

1-1-2008

# Design And Development Of Power And Attitude Control Subsystems For RYESAT

Michael William Richard. Alger  
*Ryerson University*

Follow this and additional works at: <http://digitalcommons.ryerson.ca/dissertations>

 Part of the [Mechanical Engineering Commons](#)

---

## Recommended Citation

Alger, Michael William Richard., "Design And Development Of Power And Attitude Control Subsystems For RYESAT" (2008). *Theses and dissertations*. Paper 1075.

This Thesis is brought to you for free and open access by Digital Commons @ Ryerson. It has been accepted for inclusion in Theses and dissertations by an authorized administrator of Digital Commons @ Ryerson. For more information, please contact [bcameron@ryerson.ca](mailto:bcameron@ryerson.ca).

TK  
3260  
ASC  
2008

# DESIGN AND DEVELOPMENT OF POWER AND ATTITUDE CONTROL SUBSYSTEMS FOR RYESAT

By Michael William Richard Alger

B.Eng. Ryerson University 2006

A Thesis Presented to Ryerson University  
in partial fulfillment of the requirement  
for the degree of Masters of Applied Science  
in the Program of Mechanical Engineering

Toronto, Ontario, Canada, 2008

©Michael William Richard Alger 2008

PROPERTY OF  
RYERSON UNIVERSITY LIBRARY



---

## Authors' Declaration

I hereby declare that I am the sole author of this thesis or dissertation. I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

---

Mike Alger

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

---

Mike Alger





---

# Abstract

## **Design and Development of Power and Attitude Control Subsystems for RyeSat**

Masters of Applied Science 2008, Mike Alger

School of Graduate Studies Ryerson University

This thesis describes the design and development of Ryerson University's first CubeSat (RyeSat) with a focus on power and attitude control subsystems. This satellite is intended to become the initial of a series of CubeSats built by Ryerson University to perform research in spacecraft control algorithms and actuators. RyeSat is built around a standard interface, which specifies both a data-bus and a switchable power supply system for non critical systems. To facilitate the development of this satellite a prototype power subsystem was created, programmed and tested. In addition to developing the system's architecture and power subsystem; analysis was preformed to size both reaction wheels and magnetic torquers. This analysis showed that a commercially available motor could be adapted to fulfill the attitude control requirements of a CubeSat and also showed that miniature magnetic torque rods would be more efficient than magnetic torque coils typically used on CubeSats. Finally, control laws for these actuators were designed and an adaptive nonlinear sliding mode controller for reaction wheels was applied to control the 3-axis attitude motion of RyeSat.



---

# Acknowledgements

I would like to thank my parents, without whose reassurance and patience I would not have had the chance to pursue this project.

I would also like to thank my supervisor, Dr Krishna Kumar, for guiding me through this research and providing me with financial support through his grants.

Thanks to all those in the lab who have kept me relatively sane; it was definitely a valiant effort on your part.

MIKE ALGER September 2008

---

# Table of Contents

Authors' Declaration .....	iii
Abstract .....	v
Acknowledgements .....	vii
Table of Contents .....	viii
List of Figures .....	xii
List of Tables .....	xv
List of Acronyms .....	xviii
Symbols Used .....	xix
<b>Chapter 1    Introduction .....</b>	<b>1</b>
1.1    Motivation .....	1
1.2    Small Satellite Design: State of Art .....	2
1.3    Research Objectives .....	3
1.4    Contributions of Thesis .....	5
1.5    Thesis Organization .....	6
<b>Chapter 2    Literature Review .....</b>	<b>7</b>
2.1    Structure .....	7
Requirements .....	8
2.2    Attitude and Orbit Determination and Control System .....	10
2.3    Communications .....	11
2.4    Command and Data Handling .....	12
2.5    Power .....	13
2.6    Payload .....	14
2.7    System Bus .....	15
2.7.1    Spacecraft Interfaces .....	15
2.7.2    Past CubeSats and Their Bus Interfaces .....	17
2.8    Fabrication for a Space Environment .....	21
2.8.1    Environment .....	21
2.8.2    Space Manufacturing .....	24

<b>2.9</b>	<b>Discussions on CubeSats Performance.....</b>	<b>25</b>
2.9.1	CubeSat Failures and Problems.....	25
2.9.2	Checklists and Documentation .....	26
2.9.3	CubeSat Successes .....	27
<b>Chapter 3</b>	<b>System Requirements.....</b>	<b>29</b>
<b>3.1</b>	<b>Overall mission.....</b>	<b>29</b>
<b>3.2</b>	<b>Mass and Power Budgets.....</b>	<b>30</b>
3.2.1	System bus .....	30
3.2.2	Environmental/ thermal.....	31
3.2.3	Power .....	32
3.2.4	Attitude Determination and Control.....	33
3.2.5	C&DH/Comm Subsystem .....	34
3.2.6	Payload.....	35
<b>Chapter 4</b>	<b>System Analysis and Design.....</b>	<b>37</b>
<b>4.1</b>	<b>Architecture of RyeSat.....</b>	<b>38</b>
4.1.1	Advantages of an I <sup>2</sup> C Data Bus .....	40
4.1.2	Disadvantages of an I <sup>2</sup> C Based Standard.....	40
<b>4.2</b>	<b>Thermal Design .....</b>	<b>41</b>
	Analysis42	
<b>4.3</b>	<b>Power Subsystem .....</b>	<b>43</b>
	Analysis45	
<b>4.4</b>	<b>Attitude Control Subsystem .....</b>	<b>46</b>
4.4.1	Spacecraft Disturbance Torques .....	47
4.4.2	Spacecraft Actuator Sizing .....	51
<b>Chapter 5</b>	<b>Hardware Testing and Simulation.....</b>	<b>61</b>
<b>5.1</b>	<b>Bus Operation .....</b>	<b>61</b>
5.1.1	Case 1 .....	62
5.1.2	Case 2 .....	62
5.1.3	Case 3 .....	63
5.1.4	Case 4 .....	63
5.1.5	Case 5 .....	64
5.1.6	Case 6 .....	64
5.1.7	Case 7 .....	64



---

5.1.8	Case 8.....	65
5.1.9	Case 9.....	65
5.1.10	Checksums and Error Codes .....	65
<b>5.2</b>	<b>Thermal Subsystem.....</b>	<b>66</b>
5.2.1	Heater Control .....	67
5.2.2	Health Monitoring .....	68
<b>5.3</b>	<b>Power subsystem .....</b>	<b>68</b>
5.3.1	Uninterruptible Feed lines .....	68
5.3.2	Controllable Feedlines .....	71
5.3.3	Feedline Control Mechanism .....	72
5.3.4	Solar Cell Measurement and Power Transfer Bus .....	74
5.3.5	Battery Charger.....	75
5.3.6	Battery Monitor .....	76
5.3.7	System Wide Interface.....	76
5.3.8	Microcontroller .....	77
5.3.9	Overall Schematic Design .....	77
5.3.10	Physical Implementation .....	80
5.3.11	Firmware Design .....	80
5.3.12	Commands to Control Power Subsystem .....	82
<b>5.4</b>	<b>Attitude Control Subsystem.....</b>	<b>84</b>
5.4.1	H-Bridge .....	85
5.4.2	Microcontroller .....	85
5.4.3	Physical Implementation .....	86
<b>5.5</b>	<b>Control Law Development .....</b>	<b>86</b>
5.5.1	Rotation Sequences .....	86
5.5.2	Spacecraft Dynamics.....	87
5.5.3	Linear Control Law .....	89
5.5.4	Magnetic PD Controller .....	91
5.5.5	Sliding Mode Adaptive Controller.....	94
<b>Chapter 6</b>	<b><i>Conclusions and Future work.....</i></b>	<b>105</b>
<b>6.1</b>	<b>Conclusions.....</b>	<b>105</b>
6.1.1	System Bus.....	105
6.1.2	Power Subsystem.....	106
6.1.3	ACS.....	106

<b>6.2</b>	<b>Future work.....</b>	<b>106</b>
6.2.1	System Bus .....	106
6.2.2	Power Subsystem .....	107
6.2.3	ACS &ADS .....	107
	<b>References .....</b>	<b>108</b>
<i>Appendix A</i>	<i>Thermal Calculations.....</i>	<i>111</i>
<i>Appendix B</i>	<i>I<sup>2</sup>C Communications Test Code .....</i>	<i>116</i>
<b>A.1.</b>	<b>Single Byte Each Way.....</b>	<b>116</b>
A.1.1.	Overview .....	116
A.1.2.	Slave .....	116
A.1.3.	Master .....	121
<i>Appendix C</i>	<i>Power Subsystem firmware .....</i>	<i>125</i>
<i>Appendix D</i>	<i>Power Subsystem Code.....</i>	<i>127</i>
<i>Appendix E</i>	<i>Additional mathematics .....</i>	<i>143</i>
<i>Appendix F</i>	<i>Cube sat specifications document .....</i>	<i>145</i>



---

# List of Figures

Figure 1: Typical CubeSat System Architecture .....	7
Figure 2: Location of highest stress (Intelligent Space Systems Laboratory 2001) .....	9
Figure 3: CubeSat Specification drawing (HutputtEanasin and Toorian 2006, 7).....	9
Figure 4: Diagram of a typical Mil-STD-1553 bus (Wikipedia 2007) .....	16
Figure 5: SpaceWire configuration (ESA 2003) .....	17
Figure 6: Example SPI interface (Wikipedia 2008) .....	18
Figure 7: KUTESat system block diagram original (Paruchuri 2006) .....	18
Figure 8: KUTESat final configuration (Paruchuri 2006) .....	18
Figure 9: Xi 3 system block diagram (Intelligent Space Systems Laboratory 2001).....	19
Figure 10: nCube System block diagram (Øverby 2004).....	20
Figure 11: CubeSat kit Interface (Pumkin 2007) .....	21
Figure 12: Satellite configuration drawing.....	37
Figure 13: System block diagram (hardware specific) .....	38
Figure 14: Systems block representation of the proposed standard. ....	40
Figure 15: Electrical Schematic of Interface used on RyeSat. ....	40
Figure 16: Thermal subsystem block diagram .....	42
Figure 17: Power subsystem function block diagram .....	44
Figure 18: ACS hardware block diagram .....	47
Figure 19: Considered shape to calculate worst case gravity gradient .....	48
Figure 20: Layout of potential current loops .....	49
Figure 21: Picture of Portescap motor.....	52
Figure 22: Flywheel: solid disc configuration.....	53
Figure 23: Mass efficient flywheel design.....	54
Figure 24: Physical network (Sturman 2007).....	61
Figure 25: Sample data sentence.....	62
Figure 26: Thermal system schematic .....	67
Figure 27: 3.3V uninterruptible supply circuit .....	69
Figure 28: 5 volt step up and supply circuit .....	71

Figure 29: 3.3V 250 mA power supply lines (PS 1 and 3) .....	71
Figure 30: 3.3V 800 mA power supply line (PS 2 and 4) .....	71
Figure 31: Feed line control mechanism .....	72
Figure 32: Two solar cell circuit .....	75
Figure 33: Battery charger circuit .....	76
Figure 34: Battery monitor circuit .....	76
Figure 35: Interface to the satellite bus .....	77
Figure 36: Microcontroller pin-out .....	77
Figure 37: Power subsystem schematic .....	78
Figure 38: Power subsystem schematic continued .....	79
Figure 39: PCB layout .....	80
Figure 40: Firmware Design (Main loop) .....	81
Figure 41: Firmware design (Interrupt) .....	81
Figure 42: Firmware design (Interrupt) continued .....	81
Figure 43: Schematic for the ACS subsystem (magnetic torquers only) .....	85
Figure 44: PCB layout (preliminary) .....	86
Figure 45: Angular response PD controller. ....	91
Figure 46: Momentum storage for a simple PD controller .....	91
Figure 47: Torques for PD controller .....	91
Figure 48: Magnetic PD controller response .....	93
Figure 49: Magnetic PD controller input .....	94
Figure 50: Attitude response (ideal cube MOI) .....	98
Figure 51: Torque and adaptive gains (ideal cube MOI) .....	99
Figure 52: Reaction wheel storage momentum (ideal cube MOI) .....	99
Figure 53: Attitude response (worst case MOI) .....	100
Figure 54: Torque and adaptive gains (worst case MOI) .....	100
Figure 55: Reaction wheel storage momentum (worst case MOI) .....	101
Figure 56: Attitude maneuver response (ideal cube MOI) .....	102
Figure 57: Torque and adaptive gains while maneuvering (ideal cube MOI) .....	103
Figure 58: Reaction wheel storage momentum while maneuvering (ideal cube MOI) .....	103
Figure 59: Attitude maneuver response (worst case MOI) .....	103

---

Figure 60: Torque and adaptive gains while maneuvering (worst case MOI) .....	104
Figure 61: Reaction wheel storage momentum while maneuvering (worst case MOI)) .....	104
Figure 62: Power subsystem main loop & WDT .....	125
Figure 63: Interrupt error checking steps .....	125
Figure 64: Interrupt continued .....	126



---

## List of Tables

Table 1: Classification of spacecraft by mass (Kumar 2006) .....	3
Table 2: Summary of missions.....	4
Table 3: Typical Requirements for a Pico-satellite .....	8
Table 4: Summary of attitude and orbit control systems .....	11
Table 5: Summary of communication systems .....	12
Table 6: Summary of C&DH systems.....	13
Table 7: Summary of power systems .....	13
Table 8: Summary of Payloads .....	15
Table 9: Overall system requirements .....	29
Table 10: Design budget.....	30
Table 11: System bus requirements.....	30
Table 12: Environmental requirements .....	31
Table 13: Power requirements.....	32
Table 14: ADCS requirements .....	33
Table 15: C&DH requirements .....	34
Table 16: Payload requirements .....	35
Table 17: Interface matrix .....	42
Table 18: Spacecraft overall temperature at 1000 km without heaters.....	43
Table 19: Spacecraft overall temperature at 1000 km with 2.35W heater .....	43
Table 20: Spacecraft overall temperature at 500 km without heaters.....	43
Table 21: Spacecraft overall temperature and at km with 2.35W heater .....	43
Table 22: Interface matrix .....	44
Table 23: Solar cell power generation assumptions .....	45
Table 24: Power outputs from solar cells.....	45
Table 25: Power consumption (without heaters on) .....	46
Table 26: Power consumption (with heaters on).....	46
Table 27: Interface matrix .....	47
Table 28: Disturbance torque models.....	48

---

Table 29: Estimated magnetic moments from current loops .....	49
Table 30: disturbance torques at 400km .....	50
Table 31: disturbance torques @ 1000km.....	50
Table 32: Reaction wheel sizing criteria.....	51
Table 33: Example motor constants .....	51
Table 34: Example motor characteristics.....	52
Table 35: Fly wheel sizing considerations .....	52
Table 36: Fly wheel: solid disc parameters .....	53
Table 37: Simple ring fly wheel parameters .....	54
Table 38: Comparison to other available wheels.....	54
Table 39: Magnetic torquer requirements .....	55
Table 40: Air core design parameters .....	57
Table 41: Ferrite core design (magnetic parameters).....	58
Table 42: Ferrite core design (mass parameters) .....	58
Table 43: Ferrite core design (size parameters) .....	59
Table 44: Comparison of available magnetic torquers .....	59
Table 45: I <sup>2</sup> C Address table for RyeSat.....	61
Table 46: Case 1 (Master sends no response wanted) .....	62
Table 47: Case 2(Master sends, and a response is wanted) .....	63
Table 48: Case 3 (Master receives a wanted response) .....	63
Table 49: Case 4 (incorrect packet size).....	64
Table 50: Case 5(lost byte from master).....	64
Table 51: Case 6 (lost byte from Slave).....	64
Table 52: Case 7 (incorrect checksum on slave) .....	65
Table 53: Case 9 (error follow up).....	65
Table 54: List of standard bus error codes.....	66
Table 55: Temperature sensor address per board .....	68
Table 56: LTC 1440 resistor sizing table .....	74
Table 57: Key Diode Characteristics.....	74
Table 58: Summary of the communications test command.....	82
Table 59: Communications test command input.....	82

---

Table 60: Communications test output.....	82
Table 61: Summary of the Power distribution control command .....	82
Table 62: Example input for the power control command .....	83
Table 63: Example output of the power control command .....	83
Table 64: Summary of the solar Cell inquiry command .....	83
Table 65: Input for the solar cell inquiry command .....	83
Table 66: Structure of the output for the solar cell inquiry command .....	83
Table 67: Summary of the Battery inquiry command .....	84
Table 68: Input for the battery inquiry command .....	84
Table 69: Structure of the battery inquiry output .....	84
Table 70: initial conditions for PD controller .....	90
Table 71: Initial conditions for magnetic PD Controller .....	93
Table 72: Initial conditions ideal cube.....	98
Table 73: initial conditions worst case MOI .....	98
Table 74: Attitude maneuver initial conditions (ideal cube).....	101
Table 75: Attitude maneuver initial conditions (worst case MOI) .....	102
Table 76: Sides 1- 3 in a 1000 km orbit (hot and cold cases) .....	111
Table 77: Sides 4- 6 in a 1000 km orbit (hot and cold cases) .....	112
Table 78 Sides 1- 3 in a 500 km orbit (hot and cold cases) .....	113
Table 79: Sides 4- 6 in a 500 km orbit (hot and cold cases) .....	114



---

## List of Acronyms

EDAC	Error detection and correction
RAM	Random access memory
ROM	Read only memory
PROM	Programmable read only memory
TNC	Terminal node controller
DTMF	Dual tone multi frequency
FM	Frequency modulated
CW	Continuous wave (Morse code)
AOCS	Attitude orbit control system
ADS	Attitude determination system
ACS	Attitude Control system
GPS	Global positioning system
PCB	Printed Circuit board
P-POD	Poly Pico-Satellite Orbital Deployer
SPI	Serial peripheral interface
I <sup>2</sup> C	Inter-integrated circuit
UART	Universal asynchronous receiver transmitter
LEO	Low earth orbit
COTS	Commercial off the shelf
CSA	Canadian Space Agency
SSDCG	Space Systems Dynamics and Controls Group
TRL	Technology Readiness levels
Vref	Voltage Reference

---

# Symbols Used

$\tau_{gg}$	Torque due to gravity gradient disturbances
$\tau_{srp}$	Torque due to Solar radiation disturbances
$\tau_{mag}$	Torque due to magnetic disturbances
$\tau_{aero}$	Torque due to aerodynamic disturbances
$\tau_{dist}$	Sum of disturbance torques
$\tau_{wheel}$	Torque from reaction wheel
$\tau_{motor}$	Torque supplied from motor
$\tau_{torquer}$	Torque supplied from magnetic torque rod
$\omega_{motor}$	Wheel speed of motor (reaction wheel)
$H_{storage}$	Storage momentum from reaction wheel
$I_{xx}; I_{yy}; I_{zz}$	Principal moments of inertia
$\mu$	Earth's gravity constant
$r_{orbit}$	Orbital Radius of satellite
$F_s$	Solar energy constant
$A_s$	Exposed area
$c$	speed of light
$q$	reflectance factor
$i$	The angle of incidence of the Sun to the panel
$C_{ps}$	Center of pressure (solar)
$C_m$	Center of mass
$\rho$	density of the atmosphere at altitude
$v$	The speed of the spacecraft (magnitude)
$C_D$	Coefficient of drag
$C_{pa}$	Center of pressure(aero)
$V_{applied}$	Applied voltage
$k_i$	Motor torque constant
$R$	Resistance
$P_{motor}$	Power required by motor [W]



---

$m_{\text{applied}}$	Applied magnetic moment
$B_{\text{earth}}$	Magnetic field of earth at a given position
$I$	Current
$A_{\text{coil}}$	Area (of coil)
$A_{\text{wire}}$	Cross-sectional area of the wire
$r_{\text{coil}}$	is the radius of the coil
$\Phi$	resistivity of material
$N$	Number of turns in the solenoid coil
$M$	Mass of the coil
$\rho_{\text{wire}}$	density of the wire
$B$	magnetic flux density from the coil to the core
$N_d$	demagnetization factor
$\mu_o$	free space permeability
$\mu_{\text{core}}$	permeability of the core
$r$	radius of the core
$L$	length of the core
$D$	diameter of the wire
$\mathbf{R}_{bo}$	Rotation Transformation matrix from orbital frame to body frame
$\alpha$	Pitch angle
$\varphi$	Roll angle
$\gamma$	Yaw angle
$k_1; k_2; k_3$	Ratios of moments of inertia
$h$	Altitude of satellite
$M_{\text{earth}}$	Earth's magnetic moment
$D_{\text{residual}}$	Residual Dipole of the satellite

# CHAPTER 1

## INTRODUCTION

The cost of launching spacecraft into orbit has always been one of the most expensive elements of any mission. In the earliest years of spaceflight, rockets were made larger to accommodate heavier satellite payloads. This paradigm has recently shifted towards developing smaller spacecraft that possess the same functionality as their earlier predecessors, but can be flown on smaller rockets or as secondary payloads on larger launch vehicles. However, due to their size, these small satellites are limited to what missions they can accomplish. The most significant limitations are: available power and pointing performance.

Power limitations will always be a major problem for small satellites, as, typically, the only way for a satellite to generate power on orbit is through solar cells. Unless there is a major breakthrough in solar cell efficiencies or in the deployment methods of solar arrays for small satellites, the power available to these missions will stay limited. Pointing performance on the other hand is one limitation that can be easily addressed, both by improving the control algorithms and by scaling down existing actuators found on larger spacecraft. In this thesis, the issues involved in the design and development of a picosatellite were investigated. Specifically, focus was given to the development of a CubeSat that could be adapted to many different controls experiments.

### 1.1 Motivation

The overall research goal of the space systems dynamics and control group (SSDCG) at Ryerson University is to improve the control methods utilized on satellites; by both investigating novel actuators and by implementing new control algorithms on existing hardware. CubeSats provide an inexpensive way to get hardware to space. The design and development of Ryerson

University's first CubeSat, and a simplified standard architecture, will allow researchers in SSDCG to develop and test novel algorithms and actuators in the space environment with minimal development time, improving the time it takes to bring theoretical research to acceptable technology readiness levels (TRLs) for higher performance missions investigated by larger government agencies such as the Canadian Space Agency.

The reason it was decided the power subsystem should be created first was based on the fact that the power subsystem plays a vital part of the devised architecture. It should be given as much time as possible to work out any problems before a launch. The power subsystem is also an ideal choice to test the interface as it is a relatively simple system from a programming perspective, sending a limited data set to a master and responding to a very limited subset of commands. Additionally, the power subsystem is unlikely to undergo further changes at a systems level unlike other subsystems such as the attitude control system.

Since the overall mission is to test new control laws and actuators the next logical step was to begin work on the attitude control subsystem (ACS). The first focus of the research was to suitably size the reaction wheels and the magnetic torquers for RyeSat. The second focus was to experiment with the control law design.

## 1.2 Small Satellite Design: State of Art

Satellites are generally classified by mass (see Table 1); currently the smallest satellites launched have been in the pico-class<sup>1</sup>. The first of these pico-satellites flew on the OPAL and Sapphire missions. The efforts applied to developing these satellites later led to the creation of the CubeSat standard. A standard that has been developed and flight tested within the last ten years by an ever growing CubeSat community (Villa 2005). In the strictest sense CubeSats are a subclass of pico satellites nominally having the dimensions of 10x10x10cm (HutputtEanasin and Toorian 2006). The strategy for deploying these satellites in orbit involves launching a group of CubeSats in a deployment device known as a Poly Pico-satellite Orbital Deployer (P-POD). The P-POD integrates with the launch vehicle and carries three normal sized CubeSats. Using the P-POD, the first CubeSats were launched on the Eurorocket from Mirny Plestsk, Russia June 30th 2003 (Durham 2008). All of these CubeSats were built by students at different universities and of

---

<sup>1</sup> Excluding the individual dipoles released in the Westford needle project



the six satellites launched; three were successful, and one a limited success. After this initial launch, many universities from around the globe have constructed or began construction on CubeSats of their own. CubeSats can be constructed relatively inexpensively and can be launched for approximately \$40,000 US per satellite (David 2004). However, the key benefit of constructing a CubeSat over another type of pico-satellite is to eliminate the hassle of organizing and integrating it with the launch vehicle, as this problem is dealt with by flying in the P-POD. This arrangement should allow the satellite design team to focus less time on logistic issues associated with the rocket launch, and spend more time developing the satellite for the mission.

Table 1: Classification of spacecraft by mass (Kumar 2006)

Satellite Class	Mass
Large satellite	>1000kg
Medium sized satellite	500-1000kg
Mini satellite	100-500kg
Micro satellite	10-100kg
Nano satellite	1-10kg
Pico satellite	0.1-1kg
Femto satellite	<100g


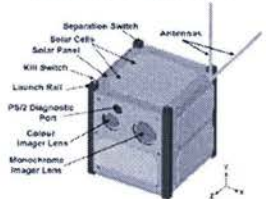

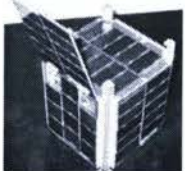


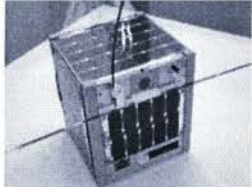
According to the CubeSat community website, there have been 16 successfully launched CubeSats (Durham 2008). The following table (Table 2) summarizes some of the earlier CubeSat configurations and missions. This table also shows even with tight volume and mass restrictions CubeSats have performed some impressive missions. Some of the more complicated missions utilized atmospheric sensors to detect carbon dioxide levels, sensitive magnetometers to sense Earthquakes, and various cameras to take images of the Earth. In addition to science based missions, CubeSats present themselves as interesting test platforms for companies planning to test technologies that require space qualification.

### 1.3 Research Objectives

The objectives of this research were to design power and attitude subsystems for RyeSat. This included designing of the system architecture, developing the power subsystem, sizing the actuators and developing attitude control algorithms.

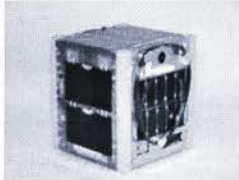

The modular system bus was designed and tested both with a breadboard and with an actual subsystem. A focus was placed on ensuring the data bus was well understood and defined, to speed up future development of other subsystems.

Table 2: Summary of missions

Satellite	Launch Vehicle	Configuration	Mission
AAU CubeSat	Eurorocket Launched June 30th 2003 from Mirny, Plestsk	 Standard CubeSat	The scientific mission of the AAU CubeSat was to take pictures of the surface of the Earth. In particular of Denmark by using the on-board camera. (Alminde, AAU CubeSat, 2003) (Wisiewski 2004)
CanX-1	Eurorocket Launched June 30th 2003 from Mirny, Plestsk	 Standard CubeSat	CanX-1 is an Engineering test bed to test technologies for Future CanX missions (Space Flight Lab 2007)
CanX-2	Falcon 1 To be launched July 2007 Omalek Island, Kwajalein	 Triple CubeSat	Designed to be an Engineering test bed for upcoming CanX-4 CanX-5 formation flying experiment. It will test various attitude control methods and will test a nano-propulsion system (Space Flight Lab 2007)
CUTE-I	Eurorocket Launched June 30th 2003 from Mirny, Plestsk	 Standard CubeSat	Designed to test a new radio protocol for amateur packet radio data. (Lab for Space Systems 2006)
Quake Sat	Eurorocket Launched June 30th 2003 from Mirny, Plestsk	 Triple CubeSat	Designed to detect ELF created by Earthquake events (Bleier, et al. 2004) (Long, et al. 2002)
KUTESat 1 Pathfinder	Dnepr 1 launch Aborted launch resulted in a Loss of the satellite	 Standard CubeSat	KUTESat had to reduce its original mission to a Radiation dosimeter experiment* (Villa, Project Management of a Student Built Space Satellite: The KUTESat- 1 2005)
Xi -IV	Eurorocket Launched June 30th 2003 from Mirny, Plestsk		Simple camera mission, carrying microfilmed messages from Earth to space. (Intelligent Space Systems Laboratory 2001) (Intelligent Space Systems Laboratory

\* Configuration after changes late in the design.



Satellite	Launch Vehicle	Configuration	Mission
		Standard CubeSat	2008)
Xi -V	Kosmos-3m 27th October 2005 Plesetsk		Simple camera mission, carrying microfilmed messages from Earth to space. Testing CIGS cell for space use. (Intelligent Space Systems Laboratory 2008)
Cape 1	To be launched on Dnepr 2 2007, Baikonur Cosmodrome Kazakhstan		An engineering test bed for University of Louisiana at Lafayette. It will primarily send housekeeping data and images back to the student run ground station. (J. Wagner 2005)
		Standard CubeSat	

The development of the power subsystem served two purposes. First, it served to develop hardware for the satellite itself, and second, it was to be utilized to further test the satellite's power and data buses. Focus was put on proving out the circuit and finding possible places for improvement with minimal changes.

The actuators were to be sufficiently sized for a single cube, CubeSat, and the research placed an emphasis on ensuring that these actuators would fit within the satellites power and volume budgets.

Research regarding control laws was to investigate the application of a sliding mode controller to control a CubeSat sized spacecraft using reaction wheels. Effort was also placed on designing a simplified controller for the magnetic torquers to de-tumble the satellite after deployment.

## 1.4 Contributions of Thesis

In this thesis the objectives were accomplished as follows:

- 1) RyeSat's system architecture was designed with a modular concept.
- 2) RyeSat's systems bus was designed and tested.
- 3) RyeSat's power subsystem was designed and tested.
- 4) Actuators were sized to fit within the CubeSat specifications and to allow full 3 axis control.

- 5) Attitude control algorithms were developed and tested through numerical simulation.

## **1.5 Thesis Organization**

This thesis is organized into five chapters.

Chapter 2 presents a review of CubeSats with a focus on subsystem design. Methods to develop smaller satellites to the standard practices used to develop larger satellites are also compared.

Chapter 3 explains the systems requirements of RyeSat. These requirements were expanded upon and related to individual subsystems.

Chapter 4 presents the systems engineering and the analysis used to develop the thermal power and attitude control subsystems.

Chapter 5 explains the detailed design including the structure and circuits of the system bus, thermal and power subsystems. This is followed by the development and simulation of attitude control laws.

Chapter 6 concludes the work presented. It summarizes the work completed on the subsystems constructed and suggests future work.

# CHAPTER 2

## LITERATURE REVIEW

Pico-satellites, like larger satellites, are designed using established engineering practices based on previous space flight missions. The basics of spacecraft design include breaking down the overall spacecraft into distinct subsystems that execute particular tasks. However, some of the traditionally segregated subsystems must be combined to save mass, space, and power. Figure 1 and Table 3 illustrate typical architectures and common requirements for smaller satellites and show some common characteristics of CubeSats that have already been built.

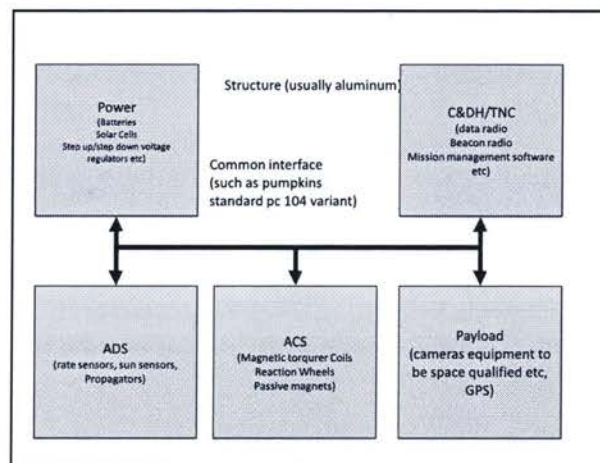


Figure 1: Typical CubeSat System Architecture

### 2.1 Structure

The purpose of a structure subsystem on a spacecraft is to maintain the spacecraft's shape while undergoing the loading conditions experienced during both launch and while in orbit. Another purpose of this subsystem is to provide a location to mount other hardware and wire harnesses. For pico-satellites, the structure subsystem represents a larger mass fraction as compared to larger satellites.



Table 3: Typical Requirements for a Pico-satellite

System	Typical requirement	Typical characteristics	Mass % Of smaller spacecraft	Mass % SMAD
Structure	Adhere to Cal Poly requirements Must be able to survive launch Must weigh as little as possible	Mostly constructed out of aluminium sometimes supported by the PCB material that the electronic packages are placed on	25%	22.7%
Attitude Orbit control system	Typically there is no orbital control requirements	Orbital control is typically not used	10%	2.7%
	There may be orbital determination requirements for imaging or attitude determination	Orbital determination is typically not used and ground stations rely on NORAD's TLEs or similar data		
	There is usually some form of attitude control requirements based on radio reception, and other payload missions, typically there are no requirements due to solar panel orientation	For active control systems magnetic torque coils are typically used. Typically passive methods are used such as gravity gradient or permanent magnetic rods	0%	11.3%
	Attitude determination will usually have some form of requirement based on the control requirements listed above; Some payloads require more accurate knowledge and use it to trigger events other than attitude control.	Attitude determination can be found by many methods see section 2.2 for details on the most common methods		
Communications	There must be communication with a ground station  Communications link must be sufficient for the mission data to be downloaded in limited ground station passes	1200bps is the most common speed 9600bps is the next most common speed Most satellites use AX.25 for transmission to ground Uplinks can be AX.25 or DTMF (touch tone phone codes)	20%	12.7%
Command and data handling	Must run mission operations independently Must be able to store mission data in between ground station passes	Can be any form of microcontroller/computer This system depends highly on payload requirements	5%	
Power	Must power the satellite Must not be dependent on the spacecraft orientation Must provide battery protection (discharging and charging)	<2W on standard CubeSat configuration Typically high efficiency Solar cells >25% Typically utilizing Lithium ion batteries	20%	24.6%
Thermal control system	Must keep components in their operating limits	Will employ heaters Specialized radiators or louvers typically not used	5%	1.7%
Payload	Vary from mission to mission		15%	24.4%

## Requirements

A CubeSat must follow strict guidelines as put forth by the Cal Poly design specification documentation in order to be launched in the P-POD deployment system (HutputtEanasin and Toorian 2006). The design specification is shown in Figure 3, and a reproduction of these requirements can be found in Appendix F.

Along with these guidelines, most CubeSats have a strict mass budget. Typically, the structure weighs approximately 200-250 grams which represents 20-25% of the standard CubeSat total mass. Although not stated in the requirements document, the structure of the CubeSat must be

designed to deal with the stresses it will experience during launch conditions, as illustrated in Figure 2.

Typically, most CubeSats strictly adhere to the Cal Poly suggestion that the frame of the CubeSat be manufactured out of aluminum 7075 or 6061 T6. An example of this traditional approach is shown in Xi- 4 and Xi-5. Only one CubeSat, CAPE-1, has not used aluminum exclusively throughout the frame. Instead, it relies on PCB material for shielding and some structural support (see Table 2).

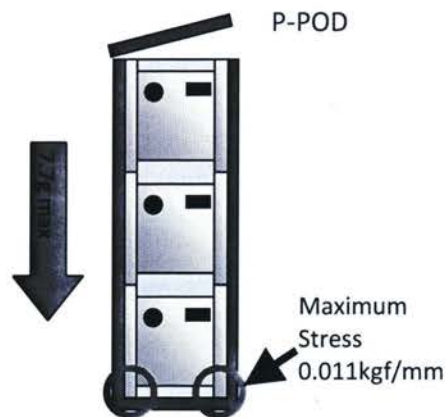


Figure 2: Location of highest stress (Intelligent Space Systems Laboratory 2001)

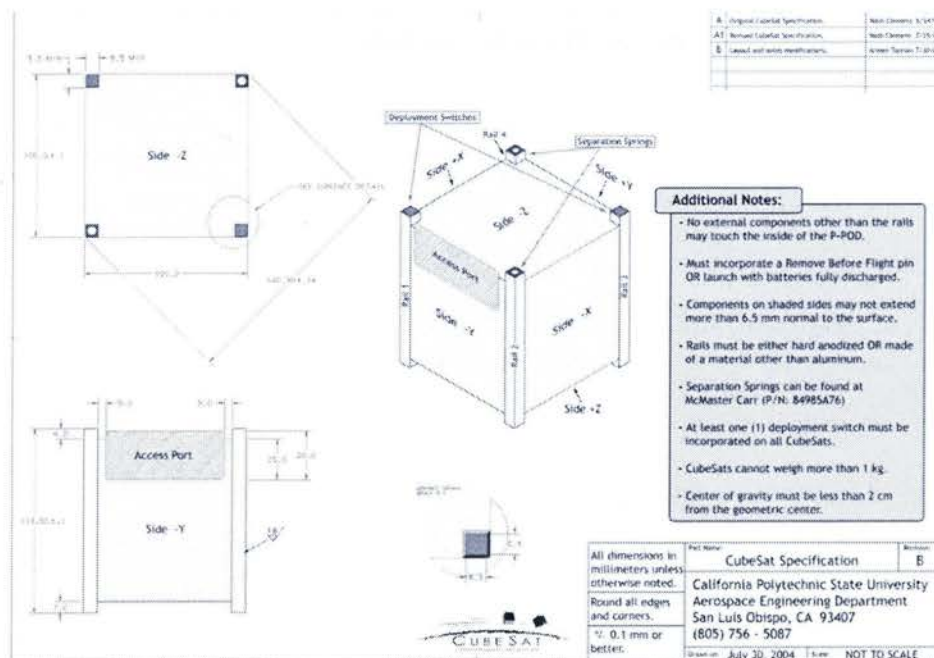


Figure 3: CubeSat Specification drawing (HutputtEanasin and Toorian 2006, 7)



## **2.2 Attitude and Orbit Determination and Control System**

The performance of the AOCS is dictated by the pointing requirements defined by the payload and communications subsystems. There are typically four components to an AOCS; orbit determination, orbit control, attitude determination, and attitude control. On some CubeSats, orbital position is determined by orbit propagation (Wisiewski 2004), while some newer CubeSat designs include low power commercial GPS receivers along with orbital propagation to obtain a better estimate (Space Flight Lab 2007). Traditionally, a CubeSat does not require orbital control because, during its short lifespan (typically 1-2 years), the satellite will not experience a large orbital drift due to environmental forces. However, the upcoming CanX-2 will fly with an experimental nano-propulsion system that could potentially change this trend (Space Flight Lab 2007). The third component of the AOCS is attitude determination. Some commonly used attitude determination sensors on CubeSats are: magnetometers, optical sensors, and gyroscopes.

Magnetometers work by determining the orientation of the Earth's magnetic field with respect to the sensor. This information along with satellite location with respect to the Earth allows the determination of one of the two required vectors to fully determine attitude. Optical sensors, such as digital sun sensors, can provide the other vector and determine the spacecraft's orientation by utilizing thin slits and linear arrays of photo sensitive circuits. When the light strikes the slit at a certain angle, certain elements of the photo array are activated. Other optical sensors such as star trackers can be used to determine the spacecraft's attitude. However commercial off the shelf components are typically too heavy and power demanding to be implemented on the typical CubeSat. The last types of common sensors used to determine a CubeSat's attitude are inertial sensors (gyroscopes). However, these devices are prone to drifting away from the true values and are only accurate for short periods of time. Hence, they are typically used in conjunction with other sensors.

As most CubeSats have differing pointing requirements, they employ vastly different methods of attitude control. The AAU CubeSat, KUTESat, and the CanX series all employed magnetic torquers to de-tumble or control the attitude of the spacecraft (Wisiewski 2004), (Villa 2005), (Wang 2004). Other CubeSats such as the Xi-series from Intelligent Space Systems Laboratory (ISSL) in Tokyo and Quakesat from Stanford have utilized passive magnetic stabilization and gravity gradient methods of attitude control (Intelligent Space Systems Laboratory 2008) (Bleier,

et al. 2004). Finally, some CubeSats such as CUTE-I have been designed without any attitude control (Lab for Space Systems 2006). A summary of AOCS's implemented on CubeSats is presented in Table 4.

## 2.3 Communications

All missions require some data to be sent back to a ground station, and, depending on the mission requirements, there are different methods of transmitting the data. For most CubeSats, there is a beacon, or a beacon mode, that transmits a CubeSat's call sign and basic housekeeping data. In addition to the beacon, there is usually a more elaborate downlink that operates as a data radio. These data radio links can be based on older modems and have speeds as low as 1200 bps or they can be newer amateur data radios that transmit at 9600 bps. The first CubeSats were operated on amateur radio frequencies, and the developers of these CubeSats published their downlink frequencies so amateur operators could listen to the CubeSats. This practice is a fairly popular means of relaying data to and from the CubeSats and can save the cost of setting up multiple ground stations. Onboard the satellite, a typical setup has a terminal node controller (TNC) connected to both an amateur handheld radio and the main onboard computer. This TNC will often handle the translation of the baseband signals (simply an audio signal) from the radio to digital signals (typically UART data streams) and vice versa. It should be noted that most CubeSats do not transmit faster than 1200bps, as most amateur radio equipment is designed for this speed, although there are a few exceptions using newer amateur radio data formats. A summary of radio configurations used on previous CubeSats is given in Table 5.

Table 4: Summary of attitude and orbit control systems

Name University	Orbit determination	Attitude sensors	Orbit control	Attitude control
AAU CubeSat University of Aalborg (Wisiewski 2004)	NORAD 2 line DATA	Magnetometer sun sensors	NONE	Magnetic torquer rods
CanX-1 University of Toronto (Wang 2004)	GPS / NORAD 2 line Data	a Honeywell three-axis digital magnetometer	None	Magnetic torquer rods
CanX-2 University of Toronto (Caillibot 2005)	GPS/ NORAD 2 line Data	Sun sensors 3 Axis magnetometers verification using cameras	a liquid-fuelled cold gas system using sulphur hexafluoride (SF <sub>6</sub> )	Orthogonal Magnetic torquer rods nano sized momentum wheel
CUTE-I Tokyo institute of technology (Lab for Space Systems 2006)	NORAD 2 line Data	Piezo-electric Vibrating Gyroscope Dual Axis Accelerometer	None	None



Name University	Orbit determination	Attitude sensors	Orbit control	Attitude control
CMOS Sun sensor				
Quake sat Stanford University (Bleier, et al. 2004)	NORAD 2 line Data	None	None	Passive control using bar magnets
KUTESat pathfinder* (Villa, Project Management of a Student Built Space Satellite: The KUTESat- 1 2005)	NORAD 2 line Data	None	None	None
Xi 4/5 (Intelligent Space Systems Laboratory 2008)	NORAD 2 line Data	None	None	Passive control using bar magnets

## 2.4 Command and Data Handling

The C&DH unit is one of the areas where CubeSats vary widely from one another. Some satellites are comprised of separate processors for each subsystem to process data created by the subsystem, while other satellites utilize a monolithic processor to handle all the tasks for every subsystem. Both methods have their merits and drawbacks. Table 6 summarizes some command and data handling configurations.

Table 5: Summary of communication systems

Name University	Uplink	Downlink	Beacon	Speed of link
AAU CubeSat (Alminde, Bisgaard, et al. 2002)	437.9 MHz Ax.25 ¼ Dipole	437.9 MHz Ax.25 ¼ Dipole	437.9 MHz CW Only when OBC is not working	9600 bps
CanX-1 (Wang 2004)	437.757 MHz MSK over FM ¼ wave dipole DTMF Fire-code to reset	437.88 MHz MSK over FM ¼ wave dipole	None	1200 bps
CanX-2 (Space Flight Lab 2007)	S-band based system 144 MHz Band Alnico DJ-C1 100mW	S-band based system Downlink 437.4000 MHz AFSK 1200 BPS Alnico DJ-C4 1110mW	None Downlink 436.8375 MHz CW	
CUTE-I (AMSAT 2006)	Acts on received DTMF commands 1/4wave dipole	Transmits using Ax.25 or SRLL ¼ wave dipole	Maki Denki Radio Sends call-sign and telemetry using CW (Morse code)	1200 bps
Quake Sat (Bleier, et al. 2004)	436.675MHz Tekk T-Net Mini 960 radio & BayPac BP-96A Modem	436.675MHz Tekk T-Net Mini 960 radio & BayPac BP-96A Modem	None	9600 bps
KUTESat (Villa, Project Management of a Student Built Space Satellite: The KUTESat- 1 2005)		437.386 MHz FSK AX.25	None	1200 bps

\* In its final launch configuration, as there was a change of mission scope during the final stages of construction

Name University	Uplink	Downlink	Beacon	Speed of link
Xi 4 /5 (Intelligent Space Systems Laboratory 2008)	145.835MHz Ax.25 Specially designed radio fire code	437.490MHz Ax.25 FSK (packet) Specially designed radio	CW (beacon) 436.8475MHz	1200 bps

Table 6: Summary of C&amp;DH systems

Name	OBC	Flight memory
AAU CubeSat	Siemens C161 micro controller	512kB of PROM For Flight application code 256kB of flash ROM for changes to code
CanX-1	The Atmel ARM7 of 40 MHz	32 MB of Flash-RAM 512 KB (triplicated for EDAC)
CanX-2	ARM7 processor running at 15 MHz	16 MB of Flash for storage of telemetry 2 MB of SRAM (triplicated for EDAC)
CUTE-I	8bit H8/300	32kbyte ROM 1kbyte RAM SRAM (4Mbit), EEPROM (256kbit)
Quakesat	PC 104 form factor computer running Linux based on ZFMicro Devices ZFx86 Processor Clock 100MHZ	16MB – 32 MB DRAM (Diamond systems Corporation 2001-2008) 2MB flash 2
KUTESat	DragonBall VZ @ 33 MHz	4 MB 8 MB RAM (Paruchuri 2006)

## 2.5 Power

Typically on a CubeSat, one cannot expect a lot of power for payloads. This is because all CubeSats rely on solar cells to generate power. The limited surface area on a CubeSat usually restricts the power output of the solar cells to under 2W. When in the absence of sunlight, stored power is used. Storage of power on CubeSats almost exclusively relies on Lithium-ion battery technology because of its high energy to weight ratio. Due to the wide spread use of Lithium ion batteries, it is not unusual to find a CubeSat that has a storage Capacity of almost 13Whrs. These limitations must be taken into consideration when designing other CubeSat subsystems. Table 7 is a summary of power systems used on CubeSats.

Table 7: Summary of power systems

Satellite	# of Solar cells	Power generated	Battery	Storage Capacity (mAh.)	Additional features
AAU CubeSat	EMCORE Triple junction GaAs cells (efficiency of 28%) 10 cells 76mm-36mm	<2W	4 Lithium-Ion polymer cells with a capacity of 940mAh each,	3680	Controls power flow to subsystems to reset in event of latch up also doubles as an emergency beacon
CanX-1	Emcore triple-junction cells (26% maximum)	1.7 W max	Polystor 3.7 V lithium-ion battery	3600	Peak power tracking Power shunting

Satellite	# of Solar cells	Power generated	Battery	Storage Capacity (mAh.)	Additional features
	efficiency). 6 solar Cells				Emergency load shedding (Stras, et al. 2003)
CanX-2	22 triple junction gallium-arsenide (GaAs) solar cells	2- 7 W	3.6 Lithium-Ion battery		
CUTE-I	Comprised of 4*6 cm cells with 17% efficiency	1.5W	4-parallel lithium ion batteries	4160	Manages batteries Monitors Temperature of power management board Deployable panel to increase surface area
Quake Sat	Multi-junction GaAs solar cells with (18% Efficiency)	5W typical 14 W Max	2 Lithium ion batteries	2800	4 double sided deployable solar panels (Bleier, et al. 2004)
KUSAT		2.2 typical			
Xi 4/5	Cell type : Si Crystal (SHARP) Efficiency : 16% 10 cells in series/panel	1.225W	Lithium Manganate 3.8 V	6240	Amount of power generated controls the beacon activation

## 2.6 Payload

The Payload subsystem varies between each individual CubeSat, as each satellite is designed for differing mission requirements. Some of the simpler payloads flown on CubeSats are imaging missions or radio experiments. More advanced payloads have included attitude control experiments, and flight testing of new space hardware. Most satellite design teams incorporate small experiments or devices to help defer the cost of launch. A good example of this is the University of Toronto's CanX-1 testing Xiphos' Q4 Card and CanX-2 carrying four different experiments on its upcoming flight. Because a CubeSat's mass, power and volume are typically limited, expression of interest by a customer in flying a payload on a CubeSat mission should begin early in the design process. This allows the satellite design team to take the payload specific mission requirements into greater consideration. Table 8 lists payloads that have flown on CubeSat missions or will be flown soon.

### Requirements

The key considerations when designing a payload experiment for a CubeSats are; mass, volume and power. Ideally, from a satellite design perspective, everything needs to be kept small, light, low powered, and be simple to interface with. These may lead to requirements for a payload similar to the following:

- A payload's mass to be less than 100 grams.



- A payload's power consumption must be under 100mA at 3.3V while in operation.
- A payload's volume should be constrained to the volume of a typical cell phone.
- Most of the payloads processing should be done on the payload subsystem.
- A payload's interface to the rest of the system must be a simple one such as UART, SPI or I<sup>2</sup>C.
- Payloads will have a limited amount of downlink bandwidth available (this could limit output to less 500 kb a pass).

Table 8: Summary of Payloads

Satellite mission& objectives	Payloads selected
AAU CubeSat	Camera module 2 Cameras
CanX-1	Xiphos Q4 card Attitude control using Magnetic torquers Atmospheric Spectrometer, a GPS Signal Occultation Experiment
CanX-2	a Atomic oxygen material degradation experiment a network communications experiment
CUTE-I	Different Radio protocols
Quake Sat	ELF magnetometer for Earthquake detection from orbit
KUTESat 1 Pathfinder	Radiation Dosimeters
Xi-IV	Camera module and Radio communications
Xi-V	Testing a new Type of Solar panel
Cape 1	Camera module

## 2.7 System Bus

CubeSats present an interesting design challenge with system bus design, as they are typically too small to allow the use of common standard spacecraft data and power buses. To deal with this limitation, many standard terrestrial interfaces & protocols are utilized.

### 2.7.1 Spacecraft Interfaces

Larger spacecraft are not without standards. However, the standards typically used are unsuitable for smaller spacecraft. To illustrate this point, a brief overview of two commonly used interfaces (MIL-STD-1553 and SpaceWire) is presented in the following subsections.

#### ***MIL-STD-1553B***

MIL-STD-1553 was first proposed in 1975 to create a reliable serial interface between subsystems on fighter aircraft. Since then, this standard has found its way into other systems such as satellites. In general, MIL-STD-1553 is an addressable serial interface, designed to transfer data at 1 Mbit/second. In high reliability applications, MIL-STD-1553 usually consists of two or more separate but identical data buses as can be seen in Figure 4 (US Department of Defense 1996).



These multiple buses are supervised by a single bus Controller, and this controller is solely in charge of the bus. However, it often is supplemented with a backup. The bus controller is responsible for directing data traffic on the data lines and has several modes of operation including communication between itself and any remote terminal, communication from the terminal to itself, directing information from one remote terminal to another remote terminal, and finally a system wide broadcast.

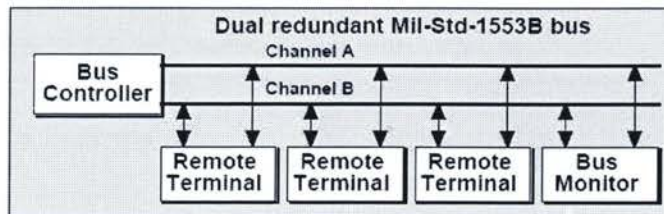


Figure 4: Diagram of a typical Mil-STD-1553 bus (Wikipedia 2007)

### ***SpaceWire***

SpaceWire is a standard developed by the European Space Agency to reduce the cost of development, improve reliability of satellites and space systems, while still meeting the processing needs of the spacecraft (ESA 2003). The physical layer of SpaceWire builds heavily on the IEEE 1355-1995 standard which is designed to be a low cost low latency serial network interface (ESA 2003). In its most general and physical terms, SpaceWire is a point to point high speed data link consisting of 8 wires and a common ground. The 8 wires are divided into 4 pairs; DataIn (+/-), Dataout (+/-), Strobeln (+/-) and Strobeout (+/-). The typical pin-out of space wire is shown in Figure 5. Each pair is used to create a low voltage differential signalling line; improving the reliability of the signal and the power consumption of the interface. Utilizing both data and strobe information lines allows for the data to be verified by relatively simple techniques (ESA 2003). With this configuration, SpaceWire has been designed with a minimum speed of 2 Mbit/s and a maximum speed of 400Mbits/s (ESA 2003).

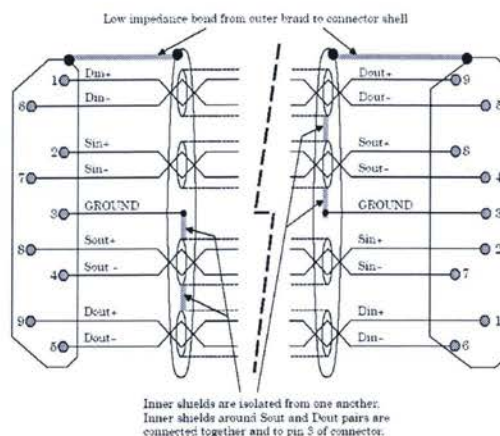


Figure 5: SpaceWire configuration (ESA 2003)

### ***Problems Using MIL-STD-1553 and SpaceWire on Small Spacecraft***

There are many problems with utilizing either SpaceWire or MIL-STD-1553B on a small satellite these include: connectors, signalling voltages, and the need for specialized transceiver devices. Typically, small spacecraft do not have the room to accommodate the standard receptacles for SpaceWire or MIL-STD-1553 (DB9 or a BNC variant, respectively). The lack of standard receptacles can be overcome through the use of more suitable connectors or printed circuit board (PCB) traces. However, this tends to defeat the purpose of using a standard. Signalling voltage can also be a problem on small spacecraft, as the available power is typically at lower voltages and limited in supply. Although this is not specifically a problem for SpaceWire, which is a low voltage differential signalling protocol, it is a significant problem for MIL-STD-1553, as its signalling voltages are well beyond the practical capability of picosatellites (peak signalling voltages are in the range of 18-27V). Finally, data transceivers for these standards are both expensive and are often too big for use on a CubeSat.

## **2.7.2 Past CubeSats and Their Bus Interfaces**

### ***KUTE SAT (SPI based)***

KUTESat was a CubeSat developed by Kansas University. Unfortunately, it was aboard the Dnepr 1 launch which failed to achieve orbit on July 26 2006. The design of KUTESat consisted of a common SPI bus and its typical configuration is shown in Figure 6. However, due to problems with the SPI bus and the payload microcontroller, a simplification of the initial plan was required. In the final design, the main communications subsystem directly measured and reported the

payload information. The changes in KUTESat's system design can be shown in Figure 7 and Figure 8.

In general, Serial Peripheral Interface (SPI) is a clocked serial interface and is capable of data speeds well over 1Mbit/s in full duplex (data in and data out on separate lines) utilizing 3 lines (plus a common data ground). Multiple devices can exist on the same SPI bus, but typically the device that will be communicating to the master must be signalled separately using a device select pin. SPI also lacks the direct ability to detect if the device it is communicating with is functioning or present, possibly leading to more complicated error checking routines.

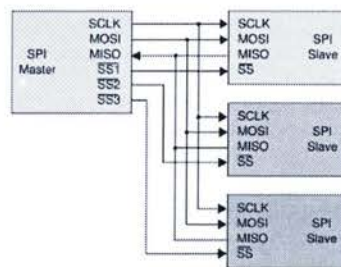


Figure 6: Example SPI interface (Wikipedia 2008)

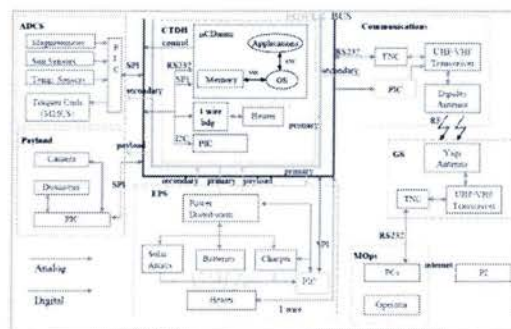


Figure 7: KUTESat system block diagram original (Paruchuri 2006)

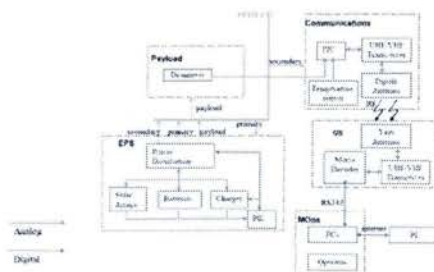


Figure 8: KUTESat final configuration (Paruchuri 2006)



### ***Xi 4 and5 (UART based)***

Xi 4 and Xi 5 were both developed by University of Tokyo. Xi 4 was successfully launched in 2003 and Xi5 was launched in 2005 (Krebs 2007). Both satellites conducted radio experiments to test a new amateur data packet method. Xi 4 and 5 both utilized separate UART based communication protocols between the main onboard computer and separate subsystem components (Intelligent Space Systems Laboratory 2001). The system design of these satellites is similar to the earlier engineering prototype Xi-3 and its configuration is shown in Figure 9.

Universal Asynchronous Receiver Transmitter (UART) is the hardware used to decode serial data transmitted across a serial Tx and Rx line. However, UART has become known as the commonly accepted term for asynchronous serial data transfer containing a 2 line plus a common data ground. Although there are no official requirements for speeds between two devices, as UART is designed to operate at any arbitrary clock speed between devices as long as it is the same on each, there are some commonly accepted speeds for which equipment is optimized for, such as 4800 bits/s, 9600 bits/s, 14.4Kbits/s up to 2.764800 Mbit/s.

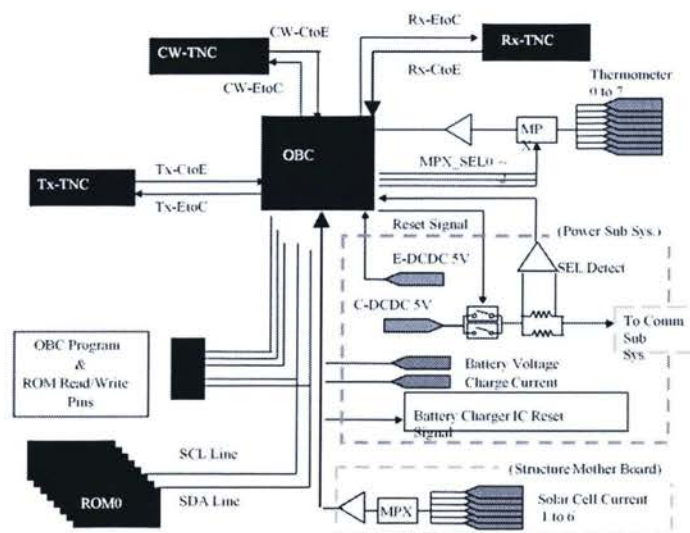


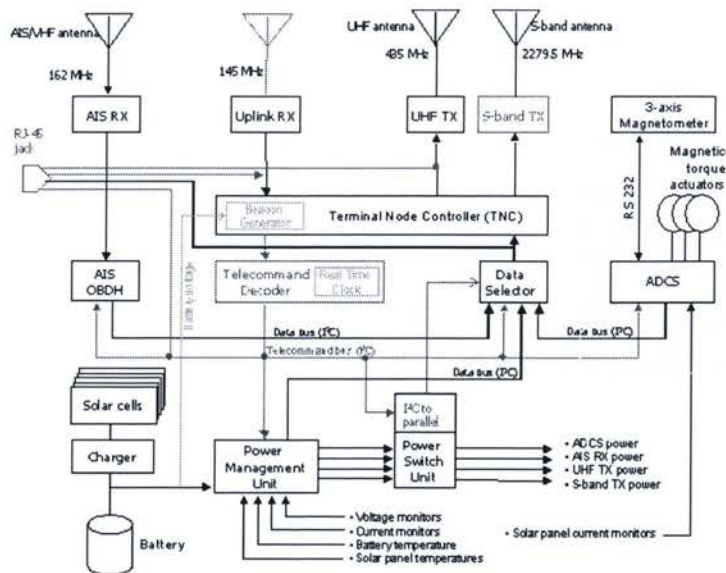
Figure 9: Xi 3 system block diagram (Intelligent Space Systems Laboratory 2001)

### ***NCUBE (I<sup>2</sup>C based)***

NCube was a CubeSat developed by the Norway University of Science and Technology (Øverby 2004) which was unfortunately among the CubeSats that were lost in the first Dnepr launch (Durham 2008). NCube's system architecture utilized two separate I<sup>2</sup>C buses. One was subsystem



commands and the other dedicated to transferring data to the radios Terminal Node Controller (TNC) (Øverby 2004). NCube's system design can be seen in Figure 10.



**Figure 10: nCube System block diagram (Øverby 2004)**

## CubeSat Kits

CubeSat Kits are being offered commercially by Pumpkin Inc. The kits offer a structure and an optional onboard computer and power subsystem. The kit is based around a stackable bus and individual cards are stacked to form an entire satellite. This kit also offers most commercial interfaces including SPI, UART, and I<sup>2</sup>C allowing designers to develop an entire subsystem based on the single low power processor. The Flight modules system architecture is found in Figure 11.

### Commonality between designs

Most of the CubeSats presented here utilize low power computing devices and none have had a radio transmitter faster than 9600 bits/s. Additionally, most of the designs presented here utilized a shared bus (I<sup>2</sup>C or SPI) to reduce the number of conductors in the spacecraft. It is also interesting to note most connections to the downlink or TNC were through a lower speed UART.

## 2.8 Fabrication for a Space Environment

### 2.8.1 Environment

## Radiation

21

environment, which causes two main modes of failure for microelectronics: single event effects, and long term dose effects.

Single event effects happen when high energy particles strike a transistor within an electronic devices core, and change the transistors material properties either temporarily or permanently depending on the strike. These radiation effects are at a transistor level, what happens to the system depends greatly on the particular transistor hit. Typically the least damaging cases would be a single event upset (SEU) where the data stored or being processed on the transistor is bit-flipped (a binary 1 is changed to a 0 or vice versa). The next damaging scenario happens when a transistor carrying important information on the chip (like the exit condition for a loop), is struck by radiation. This causes the device to lock up in an inescapable loop and this event is classified as a single event latch-up (SEL). Typically this problem can be corrected by turning the device on and off. One of the worst single event effects happens when the transistor controls a large flow of current as in the case of a power MOSFET. This damage even if temporary could cause other components to burnout, destroying the device. This effect is known as a single event burnout (SEB) (Larson and Wertz 2005, 220). Fortunately there are several methods to overcome SEU's, SEL's and even some SEB's. These include error detection and correction methods (EDAC) in hardware and software, core shielding, selection of device's substrate material, and changes to the device's core design for radiation hardening.

Long term dose effects can be equally devastating. They can be modeled as a gradual addition of electron holes to the transistors gate or base (depending on variety of transistor). This effect then changes the threshold voltage of the gate or base. Fortunately Galysh et al. (2000) have done testing and found most typical devices can handle up to 20 Krads which means most devices in low earth orbit can safely run for 6 months to a year in low earth orbit without additional measures or shielding.

### ***Thermal Management***

Thermal management is a huge issue in spacecraft design. Although most electronics function best in cold conditions, a spacecraft can easily experience temperature swings from -35°C to 60°C. One key problem for pico-satellite developers is that most components used in small spacecraft design were intended to be at least partially cooled by convective air currents. Another problem with consumer COTS parts is they are not intended to be thermally cycled as



often and as quickly as a device in the space environment. However, these problems can be overcome with the use of heaters and smart PCB design. Most electronic devices utilize ground pins or planes to eliminate their waste heat, providing a direct thermal path for heating and cooling of individual devices. These pins allow the PCB designer to link heat sensitive parts to thermal management systems. In the realm of spacecraft design these thermal systems can be as simple as parts of the spacecrafts structure acting as thermal capacitors or radiators.

### ***Micrometeorites/ Space debris***

Satellites in low earth orbit (LEO) typically travel at around 7 km/s. A collision with even a small screw in a complementary elliptical orbit may cause great amounts of damage. However, without knowing the location of every piece of debris in space (as most are too small to track), only estimates can be provided on which orbits may experience the greatest risks of collision. Typically, when this issue is analyzed, the risk assessment estimates are determined from formulas based on the exposed cross-sectional area. Fortunately because of a CubeSats size, the risk of a collision is greatly reduced. However, CubeSats themselves represent a potential hazard to many other larger spacecraft. To minimize this risk, it is necessary that CubeSat launchers take great care in ensuring their payloads reach to their published and desired orbits. It is also important for CubeSat developers to design the spacecraft with the fewest separable parts in order to reduce the chance of adding to the space debris problem.

### ***Out-Gassing***

Out-gassing is the escaping of minute amounts of solvents or other gases trapped in components used on a spacecraft. If there is a larger amount of solvents or composites used on a spacecraft, the torques created by out-gassing could result in an attitude control problem. Even if there is not a large force created by the out-gassing there is the potential for the solvent or other material to affect solar cells or optical equipment on the satellite or another satellite on the same launch vehicle. Fortunately most suppliers of adhesives and solvents offer low out-gassing varieties of their products and these should be used whenever possible to avoid this problem. Another way to reduce out-gassing is to place the satellite in a vacuum chamber on the ground for an extended period of time cycling between high and low temperatures. This procedure known as a bake-out will allow the gasses in the adhesives and materials to bubble out while the spacecraft is still accessible for cleanup.



### **2.8.2 Space Manufacturing**

As it has been stated earlier there is a special methodology surrounding spacecraft manufacturing. This methodology typically mandates that all components used on a spacecraft be from specialized suppliers and be accompanied by detailed documentation of their production, testing, and storage. Regardless of the methods used by the CubeSat developer there is typically a need to adhere to of the rules regarding the documentation and manufacturing of spacecraft to be launched as a secondary payload.

#### ***Traditional Approach***

Traditionally, manufacturing for a space environment required a different approach to manufacturing. First, most early spacecraft were manufactured using prototype shop techniques, with expert technicians in clean rooms assembling the spacecraft, using only space qualified parts. This traditional approach was born from the early space missions like Pioneer, Apollo and Voyager, and at the time most failures could be attributed to faulty components. From the problems experienced in earlier missions, a focus was placed on improving the reliability of the mission by improving the quality of the individual parts (Fleeter 2000). Despite the improvement of commercial processes, the space industry is reluctant to abandon space qualified parts and the sometimes error prone hand manufacturing and testing procedures associated with them. Whether or not this is the correct way to improve the reliability of the spacecraft today, is not an issue, because in the end, it is typically a large space agency that will review the Pico-satellite before it is placed on the launch vehicle. In the end this will require the Pico satellite to be built in clean room conditions, or to be cleaned thoroughly prior to launch.

#### ***Typical CubeSat Approach***

Because CubeSats typically don't have the luxury of expert technicians, perfect clean rooms and expensive space qualified parts, CubeSat developers have to look at other ways to improve reliability of the spacecraft. Fortunately for CubeSat developers consumer COTS parts have drastically improved in reliability since the 1960's, because of improvements in both the manufacturing process and integration of more external components into the device. Although there is a risk associated with building more devices into a single package, automation in the manufacturing and testing processes has improved these devices' overall reliability and defects can be easily tested for. With these improvements to consumer COTS parts, the time and money previously spent tracking individual space qualified components can now be better spent on

improving the design and speeding up the development schedule (Fleeter 2000). An additional benefit of using consumer COTS parts is that these devices typically contain more of the supporting circuitry inside. Hence there are fewer connections for the designer/assembler to get wrong and less components to potentially fail when in orbit. Spacecraft reliability cannot be gauged directly from the choice of space qualified or consumer COTS parts, and in no way is it possible to tell which choice is better without doing a failure modes analysis study. However, consumer COTS parts do have the advantage of simplifying that analysis.

## **2.9 Discussions on CubeSats Performance**

Missions involving CubeSats have not been highly successful for several reasons. The author believes one of the primary reasons may be due to lack of experience of a CubeSat design team begins with. However, some common mistakes listed here can be avoided, and some particular missions can be used as a guide for an inexperienced satellite design team.

### **2.9.1 CubeSat Failures and Problems**

On a CubeSat mission, many problems could happen. The most common serious problems include launch vehicle failures, communication failures, power failures, and electronic interface issues.

#### ***Launch Vehicle Failures***

This problem has claimed the most CubeSats to date. The launch failure of Dnepr 1 caused 14 teams to lose their satellites including KUTESat and NCube (Durham 2008). Even if all types of precautions are taken to avoid this problem, there is always a risk of launch failure. One method to mitigate this risk is for a team to build two identical satellites, although this approach may seem expensive. A team should be in the practice of ordering operational spares for the inevitable damaging of a component that happens during the development and integration phase. By utilizing spares and purchasing additional components as needed, the team could easily build an identical satellite in the period of time it takes to organize another launch.

#### ***Communications Failures***

This problem was experienced by CanX-1. The satellite was launched and deployed but the communication between the ground station and the spacecraft failed (AMSAT 2006). There are a few methods to mitigate this problem. The first approach is to have a large margin on the link



budget to allow for an improperly aligned or stuck antenna. Also, the frequencies of the satellite can be widely published so amateur radio operators with better equipment can listen for the satellite. Another good approach is to implement an emergency beacon that will transmit in the event that other systems fail.

### ***Power Failures***

After Quakesat had been in orbit for 6 months it experienced the loss of both of its main batteries. This problem led to the frequent rebooting of the satellite. The loss of the batteries was perhaps due to the electrolyte drying out in the spacecraft's battery pack (Bleier, et al. 2004). The Quakesat team reasoned that the power failures could have been avoided if the battery packs were sealed better or protected from high temperatures using insulation (Bleier, et al. 2004). Another approach to mitigate this problem would have been to place redundant battery packs in different locations to avoid all of the battery packs being damaged the same way at the same time. Although given the size of CubeSats in general this may not be as beneficial as the other suggestions provided by The Quakesat team.

### ***Electronic Interface Issues***

This problem was encountered by the KUTESat team late in the design process (Villa 2005). In this particular case, part of the problem was getting two different microcontrollers to communicate over a standard but more complicated interface. The other part of the problem was not realizing the limitation of one of the microcontrollers. The easiest method to avoid this problem is to use extremely simple interfaces such as UART, or TTL (Transistor-Transistor Logic). Checking data sheets of the candidate devices with specific attention given to the interfaces and features most likely to be used will also help mitigate this problem. A more reliable approach would be to test this type of communications link early on a bread board, with examples of the features to be used on the microcontrollers being implemented.

## **2.9.2 Checklists and Documentation**

One of the most over looked aspects of pico-satellite design is the preparation of full documentation. Some of the reasons for poor documentation are rushed schedules, inexperienced teams and a lack of industry experience. The author has found that developing documentation and more importantly, easy to reference checklists ensures that no major steps are missed in operation. These checklists also remove guess work when troubleshooting

problems. Checklists should be used regularly and should act as a program for the team to work through without any deviation. It should be noted that even limited checklists are invaluable to any satellite operations team. Documentation should be created to supplement any checklist and should serve to explain why routines in a checklist are followed. This said, writing checklists and documentation for every contingency is almost impossible, but even the act of writing some checklists allows for a greater understanding of the system on a whole, and serves to ensure the development team fully understands the system.

### **2.9.3 CubeSat Successes**

Some of the most successful missions utilized the simplest approaches to solve their problems. In addition, they considered simple mission requirements. The more notably successful missions are CUTE-I, Xi-4, and Xi-5.

#### ***CUTE-I***

CUTE-I has been operational ever since its launch in 2003 (AMSAT 2006). Its simple radio link layer (SRLL) radio experiment has been very successful and many amateur radio operators have received its signal (including the author). The ground station communicates with the satellite via DTMF codes, while the satellite transmits data back to the ground station using multiple redundant radios. The CUTE-I's success is attributed largely to its overall simplicity in its solutions for its mission goals.

#### ***Xi-4 and Xi-5***

Both of these satellites are highly successful missions returning images of the Earth. In the authors' opinion, based on all the CubeSats reviewed, these satellites are the most successful. This success is largely in part due to the excellent design process applied by the ISSL development team, building three consecutive prototypes before developing the final space ready hardware. Each prototype provided the ISSL design team insight on the overall design and offered a chance for the team to make changes to both the hardware and the software design. Xi-4 and Xi-5 also didn't require strict pointing budgets and relied on passive stabilization.

These successes show that CubeSats can be an extremely cost effective way of testing hardware for space qualification. Also it should be evident that most issues that can adversely affect a CubeSat can be dealt with in the design stages of a mission. Despite the possibility of failure,



development of a CubeSat is a great opportunity to allow small companies and universities to venture into the realm of spacecraft design.

# CHAPTER 3

## SYSTEM REQUIREMENTS

When first proposed, it was decided that RyeSat would be a standard single cube CubeSat, weigh no more than one kilogram and not exceed the dimensions set out by California Polytechnic Institute and Stanford University. It was also decided that this satellite would be used as a learning tool for the satellite design group at Ryerson University. This satellite would explore modular satellite construction techniques and would carry a camera and an additional payload to an orbit of opportunity not exceeding 650 km. Furthermore, RyeSat would experiment with MEMS-based attitude sensors for future use on small satellite missions.

### 3.1 Overall mission

In addition to the original mission objectives, RyeSat will be constrained to the following requirements proposed in the initial proposal and the lab development environment. These exact requirements are found in Table 9.

Table 9: Overall system requirements

#	Requirement	Origin	Verification method
O1	RyeSat shall test and Demonstrate MEMS based Rate sensors in the space environment.	Proposal	Design
O2	RyeSat shall take images of the Earth and moon.	Proposal	Design
O3	RyeSat will conform to the standard size CubeSat standards (V9.0).	Proposal	Design
O4	RyeSat will document all of the design process for future reference.	In house Requirement	Design
O5	RyeSat will utilize as many COTS (commercial off the shelf) parts as possible.	Proposal	Design
O6	RyeSat will be programmed in a higher level programming language such as C	In house Requirement	Design
O7	RyeSat will utilize the same design process as Ryerson's 2005, 2006 CanSat project.	In house Requirement	Design

#	Requirement	Origin	Verification method
O8	RyeSat shall be designed to survive the mission life of 6 months to a year	Proposal	Design
O9	RyeSat shall be designed to send mission health and payload data back to Earth via an amateur radio link.	Proposal	Design
O10	The satellite shall be as simple in design as possible from a systems level perspective.	Proposal	Design
O11	RyeSat shall store important payload data until it can relay the data to a ground station.	O9	Design
O12	The team will obtain appropriate licenses for the frequencies used.	O3 CubeSat specifications V9.0	Design
O13	The team will obtain the appropriate orbital debris mitigation documentation.	O3 CubeSat specifications V9.0	Design
O14	RyeSat shall use modular design techniques	Proposal	

### 3.2 Mass and Power Budgets

To also give some guidelines in the design of other subsystems the following mass and power budget was created (Table 10).

Table 10: Design budget

System	mass(g)			power (mW)					
		Voltage (V)	Draw (mA)	all	normal	Normal +transmit	Normal +ADS	Normal +ACS	Normal payload
Structure	400	0	0	0	0	0	0	0	0
Power	100	3.3	50	165	165	165	165	165	165
ADS	50	3.3	300	990	0	0	990	0	0
ACS	100	3.3	800	2640	0	0	0	2640	0
Payload	50	3.3	300	990	0	0	0	0	990
C&DH	100	3.3	300	990	990	990	990	990	990
COMM	100	6	280	1400	0	1680	0	0	0
Thermal	50	3.3	700	2300	0	0	0	0	0
Connectors	50	3.3	~2	~5	~5	~5	~5	~5	~5
Total	1000			9480	1160	2840	2150	3800	2150

\*note structure contains the mass of circuit boards

#### 3.2.1 System bus

It was decided that requirements for each subsystem should be developed next and the requirements for the way these systems interact should be described first as these choices would greatly affect the subsystem design. Table 11 is a list of requirements considered for the design of the system bus. The end result of these requirements lead to the choice to use I<sup>2</sup>C.

Table 11: System bus requirements

#	Requirement	Origin	Verification method
SB1	The system bus standard must accommodate modular subsystem design	O14	Design
SB2	The standard must allow a means of isolating malfunctioning subsystems	O9, O8, O14	Design Testing
SB3	The standard must be suitable for a CubeSat and smaller spacecraft	O14	Design
SB4	The data transfer standard must be a lower signalling voltage standard	O3, O4	Design
SB5	The standard must utilize physically smaller connectors or be independent of connector design.	O11	Design
SB6	The standard should be low cost	O5	Design Analysis
SB7	The standard shall not require any additional or highly specialized equipment.	O10	Design
SB8	The standard shall have a terrestrial industry equivalent (to allow for the use of lower cost parts).	O14	Design Analysis
SB9	The standard must be reliable and simple	O10	Design Testing
SB10	The standard should utilize the fewest interconnects possible	O10	Design
SB11	The standard should not require substantial amounts of code to operate	O6, O10, O14	Design Testing
SB12	The standard should be able to automatically detect errors and attempt to correct them.	O8	Design Testing
SB13	The standard must have methods to recover from data errors.	O8	Design Testing
SB14	The standard must be able to remain reliable for prolonged periods of continuous use.	O8, O10, O14	Design Testing
SB15	The standard must show some immunity to magnetic fields.	O8	Design
SB16	The standard must show some immunity to temperature changes.	O8	Design
SB17	The standard should allow for faster data rates in the future.	O14	Design

### 3.2.2 Environmental/ thermal

The environmental requirements were derived from initial analysis of the space environment and the design lifespan specified in the overall requirements. Table 12 highlights these requirements.

Table 12: Environmental requirements

#	Requirement	Origin	Verification method
E1	The thermal subsystem shall only use passive means of heat dissipation.	O8, O14	Design



#	Requirement	Origin	Verification method
E2	RyeSat's thermal system will keep the all components in their normal operating temperatures which will be assumed to be -40°C to 80°C unless tighter tolerances are specified within a particular subsystem.	O8, O14	Analysis Testing
E3	Thermal management subsystem shall be autonomous.	O8, O14	Design
E4	Thermal management systems will be placed on every subsystem and they shall as similar in design as possible.	O14	Design

### 3.2.3 Power

The power requirements were derived mainly from the overall mission lifespan and our system bus choices. Table 13 lists these requirements.

Table 13: Power requirements

#	Requirement	Origin	Verification method
P1	The power system should provide adequate power to perform operations while in orbit.	O8	Analysis Design
P2	The power system will be rechargeable.	O10	Design
P3	The power system will be fault tolerant.	O8	Design Testing
P4	The power system will be capable of reporting its condition.	O10	Design
P5	The power system shall be designed to survive the vacuum of space.	O8	Design Analysis Test
P6	The batteries selected for the final design shall be selected to survive the vacuum of space	O8	Design Test
P7	The system must supply a 3.3V uninterruptible line for the C&DH module.	O8,O14	Design
P8	The system will supply a 6- 5 V uninterruptible line for the radio.	O8,O14	Design
P9	The system shall have controllable lines for non essential systems.	O8,O14	Design
P10	The system will supply a switchable 3.3V 250mA line for the ADS.	O8,O14, Design budget	Design
P11	The system will supply a switchable 3.3V 750mA line for the ACS.	O8,O14, Design budget	Design
P12	The system will supply a switchable 3.3V 250mA line for the payload.	O8,O14, Design budget	Design
P13	The system will supply a switchable 3.3V 800mA line for the thermal management system.	O8,O14, Design budget	Design
P14	The system shall only use the top surface of the board (heater and its control circuit will be placed on the bottom).	E4	Design
P15	The system shall perform a check before enabling itself on the system bus.	SB2	Design Test

#	Requirement	Origin	Verification method
P16	The solar cells shall be used as a rough means of attitude estimation.	A2	Design Analysis
P17	The power subsystem shall be designed to survive be functional between -40° to 85°C	E2	Design Test

### 3.2.4 Attitude Determination and Control

Attitude pointing requirements came from both the initial proposal and the payload. The decision to split the ADCS up into two separate subsystems was done to simplify the work breakdown of the team. Table 14 lists these requirements

Table 14: ADCS requirements

#	Requirement	Origin	Verification method
A1	The ADCS will be split into two parts	O10, O14	Design
A2	RyeSat shall be designed to be stabilized on all 3 axes.	O2,	Design
AD1	The ADS shall be designed to allow for integration of a GPS device.	O14, proposal	Design
AD2	The ADS shall determine and record its orientation.	O1	Design
AD3	The ADS will calibrate sensors in software	O1	Design Test
AD4	The ADS will be functional between -40° and 85°C	E2	Analyze Test
AD5	The ACS will perform a check before it enables its self on the system bus.	SB2	Design Test
AD6	The ACS shall only utilize only the top surface of the board	E4	Design
AC1	The ACS shall control its attitude using magnetic torque rods.	A2	Analysis
AC2	The ACS shall have provisions for a 3 reaction wheels. <sup>2</sup>		Design
AC3	The ACS will be effective between 400-1000 km orbits	O2	Analyze
AC4	The ACS shall have modes that automatically de-tumble the spacecraft	O1,O2,O10	Design Analyze
AC5	The ACS will allow for the use of different control algorithms <sup>2</sup>		
AC6	The ACS will perform a check before it enables its self on the system bus.	SB2	Design Test
AC7	The ACS shall only utilize only the top surface of the board	E4	Design
AC8	The ADS will be functional between -40° and 85°C	E2	Design Test

<sup>2</sup> Late changes to requirements



### 3.2.5 C&DH/Comm Subsystem

The requirements for the C&DH came from a literature review of existing CubeSats and the design life time and the system bus choices. Some of the communications requirements were outlined from the CubeSat design specifications, while others were derived from the initial proposal and its requirements. Table 15 shows lists these requirements

Table 15: C&DH requirements

#	Requirement	Origin	Verification method
OBC1	The onboard computer shall be fault tolerant.	O8	Design Test
OBC2	The onboard computer shall be reprogrammable in orbit.	O8, O14, O11,O2	Design
OBC3	The onboard computer shall be programmable in C.	O6,O14	Design
OBC4	The onboard computer shall be radiation tolerant for the mission life.	O8	Design Test
OBC5	The onboard computer will be supervised by a watchdog or another more reliable but simpler microcontroller.	O8 OBC1	Design
OBC6	The onboard computer Shall have a "fire code" reset option.	O8 OBC1	Design
OBC7	The OBC shall be designed to monitor its system health and report failures and errors as the mission proceeds.	O9,O11	Design
OBC8	The OBC shall intelligently decide if it is safe to perform a requested mission task (i.e., check battery levels; ensure devices are deployed ... etc.)	OBC1 O8	Design
OBC9	All subsystems shall interface with the OBC directly using a simple protocol	SB1, O14	Design
OBC10	The radio shall be capable of transmitting the payload's data back to Earth.	O9	Analysis
OBC11	The radio shall be designed to have the capability to be turned off by remote command.	O3, CubeSat specifications V9.0 (as per FCC Rules)	Design
OBC12	The radio shall not transmit until 15 minutes after deployment.	O3, CubeSat specifications V9.0	Design
OBC13	The radio shall transmit on its lowest power settings 15 minutes to 30 minutes after deployment.	O3,CubeSat specifications V9.0	Design
OBC14	The radio shall begin transmitting at its highest power setting 30 minutes after deployment.	O3,CubeSat specifications V9.0	Design
OBC15	There shall be a "fire code" reset signal that the communications system will forward to reset the OBC.	O8	Design Test
OBC16	The system must operate on a 3.3V uninterruptible line for the processors.	P7	Design
OBC17	The system must operate on a 5 V uninterruptible line for the radio.	P8	Design
OBC18	The system will have two digital modems, one DTMF and one Bell 202.	O8, O9, O11	Design



#	Requirement	Origin	Verification method
OBC19	The system will have EEPROM space onboard for local storage.	O11	Design
OBC20	The system will monitor amateur radio via bell 202 modem chip	O9, O8	Design Test
OBC21	Both processors will enable the WDT timers and service them accordingly.	O8	Design Test
OBC22	The systems backup processor will monitor a fire-code detector.	O8,O9	Design Test
OBC23	The systems backup processor will reset the entire satellite should it detect aberrant behaviour.	O8	Design Test
OBC24	The backup processor will be subject to a mandatory reset every 60s.	O8	Design
OBC25	The system will be the only source of I <sup>2</sup> C bus pull-up resistors.	O8, SB specifications	Design
OBC26	The system will be the only source of the MCLR (master reset) pull-up resistor.	O8	Design
OBC27	All mission data will be stored (mission, progress, payload data) in non-volatile memory at regular (short) intervals and only erased when confirmed archived by ground station.	O9, O11	Design
OBC28	All data packets communicated between processors will be verified by an embedded checksum.	SB specifications	Design Test
OBC29	The Payload will perform a check before it enables its self on the system bus.	SB2	Design Test
OBC30	The Payload shall only utilize only the top surface of the board	E4	Design Test
OBC31	The Payload will be functional between -40° and 85°C	E2	Design Test

### 3.2.6 Payload

The requirements for the payload came directly from the decisions made regarding the OBC and the initial proposal.

Table 16: Payload requirements

#	Requirement	Origin	Verification method
PL1	RyeSat shall take images of the Earth and moon.	O2	Design
PL2	The images shall have a resolution of 5-10 km per pixel when viewing the Earth.	P1_1	Analysis
PL3	The images shall have a resolution of 50-100 km per pixel when viewing the moon.	P1_1	Analysis
PL4	Payloads will utilize an I <sup>2</sup> C communication interface to the onboard computer.	SB specifications	Design
PL5	The Payload will perform a check before it enables its self on the system bus.	SB2	Design Test
PL6	The Payload shall only utilize only the top surface of the board	E4	Design Test
PL7	The Payload will be functional between -40° and 85°C	E2	Design Test



# CHAPTER 4

## SYSTEM ANALYSIS AND DESIGN

RyeSat is comprised of 5 subsystems: C&DH, Power, ADS, ACS, payload (Figure 12). These subsystems are connected using a standard interface based on the I<sup>2</sup>C bus. The satellite's system block diagram is shown in Figure 13: System block diagram (hardware specific)Figure 13

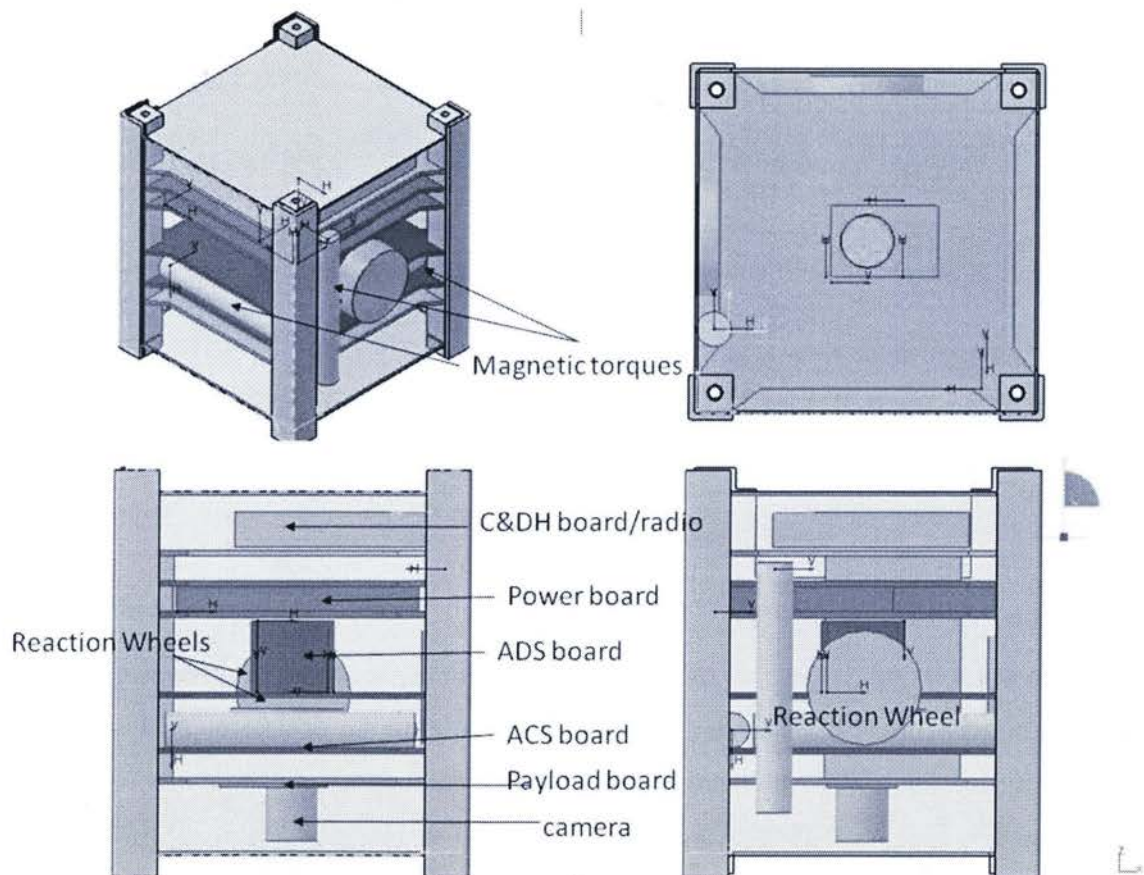


Figure 12: Satellite configuration drawing



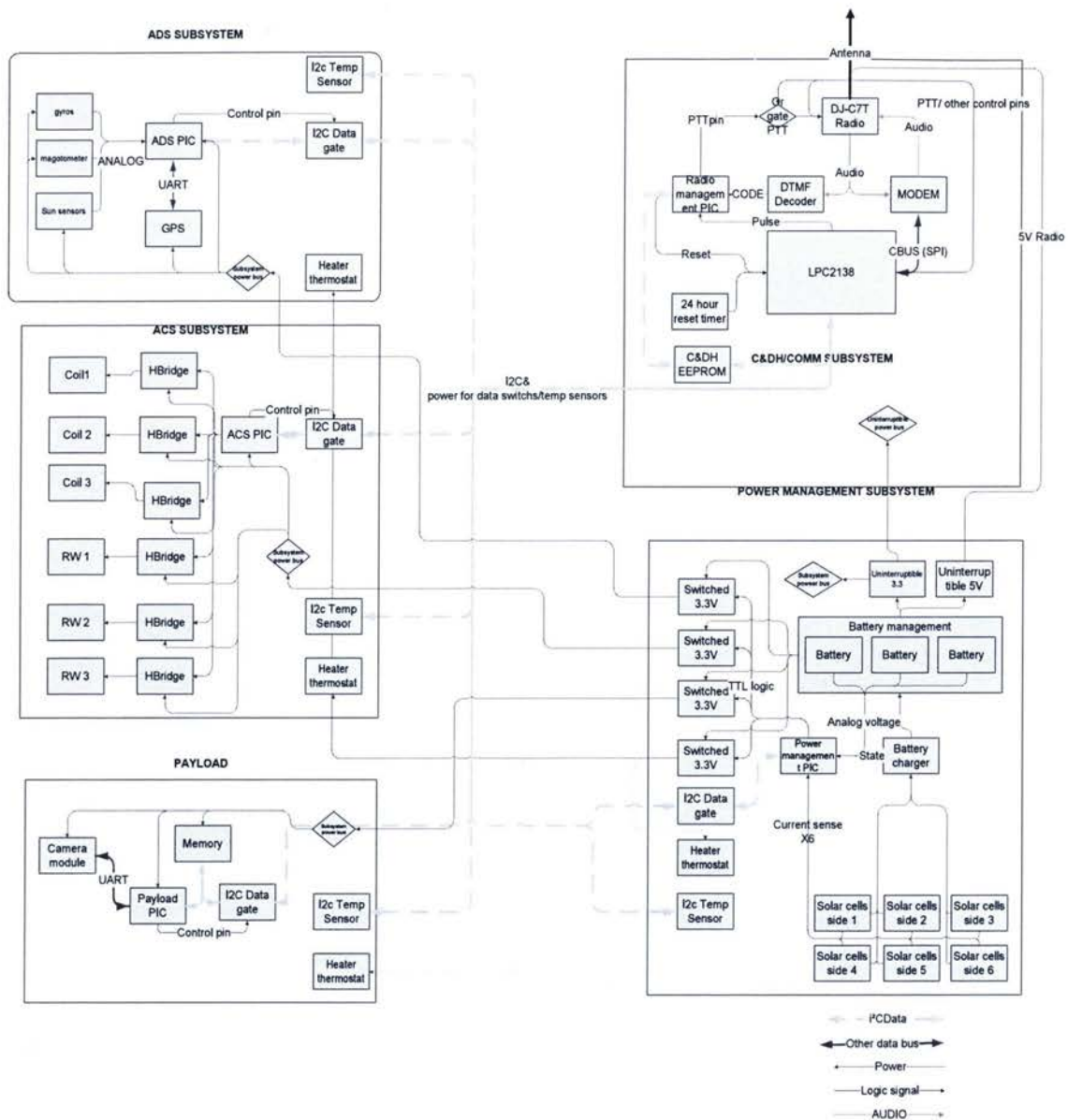


Figure 13: System block diagram (hardware specific)

## 4.1 Architecture of RyeSat

One of the key tasks for RyeSat was to define a method for communication between each of the individual subsystems. It was decided that the interface must be flexible to deal with future design considerations, be proven reliable and be standard in industry. Furthermore, it was noted early in the design phase that there was little chance that this mission or future missions would need an internal data transfer rate greater than 100,000 bit/sec, as the available radios designed for smaller satellites are typically limited to 9600 bits/sec. It was also noted that there are only 5

subsystems on the spacecraft capable of creating data. These considerations led to the adoption of the I<sup>2</sup>C based standard.

The standard chosen for the RyeSat specifies both a data bus and a power control bus (Figure 14). This type of standard is useful for isolating each subsystem from the rest of the spacecraft. This standard also has the potential to produce individual subsystems that can later be marketed to other interested companies as proven hardware. The spacecraft bus has a 3.3V uninterruptible power source for mission critical subsystems, a 5 volt uninterruptible source for the radio, 4 separate controllable power sources for each of the subsystems, and a 2 line data bus used to communicate between the subsystems. Figure 15 illustrates the full data interface (Alger, et al. 2007). The data bus is based on the I<sup>2</sup>C standard which is already designed to handle multiple masters, corrupted commands, and slight variations in clock speeds. The I<sup>2</sup>C standards also offer standard data transfer rates of 100 Kbits/s, 400 Kbits/s, and 3.4Mbits/s. To improve the reliability on the first RyeSat, the I<sup>2</sup>C bus clock speed is limited to 100Kbits/s and a data switch is placed between each subsystem and the main data bus. The data switch is designed to remain closed until the subsystem performs its own start-up checks and deems itself operational and able to communicate with the rest of the satellite. However, such a check will only ensure that the subsystems firmware is still functioning properly and will not act to disconnect the subsystem in the event of its own single event latch-up. In future revisions, a windowed watchdog timer ("dead-man" style switch) will be developed and the subsystem will be expected to continually send a varying pulse to ensure it is working correctly, if the pulse stops (or is irregular) the switch will disconnect the subsystem from the data bus. Furthermore, if there happens to be a subsystem that is constantly causing problems, the power subsystem will simply turn off the faulty subsystem.



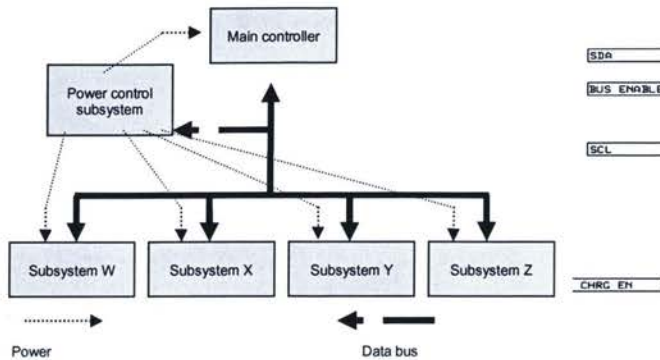


Figure 14: Systems block representation of the proposed standard.

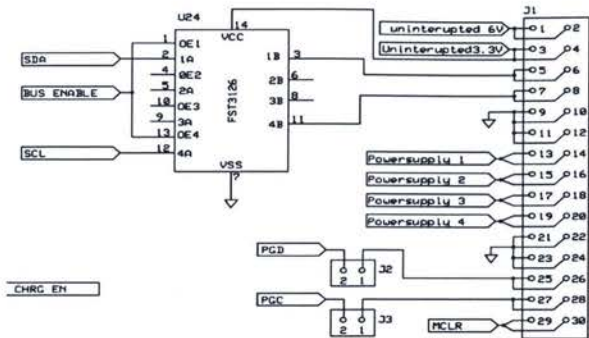


Figure 15: Electrical Schematic of Interface used on RyeSat.

#### 4.1.1 Advantages of an I<sup>2</sup>C Data Bus

Phillips originally developed the I<sup>2</sup>C standard in the late 1980's to simplify the number of interconnects between devices in consumer electronics such as televisions and VCRs (Phillips NXP Semiconductor 2000). Since then, I<sup>2</sup>C has become a widely accepted standard for lower speed (100Kbits/s-400Kbits/s) devices such as analog to digital converters, EEPROMS and sensors. The key advantage of I<sup>2</sup>C is its software addressable interface. This style of protocol allows for a drastic simplification of the hardware used to connect devices in spacecraft and allows for a simple and elegant system design. Another benefit to I<sup>2</sup>C is the possibility of adding completely different subsystems to a known working system without any modification to the hardware of the existing system, allowing older proven parts to seamlessly integrate with new payloads. Furthermore, I<sup>2</sup>C is capable of running multiple masters (one microcontroller is capable of requesting data from another microcontroller on the bus without going through an intermediary bus controller), allowing future designers to utilize similar design architectures as found in MIL-1553b at a faster speed. Additionally, I<sup>2</sup>C has natural noise immunity, a wide working voltage range; it is also insensitive to temperature swings and consumes a minimum amount of power (Phillips NXP Semiconductor 2000). Finally, I<sup>2</sup>C transceivers are already found in many microcontrollers and flight computers, allowing for a relatively easy adaptation of technology developed on smaller satellites to larger ones.

#### 4.1.2 Disadvantages of an I<sup>2</sup>C Based Standard

The interface proposed is not entirely free of problems and has its tradeoffs including reliability issues associated with sharing a common bus, lack of speed associated with time division data



transfer methods, and the possibility of utilizing all available addresses on a complex satellite. Most reliability issues with an I<sup>2</sup>C based interface can be addressed with simple solutions; some of these solutions can be hardware based, like the proposed dead man switch, and others can be software based solutions, like an intelligent power/bus monitor subsystem. Either way reliability can be improved upon with little more than minor design changes that are already within the I<sup>2</sup>C specification. Additional ways to improve bus reliability would be to run a secondary I<sup>2</sup>C bus in parallel, similar to a standard MIL-STD-1553 system, this could potentially allow the system to recover gracefully if only one bus was experiencing problems. However, this interface will always have problems associated with shared time access busses, which must be kept in mind when designing a subsystem using this specification. Nevertheless, this problem can be mitigated if it is encountered by using faster data transfer speeds. Address limitations will most likely never be a problem on small satellites as there are theoretically 128 possible combinations with 7 bit I<sup>2</sup>C addressing. The only real addressing problems will happen when specialized components, like temperature sensors, share identical addresses.

## **4.2 Thermal Design**

The design process illustrated the need for a thermal control subsystem to protect thermally sensitive subsystems such as the battery and the camera. It should also be noted that most of the commercially available integrated circuits (ICs) are designed for automotive thermal environments (-40°C to 85°C in operation). To mitigate the cold case, a thermostat system was developed and heaters were appropriately sized to keep the spacecraft warm in the cold case. To augment this system, I<sup>2</sup>C based temperature sensors are attached to each of the subsystems to monitor and gather health data to help designs for future missions. From base line calculations, it was found that the hot cases were not a major concern for the spacecraft and additional methods to cool the spacecraft would greatly affect the mass and power budgets. Figure 16 and Table 17 illustrate how the thermal subsystem interacts with the rest of the spacecraft.

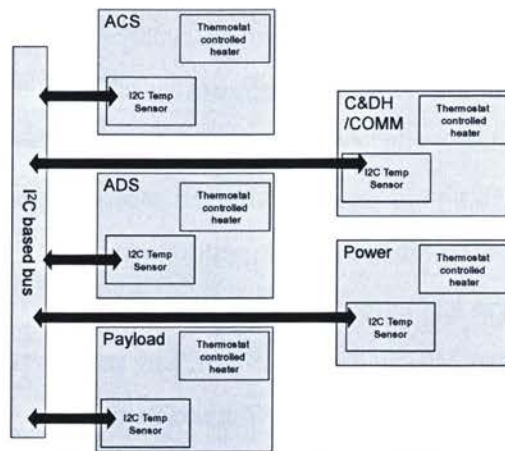


Figure 16: Thermal subsystem block diagram

Table 17: Interface matrix

Type	Input	Output
Power	3.3V 850 mA source 3.3V uninterruptible source	
Data	Standard I <sup>2</sup> C based interface (temperature sensors)	Temperature in I <sup>2</sup> C format
Structure	Attached to all cards	
Thermal	Local temperatures	6 micro heaters (system wide total of 2.3 W of heat) Heat loss through structure
Electrical noise		Possible noise due to Switching of heater ( $f < .5$ Hz)

### Analysis

Thermal analysis was conducted using the methods outlined in (Larson and Wertz 2005). Table 18 to Table 21 and the additional tables found in Appendix A were used to ensure the heaters were adequately sized for the mission.

First, the hot case was calculated using the assumption that one face of the CubeSat was facing the Sun; another was facing the Earth, while the remaining sides were facing cold space. This simplified the process of determining the geometric factors for albedo and Earth infrared effects. From these assumptions, the heat flux for each side was calculated and summed together. Next, assumptions regarding heat generated by the parts within the spacecraft were added to this heat flux. Finally, radiative means of expelling heat were modeled and temperatures on the surface and within the spacecraft were calculated. This process was carried out similarly for the cold case but with the assumption the sun facing side was actually facing cold space.

The 1000 km orbit was determined to be the worst case for RyeSat thermally as it was the coldest. It was determined that a minimum of 2.3W of heating was needed to satisfy the typical minimum temperature requirements for most components on the spacecraft (-40°C). For the other extremes (500km), it was established that the spacecraft would have no problems with components over-heating as the spacecraft's overall temperatures would be around 79°C, within the components cooling requirements.

**Table 18: Spacecraft overall temperature at 1000 km without heaters**

overall	HOT	COLD
Q power consumed [W]	3	0.5
Q total in HOT [W]	23.93419004	6.567948
T average surface (°C)	28.61755116	-54.6972
T average inside(°C)	70.45526918	-43.2162

**Table 19: Spacecraft overall temperature at 1000 km with 2.35W heater**

overall	HOT	COLD
Q power consumed [W]	3	0.5
Q total in HOT [W]	23.93419004	8.917948
T average surface (°C)	28.61755116	-37.3496
T average inside(°C)	70.45526918	-21.7608

**Table 20: Spacecraft overall temperature at 500 km without heaters**

overall	HOT	COLD
Q power consumed [W]	3	0.5
Q total in HOT [W]	25.7198	7.9041
T average surface (°C)	34.0923	-44.3531
T average inside(°C)	79.0513	-30.5365

**Table 21: Spacecraft overall temperature and at km with 2.35W heater**

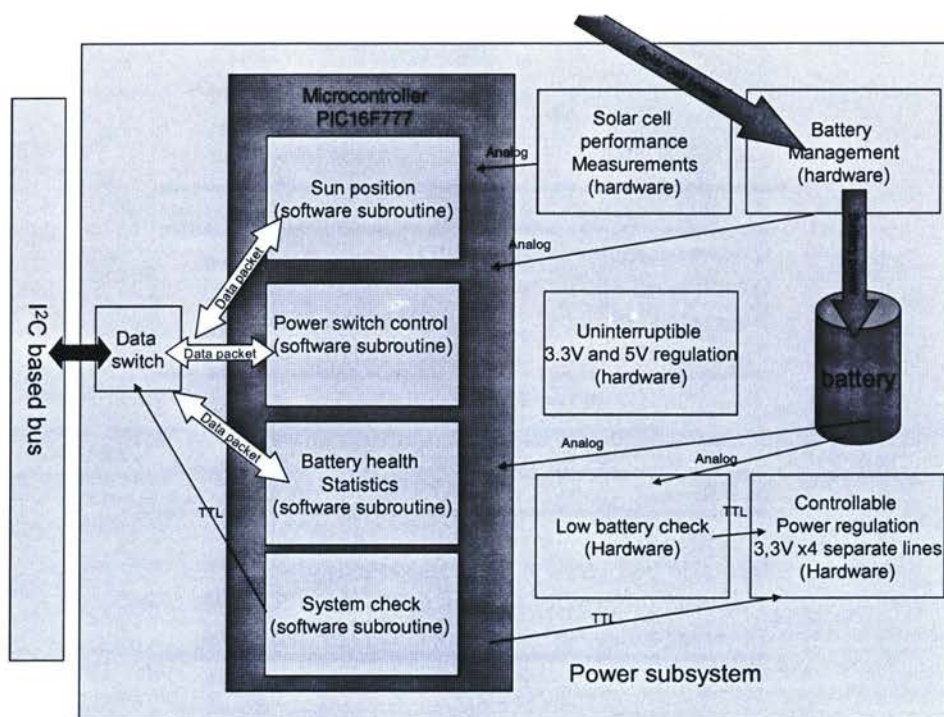
overall	HOT	COLD
Q power consumed [W]	3	0.5
Q total in HOT [W]	25.7198	10.2541
T average surface (°C)	34.0923	-28.9794
T average inside(°C)	79.0513	-11.0548

### 4.3 Power Subsystem

The power subsystem is one of the key subsystems on RyeSat; it is responsible for charging the satellite's battery, using power from the solar cells. This subsystem will also be regulating and controlling the flow of power to secondary systems such as the ACS, ADS and payload. This



system will also be providing uninterruptible power to mission critical systems such as the radio and C&DH module. To accomplish these tasks, the functions and structure illustrated in Figure 17 was implemented into a single subsystem. Table 22 illustrates the interaction of the subsystem with the rest of the spacecraft.



**Figure 17: Power subsystem function block diagram**

Table 22: Interface matrix

Type	Input	Output
Power	11 Raw Solar cells Battery	3.3 uninterrupted
		5V uninterrupted
		Battery
		Feed line 1 (3.3V 200 mA)
		Feed line 2 (3.3V 800 mA)
Data	Programming interface (PGC PGD) System wide reset (activates when brought low) Commands to turn on supply lines Commands to turn off supply lines Commands requesting Battery Voltage Commands Requesting battery current Commands requesting remaining battery life	Feed line 3 (3.3V 200 mA)
		Feed line 4 (3.3V 800 mA)
		Battery voltage
		Battery current draw
		Estimated Battery life in mAHrs
Structure	1 Standard PCB card (see structure document for details)	Connection to structure in corners
Thermal	1 micro heater	4 thermal pads in corner
Electrical noise		Possible 1 MHz from DC-DC converters

### Analysis

To provide a base line of the power available, three test cases were considered; these are the Sun directly facing a single side of the spacecraft, the Sun striking a edge directly on exposing two sides at an equal angle, and the Sun facing a corner of the spacecraft striking three faces at an equal angles. It should be noted that these cases are only ideal cases and, without a maximum power point tracking system (MPPT) it is unlikely to see such power output from the solar cells. Table 25 and Table 26 show power budgets, and were used to determine if at any point the power subsystem would have to cut off a secondary subsystem to maintain power for mission critical systems. The equation that is used to estimate the output of the solar cells is shown below in 4-1.

$$P = \cos(\theta) \cdot \eta_{\text{solarcell}} \cdot S \cdot A \quad 4-1$$

Where  $\theta$  is the angle of the cell to the Sun,  $\eta_{\text{solarcell}}$  is the efficiency of the solar cell,  $A$  is the area of the cell, and  $S$  is the solar flux,  $S = 1320 \text{ W/m}^2$ .

**Table 23: Solar cell power generation assumptions**

Satellite power calculations temperature	28 deg c
Solar constant	1320
Solar cell efficiency BOL loaded	26.5%
Solar cell efficiency EOL	22.1%
Solar cell voltage @ max power	2.4
Solar cell current @ max power(cm <sup>2</sup> )	0.016
Solar cell area cm <sup>2</sup>	26.62
# of cells per side	2

**Table 24: Power outputs from solar cells**

	BOL	EOL
Power generated Case 1[W] (direct sunlight against one face)	1.8623352	1.55311728
Power generated Case 2[W] (two sides)	2.633739698	2.196439521
Power generated Case 3 [W]	4.562018089	3.804550934



Table 25: Power consumption (without heaters on)

Idle conditions	Voltage	Current amps	Power(W)	notes
Radio receive	5.0	0.080	0.400	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.000	0.000	powered off
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.000	0.000	powered off
Thermal	3.3	0.000	0.000	powered off
Total			0.549	

Transmit	Voltage	Current amps	Power(W)	notes
Radio transmit	5.0	0.320	1.600	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.000	0.000	powered off
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.000	0.000	powered off
Thermal	3.3	0.000	0.000	powered off
Total			1.749	

Payload mission	Voltage	Current amps	Power(W)	notes
Radio receive	5.0	0.080	0.400	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.100	0.330	worst case estimate
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.000	0.000	powered off
Thermal	3.3	0.000	0.000	powered off
Total			0.879	

Attitude control	Voltage	Current amps	Power(W)	notes
Radio receive	5.0	0.080	0.400	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.000	0.000	powered off
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.600	1.980	worst case estimate
Thermal	3.3	0	0.000	powered off
Total			2.529	

Table 26: Power consumption (with heaters on)

Idle conditions	Voltage	Current amps	Power(W)	notes
Radio receive	5	0.080	0.400	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.000	0.000	powered off
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.000	0.000	powered off
Thermal	3.3	0.800	2.640	worst case
Total			3.189	

Transmit	Voltage	Current amps	Power(W)	notes
Radio transmit	5	0.320	1.600	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.000	0.000	powered off
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.000	0.000	powered off
Thermal	3.3	0.800	2.640	worst case
Total			4.389	

Payload mission	Voltage	Current amps	Power(W)	notes
Radio receive	5	0.080	0.400	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.100	0.330	worst case estimate
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.000	0.000	powered off
Thermal	3.3	0.800	2.640	worst case
Total			3.519	

Attitude control	Voltage	Current amps	Power(W)	notes
Radio receive	5	0.080	0.400	radio data sheet
C&DH idle	3.3	0.010	0.033	estimate
Power	3.3	0.010	0.033	estimate
Payload	3.3	0.000	0.000	powered off
ADS	3.3	0.025	0.083	estimate
ACS	3.3	0.600	1.980	worst case estimate
Thermal	3.3	0.800	2.640	worst case
Total			5.169	

## 4.4 Attitude Control Subsystem

The attitude control subsystem is an I<sup>2</sup>C based magnetic torque controller; its task is to control power to three orthogonally placed magnetic torquers. Figure 18 shows its configuration and Table 27 illustrates the subsystem's interaction with the rest of the spacecraft.



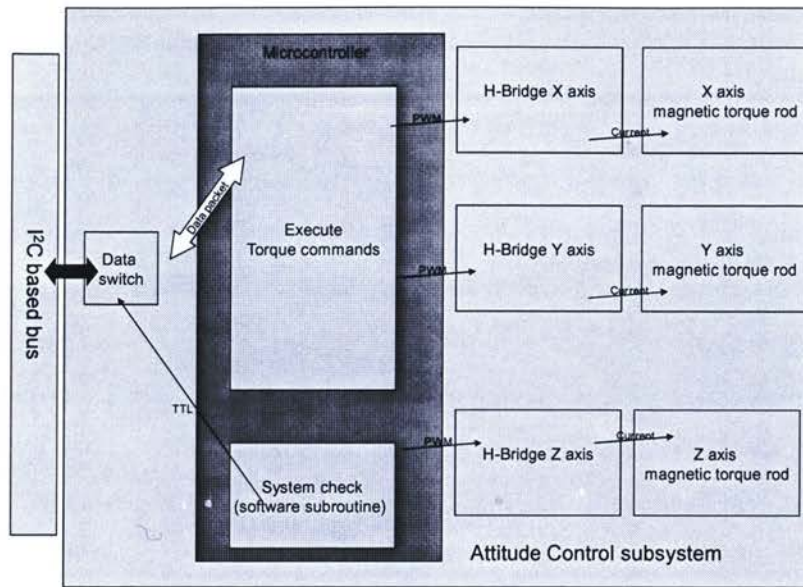


Figure 18: ACS hardware block diagram

Table 27: Interface matrix

Type	Input	Output
Power	3.3V 800mA	
Data	Command to de-tumble satellite Command to turn on torque coil, duration , and power	Error messages Acknowledgements Current usage
Structure	1 Standard PCB card (see structure document for details) 1 magnetic torque coils attached to wall of spacecraft	Connection to structure in corners
Thermal	1 micro heater	4 thermal pads in corner
Electrical noise		
Magnetic moment	Residual magnetic field on torquer Magnetic moments due Current loops in the spacecraft	3 independent magnetic torque rods

#### 4.4.1 Spacecraft Disturbance Torques

One of the first steps in designing an ACS is to determine the magnitude of the disturbances acting on the spacecraft. CubeSats are subjected to various disturbance torques including gravity gradient, solar radiation pressure, magnetic and aerodynamic disturbances.

The following sections and tables illustrate the methods and assumed values used to estimate disturbance torques in 400 and 1000km orbits; these tables were adapted from methods presented in Larson and Wertz. Table 28 shows the standard formulas for estimating disturbance torques.

Table 28: Disturbance torque models

Type of disturbance torque	Standard formula (Larson and Wertz 2005)
Gravity Gradient	$\tau_{gg} = \frac{3\mu}{2r_{orbit}^3}  I_{max} - I_{min}  \sin 2\theta$
Solar Radiation Pressure	$F = \frac{F_s}{c} A_s (1 + q) \cos i$ $\tau_{srp} = F (C_{ps} - C_m)$
Magnetic	$\tau_{mag} = D_{residual} B_{earth}$ $\tau_{mag_{max}} = D_{residual} \frac{2M_{earth}}{R_{orbit}^3}$
Aerodynamic	$F = \frac{1}{2} \rho v^2 A C_D$ $\tau_{aero} = F (C_{pa} - C_m)$

### Assumptions

To estimate the largest gravity gradient torque on a CubeSat, the center of mass was located 2 cm away from the volumetric center of the CubeSat (adhering to the limit imposed by the CubeSat standard). Considering the simple planer case as shown in Figure 19 the masses (m1, m2, m3) can be analytically solved given the desired center of mass and the spacecrafts total mass.

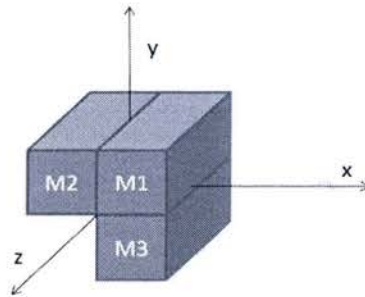


Figure 19: Considered shape to calculate worst case gravity gradient

Assuming the center of mass was located at  $x=2, y=2, z=0$  cm (coordinate frame at the geometric center) and the total mass of the system was 1kg, it is determined that the masses shown in Figure 19 m1,m2,m3 are 0.4kg, 0.3kg, 0.3kg, respectively and the system's principal moments of inertia would be  $I_{xx} = .002, I_{yy} = .003, I_{zz} = .005$  kg·m<sup>2</sup>. Solar radiation pressure and aerodynamic drag disturbance torques are calculated considering that the force was acting at the

center of pressure located at  $x=-5$ ,  $y=0$ ,  $z=0$  cm, the center of one of the plates furthest from the worst case centre of mass.

As a CubeSat is limited in size, there is a practical limit to the size of a potential magnetic dipole. As magnetic dipoles are created from current loops in the satellite, they are limited in cross-sectional area by the size of the spacecraft. Another limiting factor is the number of possible loops on a single board, as well as the amount of current through the loop. In this particular case, it was assumed that the maximum area for a current loop was 10cm x 10cm, a maximum of 2 current loops could exist on a single board and depending on the subsystem the maximum current was either 250 or 800mA. The other possible magnetic dipoles that could be created in the satellite would be from the vertical supply lines; these dipoles were modeled as single current loops with varying distances from the ground pin. An example of how these fields were oriented is shown in Figure 20, and the estimated worst case magnetic moments are tabulated in Table 29.

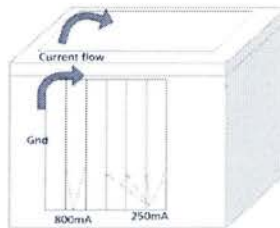


Figure 20: Layout of potential current loops

Table 29: Estimated magnetic moments from current loops

	Magnetic moment Horizontal ( $\text{Am}^2$ )	Magnetic moment Vertical ( $\text{Am}^2$ )
power on a board 1 (3.3@800mA)	0.016	0.0008
power on a board 2 (3.3@800mA)	0.016	0.0016
power on a board 3 (3.3@250mA)	0.005	0.00075
power on a board 4 (3.3@250mA)	0.005	0.001
power on a board 5 (3.3@250mA)	0.005	0.00125
power on a board 6 (5V@80mA)	0.0016	0.00048
total magnetic moment dipole	0.0486	0.00588

This analysis illustrates that the maximum dipole of the satellite can be estimated to be on the order of .05 [ $\text{A}\cdot\text{m}^2$ ].



Table 30: disturbance torques at 400km

Space craft parameters	
X [cm]	10.0
Y[cm]	10.0
Z[cm]	10.0
Mass [kg]	1.00
Ixx (ideal)[kg·m <sup>2</sup> ]	1.67E-03
Iyy (ideal)[kg·m <sup>2</sup> ]	1.67E-03
Izz (ideal)[kg·m <sup>2</sup> ]	1.67E-03
Ixx (worst case)[kg·m <sup>2</sup> ]	2.00E-03
Iyy (worst case)[kg·m <sup>2</sup> ]	3.00E-03
Izz (worst case)[kg·m <sup>2</sup> ]	5.00E-03
max moment of inertia[kg·m <sup>2</sup> ]	5.00E-03
minimum moment of inertia[kg·m <sup>2</sup> ]	2.00E-03
Centre of pressure x [cm]	5.00
Centre of pressure y [cm]	0.00
Centre of pressure z [cm]	0.00
Centre of mass x [cm]	2.00
Centre of mass y [cm]	2.00
Centre of mass z [cm]	0.00
Centre of solar pressure x[cm]	5.00
Centre of solar pressure y[cm]	0.00
Centre of solar pressure z [cm]	0.00
C solar pressure -C gravity [cm]	7.28
C pressure -C gravity [cm]	7.28
Area side 1 [m <sup>2</sup> ]	1.00E-02
Area side 2 [m <sup>2</sup> ]	1.00E-02
Area side 3 [m <sup>2</sup> ]	1.00E-02
Area side 4 [m <sup>2</sup> ]	1.00E-02
Area side 5 [m <sup>2</sup> ]	1.00E-02
Area side 6 [m <sup>2</sup> ]	1.00E-02
Environmental parameters	
p(atmosphere) [kg/m <sup>3</sup> ]	1.05E-11
C (speed of light) [m/s]	3.00E+08
Magnetic moment of earth [T·m <sup>3</sup> ]	7.96E+15
orbit parameters assume circular	
h [km]	400
orbital period [s]	5.55E+03
mass of earth [kg]	5.97E+24
gravitational constant [m <sup>3</sup> /kg/s <sup>2</sup> ]	6.67E-11
Radius of earth [km]	6380
R [m]	6780
μearth [m <sup>3</sup> /s <sup>2</sup> ]	3.99E+14
Gravity Gradient	
Max Torque due to gravity gradient [N·m]	5.76E-09
Solar radiation	
F [N]	8.43E-08
Torque due to solar radiation [N·m]	6.14E-09
magnetic field	
natural residual dipole of space craft [A·m <sup>2</sup> ]	4.86E-02
strongest expected magnetic field	5.11E-05
torque due to magnetic forces [N·m]	2.48E-06
Aerodynamic	
Cd	2.50E+00
V [m/s]	7.67E+03
F [N]	7.72E-06
Torque due to aerodynamic forces [N·m]	5.62E-07
total disturbance torque [N·m]	3.06E-06

Table 31: disturbance torques @ 1000km

Space craft parameters	
X [cm]	10.0
Y[cm]	10.0
Z[cm]	10.0
Mass [kg]	1.00
Ixx (ideal)[kg·m <sup>2</sup> ]	1.67E-03
Iyy (ideal)[kg·m <sup>2</sup> ]	1.67E-03
Izz (ideal)[kg·m <sup>2</sup> ]	1.67E-03
Ixx (worst case)[kg·m <sup>2</sup> ]	2.00E-03
Iyy (worst case)[kg·m <sup>2</sup> ]	3.00E-03
Izz (worst case)[kg·m <sup>2</sup> ]	5.00E-03
max moment of inertia[kg·m <sup>2</sup> ]	5.00E-03
minimum moment of inertia[kg·m <sup>2</sup> ]	2.00E-03
Centre of pressure x [cm]	5.00
Centre of pressure y [cm]	0.00
Centre of pressure z [cm]	0.00
Centre of mass x [cm]	2.00
Centre of mass y [cm]	2.00
Centre of mass z [cm]	0.00
Centre of solar pressure x[cm]	5.00
Centre of solar pressure y[cm]	0.00
Centre of solar pressure z [cm]	0.00
C solar pressure -C gravity [cm]	7.28
C pressure -C gravity [cm]	7.28
Area side 1 [m <sup>2</sup> ]	1.00E-02
Area side 2 [m <sup>2</sup> ]	1.00E-02
Area side 3 [m <sup>2</sup> ]	1.00E-02
Area side 4 [m <sup>2</sup> ]	1.00E-02
Area side 5 [m <sup>2</sup> ]	1.00E-02
Area side 6 [m <sup>2</sup> ]	1.00E-02
Environmental parameters	
p(atmosphere) [kg/m <sup>3</sup> ]	1.43E-12
C (speed of light) [m/s]	3.00E+08
Magnetic moment of earth [T·m <sup>3</sup> ]	7.96E+15
orbit parameters assume circular	
h [km]	1000
orbital period [s]	6.31E+03
mass of earth [kg]	5.97E+24
gravitational constant [m <sup>3</sup> /kg/s <sup>2</sup> ]	6.67E-11
Radius of earth [km]	6380
R [m]	7380
μearth [m <sup>3</sup> /s <sup>2</sup> ]	3.99E+14
Gravity Gradient	
Max Torque due to gravity gradient [N·m]	4.47E-09
Solar radiation	
F [N]	8.43E-08
Torque due to solar radiation [N·m]	6.14E-09
magnetic field	
natural residual dipole of space craft [A·m <sup>2</sup> ]	4.86E-02
strongest expected magnetic field	3.96E-05
torque due to magnetic forces [N·m]	1.93E-06
Aerodynamic	
Cd	2.50E+00
V [m/s]	7.35E+03
F [N]	9.66E-07
Torque due to aerodynamic forces [N·m]	7.03E-08
total disturbance torque [N·m]	2.01E-06

#### 4.4.2 Spacecraft Actuator Sizing

There are two major types of spacecraft actuators: those that store momentum on the spacecraft, such as reaction wheels, and those that apply an external torque to the satellite, such as magnetic torquers. The following sections are used to size reaction wheels and magnetic torquers for RyeSat.

##### A. Reaction Wheel Sizing

There are 3 main criteria for sizing reaction wheels. These criteria are: minimum torque required to reject disturbances, the minimum torque required to slew at a desired rate, and the minimum amount of angular momentum storage needed over an orbital period. Table 32 summarizes the methods used to estimate these criteria.

Table 32: Reaction wheel sizing criteria

Requirement	Standard formula (Larson and Wertz 2005)	
Ability to reject disturbance torques	$\tau_{\text{wheel}} = \tau_{\text{Dist}}(SF)$ Design Margin =2	6.10E-06 [N·m]
Ability to slew	$\tau_{\text{wheel}} = \frac{4\theta I}{t^2}$ Required slew angle 90[deg] Time to slew 60 [s]	8.72665E-06 [N·m]
Momentum storage	$H_{\text{storage}} = \frac{\tau_{\text{Dist}} P}{4\sqrt{2}}$ Orbital period is 5553[s]	3.00E-03[N·m·s]

To investigate if it was possible to scale down a reaction wheel for use on RyeSat, a study was performed to determine if a motor and flywheel could be selected from those available on the market. To ensure the selected motor was capable of controlling the spacecraft the following methods were used.

The following relationship is used to relate torque to an applied voltage

$$\tau_{\text{motor}} = -\frac{k_i^2}{R} \omega_{\text{motor}} + k_i \frac{V_{\text{applied}}}{R} \quad 4-2$$

A candidate motor (Portescap 2007) was found with the following motor characteristics (Table 33 and Figure 21 ). From a design perspective, this motor was ideal for several reasons including its brushless DC design, its maximum design spin rate of 50,000 RPM, and its large rotator inertia.

Table 33: Example motor constants

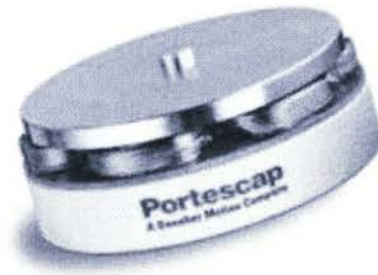
Ki [N·m/A]	7.80E-03
L [mH]	36
R [ohms]	3.75
$I_{\text{motor}}$ [kg·m <sup>2</sup> ]	1.13E-06
$V_{\text{applied}}$ [V]	3.00
Mass of motor (datasheet) [kg]	0.026

With these basic motor constants the following motor characteristics presented in Table 34 were calculated.

**Table 34: Example motor characteristics**

Max wheel speed [rad/second]	385
Max wheel speed [rpm]	3625
Torque @ max power[mNm]	3.12
Max power [W]	0.6
Max current[A]	0.2
Max momentum storage (motor only) [mN·m·s]	0.0435

This particular motor has a suitable amount of torque to accomplish both slew and disturbance rejection; however, despite the motors large rotor inertia, its momentum storage is insufficient for the proposed mission. To make this motor suitable for RyeSat, a flywheel must be added to increase the motor's moment of inertia.



**Figure 21: Picture of Portescap motor**

The size of the flywheel is determined by the following relationship:

$$(I_{\text{flywheel}})_{\min} = (I_{\text{rw}} - I_{\text{motor}}) = \left( \frac{H_{\text{storage}}}{\omega_{\max}} \right) - I_{\text{motor}} \quad 4-3$$

**Table 35: Flywheel sizing considerations**



Minimum Momentum storage [mN·m·s]	3.00
max wheel speed [rad/second]	385
required moment of inertia (Reaction wheel) [kg·m <sup>2</sup> ]	7.79E-06
minimum moment of inertia (flywheel) [kg·m <sup>2</sup> ]	6.66E-06

Assuming the flywheel is a solid disc, the moment of inertia of the flywheel is given as

$$I = \frac{1}{2}mr^2$$

$$m = \rho h\pi r^2$$

$$I = \frac{1}{2}\rho h\pi * r^4$$

4-4

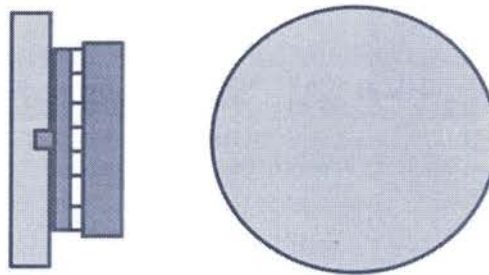


Figure 22: Flywheel: solid disc configuration

Table 36: Flywheel: solid disc parameters

Density [kg/m <sup>3</sup> ]	7800
Radius of needed wheel [mm]	20
Thickness of needed wheel [mm]	6
I flywheel [kg·m <sup>2</sup> ]	1.176E-05
Mass of flywheel [kg]	0.0588
Mass of motor (datasheet)[kg]	0.026
Mass of assembly[kg]	0.085
Reaction wheel inertia [kg·m <sup>2</sup> ]	1.29E-05
Momentum storage [mN·m·s]	4.96
Time to go from max speed on direction to another [s]	3.18

It is obvious from Eq. (4-4) that this design is an inefficient use of mass. However, this design will allow for mounting on either the motor's shaft, or directly to the flat rotor on the top of the motor.

A mass efficient design would use a ring mounted to the flat rotor on top of the motor. Although balancing this reaction wheel will be more difficult, and further analysis is needed to determine if this is physically possible with the candidate motor.

$$I = \frac{1}{2}m(r_{\text{outer}}^2 + r_{\text{inner}}^2)$$

$$m = \rho h \pi (r_{\text{outer}}^2 - r_{\text{inner}}^2)$$

$$I = \frac{1}{2} \rho h \pi (r_{\text{outer}}^4 - r_{\text{inner}}^4)$$

4-5

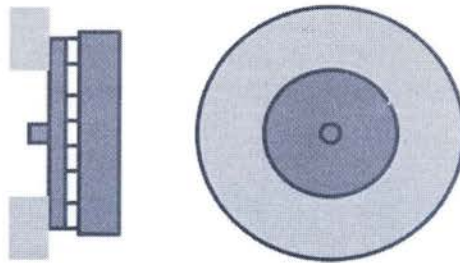


Figure 23: Mass efficient flywheel design

Table 37: Simple ring flywheel parameters

Density [kg/m <sup>3</sup> ]	7800
R_inner [mm]	10
R_outer [mm]	20
Thickness of needed wheel [mm]	4
Mass of flywheel [g]	29.4
Mass of assembly[g]	55
I flywheel [kg·m <sup>2</sup> ]	7.35E-06
Reaction wheel inertia [kg·m <sup>2</sup> ]	8.48E-06
Momentum storage [mN·m·s]	3.26
Time to go from max speed on direction to another [s]	1.05

For comparison a few currently available or proposed reaction wheels were compared with this design in Table 38.

Table 38: Comparison to other available wheels

	CubeSat kit ADACS (Pumpkin 2008)	Sinclair interplanetary (Sinclair Interplanetary 2008)	TU Berlin (Kayal, et al. 2005)	Proposed
Nominal torque [mN·m]	6	2	.003	3.4

		CubeSat kit ADACS (Pumpkin 2008)	Sinclair interplanetary (Sinclair Interplanetary 2008)	TU Berlin (Kayal, et al. 2005)	Proposed
Nominal momentum storage	[mN·m·s]	1.1	30	.1	3.26-4.96
Voltage	[V]	12	3-6	3-5	3.3
Power	[W]	1.5 <sup>3</sup>	2	.3W - .7W	.8
Mass	[g]	910 <sup>4</sup>	185	44	55-85
Dimensions	[cm]	[10,10,7]	[5,5,4]	[2.5,2.5,1.2]	[4,4,2]

### B. Magnetic Torquer Sizing

Magnetic torquers produce an external torque on a spacecraft by creating a magnetic dipole. This magnetic dipole then creates the torque described in Eq. (4-6). Magnetic torquers are traditionally used on larger spacecraft to offload momentum from storage devices such as reaction wheels, momentum wheels, or control moment gyros. On some smaller spacecraft (and some CubeSats) magnetic torquers have been used as the sole means of attitude control.

$$\tau_{\text{torquer}} = m_{\text{applied}} \times B_{\text{Earth}} \quad 4-6$$

To properly size the magnetic torquer to be the sole means of attitude control in the spacecraft, the torquer must be designed to provide the required amount of torque to counteract disturbances even when far away from the optimal orientation with the Earth's magnetic field. Even with oversized torque coils there will be instances when the spacecraft is not correctly oriented in the Earth's magnetic field to apply a torque to counteract disturbances. However, limits can be set on how far the magnetic torquer and the Earth's magnetic field can be from an optimal orientation before the coil loses its effectiveness, and the control algorithm can be designed to account for periods when torque is unavailable.

Table 39: Magnetic torquer requirements

orbit parameters assume circular	
$h$ lowest [km]	400
$a$ lowest [km]	6778.1
$h$ highest [km]	1000
$a$ highest [km]	7378.1

<sup>3</sup> Estimate for one wheel in a sealed box

<sup>4</sup> Includes magnetic torquers and 2 other wheels



orbit parameters assume circular	
Magnetic moment constant [T·m <sup>3</sup> ] (Larson Wertz)	7.96E+15
Strongest earth magnetic field (lowest alt ~poles) [T]	5.11E-05
Weakest earth magnetic field (highest alt ~equator) [T]	1.98E-05
Torque required from coils [mN·m]	5.00E-03
$m \times B$ min angle between torquer and earths magnetic field to be effective [deg]	45
required m (highest alt) [Am <sup>2</sup> ]	0.3568
required m (lowest alt) [Am <sup>2</sup> ]	0.1383

### Coreless Magnetic Torque Coil

In an open core design the magnetic moment is produced orthogonally to the windings. This style of coil's magnetic moment can be calculated using the following equation

$$m = NIA_{\text{coil}} \quad 4-7$$

Where  $N$  is the number of loops,  $I$  is the current, and  $A$  is the area of the coil.  $I$  is then found using ohm's law and the material properties of the wire

$$I = \frac{V_{\text{applied}}}{R} \quad 4-8$$

$$R = \frac{N(2\pi r_{\text{coil}})\Phi}{A_{\text{wire}}}$$

$$I = \frac{V_{\text{applied}}A_{\text{wire}}}{N(2\pi r_{\text{coil}})\Phi} \quad 4-9$$

And assuming a circular coil,

$$A_{\text{coil}} = \pi r_{\text{coil}}^2 \quad 4-10$$

Combining Eq. (4-7), Eq. (4-9), and Eq. (4-10) yields

$$m = \frac{A_{\text{wire}}r_{\text{coil}}V_{\text{applied}}}{2\Phi} \quad 4-11$$

If there is a limit on the coil mass, the number of turns allowed in the coil is limited. If the number of coils is limited too much, the current demands will be too high to maintain the desired magnetic moment and may even be too high for the wire to safely conduct. Conversely if the current is highly limited, the number of turns required will cause the magnetic torquer to weigh too much to comply with the mass budget.

$$N = \frac{M}{2\pi r_{\text{coil}} A_{\text{wire}} \rho_{\text{wire}}} \quad 4-12$$

Table 40: Air core design parameters

cross-sectional area of the wire [mm <sup>2</sup> ]	.102
Radius of coil [cm]	4
Voltage applied [V]	3.3
resistivity [Ω/m <sup>3</sup> ]	1.72E-08
Maximum magnetic moment [Am <sup>2</sup> ]	0.392
N [turns]	400
Current (I) [A]	0.194
density of copper [kg/m <sup>3</sup> ]	8930
Mass [g]	91.7

From Eq. (4-11) it is found that there are only three obvious parameters to change the magnitude of the magnetic dipole; applied voltage, radius of the coil, and the cross-sectional area of the wire (gauge). The resistivity of copper will vary when the device is used (due to heating) however as long as currents are within limits the coil will not heat up appreciably.

#### Ferrite Core Magnetic Torquer Design

This type of magnetic torquer is more commonly found on larger spacecraft. The magnetic dipole created by this torquer is along the axis of the core. Although this method is more complex to analyze it has the added benefit of a stronger magnetic dipole (at least 50 times stronger with a poorly selected core). One of the problems encountered when designing a cored magnetic torquer is to ensure the coil is properly designed for the core. If the designed coil is too strong for the core selected it may induce a constant magnetic field along the core creating a constant torque on the satellite. The magnitude of the magnetic moment created by this torquer can be modeled using Eqs. (4-13 to 4-15). This analysis is adapted from the one originally found in the Lionsat magnetic torquer design (Wagner, et al. 2003).

$$B = \frac{\mu_o N I}{L \left( \frac{1}{\mu_{\text{core}}} + N_d \right)} \quad 4-13$$

$$N_d = \frac{4 \left[ \ln \left( \frac{L}{r} \right) - 1 \right]}{\left( \frac{L}{r} \right)^2 - 4 \ln \left( \frac{L}{r} \right)} \quad 4-14$$

$$m = \frac{B\pi r^2 L}{\mu_o} \quad 4-15$$

Rearrange the preceding equations to solve for m

$$m = \frac{\pi r^2 N I}{\frac{1}{\mu_{core}} + N_d} \quad 4-16$$

Mass is then calculated using the following method:

$$\begin{aligned} M &= M_{core} + M_{coil} \\ M_{core} &= \rho_{core} \pi r^2 L \\ M_{coil} &= \rho_{wire} \left( \frac{\pi}{4} D^2 \right) 2\pi r N \end{aligned} \quad 4-17$$

The power consumed is determined as

$$\begin{aligned} P &= VI \\ \text{or} \\ P &= \frac{V^2}{R} \\ R &= \Phi_{wire} \cdot A_{wire} \cdot L_{wire} \\ R &= \Phi_{wire} \left( \frac{\pi}{4} D^2 \right) 2\pi r N \end{aligned} \quad 4-18$$

Table 41: Ferrite core design (magnetic parameters)

$\mu_o$	1.25664E-06
$\mu_{r1}$	125
maximum Voltage [V]	3.3
maximum current[A]	0.1185
Resistance[Ω]	27.8551
maximum power consumed[W]	0.3910
wire diameter (31gague) [mm]	.226
resistivity per volume[Ω/m <sup>3</sup> ]	1.72E-08
estimated resistivity per m[Ω/m]	0.4285
length of wre used [m]	65
Radius of core [mm]	3.175
Length of coil on core [cm]	6.5
Number of turns in the coil	3258
Demagnetization factor(Nd)	0.0198
B	0.2680
Maximum magnetic moment [Am <sup>2</sup> ]	0.4391

Table 42: Ferrite core design (mass parameters)

density of core [kg/m <sup>3</sup> ]	7870
--------------------------------------	------



density of wire[kg/m <sup>3</sup> ]	8930
mass of core [g]	17.4
Mass of coil [g]	23.3
total mass [g]	40.7

Table 43: Ferrite core design (size parameters)

number of coils per layer	288
number of layers	12
overall diameter [cm]	1.18
estimated rod length [cm]	7

Table 44: Comparison of available magnetic torquers

		(Space Quest 2008) <sup>5</sup>	Air core	Ferrite core
Magnetic moment	[Am <sup>2</sup> ]	5	.39	.44
Voltage	[V]	10	3.3	3.3
Power	[W]	3	.65	.4
Mass	[g]	200	90	40

Comparing the results in Table 40 to those in Table 41 it is found that the ferrite core design weighs almost half the mass of a similarly sized air coil and consumes less power. Therefore, the ferrite core is a superior choice.

<sup>5</sup> Nominal design custom ones are available



# CHAPTER 5

## HARDWARE TESTING AND SIMULATION

The bus will consist of the C&DH module running as a master and the remaining subsystems will act as slaves on the I<sup>2</sup>C bus (Figure 24). In addition to the subsystems, sensors and EEPROMs will also be present on the bus. A table of these devices and their addresses is presented in Table 45.

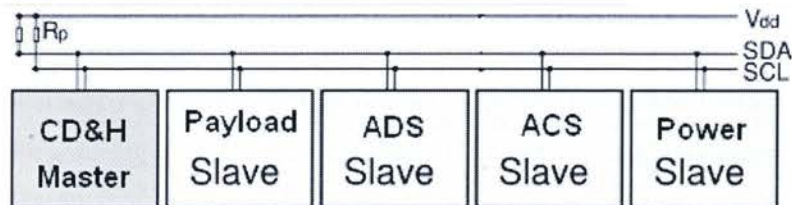


Figure 24: Physical network (Sturman 2007)

Table 45: I<sup>2</sup>C Address table for RyeSat

Modules		Temperature sensors		EEPROMs	
CDH Main	0xE0	CDH	0x90	CDH	0xA0
CDH backup	0xEE	Radio(possible)	0x92		
payload	0xE8	Power	0x94		
ADS	0xE2	ADS	0x98		
ACS	0xE4	ACS	0x9A		
Power	0xE6	Payload	0x9C		
		Payload camera	0x9E		

### 5.1 Bus Operation

The bus will have a common data sentence containing not only the obligatory I<sup>2</sup>C start, stop, collision and addressing commands but a control byte length of incoming packets and a checksum (see Figure 25). With this structure, nine cases have been defined to deal with most



data transmissions with packet sizes up to 65536 bytes long. Of these 9 cases, 6 are potential error cases that have been identified.

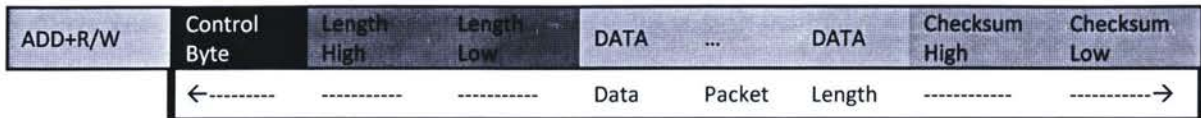


Figure 25: Sample data sentence

### 5.1.1 Case 1

The first case is when the master would like to address one system and not expect a reply. The Master (C&DH) will check to see if the bus is clear to take over. If so, it will send a start condition to the bus (as defined in the I<sup>2</sup>C specification (Phillips NXP Semiconductor 2000)). It will then send the desired slave address with the read/write bit set high, at which point the correct slave will respond with an acknowledge bit. The master will then send a control byte to inform the slave what command is being issued to it. Next, the length of the data packet is sent (from the beginning of the control byte to the end of the check sum). Subsequently, the slave will initiate a loop to receive and store the data and the checksum. After this data sentence is sent, the master will send the stop condition to the bus and the addressed slave will revert back to an idle state on the bus. Table 46 summarizes this case.

Table 46: Case 1 (Master sends no response wanted)

Master status	Tx											
Slave status	Rx+      ack bits											
Data sent/received												
Master sends	start	Slave address + write bit	Control byte	Length MSB	Length LSB	Data [0]	Data [1]	...	Data [n]	Checksum MSB	Checksum LSB	S T O
Addressed slave		Ack	ack	ack	ack	ack	ack	...	ack	ack	ack	

### 5.1.2 Case 2

The next case is similar to the previous; the master will wait for a clear bus, send an address, control-byte, length data, and checksum. However instead of sending the stop command, it will repeat the slave's address and set the read/write bit low allowing the slave to send data back to the master. At the end of this case the slave will hold the clock low to prepare for Case 3. This case is summarized in Table 47.

Table 47: Case 2(Master sends, and a response is wanted)

Master status	Tx											
Slave status	Rx + ack bits											
Data sent/received												
Master sends	start	Slave address + write bit	Control byte	Length MSB	Length LSB	Data [0]	Data [1]	...	Data [n]	Checksum MSB	Checksum LSB	Slave address + read bit
Addressed slave		ack	ack	ack	ack	ack	ack	...	ack	ack	ack	ack

### 5.1.3 Case 3

As stated in Case 2, the clock will be held low after the master sends the slave address and the read bit. While the clock is held low, the slave will be preparing its response to the command data. Once the slave is ready, it will resend the control byte which the master can use for verification purposes. The slave will then send its response length data and a checksum. If this transfer is successful, the master will then stop the bus and both the master and slave will revert to an idle state. Table 48 summarizes case 3.

Table 48: Case 3 (Master receives a wanted response)

Master status	Rx+ ack bits										
Slave status	Tx										
Data sent/received											
Master sends	ack	ack	ack	ack	ack	...	ack	ack	Nack	STOP	
Addressed slave	Control byte	Length MSB	Length LSB	Data[0]	Data[1]	...	Data[n]	Checksum MSB	Checksum LSB		

### 5.1.4 Case 4

Case 4 is the beginning of the error cases; this particular case is designed to catch programming errors that could overwhelm the slave with data. This case will occur when the master “accidentally” sends a command with a data packet larger than expected to the slave. It will begin normally with the master sending start conditions address and read/write bit, and length. After the length is received, the slave will perform a simple check before it begins its data collection loop to determine if the length is smaller than its previously designed and documented buffer length. In the event the master is attempting to send a packet that is too large, the slave will first log an error code and then will not acknowledge (Nack) the first data byte. The master will eventually stop the bus resetting both the slave and master to idle states. Its structure is found in Table 49.

Table 49: Case 4 (incorrect packet size)

Master status	Tx						
Slave status	Rx+ ack bits						
Data sent/received							
Master sends	start	Slave address + write bit	Control byte	Length MSB	Length LSB	Data[0]	stop
Addressed slave		ack	ack	ack	ack	NACK	

### 5.1.5 Case 5

This case is designed to account for lost bytes or bus error when the master is transmitting. If during any transmission by the master a byte is corrupted (i.e., bit missing from byte), the slave will record this as the last occurred error and will “nack” as this is implemented on most microcontrollers as an automatic function, at which point the master will send a stop command to free up the bus. Its structure is found in Table 50.

Table 50: Case 5 (lost byte from master)

Master status	Tx									
Slave status	Rx+ ack bits									
Data sent/received										
Master sends	start	Slave address + write bit	Control byte	Length MSB	Length LSB	Data[0]	Data[1]	...	Data[i]	STOP
Addressed slave		ack	ack	ack	ack	ack	ack	...	NACK	

### 5.1.6 Case 6

This case is similar to Case 5 except that the error occurs when the slave is in the middle of transmitting a data sentence. If during any transmission by the slave a byte is corrupted, the master will send a “nack”, the slave will then detect this “nack” and record an error. As a “nack” is traditionally used by the master to close communication with the slave transmitting, the slave will automatically be reset to an idle state and the master can relinquish control of the bus by sending a stop command. Its structure is found in Table 51.

Table 51: Case 6 (lost byte from Slave)

Master status	Rx+ ack bits							
Slave status	Tx							
Data sent/received								
Master sends	ack	ack	ack	ack	ack	...	NACK	STOP
Addressed slave	Control byte	Length MSB	Length LSB	Data[0]	Data[1]	...	Data[i]	

### 5.1.7 Case 7

Case 7 occurs when the slave detects an incorrect checksum. This will occur after a Case 2 so, at this point, the slave will be holding the bus. Before the slave releases the bus, an error will be



recorded in the most recent error register and the slave will immediately send the error code out as its first piece of data, the master will then detect that the ctrlbyte is not the same and send the stop command, allowing both devices and the bus to return to an idle state. This case's structure is found in Table 52.

**Table 52: Case 7 (incorrect checksum on slave)**

Master status	Rx+ ack bits	
Slave status	Tx	
Data sent/received		
Master sends	Ack	STOP
Addressed slave	error code (0xE3)	

### 5.1.8 Case 8

Case 8 occurs when the response checksum is determined to be incorrect, at which point the bus should be already free, as a complete Case 2 and Case 3 would have been sent. At this point, it is up to the master to choose to re-run the command.

### 5.1.9 Case 9

Case 9 is a follow up command and is simply a modified example of Case 2 and 3 with a single byte reply. Its structure is found in Table 53.

**Table 53: Case 9 (error follow up)**

Master status	Tx							
Slave status	Rx+ ack bits							
Data sent/received								
Master sends	start	Slave address + write bit	Control byte (0xE0)	Length MSB	Length LSB	Checksum MSB	Checksum LSB	Slave address+ read bit
Addressed slave		ack	ack	ack	ack	ack	ack	Ack
Slave sends	Ack	ack	ack	ack	ack	Nack	STOP	
Addressed slave	Control byte	Length MSB	Length LSB	Error code	Checksum MSB	Checksum LSB		

### 5.1.10 Checksums and Error Codes

Checksums for these data sentences are calculated as follows:

- Formulate the response packet in memory without the checksum bytes
- Create and initialize an unsigned integer (16b) to zero.
- Iterate byte for byte through the packet, adding the byte value to the checksum integer

- Stop after the last byte of data.
- Append the checksum integer to the packet

Cautions:

- It must be ensured that the integer is in the correct format, with the Most Significant Bit (MSB) sent before the Least Significant Bit (LSB)
- It must be ensured that the integer sum can "wrap around", meaning that:  $65534 + 3 = 1$

The standard error codes all slave devices will use is listed in Table 54. Bus collision errors are only present on devices which can detect collisions on the I<sup>2</sup>C bus; this can be useful in determining if the device is faulty or if there are multiple devices trying to respond to a single command.

Table 54: List of standard bus error codes

Error codes	Code
No detected error	0xE0
Too Much Data for slave	0xE1
byte lost on Master transmission	0xE2
byte lost on slave transmission	0xE3
Incorrect checksum	0xE4
Bus Collision ~variant of byte lost on master	0xE5
Bus Collision ~variant of byte lost on master	0x06

## 5.2 Thermal Subsystem

The thermal system will utilize a hardware controlled thermostat and an I<sup>2</sup>C based temperature sensor. This will allow the thermal subsystem to operate heaters without intervention from any other system and allow reporting of individual board temperatures to the C&DH and the ground station. The circuit for the heating subsystem is shown in Figure 26.

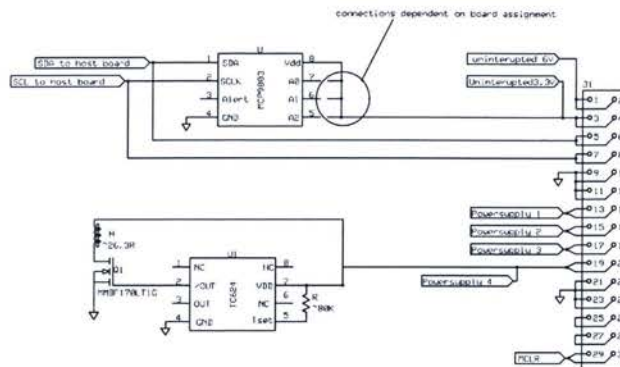


Figure 26: Thermal system schematic

### 5.2.1 Heater Control

The heater control circuit consists of a thermostat control IC (TC624) an N channel MOSFET, a Minco thin film heater and a resistor.

#### *Sizing the resistor*

The data sheet for the TC624 indicates that the /OUT pin is high when the temperature is below the trip point. This configuration is ideal for driving a heater and is even outlined in the TC624's datasheet. It is recommended that the heater be turned on 5 to 10°C before the minimum temperature (i.e., heater turns on at -35°C when the lowest minimum temperature is -40 °C)

#### *MOSFET Selection*

A MOSFET is suggested in the TC624 datasheet to act as a switch for the heater. Although the datasheet does not suggest a particular device, it does stress that it should be able to be driven with CMOS logic levels (3-5 V 250uA typically). The MOSFET should also have a low voltage drop across it to minimize the power lost before it reaches the heater coil. The chosen MOSFET is the same as the one used in the power subsystem (IRLML2502), as it meets the minimum requirements, and, utilizing similar parts, simplifies both the assembly and ordering processes.

#### *Heater Selection*

At the time of writing, MINCO offered many tools to size heaters for particular applications. For the pico-satellite application, it was determined system wide that 2.3W of heating power is needed to heat the spacecraft in a worst case scenario. With 5 individual boards in the spacecraft, analysis shows that each of the heaters must provide an average of 0.46 Watts.



Fortunately, Minco offers a heater, 3 x3 inches with the correct resistance; the part number is HK5464R23.3L12B. This device also comes with an adhesive backing approved by NASA for out-gassing. However, it is not rated for temperatures below -32°C so there should also be a mechanical means to hold this heater to the board. Alternatively, the rear plane of the board could be etched with a similar coil to heat the board.

### 5.2.2 Health Monitoring

Temperature will be taken by the MPC9803 I<sup>2</sup>C temperature sensor and passed directly to the C&DH module using the system wide I<sup>2</sup>C bus. To ensure proper operation of these devices on the bus, each will have to have a unique address. Table 55 shows the predetermined addresses for the temperature sensors board by board.

Table 55: Temperature sensor address per board

Board	Address	A2	A1	A0
C&DH	1001 000	GND	GND	GND
Radio(possible)	1001 001	GND	GND	+3.3V U
Power	1001 010	GND	+3.3V U	GND
ADS	1001 100	+3.3V U	GND	GND
ACS	1001 101	+3.3V U	GND	+3.3V U
Payload	1001 110	+3.3V U	+3.3V U	GND
Payload camera	1001 111	+3.3V U	+3.3V U	+3.3V U

## 5.3 Power subsystem

The design of this subsystem can be broken up into several subcomponents: voltage regulation, battery monitoring and conversion of power from the solar cells.

### 5.3.1 Uninterruptible Feed lines

This satellite has two uninterruptible power supply lines; one is for the main C&DH board which runs at 3.3 V and the other is a 5V line for the amateur radio.

#### 3.3V line

The 3.3V line consists of three main IC's, capacitors and a resistor. It is best to explain each device with its supporting components in the following breakdown: circuit breakers, MOSFET, and voltage regulation. The circuit for the 3.3 Volt line is shown in Figure 27.

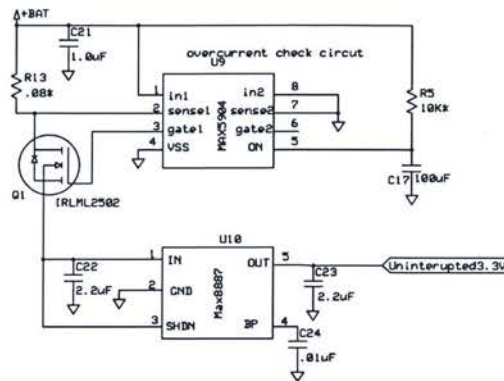


Figure 27: 3.3V uninterruptible supply circuit

### Automatic Resetting Circuit Breakers

The MAX5904 acts as an automatically resetting circuit breaker to prevent the system from drawing too much current and damaging other systems. The time it stays off after it is triggered is between .25sec - 1sec, which should be a suitable amount of time to allow for a device to recover from a single event upset or latch-up. This device uses an external current sensing resistor to allow the designer to set a unique current limit for the circuit. This particular device measures the voltage difference across the current sensing resistor and will trigger the breaker when it exceeds 25mV. Analysis to size this resistor must also consider the manufacturing tolerances of the resistor and the MAX5904 to ensure it will trigger as desired. To guarantee the circuit breaker will trigger before 250mA, a 0.08Ω current sensing resistor was chosen.

### MOSFET Selection

The MAX5904 Drives an N-Channel MOSFET to control the flow of current in the circuit; if the MAX5904 is below the 250mA limit (and everything else is connected correctly), the MAX5904 will send a high signal to the MOSFET, allowing current to flow. If the current exceeds the 250mA limit, the MAX5904 will pull the MOSFET's gate low, preventing current from flowing. This MOSFET must be selected with some care to ensure it will switch at the voltage levels provided by the MAX5904 and be able to sustain a constant flow of current and not incur large power losses.

The MAX5904 supplies a high signal between 3.6V and 5.8V. Since MOSFETS are charge controlled devices, one only has to ensure the gate threshold voltage is met and there is ample current supplied to guarantee a fast switching rate. For the chosen N-Channel MOSFET

(IRLML2502), the switching period was found to be .15ms which will be adequate for power supply line switching. Also important in MOSFET selection is the devices Current consumption, as MOSFETS have a resistance associated with them when enabled. With this particular device, the power loss was 0.005W and this related to a voltage drop of 0.02V, which is negligible for the voltage regulator.

### Voltage Regulation

The MAX8887 is a low drop out linear regulator able to reliably deliver a constant voltage (3.3V) up to a cut off point 0.2 mV (3.5V) above its designed output. This device also is equipped with a bypass capacitor that can be used to limit the device's noise on the ground plane, causing less distortion for the radio and sensors. For this device to work properly, the data sheet suggests that filtering capacitors be placed on both the inputs and the outputs

### ***5 volt line***

The 5V line will only power the onboard handheld radio and will not require a protection circuit. As it is better to power a failing radio and be short power for other systems rather than to limit power to the radio and not be able to receive or transmit. 5 Volts is generated from the battery voltage using a Max1790 dc-dc converter, which is capable of supplying 5V, 0.8A in the configuration shown in Figure 28.

There are several required components for this device to operate correctly; these include the 3.3 $\mu$ H inductor, a 4.7 $\Omega$  resistor and two bypass capacitors (.22 $\mu$ F and .68 $\mu$ F). The data sheet for the MAX1790 goes into detail regarding the selection of these components. As this device is outputting 5V, an external protection diode must be located before the output to the system to prevent current from flowing in reverse through the inductor and damaging the MAX1790 device. The diode chosen is the same Schottky diode chosen to protect the solar cells and is capable of handling the 500mA start-up current and fast switching requirements.



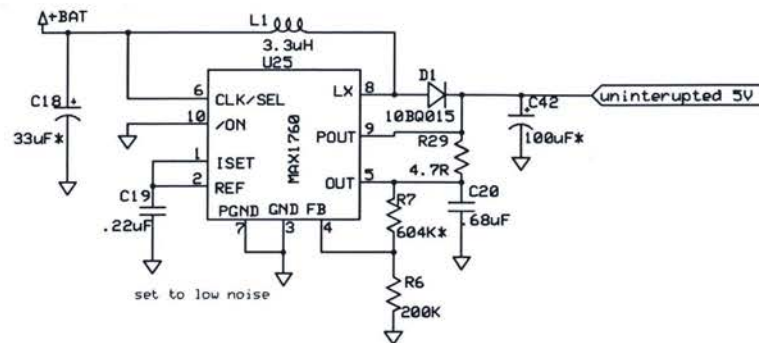


Figure 28: 5 volt step up and supply circuit

### 5.3.2 Controllable Feedlines

The controllable feed lines are very similar to the 3.3V uninterrupted line presented earlier. The major difference is the automatically resetting breaker is replaced with a latched breaker that requires a toggling from the microcontroller to reset if it is tripped. The other difference is with some of the supply lines which require a different voltage regulator capable of supplying a higher current. The circuits of these two variants are shown in Figure 29 and Figure 30.

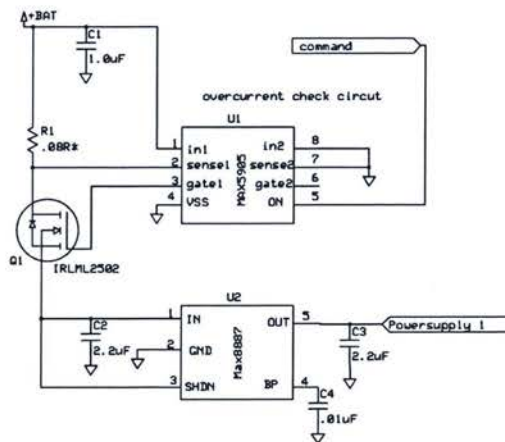


Figure 29: 3.3V 250 mA power supply lines (PS 1 and 3)

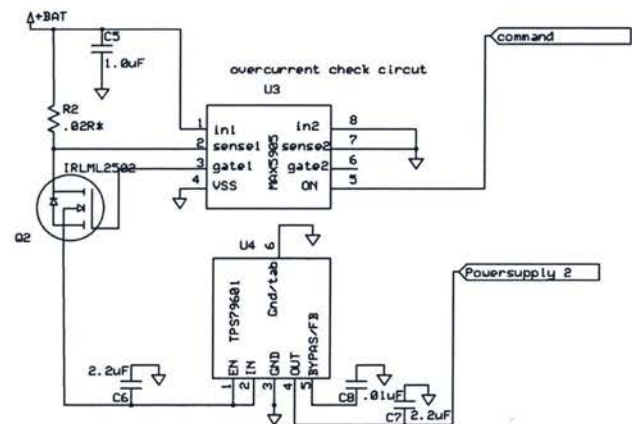


Figure 30: 3.3V 800 mA power supply line (PS 2 and 4)

#### Max5905 Circuit Breaker

Early in the design process for the satellite, there was a decision to limit most subsystems to 250mA at 3.3V; however, the actuators on the ACS and the heaters required around 800mA to operate effectively. Again, similar analysis to the Max5904 can be used to size the current

sensors for this device. It was found that  $0.08\Omega$  resistors can be used for supply lines requiring only 250mA, and  $0.02\Omega$  resistors can be used for controlling supply lines carrying 850mA.

### TPS79633 Voltage Regulator

The TPS79633 regulator was selected to replace the MAX8887 for supplies requiring a larger current draw, as the MAX8887 is only capable of sourcing 300mA. The main difference is the package design and the ability to source more current. The filtering and bypass capacitors are the same as the MAX8887 to simplify the ordering process.

### 5.3.3 Feedline Control Mechanism

The feed line control mechanism acts as a hardware AND gate, verifying the battery is beyond the low voltage threshold and the microcontroller is commanding the supply line on. The circuit for the feed line control mechanism is shown in Figure 31.

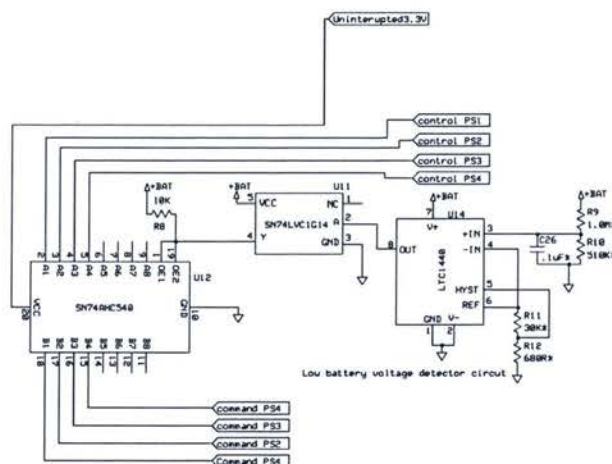


Figure 31: Feed line control mechanism

### Minimum Voltage Detection

Most lithium ion cells are capable of discharging well below their safe recharge voltage (typically 3V), so a protection circuit must be in place to ensure the battery does not over discharge. This problem is a bit of a paradox for the C&DH/COMM subsystem, which must always be on and functioning after deployment. To compromise, the minimum cut off voltage has been raised from 3V to 3.4V and the cut-off circuit will only power off non-critical subsystems. Assuming this measured voltage was the open circuit voltage (not the case as the battery is under load at all times), there will be at a very minimum of 5-10% of the battery's usable capacity left over to run

vital systems; this in turn equates to an absolute minimum of 20 minutes of run time for critical systems. These conservative estimates should prevent the battery from being over discharged and allow the C&DH/COMM subsystem to remain powered during the eclipsed periods of the orbit.

#### LTC1440 and SN74LVC1G14 Configuration

The LTC1440 is a comparator device. These devices are used to determine when a voltage has passed a certain threshold and will send a logic high signal once it has. The inverter (SN74LVC1G14) is used to switch the logic of the comparator around. The trigger point on the comparator is set according to the device's datasheet (This process is also shown in Eqs. (5-1 to 5-3)) and is set to trigger when the battery voltage drops below 3.4V. The comparator will stay triggered until the battery voltage increases to 3.65V. This was done to ensure the battery pack has time to recharge properly even without the aid of the microcontroller. The datasheet for the LTC1440 indicates a design process for selecting resistor values. The results of this process are presented in Table 56.

$$I_{\text{ref}} = \frac{V_{\text{HB}}}{2R_{12}} \quad 5-1$$

$$I_{\text{ref}} = \frac{V_{\text{ref}} - \frac{V_{\text{HB}}}{2}}{2R_{11}}$$

Solving for  $V_{\text{HB}}$

$$V_{\text{HB}} = \frac{V_{\text{ref}}}{\frac{R_{12}}{R_{11}} + 1/2} \quad 5-2$$

Solving for the trip point

$$\frac{R_9}{R_{10}} = \left[ \frac{V_{\text{trip}}}{V_{\text{ref}} + \frac{V_{\text{HB}}}{2}} - 1 \right] \quad 5-3$$

$$V_{\text{trip}} = \left( V_{\text{ref}} + \frac{V_{\text{HB}}}{2} \right) \left( \frac{R_9}{R_{10}} + 1 \right)$$

$$V_{\text{Hysteresis actual}} = \frac{V_{\text{HB}} V_{\text{trip}}}{V_{\text{ref}}}$$



### 5.3.4 Solar Cell Measurement and Power Transfer Bus

This part of the circuit has been designed to function with a wide variety of solar cells and is adaptable to different configurations for future missions. The input bus consists of six diode protected input ports and one unprotected crossover link to allow for charging on the ground or the grouping of multiple power subsystems in future missions.

Table 56: LTC 1440 resistor sizing table

Resistors selected (same as Fig.33)				
R10	R9	R11	R12	Vref
510000Ω	1000000Ω	300000Ω	6800Ω	1.182V
Output voltages				
Vtrip	Vhin	Vlow	Vhigh	
3.539V	0.079V	3.460V	3.618V	
VHB cannot exceed .05V Iref and Ibais should be as small as possible				
Vhb	Iref	Ibias		
0.026V	25.97E-9A	2.32E-6A		

#### Diode Protected Inputs

These inputs were designed to measure the current being produced by the solar cells on one side of the spacecraft utilizing a LT1787 current sensor. A 10bq015 schottky diode is used to protect the solar cell and the current sensor from reverse currents. These diodes will prevent up to 15V flowing in the reverse direction. The circuit is shown in Figure 32.

#### Sizing the Current Sensor

The candidate solar cell manufacturer indicates that the peak power of the cells occurs at 2.4 V per cell and 400mAhrs for the given size selected. It is planned that two cells in series will be used per side giving a maximum voltage and current around 4.8V and 400mAhrs, respectively.

#### Diode Selection

The schottky diode selected (10bq015) was chosen to minimize the voltage drop across the diode to maximize the power supplied to the charging bus. It was also chosen to prevent a maximum current of 800mA from traveling in reverse through solar cells shadowed and not producing power. This diode was utilized throughout the design to minimize ordering errors. The key characteristics of the chosen diode are shown in Table 57.

Table 57: Key Diode Characteristics

Max average forward current	1A
Max peak forward current	140 A
Max Forward voltage drop	.34V
Max reverse leakage current 25°C	.50 mA
Max Reverse leakage current 100°C	12 mA
Max Reverse DC voltage	15V (25V Peak)

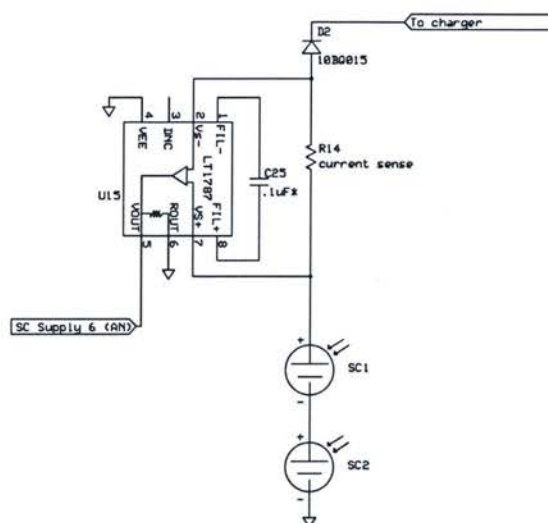


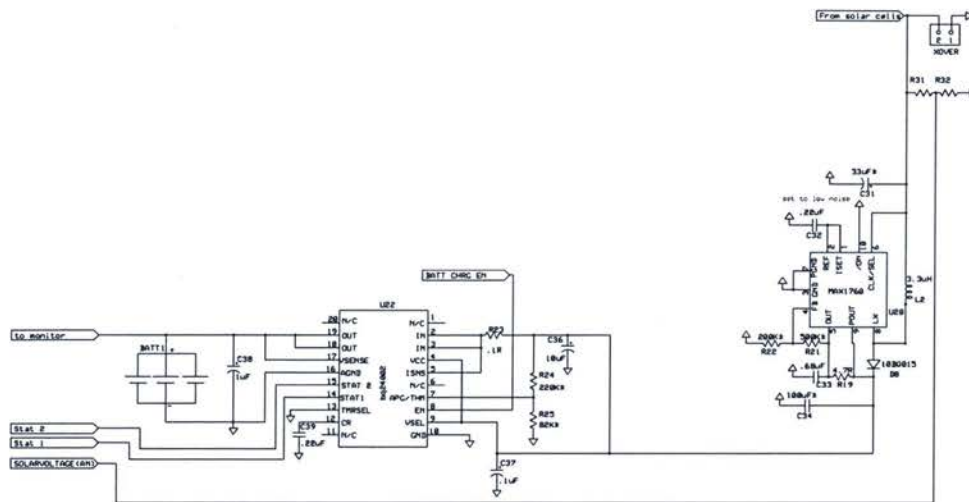
Figure 32: Two solar cell circuit

### 5.3.5 Battery Charger

The main component of the battery charging system is the bq24002 from Texas instruments. This device, along with a DC- DC boost voltage converter (MAX1760) and voltage divider, will charge the battery and supply current for the system when it is exposed to sunlight. The circuit is shown in Figure 33.

#### ***BQ24002 Monolithic Charger IC***

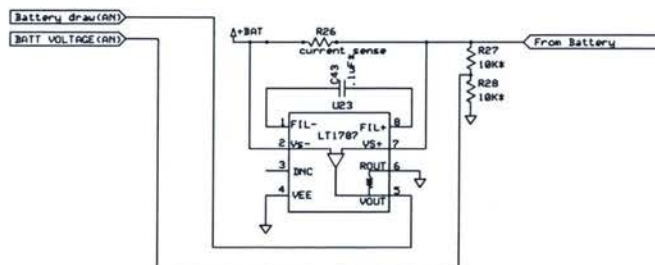
This device was selected amongst the many available on the market for its simplicity, ability to change charge current, indication of charge status features, and easy to solder TSSOP package. This device, however, has a slightly higher input voltage. This requires the use of a DC-DC converter to ensure power will transfer from the cells on non ideal orientations of the solar cells and the Sun. To ensure this device utilizes the most amount of available solar power and does not burn out the DC-DC voltage step up converter, the charge current will be limited to 0.5A.



**Figure 33: Battery charger circuit**

### 5.3.6 Battery Monitor

The battery monitor will consist of the subsystem microcontroller estimating the battery draw/recharge current over time. A current sensor (LT1787) will be used to directly measure the current. To help verify these estimates, the battery voltage will be measured using a simple voltage divider. The circuit for this is shown in Figure 34.



**Figure 34: Battery monitor circuit**

### 5.3.7 System Wide Interface

The power subsystem will be connected to the rest of the satellite using a standardized interface, which includes a 30 pin connector a bus switch and programming enable jumpers. Figure 35 shows the circuit for the systems interface.





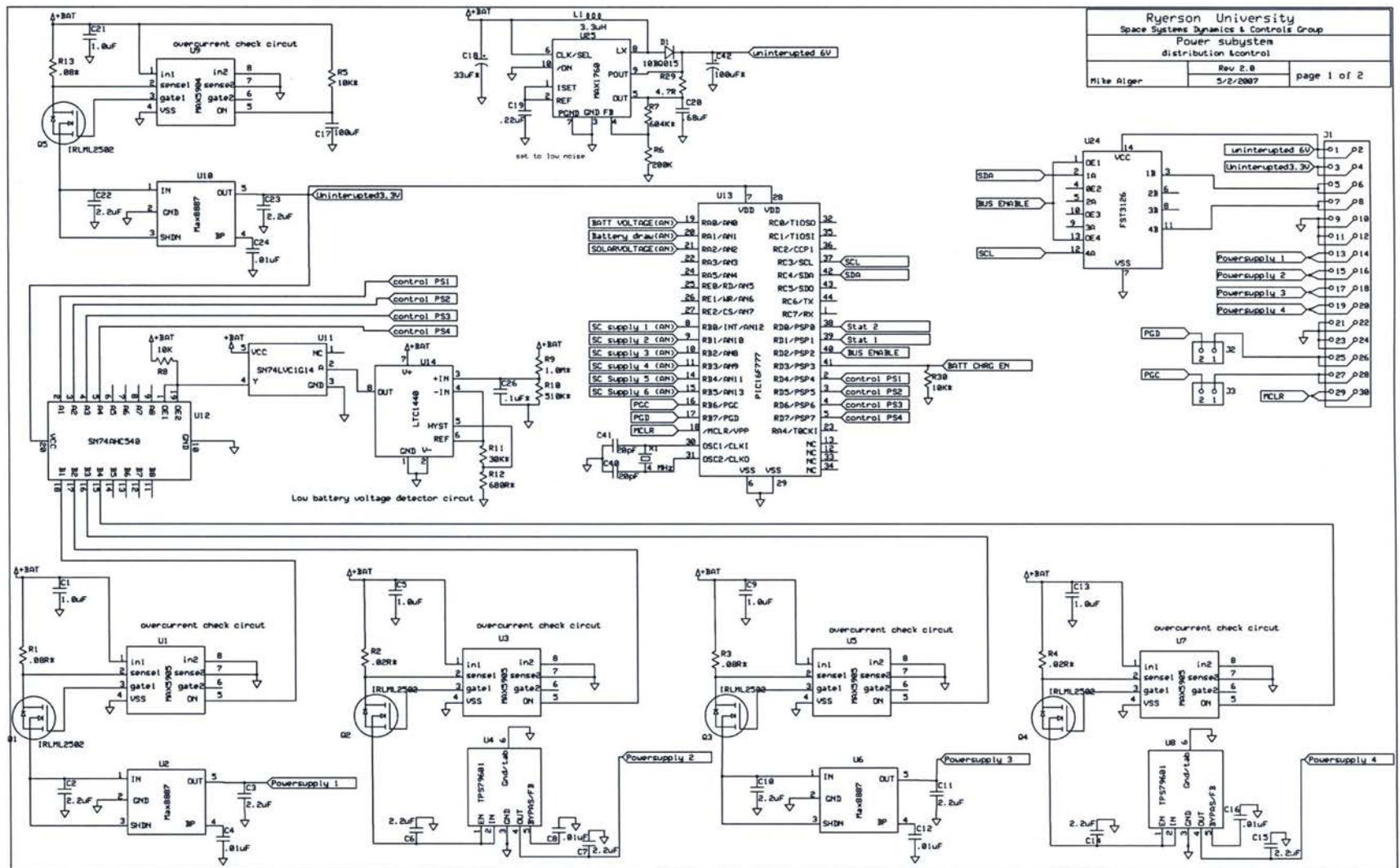
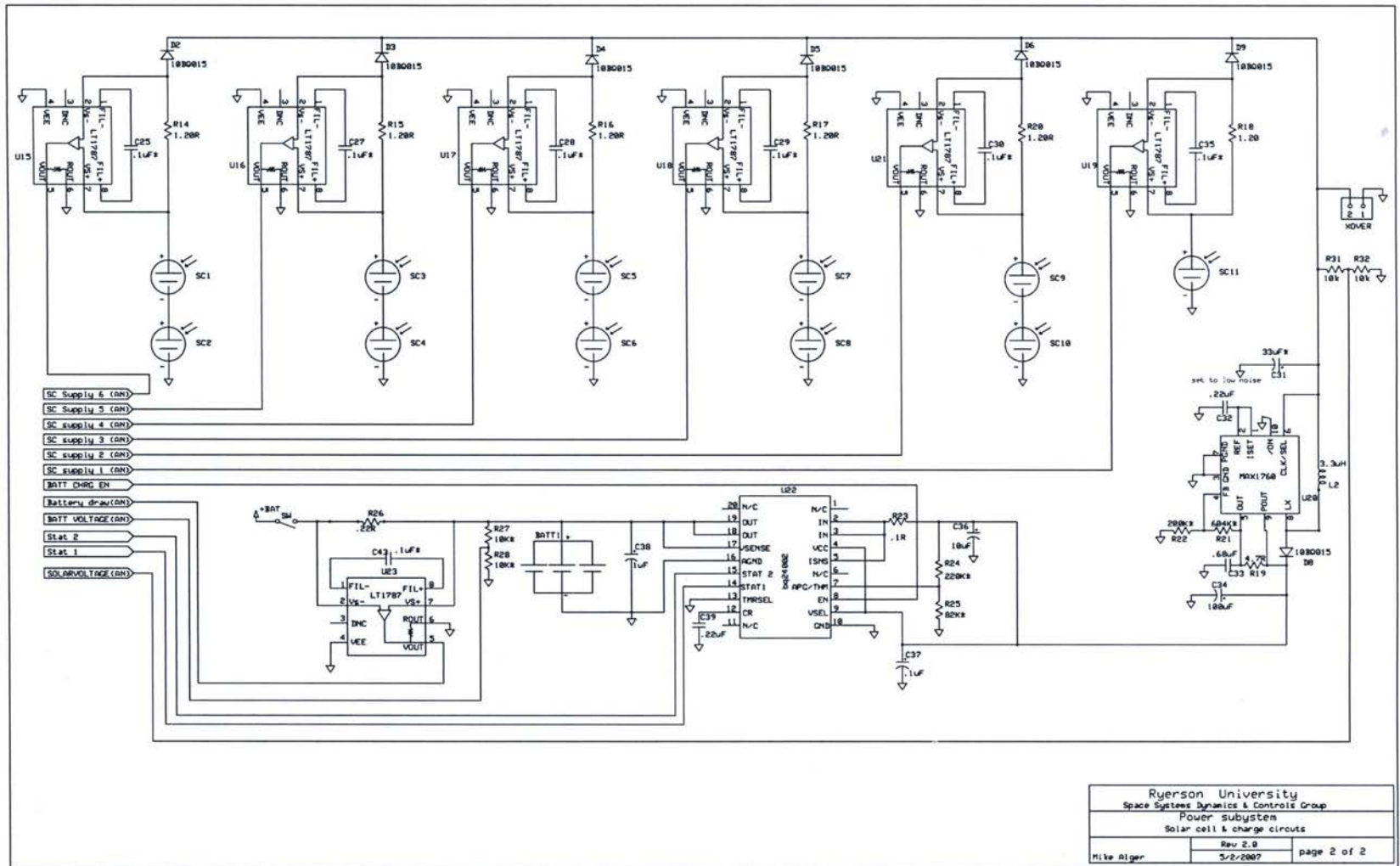


Figure 37: Power subsystem schematic



**Figure 38: Power subsystem schematic continued**



### 5.3.10 Physical Implementation

The power subsystem circuit must only occupy a single PCB card in the satellite. A battery will rest on a second blank PCB and will be connected to the first using the BATT1 input. Figure 39 illustrates the layout of the final power subsystem.

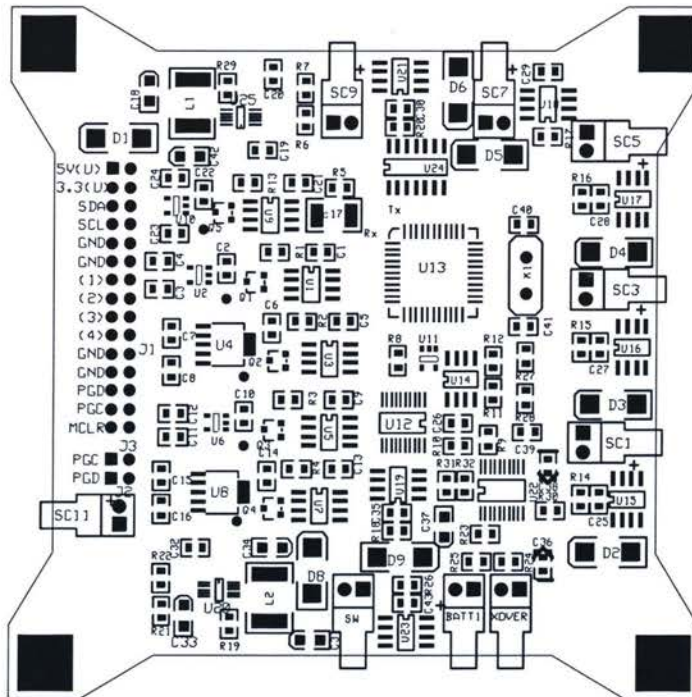
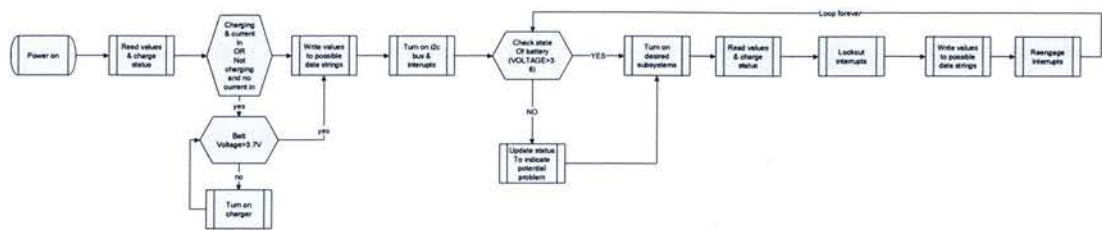


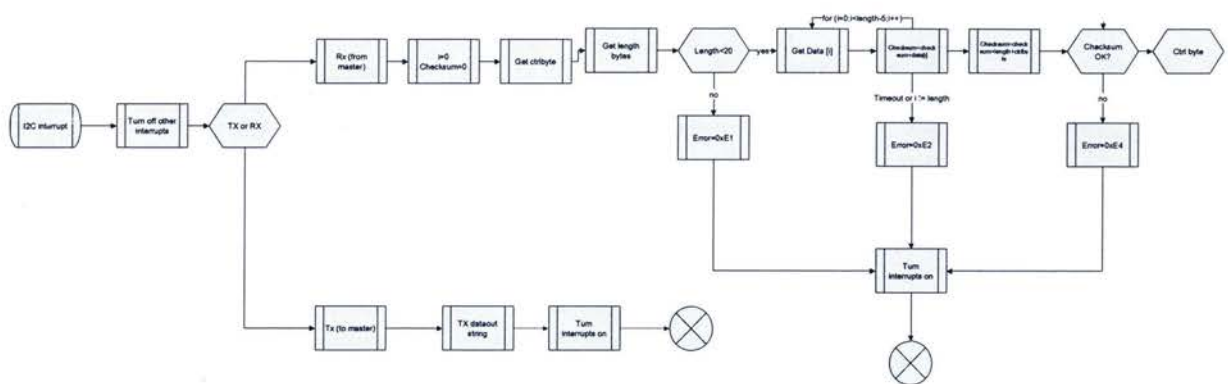
Figure 39: PCB layout

### 5.3.11 Firmware Design

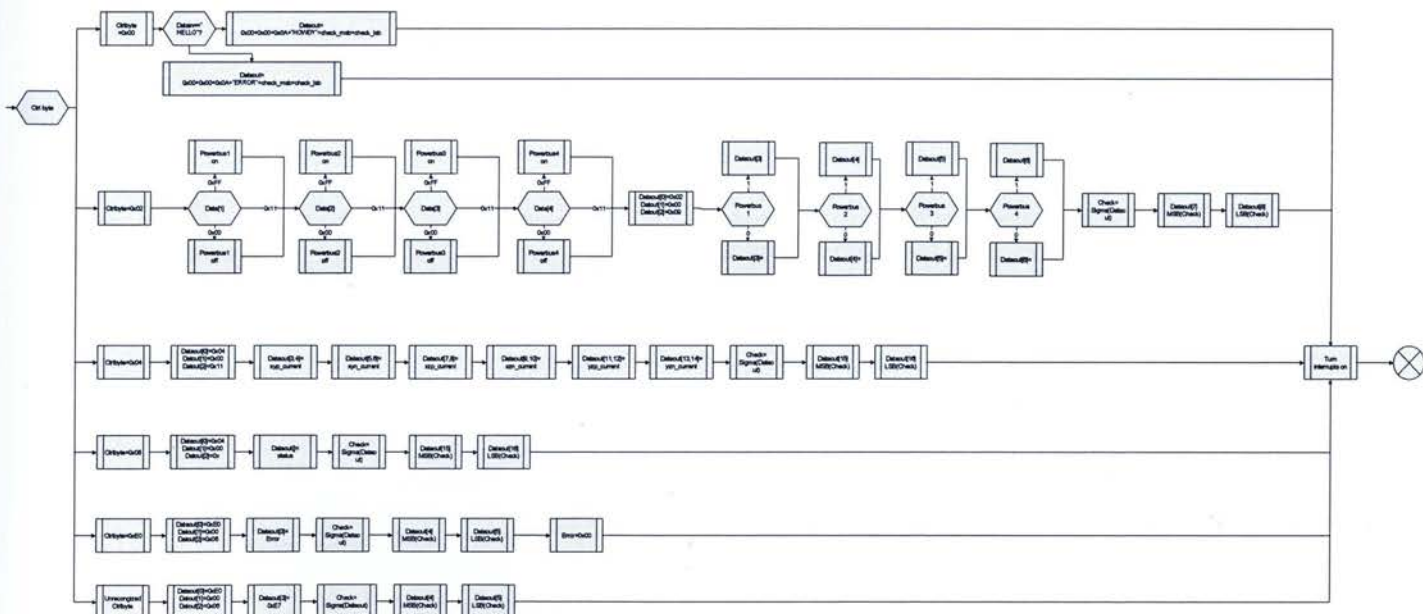
The firmware design consists of two main parts: the main continuous loop and the interrupt routine (Figure 40). The main loop will constantly measure the charge status, the current drawn from the solar cells and battery and write these values to registers accessible during an interrupt. The interrupt will then send data out to the system that requested it (C&DH or ADS). The interrupt will perform error checking on the incoming signal to ensure it is valid (Figure 41) and then, if the command is correct, it will then execute the appropriate case to load the data output variable (Figure 42).



**Figure 40: Firmware Design (Main loop)**



**Figure 41: Firmware design (Interrupt)**



**Figure 42: Firmware design (Interrupt) continued**

### 5.3.12 Commands to Control Power Subsystem

Commands to the power subsystem will be sent via I<sup>2</sup>C and will be formatted as shown in Section 5.1 (pg 61). At the time of writing, the power subsystem will respond to the commands listed in this subsection with the following responses.

#### Communications Test

This command simply tests if the I<sup>2</sup>C interface is working properly. Table 58 summarizes the command; Table 59 and Table 60 show the input and output of this command.

Table 58: Summary of the communications test command

Description	Communications interface testing.	description	Comm. IF test response.
Control byte	0x00	Control byte	0x00
packet length	0x000A	packet length	0x000A
DATA	HELLO	DATA	HOWDY
checksum	calculate	checksum	calculate

Table 59: Communications test command input

Ctrlbyte	length MSB	Length LSB	Data					Check MSB	Check LSB
0x00	0x00	0x0A	'H'	'E'	'L'	'L'	'O'	0x01	0x7E

Table 60: Communications test output

Ctrlbyte	length MSB	Length LSB	Data					Check MSB	Check LSB
0x00	0x00	0x0A	'H'	'O'	'W'	'D'	'Y'	0x01	0x95

#### Power Distribution Control

PS1 PS2, PS3, and PS4 correspond to the states of the controllable supply lines; if the supply needs to be turned on, a 0xFF is sent; if it needs to be turned off, a 0x11 is sent; if no change is desired, a 0x55 will be sent. The responses are similar, with a 0xFF indicating the supply line is on and 0x11 indicating the line is off. Table 61 summarizes this command and Table 62 to Table 63 show an example of this command.

Table 61: Summary of the Power distribution control command

Description	power distribution control -	Description	Power dist response.
Control byte	0x02	Control byte	0x02
packet length	0x0009	packet length	0x0009
DATA	0x??(Power bus one) 0x??(2) 0x??(3) 0x??(4)	DATA	0x?? 0x?? 0x?? 0x??
checksum	Calc	checksum	Calc



Table 62: Example input for the power control command

Ctrlbyte	length MSB	Length LSB	Data				Check MSB	Check LSB
0x02	0x00	0x09	0x55	0x11	0x55	0x55	0x01	0x1B

Table 63: Example output of the power control command

Ctrlbyte	length MSB	Length LSB	Data				Check MSB	Check LSB
0x02	0x00	0x09	0xFF	0x11	0xFF	0xFF	0x03	0x19

### Solar Cell Currents and Solar Cell Bus Voltage

This command will return the measured current and voltages as 10bit ADC readings (lossless and compact data format). This command is summarized in Table 64. Table 65 and Table 66 show the command and the structure of the response. The relationship relating the solar cell currents [A] to the ADC Reading is:

$$I_{sc} \cdot 064 \left( \frac{1024}{3.3} \right) = ADC_{SC}$$

And the relationship relating the voltage [V] to the ADC reading is

$$\frac{V_{scvoltage}}{2} \left( \frac{1024}{3.3} \right) = ADC_{SCvoltage}$$

Table 64: Summary of the solar Cell inquiry command

Description	Solar cells status	Description	6 current readings + 1 Bus Voltage reading
Control byte	0x04	Control byte	0x04
packet length	0x0005	packet length	0x0013
DATA		DATA	7 unsigned int (16b)
Checksum	calc	checksum	calc

Table 65: Input for the solar cell inquiry command

Ctrlbyte	length MSB	Length LSB	Check MSB	Check LSB
0x04	0x00	0x05	0x00	0x09

Table 66: Structure of the output for the solar cell inquiry command

Ctrlbyte	length MSB	Length LSB	SC1		SC2		SC3		SC4		SC5		SC6		Volt		Check MSB	Check LSB
0x04	0x00	0x13																

### Battery Charger Status

This command will return the status of the battery charger (one byte), the measured battery voltage (two bytes), and the measured battery draw (two bytes). The status of the monolithic charger is read off the port and a high condition on of the status pin is indicated with half a byte with a 0xF as a high, and a low is indicated with a 0x0. The upper 4 bits of the byte represent status pin one and the lower 4 bits representing status pin two (e.g., 0xF0 indicates a high on status 1 and a low on status 2). A summary of this command can be found in Table 67, and its command and response can be seen in Table 68 and Table 69. The battery current drawn [A] and voltage [V] can be related to the reported ADC values using the following relationships:

$$I_{\text{batt draw}} \cdot 0.64 \left( \frac{1024}{3.3} \right) = \text{ADC}_{\text{batt draw}}$$

$$\frac{V_{\text{batt}}}{2} \left( \frac{1024}{3.3} \right) = \text{ADC}_{\text{batt}}$$

Table 67: Summary of the Battery inquiry command

Description	Solar cells status	Description	6 current readings + 1 Bus Voltage reading
Control byte	0x04	Control byte	0x04
packet length	0x0005	packet length	0x0013
DATA		DATA	1 byte + 2 unsigned int (16b)
checksum	Calc	checksum	calc

Table 68: Input for the battery inquiry command

Ctrlbyte	length MSB	Length LSB	Check MSB	Check LSB
0x08	0x00	0x05	0x00	0x11

Table 69: Structure of the battery inquiry output

Ctrlbyte	length MSB	Length LSB	Status	Batt volt	Batt Draw	Check MSB	Check LSB
0x02	0x00	0x0A					

## 5.4 Attitude Control Subsystem

The design of the ACS subsystem will integrate two of the three coils on a single board, along with the H-bridge controllers and the microcontroller

### 5.4.1 H-Bridge

An H-Bridge was chosen to control the magnetic coils, as it can vary the amount of Voltage (and indirectly current). The H-Bridge chosen is the MPC1751a which is a single monolithic device capable of driving its own internal MOSFETS to drive separate coils. These devices are typically used in small stepper motor control in terrestrial applications. To control a coil with this device, there needs to be two inputs from the external controller, these being labelled for ease as: direction (DIR) and duty cycle (PWM). Because there is a natural inductance in the coil being used to change the spacecrafts attitude, pulse width modulation can be effectively used to set the coil's current. All that needs to be determined is the coil's true inductance and a minimum frequency can be established.

### 5.4.2 Microcontroller

The pulse width modulation (PWM) controller on the microcontroller is a fairly efficient way of varying the duty cycle (time on / time off) to the H-bridge. PWM works by setting a duty cycle and an overall frequency. These values must be within the range of the H-Bridges response range (<200 KHz) and faster than the minimum frequency of the coil. Failing to do so will cause the H-bridge to be unable to switch the coils fast enough or cause the output of the coil to become discrete. The circuit used for a magnetic torquer only ACS subsystem is found in Figure 43.

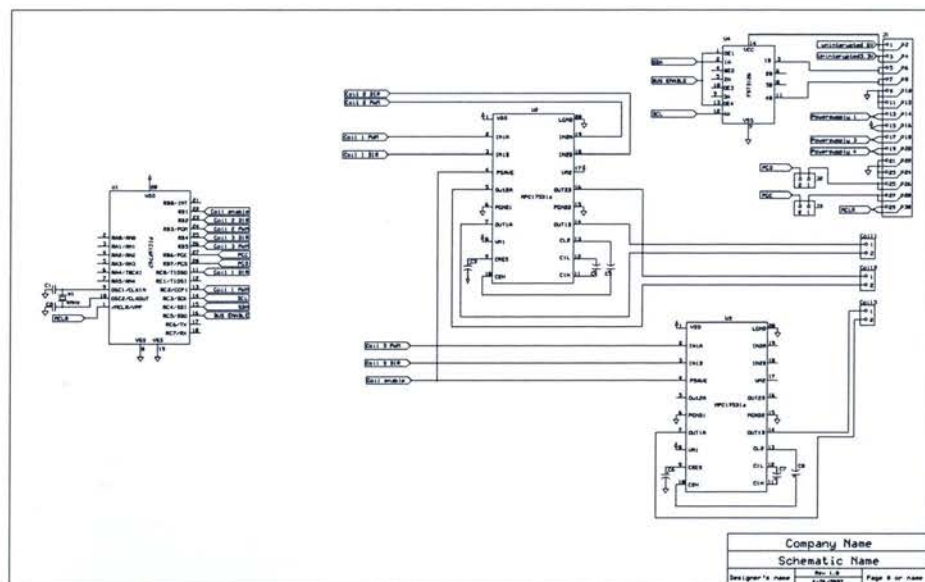


Figure 43: Schematic for the ACS subsystem (magnetic torquers only)



### 5.4.3 Physical Implementation

The following figure illustrates the layout of the ACS subsystem.

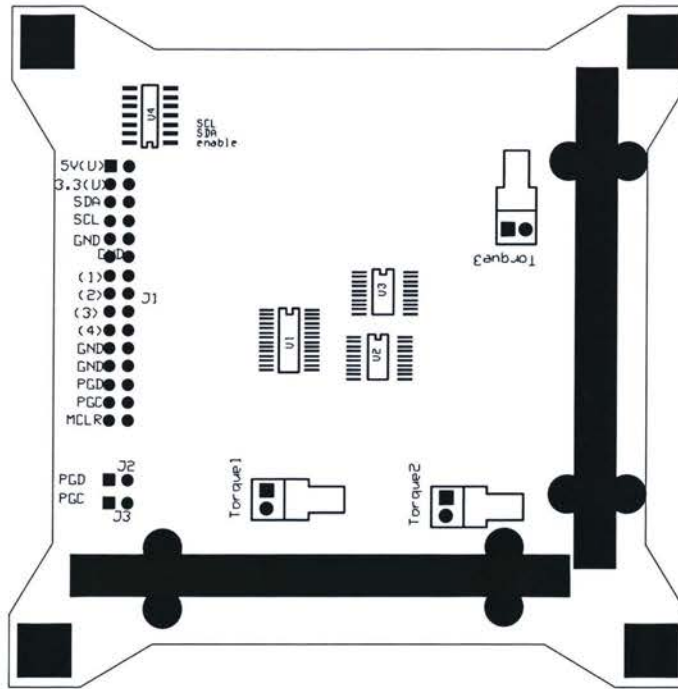


Figure 44: PCB layout (preliminary)

## 5.5 Control Law Development

### 5.5.1 Rotation Sequences

The following Rotation sequence is used to develop the control laws illustrated in further sections. This development of the Euler rotation sequence will follow the conventions presented in Kumar (2006).

$$\mathbf{R}_{bo} = \mathbf{R}_1(\gamma)\mathbf{R}_2(\phi)\mathbf{R}_3(\alpha) \quad 5-4$$

$$\mathbf{R}_{bo} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 5-5$$

$$\mathbf{R}_{bo} = \begin{bmatrix} \cos\alpha\cos\gamma & \sin\alpha\cos\phi & -\sin\phi \\ \cos\alpha\sin\phi\sin\gamma - \sin\alpha\cos\gamma & \sin\alpha\sin\phi\sin\gamma + \cos\alpha\cos\gamma & \cos\phi\sin\gamma \\ \cos\alpha\sin\phi\cos\gamma + \sin\alpha\sin\gamma & \sin\alpha\sin\phi\cos\gamma - \cos\alpha\sin\gamma & \cos\phi\cos\gamma \end{bmatrix} \quad 5-6$$

### 5.5.2 Spacecraft Dynamics

The standard vector notation for the dynamics of a rigid spacecraft is expressed as:

$$\boldsymbol{\tau} = I\dot{\boldsymbol{\omega}}_b + \boldsymbol{\omega}_b \times I\boldsymbol{\omega}_b \quad 5-7$$

This vector equation can then be expanded to the standard Euler-Newton equations of motion:

$$\begin{aligned} \tau_x &= I_{xx}\dot{\omega}_x - (I_{yy} - I_{zz})\omega_y\omega_z \\ \tau_y &= I_{yy}\dot{\omega}_y - (I_{zz} - I_{xx})\omega_z\omega_x \\ \tau_z &= I_{zz}\dot{\omega}_z - (I_{xx} - I_{yy})\omega_x\omega_y \end{aligned} \quad 5-8$$

For ease of development of the control laws, it is convenient to express time varying parameters (expressed as  $\dot{q}$  and  $\ddot{q}$ ) with ones that vary with the orbital position (expressed as  $q'$  and  $q''$ ). It can be shown that the relationship between time varying parameters and orbital position varying parameters is as follows;

$$\begin{aligned} \dot{q} &= \frac{\partial q}{\partial t} = \frac{\partial q}{\partial \theta} \frac{\partial \theta}{\partial t} = q' \dot{\theta} \\ \ddot{q} &= \frac{\partial^2 q}{\partial t^2} = \frac{\partial(q' \dot{\theta})}{\partial t} = q'' \dot{\theta}^2 + q' \ddot{\theta} \\ \text{Where } q' &= \frac{\partial q}{\partial \theta} \end{aligned} \quad 5-9$$

For a circular orbit these relations reduce down to:

$$\begin{aligned} \dot{q} &= q' \omega_o \\ \ddot{q} &= q'' \omega_o^2 \end{aligned} \quad 5-10$$

Using a 3-2-1 rotation sequence as shown in Kumar 2006, the angular velocity terms can be expressed in terms of rates of change of body angles as follows:

$$\begin{aligned} \omega_x &= -(\dot{\theta} + \dot{\alpha})\sin\phi + \dot{\gamma} \\ \omega_y &= (\dot{\theta} + \dot{\alpha})\cos\phi\sin\gamma + \dot{\phi}\cos\gamma \\ \omega_z &= (\dot{\theta} + \dot{\alpha})\cos\phi\cos\gamma + \dot{\phi}\sin\gamma \end{aligned} \quad 5-11$$

Angular acceleration can then be expressed as follows:

$$\dot{\omega}_{bo} = \begin{pmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{pmatrix} = \begin{pmatrix} \ddot{\gamma} - (\ddot{\alpha} + \ddot{\theta})\sin(\phi) - \dot{\phi}(\dot{\alpha} + \dot{\theta})\cos(\phi) \\ (\ddot{\alpha} + \ddot{\theta})\cos(\phi)\sin(\gamma) - \dot{\phi}(\dot{\alpha} + \dot{\theta})\sin(\phi)\sin(\gamma) + \dot{\gamma}(\dot{\alpha} + \dot{\theta})\cos(\phi)\cos(\gamma) + \ddot{\phi}\cos(\gamma) - \dot{\phi}\dot{\gamma}\sin(\gamma) \\ (\ddot{\alpha} + \ddot{\theta})\cos(\gamma)\cos(\phi) - \dot{\phi}(\dot{\alpha} + \dot{\theta})\cos(\gamma)\sin(\phi) - \dot{\gamma}(\dot{\alpha} + \dot{\theta})\sin(\gamma)\cos(\phi) - \ddot{\phi}\sin(\gamma) - \dot{\phi}\dot{\gamma}\sin(\gamma) \end{pmatrix} \quad 5-12$$

Using Eq. (5-8 to 5-12) the normalized and coupled equations of motion for a satellite in a circular orbit can be expressed as:

$$\begin{aligned} & \alpha'' \cos \phi \cos \gamma - \phi'' \sin \gamma - (1 - k_{xz} + k_{yz})(1 + \alpha')\phi' \sin \phi \cos \gamma \\ & - (1 + k_{xz} - k_{yz})(1 + \alpha')\gamma' \cos \phi \sin \gamma + \phi'\gamma' \cos \gamma \\ & + (k_{xz} - k_{yz})(1 + \alpha')^2 \sin \phi \cos \phi \sin \gamma = \frac{\tau_z}{I_z \omega_o} \end{aligned} \quad 5-13$$

$$\begin{aligned} & k_{yz}\alpha'' \cos \phi \sin \gamma + k_{yz}\phi'' \cos \gamma - (1 - k_{xz} + k_{yz})(1 + \alpha')\phi' \sin \phi \sin \gamma \\ & - (1 - k_{xz} - k_{yz})[(1 + \alpha')\gamma' \cos \phi \cos \gamma - \phi'\gamma' \sin \gamma] \\ & + (1 - k_{xz})(1 + \alpha')^2 \sin \phi \cos \phi \cos \gamma = \frac{\tau_y}{I_z \omega_o} \end{aligned} \quad 5-14$$

$$\begin{aligned} & -k_{xz}\alpha'' \sin \phi + k_{xz}\gamma'' + [(1 - k_{yz}) \cos 2\gamma - k_{xz}](1 + \alpha')\phi' \cos \phi \\ & + (1 - k_{yz})[(1 + \alpha')^2 \cos^2 \phi - \phi'^2] \sin \gamma \cos \gamma = \frac{\tau_x}{I_z \omega_o} \end{aligned} \quad 5-15$$

Where:

$$k_{xz} = \frac{I_{xx}}{I_{zz}} = \frac{1 - k_1}{1 - k_1 k_2}$$

$$k_{yz} = \frac{I_{yy}}{I_{zz}} = \frac{1 - k_2}{1 - k_1 k_2}$$

And  $k_1$  and  $k_2$  can also be found from:

$$k_1 = \frac{I_{zz} - I_{xx}}{I_{yy}}; k_2 = \frac{I_{zz} - I_{yy}}{I_{xx}}$$

If the controller is only to take into account the effect of gravity gradient to control its actuators, the gravity gradient force will have to be modeled into the previous equations. Gravity gradient can be expressed with the following vector equation:

$$\tau_b = 3 \frac{\mu}{R^3} \mathbf{c}_3 \times \mathbf{I} \mathbf{c}_3 \quad 5-16$$

And this equation can be expanded to its individual parts expressed in the body frame as:



$$\tau_b \approx \begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} -3 \frac{\mu}{R^3} (I_{yy} - I_{zz}) (\cos \alpha \sin \phi \cos \gamma + \sin \alpha \sin \gamma) (\cos \alpha \sin \phi \sin \gamma - \sin \alpha \cos \gamma) \\ -3 \frac{\mu}{R^3} (I_{zz} - I_{xx}) (\cos \alpha \sin \phi \cos \gamma + \sin \alpha \sin \gamma) \cos \alpha \cos \phi \\ -3 \frac{\mu}{R^3} (I_{xx} - I_{yy}) (\cos \alpha \sin \phi \sin \gamma - \sin \alpha \cos \gamma) \cos \alpha \cos \phi \end{pmatrix} \quad 5-17$$

Combining Eq. (5-13 to 5-15) and Eq. (5-17), the equations of motion expressed can be expressed as follows:

$$\begin{aligned} & \alpha'' \cos \phi \cos \gamma - \phi'' \sin \gamma - (1 - k_{xz} + k_{yz})(1 + \alpha')\phi' \sin \phi \cos \gamma \\ & - (1 + k_{xz} - k_{yz})(1 + \alpha')\gamma' \cos \phi \sin \gamma + \phi' \gamma' \cos \gamma \\ & + (k_{xz} - k_{yz})(1 + \alpha')^2 \sin \phi \cos \phi \sin \gamma \\ & + 3(k_{xz} - k_{yz})(\cos \alpha \sin \phi \sin \gamma + \sin \alpha \cos \gamma) \cos \alpha \cos \phi = 0 \end{aligned} \quad 5-18$$

$$\begin{aligned} & k_{yz}\alpha'' \cos \phi \sin \gamma + k_{yz}\phi'' \cos \gamma - (1 - k_{xz} + k_{yz})(1 + \alpha')\phi' \sin \phi \sin \gamma \\ & - (1 - k_{xz} - k_{yz})[(1 + \alpha')\gamma' \cos \phi \cos \gamma - \phi' \gamma' \sin \gamma] \\ & + (1 - k_{xz})(1 + \alpha')^2 \sin \phi \cos \phi \cos \gamma \\ & + 3(1 - k_{xz})(\cos \alpha \sin \phi \cos \gamma + \sin \alpha \sin \gamma) \cos \alpha \cos \phi = 0 \end{aligned} \quad 5-19$$

$$\begin{aligned} & -k_{xz}\alpha'' \sin \phi + k_{xz}\gamma'' + [(1 - k_{yz}) \cos 2\gamma - k_{xz}](1 + \alpha')\phi' \cos \phi \\ & + (1 - k_{yz})[(1 + \alpha')^2 \cos^2 \phi - \phi'^2] \sin \gamma \cos \gamma \\ & - 3(1 - k_{yz})(\cos \alpha \sin \phi \cos \gamma + \sin \alpha \sin \gamma) (\cos \alpha \sin \phi \sin \gamma - \sin \alpha \cos \gamma) = 0 \end{aligned} \quad 5-20$$

### 5.5.3 Linear Control Law

For a point of comparison with the adaptive sliding mode controller, a linear control law can be compared. The development of this controller requires linearized versions of Eq. (5-18 to 5-20), and these can be expressed as:

$$\alpha'' + \frac{3(k_2 - k_1)}{1 - k_1 k_2} \alpha = 0 \quad 5-21$$

$$\phi'' + (1 - k_1)\gamma' + 4k_1\phi = 0 \quad 5-22$$

$$\gamma'' + (k_2 - 1)\phi' + k_2\gamma = 0 \quad 5-23$$

These equations can be arranged into the standard  $\dot{X} = AX + BU$  form:

$$\begin{bmatrix} \alpha' \\ \alpha'' \\ \phi' \\ \phi'' \\ \gamma' \\ \gamma'' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{3(k_2 - k_1)}{1 - k_1 k_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -4k_1 & 0 & 0 & k_1 - 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -k_2 & -k_2 & 0 \end{bmatrix} \begin{bmatrix} \alpha' \\ \alpha'' \\ \phi' \\ \phi'' \\ \gamma' \\ \gamma'' \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_\alpha \\ U_\phi \\ U_\gamma \end{bmatrix} \quad 5-24$$

From Eq. (5-24), a linear control law can be solved for using pole placement, knowing the desired performance characteristics. Kumar (2006) shows the development of a simple linear controller and the final outcome is shown in Eq. 5-25.

$$\begin{aligned} u_\alpha &= -2 \zeta_\alpha \omega_\alpha \delta\alpha' - \omega_\alpha^2 \delta\alpha - \frac{3(k_2 - k_1)}{1 - k_1 k_2} \delta\alpha \\ u_\phi &= -2 \zeta_\phi \omega_\phi \delta\phi' - \omega_\phi^2 \delta\phi + (1 - k_1)\delta\gamma' + 4k_1\delta\phi \\ u_\gamma &= -2 \zeta_\gamma \omega_\gamma \delta\gamma' - \omega_\gamma^2 \delta\gamma + (k_2 - 1)\delta\phi' + k_2\delta\gamma \end{aligned} \quad 5-25$$

The controller from Eq. (5-25) with  $\zeta_\alpha = \zeta_\phi = \zeta_\gamma = 1$  and  $\omega_\alpha = \omega_\phi = \omega_\gamma = 2$  was simulated with the initial conditions shown in Table 70. Figure 45 shows this controller reaching the desired attitude in about half an orbit, and Figure 46 shows how much momentum would have to be stored if reaction wheels were used, finally the torques required are shown in Figure 47.

Table 70: initial conditions for PD controller

Principal MOI [kg · m <sup>2</sup> ]	Stored momentum [mNms]	Angle[roll pitch yaw)	Rate $\begin{bmatrix} \alpha \\ \phi \\ \gamma \end{bmatrix} \left[ \frac{rad}{s} \right]$	Desired angle
$\begin{bmatrix} 0.002 \\ 0.003 \\ 0.005 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 70^\circ \\ 70^\circ \\ 70^\circ \end{bmatrix}$	$\begin{bmatrix} 2\omega_o \\ \omega_o \\ \frac{\omega_o}{5} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

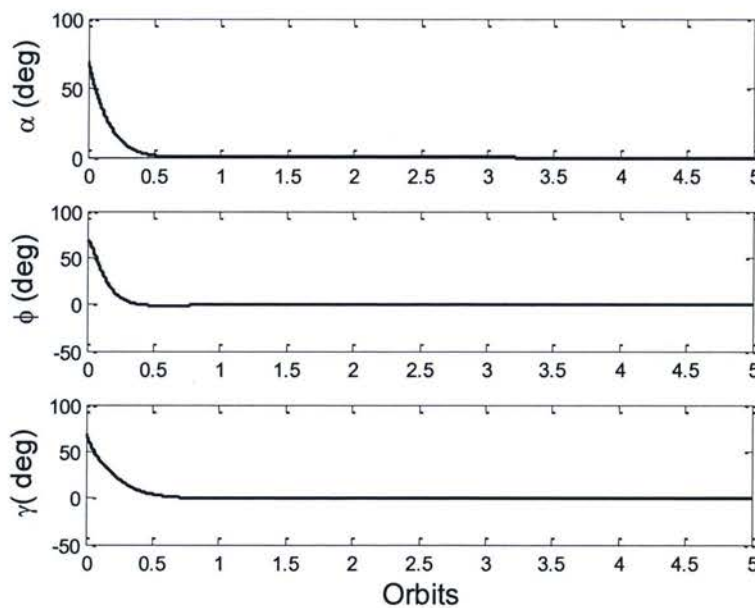


Figure 45: Angular response PD controller.

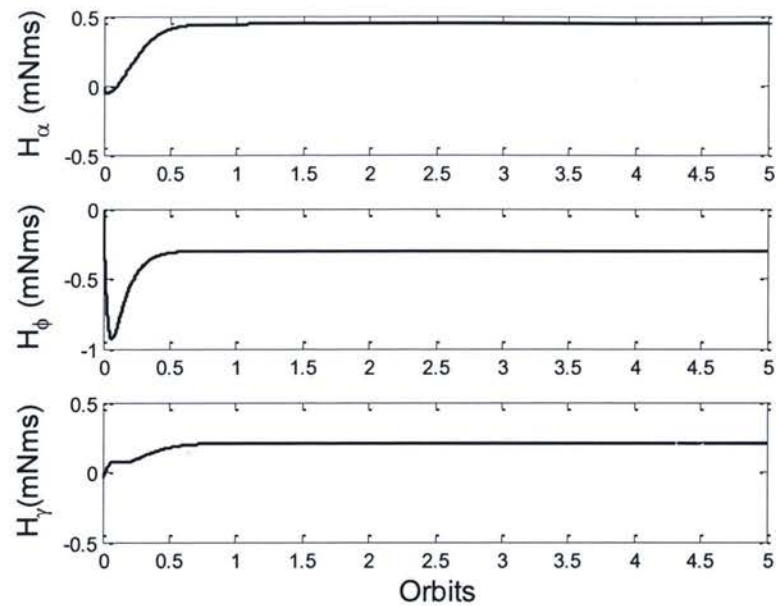


Figure 46: Momentum storage for a simple PD controller

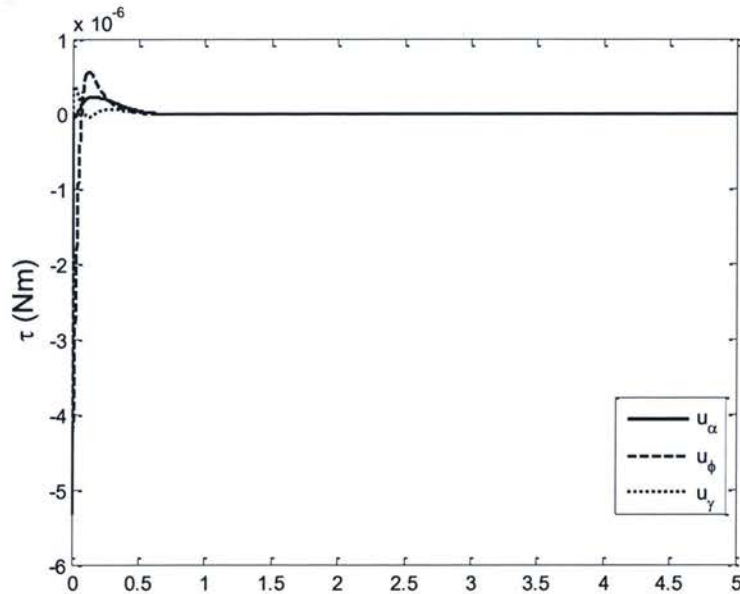


Figure 47: Torques for PD controller

#### 5.5.4 Magnetic PD Controller

In a simplified model, the magnetic field vectors can be calculated at any point using the following equation:



$$\vec{B}(R) = \frac{R_e^3 H_0}{R^3} (3(\hat{i}_m \hat{i}_R) \hat{i}_R - \hat{i}_m) \quad 5-26$$

This equation can be further expanded out and expressed in terms of the orbital frame as follows:

$$\vec{B}_o = \frac{R_e^3 H_0}{R^3} \begin{Bmatrix} -2 \sin \theta \sin i_m \\ \cos \theta \sin i_m \\ \cos i_m \end{Bmatrix} \begin{Bmatrix} i_o \\ j_o \\ k_o \end{Bmatrix} \quad 5-27$$

And finally the magnetic field vectors can be expressed in terms of the body frame:

$$\vec{B}_B = R_{Bo} \vec{B}_o \quad 5-28$$

$$\vec{B}_o = R_{Bo} \left( \frac{R_e^3 H_0}{R^3} \begin{Bmatrix} -2 \sin \theta \sin i_m \\ \cos \theta \sin i_m \\ \cos i_m \end{Bmatrix} \begin{Bmatrix} i_o \\ j_o \\ k_o \end{Bmatrix} \right) \quad 5-29$$

Torque from a magnetic torquer is expressed as follows:

$$\vec{\tau}_B = \vec{M}_B \times \vec{B}_B \quad 5-30$$

As only the components of  $\vec{M}$  that are perpendicular to  $\vec{B}$  create a torque, the following relationship is used to obtain the optimal torque to stabilize the spacecraft. This approach was also utilized by Makovec (2001) and Wisiewski (2004).

$$\vec{M}_{\text{optimal}} \rightarrow \vec{M}_B: \vec{M}_B = \frac{\vec{M}_{\text{optimal}} \times \vec{B}_B}{\|\vec{B}_B\|} \quad 5-31$$

Eq.(5-30) and Eq.(5-31) can then be expanded out to

$$\begin{aligned} \tau_y = \tau_{mx} &= \frac{1}{\|\vec{B}_B\|} \left( -(B_{By}^2 + B_{Bz}^2) \tilde{M}_{\text{optimal}x} + B_{Bx} B_{By} \tilde{M}_{\text{optimal}y} + B_{Bx} B_{Bz} \tilde{M}_{\text{optimal}z} \right) \\ \tau_\phi = \tau_{my} &= \frac{1}{\|\vec{B}_B\|} \left( B_{Bx} B_{By} \tilde{M}_{\text{optimal}x} - (B_{Bx}^2 + B_{Bz}^2) \tilde{M}_{\text{optimal}y} + B_{By} B_{Bz} \tilde{M}_{\text{optimal}z} \right) \\ \tau_\alpha = \tau_{mz} &= \frac{1}{\|\vec{B}_B\|} \left( B_{Bx} B_{Bz} \tilde{M}_{\text{optimal}x} + B_{By} B_{Bz} \tilde{M}_{\text{optimal}y} - (B_{Bx}^2 + B_{By}^2) \tilde{M}_{\text{optimal}z} \right) \end{aligned} \quad 5-32$$

A simple PD controller can be applied to Eq. (5-32) with  $\vec{M}_B$  as the control torque as shown in the following equation:

$$\tilde{M}_{\text{optimalx}} = -k_{1\gamma}(\gamma) - k_{2\gamma}(\gamma')$$

$$\tilde{M}_{\text{optimaly}} = -k_{1\phi}(\phi) - k_{2\phi}(\phi')$$

5-33

$$\tilde{M}_{\text{optimalz}} = -k_{1\alpha}(\alpha) - k_{2\alpha}(\alpha')$$

And the gains for this controller were selected to be as follows:

$$k_{1\gamma} = k_{1\phi} = k_{1\alpha} = 0.0000225$$

$$k_{2\gamma} = k_{2\phi} = k_{2\alpha} = 0.0000801$$

With this control law and the previously selected gains, a test case was run using the initial conditions found in Table 71. The plot in Figure 48 shows that this particular controller with these gains will settle the spacecrafts attitude at the desired orientation in about 3-4 orbits. Figure 49 show that this controller produces the required amount of torque without exceeding the available magnetic moment of the previously designed actuator.

Table 71: Initial conditions for magnetic PD Controller

Principal MOI [ $kg \cdot m^2$ ]	Stored momentum [ $mNm s$ ]	Angle[roll pitch yaw]	Rate $\begin{bmatrix} \alpha' \\ \phi' \\ \gamma' \end{bmatrix} \left[ \frac{rad}{s} \right]$	Desired angle
$\begin{bmatrix} 0.002 \\ 0.003 \\ 0.005 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 70^\circ \\ 70^\circ \\ 70^\circ \end{bmatrix}$	$\begin{bmatrix} 2\omega_0 \\ \omega_0 \\ 5 \\ \omega_0 \\ 5 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

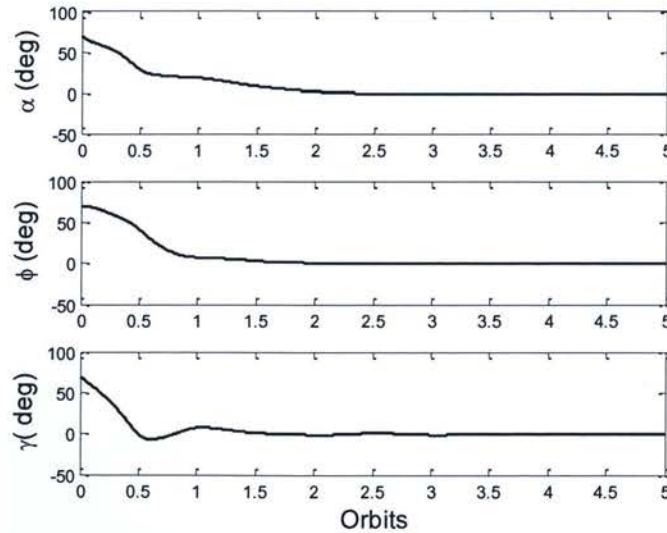


Figure 48: Magnetic PD controller response

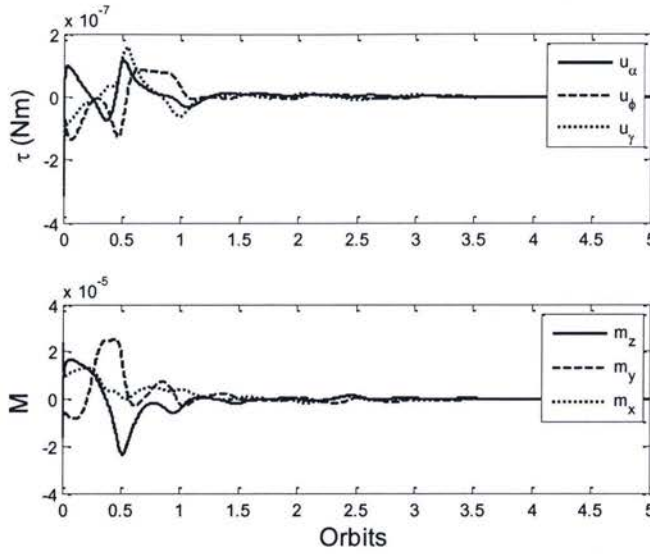


Figure 49: Magnetic PD controller input

### 5.5.5 Sliding Mode Adaptive Controller

To ease the development of the adaptive control law, it is easier to express the spacecraft nonlinear equations of motion in Lagrangian form, as this will provide a symmetric inertia matrix with  $\alpha'', \phi'', \gamma''$  as coefficients. Although this can be done from the energy equations directly, it is easiest in this case to use the following relationship and Eqs. (5-18 to 5-20):

$$\begin{aligned} L_\alpha &= E_\alpha \cos \phi \cos \gamma + E_\phi \cos \phi \sin \gamma - E_\gamma \sin \gamma \\ L_\phi &= -E_\alpha \sin \gamma + E_\phi \cos \gamma \\ L_\gamma &= E_\gamma \end{aligned} \quad 5-34$$

The full derivation of these equations can be found in Kumar and the final result is reproduced in Appendix E. To design a sliding mode controller, first the sliding planes are selected as follows:

$$\begin{aligned} S_1 &= c_1 \alpha' + c_2 \alpha \\ S_2 &= c_3 \phi' + c_4 \phi \\ S_3 &= c_5 \gamma' + c_6 \gamma \end{aligned} \quad 5-35$$

Where  $c_1 - c_6$  positive constants (control gains) of the sliding are plane and will specify the performance of the controller.

The adaptive parameters in this control law will be the inertias of the spacecraft. Doing so will allow the controller to adapt to changing moments of inertia of the spacecraft (caused due to



antenna deployment, fuel sloshing etc.). It will also adapt the controller to minor actuator failures, and misalignments.

Applying Eq. (5-34) to Eqs. (5-18)(5-19)(5-20) and then dividing by  $I_{zz}$  will allow the expressions to be written in the following form:

$$\begin{aligned} C_\alpha \alpha'' + C_{\alpha\phi} \phi'' + C_{\alpha\gamma} \gamma'' + f_\alpha &= U_\alpha \\ C_{\phi\alpha} \alpha'' + C_\phi \phi'' + C_{\phi\gamma} \gamma'' + f_\phi &= U_\phi \\ C_{\gamma\alpha} \alpha'' + C_{\gamma\phi} \phi'' + C_\gamma \gamma'' + f_\gamma &= U_\gamma \end{aligned} \quad 5-36$$

This will allow the design of a candidate Lyapunov function with the following form:

$$V = \frac{1}{2} C_\alpha S_1^2 + \frac{1}{2} C_\phi S_2^2 + \frac{1}{2} C_\gamma S_3^2 \quad 5-37$$

The derivative of V gives:

$$V' = C_\alpha S_1 S_1' + C_\phi S_2 S_2' + C_\gamma S_3 S_3' \quad 5-38$$

$$V' = C_\alpha S_1 (c_1 \alpha'' + c_2 \alpha') + C_\phi S_2 (c_3 \phi'' + c_4 \phi') + C_\gamma S_3 (c_5 \gamma'' + c_6 \gamma') \quad 5-39$$

Substituting Eq. (5-36) into Eq. (5-39) and rearranging, the following relationship can be shown:

$$V' = S_1 (F_\alpha + c_1 U_\alpha) + S_2 (F_\phi + c_3 U_\phi) + S_3 (F_\gamma + c_5 U_\gamma) \quad 5-40$$

Where;

$$F_\alpha = (-C_{\alpha\phi} \phi'' - C_{\alpha\gamma} \gamma'' - f_\alpha) c_1 + C_\alpha c_2 \alpha' \quad 5-41$$

$$F_\phi = (-C_{\phi\alpha} \alpha'' - C_{\phi\gamma} \gamma'' - f_\phi) c_3 + C_\phi c_4 \phi' \quad 5-42$$

$$F_\gamma = (-C_{\gamma\alpha} \alpha'' - C_{\gamma\phi} \phi'' - f_\gamma) c_5 + C_\gamma c_6 \gamma' \quad 5-43$$

These equations can be expressed in terms of the spacecraft's principal moments of inertia which will be changed by the controller:

$$F_\alpha = \theta_1 I_{xx} + \theta_2 I_{yy} + \theta_3 I_{zz} \quad 5-44$$

$$F_\phi = \theta_4 I_{xx} + \theta_5 I_{yy} + \theta_6 I_{zz} \quad 5-45$$

$$F_\gamma = \theta_7 I_{xx} + \theta_8 I_{yy} + \theta_9 I_{zz} \quad 5-46$$

The adaptation of the inertial parameters is handled with another adaptive function based on the previous sliding mode controller

$$V_a = V + \frac{1}{2\gamma_1} \tilde{I}_{xx}^2 + \frac{1}{2\gamma_2} \tilde{I}_{yy}^2 + \frac{1}{2\gamma_3} \tilde{I}_{zz}^2 \quad 5-47$$

$$V'_a = V' + \frac{1}{\gamma_1} \tilde{I}_{xx} \hat{I}'_{xx} + \frac{1}{\gamma_2} \tilde{I}_{yy} \hat{I}'_{yy} + \frac{1}{\gamma_3} \tilde{I}_{zz} \hat{I}'_{zz} \quad 5-48$$

Eq.(5-48) expands out to:

$$V'_a = S_1(\theta_1 I_{xx} + \theta_2 I_{yy} + \theta_3 I_{zz} + c_1 U_\alpha) + S_2(\theta_4 I_{xx} + \theta_5 I_{yy} + \theta_6 I_{zz} + c_3 U_\phi) + S_3(\theta_7 I_{xx} + \theta_8 I_{yy} + \theta_9 I_{zz} + c_5 U_\gamma) + \frac{1}{\gamma_1} \tilde{I}_{xx} \hat{I}'_{xx} + \frac{1}{\gamma_2} \tilde{I}_{yy} \hat{I}'_{yy} + \frac{1}{\gamma_3} \tilde{I}_{zz} \hat{I}'_{zz} \quad 5-49$$

Knowing that:

$$\begin{aligned} \tilde{I}_{xx} &= \hat{I}_{xx} - I_{xx} \\ \tilde{I}_{yy} &= \hat{I}_{yy} - I_{yy} \\ \tilde{I}_{zz} &= \hat{I}_{zz} - I_{zz} \end{aligned} \quad 5-50$$

$V'_a$  Becomes:

$$\begin{aligned} V'_a &= S_1(\theta_1 \hat{I}_{xx} + \theta_2 \hat{I}_{yy} + \theta_3 \hat{I}_{zz} + c_1 U_\alpha) + S_2(\theta_4 \hat{I}_{xx} + \theta_5 \hat{I}_{yy} + \theta_6 \hat{I}_{zz} + c_3 U_\phi) \\ &+ S_3(\theta_7 \hat{I}_{xx} + \theta_8 \hat{I}_{yy} + \theta_9 \hat{I}_{zz} + c_5 U_\gamma) - S_1(\theta_1 \tilde{I}_{xx} + \theta_2 \tilde{I}_{yy} + \theta_3 \tilde{I}_{zz}) \\ &- S_2(\theta_4 \tilde{I}_{xx} + \theta_5 \tilde{I}_{yy} + \theta_6 \tilde{I}_{zz}) - S_3(\theta_7 \tilde{I}_{xx} + \theta_8 \tilde{I}_{yy} + \theta_9 \tilde{I}_{zz}) \\ &+ \frac{1}{\gamma_1} \tilde{I}_{xx} \hat{I}'_{xx} + \frac{1}{\gamma_2} \tilde{I}_{yy} \hat{I}'_{yy} + \frac{1}{\gamma_3} \tilde{I}_{zz} \hat{I}'_{zz} \end{aligned} \quad 5-51$$

Setting the following terms from Eq. (5-51) to zero

$$-S_1 \theta_1 \tilde{I}_{xx} - S_2 \theta_4 \tilde{I}_{xx} - S_3 \theta_7 \tilde{I}_{xx} + \frac{1}{\gamma_1} \tilde{I}_{xx} \hat{I}'_{xx} = 0 \quad 5-52$$

$$-S_1 \theta_2 \tilde{I}_{yy} - S_2 \theta_5 \tilde{I}_{yy} - S_3 \theta_8 \tilde{I}_{yy} + \frac{1}{\gamma_2} \tilde{I}_{yy} \hat{I}'_{yy} = 0 \quad 5-53$$

$$-S_1 \theta_3 \tilde{I}_{zz} - S_2 \theta_6 \tilde{I}_{zz} - S_3 \theta_9 \tilde{I}_{zz} + \frac{1}{\gamma_3} \tilde{I}_{zz} \hat{I}'_{zz} = 0 \quad 5-54$$

Allows the estimated parameters to be found as:

$$\hat{I}'_{xx} = \gamma_1 (S_1 \theta_1 + S_2 \theta_4 + S_3 \theta_7) \quad 5-55$$

$$\hat{I}'_{yy} = \gamma_2 (S_1 \theta_2 - S_2 \theta_5 - S_3 \theta_8) \quad 5-56$$

$$\hat{I}'_{zz} = \gamma_3 (S_1 \theta_3 - S_2 \theta_6 - S_3 \theta_9) \quad 5-57$$

To ensure that  $V'_a$  remains negative, definite and asymptotically stable the remaining terms are set to:

$$(\theta_1 \hat{I}_{xx} + \theta_2 \hat{I}_{yy} + \theta_3 \hat{I}_{zz} + c_1 U_\alpha) = -\eta \operatorname{sgn}(S_1) - K_1 S_1 \quad 5-58$$

$$(\theta_4 \hat{I}_{xx} + \theta_5 \hat{I}_{yy} + \theta_6 \hat{I}_{zz} + c_3 U_\phi) = -\eta \operatorname{sgn}(S_2) - K_2 S_2 \quad 5-59$$

$$(\theta_4 \hat{I}_{xx} + \theta_5 \hat{I}_{yy} + \theta_6 \hat{I}_{zz} + c_5 U_\gamma) = -\eta \operatorname{sgn}(S_3) - K_3 S_3 \quad 5-60$$

This leads to the final control law in terms of angular acceleration with respect to the orbital velocity:

$$U_\alpha = -\frac{\theta_1 \hat{I}_{xx} + \theta_2 \hat{I}_{yy} + \theta_3 \hat{I}_{zz} + \eta \operatorname{sgn}(S_1) + K_1 S_1}{c_1} \quad 5-61$$

$$U_\phi = -\frac{\theta_4 \hat{I}_{xx} + \theta_5 \hat{I}_{yy} + \theta_6 \hat{I}_{zz} + \eta \operatorname{sgn}(S_2) + K_2 S_2}{c_3} \quad 5-62$$

$$U_\gamma = -\frac{\theta_7 \hat{I}_{xx} + \theta_8 \hat{I}_{yy} + \theta_9 \hat{I}_{zz} + \eta \operatorname{sgn}(S_3) + K_3 S_3}{c_5} \quad 5-63$$

To relate these commanded accelerations to torque from an actuator (e.g., reaction wheel) the following relationships are used;

$$\tau_\alpha(t) = U_\alpha(\theta_{orb}) I_{zz} \omega_{orb}^2 = -\frac{\theta_1 \hat{I}_{xx} + \theta_2 \hat{I}_{yy} + \theta_3 \hat{I}_{zz} + \eta \operatorname{sgn}(S_1) + K_1 S_1}{c_1} I_{zz} \omega_{orb}^2 \quad 5-64$$

$$\tau_\phi(t) = U_\phi(\theta_{orb}) I_{zz} \omega_{orb}^2 = -\frac{\theta_4 \hat{I}_{xx} + \theta_5 \hat{I}_{yy} + \theta_6 \hat{I}_{zz} + \eta \operatorname{sgn}(S_2) + K_2 S_2}{c_3} I_{zz} \omega_{orb}^2 \quad 5-65$$

$$\tau_\gamma(t) = U_\gamma(\theta_{orb}) I_{zz} \omega_{orb}^2 = -\frac{\theta_7 \hat{I}_{xx} + \theta_8 \hat{I}_{yy} + \theta_9 \hat{I}_{zz} + \eta \operatorname{sgn}(S_3) + K_3 S_3}{c_5} I_{zz} \omega_{orb}^2 \quad 5-66$$

### ***Initial Attitude Control Using Sliding Mode Technique***

To prove this controller is adaptable to different spacecraft both the ideal spacecraft MOI and the worst-case MOI were tested with the same initial conditions. The initial conditions for the ideal MOI are shown in Table 72 and the worst case in Table 73. It can be seen in both Figure 50 and Figure 53 that the controller stabilizes the system in less than half an orbit. Figure 52 and Figure 55 both show that the angular momentum used in this maneuver is lower than that of the previously designed reaction wheel. Figure 52 and Figure 54 show that the torque required to settle the system is well within the capabilities of the motor. Additionally those figures show that the adaptive gains settle down after 0.5 orbits.



Table 72: Initial conditions ideal cube

Principal MOI [ $kg \cdot m^2$ ]	Stored momentum [ $mNms$ ]	Angle[roll pitch yaw]	Rate $\begin{bmatrix} \alpha \\ \phi \\ \gamma \end{bmatrix} \left[ \frac{rad}{s} \right]$	Desired angle
$\begin{bmatrix} 0.001666 \\ 0.001666 \\ 0.001666 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 70^\circ \\ 70^\circ \\ 70^\circ \end{bmatrix}$	$\begin{bmatrix} 2\omega_0 \\ \omega_0 \\ \frac{\omega_0}{5} \\ \frac{\omega_0}{5} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Table 73: initial conditions worst case MOI

Principal MOI [ $kg \cdot m^2$ ]	Stored momentum [ $mNms$ ]	Angle[roll pitch yaw]	Rate $\begin{bmatrix} \alpha \\ \phi \\ \gamma \end{bmatrix} \left[ \frac{rad}{s} \right]$	Desired angle
$\begin{bmatrix} 0.002 \\ 0.003 \\ 0.005 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 70^\circ \\ 70^\circ \\ 70^\circ \end{bmatrix}$	$\begin{bmatrix} 2\omega_0 \\ \omega_0 \\ \frac{\omega_0}{5} \\ \frac{\omega_0}{5} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

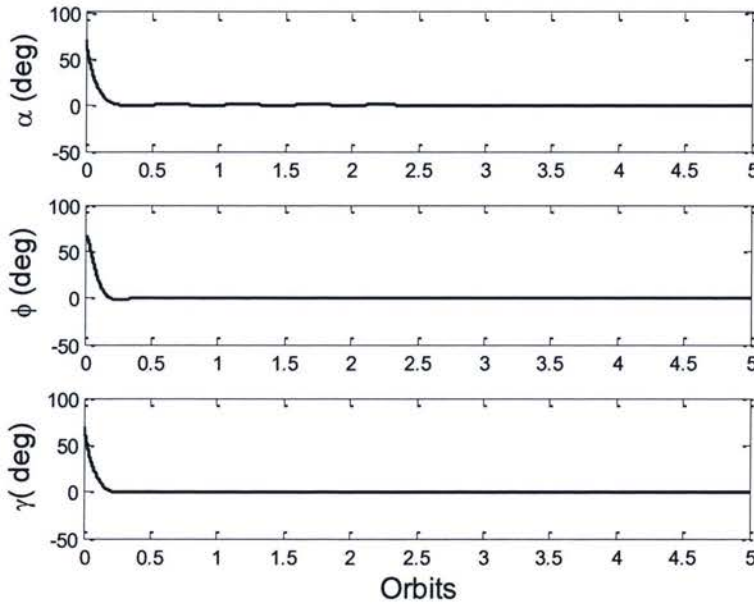


Figure 50: Attitude response (ideal cube MOI)

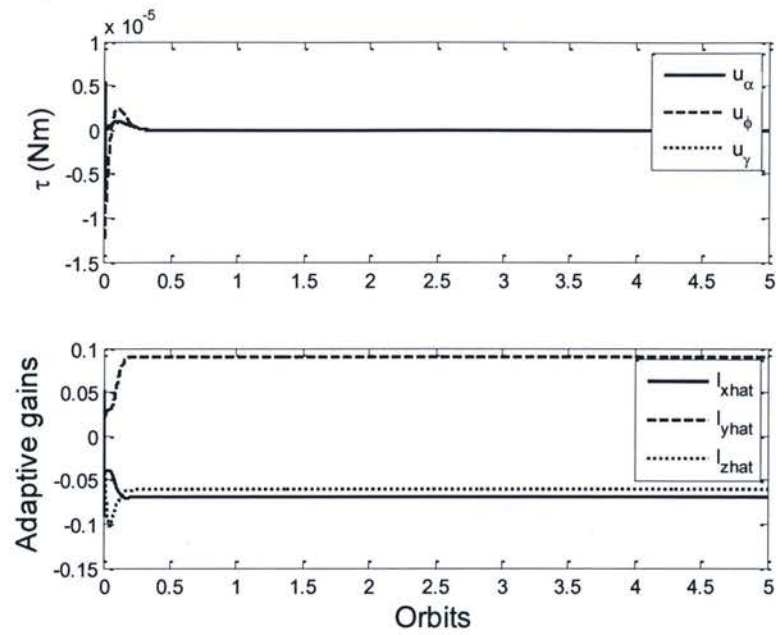


Figure 51: Torque and adaptive gains (ideal cube MOI)

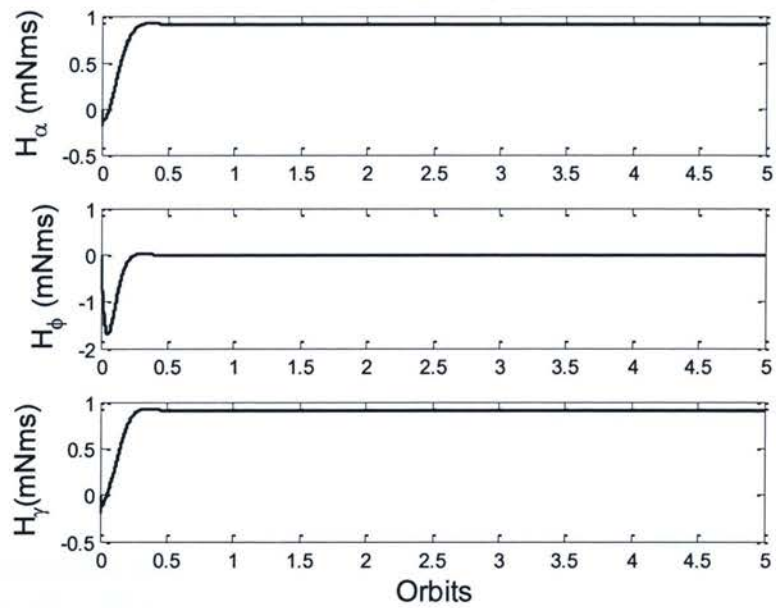


Figure 52: Reaction wheel storage momentum (ideal cube MOI)

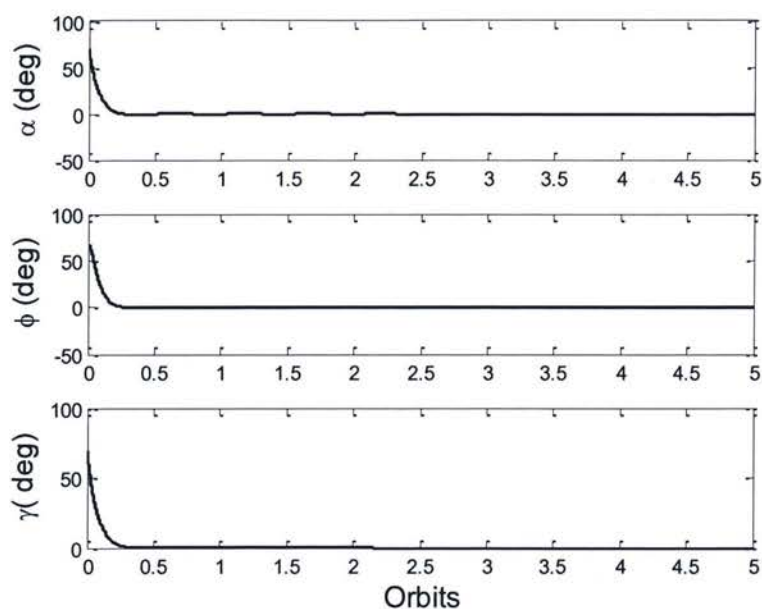


Figure 53: Attitude response (worst case MOI)

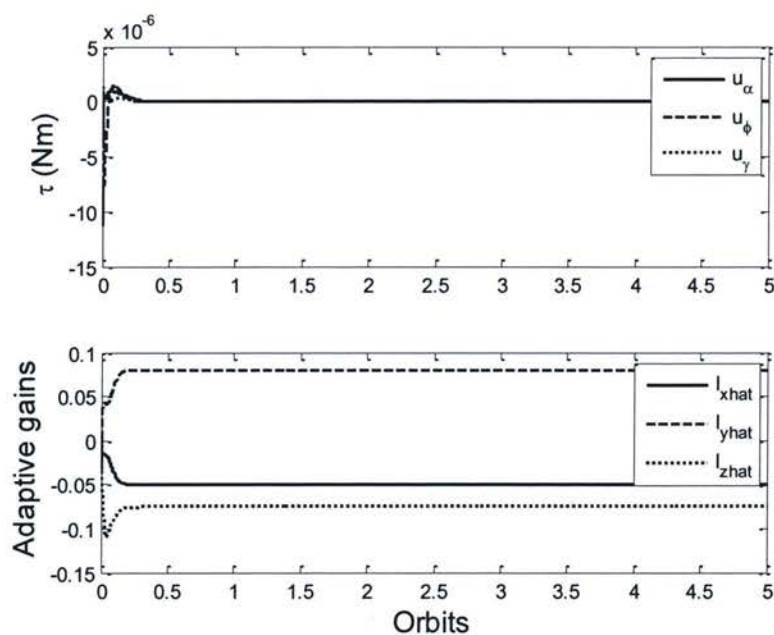


Figure 54: Torque and adaptive gains (worst case MOI)



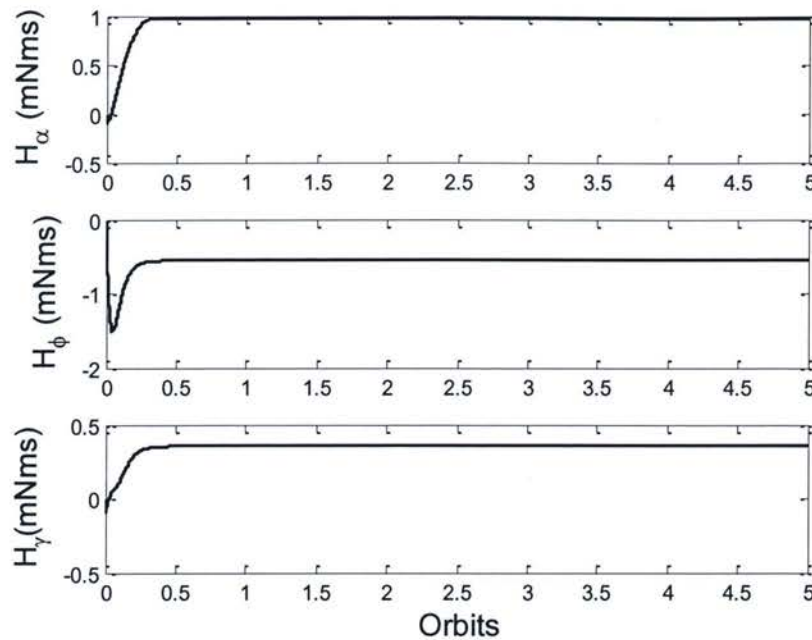


Figure 55: Reaction wheel storage momentum (worst case MOI)

### Attitude Maneuver Response

A second test case was run to test this control laws ability to track a desired attitude angle, again similar to the previous case both the ideal cube and worst case MOI simulated with the initial conditions found in Table 74 and Table 75. Again Figure 56 and Figure 59 show the control law settling to the desired attitude in under half an orbit. Figure 57 and Figure 60 show the torque required is within the capability of the actuator, more interestingly the adaptive gains both seem to settle to a constant slope. Figure 58 shows the momentum settling down to a final value as expected (as the moments of inertia are equal) and Figure 61 shows a minor increase in the angular momentum as expected as a constant torque is needed to overcome a gravity gradient torque.

Table 74: Attitude maneuver initial conditions (ideal cube)

Principal MOI [ $kg \cdot m^2$ ]	Stored momentum [ $mNms$ ]	Angle[roll pitch yaw)	Rate $\begin{bmatrix} \alpha \\ \phi \\ \gamma \end{bmatrix} \left[ \frac{rad}{s} \right]$	Desired angle
$\begin{bmatrix} 0.001666 \\ 0.001666 \\ 0.00166 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0^\circ \\ 0^\circ \\ 0^\circ \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 45^\circ \\ 45^\circ \\ 45^\circ \end{bmatrix}$

Table 75: Attitude maneuver initial conditions (worst case MOI)

Principal MOI [ $kg \cdot m^2$ ]	Stored momentum [ $mNms$ ]	Angle[roll pitch yaw]	Rate $\begin{bmatrix} \alpha \\ \phi \\ \gamma \end{bmatrix} \left[ \frac{rad}{s} \right]$	Desired angle
$\begin{bmatrix} 0.002 \\ 0.003 \\ 0.005 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0^\circ \\ 0^\circ \\ 0^\circ \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 45^\circ \\ 45^\circ \\ 45^\circ \end{bmatrix}$

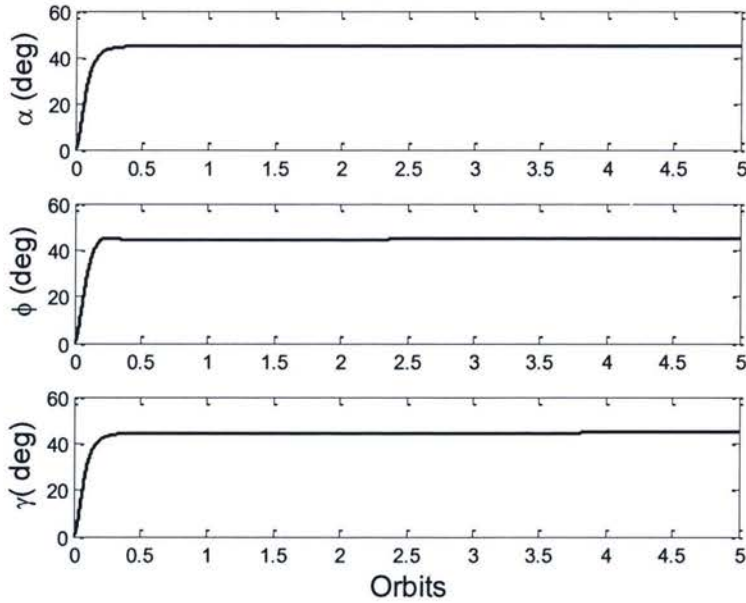


Figure 56: Attitude maneuver response (ideal cube MOI)

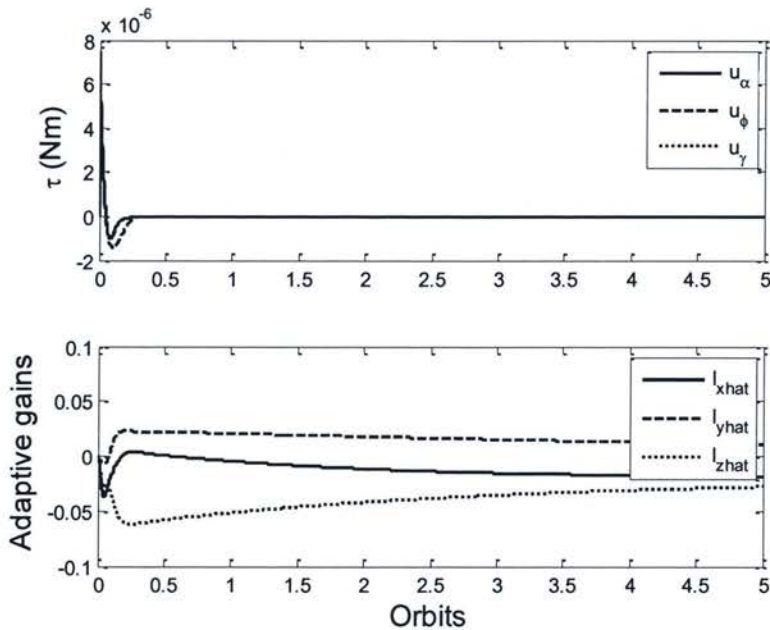


Figure 57: Torque and adaptive gains while maneuvering (ideal cube MOI)

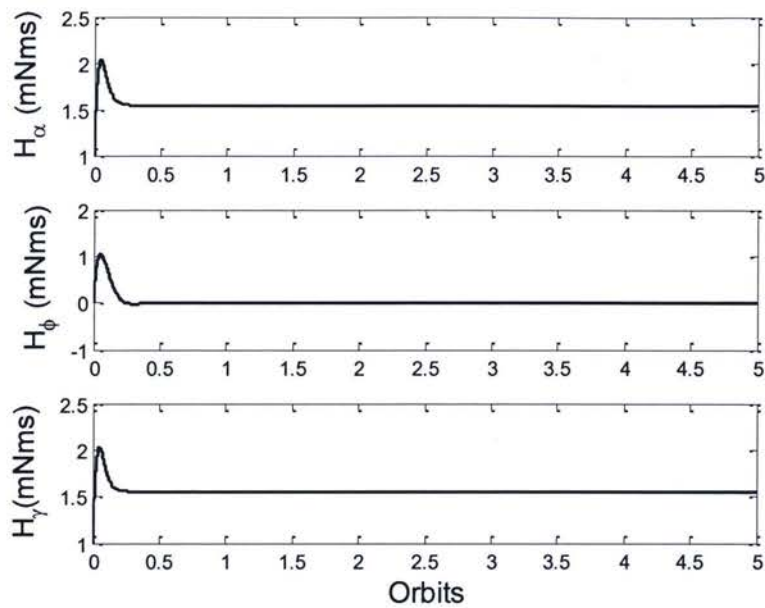


Figure 58: Reaction wheel storage momentum while maneuvering (ideal cube MOI))

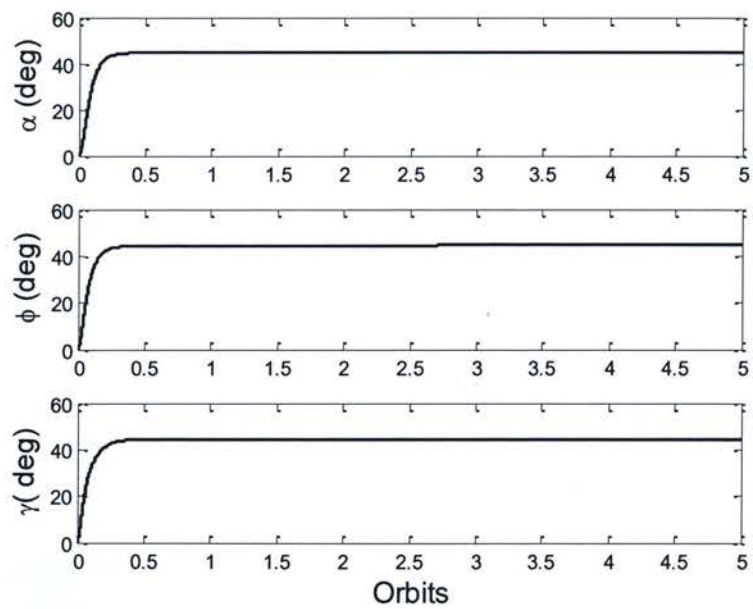


Figure 59: Attitude maneuver response (worst case MOI)



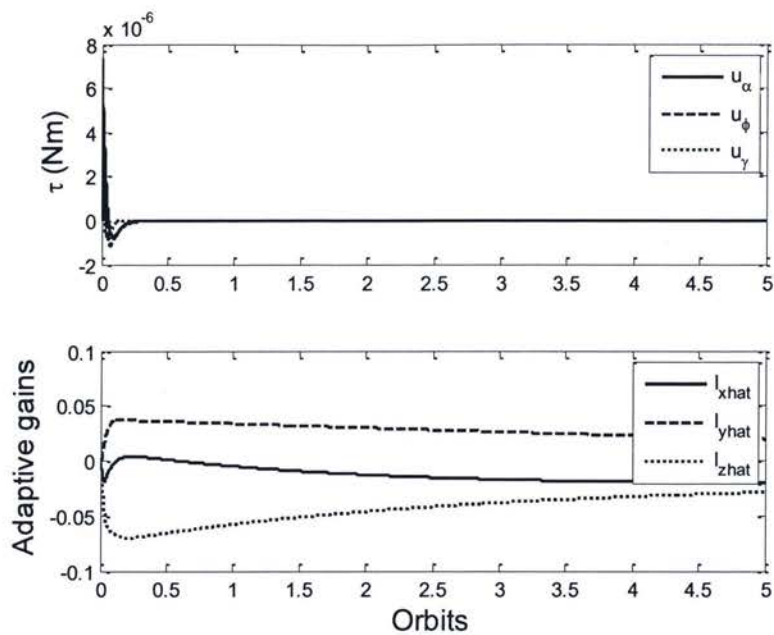


Figure 60: Torque and adaptive gains while maneuvering (worst case MOI)

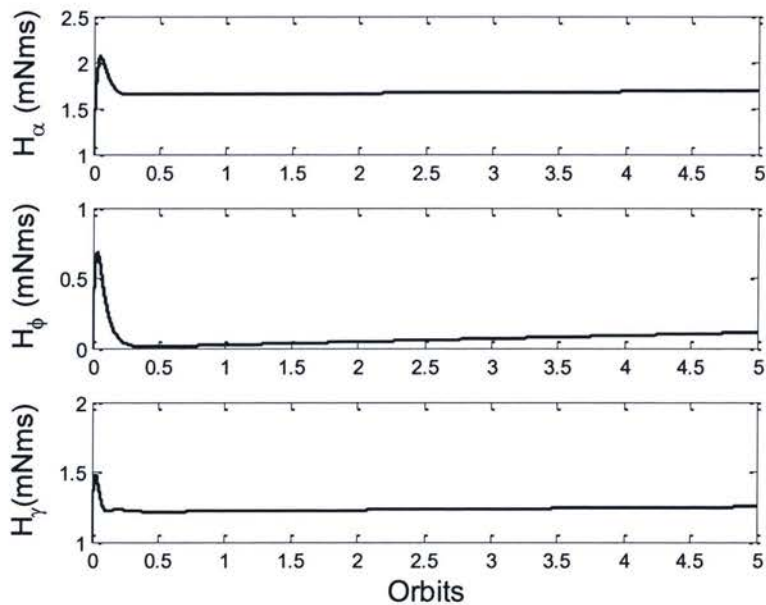


Figure 61: Reaction wheel storage momentum while maneuvering (worst case MOI)

# CHAPTER 6

## CONCLUSIONS AND FUTURE WORK

### 6.1 Conclusions

Pico-satellites are viable development platforms for testing new technologies in the space environment. The opportunities given to fly the earliest pico-satellites as secondary payloads have benefited all facets of the space community, both by effectively training the next generation of spacecraft designers and by pushing the boundaries of space technology at largely reduced risk and cost. The CubeSat standard is a wonderful way for universities and companies to participate in small satellite design. With a larger number of universities and companies doing research in small satellites, one can expect complex space technologies to be developed even faster. Already the development of space technology through CubeSats is evident with missions like CanX-2. It is felt by the author that future CubeSat missions will become even more complex, and these satellites will eventually be performing the tasks traditionally expected by nano and micro-satellites, such as operating inexpensive telescopes. And even if physical requirements prevent CubeSats from performing these tasks (such as inability to scale down optics), the technology developed for CubeSats will influence larger systems by creating technology that has lower mass and power requirements. In this thesis, a novel systems bus and a power subsystem were designed, developed and tested. Additionally, actuators and control algorithms for the satellite were designed.

#### 6.1.1 System Bus

The standard used in RyeSat is similar to those used in other CubeSats. Because of this similarity, a standard based on this work could be widely accepted as a future industry standard. Furthermore, this standard is also relatively low cost and requires no special connectors or specialized hardware. Early breadboard tests have shown no problems with the interface,

illustrating an early level of repeatability and reliability. The subsystems developed thus far are fully compatible with earlier breadboard models, proving that the standard can allow for incremental improvements and development of a system. In addition, only the slowest speed data rate has been used, implying there can be further improvement in the data rate. Although no standard can ever solve all problems, the standard proposed by the author does handle low data transmission requirements satisfactorily and is scalable for future missions. Additionally, this bus setup is similar to the already proven MIL-STD-1553 bus and is scalable both upwards and downwards to allow for larger and smaller satellites and subsystems.

### **6.1.2 Power Subsystem**

The power subsystem as presented is a working breadboard level of the actual spacecraft part; it is fully capable of supplying the 3.3V and 5V uninterruptible supply lines, and switching on and off the controllable 3.3V supply lines. The power subsystem is also capable of charging a lithium ion battery from a simulated solar cell input and measures various currents and voltages within the system to verify its correct operation. Furthermore, the power subsystem fits the standard size PCB and is capable of communicating with the rest of the satellite using the standard bus proposed.

### **6.1.3 ACS**

Both magnetic torque rods and reaction wheels were sized for use on RyeSat, and it was found that these actuators would fit within a single cube CubeSat and will be highly effective in controlling the attitude of RyeSat along all three axes. The PD magnetic controller was able to stabilize the tumbling motion without exceeding the available magnetic moment from the designed torque coils. Finally, nonlinear adaptive sliding mode controllers for the reaction wheels were successful in stabilizing the 3-axis attitude motion of the RyeSat with a high degree of attitude precision and within half an orbit. Furthermore, these control laws did not exceed the torque available from the designed reaction wheels.

## **6.2 Future work**

### **6.2.1 System Bus**

An effort should be made to create a module that will allow previously developed systems to interact with the industry standard CubeSat kit. The power board would be a logical place to add



a bridge between the previous Ryerson devised pin out and that of the CubeSat kits. The major difference between the two standards is the physical interface and the way power to subsystems is individually controlled by the power subsystem. The modification would divert the 3.3V and 5V uninterruptible lines to the base CubeSat kit flight module and carry the I<sup>2</sup>C lines from the Flight module to the standard interface. Also, a re-examination of the board standard should be undertaken to allow it to fit together with the standard CubeSat kit card standard.

### 6.2.2 Power Subsystem

The power subsystem can be further improved in order reduce the number of parts on the board and simplify the design. This subsystem also needs to undergo several tests before actual implementation on the RyeSat; these tests include a vacuum test of the selected lithium ion cells, a full vacuum test of the charging system and battery, and a test with the estimated extreme temperatures.

### 6.2.3 ACS & ADS

In this thesis, the ADS was used to filter, calibrate and gather data from analog sensors. The C&DH subsystem would then apply the control law and send commands to the ACS which would then turn on the torque coils. One problem with this approach is that it involved the active participation of the C&DH module, a system that could be already busy doing other things such as decoding radio signals or dealing with payload data. The other problem with this setup is the magnetometer is currently located on the ADS subsystem preventing the ACS from directly knowing what magnetic field it is torquing against; this hinders the capability of the ACS greatly. A better strategy would be to add a remote I<sup>2</sup>C based magnetometer available to both the ADS and ACS; this would allow for placement of the magnetometer further away from the torquing coils and the radio. This would also allow the ACS to act more intelligently in its application of the magnetic torque coils. The other change that should be made to the overall design is the division of work for each component of the attitude determination and control system (ADCS). First, the ADS module should pre-process the raw data and produce the most up to date estimation of the attitude; this device could then just be polled for the spacecraft's current attitude states by either the C&DH for transmission to earth or by the ACS for determining the control outputs. The ACS should also be modified to be an on orbit reprogrammable device.

# References

- Alger, M., G. McVittie, T. Patel, and Shaild. "Picosatellite CDR document." Department of Aerospace, Space Systems Dynamics & Controls Group, Ryerson University, Toronto, 2007.
- Alminde, L. *AAU CubeSat*. Alaborg University. 2003. <http://www.cubesat.auc.dk/> (accessed September 20, 2007).
- Alminde, L., M. Bisgaard, D. Vinter, T. Viscor, and K. Ostergaard. "AAU CubeSat Documentation." *Robustness of Radio Link Between AAU Cubesat and GroundStation*. Alaborg University. 2002. <http://www.cubesat.aau.dk/> (accessed September 10th, 2007).
- AMSAT. *AMSAT - Satellite Detail -CubeSat OSCAR 55*. The Radio Amateur Satellite Corporation. April 6, 2006. [http://www.amsat.org/amsat-new/satellites/satInfo.php?satID=69&retURL=satellites/all\\_oscars.php](http://www.amsat.org/amsat-new/satellites/satInfo.php?satID=69&retURL=satellites/all_oscars.php) (accessed July 22, 2008).
- Bleier, T., et al. "Lessons Learned, Quakefinder." *QuakeSat*. Stanford University. 2004. <http://www.quakefinder.com/lessons.htm>. (accessed June 15, 2007).
- Caillibot, P. *Systems Engineering and Ground Station Software for the CanX-2 Nanosatellite*. Masters Thesis, Toronto, Ontario: University of Toronto, UTIAS, 2005.
- David, L. *CubeSats: Tiny Spacecraft, Huge Payoffs*. Imaginova. September 8, 2004. [http://www.space.com/business/technology/cube\\_sats\\_040908.html](http://www.space.com/business/technology/cube_sats_040908.html). (accessed July 10, 2007).
- Diamond systems Corporation. *Earthquake Research Satellite Demonstrates Ruggedness of Diamond Systems' CPUs*. Diamond systems Corporation. 2001-2008. <http://www.diamondsystems.com/articles/13> (accessed March 9, 2008).
- Durham, Matthew. *Cubesat Community- Missions*. 2008. <http://www.cubesat.org/> (accessed March 23, 2008).
- ESA. *Space Engineering SpaceWire - Links Nodes routers and networks*. ESA Publications Division, ESA, Noordwijk Netherlands: ESTEC, 2003.
- Fleeter, R. *The Logic of Microspace*. El Segundo, CA.: Space Technology Library, Microcosm Press, 2000.
- Galysh, I., K. Doherty, J. McGuire, H. Heidt, D. Niemi, and G. Dutchover. "CubeSat: Developing a Standard Bus for Picosatellites." *Proc. SPIE (Society of Photo-Optical Instrumentation Engineers)* 4136, (November 2000): p. 64-71.
- Holbert, K. *Space Radiation Environmental Effects*. Arizona state University. February 26th, 2008. <http://www.eas.asu.edu/~holbert/eee460/spacerad.html> (accessed March 9, 2008).
- Hughes, P. *Spacecraft Attitude Dynamics*. 1st Dover edition. Mineola, New York: Dover Publications, 2004.
- HutputtEanasin, A., and A. Toorian. "CubeSat Design Specifications Revision 9, Cal Poly." *Cubesat Community*. California Polytechnic State University. August 2, 2006. [http://cubesat.atl.calpoly.edu/media/CDS\\_rev10.pdf](http://cubesat.atl.calpoly.edu/media/CDS_rev10.pdf) (accessed January 1, 2007).
- Intelligent Space Systems Laboratory. *University of Tokyo CubeSat Project Critical Design Review*. Power Point Presentation File. ISSL at the University of Tokyo, Tokyo,, April 6, 2001.
- . *XI Series of CubeSats*. ISSL Tokyo University. March 19, 2008. <http://www.space.t.u-tokyo.ac.jp/cubesat/index-e.html> (accessed March 26, 2008).



---

Kayal, H., K. Brieß, H. Driesher, J. Eckler, and O. Hillenmaier. "Miniaturization - a New Evolution Level of Developmetn of Reaction Wheels." *5th International Symposium of the International Academy of Astronautics on Small Satellites for Earth Observation*. Berlin, 2005.

Krebs, G. *Gunter's space page*. December 08, 2007.

[http://space.skyrocket.de/index\\_frame.htm?http://space.skyrocket.de/doc\\_sdat/xi-4.htm](http://space.skyrocket.de/index_frame.htm?http://space.skyrocket.de/doc_sdat/xi-4.htm) (accessed March 30, 2008).

Kumar, K. D. *Fundamentals of Space Systems*. 1st edition. Toronto, Ontario: Ryerson Bookstore, 2006.

Lab for Space Systems. "CUTE I Satellite Homepage, Tokyo Institute of Technology." *CUTE I*. TITECH CUBESAT PROJECT TEAM. Febuary 21, 2006. [http://lss.mes.titech.ac.jp/ssp/cubesat/index\\_e.html](http://lss.mes.titech.ac.jp/ssp/cubesat/index_e.html) (accessed March 9, 2008).

Larson, W., and J. Wertz. *Space mission analysis and design*. 3rd Edition, 7th printing . El. Segundo, California : Space Technology Library/Microcosm, 2005.

Long, M., et al. "A cubesat derived design for a unique academic research mission." *16th Annual USU Conference on Small Satellites*. Logan, Utah, 2002.

Makovec, K. "A Nonlinear Magnetic Controller for Three Axis Stability of NanoSatellites." Masters thesis, Aerospace Engineering , Virginia Polytechnic Institute and State University, Blacksburg, Virginia , 2001.

Microchip. "16 -Bit 28pin Starter Development Board User's guide." 2007.

<http://ww1.microchip.com/downloads/en/DeviceDoc/DS-51656A.pdf> (accessed May 3, 2008).

Øverby, E. *Attitude control for the Norwegian student satellite nCube*. Masters Thesis, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim: Norwegian University of Science and Technology, 2004.

Paruchuri, N. *KUHABS, KUBESAT&KUTESAT-1 Technical Report, Design of a Modular*. Thesis, Lawrence, Kansas: Kansas State University, 2006.

Phillips NXP Semiconductor. *The I2C-Bus Specification Version 2.1*. Data sheet, Eindhoven, The Netherlands: NXP semiconductor, 2000.

Portescap. "The nuvoDisc Motor." *Portescap - Minature Motors*. 2007.

<http://www.portescap.com/catalog/nuvoDisc%2032BF%20promo%20flyer%208.07.pdf> (accessed June 5, 2008).

Pumkin. "CubeSat Kit ." *CubeSat Kit.FM430 Flight ModuleHardware Revision: C*. Pumkin Incorporated, San Francisco, California. 2007. [http://www.cubesatkit.com/docs/datasheet/DS\\_CSK\\_FM430\\_710-00252-C.pdf](http://www.cubesatkit.com/docs/datasheet/DS_CSK_FM430_710-00252-C.pdf) (accessed July 20, 2007).

Pumpkin. *CubeSat kit datasheets*. April 2008.

[http://www.cubesatkit.com/docs/datasheet/DS\\_CSK\\_ADACS\\_634-00412-A.pdf](http://www.cubesatkit.com/docs/datasheet/DS_CSK_ADACS_634-00412-A.pdf) (accessed June 3, 2008).

Sinclair Interplanetary. *Sinclair interplanetary Reaction wheels*. 2008.

<http://www.sinclairinterplanetary.com/30mNm-secwheel2008c.pdf> (accessed June 3, 2008).

Space Flight Lab. *CanX-1*. University Of Toronto Inistitute for Aerospace Studies. 2007. <http://www.utias-sfl.net/nanosatellites/CanX1> (accessed 08 10, 2007).

—. *CanX-2*. 2007. <http://www.utias-sfl.net/nanosatellites/CanX2> (accessed 08 10, 2007).

Space Quest. *SpaceQuest products*. 2008. <http://www.spacequest.com/products/TQR-5.pdf> (accessed June 3, 2008).



---

Stras, L.N., D.D Kekez, G.J. Wells, T. Jeans, R.E.Pranajya,F.M Zee, and Foisy.D.G. "The design and operation of the Canadian advanced nanospace eXperiment (CanX-1)." *AMSAT-NA 21st Space Symposium*. Toronto, Ontario, Canada: Proc. AMSAT-NA 21st Space Symposium, 2003.

Sturman, Bryan. "RyeSat I2C Comm spec- ICD." Internal Interface Control Document, SpaceSystems Dynamics and Controls group , Ryerson University, Toronto, 2007.

US Department of Defense. *Digital time Division Comand/Response Multiplex Data Bus*. Online Document. 1996.

Villa, M. *Project Management of a Student Built Space Satellite: The KUTESat- 1*. PhD Thesis, University of Kansas, Lawrence, Kansas, USA: University of Kansas, 2005.

Villa, M. "Project Management of a Student Built Space Satellite: The KUTESat- 1." PhD Thesis, University of Kansas, Lawrence, Kansas, USA, 2005.

Wagner, A., M. Sams, R. Krauland, and A. Salerno. "Lion sat Team 4 Magnetic Torquer." Pennsylvania State University . 2003. <http://www.mne.psu.edu/me415/fall03/lionsat4/designtr1.htm> (accessed May 28, 2008).

Wagner, J. *Cajun Advanced Pico-satellite Experiment*,. University of Louisiana at Lafayette . 2005. <http://cape.louisiana.edu/> (accessed March 26, 2008).

Wang, T. "Development, Implementation, and Testing of Software for the CanX-1 Onboard." Masters Thesis, UTIAS, University of Toronto, Toronto,Ontario, Canada, 2004.

Wikipedia. *MIL-STD-1553*. Wikipedia. July 22, 2007. URL: <http://en.wikipedia.org/wiki/MIL-STD-1553> (accessed July 22, 2007).

—. *Serial Peripheral Interface Bus*. June 2008.

[http://en.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface\\_Bus](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus) (accessed July 31, 2007).

Wisiewski, R. *Aalborg University Student Satellite*. Aalborg University . 2004. [www.cubesat.auc.dk](http://www.cubesat.auc.dk) (accessed 08 20, 2007).

## Appendix A Thermal Calculations

Table 76: Sides 1- 3 in a 1000 km orbit (hot and cold cases)

Side 1	earth facing		Side 2	Radial outward facing		Side 3	space facing	
500 km alt	Hot	Cold	500 km alt	Sun facing	Space facing	500 km alt	Hot	Cold
Solar constant	1420	1360	Solar constant	1420	1360	Solar constant	1420	1360
Albedo	0.3	0.23	Albedo	0.3	0.23	Albedo	0.3	0.23
Earth IR	244	218	Earth IR	244	218	Earth IR	244	218
Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85
Absorption of side	0.92	0.92	Absorption of side	0.92	0.92	Absorption of side	0.92	0.92
Area	0.01	0.01	Area	0.01	0.01	Area	0.01	0.01
F solar	0	0	F solar	1	1	F solar	0	0
F Albedo	0.8587	0.8587	F Albedo	0	0	F Albedo	0.0232	0.0232
FEIR	0.8618	0.8618	FEIR	0	0	FEIR	0.2746	0.2746
q solar	0	0	q solar	1306.4	0	q solar	0	0
q albino	178.09438	159.11711	q albino	0	0	q albino	4.81168	4.29896
q EarthIR	337.756656	248.0053568	q EarthIR	0	0	q EarthIR	107.6212	79.02329
Q solar	0	0	Q solar	13.064	0	Q solar	0	0
Q albino	1.7809438	1.5911711	Q albino	0	0	Q albino	0.048117	0.04299
Q Earth Ir	3.37756656	2.480053568	Q Earth Ir	0	0	Q Earth Ir	1.076212	0.790233
Q backload	0	0	Q backload	0	0	Q backload	0	0
Qface1	5.15851036	4.071224668		13.064	0		1.124329	0.833222

Table 77: Sides 4- 6 in a 1000 km orbit (hot and cold cases)

Side 4	space facing		Side 5	space facing		Side 6	space facing	
500 km alt			500 km alt			500 km alt		
	Hot	Cold		Hot	Cold		Hot	Cold
solar constant	1420	1360	solar constant	1420	1360	solar constant	1420	1360
Albedo	0.3	0.23	Albedo	0.3	0.23	Albedo	0.3	0.23
Earth IR	244	218	Earth IR	244	218	Earth IR	244	218
Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85
Absorption of side	0.92	0.92	Absorption of side	0.92	0.92	Absorption of side	0.92	0.92
Area	0.01	0.01	Area	0.01	0.01	Area	0.01	0.01
F solar	0	0	F solar	0	0	F solar	0	0
F Albedo	0.0232	0.0232	F Albedo	0.0232	0.0232	F Albedo	0.0232	0.0232
FEIR	0.2746	0.2746	FEIR	0.2746	0.2746	FEIR	0.2746	0.2746
q solar	0	0	q solar	0	0	q solar	0	0
q albino	4.81168	4.29896	q albino	4.81168	4.29896	q albino	4.81168	4.29896
q EarthIR	107.621	79.0232	q EarthIR	107.621	79.0232	q EarthIR	107.621	79.0232
	2	9		2	9		2	9
Q solar	0	0	Q solar	0	0	Q solar	0	0
Q albino	0.04811	0.04299	Q albino	0.04811	0.04299	Q albino	0.04811	0.04299
	7			7			7	
Q Earth Ir	1.07621	0.79023	Q Earth Ir	1.07621	0.79023	Q Earth Ir	1.07621	0.79023
	2	3		2	3		2	3
Q backload	0	0	Q backload	0	0	Q backload	0	0
	1.12432	0.83322		1.12432	0.83322		1.12432	0.83322
	9	2		9	2		9	2



Table 78 Sides 1- 3 in a 500 km orbit (hot and cold cases)

Side 1	earth facing		Side 2	Sun facing		Side 3	space facing	
500 km alt			500 km alt			500 km alt		
	Hot	Cold		Hot	Cold		Hot	Cold
solar constant	1420	1360	solar constant	1420	1360	solar constant	1420	1360
Albedo	0.3	0.23	Albedo	0.3	0.23	Albedo	0.3	0.23
Earth IR	244	218	Earth IR	244	218	Earth IR	244	218
Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85
Absorption of side	0.92	0.92	Absorption of side	0.92	0.92	Absorption of side	0.92	0.92
Area	0.01	0.01	Area	0.01	0.01	Area	0.01	0.01
F solar	0	0	F solar	1	1	F solar	0	0
F Albedo	0.8587	0.8587	F Albedo	0	0	F Albedo	0.0232	0.0232
FEIR	0.8618	0.8618	FEIR	0	0	FEIR	0.2746	0.2746
q solar	0	0	q solar	1306.4	0	q solar	0	0
q Albedo	178.09438	159.11711	q Albedo	0	0	q Albedo	4.81168	4.29896
q EarthIR	337.756656	248.0053568	q EarthIR	0	0	q EarthIR	107.6212	79.02329
Q solar	0	0	Q solar	13.064	0	Q solar	0	0
Q Albedo	1.7809438	1.5911711	Q Albedo	0	0	Q Albedo	0.048117	0.04299
Q EarthIR	3.37756656	2.480053568	Q EarthIR	0	0	Q EarthIR	1.076212	0.790233
Q backload	0	0	Q backload	0	0	Q backload	0	0
Qface1	5.15851036	4.071224668		13.064	0		1.124329	0.833222

Table 79: Sides 4- 6 in a 500 km orbit (hot and cold cases)

Side 4	space facing		Side 5	space facing		Side 6	space facing	
500 km alt			500 km alt			500 km alt		
	Hot	Cold		Hot	Cold		Hot	Cold
solar constant	1420	1360	solar constant	1420	1360	solar constant	1420	1360
Albedo	0.3	0.23	Albedo	0.3	0.23	Albedo	0.3	0.23
Earth IR	244	218	Earth IR	244	218	Earth IR	244	218
Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85	Emissivity of side	0.85	0.85
Absorption of side	0.92	0.92	Absorption of side	0.92	0.92	Absorption of side	0.92	0.92
Area	0.01	0.01	Area	0.01	0.01	Area	0.01	0.01
F solar	0	0	F solar	0	0	F solar	0	0
F Albedo	0.0232	0.0232	F Albedo	0.0232	0.0232	F Albedo	0.0232	0.0232
FEIR	0.2746	0.2746	FEIR	0.2746	0.2746	FEIR	0.2746	0.2746
q solar	0	0	q solar	0	0	q solar	0	0
q Albedo	4.81168	4.29896	q Albedo	4.81168	4.29896	q Albedo	4.81168	4.29896
q EarthIR	107.6212	79.02329	q EarthIR	107.6212	79.02329	q EarthIR	107.6212	79.02329
Q solar	0	0	Q solar	0	0	Q solar	0	0
Q Albedo	0.048117	0.04299	Q Albedo	0.048117	0.04299	Q Albedo	0.048117	0.04299
Q EarthIR	1.076212	0.790233	Q EarthIR	1.076212	0.790233	Q EarthIR	1.076212	0.790233
Q backload	0	0	Q backload	0	0	Q backload	0	0
	1.124329	0.833222		1.124329	0.833222		1.124329	0.833222

The following is an elaboration of the methods used in the previous tables

$$Q_{net} = \sum_{sides} Q_{solar} + Q_{albedo} + Q_{earthIR} + Q_{backload} + Q_{heatgenerated} \quad 6-1$$

$$Q_{net} = \frac{T_{inside} - T_{space}}{R_{total}} = \frac{T_{surface} - 0}{\frac{1}{h_{radiative}A}} = \frac{T_{surface} - T_{inside}}{\frac{L}{K_{solarcell(glass)}A}} \quad 6-2$$

$$and$$

$$R_{total} = \frac{1}{h_{radiative}A} + \frac{L}{K_{solarcell(glass)}A}$$

$$h_{radiative} \approx \varepsilon \sigma A T^4 \quad 6-3$$

Where:

$\varepsilon$  is the emissivity of the solar cell (major covering on the spacecraft)

$\sigma$  is the Boltzmann constant  $5.67051 \times 10^{-8} \text{W/m}^2\text{K}^4$

A is the area of the face of the spacecraft

T is the temperature (for the given subscript)

The following is the equation relating the trip point on the thermostat to a particular resistor value

$$R_{Trip} = 0.5997 \times (T_{desired})^{2.1312} \quad 6-4$$



# Appendix B I<sup>2</sup>C Communications Test Code

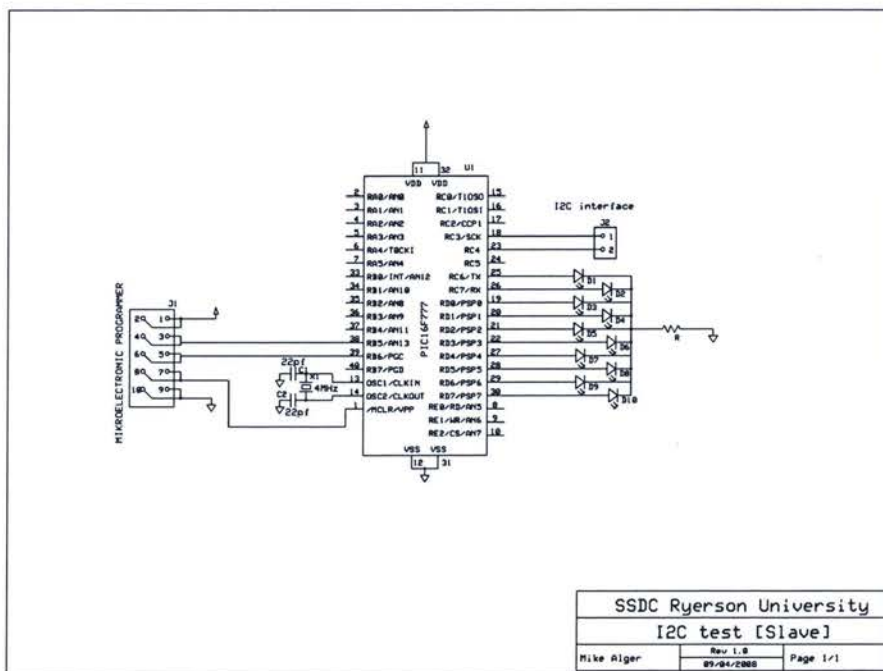
## A.1. Single Byte Each Way

### A.1.1. Overview

The following code was used as a reference to test and demonstrate the proposed system bus communication standard. This early iteration of the code utilized two different series of PIC microcontrollers a PIC24F as the master and a PIC16 as the slave. Either of these processors could have been the master or slave however in the interest to save power and increase reliability the 8Bit Pic16 series is more commonly used throughout the spacecraft as a slave device. So for this test it made sense to have the 16 series as the slave

### A.1.2. Slave

The slave is a PIC16F777 is shown below.



Code Excerpt 1: Slave Code for PIC 16 series

```
//unit Powerboardfirmwarev0002;

//control port/ttl port definition
#define PS1          PORTD.F4
#define PS2          PORTD.F5
#define PS3          PORTD.F6
#define PS4          PORTD.F7
#define busenable    PORTD.F2
#define chargeenable PORTD.F3
#define STAT1        PORTD.F1
#define STAT2        PORTD.F0
```

```

//#define SCLK          PORTC.F3
//#define SDA           PORTC.F4

//analog channel definition
#define getSolCel_1      Adc_Read(12)
#define getSolCel_2      Adc_Read(10)
#define getSolCel_3      Adc_Read(8)
#define getSolCel_4      Adc_Read(9)
#define getSolCel_5      Adc_Read(11)
#define getSolCel_6      Adc_Read(13)
#define getSolVolt       Adc_Read(2)
#define getBattVolt      Adc_Read(0)
#define getBattDraw      Adc_Read(1)

//Built in functions from MikroC
#define Lo(param) ((char *)&param)[0]
#define Hi(param) ((char *)&param)[1]
#define Higher(param) ((char *)&param)[2]
#define Highest(param) ((char *)&param)[3]

#define lo(param) ((char *)&param)[0]
#define hi(param) ((char *)&param)[1]
#define higher(param) ((char *)&param)[2]
#define highest(param) ((char *)&param)[3]

/*****
 * Global Variables
 *****/
char junk; // a place to dump extraneous data i.e. Address after it is checked
unsigned char ctrlbyte;
unsigned char length_msb, length_lsb, check_msb, check_lsb;
char Datain[20];
unsigned char dataout[20]={"\x00\x00\x05\x00\x05\xFF"};
char i,j=0;
unsigned int length, checkcalc=0, checkreceived, lasterror=0;
unsigned int BattVolt, Battdraw, Sol_1, Sol_2, Sol_3, Sol_4, Sol_5, Sol_6, Sol_volt;

unsigned char temp3[13]="sucks";
////////////////////////////////////

char readi2cbyte(void);
void S_writei2cbyte(unsigned char data_out);
void S_PUTS( char * wrptr);
void readADC (void);
void clearwdt(void);
void howdy (void);

/*****
interrupt function
 *****/

void interrupt (void)
{
    INTCON.GIE=0; /* turn of further interrupts*/
    //PORTD=0xFF; // turn on test port

```

```

if(!SSPSTAT.F2 && !SSPSTAT.F5){ /* checks i2c state: for our protocol only
                                state 1 works slave receive and last packet
                                was and address other states are interpreted as errors*/

    PORTC.F7=1; /* flick a bit to indicate receive mode has started*/
    checkcalc=0; /* reset the checksum as some is done in a loop*/
    junk=readi2cbyte(); /* empty the SSPBUFF OF THE ADDRESS*/

    ctrlbyte=readi2cbyte(); /* read control byte */
    length_msb=readi2cbyte(); /* read the high byte of length */
    length_lsb=readi2cbyte(); /* read the low byte of sentence length*/

    length=length_msb*0x100+length_lsb; /* calculate length for the data loop*/
    if (length<20){ /* check to see if its smaller then predefined size*/
        for (i=0;i<(length_lsb-5);i++){
            Datain[i]=readi2cbyte(); /*read in data bytes into an array*/
            checkcalc=checkcalc+Datain[i]; /*add them up as a checksum now; to save a loop later*/
        }
    }
    else{
        lasterror=0xE1; /*error condition sentence is to big */
        // TRISD=~0xE1; /* flash the error on the test port */
    }

    check_msb =readi2cbyte(); /* read the high byte of checksum*/
    check_lsb =readi2cbyte(); /* read the low byte of checksum*/
    /* check the checksum first turn the received one back to an int */
    checkreceived=(unsigned int)check_msb*0x100 +(unsigned int)check_lsb ;
    /* then add up all the received bytes*/
    checkcalc=ctrlbyte+length_msb+length_lsb+ checkcalc ;

    if (checkcalc==checkreceived){
        /* this is the main heart of the isr thus far, the slave is synced with the master
        the datachecksum is ok and the sentence is an appropriate length
        now is the time to interpret the ctrlbyte and utilize the data received
        */

        switch (ctrlbyte) {
            case 0x00: howdy(); break;
        }

        // PORTD=Datain[0];

    }
    else if (lasterror!=0xE1){ /*check to see if this is not because of previous error*/

        lasterror=0xE4 ; /*error condition Checksum wrong*/
        // TRISD=~0xE4; /* flash the error on the test port */
    }
}

else if(SSPSTAT.F2&&!SSPSTAT.F5){ /* checks i2c state: for our protocol only
                                state 3 works slave transmit and last packet
                                was and address other states are interpreted as errors*/

// PORTC.F6=1 ;
S_PUTS(dataout) ;
}
else if(SSPSTAT.F2){
    lasterror=0xE3;
}
else if(!SSPSTAT.F2){
    lasterror=0xE2;
}

else{lasterror=0xEE;}

/* Reset interrupts and before leaving interrupt function*/

```



```

PIR1.SSPIF=0;
INTCON.GIE=1;
}
////////////////////////////////////

/*-----
* Main
-----*/
void main (void)
{

/*-----
|configuration of i2c on MSSP port
-----*/

//(*pic16)
TRISC.F3=1;// configure clock as input
TRISC.F4=1;// configure data as input
// SSPBUF =0x00;
SSPADD =0xE6;
SSPSTAT =0b10000000;
SSPCON =0b00110110; // see section 10.4 pic 16f7x7 data sheet for specifics
// SSPCON2 =0b00 ;

//18 series has two available i2c ports [4 registers] (SSP1CON,SSP1CON2) &(SSP2CON,SSP2CON2)
// although their configuration is similar if not identical to the following

/*-----
|interrupt setup
-----*/

//(*pic16)
PIE1=0x00;
PIE1.SSPIF=1;
PIE2=0x00;// turn off all other
INTCON=0;
INTCON.PEIE=1;
INTCON.GIE=1;
PORTD=0b00001100;
//busenable=1;
// chargeenable=1;

/*-----
|ADC setup
-----*/
ADCON1 = 0x00; // Configure analog inputs and Vref

TRISA=0b00000111;
TRISB=0b00111111;

TRISD = 0b00000011; // set direction of control outputs and charge stat inputs
//PORTD=0xFF;

readADC();

do
{
Delay_ms(500);
PS1=~PS1;
PS2=~PS2;
PS3=~PS3;
}

```

```

PS4=~PS4;

INTCON.GIE=0;
readADC();
INTCON.GIE=1;

// clearwdt();
}
while (1);
}

/*****
 * Watchdog reset function
 *****/
void clearwdt(void) {
asm {
    CLRWDT
}
}
//*****

//Slave Read
//*****
char readi2cbyte(void)
{
    while(!SSPSTAT.BF);    // wait till the buffer is full
    PIR1.SSPIF=0;          // reset the overflow interrupt(sspov is tied to this)
    return (SSPBUF);        // return databyte in buffer
}
//*****

//Slave Write
//*****
void S_writei2cbyte(char data_out)
{
    SSPBUF = data_out;      /* data byte loaded into buffer */
    SSPCON.CKP = 1;         /* Release the clock */
}
//*****

void S_PUTS(char *wrptr) {
    while(*wrptr!=0xFF && *wrptr+1!=0x00) {
        SSPBUF=(*wrptr++);
        SSPCON.CKP=1;
        while(SSPSTAT.BF);
    }
}

void readADC (void)
{
    BattVolt = getBattVolt;
    BattDraw = getBattDraw;
    Sol_1    = getSolCel_1;
    Sol_2    = getSolCel_2;
    Sol_3    = getSolCel_3;
    Sol_4    = getSolCel_4;
    Sol_5    = getSolCel_5;
    Sol_6    = getSolCel_6;
}

```

```

Sol_volt = getSolVolt;

WordToStr(BattVolt, temp3);

}

void howdy (void){
    dataout[0]='b';
    dataout[1]=0;
    dataout[2]=0x0A;
    dataout[3]=temp3[0];
    dataout[4]=temp3[1];
    dataout[5]=temp3[2];
    dataout[6]=temp3[3];
    dataout[7]=temp3[4];
    dataout[8]='h';
    dataout[9]='i';
    dataout[10]=0xFF;//null to end string
    dataout[11]=0x00;
}

```

### A.1.3. Master

The master is built around the 28 pin starter kit from microchip details are found in (Microchip 2007), for this particular application the second I<sup>2</sup>C bus on the microchip kit was used.

#### Code Excerpt 2: Slave Code for PIC 16 series

```

#include "p24fj64ga002.h"
// the above include path may be different for each user. If a compile
// time error appears then check the path for the file above and edit
// the include statement above.

#include<i2c.h>

#define XT_FREQ      7372800           //On-board Crystal frequency
#define PLLMODE      2                //On-chip PLL setting
#define FCY          XT_FREQ*PLLMODE  //Instruction Cycle Frequency
#define BAUDRATE      9600
#define BRGVAL        ((FCY/BAUDRATE)/16)-1
#define I2CBAUD        400000         // i2c clock running at 400khz

//_CONFIG1(JTAGEN_OFF&GCP_OFF&GWRP_OFF &COE_OFF& ICS_PGx1 &FWDTEN_OFF );
//_CONFIG2(IESO_OFF &FNOSC_PRIPLL&FCKSM_CSDCMD&OSCIOFNC_OFF &I2C1SEL_SEC &POSCMOD_XT );

//

void delay(void)
{
    int var1,var2;
    for(var1=0;var1!=100;var1++)
    {
        for(var2=0;var2!=10000;var2++);
    }
}

//      EnableIntIM2C2;
//void __attribute__((interrupt, no_auto_psv)) _MI2C2Interrupt(void)

```



```

//{
//      //      jDone=1;
//      IFS3bits.MI2C2IF = 0;          //Clear the DMA0 Interrupt Flag;
//
//}

void main(void)
{
    CLKDIVbits.RCDIV = 0;
    RPINR10bits.U1RXR = 9;          // Make Pin RP9 U1RX
    RPOR4bits.RP8R = 3;            // Make Pin RP8 U1TX

    TRISB = 0x0300;
    U1BRG = BRGVAL;
    U1MODE = 0x8000;                // Reset UART to 8-n-1, alt pins, and enable
    U1STA = 0x0440;                // Reset status register and enable TX & RX

    PADCFG1 = 0xFF;                // Make analog pins digital
    LATB = 0xF000;                 //Toggle LED's
    TRISB = 0x00;                 // Configure LED pins as output

    unsigned int config2, config1;
    char check=0,ZZ=90;
    unsigned int checksum;
    unsigned char BB,i=0,X;

    char Datin[]={'Z','Z','Z','Z'};
    unsigned char ctrlbyte='Z';
    unsigned char length_msb='Z', length_lsb='Z',check_msb='Z',check_lsb='Z';
    unsigned int length, checkcalc=0, checkreceived, lasterror=0;

    /* Baud rate is set for 100 Khz */
    config2 = 0x90;
    /* Configure I2C for 7 bit address mode */
    config1 = 0xF20; //(I2C_ON & I2C_IDLE_CON & I2C_CLK_HLD &
//      I2C_IPMI_DIS & I2C_7BIT_ADD &
//      I2C_SLW_DIS & I2C_SM_DIS &
//      I2C_GCALL_DIS & I2C_STR_DIS &
//      I2C_NACK & I2C_ACK_DIS & I2C_RCV_DIS &
//      I2C_STOP_DIS & I2C_RESTART_DIS &
//      I2C_START_DIS);
    OpenI2C2(config1,config2);
    while(1){

        IdleI2C2();
        StartI2C2();
        /* Wait till Start sequence is completed */
        while(I2C2CONbits.SEN );
        /* Write Slave address and set master for transmission */
        check = MasterWriteI2C2(0xE6);
        /* Wait till address is transmitted */
        if (check==0 ){
            checksum=BB+7+0;
            check_msb=checksum/256;
            check_lsb=checksum%256;
            LATB=0xA000;
            IdleI2C2();
            MasterWriteI2C2(0x00);
            IdleI2C2();
            MasterWriteI2C2(0x00);
            IdleI2C2();
            MasterWriteI2C2(0x07);
            IdleI2C2();
            MasterWriteI2C2(BB);
            IdleI2C2();
            MasterWriteI2C2(0x00);          IdleI2C2();
            MasterWriteI2C2(check_msb);
            IdleI2C2();
        }
    }
}

```

```

        MasterWriteI2C2(check_lsb);
        IdleI2C2();
        StopI2C2();

    }

//    else{CloseI2C2();LATB=0x0000;}

    BB++;
//if(BB==0xF0){BB=0x00;}
    delay();
//    LATB=0x5000;
//    delay();

    IdleI2C2();
    StartI2C2();
    while(I2C2CONbits.SEN );
    check = MasterWriteI2C2(0xE7); // Write Slave address and set master for receive
    if (check==0 ){
        LATB=0x5000;
        IdleI2C2();

        ctrlbyte= MasterReadI2C2();
        AckI2C2();
        while(I2C1CONbits.ACKEN == 1);
        IdleI2C2();

        length_msb=MasterReadI2C2(); /* read the high byte of length */
        AckI2C2();
        while(I2C1CONbits.ACKEN == 1);
        IdleI2C2();

        length_lsb=MasterReadI2C2(); /* read the low byte of sentence length*/
        AckI2C2();
        while(I2C1CONbits.ACKEN == 1);
        IdleI2C2();

        length=length_msb*0x100+length_lsb; /* calculate length for the data loop*/
        //        if (length<20){ /* check to see if its smaller then predefined size*/
        X=length_lsb-5;
        for (i=0;i<(X);i++){
            Datain[i]=MasterReadI2C2(); /*read in data bytes into an array*/
            AckI2C2();
            while(I2C1CONbits.ACKEN == 1);
            IdleI2C2();
            checkcalc=checkcalc+Datain[i]; /*add them up as a checksum now; to save a loop later*/
        }

//        else {
//            Datain[0]=127;
//            Datain[1]=127;
//            Datain[2]=127;
//        }
//    }

    check_msb =MasterReadI2C2(); /* read the high byte of checksum*/
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();

    check_lsb =MasterReadI2C2(); /* read the low byte of checksum*/
    NotAckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();
    StopI2C2();

}
else{
    ZZ=~check;//0x02;
    LATB=0x0000;
    CloseI2C2();
}

```

```
        while(!U1STAbits.TRMT);
        U1TXREG = ' ';
        while(!U1STAbits.TRMT);           // Echo Back Received Character with quotes
        U1TXREG = ctrlbyte;
//      while(!U1STAbits.TRMT);
//      U1TXREG = length_msb;
//      while(!U1STAbits.TRMT);
//      U1TXREG = length_lsb;

for(i=0;i<length_lsb-5;i++){
    while(!U1STAbits.TRMT);
    U1TXREG = Datain[i];
}

    while(!U1STAbits.TRMT);
    U1TXREG = check_msb;
    while(!U1STAbits.TRMT);
    U1TXREG = check_lsb;
    while(!U1STAbits.TRMT);
    U1TXREG = ' ';
    while(!U1STAbits.TRMT);
    U1TXREG = 10;
    while(!U1STAbits.TRMT);
    U1TXREG = 13;
    delay();
}
return;
}
```



## Appendix C Power Subsystem firmware

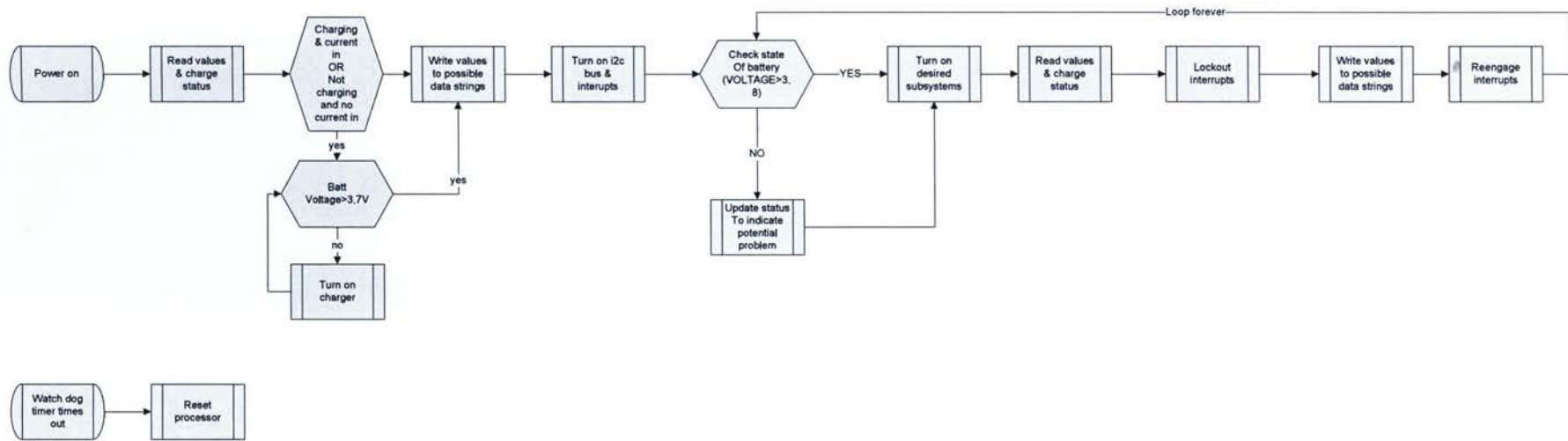


Figure 62: Power subsystem main loop & WDT

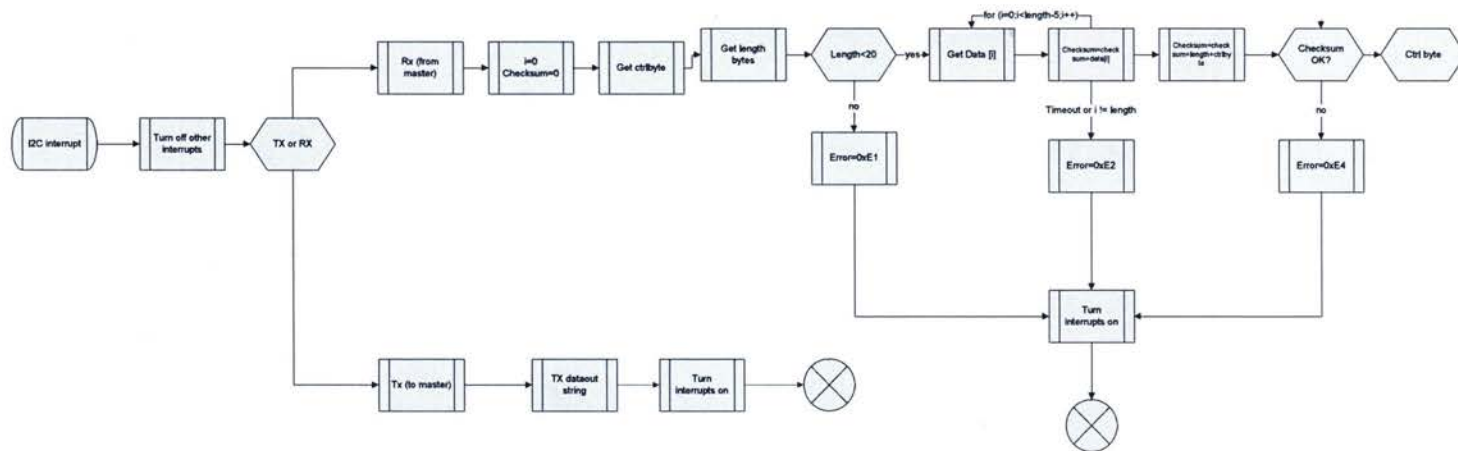


Figure 63: Interrupt error checking steps



## Appendix D Power Subsystem Code

Code Excerpt 3: Complete listing of the power subsystem code

```
/*unit Powerboardfirmwarev0006;
//notes:
Watch dog timer is turned on with a 1:128 postscaler this
should allow for at time out period of 2.3 seconds or so
-----*/

//control port/ttl port definition
#define PS1          PORTD.F4
#define PS2          PORTD.F5
#define PS3          PORTD.F6
#define PS4          PORTD.F7
#define busenable    PORTD.F2
#define chargeenable PORTD.F3
#define STAT1        PORTD.F1
#define STAT2        PORTD.F0

//#define SCLK        PORTC.F3
//#define SDA         PORTC.F4

//analog channel definition
#define getSolCel_1    Adc_Read(12)
#define getSolCel_2    Adc_Read(10)
#define getSolCel_3    Adc_Read(8)
#define getSolCel_4    Adc_Read(9)
#define getSolCel_5    Adc_Read(11)
#define getSolCel_6    Adc_Read(13)
#define getSolVolt     Adc_Read(2)
#define getBattVolt    Adc_Read(0)
#define getBattDraw    Adc_Read(1)

#define turn_on_charger PORTD=0b00000100
#define active_on_bus   PORTD=0b00001100

//#define TMR1intervalH (65535-50000)/256
//#define TMR1intervalL (65535-50000)%256
//#define I2Ctimeout 1

//Built in functions from MikroC
#define Lo(param) ((char *)&param)[0]
#define Hi(param) ((char *)&param)[1]
#define Higher(param) ((char *)&param)[2]
#define Highest(param) ((char *)&param)[3]

#define lo(param) ((char *)&param)[0]
#define hi(param) ((char *)&param)[1]
#define higher(param) ((char *)&param)[2]
#define highest(param) ((char *)&param)[3]

/*****
* Global Variables
*****/
char junk; // a place to dump extraneous data i.e. Address after it is checked
unsigned char ctrlbyte;
unsigned char length_msb, length_lsb, check_msb, check_lsb, chargestatus, lasterror=0;
char Datain[20];
unsigned char dataout[25];
unsigned int length, checkcalc=0, checkreceived,i ;
unsigned int BattVolt, Battdraw, Sol_1, Sol_2, Sol_3, Sol_4, Sol_5, Sol_6, Sol_volt;
unsigned long RTCcounts=0, TIMEOUT;
unsigned char powerstat;
//*****
```



```

char readi2cbyte(void);
//void S_writei2cbyte(unsigned char data_out);
void S_PUTS( char * wrptr);
void readADC (void);
void clearwdt(void);
void howdy (void);

void powerswitch(char *datain);
void solarpanelstatus(void);
void batterystatus(void);
void errorstatus(void);
void commanderror(void);

/*****
interrupt function
*****/
void interrupt (void)
{
  INTCON.GIE=0; /* turn of further interrupts*/

  /*-----
  Timer counter increment (for future use)
  -----*/

  //if (PIR1==0x01 ||PIR1==0x09){
  // //if (RTCCOUNTS%10==0){PS1=~PS1; }
  // PS1=~PS1;
  //   RTCCOUNTS=RTCCOUNTS+1;
  //   T1CON = 0b00000000;
  //   TMR1H = TMR0intervalH;
  //   TMR1L = TMR0intervalL;
  //   T1CON = 0b00000001;
  //   PIR1.TMR1IF=0;
  // }
  /*-----

  /*-----
  I2C STATE MACHINE
  -----*/

  //else if(PIR1==0x08 ||PIR1==0x09){
  PS2=~PS2;
  if(!SSPSTAT.F2 && !SSPSTAT.F5){ /* checks i2c state: for our protocol only
                                  state 1 works slave receive and last packet
                                  was and address other states are interpreted as errors*/

    //PORTC.F7=1; /* flick a bit to indicate receive mode has started*/
    checkcalc=0; /* reset the checksum as some is done in a loop*/
    junk=readi2cbyte(); /* empty the SSPBUFF OF THE ADDRESS*/

    ctrlbyte=readi2cbyte(); /* read control byte */
    length_msb=readi2cbyte(); /* read the high byte of length */
    length_lsb=readi2cbyte(); /* read the low byte of sentence length*/
    length=length_msb*0x100+length_lsb; /* calculate length for the data loop*/
    if (length<20){ /* check to see if its smaller then predefined size*/
      for (i=0;i<(length-5);i++){ /*-5 accounts for ctrlbyte length & checksum */
        Datain[i]=readi2cbyte(); /*read in data bytes into an array*/
        checkcalc=checkcalc+Datain[i]; /*add them up as a checksum now; to save a loop later*/
      }
    }
    else{
      lasterror=0xE1; /*error condition sentence is to big */
      errorstatus();
      // TRISD=~0xE1; /* flash the error on the test port */
    }

    check_msb =readi2cbyte(); /* read the high byte of checksum*/

```

```

check_lsb =readi2cbyte(); /* read the low byte of checksum*/
/* check the checksum first turn the received one back to an int */
checkreceived=(unsigned int)check_msb*0x100 +(unsigned int)check_lsb ;
/* then add up all the received bytes*/
checkcalc=ctrlbyte+length_msb+length_lsb+ checkcalc ;

if (checkcalc==checkreceived){
/* this is the main heart of the isr. thus far, the slave is synced with the master
the datachecksum is ok and the sentence is an appropriate length
now is the time to interpret the ctrlbyte and utilize the data received
*/

    switch (ctrlbyte) {
        case 0x00: howdy(); break;
        case 0x02: powerswitch(Datain); break;
        case 0x04: solarpanelstatus(); break;
        case 0x08: batterystatus(); break;
        case 0xE0: errorstatus(); break;
        default : commanderror(); break;
    }
}
else if (lasterror!=0xE1){ /*check to see if this is not because of previous error*/
    lasterror=0xE4 ; /*error condition Checksum wrong*/
    errorstatus();
}
}

else if(SSPSTAT.F2&&!SSPSTAT.F5){
/* checks i2c state: for our protocol only state 3 works slave transmit and
last packet was and address other states are interpreted as errors*/
    S_PUTS(dataout) ;
}
else if(SSPSTAT.F2){
    lasterror=0xE3;
    errorstatus();
}
else if(!SSPSTAT.F2){
    lasterror=0xE2;
    errorstatus();
}
else{
    lasterror=0xEE;
    errorstatus();
}
// }
/*-----*/

/* Reset interrupts and before leaving interrupt function*/
PIR1=0;
INTCON.GIE=1;
}
////////////////////////////////////

/*-----
* Main
-----*/
void main (void)
{
/*WDT setup */
OPTION_REG=0b00001000; // see note on WDT chapter 26 in Family guide
OPTION_REG=0b00001111;
/*-----
|configuration of i2c on MSSP port
-----*/

```

```

//(*pic16)
TRISC.F3=1;// configure clock as input
TRISC.F4=1;// configure data as input

SSPAD = 0xE6;
SSPSTAT = 0b10000000;
SSPCON = 0b00110110; // see section 10.4 pic 16f7x7 data sheet for specifics
// SSPCON2.SEN = 1;

//18 series has two available i2c ports [4 registers] (SSP1CON,SSP1CON2) & (SSP2CON,SSP2CON2)
// although their configuration is similar if not identical to the following
//-----

/*-----
|configuration of Timer (for future use)
|-----*/

// TMR1H = TMR1intervalH;
// TMR1L = TMR1intervalL;
// T1CON = 0b00000001; /* timer turned on and works off of internal osc*/
//-----

/*-----
|interrupt setup
|-----*/

//(*pic16)
INTCON=0; // Turn off interrupts while configuring them (to be safe)
Delay_ms(500); // startup delay to allow crystal to stabilize
PIE1=0x08; // PIE1.SSPIF=1; turn on only I2C interrupt
//PIE1=0x09 // PIE1.TMR1IF=1 & PIE1.SSPIF=1; turn on timer & I2C interrupt
PIE2=0x00; // turn off all other interrupts in different register
INTCON.PEIE=1; // turn on interrupts from PIE1&PIE2 registers
//-----

/*-----
|ADC & IO port setup
|-----*/

ADCON1 = 0x00; // Configure analog inputs and Vref
TRISA = 0b00000111;
TRISB = 0b00111111;
TRISD = 0b00000011; // set direction of control outputs and charge stat inputs

/*-----
|Main
|-----*/

readADC();
//if (BattVolt<=574) {
// PORTD=0b11111000;
// delay_ms(1000);
// }
//else
// {
// INTCON.GIE=1;
// PORTD=0b11110100;
// do
// {
// Delay_ms(1000); //Waiting loop
// {
// INTCON.GIE=0;
// readADC();
// INTCON.GIE=1;
// if (BattVolt>620){PORTD=PORTD&0b11110111;} // turn off charger
// else if (BattVolt<=600){PORTD=PORTD|0b00001000;} // turn on charger
// }
// else if (BattVolt<=570){PORTD=PORTD|0b01111100;}

// clearwdt();
}while (1);

```



```

}

}

/*.....
* Watchdog reset function
*.....*/
void clearwdt(void){
asm {
    CLRWDT
}
}

////////////////////////////////////

//.....
//Slave Read
//.....
char readi2cbyte(void)
{
    while(!SSPSTAT.BF); // wait till the buffer is full
    PIR1.SSPIF=0; // reset the overflow interrupt(sspov is tied to this)
    SSPSTAT.BF=0;
    return (SSPBUF); // return databyte in buffer
}
//.....

//.....
//Slave Write
//.....

void S_PUTS(char *wrptr) {
    while((*wrptr!=0xFF && *wrptr+1!=0x00)) {
        SSPBUF=(*wrptr++);
        SSPCON.CKP=1;
        while(SSPSTAT.BF);
    }
}

//.....
////////////////////////////////////

//.....
//ADC Read
//.....
void readADC (void)
{
    BattVolt = getBattVolt;
    Battdraw = getBattDraw;
    Sol_1 = getSolCel_1;
    Sol_2 = getSolCel_2;
    Sol_3 = getSolCel_3;
    Sol_4 = getSolCel_4;
    Sol_5 = getSolCel_5;
    Sol_6 = getSolCel_6;
    Sol_volt = getSolVolt;
    powerstat=PORTD;
    chargestatus='G';

}

//.....

//.....
//COMMUNICATIONS TEST
//.....
void howdy (void){

```

```

char k;
unsigned int check2;

dataout[0]=0x00;
dataout[1]=0x00;
dataout[2]=0x0A;
dataout[3]='H';
dataout[4]='O';
dataout[5]='W';
dataout[6]='D';
dataout[7]='Y';
dataout[8]='x';//null to end string
dataout[9]='y';
dataout[10]=0xFF;//null to end string
dataout[11]=0x00;

check2=0;
k=0;
while(dataout[k]!='x' && dataout[k+1]!='y')
{
    check2=check2 + dataout[k];
    k++;
}
dataout[k]=Hi(check2);
dataout[k+1]=Lo(check2);
}
//*****

//*****
//POWER DISTRIBUTION CONTROL
//*****
void powerswitch(char *datain){
    char k;
    unsigned int check2;

    char power;
    power=PORTD;

    if (datain[0]==0xFF){ power.F4=1; } else if (datain[0]==0x11){power.F4=0;}
    if (datain[1]==0xFF){ power.F5=1; } else if (datain[1]==0x11){power.F5=0;}
    if (datain[2]==0xFF){ power.F6=1; } else if (datain[2]==0x11){power.F6=0;}
    if (datain[3]==0xFF){ power.F7=1; } else if (datain[3]==0x11){power.F7=0;}

    PORTD=power ;

    dataout[0]=0x02;
    dataout[1]=0x00;
    dataout[2]=0x09;
    if (PS1) {dataout[3]=0xFF;} else{dataout[3]=0x11;}
    if (PS2) {dataout[4]=0xFF;} else{dataout[4]=0x11;}
    if (PS3) {dataout[5]=0xFF;} else{dataout[5]=0x11;}
    if (PS4) {dataout[6]=0xFF;} else{dataout[6]=0x11;}
    dataout[7]='x';
    dataout[8]='y';
    dataout[10]=0xFF;//null to end string
    dataout[11]=0x00;

    check2=0;
    k=0;
    while(dataout[k]!='x' && dataout[k+1]!='y')
    {
        check2=check2 + dataout[k];
        k++;
    }
    dataout[k]=Hi(check2);
    dataout[k+1]=Lo(check2);
}
//*****

```

```

//*****
//SOLAR POWER STATUS
//*****
void solarpanelstatus(void){
    char k;
    unsigned int check2;
    dataout[0]=0x04;
    dataout[1]=0x00;
    dataout[2]=0x13;
    dataout[3]=Hi(Sol_1);
    dataout[4]=Lo(Sol_1);
    dataout[5]=Hi(Sol_2);
    dataout[6]=Lo(Sol_2);
    dataout[7]=Hi(Sol_3);
    dataout[8]=Lo(Sol_3);
    dataout[9]=Hi(Sol_4);
    dataout[10]=Lo(Sol_4);
    dataout[11]=Hi(Sol_5);
    dataout[12]=Lo(Sol_5);
    dataout[13]=Hi(Sol_6);
    dataout[14]=Lo(Sol_6);
    dataout[15]=Hi(Sol_volt);
    dataout[16]=Lo(Sol_volt);
    dataout[17]='x';
    dataout[18]='y';
    dataout[19]=0xFF;//null to end string
    dataout[20]=0x00;

    check2=0;
    k=0;
    while(dataout[k]!='x' && dataout[k+1]!='y')
    {
        check2=check2 + dataout[k];
        k++;
    }
    dataout[k]=Hi(check2);
    dataout[k+1]=Lo(check2);

}
//*****

//*****
//BATTERY STATUS
//*****
void batterystatus(void){
    char k;
    unsigned int check2;
    dataout[0]=0x08;
    dataout[1]=0x00;
    dataout[2]=0x0A;
    dataout[3]=chargestatus;
    dataout[4]=Hi(BattVolt);
    dataout[5]=Lo(BattVolt);
    dataout[6]=Hi(Battdraw);
    dataout[7]=Lo(Battdraw);

    dataout[8]='x';
    dataout[9]='y';
    dataout[10]=0xFF;//null to end string
    dataout[11]=0x00;

    check2=0;
    k=0;
    while(dataout[k]!='x' && dataout[k+1]!='y')
    {
        check2=check2 + dataout[k];

```



```

    k++;
}
dataout[k]=Hi(check2);
    dataout[k+1]=Lo(check2);

}
//*****

//*****
//ERROR STATUS
//*****
void errorstatus(void){
char k;
    unsigned int check2;
    dataout[0]=0xE0;
    dataout[1]=0x00;
    dataout[2]=0x06;
    dataout[3]=lasterror;
    dataout[4]='x';
    dataout[5]='y';
    dataout[6]=0xFF;//null to end string
    dataout[7]=0x00;

    check2=0;
    k=0;
    while(dataout[k]!='x' && dataout[k+1]!='y')
    {
        check2=check2 + dataout[k];
        k++;
    }
    dataout[k]=Hi(check2);
        dataout[k+1]=Lo(check2);
}
//*****

//*****
//POWER DISTRIBUTION CONTROL
//*****
void commanderror(void){
    dataout[0]=0xE0;
    dataout[1]=0x00;
    dataout[2]=0x06;
    dataout[3]=0xE7;// bad ctrlbyte error
    dataout[4]=0x01;// checksum MSB
    dataout[5]=0xCD;// checksum LSB
    dataout[6]=0xFF;//null to end string
    dataout[7]=0x00;
}
//*****

```

## Code Excerpt 4: Simulated master code

```

#include "p24fj64ga002.h"
// the above include path may be different for each user. If a compile
// time error appears then check the path for the file above and edit
// the include statement above.

#include<i2c.h>

#define XT_FREQ          7372800                //On-board Crystal frequency
#define PLLMODE          2                    //On-chip PLL setting

```

```

#define FCY          XTFREQ*PLLMODE          //Instruction Cycle Frequency
#define BAUDRATE      9600
#define BRGVAL        ((FCY/BAUDRATE)/16)-1
#define I2CBAUD        400000                // i2c clock running at 400khz

//_CONFIG1(JTAGEN_OFF&GCP_OFF&GWRP_OFF &COE_OFF& ICS_PGx1 &FWDTEN_OFF );
//_CONFIG2(IESO_OFF &FNOSC_PRIPLL&FCKSM_CSDCMD&OSCIOFNC_OFF &I2C1SEL_SEC &POSCMOD_XT );

//
char buffins[100]="testing123";

void delay(void)
{
    int var1,var2;
    for(var1=0;var1!=100;var1++)
    {
        for(var2=0;var2!=10000;var2++);
    }
}

void solarcheck(void);
void commcheck(void);
void battcheck(void);
void UARTwritestring(char *stoorng);
void config_initialize_UART(void);
void main(void)
{
    CLKDIVbits.RCDIV = 0;
    RPINR18bits.U1RXR = 9;          // Make Pin RP9 U1RX
    RPOR4bits.RP8R = 3;             // Make Pin RP8 U1TX

    TRISB = 0x0300;
    U1BRG = BRGVAL;
    U1MODE = 0x8000;                // Reset UART to 8-n-1, alt pins, and enable
    U1STA = 0x0440;                // Reset status register and enable TX & RX

    PADCFG1 = 0xFF;                // Make analog pins digital
    LATB = 0xF000;                 //Toggle LED's
    TRISB = 0x00;                  // Configure LED pins as output

    unsigned int config1;
    //unsigned char *wrptr;
    //unsigned char tx_data[] = {0x00,0xFF,0x00,0xFF,};
    // char check=0,ZZ=90;
    // unsigned int checksum;
    // unsigned char i=0,X;

    // char Datain[20];
    // unsigned char ctrlbyte='Z';
    // unsigned char length_msb='Z', length_lsb='Z',check_msb='Z',check_lsb='Z';
    // unsigned int length, checkcalc=0, checkreceived, lasterror=0;

    /* Baud rate is set for 100 Khz */
    config2 = 0x90;
    /* Configure I2C for 7 bit address mode */
    config1 = 0xF200; //(I2C_ON & I2C_IDLE_CON & I2C_CLK_HLD &
    // I2C_IPMI_DIS & I2C_7BIT_ADD &
    // I2C_SLW_DIS & I2C_SM_DIS &
    // I2C_GCALL_DIS & I2C_STR_DIS &
    // I2C_NACK & I2C_ACK_DIS & I2C_RCV_DIS &
    // I2C_STOP_DIS & I2C_RESTART_DIS &
    // I2C_START_DIS);
    OpenI2C2(config1,config2);

    config_initialize_UART();
    UARTwritestring(buffins);
    while(1){

```

```

commcheck();
battcheck();
solarcheck();

}
return;
}

void commcheck(void){
    char check=0,ZZ=90;
    unsigned int checksum;
    unsigned char i=0,X;

    unsigned char Datain[20];
    unsigned char ctrlbyte = 'Z';
    unsigned char length_msb='Z', length_lsb='Z',check_msb='Z',check_lsb='Z';
    unsigned int length, checkcalc=0, checkreceived, lasterror=0;

    IdleI2C2();
    StartI2C2();
    /* Wait till Start sequence is completed */
    while(I2C2CONbits.SEN );
    /* Write Slave address and set master for transmission */
    check = MasterWriteI2C2(0xE6);
    /* Wait till address is transmitted */
    if (check==0 ){
        checksum=5;
        check_msb=checksum/256;
        check_lsb=checksum%256;
        LATB=0xA000;
        IdleI2C2();
        MasterWriteI2C2(0x00); // ctrlbyte
        IdleI2C2();
        MasterWriteI2C2(0x00); // length_msb
        IdleI2C2();
        MasterWriteI2C2(0x05); // length_lsb
        IdleI2C2();
        MasterWriteI2C2(check_msb);
        IdleI2C2();
        MasterWriteI2C2(check_lsb);
        IdleI2C2();
        StopI2C2();
    }

    delay();
    //getI2Cdata ();

    IdleI2C2();
    StartI2C2();
    while(I2C2CONbits.SEN );
    check = MasterWriteI2C2(0xE7); // Write Slave address and set master for receive
    if (check==0 ){
        LATB=0x5000;
        IdleI2C2();

        ctrlbyte= MasterReadI2C2();
        AckI2C2();
        while(I2C1CONbits.ACKEN == 1);
        IdleI2C2();

        length_msb=MasterReadI2C2(); /* read the high byte of length */
        AckI2C2();
        while(I2C1CONbits.ACKEN == 1);
        IdleI2C2();

        length_lsb=MasterReadI2C2(); /* read the low byte of sentence length*/
        AckI2C2();
        while(I2C1CONbits.ACKEN == 1);
        IdleI2C2();
    }
}

```



```

length=length_msb*0x100+length_lsb; /* calculate length for the data loop*/
if (length<20){ /* check to see if its smaller then predefined size*/
    X=length_lsb-5;
    for (i=0;i<(X);i++){
        Dain[i]=MasterReadI2C2(); /*read in data bytes into an array*/
        AckI2C2();
        while(I2C1CONbits.ACKEN == 1);
        IdleI2C2();
        checkcalc=checkcalc+Dain[i]; /*add them up as a checksum now; to save a
loop later*/
    }
}
else{
    length_msb=0;
    length_lsb=8;
    Dain[0]='B';
    Dain[1]='A';
    Dain[2]='D';

}

check_msb =MasterReadI2C2(); /* read the high byte of checksum*/
AckI2C2();
while(I2C1CONbits.ACKEN == 1);
IdleI2C2();
check_lsb =MasterReadI2C2(); /* read the low byte of checksum*/
NotAckI2C2();
while(I2C1CONbits.ACKEN == 1);
IdleI2C2();
StopI2C2();

}
else{
    ZZ=-check;//0x02;
    LATB=0x0000;
    CloseI2C2();
}

while(!U1STAbits.TRMT);
UITXREG = '';
while(!U1STAbits.TRMT); /* Echo Back Received Character with quotes
UITXREG = ctrlbyte;
// while(!U1STAbits.TRMT);
// UITXREG = length_msb;
// while(!U1STAbits.TRMT);
// UITXREG = length_lsb;

for(i=0;i<length_lsb-5;i++){
    while(!U1STAbits.TRMT);
    UITXREG = Dain[i];
}
while(!U1STAbits.TRMT);
UITXREG = '';
// while(!U1STAbits.TRMT);
// UITXREG = check_msb;
// while(!U1STAbits.TRMT);
// UITXREG = check_lsb;

while(!U1STAbits.TRMT);
UITXREG = 10;
while(!U1STAbits.TRMT);
UITXREG = 13;
delay();
}
void solarcheck(void)

```

```

char check=0,ZZ=90;
unsigned int checksum;
unsigned char i=0,X;

unsigned char Datain[20];
unsigned char ctrlbyte='Z';
unsigned char length_msb='Z', length_lsb='Z',check_msb='Z',check_lsb='Z';
unsigned int length, checkcalc=0, checkreceived, lasterror=0;

IdleI2C2();
StartI2C2();
/* Wait till Start sequence is completed */
while(I2C2CONbits.SEN );
/* Write Slave address and set master for transmission */
check = MasterWriteI2C2(0xE6);
/* Wait till address is transmitted */
if (check==0 ){
    checksum=9;
    check_msb=checksum/256;
    check_lsb=checksum%256;
    LATB=0xA000;
    IdleI2C2();
    MasterWriteI2C2(0x04); // ctrlbyte
    IdleI2C2();
    MasterWriteI2C2(0x00); // length_msb
    IdleI2C2();
    MasterWriteI2C2(0x05); // length_lsb
    IdleI2C2();
    MasterWriteI2C2(check_msb);
    IdleI2C2();
    MasterWriteI2C2(check_lsb);
    IdleI2C2();
    StopI2C2();
}
delay();
//getI2cdata ();

IdleI2C2();
StartI2C2();
while(I2C2CONbits.SEN );
check = MasterWriteI2C2(0xE7); // Write Slave address and set master for receive
if (check==0 ){
    LATB=0x5000;
    IdleI2C2();

    ctrlbyte= MasterReadI2C2();
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();

    length_msb=MasterReadI2C2(); /* read the high byte of length */
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();

    length_lsb=MasterReadI2C2(); /* read the low byte of sentence length*/
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();

    length=length_msb*0x100+length_lsb; /* calculate length for the data loop*/
    if (length<20){ /* check to see if its smaller then predefined size*/
        X=length_lsb-5;
        for (i=0;i<(X);i++){
            Datain[i]=MasterReadI2C2(); /*read in data bytes into an array*/
            AckI2C2();
            while(I2C1CONbits.ACKEN == 1);
            IdleI2C2();
            checkcalc=checkcalc+Datain[i]; /*add them up as a checksum now; to save a

```

```

loop later*/
    }

    }
    else{

        length_msb=0;
        length_lsb=8;
        Datain[0]='B';
        Datain[1]='A';
        Datain[2]='D';

    }

    check_msb =MasterReadI2C2(); /* read the high byte of checksum*/
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();
    check_lsb =MasterReadI2C2(); /* read the low byte of checksum*/
    NotAckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();
    StopI2C2();

}
else{
    ZZ=-check;//0x02;
    LATB=0x0000;
    CloseI2C2();
}

unsigned int temp;
double sc1=0.00,sc2=0.00,sc3=0.00,sc4=0.00,sc5=0.00,sc6=0.00,volt=0.00;

temp= (unsigned int) Datain[0]*0x100+ (unsigned int) Datain[1];
sc1=(double) temp* 0.05035400390625; //3.3/1024.0/.064

temp= (unsigned int) Datain[2]*0x100+ (unsigned int) Datain[3];
sc2=(double) temp* 0.05035400390625; //3.3/1024.0/.064

temp= (unsigned int) Datain[4]*0x100+ (unsigned int) Datain[5];
sc3=(double) temp* 0.05035400390625; //3.3/1024.0/.064

temp= (unsigned int) Datain[6]*0x100+ (unsigned int) Datain[7];
sc4=(double) temp* 0.05035400390625; //3.3/1024.0/.064

temp= (unsigned int) Datain[8]*0x100+ (unsigned int) Datain[9];
sc5=(double) temp* 0.05035400390625; //3.3/1024.0/.064

temp= (unsigned int) Datain[10]*0x100+ (unsigned int) Datain[11];
sc6=(double) temp* 0.05035400390625; //3.3/1024.0/.064

temp= (unsigned int) Datain[12]*0x100 + (unsigned int) Datain[13];
volt=(double)temp* 0.0064453125; //3.3/1024.0*2

sprintf(buffins, "SC1:%5.3f SC2:%5.3f SC3:%5.3f SC4:%5.3f SC5:%5.3f ,SC6:%5.3f",
SolarVolt:%5.3f\r\n",sc1,sc2,sc3,sc4,sc5,sc6,volt);
UARTwritestring(buffins);
delay();

}

void battcheck(void){
    char check=0,ZZ=90;
    unsigned int checksum;

```



```

unsigned char i=0,X;

unsigned char Datain[20];
unsigned char ctrlbyte ='Z';
unsigned char length_msb='Z', length_lsb='Z',check_msb='Z',check_lsb='Z';
unsigned int length, checkcalc=0, checkreceived, lasterror=0;

IdleI2C2();
StartI2C2();
/* Wait till Start sequence is completed */
while(I2C2CONbits.SEN );
/* Write Slave address and set master for transmission */
check = MasterWriteI2C2(0xE6);
/* Wait till address is transmitted */
if (check==0 ){
    checksum=13;
    check_msb=checksum/256;
    check_lsb=checksum%256;
    LATB=0xA000;
    IdleI2C2();
    MasterWriteI2C2(0x08); // ctrlbyte
    IdleI2C2();
    MasterWriteI2C2(0x00); // length_msb
    IdleI2C2();
    MasterWriteI2C2(0x05); // length_lsb
    IdleI2C2();
    MasterWriteI2C2(check_msb);
    IdleI2C2();
    MasterWriteI2C2(check_lsb);
    IdleI2C2();
    StopI2C2();
}

delay();
//getI2Cdata ();

IdleI2C2();
StartI2C2();
while(I2C2CONbits.SEN );
check = MasterWriteI2C2(0xE7); // Write Slave address and set master for receive
if (check==0 ){
    LATB=0x5000;
    IdleI2C2();

    ctrlbyte= MasterReadI2C2();
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();

    length_msb=MasterReadI2C2(); /* read the high byte of length */
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();

    length_lsb=MasterReadI2C2(); /* read the low byte of sentence length*/
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();

    length=length_msb*0x100+length_lsb; /* calculate length for the data loop*/
    if (length<20){ /* check to see if its smaller then predefined size*/
        X=length_lsb-5;
        for (i=0;i<(X);i++){
            Datain[i]=MasterReadI2C2(); /*read in data bytes into an array*/
            AckI2C2();
            while(I2C1CONbits.ACKEN == 1);
            IdleI2C2();
            checkcalc=checkcalc+Datain[i] ; /*add them up as a checksum now; to save a
loop later*/
        }
    }
}

```

```

    }
    else{

        length_msb=0;
        length_lsb=8;
        Datain[0]='B';
        Datain[1]='A';
        Datain[2]='D';

    }

    check_msb =MasterReadI2C2(); /* read the high byte of checksum*/
    AckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();
    check_lsb =MasterReadI2C2(); /* read the low byte of checksum*/
    NotAckI2C2();
    while(I2C1CONbits.ACKEN == 1);
    IdleI2C2();
    StopI2C2();

}
else{
    ZZ=-check;//0x02;
    LATB=0x0000;
    CloseI2C2();
}

}

unsigned int temp;
double Battvolt=0.00,Battdraw=0.00;

temp= (unsigned int) Datain[1]*0x100 + (unsigned int) Datain[2];
Battvolt=(double)temp* 0.0064453125; //3.3/1024.0*2
temp= (unsigned int) Datain[3]*0x100+ (unsigned int) Datain[4];
Battdraw=(double) temp* 0.05035400390625; //3.3/1024.0/.064

sprintf(buffins, "Charge Status:Off BattVolt:%5.3f BattDraw:%5.3f\r\n",Battvolt, Battdraw);
UARTwritestring(buffins);
delay();
}

void config_initialize_UART(void)
{
    unsigned int baudvalue;
    unsigned int U1MODEvalue;
    unsigned int U1STAvalue;

    CloseUART1();

    baudvalue = BRGVAL;
    U1MODEvalue = 0x8000; // Reset UART to 8-n-1, alt pins, and enable
    U1STAvalue = 0x0440; // Reset status register and enable TX & RX

    // ConfigIntUART1 (UART_RX_INT_DIS&UART_TX_INT_DIS);
    // U1MODEvalue =UART_EN & UART_IDLE_CON &UART_EN_WAKE& UART_DIS_LOOPBACK & UART_DIS_ABAUD&UART_NO_PAR_8BIT
    // &UART_1STOPBIT & UART_ALTRX_ALTTX;
    // U1STAvalue =UART_TX_PIN_NORMAL&UART_TX_ENABLE&UART_ADR_DETECT_DIS&UART_RX_OVERRUN_CLEAR;
    // OpenUART1 (U1MODEvalue, U1STAvalue, baudvalue);
    // OpenUART1 (U1MODEvalue, U1STAvalue, baudvalue);
}

void UARTwritestring(char *stoorng)
{
    int stoorng_index=0;
    while(stoorng[stoorng_index]!=0){ //loop until null string

```

```
terminator.  
                                WriteUART1(storing[storing_index++]);           //write  
character to the uart  
                                while(BusyUART1());                           //spin  
until the uart is finished with the last char.  
                                };  
}
```

## Appendix E Additional mathematics

Lagrange's Equations of motion (reproduced from Kumar 2006)

$$\begin{aligned}
 L_{\alpha} :: & (\cos^2 \phi \cos^2 \gamma + k_{yz} \cos^2 \phi \sin^2 \gamma + k_{xz} \sin^2 \phi) \alpha'' \\
 & + (-\sin \gamma \cos \phi \cos \gamma + k_{yz} \cos \gamma \cos \phi \sin \gamma) \phi'' \\
 & - k_{xz} \sin \phi \gamma'' + ((1 - k_{xz}) \sin \phi \cos \phi \cos \gamma \cos \phi \sin \gamma + (k_{xz} - k_{yz}) \sin \phi \cos \phi \sin \gamma \cos \phi \cos \gamma \\
 & - (1 - k_{yz}) \cos^2 \phi \sin \gamma \cos \gamma \sin \phi) \alpha'^2 + \{(-1 + k_{xz} - k_{yz}) \cos \phi \sin \gamma \cos \phi \cos \gamma \\
 & - (1 - k_{xz} - k_{yz}) \cos \phi \cos \gamma \cos \phi \sin \gamma\} \gamma' + (-1 - k_{xz} + k_{yz}) \sin \phi \cos^2 \gamma \cos \phi \\
 & - (1 - k_{xz} + k_{yz}) \sin \phi \sin^2 \gamma \cos \phi - ((1 - k_{yz}) \cos 2\gamma - k_{xz}) \cos \phi \sin \phi \phi' \\
 & + (2k_{xz} - 2k_{yz}) \sin \phi \cos \phi \sin \gamma \cos \phi \cos \gamma + (2k_{xz} - 2k_{yz}) \sin \phi \cos \phi \cos \gamma \cos \phi \sin \gamma \\
 & - 2(1 - k_{yz}) \cos^2 \phi \sin \gamma \cos \gamma \sin \phi \alpha' + \{(-1 + k_{xz} - k_{yz}) \cos^2 \gamma \cos \phi \\
 & - (1 - k_{xz} - k_{yz}) \sin^2 \gamma \cos \phi\} \phi' - (1 + k_{xz} + k_{yz}) \cos \phi \sin \gamma \cos \phi \cos \gamma \\
 & - (1 - k_{xz} - k_{yz}) \cos \phi \cos \gamma \cos \phi \sin \gamma \gamma' - (-1 + k_{yz}) \sin \gamma \cos \gamma \sin \phi \phi'^2 \\
 & + (-1 - k_{xz} + k_{yz}) \sin \phi \cos^2 \gamma \cos \phi - (1 - k_{xz} + k_{yz}) \sin \phi \sin^2 \gamma \cos \phi \\
 & - ((1 - k_{yz}) \cos 2\gamma - k_{xz}) \cos \phi \sin \phi \phi' + \{(k_{xz} - k_{yz}) \sin \phi \cos \phi \sin \gamma \\
 & + 3(k_{xz} - k_{yz})(\cos \alpha \sin \phi \sin \gamma - \sin \alpha \cos \gamma) \cos \alpha \cos \phi\} \cos \phi \cos \gamma \\
 & + ((1 - k_{xz}) \sin \phi \cos \phi \cos \gamma + 3(1 - k_{xz})(\cos \alpha \sin \phi \cos \gamma \\
 & + \sin \alpha \sin \gamma) \cos \alpha \cos \phi) \cos \phi \sin \gamma - \{-3(1 - k_{yz})(\cos \alpha \sin \phi \cos \gamma + \sin \alpha \sin \gamma) \\
 & \cdot (\cos \alpha \sin \phi \sin \gamma - \sin \alpha \cos \gamma) + (1 - k_{yz}) \cos^2 \phi \sin \gamma \cos \gamma\} \sin \phi = 0
 \end{aligned}$$

6-5

$$\begin{aligned}
 L_{\phi} :: & (-\sin \gamma \cos \phi \cos \gamma + k_{yz} \cos \gamma \cos \phi \sin \gamma) \alpha'' \\
 & + (\sin^2 \gamma + k_{yz} \cos^2 \gamma) \phi'' + \{-(k_{xz} - k_{yz}) \sin \phi \cos \phi \sin^2 \gamma \\
 & + (1 - k_{xz}) \sin \phi \cos \phi \cos^2 \gamma\} \alpha'^2 \\
 & + \{((1 + k_{xz} - k_{yz}) \cos \phi \sin^2 \gamma - (1 - k_{xz} - k_{yz}) \cos \phi \cos^2 \gamma) \gamma' \\
 & - (2k_{xz} - 2k_{yz}) \sin \phi \cos \phi \sin^2 \gamma + (2 - 2k_{xz}) \sin \phi \cos \phi \cos^2 \gamma\} \alpha' \\
 & + \{((1 + k_{xz} - k_{yz}) \cos \gamma \sin \gamma + (1 - k_{xz} - k_{yz}) \sin \gamma \cos \gamma) \phi' \\
 & + (1 + k_{xz} - k_{yz}) \cos \phi \sin^2 \gamma - (1 - k_{xz} - k_{yz}) \cos \phi \cos^2 \gamma\} \gamma' \\
 & - ((k_{xz} - k_{yz}) \sin \phi \cos \phi \sin \gamma + 3(k_{xz} - k_{yz})(\cos \alpha \sin \phi \sin \gamma) \\
 & - \sin \alpha \cos \gamma) \cos \alpha \cos \phi \sin \gamma + \{(1 - k_{xz}) \sin \phi \cos \phi \cos \gamma \\
 & + 3(1 - k_{xz})(\cos \alpha \sin \phi \cos \gamma + \sin \alpha \sin \gamma) \cos \alpha \cos \phi\} \cos \gamma = 0
 \end{aligned}$$

6-6

$$\begin{aligned}
 L_{\gamma} :: & -k_{xz} \alpha'' \sin \phi + k_{xz} \gamma'' - [(1 - k_{yz}) \cos 2\gamma - k_{xz}](1 + \alpha') \phi' \cos \phi \\
 & + (1 - k_{yz})[(1 + \alpha')^2 \cos^2 \phi' - \phi'^2] \sin \gamma \cos \gamma \\
 & - 3(1 - k_{yz})(\cos \alpha \sin \phi \cos \gamma + \sin \alpha \sin \gamma)(\cos \alpha \sin \phi \sin \gamma - \sin \alpha \cos \gamma) = 0
 \end{aligned}$$

6-7

$\theta_1 - \theta_9$  terms

$$\begin{aligned}
 \text{thetal} = & -C1 * \sin(\text{phi}) * \text{phiD} * \cos(\text{phi}) * \alpha\text{phad} - C1 * \cos(\text{phi}) * \text{phiD} * \sin(\text{phi}) - C1 * \cos(\text{phi}) * \text{phiD} * \\
 & \sin(\text{phi}) * \alpha\text{phad} - C1 * \sin(\text{phi}) * \text{phiD} * \cos(\text{phi}) + C1 * \sin(\text{phi}) * \gamma\text{amadd} + 0.3e1 * C1 * \cos(\text{phi}) * \\
 & \cos(\alpha\text{phad}) * \cos(\text{phi}) * \sin(\alpha\text{phad}) - C2 * \alpha\text{phad} * \cos(\text{phi})^2 + C2 * \alpha\text{phad} + C1 * \cos(\text{phi}) * \text{phiD} * \gamma\text{amadd};
 \end{aligned}$$

6-8

$$\begin{aligned}
 \text{theta2} = & 0.3e1 * C1 * \sin(\text{phi}) * \sin(\alpha\text{phad}) * \cos(\alpha\text{phad}) * \sin(\text{phi}) + C2 * \alpha\text{phad} * \cos(\text{phi})^2 + C1 * \cos(\text{phi}) \\
 & * \text{phiD} * \gamma\text{amadd} + C1 * \sin(\text{phi}) * \text{phiD} * \cos(\text{phi}) + C1 * \cos(\text{phi}) * \text{phiD} * \sin(\text{phi}) - C1 * \text{phiDD} * \cos(\text{phi}) * \\
 & \sin(\gamma\text{amadd}) * \cos(\gamma\text{amadd}) - C1 * \sin(\text{phi}) * \sin(\gamma\text{amadd}) * \cos(\gamma\text{amadd}) * \cos(\text{phi})^2 - C1 * \sin(\text{phi}) * \sin(\gamma\text{amadd}) *
 \end{aligned}$$

6-9



$$\begin{aligned} & \cos(\gamma) * \cos(\phi) ^ 2 * \alpha^D ^ 2 - 0.3e1 * C1 * \cos(\phi) * \cos(\gamma) ^ 2 * \cos(\alpha) * \cos(\phi) * \\ & \sin(\alpha) - 0.2e1 * C1 * \sin(\phi) * \phi^D * \cos(\phi) * \cos(\gamma) ^ 2 - 0.6e1 * C1 * \sin(\phi) * \cos(\alpha) * \\ & \sin(\phi) * \cos(\gamma) ^ 2 * \sin(\alpha) - 0.2e1 * C1 * \cos(\phi) * \cos(\gamma) ^ 2 * \phi^D * \gamma^D - 0.2e1 * C1 * \\ & \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 * \alpha^D + 0.6e1 * C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \\ & \cos(\alpha) ^ 2 - 0.3e1 * C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \cos(\alpha) ^ 2 * \cos(\phi) ^ 2 + C1 * \sin(\phi) * \\ & \sin(\gamma) * \cos(\gamma) * \phi^D ^ 2 + C1 * \cos(\phi) * \sin(\gamma) * \cos(\phi) * \cos(\gamma) - 0.2e1 * C1 * \\ & \sin(\phi) * \phi^D * \cos(\phi) * \cos(\gamma) ^ 2 * \alpha^D - 0.2e1 * C1 * \cos(\phi) * \cos(\gamma) * \gamma^D * \cos(\phi) * \\ & \sin(\gamma) + C1 * \sin(\phi) * \phi^D * \cos(\phi) * \alpha^D + C1 * \cos(\phi) * \phi^D * \sin(\phi) * \alpha^D + C1 * \cos(\phi) * \\ & \sin(\gamma) * \sin(\phi) * \cos(\phi) * \cos(\gamma) * \alpha^D ^ 2 + 0.3e1 * C1 * \cos(\phi) * \sin(\gamma) * \cos(\alpha) ^ 2 * \\ & \cos(\phi) * \sin(\phi) * \cos(\gamma) - 0.2e1 * C1 * \cos(\phi) * \cos(\gamma) * \gamma^D * \cos(\phi) * \sin(\gamma) * \\ & \alpha^D + 0.2e1 * C1 * \cos(\phi) * \sin(\gamma) * \sin(\phi) * \cos(\phi) * \cos(\gamma) * \alpha^D - 0.3e1 * C1 * \sin(\phi) * \\ & \sin(\gamma) * \cos(\gamma) - C2 * \alpha^D * \cos(\phi) ^ 2 * \cos(\gamma) ^ 2; \end{aligned}$$

$$\begin{aligned} \theta_3 = & -0.3e1 * C1 * \sin(\phi) * \sin(\alpha) * \cos(\alpha) * \sin(\phi) - C1 * \cos(\phi) * \phi^D * \gamma^D - C1 * \\ & \sin(\phi) * \phi^D * \cos(\phi) + C1 * \cos(\phi) * \phi^D * \sin(\phi) + C1 * \phi^{DD} * \cos(\phi) * \sin(\gamma) * \cos(\gamma) + \\ & C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 + C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 * \\ & \alpha^D ^ 2 + 0.3e1 * C1 * \cos(\phi) * \cos(\gamma) ^ 2 * \cos(\alpha) * \cos(\phi) * \sin(\alpha) + 0.2e1 * C1 * \sin(\phi) * \\ & \phi^D * \cos(\phi) * \cos(\gamma) ^ 2 + 0.6e1 * C1 * \sin(\phi) * \cos(\alpha) * \sin(\phi) * \cos(\gamma) ^ 2 * \sin(\alpha) + \\ & + 0.2e1 * C1 * \cos(\phi) * \cos(\gamma) ^ 2 * \phi^D * \gamma^D + 0.2e1 * C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \\ & \cos(\phi) ^ 2 * \alpha^D - 0.6e1 * C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \cos(\alpha) ^ 2 + 0.3e1 * C1 * \sin(\phi) * \\ & \sin(\gamma) * \cos(\gamma) * \cos(\alpha) ^ 2 * \cos(\phi) ^ 2 - C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) * \phi^D ^ 2 - \\ & C1 * \cos(\phi) * \sin(\gamma) * \sin(\phi) * \cos(\phi) * \cos(\gamma) + 0.2e1 * C1 * \sin(\phi) * \phi^D * \cos(\phi) * \\ & \cos(\gamma) ^ 2 * \alpha^D + 0.2e1 * C1 * \cos(\phi) * \cos(\gamma) * \gamma^D * \cos(\phi) * \sin(\gamma) - C1 * \sin(\phi) * \\ & \phi^D * \cos(\phi) * \alpha^D - 0.3e1 * C1 * \cos(\phi) * \cos(\alpha) * \cos(\phi) * \sin(\alpha) + C1 * \cos(\phi) * \phi^D * \\ & \sin(\phi) * \alpha^D - C1 * \cos(\phi) * \sin(\gamma) * \sin(\phi) * \cos(\phi) * \cos(\gamma) * \alpha^D ^ 2 - 0.3e1 * C1 * \\ & \cos(\phi) * \sin(\gamma) * \cos(\alpha) ^ 2 * \cos(\phi) * \sin(\phi) * \cos(\gamma) + 0.2e1 * C1 * \cos(\phi) * \cos(\gamma) * \\ & \gamma^D * \cos(\phi) * \sin(\gamma) * \alpha^D - 0.2e1 * C1 * \cos(\phi) * \sin(\gamma) * \sin(\phi) * \cos(\phi) * \cos(\gamma) * \\ & \alpha^D + 0.3e1 * C1 * \sin(\phi) * \sin(\gamma) * \cos(\gamma) + C2 * \alpha^D * \cos(\phi) ^ 2 * \cos(\gamma) ^ 2; \end{aligned}$$

6-10

$$\begin{aligned} \theta_4 = & 0.3e1 * C3 * \cos(\alpha) ^ 2 * \cos(\phi) * \sin(\phi) + 0.2e1 * C3 * \sin(\phi) * \cos(\phi) * \alpha^D + C3 * \\ & \sin(\phi) * \cos(\phi) * \alpha^D ^ 2 - C3 * \gamma^D * \cos(\phi) * \alpha^D - C3 * \gamma^D * \cos(\phi) + C3 * \sin(\phi) * \\ & \cos(\phi); \end{aligned}$$

6-11

$$\begin{aligned} \theta_5 = & -C3 * \sin(\phi) * \cos(\phi) * \alpha^D ^ 2 + C3 * \gamma^D * \cos(\phi) + C3 * \sin(\phi) * \cos(\phi) * \alpha^D ^ 2 * \\ & \cos(\gamma) ^ 2 + 0.3e1 * C3 * \cos(\alpha) ^ 2 * \cos(\phi) * \sin(\phi) * \cos(\gamma) ^ 2 + 0.3e1 * C3 * \cos(\gamma) * \\ & \cos(\alpha) * \cos(\phi) * \sin(\alpha) * \sin(\gamma) + C3 * \gamma^D * \cos(\phi) * \alpha^D - 0.2e1 * C3 * \sin(\phi) * \\ & \cos(\phi) * \alpha^D + 0.2e1 * C3 * \sin(\phi) * \cos(\phi) * \alpha^D * \cos(\gamma) ^ 2 + 0.2e1 * C3 * \sin(\gamma) * \phi^D * \\ & \gamma^D * \cos(\gamma) - 0.2e1 * C3 * \gamma^D * \cos(\phi) * \cos(\gamma) ^ 2 * \alpha^D - C3 * \sin(\phi) * \cos(\phi) + C3 * \\ & \sin(\phi) * \cos(\phi) * \cos(\gamma) ^ 2 - C3 * \alpha^{DD} * \cos(\phi) * \sin(\gamma) * \cos(\gamma) - 0.2e1 * C3 * \gamma^D * \\ & \cos(\phi) * \cos(\gamma) ^ 2 + C4 * \phi^D * \cos(\gamma) ^ 2 - 0.3e1 * C3 * \cos(\alpha) ^ 2 * \cos(\phi) * \sin(\phi); \end{aligned}$$

6-12

$$\begin{aligned} \theta_6 = & -C4 * \phi^D * \cos(\gamma) ^ 2 + C4 * \phi^D + C3 * \alpha^{DD} * \cos(\phi) * \sin(\gamma) * \cos(\gamma) - 0.2e1 * C3 * \\ & \sin(\gamma) * \phi^D * \gamma^D * \cos(\gamma) - C3 * \gamma^D * \cos(\phi) * \alpha^D - C3 * \gamma^D * \cos(\phi) - C3 * \\ & \sin(\phi) * \cos(\phi) * \alpha^D ^ 2 * \cos(\gamma) ^ 2 - 0.3e1 * C3 * \cos(\gamma) * \cos(\alpha) * \cos(\phi) * \sin(\alpha) * \\ & \sin(\gamma) - 0.2e1 * C3 * \sin(\phi) * \cos(\phi) * \alpha^D * \cos(\gamma) ^ 2 + 0.2e1 * C3 * \gamma^D * \cos(\phi) * \\ & \cos(\gamma) ^ 2 * \alpha^D + 0.2e1 * C3 * \gamma^D * \cos(\phi) * \cos(\gamma) ^ 2 - C3 * \sin(\phi) * \cos(\phi) * \cos(\gamma) * \\ & ^ 2 - 0.3e1 * C3 * \cos(\alpha) ^ 2 * \cos(\phi) * \sin(\phi) * \cos(\gamma) ^ 2; \end{aligned}$$

6-13

$$\theta_7 = C5 * \phi^D * \cos(\phi) * \alpha^D + C5 * \sin(\phi) * \alpha^{DD} + C6 * \gamma^D + C5 * \phi^D * \cos(\phi);$$

6-14

$$\begin{aligned} \theta_8 = & 0.2e1 * C5 * \phi^D * \cos(\phi) * \cos(\gamma) ^ 2 + 0.2e1 * C5 * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 * \\ & \alpha^D + C5 * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 * \alpha^D ^ 2 + 0.6e1 * C5 * \cos(\alpha) * \sin(\phi) * \\ & \cos(\gamma) ^ 2 * \sin(\alpha) + 0.2e1 * C5 * \phi^D * \cos(\phi) * \cos(\gamma) ^ 2 * \alpha^D - 0.3e1 * C5 * \sin(\alpha) * \\ & \cos(\alpha) * \sin(\phi) - C5 * \phi^D * \cos(\phi) * \alpha^D + C5 * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 + 0.3e1 * C5 * \\ & \sin(\gamma) * \cos(\gamma) - 0.6e1 * C5 * \cos(\alpha) ^ 2 * \cos(\gamma) * \sin(\gamma) + 0.3e1 * C5 * \cos(\alpha) ^ 2 * \\ & \cos(\gamma) * \sin(\gamma) * \cos(\phi) ^ 2 - C5 * \sin(\gamma) * \cos(\gamma) * \phi^D ^ 2 - C5 * \phi^D * \cos(\phi); \end{aligned}$$

6-15

$$\begin{aligned} \theta_9 = & -0.2e1 * C5 * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 * \alpha^D - 0.6e1 * C5 * \cos(\alpha) * \sin(\phi) * \\ & \cos(\gamma) ^ 2 * \sin(\alpha) - 0.2e1 * C5 * \phi^D * \cos(\phi) * \cos(\gamma) ^ 2 + C5 * \sin(\gamma) * \cos(\gamma) * \phi^D * \\ & ^ 2 - C5 * \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 + 0.3e1 * C5 * \sin(\alpha) * \cos(\alpha) * \sin(\phi) - C5 * \\ & \sin(\gamma) * \cos(\gamma) * \cos(\phi) ^ 2 * \alpha^D ^ 2 + C5 * \phi^D * \cos(\phi) * \alpha^D - 0.3e1 * C5 * \sin(\gamma) * \\ & \cos(\gamma) - 0.2e1 * C5 * \phi^D * \cos(\phi) * \cos(\gamma) ^ 2 * \alpha^D - 0.3e1 * C5 * \cos(\alpha) ^ 2 * \cos(\gamma) * \\ & \sin(\gamma) * \cos(\phi) ^ 2 + 0.6e1 * C5 * \cos(\alpha) ^ 2 * \cos(\gamma) * \sin(\gamma) + C5 * \phi^D * \cos(\phi); \end{aligned}$$

6-16

# Appendix F Cube sat specifications document

(Reproduced)

## CubeSat Design Specification (CDS)

Revision 9



Release State		Additional Restrictions	
X	Public		
	Internal		Confidential
	Controlled		ITAR
	Work in Progress		
	Inactive		

Revision	Date	Authored by	Notes
8.1	05/26/04	Amy Hutputtanasin	
9	5/15/05	Armen Toorian	Updated specification.

Last printed 3 June 2004

## Overview

The CubeSat Project is an international collaboration of over 40 universities, high schools, and private firms developing picosatellites containing scientific, private, and government payloads. A CubeSat is a 10 cm cube with a mass of up to 1 kg. Developers benefit from the sharing of information within the community. If you are planning to start a CubeSat project, please contact California Polytechnic State University (Cal Poly). Visit the CubeSat website at <http://cubesat.calpoly.edu> for more information.

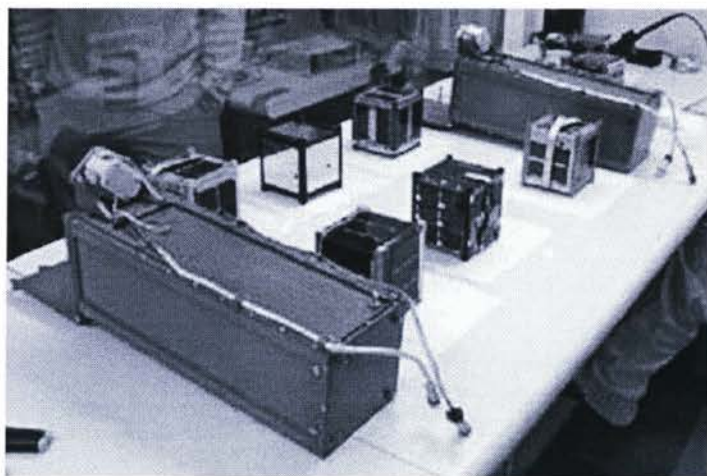


Figure 1: Six CubeSats and their deployment systems.

The primary mission of the CubeSat Program is to provide access to space for small payloads. The primary responsibility of Cal Poly as a launch coordinator is to ensure the safety of the CubeSats and protect the launch vehicle (LV), primary payload, and other CubeSats. CubeSat developers should play an active role in ensuring the safety and success of CubeSat missions by implementing good engineering practice, testing, and verification of their systems. Failures of CubeSats, the P-POD, or interface hardware can damage the LV or a primary payload and put the entire CubeSat Program in jeopardy. As part of the CubeSat Community, all participants have an obligation to ensure safe operation of their systems and to meet the design and testing requirements outlined in this document.



### P-POD Interface

The Poly Picosatellite Orbital Deployer (P-POD) is Cal Poly's standardized CubeSat deployment system. It is capable of carrying three standard CubeSats and serves as the interface between the CubeSats and L.V. The P-POD is an aluminum, rectangular box with a door and a spring mechanism. CubeSats slide along a series of rails during ejection into orbit. CubeSats must be compatible with the P-POD to ensure safety and success of the mission, by meeting the requirements outlined in this document. Additional unforeseen compatibility issues will be addressed as they arise.

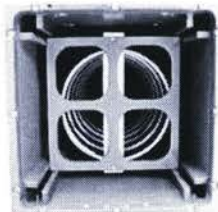


Figure 1: Cross section of the P-POD.

### General Responsibilities

1. CubeSats **must not** present any danger to neighboring CubeSats in the P-POD, the L.V., or primary payloads:
  - All parts must remain attached to the CubeSats during launch, ejection and operation. No additional space debris may be created.
  - CubeSats must be designed to minimize jamming in the P-POD.
  - Absolutely no pyrotechnics are allowed inside the CubeSat.
2. NASA approved materials should be used whenever possible to prevent contamination of other spacecraft during integration, testing, and launch.
3. The newest revision of the CubeSat Specification is always the official version
  - Developers are responsible for being aware of changes.
  - Changes will be made as infrequently as possible bearing launch provider requirements or widespread safety concerns within the community.
  - Cal Poly will send an update to the CubeSat mailing list upon any changes to the specification.
  - CubeSats using an older version of the specification *may* be exempt from implementing changes to the specification on a case-by-case basis.



Figure 1: Poly Picosatellite Orbital Deployer (P-POD)

Cal Poly holds final approval of all CubeSat designs. Any deviations from the specification must be approved by Cal Poly launch personnel. **Any CubeSat deemed a safety hazard by Cal Poly launch personnel may be pulled from the launch**

### Dimensional and Mass Requirements

CubeSats are cube shaped picosatellites with a nominal length of 100 mm per side. Dimensions and features are outlined in the CubeSat Specification Drawing (Attachment 1).

- 1). General features of all CubeSats are:
  - Each single CubeSat may not exceed 1 kg mass.
  - Center of mass must be within 2 cm of its geometric center.
  - Double and triple configurations are possible. In this case allowable mass 2 kg or 3 kg respectively. Only the dimensions in the Z axis change (227 mm for doubles and 340.5 mm for triples). X and Y dimensions remain the same.



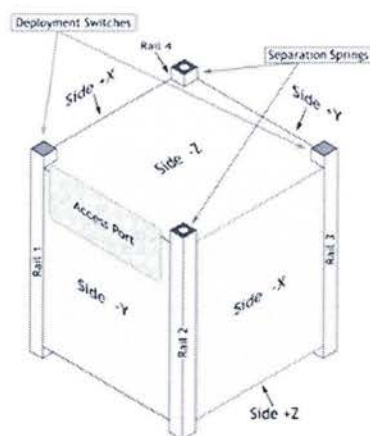


Figure 2: CubeSat isometric drawing.

### Structural Requirements

The structure of the CubeSat must be strong enough to survive maximum loading defined in the testing requirements and cumulative loading of all required tests and launch. The CubeSat structure **must** be compatible with the P-POD.

- Rails must be smooth and edges must be rounded to a minimum radius of 1 mm.
- At least 75% (85.125 mm of a possible 113.5mm) of the rail must be in contact with the P-POD rails. 25% of the rails may be recessed and **NO** part of the rails may exceed the specification.
- All rails must be hard anodized to prevent cold-welding, reduce wear, and provide electrical isolation between the CubeSats and the P-POD.
- Separation springs must be included at designated contact points (Attachment 1) Spring plungers are recommended (McMaster-Carr P/N: 84985A76 available at <http://www.mcmaster.com>). A custom separation system may be used, but must be approved by Cal Poly launch personnel.
- The use of Aluminum 7075 or 6061-T6 is suggested for the main structure. If other materials are used, the thermal expansion must be similar to that of Aluminum 7075-T73 (P-POD material) and approved by Cal Poly launch personnel.
- Deployables must be constrained by the CubeSat. The P-POD rails and walls are **NOT** to be used to constrain deployables.



Figure 3: Spring plunger.

## Electrical Requirements

Electronic systems must be designed with the following safety features:

- No electronics may be active during launch to prevent any electrical or RF interference with the launch vehicle and primary payloads. CubeSats with rechargeable batteries must be fully deactivated during launch or launch with discharged batteries.
- One deployment switch is required (two are recommended) for each CubeSat. The deployment switch should be located at designated points (Attachment 1).
- Developers who wish to perform testing and battery charging after integration must provide ground support equipment (GSE) that connects to the CubeSat through designated data ports (Attachment 1).
- A remove before flight (RBF) pin is required to deactivate the CubeSats during integration outside the P-POD. The pin will be removed once the CubeSats are placed inside the P-POD. RBF pins must fit within the designated data ports (Attachment 1). RBF pins should not protrude more than 6.5 mm from the rails when fully inserted.

## Operational Requirements

CubeSats must meet certain requirements pertaining to integration and operation to meet legal obligations and ensure safety of other CubeSats.

- CubeSats with rechargeable batteries must have the capability to receive a transmitter shutdown command, as per FCC regulation.
- To allow adequate separation of CubeSats, antennas may be deployed 15 minutes **after** ejection from the P-POD (as detected by CubeSat deployment switches). Larger deployables such as booms and solar panels may be deployed 30 minutes **after** ejection from the P-POD.
- CubeSats may enter low power transmit mode (LPTM) 15 minutes **after** ejection from the P-POD. LPTM is defined as short, periodic beacons from the CubeSat. CubeSats may activate all primary transmitters, or enter high power transmit mode (HPTM) 30 minutes **after** ejection from the P-POD.
- Operators must obtain and provide documentation of proper licenses for use of frequencies. For amateur frequency use, this requires proof of frequency coordination by the International Amateur Radio Union (IARU). Applications can be found at [www.iaru.org](http://www.iaru.org).
- Developers must obtain and provide documentation of approval of an orbital debris mitigation plan from the Federal Communications Commission (FCC). Contact Robert Nelson at [rnelson@fcc.org](mailto:rnelson@fcc.org).
- Cal Poly will conduct a minimum of one fit check in which developer hardware will be inspected and integrated into the P-POD. A final fit check will be conducted prior to launch. The CubeSat Acceptance Checklist (CAC) will be used to verify compliance of the specification (Attachment 2). Additionally, periodic teleconferences, videoconferences, and progress reports may be required.

### Testing Requirements

Testing must be performed to meet all launch provider requirements as well as any additional testing requirements deemed necessary to ensure the safety of the CubeSats and the P-POD. All flight hardware will undergo qualification and acceptance testing. The P-PODs will be tested in a similar fashion to ensure the safety and workmanship before integration with CubeSats. At the very minimum, all CubeSats will undergo the following tests.

- Random vibration testing at a level higher than the published launch vehicle envelope outlined in the MTP.
- Thermal vacuum bakeout to ensure proper outgassing of components. The test cycle and duration will be outlined in the MTP.
- Visual inspection of the CubeSat and measurement of critical areas as per the CubeSat Acceptance Checklist (CAC).

### Qualification

All CubeSats must survive qualification testing as outlined in the Mission Test Plan (MTP) for their specific launch. The MTP can be found on the CubeSat website. Qualification testing will be performed at above launch levels at developer facilities. In some circumstances, Cal Poly can assist developers in finding testing facilities or provide testing for the developers. A fee may be associated with any tests performed by Cal Poly. CubeSats must **NOT** be disassembled or modified after qualification testing. **Additional testing will be required if modifications or changes are made to the CubeSats after qualification.**

### Acceptance

After delivery and integration of the CubeSats, additional testing will be performed with the integrated system. This test assures proper integration of the CubeSats into the P-POD. Additionally, any unknown, harmful interactions between CubeSats may be discovered during acceptance testing. Cal Poly will coordinate and perform acceptance testing. No additional cost is associated with acceptance testing. After acceptance testing, developers may perform diagnostics through the designated P-POD diagnostic ports, and visual inspection of the system will be performed by Cal Poly launch personnel. The P-PODs **WILL NOT** be deintegrated at this point. If a CubeSat failure is discovered, a decision to deintegrate the P-POD will be made by the developers in that P-POD and Cal Poly based on safety concerns. The developer is responsible for any additional testing required due to corrective modifications to deintegrated CubeSats.

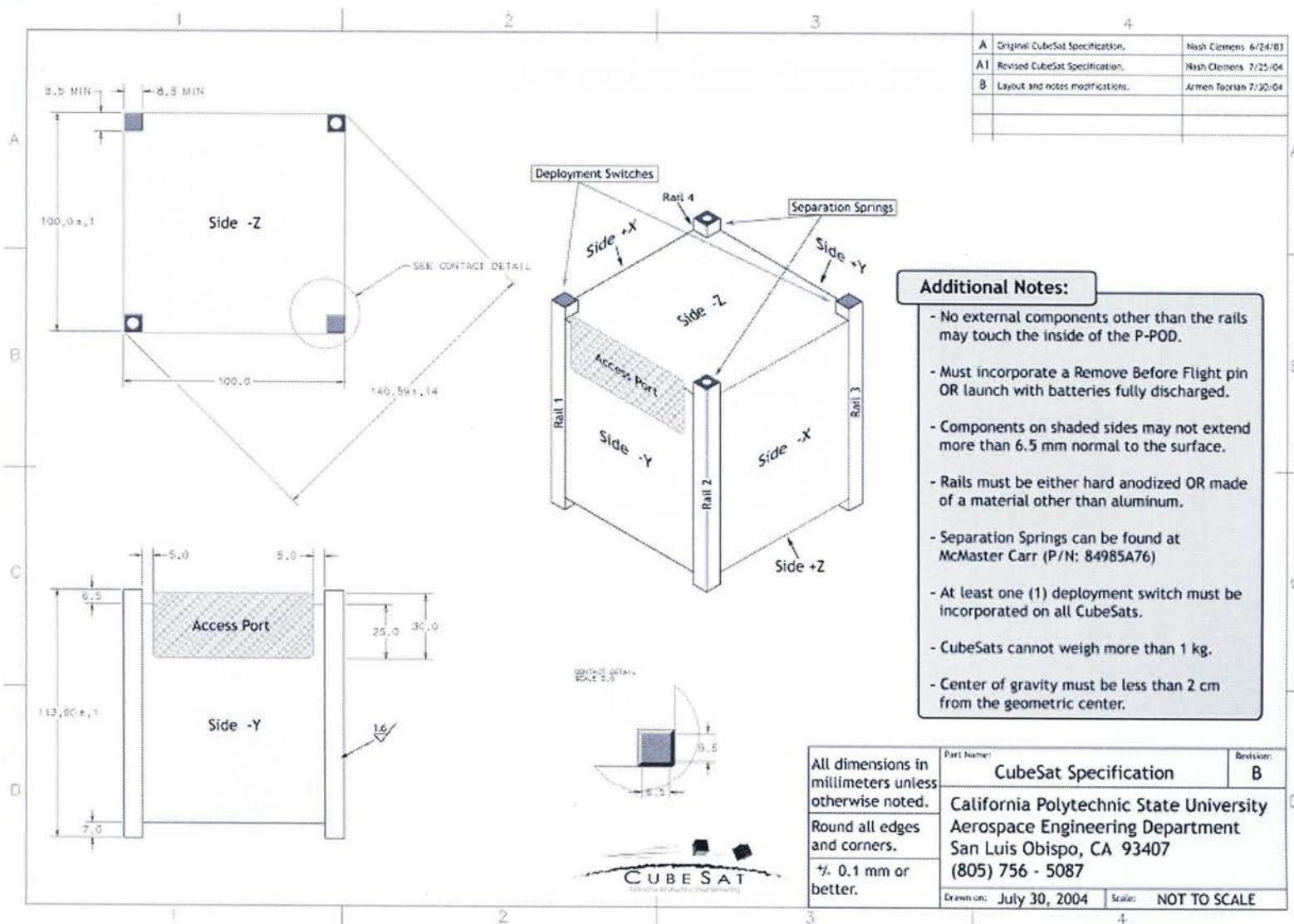
### Contacts

Cal Poly, San Luis Obispo  
Prof. Jordi Puig-Suan  
Aerospace Engineering Dept.  
(805) 756-5087  
(805) 756-2376 fax  
jpuiqsua@calpoly.edu

Student Contacts:  
Simon Lee  
slee@calpoly.edu  
Armen Toorian  
atorian@calpoly.edu

Stanford University  
Prof. Bob Twigg, Director  
Space Systems Development Lab. (SSDL)  
Dept. of Aeronautics and Astronautics  
(650) 723-8651  
(650) 723-1685 fax  
btwigg@leland.stanford.edu



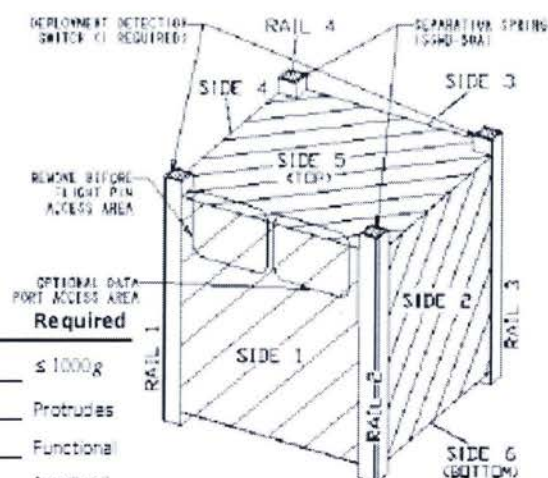




## Cubesat Acceptance Checklist

Revision Date: April 4, 2004  
Author: Armin Topalian

This document is intended to be used concurrently with the Cubesat Integration Procedure (CIP)



List Item	Actual	Required
Mass	_____	≤ 1000g
Remove Before Flight	_____	Protrudes
Spring Plungers	_____	Functional
Rails	_____	Anodized
Deployment Switches	_____	Functional
<b>Width [x-y]</b>		
Side 1	_____	100.0 ± 0.1mm
Side 2	_____	100.0 ± 0.1mm
Side 3	_____	100.0 ± 0.1mm
Side 4	_____	100.0 ± 0.1mm
<b>Height [z]</b>		
Rail 1	_____	113.5 ± 0.1mm
Rail 2	_____	113.5 ± 0.1mm
Rail 3	_____	113.5 ± 0.1mm
Rail 4	_____	113.5 ± 0.1mm
<b>Diagonal [x-y]</b>		
Top 1&3	_____	141.2 <sup>+0</sup> <sub>-1.5</sub> mm
Top 2&4	_____	141.2 <sup>+0</sup> <sub>-1.5</sub> mm
Bottom 1&3	_____	141.2 <sup>+0</sup> <sub>-1.5</sub> mm
Bottom 2&4	_____	141.2 <sup>+0</sup> <sub>-1.5</sub> mm

Authorized By:

IT #1: \_\_\_\_\_

IT #2: \_\_\_\_\_

Testing Info:

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

Passed: Y / N