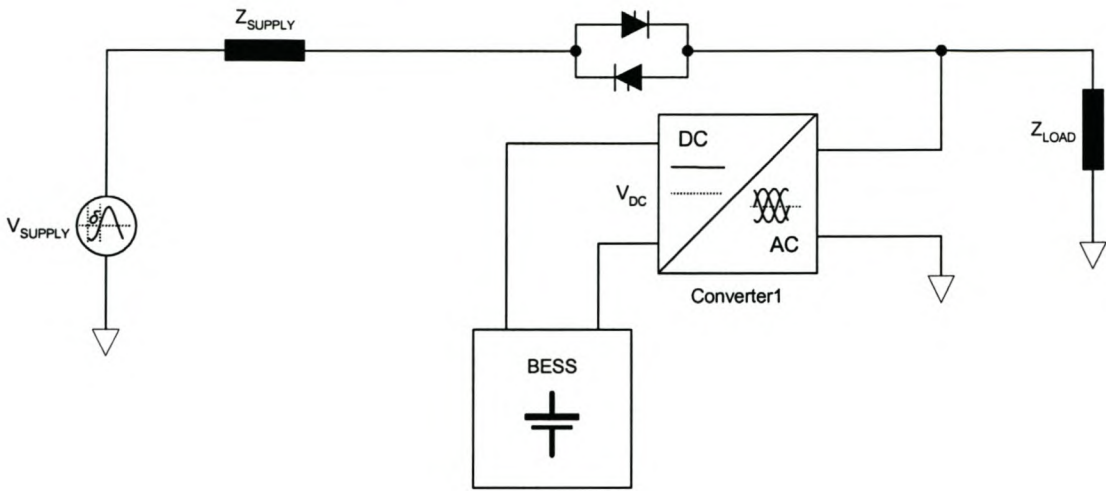
The converter is electrically isolated from the controller using fibre optic interconnections. Ten transmitters and ten receivers are used to send PWM signals to the converter and to monitor the status.

The MC switches the converter at 5 kHz. The various currents and voltages are measured and sampled at the switching frequency. Due the high switching harmonic content of the measured signal it has to be filtered before being sampled by the analog to digital converters. This process will be fully explained in section 2.3.3.1.



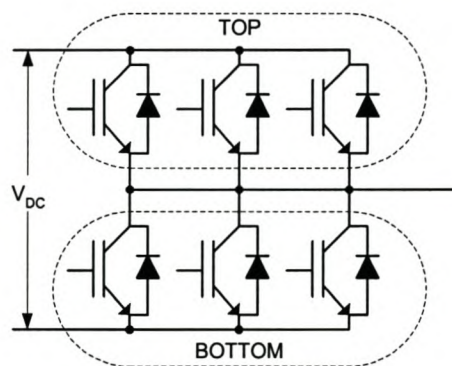
**Figure 2-8: Main Controller Overview**

### 2.3.2 Single Converter Configuration



**Figure 2-9: Normal Line-Interactive Configuration**

This UPS system consists of a single 250 kVA four-phase switched neutral converter. Each phase of the converter is made up of a three-phase IGBT module containing six switches. The top three and bottom three switches are switched in parallel. The converter phase current is shared between the three phases of the module. The reason for this is to be able to reach a higher power rating.



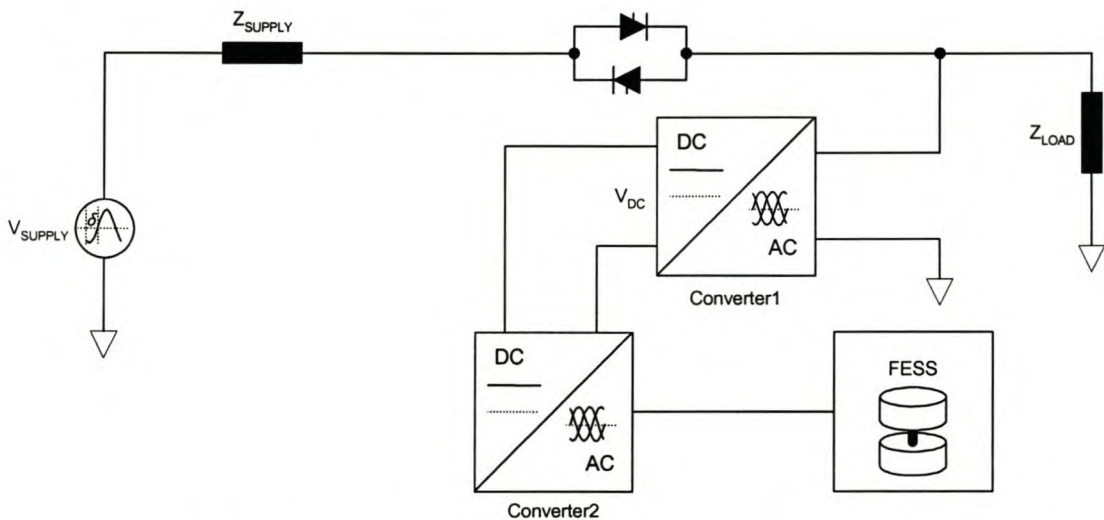
**Figure 2-10: IGBT Module**

Each IGBT module is optically isolated from the MC. Two error signals, for the top and bottom switches respectively, monitor the health of each module. An



error will be generated internally by the module on over-current, over-temperature and under-voltage on the gating signals. The dc-bus is connected to an 800V lead acid battery bank. A dc-dump configuration is connected to the centre point of the dc-bus to protect it from over-voltage.

### 2.3.3 Dual Converter Configuration



**Figure 2-11: Line-Interactive Configuration with Auxiliary Converter**

Adding a second converter to the UPS system allows complex energy storage systems to be utilized. Both the converters are identical. The main converter controls the output waveform while the second converter controls the energy storage system. When ac energy storage is used, such as with a flywheel, the second converter is used in an ac to dc configuration. A dc to dc configuration can also be used to step the voltage up or down for a dc energy storage such as lead acid or flow batteries. The converter keeps the dc-bus voltage constant under variable battery loads.

In Figure 2-12 the various connections and measurements which control the second converter can be seen. Note that the connections for converter 1 are not shown in the figure.

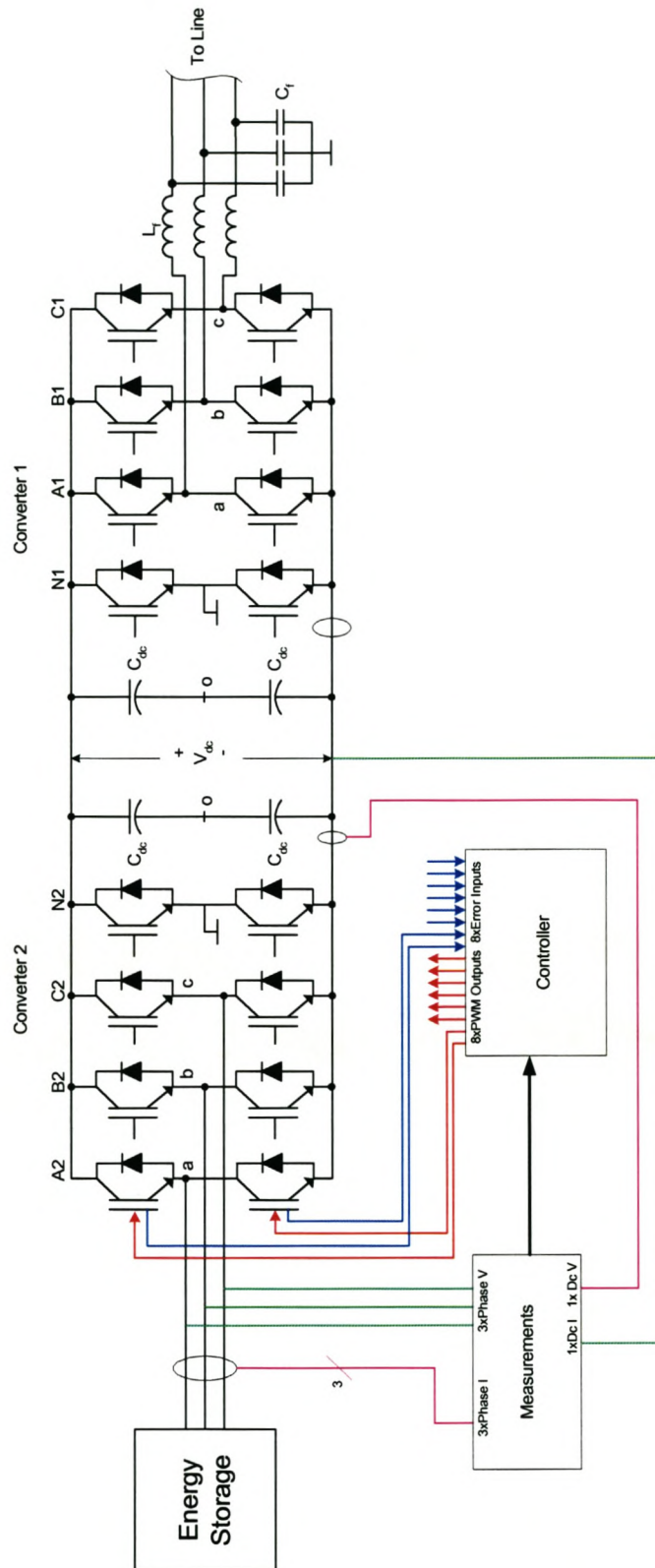


Figure 2-12: Control and Measurement Connections

### **2.3.3.1 DC to DC Configuration**

When converter 2 is used in a dc to dc configuration, all four phases are connected in parallel. All the top (and bottom) switches are switched simultaneously. Flow batteries (section 3.4) have a relatively low output voltage and to reach the maximum system output power of 250 kW, large currents are drawn. For example:

$$\begin{aligned} I &= \frac{P}{V} \\ &= \frac{250 \text{ kW}}{150 \text{ V}} \\ &= 1.66 \text{ kA} \end{aligned} \tag{2-3}$$

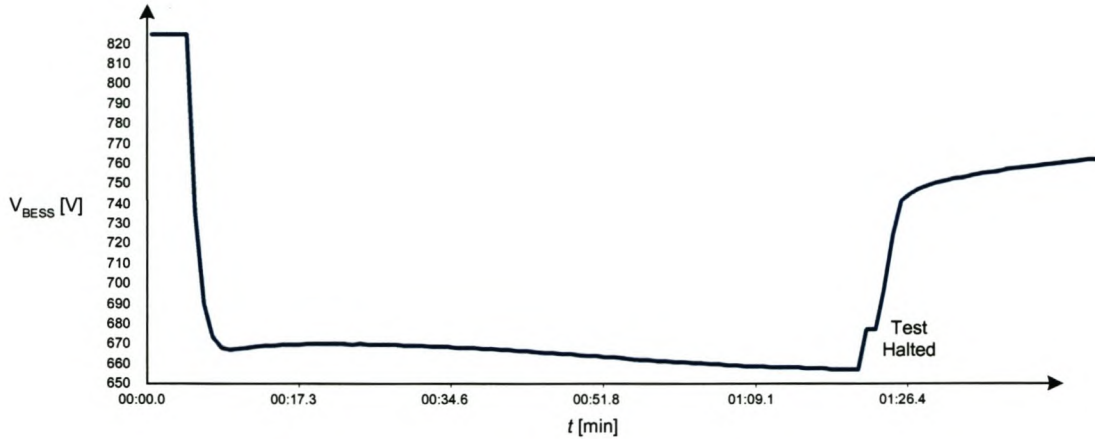
A single switch is not able to handle 1.66 kA, which is the reason why the current has to be shared by the phases (Figure 2-15).

The phases of the converter cannot be controlled individually because of the limitations of the measuring system of the MC. All the measurements in the MC are sampled only once in a 5 kHz period. If the timing of the phase measurements can be controlled individually or oversampled, the problem can be solved. This allows the output to be split between two energy storage systems (Figure 2-16), for instance two flow batteries or a combination of flow- and lead acid batteries (section 3.3). Each of the energy storage systems can now be individually controlled with two phase arms. If only one energy storage system has to be serviced, the system can function at half capacity.

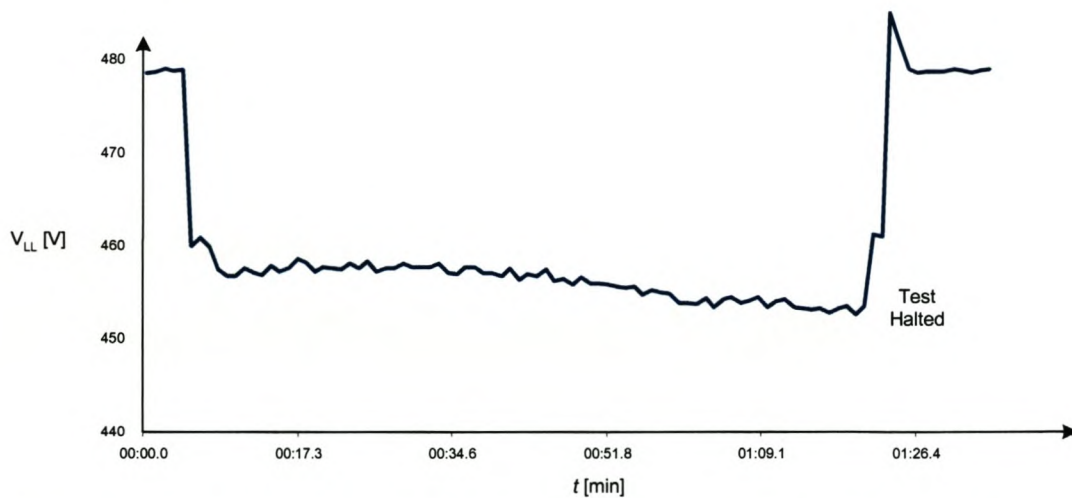
When large amounts of power are drawn from batteries the voltage can drop significantly due to internal resistance. This limits the output voltage of the system when the batteries are connected directly on the dc-bus. Figure 2-13 shows how the dc-bus voltage drops when a large amount of power is extracted from the batteries. This in turn affects the system output voltage.

---

The line-to-line output voltage of the shunt power compensator can be seen in Figure 2-14. These tests were done externally.

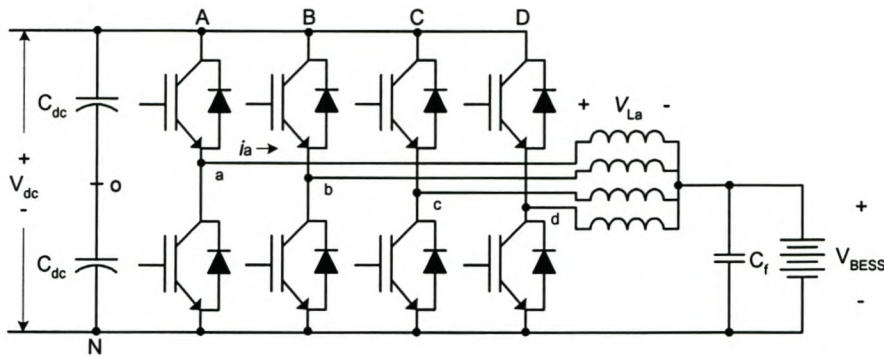


**Figure 2-13: DC-Bus Voltage with 250kW Battery Load Test**

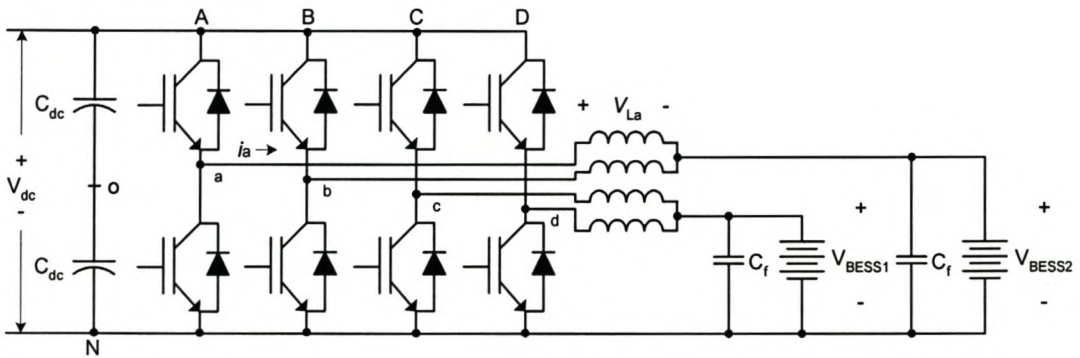


**Figure 2-14: Line-to-Line Output Voltage with 250kW Load Test**

To keep the dc-bus voltage constant, the battery voltage has to be stepped up, or the dc-bus voltage stepped down, depending on the direction of the power flow. The configuration for the step-up and step-down converter stays the same. The converter is a two-quadrant configuration i.e. power can flow in both directions. Taking the dc-bus as source, a step-down converter is implemented.



**Figure 2-15: Single Energy Storage DC to DC Configuration**



**Figure 2-16: Dual Energy Storage DC to DC Configuration**

The battery voltage can be controlled by changing the duty cycle,  $D$ , of the switches. The PWM control signal, at a specified  $D$ , is generated by comparing a triangular wave to the control voltage  $V_{control}$ .

$$D = \frac{t_{TOP}}{T_S} = 1 - \frac{t_{BOT}}{T_S} = \frac{V_{BESS}}{V_{DC}} \tag{2-4}$$

$$(1 - D) = \frac{t_{BOT}}{T_S}$$

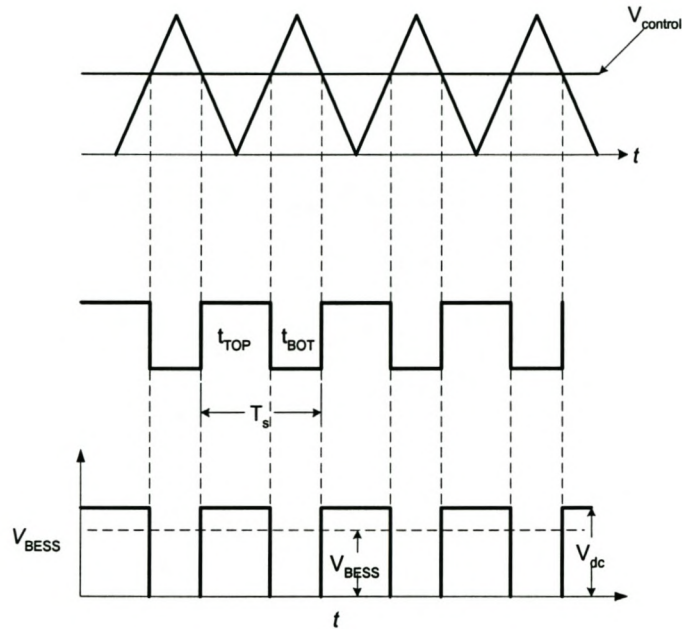
Where:

$t_{TOP}$  = top switches on

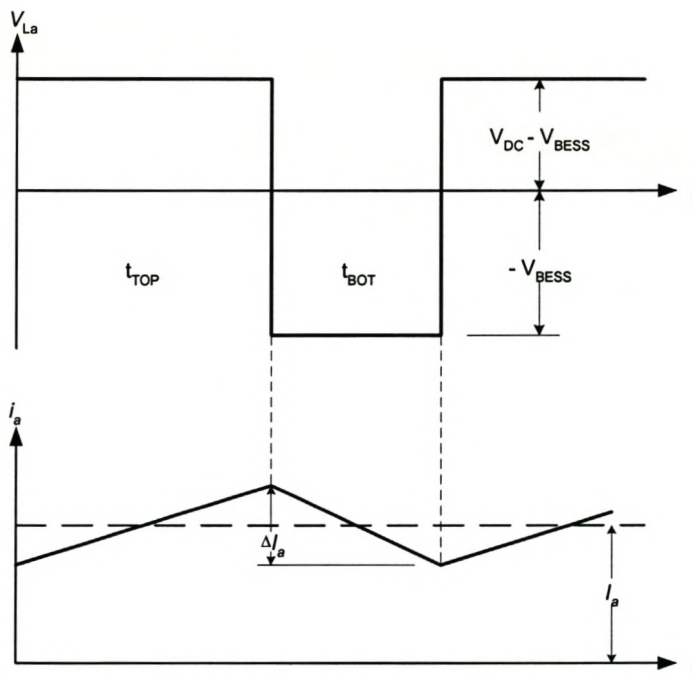
$t_{BOT}$  = bottom switches on

$T_S$  = switching period

Equation (2-4) is derived from the fact that the average voltage across an inductor is equal to zero (Figure 2-18).



**Figure 2-17: PWM Control Signal Generation**



**Figure 2-18: Phase Output Waveforms**

The phase current contains a high harmonic content at the switching frequency. By looking at the voltage over the inductor, the ripple current can be calculated as follows:

$$V = L \frac{di}{dt}$$

$$V_{BESS} = L \frac{di_a}{(1-D)T_s} \quad (2-5)$$

$$\Delta I_a = \frac{V_{BESS}}{L} (1-D)T_s$$

For example:

$$V_{BESS} = 400 \text{ V}$$

$$L = 800 \text{ uH}$$

$$D = 0.5$$

$$T_s = \frac{1}{f_s} = 0.2 \text{ ms} \quad (2-6)$$

$$\Delta I_a = \frac{V_{BESS}}{L} (1-D)T_s$$

$$= \frac{400}{800 \times 10^{-6}} (0.5)(0.0002)$$

$$= 50 \text{ A}$$

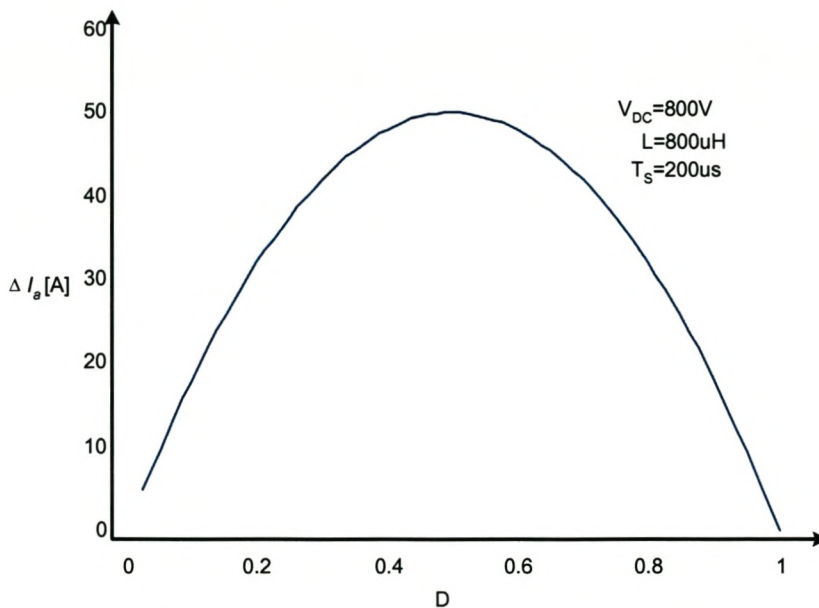
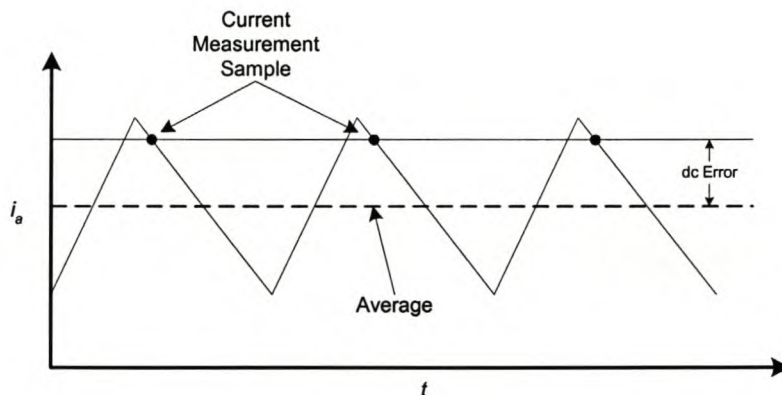


Figure 2-19: Phase Current Ripple

In Figure 2-19 it can be seen that the ripple current is at a maximum when the duty ratio is 50%. To reduce this ripple current, each phase arm can be switched 90 degrees out of phase (interleaved switching). The problem is that the measurements are all taken at the same time for all the phases. Because of the high ripple content, this will result in a different current reading for each phase. This ripple content plays a significant role in the closed-loop current control of the step down converter. As mentioned in section 2.3.1, the currents for the MC are sampled at the switching frequency of 5 kHz. With high ripple content these measurements are not very reliable. If the converter is switching at a fixed duty ratio the ripple current waveform will stay the same. The MC measurement system always samples at the same place (time) in the current waveform. By doing this the measuring system makes a dc error (aliasing) on the current measurement as shown in Figure 2-20.



**Figure 2-20: Current Measurement Error**

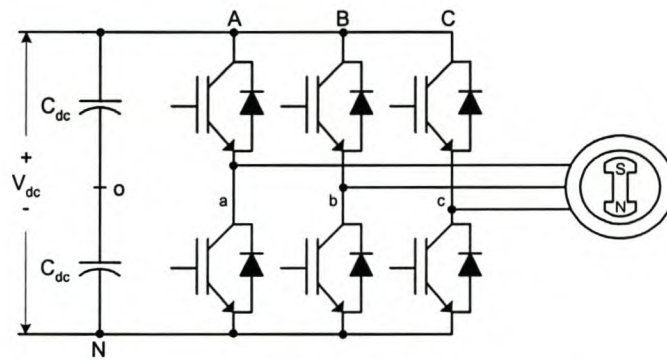
This problem becomes worse if the duty ratio varies in time. The error changes from linear (dc) to non-linear (varying error).

To counter this problem, an analog low pass filter is implemented in the MC measurement system before sampled by the analog to digital converters. This in turn affects the dynamic response of the control loop.



### 2.3.3.2 AC to DC Converter Configuration

When ac to dc operation is required, converter 2 is configured as a standard three-phase converter. This configuration is used when flywheels are connected as energy storage. A flywheel is, in essence, an electric machine connected to a rotating mass. The control of the flywheel depends on which type of machine is used.



**Figure 2-21: AC to DC Configuration**

In high-speed flywheels, friction losses have to be kept to a minimum. This is achieved by sealing the flywheel in a vacuum and using magnetic bearings. To prevent frictional losses, dc machines are not used because of the physical contact of the brushes on the rotor. Dc machines are also expensive and require frequent maintenance. Induction machines seem like a perfect solution, they are inexpensive and do not require continuous maintenance, but the generative properties of the induction machine are not satisfactory [7]. Synchronous machines have better generative properties because of their ability to control the power factor by changing the dc excitation, but again the synchronous machine has sliprings (brushes) that connect to the rotor windings. The most popular choice is the permanent magnet synchronous machine. The rotor has no contacts except the top and bottom bearings, reducing the friction losses. Therefore most manufacturers of high-speed flywheels use this type of machine in their designs.

## **2.4 Conclusion**

The MC was not designed to control two converters in addition to an energy storage system. It does not have sufficient I/O ports and processing power to handle the additional overhead of the second converter and energy storage system.

To control and utilize the full potential of a complex energy storage system a flexible measuring system is needed. This will allow the use of interleaved switching and the ability to incorporate dual energy storages. The measuring system of the MC was not designed to handle the additional complex measurements involving a second converter and energy storage.

---

### 3. Energy Storage Systems

#### 3.1 Introduction

Electrical energy can be stored in many forms. These can be chemical, as in batteries, electrostatic, as charge in capacitors, the gravitational potential energy of water generating electricity or in the kinetic energy of a rotating mass. Most active power conditioners, e.g. power quality compensators, end-of-line support systems and peak-shaving systems, use lead-acid batteries for energy storage [11]. Due to the constant dc voltage nature of these batteries, they can usually be connected directly to the dc side of a converter. When full capacity needs to be used, the voltage variation between charge and discharge can become too high.

Using the direct battery connection limits the energy storage of the power conditioner to lead acid or equivalent battery technology. By adding a second converter to the power conditioner, different energy storage systems can be combined with the existing system. This allows for the dc voltage to be stepped up or down in a dc to dc configuration or power to be absorbed from a flywheel in an ac to dc configuration. This Chapter will focus on flywheels, lead-acid batteries and flow batteries, in order to establish the requirements for the GESS control. The layout and configuration of the compensator were discussed in Chapter 2.



**Figure 3-1: Converter Topologies**

## **3.2 Flywheel Energy Storage**

### **3.2.1 Introduction**

Flywheels are one of the oldest forms of energy storage, and are now being considered again for a much wider field of utilization, competing with electrochemical batteries. Flywheels operate on a simple principle of storing kinetic energy in a rotating mass.

In ancient potteries, a kick at the lower wheel of the rotating table was the energy input to maintain rotation. The rotating mass stores the short energy input so rotation can be maintained at a fairly constant rate. Flywheels have been applied in steam and combustion engines for the same purpose since the time of their invention. The application of flywheels for longer storage times is much more recent, and has been made possible by developments in materials science and bearing technology.

Traditionally, flywheels have been large steel wheels rotating at low speed. Consequently their utilization has been limited due to space constraints.

With the advent of modern materials and techniques such as high strength composites, permanent magnets and power electronics, it is now possible to utilize a "modern flywheel" to store energy for electrical applications [6].

### **3.2.2 Energy Content**

Flywheels are capable of storing large amounts of power, which makes them useful in short events requiring large amounts of power like UPS systems. The size of the flywheel and of the motor generator depends on whether the application requires a long or short discharge. A long discharge would require a large flywheel and a small motor generator, while a short discharge would require the opposite. The rate at which energy can be exchanged into or out of the flywheel is limited only by the motor/generator design [38]. Therefore, it is

---

possible to withdraw large amounts of energy in a far shorter time than with traditional chemical batteries (high turn-around efficiency) [6], [38].

The energy content of a rotating mechanical system is:

$$W = \frac{1}{2} J \omega^2 \quad (3-1)$$

Where:

$J$  = moment of inertia.

$\omega$  = angular velocity.

The moment of inertia is a function of the mass and shape of the object. This is defined as:

$$J = \sigma \int x^2 dm \quad (3-2)$$

$\sigma$  = inertial constant (depends on shape).

$x$  = distance from the axis.

$dm$  = differential mass.

With  $x$  as a constant radius  $r$  the solution to the integral is:

$$J = \sigma m r^2 \quad (3-3)$$

$$W = \frac{1}{2} \sigma r^2 m \omega^2 \quad (3-4)$$

and the equation per unit of mass:

$$W_m = \frac{1}{2} \sigma r^2 \omega^2 \quad (3-5)$$

---

This equation shows that angular velocity is much more important than the mass to obtain high stored energy.

The tensile strength of the material dictates the upper limit of the angular velocity, thus to extract maximum energy from a flywheel with fixed dimensions, a material which combines low density with high tensile strength should be used [39].

### 3.2.3 Flywheel shape factor

For a simple ring flywheel the inertial constant (shape factor)  $\sigma$  is equal to 1. The shape factor is a measure of the efficiency with which the flywheel's geometry uses the material's strength. The ideal shape of the flywheel would result in biaxial, tangential and radial stresses in the material being uniform and  $\sigma = 2$ . These flywheels are thickest near the axis and thinnest at the rim. This is the classical steam engine design. Table 1 [6], [38] shows the shape factor for different flywheel designs

**Table 1: Flywheel Shape Factor**

<b>Flywheel shape</b>	<b><math>\sigma</math></b>
Constant stress disk	1.862
Flat unpierced disc	1.212
Thin ring	1.000
Flat pierced disc	0.610

Designs where composite materials are used differ from the classic flywheel design. The reason is that in fibre materials the tensile strength is high in only one direction. This means the biaxial and tangential stresses must be limited. In practice, rims of finite thickness are used and are not stressed uniformly. For this design the shape factor is typically  $\sigma = 0.8$ .

To design a flywheel with maximum stored energy depends on many different factors: Material design stress, material density, total mass and flywheel shape. Table 2 [6] shows how certain materials cope with rotating stresses.

**Table 2: Flywheel Energy and Design Stress**

Material	Design stress [ $10^6 N/m$ ]	Useful energy [ $10^3 J/kg$ ]
Wood	30	21
Mild steel	300	29.5
E-glass 60% fibre/epoxy	250	50.4
S-glass 60% fibre/epoxy	350	70.5
Maraging steel	900	86.4
Titanium alloy	650	110.8
Carbon 60% fibre/epoxy	750	185.7
Kevlar 60% fibre/epoxy	1000	274.3

### 3.2.4 Energy Discharge

When discharging a flywheel the angular velocity must be kept within a certain range. The mechanical design (bearings, electrical machine) limits the operating speed. The flywheel could become unstable and disintegrate at lower or higher speeds. This means that not all the stored energy can be extracted. The useful energy per mass unit can be given as follows:

$$W_m = \frac{1}{2} \sigma r^2 \omega^2 (1 - s^2) \quad (3-6)$$

$s$  = ratio of minimum to maximum speed.

This ratio is usually taken to be 0.2.

### **3.3 Battery Energy Storage**

#### **3.3.1 Introduction**

Lead acid batteries are the most commonly used batteries in power compensators [11]. The reasons are: low cost, high voltage per cell and reasonably good capacity life [6]. The batteries can be connected directly to the dc-bus of the compensator. No advanced control algorithms are needed for charge or discharge.

In normal mode, when the line voltage is present, the battery uses only a small trickle charge to keep it at a nominal voltage.

When large currents are drawn the output voltage can drop considerably. Because the batteries are stacked in series to reach the voltage level, the internal resistance of each battery is added to that of the others. The result is a fairly large internal resistance of the battery bank. Lead acid batteries can also not be left in the discharge state for too long without being permanently damaged.

#### **3.3.2 Chemical Reactions**

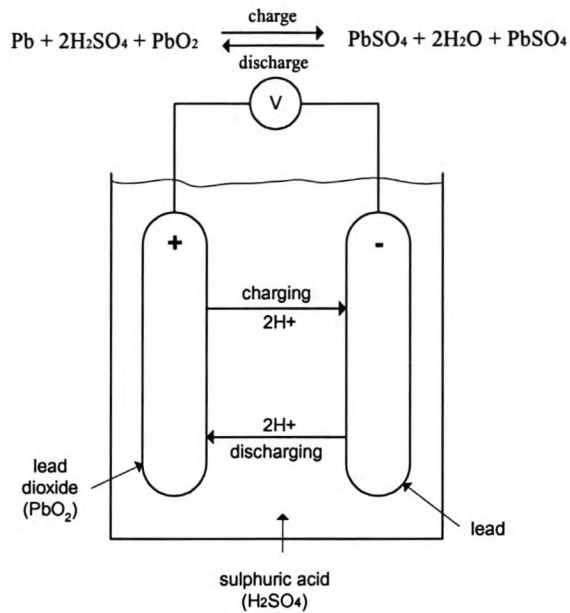
The battery consists of alternate pairs of plates, one of lead and the other lead coated with lead dioxide. These plates are immersed in a solution of sulphuric acid which serves as an electrolyte.

When the battery is discharging, energy is produced by the acid in the electrolyte gradually combining with active material of the plates. This combination produces lead sulphate in both negative and positive plates. The battery is completely discharged when both plates are entirely sulphated. The two plates are now composed of the identical material and then the voltage collapses. This is the reason why lead acid batteries cannot be fully discharged. If this state is reached it cannot be reversed.

---



Charging the battery drives the acid out of the plates into the electrolyte. When the battery is fully charged, the plates return their original state which is lead and lead dioxide. The concentration of acid in the electrolyte is now at a maximum.



**Figure 3-2: Lead Acid Battery Cell**

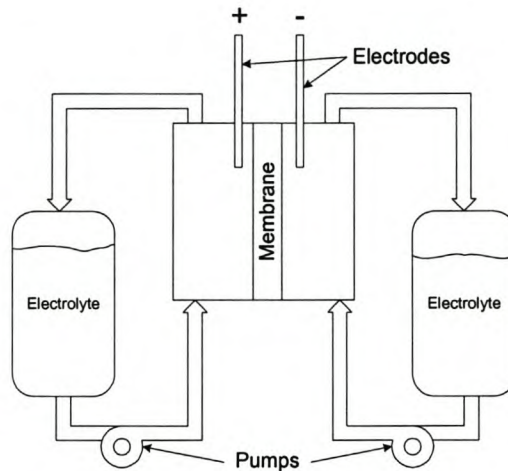
### 3.4 Flow Batteries

#### 3.4.1 Introduction

A flow battery is nothing other than an electrochemical battery where the oxidation and reduction agents are pumped through (over) the stack (electrodes). The advantage of this, over just submerging the electrodes into the agent, is that the battery capacity is limited only by the size of the storage tanks [11]. The electrolyte can also be replaced without interfering with the stack. This allows the battery to be “charged” instantly, merely by replacing the electrolyte with new, charged electrolyte. The battery can also be charged by supplying a charge voltage.

The electrodes of flow batteries act only as an electron transfer surface. By increasing the size (surface area) of the electrodes, more electrons can be

transferred and the power output of the battery is increased. The electrodes do not take part in the electrochemical process and therefore do not limit the energy storage capacity. Electrical energy is stored or released by means of reversible electrochemical reactions between two solutions (electrolytes). The electrolytes are kept separate inside the cell stack by an ion exchange membrane. [15]



**Figure 3-3: Flow Battery**

### 3.4.2 Electrochemical Cell

In an oxidation/reduction reaction (Redox), electrons are transferred from one chemical species to another [10]. A species is the agent which takes part in the chemical reaction. The species which loses electrons is oxidized and is referred to as the reduction agent. The species which gains electrons is reduced and is referred to as the oxidation agent. Usually, a redox reaction is written as two half-reactions, showing the movement of electrons from one chemical species to another. For example:



These expressions are termed half-reactions, because the equations have not yet been combined. In order to have oxidation occur, reduction must take place. This can be done by simply adding the equations above together:

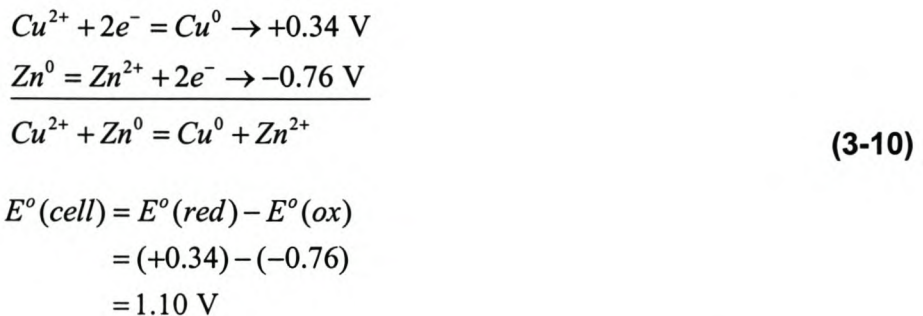


The electrons in the equation always cancel out. The final equation still represents an electron transfer, or a redox reaction, but with no electrons showing up in the equation.

Each half-reaction has a standard potential,  $E^{\circ}(\text{ox})$  and  $E^{\circ}(\text{red})$ , associated with it. The addition of the standard potentials for each half reaction results in a reaction potential,  $E^{\circ}(\text{cell})$ . If the overall reaction potential is positive, the redox reaction is spontaneous.

$$\begin{aligned}
 E^{\circ}(\text{cell}) &= E^{\circ}(\text{right}) - E^{\circ}(\text{left}) \\
 &= E^{\circ}(\text{red}) - E^{\circ}(\text{ox})
 \end{aligned}
 \quad (3-9)$$

For example:



This reaction will occur spontaneously because the cell potential is positive. If the cell potential is negative the redox reaction will not take place spontaneously.

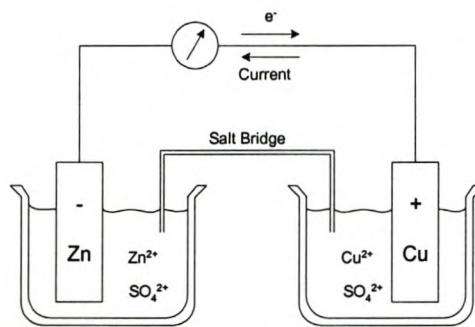
---

This does not mean that the reaction cannot take place but some sort of energy (external voltage source) has to be added to make it happen. For example:



$$\begin{aligned}
 E^\circ(\text{cell}) &= E^\circ(\text{red}) - E^\circ(\text{ox}) \\
 &= (0.00) - (+1.36) \\
 &= -1.36 \text{ V}
 \end{aligned}$$

This reaction will not occur spontaneously.



**Figure 3-4: Electrochemical Cell**

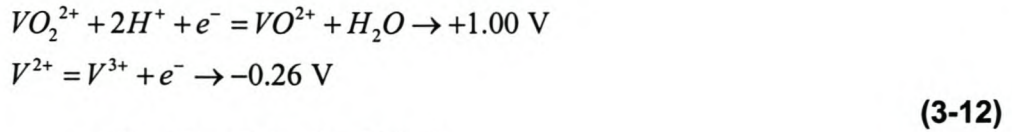
### 3.4.3 Vanadium Redox Flow Battery

#### 3.4.3.1 Principal of the Vanadium Battery

The Vanadium Redox Flow Battery (VRB) uses sulphuric acid solutions of vanadium in both electrolytes. The electrolytes are circulated by pumps that force the electrolyte through the stack. The stack is made up of electrochemical cells. A membrane (like salt bridge) in each cell allows ion exchange between the two vanadium electrolytes. Carbon fiber and carbon plate (bipolar plate) are used as

electrodes and do not take part in the reaction other than in the exchange of electrons (See Figure 3-3) [17].

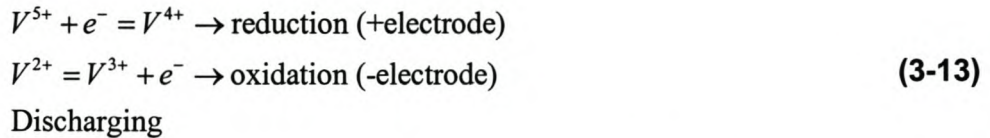
The two half-reactions for the VRB are:



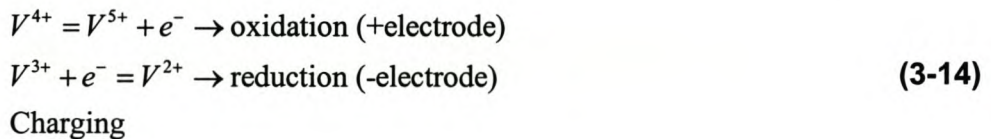
$$\begin{aligned} E^{\circ}(\text{cell}) &= E^{\circ}(\text{red}) - E^{\circ}(\text{ox}) \\ &= (+1.00) - (-0.26) \\ &= 1.26 \text{ V} \end{aligned}$$

The cell potential is positive; this implies the redox reaction is spontaneous.

These half-reactions can be simplified to:



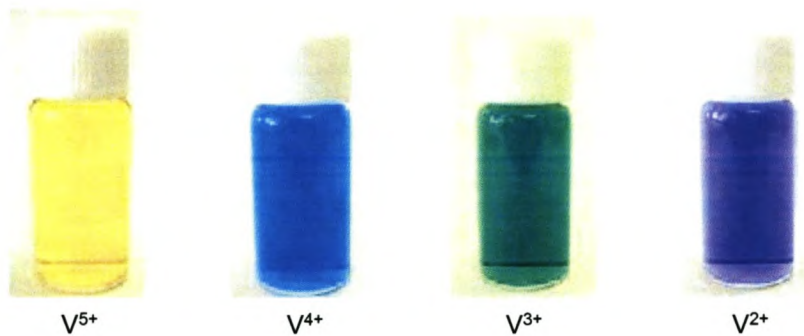
To charge the battery the redox reaction has to change around. This redox reaction can work in reverse, but this is not spontaneous. An external voltage has to be applied for the reverse redox reaction to occur, thereby charging the VRB. This is explained in section 3.4.2.



### 3.4.3.2 Features and Applications of VRB

As the VRB uses electrolytic solutions of the same species (vanadium) in both half cells, it has the advantage that the electrolyte cannot be cross contaminated and so has an indefinite life [11].





**Figure 3-5: Vanadium Electrolytes**

The advantages of the VRB are [11], [12], [15]:

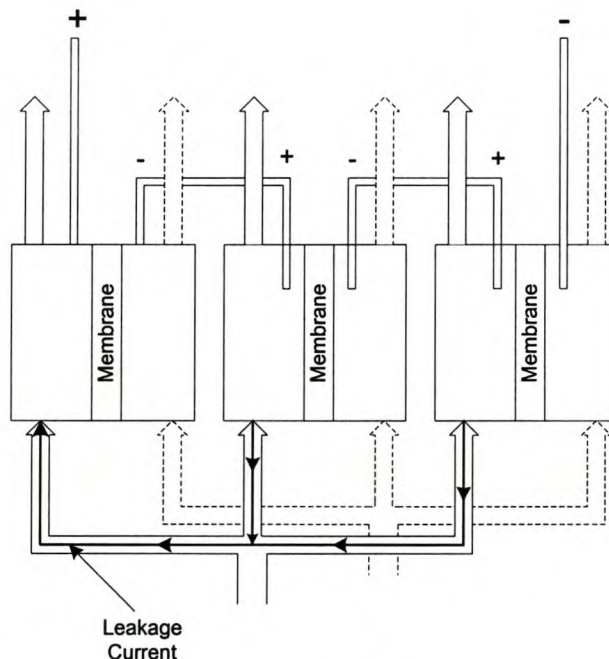
- Can be recharged at high rates in a fraction of the time needed for lead acid batteries.
- Can be fully discharged without harm to the battery.
- Has a fast response and is capable of high rate discharge over short periods.
- Long life
- Capacity can be easily increased
- Vanadium is readily available and relatively cheap

As the VRB can supply high power output and long time capacity, it is suitable for load levelling in substations and factories [13]. The VRB can also be used in power quality applications where fast response is necessary, as it displays a response time of less than 1 ms and the maximum short time overload output of several times that of normal output [14].

---

The vanadium electrolyte conducts a small leakage current. The conductive properties of the vanadium electrolyte do not allow more than a certain number of cell stacks to be placed in series. When too many cell stacks are placed in series, a leakage current flows, due to the high potential difference between the top and bottom electrodes. This is because the same electrolyte is pumped through all the stacks (Figure 3-6).

Because of this problem the output voltage of the VRB is limited to +/- 400 V and a converter is needed to step the voltage up. The VRBs charge and discharge cycles can then be controlled to specifically suite the application.



**Figure 3-6: Vanadium Electrolyte Current Leakage**

A disadvantage of the VRB is the standby losses of the battery. The vanadium electrolyte has to be pumped through the stacks at all times for the battery to have a fast reaction time. This results in losses due to the necessity of having the pumps running all the time. In most applications where VRBs are used it is required that the system is active all the time.

### **3.5 Conclusion**

To be able to utilize a wide range of energy storage types in the same power system, the energy storage needs to be controlled. This can be done by connecting the energy storage to a converter. The output voltages for lead acid batteries and flow batteries can be stepped up or down and dc can be converted to ac and ac to dc for controlling a flywheel.

For the successful control of these storage types, various currents and voltages have to be measured and the measurements then used by a digital controller to control the converter, which in turn controls the energy storage.

Table 3 shows the requirements for controlling each of the energy storage systems with the second converter added to the UPS system.

**Table 3: Energy Storage Control System Requirements**

	Lead acid batteries	Flow batteries	Flywheels
Processor	Yes	Yes	Yes
PWM outputs	8	8	8
Error inputs	8	8	8
Relay outputs	4	4	4
Voltage measurements	2	2	4
Current measurements	5	5	5
Additional sensors	0	0	Speed/Position



## ***4. Development of the Generic Energy Storage System***

### ***4.1 Introduction***

The MC was designed to control the single converter compensator. As mentioned in section 2.3.1 the MC cannot handle the additional processing and also does not have sufficient I/O ports to control a second converter and energy storage. The existing measuring system on the MC is not optimally suited for the measuring of waveforms containing high harmonic content (discussed in section 2.3.3.1). The Generic Energy Storage System (GESS) was developed to control the energy storage for the dual converter line-interactive compensator discussed in Chapter 2.

The GESS is a multi-functional extension of the MC. The GESS consists of three modules: the Generic Energy Storage Controller (GESC), Generic Energy Storage Measuring System (GESMS) and an Input/Output Board (IOB). The function of the GESS is to control a wide range of energy storage types, with the focus on lead acid batteries, flow batteries and flywheels discussed in Chapter 3, and incorporate them into the existing line-interactive compensator.

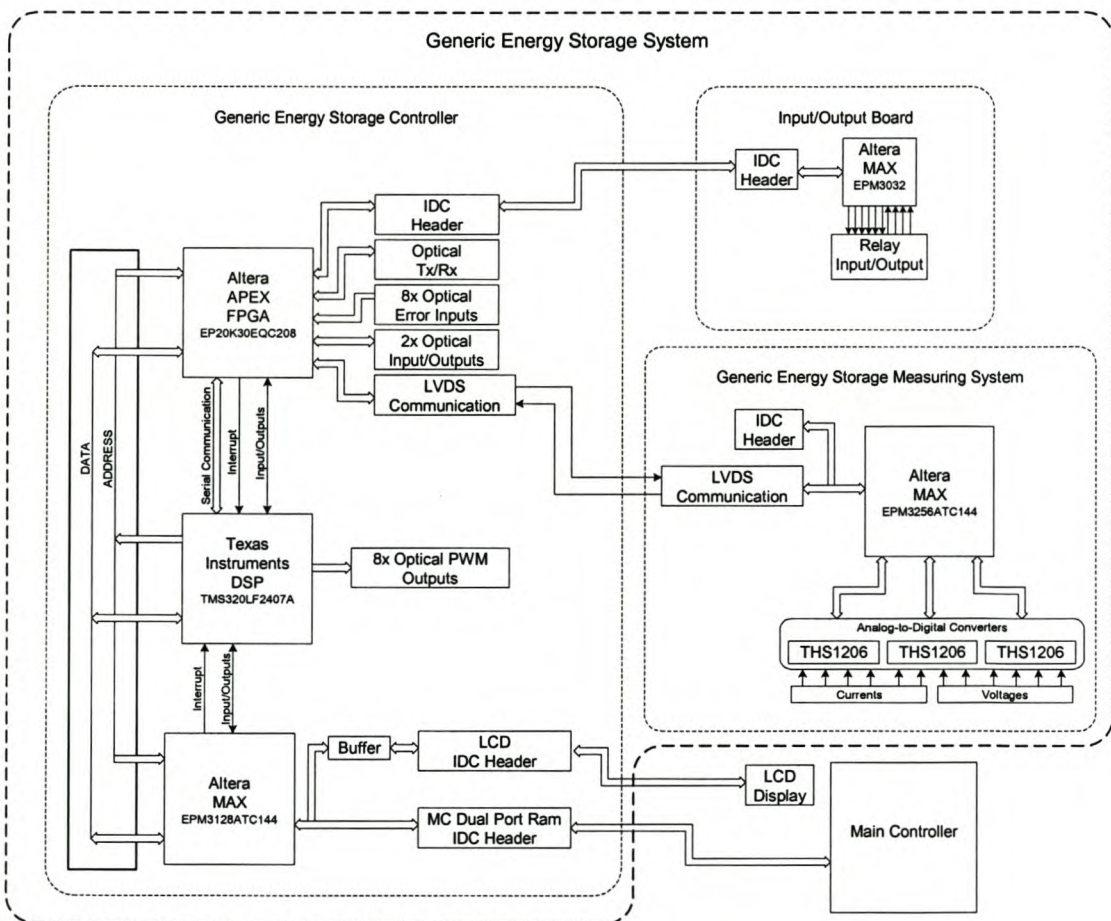
### ***4.2 System Description***

The successful control of an energy storage system requires the following basic components:

- Digital Signal Processor
  - Programmable Logic Devices
  - Inputs and Outputs
  - Measurements
  - Communications
-

The following ports for interfacing were identified:

- 4x Optical PWM pairs to switch 4 phase arms
- 8x Optical error inputs from the converter
- 6x Current measurements
- 6x Voltage measurements
- 4x Relay inputs
- 6x Relay outputs
- 1x Optical Tx/Rx serial communication
- 2x Optical inputs/outputs
- 1x LVDS port to measuring system
- 1x IDC header to Main Controller
- 1x IDC header to LCD display



**Figure 4-1: Generic Energy Storage System Overview**

### **4.3 Generic Energy Storage Controller**

The Generic Energy Storage Controller (GESC) consists of a DSP (Digital Signal Processor), FPGA (Field Programmable Gate Array), EPLD (Erasable Programmable Logic Device) and miscellaneous drivers and interfaces. The DSP executes a software programme that implements the control algorithms. The FPGA is responsible for processing the data from the measuring system, input/output board and error signals from optical inputs. The measurement data is received from the high speed LVDS connection. The EPLD controls the data flow between the DSP, MC and LCD display.

#### **4.3.1 Digital Signal Processor**

The DSP used for the GESC is the TMS320LF2407A from Texas Instruments. This a medium-performance DSP specially targeted at drives and other power conversion applications. Features of the TMS320LF2407A are: [20]

1. High-Performance Static CMOS Technology
    - 25-ns Instruction Cycle Time (40 MHz)
    - 40-MIPS Performance
  2. On-Chip Memory
    - Up to 32K Words x 16 Bits of Flash EEPROM (4 Sectors) or ROM
    - Programmable "Code-Security" Feature for the On-Chip Flash/ROM
    - Up to 2.5K Words x 16 Bits of Data/Program RAM
      - 544 Words of Dual-Access RAM
      - Up to 2K Words of Single-Access RAM
  3. Boot ROM
    - SCI/SPI Bootloader
  4. Two Event-Manager (EV) Modules (EVA and EVB), each Includes:
-

- Two 16-Bit General-Purpose Timers
- Eight 16-Bit Pulse-Width Modulation (PWM) Channels which enable:
  - Three-Phase Inverter Control
  - Centre or Edge-Alignment of PWM Channels
  - Emergency PWM Channel Shutdown with External PDPINTx\ Pin
- Programmable Deadband (Deadtime) Prevents Shoot-Through Faults
- On-Chip Position Encoder Interface Circuitry

Each is:

- Designed for AC Induction, BLDC, Switched Reluctance, and Stepper Motor Control
- Applicable for Multiple Motor and/or Converter Control

5. External Memory Interface

- 192K Words x 16 Bits of Total Memory:  
64K Program, 64K Data, 64K I/O

6. Watchdog (WD) Timer Module

7. Controller Area Network (CAN) 2.0B Module

8. Serial Communications Interface (SCI)

9. 16-Bit Serial Peripheral Interface (SPI)

10. Phase-Locked-Loop (PLL)-Based Clock Generation

11. Up to 40 Individually Programmable, Multiplexed General-Purpose Input/Output (GPIO) Pins

12. Up to Five External Interrupts (Power Drive Protection, Reset, Two Maskable Interrupts)

13. Power Management:

- Three Power-Down Modes
- Ability to Power Down Each Peripheral Independently

14. Real-Time JTAG-Compliant Scan-Based Emulation

---

### 4.3.1.1 DSP Configuration

The Code Composer integrated development environment is used as programming platform. This program is specifically targeted at configuring Texas Instruments DSPs. Code Composer configures the device through a JTAG port that allows the user to have almost total control over the DSP. This enables realtime debugging via the JTAG connection to the PC (Personal Computer) and is very handy when new software is developed. The XDS510PP JTAG POD from Spectrum Digital is used as connection between the PC and GESC. The JTAG device is very expensive and is usually used only when new software is developed. When in the field the DSP can be booted through the serial port.

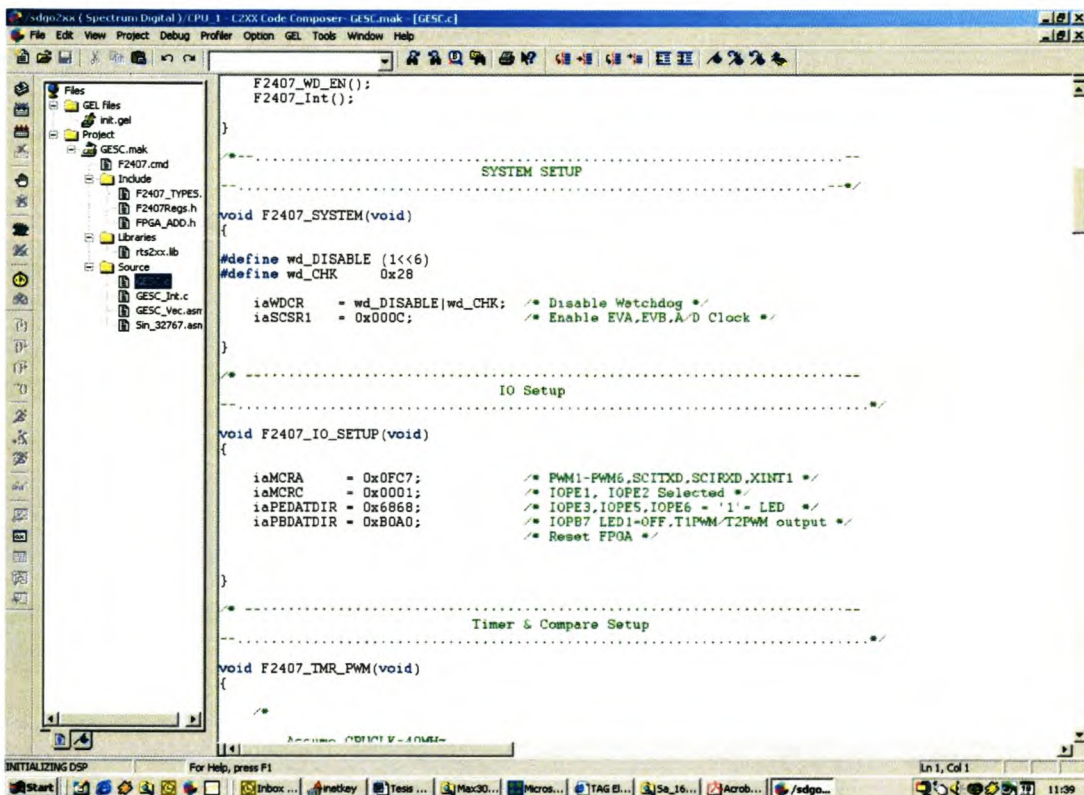
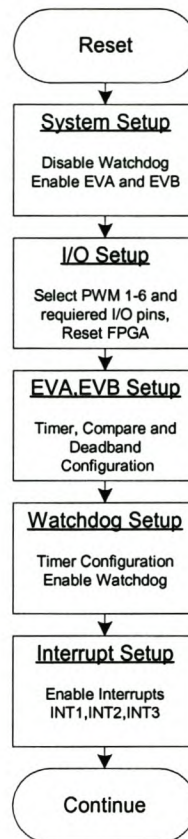


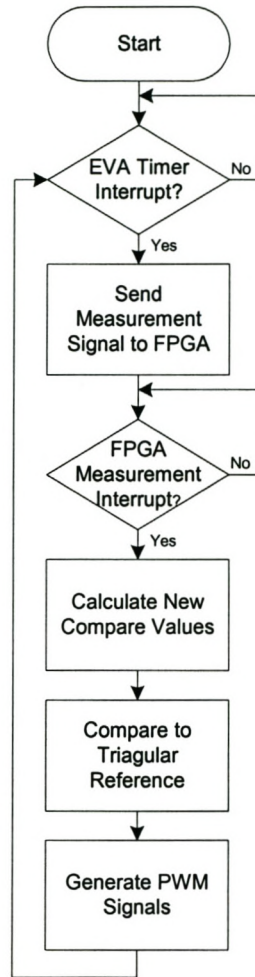
Figure 4-2: Code Composer Programming Environment

### 4.3.1.2 DSP Software Setup



**Figure 4-3: DSP Software Configuration**

After reset all the DSP required functions must be configured. As soon as the interrupts are enabled, the whole system starts. An up-down counter (triangular wave) in EVA activates an interrupt every 5 KHz. The DSP sends a signal to the FPGA to request the measurements. The FPGA generates an external interrupt to let the DSP know the measurements are ready. The DSP reads the required measurements from the FPGA. New compare values are calculated and loaded into the compare registers. The PWM signals are generated by comparing these values to the triangular wave.



**Figure 4-4: Interrupt Flow**

#### 4.3.1.3 DSP JTAG Interface

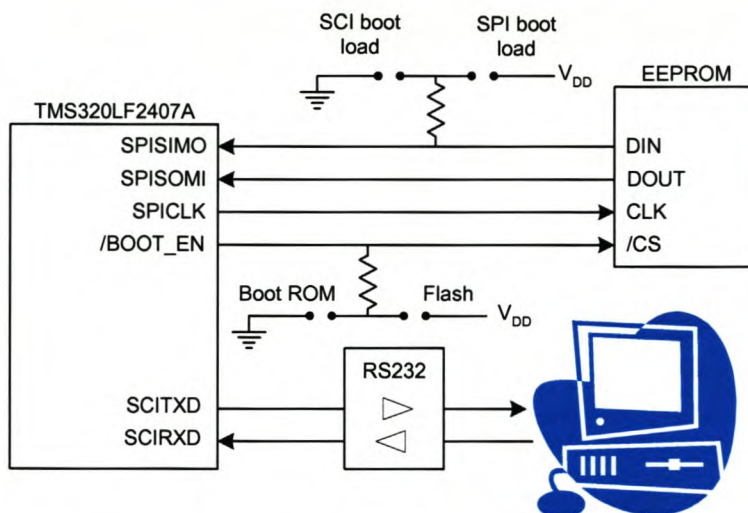
The 14 pin IDC header on the GESC is the standard interface used by emulators to interface to Texas Instruments DSPs. The pinouts for the connector are shown in Figure 4-5.

TMS	1	2	/TRST	TMS	Test mode select
TDI	3	4	GND	TDI	Test data input
+5V	5	6	no pin(key)	TDO	Test data output
TDO	7	8	GND	TCK	Test clock at 10.368MHz
TCK-RET	9	10	GND	/TRST	Test reset
TCK	11	12	GND	EMU0	Emulation pin 0
EMU0	13	14	EMU1	EMU1	Emulation pin 1
				TCK-RET	Test clock return
				GND	Ground

**Figure 4-5: JTAG Connector Pinout**

**4.3.1.4 Boot ROM Loader**

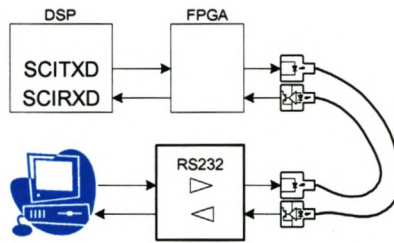
The DSP includes on chip read only memory (ROM) containing bootloader code. This code loads code from an external serial boot device (PC or EEPROM) at reset and transfers control to the code loaded from external device. There are two boot ROM options. Code can be loaded through either asynchronous or synchronous serial transfer. The synchronous transfer is done through the serial peripheral interface (SPI), and the asynchronous transfer is done through the serial communications interface (SCI). The selection between the two can be seen in Figure 4-6: F2407A Boot ROM Configuration.



**Figure 4-6: F2407A Boot ROM Configuration**



The internal flash memory of the DSP is sufficient and the GESC does not need to boot from an on board external memory device. The SCITXD and SCIRXD pins are connected through the FPGA to an optical receiver and transmitter. The reason for this is to galvanically isolate the serial communications to the outside. When GESC needs to be booted from a PC an optical to RS232 conversion has to be made.



**Figure 4-7: GESC Boot ROM Configuration**

### 4.3.2 GESC Memory Interface

The GESC can access 2k Words of external memory. This is split equally between the FPGA and EPLD. The external memory is mapped in data space from 8000h to 87FFh.

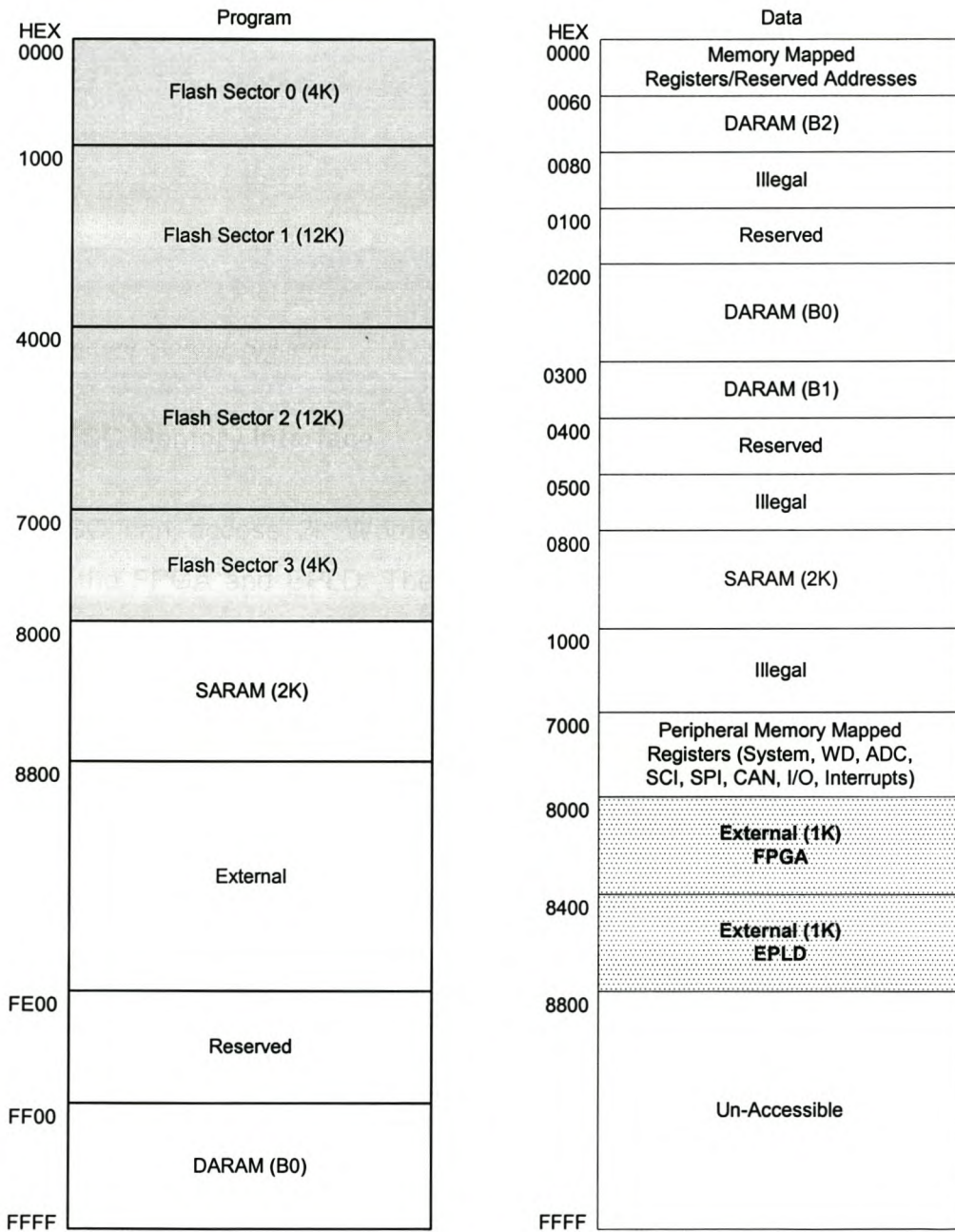


Figure 4-8: GESC Memory Map

The external memory interface of the DSP has 16 bit data and address lines. In the GESC only the eleven least significant address lines are used. This implies that 2048 words of external data can be accessed. The DSP, EPLD and FPGA share the same data bus. The DSP has total control over the data bus. The memory is split into 1024 words each for the FPGA and EPLD.

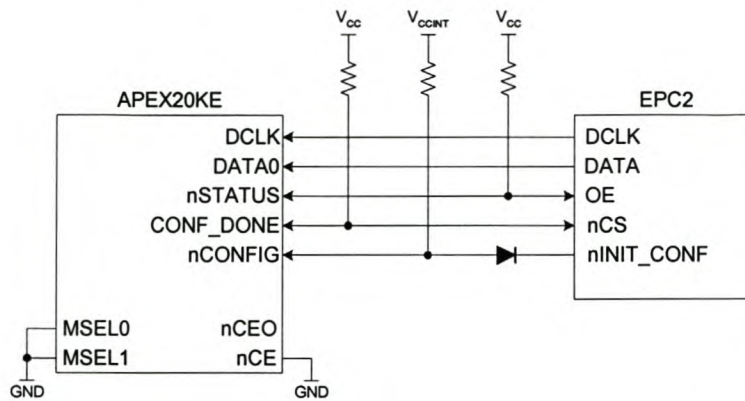
**Table 4: Address Decoding**

A[10..0]	10	9	8	7	6	5	4	3	2	1	0
<b>EPLD</b>	1	X	X	X	X	X	X	X	X	X	X
<b>FPGA</b>	0	X	X	X	X	X	X	X	X	X	X
Measurements	0	0	0	0	0	X	X	X	X	X	X
Write to outputs	0	0	0	0	1	0	X	X	X	X	X
Read from inputs	0	0	0	0	1	1	X	X	X	X	X

Table 4 show how addresses are dedicated to specific sections in the FPGA. The FPGA does not use the full 1024 words of memory but only 128 words. This is more than sufficient at this stage. It can easily be re-configured by changing the firmware in the FPGA. At this stage the GESC is used as a stand alone controller and does not need to communicate with the MC. The 1024 addresses which are dedicated to the EPLD are reserved for communications with the MC.

#### **4.3.3 Field Programmable Gate Array**

The APEX EP20K30EQC208-2X FPGA from Altra is used. This device contains 30000 gates, 1200 logic elements, 24376 bits of RAM and up to 125 user I/O pins. The APEX20KE includes two PLLs (Phase Lock Loops) [23]. This feature enables the multiplication and division of clock signals. The FPGA has a volatile internal memory (If the device is switched off it loses its configuration). The FPGA is configured by an EPC2 configuration device at start-up. The EPC2 can be re-programmed and has non-volatile memory.



**Figure 4-9: Configuration Device Connection**

*Error Detection:* The FPGA combines all the optical error input signals from the converter and LVDS communication link. If an error is detected the power drive protect interrupt (PDPINTA) pin on the DSP is driven low. This puts the PWM outputs in a high impedance state. The power drive protect interrupt (PDPINT) is implemented on hardware level to minimise the time delay for the converter to trip.

*Measurement Data:* The FPGA also receives the measurement data through the LVDS communication link. This data is processed and then stored in dual port RAM implemented in the FPGA. The DSP can then access the data when needed.

*Input Output Board:* The IOB is also controlled by the FPGA. The DSP can write the required values for the outputs into the output register. The FPGA then sends the values to the IOB. The inverse is implemented for the inputs from the IOB.

The firmware will be discussed in Chapter 5.

#### 4.3.4 Erasable Programmable Logic Device

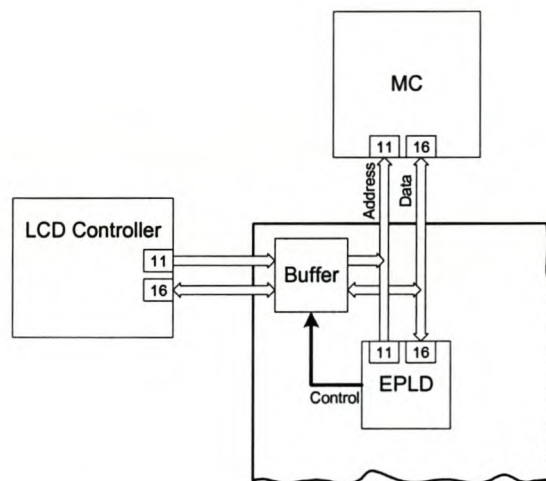
The EPM3128ACT144 EPLD from Altera is used. The device has 2500 gates and 128 macrocells. This is a multi-voltage device and is 2.5V, 3.3V and 5V I/O

compatible. This is necessary because the MC and LCD controller uses 5V and the GESC 3.3V.

The GESC uses the header which was previously dedicated for communications with the LCD controller in order to connect to the MC. This communications port is shared between the GESC and the LCD controller. The GESC uses a piggy back connection that slots onto the IDC header on the MC. The LCD controller then connects to a header on the GESC. The LCD controller still has priority on data flow; this means that the GESC must always check if the connection is available before data exchange with the MC.

The EPLD controls the data flow between the MC, LCD controller and GESC. To avoid the interference from the LCD controller when the GESC and the MC are exchanging data, the data and address lines are disconnected with a buffer.

The firmware will be discussed in the next Chapter.

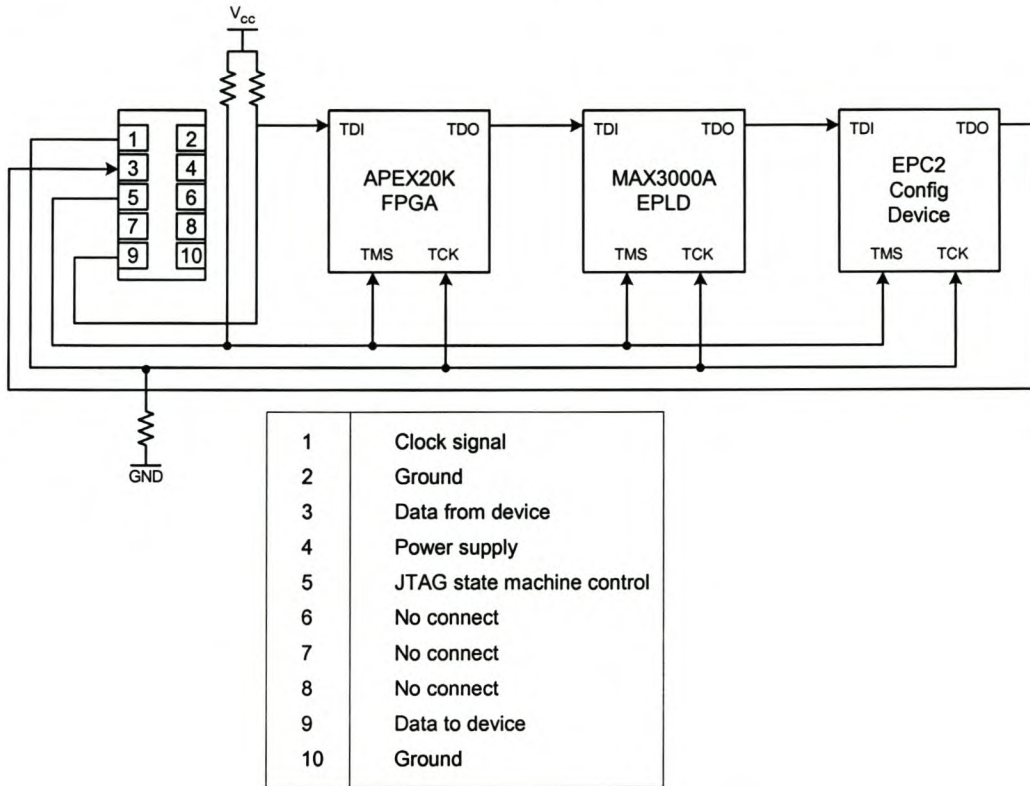


**Figure 4-10: Connection between the GESC, MC and LCD Controller**

#### 4.3.5 PLD JTAG Interface

All the PLDs are configured by a single JTAG interface dedicated to the PLDs. This is done with a ByteBlaster JTAG cable connected to a Personal Computer.

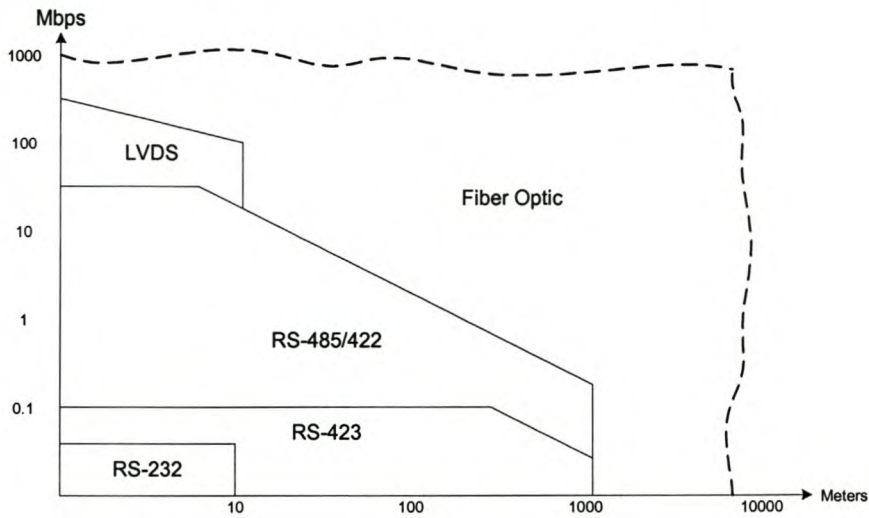
The PLDs are connected in a JTAG chain. This enables the Quartus II configuration software to detect all the devices connected. The firmware development is also done in the Quartus II programming environment.



**Figure 4-11: PLDs JTAG Chain**

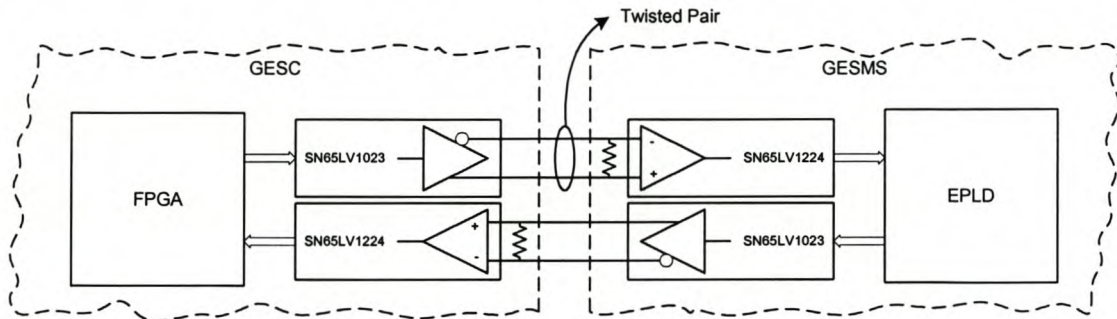
#### 4.3.6 LVDS Communication Port

The Low Voltage Differential Signalling (LVDS) is a high bandwidth communication medium. Data transfer rates of 100Mbps to 660Mbps can be achieved. These transfer rates compare with transmission over optical fibre, but at a fraction of the cost. Working in a high noise environment, such as a high power UPS system, LVDS decreases noise interference. The differential nature of LVDS enables the communication link to operate in relatively high levels of EMI, as it rejects most of the common mode noise.



**Figure 4-12: Data Rate vs. Cable Length**

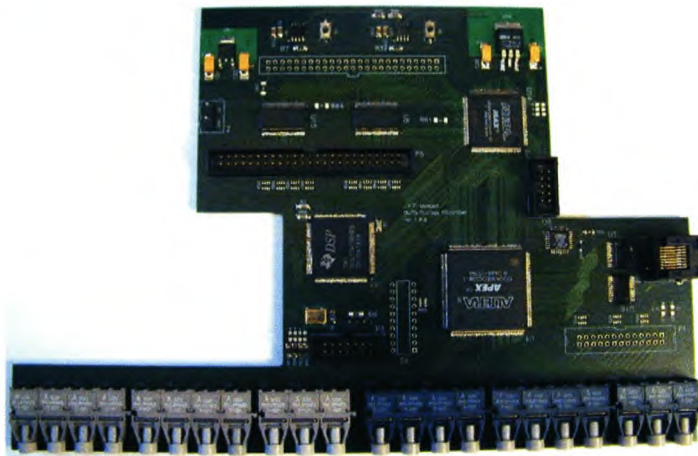
The reason why LVDS is so effective is the differential pair transmission. This is also a critical part of the design. Common noise can be rejected only if the noise on each transition path is the same. When routing the PCB these transmission paths must be of similar length and kept as close as possible to each other.



**Figure 4-13: LVDS Communication Link**

To connect the GESC to the GESMS, modular jacks with shielded, twisted pair network cable are used. The LVDS port handles data transfer to and from the measuring system. It is important to have a reliable high speed data link to the measuring system. Communication link failure or faulty measurements being received can lead to the destruction of the converter.

The SN65LV12021/SN65LV1212 LVDS Serializer/Deserializer chip set is used. The 10 bit chipset transmits and receives serial data over LVDS differential backplanes. The serializer clocks the parallel data in and converts it to a serial differential signal. The clock information is encoded in the data. The deserializer converts the differential signal back to a 10 bit parallel output. The deserializer extracts the clock signal from the data. The receive clock locks onto the extracted clock. When the frequencies lock the LVDS lock signal goes low. New data arrives at the rising edge of the receive clock and can be used.



**Figure 4-14: Photograph of the GESC**

#### **4.4 Generic Energy Storage Measuring System**

A new Generic Energy Storage Measuring System (GESMS) was developed in combination with the GESC to overcome the shortcomings of the existing measuring system on the MC.

The GESC consists of three THS1206 10 bit analog to digital converters (ADC) from Texas Instruments, with the ability to sample four measurement channels at 1.5 MSPS. The data is captured and the ADCs are configured by an EPLD. The data is processed and then sent at 200 Mbps over the LVDS communication to

---



the GESCS. The measurement data is received and processed by the FPGA on the GESCS. The GESMS has a 20 MHz crystal oscillator. All the timing on the GESMS is controlled by the EPLD.

On the existing measuring system the voltages and currents are sampled at 5 kHz, which is the same as the converter switching frequency. This poses problems, because aliasing can occur due to the high frequency content on the measured signals.

There are two solutions to this problem. The measured signals can be filtered before sampling by the ADC or the signal can be over-sampled by the ADC. With over-sampling, the signal is sampled at a higher rate and filtered using a digital filter. The filter is typically a moving average filter.

The GESMS samples 6 current and 6 voltage measurements at 833.33 KSPS. The FPGA take each measurement sample and adds it to the previous samples. The DSP sends a signal to the FPGA when it requires a measurement; the accumulated value is stored, together with the number of values added. The DSP reads the data and works out the average. The average of 167 samples within a 5 KHz period is used in the control calculations. This removes the high frequency content on the measured signals.

#### **4.4.1 Analog to Digital Conversion**

*Digital Section:* The THS1206 is a CMOS, 12 bit, 6 MSPS analog to digital converter (ADC). An EPLD programs the internal control registers of the ADC to the desired mode. The THS1206 consists of four analog inputs, which are sampled simultaneously. Each of the inputs can be selected independently, or two inputs can be combined as a differential input. The ADC can operate in single conversion mode or continuous conversion mode. These options depend on the values written to control registers CR0 and CR1. An internal reference voltage of 2.5V is provided.

---

**Table 5: Bit Definitions of ADC Control Registers**

REG	BIT9	BIT8	Bit7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CR0	TEST1	TEST0	SCAN	DIFF1	DIFF0	CHSEL1	CHSEL0	PD	MODE	VREF
CR1	RBACK	OFFSET	BIN/2's	R/W	DATA_P	DATA_T	TRIG1	TRIG0	OVFL	RESET

**Table 6: Control Register C0 Bit Functions**

BITS	NAME	FUNCTION
0	VREF	0 = The internal reference voltage is selected; 1 = The external reference voltage is selected
1	MODE	0 = Continuous conversion mode selected; 0 = Single conversion mode
2	PD	0 = ADC active; 1= Power down
3,4	CHSEL	Analog input selection
5,6	DIFF	Number of differential inputs
6	SCAN	1 = Autoscan through analog inputs
8,9	TEST	Test inputs

**Table 7: Control Register C1 Bit Functions**

BITS	NAME	FUNCTION
0	RESET	Writing a 1 into this bit resets the device
1	OVFL	Indicates an overflow in the FIFO
2,3	TRIG	Sets the trigger level for the FIFO register
4	DATA_T	0 = DATA_AV output is a puls; 1 = DATA_AV output active level
5	DATA_P	0 = DATA_AV is active low; 1 = DATA_AV is active high
6	R/W	R/w or RD,/WR selection
7	BIN/2s	1 = Output twos complement; 0 = output normal binary
8	OFFSET	Offset cancellation
9	RBACK	Debug mode

For GESCS the three ADCs are set up for continuous conversion of all four analog inputs. The sample rate is equal to the conversion clock divided by the number of channels selected, in this case four. A conversion clock of 3.33 MHz is used. Each of the twelve channels is sampled at 833.33 KHz.

The EPLD plays a complex part in the GESMS. First, the EPLD configures the ADCs and then activates the conversion clock. The data of three measurements from three ADC arrive at the same time. All that data has to be transmitted over the LVDS port before the next set of data arrives. This requires complex timing analysis.

The EPLD used for the GESMS is the same as on the GESCS. The only difference is that this device has 5000 gates and 256 microcells, which is twice the number that is used on GESCS. The ADCs use a 5V supply and the LVDS chip set 3.3V. This utilizes the multivoltage ability of the EPLD.

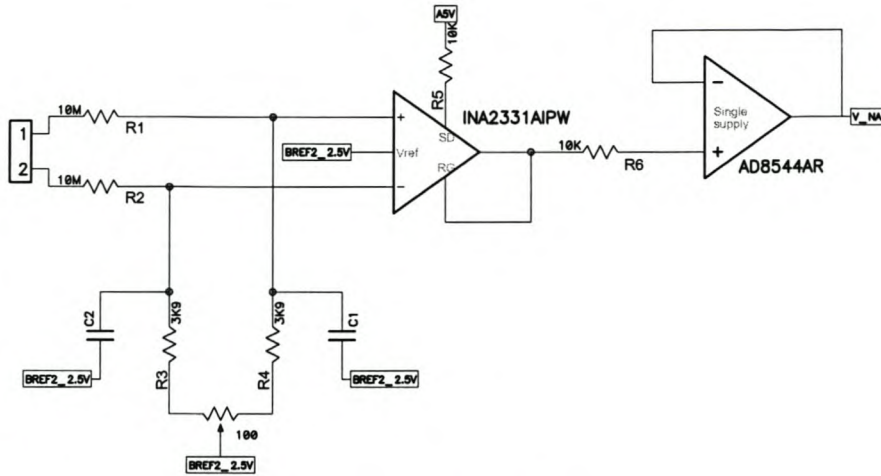
The firmware for the GESMS EPLD will be discussed in the next Chapter.

*Analog Section:* In Figure 2-15 and Figure 2-16 the current and voltage measurements can be seen (design will follow below). This measuring system was developed for a four-phase converter. To control the converter in dc to dc configuration, the two dc voltages (dc-bus and output voltage) are measured. The currents in each phase and the dc-bus current are measured. The total measurements for the dc to dc configuration are two voltages and five currents.

For the ac to dc configuration, the voltage from each phase to the switched neutral, and the dc bus voltage are measured. Four phase currents and the dc bus current have to be measured. The total measurements for the ac to dc configuration are four voltages and five currents.

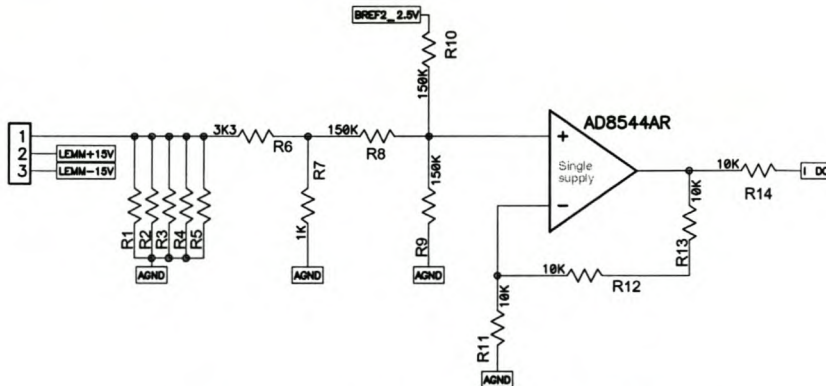
The GESMS has six voltage and six current measurements for flexibility.

---



**Figure 4-15: Voltage Measurement**

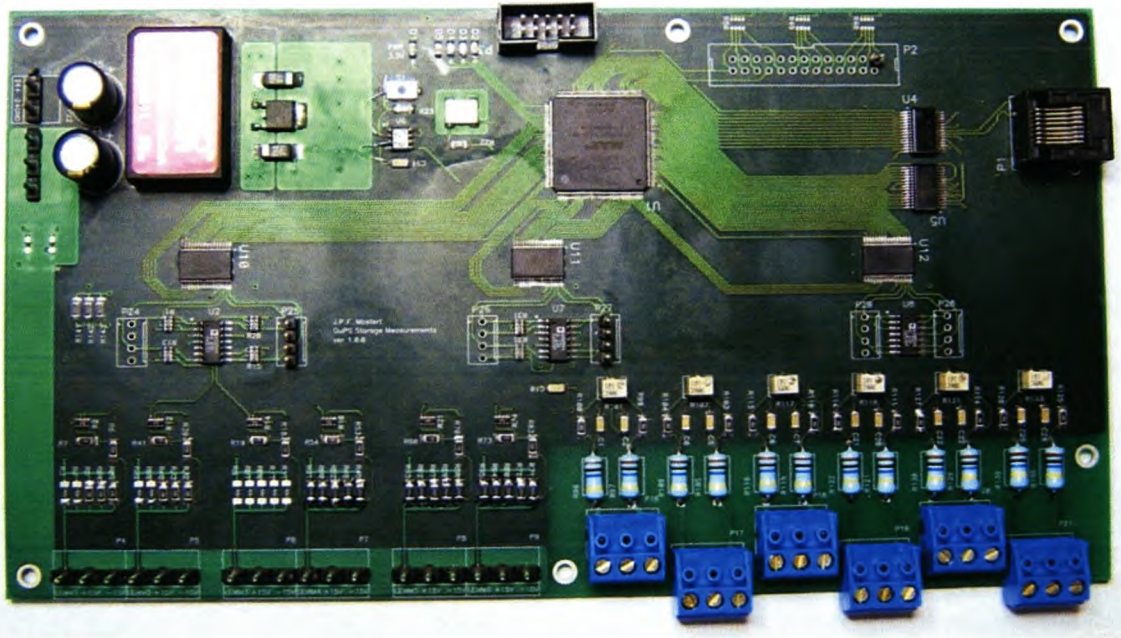
The INA2331 instrumentation amplifier was used in the voltage measurement [34]. It is a low cost, single supply device from Texas Instruments. The inputs on the PCB connect directly across the measured voltage. Great care was taken with the clearances on the PCB. The voltage measurement was designed to measure voltages up to 500 V. Because of the ADC input voltage range the output was biased around 2.5 V.



**Figure 4-16: Current Measurement**

The AD8544 operational amplifier was used in the current measurement [35]. It is a single supply, wide bandwidth, quad amplifier from Analog Devices. The inputs connect to a CT or Hall Effect current sensor. Depending on the gain (turns ratio) of the sensor, resistors R1 to R5 can be chosen so that the maximum measured

current representation falls inside the voltage range of the ADC. The output is also biased around 2.5 V because of the ADC input voltage range.



**Figure 4-17: Photograph of the GESMS**

#### **4.5 Input Output Board**

The Input Output Board (IOB) can handle four inputs and six outputs. The outputs are “form C dry relay contacts”. The inputs are optically isolated and are used to sense dry contact outputs. These inputs and outputs are usually used to drive bigger switches. Before connecting the batteries to the dc bus the voltages must be equal, otherwise large currents will flow.

A small EPLD receives and sends the digital values to the GESCS.

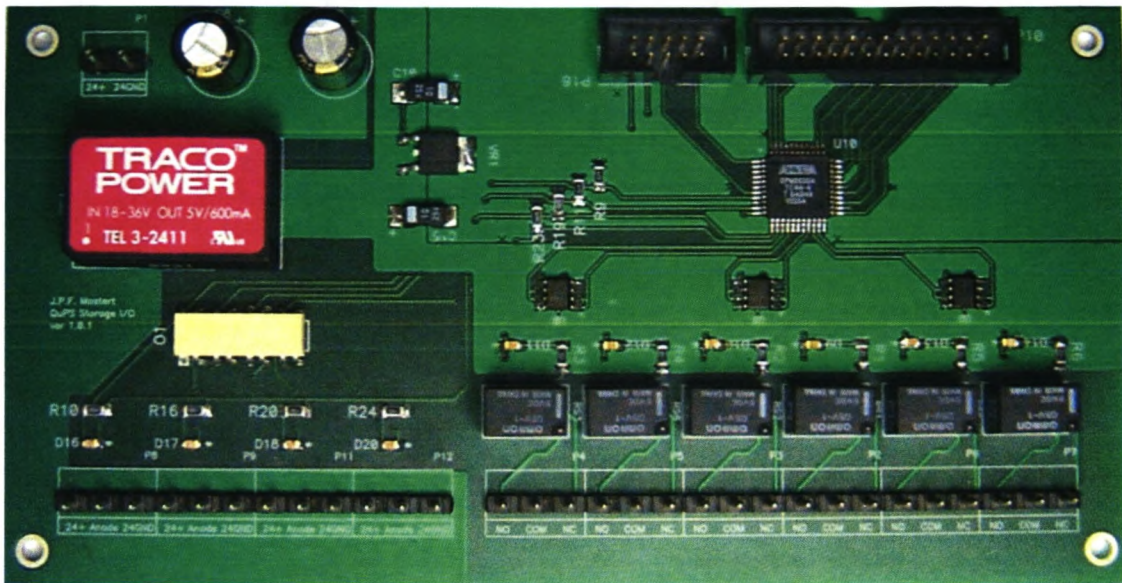


Figure 4-18: Photograph of the IOB

## ***5. Development of Firmware for PLDs***

### ***5.1 Introduction***

In today's fast changing electronic environment flexibility is very important. PLD allows the user to re-configure physical gates inside a device for a specific task. This gives the designer a flexible hardware setup.

When interfacing with different controllers and devices which do not require the same hardware configurations, PLDs is a necessity. This is the case with the GESS. The GESS has to be able to work as a stand alone system or interact and work in combination with the MC. When, in future, different energy storages are incorporated and added to the UPS system, the firmware on the PLDs can be re-configured to suit the specific application.

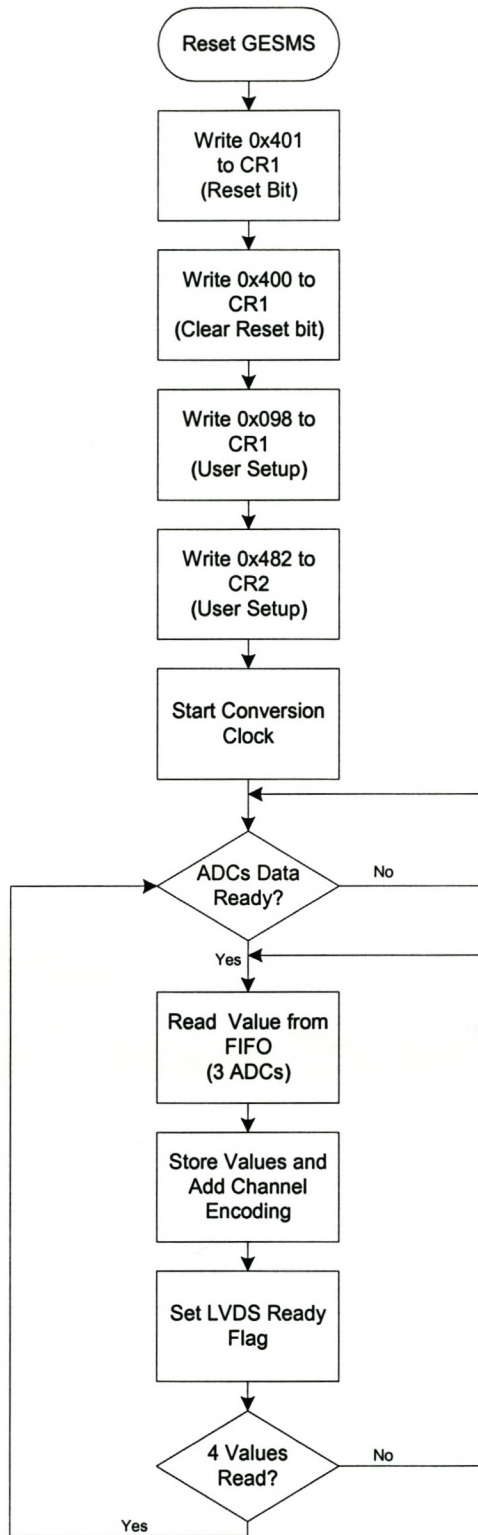
In this chapter the firmware design of all the PLDs is discussed. The description of firmware on the PLDs is not discussed from device to device, but rather as firmware sections according to the application. Each section can include more than one PLD. The main firmware sections are: data flow from the ADCs to the DSP, input output board data flow, error detection, and data flow from the GESMS to the MC.

### ***5.2 Measurement Data Flow from ADC to the DSP***

Before the DSP can read the required (specific) measurement value from the FPGA, the ADC has had to sample the measurement, the GESMS EPLD has received the value and sent it through the LVDS connection to the FPGA where the data is processed and stored. This happens 167 times for 12 channels in every 5 KHz period. The firmware for the measurement data flow involves the GESMS EPLD and FPGA.

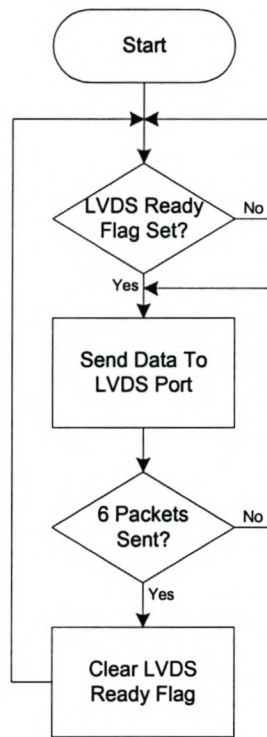
In Figure 5-1 and Figure 5-2 the flow charts show the working of the firmware on the GESMS EPLD.

---



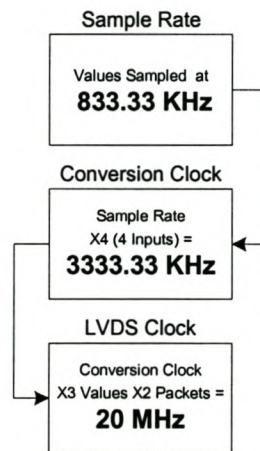
**Figure 5-1: Receiving Data from ADCs on the GESMS**



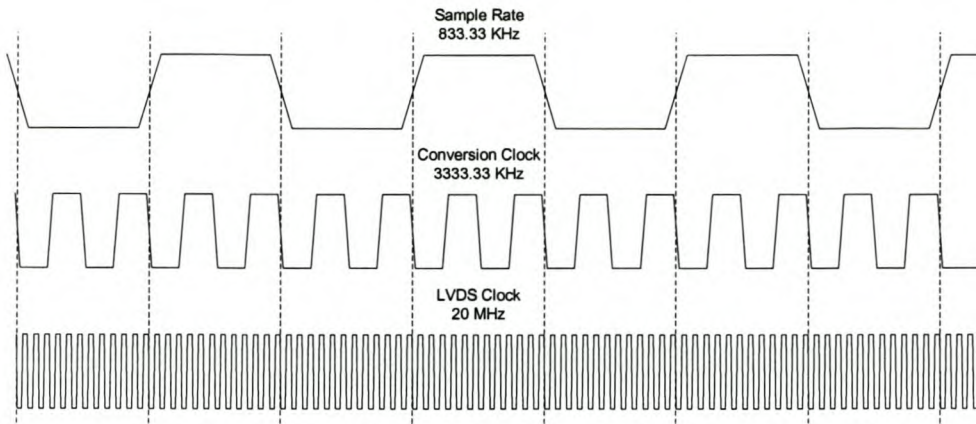


**Figure 5-2: Sending Measurement Data over LVDS Port**

All the timing on the GESMS is done by the EPLD. The three ADCs use the same conversion clock. This implies that for each conversion clock cycle, three 12 bit values are sampled. Then the data is encoded to be able to differentiate between the measurements. Before the next sampled values arrive all the previous data has had to be sent.



**Figure 5-3: Clock Signals on GESMS**



**Figure 5-4: Clock Frequency Comparison**

The LVDS port has only a 10 bit parallel input. This means each value must be sent in two packets over the LVDS port to the FPGA. Within the conversion clock period the EPLD has to send six packets of measurement data.

**Table 8: Encoding Measurement Data**

	LVDS Packet 1										LVDS Packet 2									
	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Start Sequence Code	0	1	0	1	0	1	A/D Bit 11	A/D Bit 10	A/D Bit 9	A/D Bit 8	1	0	A/D Bit 7	A/D Bit 6	A/D Bit 5	A/D Bit 4	A/D Bit 3	A/D Bit 2	A/D Bit 1	A/D Bit 0
Normal Code	0	0	0	1	0	1	A/D Bit 11	A/D Bit 10	A/D Bit 9	A/D Bit 8	1	0	A/D Bit 7	A/D Bit 6	A/D Bit 5	A/D Bit 4	A/D Bit 3	A/D Bit 2	A/D Bit 1	A/D Bit 0

In Table 8 it can be seen how the data is encoded. It is split up into two 10 bit packets. The first of the twelve measurement data is encoded differently. The measurements are always sent in the same order. The start of sequence code is used to decode the data and ensure that the measurements are stored in the correct addresses on the FPGA. The FPGA checks if the data is received correctly by making sure the right codes are received.

In Figure 5-5 the flow chart shows how the measurement data on the FPGA is processed.

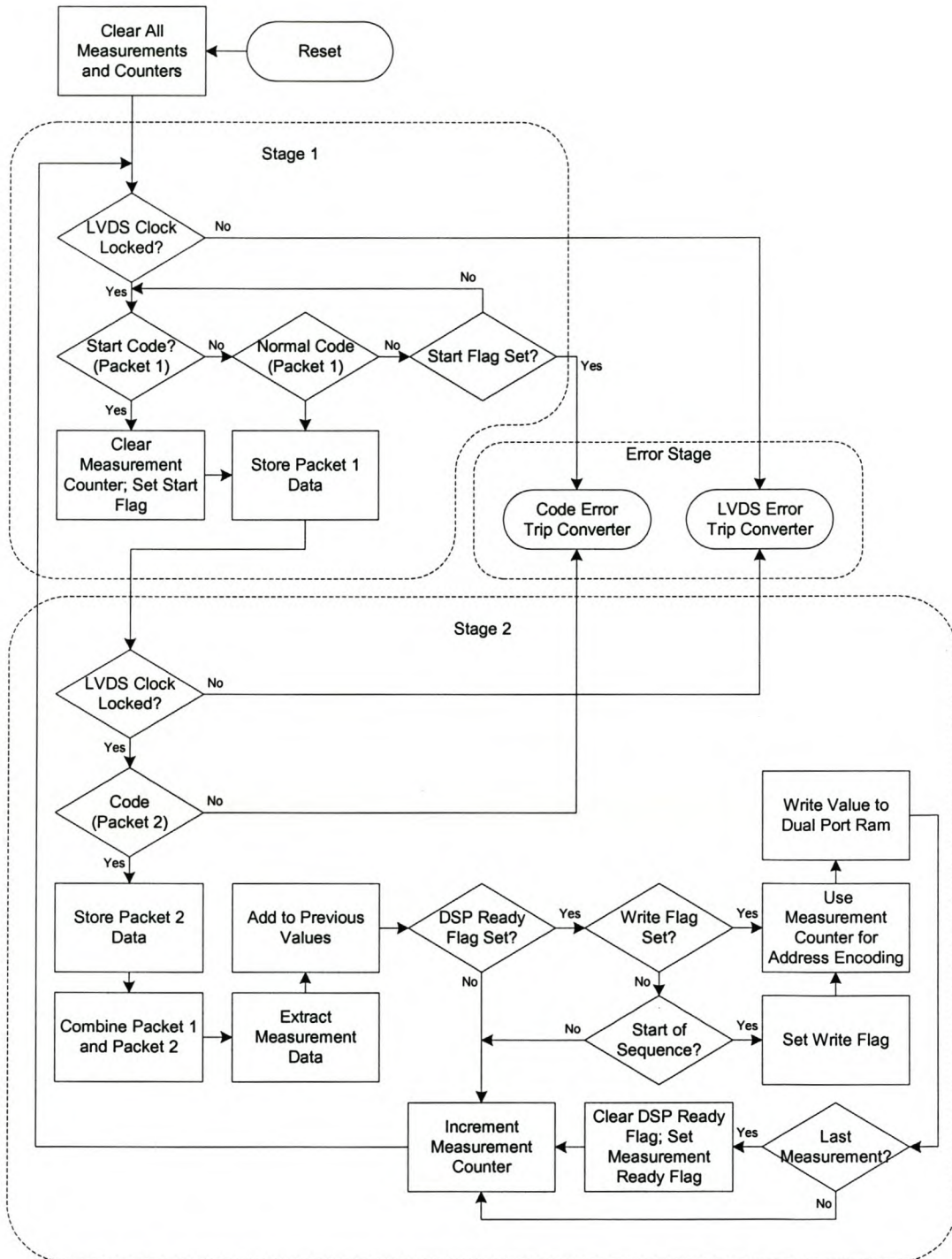
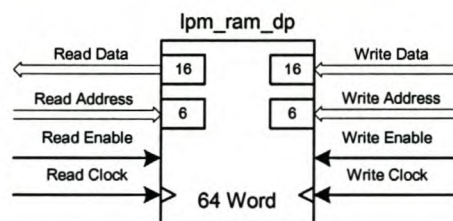


Figure 5-5: Processing of the Measurement Data in FPGA

Before any LVDS data can be received, the receive clock of the deserializer has to lock onto the clock encoded in the incoming data. When the frequencies lock the LVDS lock signal goes low. The FPGA samples the 10 bit data at the rising edge of the receive clock. The measurement data is always sent in the same order. The FPGA takes the two LVDS packets and extracts the measurement value. That value is then added to the previous values. When the DSP requires the measurement values, the DSP ready signal goes high. The data and number of additions are written to the dual port ram (DPR), implemented in the FPGA. The twelve measurement values are written in sequentially, the FPGA sends an interrupt to the DSP when this has been completed. The DSP reads the measurement values stored at its specific addresses and calculates the average values.

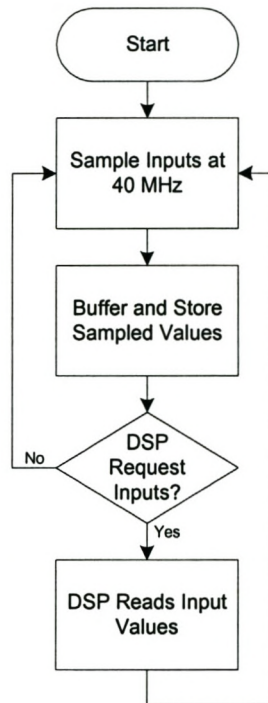
The 64 Word DPR implemented in the FPGA is a standard megafunction called `lpm_ram_dp`. The `lpm_ram_dp` function uses the embedded system blocks (ESBs) in the FPGA and not the logic elements, thus saving the resources on the FPGA. The DPR is configured for the specific needs of the system. The twelve measurements are stored at the first twelve addresses. The address decoding can be seen in Table 4.



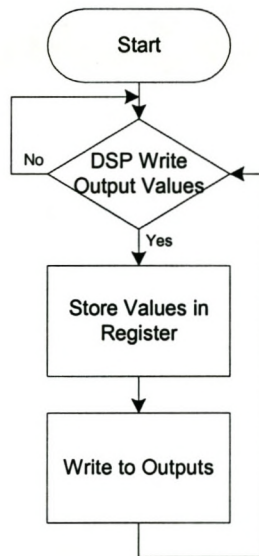
**Figure 5-6: DPR Implemented in the FPGA**

### 5.3 Input Output Board Data Flow

The input output board data flow is discussed in section 4.3.3.



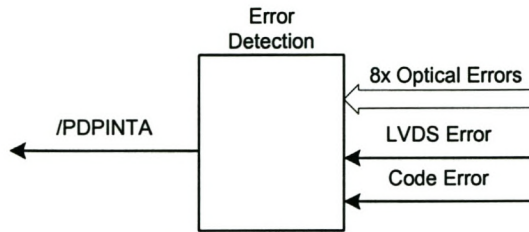
**Figure 5-7: Inputs Flow Chart**



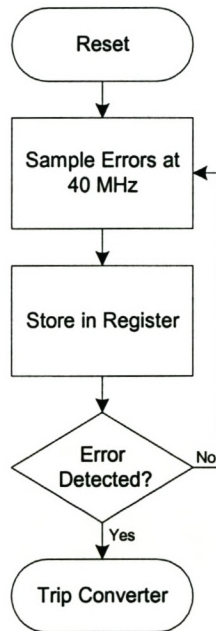
**Figure 5-8: Outputs Flow Chart**

### 5.4 Error Detection

The error detection is discussed in section 4.3.3.



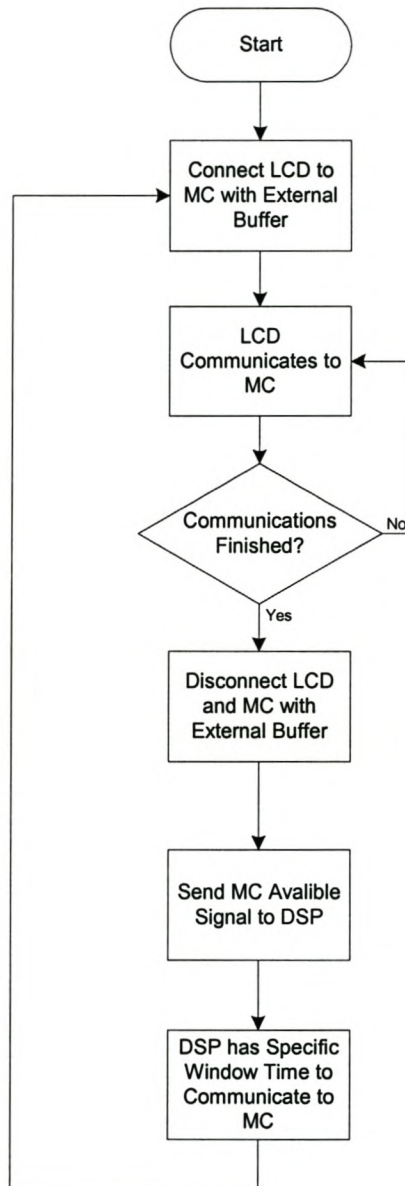
**Figure 5-9: Error Detection Block Diagram**



**Figure 5-10: Error Detection Flow Diagram**

### 5.5 The GESC and MC Connection

The GESC and MC connection is discussed in Chapter 4.3.4. Figure 4-10 shows the physical connections between the GESS, LCD Controller and the MC. The communication between the MC and LCD Controller is a priority. Communications on the bus have to be multiplexed between the LCD Controller and the GESC.



**Figure 5-11: Flow Chart of GESC and MC Connection**

## **5.6 Summary**

In this chapter all the firmware for the various PLDs was discussed. This firmware enabled the results and measurements obtained in chapter 6.

---



## 6. Evaluation of the Energy Storage Controller

### 6.1 Introduction

To evaluate the GESS, two topologies were implemented; the control of a dc to dc and an ac to dc converter. The implementation and control of the dc to dc topology is discussed in detail. A closed-loop control system is implemented in the GESC. This will fully test the ability of the GESMS to measure signals with high harmonic content associated with the dc to dc converter waveforms, and of the GESC to use these measurements to control a closed loop system.

A dc to ac topology was implemented in an open loop system for the control of an induction machine flywheel. The closed loop control is not implemented because of the complex control algorithms. This falls outside the scope of this thesis.

### 6.2 Practical Setup

A three-phase converter was used for both the dc to dc and dc to ac topologies. The converter is a pre-assembled module from Semikron. The phases are controlled by six optical PWM inputs. The error signals for each phase are combined to one optical error output.

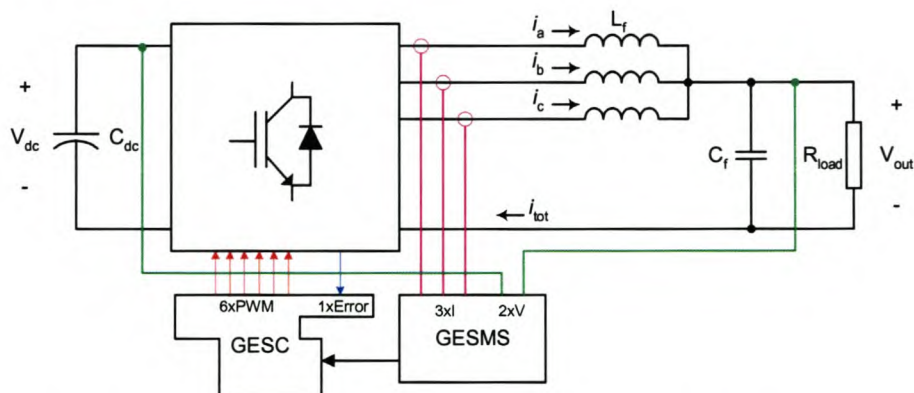


Figure 6-1: DC to DC Converter Experimental Setup

In Figure 6-1 the practical setup for the dc to dc topology can be seen. The inductance of the output filter inductors ( $L_f$ ) is 700  $\mu$ H per phase. The capacitance of the output filter capacitors ( $C_f$ ) is 300  $\mu$ F. A dc-bus voltage of 400 V was used.

### **6.3 Control Algorithm**

#### **6.3.1 Introduction**

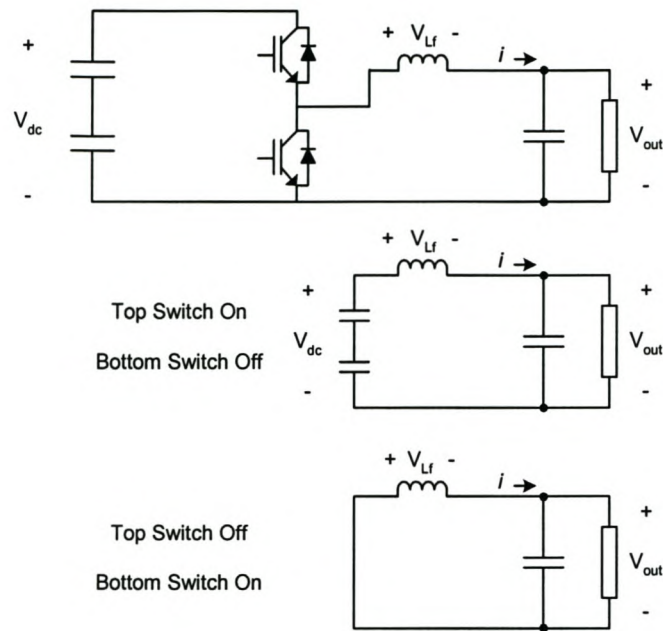
Predictive control has good dynamic performance and is relatively simple to implement [18]. Steady state error can occur due to the absence of an integrating term in the control equation.

Predictive algorithms that sample the dc bus and output voltages enable the controller to compensate for varying bus and output voltages, depending on the direction of the power flow. This is particularly handy in an energy storage application where the dc bus or output voltage must be kept constant as energy is extracted from the storage system.

#### **6.3.2 Derivation**

The control equation is derived for the dc to dc converter. As mentioned in section 2.2, all the phases are connected in parallel. The current in each half-bridge phase-arm is controlled individually and these are then combined to form the total load current. This is one of the main advantages of the GESMS. If the system is controlled by using only the output current, all the top and bottom switches will switch simultaneously, as the same duty cycles are given to each phase-arm. This can result in currents differing in each phase because of unbalances in the inductors and series resistance. The control equation is derived by looking at one of the phases. There are two switching states. In Figure 6-2 it can be seen that either the top or the bottom switch can be on at any specific time.

---



**Figure 6-2: Switching States for Dc to Dc Converter**

$V_{out}$  is assumed to stay constant over the switching period. When the top switch is on and the bottom off, the current through the inductor can be calculated as follows.

$$t_{TOP} : \Delta i = \frac{V_{dc} - V_{out}}{L} \quad (6-1)$$

For the bottom switch on and top switch off, the current equation is

$$t_{BOTTOM} : \Delta i = \frac{-V_{out}}{L} \quad (6-2)$$

where  $\Delta i$  denotes the change in current during each time interval. The total change in inductor current in a switching period can be calculated by superposition of equations (6-1) and (6-2).

$$\Delta i = \left( \frac{V_{dc} - V_{out}}{L} \right) t_{TOP} + \left( \frac{-V_{out}}{L} \right) t_{BOTTOM} \quad (6-3)$$

From equation (2-4):

$$T = t_{TOP} + t_{BOTTOM} \quad (6-4)$$

$$\begin{aligned} d &= \frac{V_{out}}{V_{dc}} \\ &= \frac{t_{TOP}}{T} \end{aligned} \quad (6-5)$$

From equations (6-3), (6-4) and (6-5) the control algorithm for predictive current control can be calculated as:

$$d = \frac{L \frac{\Delta i}{T} + V_{out}}{V_{dc}} \quad (6-6)$$

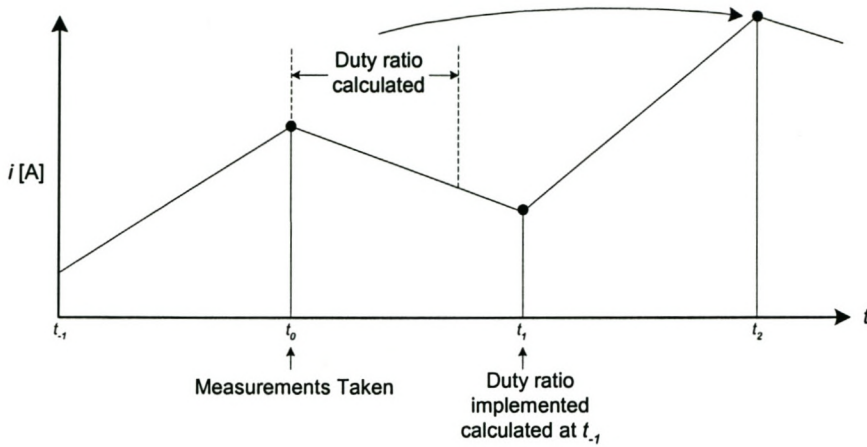
$$0 < d < 1$$

### 6.3.3 Double Prediction

Physical limitations in the control and measurement system results in a switching period delay in the duty ratio given to the switches. The reason is that the controller cannot take the measurements and calculate the new duty ratios in zero time. The PWM signals in the DSP are double buffered. This means that the PWM signals calculated now will only be implemented at the start of the next switching period.

To compensate for this, double prediction is implemented. Double prediction uses the previous duty ratio and predicts what the change in current is going to be. This value is added to the measured current values. The controller now predicts what the currents are going to be at the end at the period and uses these values to calculate the new duty ratio.

---



**Figure 6-3: Double Prediction**

Equation (6-6) is rearranged to make the  $\Delta i$  the subject. It is assumed the output and dc voltages change slowly enough over a switching period not to have a significant effect.

$$\Delta i_{pred} = \frac{\left( d_{prev} - \frac{V_{out}}{V_{dc}} \right) V_{dc}}{\frac{L}{T}} \quad (6-7)$$

$$d = \frac{L \left( \Delta i_{meas} + \Delta i_{pred} \right) + V_{out}}{V_{dc} T} \quad (6-8)$$

## 6.4 Simulations

### 6.4.1 Introduction

The Simplorer simulation package was used to simulate the experimental setup and to see how the control system would react. This package allows the detailed simulation of power electronic devices and control schemes.

Ideal switches and components were used for this simulation. The filter components in the simulation have the same values as the experimental setup to correlate between simulation and practical results.

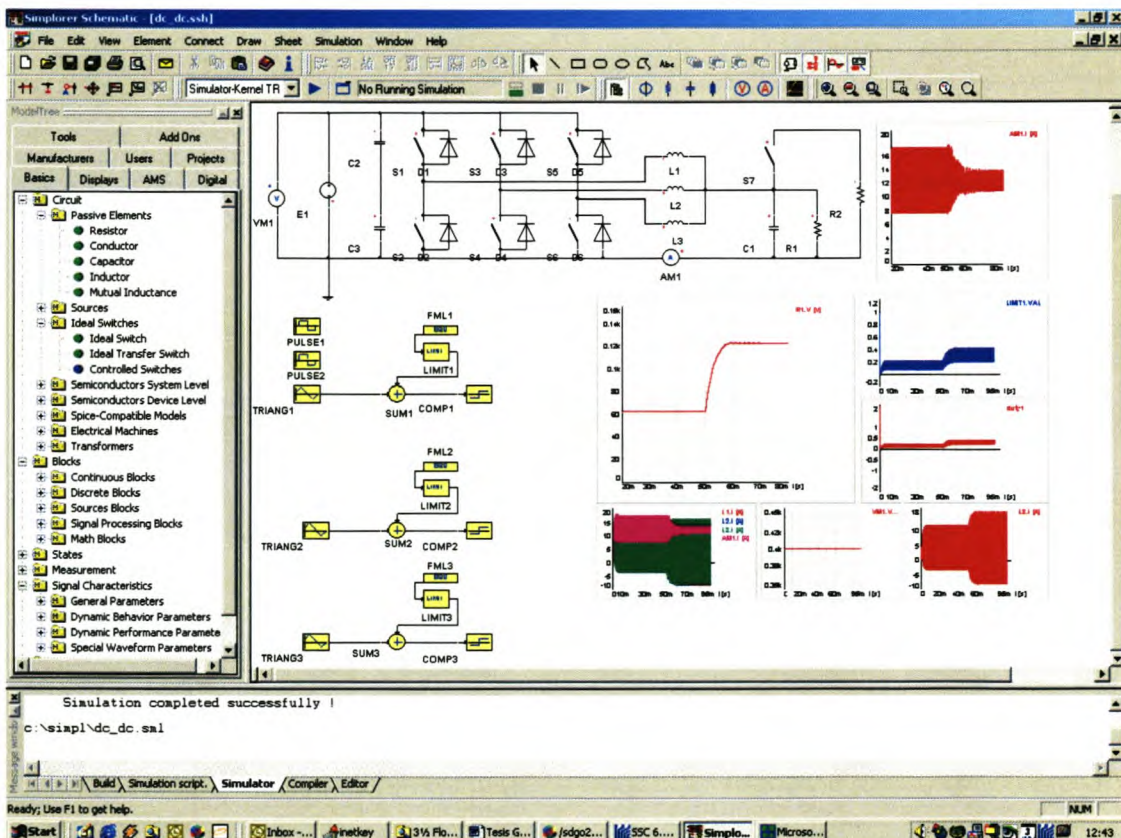
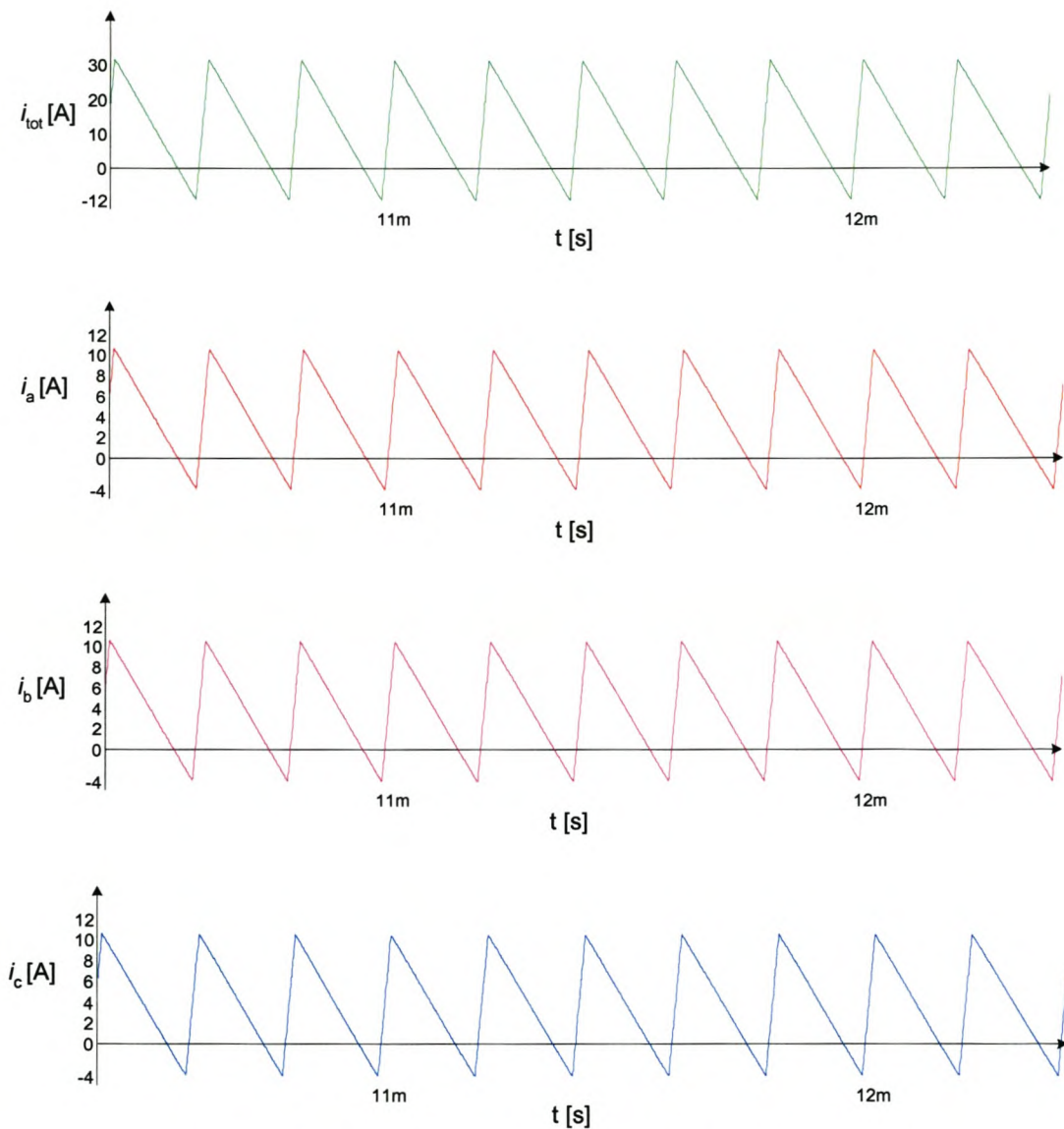


Figure 6-4: Simplorer Simulation Package

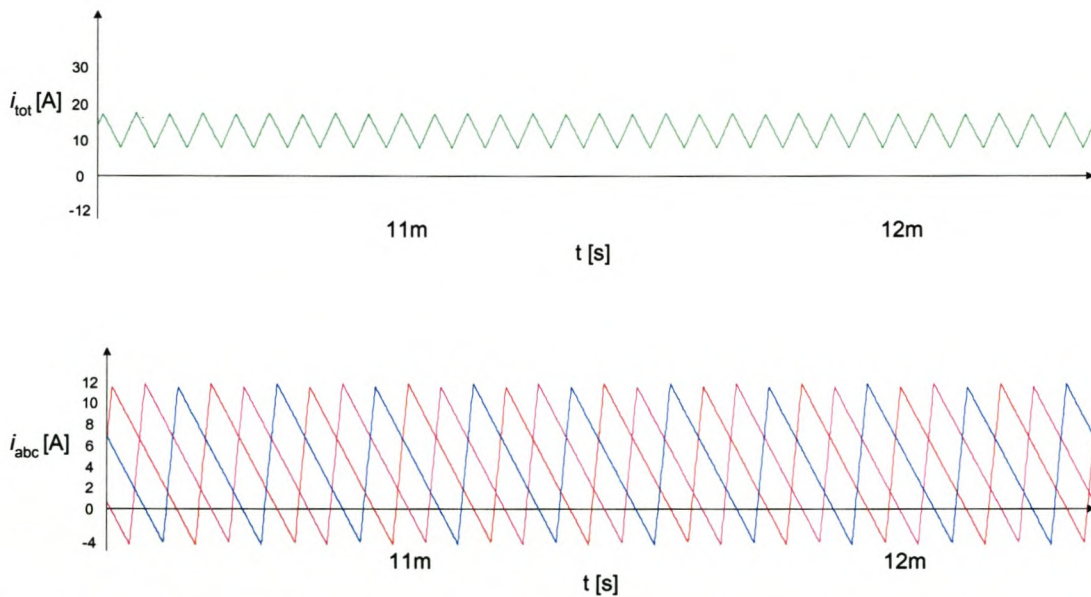
### 6.4.2 Simulation Results

In Figure 6-5 the results of the closed loop control can be seen. A reference current of 5 A is implemented in all three phases. The result is a combined output current of 15 A. The high current ripple associated with dc to dc converters can be seen in this simulation. Note how the ripple in each phase adds up in the total output current.



**Figure 6-5: Phase Currents and Total Current**

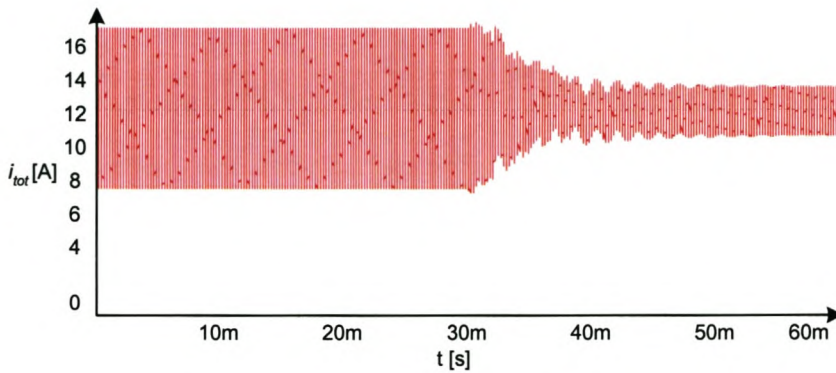
In Figure 6-6 the same measurements are taken as in Figure 6-5 but interleaved switching was used. Note how the ripple in the output current is significantly reduced. In the practical setup a three phase converter is used and the reference waveforms are thus shifted by 120 degrees. In a four phase system they would be shifted by 90 degrees, reducing the ripple even further. The frequency is also three times higher than before.



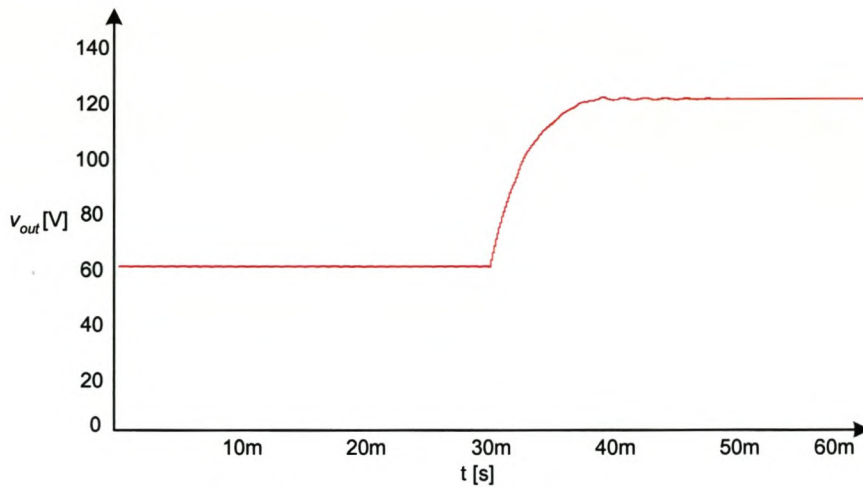
**Figure 6-6: Phase Currents and Total Current with Interleaved Switching**



Figure 6-7 and Figure 6-8 show the total output current and voltage respectively for a load step simulation. At 30 ms the load was doubled. It can be seen how the current control loop keeps the average current the same at 12 A while the output voltage doubles from 60 V to 120 V.



**Figure 6-7: Total Output Current with Load Step**



**Figure 6-8: Output Voltage with Load Step**

The exponential increase in output voltage is due to the charging of the capacitors in the output filter.

---

## 6.5 Practical Results

### 6.5.1 Control Software

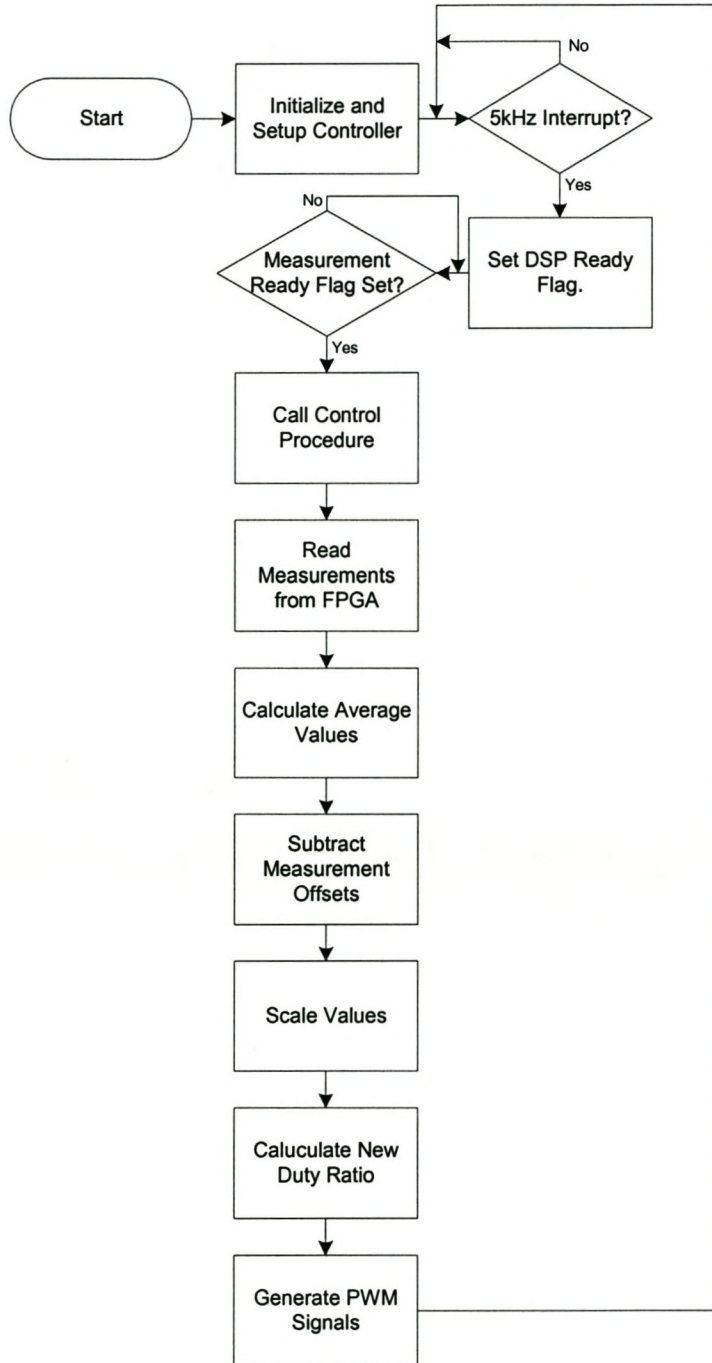


Figure 6-9: Flow Chart of Control Software

### 6.5.2 DC to DC Conversion Results

In all the current measurements where 'A' is not stated 1 mV represents 1 A.

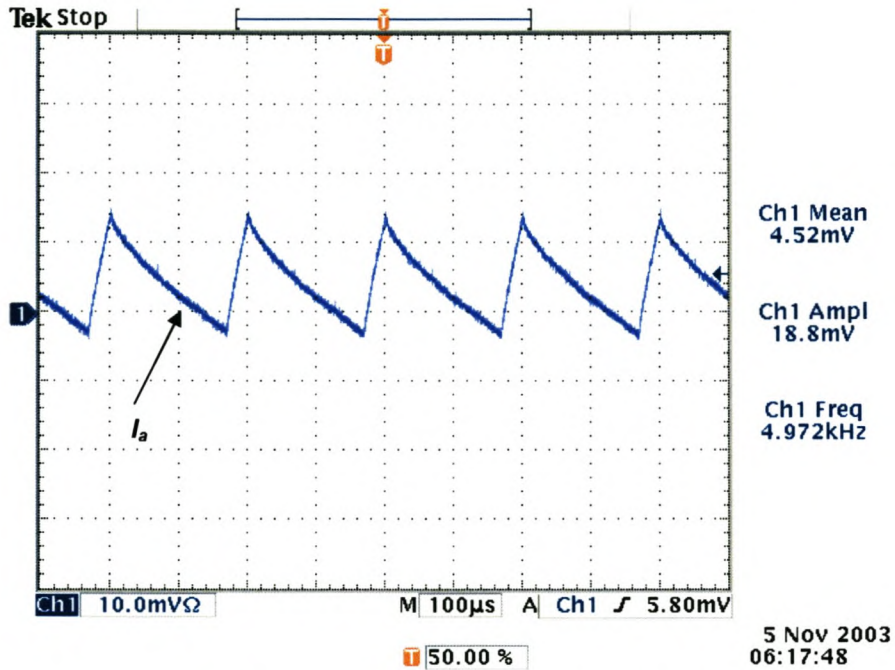


Figure 6-10: Phase Current with 4.5 A Reference

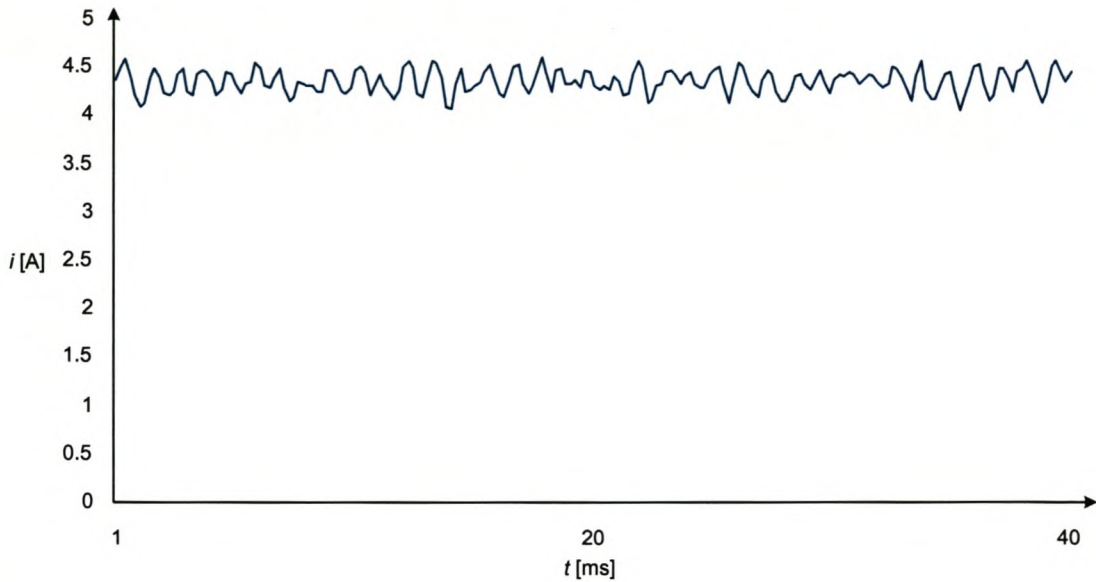
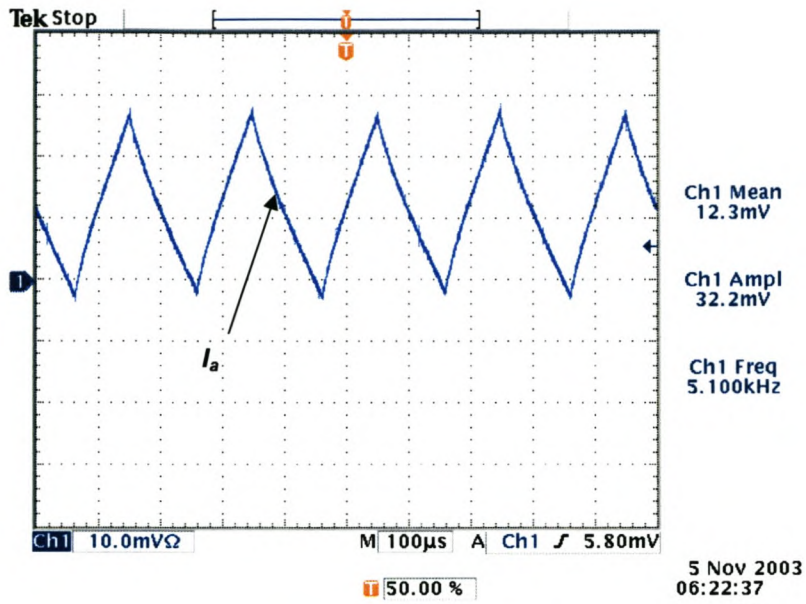
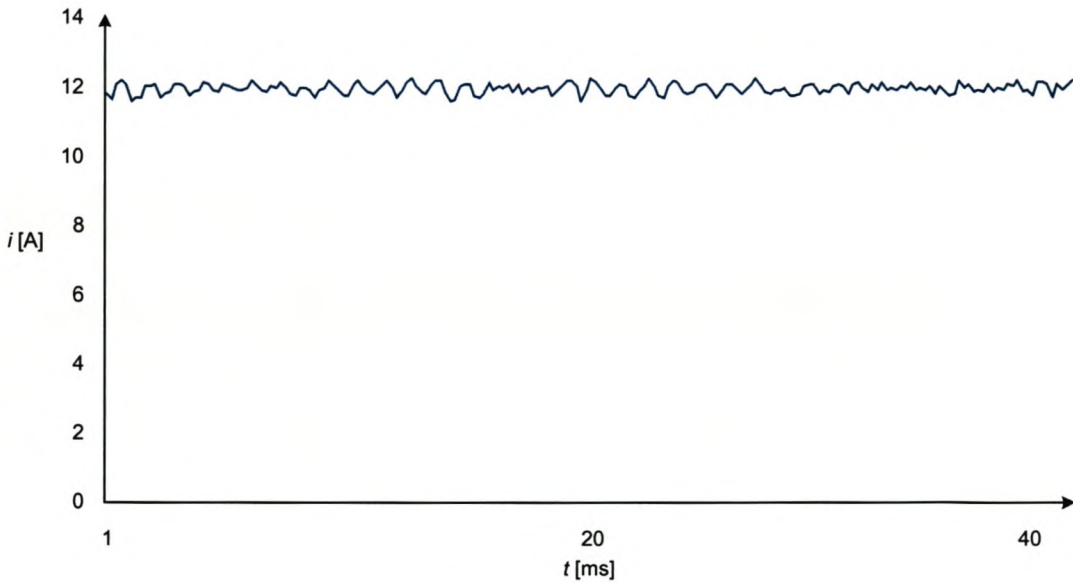


Figure 6-11: Average Measured Phase Currents in DSP with 4.5 A Reference

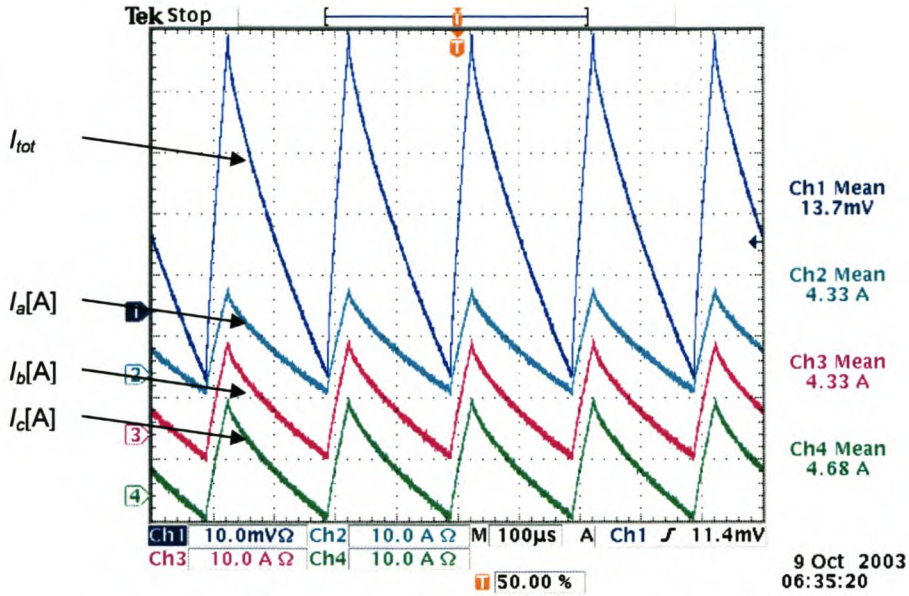


**Figure 6-12: Phase Current with 12 A Reference**

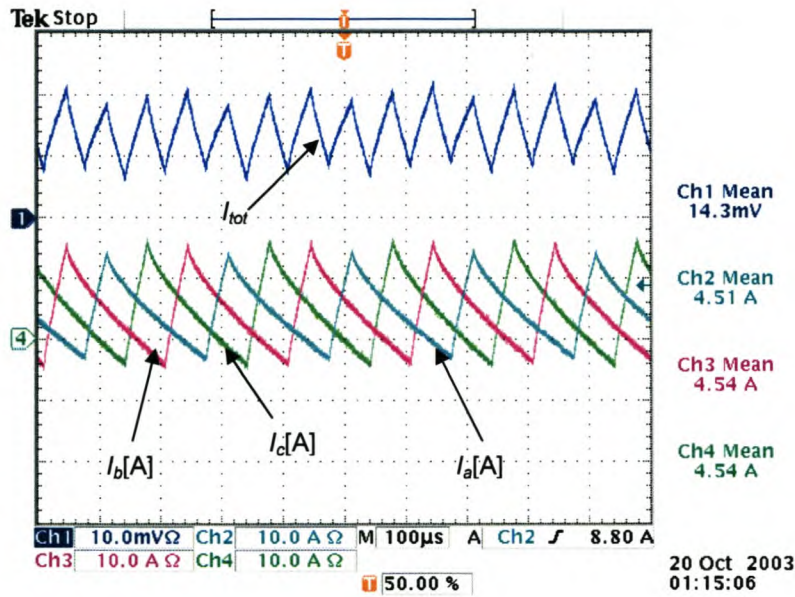


**Figure 6-13: Average Measured Phase Current in DSP with 12 A Reference**

Figure 6-10 to Figure 6-13 show the measuring system's ability to produce accurate measurements under extreme conditions. It also shows how the control loop is able to regulate the current.



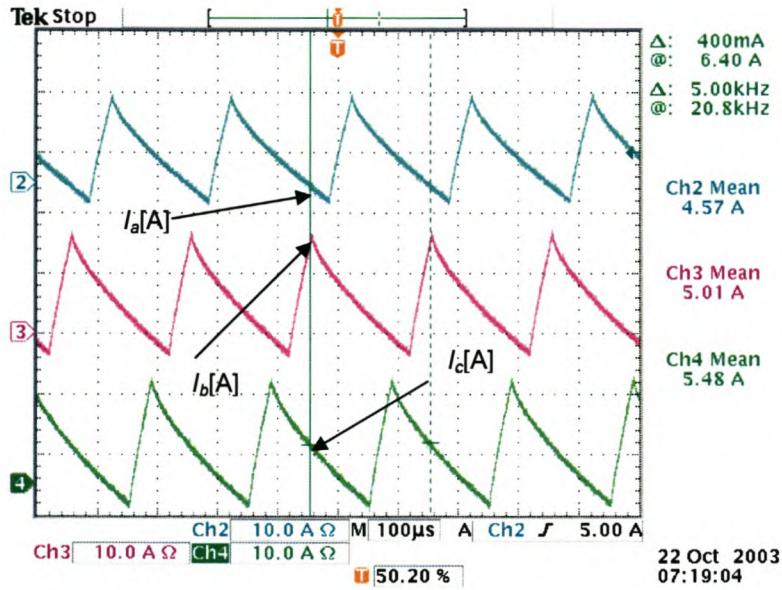
**Figure 6-14: Close-Loop Phase and Output Currents**



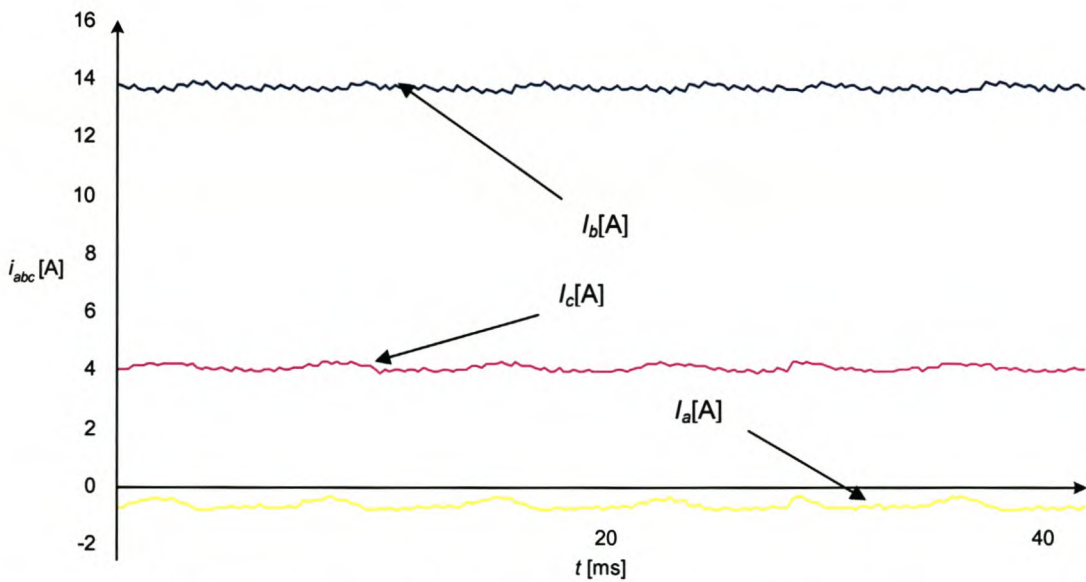
**Figure 6-15: Close-Loop Phase and Output Currents with Interleaved Switching**

Figure 6-14 and Figure 6-15 show the significant reduction in output current ripple with interleaved switching, as opposed to normal switching. This enables

the converter to use smaller inductors, reducing size and cost. These measurements correlate with the simulated results in Figure 6-5 and Figure 6-6



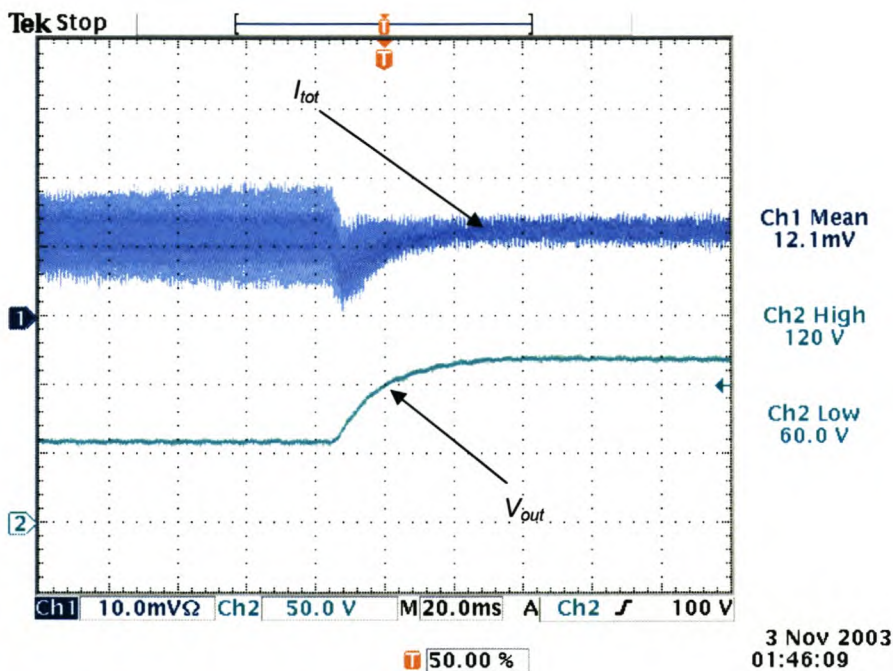
**Figure 6-16: Open Loop Single Sample Phase Currents**



**Figure 6-17: Single Sample Phase Currents in DSP**

In Figure 6-16 the open-loop interleaved phase currents can be seen. The firmware of the FPGA was changed so that the DSP receives only a single

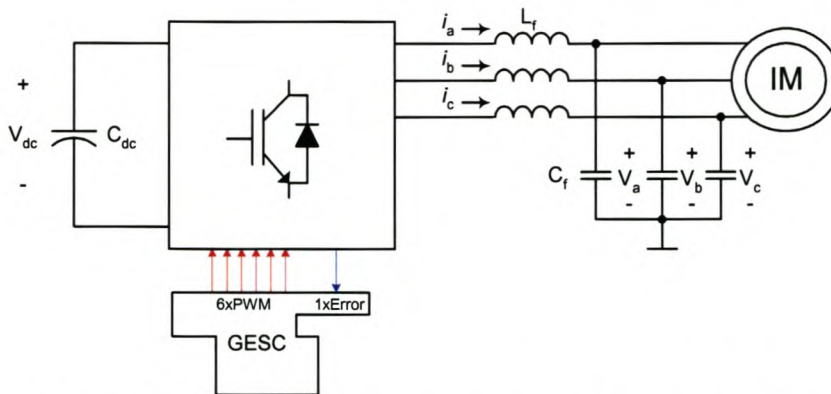
sample. All the phase currents are sampled simultaneously once every 5 kHz as on the measuring system on the MC. In Figure 6-17 the values received by the DSP can be seen. The aliasing mentioned in section 2.3.3.1 is clearly visible. There is a huge improvement from the GESMS measurements seen in Figure 6-10 to Figure 6-13 compared the single sample measurements in Figure 6-16 and Figure 6-17. It was impossible to implement a stable closed loop system with the single sample measurements and control scheme.



**Figure 6-18: Output Current and Voltage with Load Step**

Figure 6-18 shows a load step. It can be seen how the control loop kept the current constant at 12 A while the output voltage doubled. This measurement correlates with the simulated result in Figure 6-7.

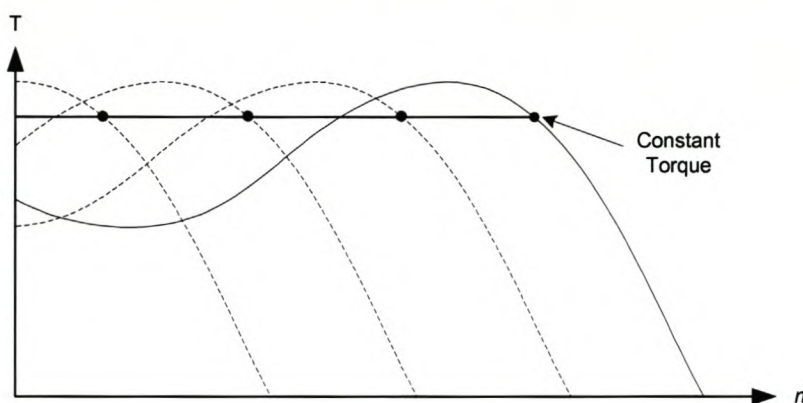
### 6.5.3 DC to AC Results



**Figure 6-19: DC to AC Converter Experimental Setup**

In Figure 6-19 the practical setup for the dc to ac topology can be seen. The inductance of the output filter inductors ( $L_f$ ) is 700  $\mu\text{H}$  per phase. The capacitance of the output filter capacitors ( $C_f$ ) is 100  $\mu\text{F}$ . A dc bus voltage of 400 V is used.

Open loop volt-hertz [4] control was used to control the induction machine. The ratio between the frequency and amplitude is always kept the same. At startup the frequency and amplitude are simultaneously swept up to the rated values. This enables the maximum torque to be available at all times [7]. Figure 6-20 shows different torque curves at various frequencies.



**Figure 6-20: Torque-Speed Curve with Volt-Hertz Control**



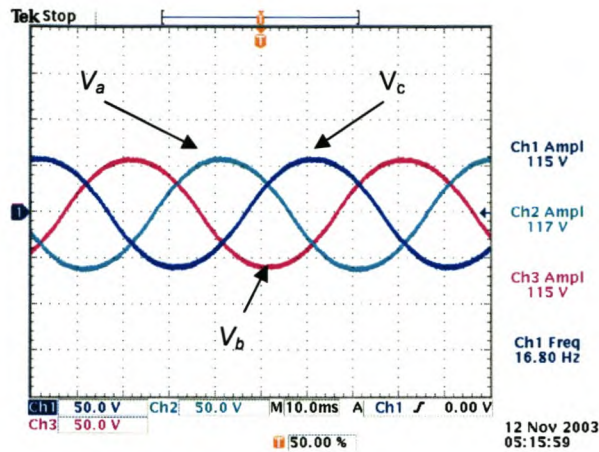


Figure 6-21: Phase Output Voltages with Modulation Index 33%

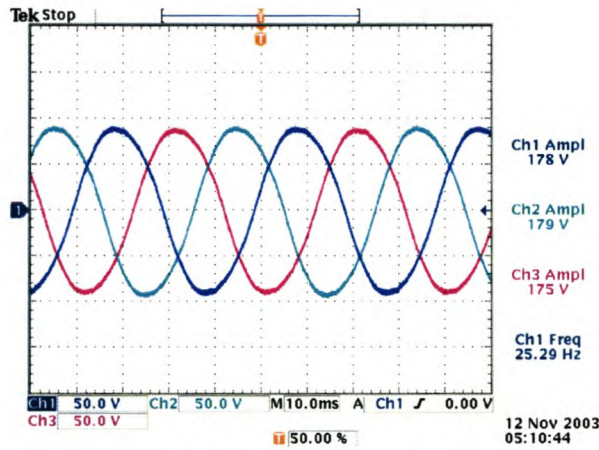


Figure 6-22: Phase Output Voltages with Modulation Index 50%

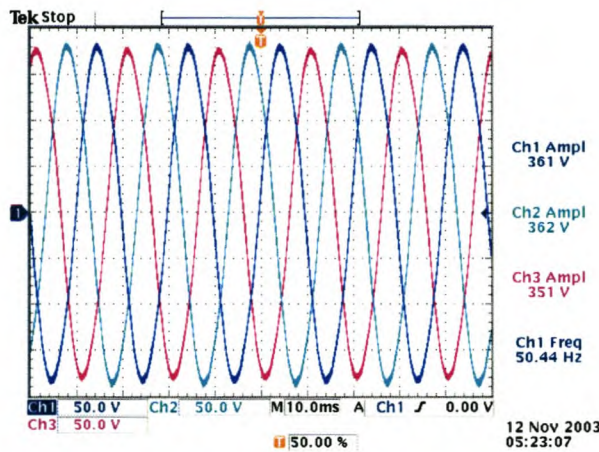


Figure 6-23: Phase Output Voltages with Modulation Index 100%

In Figure 6-21 to Figure 6-23 the phase voltages for a dc to ac converter can be seen at various modulation indexes.

## **6.6 Summary**

In this chapter a dc to dc converter and dc to ac converter were successfully implemented. The dc to dc converter control was the main focus of this chapter and was discussed in depth. These two topologies tested the ability of the GESS to control an energy storage system.

---

## **7. Conclusions**

### **7.1 Overview**

This thesis covers the design of a new DSP based controller and measuring system, able to control lead acid batteries, flow batteries and flywheels and to incorporate them into an existing UPS system. A background study has been done on these systems and the basic operation of a power electronic converter.

New energy storage systems are more complex and need more control features. They can no longer simply be connected to the dc side of an UPS converter. Each system needs to be controlled according to its specific application, thus needing its own power electronic converter.

The addition of a second converter to the power compensator required the development of a new control and measuring system. The existing MC was not capable of handling two converters and complex energy storage. It does not have sufficient I/O ports and processing power to handle the additional overhead. The new GESM is specifically designed to control the energy storage.

To control and utilize the full potential of a complex energy storage system, a flexible measuring system was needed. The new GESMS is a very powerful measuring system and is able to sample currents and voltages up to 1.5 MSPS. These measurements can also be passed on to the MC. The GESM is a modular design, allowing any part of the design to be upgraded without influencing the other.

The new GESM will enable the UPS system to utilize a wide range of energy storage types. The UPS system is no longer limited to lead acid batteries alone, but can use flow batteries, lead acid batteries or flywheels. The previous shortcomings involving lead acid batteries are also eliminated. The new control

---

system also allows for a combination of energy storage systems to be integrated with a single conditioner.

## **7.2 Practical Evaluation**

The main objective was to test the GESS and evaluate whether it can fill the shortcomings in the existing control system and also have the ability and resources to control an energy storage system.

The GESS was tested by implementing a closed-loop dc to dc converter for battery applications and an open loop dc to ac converter for flywheel applications. The GESS allowed the use of interleaved switching and the ability to control each phase separately. The four phases of the converter can now be split into two full bridges. This will enable the control of dual energy storages in the same system. The oversampling ability of the GESMS allowed it to cope with the high harmonic content in the measured signals.

## **7.3 Future Work and Recommendations**

With new technology moving faster than the hardware engineer can design, newer, faster and better components are always available. The main upgrades considered for the GESS are:

- The APEX FPGA currently used is an expensive device not optimally suited for this application. Altera has since introduced the new Cyclone FPGA which is a low cost high volume device with more logic elements than the APEX device. The Cyclone was not yet available when design started for the GESS.
  - A new F2813 DSP from Texas Instruments, targeted for drives and power electronics, is now available. It is three times more powerful than the F2407A with many more new features.
-

- All the measurement processing is done in the FPGA on the GESC. An option is to shift that processing to the GESMS. This will result in significantly less data transfer between the GESMS and the GESC because only the processed values will need to be transmitted and not all the sampled measurements.
- If the change mentioned above is made, the communications between the GESMS and the GESC will be far fewer. This will eliminate the need for the high speed LVDS link. A fiber optic link can be used, which will considerably reduce the risk of interface and data corruption.

The final proposal is very bold. The redesign of the total controller system. The design of a new controller will include the functions of the MC and the GESC. This will be the ultimate solution. Components for such a system have already been identified. The new C6713 floating point DSP from Texas Instruments is capable of 1000 MFLOPS. This will eliminate the need for multiple processing devices. The EP1C12 FPGA from Altera has 12000 logic elements. This will open possibilities for upgrading the firmware and extending the functionality of the system.

---

## References

- [1] Clive D.S. Tuck, *Modern Battery Technology*, Ellis Horwood Limited, 1991.
  - [2] T.R. Crompton, *Battery Reference Book*, Butterworth-Heinemann Ltd, 1995.
  - [3] Muhammad H. Rashid, *Power Electronics Handbook*, Academic Press, 2001.
  - [4] N. Mohan, T.M. Underland, W.P. Robbins, *Power Electronics*, John Wiley & Sons Inc, 1995
  - [5] D. Christiansen, *Electronic Engineers Handbook*, McGraw-Hill, 1996.
  - [6] A. Ter-Gazarian, *Energy Storage for Power Systems*, Peter Peregrinus Ltd, 1994
  - [7] P.C. Sen, *Principles of Electric Machines and Power Electronics*, John Wiley & Sons Inc, 1997
  - [8] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, 1997
  - [9] J.W. Nilsson, S.A. Riedel, *Electric Circuits*, Addison-Wesley Publishing Company Inc, 1996
  - [10] J.J. Cruywagen, J.B.B. Heys, H.G. Raubenheimer, P.C. van Berge, *Inleiding tot die Anorganiese en Fisiese Chemie*, Butterworth Uitgewers (Edms) Bpk, 1993
  - [11] M. Skyllas-Kazacos, C. Menictas, *The Vanadium Redox Battery for Emergency Back-Up Applications*, IEEE, 1997
  - [12] S. Miyake, N. Tokuda, *Vanadium Redox Flow Battery for Variety of Applications*, IEEE, 2001
  - [13] N. Tokuda, Y. Miki H. Arai, K. Yamamoto, K. Emmura, K. Motoi, T. Shinzato, T. Konno, *Vanadium Redox Flow Battery System for Use in Office Buildings*, Electric Energy Storage Applications and Technologies. 2000
-

- 
- [14] T. Kaizuka, T. Sasaki, *Evaluation of Control Maintaining Electric Power Quality by Use of Rechargeable Battery System*, IEEE, 2000
- [15] A. Price, *Technologies for Energy Storage – Present and Future: Flow Batteries*, IEEE, 2000
- [16] T. Shigematsu, T. Kumamoto, H. Deguchi, T. Hara, *Applications of Vanadium Redox Flow Battery to Maintain Power Quality*, IEEE, 2002
- [17] A. Shibata, K. Sato, *Development of Vanadium Redox Flow Battery for Electricity Storage*, Power Engineering Journal, June 1999
- [18] D.D Bester, Control of Series Compensator for Power Quality Conditioner, March 1999
- [19] TMS320LF2407 Evaluation Module, Technical Reference, Spectrum Digital
- [20] TMS320LF2407A Datasheet, Texas Instruments, SPRS145G, July 2000, Revised February 2002
- [21] TMS320LF/LC240xA DSP Controllers Reference Guide, Texas Instruments, SPRU 357B, December 2001
- [22] A.L. Lilein, Digital Signal Processors vs. Universal Microprocessors, Texas Instruments, SPRA344, September 1996
- [23] APEX 20K Programmable Logic Device Family, Altera Corporation, February 2002
- [24] Configuration Devices for SRAM-Based LUT Devices Datasheet, Altera Corporation, February 2002
- [25] MAX3000A Programmable Logic Device Family, Altera Corporation, October 2001
- [26] In-System Programmability in MAX Devices, Altera Corporation, June 2000
- [27] SN65LV1021/1212 10MHz to 40MHz 10:1 LVDS Serializer/Deserializer, Texas Instruments, SLLS526E, February 2002, Revised September 2002
-

- 
- [28] Low-Voltage Differential Signaling (LVDS) Design Notes, Texas Instruments, SLLA014A, 2000
- [29] Reducing Electro Magnetic Interference (EMI) with Low Voltage Differential Signaling (LVDS), Texas Instruments, SLLA030C, September 2000, Revised June 2002
- [30] THS1206 12-BIT 4 Analog 6 MSPS Simultaneous Sampling Analog-to-Digital Converters, Texas Instruments, SLAS217H, May 1999, Revised July 2003
- [31] R. Mancini, Opamps for Everyone Design Reference, SLOD006B, August 2002
- [32] TMS320C31 Datasheet, Texas Instruments, SPRS035B, March 1996, Revised January 1999
- [33] TMS320C50 Datasheet, Texas Instruments, SPRS030A, April 1995, Revised April 1996
- [34] INA2331 Datasheet, Texas Instruments, SBOS215A, December 2000
- [35] AD8544 Datasheet, Analog Devices, Rev B , 2000
- [36] Texas Instruments, <http://www.ti.com>
- [37] Altera Corporation, <http://www.altera.com>
- [38] Flywheel Energy Storage,  
<http://www.upei.ca/~physics/p261/projects/flywheel1/flywheel1.htm>
- [39] Flywheels and Energy Storage,  
<http://zebu.uoregon.edu/1996/ph162/l10a.html>
- [40] Hybrid Electric Vehicle Program, <http://www.ott.doe.gov/hev/flywheels.html>
- [41] Electrolytic Capacitors, [http://www.faradnet.com/deeley/book\\_toc.htm](http://www.faradnet.com/deeley/book_toc.htm)
-



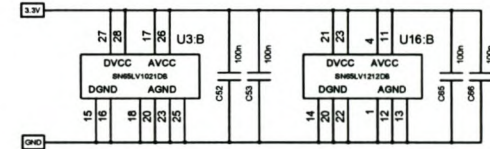
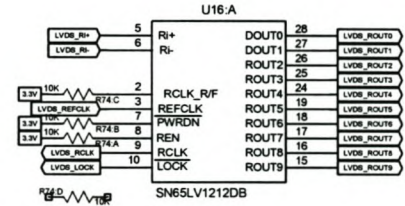
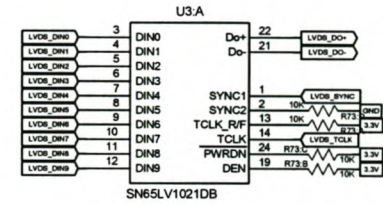
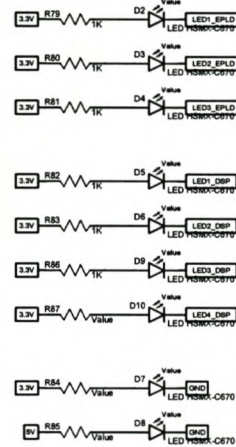
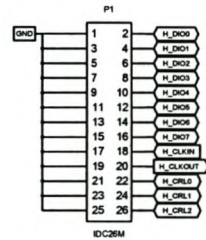
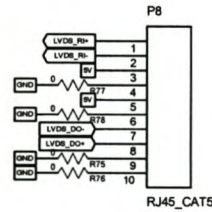
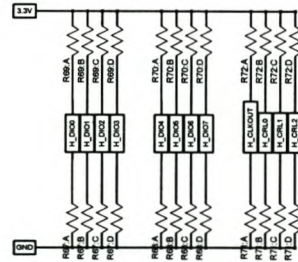
A

***Appendix A*    *GESC Schematics***

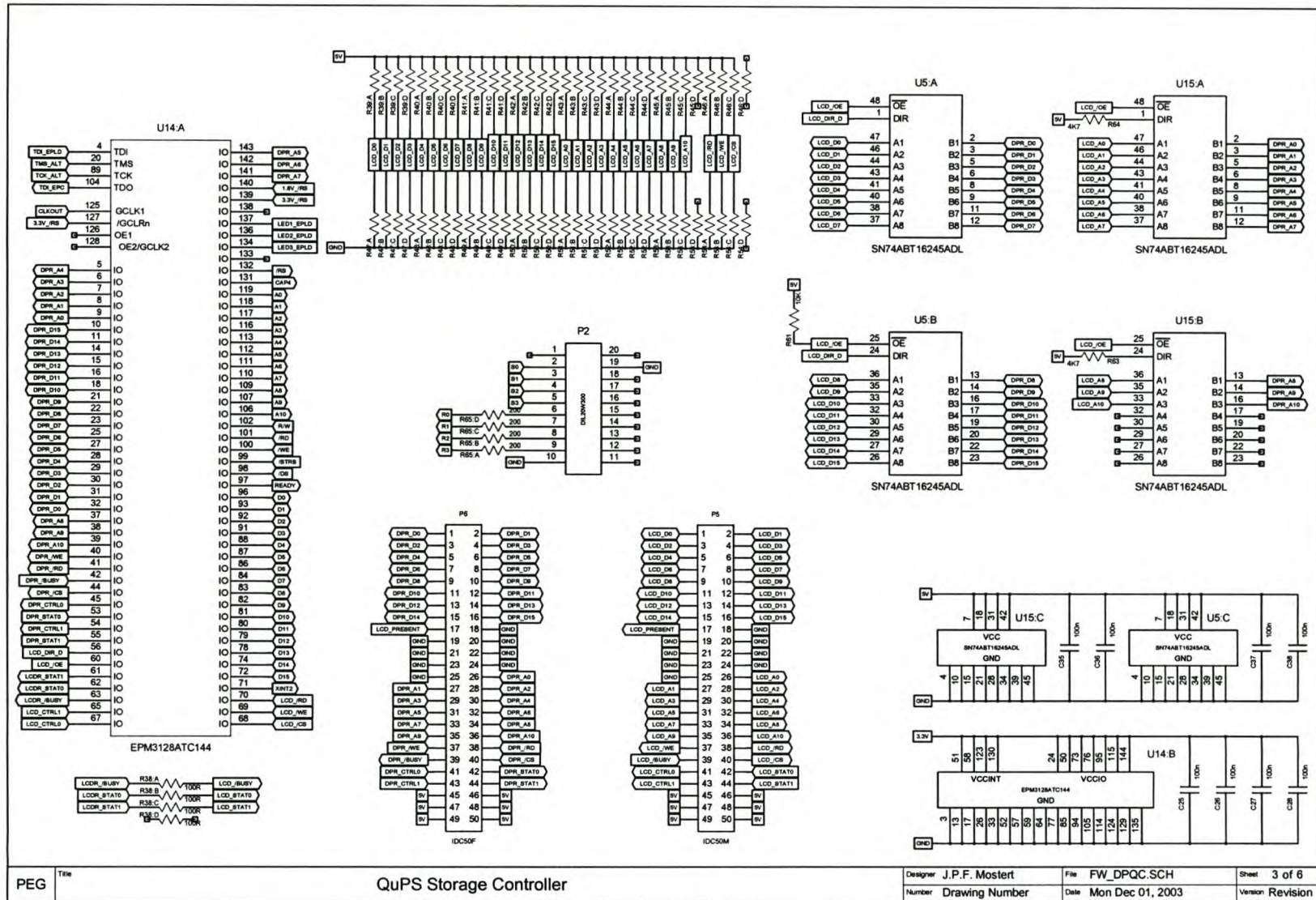
---



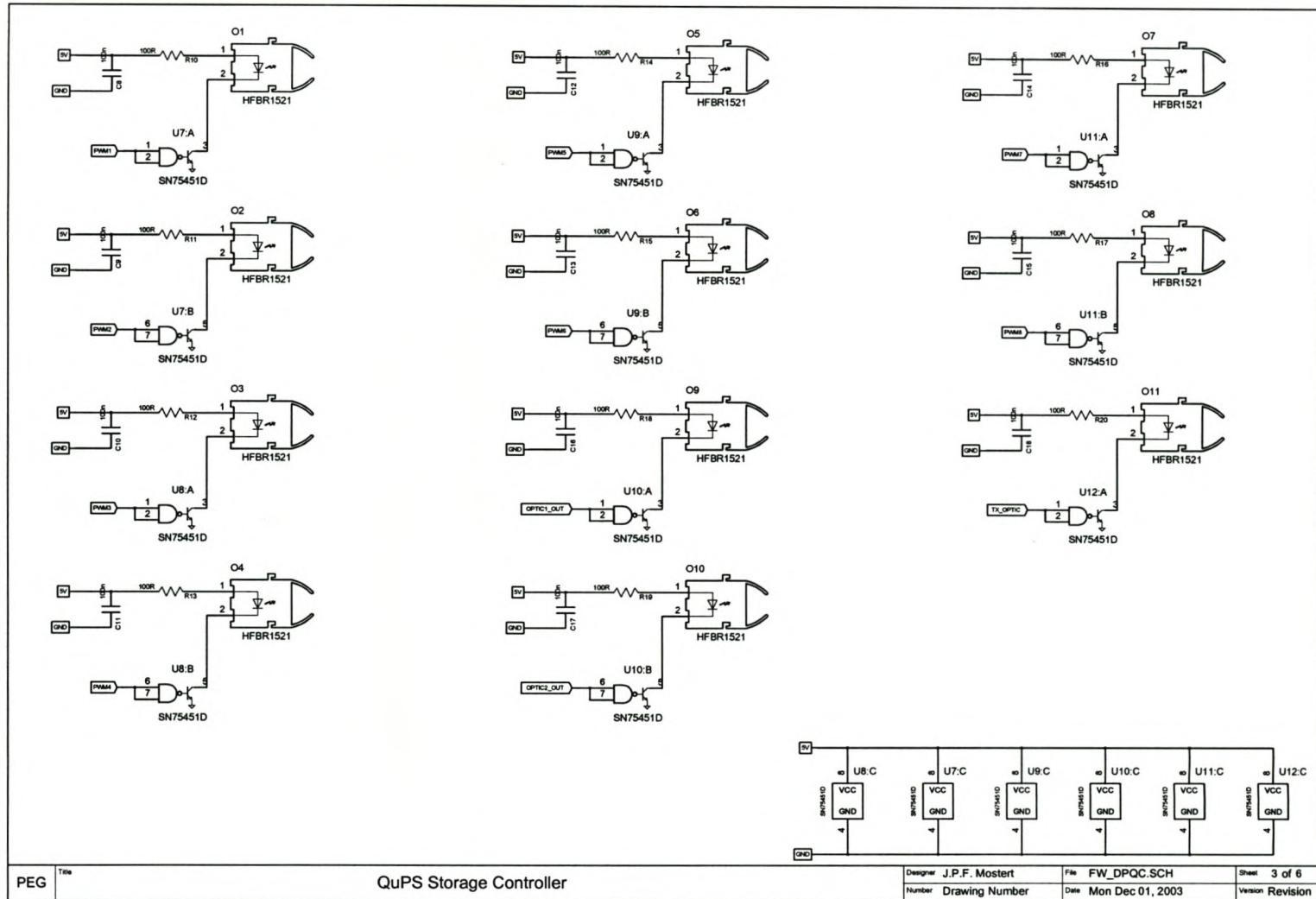


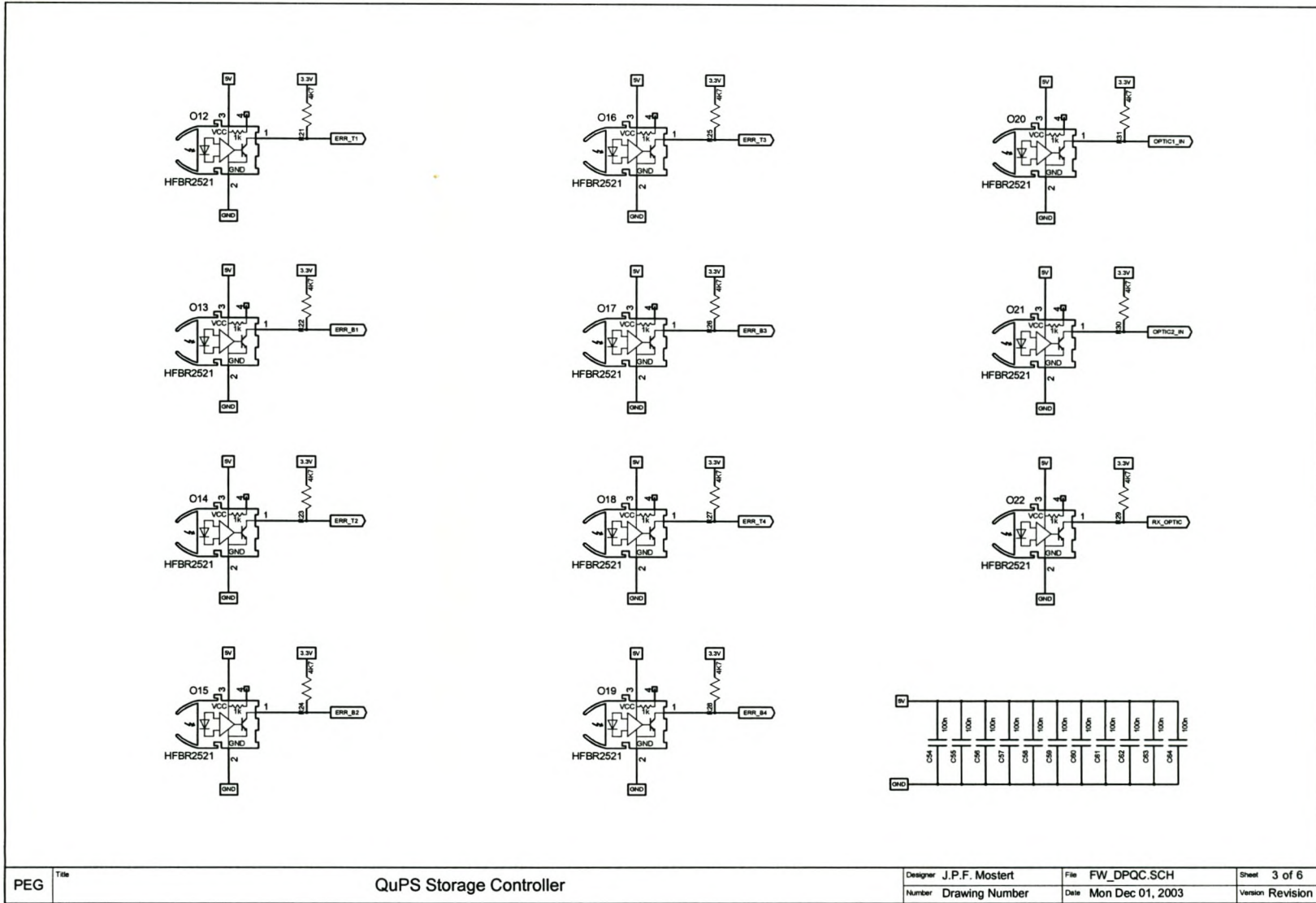


PEG	Title	QuPS Storage Controller	Designer	J.P.F. Mostert	File	FW_DPQC_SCH	Sheet	3 of 6
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision



PEG	Title	QuPS Storage Controller	Designer	J.P.F. Mostert	File	FW_DPQC.SCH	Sheet	3 of 6
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision





PEG	Title	QuPS Storage Controller	Designer	J.P.F. Mostert	File	FW_DPQC.SCH	Sheet	3 of 6
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision

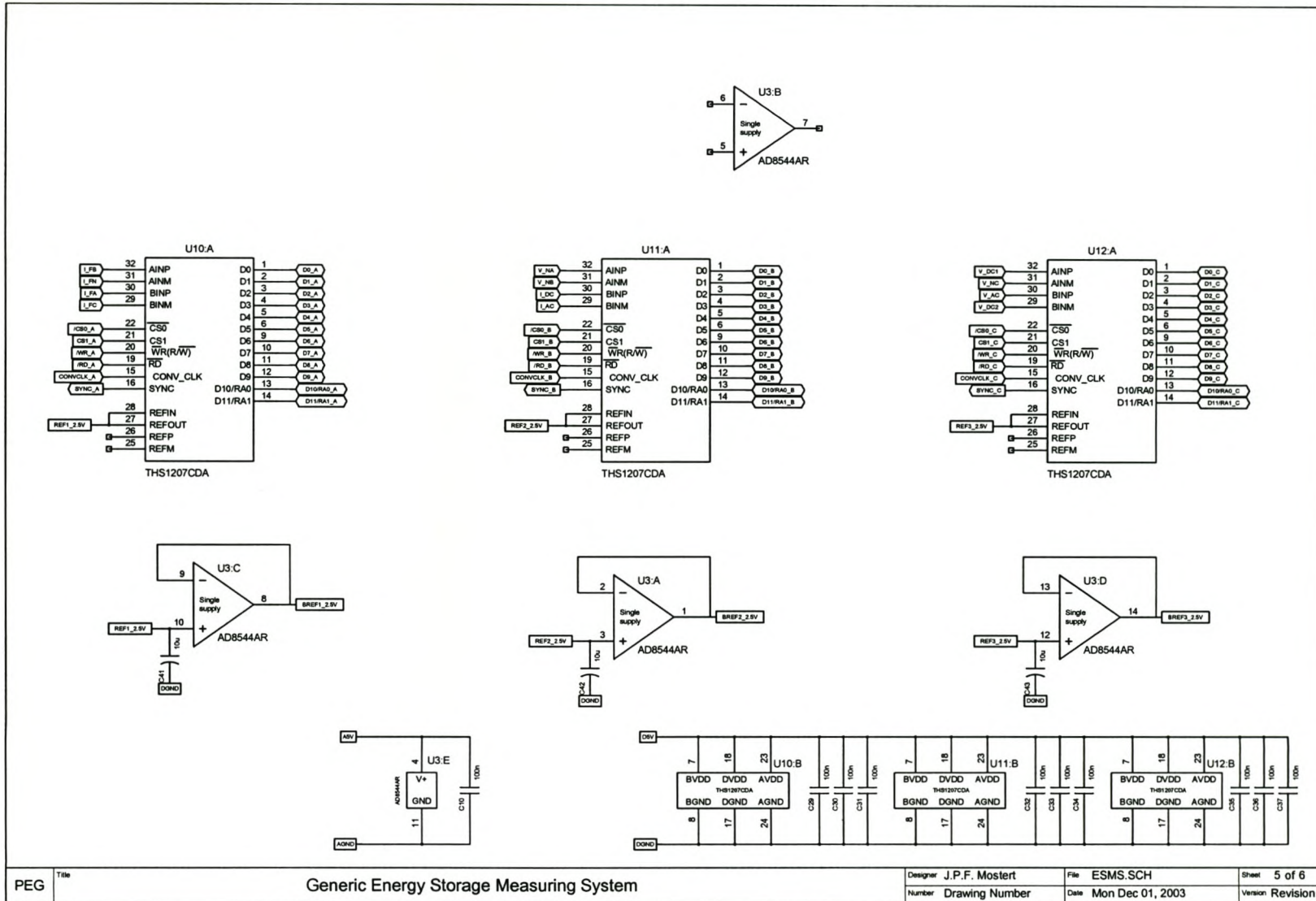
B

***Appendix B    GESMS Schematics***

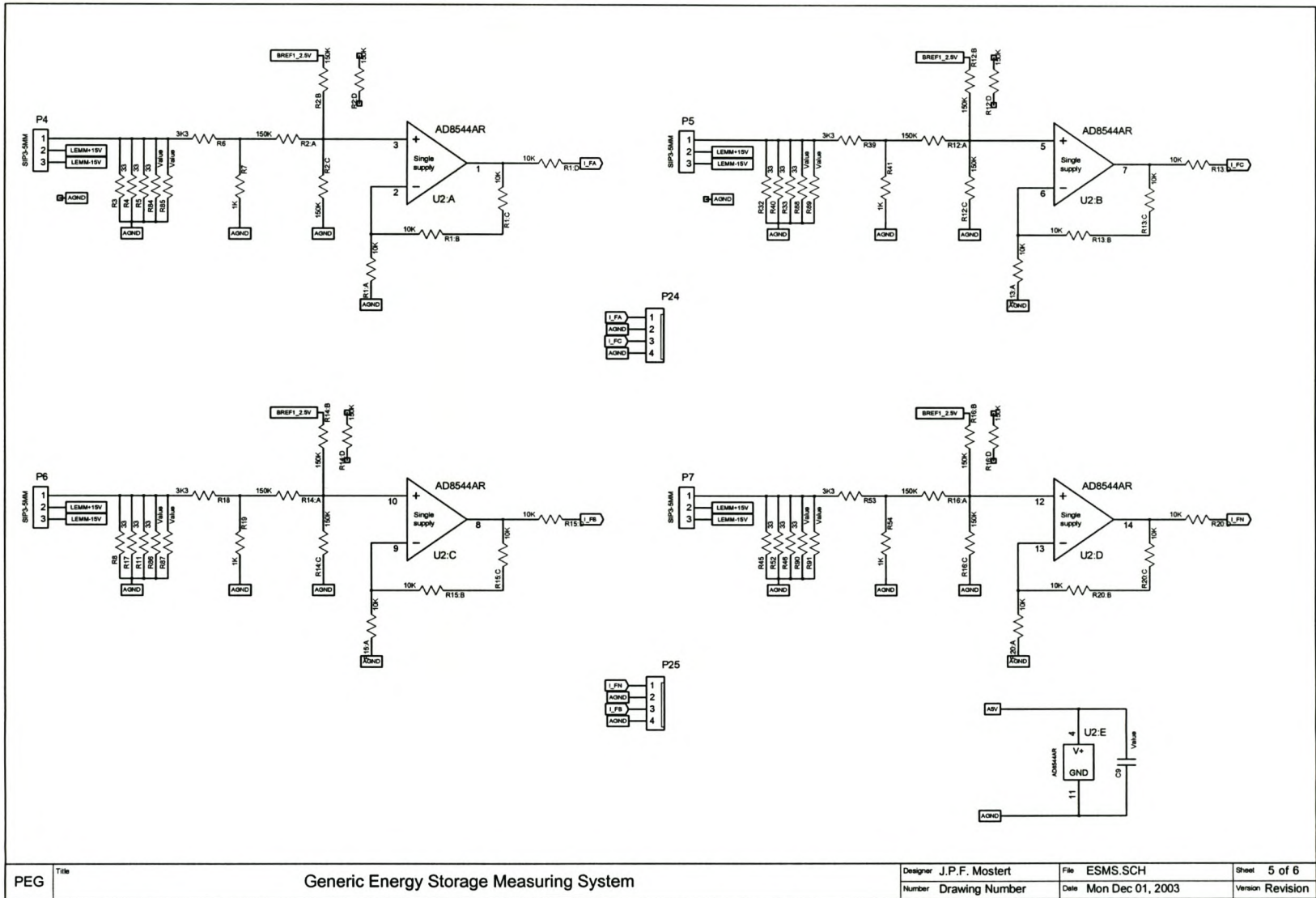
---



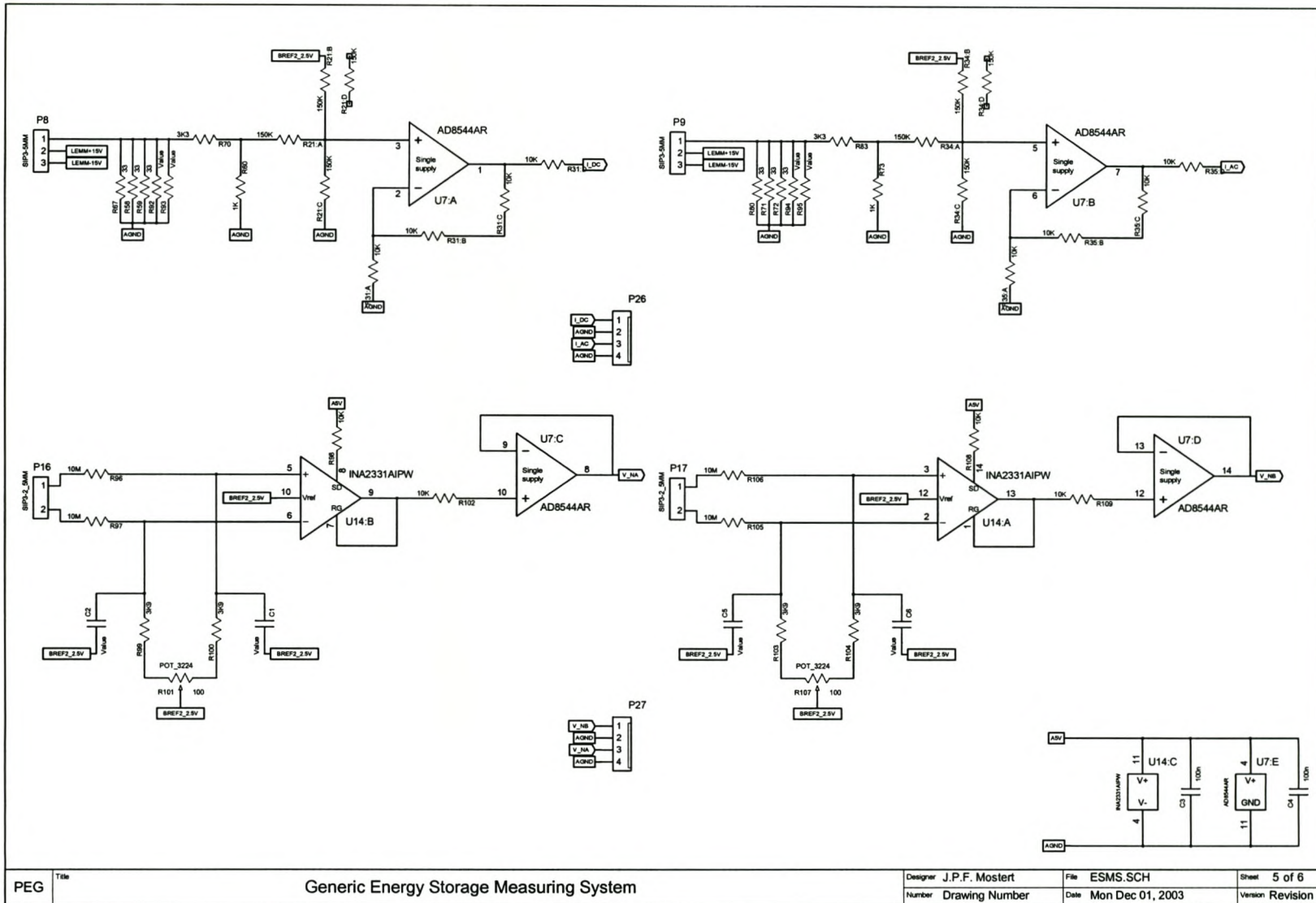




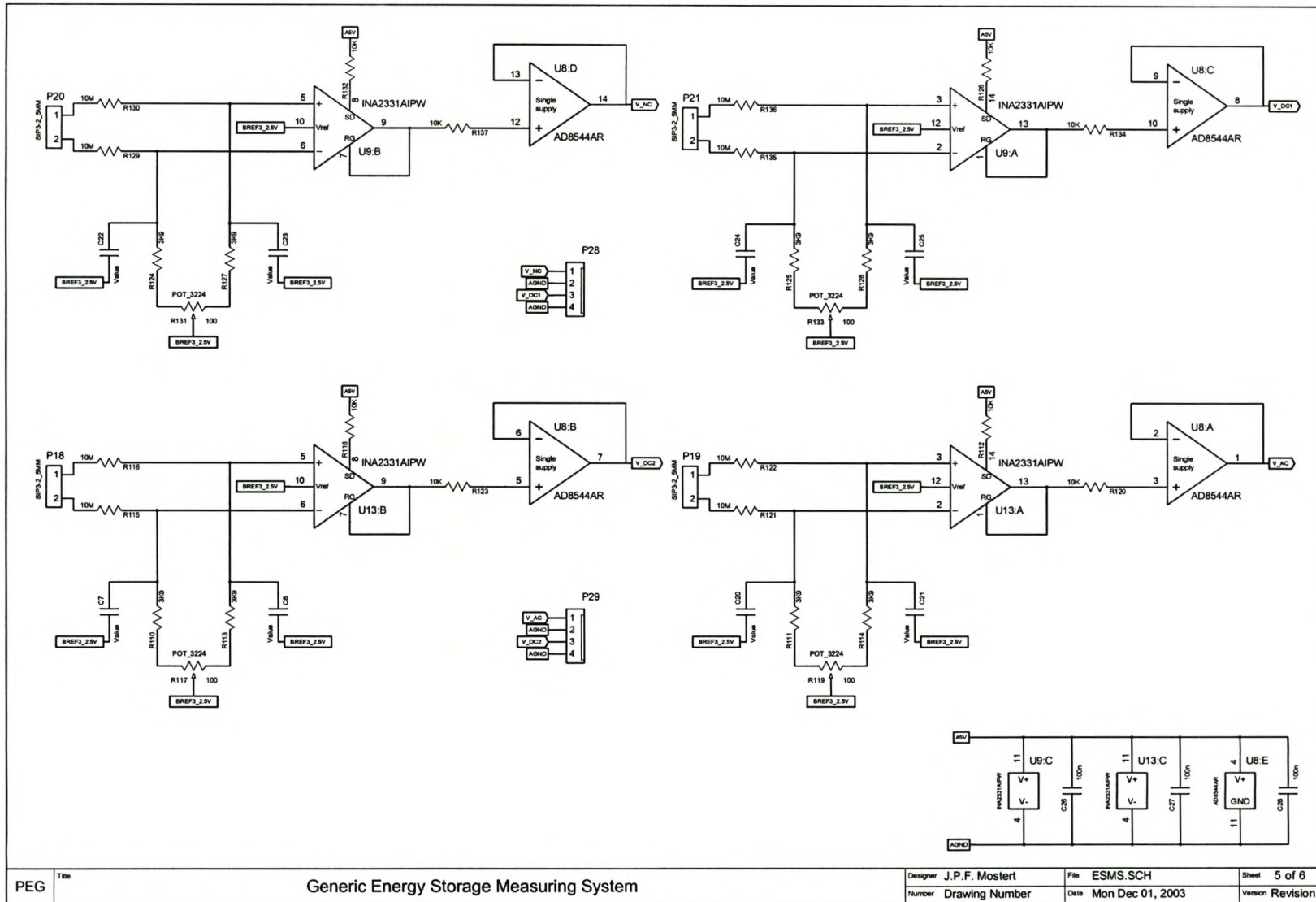
PEG	Title	Generic Energy Storage Measuring System	Designer	J.P.F. Mostert	File	ESMS.SCH	Sheet	5 of 6
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision



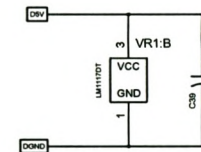
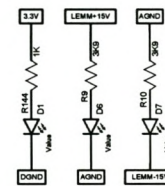
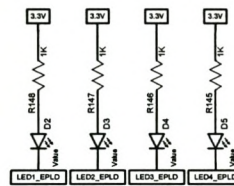
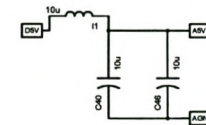
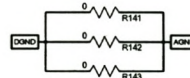
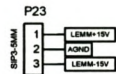
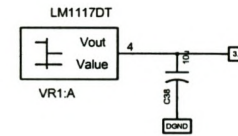
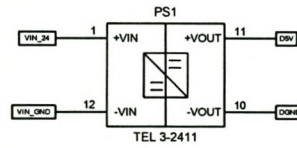
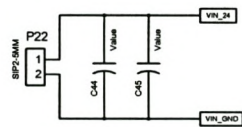
PEG	Title	Generic Energy Storage Measuring System	Designer	J.P.F. Mostert	File	ESMS.SCH	Sheet	5 of 6
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision



PEG	Title	Generic Energy Storage Measuring System	Designer	J.P.F. Mostert	File	ESMS.SCH	Sheet	5 of 6
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision



PEG	Tit	Generic Energy Storage Measuring System	Designer J.P.F. Mostert	File ESMS.SCH	Sheet 5 of 6
			Number Drawing Number	Date Mon Dec 01, 2003	Version Revision

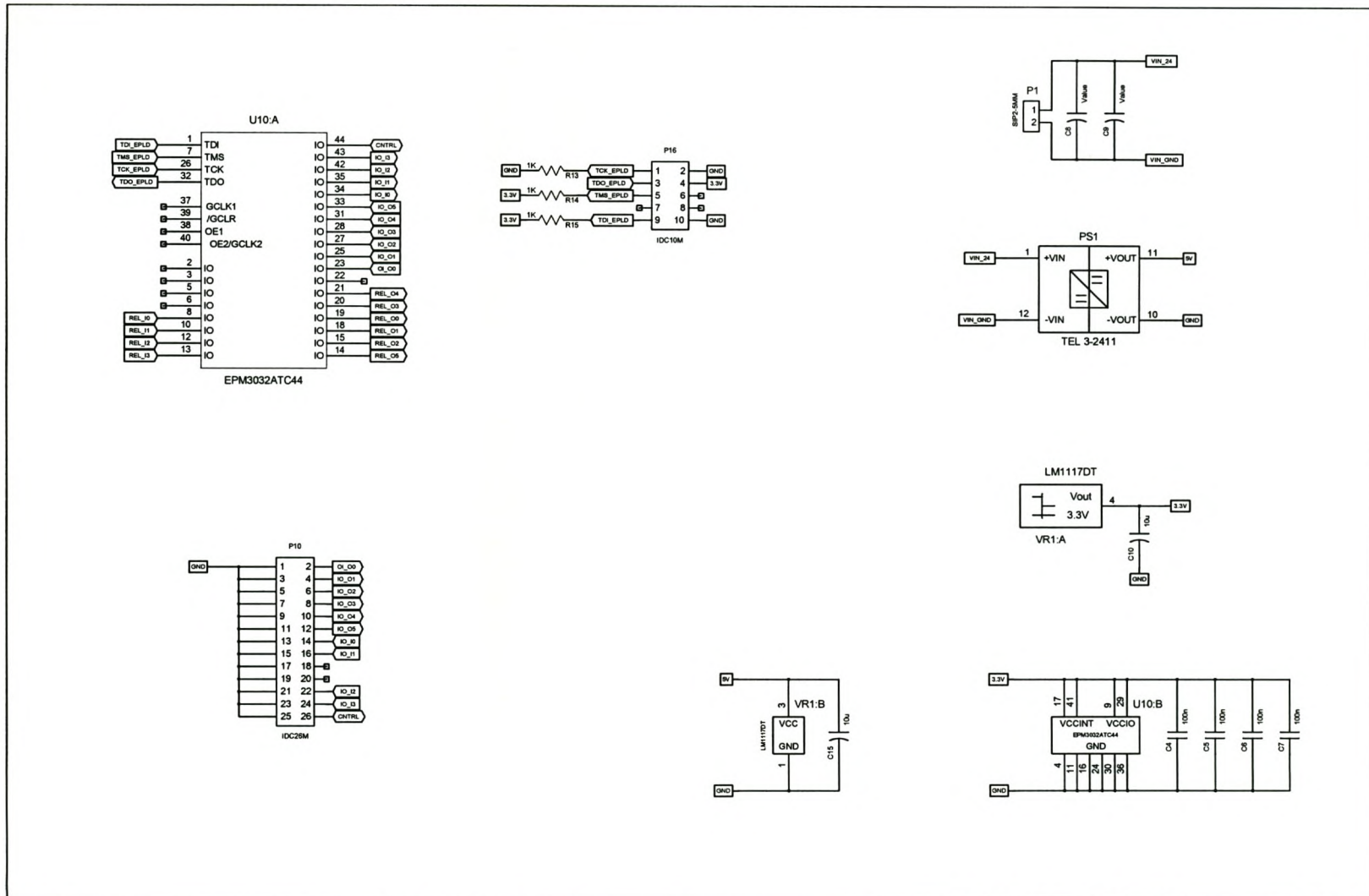


PEG	Title	Generic Energy Storage Measuring System	Designer	J.P.F. Mostert	File	ESMS.SCH	Sheet	5 of 6
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision

C

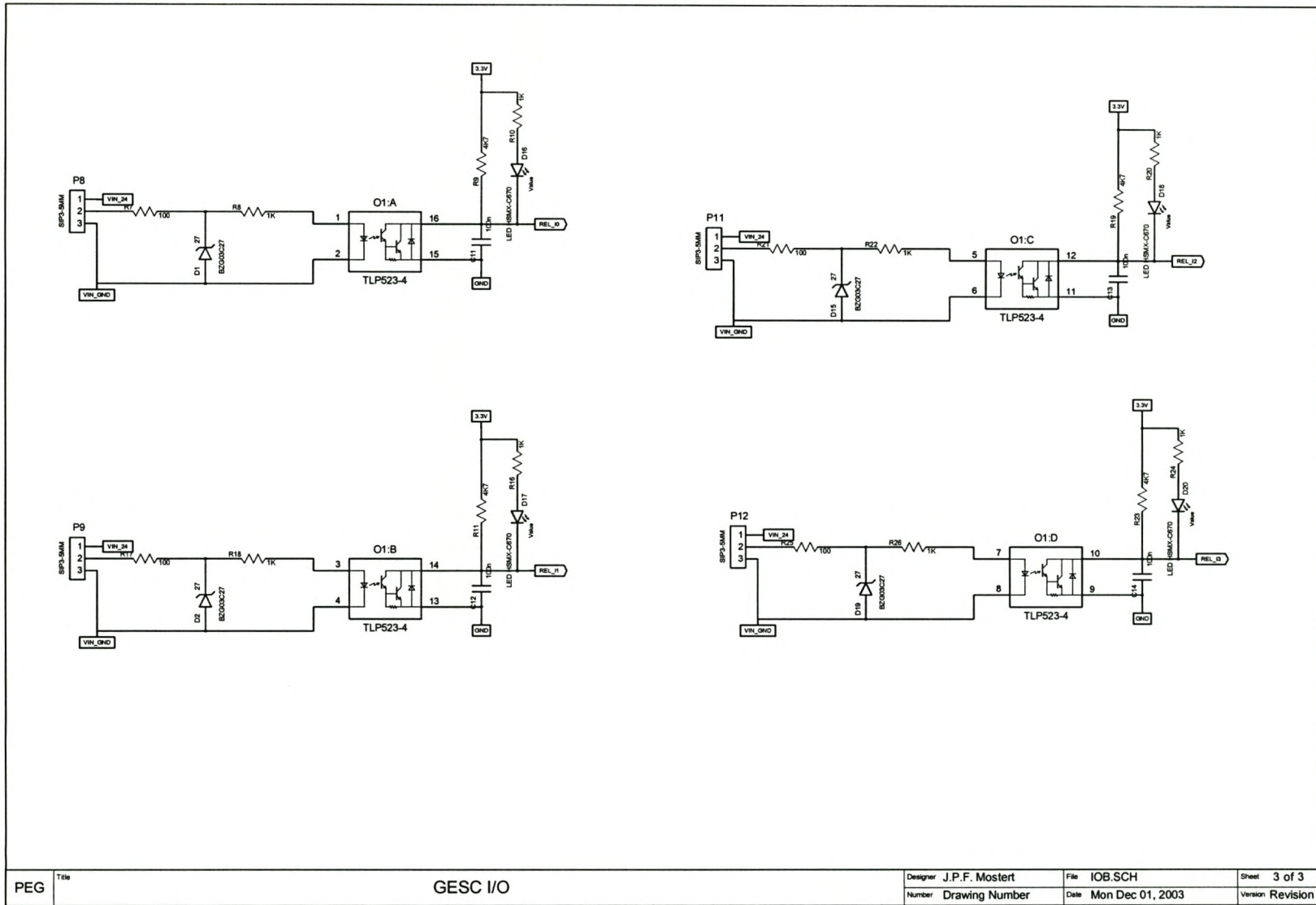
***Appendix C    IOB Schematics***

---

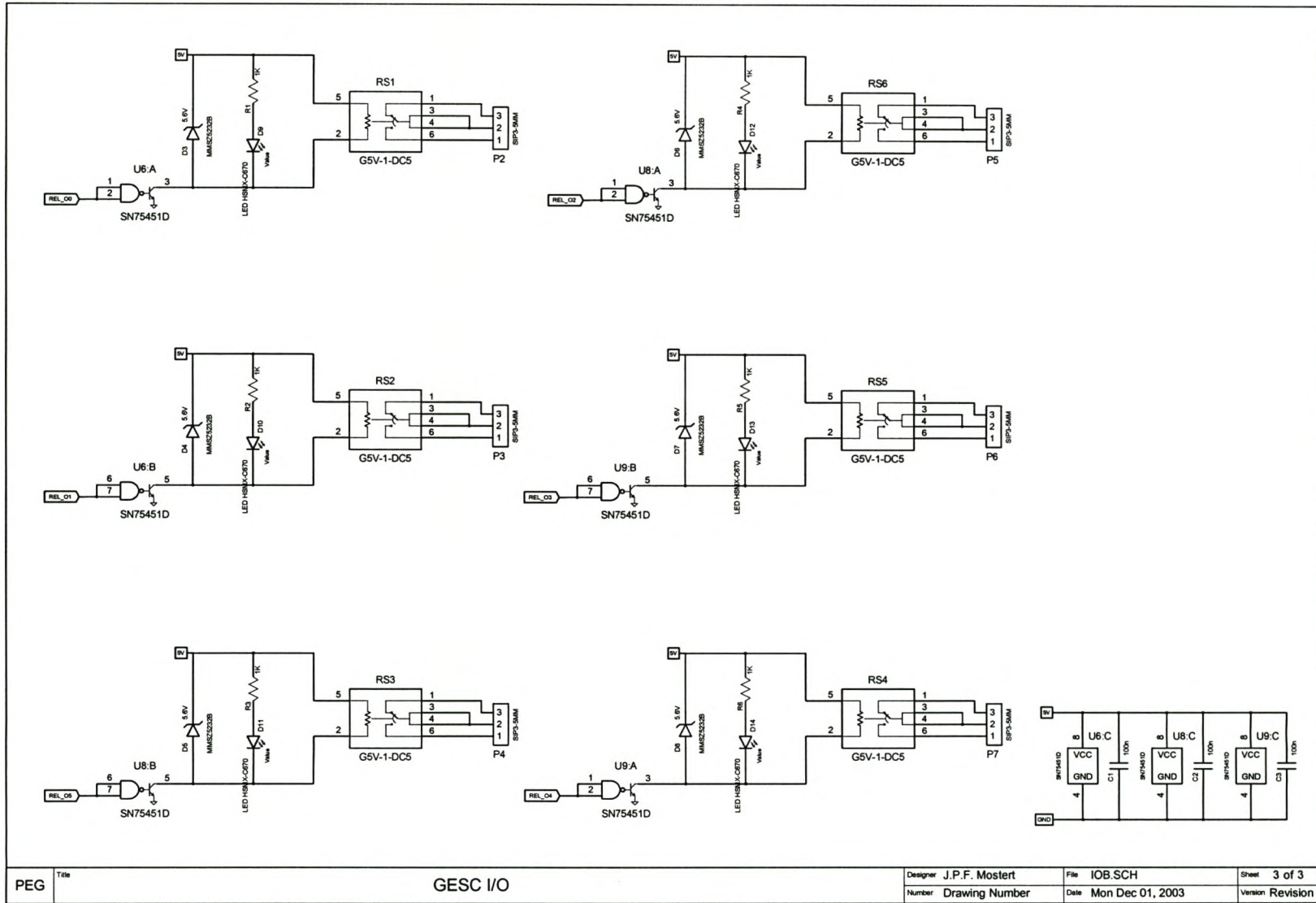


PEG	Title	GESC I/O	Designer	J.P.F. Mostert	File	IOB.SCH	Sheet	3 of 3
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision





PEG	Title	GESC I/O	Designer	J.P.F. Mostert	File	IOB.SCH	Sheet	3 of 3
			Number	Drawing Number	Date	Mon Dec 01, 2003	Version	Revision



D

***Appendix D    DSP Software***

---

---

```
/*.....-----
file:      GESC.c
project:   Generic Energy Storage Controller
developer: J.P.F. Mostert
company:   University of Stellenbosch
-----

2002/12/10 (ver.1.0.0)

/* ..... */

#include "F2407Regs.h"
#include "FPGA_ADD.h"
#define _GESC

void F2407_SYSTEM(void);
void F2407_IO_SETUP(void);
void F2407_TMR_PWM(void);
void MD_Delay(void);
void F2407_WD_EN(void);
void F2407_Int(void);
void Sin_Comp(void);
void F2407_Kick_Dog(void);
void DC_DC(void);

int Sin32767(int);
void WSSetup(void);

/*--
.....-----
Main Program
-----*/

void main(void)
{
    F2407_SYSTEM();
    F2407_IO_SETUP();
    F2407_TMR_PWM();
    MD_Delay();
    F2407_WD_EN();
    F2407_Int();
    while(1);
}

/*--.....-----
SYSTEM SETUP
-----*/

void F2407_SYSTEM(void)
{
```

---

---

```

#define wd_DISABLE (1<<6)
#define wd_CHK      0x28

iaWDCR   = wd_DISABLE|wd_CHK;      /* Disable Watchdog */
iaSCSR1  = 0x000C;                /* Enable EVA,EVB,A/D Clock */
WSSetup();                          /* Calling External ASM Program */

}

/* -----
                                     IO Setup
----- */

void F2407_IO_SETUP(void)
{
iaMCRA   = 0x2FC4;                /* PWM1-PWM6,IOPA0,IOPA1,XINT1,T2PWM */
iaMCRC   = 0x0007;                /* PWM7/8, CLKOUT selected*/
iaPADATDIR = 0x0303;             /* Set reset for FPGA, ext PWM high imp */
iaPEDATDIR = 0x6868;             /* IOPE3,IOPE5,IOPE6 = '1'= LED */
iaPBDATDIR = 0x9080;             /* IOPB7 LED1=OFF,T1PWM output */
}

/* -----
                                     Timer & Compare Setup
----- */

void F2407_TMR_PWM(void)
{
    /* Assume CPUCLK=40MHz

    Timer1: To Generate PWM
    Timer2: To Generate Accurate Timing

    Set to 5kHz (taking into account up/down counting mode,
    Period = TPRD*2).
    Set to continuous up/down mode (for PWM generation).
    Set prescale to 1 (clock divide by two).*/

iaCOMCONA = 0x0000; /* Put PWM EVA Output in High Impedance State */
iaCOMCONB = 0x0000; /* Put PWM EVB Output in High Impedance State */
iaGPTCONA = 0x0004; /* T2PIN Active low, High Impedance state */

iaT1PR    = 2000; /* 40MHz/5kHz/2/2=2000 ; Up-Down Count */
iaT2PR    = 2000;
iaT3PR    = 2000;

iaACTRA   = 0x0999; /* PWM2,4,6 Active High. PWM1,3,5 Active Low */
iaACTRB   = 0x0FF9; /* PWM8 Active High.PWM7 Active Low*/

iaCMPR1   = 1000; /* Gets Updated Later */
iaCMPR2   = 1000;
iaCMPR3   = 1000;
iaCMPR4   = 1000;
iaT2CMPR  = 1000; /* Gets Updated Later */

```

---



---

```

void F2407_Int(void)
{
    /*
        INT1: Power Drive Protection, External Int.
        INT2: Timer 1 Period Interrupt
        INT3: Timer 2 Period Interrupt
        INT5: None
        INT6: None
    */
    /*
asm (" CLRC INTM");    /* Enable Interrupts Globally (assembler Code) */

iaIFR      = 0xFFFF;    /* Clear all Flags */
iaIMR      = 0x0007;    /* Enable Int1,2,3 */
iaXINT1CR  = 0x0001;    /* Enable XINT1 */

iaEVAIFRA  = 0xFFFF;    /* Clear all Flags */
iaEVAIFRB  = 0xFFFF;    /* Clear all Flags */
iaEVBIFRA  = 0xFFFF;    /* Clear all Flags */
iaEVAIMRA  = 0x0081;    /* PDPINTA,Timer1 Period Int Enable */
iaEVAIMRB  = 0x0001;    /* Timer2 Period Interrupt Enable */
iaEVBIMRA  = 0x0080;    /* Timer3 Period Interrupt Enable */

}

/*.....
...
                                Sine Compare
--
.....
*/
void Sin_Comp(void)
{
    static int Freq_Cnt = 0;
    static int Amp_Cnt = 0;
    static int Amplitude = 0;
    static int Freq = 0;
    static int Ref_A;
    static int Ref_B;
    static int Ref_C;

    /* Load the counters that sine waves will be 120deg out of phase */

    static unsigned int cnt1 = 0;
    static unsigned int cnt2 = (64000/3);
    static unsigned int cnt3 = 2*(64000/3);

    if (++Freq_Cnt >= 80)    /* 5kHz/80 = 62.5 */
    {
        Freq_Cnt = 0;
        if (Freq < 648) Freq++;
    }
}

```

---

---

```

if (++Amp_Cnt >= 25)          /* 5kHz/25 = 400 */
{
    Amp_Cnt = 0;             /* 2000/200 = 10sek */
    if (Ampl < 1999) Ampl++;
}

cnt1 += Freq;                /* 64000/640=100 ; 5kHz/100=50Hz */
cnt2 += Freq;
cnt3 += Freq;

/*--*/
if (cnt1 >= 64000) cnt1 -= 64000; /* 2^6=64 ; 1000 Pts Sine Tabel */
Ref_A = (Sin32767(cnt1 >> 6)); /* 64*1000=64000 */
Ref_A = ((long)Ref_A * (long)Ampl) >> 16; /* Skaal -+ 32767 af na */
iaCMPR1 = 1000 + (Ref_A); /* -+ 1000 */

/*--*/
if (cnt2 >= 64000) cnt2 -= 64000;
Ref_B = (Sin32767(cnt2 >> 6));
Ref_B = ((long)Ref_B * (long)Ampl) >> 16;
iaCMPR2 = 1000 + (Ref_B);

/*--*/
if (cnt3 >= 64000) cnt3 -= 64000;
Ref_C = (Sin32767(cnt3 >> 6));
Ref_C = ((long)Ref_C * (long)Ampl) >> 16;
iaCMPR3 = 1000 + (Ref_C);
}

/*.....--
                                DC to DC
--.....*/

void DC_DC(void)
{
/*   static long cnt_table[21] = {53444,53106,52772,52442,52117,51795,
    51477,51163,50853,50547,50244,49945,49650,49358,49069,48784,48502,
    48223,47947,47675,47406}; /* For 32767 table */

    static long cnt_table[21] = {59858,59479,59105,58736,58372,58011,
    57655,57303,56956,56613,56274,55939,55608,55281,54958,54639,54323,
    54010,53701,53396,53095}; /* 12% aanpassing */

/*Measurement div by 2^4 with shift in FPGA;4096*8 = 32767 ;
[(measurement_sum) / (meas_cnt/16)]*16; */

unsigned int cnt_meas = 0;
unsigned int meas1 = 0;
unsigned int meas2 = 0;

```

---



```
unsigned int meas3 = 0;
unsigned int meas4 = 0;
unsigned int meas5 = 0;
unsigned int meas6 = 0;
unsigned int meas7 = 0;
unsigned int meas8 = 0;
unsigned int meas9 = 0;
unsigned int meas10 = 0;
unsigned int meas11 = 0;
unsigned int meas12 = 0;

static long meas1_off = 0;
static long meas2_off = 0;
static long meas3_off = 0;
static long meas4_off = 0;
static long meas5_off = 0;
static long meas7_off = 0;

static long meas1_zero = 0;
static long meas2_zero = 0;
static long meas3_zero = 0;
static long meas4_zero = 0;
static long meas5_zero = 0;
static long meas7_zero = 0;

float meas1_flt = 0.0;
float meas2_flt = 0.0;
float meas3_flt = 0.0;
float meas5_flt = 0.0;
float meas7_flt = 0.0;

static float duty1 = 0.0;
static float duty2 = 0.0;
static float duty3 = 0.0;

int duty1_int = 0;
int duty2_int = 0;
int duty3_int = 0;

float crrnt1 = 0.0;
float crrnt2 = 0.0;
float crrnt3 = 0.0;
float Ref_I_flt = 0.0;

unsigned static int Ref_I = 0;
unsigned static int Ref_Cnt = 0;
static int cnter1 = 0;
static unsigned int off_cnt = 0;
static unsigned int meas_flag = 0;

cnt_meas = *(MEAS_CNT);
meas1 = *(I_FN);
meas2 = *(I_AC);
meas3 = *(I_DC);
meas4 = *(I_FB);
```

---

---

```

meas5 = *(V_NA);
meas6 = *(V_DC1);
meas7 = *(V_NB);
meas8 = *(V_NC);
meas9 = *(I_FA);
meas10 = *(V_AC);
meas11 = *(I_FC);
meas12 = *(V_DC2);

if (cnt_meas > 177)
{
    iaCOMCONA = 0x0000; /* Put PWM Output in High Impedance State */
    iaCOMCONB = 0x0000;
    iaPADATDIR |= (1<<1); /* Disable ext PWM */
    iaGPTCONA &= ~(1<<6); /* Disable timer compare */
    iaPEDATDIR &= ~(1<<3); /* LED4 ON */
    iaPEDATDIR |= (1<<5); /* LED3 OFF */
    iaPEDATDIR |= (1<<6); /* LED2 OFF */
    while(1) F2407_Kick_Dog();
}

if (cnt_meas < 165)
{
    if (meas_flag == 1)
    {
        iaCOMCONA = 0x0000; /* Put PWM in High Impedance State */
        iaCOMCONB = 0x0000;
        iaPADATDIR |= (1<<1); /* Disable ext PWM */
        iaGPTCONA &= ~(1<<6); /* Disable timer compare */
        iaPEDATDIR &= ~(1<<3); /* LED4 ON */
        iaPEDATDIR |= (1<<5); /* LED3 OFF */
        iaPEDATDIR |= (1<<6); /* LED2 OFF */
        while(1) F2407_Kick_Dog();
    }
    else
    {
        meas1 = 21370;
        meas2 = 21370;
        meas3 = 21370;
        meas4 = 21370;
        meas5 = 21370;
        meas7 = 21370;
    }
}

cnt_meas -= 157;

/* Div and scale to 32767 */
meas1 = ((long)meas1 * cnt_table[cnt_meas]) >> 16;
meas2 = ((long)meas2 * cnt_table[cnt_meas]) >> 16;
meas3 = ((long)meas3 * cnt_table[cnt_meas]) >> 16;
meas4 = ((long)meas4 * cnt_table[cnt_meas]) >> 16;
meas4 = ((long)meas4 * cnt_table[cnt_meas]) >> 16;
meas5 = ((long)meas5 * cnt_table[cnt_meas]) >> 16;
meas7 = ((long)meas7 * cnt_table[cnt_meas]) >> 16;

```

---

---

```

if (off_cnt++ < 1024) /* 2^10 */
{
    meas1_off += ((meas1)-16383);          /* Calculating offset */
    meas2_off += ((meas2)-16383);
    meas3_off += ((meas3)-16383);
    meas4_off += ((meas4)-16383);
    meas5_off += ((meas5)-16383);

    meas1_zero += meas1;                  /* Calculating zero value */
    meas2_zero += meas2;
    meas3_zero += meas3;
    meas4_zero += meas4;
    meas5_zero += meas5;
}
else if (off_cnt == 1025)
{
    off_cnt += 1;
    meas1_off = meas1_off >> 10; /* Average offset */
    meas2_off = meas2_off >> 10;
    meas3_off = meas3_off >> 10;
    meas4_off = meas4_off >> 10;
    meas5_off = meas5_off >> 10;

    meas1_zero = meas1_zero >> 10; /* Average zero value */
    meas2_zero = meas2_zero >> 10;
    meas3_zero = meas3_zero >> 10;
    meas4_zero = meas4_zero >> 10;
    meas5_zero = meas5_zero >> 10;

    meas1_zero = meas1_zero-meas1_off; /* Zero value */
    meas2_zero = meas2_zero-meas2_off;
    meas3_zero = meas3_zero-meas3_off;
    meas4_zero = meas4_zero-meas4_off;
    meas5_zero = meas5_zero-meas5_off;
}
else
{
    off_cnt = 2000;
    iaCOMCONA = 0x8200; /* Enable PWM */
    iaCOMCONB = 0x8200;
    iaPADATDIR &= ~(1<<1); /* Enable ext PWM */
    iaGPTCONA |= (1<<6); /* Enable timer compare */

    meas7_off = 2135;
    meas7_zero = 16319;

/*--*/
if (meas1 <= meas1_off)
{
    meas1 = 0;
}
else
{
    meas1 = meas1-meas1_off;
    if (meas1 <= meas1_zero)
    {

```

---

```
        meas1 = 0;
    }
    else
    {
        meas1 = meas1-meas1_zero;
    }
}

/*--*/
if (meas2 < meas2_off)
{
    meas2 = 0;
}
else
{
    meas2 = meas2-meas2_off;
    if (meas2 <= meas2_zero)
    {
        meas2 = 0;
    }
    else
    {
        meas2 = meas2-meas2_zero;
    }
}

/*--*/
if (meas3 <= meas3_off)
{
    meas3 = 0;
}
else
{
    meas3 = meas3-meas3_off;
    if (meas3 <= meas3_zero)
    {
        meas3 = 0;
    }
    else
    {
        meas3 = meas3-meas3_zero;
    }
}

/*--*/
if (meas4 <= meas4_off)
{
    meas4 = 0;
}
else
{
    meas4 = meas4-meas4_off;
    if (meas4 <= meas4_zero)
    {
        meas4 = 0;
    }
    else

```

---

---

```
        {
            meas4 = meas4-meas4_zero;
        }
    }

/*--*/
if (meas5 <= meas5_off)
{
    meas5 = 0;
}
else
{
    meas5 = meas5-meas5_off;
    if (meas5 <= meas5_zero)
    {
        meas5 = 0;
    }
    else
    {
        meas5 = meas5-meas5_zero;
    }
}

/*--*/
if (meas7 <= meas7_off)
{
    meas7 = 0;
}
else
{
    meas7 = meas7-meas7_off;
    if (meas7 <= meas7_zero)
    {
        meas7 = 0;
    }
    else
    {
        meas7 = meas7-meas7_zero;
    }
}
}

if (++Ref_Cnt <= 16383 )      /* Generating a reference current step */
{
    Ref_I_flt = 3.0;
    cnter1 = 0;
}
else
{
    Ref_Cnt = 17000;
    Ref_I_flt = 12.0;
}

meas1_flt = (float)meas1;
meas2_flt = (float)meas2;
meas3_flt = (float)meas3;
meas5_flt = (float)meas5;
```

---

---

```
meas7_flt = (float)meas7;

meas1_flt = (meas1_flt/573.433);    /* Scale currents 1:1 */
meas2_flt = (meas2_flt/573.433);
meas3_flt = (meas3_flt/573.433);
meas5_flt = (meas5_flt/33.74);    /* Scale voltages 1:1 */
meas7_flt = (meas7_flt/33.74);

/* Start of control algorithm */

crrnt1 = (((duty1*meas7_flt) - meas5_flt)/4.8);
crrnt2 = (((duty2*meas7_flt) - meas5_flt)/4.28);
crrnt3 = (((duty3*meas7_flt) - meas5_flt)/4.14);
duty1 = (((3.40*(12.0 - (meas1_flt+crrnt1))) + meas5_flt)/meas7_flt) +
0.003); /* 3.40 */
duty2 = (((3.40*(12.0 - (meas2_flt+crrnt2))) + meas5_flt)/meas7_flt) +
0.003); /* 3.57 */
duty3 = (((3.40*(12.0 - (meas3_flt+crrnt3))) + meas5_flt)/meas7_flt) +
0.003); /* 3.45 */

duty1_int = (int)(2000.0*duty1);
duty2_int = (int)(2000.0*duty2);
duty3_int = (int)(2000.0*duty3);

if (cnter1 > 199) cnter1 = 199;

meas_test1[cnter1] = meas1;
meas_test2[cnter1] = meas2;
meas_test3[cnter1] = meas3;

/*--*/    if (duty1_int >= 2000)
    {
        duty1_int = 2000;
    }
    else if (duty1_int <= 0)
    {
        duty1_int = 0;
    }

/*--*/    if (duty2_int >= 2000)
    {
        duty2_int = 2000;
    }
    else if (duty2_int <= 0)
    {
        duty2_int = 0;
    }

/*--*/    if (duty3_int >= 2000)
    {
        duty3_int = 2000;
    }
    else if (duty3_int <= 0)
    {
        duty3_int = 0;
    }
```

---

---

```
iaCMPR1    = duty2_int;    /* Phase C meas2 */
iaCMPR4    = duty3_int;    /* Phase B with interleave switching */
iaT2CMPR   = duty1_int;    /* Phase A with interleave switching */
iaCMPR2    = duty3_int;    /* Phase B meas3, normal switching */
iaCMPR3    = duty1_int;    /* Phase A meas1, normal switching */

meas_flag = 1;
cnter1 += 1;
iaPBDATDIR |= (1<<7); /* LED1 OFF */
}

/*-----
                                   Kick Dog
-----*/

void F2407_Kick_Dog(void)
{
iaWDKEY = 0x55;
iaWDKEY = 0xAA;
}

/*-----

file:      GESC_Int.c
project:   Generic Energy Storage Controller
developer: J.P.F. Mostert
company:   University of Stellenbosch

-----*/

/* ----- */

#include "F2407Regs.h"
#define _GESC_Int

int IntErrCnt = 0;

/* ----- */

interrupt void Int_ERR(void)
{
    IntErrCnt++;
}

/* ----- */

interrupt void Int_1()
{
```

---

---

```

if ((iaEVAIFRA)&(1))
{
    iaPADATDIR |= (1<<1); /* Disable ext PWM */
    iaGPTCONA  &= ~(1<<6); /* Disable timer compare */
    iaCOMCONB = 0x0000; /* Put PWM Output in High Imp State */
    iaEVAIFRA |= (1);
    iaPEDATDIR &= ~(1<<3); /* LED4 ON */
    iaPEDATDIR |= (1<<5); /* LED3 OFF */
    iaPEDATDIR |= (1<<6); /* LED2 OFF */
    while(1) F2407_Kick_Dog();
}

if ((iaXINT1CR)&(1<<15))
{
    iaXINT1CR |= (1<<15); /* Clear XINT1 Flag */
    iaPEDATDIR &= ~(1<<6); /* LED2 ON */
    iaPBDATDIR &= ~(1<<4); /* Clear measurement */
/* DC_DC(); /* Call DC to DC programme */
   Sin_Comp(); /* Call AC to DC programme */
}

}

/* --
.....-- */

interrupt void Int_2()
{
if ((iaEVAIFRA)&(1<<7))
{
    iaEVAIFRA |= (1<<7); /* Clear Timer1 Period Flag */
    F2407_Kick_Dog(); /* Kick Watchdog */
    iaPBDATDIR |= (1<<4); /* Set measurement */
    iaPEDATDIR &= ~(1<<5); /* LED3 ON */
    iaPBDATDIR &= ~(1<<7); /* LED1 ON */
}

if ((iaEVBFRA)&(1<<7))
{
    iaEVAIFRB |= (1<<7); /* Clear Timer3 Period Flag */
    F2407_Kick_Dog(); /* Kick Watchdog */
}

}

/* --
.....-- */

interrupt void Int_3()
{
if ((iaEVAIFRB)&(1))
{

```

---



---

```

        iaEVAIFRB |= (1);          /* Timer2 Period Flag */
    }
}

; -----
;
; file:          GESC_Vec.asm
; project:       Generic Energy Storage Controller
; developer:     J.P.F. Mostert
; company:       University of Stellenbosch
; date:          10/12/2002
; version:       1.0.0
; -----

.sect "vectors" ; map to 0h in program space

.ref _Int_ERR
.ref _Int_1, _Int_2, _Int_3, _c_int0

B _c_int0          ;
B _Int_1           ; Power Drive Protection, XINT1
B _Int_2           ; Timer1/Timer3 Period Interrupt
B _Int_3           ; Timer2 Period Interrupt
B _Int_ERR         ; INT4 (maskable)
B _Int_ERR         ; INT5 (maskable)
B _Int_ERR         ; INT6 (maskable)
B _Int_ERR         ; Reserved
B _Int_ERR         ; INT8 (software)
B _Int_ERR         ; INT9 (software)
B _Int_ERR         ; INT10 (software)
B _Int_ERR         ; INT11 (software)
B _Int_ERR         ; INT12 (software)
B _Int_ERR         ; INT13 (software)
B _Int_ERR         ; INT14 (software)
B _Int_ERR         ; INT15 (software)
B _Int_ERR         ; INT16 (software)
B _Int_ERR         ; TRAP instruction
B _Int_ERR         ; Non Maskable
B _Int_ERR         ; Reserved
B _Int_ERR         ; INT20 (software)
B _Int_ERR         ; INT21 (software)
B _Int_ERR         ; INT22 (software)
B _Int_ERR         ; INT23 (software)
B _Int_ERR         ; INT24 (software)
B _Int_ERR         ; INT25 (software)
B _Int_ERR         ; INT26 (software)
B _Int_ERR         ; INT27 (software)
B _Int_ERR         ; INT28 (software)
B _Int_ERR         ; INT29 (software)
B _Int_ERR         ; INT30 (software)
B _Int_ERR         ; INT31 (software)
; -----

```

---

---

```
/*-----  
file:      FPGA_ADD.h  
project:   GESC  
developer: J.P.F. Mostert  
company:   University of Stellenbosch  
-----  
2003/04/22 (ver.1.0.0):  
*/  
  
#ifndef _FPGA_ADD  
#define _FPGA_ADD  
/* ----- */  
  
#include "F2407_TYPES.h"  
  
/* ----- */  
  
#define I_FB          ((VWORD*) 0x8000)  
#define V_NA          ((VWORD*) 0x8001)  
#define V_DC1         ((VWORD*) 0x8002)  
#define I_FN          ((VWORD*) 0x8003)  
#define V_NB          ((VWORD*) 0x8004)  
#define V_NC          ((VWORD*) 0x8005)  
#define I_FA          ((VWORD*) 0x8006)  
#define I_DC          ((VWORD*) 0x8007)  
#define V_AC          ((VWORD*) 0x8008)  
#define I_FC          ((VWORD*) 0x8009)  
#define I_AC          ((VWORD*) 0x800A)  
#define V_DC2         ((VWORD*) 0x800B)  
#define MEAS_CNT      ((VWORD*) 0x800C)  
  
/*-----*/  
  
#endif
```

---

E

***Appendix E    FPGA Firmware***

---

---

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY CUSTOM_DPRAM IS

    PORT(
        RAM_WREN          : IN  STD_LOGIC;
        RAM_WR_CLK       : IN  STD_LOGIC;
        RAM_WR_A         : IN  STD_LOGIC_VECTOR(5 DOWNTO 0);
        RAM_WR_D         : IN  STD_LOGIC_VECTOR(15 DOWNTO 0);

        WE               : IN  STD_LOGIC;
        RW               : IN  STD_LOGIC;
        RD               : IN  STD_LOGIC;
        DS               : IN  STD_LOGIC;
        DSP_CLK          : IN  STD_LOGIC;
        A                : IN  STD_LOGIC_VECTOR(10 DOWNTO 0);
        D                : INOUT STD_LOGIC_VECTOR(15 DOWNTO 0));

END CUSTOM_DPRAM;

ARCHITECTURE a OF CUSTOM_DPRAM IS

    COMPONENT altdpram1 IS
        PORT
        (
            data          : IN  STD_LOGIC_VECTOR (15 DOWNTO 0);
            wraddress     : IN  STD_LOGIC_VECTOR (5 DOWNTO 0);
            rdaddress     : IN  STD_LOGIC_VECTOR (5 DOWNTO 0);
            wren          : IN  STD_LOGIC := '1';
            rden          : IN  STD_LOGIC := '1';
            clock         : IN  STD_LOGIC ;
            q             : OUT STD_LOGIC_VECTOR (15 DOWNTO 0));
    END COMPONENT altdpram1;

    SIGNAL LPM_D_IN, LPM_D_OUT : STD_LOGIC_VECTOR(15 DOWNTO 0);
    SIGNAL LPM_RD_A, LPM_WR_A  : STD_LOGIC_VECTOR(5 DOWNTO 0);
    SIGNAL LPM_RDEN, LPM_WREN  : STD_LOGIC;
    SIGNAL LPM_CLK             : STD_LOGIC;

BEGIN

    S1 : altdpram1 PORT MAP
        (data => LPM_D_IN, wraddress => LPM_WR_A,
         rdaddress => LPM_RD_A, wren => LPM_WREN,
         rden => LPM_RDEN, clock => LPM_CLK,
         q => LPM_D_OUT);

    LPM_WREN <= RAM_WREN;
    LPM_WR_A <= RAM_WR_A;
    LPM_D_IN <= RAM_WR_D;
    LPM_CLK <= RAM_WR_CLK;

    PROCESS (RD)

```

---

---

```
BEGIN
  IF RD = '0' THEN
    IF A(10 DOWNT0 6) = "00000" THEN

      LPM_RD_A <= A(5 DOWNT0 0);
      LPM_RDEN <= '1';
      D <= (OTHERS => 'Z');
      D <= LPM_D_OUT;

    ELSE

      LPM_RDEN <= '0';
      D <= (OTHERS => 'Z');
    END IF;
  ELSE

    LPM_RDEN <= '0';
    D <= (OTHERS => 'Z');
  END IF;

END PROCESS;
END a;

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY LVDS_RECV IS

  PORT(

    RS                : IN  STD_LOGIC;
    T1PWM             : IN  STD_LOGIC;
    TXD               : IN  STD_LOGIC;
    PDPINTA           : OUT STD_LOGIC;
    XINT1             : OUT STD_LOGIC;
    OPTIC1_OUT        : OUT STD_LOGIC;
    OPTIC2_OUT        : OUT STD_LOGIC;
    TX_OPTIC          : OUT STD_LOGIC;
    ERR_T1            : IN  STD_LOGIC;

    RAM_WREN          : OUT STD_LOGIC;
    RAM_WR_CLK        : OUT STD_LOGIC;
    RAM_WR_A          : OUT STD_LOGIC_VECTOR(5 DOWNT0 0);
    RAM_WR_D          : OUT STD_LOGIC_VECTOR(15 DOWNT0 0);

    LVDS_RCLK         : IN  STD_LOGIC;
    LVDS_LOCK         : IN  STD_LOGIC;
    LVDS_ROUT         : IN  STD_LOGIC_VECTOR(9 DOWNT0 0);
    LVDS_DIN          : OUT STD_LOGIC_VECTOR(9 DOWNT0 0);
    LVDS_SYNC         : OUT STD_LOGIC);

END LVDS_RECV;

ARCHITECTURE a OF LVDS_RECV IS
```

---

---

```
TYPE STATE_TYPE IS (LVDS_S1, LVDS_S2, LVDS_ERR1, LVDS_ERR2);

SIGNAL state           : STATE_TYPE;
SIGNAL CNT             : INTEGER RANGE 0 TO 255;
SIGNAL CNT_HLD        : INTEGER RANGE 0 TO 255;
SIGNAL DSP_READY      : STD_LOGIC;
SIGNAL ADD_WR         : STD_LOGIC;
SIGNAL T1PWM_HLD      : STD_LOGIC;
SIGNAL T1PWM_HLD1     : STD_LOGIC;
SIGNAL LVDS_ERR1_FLG  : STD_LOGIC;
SIGNAL LVDS_ERR2_FLG  : STD_LOGIC;
SIGNAL XINT_FLG       : STD_LOGIC;
SIGNAL PDPINT_FLG     : STD_LOGIC;

BEGIN

PDPINTA <= NOT(PDPINT_FLG OR ERR_T1);
OPTIC2_OUT <= '0';

PROCESS (LVDS_RCLK)

TYPE ARRAY_MEAS IS ARRAY (0 TO 11) OF INTEGER RANGE 0 TO 4096;
VARIABLE MEAS_ARR : ARRAY_MEAS;
VARIABLE LVDS_DAT1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
VARIABLE LVDS_DAT2 : STD_LOGIC_VECTOR(7 DOWNTO 0);
VARIABLE MEAS_VAL : STD_LOGIC_VECTOR(11 DOWNTO 0);
VARIABLE MEAS_VAL_INT : INTEGER RANGE 0 TO 4095;
VARIABLE MEAS_CNT : INTEGER RANGE 0 TO 11;
VARIABLE MEAS_CNT_VEC : STD_LOGIC_VECTOR(5 DOWNTO 0);
VARIABLE STRT_FLG : STD_LOGIC;
VARIABLE MEAS_VEC : STD_LOGIC_VECTOR(11 DOWNTO 0);
VARIABLE DPR_WR : STD_LOGIC;

BEGIN
IF TXD = '1' THEN

STRT_FLG := '0';
CNT <= 2;
CNT_HLD <= 2;
PDPINT_FLG <= '0';
XINT1 <= '1';
OPTIC1_OUT <= '1';
OPTIC2_OUT <= '1';
TX_OPTIC <= '1';
MEAS_ARR(0) := 0;
MEAS_ARR(1) := 0;
MEAS_ARR(2) := 0;
MEAS_ARR(3) := 0;
MEAS_ARR(4) := 0;
MEAS_ARR(5) := 0;
MEAS_ARR(6) := 0;
MEAS_ARR(7) := 0;
MEAS_ARR(8) := 0;
MEAS_ARR(9) := 0;
MEAS_ARR(10) := 0;
```

---

---

```

MEAS_ARR(11) := 0;
state <= LVDS_S1;

ELSIF LVDS_RCLK'EVENT AND LVDS_RCLK = '1' THEN

CASE state IS
WHEN LVDS_S1 =>

    RAM_WR_CLK <= '1';

    IF LVDS_LOCK = '0' THEN
        IF LVDS_ROUT(9 DOWNT0 4) = "010101" THEN

            LVDS_DAT1 := LVDS_ROUT(3 DOWNT0 0);
            STRT_FLG := '1';
            MEAS_CNT := 0;
            state <= LVDS_S2;

        ELSIF LVDS_ROUT(9 DOWNT0 4) = "000101" AND
STRT_FLG = '1' THEN

            LVDS_DAT1 := LVDS_ROUT(3 DOWNT0 0);
            state <= LVDS_S2;

        ELSIF STRT_FLG = '1' AND LVDS_ROUT(9 DOWNT0 4)
/= "000101" THEN

            state <= LVDS_ERR1;

        ELSE

            CNT <= CNT_HLD;
            STRT_FLG := '0';
            state <= LVDS_S1;

        END IF;
    ELSE

        state <= LVDS_ERR2;

    END IF;

WHEN LVDS_S2 =>

    RAM_WR_CLK <= '0';

    IF LVDS_LOCK = '0' THEN
        IF LVDS_ROUT(9 DOWNT0 8) = "10" THEN

            PDPINT_FLG <= '0';
            LVDS_DAT2 := LVDS_ROUT(7 DOWNT0 0);
            MEAS_VAL := (LVDS_DAT1)&(LVDS_DAT2);
            MEAS_VAL_INT :=
CONV_INTEGER(UNSIGNED(MEAS_VAL));
            MEAS_ARR(MEAS_CNT) := MEAS_VAL_INT;

```

---

---

```

        IF DPR_WR = '1' THEN

            RAM_WREN <= '1';
            MEAS_CNT_VEC :=
CONV_STD_LOGIC_VECTOR(MEAS_CNT,6);
            RAM_WR_A <= MEAS_CNT_VEC;
            MEAS_VEC :=
CONV_STD_LOGIC_VECTOR(MEAS_ARR(MEAS_CNT),12);
            MEAS_ARR(MEAS_CNT) := MEAS_VAL_INT;
            RAM_WR_D <= MEAS_VAL(11 DOWNTO
0)&("0000"); -- Change(19 down to 4)
            ADD_WR <= '1';

            IF MEAS_CNT = 11 THEN

                DPR_WR := '0';
                ADD_WR <= '0';
                DSP_READY <= '0';
                XINT_FLG <= '0';
                XINT1 <= '0';

            END IF;

        ELSE

            XINT_FLG <= '1';
            XINT1 <= XINT_FLG;
            RAM_WREN <= '0';

        END IF;

        IF MEAS_CNT = 11 THEN

            STRT_FLG := '0'; -- New cycle start
            CNT <= CNT+1;

            IF DSP_READY = '1' THEN
                IF ADD_WR = '0' THEN

                    RAM_WREN <= '1';
                    RAM_WR_A <= "001100";
                    RAM_WR_D <=

CONV_STD_LOGIC_VECTOR(CNT,16);

                    CNT <= 2;
                    DPR_WR := '1';

                END IF;
            ELSE

                DPR_WR := '0';

            END IF;
        END IF;

```

---



---

```
T1PWM_HLD <= T1PWM;
T1PWM_HLD1 <= T1PWM_HLD;

IF T1PWM_HLD = '0' AND T1PWM = '1' THEN

    DSP_READY <= '1';

ELSIF T1PWM_HLD1 = '0' AND T1PWM = '1'
THEN

    SP_READY <= '1';

END IF;

MEAS_CNT := MEAS_CNT+1;
CNT_HLD <= CNT;

state <= LVDS_S1;

ELSE

    state <= LVDS_ERR1;

END IF;

ELSE

    state <= LVDS_ERR2;

END IF;

WHEN LVDS_ERR1 =>

    STRT_FLG := '0';
    LVDS_ERR1_FLG <= NOT(LVDS_ERR1_FLG);
    OPTIC1_OUT <= LVDS_ERR1_FLG; -- LED AAN AS ERR
    state <= LVDS_S1;

WHEN LVDS_ERR2 =>

    PDPINT_FLG <= '1';
    LVDS_ERR2_FLG <= '1';
    TX_OPTIC <= '0'; -- LED AAN AS ERR

END CASE;
END IF;
END PROCESS;
END a;
```

---

F

***Appendix F      GESMS Firmware***

---

---

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY TEST_PROJ IS

    PORT(

        RW_INT           : IN  STD_LOGIC;
        RD_INT           : IN  STD_LOGIC;
        CS0_INT          : IN  STD_LOGIC;
        CS1_INT          : IN  STD_LOGIC;
        D_INT            : IN  STD_LOGIC_VECTOR(11 DOWNTO 0);

        RW_A_EXT         : OUT STD_LOGIC;
        RD_A_EXT         : OUT STD_LOGIC;
        CS0_A_EXT        : OUT STD_LOGIC;
        CS1_A_EXT        : OUT STD_LOGIC;
        D_A_EXT          : INOUT STD_LOGIC_VECTOR(11 DOWNTO 0);
        D_A_INT_OUT      : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);

        RW_B_EXT         : OUT STD_LOGIC;
        RD_B_EXT         : OUT STD_LOGIC;
        CS0_B_EXT        : OUT STD_LOGIC;
        CS1_B_EXT        : OUT STD_LOGIC;
        D_B_EXT          : INOUT STD_LOGIC_VECTOR(11 DOWNTO 0);
        D_B_INT_OUT      : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);

        RW_C_EXT         : OUT STD_LOGIC;
        RD_C_EXT         : OUT STD_LOGIC;
        CS0_C_EXT        : OUT STD_LOGIC;
        CS1_C_EXT        : OUT STD_LOGIC;
        D_C_EXT          : INOUT STD_LOGIC_VECTOR(11 DOWNTO 0);
        D_C_INT_OUT      : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);

    END TEST_PROJ;

ARCHITECTURE a OF TEST_PROJ IS

    SIGNAL KAAS : STD_LOGIC_VECTOR(11 DOWNTO 0);

BEGIN

    RW_A_EXT <= RW_INT;
    RD_A_EXT <= RD_INT;
    CS0_A_EXT <= CS0_INT;
    CS1_A_EXT <= CS1_INT;

    RW_B_EXT <= RW_INT;
    RD_B_EXT <= RD_INT;
    CS0_B_EXT <= CS0_INT;
    CS1_B_EXT <= CS1_INT;

```

---

---

```

RW_C_EXT <= RW_INT;
RD_C_EXT <= RD_INT;
CS0_C_EXT <= CS0_INT;
CS1_C_EXT <= CS1_INT;

PROCESS (RW_INT)
BEGIN
    IF RW_INT = '0' THEN

        D_A_EXT <= D_INT;
        D_B_EXT <= D_INT;
        D_C_EXT <= D_INT;

        D_A_INT_OUT <= (OTHERS => 'Z');
        D_B_INT_OUT <= (OTHERS => 'Z');
        D_C_INT_OUT <= (OTHERS => 'Z');

    ELSE

        D_A_EXT <= (OTHERS => 'Z');
        D_B_EXT <= (OTHERS => 'Z');
        D_C_EXT <= (OTHERS => 'Z');

        D_A_INT_OUT <= D_A_EXT;
        D_B_INT_OUT <= D_B_EXT;
        D_C_INT_OUT <= D_C_EXT;

    END IF;
END PROCESS;
END a;

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;

ENTITY EPLD_MEAS IS

    PORT(
        CLK                : IN  STD_LOGIC;
        RS                 : IN  STD_LOGIC;
        LED                : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);

        LVDS_TCLK          : OUT STD_LOGIC;
        LVDS_SYNC          : OUT STD_LOGIC;
        LVDS_DIN           : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
        LVDS_ROUT          : IN  STD_LOGIC_VECTOR(9 DOWNTO 0);
        LVDS_LOCK          : IN  STD_LOGIC;
        LVDS_RCLK          : IN  STD_LOGIC;

        CS0_ABC            : OUT STD_LOGIC;
        CS1_ABC            : OUT STD_LOGIC;
        RW_ABC             : OUT STD_LOGIC;
        RD_ABC             : OUT STD_LOGIC;
        D_ABC              : OUT          STD_LOGIC_VECTOR(11 DOWNTO 0);

        SYNC_A             : IN  STD_LOGIC;

```

---

---

```

        SYNC_B           : IN  STD_LOGIC;
        SYNC_C           : IN  STD_LOGIC;
        CONVCLK_A        : OUT STD_LOGIC;
        CONVCLK_B        : OUT STD_LOGIC;
        CONVCLK_C        : OUT STD_LOGIC;
        D_A_RECV         : IN  STD_LOGIC_VECTOR(11 DOWNT0 0);
        D_B_RECV         : IN  STD_LOGIC_VECTOR(11 DOWNT0 0);
        D_C_RECV         : IN  STD_LOGIC_VECTOR(11 DOWNT0 0));

END EPLD_MEAS;

ARCHITECTURE a OF EPLD_MEAS IS
    TYPE STATE_TYPE IS(WR_S1, WR_S2, WR_S3, WR_S4, WR_S5, WR_S6,
WR_S7, WR_S8, WR_S9, WR_S10, WR_S11, WR_S12, WR_S13, WR_S14, WR_S15);
    TYPE STATE1_TYPE IS (CLK_S1, CLK_S2, CLK_S3, CLK_S4, CLK_S5,
CLK_S6, CLK_S7, CLK_S8);
    TYPE STATE2_TYPE IS(RD_S1, RD_S2, RD_S3, RD_S4);
    TYPE STATE3_TYPE IS(LVDS_S1, LVDS_S2, LVDS_S3, LVDS_S4, LVDS_S5,
LVDS_S6, LVDS_S7, LVDS_S8, LVDS_S9, LVDS_S10, LVDS_S11, LVDS_S12);
    TYPE STATE4_TYPE IS (TMG_S1, TMG_S2, TMG_S3, TMG_S4, TMG_S5,
TMG_S6, TMG_S7, TMG_S8, TMG_S9, TMG_S10, TMG_S11, TMG_S12, TMG_S13,
TMG_S14, TMG_S15, TMG_S16, TMG_S17, TMG_S18, TMG_S19, TMG_S20, TMG_S21,
TMG_S22, TMG_S23);

    SIGNAL      state      : STATE_TYPE;
    SIGNAL      state1     : STATE1_TYPE;
    SIGNAL      state2     : STATE2_TYPE;
    SIGNAL      state3     : STATE3_TYPE;
    SIGNAL      state4     : STATE4_TYPE;
    SIGNAL      RD_STRT    : STD_LOGIC_VECTOR(1 DOWNT0 0);
    SIGNAL      SLOWCLK    : STD_LOGIC;
    SIGNAL      RDCLK      : STD_LOGIC;
    SIGNAL      CSCLK      : STD_LOGIC;
    SIGNAL      RDPLS      : STD_LOGIC;
    SIGNAL      LVDSPLS    : STD_LOGIC;
    SIGNAL      CS0_TMP    : STD_LOGIC;
    SIGNAL      WR_TMP     : STD_LOGIC;
    SIGNAL      SYNC_ABC   : STD_LOGIC;
    SIGNAL      CNT        : INTEGER RANGE 0 TO 1023;
    SIGNAL      TMP_D_A1   : STD_LOGIC_VECTOR(19 DOWNT0 0);
    SIGNAL      TMP_D_B1   : STD_LOGIC_VECTOR(19 DOWNT0 0);
    SIGNAL      TMP_D_C1   : STD_LOGIC_VECTOR(19 DOWNT0 0);
    SIGNAL      TMP_D_C2   : STD_LOGIC_VECTOR(19 DOWNT0 0);

BEGIN

    LVDS_TCLK <= NOT(CLK);
    SYNC_ABC <= (SYNC_A OR SYNC_B OR SYNC_C);

    PROCESS (CLK)
    BEGIN
        IF RS = '0' THEN

            RD_STRT <= "00";
            LVDS_SYNC <= '1';
            CNT <= 0;

```

---

---

```
state <= WR_S1;
state2 <= RD_S1;

ELSIF CLK'EVENT AND CLK = '1' THEN
CASE state IS
  WHEN WR_S1 =>

    IF RD_STRT = "00" THEN

      CS0_TMP <= '0';
      WR_TMP <= '0';
      D_ABC <= "010000000001"; --0x401

      state <= WR_S2;

    ELSE

      state <= WR_S1;

    END IF;

  WHEN WR_S2 =>

      CS0_TMP <= '0';
      WR_TMP <= '1';
      state <= WR_S3;

  WHEN WR_S3 =>

      CS0_TMP <= '1';
      WR_TMP <= '1';
      state <= WR_S4;

  WHEN WR_S4 =>

      CS0_TMP <= '0';
      WR_TMP <= '0';
      D_ABC <= "010000000000";--0x400
      state <= WR_S5;

  WHEN WR_S5 =>

      CS0_TMP <= '0';
      WR_TMP <= '1';
      state <= WR_S6;

  WHEN WR_S6 =>

      CS0_TMP <= '1';
      WR_TMP <= '1';
      state <= WR_S7;

  WHEN WR_S7 =>
```

---

```
CS0_TMP <= '0';
WR_TMP <= '0';
D_ABC <= "000010011000";
state <= WR_S8;

WHEN WR_S8 =>

    CS0_TMP <= '0';
    WR_TMP <= '1';
    state <= WR_S9;

WHEN WR_S9 =>

    CS0_TMP <= '1';
    WR_TMP <= '1';
    state <= WR_S10;

WHEN WR_S10 =>

    CS0_TMP <= '0';
    WR_TMP <= '0';
    D_ABC <= "010010000010";--0x482

    state <= WR_S11;

WHEN WR_S11 =>

    CS0_TMP <= '0';
    WR_TMP <= '1';
    LVDS_SYNC <= '0';
    state <= WR_S12;

WHEN WR_S12 =>

    CS0_TMP <= '1';
    WR_TMP <= '1';
    LVDS_SYNC <= '0';
    state <= WR_S13;

WHEN WR_S13 =>

    CNT <= CNT+1;

    IF CNT = 1029 THEN

        RD_STRT <= "10";
        state <= WR_S14;

    END IF;

    state <= WR_S14;
```

---

---

```

        WHEN WR_S14 =>

            IF SYNC_ABC = '0' THEN
                RD_STRT <= "11";
                state <= WR_S14;
            ELSE
                RD_STRT <= "10";
                state <= WR_S14;
            END IF;

        WHEN WR_S15 =>

            RD_STRT <= "01";
            state <= WR_S1;

    END CASE;
END IF;
END PROCESS;

PROCESS (CLK)
BEGIN
    IF CLK'EVENT AND CLK = '0' THEN
        CASE state2 IS

            WHEN RD_S1 =>

                IF RDPLS = '1' THEN

                    TMP_D_A1 <= ("010101")&(D_A_RECV(11 DOWNTO
8))&("10")&(D_A_RECV(7 DOWNTO 0));
                    TMP_D_B1 <= ("000101")&(D_B_RECV(11 DOWNTO
8))&("10")&(D_B_RECV(7 DOWNTO 0));
                    TMP_D_C1 <= ("000101")&(D_C_RECV(11 DOWNTO
8))&("10")&(D_C_RECV(7 DOWNTO 0));
                    state2 <= RD_S2;

                END IF;

            WHEN RD_S2 =>

                IF RDPLS = '1' THEN

                    TMP_D_A1 <= ("000101")&(D_A_RECV(11 DOWNTO
8))&("10")&(D_A_RECV(7 DOWNTO 0));
                    TMP_D_B1 <= ("000101")&(D_B_RECV(11 DOWNTO
8))&("10")&(D_B_RECV(7 DOWNTO 0));
                    TMP_D_C2 <= ("000101")&(D_C_RECV(11 DOWNTO
8))&("10")&(D_C_RECV(7 DOWNTO 0));
                    state2 <= RD_S3;

                END IF;

            WHEN RD_S3 =>

                IF RDPLS = '1' THEN

```

---





```
WHEN LVDS_S4 =>

    LVDS_DIN <= TMP_D_B1(9 DOWNT0 0);
    state3 <= LVDS_S5;

WHEN LVDS_S5 =>

    LVDS_DIN <= TMP_D_C1(19 DOWNT0 10);
    state3 <= LVDS_S6;

WHEN LVDS_S6 =>

    LVDS_DIN <= TMP_D_C1(9 DOWNT0 0);
    state3 <= LVDS_S7;

WHEN LVDS_S7 =>

    IF LVDSPLS = '1' THEN
        LVDS_DIN <= TMP_D_A1(19 DOWNT0 10);
        LED(0) <= '1';
        state3 <= LVDS_S8;
    ELSE
        LVDS_DIN <= ("0011111111");
        LED(0) <= '0';
        state3 <= LVDS_S7;
    END IF;

WHEN LVDS_S8 =>

    LVDS_DIN <= TMP_D_A1(9 DOWNT0 0);
    state3 <= LVDS_S9;

WHEN LVDS_S9 =>

    LVDS_DIN <= TMP_D_B1(19 DOWNT0 10);
    state3 <= LVDS_S10;

WHEN LVDS_S10 =>

    LVDS_DIN <= TMP_D_B1(9 DOWNT0 0);
    state3 <= LVDS_S11;

WHEN LVDS_S11 =>

    LVDS_DIN <= TMP_D_C2(19 DOWNT0 10);
    state3 <= LVDS_S12;

WHEN LVDS_S12 =>

    LVDS_DIN <= TMP_D_C2(9 DOWNT0 0);
```

---

```
state3 <= LVDS_S1;

        END CASE;
    END IF;
END PROCESS;

PROCESS (CLK)
BEGIN
    IF CLK'EVENT AND CLK = '0' THEN
        CASE state1 IS
            WHEN CLK_S1 =>

                SLOWCLK <= '0';
                state1 <= CLK_S2;

            WHEN CLK_S2 =>

                SLOWCLK <= '0';
                state1 <= CLK_S3;

            WHEN CLK_S3 =>

                SLOWCLK <= '0';
                state1 <= CLK_S4;

            WHEN CLK_S4 =>

                SLOWCLK <= '1';
                state1 <= CLK_S5;

            WHEN CLK_S5 =>

                SLOWCLK <= '1';
                state1 <= CLK_S6;

            WHEN CLK_S6 =>

                SLOWCLK <= '1';
                state1 <= CLK_S1;

            WHEN CLK_S7 =>

                SLOWCLK <= '1';
                state1 <= CLK_S8;

            WHEN CLK_S8 =>

                SLOWCLK <= '1';
                state1 <= CLK_S1;

        END CASE;
    END IF;
END PROCESS;

PROCESS (CLK)
BEGIN
```

---

---

```
IF CLK'EVENT AND CLK = '0' THEN
  CASE state4 IS
    WHEN TMG_S1 => --start

      CSCLK <= '1';
      RDCLK <= '1';
      RDPLS <= '0';
      LVDSPLS <= '0';

      IF RD_STRT = "11" THEN

        state4 <= TMG_S2;

      ELSE

        state4 <= TMG_S1;

      END IF;

    WHEN TMG_S2 => --1

      CSCLK <= '0';
      RDCLK <= '0';
      RDPLS <= '1';
      LVDSPLS <= '0';

      state4 <= TMG_S3;

    WHEN TMG_S3 => --1

      CSCLK <= '0';
      RDCLK <= '1';
      RDPLS <= '0';
      LVDSPLS <= '0';

      state4 <= TMG_S4;

    WHEN TMG_S4 => --1

      CSCLK <= '1';
      RDCLK <= '1';
      RDPLS <= '0';
      LVDSPLS <= '1';

      state4 <= TMG_S5;

    WHEN TMG_S5 => --2

      CSCLK <= '1';
      RDCLK <= '1';
      RDPLS <= '0';
      LVDSPLS <= '0';
```

---

```
state4 <= TMG_S6;

WHEN TMG_S6 => --2

    CSCLK <= '1';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '0';

    state4 <= TMG_S7;

WHEN TMG_S7 => --2

    CSCLK <= '1';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '0';

    state4 <= TMG_S8;

WHEN TMG_S8 => --2

    CSCLK <= '0';
    RDCLK <= '0';
    RDPLS <= '1';
    LVDSPLS <= '0';

    state4 <= TMG_S9;

WHEN TMG_S9 => --2

    CSCLK <= '0';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '0';

    state4 <= TMG_S10;

WHEN TMG_S10 => --2

    CSCLK <= '1';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '1';

    state4 <= TMG_S11;

WHEN TMG_S11 => --3

    CSCLK <= '1';
    RDCLK <= '1';
```

---

---

```
RDPLS <= '0';
LVDSPLS <= '0';

state4 <= TMG_S12;

WHEN TMG_S12 => --3

    CSCLK <= '1';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '0';

    state4 <= TMG_S13;

WHEN TMG_S13 => --3

    CSCLK <= '1';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '0';

    state4 <= TMG_S14;

WHEN TMG_S14 => --3

    CSCLK <= '0';
    RDCLK <= '0';
    RDPLS <= '1';
    LVDSPLS <= '0';

    state4 <= TMG_S15;

WHEN TMG_S15 => --3

    CSCLK <= '0';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '0';

    state4 <= TMG_S16;

WHEN TMG_S16 => --3

    CSCLK <= '1';
    RDCLK <= '1';
    RDPLS <= '0';
    LVDSPLS <= '1';

    state4 <= TMG_S17;

WHEN TMG_S17 => --4
```

---

```
        CSCLK <= '1';
        RDCLK <= '1';
        RDPLS <= '0';
        LVDSPLS <= '0';

        state4 <= TMG_S18;
WHEN TMG_S18 => --4

        CSCLK <= '1';
        RDCLK <= '1';
        RDPLS <= '0';
        LVDSPLS <= '0';

        state4 <= TMG_S19;
WHEN TMG_S19 => --4

        CSCLK <= '1';
        RDCLK <= '1';
        RDPLS <= '0';
        LVDSPLS <= '0';

        state4 <= TMG_S20;
WHEN TMG_S20 => --4

        CSCLK <= '0';
        RDCLK <= '0';
        RDPLS <= '1';
        LVDSPLS <= '0';

        state4 <= TMG_S21;
WHEN TMG_S21 => --4

        CSCLK <= '0';
        RDCLK <= '1';
        RDPLS <= '0';
        LVDSPLS <= '0';

        state4 <= TMG_S22;
WHEN TMG_S22 => --4

        CSCLK <= '1';
        RDCLK <= '1';
        RDPLS <= '0';
        LVDSPLS <= '1';
```

---

---

```
        state4 <= TMG_S23;

        WHEN TMG_S23 => --end

        CSCLK <= '1';
        RDCLK <= '1';
        RDPLS <= '0';
        LVDSPLS <= '0';

        state4 <= TMG_S1;

        END CASE;
    END IF;
END PROCESS;

WITH RD_STRT(1) SELECT

CS0_ABC <=  CS0_TMP    WHEN '0',
            CSCLK      WHEN '1',
            '1'        WHEN OTHERS;

WITH RD_STRT(1) SELECT

CS1_ABC <=  NOT(CS0_TMP) WHEN '0',
            NOT(CSCLK)   WHEN '1',
            '0'          WHEN OTHERS;

WITH RD_STRT(1) SELECT

RD_ABC <=  '1'        WHEN '0',
            RDCLK      WHEN '1',
            '1'        WHEN OTHERS;

WITH RD_STRT(1) SELECT

RW_ABC <=  WR_TMP     WHEN '0',
            '1'        WHEN '1',
            '1'        WHEN OTHERS;

WITH RD_STRT(1) SELECT

CONVCLK_A <= SLOWCLK  WHEN '1',
            '1'        WHEN OTHERS;

WITH RD_STRT(1) SELECT

CONVCLK_B <= SLOWCLK  WHEN '1',
            '1'        WHEN OTHERS;

WITH RD_STRT(1) SELECT

CONVCLK_C <= SLOWCLK  WHEN '1',
            '1'        WHEN OTHERS;
```

---



```
LED(3 DOWNT0 1) <= "111";  
END a;
```