Electronic Architecture and Technology Development of Astronaut Spaceflight Load Sensors

by

Sylvie Loday

French Engineering Diploma, Electrical Engineering
Ecole Supérieure d'Electricité, Paris, France, 2000

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

February 2001

Signature of Author .... [illegible]
Department of Electrical Engineering and Computer Science
January 12, 2001

Certified by .... [illegible]
Dava J. Newman, Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .... [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Electronic Architecture and Technology Development of Astronaut Spaceflight Load Sensors

by

Sylvie Loday

Submitted to the Department of Electrical Engineering and Computer Science on January 12, 2001 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering and Computer Science

ABSTRACT

Achieving a microgravity environment for scientific experiments is one of the primary objectives of the International Space Station (ISS). While disturbances to the spacecraft due to the vibration of mechanical equipment and electronics systems can be well estimated by mathematical models, the disturbances due to astronaut intra-vehicular activity (IVA) is far more difficult to predict due to the inherent randomness.

Previous experiments aboard the Space Shuttle and the Russian Mir Space Station provided a quantification of nominal astronaut reactions applied to a spacecraft and a description of typical astronaut motions in microgravity. As the ISS becomes operational, there is a need to further quantify astronaut IVA and to help astronauts optimize their motions in microgravity to maintain a quiescent environment inside the Space Station. The Microgravity Investigation and Crew Reactions in 0-G (MICR0-G) research effort proposes to develop for the ISS a prototype of advanced load sensors providing the same functionality as the foot loops and the hand rails on the Space Shuttle and ISS, but able to record the applied forces and moments along the *x*-, *y*- and *z*- axes (6 degrees-of-freedom).

This thesis describes the engineering work that has been carried on at MIT within the framework of the MICR0-G project. A historical overview of the research that has been conducted to quantify the astronaut-spacecraft interaction is provided to better understand the motivation and objectives of the MICR0-G project. This thesis concentrates on the design of advanced load sensors for use on the ISS and the technology development of an on-ground prototype system. The advanced load sensor prototype has been tested in microgravity aboard the KC-135 aircraft during parabolic flights in January 2001. The in-flight operations are detailed and early post-flight results are provided.

Thesis Supervisor: Dava J. Newman, Ph.D.
Title: Associate Professor of Aeronautics and Astronautics

ACKNOWLEDGMENTS

My foremost thanks are due to my advisor Professor Dava Newman. I am deeply grateful that you provided me with the chance to work with NASA on an exciting project and offered me the financial support that made my studies at MIT and my life in Boston so much easier. Thank you also for giving me the invaluable opportunity to fly aboard the KC-135 aircraft: it was so exciting!

My deepest admiration and gratitude goes to Amir. You did a terrific work on EDLS and all the documents and articles you wrote were an inestimable help for me through my research effort. Thanks for all your technical and personal advice, and above all for becoming such a good friend.

Jesse, I want to congratulate you for your invaluable contribution to the MICR0-G project. I learned a lot from your scientific knowledge and it was a great pleasure to work with you. The software development was a challenging part and you completed it wonderfully! Charlie, your enthusiasm for the project and your bright made it very pleasant to work with you. I thank you both for your enormous help on this project and for having been such a great team.

I am very grateful to Ed Bokhour, who assisted me all along the project and in particular designed the signal conditioning board. You always had time for me to answer all my questions and helped me out with building most of the hardware. You really became a technical mentor for me and nothing would have been possible without your tremendous support.

Several faculty members at MIT have helped me throughout this project and I wanted to express my sincere appreciation to them. I am grateful to Steven Schwartz, a research scientist at the Media Lab, for being the source of great ideas and conveying so much happiness. Thanks are also due to Paul Bauer for his valuable technical advice and to Don Weiner for helping me building the Lexan boxes. I am also indebted to Jennie Leith for her help and efficiency.

To prepare for the experiment aboard the KC-135 aircraft, I worked with several people from the Johnson Space Center. Thanks to Noel Skinner, who advised me on the technical preparation for the parabolic flights. Mike Fox has been an invaluable help in completing the medical requirements and physiological training.

I had the chance to work in collaboration with Politecnico di Milano University in order to integrate and test our advanced load sensor prototype in conjunction with their motion analyzer ELITE-S. Guido, it was a real pleasure to work with you. Thanks for your help in preparing the flight experiment and your enthusiasm for our "kindacool" project ;-).

Bastien, I would like to express you my deep affection. The best moments of my life, I spent them with you. You convinced me to come to Boston and gave me the strength and optimism to enjoy this experience. You are the most important person of my life and what I am, I owe to you.

Last but not least, Joe Saleh. The first day I met you, you gave me an invaluable gift. Since then, you became my officemate and my best friend. You always found the words to make me feel better and believe that all I experienced was an inestimable chance. Thank you so much for your time, advice, support, presence and love.

TABLE OF CONTENTS

Appendices

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AC	Alternating Current
A/D	Analog-to-Digital
ASI	Italian Space Agency
ATM	Apollo Telescope Mount
AUP	Advanced Undergraduate Project
BNC	Barrel Nut Connector
CC	Central Computer
CCD	Charge Coupled Device
CMG	Control Moment Gyro
CNES	Centre National d'Etudes Spatiales
COF	Columbus Orbital Facility
COTS	Commercial Off-the-Shelf
COUHES	Committee On the Use of Humans as Experimental Subjects
CPU	Central Processing Unit
CRT	Cathodic Ray Tube
D/A	Digital-to-Analog
DC	Direct Current
DLS	Dynamic Load Sensors
dof	degree(s) of freedom
DOS	Disk Operating System
EDAC	Error Detection And Correction
EDLS	Enhanced Dynamic Load Sensors
ELITE	Elaboratore di Immagini Televisive
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
ESA	European Space Agency
ESM	Experiment Support Module
EXPRESS	EXpedite the PRocessing of Experiments to Space Station
FD	Flight Day
FR	Foot Restraint
GNU	Gnu's Not Unix
GPS	Global Positioning System
GUI	Graphical User Interface
HH	Handhold
HP	Hewlett Packard
HTML	Hypertext Markup Language
IDE	Integrated Device Electronics
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input / Output
IR	Infrared
IRB	Institutional Review Board
ISA	Industry Standard Architecture
ISS	International Space Station
IVA	Intra-Vehicular Activity

JEM	Japanese Experiment Module
JSC	NASA Johnson Space Center
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LEO	Low Earth Orbit
LILO	LInux LOader
MASU	Mir Auxiliary Sensor Unit
MICR0-G	Microgravity Investigations and Crew Reaction in 0-Gravity
MIT	Massachusetts Institute of Technology
MODE	Middeck 0-Gravity Dynamics Experiment
MWNE	Mir Wireless Network Experiment
NASA	National Aeronautics and Space Administration
NetBEUI	NetBIOS Extended User Interface
NetBIOS	Network Basic Input / Output System
NTP	Network Time Protocol
OS	Operating System
PCB	Printed Circuit Board
PCMCIA	Personal Computer Memory Card International Association
PC	Personal Computer
PCS	Pointing Control System
PDA	Portable Digital Assistant
PI	Principal Investigator
PSD	Power Spectral Density
PSI	Payload Systems Inc.
R&MA	Restraint & Mobility Aid
RAM	Random Access Memory
RF	Radio Frequency
RG	Radial Ground
RMS	Root Mean Square
RSA	Russian Space Agency
RTOS	Real-Time Operating System
SCB	Signal Conditioning Board
SEU	Sensor Electronics Unit
SMU	Sensor Mechanical Unit
SNB	SubnoteBook
SSC	Station Support Computer
SSD	Solid State Disk
STS	Space Transportation System
TCP/IP	Transmission Control Protocol/ Internet Protocol
TEA	Torque Equilibrium Attitude
TEDP	Test Equipment Data Package
TP	Touchpad
TRR	Test Readiness Review
TVC	Television Camera
VGA	Video Graphics Array
WNS	Wireless Network Experiment

CHAPTER

1

INTRODUCTION

This thesis describes the technology development of an advanced load sensor ground-based prototype and details the preliminary tests in microgravity during parabolic flights. The research effort has been conducted within the framework of the Microgravity Investigation and Crew Reactions in 0-G (MICR0-G), a research project funded by the National Aeronautics and Space Administration (NASA) organization. The MICR0-G project was a follow-on to the Enhanced Dynamic Load Sensors (EDLS) spaceflight experiment flown on the Russian Space Station Mir. The technology development of the advanced load sensor prototype has been carried out by the Massachusetts Institute of Technology (MIT), with collaboration from Politecnico di Milano University and the Italian Space Agency (ASI). The key hardware of the advanced sensor prototype is a set of three types of load sensors – a handhold, a foot restraint and a touchpad – similar in appearance to the mobility aids found in the Space Shuttle orbiter to assist the crew in moving inside the spacecraft, but able to measure the applied forces and moments about the *x*-, *y*-, and *z*- axes.

The aim of Chapter 1 is to give a brief overview of the thesis' content. The first section summarizes the previous research efforts on astronaut-induced loads in microgravity. The second section provides information on the MICR0-G research project and the technology development work conducted at MIT. Section 1.3 details the motivation for designing a new generation of load sensors and describes the main improvements of the advanced load sensors design compared to the EDLS system. Finally, the last section presents the outline of the thesis.

1.1 Astronaut Loads and Motions in Microgravity

Achieving a quiescent microgravity environment for scientific experiments is a key objective for spaceflight missions. Today, many research activities require an absolutely quiescent environment of 10^{-6} g (g or G are used through the thesis for gravity (9.8 m.s^{-2})) or less. As a result, vibrations and disturbances induced by the equipment and the astronauts should be minimized to assure that material science, physics, biological and medical experiments have the best possible conditions in orbit.

While disturbances inside the spacecraft due to the operation of mechanical systems can be well estimated by mathematical models, predicting astronaut-induced disturbances represents a far more challenging task. In early investigations, the effects of astronaut motions on the spacecraft consisted of modeling an astronaut as a point source mass and the spacecraft as a rigid body and analyzing their dynamic interaction [Roberson, 1963]. Through ground-based experiments, it became possible to define a set of typical astronaut motions in weightlessness and estimate the magnitude and the energy of the corresponding loads produced by the astronauts [Fuhrmeister, 1966]. The first spaceflight experiment carried out to measure crew-induced disturbances was the Skylab Experiment T-013 in 1973, in which an astronaut executed a set of pre-determined motions to measure the resulting forces. The experiment showed astronaut loads from a few Newtons to 400 Newtons and torques up to 1000 Newton·meters [Conway, 1976].

The Dynamic Load Sensors (DLS) experiment was conducted aboard the Space Shuttle in March 1994 to quantify the forces and moments exerted by the astronauts on the spacecraft during routine on-orbit operations [Van Schoor, 1992]. From over 67 hours of DLS force/moment data recording, 301 distinct astronaut motions were identified and led to an average peak force level of 53 N with a 41 N standard deviation [Amir, 1999 B]. The Enhanced Dynamic Load Sensors (EDLS) experiment was proposed as a follow-on to DLS to expand the database on astronaut-induced disturbances. The EDLS spaceflight experiment was conducted on the Russian Mir Space Station from May 1996 to May 1997, as part of the US-Russian cooperation and Phase I of the International Space Station (ISS) project. The EDLS experiment was performed in larger modules than the DLS experiment, involved more astronauts and force/moment data were acquired during long duration spaceflight missions, yielding a significant database from astronaut intravehicular activity (IVA). The EDLS experimental hardware was returned to Earth on flight STS-89 in January 1998 and the EDLS spaceflight data has been analyzed [Amir, 1998]. From the overall 4,480 events identified for the EDLS data, 96% of the time the maximum force magnitude was below 60 N and about 99% of the time the maximum force magnitude was below 90 N [Amir, 2000].

1.2 The MICR0-G Project

The International Space Station is the first space vehicle designed with microgravity requirements. As a result there is a great need to predict, quantify and limit to the extent possible the disturbances to the microgravity environment due to astronaut activities inside space modules. The main objective of the MICR0-G project is to develop a ground-based prototype of advanced load sensors that quantify astronaut activity inside the ISS [Newman, 1997 B]. At the same time, the Italian Space Agency (ASI) supports the development of the ELITE-S2 system, an enhanced version of the real-time opto-electronic motion analyzers ELITE-S and Kinelite, flown respectively on the EuroMir '95 Mission and on the Neurolab Space Shuttle mission [Ferrigno, 1985]. The MICR0-G and ELITE-S2 systems will be integrated and space-qualified to ultimately equip the ISS with a system capable of precisely measuring the forces and moments that the astronauts apply during IVA motions, as well as analyzing their postures and movements.

A ground-based technology demonstrator of the advanced load sensor system has been developed at MIT and tested in conjunction with the ELITE-S system during parabolic flights in January 2001. This experiment provided early data acquired in microgravity and useful information to make the final modifications and finalize the ground-based system development. In the future, a formal proposal will be submitted to NASA to pursue integrating the advanced load sensors with the ELITE-S2 motion analyzer, space-qualifying the integrated system, and manifesting the final version on the ISS.

1.3 Motivation

This thesis describes the technology development of an advanced load sensor prototype for the ISS to aid the astronauts in the operation of the station while measuring the applied forces and torques in the *x*-, *y*- and *z*- axes (6 degrees of freedom). The MICR0-G advanced sensor prototype uses the same load sensors as the EDLS system but incorporates miniaturized electronics and enhances the overall capability of the kinetic system.

The DLS/EDLS experimental hardware was designed in the late 1980s as a kinetic data acquisition and storage system to support space shuttle middeck experiments. The EDLS experiment used two foot restraints, a handhold and a touchpad. The four sensors were connected via long umbilical cables to the Experiment Support Module (ESM) processing and storing the data acquired by each sensor. All ESM elements were contained within a rigid aluminum case of 49.1 cm x 22.8 cm x 41.5 cm. The MICR0-G load sensor prototype makes use of the latest advancements in the electronics field to miniaturize the system so that the entire data acquisition and processing components are incorporated into a sensor itself. The redundancy of the electronics increases the ability of the system to recover from a failure and enhances its flexibility since potentially any number of sensors can be used. Using four sensors as in the EDLS experiment, the advanced sensor electronics achieves a reduction in size and mass by a factor of four in comparison to the ESM module. The advanced load sensor system provides real-time feedback to the astronauts on the magnitude of the reaction forces they cause. Immediate feedback is a valuable tool for the crew in determining whether they need to adjust their motion or not. A major enhancement of the MICR0-G kinetic system compared to the EDLS system is the use of a wireless Radio Frequency (RF) capability for communication between the sensors and a central computer storing the data. The lack of a long umbilical cable from each sensor to an external computer system results in a simpler, safer and mobile system. Aboard the Mir Space Station, the EDLS operator interfaced with a 25-button keypad and a 4 x 20 character alphanumeric display. The interface of the advanced load sensor system is a laptop computer LCD (Liquid Crystal Display) screen and keyboard. Familiar to any astronaut, a laptop interface facilitates the operations and optimizes the time spent on the experiment set-up. Finally, the laptop computer can be easily moved around and be shared with other experiments.

1.4 Thesis Outline

This thesis encompasses the design and development of the advanced load sensor system as well as the experiment aboard the KC-135 microgravity aircraft testing the ground-based sensors prototype together with the Italian ELITE system. Chapter 2 "Previous Research on Astronaut-Induced Disturbances" provides general information on the International Space Station, for which the advanced load sensors have been designed, as well as an overview of the microgravity environment. The chapter ends with a discussion on prior research efforts on astronaut-induced disturbances and details the corresponding key results.

As the name implies, Chapter 3 "Design of the Advanced Load Sensors" concentrates on the design of the advanced load sensor system for the ISS and the development of a ground-based prototype.

Chapter 3 presents the motivation, objectives, and timetable of the MICR0-G project, and then details two innovative design characteristics of the advanced load sensor system: the use of the PC/104 form factor and of wireless communications. Section 3.4 presents the design of the advanced kinetic system for the ISS. Finally, the last two sections introduce the preliminary software development and calibration of the prototype.

Chapter 4 "Flight Operations" presents the experimental protocol designed for and flown aboard the KC-135 aircraft in January 2001 in order to test the advanced load sensor prototype synchronized with the ELITE-S three-dimensional opto-electronic motion analyzer. The chapter begins with a brief overview of the objectives of the flight experiment and the results expected. Then, general information on the KC-135 aircraft and parabolic flights outlines the context of the experiment. The chapter further details the equipment of both systems and explains the experimental set-up as well as in-flight operations. The final section provides recommendations to complete the development of the advanced load sensor system and to ultimately build the space-qualified version for the ISS.

CHAPTER

2

PREVIOUS RESEARCH ON ASTRONAUT-INDUCED DISTURBANCES

This chapter begins with a description of the International Space Station (ISS), the different phases of its assembly and its major characteristics once assembly is complete. The major objective of the International Space Station is to provide a permanent laboratory for scientific research in microgravity. The second section briefly explains the microgravity environment and the importance of conducting experiments in reduced gravity. Astronauts intra-vehicular activity (IVA) as a potential source of disturbance to the microgravity environment is investigated. Section 2.3 reviews previous research on the effects of astronaut motion on a spacecraft and provides details on the Enhanced Dynamic Load Sensors (EDLS) experiment, which is currently the latest space experiment on astronaut-induced disturbances and motions.

2.1 The International Space Station

The International Space Station represents the largest scientific cooperative program in history, drawing on the resources and scientific expertise of sixteen nations – Belgium, Brazil, Canada, Denmark, France, Germany, Italy, Japan, Netherlands, Norway, Russia, Spain, Sweden, Switzerland, United Kingdom, and USA [NASA, 2000 B]. The Space Station will be a test bed for the technologies of the future and a laboratory for research on advanced materials, communications technology, fluids, biology, and medical research.

2.1.1 Space Station Assembly

The ISS with its various capabilities, will be assembled in space step-by-step, combining components carried by more than 40 launch missions. Rockets and Space Shuttles will be used to carry the components to the ISS. In total, 460,000 kg of structure, modules, equipment and supplies will be placed in orbit. To assemble and maintain the ISS, space-walking astronauts will work with a new generation of space robotics. The Space Shuttle's mechanical arm and a new space station arm will operate as space cranes to maneuver large components and serve to maneuver astronauts to work areas. The first component of the ISS was launched on November 20, 1998 from Russia to start its construction, and the objective is to complete the Space Station by 2006 [NASA, 2000 A]. This time frame was divided into three phases corresponding to specific stages in the development and assembly of the ISS. The schedule and purposes of each phase are summarized herein and the portions set in italic identify the main objectives.

Phase I

In preparation for the assembly and operation of the ISS, NASA and the Russian Space Agency (RSA) worked together in a series of missions using U.S. Space Shuttles and the Russian Space Station Mir to gather operational experience and to perform early scientific experiments. Beginning in February 1994, the U.S. Space Shuttle-Mir program, also known as Phase I, was comprised of a number of Space Shuttle-Mir rendezvous and docking missions. Seven U.S. astronauts went to the Russian Mir space station as for long-duration flights and nine Russian cosmonauts flew on U.S. Space Shuttles [Moore, 1998]. Phase I ended in June 1998. The Space Shuttle-Mir Program provided the United States with the chance to carry out long-term experiments in microgravity aboard Mir, for periods of time much longer than the capability of the Shuttle. Phase I allowed U.S. astronauts, in four years, to spend 975 days on Mir. This is more time in orbit than had been previously accumulated since the Space Shuttle program began in 1981.

Many in-orbit life science, microgravity and environmental experiments were performed during Phase I in preparation for the International Space Station [Robey, 1997]. Over 140 scientific experiments were conducted by leading scientists from Canada, France, Hungary, Japan, the United Kingdom and the United States. Research in the life sciences included human metabolic, neurosensory, pulmonary systems and cardiovascular studies. In the microgravity sciences, research was conducted on Mir in fluid dynamics, material science, and combustion science. The Enhanced Dynamic Load Sensors (EDLS) experiment was a Mir Phase I investigation [NASA, 1997 A]. The results of the EDLS experiment provided data on astronaut IVA motions and served as a reference to design an enhanced system of load sensors for the ISS.

Another important objective of the Phase I program was *to learn how to reduce risks dealing with assembling and operating the ISS* [Robey, 1997]. Space operations on Mir have led to improvements in the design of the space station. U.S. astronaut experience on Mir has led to software, hardware and procedural changes. Some of these modifications include single-command shutdown of ventilation systems to prevent fire from spreading, simplified location of medical kits and fire extinguishers, and quick disconnect capability for cables in the event of depressurization of a module. Other changes being made as a result of the experience gained from Phase I are the addition of extra tracking lights to the space station to provide better use of navigation systems during rendezvous operations [NASA, 1998 A]. Phase I provided technical lessons to reduce the risk associated with the construction and operation of the ISS, as well as scientific lessons to enhance long duration experiment performance.

Phase II

Phase II is the first step in the construction of the ISS in orbit. It began with the launch of the Zarya (meaning "sunrise" in Russian) module in November 1998 and the Unity node in December 1998. The first entirely Russian contribution, a component called the Zvezda Service Module, was launched from Russia in July 2000. This component *provides the initial living area and life support systems* [NASA, 2000 D]. After additional assembly and supply flights, a three-person crew (two Russian cosmonauts and an American astronaut) was launched aboard a Russian capsule in October 2000. This crew will be replaced by a second three-person crew in February 2001.

Phase III

Phase III will include the final assembly and full operational capabilities of the International Space Station. It will finally be equipped with laboratory modules supplied by the United States, Japan, Europe, and Russia. It will also be equipped with a robot arm supplied by Canada as well as pallets supplied by Brazil. After its completion, a crew of up to seven members will be able to live and work aboard the station.

2.1.2 The ISS Major Characteristics

The International Space Station will be in a circular orbit at a nominal altitude of approximately 400 km and at an inclination of 51.6 degrees, and will usually maintain Torque Equilibrium Attitude (TEA). Attitude control on the ISS can be achieved using non-propulsive effectors such as CMGs

(Control Moment Gyros) and/or propulsive effectors such as service module thrusters [NASDA, 2000]. The ISS will orbit the earth approximately every 90 to 94 minutes. The solar beta angle, which is the angle between the ISS orbit plane and the earth-sun line, will vary between ±75°. A wide array of resources (i.e., power, crew support, payload volume, data communications, etc.) will support scientists and engineers from the governmental, academic and industrial/commercial sectors. The availability of the space station resources and accommodations will be time-phased, allowing for a systematic build-up of instruments and research facilities over a multi-year period. By its completion, the ISS will supply the following research accommodations [NASA, 1998 C]:

- microgravity research laboratories and orbital research platforms.
- multi-user experiment facilities, experiment-specific instruments, and laboratory support equipment.
- transportation systems for the periodic supply and return of research equipment, supplies, and materials.
- power, thermal control, data processing, communications, and associated resources for the operation of experiment apparatus and facilities.
- a worldwide distributed ground systems network for data distribution and remote command, control, and teleoperation of experiment apparatus and facilities to support crew operation.
- worldwide ground-based crew training facilities and continuous presence of trained crew for on-orbit research.

Many payloads, particularly commercial payloads, will fit into specialized experiment racks designed to service a number of diverse, moderate-sized payloads. These racks are named EXPRESS, for EXpedite the PRocessing of Experiments to Space Station. This will provide multiple users with a simple, standard interface to plug into the ISS research capabilities, such as power and data handling. The ISS will support four EXPRESS racks during its period of assembly in support of early research capability. In addition, two EXPRESS pallets will be attached to the outside of the station to support exposed payloads.

The primary areas the astronauts utilize are six "experiment modules" where they work and conduct research, and a habitation module where they live. The six laboratories together are four times larger and more capable than any previous space station. The United States will provide two laboratories (the United States Laboratory and the Centrifuge Accommodation Module) and a habitation module for four crew members. There will be two Russian research modules; one Japanese laboratory referred to as the Japanese Experiment Module (JEM) named “Kibo” (Hope); and one European Space Agency (ESA) laboratory called the Columbus Orbital Facility (COF) [NASA, 1997 B].

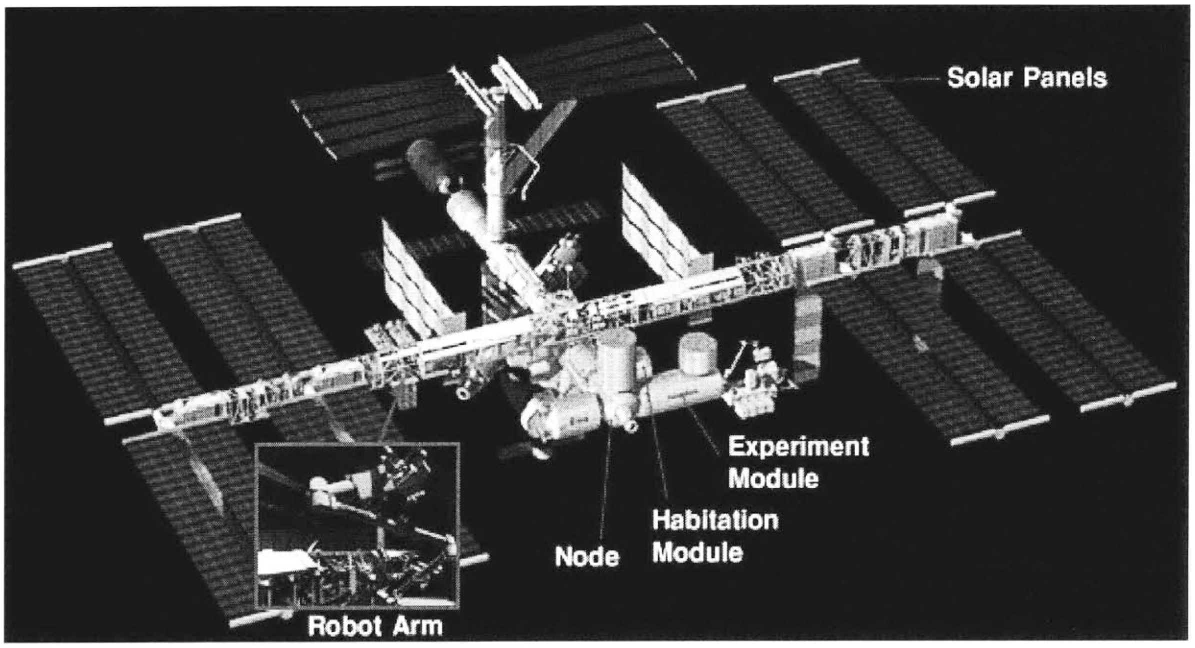


Figure 2.1: The ISS and its main components.

Inside the experiment and habitation modules, the atmosphere is kept almost the same as on the ground so that astronauts work in a shirt sleeve environment. The pressurized living and working space aboard the completed ISS will be roughly equivalent to the passenger cabin volume of two Boeing 747 jets or about three times the Mir space station volume. Solar panels are attached to generate power, and a robot arm will be used to support the tasks outside the ISS.

The ISS major characteristics upon completion are summarized herein [NASA, 1998 C]:

- Volume: 1200 m^3
- Total mass: 470, 000 kg (940,000 pounds)
- Dimensions (including solar panels): 109 m x 80 m
- Orbital inclination: 51.6 degrees
- Average orbital altitude: 400 km
- Atmosphere: 14.7 pounds per square inch (same as on Earth)
- Crew size: up to 7

2.1.3 The ISS Modes

The ISS modes described in Table 2.1 are defined to support the mission objectives accomplishment of all the experiments conducted aboard the ISS [NASA, 1999 B]. An example of an ISS mode cycle is shown in Figure 2.2. Mode changes are usually controlled by the ground controller or by on-orbit crew input commands. However, the transition to survival mode may be automatically initiated by the ISS safety system.

Table 2.1: Summary of the ISS modes [Boeing, 1998].

Mode	Summary
Standard	This mode represents the core operations of the ISS.
Reboost	This mode is used to control the translation maneuver including reboost.
Maneuver	This mode consists of the functionality required to change the ISS attitude orientation.
Microgravity	In this mode, microgravity capability applies as shown in Table 2.2.
Survival	This mode may be initiated when a warning of imminent threat (e.g. attitude, power inadequate) occurs.
Proximity	This mode provides the capabilities related to supporting safe operations with other vehicles.
Assured Safe Crew Return (ASCR)	This mode is used to assure return to earth of ISS crew in the case of a crew life threatening illness and so on.
External Operations	This mode utilizes functionality related to supporting ISS based external operations (EVA and external robotic operations).

User payload operations for microgravity research are supported in the microgravity mode. This mode includes the effects of crew equipment such as exercise devices and latched or hinged enclosures, but "crew effects are mitigated to the extent possible", according to the "System Specification for the ISS" [Boeing, 1998]. The ISS altitude is controlled using only non-propulsive effectors. The ISS can maintain the microgravity environment with an acceleration on the order of 10^{-6} g.

The modes reboost, proximity, and external operations are generally of short-term duration (less than 24 hours). User payload operations are supported in these modes. The survival mode is the only timeframe when the ISS does not assure to support user payload operations.

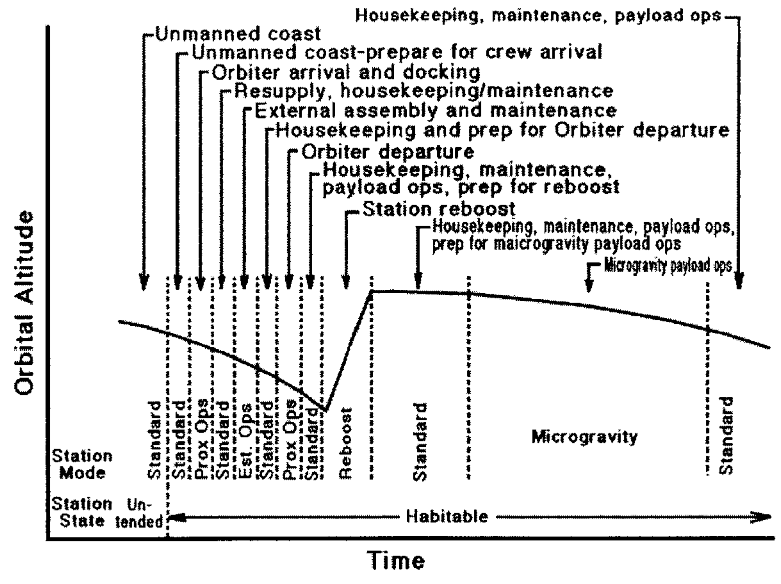
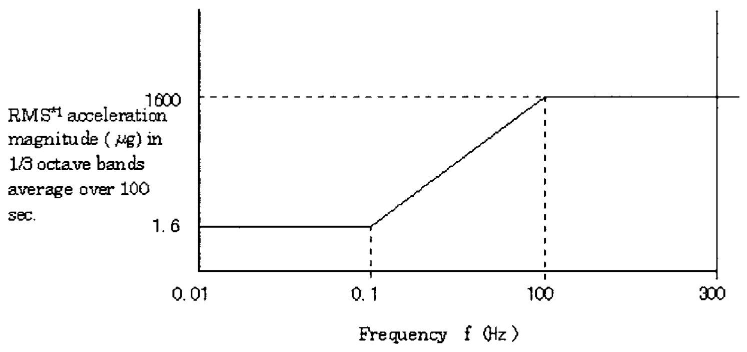


Figure 2.2: The chart shows an example of ISS mode cycles [NASDA, 2000].

Table 2.2: Acceleration performance in the microgravity mode [NASDA, 2000]

Quasi-steady acceleration	Acceleration magnitude limit	Less than or equal to 1.0 μg
	Applied locations	At least 50% of the centers of the internal payload locations.
Vibratory acceleration		Apply to total combined acceleration level produced by all the disturbance sources (without payloads) which occur simultaneously.  $0.01 \leqq f \leqq 0.1$ Hz : $a \leqq 1.6$ μg $0.1 < f \leqq 100$ Hz : $a \leqq f \times 16$ μg $100 < f \leqq 300$ Hz : $a \leqq 1600$ μg
	Transient acceleration limit	Apply to acceleration level produced by an individual transient disturbance source - less than or equal to 1000 μg/axis - less than or equal to 10 μgsec/axis over a 10 sec. interval
	Applied locations	At least 50% of the structural mounting interfaces to the internal payload locations.

2.2 The Microgravity Environment

2.2.1 What is Microgravity?

Microgravity is defined as a condition where the apparent weight of an object is much less than its true weight (on Earth). In other words, it is a term describing apparent weightlessness or reduced weight. Apparent weight is the weight measured when an object is put on a scale in a given environment, where true weight is the force of Earth's gravity on the object. [NASA, 1998 D]

Microgravity occurs where gravity is low. However, to reach a point where Earth's gravity is reduced to one-millionth (10^{-6} g) of that on the Earth's surface, one would have to be 6.37 million kilometers away from Earth (almost 17 times further away than the Moon). Since spacecraft usually orbit only 200-450 km above the Earth's surface, the microgravity environment found aboard these vehicles is not due to the absence of gravity (e.g., the Space Shuttle operating at 350 km above the Earth's surface, experiences about 90% of the gravity on the surface of Earth). Rather it is due to free-fall [NASA, 1998 E].

Free-fall occurs when an object falls toward the Earth with an acceleration equal to that due to gravity alone (approximately 9.8 m/s^2, or 1 g at the Earth's surface). The microgravity environment associated with spacecraft is the result of being in orbit, which is a state of continuous free-fall around the Earth. A circular orbit results when the centripetal acceleration of uniform circular motion (v^2/r; v = velocity of the object, r = distance from the center of the object to the center of the Earth) is the same as that due to gravity alone [NASA, 1997 D].

The absence of a gravity effect is called weightlessness or zero-g. In practice, zero-g cannot be achieved, since an orbiting spacecraft is subject to various small forces produced by the space environment. As a result, for most practical applications in Low Earth Orbit (LEO), which implies altitudes from 150 to 600 km, the gravitational effect can be reduced to 10^{-6}g ($=1\mu$g); a level of 10^{-7}g can be achieved over a very small region near the center of mass of the spacecraft. For this reason the term *microgravity* (μg) rather than zero-g is best used to precisely describe the condition in orbit.

Brief periods of microgravity can be achieved on Earth by dropping objects from tall structures. Longer periods are created through the use of airplanes, rockets, and spacecrafts [NASA, 1996]. These different methods are described below and illustrated in Figure 2.3.

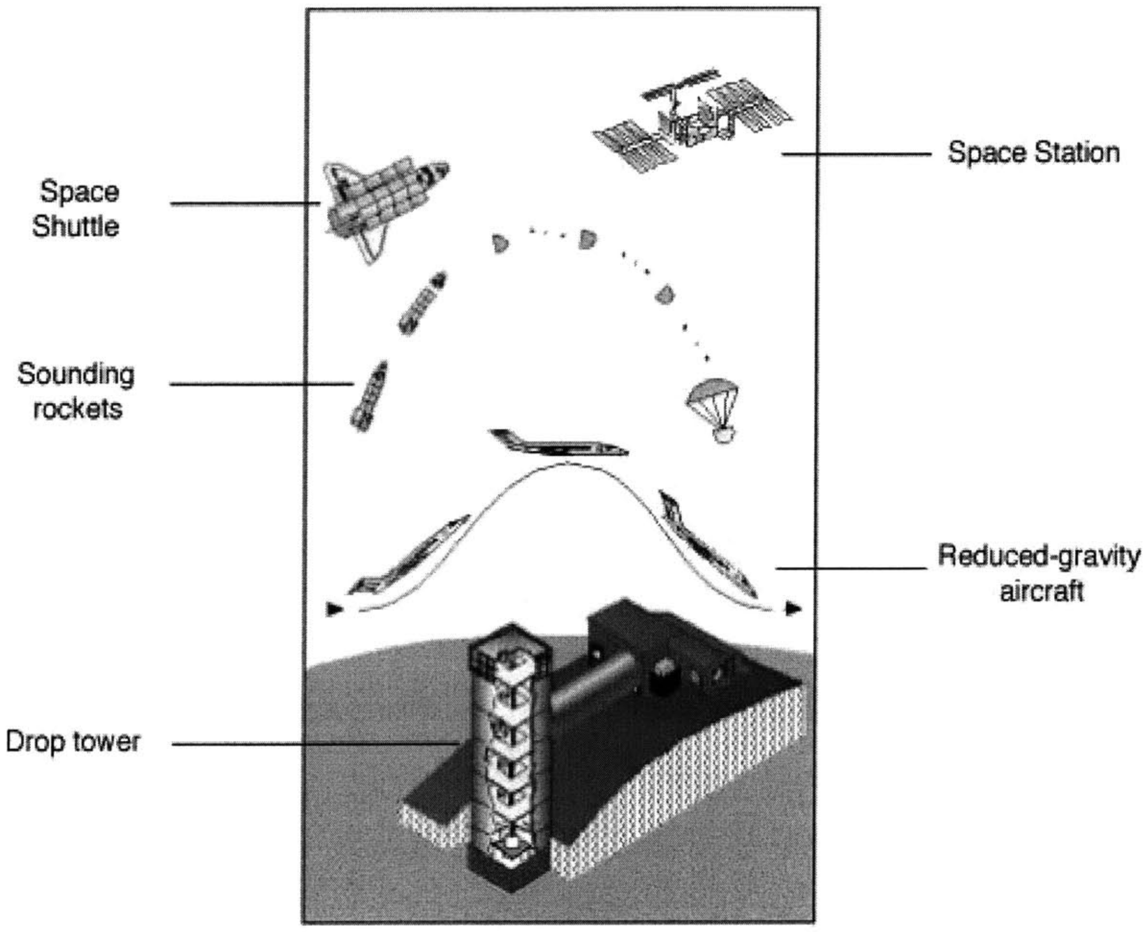


Figure 2.3: Illustration of the facilities used to achieve a microgravity environment [NASA, 1996].

- **A drop tower** is a long vertical shaft used for dropping experiment packages, enabling them to achieve microgravity through free-fall. Various methods are used to minimize or compensate for air drag on the experiment packages as they fall. The Glenn Research Center in Cleveland, Ohio, has two drop facilities (one 24 m tall and one 132 m deep) that can accommodate experiments, which need only a limited amount of time (2.2 or 5.2 s) of microgravity or are test runs of experiments that will later be performed for longer periods in an aircraft, a rocket, or a spacecraft.
- **Reduced-gravity aircrafts** are flown in parabolic arcs to achieve longer periods of microgravity. The airplane climbs rapidly until its nose is at an approximate 45° angle to the horizon. Then the

engines are briefly cut back, the airplane slows, and the nose is pitched down to complete the parabola. As the plane traces the parabola, microgravity conditions are created for 20-25 seconds. As many as 40 parabolic trajectories are performed on a typical flight. Details on the KC-135 reduced-gravity aircraft are provided in Chapter 4.

- **Sounding rockets** produce higher-quality microgravity conditions for longer periods of time than airplanes. An experiment is placed in a rocket and launched along a parabolic trajectory. Microgravity conditions are achieved during the several minutes when the experiment is in free-fall prior to re-entering Earth's atmosphere.
- **A space shuttle** is a reusable launch vehicle that can maintain a consistent orbit and provide up to a month of high-quality microgravity conditions. The shuttle, which can accommodate a wide range of experiment payloads, provides a laboratory environment in which scientists can conduct microgravity investigations.
- **A space station** is a permanent facility that maintains a low Earth orbit for up to several decades. The facility enables scientists to conduct their research in microgravity over a period of several months without having to return the entire laboratory to Earth each time an experiment is completed.

2.2.2 Why Conduct Research in Microgravity?

A microgravity environment provides the basis for a unique laboratory in which scientists can investigate the three fundamental states of matter: solid, liquid, and gas. Microgravity conditions allow scientists to observe and explore phenomena and processes that are normally masked by the effects of Earth's gravity. Microgravity investigations are particularly interested in the fields of life sciences, biotechnology, combustion science, fluid physics, fundamental physics, and material science [NASA, 1997 B], [NASA, 1997 C]. A few examples where microgravity is used to identify new physical phenomena, validate contemporary scientific theories, and develop new theories, are described below [De Lombard, 1996].

Biotechnology

A microgravity environment provides new tools to address two fundamental aspects of biotechnology: the growth of high-quality crystals for the study of proteins, and the growth of three-dimensional tissue samples in laboratory cultures.

- Research has shown that the effects of gravity adversely influence crystals' development. Growing crystals in microgravity yields to substantially better structural information than crystals grown on Earth.
- In the past, gravity-induced sedimentation in cellular cultures made it virtually impossible for researchers to grow representative tissue samples outside of the human body. Often, the result was more two-dimensional than three-dimensional, limiting the sample's usefulness as a research tool. After experimenting in microgravity, scientists have found that a low-gravity environment allows cells to cluster together in three dimensions, often closely resembling the shape such tissue takes in the human body. By using space-based experiments as a model, researchers have developed a "bioreactor" for terrestrial applications that uses horizontal rotation to mimic the microgravity environment. Today, researchers are using the bioreactor to successfully culture tissue samples as diverse as liver, muscle, cartilage, and bone.

Combustion Science

The Earth's gravity causes hot flame gases to rise, leading to unsteady, fast-moving, and distorted flames. Measurements on combustion reactions are therefore difficult. A microgravity environment eliminates gravity-driven buoyant convection thus produces large, steady, slow-moving, and symmetric flames. These well-behaved flames are much easier to study.

Physics of Fluids

Forces such as surface tension or control fluid behavior normally masked by gravity here on Earth, can be studied in microgravity. As a result, many of our intuitive expectations about fluids do not hold up in orbit. For instance, surface tension causes drops of any liquid to form almost-perfect spheres when the influence of gravity is absent. A better understanding of the behavior of fluids will have many applications in materials science, biotechnology, and combustion science. For example, impurities in materials such as glasses and alloys can be reduced by managing fluid behavior while the material is in molten state.

Materials Science

The production processes for most materials include steps that are heavily influenced by the force of gravity. The chance to implement and observe these processes in microgravity promises to increase our fundamental understanding of production operations and of the materials they produce. By carefully studying and controlling the processes by which materials are formed, commercial researchers can design new alloys, ceramics, glasses, polymers, and semiconductors to improve the performance of products ranging from contact lenses to car engines and medical instruments.

Fundamental Science

The ISS will support many different studies in fundamental science, including low-temperature and condensed matter physics, gravitational and relativistic physics, and laser cooling and atomic physics. For example, some studies involve the collection of atoms cooled to extremely low temperatures by slowing down their motion with laser beams (the rate of motion of an atom is directly tied to its temperature). On Earth, these ultra cooled samples would be disturbed by contact with the walls of the holding chamber – contact that will not occur in the microgravity environment of space. In turn, these samples can be used to measure time by observing the light emitted from the atoms (resulting from electronic transitions within the atoms themselves). These clocks will be much more accurate than those operating on Earth. As a result, they may serve the practical and immediate application of a highly accurate clock for corrections to time-dependent systems such as the Global Positioning System (GPS).

2.3 Research on Astronaut-Induced Disturbances

External disturbances to a spacecraft in orbit such as aerodynamic drag or solar pressure can be described in a simple analytical form and estimated well from vehicle and environmental parameters. Similarly, disturbances to the microgravity environment inside the spacecraft due to the operation of mechanical equipment such as pumps, fans, and valves can be foreseen and computed. Predicting astronaut-induced disturbances represents a far more challenging task due to the inherent randomness [Amir, 1999 B]. The assessment of astronaut-induced disturbances has involved scientist expertise for more than 40 years. Mathematical models combined with on-ground and space experiments led to a better understanding of astronaut motions, their adaptation to weightlessness and the side effects of astronaut on-orbit activities to the microgravity environment.

2.3.1 Past Experiments

NASA's initial concern was that the magnitude of the forces and moments exerted by the astronauts inside the modules would be the largest disturbance source to the vehicle's attitude control system and thus represent the design driver for the system [Kullas, 1979]. By the late 1960s/early 1970s, development of mathematical models [Hendricks, 1971], on-ground and flight experiments [Goodman, 1969], together with the development of larger spacecraft, led to the conclusion that astronaut-induced disturbances do not represent the major challenge for the control system.

The prevalent concern became that the magnitude of the forces and moments exerted by the astronauts on a spacecraft or a space station would not permit high precision pointing for astronomical observations. The Crew/Vehicle Disturbance Experiment T-013 on the Skylab Space station in 1973 [Conway, 1972] addressed this issue by collecting force and moment data exerted by the astronauts on the Skylab station. The objective of the experiment was to "assess the characteristics and effects of astronaut crew-motion disturbances aboard a manned spacecraft, and to investigate the response of the Apollo Telescope Mount (ATM) Pointing Control System (PCS) to known disturbance inputs" [Conway, 1974]. The analysis of the data showed that astronauts were able to produce forces close to 500 N if they so desired, but that the average force level for all activities did not exceed 100 N [Conway, 1976]. No quantification of crew-induced reaction forces and moments had been conducted in orbit since the Skylab experiment until DLS was performed on STS-62 in 1994 [Van Schoor, 1992]. Therefore, the results of the Skylab experiment T-013 were taken as a reference on astronaut-induced disturbances for the preliminary design of the ISS. However, other experiments were scheduled at that time because the Skylab results were questionable for various reasons. First, the loads were measured while the astronaut performed a set of prescribed activities, so the data acquired did not reflect the average typical loads applied on the spacecraft. In particular, maximum loads were recorded while the astronaut performed "vigorous soaring" with the clear objective of assessing the worst disturbance case. Second, almost the entire data set was recorded on August 16, 1973 in a single session lasting less than 80 minutes and involving only one astronaut subject [Amir, 1998]. Thus, the T-013 Skylab experiment provided the first data collected in space on astronaut-induced disturbances but is an extremely limited non-representative database.

In the 1980's, the purpose of assessing astronaut-induced disturbances fundamentally changed. Precise astronomical observations became best conducted from non-human tended craft, whereas much of the current microgravity research and scientific experiments benefited greatly from the presence of astronauts. Achieving a microgravity environment for microgravity-sensitive science experiments became the primary motivation for assessing astronauts-induced disturbances [NASA, 1998 E]. As a result, the

DLS (Dynamic Load Sensors) experiment on-board the Space Shuttle in March 1994 proposed to describe, quantify, and predict astronaut motions during normal on-orbit activities in order to provide a more comprehensive space database than the T-013 Skylab experiment [Van Schoor, 1992]. The key hardware components of DLS were a set of three sensors – a touchpad (TP), a foot restraint (FR), and a handhold (HH) – similar in appearance to the devices found in the Orbiter to assist the crew in moving inside the vehicle but designed to record the forces and torques applied by the astronauts [Bokhour, 1992]. Data was recorded over a period of 67 hours during Flight Day 7, 8, and 11 of the 14-day mission. Altogether 301 astronaut motions were found through a correlation with video footage recorded inside the Shuttle Middeck. An examination of the video footage led to the identification of nine characteristic motions that the astronauts performed in the microgravity environment of the Orbiter. The description and characteristics of these motions are provided in Table 2.3 [Amir, 1998].

Table 2.3: Characteristic astronaut motion identified in the DLS experiment [Amir, 1998].

Characteristic Motion	Description of Astronaut Motion	Sensors Used
Landing	Flying across middeck and landing on a sensor.	HH, FR, TP
Push-off	Pushing off a sensor and floating away.	HH, FR, TP
Flexion	Flexing a limb while using a sensor.	HH, FR
Extension	Extending a limb while using sensor.	HH, FR
Double Support	Using two limbs for support.	HH, FR
Single Support	Using only one limb for support.	HH, FR
Twisting	Twisting body motion.	HH, FR
Vertical Orienting	Vertical re-orienting usually during posture control.	HH, FR, TP
Horizontal Orienting	Horizontal re-orienting usually during posture control.	HH, FR, TP

The analysis of the DLS data led to an average peak force level of 53 N and a standard deviation of 41 N. Figure 2.4 shows a histogram of the peak force magnitude for the 301 analyzed events. Two data points, one at 286 N and the other at 466 N, are clearly outliers of the peak force data but the true source of these loads remains unknown. The significance of the astronaut loads is more evident from the root-mean-square (rms) forces shown in Figure 2.5. The average rms was 24 N and the standard deviation 16 N. The distribution of the energy in the astronaut-induced disturbances was determined by computing the frequency below which 95% of the power spectral density is contained (the 95% PSD). As is expected for human motion, the energy is contained in the low-frequency regime, with approximately 86% percent of all astronaut activities having a 95% PSD below 6 Hz. [Amir, 1999 B]

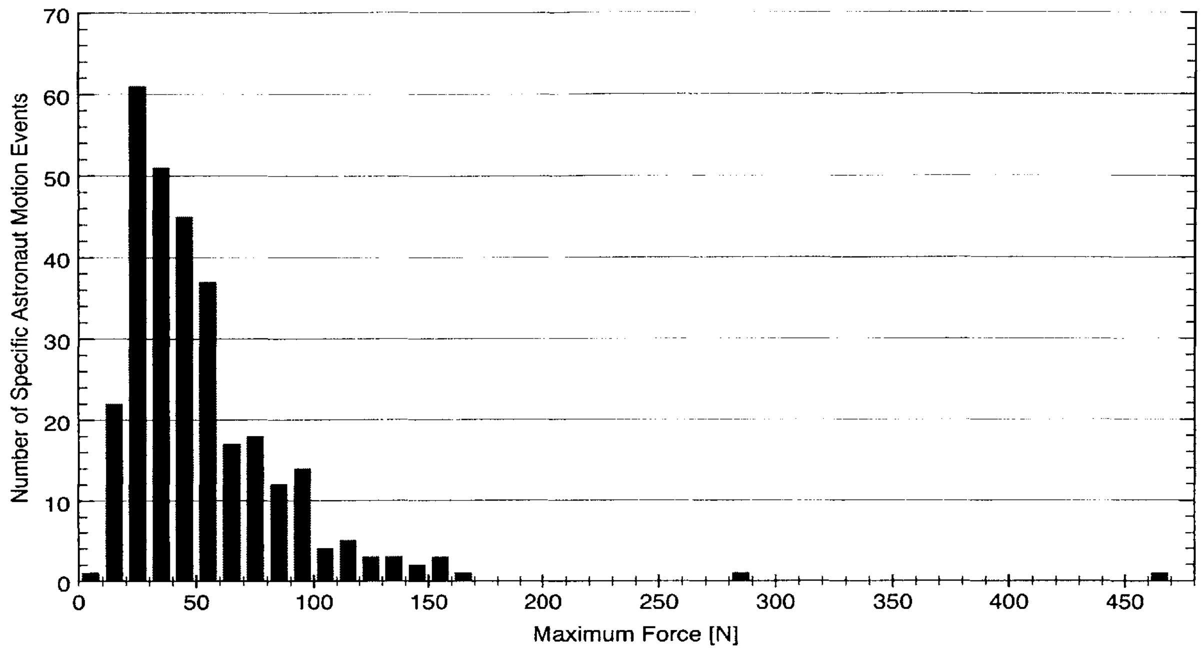


Figure 2.4: Maximum forces recorded in the DLS Experiment on the Space Shuttle [Amir, 1999 B].

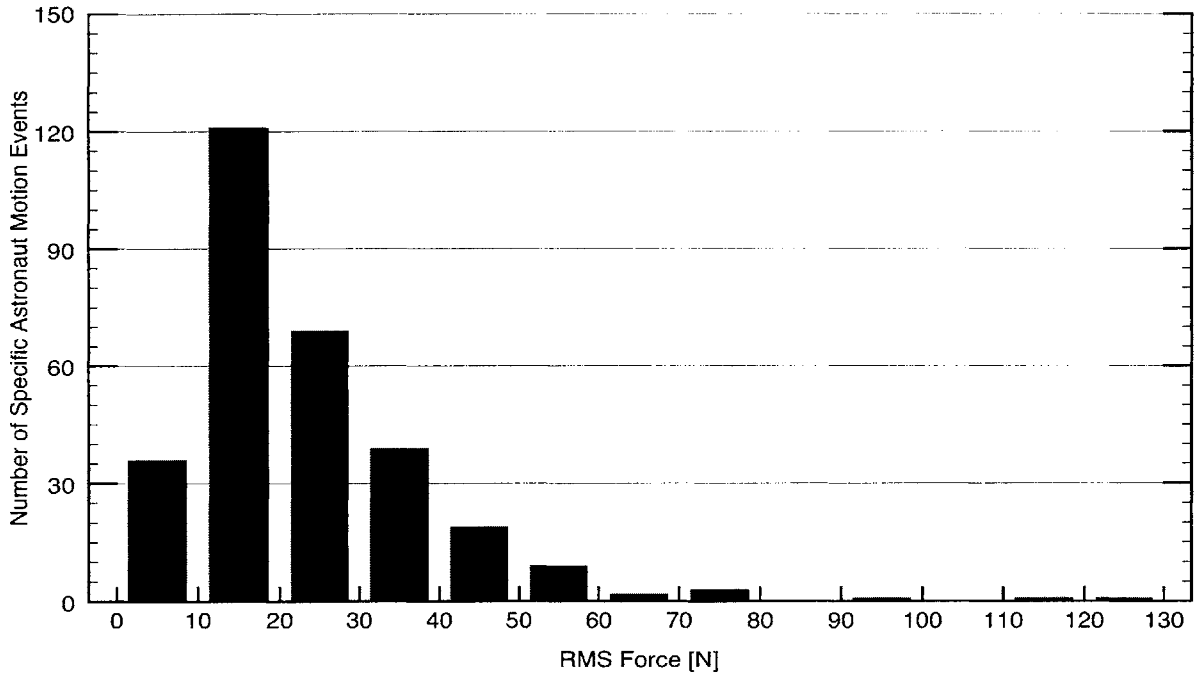


Figure 2.5: Root-mean-square forces recorded in the DLS Experiment on the Space Shuttle [Amir, 1999 B].

The Enhanced Dynamic Load Sensors (EDLS) experiment was proposed as a follow-on to the DLS experiment to expand the Shuttle database by collecting astronaut reaction forces and moments on the Russian *Mir* space station during long-duration space missions [Newman, 1994]. It was conducted as part of Phase I of the ISS Program involving U.S. experiments and astronauts on *Mir.* The ultimate goal was to define how to obtain a quiescent microgravity environment on-board the ISS, at least during the recurring 30-day quiescent periods (or "microgravity mode") in which disturbances to the acceleratory environment must be minimized or eliminated. The EDLS experiment and results are detailed below in section 2.3.2.

As the ISS becomes operational, there is a need for advanced load sensors to further quantify astronaut IVA and to help astronauts optimize their motions in the microgravity environment. The design and development of an on-ground prototype system is the main purpose of the MICR0-G (Microgravity Investigation and Crew Reactions in 0-G) research effort [Newman, 1997 B]. The MICR0-G project is comprehensively encompassed in Chapter 3.

2.3.2 The EDLS Experiment

Motivation and Objectives

The limited volume of the Space Shuttle middeck coupled with the constant activity of numerous crew members does not reflect operations inside space stations. In addition, Shuttle missions normally last about two weeks and as a result there is insufficient time to observe long-term astronaut adaptation to weightlessness and the learning effects that might occur on the ISS. Therefore, the EDLS experiment was proposed in May 1994 to expand upon the database acquired with DLS by collecting astronaut IVA forces and moments during long-duration missions on the Russian *Mir* space station.

This motivation led to the establishment of two objectives for the EDLS experiment:

- The primary objective was "to assess nominal crew-induced reactions for long-duration space station missions" with the ultimate goal "to provide a human factor assessment of crew reactions for engineering design requirements" [Newman, 1994].
- The secondary objective of the research effort was "to provide a detailed model that identifies and characterizes the adaptive control strategies adopted by crew members in microgravity" [Newman, 1994].

Based on the primary objectives, three specific goals were set for EDLS [Amir, 1998]:

1. Quantify the nominal crew-induced forces and moments during an extended stay in orbit.
2. Quantify changes in the forces and moments over time as the crew adapts to microgravity.
3. Characterize typical astronaut motions in microgravity.

The Experiment

The EDLS experiment measured daily astronaut IVA activities and prescribed motions [Amir, 2000]. In a daily IVA session, the astronauts went about their scientific or operational activities and used the sensors as restraint devices to help them in stationary tasks translation and moving about Mir. Figure 2.6 shows U.S. astronaut Jerry Linenger using the EDLS handhold to guide himself in the *Priroda* module of Mir. In prescribed motion sessions, the astronaut subjects used one or more foot restraint sensors while throwing a small ball of approximately 2 cm in diameter at a target 1.5 m–2 m away for repeated trials with their eyes either open or closed. The purpose of the throwing activity was to quantify the adaptation of human motor control in microgravity. Figure 2.7 shows a computer-rendered image of how the active EDLS sessions were performed.

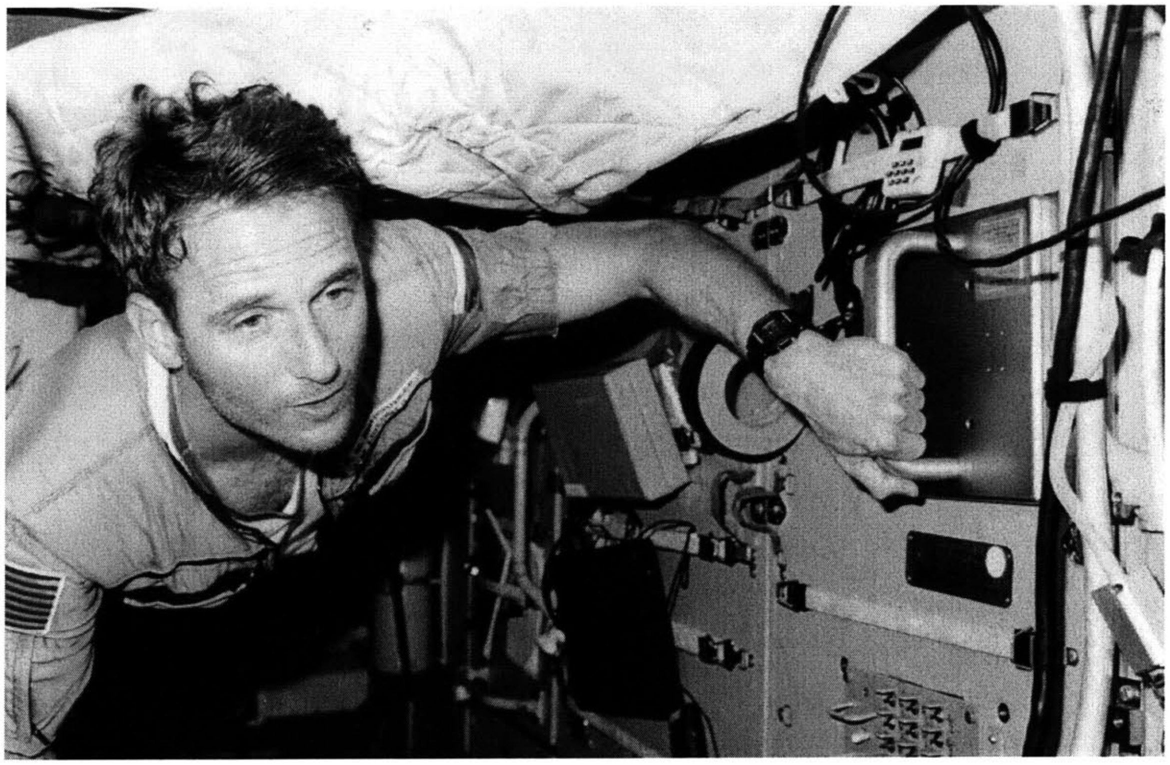


Figure 2.6: U.S. astronaut Jerry Linenger (NASA 4 mission) using the EDLS handhold to guide himself in the *Priroda* module of Mir. [NASA Image NM23-01-013].

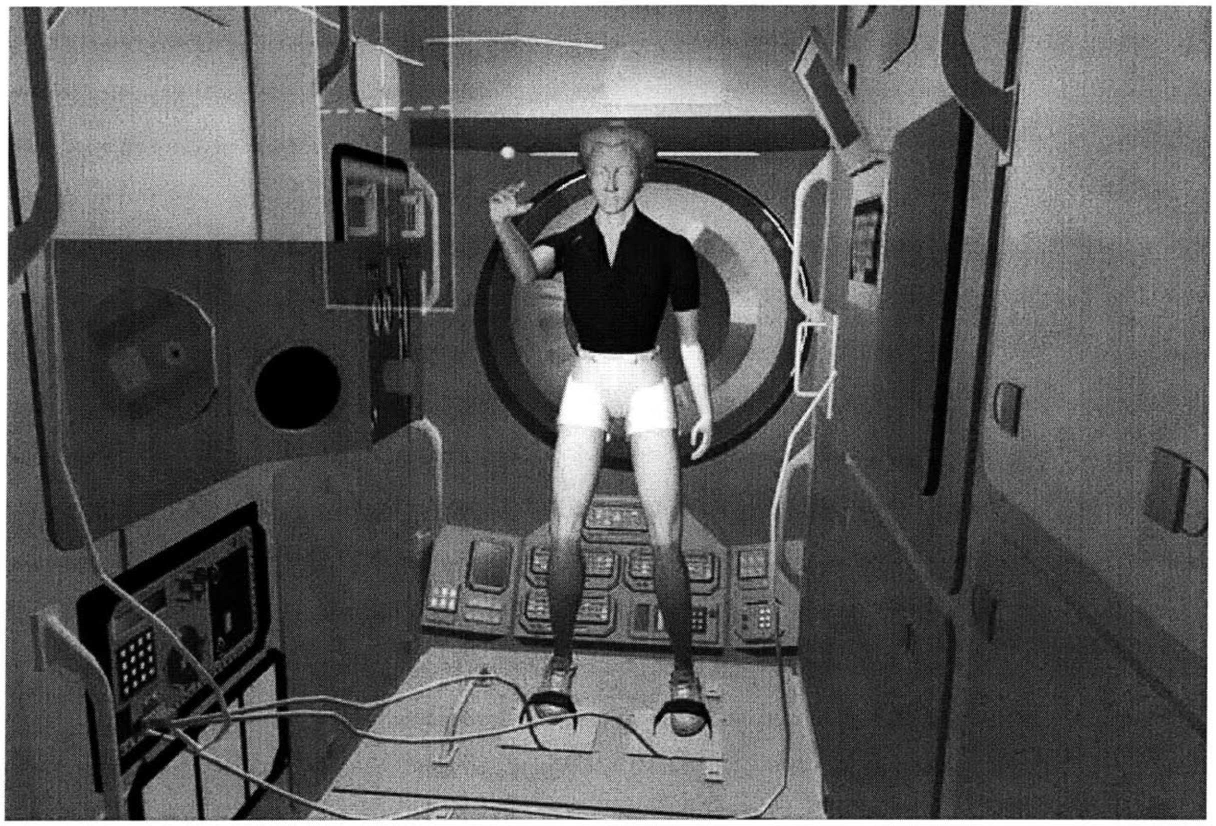


Figure 2.7: This scene from a computer-rendered animation made by Designer's CADD shows the set-up used to conduct the throwing experiment EDLS sessions [Amir, 1998].

The first EDLS data was taken on May 23, 1996. By August 1996, Shannon Lucid and her Russian colleagues Commander Yuri Onufrienko and Flight Engineer Yuri Usachyov performed 34 EDLS sessions of which two were active. Due to a failure of the data acquisition computer in August 1998, no data was acquired during the NASA 3 Mir mission. A new computer was brought to Mir in time for the NASA 4 Mir mission. From February to May 1997, twelve daily IVA activity sessions and six active sessions were performed by Jerry Linenger and Russian cosmonauts Vasili Tsibliev and Alexander Lazutkin [Amir, 1999 B].

The EDLS experiment used the same basic design for load sensors as the DLS experiment on board the Shuttle. A set of four sensors consisting of an instrumented handhold, a touchpad, and two foot restraints were used for EDLS. The handhold and the foot restraints were specifically designed to provide the same functionality as the foot loops and hand rails built into the *Mir* orbital complex for astronaut use. The touchpad's functionality was envisioned to be that of a flat surface the astronauts would use to push themselves off using either their hands or feet.

Each sensor was connected to the Experiment Support Module (ESM) housing all the electronics necessary for the experiment (i.e. the signal conditioning system, the experiment computer, the storage devices and the power supply). The module used during NASA 2 mission was called the MODE (Mid-deck 0-Gravity Dynamics Experiment) ESM and an advanced version of this computer, called MASU (Mir Auxiliary Sensor Unit) ESM, was used for NASA 4 mission [Bokhour, 1992].

The key features of the MASU ESM are provided below:

- 33 MHz Ziatech processor
- 32 channels of signal conditioning
- 4^{th}-order Bessel low-pass filter
- 16-bit A/D conversion
- 250 Hz sampling frequency
- 260 MB Type III PCMCIA cards for data storage
- Operator interface through a 25 button keypad and a 4 x 20 character supertwist backlit LCD

The ESM modules processed and stored data from all the sensors. The PCMCIA storage media were brought back to Earth in 1997 and the collected data were analyzed at MIT.

The Results

Overall 4,480 events were identified with the breakdown by mission and type provided in Table 2.4. Whenever the EDLS sensors detected loads exceeding a specified minimum threshold level, data were recorded on the storage medium. One such recording is known as an *event*, which is understood as either a specific astronaut motion lasting from a few seconds to several minutes or several motions together without an interruption in the loading to a sensor. If a distinction of individual motions in the latter case is needed, they are referred to as *subevents*. The characteristic motions observed during EDLS were very similar to those seen during DLS, which have been described in the previous section.

Table 2.4: Number of astronaut motions analyzed for EDLS [Amir, 2000].

Type of Sensor	NASA 2 Mission	NASA 4 Mission	Total
Foot restraint Sensor 1	1,817	326	2,143
Foot restraint Sensor 2	362	0	362
Handhold Sensor	230	71	301
Touchpad Sensor	1,089	585	1,674
Total	**3,498**	**982**	**4,480**

Figure 2.8 shows a histogram of the maximum force magnitudes recorded by the two foot restraint sensors and the handhold sensor during the EDLS experiment. For these 2,806 events, 96% of the time the maximum force magnitude was below 60 N and about 99% of the time the maximum force magnitude was below 90 N. For each force magnitude interval, the frequency of occurrence is provided. For example, about 21.7% of all events have a maximum force magnitude between 10 N and 20 N. The highest force magnitude measured by the two foot restraint sensors was 137 N and for the handhold sensor 124 N. Assuming a total mass of 400 metric tons for the ISS at complete assembly, an applied force of 137 N would result in an acceleration of about 3.5×10^{-5} g. The average force magnitude recorded was 21 N with a 18 N standard deviation and the median force magnitude was 16 N.

The significance of the astronaut loads is more evident from the root-mean-square (rms) forces recorded for the previous 2,806 events. The histogram of the root mean square forces recorded by the two foot restraint sensors and the handhold sensor is shown in Figure 2.9. The largest rms force was 35.6 N, the average value was 2.4 N and the median value 1.3 N. For 96% of the astronaut motions, the rms force value was below 9.0 N.

A histogram of the maximum moment magnitudes is shown in Figure 2.10. The largest moment was measured by a foot restraint with a value of 18.3 N·m. The average moment was 1.9 N·m. The distribution of the moment magnitude for the 2,806 events was such that 96% of all events had a maximum moment magnitude below 7 N·m.

An analysis of the power spectral density distribution in the events showed that on average 95% of the power was contained below a frequency of 2.6 Hz. The histogram of the frequency below which 95% percent of the power of an event is contained is shown on Figure 2.11 [Amir, 2000].

Conclusion

The EDLS experiment produced the first comprehensive description of IVA astronaut motions and quantified astronaut-induced loads in microgravity for long-duration space flights. As the astronauts adapt to the microgravity environment in orbit during the first several weeks in space, it is expected that their motions decrease in velocity and magnitude. Due to the high operational demand on the crew during the spacecraft hand-over time and shortly thereafter, the astronauts recorded EDLS sessions after the first three weeks in space. Overall, the EDLS experiment did not observe any significant change in the average force level observed.

The development of enhanced sensors for the ISS combined with a system based on measuring kinematics data via infrared video cameras, will describe and quantify the adaptation of the astronauts to the microgravity environment. In addition, the magnitude of astronaut-induced loads being difficult to infer from visual observation, active sensing with *real-time feedback* of the applied force and torque levels is a valuable tool for the astronauts to minimize the disturbances they induce, especially during microgravity operations.

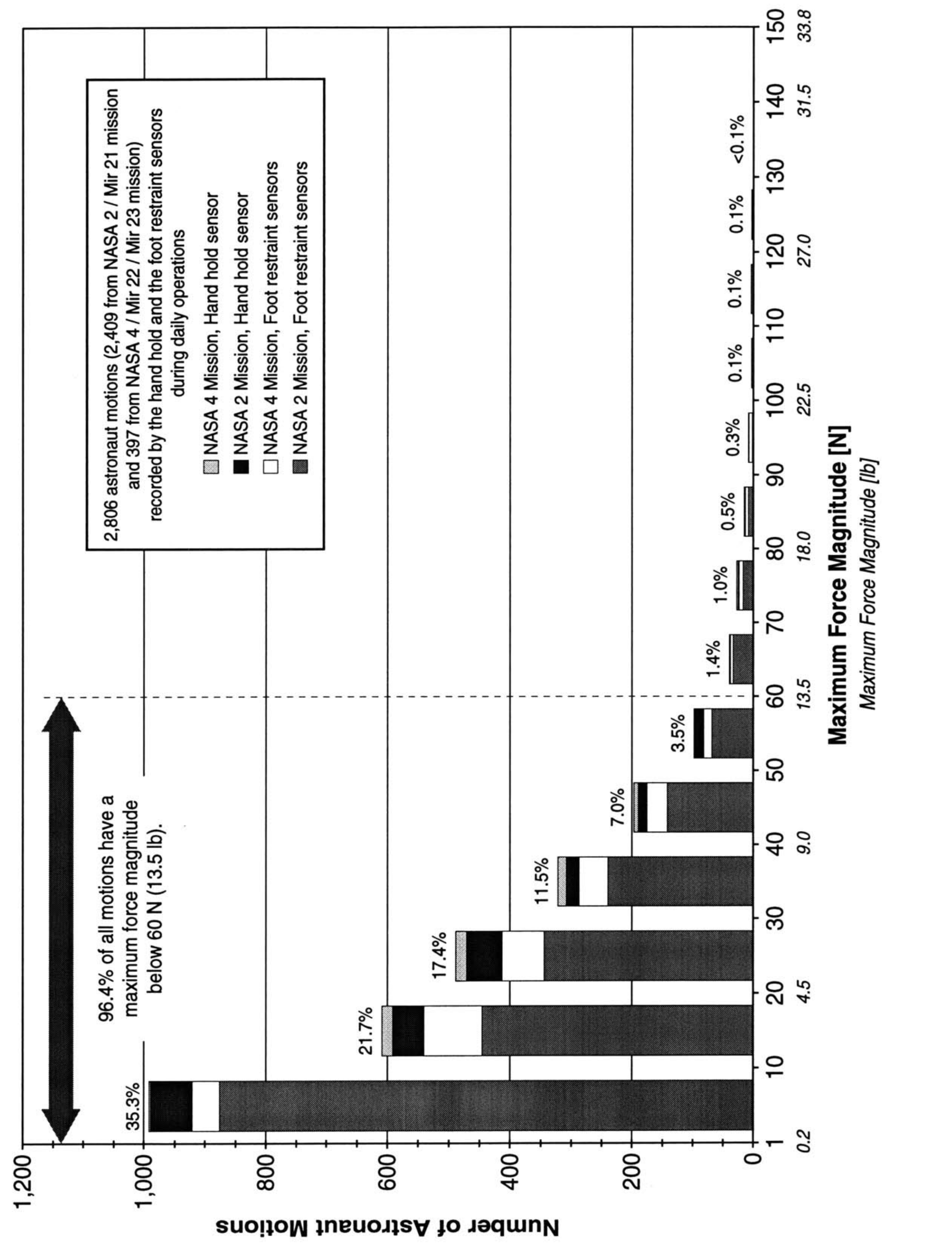


Figure 2.8: Histogram of the maximum force magnitudes recorded by the two foot restraint sensors and the handhold sensor [Amir, 2000].

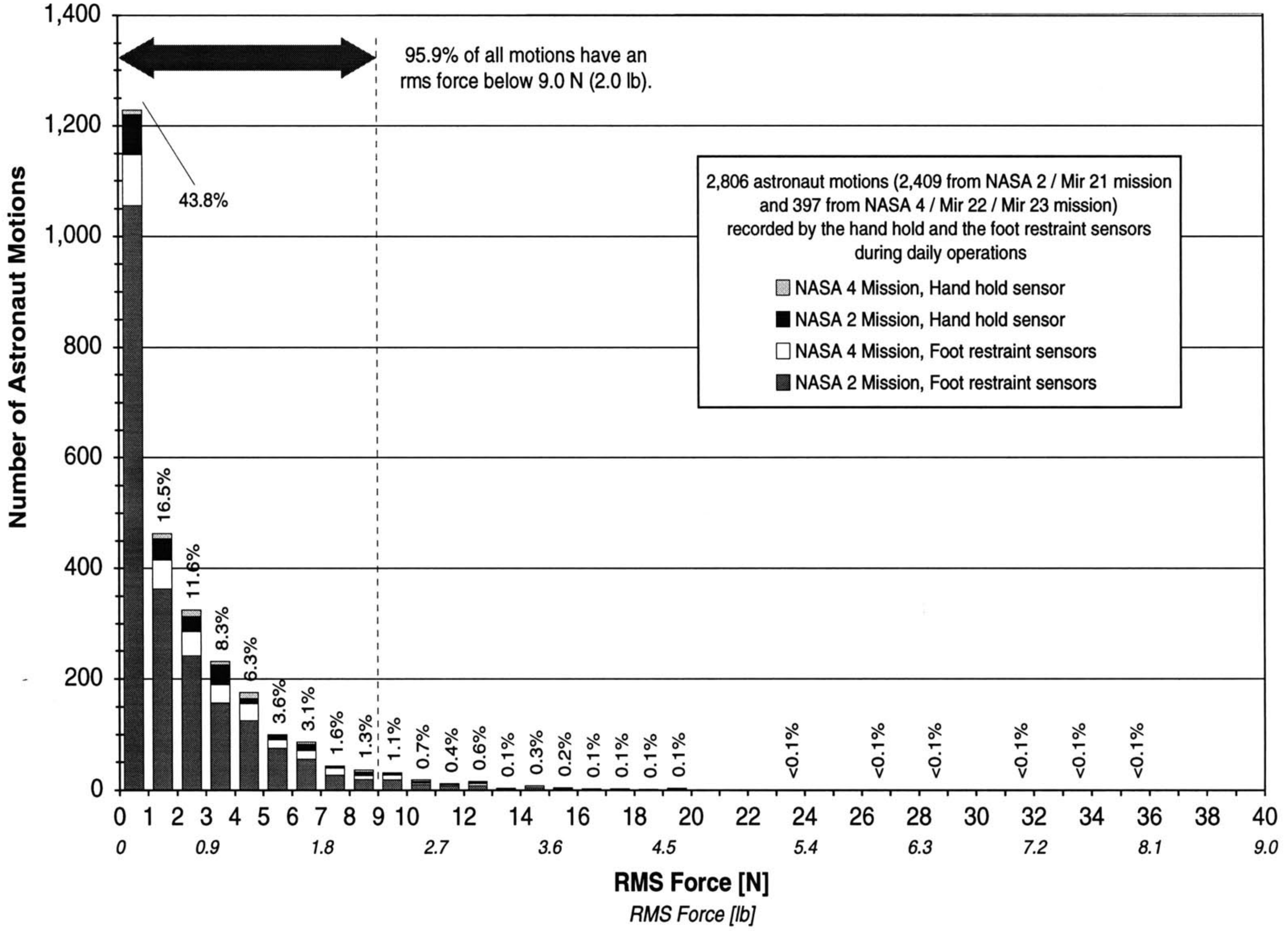


Figure 2.9: Histogram of the root mean square forces recorded by the two foot restraint sensors and the hand-hold sensor [Amir, 2000].

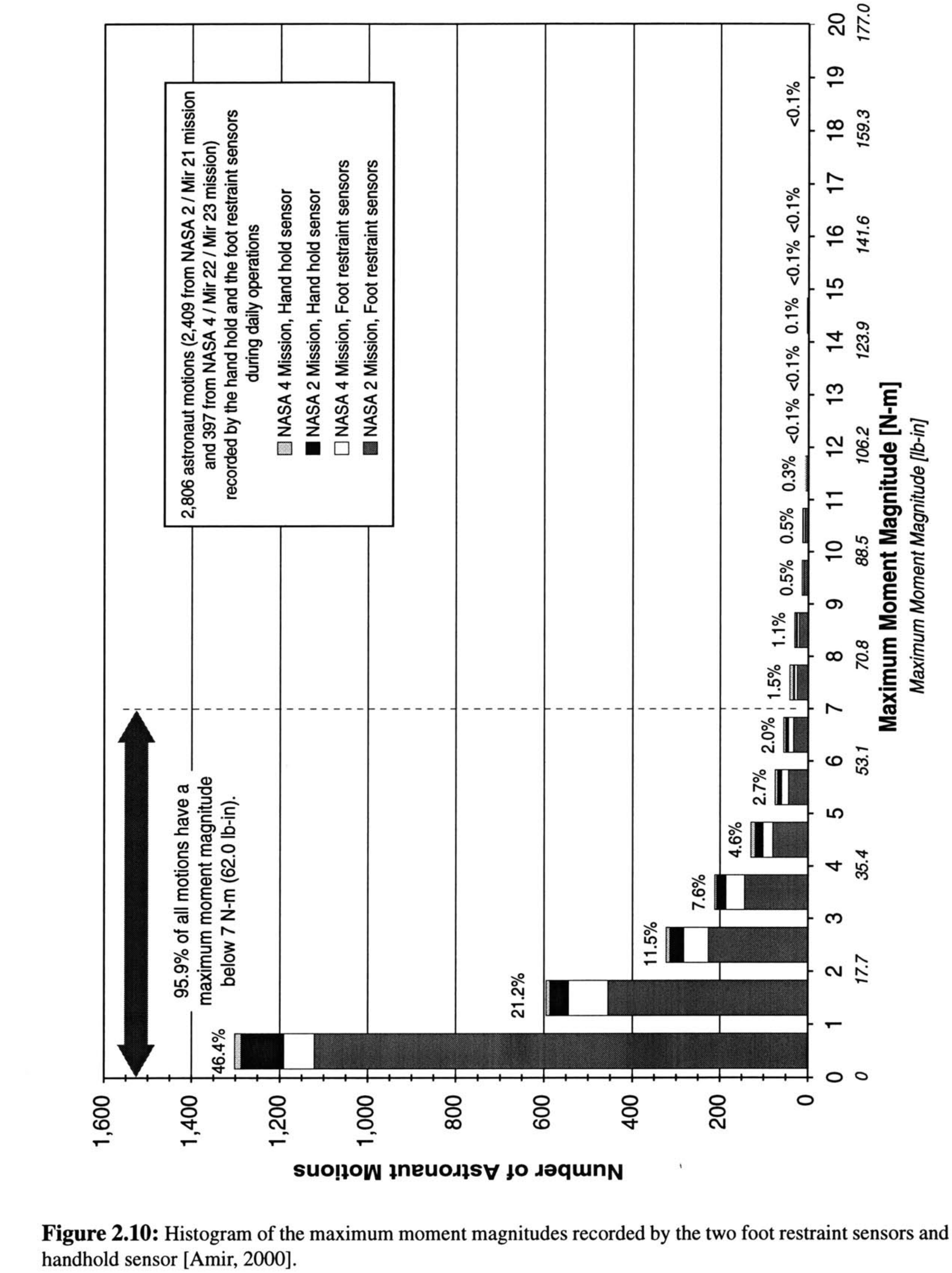


Figure 2.10: Histogram of the maximum moment magnitudes recorded by the two foot restraint sensors and the handhold sensor [Amir, 2000].

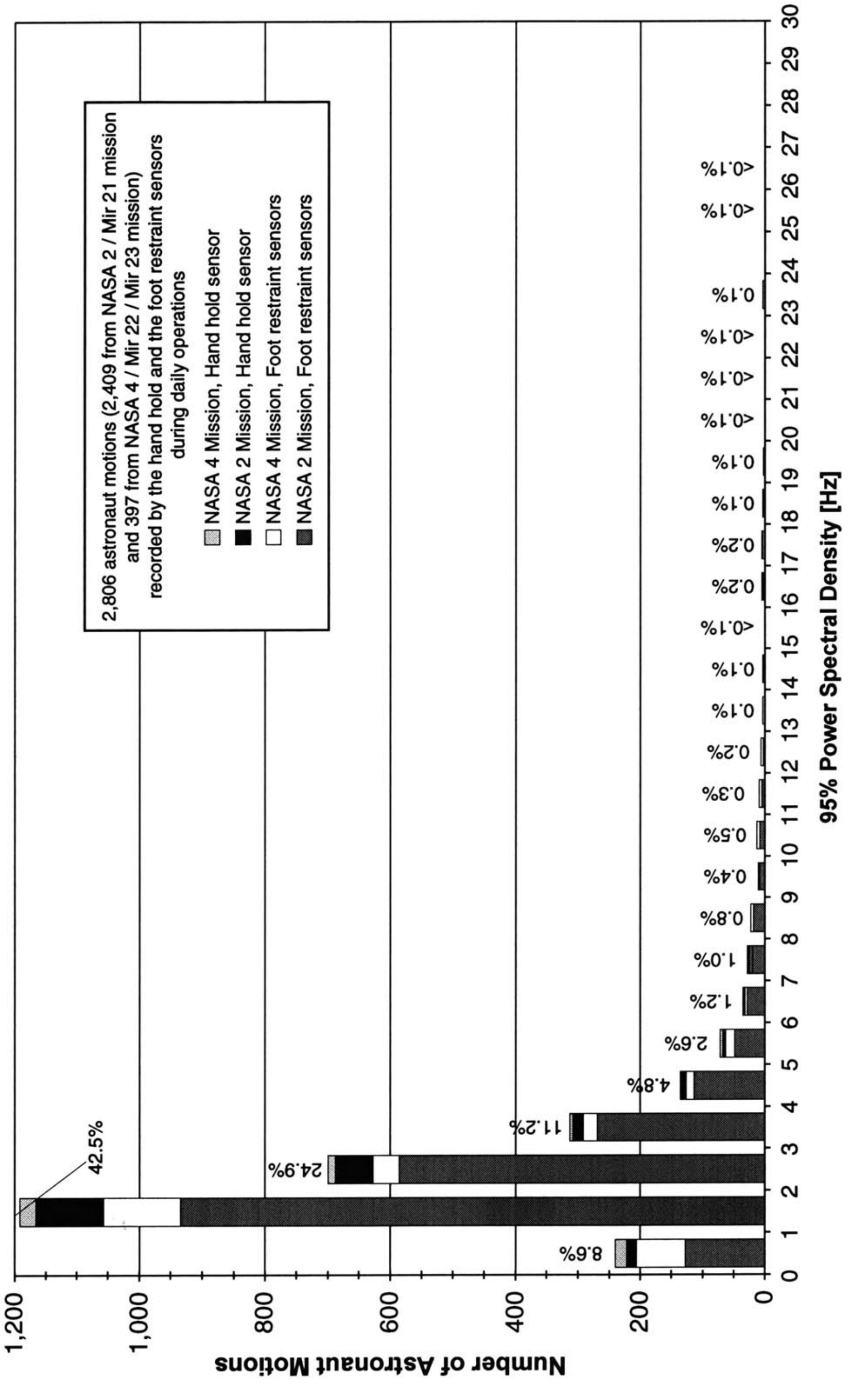


Figure 2.11: Histogram of the frequency below which 95% of the Power Spectrum Density is contained for the events recorded by the foot restraints and the handhold [Amir, 2000].

CHAPTER

3

DESIGN OF THE MICR0-G ADVANCED LOAD SENSORS

The previous chapter reviewed the motivation for investigating astronaut-induced disturbances aboard spacecraft and gave a brief overview of the past experiments and results related to quantifying astronaut intravehicular activity (IVA). As the ISS becomes operational, there is a need for advanced load sensors to further quantify astronaut IVA and motions to help optimize their tasks in the microgravity environment. The design and development of a ground-based advanced load sensor prototype system is the main goal of the MICR0-G (Microgravity Investigation and Crew Reactions in 0-G) research effort. This chapter introduces the MICR0-G project, addresses the research motivation and objectives, and list the technical requirements for the design of the advanced load sensors. The two following sections detail the two most important features that led to a vastly improved design for the advanced load sensor system: the wireless capability and the use of the PC/104 form factor. Section 3.4 presents the design and related hardware of the system. Finally, the last two sections give an overview of the software development and the calibration of the ground-based prototype.

3.1 The MICR0-G Project

The MICR0-G research effort is carried out by MIT, with collaboration from NASA, Politecnico di Milano University, the Italian Space Agency (ASI) and contributions from the French Space Agency (CNES). The objective is to develop an integrated ground-based technology demonstrator to assess crew motor behavior and nominal crew-induced reactions in weightlessness. The kinetic data (force and torque) will be measured by an advanced version of the Dynamic Load Sensors (DLS) flown on the Space Shuttle and on Mir [Newman, 1997 A]. The astronaut motions will be captured by the ELITE-S2 system, an enhanced version of the real-time opto-electronic motion analyzers ELITE-S and Kinelite, flown respectively on the EuroMir '95 Mission and on the Shuttle Neurolab mission [Ferrigno, 1985].

The primary MICR0-G project aim is to design and build a ground-based prototype for the new generation of load sensors with miniaturized electronics architecture, wireless capability and real-time feedback. Once complete, the prototype system will be proposed to NASA as a space-qualified experiment to be manifest on the ISS.

3.1.1 MICR0-G Objectives

The importance of a quiescent environment for microgravity research on the ISS and the resulting motivation of investigating astronaut-induced disturbances were presented in Chapter 2. The main objectives of the MICR0-G project are:

1. to characterize crew-induced reactions and nominal human motor strategies in weightlessness.
2. to assess astronaut adaptation to microgravity during long-duration space missions.
3. to provide real-time feedback to the crew on their motor behavior and disturbances to the microgravity environment.

These main objectives lead to four specific goals:

1. to provide a human factor assessment of astronaut reactions for future engineering design requirements of orbital modules.
2. to develop a dynamic motor control model to examine motor control strategies for long duration space flight.
3. to improve ground-based training for astronauts.
4. to help astronauts maintain a quiescent microgravity environment aboard the ISS.

3.1.2 Lessons Learned from Past Experiments

This section summarizes the problems encountered with the first-generation of load sensors that flew on the Space Shuttle for the DLS experiment and on Mir for the EDLS experiment. Examination of video tapes showing astronauts using the sensors and the spaceflight hardware as well as interviews with the astronauts provided valuable lessons for the redesign of the system. The following observations reflect space experiences and the portions set in italics identify the guidance for the development of advanced load sensors:

- Aboard the Space Shuttle, three DLS sensors were connected to the ESM (Experiment Support Module) computer via umbilical cables. Video tape recordings and direct feedback from the crew showed that all the aggregate cables floating around hinder astronauts in their motion and clutter space modules. *Optimization of the equipment size and the use of wireless capabilities should guide the design requirements for the next generation of sensors.*

- The EDLS system aboard Mir performed an auto-zeroing before beginning data acquisition so that, when no load is applied, the signal was zero. Each sensor being unique, the auto-zeroing of one sensor does not apply to another. In addition, power was provided to the system continuously during data acquisition. For these two reasons, hot-swapping sensors was not allowed (i.e., exchanging two sensors while power is on). However, this did occur during flight numerous times affecting the signal quality and making data analysis difficult [Amir, 1998]. *An enhanced architecture, or the use of a mechanical and/or electronic mechanism should be used to prevent the detrimental effects of hot-swapping sensors.*

- The ESM computer provided the main interface to the operator through a 16-button keypad and a 2 x 16 character alphanumeric display. When the system was booted up, each hardware component was tested and the status reported on the screen ("passed" or "failed"). Since there were many components and a limited text display on the screen, it made it difficult for the operator to check the possible failures [Amir, 1998]. *The new generation of sensors should provide a convenient and familiar graphical interface, and detailed information should be given only when necessary.*

- Because of the high workload that the astronauts face in orbit and the significant number of experiments they have to conduct, short set-up and operational procedures would significantly help them and reduce the probability of errors during the experiment. *Experimental protocols should be simplified and shorten to the possible extent.*

- The load sensors should provide all the functionality of restraint mobility aids to help astronauts in their daily activities. Real-time feedback is of value to the astronauts in optimizing their loading profiles and in determining whether they need to adjust their motions. *Any functionality that could be added to the experiment and that could help the astronauts in their daily activities should be investigated.*

3.1.3 Requirements for the Advanced Load Sensors

The MICR0-G research effort will develop advanced load sensors capable of precisely measuring astronaut loading profiles as they work and move in the microgravity environment during extended space missions. The load sensors will record and store force and moment data as well as provide real-time feedback to the astronauts on the magnitude of the reaction forces they cause [Newman, 1997 B].

The system requirements for the advanced load sensors are stated as follows:

- The system shall acquire and process force and moment data in real-time and provide immediate visual and/or audio feedback to the astronauts.
- The system interface shall be simple enough so that the crew-time required for setting-up the system is no more than 3 minutes.
- The mobility and flexibility of the system shall be maximized by making use of the miniaturization of electronics and the latest wireless capabilities.
- As many commercial off-the-shelf (COTS) technologies as possible shall be used to reduce development and production costs.
- The system shall be able to communicate and to be synchronized with the opto-electronic motion analyzer ELITE-S2.
- On-orbit maintenance and reparability shall be possible.

The technical requirements for the re-design of the load sensor system are stated as follows:

- Each sensor shall record force and moment components along the x-, y-, and z- axis.
- The system shall be fully compatible with the ISS (power, restraint aids, data bus, etc.).
- The system should acquire data only if a load is applied on the sensors.
- The storage capacity shall be increased up to 1GB.
- The signal conditioning shall include a low-pass filter and an adjustable amplifier.
- The sampling frequency shall be no less than 250Hz.
- The minimum force and moment resolutions shall be 2 N and 0.2 N·m, respectively.
- The full scale load shall be 400 N for forces and 50 N·m for moments.

This set of requirements led the development of the advanced load sensors. The two following sections detail the main innovative features in the design of the ground-based prototype: the wireless capability and the use of the PC/104 compact form factor. In particular, the suitability of these technologies for space applications is discussed.

3.2 Wireless Communications On-Board Space Stations

3.2.1 A Need for Wireless Networks On-Board Space Stations

Aboard space stations, astronauts are surrounded by a vast amount of cable. The whole set of cables is incredibly cumbersome (see Figure 3.1) and makes crew operations as well as housekeepping very difficult. Hooking up endless wires between devices is messy, frustrating and, above all, inefficient in astronauts' daily activities.

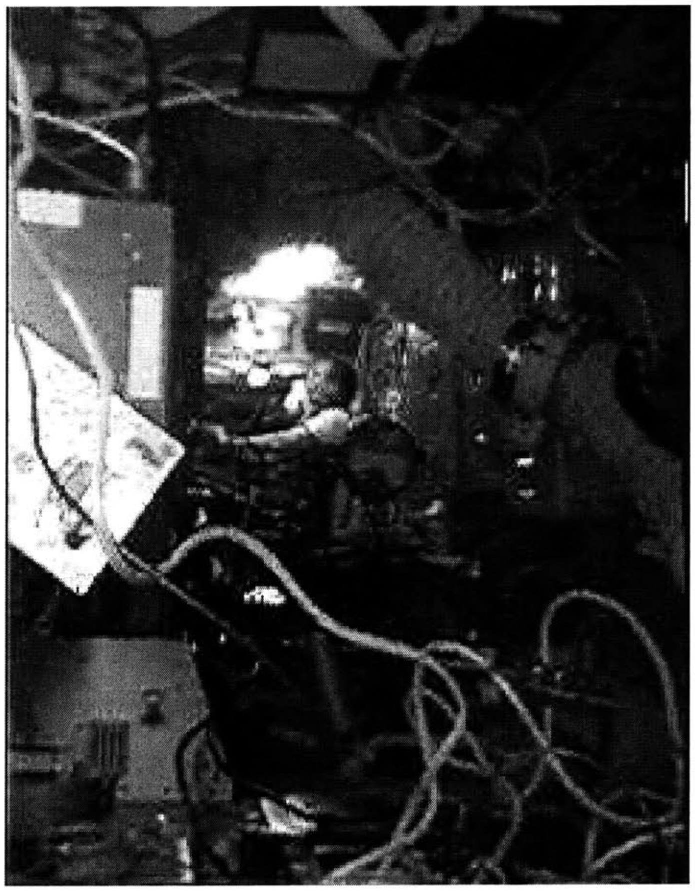


Figure 3.1: The image shows the cumbersome interior of the Mir Space Station where the astronauts were working. (NASA Mir WB8ERJ, 6-27-1999)

Wireless communications eliminate the need for cables. Since mass and volume are two severe constraints for space applications, the lack of cables greatly enhances payloads design characteristics. In addition, crew members improve their freedom to move around and gain flexibility using wireless connectivity on remote computer devices. They can be anywhere on the space station and optimally perform operations. However NASA has lagged in endorsing the use of wireless devices on spacecrafts primarily due to the critical issues of radiation, electromagnetic interferences (EMI) and electromagnetic compatibility (EMC).

3.2.2 The Mir Wireless Network Experiment (MWNE)

Mir: A Test Bed for Building the ISS

To prepare for the ISS, NASA undertook a joint technology program with the Russian Government to obtain operational experience gained on their Mir Space Station. This US/Russian "Risk Mitigation" joint program aimed to reduce the technological risks for developing, deploying, and operating the ISS, as discussed in Chapter 2. The Mir Wireless Network Experiment (MWNE) was part of this program as a "Risk Mitigation Technology Demonstration" [Kiser, 1995]. The Mir WNE was launched by the Space Shuttle flight STS-74 in November 1995. It was successfully conducted aboard the Space Shuttle Atlantis and the Mir Space Station on March 1996 during STS-76.

MWNE Hardware

The MWNE consists of three computers. The prime component is the Wireless Network Server (WNS), a ruggedized Pentium-based portable computer, with 32 MB of error-correcting memory, a 90 MHz Pentium processor, a 1024x768 Super VGA display, and NTSC video. The other two components are an HP Omnibook Subnotebook (SNB) computer and a NORAND Pen*Key Personal Digital Assistant (PDA). PDAs are hand-held computer devices that may be used to enter, manipulate, and/or display data. All three units have an RF network adapter for wireless communications between the WNS and the mobile computing nodes (i.e., the PDA and the SNB) [Hubbard, 1995].

The location of the computers were changed periodically, the performance evaluated, and any constraint on using the computers in different locations were noted. The network monitor program measured system performance and recorded test data. The PDA display prompted the crew for location changes and displayed basic data. Test results were recorded on the MWNE hard disk [Gawdiak,1998].

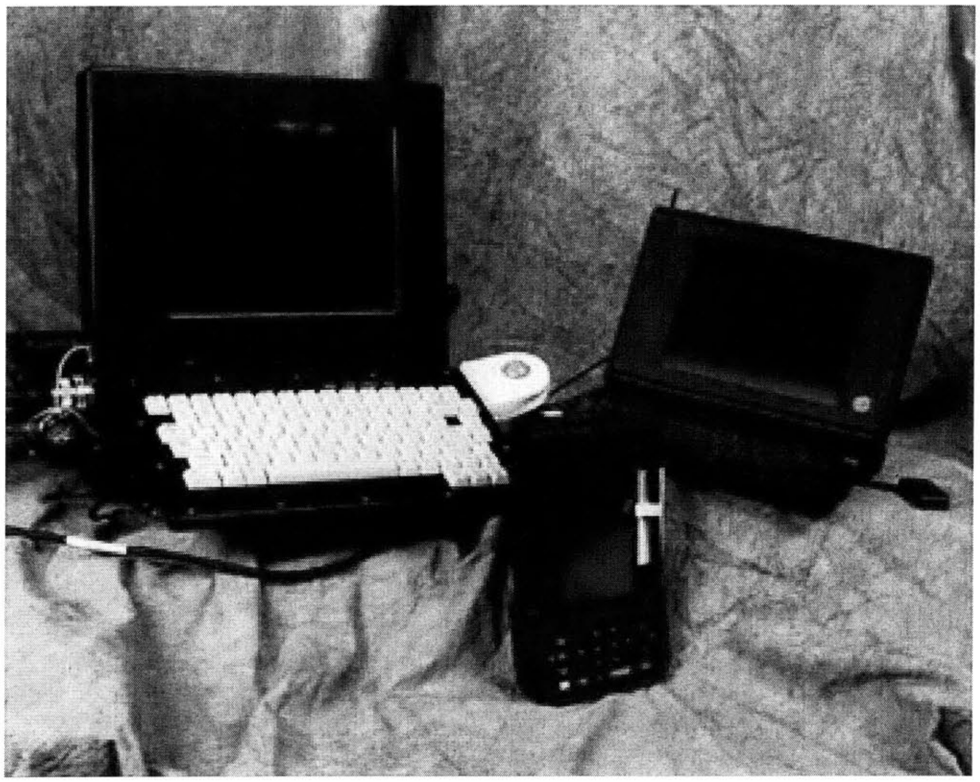


Figure 3.2: WNE (computers) components, left to right: Wireless Network Server (WNS); NORAND PenKey Personal Digital Assistant (PDA); and HP Omnibook Subnotebook computer [Alena, 1995].

MWNE Objectives

The MWNE was the first test of a wireless client-server network in the space environment. This technological evaluation tested the wireless network system performance on Mir as a possible network for the operation on the International Space Station. It has been designed to study and evaluate electromagnetic compatibility (EMC), computer devices performance, wireless network performance, and human/computer interaction factors in microgravity [Sander, 1995].

The MWNE investigated the use of spread-spectrum Radio Frequency technology in the 2.400 to 2.483 GHz frequency range. This frequency band is the standard for wireless digital Local Area Networks (LANs). Wireless LAN performance metrics were characterized by range, distribution of coverage, and maximum sustainable data rates. The variation of data throughput and bit error rates were measured as the mobile computing devices were moved within the module containing the wireless server, and as they were moved into other (remote) modules on the Mir Space Station.

MWNE Results

The results of this experiment were helpful in designing the wireless network aboard the ISS. In particular, NASA decided to keep the same protocols, frequency band and Spread Spectrum technology as in the Mir Experiment to develop the ISS wireless LAN.

The space qualification process, the empirical data, and operational experience obtained during the Mir WNE were useful in developing guidelines for deploying commercial (mobile/wireless) computing devices for space applications. The empirical data were invaluable in evaluating electronic devices' ability to operate in the space environment and helped quantify their tolerance/susceptibility to single event upsets [Kiser, 1995]. Single event upsets occur when cosmic particles strike electronic devices – such as a Random Access Memory (RAM) – and cause them to fail or change state unpredictable. Finally human/computer interaction gave an evaluation on portable computer display visibility, input accuracy, response times, and flexibility for various applications.

Through the MWNE, NASA gained experience in designing wireless applications and space qualifying specific electronics devices. The Mir Wireless Experiment results served as a basis for space engineers to develop the ISS network based on both wired and wireless communications.

3.2.3 The ISS Network

The ISS LAN uses a multipoint Ethernet bus topology for network communications. It implements both the IEEE 802.3 (Standard Ethernet) and 802.11 (Radio Frequency) connectivity standards. The File Server is an IBM ThinkPad 760XD laptop computer. The Station Support Computers (SSC) – IBM 760XD laptops – will connect to the File Server and the LAN using Radio Frequency via wireless network PC-Cards. The LAN utilizes NetBEUI (NetBIOS Extended User Interface) and TCP/IP as network communications protocols [NASA, 2000 C].

The cabled connectivity relies on RG-58 coax cabling, BNC connectors and terminators. Coaxial cables have several advantages, including high resistance to EMI, a history of reliable service, and durability [NASA, 1998 B].

Wireless communications aboard the ISS are based on Radio Frequency connectivity, which operates in the 2.4 GHz frequency band (2.4-2.484 GHz), and behave in the same manner as a standard LAN cable bus topology. Advertised throughput via RF PC-cards is 12 MB/min. Actual data throughput is predicted to be between 3.5 and 5.5 MB/min. To immunize against RF interferences and unauthorized eavesdropping, the Spread Spectrum technology (also knows as "Frequency Hopping") is employed [NASA, 1999 A]. In frequency hopping the radio signal "hops" from frequency to frequency within a specified band (2.4000-2.4825 GHz) over a set time. Both the transmitter and receiver know the hopping pattern which is called a "channel".

RF connectivity depends on Network Cards and RF Access Points for wireless transfer of network packets. The ISS LAN uses RangeLAN2 7400 PC-cards from Proxim (Sunnyvale, CA) for RF communication between laptops and RF access points. The Proxim RangeLAN2 products provide 15 frequency hopping sequences that are orthogonal patterns. Each RF access point is a 7520 RangeLAN2/AP-II MAC layer bridge that connects to the coaxbone. The dipole antenna relays RF packet communication between the backbone and RF-equipped laptops [NASA, 2000 C].

The network designed for the sensors utilizes the same protocols and some of the hardware of the ISS network. As a result, the advanced load sensors can be fully integrated into the ISS global network, or can be an independent unit able to talk to the ISS LAN when necessary.

3.3 Miniaturized Electronics: the PC/104 Solution

3.3.1 A Standard Needed for Embedded Processor Systems

Since the birth of the microprocessor in the mid-1970s, CPU performance and memory capacity have been growing exponentially. Currently, embedded computer CPUs are clocked at up to 400 MHz, RAM capacities surpass 128 MB, and mass storage is commonly measured in GB. As embedded computer speeds and memories grow ever larger, embedded applications become increasingly decoupled from the underlying embedded computer architecture. An obvious way to increase the efficiency of embedded system development is to employ standardized – even off-the-shelf – hardware and software, if available. This minimizes the need to design from scratch.

With this concept in mind, embedded system designers have looked at the highly popular PC architecture to provide a standardized hardware and software toolkit. The enormous popularity of the PC architecture has generated vast software and hardware desktop resources. But can the PC desktop architecture be successfully adapted to embedded systems?

In the PC market, price pressures severely constrain such factors as reliability, ruggedness, quality, and product longevity. Embedded computers, on the other hand, must satisfy a whole different set of objectives – most of which run counter to the priorities of the desktop market [Lehrbaum, 1997]. These include:

- minimized size and weight (so they can fit within space-limited products)
- reduced power consumption (to satisfy limited power budgets or even run off batteries)
- resistance to shock and vibration (to survive harsh embedded system environments)
- extended operating temperature (for operation in non-office applications)
- enhanced functional reliability (to eliminate the likelihood of system failures)

Furthermore, unlike desktop PCs, embedded computers are not general-purpose, user-programmable systems, but have highly specialized requirements. So they must be easily adaptable to application-specific requirements, which often require a custom electronics interface.

3.3.2 The PC/104 Solution

In the late 1980's, Ampro Computers (San Jose, CA) developed the PC/104 form factor defining how to repackage desktop PC functions in a manner that satisfies the ruggedness, reliability, and size constraints of embedded systems. The PC/104 standard offers a full hardware and software PC-compatible architecture, but in the form of compact (9.0 cm x 9.6 cm), self-stacking modules (see Figure 3.3) [Becker, 1998].

The PC/104 Consortium was established to maintain the specifications, publish a resource guide, participate in standards activities, and promote PC/104 at trade shows and through news releases. Almost any type of module you can think of is now available for the PC/104 bus including CPUs, video controllers, network interfaces, sound I/O, data acquisition boards, and specialized interfaces. More than 200 vendors world-wide supply hardware, software, and engineering services to support the growing PC/104 standard [Miller, 1997].

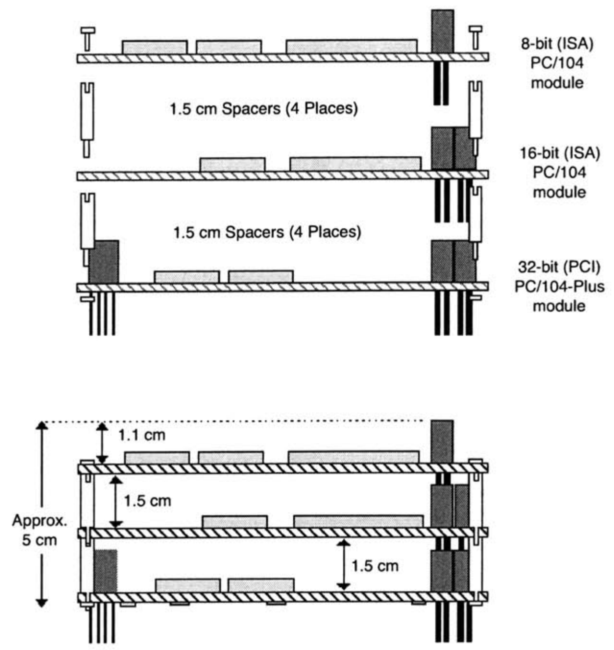

8-bit (ISA) PC/104 module
1.5 cm Spacers (4 Places)
16-bit (ISA) PC/104 module
1.5 cm Spacers (4 Places)
32-bit (PCI) PC/104-Plus module
1.1 cm
1.5 cm
Approx. 5 cm
1.5 cm

Figure 3.3: A typical PC/104 module stack [Becker, 1998].

Consequently, the decision to use a PC/104 architecture represents several advantages. Beyond obvious gains in economy of size, a wide array of choices in the configuration of the system becomes possible since the PC/104 standard is widely supported by board manufacturers. Another advantage revolves around the interchangeability of the components. For example, a PC/104 CPU could potentially be upgraded or replaced with one or two PC/104 modules that fit into the same space.

3.3.3 The PC/104 Form Factor

The PC/104 standard gets its name from the popular desktop personal computers initially designed by IBM called the PC, and the 104-pin headers used to connect the cards together. PC/104 cards are much smaller than ISA-bus cards found in PC's and can be stacked together which eliminates the need for a motherboard, backplane, and/or card cage. Because PC/104 systems are very similar to standard desktop personal computers, they can be programmed with the same development tools used with full-size PCs, therefore reducing the need and cost of custom development efforts.

Mechanical Specifications

The differences between the PC/104 and the "normal" PC standards are primarily mechanical. The PC/104 specifications are defined as a compact form factor of 9.0 cm by 9.6 cm, reduced bus drive (for lower power consumption and minimized components), and a self-stacking 104-signal pin-and-socket connector bus. As shown on Figure 3.4, the 104 pins are on the two bus connectors which carry signals between modules. P1 has 64 pins and P2 has 40 pins. Each connector is a dual-row header having 2.5 mm pin spacing.

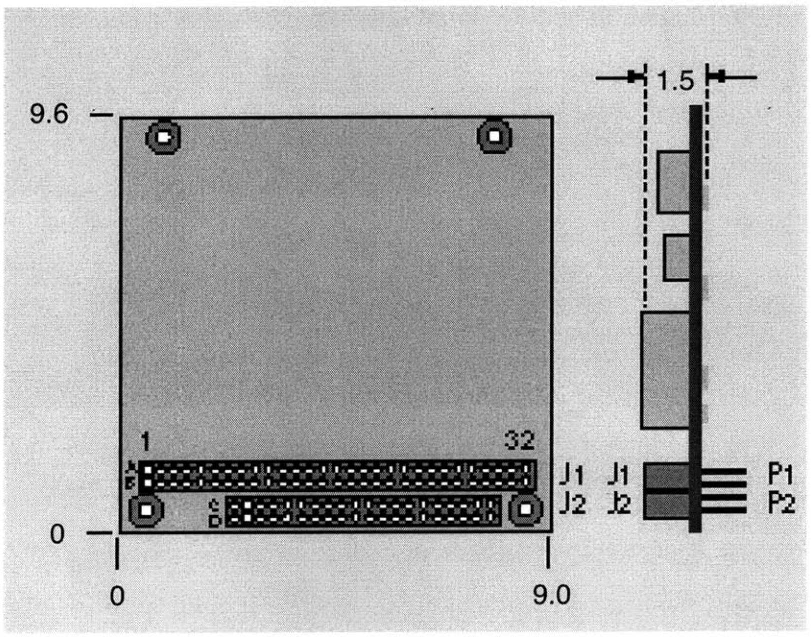


Figure 3.4: The mechanical dimensions (in cm) of a PC/104 module [Lehrbaum, 1997].

Electrical Specifications

PC/104 bus signals are identical to the standard PC bus in definition and function. Additional grounds enhance bus integrity and connector keys insure proper mating. The power requirement for driving the bus is only 4 mA (a normal PC requires 24 mA). This low power requirement reduces heat dissipation and power consumption to 1-2 Watts per module and decreases the amount of radiated emissions compared with the standard PC/AT bus [PC/104 FAQ].

Usually a PC/104 board requires a voltage source of +5 V and occasionally +/- 12 V as well. The total power requirement of a PC/104 module stack is the sum of that required by each of the modules in the stack.

3.3.4 NASA and Space Qualification of PC/104 Boards

Space applications exhibit severe constraints on weight, volume and power. Radiation, EMI and EMC are also significant issues [NASA, 1998 B]. In addition, NASA is always concerned about achieving cost effective solutions that are compatible with safety and launch constraints.

NASA approved PC/104 boards as an appropriate electronics solution for space payloads. Characterized by low power, small size, COTS effective hardware and reliable electronics, PC/104 boards meet most of the requirements for space systems. However PC/104 boards are not acceptable "as is" for space applications. Rather they require some electrical and mechanical modifications to address the microgravity environment and its implication. The challenge was to space qualify PC/104 boards while maintaining the full compatibility with the PC architecture and form factor. The modifications advised by NASA for space-qualification of PC/104 boards are explained below [Abbott, 2000].

Thermal Modifications

Convection cooling is the largest component for ground based applications, but in the microgravity environment of orbit, there is no convection. As a result, thermal conduction is the primary method of thermal control in a microgravity environment. An electronically isolated thermal strip is added to each side of the board to dissipate heat by conduction.

Power Issue

Most PC/104 processors consume as little as 4 W of power. When power management is invoked, the power consumption falls to 1.3 W.

Software Modifications

Like most COTS products, PC/104 boards do not have protection from radiation making them unusable in space applications. Typically, radiation induced-errors take the form of unexpected bit changes in memory locations. EDAC (Error Detection And Correction) is a coding method used to protect memory and to transform PC/104 processors into radiation tolerance boards. In addition, the electronics is encapsulated in space qualified aluminum housing to limit EMI.

3.3.5 How PC/104 are Used in Real Applications

Although configuration and application possibilities are practically limitless, PC/104 modules are generally used in two ways in actual embedded systems. One approach is to stack multiple PC/104 modules together to create a full-featured system as shown on Figure 3.3. In this configuration all the functions of the embedded system are contained on PC/104 modules. The number of boards that can be stacked on top of each other depends on the amount of bus drive capability on the CPU as well as the bus loading of the add-on modules. Some PC/104 manufacturers will only guarantee 5 or 6 cards in a stack, while others have tested their CPU modules with up to 10 boards or more [PC/104 FAQ]. In a second approach, PC/104 modules are distributed horizontally – plugged into custom, application baseboards like multi-chip "macrocomponents" (see Figure 3.5).

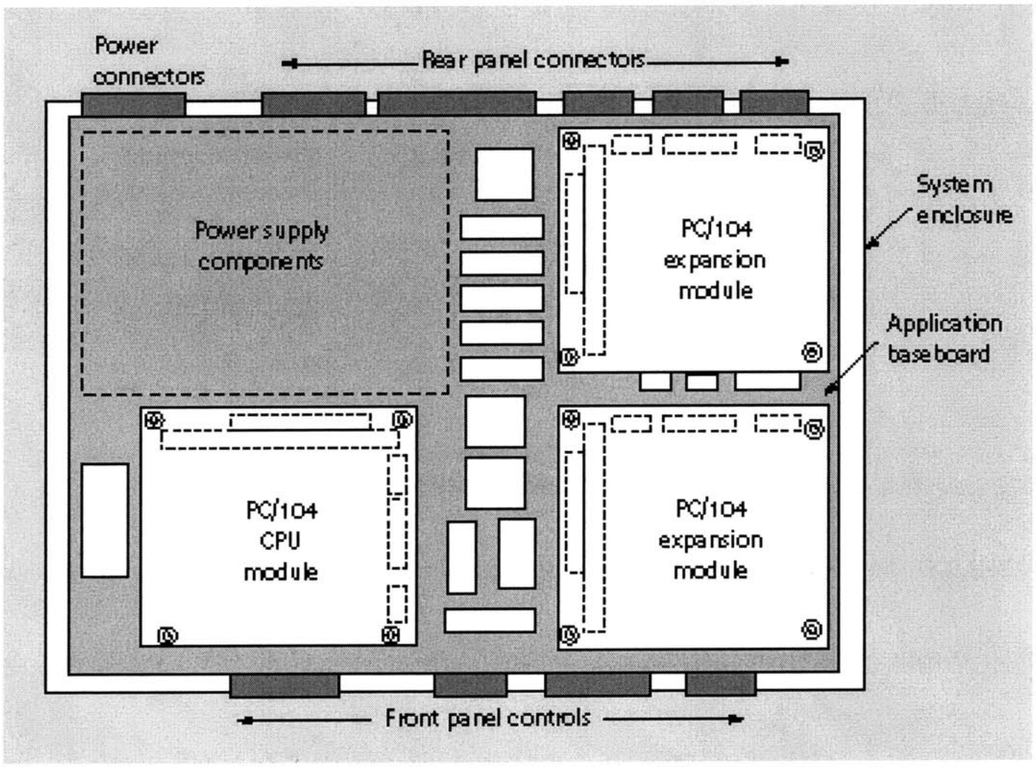


Figure 3.5: This drawing shows PC/104 boards used as macrocomponents on an application baseboard in order to optimize the application specific size requirements [Lehrbaum, 1997].

The 6-DE-104/16 board from Douglas Electronics Inc. (San Leandro, CA), allows two PC/104 modules to be placed side-by-side. Moving one or more PC/104 boards completely out of the PC/104 stack allows more flexibility on the shape of the designed system. If the application specific baseboard is designed, it can be any shape. PC/104 application baseboards usually provide several PC/104 locations and include power supply components, signal conditioning, I/O connectors and specialized interfaces [Ampro, 2000].

3.4 Architecture and Design of the Advanced Load Sensors

3.4.1 MICR0-G Design Guidelines

The MICR0-G project is designed to provide an enhancement to the EDLS system that flew on Mir. It utilizes essentially the same sensors – including the load cells and restraint devices – but attempts to take advantage of the drastic increase in computing hardware and software capabilities to produce a far more robust, flexible, and mobile interface to the sensors.

In order to reduce development cost, and possibly later production cost, as many commercial off-the-shelf (COTS) products as possible have been used while designing the advanced sensors. Except for specialized electronics, most boards exist as off-the-shelf products and have great capabilities as well as optimized size and power consumption. Off-the-shelf hardware also leads to modularity in the design, increasing the upgradeability of the system. The main disadvantage of using COTS hardware is that few off-the-shelf boards are space-qualified. However, since our system is based on boards commonly used in space applications (embedded CPUs, PC-Card modules, A/D converters, wireless networking PC-Cards, etc.), we expect that other systems in development for the ISS will make use of the same components. The cost for space-qualification can then be shared with other developers to reduce the global development budget.

Space-qualification of payload systems is a long process that can take up to 10 years. As a result, it is critical to make use of the latest technical developments to avoid rapid obsolescence of space systems. Another important requirement for space applications is a reasonable level of on-orbit maintainability and repairability to assure a system lifetime longevity.

The design of the advanced sensors emerged from the previous guidelines and the technical lessons learned from the EDLS experiment summarized in section 3.1.2.

3.4.2 Design Summary

The advanced sensors consist of three parts: (1) the Sensor Mechanical Unit (SMU) (2) the Sensor Electronics Unit (SEU) and (3) The Central Computer (CC). The SMU contains the astronaut restraint mechanism (either a foot loop or a handle) as well as the load cells to measure forces and moments. The SEU contains all the necessary electronics for data processing and networking. Each sensor incorporates its own SEU that is contained in a case screwed up underneath the sensor. The Central Computer is used as a server for networking and incorporates the storage devices to store all the data recorded by the sensors. This base station communicates wirelessly with the whole set of sensors and provides the main user interface with the system through a laptop computer display. The Central Computer can be installed wherever it looks convenient for the crew and can be moved at any time inside the space module. This MICR0-G advanced sensors' architecture is illustrated in Figure 3.6.

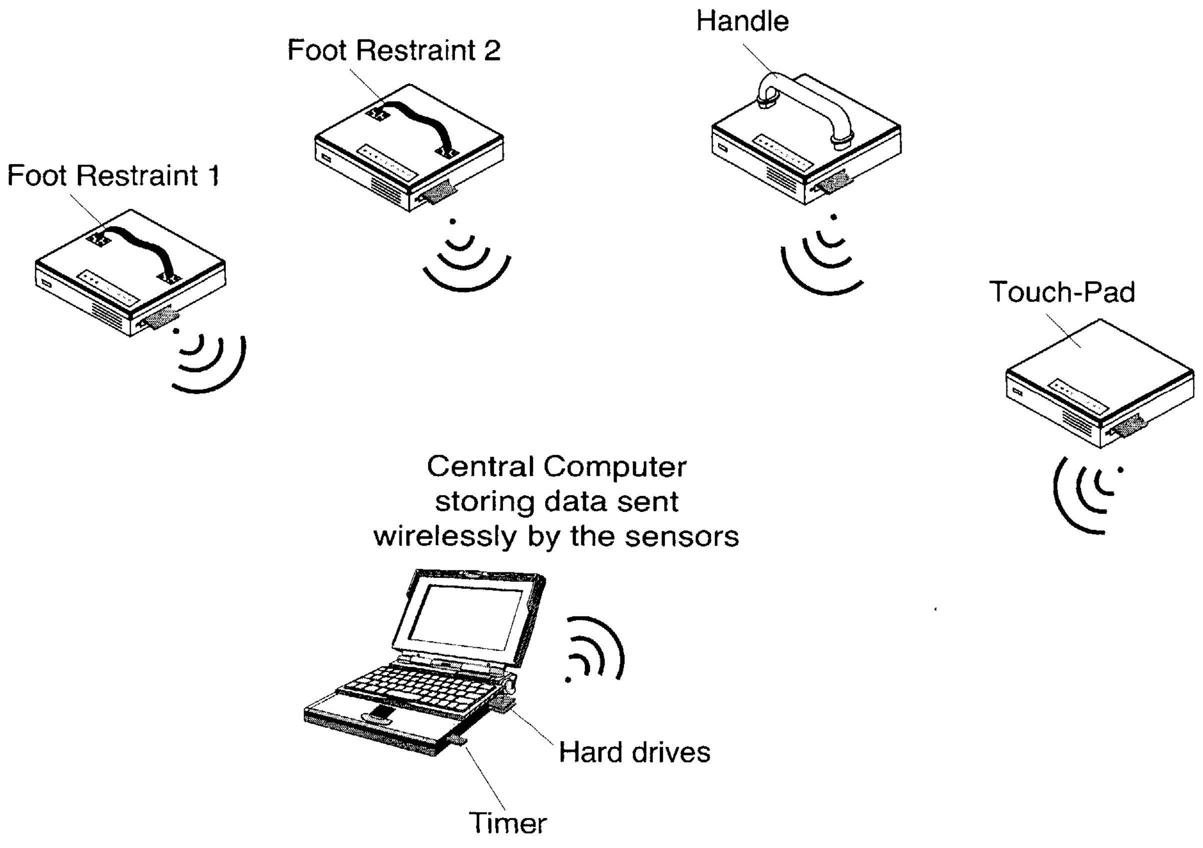


Figure 3.6: The advanced sensors' architecture consists of several sensors incorporating the data processing electronics and the RF interface, and communicating wirelessly with a central laptop computer.

Compared to the DLS/EDLS experiments, the major improvement in the design is the lack of long umbilical cables connecting the sensors to the main computer. The advantage of this approach is the vastly increased mobility of the system. The sensors can be placed wherever the crew members feel they are the most useful (i.e. high traffic areas), and the central computer can be easily moved around the module depending on the needs and preferences of the users as well as available space.

Instead of a central computer processing data from the whole set of sensors, as was the case in the DLS and EDLS experiments, each advanced sensor incorporates its own processing electronics and thus becomes an independent unit. Redundancy increases flexibility of the system and its ability to recover from a failure. If one sensor's electronics fails, the experiment is not completely deadlocked, as was the case when the MODE ESM computer failed on the NASA/Mir 22-23 mission [Amir, 1998]. The advanced load sensor system can keep on working without the defective sensor. In addition, if the astronauts want to use additional sensors, the system can easily accommodate more than four sensors.

Finally, the laptop computer provides a nice and easy interface to other payloads on-board the ISS. In particular, it provides various I/O connectors to synchronize the sensors with the ELITE-S2 video capture system.

3.4.3 The Load Sensors

The key experiment hardware is a set of four load sensors including a handhold, a touchpad, and two foot restraints. Post-flight tests of the original EDLS sensors returned from Mir, revealed an excellent overall quality and accuracy. After two years in orbit, the error of force and moment measurements was less than ±1.6% of the full scale load (400 N, 50 N·m) for which the sensors were designed for [Amir, 1998]. Therefore, NASA decided to keep the same load sensors for the development of advanced load sensors for the ISS. The sensors are 6 degree-of-freedom (d.o.f.) units able to measure applied forces and moments about the x-, y-, and z- axes. All four space-qualified sensors as used for the EDLS experiment on Mir are shown in Figure 3.7.

As part of the final design of the advanced load sensors, the top plate of the sensors will incorporate a LED device to provide real-time feedback to the astronauts on the magnitude of the applied loads. This slight modification of the enclosure is the only difference between the first and the second generation of load sensors.

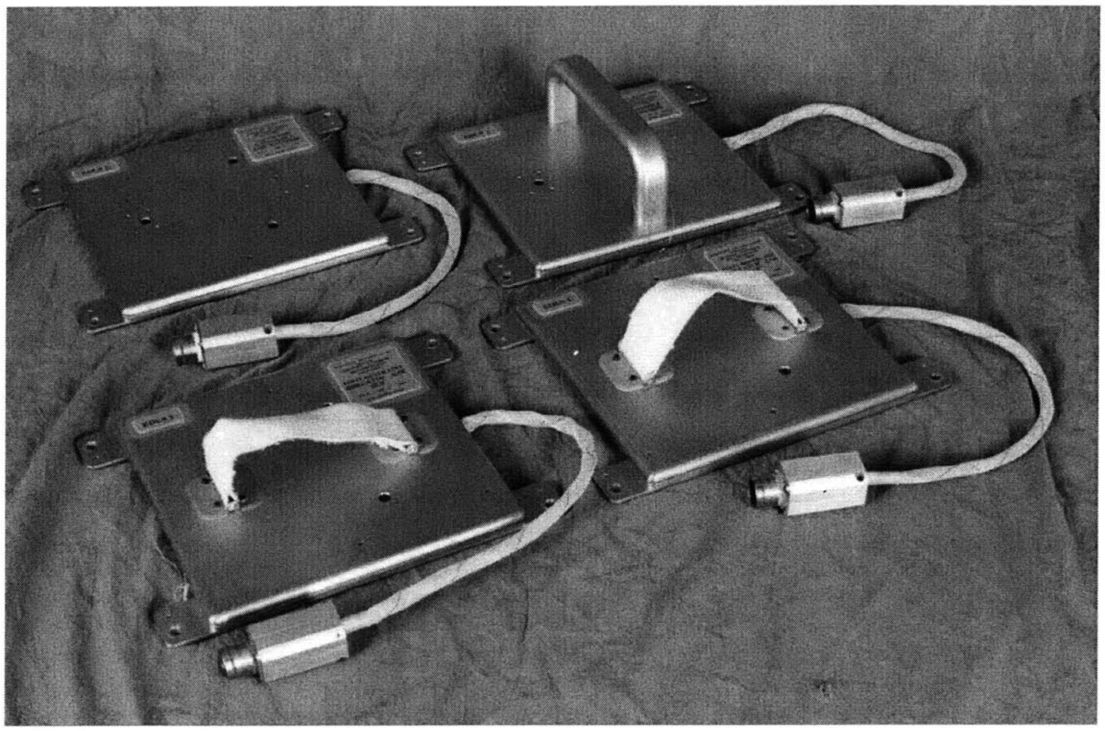


Figure 3.7: The four EDLS sensors: handhold, a touchpad and two foot restraints.

All three sensor types – the handhold, the foot restraint and the touchpad – share the same design to simplify manufacturing, assembly, exchangeability, and reduce cost. The sensors are basically square in shape (measuring 24.1 cm by 24.1 cm) and have small protrusions for screws to attach them to the box containing the electronics underneath. The touchpad's functionality was envisioned to be a flat surface that the astronauts would use to push themselves off using either their hands or their feet. The handhold has an additional handle and the two foot restraints each have a soft loop mounted on the top plate. The sensors are made from aluminum AL 7075-T63 with an Iridite finish per Mil-C-5541, Class 3 specifications. AL 7075 is a NASA approved material and the T63 heat-treated form has a yield strength of 420 MPa.

3.4.4 The Sensor Electronics Unit

Restraints and Mobility Aids (R&MAs) on-board the ISS are provided to support IVA equipment. Each sensor will be attached to standard ISS hand rails or seat tracks that are standard R&MAs interfaces for ISS payloads. In any case, a whole of about 8 cm exists between the attached sensor and the racks. The advanced sensors make use of this substantial space to incorporate some electronics in an aluminum case screwed up underneath each sensor as shown in Figure 3.8.

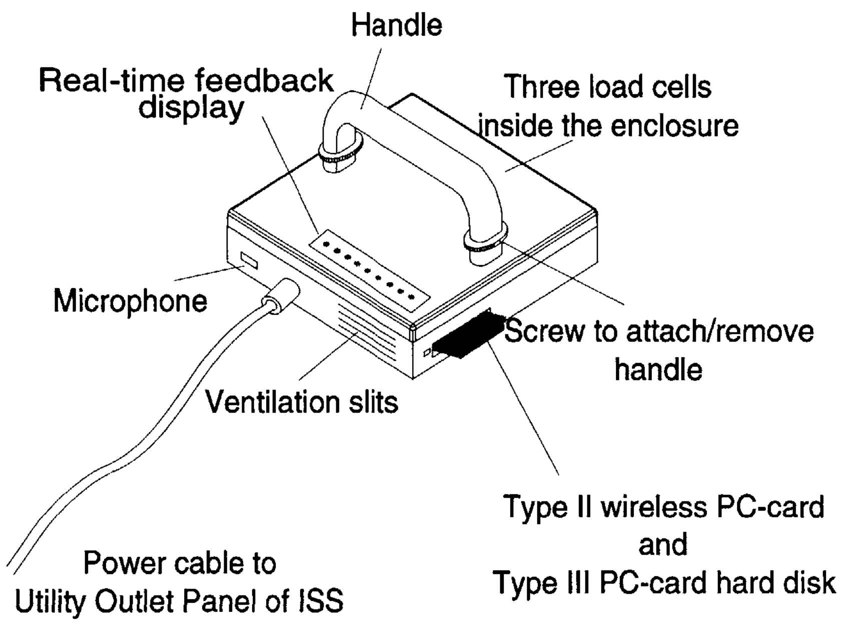

Handle
Real-time feedback display
Three load cells inside the enclosure
Microphone
Screw to attach/remove handle
Ventilation slits
Type II wireless PC-card and Type III PC-card hard disk
Power cable to Utility Outlet Panel of ISS

Figure 3.8: Drawing of a sensor (e.g., a handle) and the SEU unit attached underneath.

Consequently, the design objective was to fit the entire signal processing and data acquisition hardware into a case with the same foot print (24 cm by 24 cm) as the original sensor and a height of no more than 8 cm. The PC/104 technology achieves such a compactness while satisfying exhaustively the specifications of the system and providing even more enhanced capabilities. With dimensions of 24 cm x 24 cm x 8 cm, the SEU's volume is only 4.6 liters. The estimated mass of this unit is less than 4 kg. Thus the advanced sensor design achieves a reduction in size by a factor of ten in comparison with the MODE ESM [Bokhour, 1992] and about a factor of four if the number of channels is taken into account. In the advanced design, the mass per channel is four times less than in the EDLS system.

The SEU consists of the following main components:

- Signal Conditioning Board
- A/D Converter
- Central Processing Unit (CPU)
- PC-Card Module
- Wireless PC-Card
- Power Supply Components
- LED Device

The Sensor Electronic Unit can be divided into three layers: the signal processing layer, the digital data processing layer and the networking layer. At the bottom of the unit is the signal conditioning board which is a customized electronic board designed and fabricated by Payload Systems, Inc. (PSI, Cambridge, MA). Because of their familiarity with the sensors and the previous DLS/EDLS hardware, PSI was selected by MIT as the primary subcontractor for the MICR0-G project. They were responsible for the design, fabrication and modification of the signal conditioning board in order to meet the advanced sensors' related requirements. Since most of the SEU's hardware is based on PC/104 electronics, the customized signal conditioning board is designed to have the same foot print (9.0 cm by 9.6 cm) as PC/104 boards. As a result, it becomes easier to arrange these parts to build the flattest possible unit. The load sensor is connected to the SEU with a cable coming out of the sensor and plugged into the signal conditioning board.

The second layer of the SEU consists of the digital data processing. This layer is based on two commercial off-the-shelf PC/104 boards: the A/D converter converts analog signals into digital data and the CPU processes the resulting data. A cable connects the A/D converter to the signal conditioning board in the layer underneath it. The CPU and the A/D converter are PC/104 boards.

The top layer of the SEU is the wireless interface with the Central Computer. A PC/104 module is used to provide two PCMCIA slots accepting any type of PC-Cards (Type I, Type II, and Type III). The possible combinations are slightly different depending on the location of the module inside the PC/104 boards stack. There is no restriction on the use of Type I and Type II PC-Cards. When two cards are used, including a Type III PC-Card, the latter can be used in either socket when the PC/104 module is on top of stack, but in lower socket only otherwise. Our design makes use of a single slot to insert a wireless network Type II PC-Card. The remaining slot may be used for a Type III space qualified hard disk as an optional storage capability. Details on the hardware that has been used to develop the advanced load sensor prototype are provided in Chapter 4.

3.4.5 The Network

The network architecture is based on the TCP/IP stack, which holds the core communication protocols commonly used today over the Internet. TCP/IP provides a reliable two-way communication channel between two computers. As a result, the involved computers can establish a client-server dialog in order to exchange information.

The sensors utilize TCP/IP to send the processed data to the wireless LAN capable central laptop and to implement additional networking capabilities such as monitoring or time synchronization. The use of a widely spread networking protocol such as TCP/IP ensures a great interoperability with the existing network. The whole system composed of the sensors and the central laptop can be integrated into the ISS wireless network or can be organized into a stand-alone LAN.

3.5 Software Development

3.5.1 A Brief Report on Operating Systems for Embedded Devices

Initially, it is appropriate to define what is meant by "embedded." An embedded device is a piece of microprocessor-based computing hardware, usually on a single circuit board, which has been built to run a specific software application. The term "embedded" refers to the fact that these devices were originally used as building blocks in larger systems. When it comes to designing embedded systems, the goal is often to use as few resources as possible. In embedded systems, less is more, in many ways. Using less resources means less cost, less heat generation, more battery life, more reliability and best of all, a more successful product.

Recent years have seen a broadening array of OS software options for embedded and real-time applications. Today, most non-desktop embedded systems are based on one of the following alternatives:

- Microsoft Windows: Win98, WinNT, WinCE.
- A wide variety of proprietary real-time OSs (RTOSs) – VxWorks, Lynx, QNX, pSOS, etc.
- "Just plain DOS" – Microsoft MS-DOS, Datalight ROM-DOS, Caldera DR DOS.
- UNIX, in many forms – SCO, BSD, Solaris, etc.
- Linux, with several major and many minor distributions [Huet, 2000].

With the increasing complexity and advanced capabilities of embedded computers, advanced OS technologies have become a necessity, in order to take advantage of the full features of the system and to provide the application sophistication that has come to be expected. Embedded systems are now commonly based on 100+ MHz 486/586/Pentium class CPUs, and many incorporate sophisticated Graphical User Interfaces (GUIs), intranet/internet connectivity, high capacity flash memory ("solid state disks"), and so forth. On top of this, there is a rapidly accelerating pace of hardware and chipset innovation accompanied by extremely rapid obsolescence of the older devices. As a result, it is increasingly challenging for OS suppliers to keep up with user demand that the latest devices be supported. The result of all this, is that the number of practical OS options – except for low end microcontroller-level embedded systems – is rapidly narrowing. [Lehrbaum, 2000]

Today's highly sophisticated and empowered intelligent embedded systems demand the full power, sophistication and currency support provided by high-end operating systems like Windows 98 or Windows NT. Yet, embedded systems demand a high degree of reliability (for non-stop, unattended operation), and the ability to customize the OS to match an application's specific requirements.

In the past, system developers have embedded DOS successfully, using it primarily as a platform for running the system's specialized application software. But DOS is no longer relevant in the world of large memory spaces, 32 bit processors, complex GUI functions, and above all Ethernet/Internet connectivity. The replacement for DOS – Windows – unfortunately doesn't provide a particularly developer-friendly toolset and is overstuffed with desktop features making Win98 and WinNT disk space consuming for embedded systems. Microsoft developed the 32-bit operating system WinCE, similar in some ways to WinNT, but designed for embedding and scaleability in smaller machines such as handheld computers, pagers, and embedded systems. At that point, reports on its practical use are pretty bad: most vendors do not provide support for WinCE, and drivers are difficult to program and lack flexibility and integration [Starner, 1998].

As an alternative, embedded system developers have used a wide variety of real-time operating systems (RTOSs) that provide support for embedded applications and are designed to accommodate the special requirements of embedded systems. A significant problem with RTOSs, however, is that they lack standardization, resulting in a high development investment and serious risks of dependence on the support capabilities of the RTOS vendor and the development capabilities of a single, highly skilled programmer.

Fortunately, a new and exciting alternative has emerged: Linux. Linux provides powerful system management facilities, a rich cadre of peripheral interface support, has an excellent reputation for robustness and reliability, is well documented, and best of all, is available at no charge with complete source code [Williams, 1999].

3.5.2 The Perfect Solution: Linux

Linux is an operating system that is similar to Unix, but is a completely independent product. It has been or is being ported to run on many processors and scales well from small embedded systems to desktop 64 bit architectures. It has many of the same commands and programming interfaces as Unix so that Unix people can easily work with it. Over the past couple of years, Linux has gained both ground and popularity with companies whose products make use of embedded hardware and software. The reasons of this success are multiple and discussed below [Linux Online!, 2000].

Small foot-print Linux

For many embedded systems, the main challenge in embedding Linux is to minimize system resource requirements in order to fit within constraints such as RAM, solid state disk (SSD), processor speed, and power consumption. Although Linux, in its common condition, is relatively large and demanding of system resources, it can be scaled down to configurations barely larger than DOS. Realistic low-end requirements for Linux are in the range of 4-8 MB RAM space, and a couple of megabytes of Flash.

License and Royalty Fees

Most commercial operating systems, like Windows, have an up front distribution fee plus a per royalty fee. By contrast, Linux is free software commonly called Open Source Software. Free software results in less expensive final products. Custom written software that runs on Linux may be proprietary and licensed or controlled in any way. However, in the spirit of Linux, software of a generic nature is encouraged to be contributed for all to use. Today, dozens of software companies support Linux in a wide range of applications, from desktop workstations, to deeply embedded devices. All share a common kernel, common drivers, common GUIs, common utilities. Yet, all provide their unique application oriented add-ons, development assistance, and support.

In addition to the basic license fee charged by commercial operating systems, there is often a steep charge for software development tools used to create and test the program. Linux provides all of the basic tools "in the box". This includes C, C++, Java, and others. Fancier tools are available for free or a modest cost. Development tools are designed to support a variety of computer architectures and debugging environments.

Documented

A wealth of documentation is available ranging from user friendly tutorials for novices to Unix style cryptic manual pages. Much of this material is publicly available on the Internet. It is just a matter of understanding where to look and who to ask. In addition of documents, the Linux community provides a great support for both novices and experts.

Network Applications

TCP/IP networking support is seen as a huge plus, as it is either expensive or unavailable with other platforms. Just about every protocol and network interface has made its appearance on Linux. Its kernel handles network protocols even more efficiently than standard Unix implementations.

X Generation

When Linux is used on a PC or workstation, the preferred GUIs are based on X-Windows. This is standard fare for Unix systems. In fact, X-Windows is a platform independent client and server model that lets users at one computer open windows on another computer. This lets programs and users located on various flavors of Unix and Linux intemperate seamlessly together over a network. Linux offers a variety of window managers that provide a choice of “look and feel” models.

For those who are used to Windows95, there is even a lookalike interface so that you can feel right at home. Unlike Microsoft Windows, Linux does not require a graphical interface. Linux is based on the GNU tool chain. This provides a very complete and seamless cross platform development tool chain from editor to source level debugger. Its' C compiler seems to have every optimization built into it and produces very efficient executable code.

Conclusion

Linux is a versatile and cost effective operating system for embedded systems. It can be embedded in a surprisingly small system to handle simple tasks and scaled up to handle more complex tasks. Linux can run on most microprocessors with a wide range of peripherals and has a ready inventory of off-the-shelf applications. Development cycles are shortened through the use of mature tools, open source code, substantial documentation and available support services.

For embedded systems having restricted disk space, with the need of a scaleable and reliable operating system supporting TCP/IP networking, Linux is simply the best choice and is the OS choice for the advanced MICR0-G miniaturized electronics system.

3.5.3 Software Development

A Customized Operating System for the MICRO-G Advanced Load Sensors

The state-of-art of operating systems for embedded applications, as described in the previous sections, led to the decision on using Linux for the software development of the MICR0-G advanced load sensor prototype. Many Linux distributions are available but they are all variations on the same theme – that is, they are pretty much collections of the same basic components, including the Linux kernel, command shells (command processors), and many common utilities. The differences tend to center around which of the many hundreds of Linux utilities have been included, what extras are added, and how the installation process is managed. The core operating system itself is a fairly simple micro-kernel architecture. Networking and file systems are layered on top of the micro-kernel in a modular fashion. Custom drivers, application programs and other features can be either compiled in or added to the kernel at run time as loadable modules. This provides a highly modular building block approach to construct a custom embedded system.

The main requirements for our operating system were a network connectivity, drivers and bootable files for the DiskOnChip, drivers for the PCMCIA wireless card and drivers for the A/D converter. The software development began with testing several "embedded" distributions of Linux, including LEM, ETLinux and ThinLinux [Huet, 2000]. Since no distribution exactly matched our application's requirements, the solution was to use the modularity of Linux to develop a customized OS by deleting unnecessary functionality and layering appropriate files, programs and drivers on top of a given Linux distribution. We started with the RedHat 6.2 distribution and created a minimal version for our system. The RedHat 6.2 distribution is almost 1 GB and our final customized OS requires only 20 MB of disk space. Most of the development of the customized OS was done on a desktop machine running RedHat 6.2 and the operating system was then installed on the PC/104 processor.

The main steps of the OS development and installation on the PC/104 processor are summarized as follows:

- Installation of RedHat 6.2 on an external hard drive, using a desktop machine as a staging platform.
- Configuration and development of a kernel for our embedded system.
- PC/104 processor board set-up to boot the RedHat distribution on the hard drive.
- Obtain a list of the essential files from the Ampro documentation of the PC/104 processor board.
- Copy these files from the RedHat distribution to a file system on the flash disk of the embedded system.
- Install the customized kernel onto the flash disk along with the file system and set up the LILO boot loader to boot this kernel and file system.
- Get the PC/104 processor board boot from its Flash disk (DiskOnChip) with no hard drive or floppy drive attached.
- Test the embedded system and progressively add on the customized OS pieces that were missing or developed later (such as drivers).

Driver for the A/D Board

The development of a driver for the Diamond-MM-16 A/D converter from Diamond Systems Inc. (Newark, CA) was the most challenging phase of the software development. Linux drivers exist for the Diamond-MM and the Diamond-MM-32 modules but not for the Diamond-MM-16 module [Diamond Systems Inc., 2000]. We obtained the source for the Diamond-MM and the Diamond-MM-32 modules from the internet and compared these two drivers to list their similarities and differences. The comparison of technical documentations for each of these three I/O modules led to the conclusion that the Diamond-MM-16 was more similar to the Diamond-MM than to the Diamond-MM-32. Consequently, we decided to choose the Diamond-MM driver as a basis to develop a driver for our board. We listed all the differences between the two boards to modify the original driver, yielding a driver that could talk to the Diamond-MM-16 board but was missing a lot of functionality we needed. The challenging part was to implement in the driver the features of the Diamond-MM-16 which did not exist in other drivers. These improvements include the configuration of ranges, hardware-triggered interrupt-driven sampling, and a mode to buffer data for uninterrupted sampling over long time periods.

Wireless Network and Development of Client/Server Programs

The network development began with the installation of linux drivers for the PCMCIA wireless LAN RoamAbout 802.11 card from Cabletron Systems Inc. (Rochester, NH) on the Ampro PC/104 processor board. The laptop computer that we used for the development was a Macintosh I-Book that has a built-in airport wireless LAN card. It uses standard Apple drivers for the airport card and communicates wirelessly with the sensors using the ad-hoc (i.e. computer-to-computer) mode. The linux drivers for the Cabletron card can be found on the web: there is an open source driver as well as a proprietary driver from Lucent Technologies. We are currently using the open source driver wvlan_cs [Hinds, 2000]. We first received mostly corrupted packets while connecting wirelessly the PC/104 processor to the MIT server and could not establish a stable connection. We finally detected connection problems between the PC/104 boards and restacking the boards with the PCMCIA module directly underneath the CPU solved the problem.

Once the network connection was working, we began the development of client/server programs to control the sensors and transfer data to the laptop. The basic idea is that the sensor has a program running all the time that listens on a network port for connections from the laptop. The laptop connects to this port and issues commands, to which the sensors send responses. The force/moment data is part of the responses. The program on the sensor is written in C and the laptop side is in Java. The source codes provided in Appendix A have been written by Jesse Byler, who worked on the software development of the advanced load sensor prototype from May 2000 to January 2001 for his Advanced Undergraduate Project (AUP). Detailed information can be found in Jesse's AUP report [Byler, 2001].

3.6 Calibration of the Advanced Load Sensors

The calibration is a key step in the development of the advanced load sensors. When a load is applied to the sensor, a calibration matrix is necessary to relate the signal measured by the sensor to the corresponding force and moment data in Newtons and Newton-meters. The calibration procedure for the sensors is exactly the same for a foot restraint, a handhold or a touchpad and is further detailed below.

There is *no inherent axis system* associated with the design of the sensors and thus an axis system can be selected arbitrarily. For the calibration, a right-handed orthogonal axis system with the z-axis pointing out of the sensor was chosen as shown in Figure 3.9.

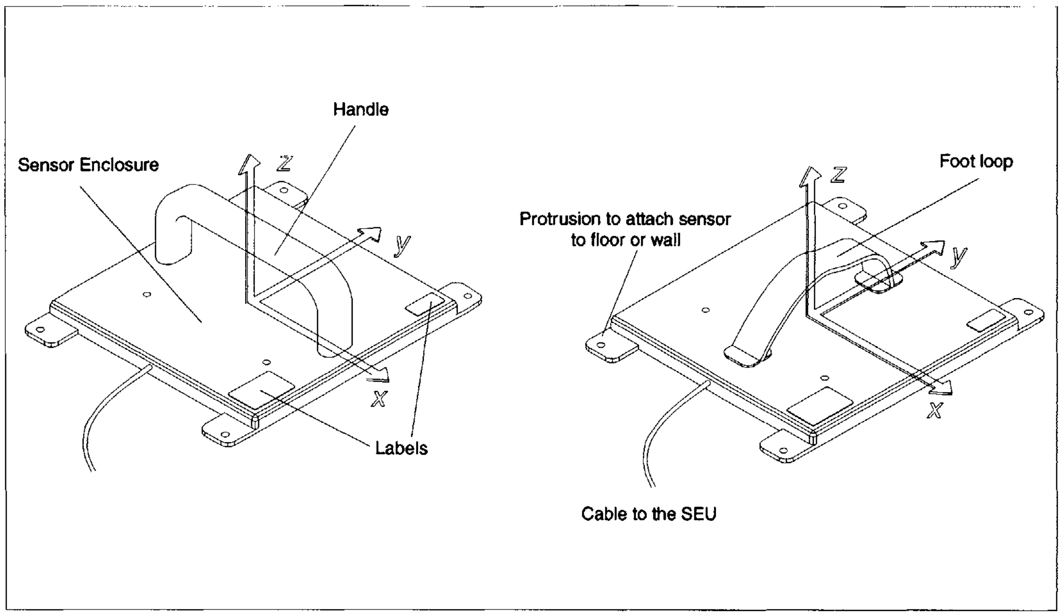


Figure 3.9: Orientation of the right-handed orthogonal axis-system used for the calibration of the handhold and the foot restraint with the advanced electronics system.

The calibration of the ground-based advanced load sensors was similar to the one conducted at Payload Systems Inc. on the EDLS sensors after they returned from Mir [Amir, 1998]. Figure 3.10 shows how the sensors were set-up for calibration when loads in the *x*- direction were applied.

The entire procedure was as follows:

- The top plate of the sensor was removed and replaced with an aluminum plate of approximately the same size with an L-shaped aluminum bar on top of it.
- The sensor with the L-shaped bar was mounted vertically on a rack, so that gravity would act either in the -*y* or in the -*x* direction.
- Fifteen weights ranging from 1.00 to 40 lb. (0.455–18.20 kg) were suspended from the bar. There were four possible locations on the bar from which the weights were hung: (1) Flushed against the plate to create no moment, (2) in the middle of the long-section of the L-shaped bar to create a small moment, (3) at the end of the long-section of the L-shaped bar to create a large moment, and (4) at the end of the short section of the L-shaped bar to create moments about two axes.
- The sensor was then rotated on the rack as to apply forces and moments about another axis.
- To record forces in the -*z* direction, the sensor was placed on a table, the L-shaped bar and its plate removed, and the weights placed at the center of the sensor, resting directly on the flexures.

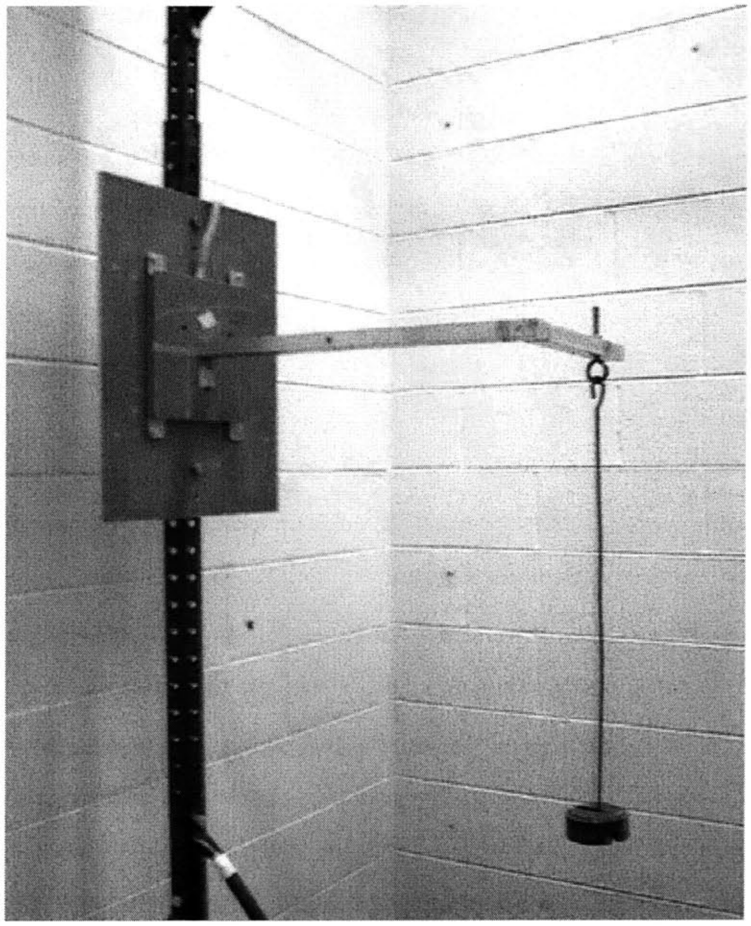


Figure 3.10: Set-up for the calibration of the sensors.

The data was recorded with the advanced electronics system described in this chapter and detailed in Chapter 4. The sensor was connected to the Sensor Electronics Unit (SEU), communicating wirelessly to a laptop computer storing the data on its hard drive. To test the reliability of the sensors and the repeatability of the data collected, the same loads were applied several times and in different time intervals.

The sensors measure loads in terms of a voltage, which is an analog signal. The signal is then amplified and filtered by the signal conditioning board. The A/D converter transforms the analog signal at the output of the signal conditioning board into discrete numbers. The input to the A/D is a voltage that ranges from -10V to 10V so the gain on the signal conditioning board was adjusted to match this range as close as possible when loads from 0 to 40 lb. were applied. The gain chosen for the calibration of both the two foot restraints and the hand hold was 200. The A/D converter is a 16-bit board, dividing the input scale of 20 V into $2^{16} = 65536$ counts. The final data stored on the laptop computer during the calibration were the voltage numbers corresponding to the input of the A/D converter.

An example of the force/torque measurements from a calibration type session is provided on Figure 3.11 and Figure 3.12. Loads were applied 5 lb. by 5 lb. in the middle of the long-section of the L-shaped bar mounted on the foot restraint 1, creating a force in the x- direction and a small moment in the y- direction. The data acquired were voltages corresponding to the input of the A/D converter. The first case corresponds to the "no loading case". Since the signals were not "zeroed" in advance, the corresponding measured values are not 0 V.

When all the calibration procedure was completed, the data stored were then analyzed and processed to obtain the calibration matrix as explained herein:

- First, the average number of counts measured under each load condition was determined. This procedure was done manually using graphs and calibration data to ensure the best results.
- The applied forces and moments (in Newtons and Newton-meters, respectively) as well as the average measured signals (in Volts) for the six channels under each load condition were entered into a spreadsheet. Figure 3.11 and Figure 3.12 show the plot of the data acquired on foot restraint 1 when loads were applied 5 lb. by 5 lb. in the x-direction. The corresponding applied loads and average voltages are reported in Table 3.1.

Table 3.1: Sample calibration data for foot restraint 1, when loads are applied 5 lb. by 5 lb. from 0 lb. to 40 lb.

Applied Forces [N]			**Applied Torques [N·m]**			**Average Measured Signal [V]**					
X	**Y**	**Z**	**X**	**Y**	**Z**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**
0	0	0	0	0	0	0.4078	0.7234	-1.0709	-0.2091	0.6320	0.1055
-22.3	0	0	0	-5.8	0	0.7430	1.0162	-1.7336	-47.2	0.9612	0.2208
-44.6	0	0	0	-11.6	0	1.0805	1.3281	-2.4011	-0.6102	1.2895	0.3382
-67.0	0	0	0	-17.4	0	1.4277	1.6451	-3.0654	-0.8254	1.6032	0.4645
-89.3	0	0	0	-23.2	0	1.7714	1.9858	-3.7269	-1.0409	1.9171	0.6039
-111.6	0	0	0	-29.0	0	2.1089	2.3559	-4.3915	-1.2764	2.2399	0.7633
-133.9	0	0	0	-34.8	0	2.4687	2.7427	-5.0862	-1.5286	2.5722	0.9463
-156.2	0	0	0	-40.6	0	2.8130	3.1012	-5.7478	-1.7568	2.8886	1.1086
-178.5	0	0	0	-46.4	0	3.1653	3.4737	-6.4132	-2.0135	3.2051	1.2872

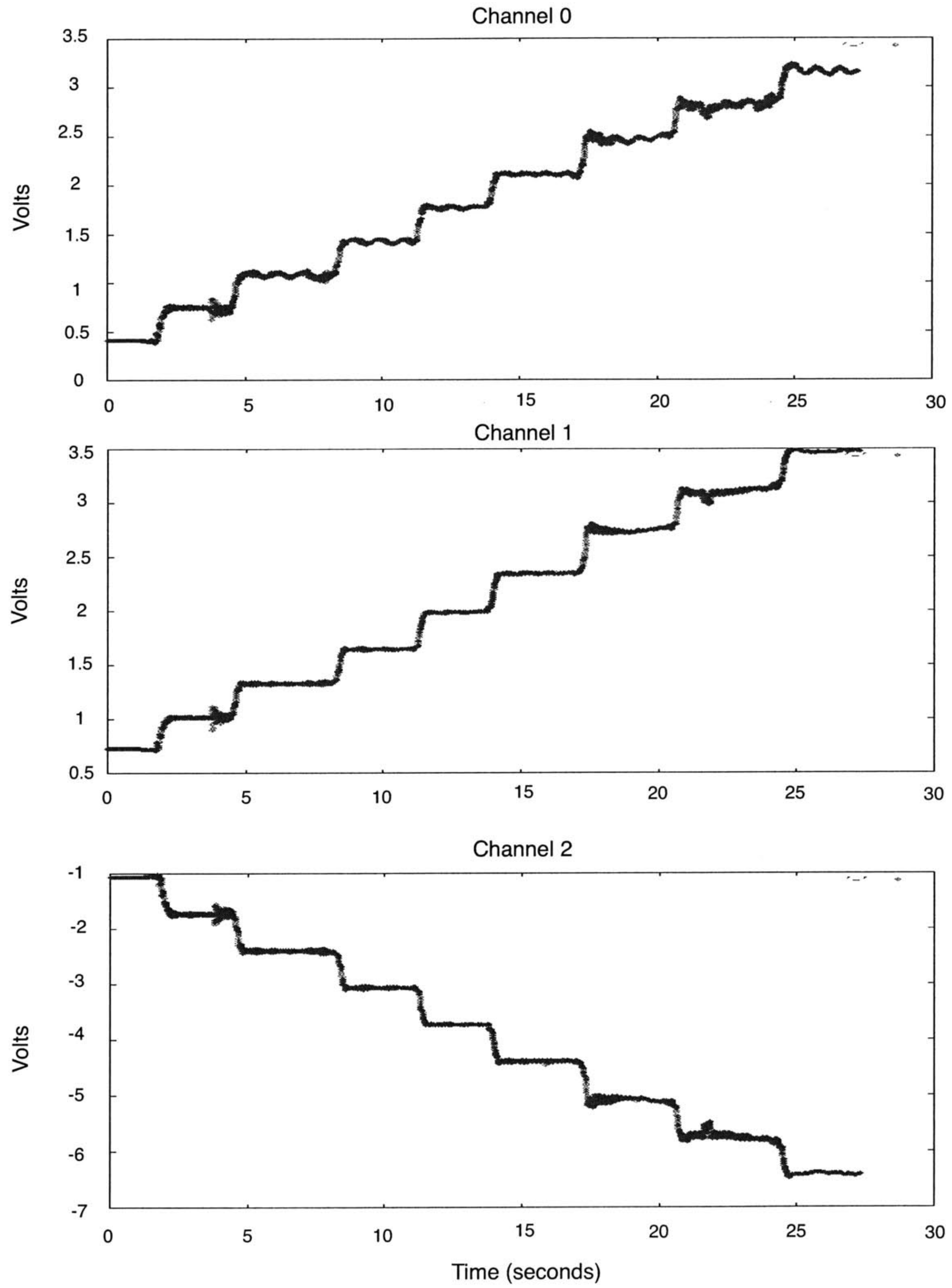


Figure 3.11: The plots show the channel 0 through channel 2 signals recorded while loads were applied to foot restraint 1 in the *x*-direction 5 lb. by 5 lb. The measured signals are in Volts.

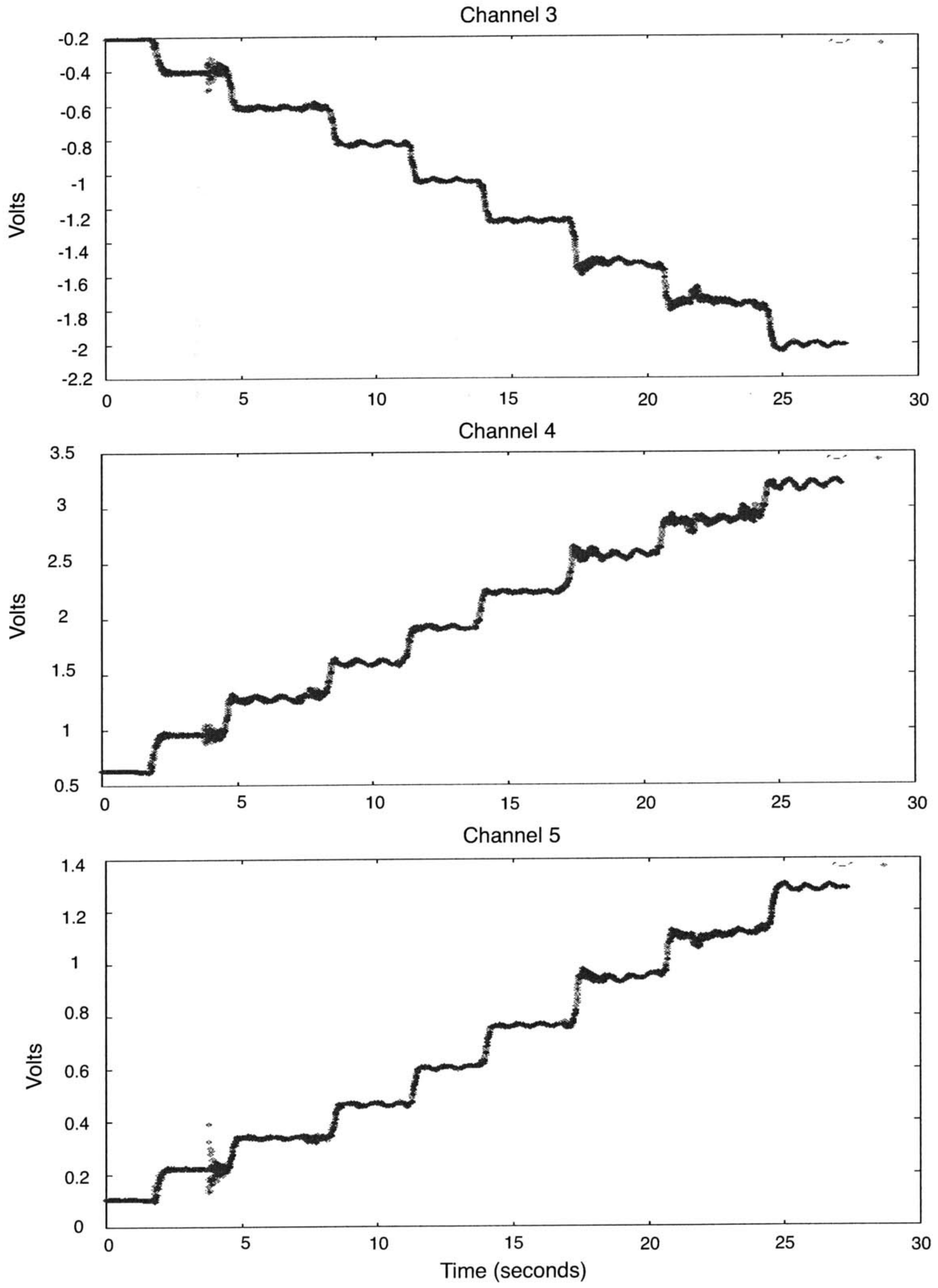


Figure 3.12: The plots show the channel 3 through channel 5 signals recorded while loads were applied to foot restraint 1 in the x-direction 5 lb. by 5 lb. The measured signals are in Volts.

- The calibration matrix was then computed from this calibration data. The calibration matrix relates the measured signal to the applied loads for each sensor. It is calculated by the method of least squares as follows:

Given a 6-element vector $\mathbf{a}$ consisting of the applied forces (in Newtons) in the three axes and the applied moments (in Newton-meters) about three axes and a vector $\mathbf{m}$ of measured signals (in volts), the relation between the two quantities is given by:

$$\mathbf{a} = C\mathbf{m}, \tag{3.1}$$

where C is the so-called calibration matrix. Since there were some $n = 120$ calibration load cases, the quantities $\mathbf{a}$ and $\mathbf{m}$ must be written as 6 x n matrices A and M, so that Eqn. 3.1 is rewritten as:

$$A = CM \tag{3.2}$$

Taking the transpose and multiplying by M yields,

$$MM^T C^T = MA^T \tag{3.3}$$

Taking the pseudo-inverse of MM^T and then the transpose of the entire equation, yields the following expression for the calibration matrix C:

$$C = ((MM^T)^{-1} MA^T))^T \tag{3.4}$$

- Once the calibration matrix is computed, the accuracy is estimated by comparing the measured loads (using the calibration matrix found by the method of least squares) with the actually applied loads. The accuracy of the sensors is evaluated in relation to the so-called *Full Scale load*, which is 400 N of force and 50 N·m of torque for the advanced load sensors. The absolute error of the sensors was computed as follows:

$$\text{Error}_{\text{absolute}} = \text{Load}_{\text{applied}} - \text{Load}_{\text{predicted}} \tag{3.5}$$

The relative error was computed as follows:

$$\text{Error}_{\text{relative}} = \frac{\text{Error}_{\text{absolute}}}{\text{Full scale load}} \tag{3.6}$$

For every load applied, the relative error was computed. If a load case resulted in a very high error, it was eliminated and a new calibration matrix was computed from the remaining values. This process was done until the error was below 2%. Consequently, not every load case that was recorded during the calibration was used in the computation of the final calibration matrix. Charles Keyes, a senior undergraduate student from Harvard, worked on the calibration of the foot restraints and the handhold during the Fall 2000 semester. Final results and calibration matrices for foot restraint 1 and the handhold are detailed in Charles Keyes' report [Keyes, 2001].

CHAPTER 4

FLIGHT OPERATIONS

The previous chapter described the hardware integration, software development and calibration of the ground-based MICR0-G prototype system to be tested during parabolic flights. The objectives of the MICR0-G flight experiment are presented in this chapter. The second section gives an overview of the KC-135 aircraft and the environment provided during microgravity flights. Then a description of the flight experiment combining the load sensor system and the Italian video acquisition system as well as details on both systems' hardware are presented. The in-flight operations during the parabolic flights is described in section 4.5 and will assist future development. Finally, the last section presents recommendations to transition the advanced load sensor ground-based prototype into a space-qualified system.

4.1 Objectives

The primary objectives of the KC-135 flights are to test the functionality of the MICR0-G hardware in a microgravity environment, to demonstrate subsystem compatibility and isolate potential problems. The experiment combines the advanced load sensor system described in Chapter 3 with the Italian video-based ELITE-S system described later in this chapter, resulting in a synchronized kinetic and kinematic system to quantify astronaut IVA loads and motions. The KC-135 flights test the following characteristics of the MICR0-G flight hardware:

- Relevance of the kinetic and kinematic data collected in microgravity
- Wireless capability of the advanced kinetic system
- Human interaction with the prototype hardware under reduced gravity conditions
- Interface and suitability of the kinetic system with the video-based ELITE-S system

The experiment aboard the KC-135 aircraft tests the MICR0-G prototype system and acquires initial kinematic and kinetic data in weightlessness with the ultimate objective of building a future system for use in the International Space Station. The post-flight analysis of the data will determine whether the performance of the entire system meets the functional requirements, defined in Chapter 3. If these requirements are not met, components will be replaced as needed, the entire design reevaluated and a new breadboard model assembled. When the requirements are met, the new MICR0-G prototype will be delivered to NASA as a technology demonstration, with a plan to develop and build spaceflight qualified hardware for the ISS based on the prototype design.

4.2 The KC-135 Reduced Gravity Aircraft

4.2.1 The Reduced Gravity Program and the MICR0-G Project

The Reduced Gravity Program operated by NASA Lyndon B. Johnson Space Center (JSC) in Houston, Texas, provides the unique "weightless" or "zero-g" environment of space flight for test and training purposes. With the development of the Space Transportation System (STS) and the current plans for the ISS, this capability is ideal for the development and verification of space hardware, experiments, astronaut training and basic research.

The MICR0-G advanced load sensors for the ISS were tested aboard the KC-135 turbo jet during January 2001. The sensors were used during four flight days by one or two subjects each day and initial data has been acquired in microgravity for further scientific analysis.

4.2.2 "The Vomit Comet"

The KC-135 is a four-engine turbojet aircraft similar to a commercial Boeing 707 but specifically modified to provide a reduced-gravity environment for research and astronaut training. Figure 4.1 shows the interior and exterior of the KC-135 aircraft. The KC-135 cargo bay test area is approximately 18 m long, 3 m wide, and 2 m high. The aircraft is equipped with electrical power, compressed gas, an overboard vent system, accelerometer data and photographic lights. The interior walls of the cabin are covered with foam padding for the protection of personnel and equipment. The aircraft is equipped with 23 seats inside the test section. [NASA, 2000 G]

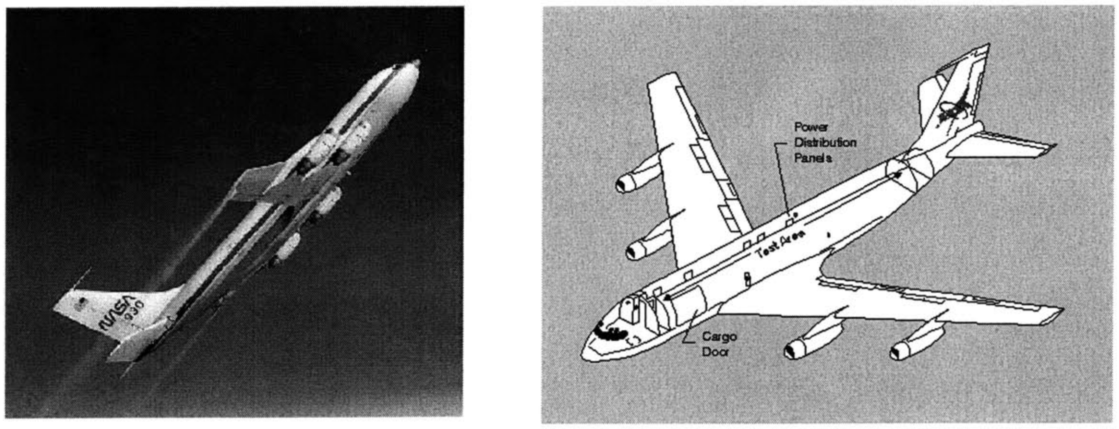


Figure 4.1: Exterior and Interior views of the KC-135 turbo jet. [NASA, 2000 F]

The main features of the KC-135 aircraft for flight payloads are summarized herein [NASA, 2000 F]:

- Electrical power available
    - -- 28VDC, 80 A
    - -- 110VAC, 400 Hz, single phase, 50 A
    - -- 110VAC, 400 Hz, three phase, 50 A per phase
    - -- 110VAC, 60 Hz, single phase, 120 A
- Photography and video support provided
- Most test equipment bolted to the floor or strapped using 50.8 cm (20") tiedown grid attachment points
- Vent/vacuum system to dump fluids overboard
- Liquid or gaseous nitrogen available
- Breathing air available

4.2.3 Trajectory and Missions

A true microgravity environment is attained as the aircraft flies in a series of parabolic maneuvers to produce weightlessness periods between 20 and 25 seconds for each parabola. Figure 4.2 shows a typical microgravity maneuver. However, the maneuver can be modified to provide any level of g-force less than one g. Some typical g-levels used on different tests and the corresponding time for each maneuver are as follows [NASA, 2000 E]:

- Negative-g (-0.1 g): approximately 15 seconds
- Micro-g: approximately 25 seconds
- Lunar-g (1/6 g): approximately 30 seconds
- Martian-g (1/3 g): approximately 40 seconds

A typical mission is 2 to 3 hours long and consists of 40 to 50 parabolas. Each parabola is initiated with a 1.8 g pull-up and terminated with a 1.8 g pull-out. The parabolas can be flown in succession or with short breaks between maneuvers to reconfigure test equipment. The Reduced Gravity Office at Ellington Field, Texas, provides scheduling, test coordination, and in-flight direction for the test programs.

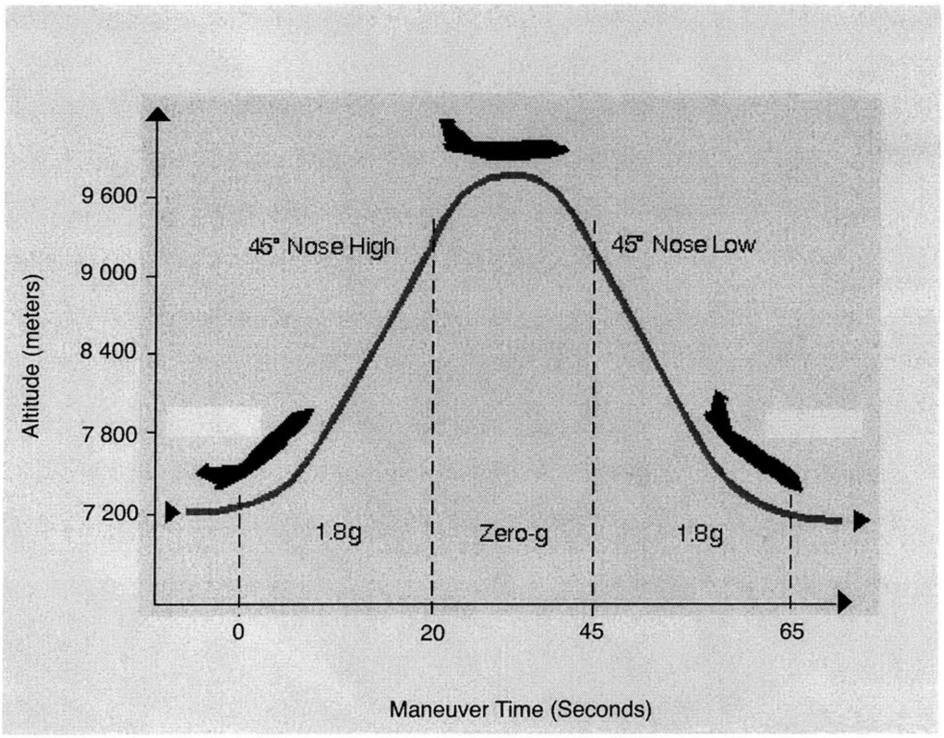


Figure 4.2: Typical parabolic trajectory of the KC-135 aircraft to produce microgravity periods. [NASA, 2000 E]

4.2.4 Requirements for the Reduced Gravity Program

The JSC Reduced Gravity Program User's Guide details all the requirements to fly an experiment aboard the KC-135 aircraft [NASA, 2000 G]. This section gives an overview of the documents and procedures that have been completed to test the MICR0-G experiment in parabolic flight.

Approvals on the Use of Human Subjects

All MIT experiments involving human subjects must obtain approval from the Committee On the Use of Humans as Experimental Subjects (COUHES). Information on the different sections required for the COUHES document is detailed on the COUHES web site (http://www.mit.edu/afs/athena.mit.edu/org/c/committees/couhes/regular.htm). All experiments planning to fly aboard the KC-135 aircraft and involving human subjects, animals, or biological tests must obtain approval from the JSC Institutional Review Board (IRB) at NASA. NASA JSC document 20483, JSC Institutional Review Board: Guidelines for Investigators Proposing Human Research for Space Flight and Related Investigations, details the IRB procedure and requirements. The COUHES and IRB documents for the MICR0-G experiment are provided respectively in Appendix B and C and were approved on December 14, 2000.

NASA Technical Approval

A Test Equipment Data Package (TEDP) must be prepared for each experiment proposed for flight on the KC-135 and submitted to the Reduced Gravity Office no later than six weeks prior to flight. This document is a technical description of the experiment including structural, electrical and hazard analysis, as well as certification for the use of hazardous materials or processes. During the Test Readiness Review (TRR), usually held in the Reduced Gravity Office one day prior to flight, the test equipment is inspected, the supporting analyses and documentation are reviewed and a final integrated verification of flight readiness is conducted. In order to fly aboard the KC-135 aircraft, any experiment requires the unanimous approval of the TRR committee.

Medical Requirements and Physiological Training

Any human subject participating in a reduced gravity flight must provide the results of a KC-135 medical examination or equivalent FAA Third Class aviation physical. The medical requirements and medical forms to be filled out are provided in the JSC Reduced Gravity Program User's Guide [NASA, 2000 G]. The MICR0-G team members used the FAA Form 8500-8 for medical qualifications. The examining physician must be certified as a FAA Medical Examiner or a designated flight surgeon. Consecutively to medical approval, each flight member must receive physiological training. Physiological training includes classroom instruction on the physiological and depressurization risks associated with parabolic flights, a high altitude chamber run to 7,500 m and a hypoxia demonstration. Hypoxia is a state of oxygen deficiency in the blood, tissues, and cells sufficient to cause an impairment of mental and body functions. All MICR0-G flight participants completed physiological training.

4.3 Flight Experiment Description

The MICR0-G experiment combines the advanced load sensors and the opto-electronic motion analyzer ELITE-S in a synchronized integrated system. The load sensor ground-based prototype has been designed and developed at MIT, while the ELITE-S system was developed at the Politecnico di Milano University in Italy.

4.3.1 Equipment Overview

The key experiment hardware of the kinetic system is based on two types of sensors – a handhold and a foot restraint. The sensors provide the functionality of crew restraint devices and mobility aids while measuring the applied forces and moments in the *x*-, *y*-, and *z*- axes (6 degrees of freedom). Each sensor is connected to a Sensor Electronics Unit (SEU) processing the force and moment signals acquired by the sensors. The SEU includes all the necessary electronics for data processing and networking, housed in a Lexan box, which is connected to a power adapter. Each SEU sends wirelessly processed data to a laptop computer using a wireless LAN PC-card. A Macintosh laptop is used as a server for networking and incorporates the main storage devices to store all the data recorded by the sensors. However, a local storage capability is available on the SEU as a possible backup for storing data. The laptop computer also provides the main user interface with the advanced load sensor system. The advanced sensors' architecture is illustrated in Figure 4.3.

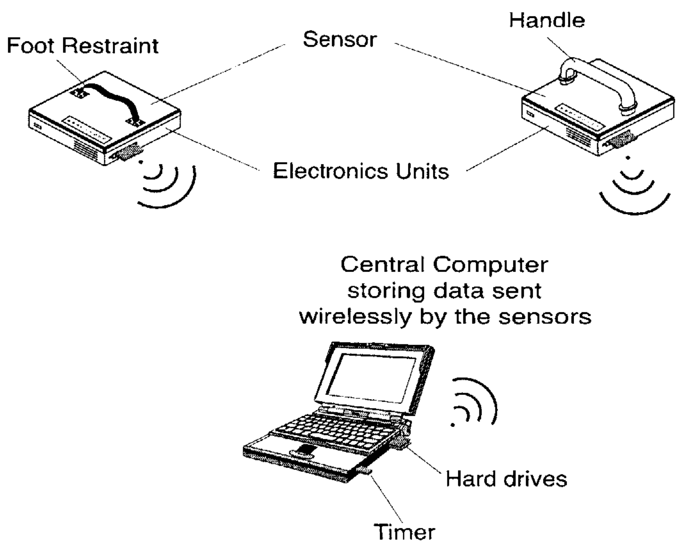

Foot Restraint
Sensor
Handle
Electronics Units
Central Computer storing data sent wirelessly by the sensors
Hard drives
Timer

Figure 4.3: The advanced MICR0-G sensors architecture of the KC-135 prototype consists of two sensors incorporating the data processing electronics and a radio frequency interface to send data wirelessly to a laptop.

The kinematic measurement component of the MICR0-G system consists of a versatile opto-electronic motion analyzer, called ELITE-S, based on the recognition of passive markers. The system is composed of TV Cameras (TVC) equipped with infrared (IR) flashes and connected through a bus to an electronics box housing the main electronics components. The electronics box is connected to a Processing Unit, namely an IBM compatible PC equipped with a I/O digital data acquisition card. Reflective markers are affixed to the subject clothes at the body joints. The IR flashes illuminate the markers, allowing infrared video acquisition of the subject motions. The identification of the markers' position is performed by the electronics box housing the dedicated pattern recognition hardware. The Processing Unit performs data processing in real-time. An LCD monitor is mounted on top of the Processing Unit monitor to provide a real-time feedback on any of the two TVC infrared views. The architecture of the ELITE-S system is shown in Figure 4.4.

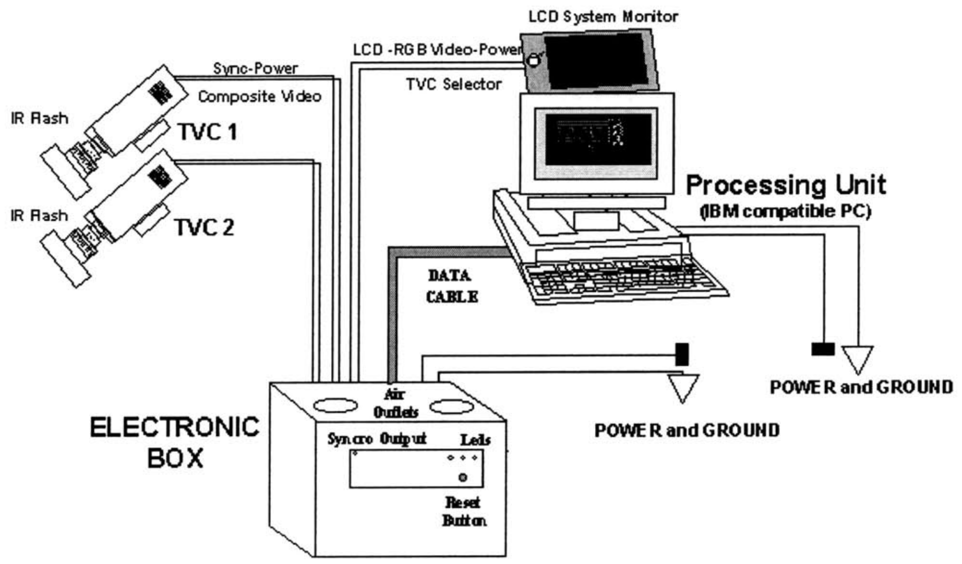


Figure 4.4: Architecture of the ELITE-S system [Politecnico di Milano, 1998].

4.3.2 Description of the MICR0-G Experiment Aboard the KC-135 Aircraft

Figure 4.5 represents the experimental set-up within the KC-135 aircraft main cabin. Due to volume constraints, only two sensors (one foot restraint and one handhold) and two TV Cameras (TVC1 and TVC2) were used. The two cameras were 3 m away from the operational volume measuring about 1 m x 2 m x 2 m. Two operators respectively worked on a laptop computer to acquire the kinetic data recorded by each sensor and on a work station to acquire the kinematic data of the motion analyzer ELITE-S.

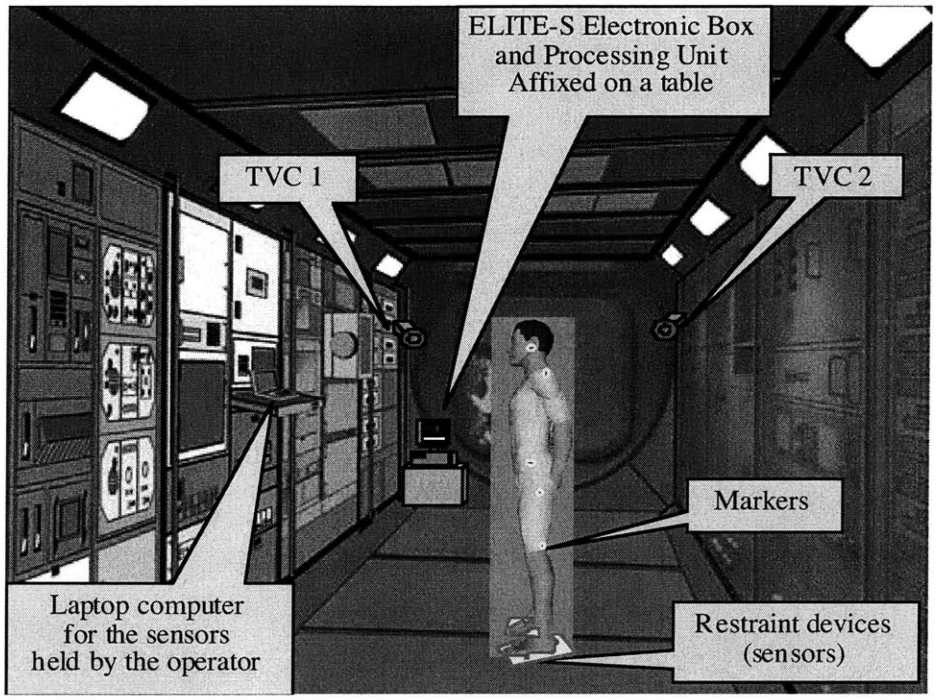

ELITE-S Electronic Box and Processing Unit Affixed on a table
TVC 1
TVC 2
Markers
Laptop computer for the sensors held by the operator
Restraint devices (sensors)

Figure 4.5: The MICR0-G system set-up aboard the KC-135 aircraft.

The subject performed different motions during the microgravity sessions using the handhold and the foot restraint as special mobility aids and restraints. The motion analyzer system captured the subject's three dimensional motions via reflective markers affixed on the subject's clothes at the body joints. Details on the motions performed by the subject for each parabola are provided in section 4.5.

4.4 Detailed Description of the MICR0-G Hardware

The following is a summary of the equipment required for the proposed KC-135 flight tests:

- Three 6 d.o.f. load sensors: a handhold and two foot restraints.
- Two Sensor Electronics Units (SEUs). Each SEU is housed in a Lexan box, incorporating the following electronics: a CPU, a PCMCIA module, a PCMCIA wireless card, an A/D converter, a signal conditioning board, and shielded cables.
- Two power adapters (one per SEU).
- The main electronics box of the ELITE system containing the dedicated hardware for real-time processing of the images coming from the two TV cameras.
- Two TV cameras equipped with infrared flashes.

- An LCD monitor providing real-time feedback on each camera's markers recognition.
- One Macintosh laptop computer for the sensors and one PC IBM workstation for ELITE-S.

The SEUs, the ELITE electronics box and the ELITE data collection unit (the IBM compatible PC) are affixed to the floor using cargo straps. Mounting poles and brackets are used to fix the TV cameras pointing towards the operational work volume. The load sensors are duct taped on the floor.

The commercial version of the ELITE system (including the TV cameras, the Main Electronics box and the IBM compatible PC) has previously flown on parabolic flights in Europe. The flight model of ELITE-S was flown on the Mir Space Station for the EUROMIR'95 Mission [Ferrigno, 1985]. The model utilized for the flight on the KC-135 is the technological model of the ELITE-S system. The same installation and operational protocols are used to ensure proper safety and reliability of the kinematics data acquisition [Politecnico di Milano, 1998].

4.4.1 The Kinetic System

Load Sensors

The key hardware of the MICR0-G experiment aboard the KC-135 aircraft is a set of three sensors, a hand hold and two foot restraint (see Figure 4.6). They are identical to the sensors that have been used on both the Space Shuttle for the DLS experiment and on Mir for the EDLS experiment. The sensors were designed at MIT and fabricated for spaceflight by Payload System Inc. (PSI) (Cambridge, MA) to provide the same functionality as the foot loops and hand rails on the Space Shuttle and ISS for astronaut use.

Both sensors are basically square in shape (measuring 24.1 cm by 24.1 cm) and made from aluminum. They share a same common design consisting of a bottom plate (3.2 mm thick), three load cells, a top plate (3.2 mm thick), and a connector cable (53 cm long including the terminating DB25 connector). The handhold has an additional handle and the foot restraint has a cloth loop mounted on the top plate. On the four corners of the sensor are small aluminum extensions with protrusions for screws to attach the sensors to the floor/wall either directly or using the mounting plates shown on Figure 4.7. Since the KC-135 aircraft is not a highly vibrating environment, the sensors were duct taped directly on the floor without using mounting plates.

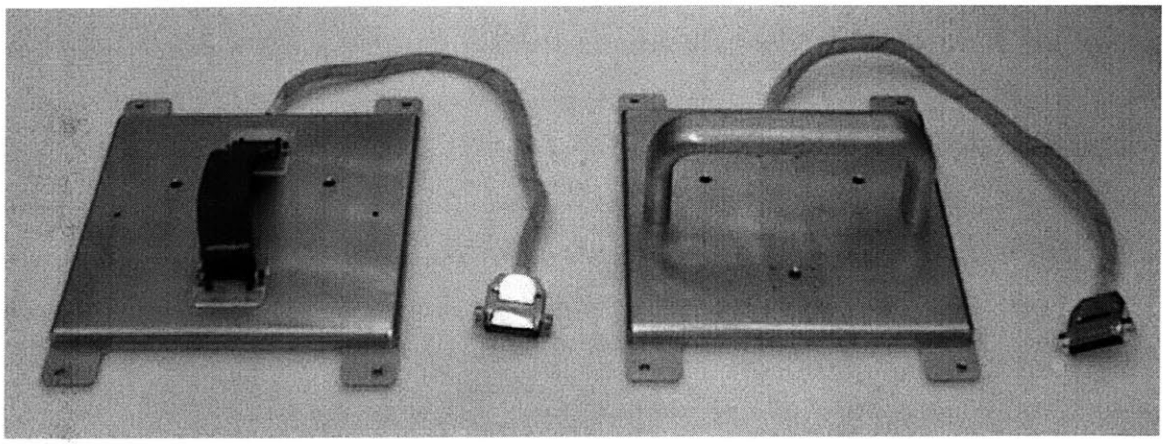


Figure 4.6: The MICR0-G hand hold and foot restraint used aboard the KC-135 aircraft.

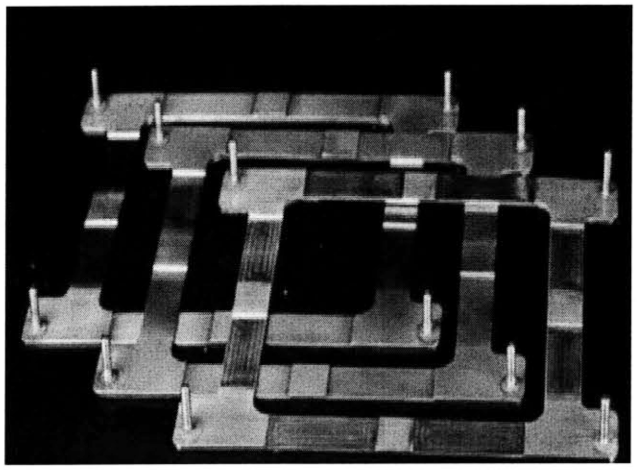


Figure 4.7: Mounting plates used to secure the sensors to the floor.

Each sensor measures the applied forces and moments in the *x*-, *y*-, and *z*- axes (6 degrees of freedom). The sensing of the forces and moments is accomplished via a full strain gage design. The sensors use bonded-resistance strain gages consisting of a grid of very fine wire with electrical resistance that varies linearly with the strain applied to the device.

Each sensor has three load cells arranged along a circle each 120° apart as is shown in Figure 4.8. Each load cell is equipped with two *full wheatstone strain gage bridges*. Each full bridge provides one of the six output signals. Signals *V1*, *V3*, *V5* are from the bridges on the center beams and signals *V2*, *V4*, and *V6* are from bridges on the side beams.

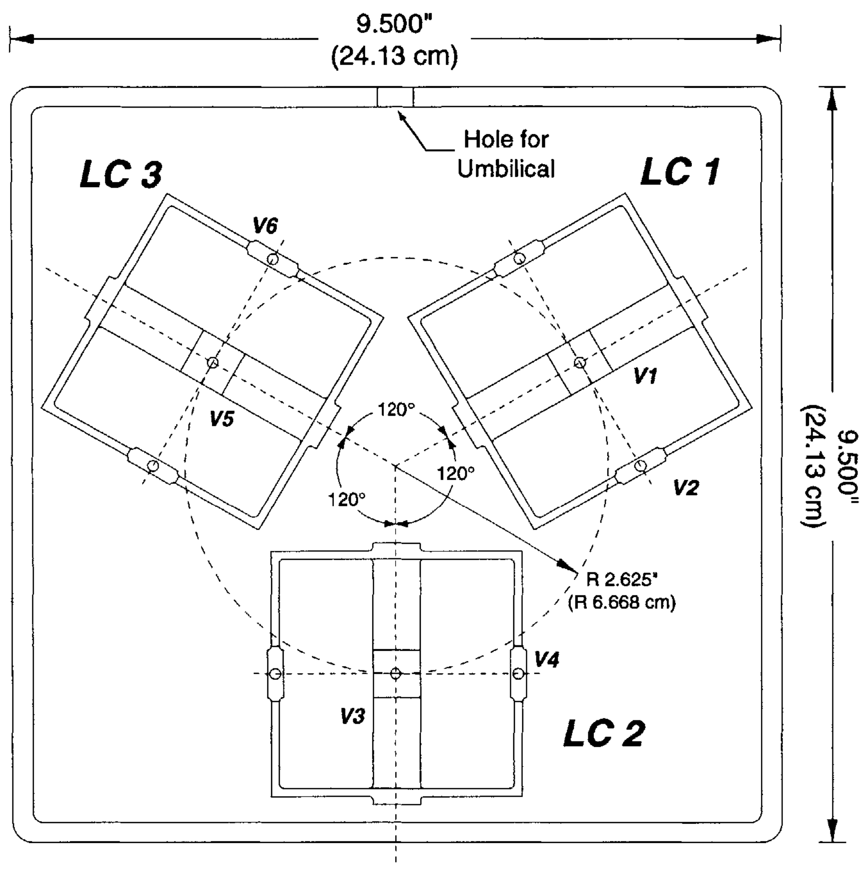
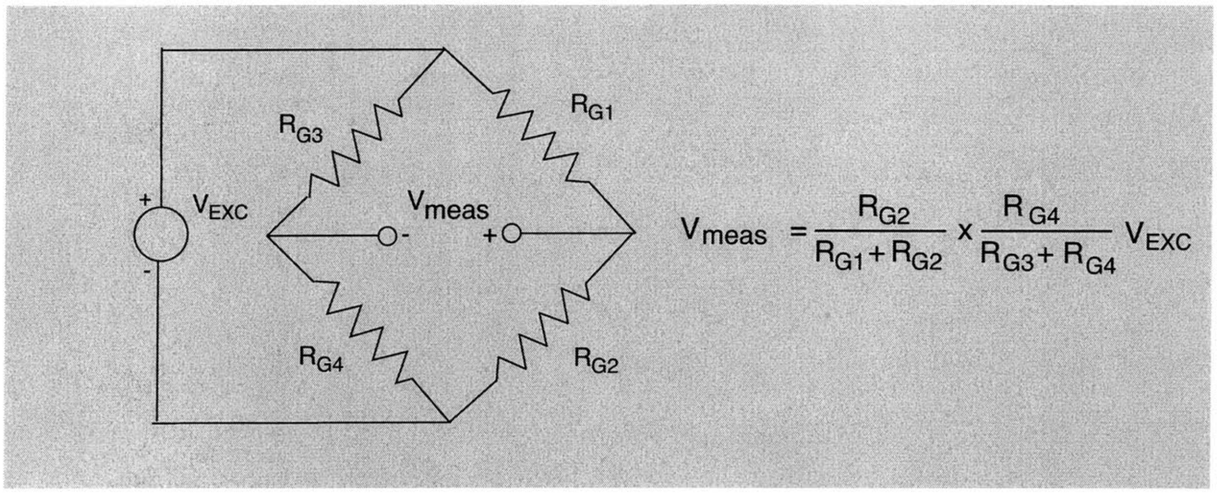


Figure 4.8: The three load cells inside each MICR0-G sensor. V1 through V6 are the six voltages that are measured and correspond to forces and moments.

The Wheatstone bridge circuit allows precise strain measurements. It consists of four resistive elements with a voltage excitation supply applied to the ends of the bridge. Strain gages can occupy one, two, or four arms of the bridge, with any remaining positions filled with fixed resistors. The MICR0-G design implements a full strain gage bridge consisting of four strain gages R_{G1}, R_{G2}, R_{G3} and R_{G4} as shown on Figure 4.9.

With an excitation voltage, V_{EXC}, powering the bridge, the measurement system measures the voltage V_{meas} across the bridge corresponding to a signal V_i (i = 1 to 6) in Figure 4.8. In the unstrained state, when the ratio of R_{G1} to R_{G2} equals the ratio of R_{G3} to R_{G4}, the measured voltage V_{meas} is 0 V. As strain is applied to the gages, their resistance values change, causing a change in the voltage V_{meas}. Full strain gage bridges are in general rare due to the higher material and labor cost involved. Instead



R_{G1}
R_{G3}
V_{EXC}
V_{meas}
R_{G4}
R_{G2}
$V_{meas} = \frac{R_{G2}}{R_{G1}+R_{G2}} \times \frac{R_{G4}}{R_{G3}+R_{G4}} V_{EXC}$

Figure 4.9: This figure shows the electrical configuration of a full bridge strain gage. R_{G1},R_{G2}, R_{G3} and R_{G4} are four strain elements.

half or quarter bridges are used where simple resistors are utilized to complete the bridge. By using full bridges, the sensitivity is nearly doubled and most, if not all, temperature effects are compensated.

The gages of the sensors have resistances of 350 Ω (value of the strain elements R_{Gi} when no load is applied) and a gage factor[1] of 2.155.

The Sensor Electronics Unit

The MICR0-G experiment aboard the KC-135 aircraft acquires data on two sensors (a hand hold and a foot restraint or two foot restraints). Each sensor acquiring data is connected to a Sensor Electronics Unit (SEU) housed in a Lexan box and incorporates the following electronics:

- Central Processing Unit (CPU)
- PCMCIA module
- PCMCIA LAN wireless card
- A/D module with a built-in D/A converter
- Six channels signal conditioning board
- Shielded cables

1. The gage factor is the ratio of the relative resistance change in a strain gage to the unit strain causing the resistance change. A unit strain is defined as the ratio of the change in length to its initial length.

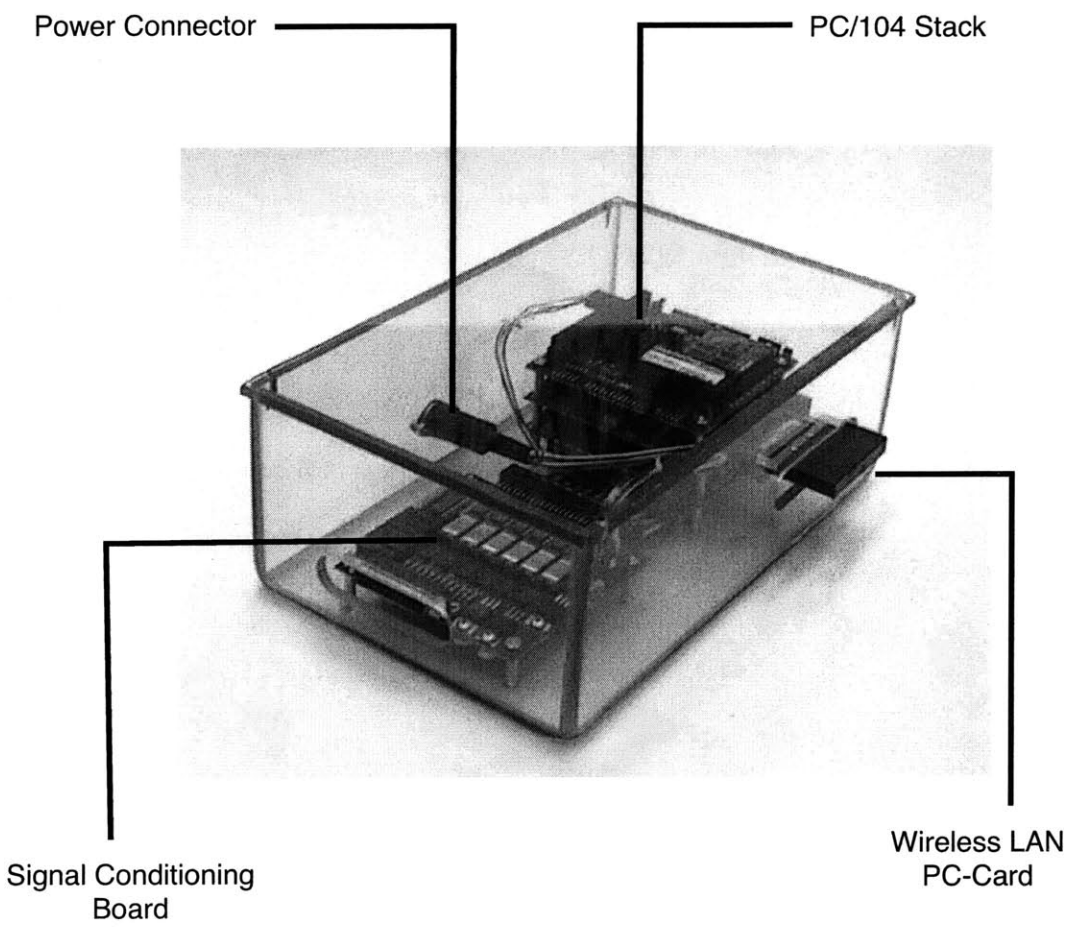


Figure 4.10: The Sensor Electronics Unit (SEU) showing the new MICR0-G electronics. The load sensor is connected to the DB25 connector on the signal conditioning board and the power supply is connected to the power connector on the rear side of the box.

The SEU housing is made of Lexan sheets of 0.63 cm (1/4 inch) thickness. The box measures 26.7 cm x 16.1 cm x 10.2 cm and has rounded corners to enhance safe operations. All the sides are glued except the top, which is screwed into the four sides of the box. One side of the box has a hole of 7.6 cm x 1.8 cm to accommodate the antenna of the wireless card to facilitate communication with the laptop computer. The opposite side has a circular port of diameter 1.5 cm for the female DB5 connector of the power adapter, which provides power to the electronics. The power adapter is connected to two power connectors inside the box, one on the CPU and the second one on the signal conditioning board. The A/D converter and the PC-card drive power from the CPU. A standard 50-pin ribbon cable connects the signal conditioning board to the A/D converter. Finally, a rectangular hole of 3.6 cm x 1.3 cm is cut on one of the sides to connect the DB25 connector of the load sensor to the signal conditioning board.

The CPU, the A/D converter and the PCMCIA module are PC/104 boards. The PC/104 form factor, detailed in Chapter 3, offers a full hardware and software PC-compatible architecture, but in the form of compact (9.0 cm x 9.6 cm), self-stacking modules. Therefore these three boards are stacked on top of each other using 1.5 cm spacers. The bottom spacer is used to screw the stack on the bottom of the Lexan box. Similar spacers and mounting technique are used to fix the signal conditioning board inside the Lexan box. Figure 4.10 shows the SEU with all the electronics and cables. The following paragraphs detail the MICR0-G SEU's electronics.

Signal Conditioning Board

The data acquisition system consists of a sensor, a signal conditioning board and an A/D converter. The signal conditioning board provides the excitation voltage to the sensor's strain gages and represents the first layer of signal processing. The characteristics and performance of the signal conditioning board are critical design parameters for the overall capability and reliability of the acquisition system. The major specifications required for the signal conditioning board are summarized herein:

- Input type: full bridge strain gages
- Number of channels: 6
- Excitation voltage: between 4 and 10 V
- Input gain: a minimum of 10 values between 50 and 1000 V/V
- Filtering: third order low-pass filter with an adjustable corner frequency around 125 Hz

Many of the COTS signal conditioning modules exceed our system's six channel specification and are excessive in size and mass. Since small size is a critical design parameter for spaceflight systems, we decided to build an embedded customized signal conditioning board to meet our specific requirements. Since the remaining modules of the SEU are PC/104 boards, the signal conditioning board was designed with respect to the PC/104 form factor to facilitate final assembly. The customized signal conditioning board was designed and built at Payload Systems Inc. and is six times smaller than the DBK43A signal conditioning module from Iotech Inc. (Cleveland, OH) proposed in an early design of the advanced MICR0-G load sensors [Amir, 1998]. A photograph of the customized signal conditioning board is shown in Figure 4.11. Technical drawings and detailed specifications of the board are provided in Appendix D.

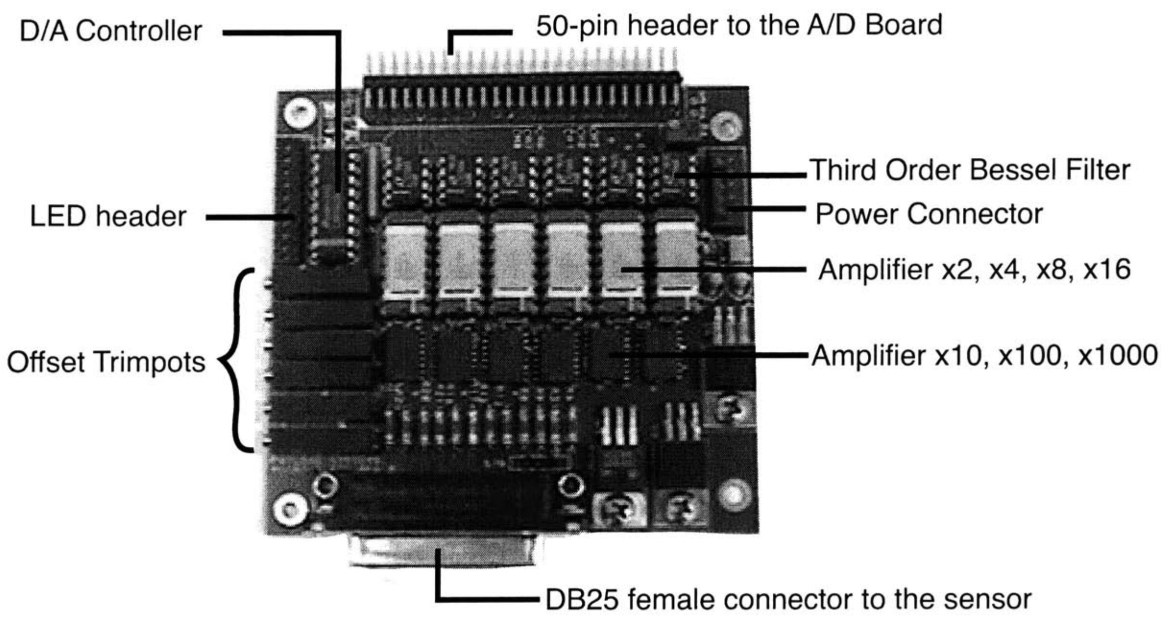

D/A Controller
50-pin header to the A/D Board
LED header
Third Order Bessel Filter
Power Connector
Amplifier x2, x4, x8, x16
Offset Trimpots
Amplifier x10, x100, x1000
DB25 female connector to the sensor

Figure 4.11: The MICR0-G customized signal conditioning board designed and built at Payload Systems Inc.

Analog to Digital Converter

The main specifications required for the A/D converter are the following:

- Number of channels: 6 differentials
- Sampling rate: 250 Hz
- Sampling resolution: 16-bit
- Built-in D/A with one analog output

For the A/D converter a commercial board was selected. Manufactured by Diamond Systems Inc., (Newark, CA), the Diamond-MM-16 shown on Figure 4.12 is an analog input/output (I/O) PC/104 data acquisition module. It offers 16 analog inputs with 16-bit resolution and programmable gain, 4 analog outputs with a 12-bit resolution D/A converter, reference voltage output for load cells, on-board pacer clock for A/D conversion timing, and 16 lines of digital I/O. The Diamond-MM-16 can sample up to eight differential channels at a rate of 100 kHz. Since the load sensor requires only six differential channels, the maximum sampling rate possible with this A/D converter is 12.5 kHz, which is far above the desired rate of 250 Hz. A sampling rate of 250 Hz has been chosen since it is the smallest sampling rate required to analyze accurately human interaction and motion data.

The A/D board samples the analog signal coming from the signal conditioning board. As explained at the end of Chapter 3, data is relevant only after multiplied by the calibration matrix. Using the D/A converter of the Diamond-MM-16 module, the calibrated data is then transformed into an analog signal, which feeds a LED device to provide real-time feedback. The feedback capability has been developed at MIT and works on the lab prototype but has not been integrated in the flight MICR0-G KC-135 flight hardware.

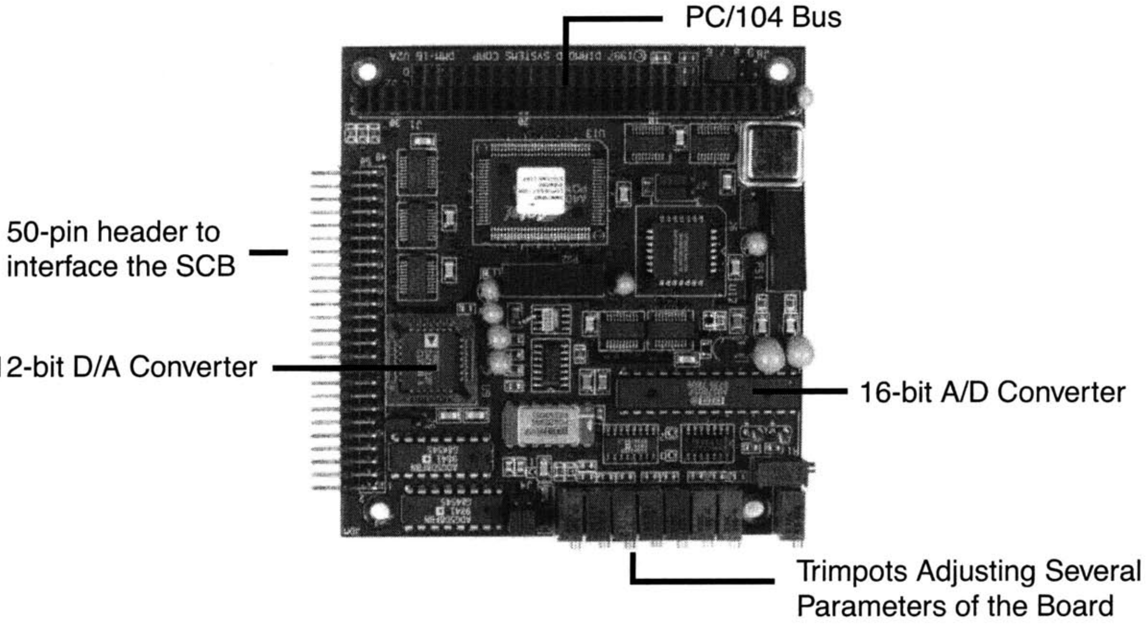

PC/104 Bus
50-pin header to interface the SCB
12-bit D/A Converter
16-bit A/D Converter
Trimpots Adjusting Several Parameters of the Board

Figure 4.12: The Diamond-MM-16 A/D converter utilized in the MICR0-G SEU.

Central Processing Unit

At the heart of the SEU is the processing unit for which the PC/104 CoreModule/4GV from Ampro (San Jose, CA) has been chosen. The CoreModule/4GV packs 486 processing power together with all the standard embedded PC product features, including CPU, memory subsystems, communication ports and disk interfaces. The CPU card includes a CRT (Cathodic Ray Tube) and flat panel video interface, a memory module of 32MB and a DiskOnChip bootable flash drive of 72 MB. Additional on board functions include two serial ports, a parallel printer port, a watchdog timer and a real time clock. The board supports a floppy drive and up to two IDE (Integrated Drive Electronics) disk drives. Figure 4.13 shows a detailed view of the CoreModule/4GV.

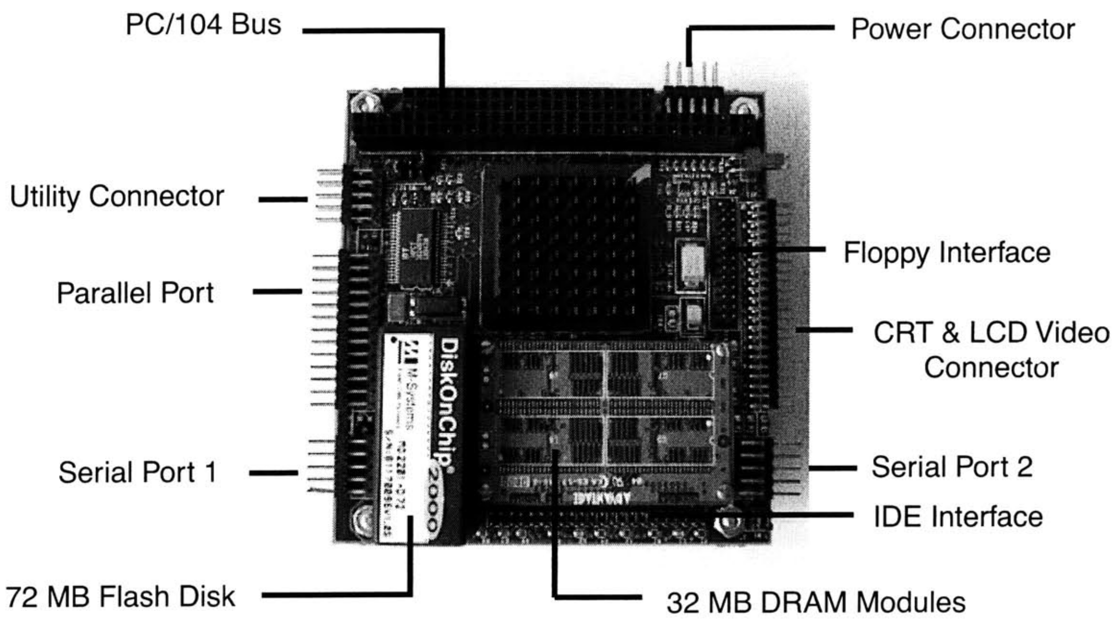


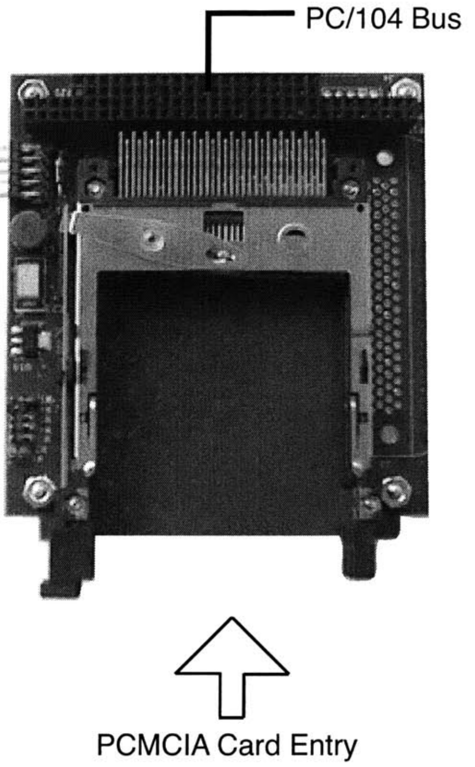
Figure 4.13: The Core Module 4GV from Ampro and details of its main characteristics.

Wireless Interface

The MiniModule/PCC from Ampro has been chosen to provide interface to the wireless PC-Card (PCMCIA). This PC/104 module, shown on Figure 4.14, provides two PCMCIA slots accepting any type of PC-Cards (Type I, Type II, and Type III). In the current design for the advanced load sensors, a single slot is used to insert a wireless network Type II PC-Card. The remaining slot may be used for a Type III space qualified hard disk as an optional storage capability. The RoamAbout 802.11 wireless network PC-Card from Cabletron Systems, Inc. (Rochester, NH) was specified for the MICR0-G prototype hardware. The RoamAbout card offers a wireless data LAN network capability utilizing the full 2.4 GHz band and is compatible with existing Linux drivers.

Power Supply

The W4232 AC/DC power adapter from Elpac Electronics Inc. (Irvine, CA), shown on Figure 4.15, is used to provide power to one SEU unit. Since two SEUs are necessary for the MICR0-G experiment, we have two power adapters total. The power adapter provides the triple output required by the electronics (+5V, +12V and -12V). The Elpac adapter incorporates an EMI filter and is fully encapsulated in a plastic case with round corners. Each adapter measures 14.6 cm x 8.3 cm x 3.8 cm. Both are secured aboard the KC-135 aircraft using velcro.



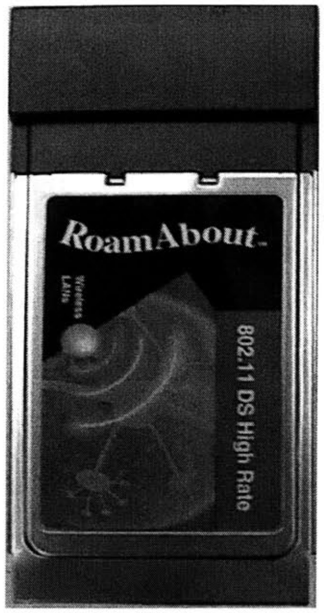


Figure 4.14: The MiniModule/PCC from Ampro (San Jose, CA) and the wireless PC-Card RoamAbout 802.11 from Cabletron (Rochester, NH) used during the MICR0-G experiment aboard the KC-135 flight in January 2001.

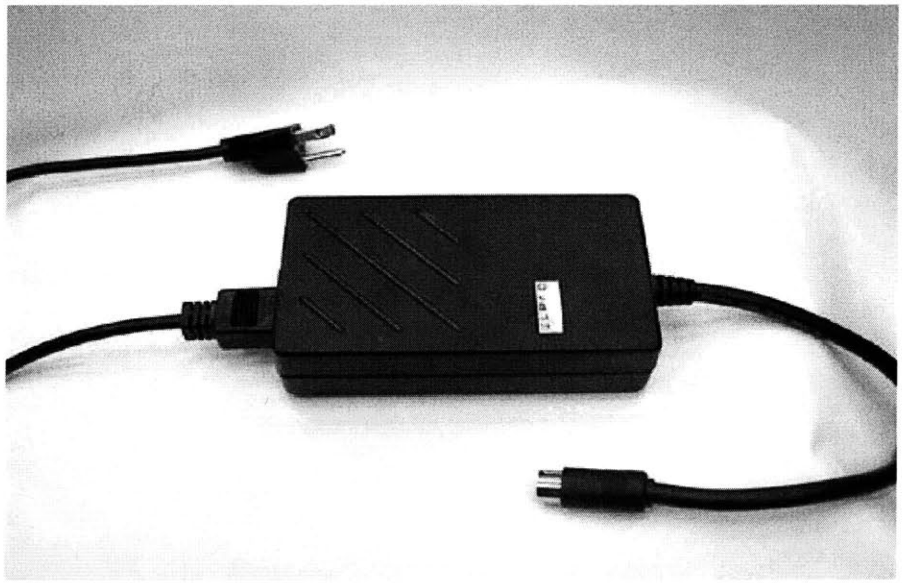


Figure 4.15: Power adapter connected to the Electronics Unit.

4.4.2 The ELITE System

The kinematic measurement component of the MICR0-G system consists of the real-time opto-electronic motion analyzer ELITE-S. The ELITE system (see Figure 4.4) is designed for three dimensional analysis of human motions within a suitably calibrated working volume without making contact with the subject or constraining movements in any way. The ELITE-S hardware is composed of the following items:

- Main Electronics Box containing the dedicated electronics for real-time processing of images
- 2 CCD TV cameras equipped with infrared flashes
- 1 calibration form
- 1 Processing Unit consisting of an IBM compatible Personal Computer, equipped with a Digital Data I/O card and the dedicated software for acquisition and storage of the kinematics data
- 1 Processing Unit Monitor and VGA cable
- 2 sets of cables (video cable and power & synchronization cable) for connecting the TVC to the Electronics Box
- 1 Digital Data Cable connecting the Electronics Box to the Processing Unit
- 1 TVC monitor
- Additional equipment (markers to affix to subjects' joints, bioadhesive stickers)

Main Electronics Box

The Main Electronics Box contains printed circuit boards coated with silicone for structural integrity and against offgassing. The coated electronic boards are mounted in a chassis and connected to the external case of the box via cables. The chassis and electronics are attached inside a vibration absorbing frame, which is mounted inside a rigid case. Grills on the upper and lower surfaces of the box facilitate ventilation. Connectors are placed in a dedicated area on the upper surface as shown in Figure 4.16. The electronics box is connected to the main power supply via cables of 1 m long and provides power to the TV cameras and the TVC monitor. Power consumption for this set of electronics is estimated between 160 and 200 W. Figure 4.16 shows a drawing of the main electronics box and specifies external dimensions in cm.

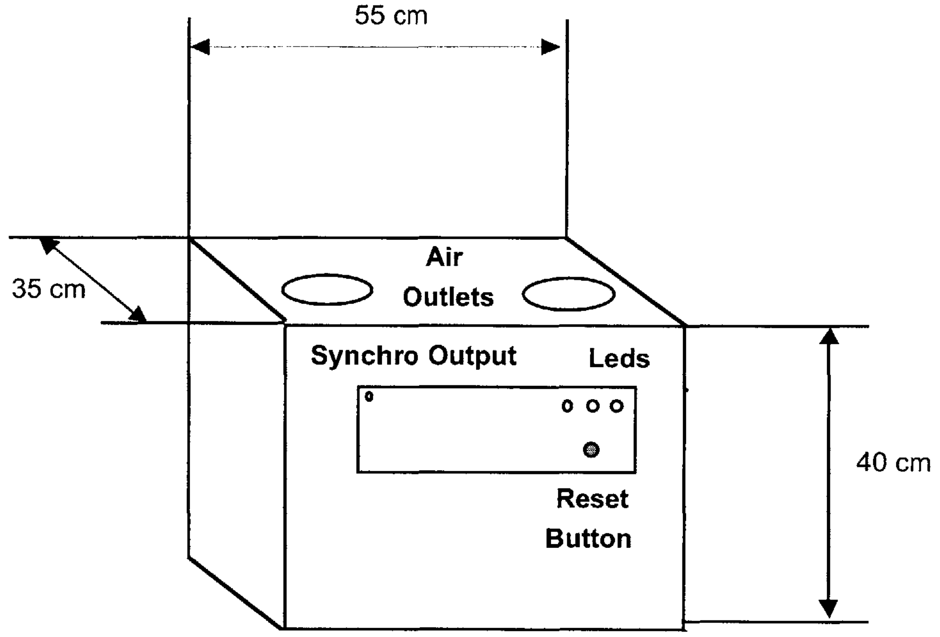


Figure 4.16: The ELITE-S main electronic box.

TV camera

The TV cameras are commercial electronics (Phillips CCD monochrome TV cameras) embedded inside a metal and plastic chassis. The electronics and chassis are mounted inside a pressed anodized aluminum case. The cameras are equipped with lenses of 8.5 mm or 6 mm (focal length) optics. Around the lenses is a ring of Velcro to interface with a complementary ring of Velcro on the flash unit. On one side of the aluminum case is a screw threaded socket allowing to fix the camera. Aboard the KC-135 aircraft, hardware can only be mounted on the floor. A mounting pole with a top plate is fixed to the aircraft floor. The camera is then screwed on the mounting plate using the camera's threaded socket. Figure 4.17 shows a drawing of the TV camera.

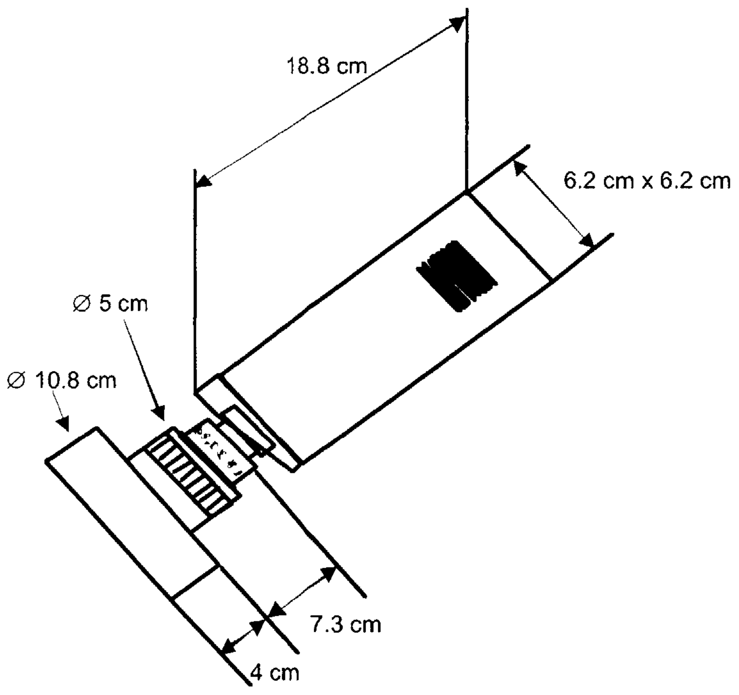


Figure 4.17: An ELITE TV camera unit.

Flash Unit

The flash unit is an annular array of 120 infrared LEDs mounted on a printed circuit board and potted with silicone. The electronics is mounted inside a plastic case coated with double aluminum Kapton to improve the thermal emissivity. A cable connects the flash unit to the TVC.

Calibration Form

The calibration form, shown in Figure 4.18, is a geodesic form of struts and joints deployed by the operator after system installation. Calibration consists of a one-shot acquisition of 22 markers, which comprise the calibration form. When deployed the form occupies a volume of about 1 m x 1 m x 1 m. When stowed, it occupies a volume of 1 m x 15 cm x 10 cm. The form has been used on the ground as well as during the flight to calibrate the system as accurately as possible.

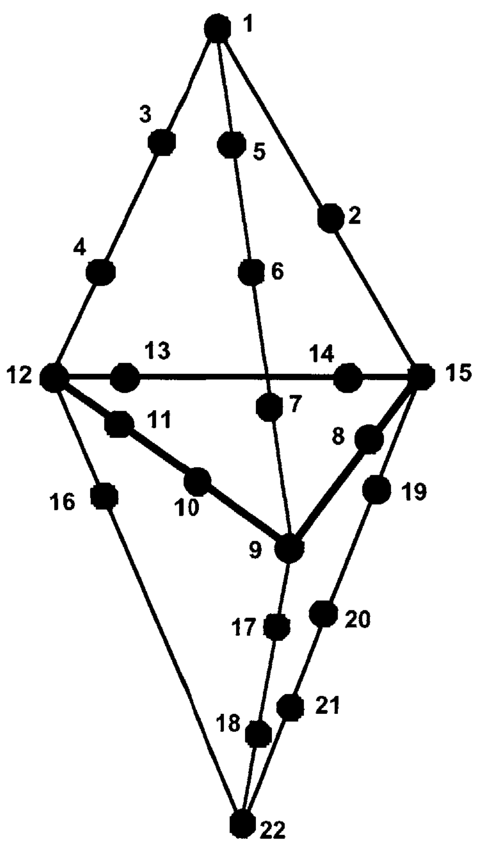


Figure 4.18: The ELITE-S calibration form.

Processing Unit

The Processing Unit is a standard IBM compatible PC equipped with a Digital Data I/O card (Keithley PDMA-32) and a dedicated software under the MS-DOS operating system. Dimensions of the Processing Unit are around 40 cm x 16 cm x 40 cm (width x height x depth). It requires power according to 100-240 V (1.2 A) at 50-60 Hz.

Processing Unit Monitor

The Processing Unit Monitor is a standard 14" PC monitor connected to the Processing Unit by means of a VGA data cable. It requires power according to 100-240 V (1.2 A) at 50-60 Hz. A Plexiglas protection is used aboard the KC-135 aircraft in order to prevent accidental monitor glass breaks. The protection is fixed to the monitor by means of complementary Velcro stripes.

TVC Cabling

For each TVC, two cables exit from the camera and run in a single sheath from the camera to the Electronics Box where they separate and attach to the Electronics Box by two connectors.

Data Cable

The Data Cable connects the Electronics Box to the Processing Unit via two dedicated interfaces. On the PC side, it interfaces with the Digital Data I/O card which is inserted in the ISA bus slot of the PC. The data cable length of 1 m implies that the Electronics Box and the Processing Unit have to be installed close to each other.

TVC Monitor

The TVC monitor allows the operator to see an infrared image of any of the two TVC views. A manual selector is placed on the TVC monitor and allows to switch from one TVC to the other. A dedicated cable (around 2 m long) connects the TVC monitor to the Electronics Box.

Additional Equipment

The additional equipment includes the markers and the bioadhesive stickers, which are needed to attach the markers to the selected anatomical landmarks of the test-subject. Markers have a hemispherical form made of plastic with a coating 3M Scotchlite vacuum pressed onto the surface. A set of 25 markers (10 mm diameter) and a set of 25 markers (15 mm diameter) are part of the additional equipment of the ELITE system. Markers are attached directly to the subject's skin or clothes by mean of double sided medical adhesive disks.

4.5 In-flight Operations

During the parabolic flights, the subject having reflective markers fixed on his clothes at the body joints is instructed to perform different motions using the combination of two sensors: either a handhold and a foot restraint (HH, FR) or two foot restraints (FR, FR). The description of the IVA motion tasks is provided in Table 4.1.

Table 4.1: Characteristic astronaut motion using mobility aids sensors.

Characteristic Motion	Description of Astronaut Motion	Sensors Used
Landing	Flying across the cabin and landing on a sensor.	HH, FR
Push-off	Pushing off a sensor and floating away.	HH, FR
Flexion/Extension	Flexing/Extending a limb while using a sensor.	HH, FR
Single/Double Support	Using one/two limbs for support.	HH, FR
Twisting	Twisting body motion.	HH, FR
Vertical/Horizontal Orienting	Vertical/Horizontal re-orienting usually for posture control.	HH, FR
Axial Movements	30° Upper trunk forward/backward bending	FR, FR
Lateral Leg Raising	45° leg abduction	FR, FR

During microgravity sessions, the forces and moments applied by the subject on the sensors are acquired and the position of the reflective markers on his body are analyzed by the TV cameras' system. The subject is required to stay in a defined working volume of 1 m x 2 m x 2 m during the experiment to stay in the field of view of the cameras. Normal flights last about two hours and consist of 40 parabolic maneuvers. The subject performs the experiment only during microgravity periods lasting between 20 and 25 seconds per parabola. Each day of flight operations aboard the KC-135 followed a similar but slightly different format. Table 4.2 summarizes the crew motions specified for each parabola on flight day 2 (FD 2).

At least three out of four team members have to be healthy to conduct the experiment, thus allowing for expected occurrences of motion sickness in microgravity. One team member operates the kinetic system and another team member operates the kinematic video system. The third person is the subject and is instructed to perform the prescribed IVA tasks interacting with the sensors according to Table 4.2. The two operators, one test director and one system operator, as well as any other passenger, are allowed to stand between the two cameras and the working volume as long as they do not interfere with the field of view of the cameras.

Table 4.2: Description of specified IVA motions for the subject during each parabola of FD 2.

Parabola Number	Sensors used	Subject Activity
1-2		Weightlessness accommodation
3		Sensor/Camera identification
4	FR, FR	Interaction with the foot restraints
5	HH, FR	Interaction with the handhold
6-7	FR	Landing/Push-off motions on one foot restraint
8-9	HH	Landing/Push-off motions on the handhold
10-11	FR, FR	Flexion/Extension on both foot restraints
12-13	HH	Flexion/Extension on the handhold
14-16	FR	Twisting motions
17-18	FR, FR	Bending Eyes Open Hands Backward
19-20	FR, FR	Bending Eyes Open Hands Forward
21-22	FR, FR	Bending Eyes Closed Hands Backward
23-24	FR, FR	Bending Eyes Closed Hands Forward
25-28	FR, FR	Pointing
29-32	FR	Leg Raising Eyes Open
33-36	FR	Leg Raising Eyes Closed
37-40	HH, FR, FR	Repeat any missed motions or second subject performs motions

Figure 4.19 shows the three postures characterizing a bending motion. The photos have been taken in microgravity aboard the KC-135 aircraft on January 25, 2001 (FD 3). Reflective markers are affixed to the subject at the body joints and secure with additional tape. The subject performs the bending motions using two foot restraints that both record the applied forces and moments. Figure 4.20 and Figure 4.21 show early results on the data acquired during a bending motion on FD 3. Two events can be clearly identified. The signals recorded are the voltages at the input of the A/D converter. Since the gain has not been modified from kinetic data acquisition in 1 g to data acquisition in microgravity, the amplitude of the signals provided in Figure 4.20 and Figure 4.21 are relatively small compared to the full input scale of the A/D set to ±10 V. Nevertheless, the accuracy of the acquisition system is good enough to distinct events from noise signals. This data will be processed during post-flight analysis to provide force and moment data respectively in Newtons and Newton-meters. The kinematic data will be analyzed at the Politecnico di Milano University and combined later with the results of the kinetic analysis in order to characterize nominal loads for each prescribed motion performed in microgravity.

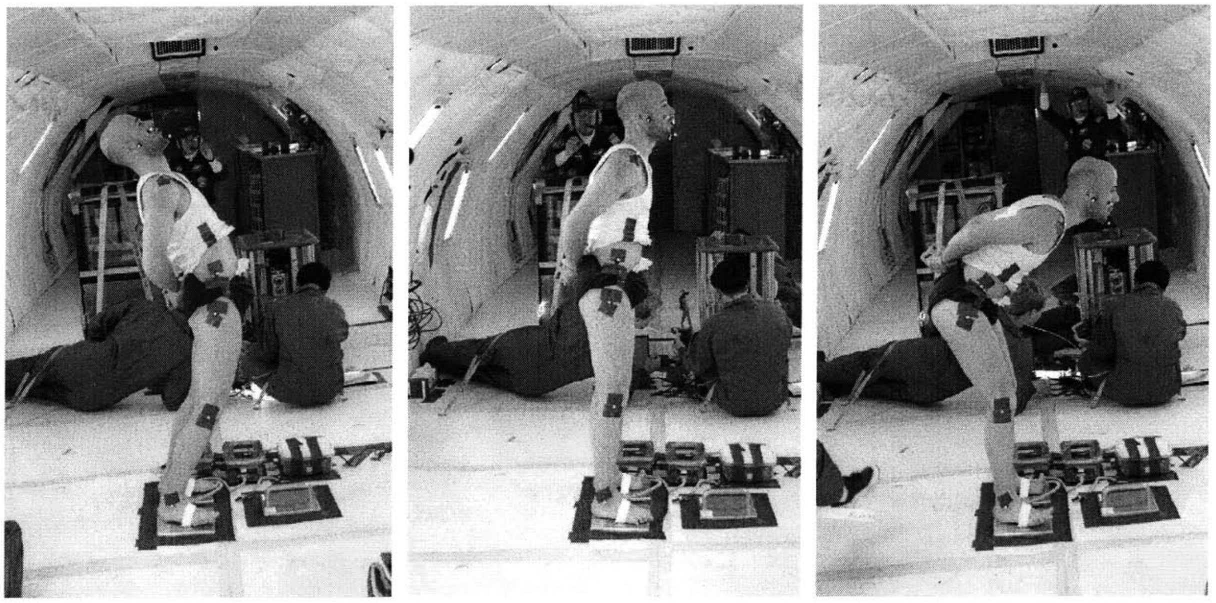


Figure 4.19: This sequence of images shows a bending motion in microgravity aboard the KC-135 aircraft on January 25, 2001 [NASA, 2001].

4.6 Recommendations for Future Development

This section is divided into four parts. Section 4.6.1 provides information to assist with the analysis of the data acquired during parabolic flights. The two following sections deal with the hardware upgrades and further software development of the ground-based prototype system. The last section summarizes the characteristics of the MICR0-G sensor prototype and presents the future plans for space-qualifying and flying the advanced load sensor system aboard the ISS in conjunction with the Italian space-qualified motion analyzer system ELITE-S2.

4.6.1 Data Analysis

Kinetic data was recorded during four flight experiments aboard the KC-135 aircraft in January 2001. The ELITE-S system recorded the corresponding kinematic data. The flight procedure describing subject's motions during each parabola has been discussed in the previous section. The kinematic data will be examined at the Politecnico di Milano University. The flight kinetic data will be analyzed at MIT and is further discussed below.

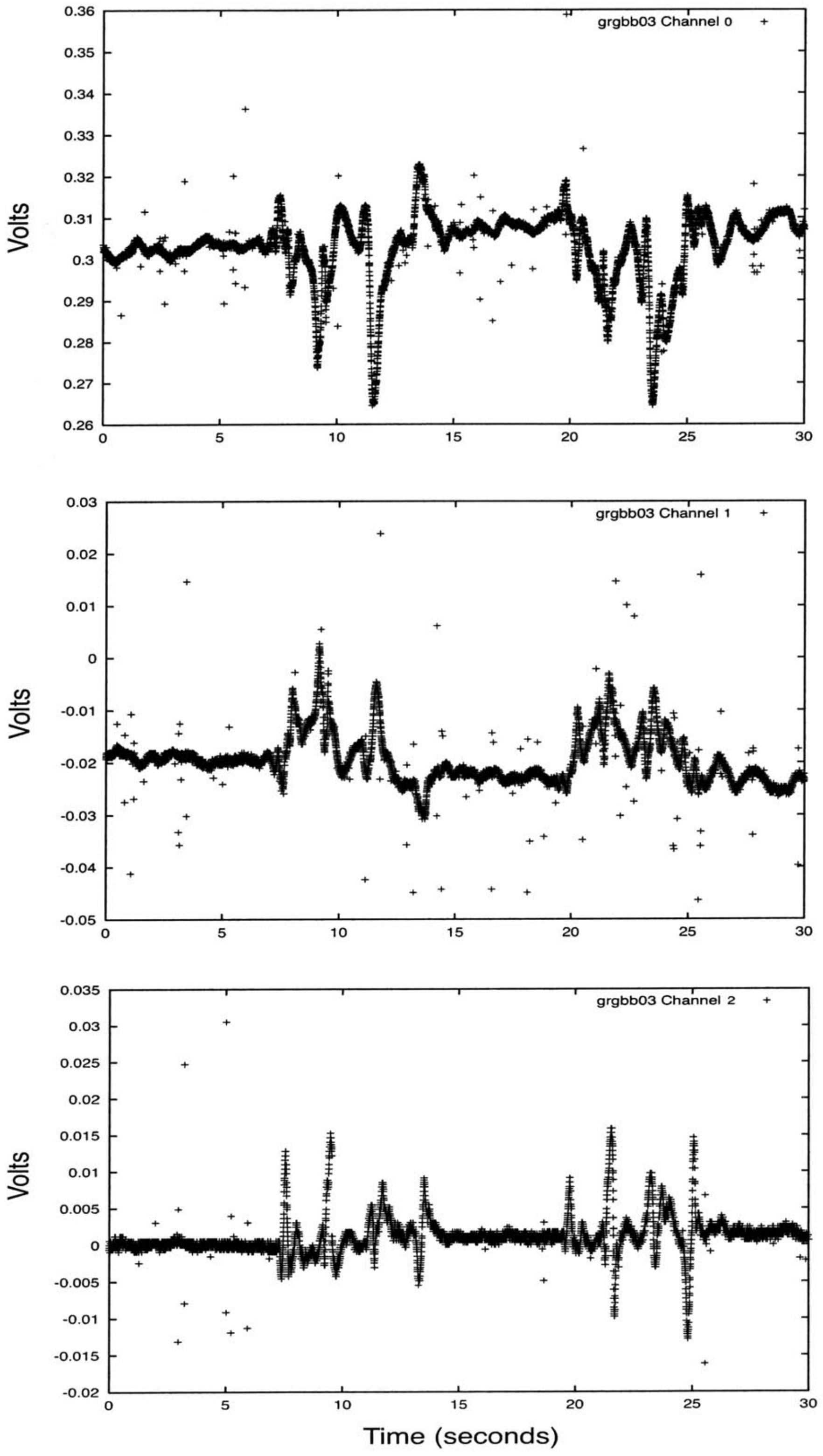


Figure 4.20: The plots show the channel 0 through channel 2 signals (in Volts) recorded on FD3 aboard the KC-135 aircraft, while the subject is performing a bending motion.

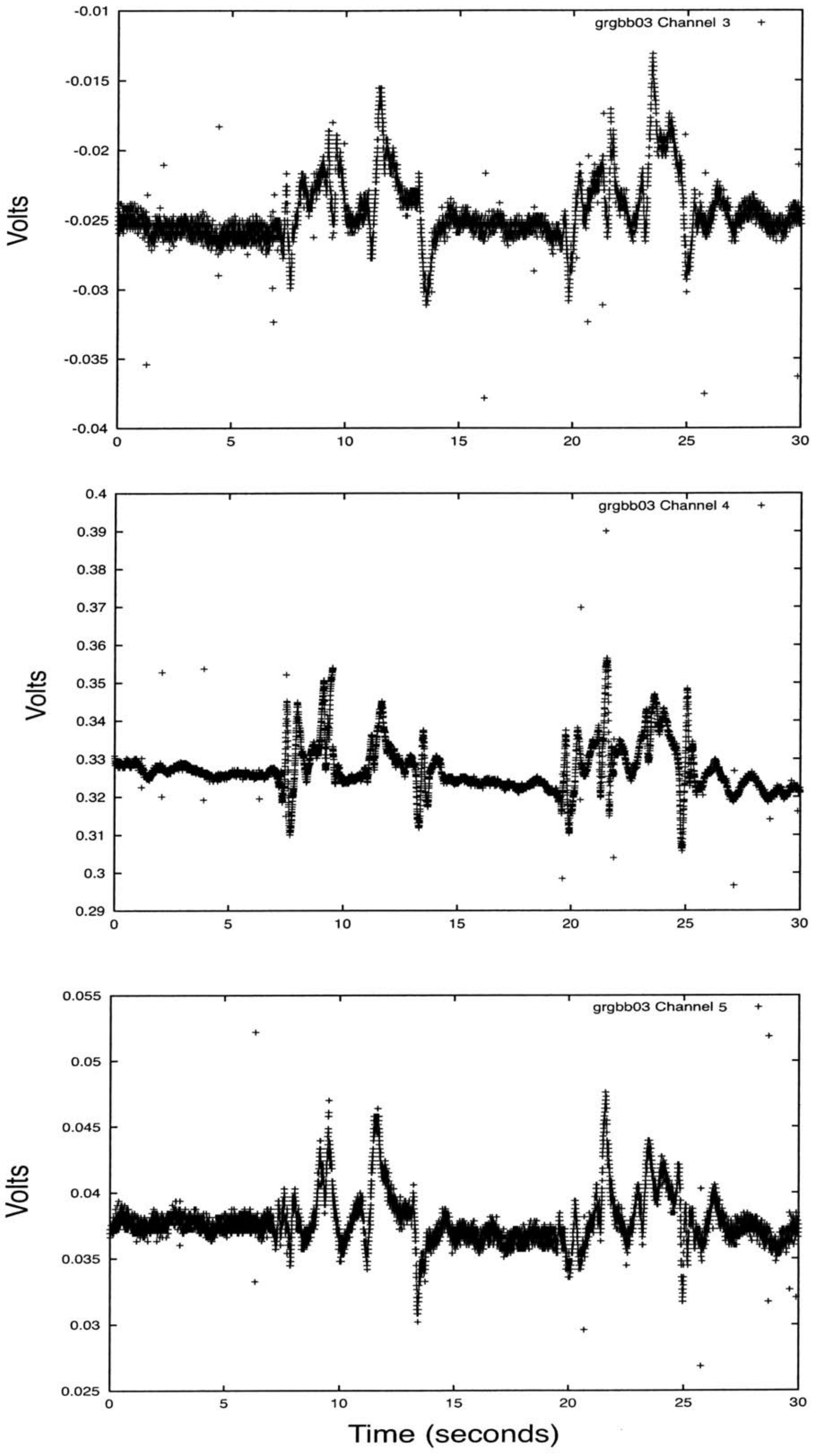
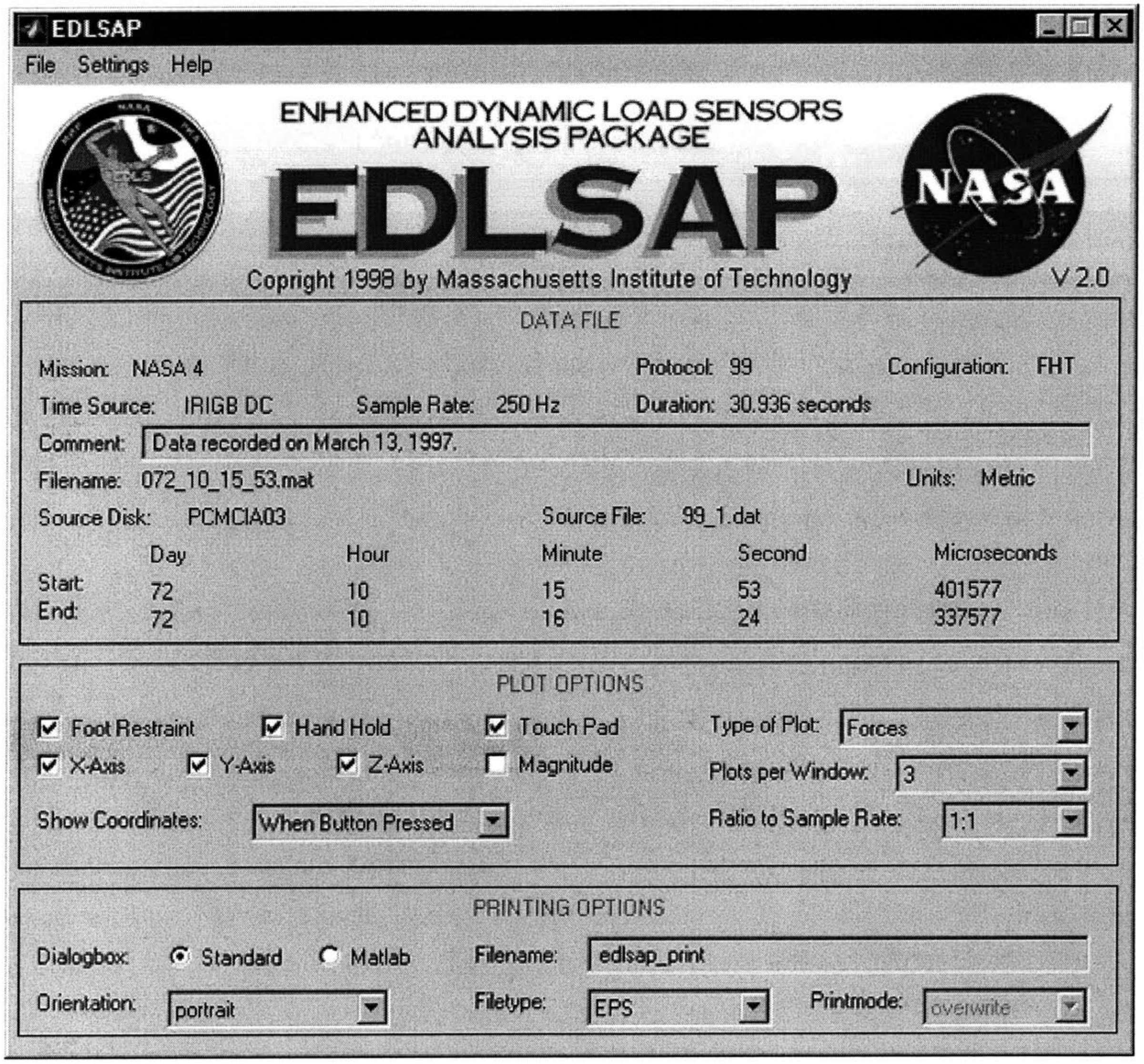


Figure 4.21: The plots show the channel 3 through channel 5 signals (in Volts) recorded on FD3 aboard the KC-135 aircraft, while the subject is performing a bending motion.

The MICR0-G advanced load system samples the six analog signals at the output of the sensor at a rate of 250 Hz and stores the corresponding data. During phase one of the data analysis, flight data will be unbiased, so that when no load is applied the average force/torque signal is zero. Then flight data will be multiplied by a calibration matrix to convert the measured voltage signals to actual applied loads. The calibration matrix for each flight sensor is obtained through the calibration procedure described in Section 3.6.

During phase two of the data analysis, the microgravity IVA motion events will be detected. Establishing a threshold (typically set at 1 N), an event is detected when the signal exceeds this threshold and then falls below it. Then the characteristics of each event will be determined. In particular, the force and moment magnitudes, the average power and the power spectrum density will be established. A software named *Enhanced Dynamic Load Sensors Analysis Package* (EDLSAP) has been developed at MIT to simplify the analysis of the space data from the Mir EDLS experiment. EDLSAP is a collection of Matlab 5 routines that were designed to run without changes under Windows, Mac OS, and Unix. The two main tasks of EDLSAP are the processing of the EDLS raw data and the analysis of the processed data. The data analysis capability of EDLSAP may be used to analyze the MICR0-G data acquired during parabolic flights. In particular, data segments such as noise, can be removed from the data by cutting unwanted portions, the signal can be unbiased, so that when no load is applied the average signal is zero. The data can also be scaled (i.e., multiplied with an arbitrary multiplication factor) and filtered with a low-pass digital elliptic filter (with a variable number of poles and corner frequency). Once a specific event has been located, EDLSAP can visualize the force vector through a 3-D animation. Finally, EDLSAP incorporates several signal processing functions to analyze the force and moment data. In particular, the average power, the root-mean-square value, the auto-correlation function, and the power spectrum density from any given signal can be computed and plot. The EDLSAP main window is shown in Figure 4.22 [Amir, 1998].

During the third round of data analysis, the force/moment data will be correlated with the ELITE system's kinematic database. Figure 4.23 and Figure 4.24 show data analysis results of the ELITE experiment aboard the Mir space station. Figure 4.23 shows the free floating posture exhibited by an astronaut at Flight Day (FD) 69 and Figure 4.24 compares the leg raising motor strategy in 1-g and in microgravity [ASI, 1998]. Combining the results of the MICR0-G kinetic and kinematic analysis will allow to compute the average load for each type of motion performed during the MICR0-G experiment



Copright 1998 by Massachusetts Institute of Technology

DATA FILE

Mission: NASA 4   Protocol: 99   Configuration: FHT
Time Source: IRIGB DC   Sample Rate: 250 Hz   Duration: 30.936 seconds
Comment: Data recorded on March 13, 1997.
Filename: 072_10_15_53.mat   Units: Metric
Source Disk: PCMCIA03   Source File: 99_1.dat

	Day	Hour	Minute	Second	Microseconds
Start:	72	10	15	53	401577
End:	72	10	16	24	337577

PLOT OPTIONS

[✓] Foot Restraint   [✓] Hand Hold   [✓] Touch Pad   Type of Plot: Forces
[✓] X-Axis   [✓] Y-Axis   [✓] Z-Axis   [ ] Magnitude   Plots per Window: 3
Show Coordinates: When Button Pressed   Ratio to Sample Rate: 1:1

PRINTING OPTIONS

Dialogbox: [✓ Standard] Matlab   Filename: edlsap_print
Orientation: portrait   Filetype: EPS   Printmode: overwrite

Figure 4.22: The Enhanced Dynamic Load Sensors Analysis Package (EDLSAP) main window [Amir, 1998].

aboard the KC-135 aircraft. A comparison with the results of space experiments, in particular the Mir EDLS experiment results, will assess the differences between astronaut motions and loads during long duration flight, and subject motions and induced-disturbances in microgravity during short duration parabolic flight.

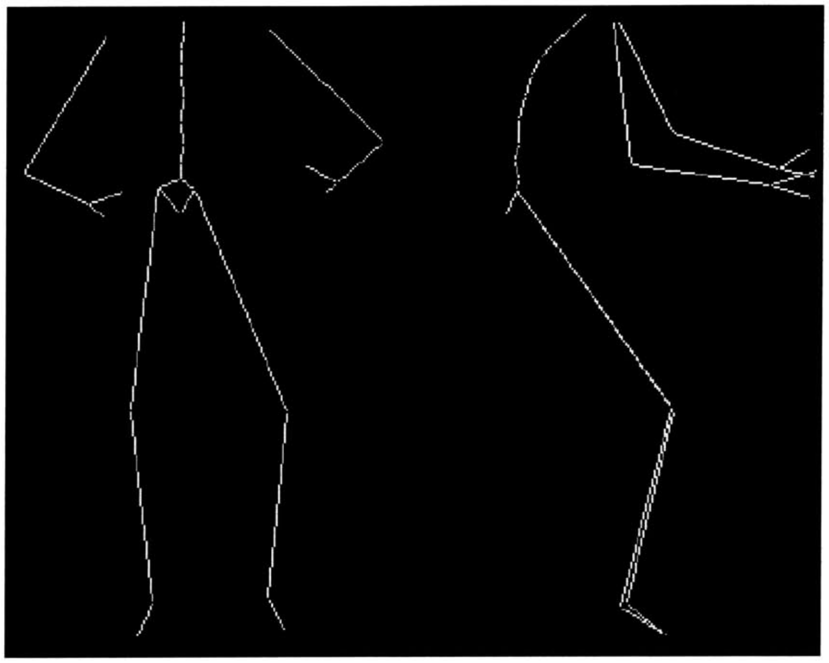


Figure 4.23: Stick diagram representation of the free floating posture at Flight Day 69 on Mir [ASI, 1998].

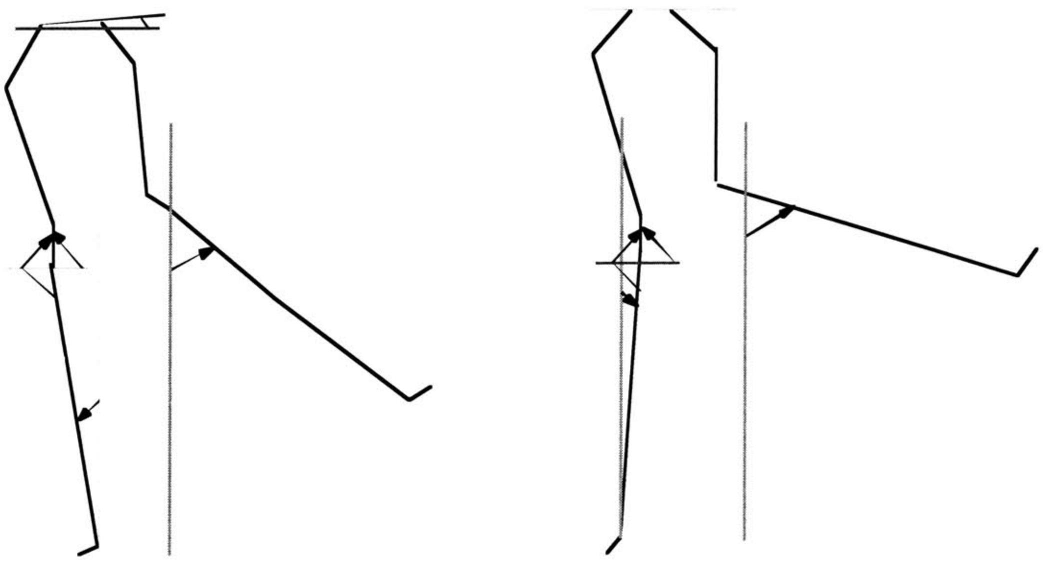


Figure 4.24: . The leg raising motor strategy in 1-g (left) and in microgravity (right) [ASI, 1998].

4.6.2 Hardware Modifications

Chapter 4 described the ground-based prototype of the advanced load sensor system. Although the prototype's development was based on the conceptual space design described in Chapter 3, it is slightly different from the system envisioned for the ISS. The following paragraphs summarize the differences between the conceptual design and the existing prototype system, and describes future hardware modifications and system improvements:

- Aboard the KC-135 aircraft, each sensor was connected via a cable to a SEU. In the final design, the SEU will be attached underneath the sensor, creating a fully integrated independent force/ torque unit. Therefore, the astronauts will be able to easily move the sensors around any ISS module in order to locate them at the most relevant places (i.e. high traffic areas).
- Since the SEU will be embedded underneath the sensor, the electronics will be integrated in order to minimize the total height with respect to the sensor's footprint for width and depth (24.1 cm by 24.1 cm). The first relevant modification will be the use of embedded power supply components instead of an external power supply. The power supply components should be encapsulated, in order to limit electromagnetic interferences (EMI) and noise signal disturbances. In addition, the PC/104 boards will be placed side by side instead of being stacked in order to minimize the SEU's height. This can be achieved using the 6-DE-104/16 board from Douglas Electronics Inc. (San Leandro, CA) or by designing a customized motherboard with a connector for each piece of electronics. Ideally only one layer of electronics will be attached underneath the load sensor, leading to a total height of only 6 cm including the load sensor.
- When the final ground-based prototype is assembled and satisfies the technical and operational requirements described in this thesis, the MICR0-G system will be used as the basis of a proposal to NASA for support to build spaceflight hardware for the ISS. The proposal will also include the integration of the advanced load sensors with the ELITE-S2 space-qualified system and the capability of the two systems to communicate and synchronize data. The space-qualification of the advanced load sensors will be simplified and the cost reduced by using previously qualified spaceflight hardware. Chapter 3 describes the modification made by NASA on PC/104 boards for space-qualification. In addition, the Proxim RangeLAN2 wireless LAN PC-card and the IBM ThinkPad laptop computer have already been flown in other space systems [Cowing, 2000] and are recommended herein.

4.6.3 Software Development

While the software development of the ground-based load sensor system led to an operational proof-of-concept of the new design, several improvements of the prototype's software may enhance its reliability, performance and ease-of-use.

Further development will include the integration of calibration programs as part of the data processing in order to record directly force and moment data in Newtons and Newton-meters. This feature is necessary to provide real-time feedback to the astronauts. In addition, analysis of data during lab tests as well as later analysis of space data will be greatly simplified.

A lesson learned from the EDLS experiment aboard the Mir space station is that hot-swapping a sensor may affect the signal quality and make further analysis difficult. The use of wireless communications in the design of the advanced load sensors reduces the risks of hot swapping since the sensor is fully integrated with the SEU. A mechanical and/or electronic mechanism could be combined with software protection to prevent problems related to sensor hot-swapping. In particular, file system corruption from sudden removal of the power can be prevented by running the Linux operating system from a RAM disk and a read-only file system. In addition, error correction codes may prevent storage of corrupted data.

An optional smart feature for the kinetic system is the store-and-forward capability of the server program. A read-write filesystem may be created on the PC/104 CPU to store data locally if the connection with the wireless server is interrupted or lost. Thus, the sensors will be able to continue collecting data locally even if the operator assigns the laptop computer (server) to other tasks or if the server is physically inaccessible via RF connection. Data will be later sent to and stored on the laptop computer when the latter becomes accessible again.

The development of a Graphical User Interface (GUI) will provide a friendly interface to the operator. The set-up procedure will be guided, simple and shorten to the extent possible. During the experiment, the operator will be able to obtain graphics either from real-time data or from previous data. The operator will also be able to modify any software-adjustable parameter such as gain, sampling frequency, or signal offset.

Ultimately the advanced load sensor system will include the capability to communicate with the ELITE-S2 system, an enhanced space-qualified version of the motion analyzer ELITE-S. Time-synchronization between the two systems will create the first global kinetic and kinematic space system. Several solutions are provided herein for time-synchronization. The easiest way to synchronize both systems is to get the ELITE-S2 system able to communicate on the advanced load sensors' wireless network, which only requires that the ELITE-S2 computer has a PCMCIA interface for a wireless LAN PC-card. Given the wireless network capability, the ELITE computer can make a TCP connection to the sensors central laptop computer and synchronize the kinetic and kinematic data. Another solution for time-synchronization is a physical connection from the ELITE computer to the advanced sensors laptop computer. If the ELITE-S2 system is running a Network Time Protocol (NTP) server, the sensors laptop computer can be connected to the Italian kinematic system via crossover ethernet cable, so that the kinetic and kinematic systems synchronize their system clocks and record time stamps with their data.

4.6.4 Conclusion

This thesis presents the conceptual design of the advanced load sensor system envisioned for the ISS, enhancing the EDLS system that flew on Mir. In addition, it provides a detailed description of the ground-based prototype delivering a proof-of-concept of the new design and allowing early tests and data acquisition in the microgravity environment of the NASA KC-135 aircraft.

The ground-based prototype system of advanced load sensors achieves most of the requirements defined in section 3.1.3 for the ISS kinetic system. In particular the MICR0-G prototype is based on three types of load sensors (a handhold, a foot restraint and a touchpad) able to record the forces and moments along the x-, y-, and z- axes (6 d.o.f.) and providing the same functionality as the foot loops and hand rails on the Space Shuttle and ISS for astronaut use. The MICR0-G advanced load sensors incorporate miniaturized electronics achieving a reduction in size and mass by a factor of four compared to the latest EDLS system. In addition, the sensor prototype successfully integrates the wireless capability yielding a flexible and mobile system. All the electronics components, apart from the signal conditioning board, are COTS products reducing past and future development and production costs. The advanced load sensor system tested at MIT is able to provide a real-time visual feedback, even though this capability has not been exploited during the MICR0-G experiment aboard the KC-135 aircraft. In order to achieve the entire set of requirements for the development of the advanced load sensors envisioned for the ISS, a few additional features have to be integrated to the current prototype. First, miniaturized power supply components will be added to the SEU and the resulting unit will be attached underneath each sensor. The design will be re-evaluated and the hardware verified to be fully compatible with the ISS (power, restraint aids, data bus, etc.). The future software development includes real-time data processing for immediate visual feedback, acquisition of data only when a load is applied on a sensor, synchronization with the ELITE-S2 system and development of a simple and friendly Graphical User Interface (GUI).

The MICR0-G project shows a successful scientific collaboration between the U.S. and Italy. The space-qualified system ELITE-S2 is currently developed at the Politecnico di Milano University in Italy. It is hoped that the advanced load sensors will receive NASA support for the development of spaceflight hardware. Flying the space qualified sensors in conjunction with the Italian real-time motion analyzer aboard the ISS represents a great opportunity to aid in the IVA operation of the station and to provide a human factor assessment of astronaut reactions and motor strategies in weightlessness for long-duration space missions.

REFERENCES

[1] L.Abbott, G. Cox, and H. Nguyen, "Space-Qualification of the PC/104 Standard", PC/104 Embedded Solutions, Summer 2000.

[2] R. Alena, "The Mir Wireless Network Experiment (MWNE) Computers", URL http://ic.arc.nasa.gov/ic/projects/WNE/wne-equip.html, December 1995.

[3] A. Amir, "Design and Development of Advanced Load Sensors for the International Space Station", E.A.A. Thesis, Massachusetts Institute of Technology, 1998.

[4] A. Amir, G. Baroni, A. Pedrocchi, D. Newman, "Measuring Astronaut Performance on the ISS: Advanced Kinematic and Kinetic Instrumentation", IEEE Transaction on Instrumentation and Measurements, Venice, Italy, May 1999 A.

[5] A. Amir, D. Newman, "Research Into the Effects of Astronauts Motion on the Spacecraft: a Review", Acta Astronautica, November 1999 B.

[6] A. Amir, D. Newman, S.M. Beck, "Astronaut-Induced Disturbances to the Microgravity Environment on Board the Mir Space Station: Results of the Enhanced Dynamic Load Sensors Spaceflight Experiment", AIAA Journal of Spacecraft and Rockets, in press, 2000.

[7] Ampro Computers, "Product Brochure", On-line document, URL http://www.ampro.com/html/white_papers.asp, 2000.

[8] M. Becker, "PC/104 bus Connector and Assembly Solutions", PC/104 Embedded Solutions, Fall 1998.

[9] Boeing Defense & Space Group, "System Specification for the International Space Station", Specification Number SSP 41000, Revision H, 25 May 1998.

[10] E. Bokhour, R. Grimes, M. Hachkowski, C. Krebs, J. Zapetis, "Middeck 0-Gravity Dynamics Experiment Flight Systems," Acta Astronautica, Vol. 29, No. 10/11, IAF Paper 92-0776, 1992.

[11] J. Byler, "Software Development of Advanced Load Sensors for the International Space Station", Advanced Undergraduate Project, Massachusetts Institute of Technology, 2001.

[12] B. Conway, "Development of Skylab Experiment T-013 Crew/Vehicle Disturbances", NASA Technical Note D-6584, January 1972.

[13] B.Conway, A. Bruce, "Investigation of Crew Motion Disturbances on Skylab-Experiment T 013", Advances in the Astronautical Sciences, 1974.

[14] B. Conway, T. Hendricks, "A Summary of the Skylab Crew/Vehicle Disturbance Experiment T-013", Nasa Technical Note D-8128, March 1976.

[15] K. Cowing, "2001: A Space Laptop", On-line document, URL http://www.spaceref.com/news/viewnews.html?id=213, September 2000.

[16] R. DeLombard, E. Nelson., "An Overview of the Microgravity Environment and Its Effects on Science," AIAA. p.1-5, January 15–18, 1996.

[17] Diamond Systems Inc., "Diamond-MM-16 Specifications", On-line document, URL http://www.diamondsystems.com/diamondmm16.htm, 2000.

[18] G. Ferrigno, A. Pedotti, "ELITE: A Digital Dedicated Hardware System for Movement Analysis via Real-time TV signal Processing", IEEE Trans. Biomed. Eng., BME 32, 943-950, 1985.

[19] G. Ferrigno, G. Baroni, A. Pedotti, "Module ELITE-S2: Technical Desrciption and Potential Application Multifactorial Movement Analysis in Microgravity", Bioengineering Department Politecnico di Milano / Italian Space Agency.

[20] W. Fuhrmeister, J. Fowler, "Experimental Study of Dynamics Effects of Crew Motions in a Manned Orbital Research Laboratory (MORL)", NASA Contract Report 66186, 1966.

[21] Y. Gawdiak, 'The Mir Wireless Network Experiment", On-line document, URL http://spaceflight.nasa.gov/history/shuttle-mir/science/shuttmir/shutmir/exphis/issmwne.htm, January 1998 A.

[22] M. Goodman,W. Middleton, "Crew Locomotion Disturbances in a Space Cabin Simulator", Journal of Spacecraft, No. 10, 1207–1209, 1969.

[23] T. Hendricks, C. Johnson, "Stochastic Crew Motion Modeling". Journal of Spacecraft, No. 2, 150–154, 1971.

[24] D. Hinds, "Linux PCMCIA Supported Device List", On-line document, URL http://pcmcia.sourceforge.org/ftp/SUPPORTED.Cards, March 2000.

[25] K. Hubbard, "Advanced Portable Workstation (APW) Development", On-line document, URL http://ic-www.arc.nasa.gov/ic/projects/PCA/apw.html, December 1995.

[26] S. Huet, "Embedded Linux Softwares", On-line document, URL http://linux-embedded.com/software.php3, 2000.

[27] C. Keyes, "Calibration of of Advanced Load Sensors for the International Space Station", Advanced Undergraduate Project, Harvard University, 2001.

[28] L. Kiser, "The Mir Wireless Experiment Background", On-line document, URL http://ic-www.arc.nasa.gov/ic/projects/mir-DTO/index.html, 1995.

[29] C. Kullas, "Handbook on Astronaut Crew Motion Disturbances for Control System Design," NASA Reference Publication 1025, 1979.

[30] R. Lehrbaum, "PC/104: The Non-backplane Alternative", PC/104 Embedded Solutions, Fall 1997.

[31] R. Lehrbaum, "Using Linux in Embedded and Real-time Systems", On-line article, URL http://linuxdevices.com/cgi-bin/article_view.cgi?artid=AT3611822672, February 2000.

[32] Linux Online!, "What is Linux", On-line document, URL http://www.linux.org/info/index.html, 2000.

[33] J. Miller "PC/104 Architecture", PC/104 Embedded Solutions, Fall 1997.

[34] C. Moore, et al., "Phase 1: A Journey to Mir 1994–98," CD-ROM, SM-G-472, Lockheed Martin, 1998.

[35] NASA Headquarters, "Creating Microgravity", On-line document, URL http://quest.arc.nasa.gov/smore/background/microgravity/MGintro3.html, 1996.

[36] NASA Langley Research Center, MIT and Payload Systems Inc.,"The Enhanced Dynamic Load Sensor (EDLS) Payload Review for NASA 5", April 1997 A.

[37] NASA Langley Research Center, "International Space Station: Main Elements", On-line document, URL http://outerplanets.larc.nasa.gov/issvc97/material.html#elements, 1997 B.

[38] NASA Headquarters, "ISS Research Plan", On-line document, URL http://www.hq.nasa.gov/office/olmsa/ISS/toc.htm, 1997 C.

[39] NASA, "Free-fall and Microgravity", On-line document, URL http://www.science.nasa.gov/msl1/ground_lab/msl1freefall.htm, 1997 D.

[40] NASA Lyndon B. Johnson Space Center, "International Space Station: Benefits from the Shuttle-Mir Program," NASA Facts, March 1998 A.

[41] NASA Johnson Space Center, "Space Station Requirements for Electromagnetic Compatibility", NASA Reference Publication SSP 30243 Revision E, June 1998 B.

[42] NASA Johnson Space Center, "The ISS Familiarization Manual", On-line document, URL http://spaceflight.nasa.gov/spacenews/factsheets/pdfs/td9702.pdf, July 1998 C.

[43] NASA, Glenn Research Center, "Understanding Microgravity", On-line document, URL http://microgravity.grc.nasa.gov/new/frame3.html, April 1998 D.

[44] NASA, Marshall Spaceflight Center, "Microgravity Research Program", On-line document, URL http://microgravity.grc.nasa.gov/new/frame3.html,, April 1998 E.

[45] NASA Johnson Space Center, "Operations LAN/File Server/SSC Operation Concepts", Rev B, January 1999 A.

[46] NASA, Glenn Research Center, "Microgravity Control Plan", SSP 57010, April 1999 B.

[47] NASA, "International Space Station Assembly Sequence" Rev. F, On-line document, URL http://spaceflight.nasa.gov/station/assembly/flights/chron.html, August 2000 A.

[48] NASA, "The ISS Partner Nations", On-line document, URL http://spaceflight.nasa.gov/station/partners.htm, September 2000 B.

[49] NASA Johnson Space Center, "Operations Local Area Network (OPS LAN) Interface Control Document", JSC 36381, February 2000 C.

[50] NASA, Glenn Research Center, "Creating a World-Class Orbiting Laboratory", On-line document, URL http://www.pbs.org/kcet/johnglenn/future/iss/creating/creating.htm, 2000 D.

[51] NASA Johnson Space Center, "Reduced Gravity Program Overview", On-line document, URL http://jsc-aircraft-ops.jsc.nasa.gov/kcprog.htm, November 2000 E.

[52] NASA Johnson Space Center, "The KC-135 Aircraft", On-line document, URL. http://jsc-aircraft-ops.jsc.nasa.gov/kcair.htm, November 2000 F.

[53] NASA Johnson Space Center, "JSC Reduced Gravity Program User's Guide", JSC 22803, Rev B, On-line document, URL http://jsc-aircraft-ops.jsc.nasa.gov/22803cpcn1.pdf, October 2000 G.

[54] NASA Johnson Space Center, On-line photos of the MICR0-G experiment aboard the KC-135 aircraft, URL http://zerog.jsc.nasa.gov/SLSD/viewer.cgi, 2001.

[55] NASDA's Space Environment Utilization Research Center, "Space Environment for the ISS", On-Line document, URL http://idb.exst.nasda.go.jp/edata/02110/199810K02110020/199810K02110020.html, August 2000.

[56] D. Newman, M. van Schoor, J. de Luis, "Crew Force Measurements: Dynamic Load Sensors (DLS) on Mir," Proposal to the NASA Office of Life & Microgravity Sciences & Applications, May 1994.

[57] D. Newman, M.Tryfonidis, M.van Schoor, "Astronaut-Induced Disturbances in Microgravity". Journal of Spacecraft and Rockets, Vol. 34, No. 2, pp. 252–254, March–April 1997 A.

[58] D. Newman, A. Pedotti, S. Beck, "Microgravity Investigations and Crew Reactions in 0-G (MICR0-G)", Proposal to the NASA Office of Life & Microgravity Sciences & Applications, March 1997 B.

[59] PC/104 Embedded Systems FAQ, On-line document, URL http://www.controlled.com/pc104faq/#howman, November 2000.

[60] R. Roberson, "Comments on the Incorporation of Man into the Attitude Dynamics of Spacecraft", The Journal of Aeronautical Sciences, No. 1, p. 27-28, 1963.

[61] J. Robey, "Planning for Microgravity Science Research on the International Space Station," 35th Aerospace Sciences Meeting & Exhibit, AIAA Paper 97-0106, January 1997.

[62] D. Sander, "Objectives of the MWNE", On-line document, URL http://spaceflight.nasa.gov/shuttle/archives/sts-74/orbit/payloads/ls/rme/w_oview.html, November 1995.

[63] ShuttlePressKit, "The ISS Program", On-line document, URL http://www.shuttlepresskit.com/ISS_OVR, June 1999.

[64] T. Starner, "The PC/104 standard and compatible software", On-line document, URL http://mevard.www.media.mit.edu/projects/wearables/pc104/pc104.html, June 1998.

[65] M. Van Schoor, D. Newman, "Dynamic Load Sensor Experiment: Background, Science Objectives, Requirements, and Preliminary Design," November 18, 1992.

[66] J. Williams, "The Case for Embedded Linux", New Technologies Magazine, November 1999.

[67] "Module ELITE-S2: Technical Description and Potential Application Multifactorial Movement Analysis in Microgravity", Bioengineering Department Politecnico di Milano / Italian Space Agancy, 1998.

APPENDIXES

APPENDIX

A

THE SOURCE CODE

A.1 Introduction

This appendix provides the source code of the Diamond-MM-16 A/D linux driver that has been developed by Jesse Byler at MIT [Byler, 2001]. A description of all the programs and how they interact is provided herein.

The source code for the user level programs that interact with the kernel driver dmm16.o is composed of the following programs:

- **d16read.c** is the source for d16read, a program that reads data from the DMM16 using the driver dmm16.o.
- **d16write.c** is the source for d16write, a program that writes data to the DMM16 using the driver dmm16.o.
- **d16ctl.c** is the source for d16ctl, a program that configures the DMM16 by using the IOCTLs of the driver dmm16.o.

The source code for the kernel driver dmm16.o is composed of the following programs:

- **dmm16.c** is the primary driver source file. Together with dmm16.h, dmm16-internal.h, config.h, and linux_version.h, it creates dmm16.o.
- **dmm16.h** is the header file which programs that use the driver include. For instance, d16read, d16write, and d16ctl all include this file.
- **dmm16-internal.h** contains definitions for low level DMM16 operations and board constructs. This is used by dmm16.c.
- **config.h** contains some compile-time parameters for dmm16.c. It sets the IRQ for the board, the major number used by the driver, the size of the buffer, and the debug level.
- **linux_version.h** contains a hack that fools the compiler into thinking we are compiling on a different kernel than we really are.

Makefile contains computer readable instructions for how to build the entire package. It takes the 8 source files above and creates 3 linux executables (d16read, d16write, d16ctl) as well as the linux kernel module dmm16.o.

A.2 Usage

The kernel module must be loaded first with the command **insmod dmm16.o** executed by the superuser. Once the module is loaded, the following shell scripts show how the card is set up and data is read and written. In the following scripts, csh aliases are defined as follows:

- The alias "0c" expands to "d16ctl /dev/dmm-c0-adc0" followed by the remaining arguments, which are an ioctl number and a value. Definitions for the ioctl numbers are in dmm16.h.
- The alias "dw" expands to "d16write /dev/dmm-c0-digital" followed by the remaining arguments, which should be a single number to be written to the digital output lines of the DMM16.
- The alias "b" expands to "d16read /dev/dmm-c0-adc-batch" followed by the remaining arguments, which are switches interpreted by d16read. In the following example, "-t" specifies that timing information is to be output, "-n 6" specifies that we are sampling 6 channels at once (set up in the kc_setup script by the line "0c 4 0x05"), and "30" specifies that 30 seconds of data are to be collected.

The script **kc_setup** initializes the card with the parameters used on our flight:

```
#!/bin/tcsh
set prompt="t"     # These two lines are a source $HOME/.cshrc # very silly hack...

0c 0 8    # Set A/D range to +/-10v
0c 2 250  # Set sampling rate to 250 Hz
0c 4 0x05 # Set batch channels to 0-5
dw 17     # Set sig. cond. gain to 200
```

The script **kc_sample** initiates a 30 second sampling period:

```
#!/bin/tcsh
set promptly"     # These two lines are a source $HOME/.cshrc # very silly hack...

# Read 30 seconds of data (250Hz * 6 channels * 30 seconds) and
# print timing information
b -t -n 6 30
```

A.3 Source Code

```
/*
 * This file is the source for d16read, a user space program which
 * interacts with the dmm16.o driver. d16read accepts a number of
 * command line arguments which control the output it produces. This
 * version of the program was used on the KC-135 flights 2-4, January
 * 24-26, 2001, with the following command line:
 *
 * d16read -t -n 6 /dev/dmm-c0-adc-batch 30
 *
 * -t indicates that time information should be printed.
 * -n 6 means that we are sampling in batch mode with 6 channels.
 * /dev/dmm-c0-adc-batch is the device special file that represents
 * the dmm16.o driver.
 * 30 is the number of seconds to sample for.
 *
 * The other switch of interest is -s, which prints summary statistics
 * for the data gathered.
 */

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <math.h>
#include <string.h>
#include "dmm16.h"
#include "sched.h" // for setting realtime priority
#include <sys/mman.h> // For mlockall
#include <sys/time.h>

#define CONV(sample) (sample + offset) * scale

void usage() {
  printf("d16read version 1.11, January 12, 2001\n");
  printf("Usage: d16read [-raw] [-s] [-i] [-t] [-n channels] \
/dev/dmm-c?-adc* [num-samples]\n");
}

// sets the process to "policy" policy, if max=1 then set at max priority,
// else use min priority
int set_realtime_priority(int policy,int max)
{
struct sched_param schp;
/*
 * set the process to realtime privs
 */
     memset(&schp, 0, sizeof(schp));
if(max)
     {
       schp.sched_priority = sched_get_priority_max(policy);
     }
     else
```

```
    {
     schp.sched_priority = sched_get_priority_min(policy);
    }


if (sched_setscheduler(0, policy, &schp) != 0) {
perror("sched_setscheduler");
return -1;
}

return 0;

}

int main(int argc, char **argv) {
 char *file_name = NULL;
 int samples = 100;
 int print_summary = 0;
 int print_volts = 1;
 int channels = 1;

 FILE *special;
 int file_desc;
 int i, j, chan;
 char *arg;
 int16_t *buffer;
 size_t hits;
 int16_t *min, *max, samp;
 long *sum;
 double *mean, *stdev;
 double scale = 1.0;
 double seconds;
 int offset = 0;
 int sample_rate = 0;
 int individual_reads_mode = 0;
 int continuous_mode = 0;
 int ret_val;
 int buf_size = -1, remaining;
 int sample_offset = 0;
 int print_time = 0;
 struct timeval begin, end;
 double begin_sec, duration;
 time_t start_t;
 struct tm *start_tm;

 for(i = 1; i < argc; i++) {
  arg = argv[i];
  if(!strcmp(arg,"-raw")) {
   print_volts = 0;
  } else if(!strcmp(arg,"-s")) {
   print_summary = 1;
  } else if(!strcmp(arg,"-c")) {
   continuous_mode = 1;
  } else if(!strcmp(arg,"-t")) {
   print_time = 1;
  } else if(!strcmp(arg,"-n")) {
   i++;
```

```
      if(!sscanf(argv[i], "%d", &channels)) {
        usage();
        exit(-1);
      }
    } else if(strstr(arg, "dmm") != NULL) {
      file_name = arg;
    } else if(!strcmp(arg, "-i")) {
      individual_reads_mode = 1; /* As in old d16read.c */
      buf_size = 10;
    } else {
// See comment below for interpretation of samples...
// Most of the time, the numeric argument is in seconds.
      if(!sscanf(arg, "%d", &samples)) {
        usage();
        exit(-1);
      }
    }
  }

  if(samples <= 0) {
    printf("Sample count must be positive.\n");
    exit(-1);
  }

  if(file_name == NULL) {
    usage();
    exit(-1);
  }

  special = fopen(file_name, "r");
  if(special == NULL) {
    printf("Could not open %s\n", argv[1]);
    exit(-1);
  }

  file_desc = open(file_name, 0);
  if(file_desc < 0) {
    printf("Could not open %s; raw mode forced\n", file_name);
    print_volts = 0;
  } else {
    ret_val = ioctl(file_desc, DMM_IOCTL_GET_ANALOG_CODE, 0);
    switch(ret_val) {
    case 0:
      scale = 5.0 / 32768.0;
      break;
    case 1:
      scale = 2.5 / 32768.0;
      break;
    case 2:
      scale = 1.25 / 32768.0;
      break;
    case 3:
      scale = 0.625 / 32768.0;
      break;
    case 8:
      scale = 10.0 / 32768.0;
      break;
  case 9:
    scale = 5.0 / 32768.0;
    break;
  case 10:
    scale = 2.5 / 32768.0;
    break;
  case 11:
    scale = 1.25 / 32768.0;
    break;
  case 12:
    offset = 32768;
    scale = 10.0 / 65536.0;
    break;
  case 13:
    offset = 32768;
    scale = 5.0 / 65536.0;
    break;
  case 14:
    offset = 32768;
    scale = 2.5 / 65536.0;
    break;
  case 15:
    offset = 32768;
    scale = 1.25 / 65536.0;
    break;
  default:
    printf("Invalid analog mode %d; raw mode forced\n", ret_val);
    print_volts = 0;
    break;
  }

  // Samples now holds a # of seconds. We multiply by the sampling rate
  // in Hz and the number of channels being sampled to get the total number
  // of samples needed.
  printf("Recording: Seconds of data: %d\n", samples);
  ret_val = ioctl(file_desc, DMM_IOCTL_GET_SAMPLE_RATE, 0);
  sample_rate = ret_val * channels; // *total* sample rate
  samples *= sample_rate;
  printf("Sample Rate: %d Hz Channels: %d Total samples: %d\n", ret_val, channels, samples);
}

buffer = calloc(samples, sizeof(int16_t));

if(buffer == NULL) {
  printf("Could not allocate memory for buffer\n");
  exit(-1);
}

if(print_summary) {
  min = malloc(channels*sizeof(int16_t));
  max = malloc(channels*sizeof(int16_t));
  sum = malloc(channels*sizeof(long));
  mean = malloc(channels*sizeof(double));
  stdev = malloc(channels*sizeof(double));

  if(min == NULL || max == NULL || sum == NULL ||
    mean == NULL || stdev == NULL) {
```

```
    printf("Could not allocate memory for analysis\n");
    exit(-1);
  }
} else { // So the compiler doesn't complain...
  min = NULL; max = NULL; sum = NULL; mean = NULL; stdev = NULL;
}

// sets realtime priority (SCHED_FIFO, max priority)
// from http://www.linuxdj.com/latency-graph/testlatency.c
set_realtime_priority(SCHED_FIFO,1);

// keep all memory locked into physical mem, to "guarantee" realtime-behavior
mlockall(MCL_CURRENT|MCL_FUTURE);

do {

if(individual_reads_mode) {
  time(&start_t);
  gettimeofday(&begin,NULL);
  for(i = sample_offset; i < samples + sample_offset; i+= buf_size) {
    remaining = samples - i;
    if(remaining > buf_size) remaining = buf_size;

    if((hits = fread(&buffer[i], 2, remaining, special)) != remaining) {
      printf("Only read %d samples...\n", i);
      exit(-1);
    }

    for(j = 0; j < remaining; j++) {  /* Print them as we get them */
      if(print_volts)
        printf("%d %+.5f\n", i+j, CONV(buffer[i+j]));
      else
        printf("%d %d\n", i+j, buffer[i+j]);
    }

    if(remaining < buf_size) break;
  }
  gettimeofday(&end,NULL);
} else {
  time(&start_t);
  gettimeofday(&begin,NULL);
  if((hits = fread(buffer, 2, samples, special)) != samples) {
    printf("Only read %d samples...\n", hits);
    exit(-1);
  }
  gettimeofday(&end,NULL);
}

if(print_time) {
  start_tm = gmtime(&start_t);
  printf("UTC at start: %04d.%02d.%02d at %02d:%02d:%02d\n",
 (start_tm->tm_year+1900), (start_tm->tm_mon+1), start_tm->tm_mday,
 start_tm->tm_hour, start_tm->tm_min, start_tm->tm_sec);
  begin_sec = begin.tv_sec+begin.tv_usec/1000000.0;
  duration = (end.tv_sec+end.tv_usec/1000000.0) - begin_sec;
  printf("Sampling started at time %.3f and lasted %.5f seconds.\n",
 begin_sec, duration);
```

```
}

if(print_summary) {

  // Initialize the arrays
  for(chan = 0; chan < channels; chan++) {
    min[chan] = max[chan] = buffer[chan];
    sum[chan] = 0;
  }

  for(i = sample_offset; i < samples + sample_offset; i++) {
    chan = i % channels;
    samp = buffer[i];
    if (samp < min[chan]) min[chan] = samp;
    if (samp > max[chan]) max[chan] = samp;
    sum[chan] += samp;
  }

  for(chan = 0; chan < channels; chan++) {
    mean[chan] = sum[chan]*channels / samples;
    sum[chan] = 0;
  }

  for(i = 0; i < samples; i++) {
    chan = i % channels;
    samp = buffer[i] - mean[chan];
    stdev[chan] += samp*samp;
  }

  for(chan = 0; chan < channels; chan++) {
    stdev[chan] = sqrt(stdev[chan]*channels / samples);
  }

  if(channels == 1) {
    printf("Min: % .5fv Max: % .5fv Mean: % .5fv Stdev: % .6fv\n",
        CONV(min[0]), CONV(max[0]), CONV(mean[0]), CONV(stdev[0]));
  } else {
    for(chan = 0; chan < channels; chan++) {
      printf("Channel %d Min: % .5fv Max: % .5fv Mean: % .5fv Stdev: % .6fv\n",
          chan, CONV(min[chan]), CONV(max[chan]),
          CONV(mean[chan]), CONV(stdev[chan]));
    }
  }
}

// Print the sampled data
for(i = sample_offset; i < samples + sample_offset; i++) {
  samp = buffer[i];
  if(!individual_reads_mode) {
    if(print_volts)
if(channels == 6) { // bad hack!!!!
  seconds = ((double)(i))/sample_rate;
        printf("%.5f %+.5f %+.5f %+.5f %+.5f %+.5f %+.5f\n", seconds,
 CONV(buffer[i]), CONV(buffer[i+1]), CONV(buffer[i+2]),
 CONV(buffer[i+3]), CONV(buffer[i+4]), CONV(buffer[i+5]));
  i += 5;
} else {
        printf("%d %+.5f\n", i, CONV(samp));
  }
    else
      printf("%d %d\n", i, samp);
   }
  }

  sample_offset += samples;

  } while(continuous_mode && !feof(special));

  /* I think exit does all this anyway, but... */
  if(print_summary) {
    free(min); free(max); free(sum); free(mean); free(stdev);
  }
  close(file_desc);
  fclose(special);
  exit(0);
}
```

==

```
/*
* This file is the source for d16write, a user space program which
* interacts with the dmm16.o driver. d16write allows writes either
 * to the D/A converter or to the digital I/O lines of the
 * Diamond-MM-16 card. The primary use for this function in the
 * KC-135 flights is to set the signal conditioning board's gain,
 * which is done by writing values to the digital I/O lines of the
 * DMM-16 card.
 */
#include <stdio.h>
#include <sys/types.h>
#include <string.h>

int main(int argc, char **argv) {
  FILE *special;
  int16_t value;
  int8_t byte_value;

  if (argc != 3) {
    printf("Usage: d16write /dev/dmm-c?-dac* value\n");
    exit(-1);
  }

  special = fopen(argv[1], "w");
  if (special == NULL) {
    printf("Could not open %s!", argv[1]);
    exit(-1);
  }

  // Kinda hacky; depends on the name of the device special file
  // following this convention...
  if (strstr(argv[1], "digital")) {
    sscanf(argv[2], "%hhd", &byte_value);

    if(fwrite(&byte_value, 1, 1, special) != 1) {
```

```
    printf("Write failed!\n");
    exit(-1);
   }
  } else {
   sscanf(argv[2], "%hd", &value);

   if(fwrite(&value, 2, 1, special) != 1) {
    printf("Write failed!\n");
    exit(-1);
   }
  }

  fclose(special);
  exit(0);
}
```

==

```
/*
 * This file is the source for d16ctl, a user space program which
 * interacts with the dmm16.o driver.  d16ctl is a very low level
 * interface to the various card parameters; IOCTLs must be specified
 * by number.  Arguments may be specified in hexadecimal by prefixing
 * the number with "0x".
 */


#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <string.h>
#include "dmm16.h"

int main(int argc, char **argv) {
  int file_desc;
  int command;
  int argument;
  int ret_val;

  if (argc != 3 && argc != 4) {
   printf("Usage: d16ctl /dev/dmm-c?-ctl command [argument]\n");
   exit(-1);
  }

  file_desc = open(argv[1], 0);
  if (file_desc < 0) {
   printf("Could not open %s!", argv[1]);
   exit(-1);
  }

  sscanf(argv[2], "%d", &command);
  if (argc == 4)
   if (strstr(argv[3], "0x"))
    sscanf(argv[3], "%x", &argument);
   else
    sscanf(argv[3], "%d", &argument);
  else
    argument = 0;

  /* This assumes dmm_ioctls contains a list of all the ioctls and
     that they are sequential */
  if (command < 0 || command > sizeof(dmm_ioctls) / sizeof(int) - 1) {
    printf("Command %d not implemented yet or does not exist.", command);
    exit(-1);
  }

  ret_val = ioctl(file_desc, dmm_ioctls[command], argument);
  printf("Ioctl returned %d\n", ret_val);

  close(file_desc); /* I think exit does this anyway, but... */
  exit(0);
}
```

==

```
/*
 * This is the kernel module dmm16.o, which is a driver for the
 * Diamond-MM-16 A/D card from Diamond Systems. It takes advantage of
 * the DMM-16's hardware clock to allow precise timing, and implements
 * a batch mode in which multiple channels can be sampled in one
 * interrupt.  It is based on a driver for the Diamond-MM-12 card
 * written by Sebastian Kuzminsky and released under the GNU General
 * Public License.
 */

#include "linux_version.h"  /* Bad Hack!  Get rid of this!! -JDB */

#include <linux/module.h>
#include <linux/modversions.h>
#include <linux/kernel.h>

#include <asm/uaccess.h>
#include <linux/ioctl.h>
#include <linux/errno.h>
#include <linux/major.h>
#include <linux/sched.h>
#include <linux/tqueue.h>

#include <linux/mm.h>
#include <linux/ioport.h>
#include <linux/interrupt.h>
#include <linux/timer.h>

#include <linux/fcntl.h>

#include <asm/io.h>

#include "config.h"
#include "dmm16-internal.h"
#include "dmm16.h"
```

```
//
// module parameters
//
int io[DMM_MAX_CARDS] = { };
MODULE_PARM(io, "1-" __MODULE_STRING(DMM_MAX_CARDS) "i");

int irq = DMM_IRQ;
MODULE_PARM(irq, "1i");
//
// This array holds information about the cards
//
static struct dmm_card_t dmm_card[DMM_MAX_CARDS];

static int num_cards = 0;


//
// This is used to sample with a hardware trigger
//
int hardware_trigger_mode = 0;
int sample_rate = -1; /* -1 signifies the sample rate is unknown */


//
// These are used by batch mode
//
int isr_runlimit = 0;


//
//
// Utility functions for sampling and scanning
//
//

// Delay about 10 microseconds (or whatever value is appropriate)
// This implementation is very hardware dependent and *bad*.
void dmm_delay() {
 int i;

 // This should be customizable via an IOCTL...
 for(i=0; i<200; i++) {
 }
}

// Delay some long lenth of time, "times" times. I don't know what
// the long length should be or even if this is necessary at all...
// This implementation is very hardware dependent and *bad*.
void dmm_delay_long(int times) {
 int i;
 int loop;

 loop = 500*times;
 // This should be customizable via an IOCTL...
 for(i=0; i<loop; i++) {
 }
}
```

```
// Read a single sample from the card
int16_t dmm_sample(int card) {
  int i;
  START_ADC_CONVERSION(card); // kick it off
  dmm_delay();  // Wait for the circuitry to settle

  for(i=0; i< 1000; i++) { // Wait for the new value to become available
    if((GET_STATUS_REGISTER(card) & STATUS_INTERRUPT_PENDING) == 0) {
      break;
    }
  }

  if(i == 1000) {
    printk(KERN_ERR DMM_ID "Status register never cleared on card %d\n", card);
    return 0;
  }

  return READ_ADC(card);
}

// Perform a scan on the card
// This function assumes the batch_channel_low and _high of the card
// are valid; it also assumes there is enough space in the card's
// buffer to hold all the samples. These checks are done in dmm_isr.
void dmm_scan(int card) {
  int config;
  int i, samples;

  DPRINTK("Beginning a/d scan...\n");
  CLEAR_INTERRUPT_FLIP_FLOP(card);
  config = GET_CONTROL_REGISTER(card); // Save config for later
  SET_CONTROL_REGISTER(card, 0); // Clear inte, trige, inttrig

  // Set up channel register
  SELECT_ADC_RANGE(card, dmm_card[card].batch_channel_high,
            dmm_card[card].batch_channel_low);

  // Not sure how long this delay needs to be... I think 10 usec
  // from the manual, but the example code has 3 delay_longs, and I'm
  // not sure how long that is... I would have expected a normal
  // delay() to be 10 usec.
  dmm_delay_long(3);

  samples = (dmm_card[card].batch_channel_high + dmm_card[card].channels -
        dmm_card[card].batch_channel_low) % dmm_card[card].channels + 1;

  for(i = 0; i<samples; i++) {
    // Take a sample
    dmm_card[card].buffer[dmm_card[card].tail] = dmm_sample(card);

    // Update card's data structures
    dmm_card[card].tail = (dmm_card[card].tail + 1) % DMM_BUFFER_SIZE;
    dmm_card[card].count -= 2;
  }

  // I don't fully understand this stuff; I'm just following the example
  // Reset config register except for INTE; may generate a conversion
  SET_CONTROL_REGISTER(card, (config & 0x7F));
  dmm_sample(card); // Wait for conversion to complete
  CLEAR_INTERRUPT_FLIP_FLOP(card);
  // Reset channels again (in case of extra conversion just above, maybe?)
  SELECT_ADC_RANGE(card, dmm_card[card].batch_channel_high, \
              dmm_card[card].batch_channel_low);
  SET_CONTROL_REGISTER(card, config); // Restore config totally
}

//
//     name:  dmm_isr()
//
//  function:  This is the interrupt service routine for the Diamond-MM
//           driver. So far it only wakes up the reading process.
//
// arguments:  The IRQ that triggered it, and some other stuff.
//
//   returns:  Nothing.
//
static void dmm_isr(int trigger_irq, void *dev_id, struct pt_regs * regs) {
  int card;
  int next;
  int samples;


  //DPRINTK("ISR running!\n");
  //return;

  // This is a hack to get us out of a squeeze during devel.
  //isr_runlimit--;
  //if(isr_runlimit < 0) {
  //  printk("ISR has exceeded runlimit!\n");
  //  DISABLE_HW_TRIGGER(0); // so the ISR doesn't keep running
  //  return;
  //}

  for (card = 0; card < num_cards; card ++) {
    // I assume this is in case multiple cards share the same interrupt...
    if (! (GET_STATUS_REGISTER(card) & STATUS_INTERRUPT_PENDING)) {
 DPRINTK("card %d needs no attention\n", card);
 continue;
    }

    if (dmm_card[card].scan) {
      samples = (dmm_card[card].batch_channel_high + dmm_card[card].channels -
             dmm_card[card].batch_channel_low) % dmm_card[card].channels + 1;
    } else {
      samples = 1;
    }

    next = (dmm_card[card].tail + samples) % DMM_BUFFER_SIZE;

#ifdef DMM_DEBUG
    // These two tests will never be true if the rest of the driver logic
    // is correct...
    if (dmm_card[card].count <= 0) {
      printk(KERN_ERR DMM_ID "card %d read ISR called with count<=0, nothing to read!\n", card);
      DISABLE_HW_TRIGGER(card); // so the ISR doesn't keep running
      CLEAR_INTERRUPT_FLIP_FLOP(card);
      wake_up_interruptible(&dmm_card[card].wait_q);
      continue;  // skip to the next card
    }

    if (FULL(dmm_card[card].head, dmm_card[card].tail, next)) {
      printk(KERN_ERR DMM_ID "card %d read ISR called with buffer full!\n", card);
      DISABLE_HW_TRIGGER(card); // so the ISR doesn't keep running
      CLEAR_INTERRUPT_FLIP_FLOP(card);
      wake_up_interruptible(&dmm_card[card].wait_q);
      continue;  // skip to the next card
    }
#endif

    if (dmm_card[card].scan) {
      dmm_scan(card);
    } else {
      dmm_card[card].buffer[dmm_card[card].tail] = READ_ADC(card);
      //DPRINTK("card %d read 0x%03x\n", card, dmm_card[card].buffer[dmm_card[card].tail]);
      CLEAR_INTERRUPT_FLIP_FLOP(card);
      dmm_card[card].tail = next;
      dmm_card[card].count -= 2;
    }

    if (dmm_card[card].count > 0) {
      next = (dmm_card[card].tail + samples) % DMM_BUFFER_SIZE;
      // there is more data to read
      if (FULL(dmm_card[card].head, dmm_card[card].tail, next)) {
        // but the buffer is full, so we should restart conversions
        // once the buffer has drained some....
        printk(KERN_ERR DMM_ID "card %d read-buffer fills up\n", card);
        dmm_card[card].need_start_conversion = 1;
        DISABLE_HW_TRIGGER(card); // so the ISR doesn't keep running
        // FIXME:  this wakeup should have maybe come a while ago...
        wake_up_interruptible(&dmm_card[card].wait_q);
      } else {
        // there's room in the buffer, so keep reading
        if(!hardware_trigger_mode) {
          START_ADC_CONVERSION(card);
        }
      }
    } else {
      // done reading
      DPRINTK("card %d done reading\n", card);
      DISABLE_HW_TRIGGER(card); // so the ISR doesn't keep running
      wake_up_interruptible(&dmm_card[card].wait_q);
    }

    // FIXME: How often should we wake the reader?
    //   The reader wakes every jiffy anyway...
    //   The ADC generates 44000 samples/second, or 440 samples per jiffy.

    //if (dmm_card[card].count % 800 == 0) {
```

```
  // DPRINTK(
  // //printk(KERN_ERR DMM_ID
  //   "card %d ISR waking reader (count=%d head=%d tail=%d)\n",
  //   card,
  //   dmm_card[card].count,
  //   dmm_card[card].head,
  //   dmm_card[card].tail
  // );
  // wake_up_interruptible(&dmm_card[card].wait_q);
  //}

 }
}


//
//     Name:  dmm_read()
//
//  Function:  Reads the values of the analog input lines.
//
// Arguments:  Hmmm...
//
//   Returns:  Number of bytes actually read.
//
static ssize_t dmm_read(
 struct file *file,
 char *buffer,
 size_t count,
 loff_t *ppos
) {
 unsigned int real_minor, card_minor;
 int card;

 int i;

 unsigned char c;

 unsigned long flags;
 int tail, num_channels;

 long timeout;


 real_minor = MINOR(file->f_dentry->d_inode->i_rdev);
 card = real_minor / DMM_MINOR_RANGE;
 card_minor = real_minor % DMM_MINOR_RANGE;

 DPRINTK("trying to read from card %d, %s (minor %u)\n", card, dmm_devices_by_minor[card_minor], real_minor);


 switch (card_minor) {
  case DMM_MINOR_ADC_0:
  case DMM_MINOR_ADC_1:
  case DMM_MINOR_ADC_2:
  case DMM_MINOR_ADC_3:
  case DMM_MINOR_ADC_4:
  case DMM_MINOR_ADC_5:
    case DMM_MINOR_ADC_6:
    case DMM_MINOR_ADC_7:
    case DMM_MINOR_ADC_8:
    case DMM_MINOR_ADC_9:
    case DMM_MINOR_ADC_10:
    case DMM_MINOR_ADC_11:
    case DMM_MINOR_ADC_12:
    case DMM_MINOR_ADC_13:
    case DMM_MINOR_ADC_14:
    case DMM_MINOR_ADC_15:
    case DMM_MINOR_ADC_BATCH:
      if (card_minor == DMM_MINOR_ADC_BATCH) {
        num_channels = (dmm_card[card].batch_channel_high + dmm_card[card].channels -
                  dmm_card[card].batch_channel_low) % dmm_card[card].channels + 1;
  if ((count % (num_channels << 1)) != 0) {
    DPRINTK("Batch reads must be in even multiples of the number of channels\n");
    return -ENODATA;
  }
      } else {
  if ((count & 1) == 1) {
    DPRINTK("ADC reads must be of even count\n");
    return -ENODATA;
  }
      }
if (count == 0) {
  // FIXME: can this even happen?
        DPRINTK("Yep, it happened (dmm_read with count = 0).\n");
  return 0;
}

DPRINTK("card %d, %s, entering critical region (sem count=%d)\n",
          card, dmm_devices_by_minor[card_minor],
          dmm_card[card].sem.count.counter);
down_interruptible(&dmm_card[card].sem);
// FIXME: check to see if we got a signal here
DPRINTK("card %d, %s, in critical region\n", card, dmm_devices_by_minor[card_minor]);

      if (card_minor == DMM_MINOR_ADC_BATCH) {
        // This probably isn't strictly necessary, since dmm_scan does this too...
        SELECT_ADC_RANGE(card, dmm_card[card].batch_channel_high,
                  dmm_card[card].batch_channel_low);
        dmm_card[card].scan = 1;
      } else {
        SELECT_ADC_RANGE(card, card_minor, card_minor);
        dmm_card[card].scan = 0;
      }

// set up state for buffered, interrupt-driven read
dmm_card[card].head = 0;
dmm_card[card].tail = 0;
dmm_card[card].count = count;
dmm_card[card].need_start_conversion = 1;

// is this the right looping condition??
while (
  (dmm_card[card].count > 0) ||
  (dmm_card[card].head != dmm_card[card].tail)
```

```
) {
  save_flags(flags);
  cli();

  if (dmm_card[card].need_start_conversion) {
        if(hardware_trigger_mode) {
          SELECT_INTERNAL_HW_ADC_TRIGGER(card);
          ENABLE_HW_TRIGGER(card);
        } else {
          START_ADC_CONVERSION(card);
        }
    dmm_card[card].need_start_conversion = 0;
    DPRINTK("card %d, reader started ADC conversion (need_start_conversion)\n", card);
  }

  if (dmm_card[card].count > 0) {
    // there's more to be read, so let the ISR chug along for
    // a while
        // [I increased the timout here so that slow
        // hardware_triggered reads work -JDB]
    timeout = interruptible_sleep_on_timeout(&dmm_card[card].wait_q, 100);

    // FIXME: Here we either got woken by the ISR or by timeout,
    // or we got a signal.  How do you tell if a signal is pending?

    // FIXME: does the return from interruptible_sleep_on
    // restore the flags?  Probably...
  }

  tail = dmm_card[card].tail;

  DPRINTK(
  //printk(KERN_ERR DMM_ID
    "card %d read-sleeper wakes up (count=%d head=%d tail=%d)\n",
    card,
    dmm_card[card].count,
    dmm_card[card].head,
    tail
  );

  // this lets the ISR keep reading, and since we have a private
  // copy of tail there wont be any confusion
  restore_flags(flags);

  if (dmm_card[card].head == tail) {
    //  the buffer is empty
    //  no data was generated by the card
    //  we awoke due to timeout or signal
      printk(KERN_ERR DMM_ID "card %d (io=0x%03X) not responding, must be missing or malfunctioning\n", card,
dmm_card[card].io);
    DPRINTK("no conversion happened, aborting read\n");
    up(&dmm_card[card].sem);
    return -ENODATA;
  }

  // copy the buffered data out to user space
```

```
    if (dmm_card[card].head > tail) {
      int chunk_size = (DMM_BUFFER_SIZE - dmm_card[card].head) * 2;
      __copy_to_user(
  buffer,
        &dmm_card[card].buffer[dmm_card[card].head],
  chunk_size
      );
      dmm_card[card].head = 0;
      buffer += chunk_size;
    }

    if (dmm_card[card].head < tail) {
      int chunk_size = (tail - dmm_card[card].head) * 2;
      __copy_to_user(
  buffer,
        &dmm_card[card].buffer[dmm_card[card].head],
  chunk_size
      );
      dmm_card[card].head = tail;
      buffer += chunk_size;
    }
  }
  up(&dmm_card[card].sem);
   DPRINTK("card %d %s has left critical region (sem count=%d)\n", card, dmm_devices_by_minor[card_minor],
dmm_card[card].sem.count.counter);
  return count;


     case DMM_MINOR_DIGITAL:
  DPRINTK("card %d, %s, reading %d bytes\n", card, dmm_devices_by_minor[card_minor], count);
       for(i = 0; i < count; i ++) {
    c = inb(dmm_card[card].io + DMM_OFFSET_DIGITAL_INPUT);
    DPRINTK("read 0x%02X\n", c);
    __put_user(c, buffer ++);
  }
       // Why is this i and not count, as in dmm32? (JDB)
  return i;


     default:
  DPRINTK("read() not implemented for this device yet (real minor=%d)...\n", real_minor);
  return -ENOSYS;
    }
  }



/****************************************************************************/
/*                                                */
/*     Name:  dmm_write()                                  */
/*                                                */
/*  Function:  Writes data to the card.                         */
/*                                                */
/* Arguments:  Hmmm...                                  */
/*                                                */
/*   Returns:  Number of bytes actually written.                 */
/*                                                      */
/******************************************************************************/
static ssize_t dmm_write(
  struct file *file,
  const char *buffer,
  size_t count,
  loff_t *ppos
){
  unsigned int real_minor, card_minor;
  int card;

  int n;

  unsigned char data = 0;


  real_minor = MINOR(file->f_dentry->d_inode->i_rdev);
  card = real_minor / DMM_MINOR_RANGE;
  card_minor = real_minor % DMM_MINOR_RANGE;


  DPRINTK("trying to write to card %d, %s (minor %u)\n", card, dmm_devices_by_minor[card_minor], real_minor);


  switch (card_minor) {
    case DMM_MINOR_DAC_0:
    case DMM_MINOR_DAC_1:
    case DMM_MINOR_DAC_2:
    case DMM_MINOR_DAC_3:
      if (count != 2) {
  DPRINTK("only 2-byte writes are allowed to the DACs\n");
  return -ENOSPC;
}
      __get_user(data, buffer ++);
      outb(data, dmm_card[card].io + DMM_OFFSET_DAC_ALL_LSB);
      __get_user(data, buffer ++);
      switch (card_minor) {
        case DMM_MINOR_DAC_0:
          outb(data, dmm_card[card].io + DMM_OFFSET_DAC_0_MSB);
          break;
        case DMM_MINOR_DAC_1:
          outb(data, dmm_card[card].io + DMM_OFFSET_DAC_1_MSB);
          break;
        case DMM_MINOR_DAC_2:
          outb(data, dmm_card[card].io + DMM_OFFSET_DAC_2_MSB);
          break;
        case DMM_MINOR_DAC_3:
          outb(data, dmm_card[card].io + DMM_OFFSET_DAC_3_MSB);
          break;
      }
      inb(dmm_card[card].io + DMM_OFFSET_DAC_ALL_UPDATE);
 return count;

    case DMM_MINOR_DIGITAL:
 for (n = 0; n < count; n ++) {
   __get_user(data, buffer ++);
   outb(data, dmm_card[card].io + DMM_OFFSET_DIGITAL_OUTPUT);
  DPRINTK("wrote 0x%02X\n", data);
 }
 return count;


   default:
 DPRINTK("write to this device is not implemented yet...\n");
 return -ENOSYS;
  }
}


//
//
//     Name:  dmm_ioctl()
//
//  Function:  Performs misc actions on Diamond-MM card and driver.
//
// Arguments:  Hmmm...
//
//   Returns:  0 if everything went well, and "what errno should be" if
//             there's a problem.
//
//     Notes:
//
static int dmm_ioctl(
  struct inode *inode,
  struct file *file,
  unsigned int command,
  unsigned long argument
){
  unsigned int real_minor, card_minor;
  int card;


  real_minor = MINOR(inode->i_rdev);
  card = real_minor / DMM_MINOR_RANGE;
  card_minor = real_minor % DMM_MINOR_RANGE;

  DPRINTK("ioctl %d called on card %d, %s (minor %d) with arg %ld\n", command,
       card, dmm_devices_by_minor[card_minor], real_minor, argument);

  switch (command) {
    case DMM_IOCTL_SET_ANALOG_CODE:
      SET_ADC_CODE(card, (int16_t)argument);
      return 0;

    case DMM_IOCTL_GET_ANALOG_CODE:
      return GET_ADC_CODE(card);
      break;

    case DMM_IOCTL_SET_SAMPLE_RATE:
      if (argument > 0) {
        PROGRAM_COUNTER_1(card, 1);
        /* I'm not sure why this is has to be 500000 (manual says
          it should be 1000000) but this works... */
        PROGRAM_COUNTER_2(card, (int)(500000 / argument));
        hardware_trigger_mode = 1;
        sample_rate = argument;
      } else {
        hardware_trigger_mode = 0;
        sample_rate = 0;
      }

      return 0;

    case DMM_IOCTL_GET_SAMPLE_RATE:
      return sample_rate;

    case DMM_IOCTL_SET_BATCH_CHANNELS:
      dmm_card[card].batch_channel_low = (argument & 0xF0) >> 4;
      dmm_card[card].batch_channel_high = argument & 0x0F;
      // I assume differential mode here (8 channels)
      DPRINTK("Setting batch range: [%d, %d]\n",
           dmm_card[card].batch_channel_low,
           dmm_card[card].batch_channel_high);
      return 0;

    case DMM_IOCTL_GET_BATCH_CHANNELS:
      return (dmm_card[card].batch_channel_low << 4) + dmm_card[card].batch_channel_high;

    /* fopening the device for writing triggers an isatty()
      invocation, which produces this TCGETS ioctl, to which I
      gather ENOTTY is the proper response.  *shrug* */
    case TCGETS:
      return -ENOTTY;

    default:
      DPRINTK("goofy ioctl!  DIR = %d, TYPE = %d, NR = %d, SIZE = %d filename = %s.\n",
           _IOC_DIR(command), _IOC_TYPE(command), _IOC_NR(command),
           _IOC_SIZE(command), file->f_dentry->d_name.name);
      return -EINVAL;
  }

  return 0;
}



/****************************************************************************/
/*                                          */
/*      Name:  dmm_open()                            */
/*                                          */
/*   Function:  Opens a Diamond-MM resource.                   */
/*                                          */
/*  Arguments:  Hmm....                          */
/*                                          */
/*    Returns:  0 if all went well, and some error code if not.            */
/*                                          */
/****************************************************************************/
static int dmm_open(
  struct inode *inode,
  struct file *file
```

```
){
  unsigned int real_minor, card_minor;
  int card;


  real_minor = MINOR(inode->i_rdev);

  card = real_minor / DMM_MINOR_RANGE;
  card_minor = real_minor % DMM_MINOR_RANGE;

  if (card_minor > DMM_MINOR_LARGEST) {
    printk(KERN_ERR DMM_ID "open called on bogus minor number %u!", real_minor);
    return -ENODEV;
  }

  DPRINTK("trying to open card %d, %s (minor %u)\n", card, dmm_devices_by_minor[card_minor], real_minor);


  // If we check to make sure the card is valid here, we dont have to
  // anywhere else.
  if (dmm_card[card].in_use[0] == -1) {
    DPRINTK("card not available\n");
    return -ENODEV;
  }


  // Everything looks good, go ahead and open.
  switch (card_minor) {
    case DMM_MINOR_ADC_0:
    case DMM_MINOR_ADC_1:
    case DMM_MINOR_ADC_2:
    case DMM_MINOR_ADC_3:
    case DMM_MINOR_ADC_4:
    case DMM_MINOR_ADC_5:
    case DMM_MINOR_ADC_6:
    case DMM_MINOR_ADC_7:
    case DMM_MINOR_ADC_8:
    case DMM_MINOR_ADC_9:
    case DMM_MINOR_ADC_10:
    case DMM_MINOR_ADC_11:
    case DMM_MINOR_ADC_12:
    case DMM_MINOR_ADC_13:
    case DMM_MINOR_ADC_14:
    case DMM_MINOR_ADC_15:
    case DMM_MINOR_ADC_BATCH:
 if ((file->f_flags & O_ACCMODE) != O_RDONLY) {
   printk(KERN_WARNING DMM_ID "ADCs can only be opened read-only, silly\n");
   return -EPERM;
 }

 break;


    case DMM_MINOR_DAC_0:
    case DMM_MINOR_DAC_1:
    case DMM_MINOR_DAC_2:
    case DMM_MINOR_DAC_3:
```

```
if ((file->f_flags & O_ACCMODE) != O_WRONLY) {
  printk(KERN_WARNING DMM_ID "DACs can only be opened write-only, silly\n");
  return -EPERM;
}
break;


    case DMM_MINOR_DIGITAL:
    case DMM_MINOR_CTL:
// FIXME: maybe we need to check read/write open stuff here, or
// maybe not..
break;


    default:
      printk(KERN_WARNING DMM_ID "that device is not implemented yet, please call back later\n");
return -ENOSYS;
  }


  DPRINTK("open of card %d, %s suceeds\n", card, dmm_devices_by_minor[card_minor]);

  dmm_card[card].in_use[card_minor] = 1;

  MOD_INC_USE_COUNT;


  isr_runlimit = 500;
  return 0;
}


//
//     Name: dmm_close()
//
// Function: Releases a Diamond-MM resource.
//
// Arguments: Hmm....
//
//  Returns: None.
//
static int dmm_release(
  struct inode *inode,
  struct file *file
) {
  unsigned int real_minor, card_minor;
  int card;


  real_minor = MINOR(inode->i_rdev);

  card = real_minor / DMM_MINOR_RANGE;
  card_minor = real_minor % DMM_MINOR_RANGE;


  DPRINTK("closing card %d, %s (minor %u)\n", card, dmm_devices_by_minor[card_minor], real_minor);
  dmm_card[card].in_use[card_minor] = 0;

  MOD_DEC_USE_COUNT;


  return 0;
}


static struct file_operations dmm_fops = {
  NULL,           // llseek
  dmm_read,       // read
  dmm_write,      // write
  NULL,           // readdir
  NULL,           // poll FIXME: implement
  dmm_ioctl,      // ioctl
  NULL,           // mmap FIXME: implement
  dmm_open,       // open
  NULL,           // flush
  dmm_release,    // release
  NULL,           // fsync
  NULL,           // fasync
  NULL,           // check_media_change
  NULL,           // revalidate
  NULL            // lock
};



//
// This is used to probe for cards.
//
static int probe_isr_ran;


//
//     name: dmm_probe_isr()
//
// function: This is the prober interrupt service routine for the
//           Diamond-MM driver. It sets a global variable to indicate
//           it got called.
//
// arguments: The IRQ that triggered it, and some other stuff.
//
//  returns: Nothing.
//
static void dmm_probe_isr(
  int trigger_irq,
  void *dev_id,
  struct pt_regs *regs
){
  DPRINTK("dmm probe isr running (irq=%d)\n", trigger_irq);
  probe_isr_ran = 1;
}
```

```
//     name:  dmm_select_irq()
//
//  function:  Selects an IRQ for the driver to use, or selects a polling
//             method.
//
// arguments:  None.
//
//   globals:  Examines and sets the 'irq' variable which holds the IRQ that
//             will be used (shared by all cards), or -1 meaning 'autoselect
//             a good IRQ now', or -2 or -3 meaning 'use polling I/O, not
//             interrupts'.
//
//   returns:  0 if it selected a good IRQ, and -EFOO if there was a problem.
//
static int dmm_select_irq(void) {
  int i;

  if ((irq >= 2) && (irq <= 7)) {
    DPRINTK("requesting IRQ %d\n", irq);
    if (request_irq(irq, dmm_probe_isr, SA_INTERRUPT, "dmm", NULL)) {
 printk(KERN_ERR DMM_ID "specified IRQ %d not available\n", irq);
 return -EBUSY;
    }
  } else if (irq == -1) {
    // autoselect a useful IRQ
    DPRINTK("autoselecting a good IRQ\n");

    for (i = 0; i < 6; i ++) {
 DPRINTK("trying IRQ %d\n", try_irq[i]);
 if (request_irq(try_irq[i], dmm_probe_isr, SA_INTERRUPT, "dmm", NULL)) {
   DPRINTK("IRQ %d not available\n", try_irq[i]);
   continue;
 }
 irq = try_irq[i];
 DPRINTK("using IRQ %d\n", irq);
 break;
    }

    if (irq == -1) {
 // no suitable IRQ was found, fall back to fast polling
 printk(KERN_ERR DMM_ID "no available IRQ found, using fast polling I/O\n");
 irq = -3;
    }
  } else if (irq == -2) {
    // use slow polling I/O
    DPRINTK("using slow polling I/O\n");
  } else if (irq == -3) {
    // use fast polling I/O
    DPRINTK("using fast polling I/O\n");
  } else {
    printk(KERN_ERR DMM_ID "invalid IRQ %d specified, bailing out\n", irq);
    return -EINVAL;
  }

  return 0;
}
```

```
//
//     name:  dmm_probe_for_cards()
//
//  function:  Probes for cards.  Configures all the cards it finds.  If
//             any cards are found, installs the driver ISR.
//
// arguments:  None.
//
//   globals:  Sets dmm_card[] to hold whatever cards are found.  Sets
//             num_cards to be the number of cards found.
//
//   returns:  0 if all went well, and -EFOO if there was a problem.
//
static int dmm_probe_for_cards(void) {
  int probe_io, i;


  DPRINTK("probing for cards...\n");

  num_cards = 0;

  for (probe_io = DMM_MIN_PROBE_IO; probe_io <= DMM_MAX_PROBE_IO; probe_io += 0x40) {
    DPRINTK("probing for card at io=0x%03X\n", probe_io);

    // claim some i/o ports
    if (check_region(probe_io, 16) == -EBUSY) {
 DPRINTK("I/O ports not available, skipping\n");
 continue;
    }
    request_region(probe_io, 16, "dmm");
    dmm_card[num_cards].io = probe_io;

    // configure this card for IRQs
    SELECT_INTERRUPT_NUMBER(num_cards, irq);
    ENABLE_INTERRUPTS(num_cards);

    // prepare the card for an ADC conversion...
    // This might be another good place to check more thoroughly... (JDB)
    CLEAR_INTERRUPT_FLIP_FLOP(num_cards);
    SELECT_ADC_RANGE(num_cards, 0, 0);

    probe_isr_ran = 0;
    dmm_card[num_cards].wait_q = NULL;

    START_ADC_CONVERSION(num_cards);
    interruptible_sleep_on_timeout(&dmm_card[num_cards].wait_q, 2);

    // here we either got woken by ISR or by timeout, or we got a signal
    // FIXME: how do you tell apart timeout and signal??

    if (!probe_isr_ran) {
 // the card isnt there or isnt responding
 DPRINTK("card not detected\n");
 release_region(dmm_card[num_cards].io, 16);
 continue;  // try the next I/O address
    }
    // the card is present!
    DPRINTK("card detected!\n");
    CLEAR_INTERRUPT_FLIP_FLOP(num_cards);

    // finish configuring it
    DISABLE_DMA(num_cards);        // DMA not supported yet...
    DISABLE_HW_TRIGGER(num_cards); //  do not use hardware trigger

    //  Initialize the ADC semaphore.
    dmm_card[num_cards].sem = MUTEX;

    //  Set card to differential mode.  This should be settable via an IOCTL.
    dmm_card[num_cards].channels = 8;

    //  Note that all parts of the card are idle, and mark the card as
    //  present in the system.
    for (i = 0; i < DMM_MINOR_LARGEST; i ++) {
dmm_card[num_cards].in_use[i] = 0;
    }

    //  Done with this card.
    printk(KERN_INFO DMM_ID "card %d (io=0x%03X) initialized\n", num_cards, dmm_card[num_cards].io);

    // print the cards state (should be sane)
    PRINT_CHANNEL_REGISTER(num_cards);
    PRINT_STATUS_REGISTER(num_cards);
    PRINT_CONTROL_REGISTER(num_cards);
    PRINT_SPECIAL_READBACK_REGISTER(num_cards);

    num_cards ++;
    if (num_cards == DMM_MAX_CARDS) {
DPRINTK("found %d cards, ending probe\n", DMM_MAX_CARDS);
break;
    }
  }

  free_irq(irq, NULL);
  if (request_irq(irq, dmm_isr, SA_INTERRUPT, "dmm", NULL)) {
    DPRINTK("couldnt change ISR, bailing\n");
    for (i = 0; i < num_cards; i ++) {
DPRINTK("card %d releasing 16 I/O ports at 0x%03x\n", i, dmm_card[i].io);
release_region(dmm_card[i].io, 16);
    }
    return -EBUSY;
  }

  return 0;
}


//
//     Name:  init_module()
//
//  Function:  Detects and initializes the Diamond-MM card(s), and
//             initializes the driver.
//
```

```
// Arguments: None.
//
//  Returns: 0 if all went well, and some E* if not.
//
int init_module(void) {
  int card, ret_val;
  int i;


  printk(KERN_INFO DMM_VERSION ", loading...\n");


  // This is the secret handshake for 'this card is missing'. We'll mark
  // it as present if we succeed in initializing it.
  for (card = 0; card < DMM_MAX_CARDS; card ++) {
    dmm_card[card].in_use[0] = -1;
  }


  // deal with irqs
  ret_val = dmm_select_irq();
  if (ret_val) {
    return ret_val;
  }

  card = 0;
  if (irq > 0) {
    // we can probe for cards
    ret_val = dmm_probe_for_cards();
    if (ret_val) {
 return ret_val;
    }
  } else {
    // initialize the cards from user parameters
    for (card = 0; card < DMM_MAX_CARDS; card ++) {
 if (io[card] == 0) {
   continue;
 }

 DPRINTK("initializing card %d at io=0x%03X\n", num_cards, io[card]);

 // claim some i/o ports
 DPRINTK("card %d requesting 16 I/O ports at 0x%03X\n", num_cards, io[card]);
 if (check_region(io[card], 16) == -EBUSY) {
   DPRINTK("I/O ports not available, skipping card\n");
   continue;
 }

 dmm_card[num_cards].io = io[card];
 request_region(dmm_card[num_cards].io, 16, "dmm");


 // Initialize the ADC semaphore and wait queue variables.
 dmm_card[num_cards].sem = MUTEX;
 dmm_card[num_cards].wait_q = NULL;

 // configure this card to not use IRQs
```

```
      // Check here more... (JDB)
DISABLE_INTERRUPTS(num_cards);

// Configure the card.
CLEAR_INTERRUPT_FLIP_FLOP(num_cards);
DISABLE_DMA(num_cards);        // DMA not supported yet...
DISABLE_HW_TRIGGER(num_cards); // do not use hardware trigger


      // Set card to differential mode. This should be settable via an IOCTL.
      dmm_card[card].channels = 8;

// Note that all parts of the card are idle, and mark the card as
// present in the system.
for (i = 0; i < DMM_MINOR_LARGEST; i ++) {
  dmm_card[num_cards].in_use[i] = 0;
}

// Done with this card.
printk(KERN_INFO DMM_ID "card %d (io=0x%03X) initialized\n", num_cards, dmm_card[num_cards].io);

// print the cards state (should be sane)
PRINT_CHANNEL_REGISTER(num_cards);
PRINT_STATUS_REGISTER(num_cards);
PRINT_CONTROL_REGISTER(num_cards);
PRINT_SPECIAL_READBACK_REGISTER(num_cards);

num_cards ++;
    }
  }


  if (num_cards == 0) {
    printk(KERN_ERR DMM_ID "no cards found, aborting\n");

    if (irq > 0) {
DPRINTK("releasing IRQ %d\n", irq);
free_irq(irq, NULL);
    }

    return -EBUSY;
  }


  // register the device with the kernel
  DPRINTK("registering driver, major number %d\n", DMM_MAJOR);
  if(register_chrdev(DMM_MAJOR, "dmm", &dmm_fops)) {
    printk(KERN_ERR DMM_ID "unable to get major number %d\n", DMM_MAJOR);
    return -EBUSY;
  }

  if (irq > 0) {
    printk(KERN_INFO DMM_ID "using IRQ %d\n", irq);
  } else if (irq == -2) {
    printk(KERN_INFO DMM_ID "using slow polling I/O\n");
  } else if (irq == -3) {
    printk(KERN_INFO DMM_ID "using fast polling I/O\n");
  }

  DPRINTK("driver initialized\n");


  return 0;
}




/****************************************************************************/
/*                                                                          */
/*     Name:  cleanup_module()                                              */
/*                                                                          */
/*  Function:  Shuts down the cards and cleans up after the driver.         */
/*                                                                          */
/* Arguments:  None.                                                        */
/*                                                                          */
/*   Returns:  None.                                                        */
/*                                                                          */
/****************************************************************************/
void cleanup_module(void) {
  int card;


  if (irq > 0) {
    DPRINTK("releasing IRQ %d\n", irq);
    free_irq(irq, NULL);
  }


  for (card = 0; card < num_cards; card ++) {
    DPRINTK("card %d releasing 16 I/O ports at 0x%03x\n", card, dmm_card[card].io);
    release_region(dmm_card[card].io, 16);

    printk(KERN_INFO DMM_ID "card %d (io=0x%03X) released\n", card, dmm_card[card].io);
  }

  DPRINTK("unregistering device\n");
  if(unregister_chrdev(DMM_MAJOR, "dmm")) {
    printk(KERN_ERR DMM_ID "device unregistering failed\n");
  } else {
    printk(KERN_INFO "Diamond-MM driver unloaded\n");
  }
}
```

==

```
/*
 * This file is part of the dmm16.o source, which is based on
 * Sebastian Kuzminski's dmm12 driver, released under the GNU General
 * Public License.
 */

#ifndef __DMM_H
#define __DMM_H
```

```
#include <linux/ioctl.h>
#include "config.h"


/*
 * This array maps minor numbers to device identifier strings.
*/
const char *dmm_devices_by_minor[] = {
  "ADC 0",
  "ADC 1",
  "ADC 2",
  "ADC 3",
  "ADC 4",
  "ADC 5",
  "ADC 6",
  "ADC 7",
  "ADC 8",
  "ADC 9",
  "ADC 10",
  "ADC 11",
  "ADC 12",
  "ADC 13",
  "ADC 14",
  "ADC 15",
  "DAC 0",
  "DAC 1",
  "DAC 2",
  "DAC 3",
  "digital I/O",
  "ADC Batch",
  "control"
};


#define DMM_IOCTL_SET_ANALOG_CODE     _IOR(DMM_MAJOR, 0, int16_t)
#define DMM_IOCTL_GET_ANALOG_CODE     _IOW(DMM_MAJOR, 1, int16_t)
#define DMM_IOCTL_SET_SAMPLE_RATE     _IOR(DMM_MAJOR, 2, int16_t)
#define DMM_IOCTL_GET_SAMPLE_RATE     _IOW(DMM_MAJOR, 3, int16_t)
#define DMM_IOCTL_SET_BATCH_CHANNELS  _IOR(DMM_MAJOR, 4, int16_t)
#define DMM_IOCTL_GET_BATCH_CHANNELS  _IOR(DMM_MAJOR, 5, int16_t)
#define DMM_IOCTL_BATCH_START         _IOR(DMM_MAJOR, 7, int16_t) // Not impl
#define DMM_IOCTL_BATCH_STOP          _IOR(DMM_MAJOR, 8, int16_t) // Not impl

const int dmm_ioctls[] = { DMM_IOCTL_SET_ANALOG_CODE,
                 DMM_IOCTL_GET_ANALOG_CODE,
                 DMM_IOCTL_SET_SAMPLE_RATE,
                 DMM_IOCTL_GET_SAMPLE_RATE,
                 DMM_IOCTL_SET_BATCH_CHANNELS,
                 DMM_IOCTL_GET_BATCH_CHANNELS,
                 DMM_IOCTL_BATCH_START,
                 DMM_IOCTL_BATCH_STOP };

#endif /* __DMM_H */

=========================================================================
/*
 * This file is part of the dmm16.o source, which is based on
 * Sebastian Kuzminski's dmm12 driver, released under the GNU General
 * Public License.
 */

#ifndef __DMM_INTERNAL_H
#define __DMM_INTERNAL_H

#include "config.h"

/****************************************************************************/
/*                                          */
/* The "version" string gets printed at init-time (module load time).    */
/*                                          */
/*                                          */
#define DMM_VERSION  "Diamond-MM-16 driver, version 01, January 12, 2001"

/****************************************************************************/
/*                                          */
/* The debug print macros.                            */
/*                                          */
/*                                          */
#ifdef DMM_DEBUG
  #define DPRINTK(fmt, args...) printk(KERN_DEBUG DMM_ID fmt, ## args)
#else
  #define DPRINTK(fmt, args...)
#endif

/****************************************************************************/
/*                                          */
/* This string is prepended to everything the DMM driver writes.         */
/*                                          */
/*                                          */
#define DMM_ID  "dmm16: "

/****************************************************************************/
/*                                          */
/* Number of minor numbers per I/O card.                       */
/*                                          */
/*                                          */
#define DMM_MINOR_LARGEST 22

/****************************************************************************/
/*                                          */
/* Alignment size, or number of minor numbers allocated to each I/O card.  */
/*                                          */
/*                                          */
#define DMM_MINOR_RANGE 32
/****************************************************************************/

/*                                          */
/* The minor number assignments themselves.                     */
/*                                          */
/*                                          */
```

```
#define DMM_MINOR_ADC_0   0
#define DMM_MINOR_ADC_1   1
#define DMM_MINOR_ADC_2   2
#define DMM_MINOR_ADC_3   3
#define DMM_MINOR_ADC_4   4
#define DMM_MINOR_ADC_5   5
#define DMM_MINOR_ADC_6   6
#define DMM_MINOR_ADC_7   7
#define DMM_MINOR_ADC_8   8
#define DMM_MINOR_ADC_9   9
#define DMM_MINOR_ADC_10  10
#define DMM_MINOR_ADC_11  11
#define DMM_MINOR_ADC_12  12
#define DMM_MINOR_ADC_13  13
#define DMM_MINOR_ADC_14  14
#define DMM_MINOR_ADC_15  15
#define DMM_MINOR_DAC_0   16
#define DMM_MINOR_DAC_1   17
#define DMM_MINOR_DAC_2   18
#define DMM_MINOR_DAC_3   19
#define DMM_MINOR_DIGITAL  20
#define DMM_MINOR_ADC_BATCH 21
#define DMM_MINOR_CTL     22


//
// The default I/O addresses and IRQ of the cards.
//
#define DMM_MIN_PROBE_IO 0x100
#define DMM_MAX_PROBE_IO 0x3C0

// Moved to config.h... -JDB
// #define DMM_IRQ    -1 // This means, 'pick a good one'.

/****************************************************************************/
/*                                                                          */
/* The I/O space offsets of the various card features you write to.        */
/*                                                                          */
/*                                                                          */
#define DMM_OFFSET_START_ADC_CONVERSION 0
#define DMM_OFFSET_DAC_ALL_LSB        1
#define DMM_OFFSET_ADC_CHANNEL        2
#define DMM_OFFSET_DIGITAL_OUTPUT     3
#define DMM_OFFSET_DAC_0_MSB          4
#define DMM_OFFSET_DAC_1_MSB          5
#define DMM_OFFSET_DAC_2_MSB          6
#define DMM_OFFSET_DAC_3_MSB          7
#define DMM_OFFSET_CLEAR_INTERRUPT    8
#define DMM_OFFSET_CONTROL            9
#define DMM_OFFSET_CT_ENABLE_SELECT   10
#define DMM_OFFSET_RANGE_CONFIG       11
#define DMM_OFFSET_CT_0_DATA          12
#define DMM_OFFSET_CT_1_DATA          13
#define DMM_OFFSET_CT_2_DATA          14
#define DMM_OFFSET_CT_CONTROL         15

/****************************************************************************/
```

```
/*                                              */
/* The I/O space offsets of the card features you read from.          */
/*                                              */
/*                                              */
#define DMM_OFFSET_ADC_LSB        0
#define DMM_OFFSET_ADC_MSB        1
#define DMM_OFFSET_ADC_CHANNEL     2
#define DMM_OFFSET_DIGITAL_INPUT   3
#define DMM_OFFSET_DAC_ALL_UPDATE  4
#define DMM_OFFSET_STATUS         8
#define DMM_OFFSET_CONTROL_READBACK 9
#define DMM_OFFSET_SPECIAL_READBACK 11
#define DMM_OFFSET_CT_0_DATA      12
#define DMM_OFFSET_CT_1_DATA      13
#define DMM_OFFSET_CT_2_DATA      14
#define DMM_OFFSET_CT_CONTROL      15

/***************************************************************************/
/*                                              */
/* The values of the bits of the Status register.                  */
/*                                              */
/*                                              */
#define STATUS_ADC_BUSY        0x80
#define STATUS_UNIPOLAR        0x40
#define STATUS_SINGLE_ENDED     0x20
#define STATUS_INTERRUPT_PENDING 0x10

/***************************************************************************/
/*                                              */
/* This struct holds all the info about a DMM card.                  */
/*                                              */
/*                                              */
struct dmm_card_t {
  int io; // FIXME: should this be some other data type?
  char in_use[DMM_MINOR_LARGEST + 1];
  struct semaphore sem;
  struct wait_queue *wait_q;

  volatile int need_start_conversion;

  // FIXME:  which of these really need to be volatile?
  //   conversion_happened worked without...
  volatile int count;
  unsigned short buffer[DMM_BUFFER_SIZE];
  volatile int head;
  volatile int tail;

  int batch_channel_high;
  int batch_channel_low;
  int channels; // 16 for single-ended, 8 for differential
  int scan; // 1 for scan mode
};

/***************************************************************************/
/*                                              */
/* A macro to determine if a circular buffer is full.                */
/* h = head, t = tail, n = next = how far the next scan will take us       */
/*                                              */
#define FULL(h, t, n) !((n>=t && t>=h) || (t>=h && h>n) || (h>n && n>=t))



/****************************************************************************/
/*                                              */
/*  Macros to do various handy things to the cards.                        */
/*                                              */
/*                                              */
#define CLEAR_INTERRUPT_FLIP_FLOP(card) \
  outb(0x00, dmm_card[card].io + DMM_OFFSET_CLEAR_INTERRUPT);

#define DISABLE_INTERRUPTS(card) \
  DPRINTK("card %d disabling interrupts\n", card); \
            outb(inb(dmm_card[card].io    +    DMM_OFFSET_CONTROL)    &    0x7F,    dmm_card[card].io    +
DMM_OFFSET_CONTROL);

#define ENABLE_INTERRUPTS(card) \
  DPRINTK("card %d enabling interrupts\n", card); \
  outb(inb(dmm_card[card].io + DMM_OFFSET_CONTROL) | 0x80, dmm_card[card].io + DMM_OFFSET_CONTROL);

#define SELECT_INTERRUPT_NUMBER(card, irq) \
  DPRINTK("card %d selecting interrupt %d\n", card, irq); \
      outb(((inb(dmm_card[card].io  +  DMM_OFFSET_CONTROL)  &  0x8F)  |  (irq  <<  4)),  dmm_card[card].io  +
DMM_OFFSET_CONTROL);

#define DISABLE_DMA(card) \
  DPRINTK("card %d disabling dma\n", card); \
            outb(inb(dmm_card[card].io    +    DMM_OFFSET_CONTROL)    &    0xFB,    dmm_card[card].io    +
DMM_OFFSET_CONTROL);

#define ENABLE_DMA(card) \
  DPRINTK("card %d enabling dma\n", card); \
  outb(inb(dmm_card[card].io + DMM_OFFSET_CONTROL) | 0x04, dmm_card[card].io + DMM_OFFSET_CONTROL);

#define DISABLE_HW_TRIGGER(card) \
  DPRINTK("card %d using software ADC trigger\n", card); \
            outb(inb(dmm_card[card].io    +    DMM_OFFSET_CONTROL)    &    0xFD,    dmm_card[card].io    +
DMM_OFFSET_CONTROL);

#define ENABLE_HW_TRIGGER(card) \
  DPRINTK("card %d using hardware ADC trigger\n", card); \
  outb(inb(dmm_card[card].io + DMM_OFFSET_CONTROL) | 0x02, dmm_card[card].io + DMM_OFFSET_CONTROL);

#define SELECT_INTERNAL_HW_ADC_TRIGGER(card) \
  DPRINTK("card %d using internal HW ADC trigger\n", card); \
  outb(inb(dmm_card[card].io + DMM_OFFSET_CONTROL) | 0x01, dmm_card[card].io + DMM_OFFSET_CONTROL);

#define SELECT_EXTERNAL_HW_ADC_TRIGGER(card) \
  DPRINTK("card %d using external HW ADC trigger\n", card); \
            outb(inb(dmm_card[card].io    +    DMM_OFFSET_CONTROL)    &    0xFE,    dmm_card[card].io    +
DMM_OFFSET_CONTROL);

// FIXME: SELECT_ADC_RANGE should wait for the lines to stabilize
#define SELECT_ADC_RANGE(card, high, low) \
  DPRINTK("card %d selecting ADC range 0x%02x\n", card, (high << 4) + low); \
  outb(((high << 4) + low), dmm_card[card].io + DMM_OFFSET_ADC_CHANNEL);

#define GET_CHANNEL_REGISTER(card) \
  inb(dmm_card[card].io + DMM_OFFSET_ADC_CHANNEL)

#define GET_STATUS_REGISTER(card) \
  inb(dmm_card[card].io + DMM_OFFSET_STATUS)

#define GET_CONTROL_REGISTER(card) \
  inb(dmm_card[card].io + DMM_OFFSET_CONTROL)

#define SET_CONTROL_REGISTER(card, value) \
  outb(value, dmm_card[card].io + DMM_OFFSET_CONTROL)

#define GET_SPECIAL_READBACK_REGISTER(card) \
  inb(dmm_card[card].io + DMM_OFFSET_SPECIAL_READBACK)

#define START_ADC_CONVERSION(card) \
  outb(0x00, dmm_card[card].io + DMM_OFFSET_START_ADC_CONVERSION);

#define READ_ADC(card)                              \
  inw(dmm_card[card].io + DMM_OFFSET_ADC_LSB)

#define SET_ADC_CODE(card, code) \
  DPRINTK("card %d selecting ADC code %d\n", card, code); \
  outb(((inb(dmm_card[card].io + DMM_OFFSET_RANGE_CONFIG) & 0xF0) | \
    (code & 0x0F)), dmm_card[card].io + DMM_OFFSET_RANGE_CONFIG);

#define GET_ADC_CODE(card) \
  inb(dmm_card[card].io + DMM_OFFSET_RANGE_CONFIG) & 0x0F

#define LSB(value) value & 0xFF

#define MSB(value) (value >> 8) & 0xFF

/* See 82c54 for the meaning of the control bits. */
#define PROGRAM_COUNTER_0(card, value) \
  DPRINTK("card %d programming counter 0 to value %d\n", card, value); \
  outb(0x34, dmm_card[card].io + DMM_OFFSET_CT_CONTROL); \
  outb(LSB(value), dmm_card[card].io + DMM_OFFSET_CT_0_DATA); \
  outb(MSB(value), dmm_card[card].io + DMM_OFFSET_CT_0_DATA);

#define PROGRAM_COUNTER_1(card, value) \
  DPRINTK("card %d programming counter 1 to value %d\n", card, value); \
  outb(0x74, dmm_card[card].io + DMM_OFFSET_CT_CONTROL); \
  outb(LSB(value), dmm_card[card].io + DMM_OFFSET_CT_1_DATA); \
  outb(MSB(value), dmm_card[card].io + DMM_OFFSET_CT_1_DATA);

#define PROGRAM_COUNTER_2(card, value) \
  DPRINTK("card %d programming counter 2 to value %d\n", card, value); \
  outb(0xB4, dmm_card[card].io + DMM_OFFSET_CT_CONTROL); \
  outb(LSB(value), dmm_card[card].io + DMM_OFFSET_CT_2_DATA); \
  outb(MSB(value), dmm_card[card].io + DMM_OFFSET_CT_2_DATA);

/****************************************************************************/
/*                                                  */
/* Macros to show the internal state of the cards.                */
/*                                          */
/*                                          */
#define PRINT_CHANNEL_REGISTER(card) \
  DPRINTK("card %d channel register 0x%02x\n", card, GET_CHANNEL_REGISTER(card));

#define PRINT_STATUS_REGISTER(card) \
  DPRINTK("card %d status register 0x%02x\n", card, GET_STATUS_REGISTER(card));

#define PRINT_CONTROL_REGISTER(card) \
  DPRINTK("card %d control register 0x%02x\n", card, GET_CONTROL_REGISTER(card));

#define PRINT_SPECIAL_READBACK_REGISTER(card) \
  DPRINTK("card %d special readback register 0x%02x\n", card, GET_SPECIAL_READBACK_REGISTER(card));



#endif /* not __DMM_INTERNAL_H */

==============================================================================

/*
 * This file is part of the dmm16.o source, which is based on
 * Sebastian Kuzminski's dmm12 driver, released under the GNU General
 * Public License.
 */

#ifndef __DMM_CONFIG__
#define __DMM_CONFIG__

/*
 *  This is the maximum number of cards that the driver can deal with.
 *  This should be 1 or greater... The smaller this is, the less
 *  kernel memory the driver uses (per-card sample-buffers are statically
 *  allocated).
 *
 *  (I changed this from 2 to 1 since I have made modifications that
 *  might break multiple-card support... -JDB)
 */
#define DMM_MAX_CARDS 1



/*
 *  The default interrupt to use.
 *     3-7, 10-12 specify the IRQ directly
 *     -1 means autoselect a good IRQ
 *     -2 means use slow polling I/O (produces low CPU load)
 *     -3 means use fast polling I/O (produces high CPU load)
 *
 * Auto-select doesn't seem to work for me, but specifying 5 does
 * work.  -JDB
 */
#define DMM_IRQ  5


/*
 *  Define this to enable debug printing from the driver.  When compiled with
 * this flag, the driver will be quite verbose.
*/
#undef DMM_DEBUG
//#define DMM_DEBUG


/*
 * This is the major number of the driver. Do not change it unless you
 * know what you are doing!
*/
#define DMM_MAJOR 61


/*
 * The size (in samples) of the per-card sample buffers. Each sample takes
 * 2 bytes to store.
*/
#define DMM_BUFFER_SIZE 4096
//#define DMM_BUFFER_SIZE 2048
//#define DMM_BUFFER_SIZE 1024


//
// When autoselecting IRQs, try them in this order.
//
int try_irq[] = { 7, 5, 4, 3, 6, 2 };


#endif /* __DMM_CONFIG__ */
================================================================================

/*
 * This code is taken from <linux/version.h> and allows me to compile
 * on one kernel and run on another without using insmod -f and
 * getting warning messages. It's a terrible hack, and I'm sure there
 * must be a better way to do it... It is part of the source for
 * dmm16.o, but was not part of Sebastian Kuzminski's dmm.o source.
 */

#ifndef __rh_kernel_version_h__
#define __rh_kernel_version_h__

/*
 * First, get the version string for the running kernel from
 * /boot/kernel.h - initscripts should create it for us
 */

#include "/boot/kernel.h"

#define UTS_RELEASE "2.2.14-5.0-EMBED-2"
#define LINUX_VERSION_CODE 131598
#define KERNEL_VERSION(a,b,c) (((a) << 16) + ((b) << 8) + (c))

#endif /* __rh_kernel_version_h__ */
=================================================================
# This file is part of the dmm16.o source, which is based on Sebastian
# Kuzminski's dmm12 driver, released under the GNU General Public
# License.

MOD_DIR = /lib/modules/`uname -r`/misc

INC_DIR = /usr/local/include

CC=gcc
CFLAGS=-Wall -O2 -I/usr/src/linux/include -DMODULE -DMODVERSIONS -DEXPORT_SYMTAB -D__KERNEL__


all: dmm16.o d16read d16write d16ctl


# install: all
# mkdir -p $(MOD_DIR)
# mkdir -p $(INC_DIR)
# install -o root -g root -m 644 dmm16.o $(MOD_DIR)
# install -o root -g root -m 644 dmm16.h $(INC_DIR)


# Doesn't really install; used for convenience during development...
install: all
cd .. && rm -f dmm16.tar.gz && tar cvzpf dmm16.tar.gz --exclude=*~ dmm16

dmm16.o: dmm16.c dmm16.h dmm16-internal.h config.h Makefile
$(CC) $(CFLAGS) -c -o dmm16.o dmm16.c

# A program used for testing direct I/O port programming of the DMM16
morgoth: morgoth.c
$(CC) $(CFLAGS) -o morgoth morgoth.c

# A user space program for acquiring data from the DMM16 by using the
# dmm16.o driver
d16read: d16read.c
$(CC) $(CFLAGS) -o d16read d16read.c

# A user space program for writing data to the DMM16 by using the
# dmm16.o driver
d16write: d16write.c
$(CC) $(CFLAGS) -o d16write d16write.c

# A user space program for controlling the DMM16 using IOCTLs of the
# dmm16.o driver
d16ctl: d16ctl.c dmm16.h config.h
$(CC) $(CFLAGS) -o d16ctl d16ctl.c

purge:  clean
rm -f $(MOD_DIR)/dmm16.o $(INC_DIR)/dmm16.h

clean:
rm -f dmm16.o morgoth d16read d16write d16ctl
```

APPENDIX

B

THE COUHES DOCUMENT

B.1 The COUHES Documentation

This section provides the document that has been submitted to the Committee on the Use of Humans as Experimental Subjects (COUHES) at MIT to use human subjects during the MICR0-G flight experiment aboard the KC-135 aircraft in January 2001. This document got approval from the COUHES committee on December 14, 2000.

Application Number: 2718

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Committee on the Use of Humans as Experimental Subjects

*Application for Approval to Use Humans as Experimental Subjects

DATE: December 13, 2000

PART I.

Title of Study: Microgravity Investigation and Crew Reactions in 0-G (MICR0-G)

Principal investigator: Dava J. Newman

Department: Department of Aeronautics and Astronautics

Room No.: Bldg. 33, Room 307

E-mail address: dnewman@mit.edu

Telephone No.: (617) 258-8799

Associated Investigator:

Antonio Pedotti, Professor, Politecnico di Milano University, Italy.

Sylvie Loday, S.M. candidate, Dept. Electrical Engineering and Computer Science, MIT.

Collaborating Institution(s), if applicable: NASA, Politecnico di Milano University.

(Please attach copies of approval documents or correspondence from collaborating institution(s) where applicable)

Financial Support: NASA, PI: Professor Newman, Account Number 66820

(Include title of research grant, granting agency and award number, if any; if not applicable, please indicate how project will be financed.)

Purpose of Study:

The purpose of this study is to test aboard the KC-135 aircraft, an integrated ground-based technology assessing crew-induced disturbances and crew motor behavior in microgravity. Subjects will use special mobility aids instrumented with strain gages to record applied forces and moments. An opto-electronic motion analyzer will capture the subject's motions and will synchronize its data with the force and moment data acquired by the sensors. Quantitative analysis of human performance in microgravity is important for both scientific investigations and spacecraft engineering design. By collecting and evaluating the kinematic and kinetic data of astronauts in space, it becomes possible to characterize human motor strategies, postural behavior in weightlessness, improve the design of orbital modules, help maintain a quiescent microgravity environment for acceleration-sensitive science experiments, and optimize the human operative capabilities during long-duration space missions. This experiment will test a ground-based prototype acquiring kinetic and kinematic data in the reduced gravity environment of the KC-135, with the objective to further develop a space-qualified version for the International Space Station (ISS).

PART II.

Motivation for the Experiment:

The Department of Aeronautics and Astronautics of the Massachusetts Institute of Technology and the Department of Bioengineering of the Politecnico di Milano University developed systems respectively, to measure crew-induced reaction forces and moments and to conduct a quantitative kinematics analysis of crew intra-vehicular activity (IVA) in space. Both, the U.S. EDLS (Enhanced Dynamic Load Sensors) and the Italian ELITE-S (Elaboratore di Immagini Televisive-S2) systems were flown successfully on the Russian orbital complex *Mir*. The Microgravity Investigation and Crew Reactions in 0-G (MICR0-G) research project is funded by NASA to develop enhanced ground-based versions of ELITE-S and EDLS and integrate both prototypes in a unique kinematic and kinetic measurement system. The ultimate goal, beyond the objectives of the MICR0-G project, is the development of an advanced space-qualified facility for multifactorial movement analysis on the ISS.

The development of both the kinematic and the kinetic advanced systems has been recently completed. The COUHES proposal describes the experiment proposed to test the integrated system during microgravity periods aboard the KC-135 aircraft.

Description of the System:

Advanced Kinetic Sensors

The key experiment hardware of the kinetic system is based on three types of sensors -- a handhold, a foot restraint and a touchpad, as shown on Figure B.1. The sensors provide the functionality of crew restraint devices and mobility aids while measuring the applied forces and moments in the x-, y-, and z- axes (6 degrees of freedom).

Each sensor is connected to an advanced electronics system processing and storing the acquired force and moment data. This electronics unit consists of two parts: (1) the *Sensor Elec-*

tronics Unit (SEU) and (2) The *Central Computer* (CC). The Sensor Electronics Unit is a box housing all the necessary electronics for data processing and networking -- namely a signal conditioning board, an A/D converter, a CPU, a PC-card module, a wireless PC-card, and power supply components. Figure B.2 shows the electronics inside the SEU box apart from the power supply components. Each sensor is connected via a cable to a corresponding SEU. The Central Computer is used as a server for networking and incorporates the storage devices to store all the data recorded by the sensors. This base station communicates wirelessly with the whole set of sensors and provides the main user interface with the system through a laptop computer display. The advanced sensors' architecture is illustrated in Figure B.3.

Enhanced Kinetic System: ELITE-S

The kinematic measurement component of the MICR0-G system consists of a versatile opto-electronic motion analyzer, called ELITE-S, based on passive markers recognition. The system is composed of up to 8 TV Cameras (TVC) connected through a bus with an IBM Thinkpad 760/770 laptop. The architecture of the system, as shown in Figure B.4, has two levels: Level 1 consists of the *Interface with The Environment* (ITE) and of a *Fast Processor for Shape Recognition* (FPSR). The components of the ITE are CCD video cameras as well as a set of reflective markers pasted on the subject. The identification of the markers by each camera is performed by a dedicated pattern recognition hardware—the FPSR. Level 2 of the architecture is a laptop computer with special software performing data processing in real-time.

The two systems are connected with a Universal Serial Bus (USB) to synchronize the acquisition of kinetic data of the advanced sensors with the kinematic data of the ELITE-S system.

Protocol

The experiment described in this section will take place on-board the KC-135 aircraft during microgravity sessions. Figure B.5 represents the experimental set-up within the aircraft main cabin. Due to volume constraints, we will use only two sensors (one foot restraint and one handhold) and two TV Cameras (TVC1 and TVC2). The two cameras will be 4 meters away from the working volume measuring about 1x2x2 meters. The foot restraint will be fixed on the floor and the handhold on a wall inside the working volume. The subject will perform different motions using the sensors and is asked to stay inside the working volume as much as possible. A list of motion tasks that can be performed is described in Table B.1 and will be provided in the consent form. Constraining our subjects to perform a set of pre-described tasks may lead to atypical motions and may result in biased kinetic and kinematic data. Consequently, we will instruct the subject to perform typical daily tasks and motions that he would naturally perform in a microgravity environment.

At least two operators are necessary to run the experiment. They will respectively work on the EDLS and ELITE-S laptop computers to verify that the acquisition software is running correctly on both systems and to modify parameters such as gains or resolution during the flight if necessary. The two experimenters, one test director and one systems operator, can stand between the two cameras and the working volume as long as they don't interfere with the field of view of the cameras.

PART III.

(Please answer each question below and indicate "NA" where not applicable to your application. Positive answers should be briefly explained with detailed information included in part II)

1. How will subjects be obtained?

Subjects are people involved in the development of the system (graduate students and professors) as well as two naive subjects.

Number of subjects needed?

A total of seven subjects will be involved in the flight experiments.

Age(s) of subjects? 21-55

2. Will women and minorities be recruited? Yes

3. Will subjects receive any payment or other compensation for participation? No

4. Will your subjects be studied outside MIT premises? Yes

The experiment will be performed on board the KC-135 aircraft as part of the Reduced Gravity Program, operated by the NASA Johnson Space Center (JSC), Houston, Texas.

5. Will the facilities of the Clinical Research Center be used? No

(If so, the approval of the CRC Advisory Committee is also required.)

For proposed investigations in social science, management, and other non-biomedical areas, please continue with question 9.

6. Will drugs be used? No (N/A)

Any Investigational New Drugs (IND)? No (N/A)

7. Will radiation or radioactive materials be employed? No (N/A)

If so, your study must also be approved by the Committee on Radiation Exposure to Human Subjects. Application forms are available from Mr. Francis X. Masse, Radiation Protection Of-

fice, 20C-207, x3-2180.

8. Will special diets be used? No (N/A)

If so, please state proposed duration(s).

9. Will subjects experience physical pain or stress?

Subjects may experience motion sickness during the flight because gravity aboard the KC-135 aircraft varies from microgravity to 2-g, but there is no risk to experience physical pain or stress due to interaction with the sensors.

10. Will a questionnaire be used? No

If so, please attach a copy.

11. Are personal interviews involved? No

If so, include an explanation in Part II and attach an outline.

12. Will subjects experience psychological stress? No

13. Does this study involve planned deception of subjects? No

14. Can information acquired through this investigation adversely affect a subject's relationships with other individuals (e.g. employee-supervisor, patient-physician, student-teacher, co-worker, family relationships)? No

15. Please explain how subject's anonymity will be protected, and/or confidentiality of data will be preserved.

Subjects will be filmed while performing the experiment, but the video system is based on system-recognition of reflective markers attached to the subject. Data acquired during the experiment will not be related to the subject involved. The anonymity of the subject is fully preserved.

PART IV.

A. Please summarize the risks to the individual subject and the benefits, if any; include any possible risk of invasion of privacy, embarrassment or exposure of sensitive or confidential data, and explain how you propose to deal with these risks.

The main risk from participating in this experiment is to fly aboard the KC-135. Subjects and researchers aboard the KC-135 aircraft may experience motion sickness during the flight. Symptoms include pallor, increased perspiration, nausea and vomiting. In attempt to avoid this syndrome, various medications have been used for many years but no one medicine has been perfect in preventing this condition. Each individual must decide for him/herself what to do. All flyers will have to discuss this matter with one of the NASA physician prior to flight. A medical officer is assigned to the flight and other medical facilities are available in the local area. In addition, any flyer must provide the results of a KC-135 Examination or equivalent FAA Third Class Aviation Physical signed by an FAA Medical Examiner. Finally, all flyers will receive a physiological training provided by NASA prior to flight.

B. Detection and reporting of harmful effects: if applicable, please describe what follow-up efforts will be made to detect harm to subjects, and how this committee will be kept informed.

No harmful effects are anticipated, however the committee will be informed if any do occur.

PART V.

INFORMED CONSENT MECHANISMS: The committee is mandated by the DHHS and Institute regulations to require documented informed consent. Under certain circumstances, the committee may waive documentation. The elements of such informed consent are:

1.Consent forms should start with a statement that participation is voluntary and that the subject is free to withdraw his/her consent and to discontinue participation in the project or activity at any time without prejudice to the subject.

2.A fair explanation of the procedures to be followed and their purposes, including identification of any procedures which are experimental.

3.A description of any attendant discomforts and risks reasonably to be expected.

4.A description of any benefits to the subject that are reasonably to be expected.

5.A disclosure of any appropriate alternative procedures that might be advantageous for the subject.

6.An offer on the part of the investigator to answer any inquiries concerning the procedures.

7.There shall be no exculpatory language making the subject seem to waive any rights.

8.In addition, the following statements shall appear on all informed consent documents, except that in certain cases in non-biomedical disciplines, COUHES may decide that it may be omitted:

"In the unlikely event of physical injury resulting from participation in this research, I understand that medical treatment will be available from the MIT Medical Department, including first aid emergency treatment and follow-up care as needed, and that my insurance carrier may be billed for the cost of such treatment. However, no compensation can be provided for medical care apart from the foregoing. I further understand that making such medical treatment available, or providing it, does not imply that such injury is the investigator's fault. I also understand that by my participation in this study I am not waiving any of my legal rights."

"I understand that I may also contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, MIT 253-6787, if I feel I have been treated unfairly as a subject."

Consent forms used in cooperating institutions must assure that the rights of the subject are protected at least to the same degree.

These elements should be clearly stated in a document to be signed by the subject or a legally authorized representative in the case of minors or incompetent individuals. The material prescented in such as document must be in clear English, easily understandable to the least educated of subjects. Where minors are involved as subjects, due consideration should be given to their capability to give consent. The informed consent document should be signed by both the subject and parent or guardian wherever possible.

In the case of Questionnaires or Interviews, the Committee may decide that a consent form is not required if the intent is merely to obtain the requested information. However, it must be made clear to the subject that:

- Participation is voluntary.
- The subject may decline to answer any questions.
- The subject may decline further participation at any time without prejudice.
- Confidentiality and/or anonymity are assured.

In addition:

- No coercion to participate will be involved. For example, handing out or collecting questionnaires personally may be so interpreted.
- The data collected will be reported in such a way that the identity of individuals is protected.
- Proper measures will be taken to safeguard the data.

Other examples of situations in which informed consent documentation is not required include use of discarded blood, certain psychological studies involving intentional deception, or record searches and use of stored data. In a case of any deception, debriefing mechanisms must be acceptable before approval of an application may be complete. The Committee expects the investigators will notify the Committee if any hazards develop in excess of those anticipated.

Signature of Principal Investigator________________________________Date__________

Print Full Name:____

Signature of Department Head__________________________________Date__________

Print Full Name:____

Please return this application with 3 photocopies to:

Leigh Firn. M.D.

COUHES Chairman

E23-230

253-6787

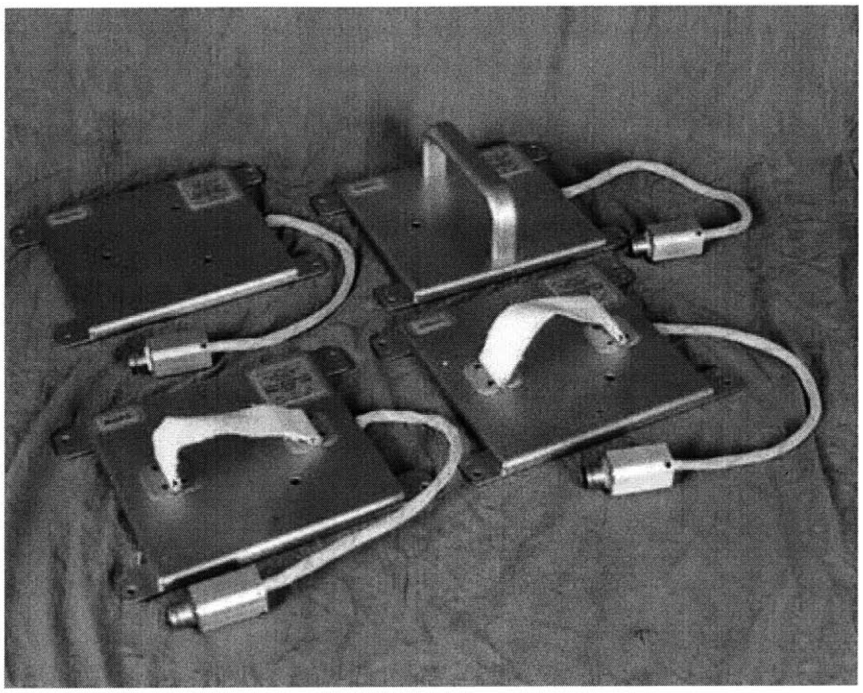


Figure B.1: This photograph shows four sensors as used for designing and testing the MICR0-G prototype. It includes a handhold, a touchpad and two foot restraints.

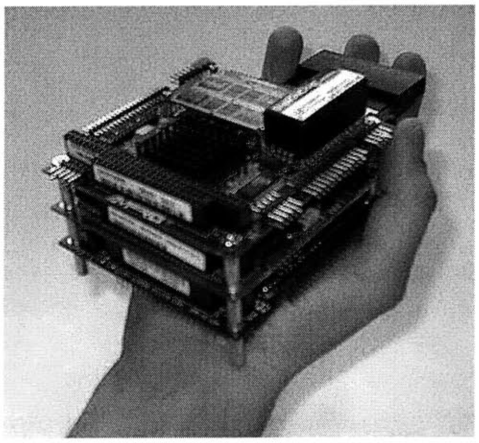


Figure B.2: This figure shows the electronics connected to each sensor apart from the power supply components.

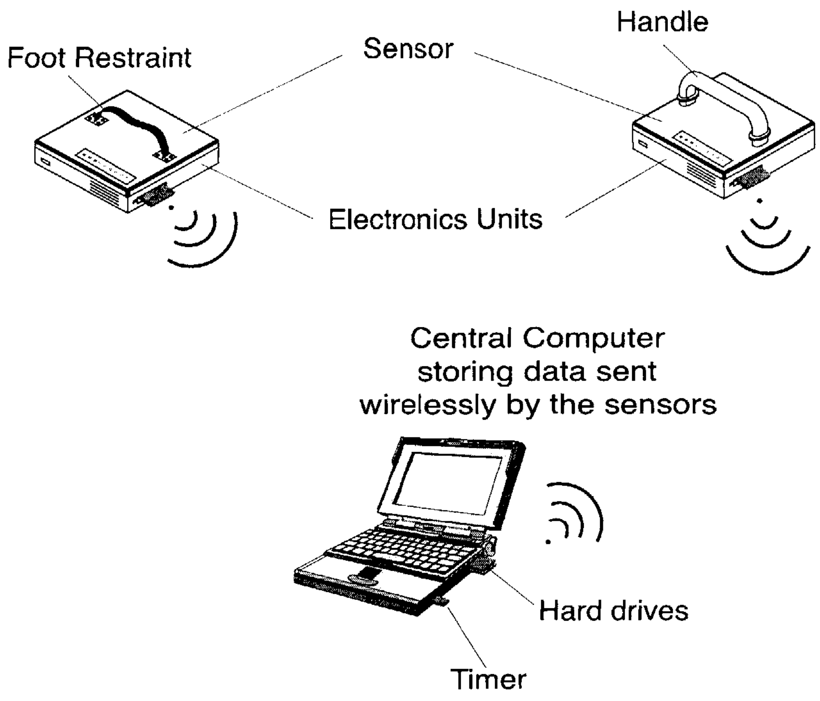

Foot Restraint
Sensor
Handle
Electronics Units
Central Computer storing data sent wirelessly by the sensors
Hard drives
Timer

Figure B.3: The advanced sensors architecture consists of several sensors incorporating the data processing electronics and an RF interface to send wirelessly data to a Central Computer

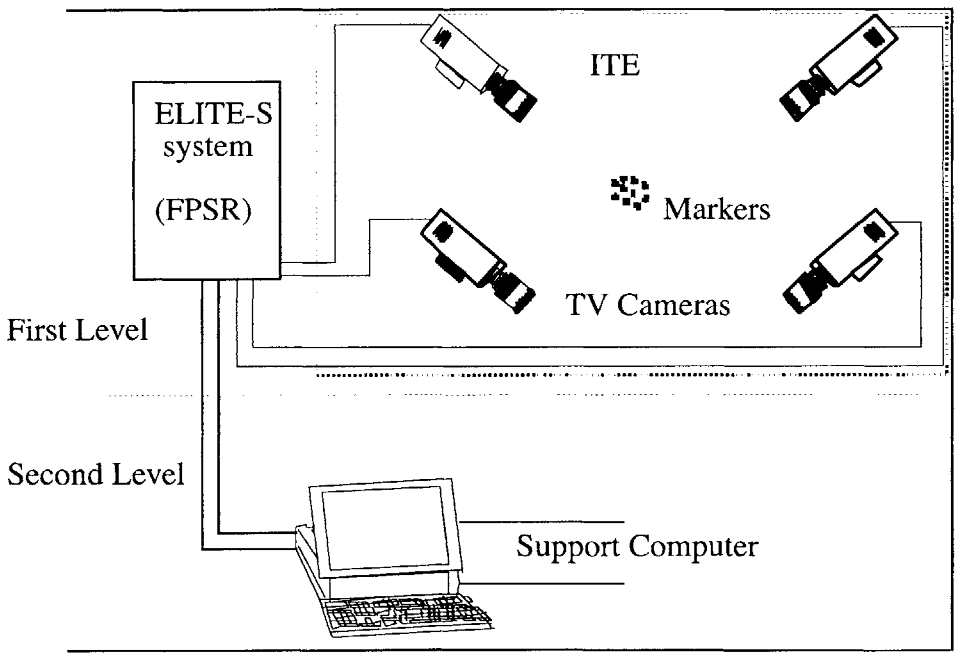


Figure B.4: Architecture of the ELITE-S system. During the experiment, markers are pasted on the subject's cloths to analyze his/her motions.

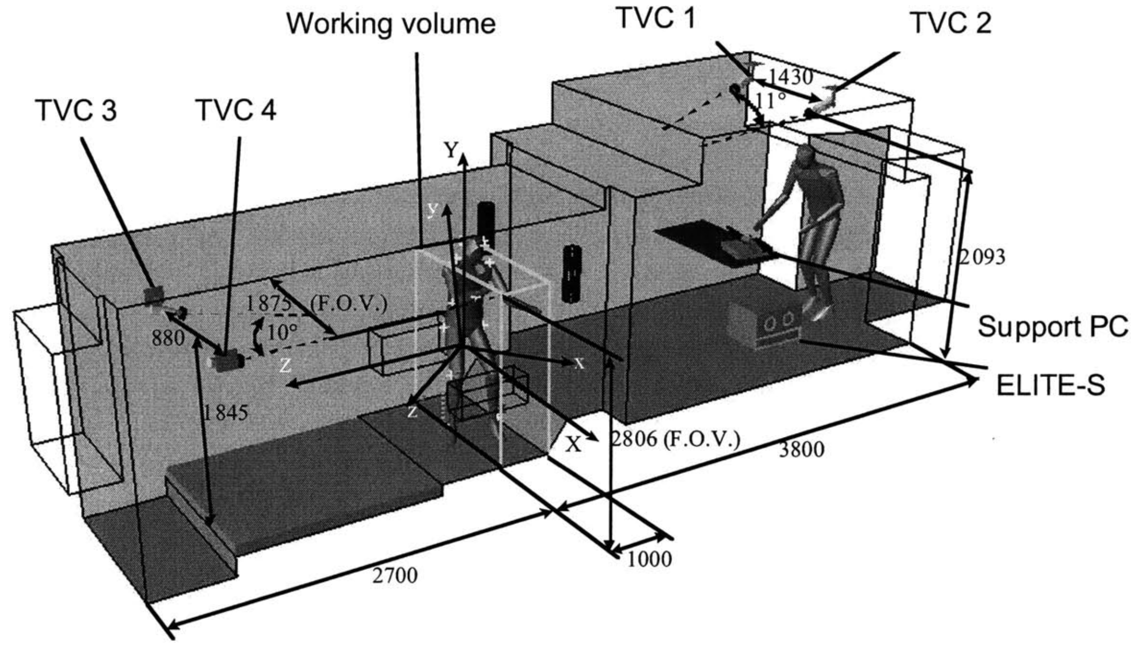


Figure B.5: This figure reports the system experimental set-up. TV cameras optics gave raise to a field of view of 1875x2506 (hxv) millimeters, which guaranteed subject's entire body visibility at each protocol.

Table B.1: Characteristic Astronaut Motion using mobility aids sensors

Characteristic Motion	Description of Astronaut Motion	Sensors Used
Landing	Flying across middeck and landing on a sensor.	HH, FR, TP
Push-off	Pushing off a sensor and floating away.	HH, FR, TP
Flexion	Flexing a limb while using a sensor.	HH, FR
Extension	Extending a limb while using sensor.	HH, FR
Double Support	Using two limbs for support.	HH, FR
Single Support	Using only one limb for support.	HH, FR
Twisting	Twisting body motion.	HH, FR
Vertical Orienting	Vertical re-orienting usually during posture control.	HH, FR, TP
Horizontal Orienting	Horizontal re-orienting usually during posture control.	HH, FR, TP

B.2 The Experimental Subject Consent Form

Microgravity Investigation and Crew Reactions in 0-G (MICR0-G)

Principal Investigator: Prof. Dava J. Newman

Co-Investigators: Antonio Pedotti, Guido Baroni, Sylvie Loday

I. VOLUNTARY PARTICIPATION, RIGHT TO WITHDRAW

Participation in this experiment is voluntary and the subject may withdraw consent and discontinue participation in this experiment at any time without prejudice.

II. PURPOSE AND OBJECTIVE OF EXPERIMENT

The purpose of this study is to test aboard the KC-135 aircraft, an integrated ground-based technology assessing crew-induced disturbances and crew motor behavior in microgravity. Subjects use special mobility aids instrumented with strain gages to record applied forces and moments. An opto-electronic motion analyzer captures the subject's motions and synchronizes its data with the force and moment data acquired by the sensors. Quantitative analysis of human performance in microgravity is important for both scientific investigations and spacecraft engineering design. By collecting and evaluating the kinematic and kinetic data of astronauts in space, it becomes possible to characterize human motor strategies, postural behavior in weightlessness, improve the design of orbital modules, help maintain a quiescent microgravity environment for acceleration-sensitive science experiments, and optimize the human operative capabilities during long-duration space missions. This experiment will test a ground-based prototype acquiring kinetic and kinematic data in the reduced gravity environment of the KC-135, with the objective to further develop a space-qualified version for the International Space Station (ISS).

III. EXPERIMENTAL PROTOCOL

The experiment described in this section takes place aboard the KC-135 aircraft during microgravity sessions. The subject has reflective markers fixed on his cloths and performs different motions using two sensors: a handhold and a foot restraint. Proposed motion tasks are provided in Table 1 but this list is purely informative. The subject is required to use the sensors as mobility devices but is free to perform the motions he wants. The subject understands that the forces and moments that he applied on the sensors will be acquired and that the position of the reflective markers on his body will be detected by video cameras. The subject is required to stay in a defined working volume of 1x2x2 meters during the experiment to stay in the field of view of the cameras. Normal flights, lasting about two hours, consist of 40 parabolic maneuvers. The subject performs the experiment only during microgravity periods lasting approximately 25 seconds per parabola.

Table 1: Characteristic Astronaut Motion Using Mobility Aids Sensors.

Characteristic Motion	Description of Astronaut Motion	Sensors Used
Landing	Flying across the cabin and landing on a sensor.	HH, FR
Push-off	Pushing off a sensor and floating away.	HH, FR
Flexion/Extension	Flexing/Extending a limb while using a sensor.	HH, FR
Single/Double Support	Using one/two limbs for support.	HH, FR
Twisting	Twisting body motion.	HH, FR
Vertical/Horizontal Orienting	Vertical/Horizontal re-orienting usually during posture control.	HH, FR

IV. FORESEEABLE INCONVENIENCE, DISCOMFORT, AND RISKS TO THE SUBJECT

The subject understands that he may experience motion sickness during the flight. Symptoms include pallor, increased perspiration, nausea and vomiting. If the subject notice inconvenience or sickness during the flight, he should disqualify him/herself from further testing. The subject understands that he/she can discontinue participation in this experiment at any time without prejudice, but he will have to withstand the 2 hours of parabolic flight even if he/she is sick.

V. RISK MINIMIZATION

Participation in this study entails the usual risks associated with flying. In attempt to avoid motion sickness, various medications have been used for many years but no one medicine has been perfect in preventing this condition. The subject will have to discuss this matter with one of the NASA physician prior to flight. A medical officer is assigned to the flight and other medical facilities are available in the local area. The subject must provide the results of a KC-135 Examination or equivalent FAA Third Class Aviation Physical signed by an FAA Medical Examiner before the flight experiment. The subject will also receive a physiological training provided by NASA prior to flight.

VI. REMEDY IN THE EVENT OF INJURY

In the event of physical injury or medical problem resulting from participation in this research, the subject understands that medical treatment will be available from medical facilities available in the local area (i.e. Kelsey-Seybold Clinic, Christus St. John Hospital, or Clear Lake Regional Medical Center), including first aid emergency treatment and follow-up care as needed, and that his/her insurance carrier may be billed for the cost of such treatment. However, no compensation can be provided for medical care apart from the foregoing. The subject further understands that making such medical treatment available, or providing it, does not imply that such injury is the investigator's fault. The subject also understands that by his/her participation in this study he/she is not waiving any of his/her legal rights.

VII. CONFIDENTIALTY

The subject will be video taped while performing the experiment by two opto-electronics cameras. This video system is based on a recognition software, which detects the position of the reflective markers sticked on the subject's clothes at the body joints. The position of the markers indicating the motions of the subject will be acquired but the subject cannot be recognized on the video tapes. In addition, data acquired during the experiment will not be related to the subject involved. The anonymity of the subject is fully preserved.

VIII. COMPENSATION

The subject will receive no compensation for participating in this experiment, but costs for remote experiments (travel, lodging, etc.) are covered.

IX. ANSWERS TO QUESTIONS

The subject may receive answers to any questions related to this experiment by asking the test conductor or contacting the Principal Investigator at (617) 258-8799. After the experiment, the kinetic data (restraint devices) will be analyzed at MIT and the kinematic data (video system) will be brought back to Politecnico di Milano University in Italy and analyzed there. The subject will be given contact information of any person who led the experiment or analyzes the corresponding data, by simply asking the Principal Investigator.

X. IN THE EVENT OF UNFAIR TREATMENT

The subject understands that he/she may also contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, MIT 253-6787, if he/she feels he/she has been treated unfairly as a subject.

XI. SIGNATURE

I, ____________________________, have read and understand the information contained in this consent form

(Subject's Printed Name)

and agree to participate as a subject in this experiment.

____

(Subject's Signature) (Date)

____

(Witness)

APPENDIX

C

INSTITUTIONAL REVIEW BOARD MICR0-G DOCUMENTATION

HUMAN RESEARCH MASTER PROTOCOL
Proposed JSC Reduced Gravity Program Research

Microgravity Investigation and Crew Reactions in 0-G (MICR0-G)
Principal Investigator: Prof. Dava J. Newman
Co-Investigators: Antonio Pedotti, Guido Baroni, Sylvie Loday

Tentative Flight Dates: January 23-26, 2000.

1. TITLE

Microgravity Investigation and Crew Reactions in 0-G (MICR0-G).

2. ORGANIZATION CONDUCTING THE RESEARCH

The Massachusetts Institute of Technology (MIT).

3. INVESTIGATORS

A. *List all investigators starting with the Principal Investigator (PI), their addresses, and phone numbers. Attach curriculum vitae for each investigator at the end of the protocol.*

Dr. Dava J. Newman
Department of Aeronautics and Astronautics
Bldg. 33, Room 307
Massachusetts Institute of Technology
77 Massachusetts Av.
Cambridge, MA 02139
Telephone No.: (617) 258-8799
E-mail address: dnewman@mit.edu

Co-I: Dr. Antonio Pedotti
Centro di Bioingegneria
Via Capecelatro,66
20148 Milano, Italy
Telephone No.: +39-(0)2-48705848
E-mail address: pedotti@mail.cbi.polimi.it

Co-I: Mr. Sherwin Beck
NASA Langley Research Center (LaRC)
Bldg. 1202, Room 146
Hampton, VA 23681
Telephone No.: 757-864-1966
E-mail address: s.m.beck@larc.nasa.gov

B. *List technical personnel who will aid in and/or conduct the research. Attach qualifications at the end of the protocol. The Committee is interested in the qualifications of the technical staff that will be interacting with the test subjects because they will be operating equipment or performing procedures on them.*

Guido Baroni, Scientist, Centro di Bioengegneria, Politecnico di Milano, Italy.
Sylvie Loday, S.M. candidate, Dept. Computer Science, MIT.
Jesse Byler, Senior Undergraduate Student, Dept. Computer Science, MIT.
Charles Keyes, Senior Undergraduate Student, Harvard.
Alessandra Pedrocchi, Graduate Student, Politecnico di Milano, Italy.

4. HYPOTHESES

The hypotheses should be clearly and succinctly stated. The Committee must consider scientific merit as a factor in weighing risks vs. benefits. This summary should abstract the details to be included in Section 5.

One key motivation for spaceflight missions is to achieve a microgravity environment to provide the best on-orbit conditions to life science, material science, physics experiments and astronomical measurements [1]. This required microgravity environment can be disturbed by external sources such as aerodynamic drag or solar pressure, as well as internal sources such as operations of mechanical equipment, and astronauts' motions. The human-induced disturbances are the more challenging to predict and assess due to their inherent randomness. The purpose of this study is to test aboard the KC-135 aircraft, an integrated ground-based technology assessing crew-induced disturbances and crew motor behavior in microgravity. Subjects will use special mobility aids instrumented with strain gages to record applied forces and moments. An opto-electronic motion analyzer will capture the subject's motions and will synchronize its data with the force and moment data acquired by the sensors. Quantitative analysis of human performance in microgravity is also important for both scientific investigations and spacecraft engineering design. By collecting and evaluating the kinematic and kinetic data of astronauts in space, it becomes possible to characterize human motor strategies, postural behavior in weightlessness, improve the design of orbital modules, and optimize the human operative capabilities during long-duration space missions [2].

5. PURPOSE OF RESEARCH

A. Historical Background

A brief background statement should trace the development of key factors or principles that led to the formulation of hypothesis. Reference to pertinent scientific literature is essential.

The first spaceflight experiment carried out to measure crew disturbances was the Skylab Crew/Vehicle Disturbance Experiment T-013 in 1973 [3]. This experiment provided the first data collected in space on astronaut-induced disturbances and verified that astronauts can produce significant disturbance forces and moments if they so desire [4]. In the 1980s, achieving a microgravity environment for acceleration-sensitive science experiments became the primary motivation for assessing astronauts-induced disturbances [5]. The DLS (Dynamic Load Sensors) experiment on-board the Space Shuttle in March 1994 [6], and the follow-on EDLS (Enhanced Dynamic Load Sensors) experiment on-board the Russian Mir space station from May 1996 to May 1997 [7], proposed to describe, quantify, and predict astronaut motions during normal on-orbit activities. The ultimate goal was to define how to obtain a stable, very low-level microgravity environment, necessary to assure that life science, material science, and astronomical investigations on board the ISS yield the most accurate data.

As the ISS becomes operational, there is a need for advanced kinematic and kinetic instruments to make precise measurements of the disturbances exerted by the astronauts and to characterize human motor strategies and postural behavior in weightlessness for long-duration space missions [8]. The design and development of an on-ground prototype system is the main purpose of the MICR0-G (Microgravity Investigation and Crew Reactions in 0-G) research effort, carried out by MIT, with collaboration from NASA and Politecnico di Milano University. The astronaut-induced forces and moments will be measured by an advanced version of Dynamic Load Sensors (DLS) flown on the Space Shuttle and on Mir. The astronaut motions will be captured by the ELITE-S2 system [9], an enhanced version of the real-time opto-electronic motion analyzers ELITE-S and Kinelite, flown respectively on the EuroMir '95 Mission and on the Neurolab mission.

B. New Information Expected

This experiment will test ground-based hardware. Specifically, a prototype system to measure kinematic and kinetic data in the reduced gravity environment of the KC-135 aircraft and eventually on the ISS. The post-flight analysis of the prototype testing will determine whether the performance of the entire system meets the functional requirement defined in the MICR0-G proposal to NASA in March 1997 [7]. If these requirements are not met, components will be replaced as needed, the entire design reevaluated and a new breadboard model assembled. When the requirements are met, the ground-based prototype will be used to secure funding in order to complete the development and build the spaceflight hardware for the ISS.

6. JUSTIFICATION FOR USE OF HUMAN SUBJECTS

Explain why humans are a necessary part of the study.

The MICR0-G experiment aboard the KC-135 will test an advanced kinematic and kinetic acquisition system in a microgravity environment. The key hardware of the experiment is a set of two sensors, a handhold and a foot restraint as shown on Figure C.1, able to record applied forces and moments. Subjects will use these special mobility aids to perform different motions in microgravity. An opto-electronic motion analyzer will capture the subject's motions using reflective markers affixed to the subject. We instruct human subjects to use the sensors during microgravity sessions aboard the KC-135 in order to acquire kinetic and kinematic data.

Although the experiment involves human subjects, the primary objective is to test a breadboard electronics system, not to assess human performance.

7. STUDY PLAN AND SCHEDULE

Give an overview of what will be accomplished during preflight training/base-line data collection sessions, in-flight experimentation, and postflight data acquisitions. For example, familiarization with the concepts of the experiment, procedures to be learned, equipment to be used, data collection, etc.

A. Dates/Duration

The preflight tests will include the calibration of the sensors, tests of the software, and acquisition of kinetic and kinematic data. Two sessions are scheduled to get the subjects familiar with the hardware and the experimental sessions aboard the KC-135 aircraft. The operators of the system in-flight have been working on the calibration and software development. Consequently, they are familiar with the experimental procedures and do not need specific training sessions.

The system will be tested aboard the KC-135 aircraft during four parabolic flights from January 23 to January 26, 2001. The in-flight experimental protocol and equipment are detailed in the following section.

Data collected in-flight will be analyzed and the performance of the entire system will be assessed. If the system does not meet the functional requirements defined in the MICR0-G proposal to NASA (March 1997) [7], the design will be reevaluated and an upgraded system assembled. Post-flight activities do not include data collection involving human subjects.

Table C.1: Summarized schedule of the MICR0-G experiment

Calibration of the system	Nov. 15 - 30, 2000
Tests of the sensors and cameras	Dec. 1 - 15, 2000
Training of subjects	Dec. 14, 2000 and Jan. 18, 2001
Tests of the in-flight experimental protocol	Dec. 18, 2000 -Jan. 5, 2001
Pre-flight acquisition of data	Jan. 8 - 12, 2001
Physiological training, tests at JSC	Jan. 15 - 19, 2001
In-flight experiment	Jan. 23 - 26, 2001
Post-flight data analysis and assessment of the system	Feb. 1-May 1, 2001

B. Place(s) of Training Test

The preflight training tests will be conducted at MIT, using the facilities of the Man Vehicle Laboratory (MVL).

C. Subjects

Provide names, dates of physicals and physiological training, and date consent form(s) signed for each subject.

The names of the persons who could be either an operator or a subject during the in-flight experiments are in italic. The other persons are naïve subjects.

Table C.2: Dates of physicals and physiological training, date consent form(s) signed for each subject.

Subject's Names	Physical exams	Physiological Training	Consent Forms Signed
Dava Newman	-	02/14/00	11/25/00
Sylvie Loday	11/28/00	01/16/01	11/25/00
Jesse Byler	11/28/00	01/16/01	11/25/00
Charles Keyes	11/28/00	01/16/01	11/25/00
Guido Baroni	11/30/00	01/16/01	11/25/00
Alessandra Pedrocchi	11/30/00	01/16/01	11/25/00
Gui Trotti	11/21/00	01/16/01	11/25/00
Jim Olive	-	01/16/01	11/25/00

8. EXPERIMENTAL PROTOCOLS AND EQUIPMENT

This section contains some of the most important information used by the Committee. It is from this section that the Committee may identify potential problems that might be overlooked by the investigators. Experience has shown that incompleteness of this section is one of the major reasons for IRB nonapproval.

A. Preflight Training and Baseline Data Collection

The pre-flight study plan encompasses the test of the overall system (hardware and software) as well as the training of the subjects to get them familiar with the key hardware of the experiment and teach them the procedure of the in-flight experiment. Preflight familiarization with all prototype equipment and desired subject activities will be administrated at MIT with a short refresher preceding the KC-135 flight. Baseline data collection and equipment calibration will be performed at MIT prior to the KC-135 flights.

B. In-flight Activities

List step-by-step procedures and equipment used, approximate duration of the testing, how many flight personnel are necessary, and how many times the experiment will be performed.

- **Equipment**

The key experiment hardware of the kinetic system is based on three types of sensors -- a handhold, a foot restraint and a touchpad, as shown on Figure C.1. The sensors provide the functionality of crew restraint devices and mobility aids while measuring the applied forces and moments in the x-, y-, and z- axes (6 degrees of freedom).Each sensor is connected to an advanced electronics system processing and storing the acquired force and moment data. This electronics unit consists of two parts: (1) the Sensor Electronics Unit (SEU) and (2) The Central Computer (CC). The Sensor Electronics Unit is a box housing all the necessary electronics for data processing and networking. Each sensor is connected via a cable to a corresponding SEU. The Central Computer is used as a server for networking and incorporates the storage devices to store all the data recorded by the sensors. This base station communicates wirelessly with the whole set of sensors and provides the main user interface with the system through a laptop computer display. The advanced sensors' architecture is illustrated in Figure C.2. The kinematic measurement component of the MICR0-G system will consist of a versatile opto-electronic motion analyzer, called ELITE-S, based on passive markers recognition. The system is composed of TV Cameras (TVC) connected via a bus to an IBM Thinkpad 760 laptop.

The architecture of the system, as shown in Figure C.3, has two levels: Level 1 consists of the Interface with The Environment (ITE) and of a Fast Processor for Shape Recognition (FPSR). The components of the ITE are CCD video cameras as well as a set of reflective markers affixed to the subject. The identification of markers in each camera image is performed by dedicated pattern recognition hardware: the FPSR. Level 2 of the architecture is a laptop computer with special software performing data processing in real-time. The two systems are connected with a Universal Serial Bus (USB) to synchronize the acquisition of kinetic data of the advanced sensors with the kinematic data of the ELITE-S system.

The hardware used aboard the KC-135 aircraft can then be summarized as follows:
- 2 full strain gages sensors (one hand hold and one foot restraint) connected to their electronics unit.
- 2 TV camera units
- Reflective markers
- 2 laptop computers

- **Experimental set-up:**

Figure C.4 represents the experimental set-up within the aircraft main cabin. Due to volume constraints, we will use only two sensors (one foot restraint and one handhold) and two TV Cameras (TVC1 and TVC2). The two cameras will be 4 meters away from the working volume measuring about 1x2x2 meters. The foot restraint will be fixed on the floor and the handhold on a wall inside the working volume. The electronics connected to the sensor will be fixed on the floor using straps. Brackets will be used to fix the TV cameras pointing towards the acquisition volume. In addition, the experiment will use two laptop computers, one to acquire the kinetic data recorded by each sensor and the other one to store the processed data of the opto-electronic motion analyzer ELITE-S.

- **Protocol:**

The experiment described in this section will take place aboard the KC-135 aircraft during microgravity sessions.

The subject will have reflective markers fixed on his cloths as shown on Figure C.5 and will perform different motions using two sensors: a handhold and a foot restraint. The description of several motion tasks is provided in Table C.3. We will instruct the subjects to perform typical daily tasks and motions in microgravity using the sensors as restraint devices. Table C.4 summarizes the motions' activity for each parabola of the flight. The forces and moments applied by the subject on the sensors will be acquired and the position of the reflective markers on his body will be analyzed by the TV cameras' system. The subject is required to stay in a defined working volume of 1x2x2 meters during the experiment to stay in the field of view of the cameras. Normal flights last about two hours and consist of 40 parabolic maneuvers. The subject performs the experiment only during microgravity periods lasting approximately 25 seconds per parabola.

In addition to the subject, at least two operators are necessary to run the experiment. They will respectively work on the EDLS and ELITE-S laptop computers to verify that the acquisition software is running correctly on both systems and to modify parameters such as gains or resolution during the flight if necessary. The two experimenters, one test director and one systems operator, can stand between the two cameras and the working volume as long as they don't interfere with the field of view of the cameras. A minimum of three people total is necessary to conduct the experiment. Considering that the subject may experience motion sickness during the flight, we require an additional potential subject, leading to a minimum of four people aboard the KC-135 to run the experiment safely.

Table C.3: Characteristic Astronaut Motion Using Mobility Aids Sensors

Characteristic Motion	Description of Astronaut Motion	Sensors Used
Landing	Flying across the cabin and landing on a sensor.	HH, FR
Push-off	Pushing off a sensor and floating away.	HH, FR
Flexion/Extension	Flexing/Extending a limb while using a sensor.	HH, FR
Single/Double Support	Using one/two limbs for support.	HH, FR
Twisting	Twisting body motion.	HH, FR
Vertical/Horizontal Orienting	Vertical/Horizontal re-orienting usually for posture control.	HH, FR
Axial Movements	30° Upper trunk forward/backward bending	FR, FR
Lateral Leg Raising	45° leg abduction	FR, FR

Table C.4: Description of the subject activity during each parabola

Parabola Number	Sensors used	Subject Activity
1-2	HH, FR	Weightlessness accommodation
3	HH, FR	Sensor/Camera identification
4	HH, FR	Interaction with the foot restraint
5	HH, FR	Interaction with the handhold
6-9	HH, FR	Perform A: Landing/Push-off motions
10-13	HH, FR	Perform B: Flexion/Extension and Single/Double support activity
14-17	HH, FR	Perform C: Twisting and Re-orienting motions
18-20	HH, FR	Perform D: Perform position body spin along yaw and pitch axes for one complete revolution
21-26	FR, FR	Perform E: Motor co-ordination protocols (30° bending and 45° leg abduction)
27-40	HH, FR, FR	Repeat any missed motions or second subject performs A,B,C,D, E motions

C. Postflight Activities

If postflight testing of flight personnel is necessary, note how many times the test will be done, when, where, and what procedures and equipment will be used.

Post-flight activities do not include data collection involving human subjects.

9. HAZARD ANALYSES AND SAFETY PRECAUTIONS

Detail the conceivable hazards that might be encountered during the study and the precautions that will be taken to avoid them. For research involving animal handling, list precautions employed for minimizing zoonoses.

A. Preflight Activities

N/A

B. In-flight Activities

- **Potential Hazards**

Subjects and/or operators could inadvertently bump into the hardware causing a slight scratch or bruise. No invasive measurements are made.

- **Protection to Minimize Risks**

The sensors and the TVC units (cameras) have already flown and are flight certified.

The new electronics components are housed in plastic boxes connected to the sensors. All the electronics boards are packaged in a LEXAN™ (plastic) box. The corners are rounded and the units will be placed about one meter from the sensors to limit the risks of the subjects to bump into them.

- **Assessment of Residual Risks**

We do not envision any residual risks.

C. Postflight Activities

N/A

10.POSSIBLE INCONVENIENCES OR DISCOMFORTS TO SUBJECTS

List additional factors that do not fall into the category of hazards, but that should be considered.

The main discomfort that the subject might experience during the experiment aboard the KC-135 aircraft is motion sickness. Symptoms include pallor, increased perspiration, nausea and vomiting. In attempt to avoid this syndrome, various medications have been used for many years but no one medicine has been perfect in preventing this condition. Each individual must decide for him/herself what to do. All flyers will have to discuss this matter with one of the NASA physician prior to flight. A medical officer is assigned to the flight and other medical facilities are available in the local area.

11.EXTENT OF PHYSICAL EXAMINATIONS

In many cases, reliance on the annual physical examination for flight personnel is all that need be stated. If a special physical examination or special test is required, describe it and state why it is needed.

The dates our subjects passed the physical examination for flight personnel and physiological training required to fly aboard the KC-135 are provided in section 7. No other physical examination is needed for their participation in the MICR0-G experiment aboard the KC-135.

12.AVAILABILITY OF A PHYSICIAN AND MEDICAL FACILITIES

State if a flight surgeon and/or facilities will be required during preflight, flight, or postflight.

No surgeon or particular medical facilities will be required during preflight, flight, or postflight.

13.FIGURES

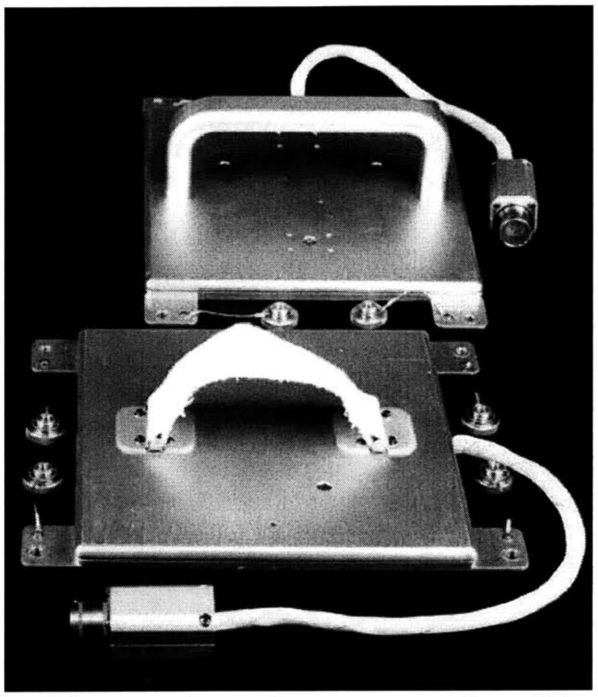


Figure C.1: This handhold and foot restraint are the two sensors that will be used aboard the KC-135.

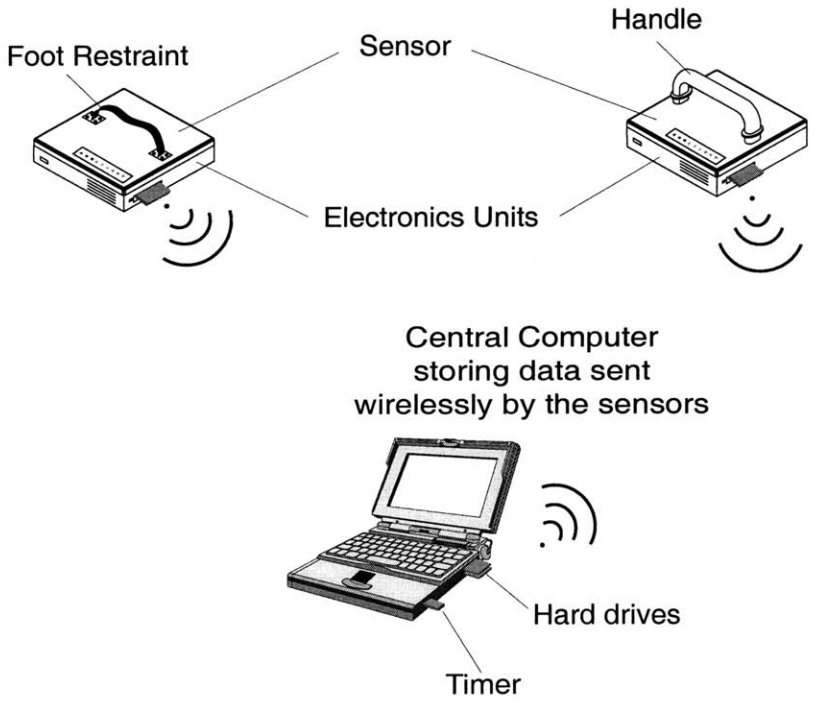

Foot Restraint
Sensor
Handle
Electronics Units
Central Computer storing data sent wirelessly by the sensors
Hard drives
Timer

Figure C.2: The advanced sensors architecture consists of two sensors incorporating the data processing electronics and an RF interface to send wirelessly data to a Central Computer.

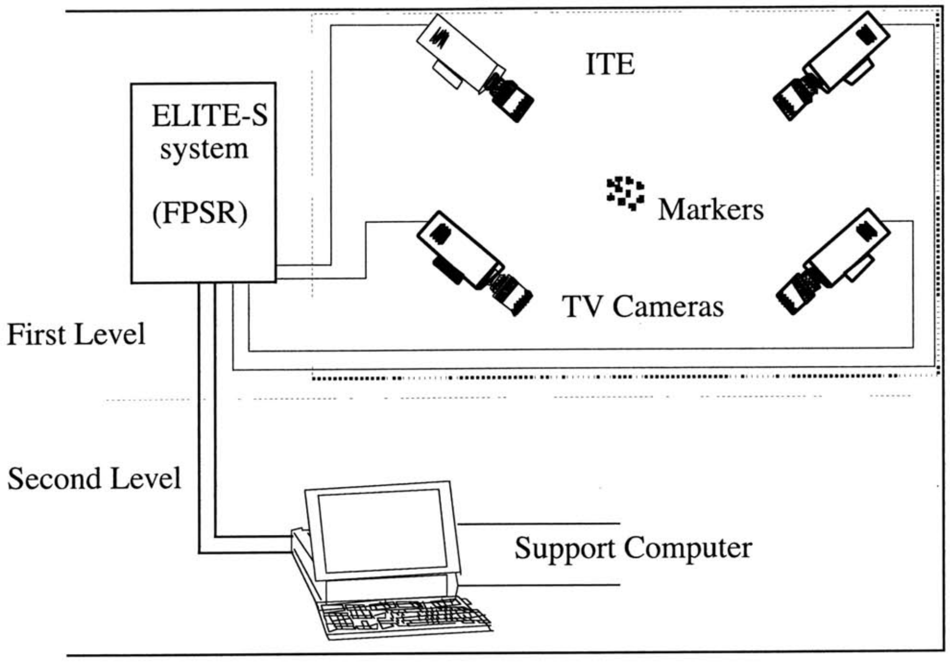


Figure C.3: Architecture of the ELITE-S system. During the experiment, the markers are affixed on the subject's cloths to analyze his/her motions.

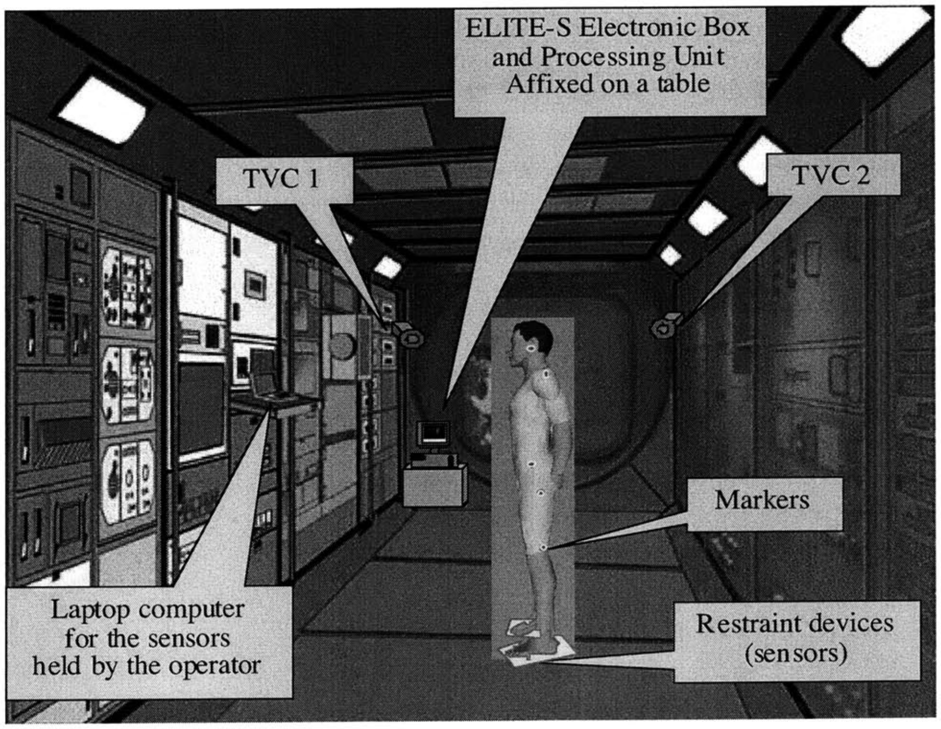
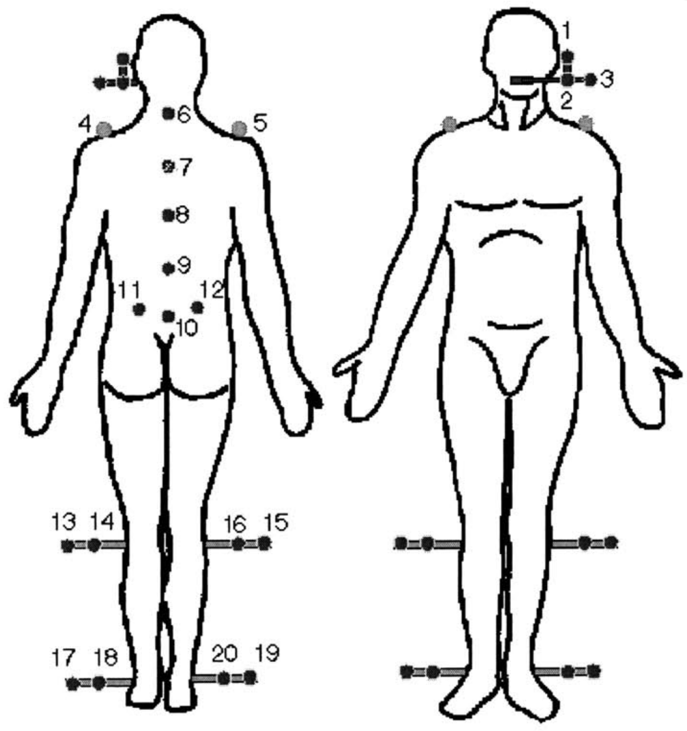

ELITE-S Electronic Box and Processing Unit Affixed on a table
TVC 1
TVC 2
Markers
Laptop computer for the sensors held by the operator
Restraint devices (sensors)

Figure C.4: Experimental flight set-up of the advanced load sensors and the ELITE system.



Markers	Position
1-2-3	Biteboard
4	left acromion
5	right acromion
6	spiny process C7
7	spiny process T7-T8
8	spiny process T12
9	spiny process L3
10	spiny process L5-S1
11	left posterior iliac spine
12	right posterior iliac spine
13-14	2 markers stick on left femural epicondyles
15-16	2 markers stick on right femural epicondyles
17-18	2 markers stick on left lateral malleolus
19-20	2 markers stick on right lateral malleolus

Figure C.5: Disposition of the reflective markers on the subject.

14.REFERENCES

[1] NASA Johnson Space Center, "Microgravity Control Plan", SSP 50036, Revision A, Type 2, NASA, 29 February 1996.

[2] A. Amir, D.J. Newman," Research Into the Effects of Astronauts Motion on the Spacecraft: a Review", Acta Astronautica, November 1999.

[3] B. Conway, T. C. Hendricks, "A Summary of the Skylab Crew/Vehicle Disturbance Experiment T-013", Nasa Technical Note D-8128, March 1976.

[4] B.Conway, Bruce A., "Investigation of Crew Motion Disturbances on Skylab-Experiment T-013", Advances in the Astronautical Sciences, 1974.

[5] Boeing Defense & Space Group, "System Specification for the International Space Station", Specification Number SSP 41000, Revision H, 25 May 1998.

[6] Newman, D. J., Marthinus van Schoor, and Javier de Luis, "Crew Force Measurements: Dynamic Load Sensors (DLS) on Mir," Proposal to the NASA Office of Life & Microgravity Sciences & Applications, May 1994.

[7] D. J. Newman, M.Tryfonidis, and M.van Schoor, "Astronaut-Induced Disturbances in Microgravity". Journal of Spacecraft and Rockets, Vol. 34, No. 2, pp. 252–254, March–April 1997.

[8] D.J. Newman, A. Pedotti, S. Beck, "Microgravity Investigations and Crew Reactions in 0-G (MICR0-G)", Proposal to the NASA Office of Life & Microgravity Sciences & Applications, March 1997.

[9] A. Amir, G. Baroni, A. Pedrocchi, D.Newman, "Measuring Astronaut Performance on the ISS: Advanced Kinematic and Kinetic Instrumentation". IEEE Transaction on Instrumentation and Measurements, Venice, Italy, May 1999.

APPENDIX

D

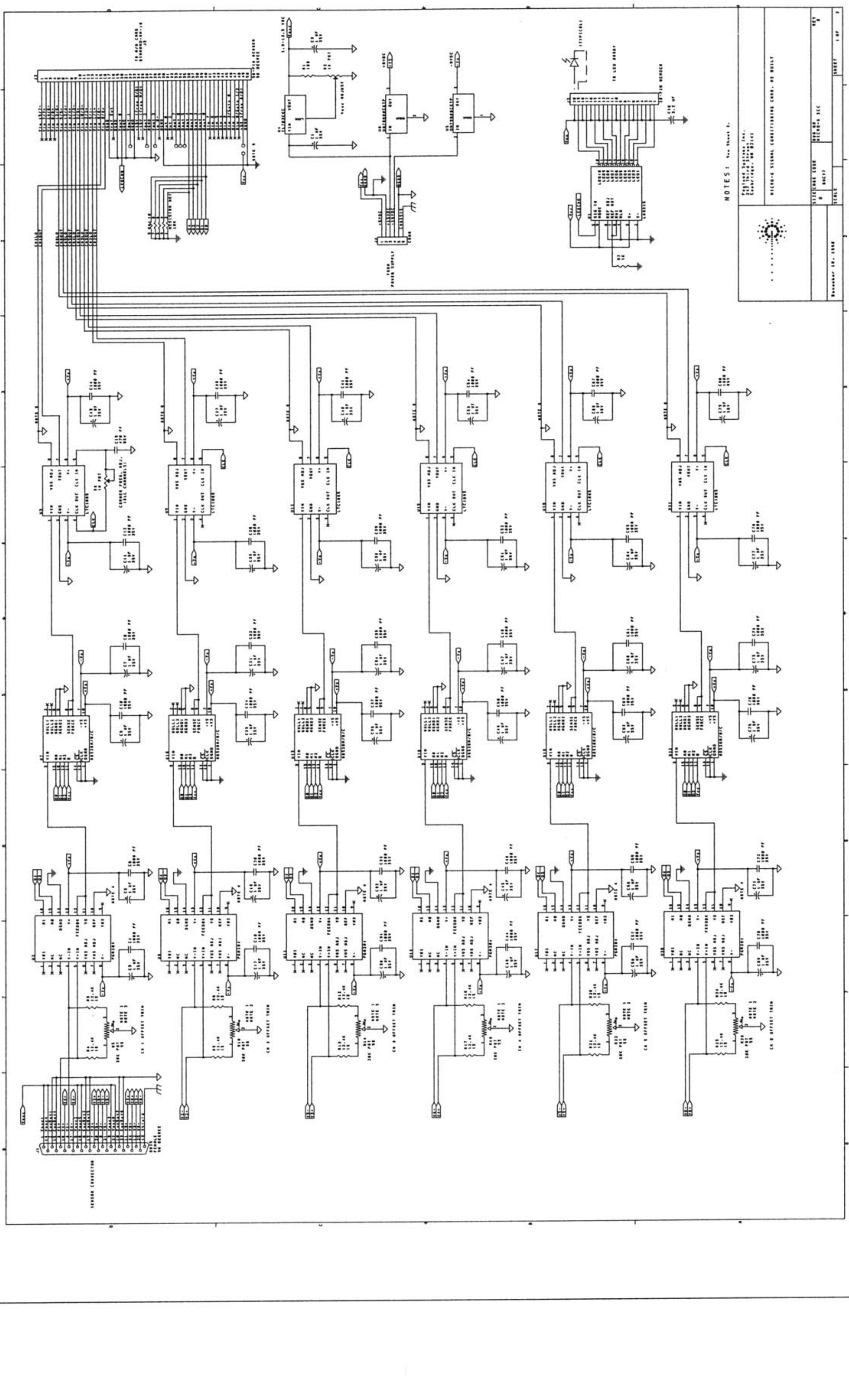
SIGNAL CONDITIONING BOARD

This appendix details the specifications and electronics drawings of the signal conditioning board that has been designed and built by Payload System Inc. (Cambridge, MA) for the advanced load sensor prototype. A total of five boards have been ordered in April 2000 and received at MIT in July 2000.

The main specifications of the board are the following:

• Form factor:	PC/104 compatible
• Input:	6 channels, differential input, compatible with full bridge strain gages
• Output:	6 channels, connector matching the input of the Diamond-MM-16 A/D converter
	Excitation for 6 strain gage bridges adjustable between 1.5 and 10 VDC
• Gain:	20 values selectable from 1 to 16000 V/V (values provided in the "Notes" below)
• Filtering:	3rd order Bessel low-pass filter, trimpot adjust of corner frequency (50 to 2000Hz)
• Dynamic Range:	Boards will be designed for compatibility with a 16-bit A/D. Noise performance of the filters may limit the dynamic range to 14-bit effective.

The next page provides the schematics of the signal conditioning board. The input connector matches the DB25 connector of the ground-based load sensors. The excitation voltage, V_{exc},can be adjusted with the trimpot R_2. The inputs are six differential signals coming from the load sensors. Each signal is processed independently but in the exact same manner. Therefore, each electronics component appear six times on the board and the drawings. The trimpots R_4, R_9, R_{13}, R_{17}, R_{21}, and R_{25} allow to adjust the offset on each channel. A first set of chips amplifies the signals by a factor of 1, 10, 100 or 1000. This gain is controlled by the inputs D_3 and D_4 from the A/D. A second set of components may amplify the signals by 1, 2, 4, 8, or 16 with the inputs D_0, D_1 and D_2 from the A/D. Finally the signals are filtered by a third order Bessel low-pass filter. The trimpot R_6 adjusts the corner frequency for all the channels at once. The output connector matches the input connector of the Diamond-MM-16 A/D board. To reduce the noise and improve the overall performance, each output signal is taken differential with one wire connected to the referenced ground.



NOTES

1. Use very low temperature coefficient resistors throughout, (5 ppm per degree C max), except regulator and LED circuits.

2. Use very low temperature coefficient potentiometers, (5 ppm per degree C max), except regulator circuit.

3. PGA204 and AD526 input circuits are sensitive. Avoid 'vias' in PCB traces. Use largest PCB traces and pads possible.

4. Keep impedance low to Ref input (tie to AGND plane directly).

5. Use square pads to indicate pin 1 on all parts.

6. Route input negative lines to ground point close to associated LTC-1065.

7. Use separate power and ground plane layers. Ground plane layer should cover entire PCB.

8. Set D0-D4 as follows to select gain:

TOTAL GAIN	D4	D3	D2	D1	D0	PGA204 GAIN	AD526 GAIN
1	0	0	0	0	0	1	1
2	0	0	0	0	1	1	2
4	0	0	0	1	0	1	4
8	0	0	0	1	1	1	8
16	0	0	1	X	X	1	16
10	0	1	0	0	0	10	1
20	0	1	0	0	1	10	2
40	0	1	0	1	0	10	4
80	0	1	0	1	1	10	8
160	0	1	1	X	X	10	16
100	1	0	0	0	0	100	1
200	1	0	0	0	1	100	2
400	1	0	0	1	0	100	4
800	1	0	0	1	1	100	8
1600	1	0	1	X	X	100	16
1000	1	1	0	0	0	1000	1
* 2000	1	1	0	0	1	1000	2
* 4000	1	1	0	1	0	1000	4
* 8000	1	1	0	1	1	1000	8
* 16000	1	1	1	X	X	1000	16

*Available but not reccomended.

9. Install jumper to use +5VDC from A/D card (only if primary Vcc input is unused).

10. Silkscreen must include: PAYLOAD SYSTEMS INC.
MICRO-G SCC REV 0
S/N _______

PAYLOAD SYSTEMS INC.
247 THIRD STREET
CAMBRIDGE, MA 02141

MICRO-G SCC NOTES

SIZE	CAGE CODE	DWG NO	REV
A		MICRO-G SCC NOTES	0

June 15, 1990

SCALE

SHEET 2 OF 2

6597-94