AAMRL-TR-90-023

# ADVANCED DYNAMIC ANTHROPOMORPHIC MANIKIN (ADAM) FINAL DESIGN REPORT

Aileen M. Bartol
Vernon L. Hazen
Joseph F. Kowalski
Brian P. Murphy
Richard P. White, Jr.

SYSTEMS RESEARCH LABORATORIES, INC.
A Division of Arvin/Calspan
2800 Indian Ripple Road
Dayton, Ohio 45440



MARCH 1990

Final Report for the Period February 1985 through June 1989

Approved for public release; distribution is unlimited

ARMSTRONG AEROSPACE MEDICAL RESEARCH LABORATORY
HUMAN SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-6573

# NOTICES



## TECHNICAL REVIEW AND APPROVAL

AAMRL-TR-90-023

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

JAMES W. BRINKLEY
Director
Biodynamics and Bioengineering Division
Harry G. Armstrong Aerospace Medical Research Laboratory

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

| | |
|---|---|
| 1a REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b RESTRICTIVE MARKINGS |
| 2a SECURITY CLASSIFICATION AUTHORITY<br>2b DECLASSIFICATION / DOWNGRADING SCHEDULE | 3 DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release. Distribution is unlimited. |
| 4 PERFORMING ORGANIZATION REPORT NUMBER(S) | 5 MONITORING ORGANIZATION REPORT NUMBER(S)<br>AAMRL-TR-90-023 |
| 6a NAME OF PERFORMING ORGANIZATION<br>Systems Research Laboratories, Inc.<br>6b OFFICE SYMBOL (If applicable) | 7a NAME OF MONITORING ORGANIZATION<br>Armstrong Aerospace Medical Research Laboratory, AAMRL/BB |
| 6c ADDRESS (City, State, and ZIP Code)<br>2800 Indian Ripple Rd<br>Dayton OH 45440-3696 | 7b ADDRESS (City, State, and ZIP Code)<br>Wright-Patterson Air Force Base OH 45433-6573 |
| 8a NAME OF FUNDING/SPONSORING ORGANIZATION<br>8b OFFICE SYMBOL (If applicable) | 9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>F33615-85-C-0535 |
| 8c ADDRESS (City, State, and ZIP Code) | 10 SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO 62202F; PROJECT NO 7231; TASK NO 36; WORK UNIT ACCESSION NO 02 |

11 TITLE (Include Security Classification)

Advanced Dynamic Anthropomorphic Manikin (ADAM) Final Design Report

12 PERSONAL AUTHOR(S) Bartol, Aileen M., Hazen, Vernon L., Kowalski, Joseph F., Murphy, Brian P., and White, Richard P., Jr.

| 13a TYPE OF REPORT | 13b TIME COVERED | 14 DATE OF REPORT (Year, Month, Day) | 15 PAGE COUNT |
|---|---|---|---|
| Final | FROM Feb 85 TO Jun 89 | 1990 March | 649 |

16 SUPPLEMENTARY NOTATION

| 17 COSATI CODES | | |
|---|---|---|
| FIELD | GROUP | SUB-GROUP |
| 23 | 02 | |
| 23 | 04 | |

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Bioengineering, biomechanics, biodynamics, anthropomorphic manikin, dummy, ejection, safety, crashes

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The USAF has embarked on a new effort to design and develop an Advanced Dynamic Anthropomorphic Manikin (ADAM) with improved biofidelity and instrumentation over currently available escape system testing manikins. This report describes the design of three prototype (one small, one mid-size, and one large) instrumentated, anthropomorphic manikins for testing, evaluating, and qualifying high-performance aircraft escape systems, including restraint and harness systems. Among the required responses are that it provide a human-like reactive live load into the ejection seat and possess realistic dynamics and kinematics due to windblast, impact, vibration, and acceleration forces representative of those encountered during ejection from aircraft. In addition to improved biomechanical response properties, the manikin has instrumentation and an on-board data acquisition system to measure, store, and transmit its responses and the data from the escape system. Two prototype manikins, a small and a large, will be fabricated and tested at Wright-Patterson AFB OH.

| | |
|---|---|
| 20 DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | 21 ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED |
| 22a NAME OF RESPONSIBLE INDIVIDUAL<br>ROY R. RASMUSSEN | 22b TELEPHONE (Include Area Code) (513) 255-3665<br>22c OFFICE SYMBOL AAMRL/BBM |

DD Form 1473, JUN 86 — Previous editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

## SUMMARY

Increases in high speed and altitude performance of current and planned high performance aircraft and the persistent high rate of fatality and injury associated with the operation of current aircraft are driving research programs to develop better crew escape and protection systems. To demonstrate the design of these escape and protective systems, it is mandatory that ejection tests be conducted with manikins designed to react dynamically as a human does. Such manikins must have individual body segments with proper centers of mass, moments of inertia, articulation flexibility and deformation similar to that of a human. Present manikins tend to be very crude human analogues with respect to dynamic and kinematic responses and are inadequate for evaluating ejection seats for which human body dynamic reactive forces are a factor in seat performance. The Advanced Dynamic Anthropomorphic Manikin (ADAM) is a United States Air Force program to design and fabricate an advanced instrumented manikin suitable for use in high performance aircraft escape system testing at airspeeds up to 700 knots equivalent air speed (KEAS). In addition to improved biomechanical response properties, the manikin has extensive sensors for acceleration, force and joint position measurement, and an on-board data acquisition system to record and transmit these responses and the data from the escape system.

Specifications for three manikins are to be developed, designated as small, mid-size, and large corresponding to a 3rd, 50th, and 97th percentile male Air Force aviator. ADAM will be based on Air Force male flight crew anthropometry with refined segment and total body inertial properties and proper joint articulation and motion resistive properties. The mechanical design of each major joint will emphasize human biofidelity. ADAM's spinal elements are designed to have elastic and viscous properties such that its seated dynamic responses are similar to those of a seated human.

Although ADAM's primary objective is to provide an instrumented manikin for high speed ejection seat tests, it has other applications as well. ADAM can be used to test head or helmet mounted equipment such as chemical defense or avionics equipment. The manikin can be used to evaluate windblast protection concepts, the effectiveness of capture/haulback active restraints and parachute opening shock. It has applications in crash attenuation seat tests involving energy absorbing seats such as those found in helicopters. ADAM can be used in car crash impact/rollover testing where a self contained instrumented manikin would be useful in evaluating occupant motion and interaction with the vehicle.

## PREFACE

The research and development program reported on herein was conducted under Air Force Contract F33615-85-C-0535, "Advanced Dynamic Anthropomorphic Manikin (ADAM)," for the Harry G. Armstrong Aerospace Medical Research Laboratory (AAMRL). The Air Force Program manager was Roy R. Rasmussen of the Biomechanics Branch, Biodynamics and Bioengineering Division of AAMRL. This program was accomplished by the Advanced Systems Development Center (ASDC) of Systems Research Laboratories, Inc. (SRL), a division of the Arvin/Calspan Corporation. The SRL Program Manager was Vernon L. Hazen of the Aerosystems Division of ASDC.

Dr. Ints Kaleps, Chief of the Biomechanics Branch, Biodynamics and Bioengineering Division, AAMRL, deserves special thanks for his continued support and direction. Gratitude is expressed to James W. Brinkley, Director of the Biodynamics and Bioengineering Division, AAMRL for his guidance throughout the program. Acknowledgement is made to Air Force personnel Stephen R. Mehaffie, Capt Ray L. Keele, 1Lt Karen E. H. Cooper, and Major Kenneth W. Nelms of the Crew Escape Technologies Program Office, Directorate of Human Systems Technology, Human Systems Division (HSD/YAH) for their valuable contributions to the ADAM program. Acknowledgment is also made to Gregory F. Zehner, Workload and Ergonomics Branch, Human Engineering Division, AAMRL and James H. Eggleston, 6585th Test Group, Holloman Air force Base, NM for their assistance throughout this development program.

Special acknowledgement is expressed to SRL Vice President and Director of ASDC, Dr. Karlheinz O.W. Ball; ASDC Senior Scientist, Richard P. White, Jr.; and Aerosystems Division Manager, William T. Carter, for their continued support and guidance throughout the program. Other SRL personnel who were key participants in the program were prior Program Managers William T. Carter, Richard Claxton, John Mantei, and Lee E. Smith; Chief Engineer William E. Nettles; Mechanical Engineers Aileen M. Bartol and Brian P. Murphy; Electronic Engineers Thomas W. Gustin, Joseph F. Kowalski, and James R. Bolton; Designer Draftsmen Glenn L. Thomas and James L. Fox; Manufacturing Engineer Edward W. LeMaster; Machinists/ Modelmakers James R. Trangenstein, Jay R. Jenkins, and Charley M. Peacock; Quality Engineer Charles A. Hipple; Quality Technician Tina M. Cooper; Production Supervisor Richard F. Fletcher; Mechanical Technician Gregory A. Thompson; Electronic Technicians Keith O. Parson and Dale L. Price; Technical Editors/Writers Sharon Seitz and Mary Wesch; and Technical Typists Marilyn Ernst and Carleen Shumway.

Accession For

Dist

A-1

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

## LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

## LIST OF TABLES

LIST OF TABLES (continued)

LIST OF TABLES (continued)

# Section 1
# INTRODUCTION

As the performance of combat aircraft increased during the World War II era, escape from disabled aircraft became increasingly more difficult and likely to cause crewmember injury. In order to increase the potential for safe escape of the crewmember from a disabled aircraft, the ejection seat concept was developed. The early seats were relatively successful in that, while generally unstable, they allowed the crewmembers to clear the aircraft and accomplish a safe parachute descent to the ground. As the aircraft speed and maneuverability characteristics were enhanced, the performance of the ejection seat was also improved.

The design approach used for the ejection seats in the 1950s was to design the seat for the aircraft in which it was going to be installed. This approach resulted in approximately 20 different ejection seats/systems, each with unique support requirements, performance envelopes, construction systems, parachutes, and procedures. In the late 1960s, the Air Force initiated a program to develop an improved ejection seat that could be used in a majority of operational aircraft. This program, Advanced Concept Ejection Seat (ACES), had as its primary objective the enhancement of successful and safe ejection at high speeds by means of improvements in reliability, stability, commonality, and improved maintainability and logistics support. These objectives have been met with the ACES II seat as indicated by the saving of 66 lives out of 71 attempts without experiencing any in-the-envelope fatalities. While the basic ACES II seat was developed using the state-of-the-art technology existing in 1967, the advances in aircraft capability have resulted in ejections using the ACES II which occur outside the operational design envelope. In recognizing the need for improvements in the operational capabilities of the seat to provide successful ejections from disabled aircraft, an ACES II upgrade program was initiated. A primary development in the ACES II upgrade program is to provide a limb restraint system to protect the crewmember from limb flail due to extreme windblast associated with ejection at high dynamic pressures. Other improvements to the ACES II seat will be the incorporation of an Advanced Recovery Sequencer (ARS) to provide a larger ejection envelope and an Automatic Inflation Modulation (AIM) parachute to provide the capability to open the recovery parachute with improved force-time characteristics.

While the upgrade of the ACES II seat will provide enhanced safety to the crewmembers during ejection from current high speed aircraft, it will not meet the needs of future aircraft which will have capabilities significantly different than the current aircraft. For this reason, the Air Force has embarked on the development of a new ejection seat incorporating advanced systems which will result in a seat having enhanced capabilities for providing safe escape from a disabled aircraft.

This advanced development program, called Crew Escape Technologies (CREST), will develop an ejection seat which will replace the fixed performance characteristics of existing escape systems with a system where performance is continually determined by a computer system that will sense the conditions encountered prior to and during an emergency, assess the life threat presented by those conditions, and control the performance of the ejection seat subsystems to ensure the greatest chance of successful escape from disabled aircraft with the least possible risk of ejection system-induced injury or fatality.

The test and evaluation of the new ejection systems being developed (ACES II Upgrade and CREST) will require a suitable manikin to duplicate the human response during an ejection sequence. The three basic requirements which the manikin must meet are humanlike dynamic response, durability, and advanced instrumentation. In order to simulate humanlike dynamic response characteristics, the manikin must have individual body segments with the proper weights, centers of mass, and moments of inertia, as well as articulation flexibility similar to that of a human.

In order to meet the durability requirements, the manikin components must be sufficiently strong to withstand the rigorous ejection environment. Meeting this requirement while simultaneously meeting the humanlike dynamic response characteristics provides a challenge. The advanced instrumentation system is unique in that its innovative design incorporates features not previously available.

At the present time, the majority of ejection tests are conducted with the GARD or center of gravity (CG) dummy designed in the 1950s that are crude representations of the human. While the body shapes and sizes are somewhat representative of the human, the body and limb articulations are very limited and the mass characteristics, weight, CG locations, and inertia properties poorly represent similar components of the human body.

While the biofidelity of manikins for ejection testing has not been improved from the original design, manikins developed for use in testing of safety devices in automobiles have been developed with increasing biofidelity over the last two decades. An in-depth review of the state-of-the-art of manikin development for both automotive and ejection testing is presented by Bateman et al. (1984). It was concluded, by Bateman et al., that the state-of-the-art of manikin development for ejection testing is far behind the technology of the ejection seats for which they were providing the human analog. It was also concluded, by Bateman et al., that, while the biofidelity of the manikins developed for injury investigation during automobile crashes was far more representative of the human, they were not suitable for use as manikins for ejection testing as they were not designed to resist the severe G loading and aerodynamic blast effects associated with ejection from a high speed aircraft.

Recognizing the lack of a suitable manikin to test the new limb restraint system being developed for the ACES II Upgrade system, the Air Force initiated the development of a new manikin to test these systems at speeds up to 700 KEAS. This new manikin, called the Limb Restraint Evaluator (LRE), incorporates significant increases in biofidelity over that present in the most advanced automotive manikin and has been designed to resist the aerodynamic forces associated with ejections at 700 KEAS (White, Gustin, and Tyler, 1984). In addition to the high degree of biofidelity incorporated into this manikin, an instrumentation system which provides for the onboard storage of 96 channels of data, as well as telemetry, has been incorporated into the 95th percentile size manikin. While the mass and size characteristics of this LRE manikin, for evaluating limb restraint systems, duplicates the dimensional/mass characteristics of the VIP-95 manikin developed for automotive crash testing, the body shape and mass characteristics are not necessarily representative of the 95th percentile size of the Air Force pilot population. Since the mass and inertial characteristics of the limbs about the joint articulation were carefully duplicated, however, the dynamic response of the limbs from dynamic G loadings and aerodynamic forces should reasonably duplicate that of a human.

While the LRE development program provided a quantum jump in the degree of biofidelic representation and in instrumentation technology incorporated into a suitable ejection dummy, its use in the CREST program is marginal since some of the important dynamic response effects associated with the human spine are not modeled and the LRE was designed to represent only the large size male. The dynamic response of the spine in the vertical direction, particularly the lumbar and cervical sections, are important motions describing the forces and moments which the human body will apply to the ejection seat and which the gimballed rockets in the CREST system must be designed to counteract.

The technical sections presented in this report will discuss, in detail, the design and test efforts that were conducted to meet and prove the stringent requirements regarding the biofidelic representation of the anthropometry, mass, and response characteristics of a small and large male human aviator.

## Section 2
## MECHANICAL DESIGN

### 2.1. GENERAL DESCRIPTION OF THE ADAM

The ADAM was designed to accurately represent a designated human population for the testing of ejection seats. Dimensional, mass property, and response characteristics representative of approximately the 3rd and 97th percentiles of a tri-service population of male aviators were designed into the small (shown in Figure 1) and large manikins. The instrumentation system within the manikin is designed to record and store various joint rotations, accelerations, and forces within the manikin and various seat parameters for a total capability of 128 channels of data.

Figure 1. Small ADAM

This section will present a physical description of the ADAM and discuss the manner by which the design of the manikin attempted to achieve the desired characteristics of the human it represents. In addition, the physical characteristics of the transducer/data instrumentation system incorporated in the ADAM will be discussed.

### 2.1.1. Specialized Features of ADAM

The sectional drawing presented in Figure 2 shows some of the special features incorporated in the ADAM design. While the majority of the ADAM components were specially designed to meet the design specifications, it was determined that four off-the-shelf items could be utilized. These four items were the head which is manufactured for the Hybrid II manikin, the Hybrid III flexible neck, and the hands and feet which are fabricated for the VIP manikin. The Hybrid II head adequately met the size requirements for both the small and large ADAM systems but had to be ballasted to properly meet the mass requirements for both the small and large ADAM. A standard four-section Hybrid III neck was used in the large manikin and a three-section neck was used in the small manikin. The hands and feet were modified to accommodate the ADAM bones. To meet the size requirements, the VIP foot was shortened for the small manikin.

A damped/elastic spine as shown in the sketch (Figure 2) provides an elastic degree of freedom between the upper torso and the pelvis in an attempt to simulate the elastic deformation of the human body in the vertical direction during dynamic Gz loading. This degree of freedom is also a major parameter in obtaining the desired impedance characteristics in the Gz direction with the frequency range of 0 to 30 Hz.

The other major features illustrated in this sketch are associated with the unique instrumentation system designed for ADAM. In order to measure the loads developed at critical areas within the manikin for comparison within critical human loadings in these same areas, six component load cells are placed at the head/neck attachment point as well as at the attachment of the spine to the pelvis. In addition, two single axis load cells are located in the lower leg to measure the loads when the tibia rotation reaches the limits of its motion. The entire instrumentation system, signal conditioners, A/D conversion circuit, and memory for 128 data channels are located within the viscera.

### 2.1.2. Details of ADAM Features

While the sketch presented in Figure 2 illustrates some of the main unique features that are incorporated into ADAM, many others are also included to meet the desired goals. These additional features will be discussed in more detail in the following paragraphs.

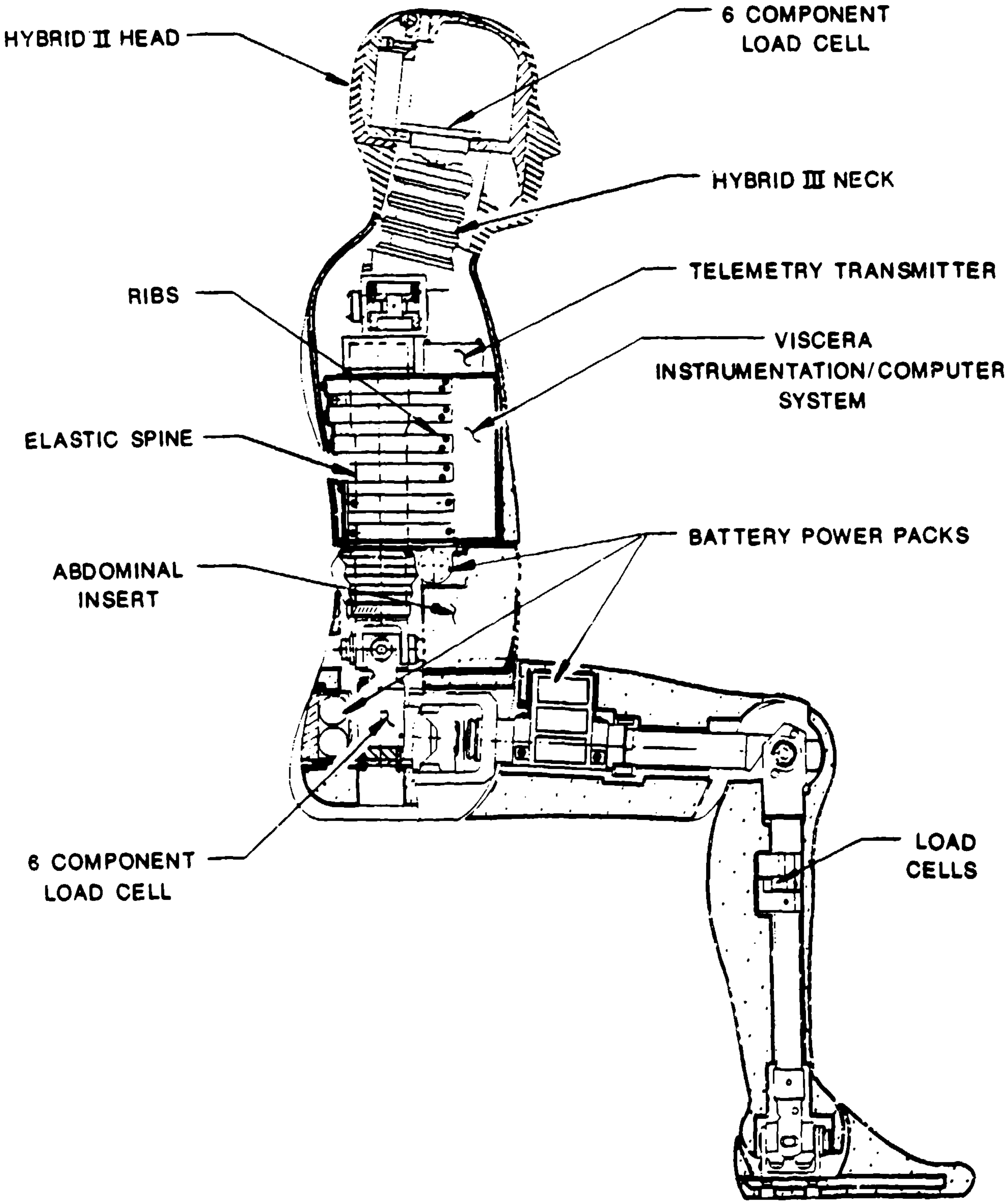



Figure 2. ADAM Special Features

### 2.1.2.1. Anthropometry and Mass Characteristics

The anthropometry and mass characteristics to which the small and large size manikins were designed are based on the tri-service database, "Anthropometry and Mass Distribution for Human Analogues, Volume I: Military Male Aviators," March 1988, AAMRL-TR-88-010. The data contained in this handbook specify the anthropometry, joint center locations, mass, center of gravity (CG) location, and inertial characteristics of the various body segments. As shown in Table 1, the measured overall manikin dimensional and mass characteristics compare favorably to the tri-service specifications. The detailed dimensional and mass characteristics of the manikins are presented and discussed in later sections of this report.

TABLE 1. COMPARISONS OF MEASURED PARAMETERS WITH SPECIFICATIONS

| Characteristic | Small Manikin | Percent Deviation From Specifications | Large Manikin | Percent Deviation From Specifications |
|---|---|---|---|---|
| Weight (pounds) | 142.3 | +2.0 | 217.0 | +0.7 |
| Sitting Height (inches) | 34.5 | -2.0 | 37.5 | -3.0 |
| Standing Height (inches) | 66.25 | 0 | 74.3 | 0 |

### 2.1.2.2. Skeletal Structure

The manikin limbs, which are the highly loaded structures of the manikin and are designed to withstand significant dynamic motions if limb flail occurs at speeds up to 700 KEAS, were constructed from 17-4PH stainless steel. This material is a precipitation hardened martensitic stainless steel used for parts requiring high strength and good corrosion and oxidation resistance up to temperatures of 600°F. The use of the high strength steel in the fabrication of the limbs allowed the design of long bones, capable of resisting the applied dynamic loading without failure, to be sized to fit within the skin line defined by the anthropometric specifications. Because of the greater volume and lower per unit loading, the torso structure was constructed from 6061-T6-57 aluminum alloy to reduce the torso weight and, thus, help maintain the proper weight distribution of the entire manikin. The small manikin, because of weight limitations, utilized aluminum parts constructed from 7075-T6 in the shoulder and pelvis areas and was designed to withstand speeds up to 450 KEAS.

2.1.2.3. Skin Contours

The skin contours for the manikins have two main functions: (1) provide the proper outside body contour, and (2) represent the compliance characteristics of human flesh. Since the manikins were built to represent the new tri-service specifications, considerable effort was required to design new mold patterns from stereophotometric data. The specified joint center locations, however, were developed from a different set of data, and the stereophotometric data did not meet the tri-service dimensions exactly. Engineering judgment and a detailed procedure were used to develop a consistent set of data for the outside body contours. The result of this extensive design effort was a set of new skin molds which produce skin contours that represent the small and large human aviator as defined in the tri-service requirements.

The flesh coverings are fabricated from heat cured vinyl plastisol which has a skin-like composition on the inner and outer surfaces and vinyl plastisol foam in between. The inside skin is formed to match the contour of the skeletal structure so that, when the flesh covering is attached, it is maintained in its proper position. At other areas, the inside skin is designed slightly larger than the skeletal structure so that the rotation of the structure does not deform the skin. Because of the need to obtain access to the instrumentation and sensors throughout the entire body, each skin covering has at least one zipper to permit easy removal of the skin from its associated structure. In addition, if a skin covering is damaged during a test, it can be immediately replaced with a new one without removing the structural component from the manikin assembly.

2.1.2.4. Joint Design

In order to duplicate, to the extent possible, the degrees of freedom in the human body, there are 43 points of rotational articulations incorporated in ADAM. A listing of these articulations and the associated ranges of motion is presented in Section 2.2.6 (Table 31).

All of the ADAM joints can be classified into two general categories: rotational sleeve joints and clevis joints. The joints in each category are similar depending on the ranges of motion and the size of the joint in question. Several joints combine the rotational sleeve and clevis concepts to allow more than one degree of freedom. The shoulder and knee joints have unique features and will be discussed in detail in the following paragraphs.

The ADAM shoulder mechanism is shown in Figure 3. This joint has five independent degrees of freedom: extension/flexion, traverse abduction/adduction, coronal abduction, elevation/ depression, and pronation/retraction. Coronal abduction and traverse abduction/adduction are achieved at the outer block through the use of two pins which intersect at the joint center. The third degree of freedom, flexion/extension, is a rotation of the outer block. A stop ring was required to allow the 235 degrees of motion. The fourth and fifth degrees of freedom are the elevation/ depression and pronation/retraction of the outer block about a point under the neck block (sterno-clavicular joint center). As shown in the photograph, concern is given to the wire routing such that minimal interference with the joint rotation is achieved. Notice the soft stops, friction mechanisms, and transducer assemblies located throughout the joint.

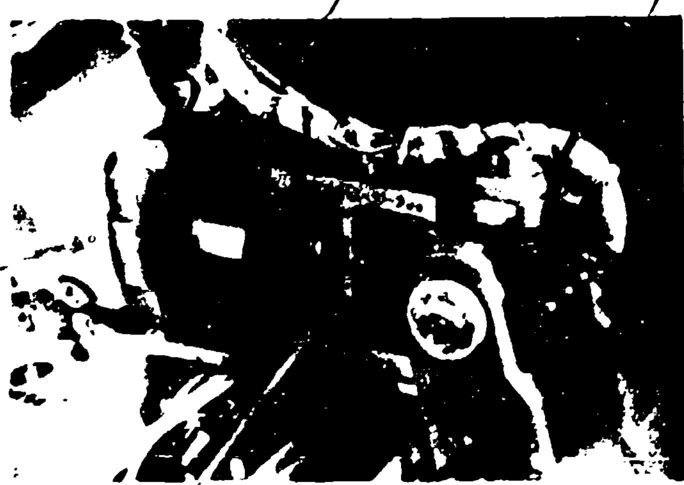



Figure 3. ADAM Shoulder Mechanism

The knee joint is shown in Figure 4. This joint is a standard clevis type joint combined with a rotational joint for lower leg rotation with the capability of allowing full lower leg rotation when the leg is in 90 or more degrees of flexion, and no rotation when the leg is in 0 degree of flexion. This feature is achieved using a triangular block which mates to a U-shaped stop at the 0 degree of flexion position.

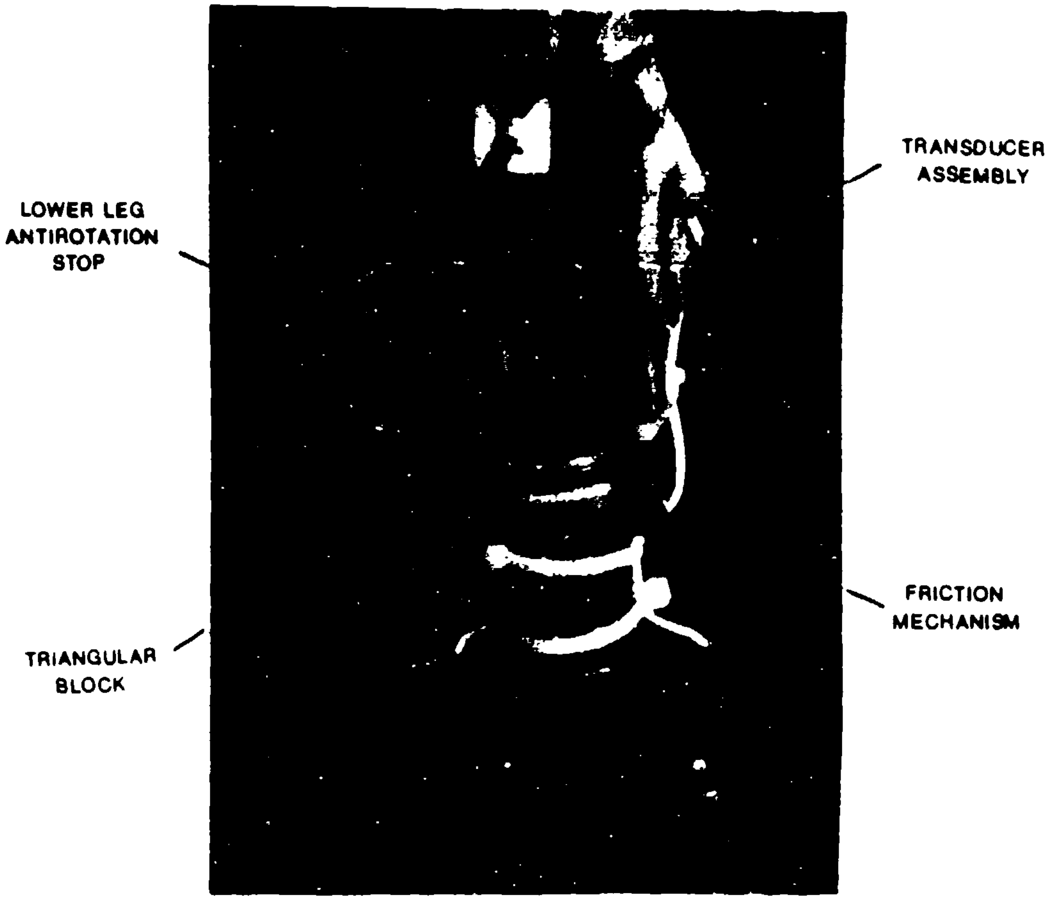



Figure 4. ADAM Knee Joint

In addition to providing points about which the various body segments can rotate, the joints provide human-like resistance to motion similar to that developed by muscles and tendons. This resistance to joint rotation manifests itself in the human body by a constant resistance torque over the ranges of free joint rotation and an increasing torque resistance as the limb reaches its limits of rotation.

Some of the features of the resistive mechanisms in the manikin joints are:

- They are insensitive to temperature, humidity, and other environmental conditions.
- They are adjustable and repeatable.
- They do not interfere with the instrumentation sensors measuring joint rotations.

Figure 5 illustrates the elbow joint in ADAM. Notice that one side of the joint is pulled against a single arm of the clevis. This reduces the effect of temperature on the set torque of the joint. Using a fine, clean thread on the pin, it was determined that a joint torque could be repeatably set to within 10 percent by measuring the torque applied to the nut.

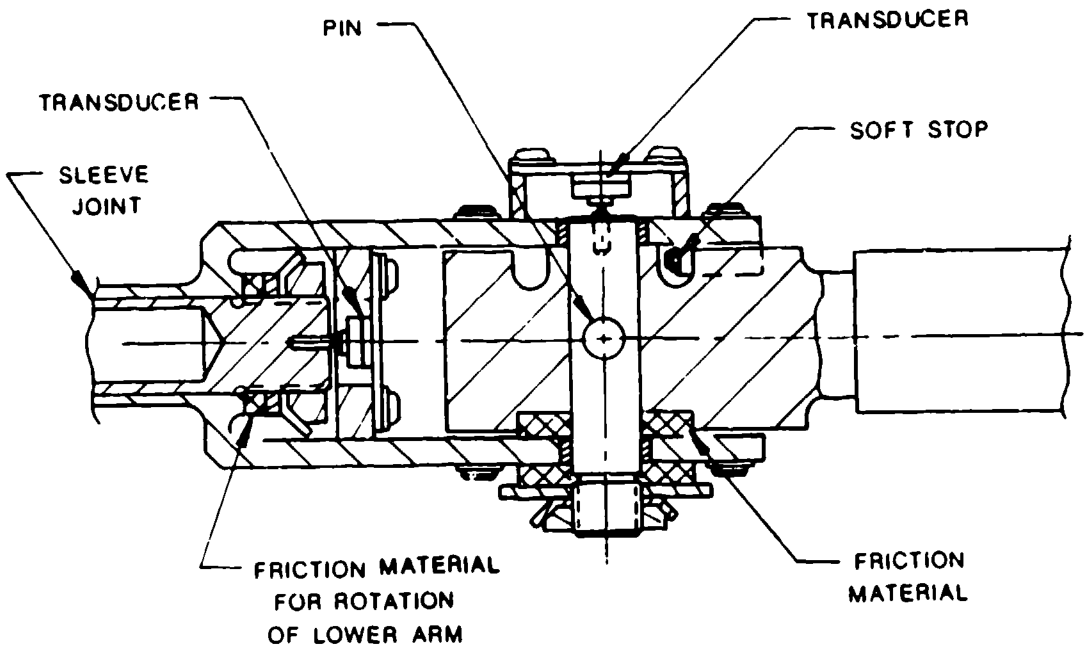



Figure 5. Small ADAM Elbow

Also shown in Figure 5 is the application of this technique to a rotational sleeve joint. The transducers used to measure the joint rotation are also illustrated in this figure. As can be seen, the resistive mechanism does not interfere with the operation of the transducer measuring joint rotation.

The design of "soft stops" to duplicate the increasing resistance to joint rotation as the limits of rotation are being approached followed a similar design/test approach that was utilized to design the resistive torque mechanisms. The stops for all joints are of a trapezoidal design and are fabricated from polyurethane. Of all the materials tested, polyurethane was chosen because of its high load-bearing capability, temperature stability, and its excellent resistance to oils, solvents, grease, etc. Static testing of the polyurethane demonstrated the increasing resistance characteristics found in humans, and dynamic testing demonstrated the integrity of the mechanism for stopping high joint rotational velocities during a typical high speed ejection if limb flail occurs.

2.1.2.5. Spine Design

During an ejection sequence, the human spinal system undergoes a complex series of deformations and bendings which are dependent on the upper body restraint system, the initial positioning of the ADAM, and the loading on the manikin. These human spine response qualities are incorporated in ADAM by using a mechanical spring/damper system located in the ADAM spine. The final design of the semielastic spine is shown in Figure 6.

In order to achieve the human-like response and impedance characteristics, the small and large ADAM spinal systems both consist of a helical spring which promoted a 10 Hz natural frequency of the torso in the z direction and a hydraulic damper to provide the required damping of the spine motion. Through an extensive testing procedure, MIL-H-5606 was selected as the damping fluid for the damper as it achieved approximately a 60 percent critical damping over a wide range of temperatures. In addition to the dynamic motion in the z direction, the mechanical spine allows the upper torso to pitch and roll with respect to the pelvis at the lumbar pivot point. Yaw motion of the upper torso is achieved by rotation between the outer and inner sleeves of the spring/hydraulic damper piston.

The dynamic testing of the spine conducted at SRL demonstrated that it will operate properly in the severe Gz ejection environment and, thus, properly approximate the reaction of the human spine to static and dynamic Gz loading.

2.1.2.6. Instrumentation System

The primary function of the instrumentation system is to provide an onboard and a redundant data gathering and recording system to ensure that all data are obtained during an ejection sequence for future analysis. The basic concept of the computer controlled instrumentation system is an extension and improvement of that previously designed for the LRE which was successfully demonstrated during ejection tests on the sled track at Holloman Air Force Base. The ADAM system, as configured for supporting the test of the CREST development program, provides the power, signal condition, and A/D conversion for 63 data channels within ADAM. In addition, 56 channels of data from the CREST seat will also be supported by the ADAM instrumentation system. The data from both sources will, herefore, utilize a majority of the 128-channel capacity of the onboard instrumentation system. The redundancy of the instrumentation system is in the ability to provide

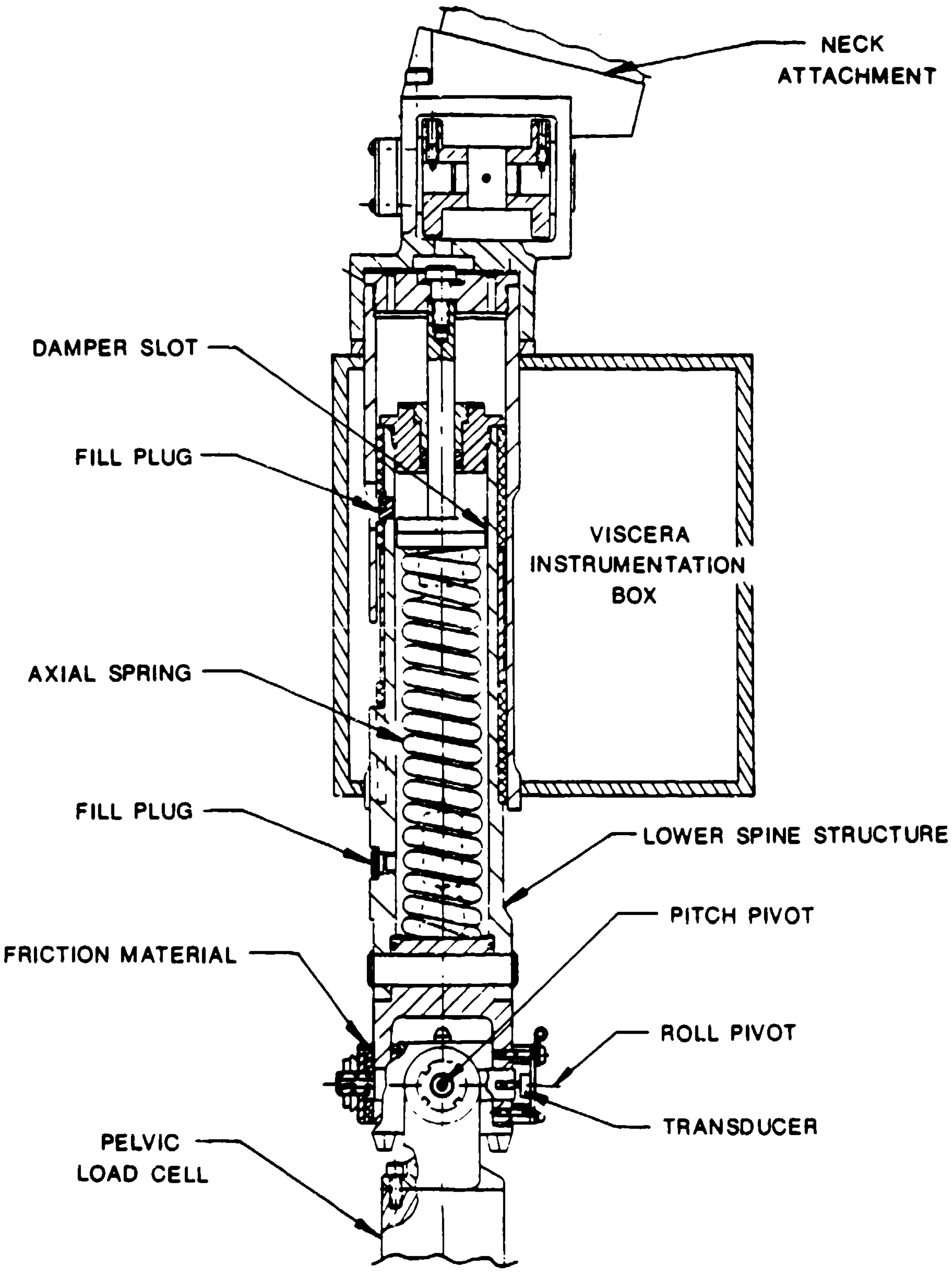



Figure 6. ADAM Semielastic Spine

complete onboard storage of all data in battery backed up SRAMs and through the telemetry of all data to a remote ground station.

Some of the major features in the instrumentation are:

- Operation at dynamic loadings up to 60 Gs.
- Operation at ambient temperatures up to 158°F.
- Packaging of the system in a limited volume.

As noted in Figure 7, the entire instrumentation system, signal conditioning, A/D conversion, memory, and computer control functions are located in the viscera/instrumentation box. The four 6 7/8-inch by 4 1/4-inch circuit boards located on the left side of the viscera are the digital boards which contain the data memory, the A/D conversion system, communi cations, and computer control system. The three boards on the right side are the analog circuits which contain the signal conditioning and multiplexers.

Figure 7. ADAM Viscera Instrumentation System

The mother boards are in the forward portion of the viscera box so that the high Gx loading during ejection will tend to force the daughter boards into the mother boards and maintain good pin connections and, thus, circuit continuity.

As previously noted and shown in the schematic presented in Figure 2, the power to operate the system during ejection is obtained from lithium batteries at different locations in ADAM.

Lithium batteries were chosen because of their high output rate and the large storage capacity. Each double D cell located in the abdomen and buttocks sections has a 30 amp hour rating, while each of the 12 D cells in the legs have a 13 amp hour rating. The battery power supply incorporated in ADAM will operate the full-up system for approximately 1 hour during the test; in the rest mode which is maintained after data collection, the system will operate for an additional 2 hours.

The above describes the system in general. The following sections will discuss the design effort involved to develop each of the major areas of design, including the instrumentation, and several elements of the mechanical system.

## 2.2. SYSTEM ANALYSES AND TESTS

### 2.2.1. Anthropometric Design

The design of the ADAM skin contours determines the drag characteristics of the manikin segments and, therefore, influences its response to aerodynamic loading. Proper skin contour design is also critical for the proper interfacing of the manikin with restraint harnesses, seat configurations, and the mounting of flight equipment. An extensive effort was required to assure that each of the manikin's contours were humanlike and matched the dimensions of the small and large male aviator as described in the ADAM Statement of Work, USAF Contract F33615-85-C-0535, Advanced Dynamic Anthropometric Manikin (ADAM), Systems Research Laboratories, Inc., 11 September 1985. The humanlike characteristic is unique to the ADAM in that the contours are mathematically derived from actual human data rather than being an artist's representation of a human form.

This section will describe the effort required to design the skin contours of the ADAM. It will cover the system requirements, the design techniques, and the final design characteristics of the ADAM.

#### 2.2.1.1. System Requirements

The two requirements of the ADAM skin design were to create humanlike skins, in general, and those which represent the small and large male aviator, specifically.

The first requirement of the design was to match the dimensions as specified in the Statement of Work for the ADAM, USAF Contract F33615-85-C-0535, Advanced Dynamic Anthropometric Manikin (ADAM), Systems Research Laboratories, Inc., 11 September 1985. Included in these dimensions are various breadths, widths, and circumferences, as well as the joint centers of each segment. Body segments were defined to include all data found between consecutive joint centers. These data describe two sizes of a standard military male aviator and will be referred to as the tri-service data.

The other requirement was that of attaining humanlike skin contours. This requirement was not satisfied by simply estimating the human shapes but by utilizing actual human data and applying the data to the manikin design.

The human test data used were compiled in 1969 by the Texas Institute for Rehabilitation and Research. Thirty-one test subjects in a standing position were measured using stereophotometric techniques. The resulting data consisted of body surface point coordinates organized in horizontal or x-y plane cross-sectional slices at two centimeter intervals. A sample set of data from one subject is shown in Figure 8. For the ADAM design, a single subject data set was not appropriate as it deviated from the tri-service data set on most segments. To obtain a data set which closely represented the tri-service data for all segments, a collection of segments from various subjects was used.

2.2.1.1.1. Data Relationships

In order to design the manikin skin contours, the locations of the tri-service joint centers had to be known with respect to the stereophotometric test data in both the average standing subject data set and the individual segment data configurations.

The set of relationships between the tri-service joint centers and the average standing data was determined using the anatomical axis systems as a common reference for each segment. These segment based axis systems are defined by surface points or landmarks which have been defined in the stereophotometric data. A global axis system, as shown in Figure 8, was used in the testing of subjects to relate the stereophotometric data to a reference frame. Each segment anatomical axis system, therefore, could be related to the global axis system in the form of a transformation matrix. Since all 31 subjects had similar but not equal transformation matrices, an average set of matrices, which was applicable to any size human, was obtained through the simple averaging of the matrix elements.

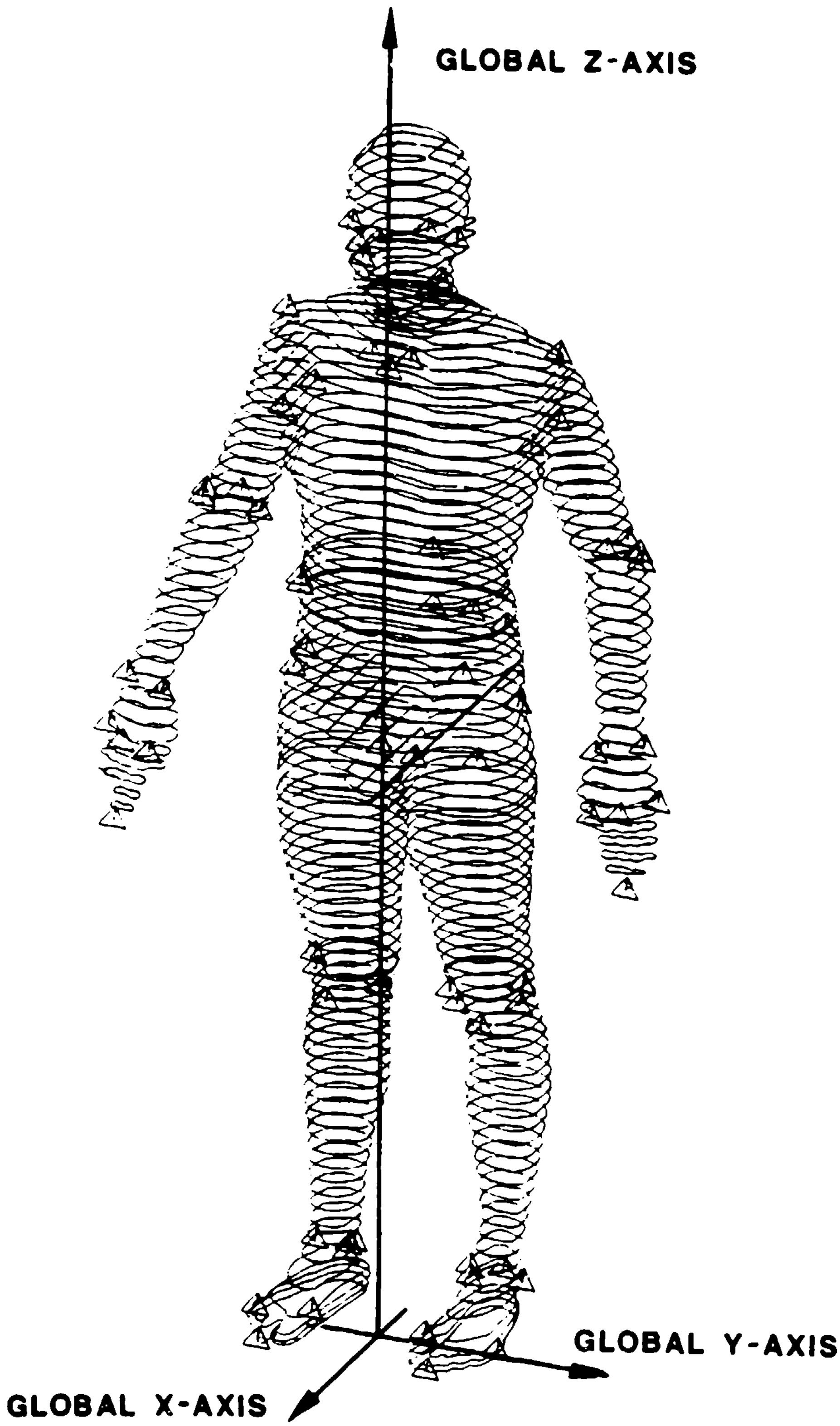



Figure 8. Stereophotometric Surface Data (Subject No. 3)

Since the joint centers in the tri-service data set were originally defined with respect to the anatomical axis systems, the relationships between the two sets of data could be defined using these axis systems. The joint centers were related to the average subject standing position through the calculation:

$$\vec{P_g} = (A)_{ga}\vec{P_a}$$

where

$\vec{P_g}$ = vector in the global axis system
$(A)_{ga}$ = transformation matrix from the anatomical to the global axes
$\vec{P_a}$ = vector in the anatomical axis system (tri-service joint centers)

The segments were then translated to form a standing man.

The result of applying the transformation matrices and translations to the joint centers was a stickman (as shown in Figure 9) consisting of the joint centers arranged in space in the position of the average standing stereophotometric subject. This information was used to determine the locations of the tri-service dimensional requirements with respect to the joint centers. The stickman was also used to define the relationships in space between any two segments in a humanlike position.

The second set of relationships needed was the associated tri-service joint centers with respect to the segment stereophotometric data. This required the placement of the joint centers in the shape data. Basically, by locating the anatomical axis system in the stereophotometric data, the joint centers are easily found as they were defined with respect to the anatomical axis systems. This information was used to define the locations of the bone axes within each segment and the locations of the points which connect adjacent segments (joint centers).

With these data known, the design of the segment skin shapes was initiated.

2.2.1.2. Design Procedure

The procedure used to develop the skin contours for the small and large ADAMs was extensive. After several manipulations of each slice of data, the ADAM was defined. The steps of this process are described in the following paragraphs.

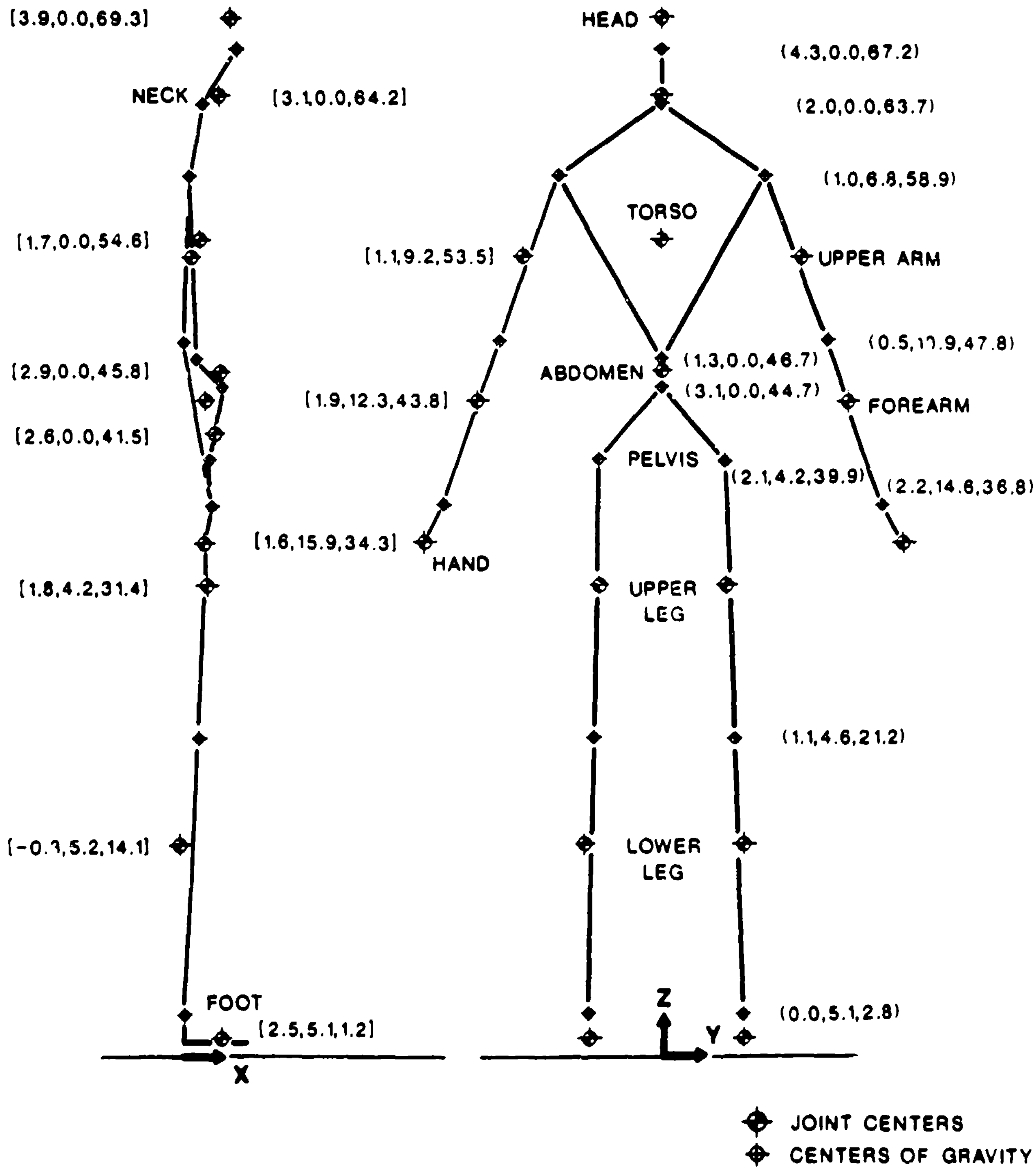



Figure 9. Large Stickman

The stereophotometric data were sorted by the University of Dayton Research Institute (UDRI) to select segments for use in the ADAM design. The sorting procedure included a comparison for each type of segment (such as a forearm) between the 31 segments from different subjects and the tri-service dimensional requirements. Those segments which best met the tri-service dimensions were selected for use in the ADAM data sets. Two sets of segments which defined the small and large manikin were assembled.

Certain additional considerations, such as how to match the connecting skin contours of segments taken from different subjects, were required because the manikin data did not originate from the same subject. These additional considerations will be discussed later.

The segment data sets consisted of the surface point coordinates arranged in planes or slices of data of point thickness which were parallel to the floor (xy plane of the global axis system) which was the position of the subject at the time of measurement. The data were developed in two forms--graphical and mathematical. The graphical form was used for the design of the ADAM skin contours, and the mathematical form was used to develop the above relationships between data sets.

The next step was to place the joint centers in the data sets using the anatomical axis systems as described previously. The connecting line between the joint centers was then located mathematically, and the location of the line passing through each slice was determined. The direction of the global x-axis was also defined on the slices in order to orient the slices with respect to each other and with respect to the segment bone axis systems.

For ease of machining and reduction of the number of unique parts for the manikins, the ability to have only one mold which could be used for the right and left for each limb segment was beneficial. To determine if this was possible, the slices were individually analyzed for a plane of symmetry which was in the direction of the x or y global axes of each slice. It was found that a plane of symmetry could be defined for each limb segment such that only one mold would be required for both the right and left versions of each segment. After the planes of symmetry were determined, each slice was averaged. The method of averaging consisted of superimposing the two halves on each other and determining the contour which halved the difference. By computer, the half contour was duplicated to form a single shape.

In order to check if the symmetry assumption was a valid one, the final slices were compared to the averaged slices. The difference was not significant as the typical distance between the averaged and actual half slices was 8 percent of half the averaged breadth of the slice at that point.

The data were also averaged from right to left on all center segments to create a symmetrical manikin. The limb segments were averaged from right to left by selecting the right hand segments, making them symmetrical, and using them for the left hand side. As well as minimizing the number of molded parts, the averaging of segments minimized any uniqueness to the skin contours. In other words, if any strange protrusions or depressions were present in the data, they were minimized by averaging two contours together.

All cross-sectional data were entered into a CAD system for ease in manipulating the shapes. The profiles of each segment in the front and side views were then generated so that the tri-service requirements could be applied to the data. The joint centers and the bone axes were drawn on the profiles. The outlines of the bone designs were also drawn on the profiles so that any discrepancies between bone and skin could be noted as the design evolved. Figure 10 shows the unmodified profiles for the upper and lower leg segments. As can be seen in Figure 10, there is a mismatch of the profiles at the knee joint.

At the connecting points (joint centers), the data were manipulated to form a continuous contour. This was a difficult step in that the contours were, in most instances, not aligned and a shifting of the contours with respect to the joint centers in one or both segments was required. The upper torso, abdomen, and pelvis connection (Figure 11) was particularly discontinuous. The correction was the movement of both the abdomen and pelvis skin contours with respect to the joint centers.

After the skin contour transitions were continuous, the positions of the tri-service dimensional requirements were located. For example, the elbow breadth was known to be across the elbow center of rotation. Some of these locations were estimated using dimensional measurements from other surveys (Churchill et al., 1978). The nearest slices to the dimensional requirement locations were moved to these locations. These slices were then either expanded or contracted to meet the circumferential, breadth, and/or depth requirements. The new slice profiles were then incorporated into the original profiles and new profiles were drawn to meet the new slices and follow the trend of the stereophotometric profiles. Figure 12 shows the relationships between the new profiles and the original stereophotometric data for the small upper and lower legs.

If any areas of discrepancy between the bone and skin profiles occurred, the area was further analyzed and either the skin or bone was redesigned to omit the conflict. For example, discrepancies in the knee and elbow areas required a shifting of the skin contours to allow the bone outlines to

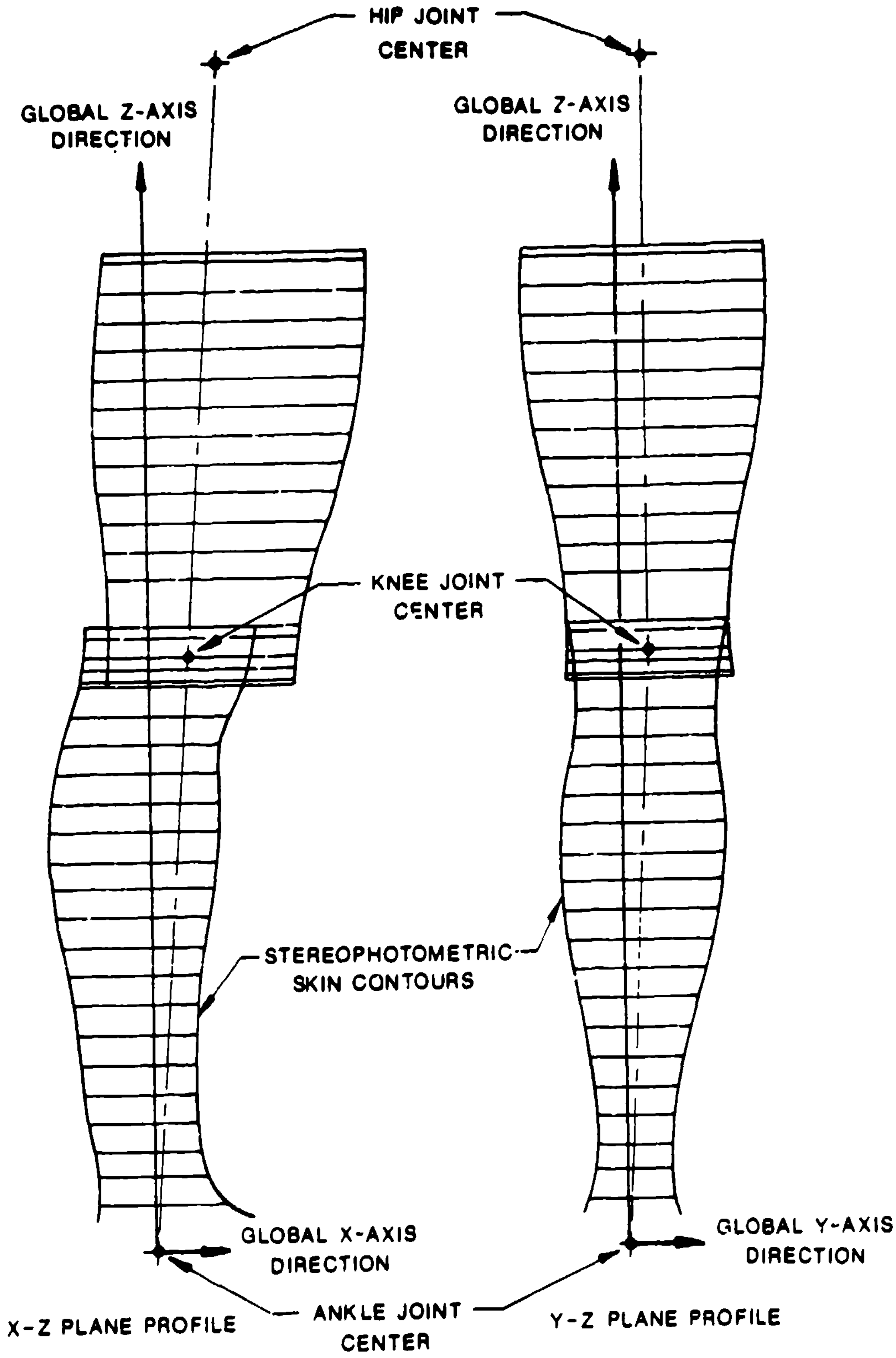



Figure 10. Small Upper and Lower Leg Profiles - Original Data

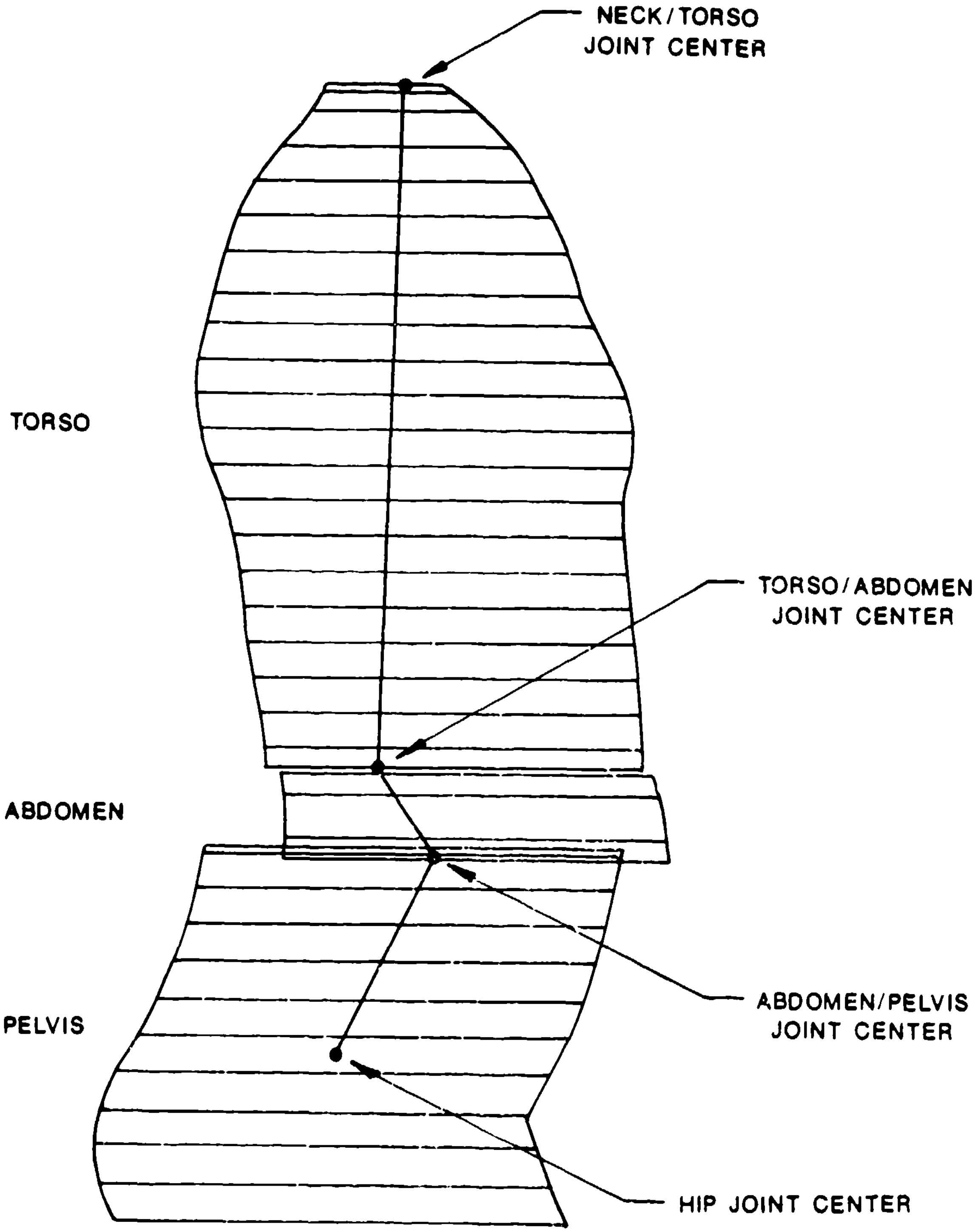



Figure 11. Small Torso, Abdomen and Pelvis Stereophotometric Data

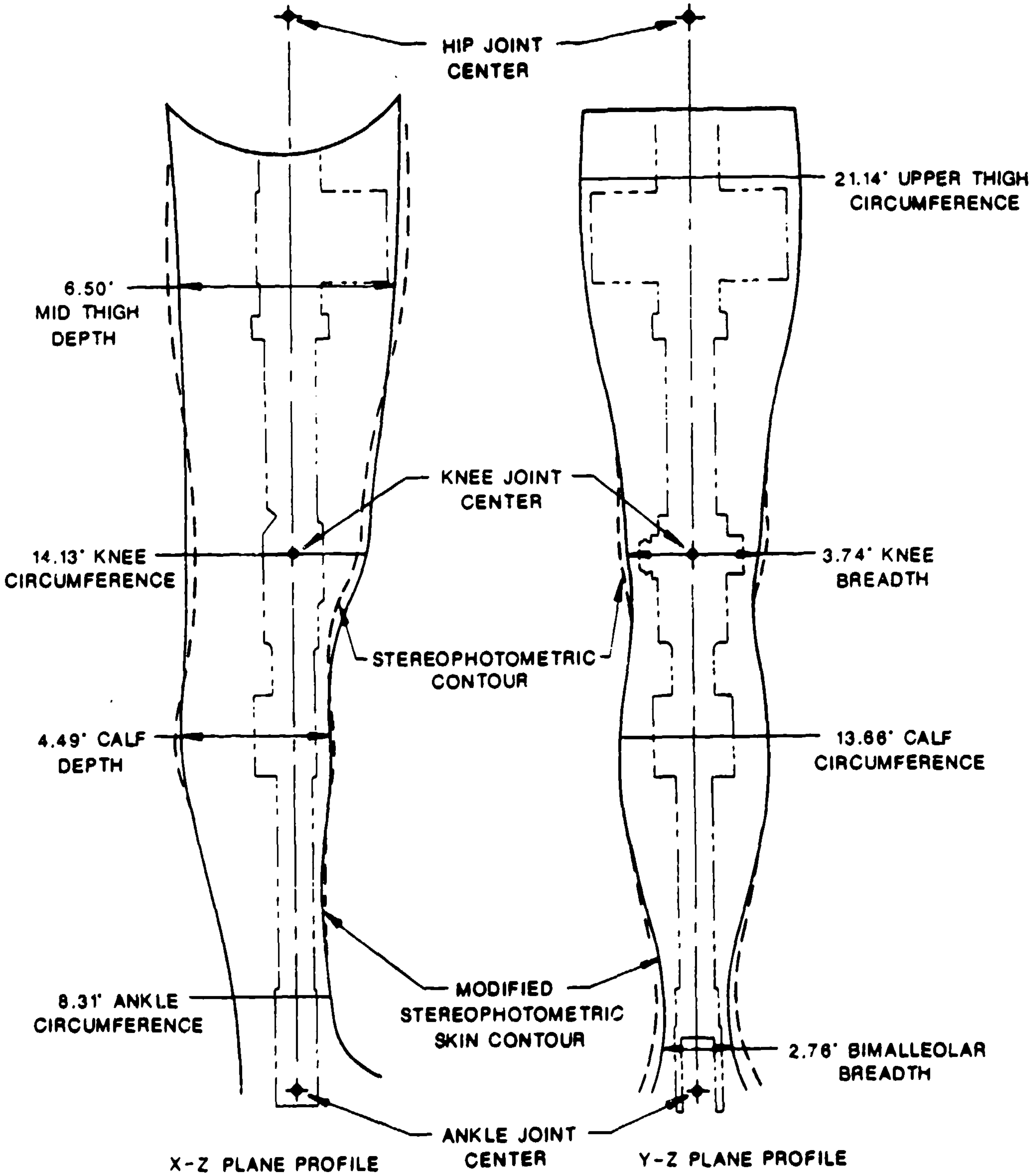



Figure 12. Small Upper and Lower Leg Profiles - Modifications

fall within the skin contours. In most instances, the contours were altered as the bone design was more critical to strength and weight requirements.

All other slices were ratioed in both the x and y directions such that they met the new profiles. This was done on a digital computer for ease and accuracy purposes. In case of discrepancies in the contours, the slices were checked by overlaying the slices on top of each other and viewing any obvious mismatching. During the making of the models, any further discrepancies were omitted by smoothing the contours.

The distances between slices and the corresponding slice shapes were then specified with respect to the joint centers of the segment. The final profile for the leg is shown in Figure 13.

The data were used to define a model for forming a cast aluminum mold. The male models formed from the contour data were used to create female molds. During this standard casting process, the interior dimensions of the mold would decrease by 1.3 percent of the model dimensions. To prevent this from altering the final molds and shapes, the finished contour data were increased by 1.3 percent in the x and y directions and in the spaces between the slices.

Once the outside contours had been defined, consideration was given to the internal areas required by the bones and the external areas required for movement between the segments to give a human-like interaction. This required modification of some of the data for allowance of the movement between segments. For example, the upper leg contour in the knee and upper thigh areas are shaped for the allowance of the movement with the lower leg and pelvis segments, respectively. Inserts to the molds were used to create inside voids in the skins which allowed space for the bones. For the nonrotating bones, these matched the bone dimensions; for the rotating bones, these allowed for the movement of the bone within the skin without altering the skin.

Not all skin segments were designed using the method described above. Three segments from each the large and small and one from the large required slightly different procedures due to the uniqueness of the data. These were the pelvis, the upper and lower arms, and the large upper torso, respectively. One other segment, the small upper torso, required a modification due to the design of the mechanical system.

The upper and lower arms are unique in that the contour data were collected with the arms in approximately 20 degrees of abduction. This resulted in the slices being nonperpendicular to the bone axes. Since the tri-service data were defined in planes perpendicular to the long axes of the

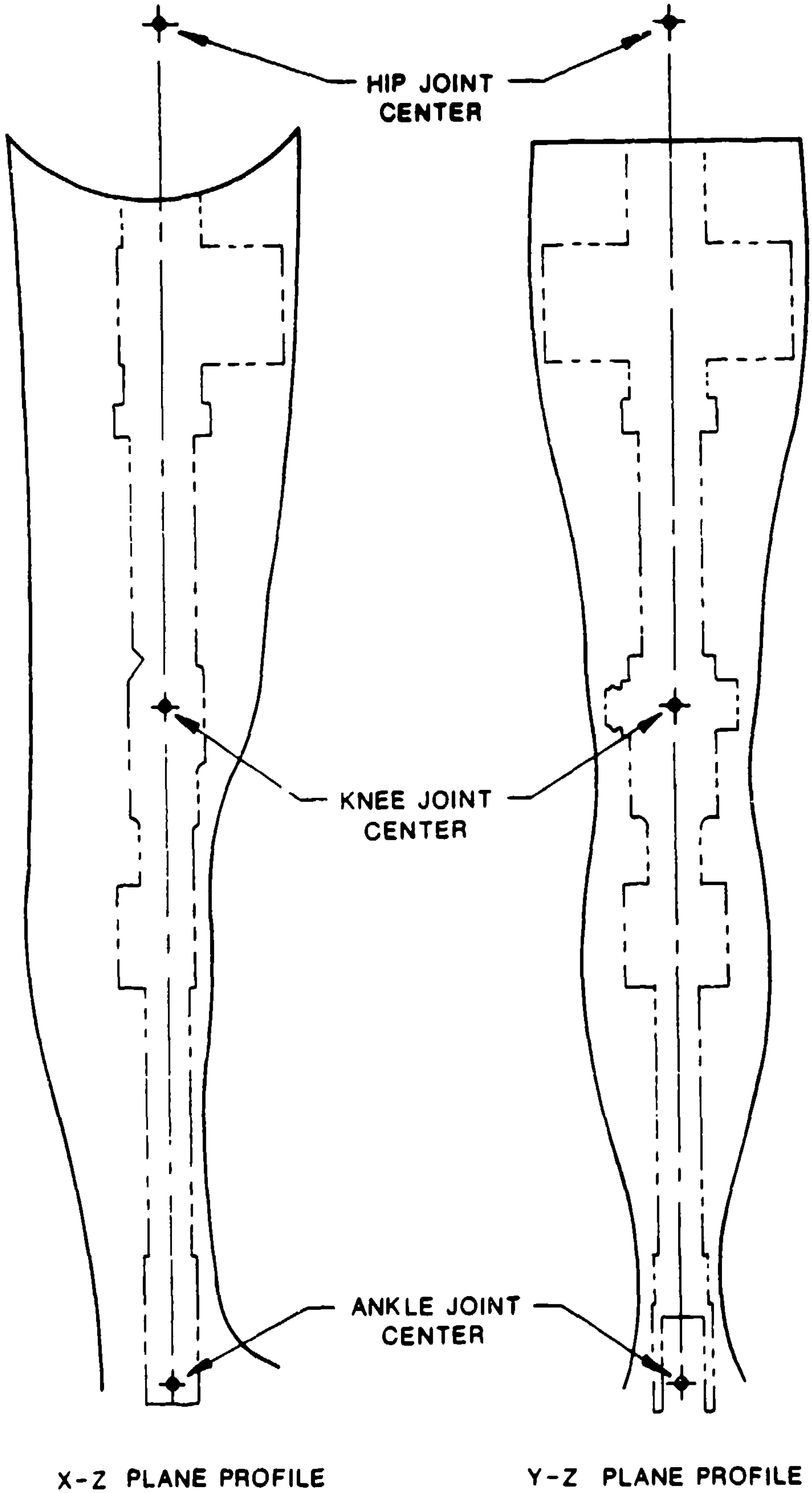



Figure 13. Small Upper and Lower Leg Profiles - Final Design

segments, the two data sets could not be related to one another. A special lofting technique was utilized to develop shapes in the perpendicular planes using the original data. Once the data were defined in perpendicular planes to the long axes, the data were fit to the tri-service data using the procedure previously outlined above.

Another segment which required special attention was the pelvis. The stereophotometric data set defines a "standing" pelvis which is not directly compatible with the ADAM design as ADAM was to be a sitting manikin with the ability to stand. Modifications to the standing pelvis were required such that the contours be continuous with the thigh skin in the sitting position keeping a smooth contour in the standing position. One example is the flattening of the bottom of the buttocks to create a human like sitting contour. Also, the hip center of rotation was moved anteriorly to create a reasonable combination of sitting and standing contours while meeting the tri-service dimensions of the pelvis and upper leg. The inserts required for the pelvis were extensive as the movement of the upper leg with respect to the pelvis was required to be unconstrained.

The data used for the large torso were not the data originally given in the stereophotometric data set as the original data represented the physique of an overweight man, not an aviator. Although the tri-service dimensions were met on this segment, the general physique was not appropriate to an aviator. To compensate for this, the finished torso skin contour from the small manikin was expanded in the manner previously described to achieve the skin contours for the large torso.

The small upper torso segment required a minor modification to allow for the length of the manikin. The length of the small spine and the stack up of the shoulder block and pelvis elements, forced the torso length (i.e., the distance between the shoulder and hip centers of rotation) to expand from the stickman dimensions by 1.1 inches. The segment was adjusted to assume this extra length by expanding the thicknesses between the slices above the chest dimensions as the chest height was a requirement of the system and was not changed.

Although these segments required several alterations from the human data, they are still representative of human skins. They meet the required dimensions of the tri-service data and are representative of the respective size male aviator as the contours follow the general shape of the stereophotometric data.

2.2.1.2.1. Existing Segment Use

Several of ADAM's skin segments were not designed based on the stereophotometric data but were obtained from other manikins. Through an analysis of existing segments, some were selected for use in ADAM as their dimensions were close to the tri-service requirements. The segments used in ADAM from other manikins include the head, hands, feet, and the small abdomen.

The Hybrid II head was used for both the small and large ADAM. The outside dimensions of this head were between the small and large tri-service requirements and most were not significantly different from either size. Since the head will be covered with a helmet in ejection seat testing, those dimensions of the head which were outside the specification will not affect the aerodynamic loadings on the head. The mass properties are more significant and were corrected by ballasting and utilizing foam. This is further discussed in the section of this report on mass properties.

The hands currently used on the VIP 95 manikin are sized for a midsize manikin. These were selected for use on the small and large ADAMs as the dimensions fell between the requirements for both sizes.

The small and large ADAM feet were molded from the VIP 95 foot mold. Since this part is representative of a 95 percentile human, its dimensions were compatible with the large ADAM and it was used directly in this manikin. For adaptation to the small manikin, it was shortened by approximately 1 inch. As in the head, the foot outside dimensions will not affect the aerodynamic loadings on the feet as they will be covered by nonflexible flight boots; therefore, the dimensions are not critical to the overall reactions of the manikin.

The abdomen segment is unique in both manikins in that the outside contours of the large and the small were not designed from the stereophotometric data. The purpose of these segments was to fill the gap between the upper torso and pelvis skins, which was about 2 inches. They also served as a protection to the instrumentation in this area. Since there were no dimensional requirements in this 2-inch section, the shapes from the upper torso and pelvis were extended to fill the gap. The abdomen from the small manikin was made using the Hybrid II 50 percentile mold. This was used because the mold could be modified to fit the small manikin and the outside contours closely followed the shapes found in the upper torso and pelvis. The large abdomen was designed for ADAM to match the contours used for the upper torso and pelvis.

2.2.1.3. Results

After the cross-sectional slice shapes were developed, they were used to create segment three dimensional models.

The models were used in a standard casting process to create aluminum molds for skin manufacturing. A photograph of the small lower leg skin is shown in Figure 14.

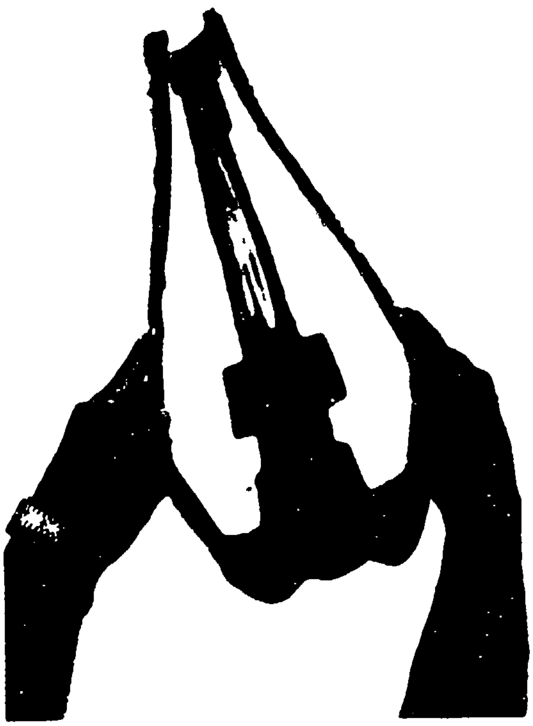


Figure 14. Small Lower Leg Skin

After molding the skins, assembling the segments, and installing the wiring, most of the dimensional requirements were met. The areas which did not meet the dimensional requirements were primarily due to the use of existing parts. For example, the wrist dimensions were increased to create a continuous contour with the VIP 95 hand. Other factors which created out of tolerance dimensions were the routing of wires expanding the skins or the allowance of the mechanical assemblies within the skin contours during the design process. All areas which were out of tolerance, with the exception of those due to the use of existing parts, were within 9 percent of the specification tolerance. The results of the small and large skin dimensions as compared to the tri-service data are presented in Tables 2 and 3, respectively.

## TABLE 2. ANTHROPOMETRIC MEASUREMENTS (SMALL ADAM)

| Item | Small Specification ( inches) | Small ADAM (inches) | Percent Difference |
|---|---|---|---|
| Stature** | 66.2 | 66.25 | 0.1 |
| Mastoid Ht. | 59.8 | 60.13 | 0.6 |
| Cervicale Ht. | 56.5 | 56.00 | 0.9 |
| Acromiale Ht. | 53.8 | 56.13 | 4.3 |
| Bottom Rib Ht. | 41.4 | N/A | -- |
| Iliocristale Ht. | 39.4 | 38.25 | 2.9 |
| Trochanterion Ht. | 34.8 | 34.75 | 0.1 |
| Gluteal Furrow Ht. | 29.9 | 32.00 | ***7.0*** |
| Tibiale Ht. | 17.7 | 18.00 | 1.7 |
| Sphyrion Ht. | 2.6 | 2.50 | 3.8 |
| Head Circ. | 22.0 | 23.13 | ***5.1*** |
| Head Breadth | 6.1 | 6.00 | 1.6 |
| Head Length | 7.7 | 8.00 | 3.9 |
| Neck Breadth | 4.6 | 3.63 | ***21.1*** |
| Neck Circ. | 14.4 | 14.63 | 1.6 |
| Chest Breadth | 12.0 | 12.00 | 0 |
| Chest Depth | 8.9 | 9.00 | 1.1 |
| Chest Circ. | 35.9 | 37.00 | 3.1 |
| Bideltoid Breadth | 17.8 | 17.38 | 2.4 |
| Bottom Rib Breadth | 10.9 | N/A | -- |
| Waist Breadth | 11.1 | 12.0 | ***8.1*** |
| Waist Depth | 8.0 | 8.00 | 0 |
| Waist Circ. | 31.1 | 33.13 | ***6.5*** |
| Bicristale Breadth | 10.2 | 10.75 | ***5.4*** |
| Bitrochanterion Breadth | 13.0 | 12.75 | 1.9 |
| Buttock Depth | 8.5 | 8.75 | 2.9 |
| Buttock Circ. | 35.9 | N/A | -- |
| Upper Thigh Circ. | 21.1 | 21.56 | 2.2 |
| Mid Thigh Depth | 6.5 | 6.47 | 0.5 |
| Knee Circ. | 14.1 | N/A | -- |
| Knee Breadth | 3.7 | 4.20 | ***13.5*** |

TABLE 2. ANTHROPOMETRIC MEASUREMENTS (SMALL ADAM) (continued)

| Item | Small Specification (inches) | Small ADAM (inches) | Percent Difference |
|---|---|---|---|
| Calf Circ. | 13.7 | 13.88 | 1.3 |
| Calf Depth | 4.5 | 4.30 | 4.4 |
| Bimalleolar Breadth | 2.8 | 2.82 | 0.7 |
| Ankle Circ | 8.3 | 8.44 | 1.7 |
| Foot Breadth | 3.7 | 4.17 | ***12.7*** |
| Foot Length | 10.1 | 9.88 | 2.2 |
| Acromio-Radiale L | 12.2 | 12.50 | 2.5 |
| Biceps Depth | 3.9 | 3.80 | 2.6 |
| Elbow Circ. | 10.2 | N/A | -- |
| Elbow Breadth | 2.7 | N/A | -- |
| Mid Forearm Circ. | 8.8 | 8.88 | 0.9 |
| Radiale-Stylion L | 10.0 | 10.30 | 3.0 |
| Mid Forearm Breadth | 3.0 | 3.05 | 1.7 |
| Wrist Breadth | 2.0 | 2.35 | ***17.5*** |
| Wrist Circ. | 6.6 | 6.88 | 4.2 |
| Hand Length | 7.2 | 7.25 | 0.7 |
| Hand Breadth | 3.3 | 3.38 | 2.4 |
| Hand Depth | 1.1 | 1.40 | ***27.3*** |
| Hand Circ. | 8.1 | 8.75 | 8.0 |
| Sitting Height** | 35.2 | 34.50 | ***2.0*** |
| Eye Height, Sitting** | 30.5 | 30.00 | ***1.6*** |
| Buttock-Knee L** | 22.3 | 22.19 | 0.5 |
| Buttock-Popliteal L** | 18.5 | 17.88 | ***3.4*** |
| Thigh Clearance | 5.9 | 5.50 | ***6.8*** |
| Knee Ht., Sitting** | 20.6 | 21.19 | ***2.9*** |
| Popliteal Ht** | 16.2 | 16.38 | ***1.1*** |
| Biceps Circ. Relaxed | 11.2 | 11.00 | 1.8 |

***Bold and Italics*** = outside specification tolerance.

Specification tolerance: ±5 percent (except **: ±1 percent).

## TABLE 3. ANTHROPOMETRIC MEASUREMENTS (LARGE ADAM)

| Item | Large Specification (inches) | Large ADAM (inches) | Percent Difference |
|---|---|---|---|
| Stature** | 74.3 | 74.26 | 0.1 |
| Mastoid Ht. | 67.5 | 66.64 | 1.3 |
| Cervicale Ht. | 64.0 | 64.76 | 1.2 |
| Acromiale Ht. | 61.3 | 63.01 | 2.8 |
| Bottom Rib Ht. | 47.2 | N/A | -- |
| Iliocristale Ht. | 45.1 | 44.01 | 2.4 |
| Trochanterion Ht. | 39.7 | 40.01 | 0.8 |
| Gluteal Furrow Ht. | 34.4 | 36.01 | 4.7 |
| Tibiale Ht. | 19.9 | 20.26 | 1.8 |
| Sphyrion Ht. | 3.0 | 3.00 | 0 |
| Head Circ. | 23.1 | 23.12 | 0.1 |
| Head Breadth | 6.3 | 6.00 | 4.8 |
| Head Length | 8.0 | 8.00 | 0 |
| Neck Breadth | 5.2 | 3.63 | ***30.2*** |
| Neck Circ. | 16.0 | 14.63 | ***8.6*** |
| Chest Breadth | 14.0 | 14.00 | 0 |
| Chest Depth | 10.6 | 10.40 | 1.9 |
| Chest Circ. | 42.3 | 43.00 | 1.7 |
| Bideltoid Breadth | 20.4 | 20.80 | 2.0 |
| Bottom Rib Breadth | 13.1 | N/A | -- |
| Waist Breadth | 13.6 | 13.25 | ***6.3*** |
| Waist Depth | 9.8 | 9.76 | 0.4 |
| Waist Circ. | 37.9 | 38.00 | 0.3 |
| Bicristale Breadth | 12.0 | 12.25 | 2.1 |
| Bitrochanterion Breadth | 15.0 | 16.00 | ***6.7*** |
| Buttock Depth | 10.6 | 10.38 | 2.1 |
| Buttock Circ. | 42.4 | N/A | -- |
| Upper Thigh Circ. | 25.7 | 25.25 | 1.8 |
| Mid Thigh Depth | 7.4 | 7.00 | ***5.4*** |
| Knee Circ. | 16.6 | N/A | -- |
| Knee Breadth | 4.2 | 4.24 | 1.0 |
| Calf Circ. | 15.9 | 15.88 | 0.1 |

TABLE 3. ANTHROPOMETRIC MEASUREMENTS (LARGE ADAM) (continued)

| Item | Large Specification (inches) | Large Adam (inches) | Percent Difference |
|---|---|---|---|
| Calf Depth | 5.1 | 5.12 | 0.4 |
| Bimalleolar Breadth | 3.1 | 2.86 | ***7.7*** |
| Ankle Circ. | 9.5 | 9.50 | 0 |
| Foot Breadth | 4.1 | 4.17 | 1.7 |
| Foot Length | 11.3 | 10.87 | 3.8 |
| Acromio-Radiale L | 13.9 | 13.50 | 2.9 |
| Biceps Depth | 4.9 | 4.76 | 2.9 |
| Elbow Circ. | 11.7 | N/A | -- |
| Elbow Breadth | 3.0 | N/A | -- |
| Mid Forearm Circ. | 9.8 | 9.75 | 0.5 |
| Radiale-Stylion L | 11.3 | 11.91 | ***5.4*** |
| Mid Forearm Breadth | 3.4 | 3.32 | 2.4 |
| Wrist Breadth | 2.3 | 2.33 | 1.3 |
| Wrist Circ. | 7.4 | 7.38 | 0.3 |
| Hand Length | 7.9 | 7.25 | ***8.2*** |
| Hand Breadth | 3.7 | 3.38 | ***8.6*** |
| Hand Depth | 1.1 | 1.40 | ***27.3*** |
| Hand Circ. | 8.9 | 8.75 | 1.7 |
| Sitting Height** | 38.6 | 37.50 | 2.8 |
| Eye Height, Sitting** | 33.5 | 33.00 | 1.5 |
| Buttock-Knee L** | 25.6 | 25.75 | 0.5 |
| Buttock-Popliteal L** | 21.4 | 20.25 | 5.4 |
| Thigh Clearance | 7.2 | 7.25 | 0.7 |
| Knee Ht., Sitting** | 23.6 | 23.75 | 0.6 |
| Popliteal Ht** | 18.4 | 18.38 | 0.1 |
| Biceps Circ. Relaxed | 13.3 | 13.00 | 2.3 |

***Bold and Italics*** = outside specification tolerance.

Specification tolerance: ±5 percent (except **: ±1 percent).

The skin contour data of the large and small ADAMs are representative of a large and small male military aviator and are mathematically linked to actual human data. This is unique to the ADAMs in that other manikins use artist representations of the human form to define the skin shapes.

Knowing that these skin contours are mathematically linked to human data, they are preferable to use for manikins as opposed to other skin sets. Another reason for using these data is that they are designed such that they can be used for the development of other manikins with minor modifications. Simply following the procedure described above, this data set can be applied to any size manikin.

Since these data are universal to all manikins, they are likely to be used in the future for the development of other manikin skin contours.

### 2.2.2. Mass Properties Analysis

The requirement to duplicate humanlike responses in the ADAM imposes specific requirements on the mass, center of gravity, and moments of inertia for each segment of both the small and large manikins. The human data used to define the segment mass properties have been specified with respect to a standard set of axis systems, designated as anatomical axis systems. The specific anatomical axis systems defined by at least three points on the surface of the skin and used for this data base were originally developed by McConville et al. (1980).

Data in the anatomical axis systems could not be directly applied to the manikin design because the surface anatomical landmarks could not be directly related to the mechanical substructure necessary for fabrication. Therefore, before using the data in the ADAM design, a transformation procedure was developed to obtain data in a form that could be directly applied to the design process. This transformation procedure first required the definition of a new axis system for each segment based on the mechanical elements of the segment. The following paragraphs document the definition of these axis systems, the procedure developed for transforming the data to the mechanical axis systems, and the development of the transformed data base used in the mechanical design. The procedure and results from the mass property analysis of the design will also be presented. A brief summary of the data base used for the description of the ADAM system and an overview of the axis systems used will be presented for background information.

2.2.2.1. Data Base Description

The specifications for ADAM, published in the ADAM Statement of Work, USAF Contract F33615-85-C-0535, Advanced Dynamic Anthropometric Manikin (ADAM), Systems Research Laboratories, Inc., 11 September 1985, are based on several sets of data. The ADAM joint centers of rotation, segment centers of mass, and anthropometry were developed along with the tri-service data set, "Anthropometry and Mass Distribution for Human Analogues, Volume I: Military Male Aviators," March 1988, AAMRL-TR-88-010, from a common data base using similar methods. Although most of the ADAM data set reflects the tri-service data, the ADAM specifications include a more extensive definition of the joint centers and centers of mass. The tri-service data required only a 1-dimensional definition of the joint centers and centers of mass, while the ADAM specification required the data to be defined in 3 dimensions. The tri-service data set, which describes a small, midsize, and large standard military aviator, was developed using a growth factor based on the 1980-1990 stature approximations, and applied directly to the dimensions from the U.S. Air Force 1967 survey of 2,420 male flying personnel (Churchill et al., 1978).

The results from the stereophotometric survey conducted in 1969 by the Texas Institute for Rehabilitation and Research of 31 male subjects (McConville et al., 1980) were used to define the inertial tensors, weights, and skin contours of the ADAM body segments. The inertial properties were derived from the volume distribution data of the survey data base. Due to the complexity involved with building a statistical data base describing shapes, specific segments were selected that best met the tri-service dimensional requirements, thus creating a collection of segments from several subjects to describe a single total body ADAM skin envelope.

Four axis systems were used in the transformations of the original ADAM data sets. Three of these were segment based and the other was a total body or global system with fixed relative segment positions. The particular global system used in this analysis was defined in the stereophotometric data base and the surface data were initially compiled in this axis system. These data, which consisted of body surface point coordinates and the surface landmark coordinates, were combined with segmentation planes as defined by McConville et al. (1980) to create individual segment surface data sets. Body segment orientations for the 31 subjects measured in the data base were averaged to arrive at a composite body position which has been used for relative adjacent segment position and motion analyses. The reconstructed body position for ADAM, in the global coordinate system with lower arm landmarks illustrated, is shown in Figure 15.

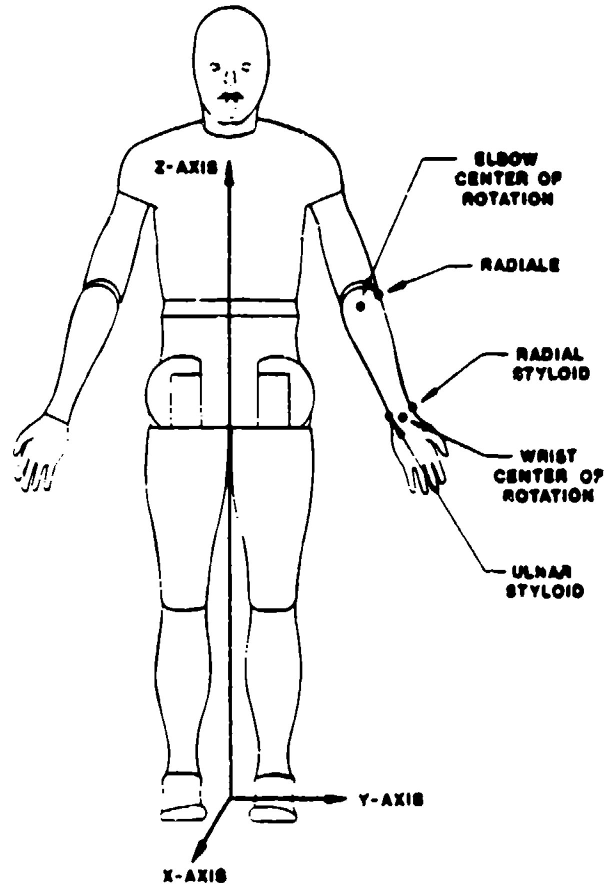



Figure 15. Total Body Global Axis System

The anatomical, principal, and mechanical axis systems were based on individual segment properties. As described earlier, the anatomical axis systems were based on defined anatomical landmarks on the skin surface. Both the tri-service and stereophotometric data bases were defined with respect to the anatomical axis systems. The principal axis systems were derived from the segment mass distribution properties. They were specified with respect to the segment center of mass and were offset from the anatomical axes by a y-axis rotation. The mechanical axis systems were based on the mechanical substructures within each segment and were developed for use in the design of the manikin. Figures 16, 17, and 18 indicate the anatomical, principal, and mechanical axis systems associated with the left forearm. These figures illustrate the use of various axis systems on a specific segment.

2.2.2.2. Mechanical Axes Definition

The transformation of segment data from an anatomical to a mechanical axis system required the calculation of the displacement matrix which relates the two axis systems by a combination of rotational and translational displacements. The displacement matrices for the ADAM segments were calculated from the mathematical definition of the mechanical axes with respect to the

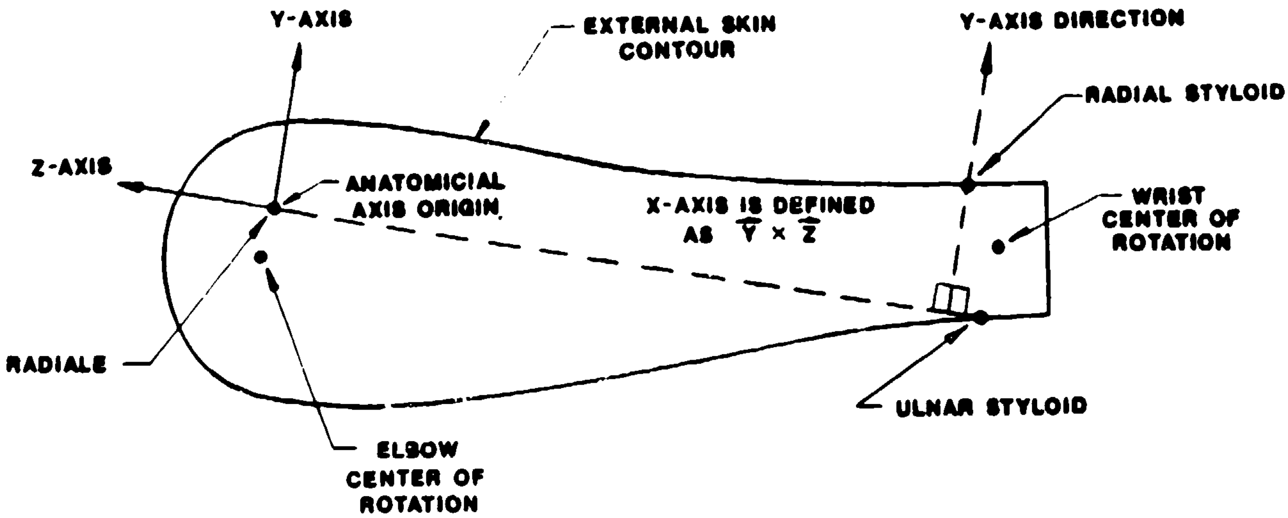



Figure 16. Forearm Anatomical Axis System

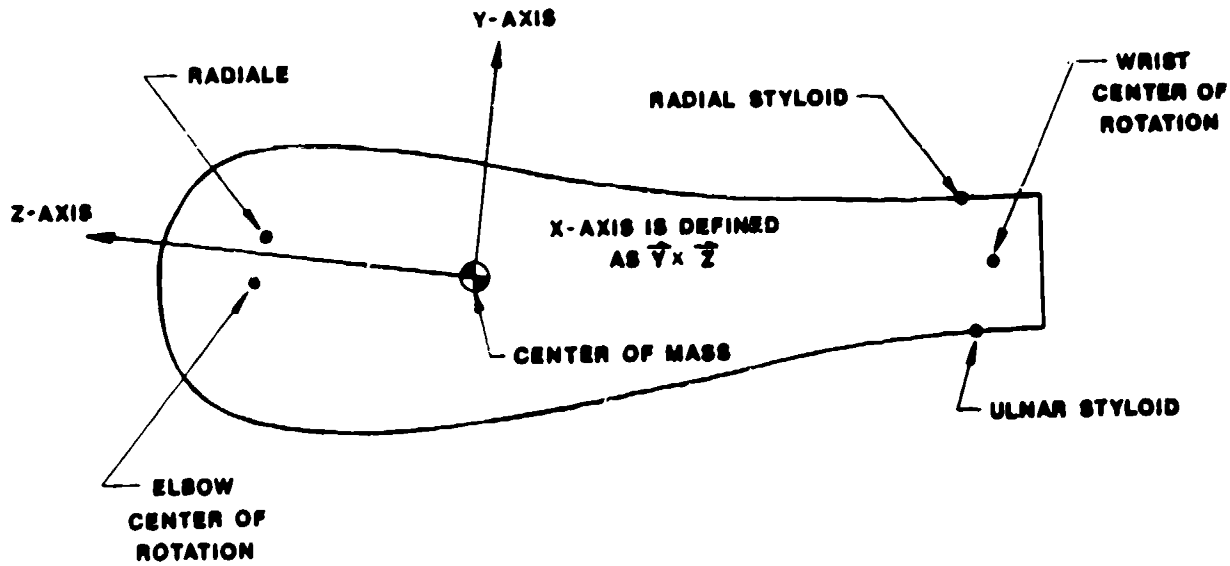



Figure 17. Forearm Principal Axis System

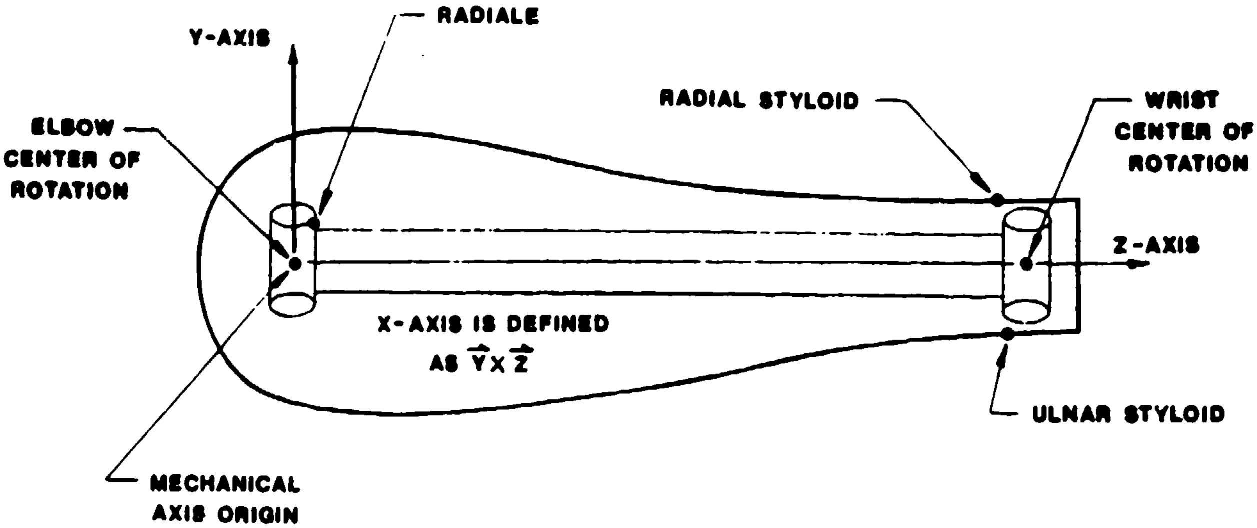



Figure 18. Forearm Mechanical Axis System

anatomical axis systems. The definition procedure was dependent on the type of body segment under consideration.

For axis definition, the body segments were separated into three general groups based on segment geometry and type of connective joints. The three groups were the torso segments, limb segments, and extremities. The mechanical axis systems were defined as described below.

2.2.2.2.1. Torso Segments

The torso inertial properties were specified with respect to three segments connected by rotational centers: the thorax, abdomen, and pelvis. This implied that two discrete articulations were associated with torso deformation. The ADAM spine design included only one articulation point in the torso which would create two torso segments connected by one rotational center. Since the rotational point in ADAM was located within the bounds of the abdomen segment and not at a rotational center, a direct application of the manikin to human torso segment data was not valid. To allow a direct comparison, the torso was resegmented to form the upper and lower torso segments, the plane of separation being normal to the plane of symmetry (midsagittal plane) and located at the manikin rotational center (lumbar pivot). The data for the upper torso were then defined with respect to the pelvis anatomical axis system.

The mechanical axis systems for both the upper and lower torso, as shown in Figures 19 and 20, were defined with respect to the physical characteristics of the manikin. The y-axis of each was

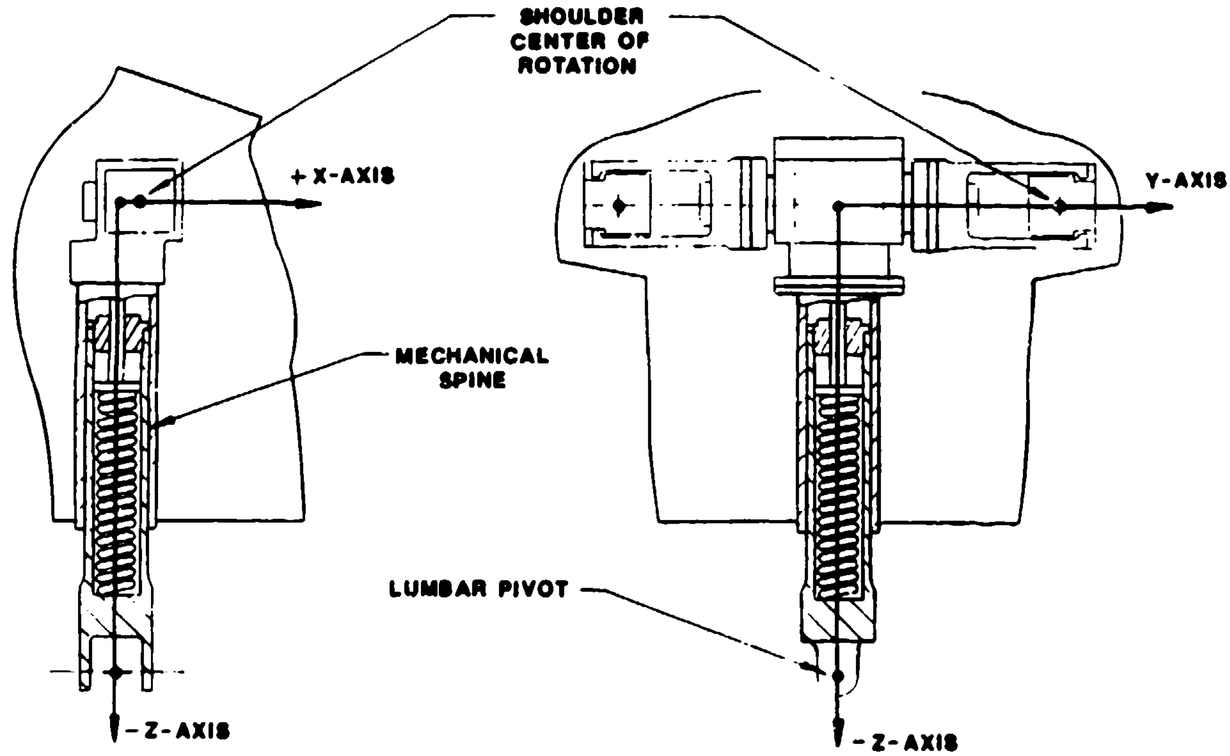



Figure 19. Upper Torso Mechanical Axis System

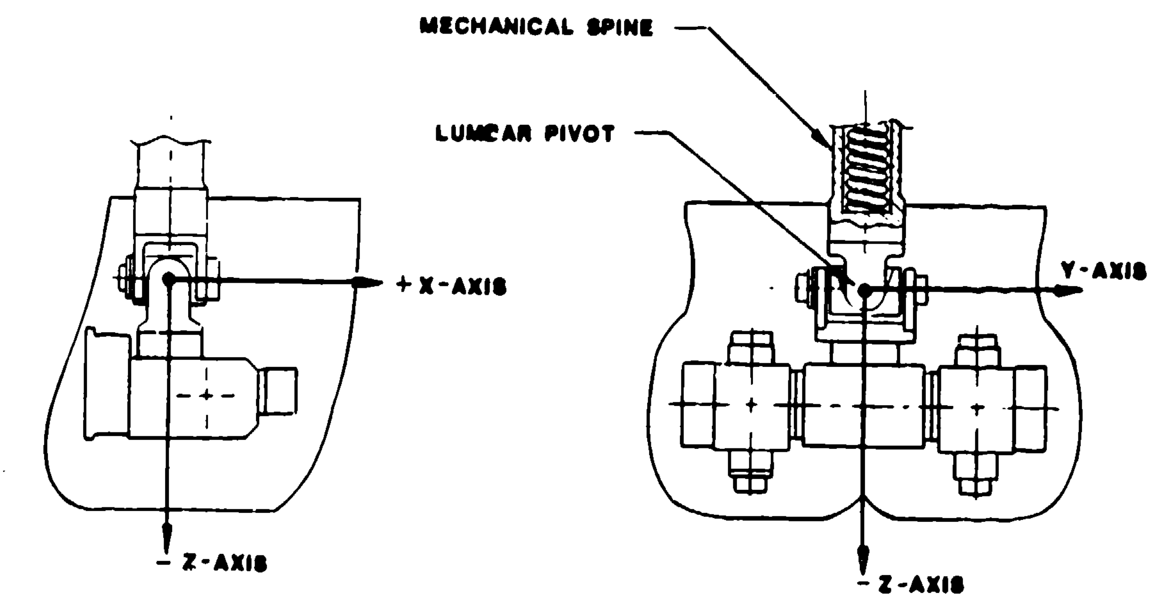



Figure 20. Lower Torso Mechanical Axis System

normal to the midsagittal plane and was positive to the left. The z-axis of the upper torso was defined by the vector from the intersection of the mechanical spine center line with the global x-y plane which passes through the shoulder centers of rotation to the lumbar pivot point. The z-axis of the lower torso was defined by the vector which originates at the lumbar pivot point and was coincident with the center line of the mechanical spine when the manikin is in the sitting position (the lumbar pivot in the 0-degree position). The origin of the upper torso mechanical axis system was located along the spine center line at the height of the shoulder centers of rotation. The lower torso mechanical system origin was located at the lumbar pivot center of rotation.

#### 2.2.2.2.2. Limb Segments

The second group of segments was that for limbs. This group included the upper and lower arms and the upper and lower legs. The mechanical axis origins of these segments were defined at the proximal joint center for each segment, with the z-axis extending from the origin to the distal joint center. The joint center locations for each segment were defined with respect to the corresponding anatomical axis systems in the human data base. The y-axes for the forearm and the lower leg were chosen to be aligned with the pin rotational axis of the elbow and knee, respectively. Since the hip and shoulder joints had more than one degree of freedom, either axis of rotation could have been used to define the second mechanical axis for the upper leg and arm, assuming the two axes are normal to each other. The axis which allows abduction/adduction for each joint was chosen to define the mechanical x-axis direction based on the procedure outlined below.

A cross product method was used to define the orientations of the joint rotational axes. The method was based on the fact that the two bones of a pinned joint move such that the center lines of the bones and the pinned joint center lie in the same plane throughout the range of motion. The axis of rotation of the pinned joint was normal to this plane and was taken as the vector cross product of the mechanical z-axes of the adjacent limb segments in the global system with the body segments in the average composite orientations described previously. The orientations in the anatomical axis systems were calculated using the program ROTRANS (see Appendix A).

#### 2.2.2.2.3. Extremities

The third group of segments consisted of the extremities. Included in this group were the head, neck, hands, and feet. Several mechanical elements and all of the skin contours except the feet of these segments used in the ADAM are standard parts from existing manikins. These segments generally met the dimensional properties; however, some changes were required to meet the inertial properties required in the ADAM. To make these changes, mechanical axis systems were required to relate the actual data to the human data.

The skin drawings of the existing parts with the ADAM mechanical elements superimposed on them were used to relate the anatomical and mechanical axes. By locating the landmarks on the drawings, the anatomical axis systems associated with the segments were identified.

The mechanical axis system definitions for these segments were defined based on the joint centers of rotation and the mechanical elements within the segments. The axis systems (Figures 21 through 24) will be briefly described. The head mechanical axis system origin, as shown in Figure 21, was located at the pin which joins the head and the neck. The X-axis was the vector from the origin in the posterior direction parallel to the bottom plate of the head in the midsagittal plane. The head Z-axis was the vector from the origin in the midsagittal plane normal to the X-axis. The neck axis origin, as shown in Figure 22, was also at the pin which attaches the head and neck segments. The Z-axis was chosen as the vector from the origin to the center of the bottom bolt. The X-axis was the vector from the origin in the midsagittal plane normal to the Z-axis. The hand mechanical axis system origin was located at the wrist center of rotation, as shown in Figure 23. The X-axis was chosen to be aligned with the wrist rotational axis for inversion/eversion. The Z-axis was defined as the vector from the origin to a point located at the center of the hand bone.

The mechanical and anatomical Y-axes for these segments were taken to be parallel and the translations and rotations between the two axis systems were explicitly defined on the drawings.

The foot mechanical axis system, as shown in Figure 24, was defined by the mechanical elements within the segment and the known position of the foot in the global axis system. The origin was located at the ankle center of rotation and the Z-axis was defined as the vector extending from the origin normal to the global X-Y plane. Because of the orientation of the foot in the stereophotometric data base which extended directly forward, the axis of rotation of the ankle was assumed to be parallel to the global Y-axis. This axis was then taken to be the Y mechanical axis.

2.2.2.2.4. Data Transformation

The methods described for defining the mechanical axis origins and the Z and Y axis directions were applied to each manikin segment. The x axis directions were calculated using the cross product of the Y and Z axes. A summary of the mechanical axis definitions is presented in Table 4.

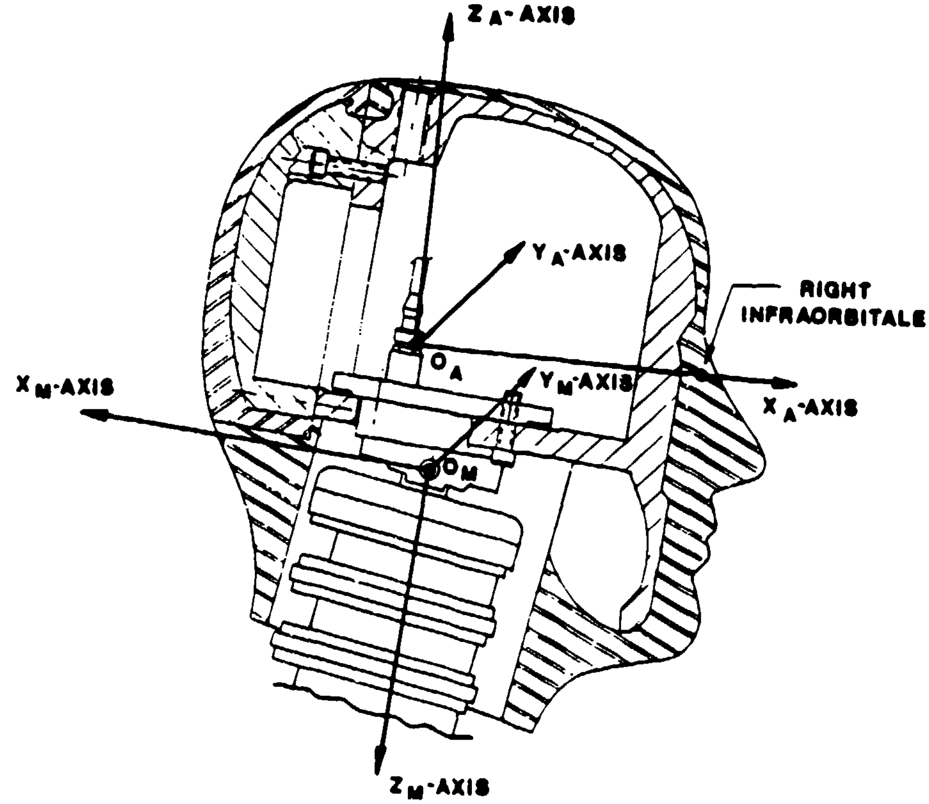


Figure 21. Head Mechanical Axis System Definition

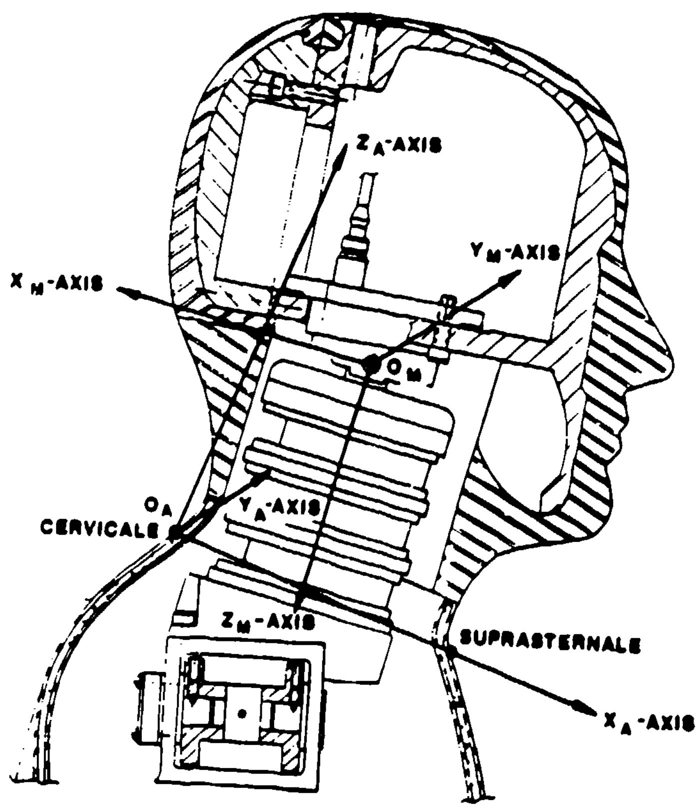


Figure 22. Neck Mechanical Axis System Definition

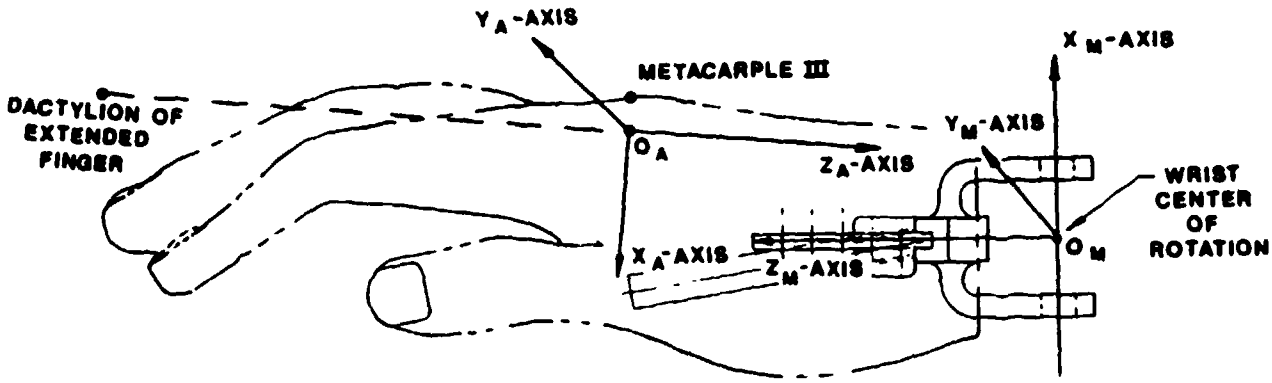



Figure 23. Hand Anatomical and Mechanical Axis System

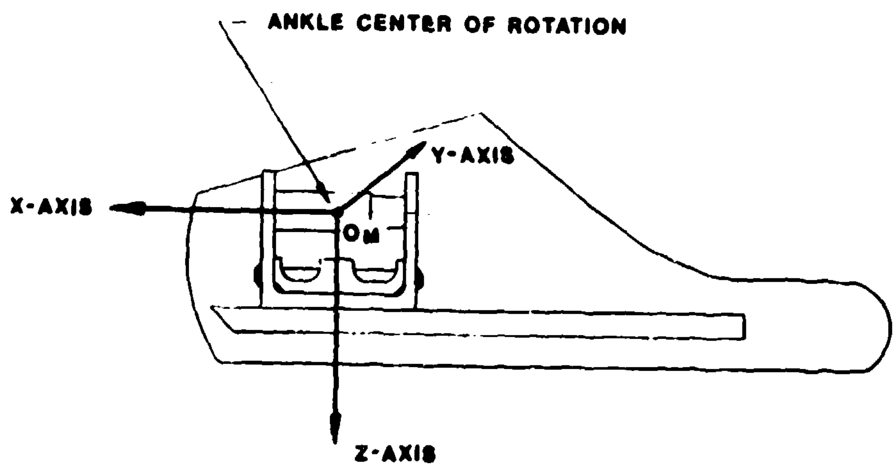



Figure 24. Foot Mechanical Axis System

TABLE 4. SUMMARY OF MECHANICAL AXIS DEFINITION

| Segment | Mechanical Z-Axis is the Vector From | To | Y-Axis Definition | X-Axis Definition |
|---|---|---|---|---|
| Head | Head/Neck Pin | Top of Head; Perpendicular to Bottom Plate | Midsagittal Symmetry | Cross Product of Y and Z |
| Neck | Head/Neck Pin | Bottom Center of Neck | Midsagittal Symmetry | Cross Product of Y and Z |
| Upper Torso | Point at Shoulder Along Spine Center Line | Lumbar Pivot Point | Midsagittal Symmetry | Cross Product of Y and Z |
| Lower Torso | Lumbar Pivot Point | Along Spine Center Line | Midsagittal Symmetry | Cross Product of Y and Z |
| Upper Arm | Shoulder | Elbow | Cross Product of Z and X | Shoulder Abduction/ Adduction Axis of Rotation |
| Forearm | Elbow | Wrist | Elbow Axis of Rotation | Cross Product of Y and Z |
| Hand | Wrist | Center Point of Distal End of Hand Bone | Wrist Extension Axis of Rotation | Flexion/ Cross Product of Y and Z |
| Thigh | Hip | Knee | Cross Product of Z and X | Hip Abduction/ Adduction Axis of Rotation |
| Calf | Knee | Ankle | Knee Axis of Rotation | Cross Product of Y and Z |
| Foot | Ankle | Bottom of Foot Bone | Ankle Flexion/ Extension Axis of Rotation | Cross Product of Y and Z |

The three points used to define the mechanical system origin and axes orientations were located with respect to the anatomical axis system by the methods described. These points were used in the computer procedure TOTAL2 (see Appendix B) which calculated the transformation matrix from the anatomical to the mechanical axis system, (D)ma. This procedure involved the establishment of the unit vectors along each axis and the formation of the rotation matrix. The translation vector was then found and combined with the rotation matrix to generate a 4 x 4 transformation matrix.

After the displacement matrices were calculated, the data were transformed from the anatomical to the mechanical axes by the operation:

$$\vec{P}_m = (D)_{ma} \vec{P}_a$$

where

$\vec{P}_m$ = vector in the mechanical axis system

$(D)_{ma}$ = the displacement matrix (4 x 4) from the anatomical to the mechanical axes

$\vec{P}_a$ = vector in the anatomical axis system

The data that were directly transformed consisted of the center of gravity locations and the joint centers of rotation. The inertial and surface shape data transformations were more involved and will be discussed in the following sections.

2.2.2.3. Inertial Transformation

The initial inertial property data provided for design consisted of principal axes and moments of inertia specified with respect to the anatomical axis system and the center of mass. In these data, the principal axes were displaced from the anatomical axes only by a rotation about the Y anatomical axis through an angle θ. The transformation operator from the principal to the anatomical axes is:

$$(A)_{ap} = \begin{vmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{vmatrix}$$

The principal moments were transformed into the anatomical axis system by:

$$(I)_a = (A)_{ap}\,(I)_p\,(A)_{ap}^T$$

where

$(I)_a$ = inertia tensor about the anatomical axes
$(I)_b$ = principal moment of inertia tensor
$(A)_{ap}^T$ = transpose of $(A)_{ap}$ matrix

The inertia tensor was then transformed into mechanical axis system alignment with the origin still at the segment center of mass, by

$$(I)_m = (A)_{ma}\,(I)_a\,(A)_{ma}^T$$

where

$(I)_m$ = inertia tensor about the mechanical axes
$(A)_{ma}$ = transformation operator from the anatomical to the mechanical axes

When transforming a diagonal matrix (the principal inertia tensor), off-diagonal terms (products of inertia) are developed. The typical product of inertia formed in the mechanical axis system was less than 10 percent of the $I_{xx}$ and $I_{yy}$ moments of inertia. A sensitivity analysis showed that neglecting these products of inertia in the mechanical axis system affects the principal moments of inertia (after a backwards transformation) about the x and y axes of all segments and the Z axes of the torso segments by less than 4 percent. However, the z axes of all but the torso segments are affected by more than 10 percent. Based on the assumption that only the $I_{zz}$ of the torso segments have a significant effect on system response, all products of inertia have been neglected and, therefore, the accuracy of the $I_{zz}$ of all limb and extremity segments is 10 percent, at best.

2.2.2.4. Analysis Technique

With the human data referenced to the segment mechanical axis systems, they could be easily compared to the mass property calculations made in conjunction with the design drawings. The following section will cover the approach to the calculations, the evaluation criteria used in the analysis of the preliminary results, and a presentation of the final results.

There were two interactive parts to the analysis procedure: the actual calculations and the design process. The calculations defined the mass properties and the design process evaluated the results in combination with the system characteristics.

2.2.2.4.1. Calculation of Mass Properties

The calculation procedure of each segment began with a separation of the segment into skin and bone subsegments. Each subsegment was then broken into elements for which the mass properties could be easily calculated. The skin subsegments were separated along the Z-axis into cylinders and then each cylinder was further separated into two elements consisting of the skin and foam. As an example of the bone subsegment elemental separation, the forearm is shown in Figure 25. Depending on the complexity of the segment, bones are divided into approximately 20 elements and the skins into approximately 10 elements.

For ease of incorporating changes and the benefit of time and accuracy, the bulk of the calculations was accomplished through the use of a computer. A program was written to calculate the mass properties of geometric elements and to combine the mass properties of the elements. The program listing can be found in Appendix C (see MASSPR).

The initial calculation for each element was to determine the weight. With the densities known, the elemental volumes were calculated. Since most elements were either parallelpipeds, cylinders, or combinations of these, the calculations were relatively straightforward. Those elements which were irregular were sectioned into geometric parts such that the volumes could be calculated.

The center of gravity calculations were more involved than those to determine the segment weights. If the element was a simple geometric shape, the center of gravity of the element was calculated directly from the drawing. The irregular elements which were sectioned required a weighted summation for the determination of the element center of gravity. The element center of gravities were then combined to find the center of gravity of the subsegment in the X, Y, and Z mechanical axis directions. This procedure was repeated to combine the subsegments in calculating the segment center of gravity.

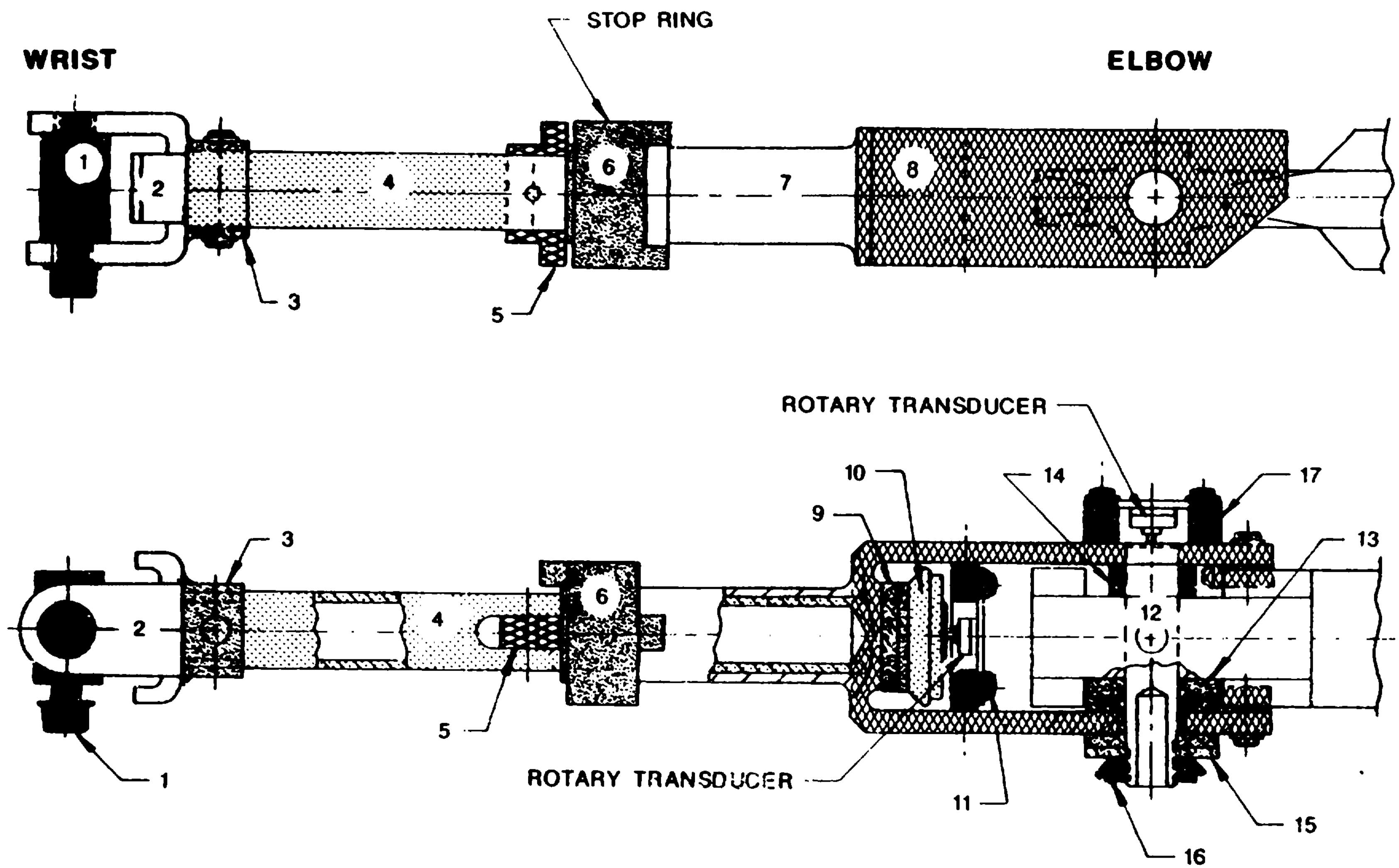



Figure 25. Small ADAM Forearm

With the center of gravities and weights of the parts known, the moments of inertia of these parts could be calculated. Parts with weights of less than .03 pound were considered negligible regarding moments of inertia. The moments of inertia of all other elements were calculated about axes aligned with the segment mechanical axes and centered at the elemental centers of gravity using simple equations. The parallel axis theorem was then used to transfer the moments of inertia to the segment center of gravity and a simple summation produced the segment moments of inertia.

The weights, centers of gravity, and moments of inertia for all small and large ADAM segments with respect to the segment mechanical axes compared to the transformed ADAM specifications are given in Tables 5 and 6.

#### 2.2.2.4.2. Design Process

The calculated data were compared to the transformed specification data. If the data did not fall within the tolerance of the specifications, the segment design was reanalyzed. Still meeting the strength requirements, the designs were modified and the mass properties were recalculated. This process continued until significant changes in the design were no longer possible based on machine-ability, strength, stability, and cost requirements. Although some segments did not meet the mass property requirements, higher priority requirements such as the strength were met. The mass properties of the system were as close as possible to the specification while still meeting the other requirements of the system.

#### 2.2.2.5. Final Transformation

The data in the mechanical axis systems could not be directly compared to the original specification data; therefore, the calculated data were transformed back into the anatomical axis systems (for a program listing, see BACK5 in Appendix D). Tables 7 and 8 show the small and large ADAM calculated data compared to the specification data with respect to the anatomical (center of gravity data) and principal (moments of inertia data) axis systems. Notice that the ADAM data contains off-diagonal terms in the "principal" inertia tensors. The reason is that the off-diagonal terms in the mechanical axes were assumed negligible and were not calculated. When the diagonal tensor was transformed, off-diagonal terms appeared. These off-diagonal terms were neglected when comparing the two sets of data but they are presented here simply because they are part of the tensors and affect the diagonal terms.

TABLE 5. SMALL ADAM MASS PROPERTY COMPARISON (MECHANICAL AXIS SYSTEMS)

| Segment | | Specification Data | | | Analytical Data | | |
|---|---|---|---|---|---|---|---|
| Head* | Weight | 9.20 | | | 9.20 | | |
| Neck* | Weight | 2.00 | | | 2.34 | | |
| Upper Torso | Weight | 45.96 | | | 48.22 | | |
| | CG | (-1.27,0,6.10) | | | (-.96,0,5.50) | | |
| | Inertia | 1514.157 | 0 | -152.041 | 1460.670 | 0 | 0 |
| | Tensor | 0 | 1226.386 | 0 | 0 | 1163. 010 | 0 |
| | | -152.041 | 0 | 801.364 | 0 | 0 | 632.360 |
| Lower Torso | Weight | 17.3 | | | | 17.94 | |
| | CG | (-0.01,0,2.39) | | | | (-.24,0,2.88) | |
| | Inertia | 188.740 | 0 | -16.180 | 271.810 | 0 | 0 |
| | Tensor | 0 | 168.380 | 0 | 0 | 104.120 | 0 |
| | | -16.180 | 0 | 217.014 | 0 | 0 | 289.540 |
| Rt Upper Arm | Weight | 3.40 | | | | 3.43 | |
| | CG | (-.17,0.20,5.40) | | | | (-.01,0,5.18) | |
| | Inertia | 28.870 | 0.748 | 2.064 | 35.310 | 0 | 0 |
| | Tensor | 0.748 | 29.116 | -1.958 | 0 | 36.101 | 0 |
| | | 2.064 | -1.958 | 6.157 | 0 | 0 | 3.680 |
| Rt Forearm | Weight | 2.50 | | | | 2.70 | |
| | CG | (-.73,-.36,4.06) | | | | (-.02,-.06,3.68) | |
| | Inertia | 20.350 | 0.159 | 1.832 | 31.493 | 0 | 0 |
| | Tensor | 0.159 | 20.323 | -2.851 | 0 | 30.908 | 0 |
| | | 1.832 | -2.851 | 3.763 | 0 | 0 | 2.036 |
| Rt Hand | Weight | 1.00 | | | | 1.316 | |
| | CG | (0.14,-.17,2.67) | | | | (-.14,0,2.38) | |
| | Inertia | 3.200 | -.423 | -.197 | 4.260 | 0 | 0 |
| | Tensor | -.423 | 2.825 | -.426 | 0 | 3.816 | 0 |
| | | -.197 | -.426 | 1.316 | 0 | 0 | 1.471 |
| Rt Upper Leg | Weight | 17.10 | | | | 16.83 | |
| | CG | (-.30,0.03,7.41) | | | | (-.23,-.01,7.30) | |
| | Inertia | 363.632 | -.821 | -.403 | 332.699 | 0 | 0 |
| | Tensor | -.821 | 383.07 | -13.019 | 0 | 345.840 | 0 |
| | | -.403 | -13.019 | 99.516 | 0 | 0 | 43.737 |
| Rt Lower Leg | Weight | 6.80 | | | | 6.75 | |
| | CG | 0.62,-.55,6.59) | | | | (0.23,.01,6.70) | |
| | Inertia | 138.120 | -.554 | -.119 | 181.457 | 0 | 0 |
| | Tensor | -.554 | 139.661 | 2.198 | 0 | 181.944 | 0 |
| | | -.119 | 2.198 | 16.268 | 0 | 0 | 8.367 |

TABLE 5. SMALL ADAM MASS PROPERTY COMPARISON (MECHANICAL AXIS SYSTEMS) (continued)

| Segment | | Specification Data | | | Analytical Data | | |
|---|---|---|---|---|---|---|---|
| Rt Foot | Weight | 1.70 | | | | 1.76 | |
| | CG | (-2.24,-.08,1.38) | | | | (-1.71,0.01,0.94) | |
| | Inertia | 3.562 | -1.129 | 3.319 | 1.639 | 0 | 0 |
| | Tensor | -1.129 | 10.656 | 0.589 | 0 | 9.993 | 0 |
| | | 3.319 | 0.589 | 9.739 | 0 | 0 | 9.919 |
| Total Manikin | Weight | 139.50 | | | | 143.27 | |

*Mechanical axis systems were not required.
Units: Weight = pounds, center of gravity = inches, moments of inertia = lb in$^2$

TABLE 6. LARGE ADAM MASS PROPERTY COMPARISON (MECHANICAL AXIS SYSTEMS)

| Segment | | Specification Data | | | Analytical Data | | |
|---|---|---|---|---|---|---|---|
| Head* | Weight | 9.8 | | | 9.84 | | |
| Neck* | Weight | 2.8 | | | 2.74 | | |
| Upper Torso | Weight | 75.91 | | | 73.61 | | |
| | CG | (-1.52,0,5.50) | | | (-1.20,0,3.97) | | |
| | Inertia | 3217.90 | -0.32 | -333.71 | 2802.250 | 0 | 0 |
| | Tensor | -0.32 | 2637.51 | 0.14 | 0 | 1822.480 | 0 |
| | | -333.71 | 0.14 | 1790.81 | 0 | 0 | 1567.480 |
| Lower Torso | Weight | 29.30 | | | 39.75 | | |
| | CG | (-2.33,0,2.22) | | | (-1.11,0,3.62) | | |
| | Inertia | 461.96 | 0 | -19.89 | 728.850 | 0 | 0 |
| | Tensor | 0 | 432.82 | 0 | 0 | 258.180 | 0 |
| | | -19.89 | 0 | 531.66 | 0 | 0 | 788.660 |
| Rt Upper Arm | Weight | 5.40 | | | 5.12 | | |
| | CG | (-.29,-.47,6.05) | | | (0.01,-.01,5.99) | | |
| | Inertia | 64.666 | 2.478 | 4.502 | 71.390 | 0 | 0 |
| | Tensor | 2.478 | 65.536 | -4.292 | 0 | 68.507 | 0 |
| | 4.502 | -4.292 | 14.312 | 0 | 0 | 7.304 | |

TABLE 6. LARGE ADAM MASS PROPERTY COMPARISON (MECHANICAL AXIS SYSTEMS) (continued)

| Segment | | Specification Data | | | Analytical Data | | |
|---|---|---|---|---|---|---|---|
| Rt | Weight | 3.70 | | | 3.70 | | |
| Forearm | CG | (-.75,-.35,4.67) | | | (0,-.07,4.48) | | |
| | Inertia | 39.553 | 0.298 | 3.544 | 59.598 | 0 | 0 |
| | Tensor | 0.298 | 39.518 | -5.516 | 0 | 58.818 | 0 |
| | | 3.544 | -5.516 | 7.482 | 0 | 0 | 3.390 |
| Rt Hand | Weight | 1.30 | | | 1.32 | | |
| | CG | (.06,-.08,3.07) | | | (-.14,0,2.38) | | |
| | Inertia | 5.106 | -.472 | -.361 | 4.260 | 0 | 0 |
| | Tensor | -.472 | 4.666 | -.735 | 0 | 3.816 | 0 |
| | -.361 | -.735 | 2.206 | 0 | 0 | 1.471 | |
| Rt Upper Leg | Weight | 25.90 | | | 26.47 | | |
| | CG | (-.20,0.08,8.50) | | | (-.25,-.01,8.44) | | |
| | Inertia | 724.17 | -1.625 | -.793 | 653.044 | 0 | 0 |
| | Tensor | -1.625 | 762.675 | -25.631 | 0 | 678.990 | 0 |
| | | -.793 | -25.631 | 204.424 | 0 | 0 | 88.298 |
| Rt Lower Leg | Weight | 10.00 | | | 9.96 | | |
| | CG | (0.90,-.47,7.20) | | | (0.47,0,7.20) | | |
| | Inertia | 271.252 | -1.218 | -.236 | 296.067 | 0 | 0 |
| | Tensor | -1.218 | 274.652 | 4.33 | 0 | 298.598 | 0 |
| | | -0.236 | 4.33 | 31.376 | 0 | 0 | 18.203 |
| Rt Foot | Weight | 2.50 | | | 2.034 | | |
| | CG | (-2.32,-.06,1.60) | | | (1.54,0.01,0.863) | | |
| | Inertia | 6.687 | -2.299 | 6.084 | 2.940 | 0 | 0 |
| | Tensor | -2.299 | 18.922 | 1.241 | 0 | 15.180 | 0 |
| | | 6.084 | 1.241 | 17.281 | 0 | 0 | 13.990 |
| Total Manikin | Weight | 215.41 | | | 223.15 | | |

*Mechanical axis systems were not required.

Units: Weight = pounds, center of gravity = inches, moments of inertia = lb $in^2$

TABLE 7. SMALL ADAM MASS PROPERTY COMPARISON (ANATOMICAL AXIS SYSTEMS)

| Segment | | Specification Data | | | Analytical Data | | | Difference | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Head | Weight | 9.2 | | | 9.2 | | | 0% | | |
| | CG | (-0.4,0,1.3) | | | (-.19,0,1.21) | | | (+.21,0,-.09) | | |
| | Principal | .171 | 0 | 0 | .222 | 0 | -.002 | 30% | | |
| | Inertia | 0 | .194 | 0 | 0 | .230 | 0 | | 19% | |
| | Tensor | 0 | 0 | .127 | -.002 | 0 | .129 | | | 2% |
| Neck | Weight | 2.0 | | | 2.34 | | | 17% | | |
| | CG | (2.1,0,1.9) | | | (2.31,0,1.42) | | | (+.21,0,-.48) | | |
| | Principal | .011 | 0 | 0 | .018 | 0 | 0 | 64% | | |
| | Inertia | 0 | .014 | 0 | 0 | .017 | 0 | | 21% | |
| | Tensor | 0 | 0 | .017 | 0 | 0 | .001 | | | 94% |
| Upper | Weight | 45.96 | | | 48.22 | | | 5% | | |
| Torso | CG | (2.36,0,5.89) | | | (2.02,0,6.53) | | | (-.34,0,+.64) | | |
| | Principal | 3.975 | 0 | .210 | 3.593 | 0 | .605 | 10% | | |
| | Inertia | 0 | 3.174 | 0 | 0 | 3.010 | 0 | | 5% | |
| | Tensor | .210 | 0 | 1.977 | .605 | 0 | 1.824 | | | 8% |
| Lower | Weight | 17.3 | | | 17.94 | | | 4% | | |
| Torso | CG | (-3.23,0,.45) | | | (-3.18,0,-.09) | | | (+.05,0,-.54) | | |
| | Principal | .473 | 0 | -.019 | .706 | 0 | .011 | 49% | | |
| | Inertia | 0 | .436 | 0 | 0 | .269 | 0 | | 38% | |
| | Tensor | -.019 | 0 | .577 | .011 | 0 | 0.747 | | | 29% |
| Rt Upper | Weight | 3.4 | | | 3.43 | | | 1% | | |
| Arm | CG | (.7,1.2,-6.2) | | | (.48,1.15,-5.95) | | | (-.22,-.05,+.25) | | |
| | Principal | .074 | 0 | 0 | .091 | 0 | -.010 | 23% | | |
| | Inertia | 0 | .077 | 0 | 0 | .093 | 0 | | 21% | |
| | Tensor | 0 | 0 | .015 | -.010 | 0 | .011 | | | 27% |
| Rt | Weight | 2.5 | | | 2.70 | | | 8% | | |
| Forearm | CG | (.9,0,-3.8) | | | (.14,.05,-3.40) | | | (-.76,+.05,+.40) | | |
| | Principal | .053 | 0 | 0 | .081 | 0 | -.002 | 53% | | |
| | Inertia | 0 | .054 | 0 | 0 | .077 | -.014 | | 43% | |
| | Tensor | 0 | 0 | .008 | -.002 | -.014 | .008 | | | 0% |
| Rt Hand | Weight | 1.0 | | | 1.32 | | | 32% | | |
| | CG | (-.3,-.2,0.4) | | | (0.04,-0.13,0.67) | | | (+.34,+.07,+.27) | | |
| | Principal | .009 | 0 | 0 | .011 | 0 | 0 | 22% | | |
| | Inertia | 0 | .007 | 0 | 0 | .010 | -.002 | | 43% | |
| | Tensor | 0 | 0 | .003 | 0 | -.002 | .004 | | | 33% |

TABLE 7. SMALL ADAM MASS PROPERTY COMPARISON (ANATOMICAL AXIS SYSTEMS) (continued)

| Segment | | Specification Data | | | Analytical Data | | | Difference | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Rt Upper Leg | Weight | 17.1 | | | 16.83 | | | 2% | | |
| | CG | (0.3,2.4,-7.4) | | | (0.23,2.37,-7.30) | | | (-.07,-.03,+.10) | | |
| | Principal | .941 | 0 | 0 | .861 | -.001 | .003 | 9% | | |
| | Inertia | 0 | .993 | 0 | -.001 | .893 | -.036 | | 10% | |
| | Tensor | 0 | 0 | .256 | .003 | -.036 | .115 | | | 55% |
| Rt Lower Leg | Weight | 6.8 | | | 6.75 | | | 1% | | |
| | CG | (-.4,-2.2,-5.6) | | | (-.20,-1.55,-5.70) | | | (+.20,+.65,-.10) | | |
| | Principal | .357 | 0 | 0 | .470 | 0 | -.002 | 32% | | |
| | Inertia | 0 | .362 | 0 | 0 | .471 | .008 | | 30% | |
| | Tensor | 0 | 0 | .042 | -.002 | .008 | .022 | | | 48% |
| Rt Foot | Weight | 1.7 | | | 1.76 | | | 4% | | |
| | CG | (-2.8,0,-0.2) | | | (-3.38,0.17,0.03) | | | (-.58,+.17,+.23) | | |
| | Principal | .005 | 0 | 0 | .008 | -.005 | -.007 | 60% | | |
| | Inertia | 0 | .028 | 0 | -.005 | .025 | -.002 | | 11% | |
| | Tensor | 0 | 0 | .029 | -.007 | -.002 | .023 | | | 21% |
| Total Manikin | Weight | 139.50 | | | 143.27 | | | 3% | | |

Units: Weight = pounds, center of gravity = inches, moments of inertia = lb in $sec^2$

TABLE 8. LARGE ADAM MASS PROPERTY COMPARISON (ANATOMICAL AXIS SYSTEMS)

| Segment | | Specification Data | | | Analytical Data | | | Difference | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Head | Weight | 9.80 | | | 9.84 | | | 0% | | |
| | CG | (-.3,0,1.1) | | | (-.31,0,1.28) | | | (-.01,0,+.18) | | |
| | Principal | .193 | 0 | 0 | .223 | 0 | 0 | 16% | | |
| | Inertia | 0 | .221 | 0 | 0 | .238 | 0 | | 8% | |
| | Tensor | 0 | 0 | .142 | 0 | 0 | .136 | | | 4% |
| Neck | Weight | 2.80 | | | 2.74 | | | 2% | | |
| | CG | (2.6,0,2.1) | | | (2.6,0,1.6) | | | (0,0,0.5) | | |
| | Principal | .020 | 0 | 0 | .020 | 0 | 0 | 0% | | |
| | Inertia | 0 | .024 | 0 | 0 | .020 | 0 | | 17% | |
| | Tensor | 0 | 0 | 0.031 | 0 | .009 | | | | 71% |

TABLE 8. LARGE ADAM MASS PROPERTY COMPARISON (ANATOMICAL AXIS SYSTEMS) (continued)

| Segment | | Specification Data | | | Analytical Data | | | Difference |
|---|---|---|---|---|---|---|---|---|
| Upper Torso | Weight | 75.91 | | | 73.61 | | | 3% |
| | CG | (2.70,0,6.69) | | | (2.30,0,8.20) | | | (-.40,0,+1.51) |
| | Principal | 8.497 | 0 | .303 | 6.985 | 0 | .885 | 18% |
| | Inertia | 0 | 6.826 | 0 | 0 | 4.717 | 0 | 31% |
| | Tensor | .303 | 0 | 4.465 | .885 | 0 | 4.324 | 3% |
| Lower Torso | Weight | 29.30 | | | 39.75 | | | 36% |
| | CG | (-3.58,0,0.34) | | | (-5.20,0,-.57) | | | -1.62,0,-.91) |
| | Principal | 1.182 | 0 | -.002 | 1.896 | 0 | .037 | 60% |
| | Inertia | 0 | 1.120 | 0 | 0 | .668 | 0 | 40% |
| | Tensor | -.002 | 0 | 1.390 | .037 | 0 | 2.032 | 46% |
| Rt Upper Arm | Weight | 5.40 | | | 5.12 | | | 5% |
| | CG | (0.6,1.2,-7.6) | | | (0.68,1.75,-7.55) | | | (+.08,+.55,+.05) |
| | Principal | .164 | 0 | 0 | .179 | -.004 | -.020 | 9% |
| | Inertia | 0 | .175 | 0 | -.004 | .180 | -.001 | 3% |
| | Tensor | 0 | 0 | .035 | -.020 | -.001 | .021 | 40% |
| Rt Forearm | Weight | 3.70 | | | 3.70 | | | 0% |
| | CG | (1.0,0,-4.3) | | | (0.21,-.02,-4.08) | | | (-.79,-.02,+.22) |
| | Principal | .103 | 0 | 0 | .154 | 0 | -.004 | 50% |
| | Inertia | 0 | .105 | 0 | 0 | .147 | -.027 | 40% |
| | Tensor | 0 | 0 | .016 | -.004 | -.027 | .014 | 13% |
| Rt Hand | Weight | 1.3 | | | 1.32 | | | 2% |
| | CG | (-.4,-.2,0.5) | | | (.04,-.13,0.67) | | | (+.44,+.07,+.17) |
| | Principal | .014 | 0 | 0 | .011 | 0 | 0 | 21% |
| | Inertia | 0 | .012 | 0 | 0 | .010 | -.002 | 17% |
| | Tensor | 0 | 0 | .005 | 0 | -.002 | .004 | 20% |
| Rt Upper Leg | Weight | 25.9 | | | 26.47 | | | 2% |
| | CG | (0.2,2.9,-8.5) | | | (.25,2.81,-8.43) | | | (+.05,-.09,+.07) |
| | Principal | 1.874 | 0 | 0 | 1.690 | -.002 | .005 | 10% |
| | Inertia | 0 | 1.977 | 0 | -.002 | 1.754 | -.070 | 11% |
| | Tensor | 0 | 0 | .526 | .005 | -.070 | .232 | 56% |
| Rt Lower Leg | Weight | 10.0 | | | 9.96 | | | 0.4% |
| | CG | (-.5,-2.4,-6.1) | | | (-.23,-1.80,-6.10) | | | (+.27,+.60,0) |
| | Principal | .701 | 0 | 0 | .767 | -.002 | -.003 | 9% |
| | Inertia | 0 | .712 | 0 | -.002 | .772 | .013 | 8% |
| | Tensor | 0 | 0 | .081 | -.003 | .013 | .047 | 42% |

TABLE 8. LARGE ADAM MASS PROPERTY COMPARISON (ANATOMICAL AXIS SYSTEMS) (continued)

| Segment | | Specification Data | | | Analytical Data | | | Difference | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Rt Foot | Weight | 2.5 | | | 2.034 | | | 19% | | |
| | CG | (-3.3,0,-.30) | | | (-4.24,-.02,.21) | | | (-.94,-.02,+.51) | | |
| | Principal | .009 | 0 | 0 | .013 | -.007 | -.009 | 44% | | |
| | Inertia | 0 | .050 | 0 | -.007 | .037 | -.003 | | 26% | |
| | Tensor | 0 | 0 | .052 | -.009 | -.003 | .032 | | | 38% |
| Total Manikin | Weight | 215.41 | | | 223.15 | | | 4% | | |

Units: Weight = pounds, center of gravity = inches, moments of inertia = lb in $sec^2$

2.2.2.6. Results

As an example of the output of the MASSPR program, the detailed results for the small forearm are shown in Figure 26. The summarized results for all small and large segments are shown in Tables 7 and 8. The results are compared to the specification data and the differences between the two sets of data are listed.

## 2.2.3. Loading Analysis

Structural design and analysis of the ADAM system requires a knowledge of the loads encountered in the ejection sequence. The purpose of this section is to define the procedures by which these load predictions were made and to document the results of the analysis. Included is a discussion of the three types of loading on the ADAM including aerodynamic loading, dynamic loading resulting from response to the aerodynamic loading, and the inertial loading on internal system elements.

In the following sections, the theory behind the aerodynamic and dynamic loading configurations will first be introduced. Then the mathematical model which represents the manikin will be discussed, and the derivation of the equations of motion of the system will be presented. The program which calculates the aerodynamic and dynamic loading on the system will then be shown. Finally, the inertial loading on the system and the conclusions of the analysis will be presented.

| SEGMENT NUM | SEGMENT NAME | NUM OF SUBSEGS | SUBSEG NUM | SUB-SEGMENT NAME | NUM OF PARTS | PART NUM | SHAPE TYPE | DENS<br>MASS | LX<br>L<br>CGX<br>IX | LY<br>OUTDIA<br>CGY<br>IY | LZ<br>INDIA AX<br>CGZ<br>IZ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | FOREARMS | 2 | | | | | | | | | | |
| | | | 1 | BONE | 17 | | | | | | | |
| | | | | | | 1 | 3 | 0.2830 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 0.00 | 0.00 | 0.00 | 0 0 |
| | | | | | | | | | 0.00 | 0.00 | 10.12 | |
| | | | | | | | | 0.19 | 0.022 | 0.022 | 0.032 | |
| | | | | | | 2 | 3 | 0.0975 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 0.00 | 0.00 | 0.00 | 0 0 |
| | | | | | | | | | 0.00 | 0.00 | 9.60 | |
| | | | | | | | | 0.08 | 0.023 | 0.037 | 0.030 | |
| | | | | | | 3 | 2 | 0.0975 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 0.56 | 0.87 | 0.69 | 3 0 |
| | | | | | | | | | 0.00 | 0.00 | 8.82 | |
| | | | | | | | | 0.01 | 0.001 | 0.001 | 0.002 | |
| | | | | | | 4 | 2 | 0.2830 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 6.34 | 0.69 | 0.53 | 3 0 |
| | | | | | | | | | 0.00 | 0.00 | 5.95 | |
| | | | | | | | | 0.28 | 0.934 | 0.934 | 0.026 | |
| | | | | | | 5 | 3 | 0.2830 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 0.00 | 0.00 | 0.00 | 0 0 |
| | | | | | | | | | 0.00 | 0.00 | 5.80 | |
| | | | | | | | | 0.05 | 0.002 | 0.009 | 0.008 | |
| | | | | | | 6 | 3 | 0.2830 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 0.00 | 0.00 | 0.00 | 0 0 |
| | | | | | | | | | 0.00 | 0.00 | 5.11 | |
| | | | | | | | | 0.17 | 0.046 | 0.046 | 0.055 | |
| | | | | | | 7 | 2 | 0.2830 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 2.54 | 0.88 | 0.69 | 3 0 |
| | | | | | | | | | 0.00 | 0.00 | 4.05 | |
| | | | | | | | | 0.17 | 0.104 | 0.104 | 0.026 | |
| | | | | | | 8 | 3 | 0.2830 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 0.00 | 0.00 | 0.00 | 0 0 |
| | | | | | | | | | 0.00 | 0.00 | 0.99 | |
| | | | | | | | | 0.65 | 1.336 | 1.064 | 0.443 | |
| | | | | | | 9 | 2 | 0.1400 | 0.00 | 0.00 | 0.00 | |
| | | | | | | | | | 0.25 | 1.00 | 0.00 | 3 0 |
| | | | | | | | | | 0.00 | 0.00 | 2.41 | |
| | | | | | | | | 0.03 | 0.002 | 0.002 | 0.003 | |

Figure 26. MASSPR Output--Small Forearm

| | | | | | | |
|---|---|---|---|---|---|---|
| 10 | 2 0.2830 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.30 | 1.15 | 0.00 | 3.0 |
| | | | 0.00 | 0.00 | 2.13 | |
| | | 0.09 | 0.008 | 0.006 | 0.015 | |
| 11 | 3 0.0975 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.00 | 0.02 | 0.00 | 0.0 |
| | | | 0.00 | 0.00 | 1.75 | |
| | | 0.03 | 0.003 | 0.006 | 0.012 | |
| 12 | 3 0.2830 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.00 | 0.00 | 0.00 | 0.0 |
| | | | 0.00 | -0.15 | 0.00 | |
| | | 0.10 | 0.043 | 0.003 | 0.043 | |
| 13 | 2 0.0854 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.30 | 1.25 | 0.50 | 2.0 |
| | | | 0.00 | -0.47 | 0.00 | |
| | | 0.03 | 0.003 | 0.006 | 0.003 | |
| 14 | 2 0.0975 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.31 | 0.75 | 0.50 | 2.0 |
| | | | 0.00 | 0.51 | 0.00 | |
| | | 0.01 | 0.000 | 0.001 | 0.000 | |
| 15 | 2 0.1520 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.25 | 1.25 | 0.50 | 2.0 |
| | | | 0.00 | -0.79 | 0.00 | |
| | | 0.04 | 0.005 | 0.009 | 0.005 | |
| 16 | 2 0.2830 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.22 | 0.86 | 0.50 | 2.0 |
| | | | 0.00 | -1.06 | 0.00 | |
| | | 0.02 | 0.002 | 0.003 | 0.002 | |
| 17 | 3 0.0975 | | 0.00 | 0.00 | 0.00 | |
| | | | 0.00 | 0.00 | 0.00 | 0.0 |
| | | | 0.00 | 1.00 | 0.00 | |
| | | 0.04 | 0.005 | 0.009 | 0.005 | |
| SUBSEGMENT TOTALS | | | | | | |
| | | | 0.00 | -0.02 | 3.65 | |
| | | 1.99 | 23.917 | 23.537 | 0.810 | |
| 2 SKIN 13 | | | | | | |
| 1 | 2 0.0100 | | 0.00 | 0.00 | 0.00 | |
| | | | 2.40 | 2.03 | 1.75 | 3.0 |
| | | | 0.00 | -0.10 | 9.60 | |
| | | 0.02 | 0.018 | 0.018 | 0.018 | |
| 2 | 2 0.0400 | | 0.00 | 0.00 | 0.00 | |
| | | | 2.40 | 2.15 | 2.03 | 3.0 |
| | | | 0.00 | -0.10 | 9.60 | |
| | | 0.04 | 0.040 | 0.040 | 0.043 | |
| 3 | 2 0.0100 | | 0.00 | 0.00 | 0.00 | |
| | | | 2.30 | 2.28 | 0.70 | 3.0 |
| | | | 0.00 | -0.27 | 7.25 | |
| | | 0.08 | 0.067 | 0.067 | 0.060 | |

Figure 26. MASSPR Output--Small Forearm (continued)

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 2 | 0.0400 | 0.00 | 0.00 | 0.00 | |
| | | | 2.30 | 2.40 | 2.28 | 3.0 |
| | | | 0.00 | -0.27 | 7.25 | |
| | | 0.04 | 0.047 | 0.047 | 0.058 | |
| 5 | 2 | 0.0100 | 0.00 | 0.00 | 0.00 | |
| | | | 1.20 | 2.55 | 1.50 | 3.0 |
| | | | 0.00 | -0.51 | 5.50 | |
| | | 0.04 | 0.027 | 0.027 | 0.044 | |
| 6 | 2 | 0.0400 | 0.00 | 0.00 | 0.00 | |
| | | | 1.20 | 2.67 | 2.55 | 3.0 |
| | | | 0.00 | -0.51 | 5.50 | |
| | | 0.02 | 0.024 | 0.024 | 0.042 | |
| 7 | 2 | 0.0100 | 0.00 | 0.00 | 0.00 | |
| | | | 2.00 | 2.85 | 0.87 | 3.0 |
| | | | 0.00 | -0.33 | 3.90 | |
| | | 0.12 | 0.103 | 0.103 | 0.126 | |
| 8 | 2 | 0.0400 | 0.00 | 0.00 | 0.00 | |
| | | | 2.00 | 2.97 | 2.95 | 3.0 |
| | | | 0.00 | -0.33 | 3.90 | |
| | | 0.05 | 0.064 | 0.064 | 0.097 | |
| 9 | 3 | 0.0100 | 0.00 | 0.00 | 0.00 | |
| | | | 0.00 | 0.00 | 0.00 | 0.0 |
| | | | 0.00 | -0.28 | 1.90 | |
| | | 0.10 | 0.111 | 0.119 | 0.161 | |
| 10 | 2 | 0.0400 | 0.00 | 0.00 | 0.00 | |
| | | | 2.00 | 3.22 | 3.10 | 3.0 |
| | | | 0.00 | -0.28 | 1.90 | |
| | | 0.05 | 0.079 | 0.079 | 0.124 | |
| 11 | 3 | 0.0100 | 0.00 | 0.00 | 0.00 | |
| | | | 0.00 | 0.00 | 0.00 | 0.0 |
| | | | -0.44 | -0.19 | -0.20 | |
| | | 0.08 | 0.331 | 0.141 | 0.372 | |
| 12 | 3 | 0.0400 | 0.00 | 0.00 | 0.00 | |
| | | | 0.00 | 0.00 | 0.00 | 0.0 |
| | | | -0.44 | -0.19 | -0.20 | |
| | | 0.07 | 0.000 | 0.000 | 0.000 | |
| 13 | 1 | 0.0000 | 0.00 | 0.00 | 0.00 | |
| | | | 0.00 | 0.00 | 0.00 | 0.0 |
| | | | 0.00 | 0.00 | 0.00 | |
| | | 0.00 | 0.000 | 0.000 | 0.000 | |
| SUBSEGMENT TOTALS | | | | | | |
| | | | -0.09 | [illegible] | 3.79 | |
| | | 0.71 | 7.481 | 7.315 | 1.177 | |
| SEGMENT TOTALS | | | | | | |
| | | | -0.02 | -0.09 | 3.69 | |
| | | 2.70 | 31.443 | 30.966 | 2.027 | |

Figure 26. MASSPR Output--Small Forearm (continued)

### 2.2.3.1. Static and Dynamic Loading Analysis Theory

#### 2.2.3.1.1. Static Aerodynamic Loading

The static aerodynamic forces and moments acting on each ADAM limb at the limit of the particular range of motion were calculated. For a more realistic analysis, each limb was separated into three segments, as defined by the joint centers of rotation, and the forces and moments acting on these segments were calculated independently. The results were integrated to create the total aerodynamic forces and moments along each limb.

The static segment aerodynamic loads were calculated using the data at the final position of the limb by means of the commonly accepted aerodynamic drag equation:

$$\text{Drag} = 1/2\, \rho\, V^2\, C_d\, C_f\, A$$

where

- $V$ = Relative air velocity calculated at the segment aerodynamic center (ft/sec)
- $\rho$ = Air density (0.002378 slugs/ft$^3$)
- $C_d$ = Drag coefficient
- Cf = Clothing factor
- $A$ = Cross-sectional area of the segment perpendicular to the air stream (ft$^2$)

The velocity used to calculate the drag for each segment as shown in Figure 27 is defined as:

$$V = [V_o - \dot{x}]\ (\sin\theta) - \sigma_m\ (\dot{\theta})$$

where

- $V_o$ = Free stream air speed (ft/sec)
- $\dot{x}$ = Torso velocity (ft/sec)
- $\theta$ = Limb angle with respect to air stream (rad)
- $\sigma_m$ = Segment aerodynamic moment arm (ft)
- $\dot{\theta}$ = Limb angular velocity (rad/sec)

As can be seen from the above expression, the free stream velocity was not used in the drag calculation as it was much higher than the velocity actually acting on the limb. Reacting to the drag forces acting on each body, the motions of the limb and torso were in the positive direction of the free stream and created a differential velocity. Therefore, the torso and limb tangential velocities were subtracted from the free stream velocity to obtain the velocity used in the aerodynamic drag equation.

The drag coefficient for the limbs was obtained from Hoerner (1965). The value of 1.8 represents the experimentally determined drag coefficient for a cylinder at a Mach number equivalent to 700 KEAS at sea level and 1.5 was determined for a 450 KEAS air speed. In this case, the drag coefficient was defined as the experimentally measured drag divided by the dynamic pressure q:

$$C_d = \frac{2D}{\rho V_o^2 A}$$

Since compressibility was inherently accounted for in the experimental measurement, no additional compressibility correction was necessary. The factor 1.4, which represents the clothing factor for all limbs, is a conservative value derived from Payne (1975, 1974). A lower clothing factor (1.3) was used for the foot as a shoe creates a lower drag than loose clothing. The hand was assumed to be bare in the testing situation and, therefore, the clothing factor was unity.

The above drag equation was also used to obtain the aerodynamic force acting on the torso/seat ($Q_x$). In this case, the drag coefficient used was 1.3 (Specker, 1985), the velocity used was the torso velocity subtracted from the free stream velocity, and the clothing factor was unity.

In calculating the drag force acting on each segment, the segment lengths were taken as the distances between the joint centers as given in the ADAM specification. The segment cross-sectional dimensions were based on tri-service data also provided by the ADAM specification. A constant cross-sectional area based on a conservatively weighted average for each segment was used in the calculation of each segment area.

The aerodynamic moment ($M_{aero}$) of the total limb evaluated at the most proximal joint center was calculated using the program described in Section 2.2.3.2 by integrating the shear values in the shear diagram from the most distal point of the limb to the joint center.

2.2.3.1.2. Dynamic Loading on ADAM Limbs

An analysis was developed to provide a procedure for predicting the dynamic loading on ADAM limbs resulting from time dependent velocity response to aerodynamic loading. This analysis focused on predicting realistic loads while using an approach that was easy to understand and apply. No attempt was made to analyze all possible combinations of complex motions because these motions depend on initial conditions that can change from test to test. Instead, emphasis was directed at defining a worst case loading experienced in reasonable test conditions. The description of the worst case and the definition procedure is described in Section 2.2.3.3.

The results from the dynamic loading analysis were used to obtain dynamic amplification factors. These are multiplication factors which were applied to the static aerodynamic loads along the limbs to obtain the total limb loading.

A description of the procedure and mathematical model used in this analysis, as well as the resulting data, will be discussed in the following sections.

2.2.3.1.2.1. Analytical Model

Originally, the mathematical model was based on conservative assumptions such as there was no deceleration of the torso/seat. The loads found through this model created stresses in the designs that were unreasonably high and could not be reduced through reasonable engineering methods. A reevaluation of the model and associated assumptions was then undertaken to provide a more detailed and realistic loading prediction. In the reanalysis, fewer conservative assumptions were made and the resulting loads were lower than the previous prediction. The revised assumptions for the analytical model are described as follows:

- The manikin is assumed to be ejected directly into the free stream and is subjected to a direct aerodynamic loading in the drag direction.

- The torso/seat is allowed motion in the x direction due to the aerodynamic drag forces but no rotation.

- Both the arm and leg are considered to be extended and locked at the elbow and knee, respectively, as shown in Figure 27.

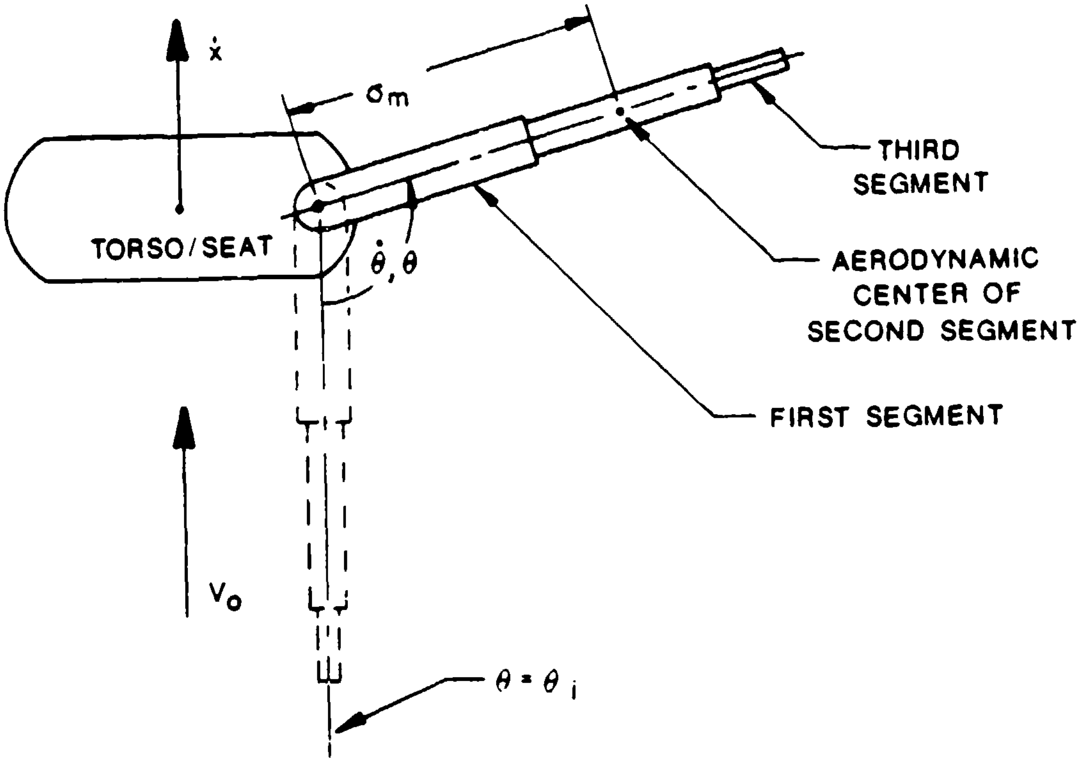



Figure 27. Mathematical Model System

- The pivot points of the arm and leg models are located at the shoulder and hip centers of rotation, respectively. They are also, by definition, the points of maximum shear and moment loading.

- The model is applicable to either the abduction/adduction or the flexion/extension directions of motion; however, only one amplification factor will be used for all directions. Both directions for the arm and leg were analyzed, and the worst case loading was used to find the dynamic amplification factor for each limb.

- When the moments produced by the inertial loading are compared to aerodynamic moments, they can be seen to be significantly lower than the aerodynamic moments. Furthermore, the two sources of moment tend to be inherently out of phase because of the order in which the loading mechanisms occur. The inertial loads certainly must be considered significant and are likely to have the effect of reducing aerodynamic loading; however, they have not been considered in the analysis documented herein as the intent was to produce a conservative set of loads.

The final limb moment total is the sum of the dynamic moment applied through the stops and the static aerodynamic moment calculated with the limb at the fully rotated position. The dynamic amplification factor is the quotient of the total moment divided by the limb static aerodynamic moment ($M_{aero}$).

2.2.3.1.2.2. Equations of Motion

The equations of motion of this system were developed using the conservation of energy equations:

$$KE = 0.5\,(M + m)\,\dot{x}^2 + (I_\theta/2)\,\dot{\theta}^2 + m\,r\,\dot{\theta}\,\dot{x}\,\sin\theta$$

$$PE = 0$$

where

KE = Kinetic energy of the system
PE = Potential energy of the system
M = Torso/seat mass (slugs)
m = Limb mass (slugs)
$I_\theta$ = Limb moment of inertia about pivot point (slugs-ft$^2$)
$\dot{x}$ = Torso/seat velocity (ft/sec)
$\dot{\theta}$ = Limb rotational velocity (rad/sec)
r = Radial distance from torso/limb joint to center of gravity of limb (ft)

and applying the Lagrangian technique. The resulting equations were:

$$(M + m)\,\ddot{x} + \left[(m\,r\,\sin\theta)\,\ddot{\theta}\right] + (m\,r\,\cos\theta)\,\dot{\theta}^2 = Q_x$$

$$I_\theta\,\ddot{\theta} + (m\,r\,\sin\theta)\,\ddot{x} = Q_\theta$$

where

$\ddot{x}$ = Torso/seat translational acceleration (ft/sec$^2$)

$\theta$ = Limb rotational motion (rad)

$\ddot{\theta}$ = Limb rotational acceleration (rad/sec2)

$Q_x$ = Aerodynamic force on torso/seat (lbs)

$Q_\theta$ = Aerodynamic moment on limb (ft-lbs)

See Figure 28 for a pictorial description of the terms.

These equations were solved simultaneously using the Runge-Kutta forward integration technique. $Q_x$ and $Q_\theta$ were calculated at each time step using the equations found in Section 2.2.3.1.1. Results from the equations included $x$, $\dot{x}$, $\ddot{x}$, $\theta$, $\dot{\theta}$, and $\ddot{\theta}$ for each time step.

2.2.3.1.2.3 Calculation of Stop Forces Using the Energy Approach

The kinetic energy of the limb with respect to the torso is:

$$KE_{limb} = \frac{I_\theta \dot{\theta}^2}{2}$$

The kinetic energy increases as the angular velocity increases under the influence of the aerodynamic force. This increase continues until the arm contacts the joint stops. The simplest way to calculate the force at the stop is to calculate the work done in compressing the soft stops and equating the work to the arm kinetic energy. For the purposes of this analysis, it was assumed that the stop or stops were located at distance $\sigma$ from the joint center of rotation as shown in Figure 29. It was further assumed that the compressive force F on the soft stop is constant over the full range of soft stop deflection. The work done by compression of the soft stop is:

$$\begin{aligned} W &= \text{(force) (deflection)} \\ &= (F)\ (\sigma\ \alpha) \end{aligned}$$

where $\alpha$ is the rotational angle through which the soft stop is deflected.

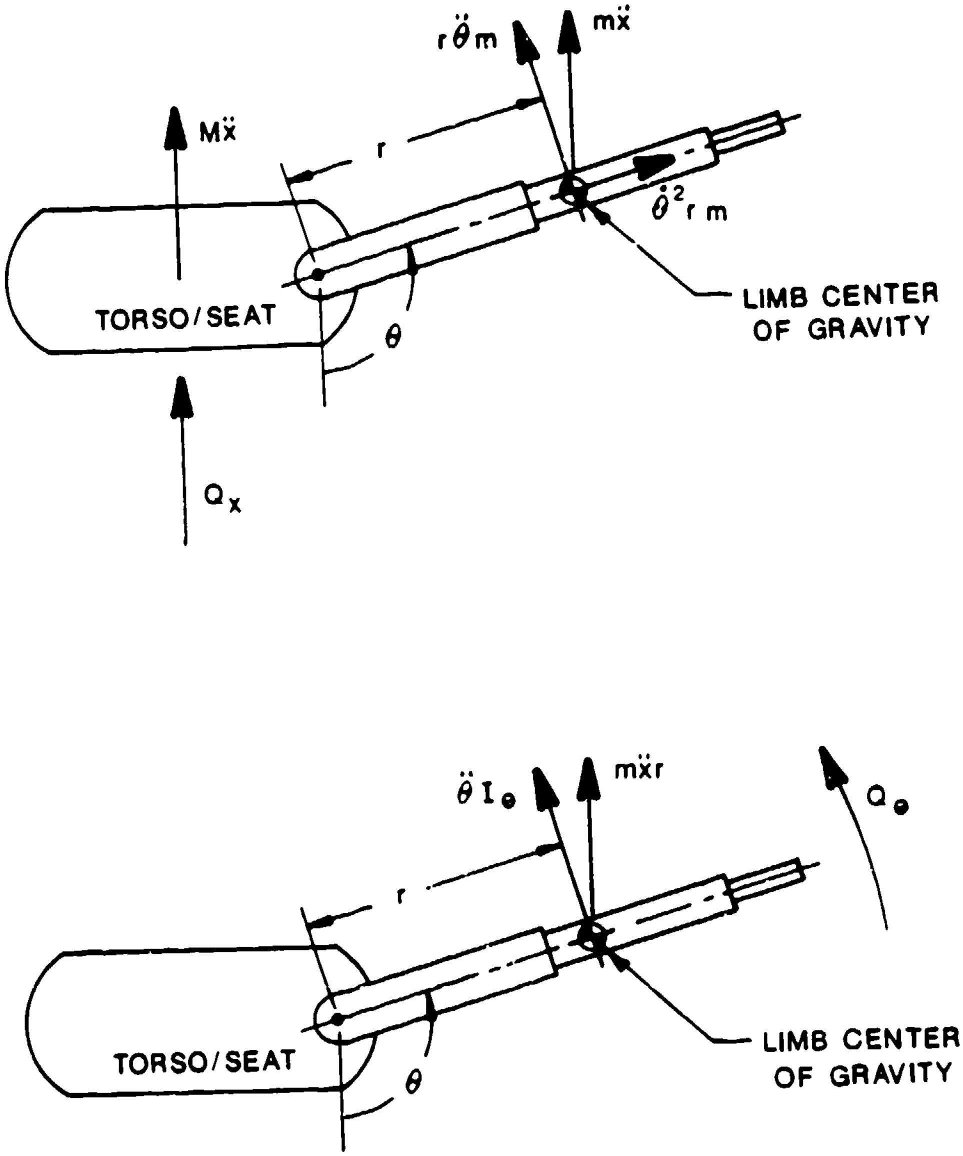



Figure 28. Mathematical Model Free Body Diagrams

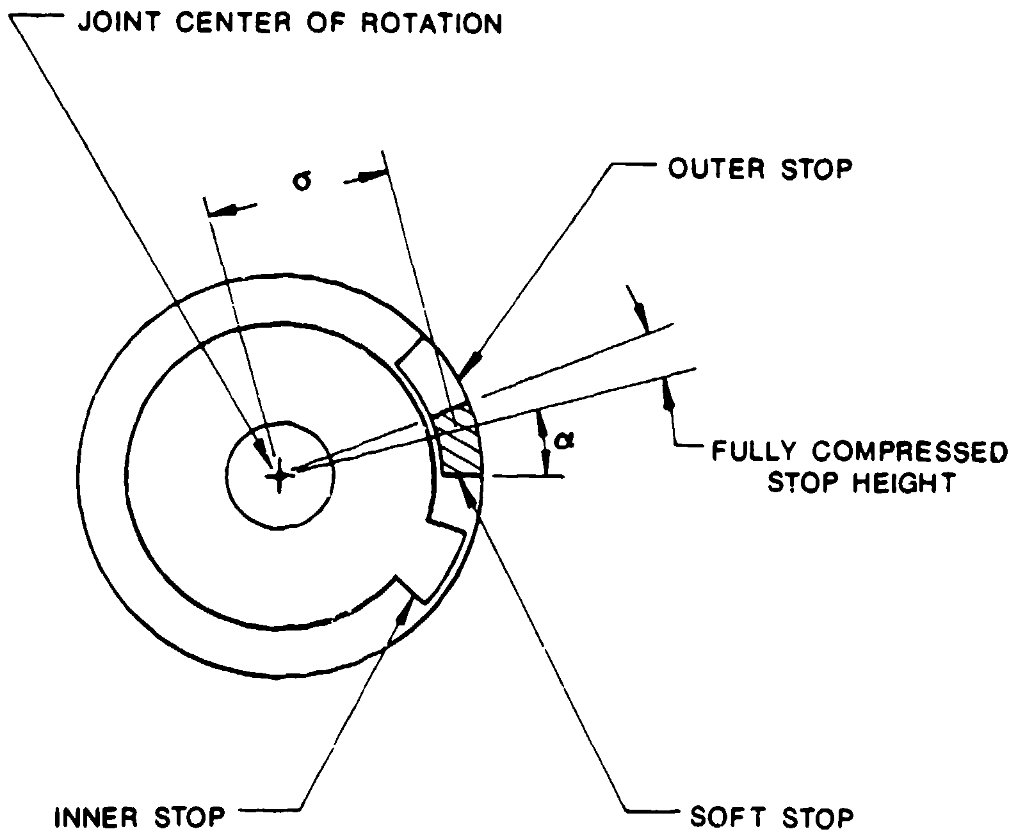



Figure 29. Joint Soft Stop Arrangement

It was assumed that, when the arm or leg motion is stopped in relation to the torso, all kinetic energy has been transferred to potential energy through work done in the stop. Thus,

$$W = KE_{limb}$$

$$F \sigma \alpha = \frac{I_\theta \dot{\theta}^2}{2}$$

$$\text{and } F = \frac{I_\theta \dot{\theta}^2}{2 \sigma \alpha}$$

Several situations existed where this calculation had to reflect the stop configurations. These configurations included the following:

- Multiple stops located opposite each other across the axis of rotation as shown in Figure 30.
- Multiple stops located on the same side of the axis of rotation as shown in Figure 31.
- Series joints as shown in Figure 32.

A calculation procedure was developed to provide for these variations in joint configurations by simply dividing the calculated force by the total number of stops, N, to determine the force per stop. Thus,

$$F_{per\ stop} = \frac{I_\theta \ \dot{\theta}^2}{2 N \sigma \alpha}$$

To calculate $\alpha$, the stop was assumed to compress fully to a zero thickness configuration. Therefore, the uncompressed stop thickness was used for the stop compression distance to calculate the corresponding $\alpha$ value based on the equation:

$$\alpha = \frac{\text{stop compression distance}}{\sigma}$$

Knowing the force per stop, the moment produced by this force is $M_{dyn} = F\,\sigma$. The dynamic amplification factor, AF, based on a moment ratio is:

$$AF = \frac{M_{dyn} + M_{aero}}{M_{aero}}$$

2.2.3.2. Loading Analysis Program

A computer program was written to provide an effective means of conducting the dynamic loading analysis. This program uses a forward integration technique, Runge-Kutta, to solve the differential equations of motion of the limb rotation and the torso linear movement under the influence of aerodynamic loading as outlined in Section 2.2.3.1.2.2. The listing for this program is shown in Appendix E (ADAMLD3).

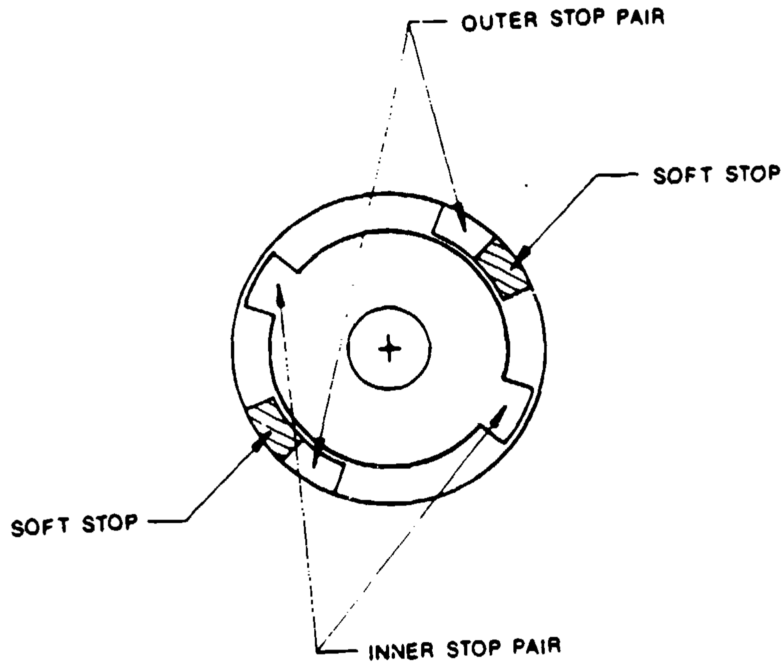



Figure 30. Joint With Multiple Stops Located Across Axis of Rotation

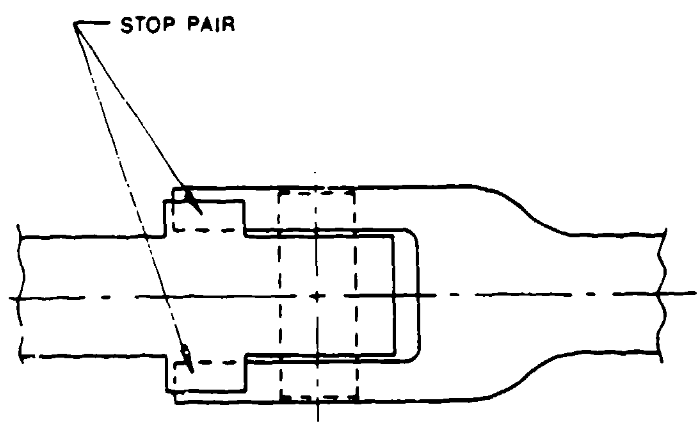



Figure 31. Multiple Stops on the Same Side of the Axis of Rotation

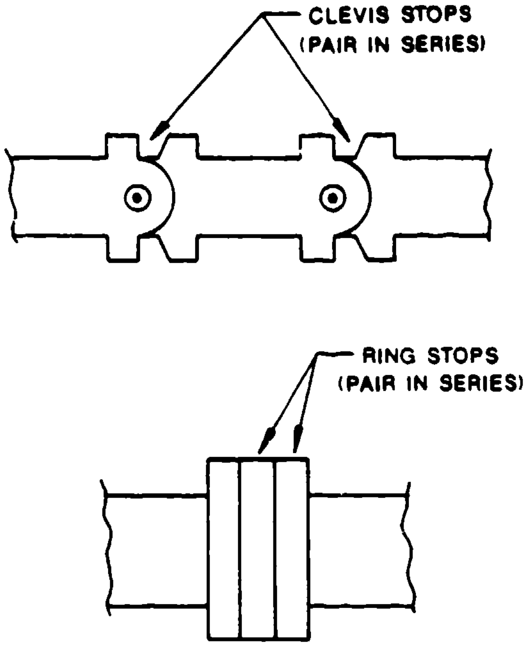



Figure 32. Joint Stops in Series

This program models the limb/torso/seat movement in the free stream. It allows input to define starting conditions including limb angle with respect to the free stream, initial torso/seat velocity and acceleration, free stream velocity, static aerodynamic torque, and integration time step. Runge-Kutta uses a step-by-step integration procedure that calculates the limb angle, angular velocity, angular acceleration, and the torso linear displacement, velocity, and acceleration for each time step.

The program also tabulates the accumulative angular position with respect to the free stream, the total elapsed time, and the kinetic energy at the end of the unrestrained angular displacement. This kinetic energy is then used to calculate the dynamic amplification factor.

The output for the case of the large arm is presented in Figure 33. As shown in the output, the displacements, velocities, and accelerations at the stop interface as well as the aerodynamic moment at the pivot point and the dynamic amplification factor are all calculated and displayed.

LARGE ADAM ARM

PROGRAM: ADAMLD3.FTN

```
ENTER THE FREESTREAM VEL(FPS) AND AIR DENSITY(LB/FT3
  1181.600       2.3779999E-03
ENTER INITIAL TORSO X DISP (FT), INITIAL VEL (FPS)AND INITIAL ACCEL (FPS2)
 0.0000000       0.0000000       0.0000000


ENTER INITIAL LIMB THETA DISP (RAD), INITIAL ANG VEL (RPS),AND ANG ACCEL (RDPS2)
  1.220000       0.0000000       0.0000000


ENTER DELTA TIME STEP (SEC)
 2.4999999E-03
ENTER RANGE OF MOTION WO SOFT STOP PRESENT (RAD)
  2.970000
ENTER SOFT STOP DEPTH AND RADIUS IN INCHES
 0.2500000       1.000000
ENTER NO OF STOPS IN SERIES AND STOP PAIRS
  2.000000       1.000000


ENTER TORSO-SEAT AREA AND DRAG COEF
  6.120000       1.300000


ENTER TORSO-SEAT AND LIMB MASS
  14.70000       0.3200000


ENTER NUMBER OF LIMB SECTIONS-INTEGER
     3


ENTER AERO AREA, MOM ARM, CD AND CLOTH FAC FOR
     1 TH LIMB SECTION
 0.4700000       0.4900000       1.800000       1.400000

ENTER AERO AREA, MOM ARM, CD AND CLOTH FAC FOR
     2 TH LIMB SECTION
 0.2100000       1 480000       1.800000       1.400000

ENTER AERO AREA, MOM ARM, CD AND CLOTH FAC FOR
     3 TH LIMB SECTION
 2.9999999E-02   2.210000       2.9999999E-02  1.000000

ENTER LIMB CG AND MOM OF INERTIA
  1.000000       0.4640000
```

| XDISP | VEL | ACCL | THETA | ANG V | ANG ACCEL | T |
|---|---|---|---|---|---|---|
| FT | FPS | FPS2 | RAD | RDPSC | RDPS2 | SEC |
| 0.000 | 0.000 | 0.00 | 1.220 | 0.000 | 0 000 | 0.000 |
| 0.003 | 2.091 | 836.42 | 1.233 | 9.991 | 3996.209 | 0.002 |
| 0.010 | 4.142 | 820.33 | 1.270 | 19.720 | 3891 778 | 0.005 |
| 0.023 | 6.178 | 814 39 | 1.331 | 29 531 | 3924.400 | 0.007 |
| 0.041 | 8.199 | 808.28 | 1.418 | 39.482 | 3980.399 | 0.010 |
| 0.064 | 10.210 | 804.76 | 1.529 | 49.911 | 4011.813 | 0.013 |
| 0.092 | 12.230 | 807.67 | 1.665 | 59.394 | 3953.111 | 0 015 |
| 0.126 | 14.283 | 821.32 | 1.825 | 68.717 | 3728.996 | 0 018 |
| 0.164 | 16.405 | 848.70 | 2.008 | 76.911 | 3277.652 | 0.020 |
| 0.208 | 18.626 | 888.69 | 2.209 | 83.383 | 2588.864 | 0.022 |

VALUES AT STOP CONTACT

| XDISP | VEL | ACCL | THETA | ANG V | ANG ACCEL | T |
|---|---|---|---|---|---|---|
| FT | FPS | FPS2 | RAD | RDPSC | RDPS2 | SEC |
| 0.177 | 17.093 | 861.09 | 2.070 | 78.916 | 3064.275 | |

| SEAT X FORCE | LIMB MOMENT | AMP FACT |
|---|---|---|
| 12828.056441 | 1433.191528 | 3.015083 |

Figure 33. Load Analysis Program Output

2.2.3.3. Loading Calculations

The computer program described above was used to obtain the loads for use in the ADAM stress analysis. The program calculated the angular accelerations and velocities, the static aerodynamic loading, and the resulting dynamic load amplification factors for the small and large ADAM arms and legs.

The initial position of the limb with respect to the air stream was a critical issue as it affected the loading situation at the end of the motion significantly. Determining the initial angle (zero, as shown in Figure 27) from actual test cases was difficult as each test is unique and involves a different subject reaction. Through a trial and error effort, it was found that a 20-degree angle for the leg and a 70-degree angle for the arm gave the worst case limb loading situations and were used for this analysis. Lower angles produced a slow start which allowed the torso to decelerate longer before rotational motion could build up. Higher starting angles tended to reduce the range over which buildup of rotational velocity could occur.

Once the initial angle was selected, the motion of the limbs in the two worst case directions was tested and the loads were calculated for each situation. The resulting moments at the hinge points are shown in Table 9. For ease of calculations and bookkeeping, the worst case loading for each limb was used throughout the limb design rather than using a different loading situation for each direction of motion. The directions of motion analyzed for limb design were abduction for the leg and flexion/extension for the arm.

As a requirement of the system, the free stream velocity was set at 700 KEAS. The loads and moments were then calculated for each limb and are presented in Table 10. However, with overall dimensional and weight requirements, only a certain level of loading could be reasonably resisted by the system. The small ADAM limb loadings were beyond this upper limit and had to be reduced. The solution in this case was to lower the speed at which limb flail was assumed to occur from the ejection speed to the speed at man-seat separation. This was not an invalid assumption since an operable ejection seat would prevent limb flail until man-seat separation. For the small manikin only, limb flail was assumed to occur at an air speed of 450 KEAS. The resulting loads and amplification factors are also shown in Table 10.

TABLE 9. DIRECTION OF MOTION LOADING COMPARISON

| Limb | Direction of Motion | Total Moment at Pivot Point (Static and Dynamic) |
|---|---|---|
| Small Arm | Flexion/Extension | 11,600 in-lb |
| Small Arm | Abduction | 10,900 in-lb |
| Large Arm | Flexion/Extension | 51,900 in-lb |
| Large Arm | Abduction | 48,800 in-lb |
| Small Leg | Flexion | 3,600 in-lb |
| Small Leg | Abduction | 21,900 in-lb |
| Large Leg | Flexion | 16,000 in-lb |
| Large Leg | Abduction | 95,500 in-lb |

TABLE 10. ADAM LIMB DYNAMIC AND AERODYNAMIC LOADING

| Limb | Flail Air Speed (KEAS) | Aero Shear At Pivot* (lb) | Aero Moment At Pivot* (in-lb) | Dynamic Amplification Factor | Total Shear At Pivot* (lb) | Total Moment At Pivot* (in-lb) |
|---|---|---|---|---|---|---|
| Small Arm | 700 | 1258.7 | 11,369 | 2.98 | 3750.9 | 33,878 |
| Small Leg | 700 | 2272.7 | 33,779 | 1.90 | 4318.1 | 64,180 |
| Small Arm | 450 | 437.2 | 3,965 | 2.93 | 1281.0 | 11,618 |
| Small Leg | 450 | 784.7 | 11,752 | 1.86 | 1459.5 | 21,858 |
| Large Arm | 700 | 1879.6 | 17,223 | 3.02 | 5676.4 | 52,014 |
| Large Leg | 700 | 2990.5 | 49,694 | 1.92 | 5741.8 | 95,413 |

*The pivot point is the shoulder center of rotation for the arm and the hip center of rotation for the leg.

### 2.2.3.4. Inertial Loadings

The ADAM requirement defines several different levels of inertial loading. The worst of these loadings is the 45 G requirement; thus, this value was used in determining the inertial loads. The loadings that have been calculated based on the inertial acceleration fall essentially into three categories: inertial loading on limb segments, inertial loading in the x and y directions on the torso elements, and inertial loading in the z direction on torso elements. These calculations are defined in the following sub. ·ctions. The weights used in these calculations are given in Table 11.

TABLE 11. ADAM SEGMENT WEIGHTS (POUNDS)

| No. | Segment | Small | Midsize | Large |
|---|---|---|---|---|
| 1 | Head | 9.2 | 9.4 | 9.8 |
| 2 | Neck | 2.0 | 2.3 | 2.8 |
| 3 | Thorax * | 39.6 | 54.9 | 66.2 |
| 4 | Abdomen * | 4.2 | 5.3 | 6.4 |
| 5 | Pelvis * | 19.5 | 26.0 | 32.6 |
| 6 | Right Upper Arm | 3.4 | 4.4 | 5.4 |
| 7 | Right Forearm | 2.5 | 3.0 | 3.7 |
| 8 | Right Hand | 1.0 | 1.1 | 1.3 |
| 9 | Left Upper Arm | 3.4 | 4.4 | 5.4 |
| 10 | Left Forearm | 2.5 | 3.0 | 3.7 |
| 11 | Left Hand | 1.0 | 1.1 | 1.3 |
| 12 | Right Thigh | 17.1 | 21.7 | 25.9 |
| 13 | Right Calf | 6.8 | 8.5 | 10.0 |
| 14 | Right Foot | 1.7 | 2.1 | 2.5 |
| 15 | Left Thigh | 17.1 | 21.7 | 25.9 |
| 16 | Left Calf | 6.8 | 8.5 | 10.0 |
| 17 | Left Foot | 1.7 | 2.1 | 2.5 |
| | | 139.5 | 179.5 | 215.4 |
| *Torso | | 63.3 | 86.2 | 105.2 |

2.2.3.4.1. Inertial Loading on Limb Segments

The inertial loading on the limb segments depend on the segment center of gravity locations from the main pivot points and the segment weights. This data as well as the loadings on the leg and arm segments, are given in Table 12. The joint forces resulting from a 45 G acceleration in the x or z direction and the moments at the pivot point produced by these forces are also listed in this table. The forces were calculated using the relationship:

$$F_{inertial} = W/G\ 45\ G = 45\ W$$

where

W = The segment weight (lbs)

G = The acceleration due to gravity ($ft/sec^2$)

TABLE 12. ADAM LIMB INERTIAL LOADING COMPARED TO THE TOTAL AERODYNAMIC LOADING

| Segment | Weight (pounds) | CG from Shoulder/Hip (inches) | 45 G Inertial Loading (pounds) | Shoulder/Hip Inertial Moment (in-lb) | Total Static and Dynamic Moment (in-lb) |
|---|---|---|---|---|---|
| Small | | | | | |
| Upper Arm | 3.4 | 5.0 | 153.0 | 765.0 | |
| Lower Arm | 2.5 | 14.5 | 112.5 | 1631.3 | |
| Hand | 1.0 | 21.1 | 45.0 | 949.5 | |
| TOTAL | | | 310.5 | 3345.8 | 11,618 |
| Upper Leg | 17.1 | 7.5 | 769.5 | 5771.3 | |
| Lower Leg | 6.8 | 22.9 | 306.0 | 7007.4 | |
| Foot | 1.7 | 32.5 | 76.5 | 2486.3 | |
| TOTAL | | | 1152.0 | 15265.0 | 21,858 |
| Large | | | | | |
| Upper Arm | 5.4 | 6.0 | 243.0 | 1458.0 | |
| Lower Arm | 3.7 | 16.6 | 166.5 | 2763.9 | |
| Hand | 1.3 | 23.6 | 58.5 | 1380.6 | |
| TOTAL | | | 468.0 | 5602.5 | 52,014 |
| Upper Leg | 25.9 | 8.5 | 1165.5 | 9906.8 | |
| Lower Leg | 10.0 | 25.9 | 450.0 | 11655.0 | |
| Foot | 2.5 | 37.2 | 112.5 | 4185.0 | |
| TOTAL | | | 1732.5 | 25747.0 | 95,413 |

The tabulated moment was defined as the inertial force multiplied by the distance from the shoulder or hip joint to the segment center of gravity.

The requirement for withstanding a 45 G loading environment was imposed for sled testing of the manikin as the maximum acceleration seen in an ejection sequence is on the order of 30 Gs. The high level of acceleration in a sled test is not accompanied with the aerodynamic loading seen in an ejection environment. Since these two loadings are not simultaneously experienced by the manikin, the two loading magnitudes were compared and the largest loading was used for the design process. This was a conservative assumption as the two loadings are in opposite directions and should be subtracted. As seen in Table 12, 60 percent of the inertial loads can be significant; however, these were neglected and only the aerodynamic loads were used for the stress analysis.

2.2.3.4.2. Inertial Loading in the X and Y Directions on Internal Torso Elements

Only two significant internal elements fell into this category. These two elements were the spine and viscera mass.

For the viscera, the load was assumed to act through the center of gravity which was assumed to be at the center of the visceral box. The weight of the box is approximately 20 pounds. Acted on by a 45 G acceleration, this produced a 900-pound force in the x and y directions.

For the spine, the estimated weight of 12 pounds was assumed to be equally distributed over the total length of 12.55 inches. Acted on by a 45 G acceleration, this produced an equally distributed load of 43 pounds per inch in the x and y directions.

2.2.3.4.3. Inertial Loading in the Z Direction on Internal Torso Elements

The vertical load of the spinal system (load along the z mechanical axis) was generated by a 45 G acceleration acting on all body segments above the lumbar-pelvis joint. The total weight of these segments is 74.7 pounds for the small manikin and 106.5 pounds for the large. When subjected to the 45 G vertical acceleration, the resulting inertial loads were 3362 and 4793 pounds, respectively.

The visceral mass of 20 pounds produced a vertical load of 900 pounds in the 45 G environment.

These loads were used to perform the stress analysis of internal components.

2.2.3.5. Results

The results from the loading analysis which include the maximum shear forces and moments about each pivot point are shown in Table 13. As noted in Section 2.2.3.3, these loads are the maximum loads developed in any one direction and were assumed to be acting in all directions for the stress analysis. The primary source of design loading was found to be the steady state aerodynamic loading; however, the steady state loading was corrected for dynamic amplification to produce realistic design loads.

TABLE 13. JOINT LOADING VALUES

| | | Loading | |
|---|---|---|---|
| Size | Point of Application | Shear (lb) | Moment (in-lb) |
| Small | Shoulder COR* | 1281.0 | 11618.0 |
| | Elbow COR | 432.9 | 2192.0 |
| | Wrist COR | 0.9 | 1.3 |
| | Hip COR | 1459.5 | 21,858.0 |
| | Knee COR | 574.3 | 5384.0 |
| | Ankle COR | 73.6 | 97.2 |
| Large | Shoulder COR | 5667.0 | 51,924.0 |
| | Elbow COR | 1558.1 | 9152.0 |
| | Wrist COR | 2.3 | 3.5 |
| | Hip COR | 5741.8 | 95,413.0 |
| | Knee COR | 2211.7 | 20,928.0 |
| | Ankle COR | 54.7 | 76.6 |

*COR = Center of Rotation

## 2.2.4. Stress Analysis

### 2.2.4.1. Introduction

The purpose of the ADAM detail design stress analysis was two fold: first, to provide design guidance and to size mechanical/structural components; and second, to assure structural integrity of the manikin system. In performing this analysis, all possible combinations of loading were not considered. Instead, the analysis focused on maximum projected loads and critical sections in order to provide the data necessary to define the parts in the most efficient manner.

### 2.2.4.2. Analysis Technique

#### 2.2.4.2.1. Stress Calculations

To assure the integrity of the manikin, stress calculations were performed on each section of the mechanical system which showed a change in cross section. The calculations were performed using standard stress equations and the loads developed in the loading section of this report. Results from the stress analysis combined with the other requirements of the system were used to design the mechanical elements.

Standard calculations based on various relationships were used in performing the structural analysis. This analysis consisted of the detailed analyses of each element considered critical based on engineering practice dictating that a change in cross section creates a change in stress. Bending stress was calculated using the following equation based on beam theory:

$$\sigma_B = \frac{Mc}{I} k$$

where

$\sigma_B$ = Bending Stress (psi)
M = Applied Moment (in-lb)
c = Distance from the Neutral Axis to the Outermost Surface (in)
I = Cross Sectional Moment of Inertia ($in^4$)
k = Stress Concentration Factor

Shear stress was calculated using the simple relationship:

$$\tau_S = \frac{V}{A} k$$

where

$\tau_S$ = Shear Stress(psi)

V = Shear Load (lb)

A = Section Shear Area (in$^2$)

The axial stresses in components under pure tension or compression were calculated using:

$$\sigma_A = \frac{P}{A} k$$

where

$\sigma_A$ = Axial Stress (psi)

P = Applied Load (lb)

A = Cross Sectional Area (in$^2$)

The torsional shear stress developed in a circular section is defined by the relationship:

$$\tau_{T_c} = \frac{Tr}{J}$$

where

$\tau_{T_c}$ = Torsional Shear Stress (psi)

T = Applied Torque (in-lb)

r = Radius to the Outer Surface (in)

J = Polar Area Moment of Inertia (in$^4$)

$$J = \frac{\pi}{32} \left(D^4 - Di^4\right)$$

where

D = Outer Diameter (in)

Di = Inner Diameter (in)

and the torsional shear stress in a rectangular section was calculated using the relationship found in Shigley and Mitchell (1983) which is:

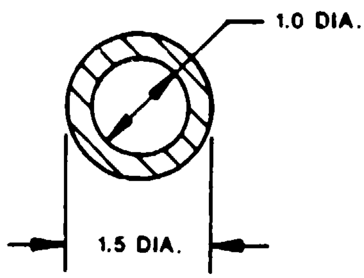
$$\tau_{T_R} = \frac{T}{wt^2}(3 + 1.8\ t/w)$$

where

$\tau_{T_R}$ = Torsional Shear Stress (psi)
T = Applied Torque (in lb)
w = Section Width (in)
t = Section Thickness (in)

As an example of the five types of stress calculations, Figure 34 shows the small upper leg with five areas of interest noted.

For Area 1, the bending stress is:



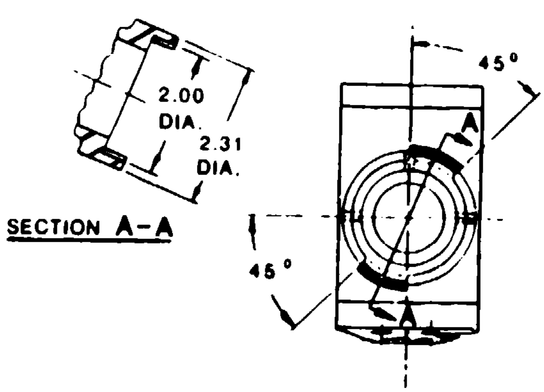
AREA 1

$$\sigma_B = \frac{Mc}{I}k$$

$$= \frac{(30{,}000)\ (.75)\ (1)}{(1.5^4 - 1.0^4)\ \frac{\pi}{64}}$$

$$\sigma_B = 90.4\ \text{ksi}$$

The shearing stress in Area 2 is calculated by the following:



AREA 2

$$\tau_S = \frac{V}{A}k$$

$$= \frac{6841(2)}{\frac{\pi}{4}\left(\frac{45}{360}\right)(2.31^2 - 2.00^2)}$$

$$\tau_S = 104\ \text{ksi}$$

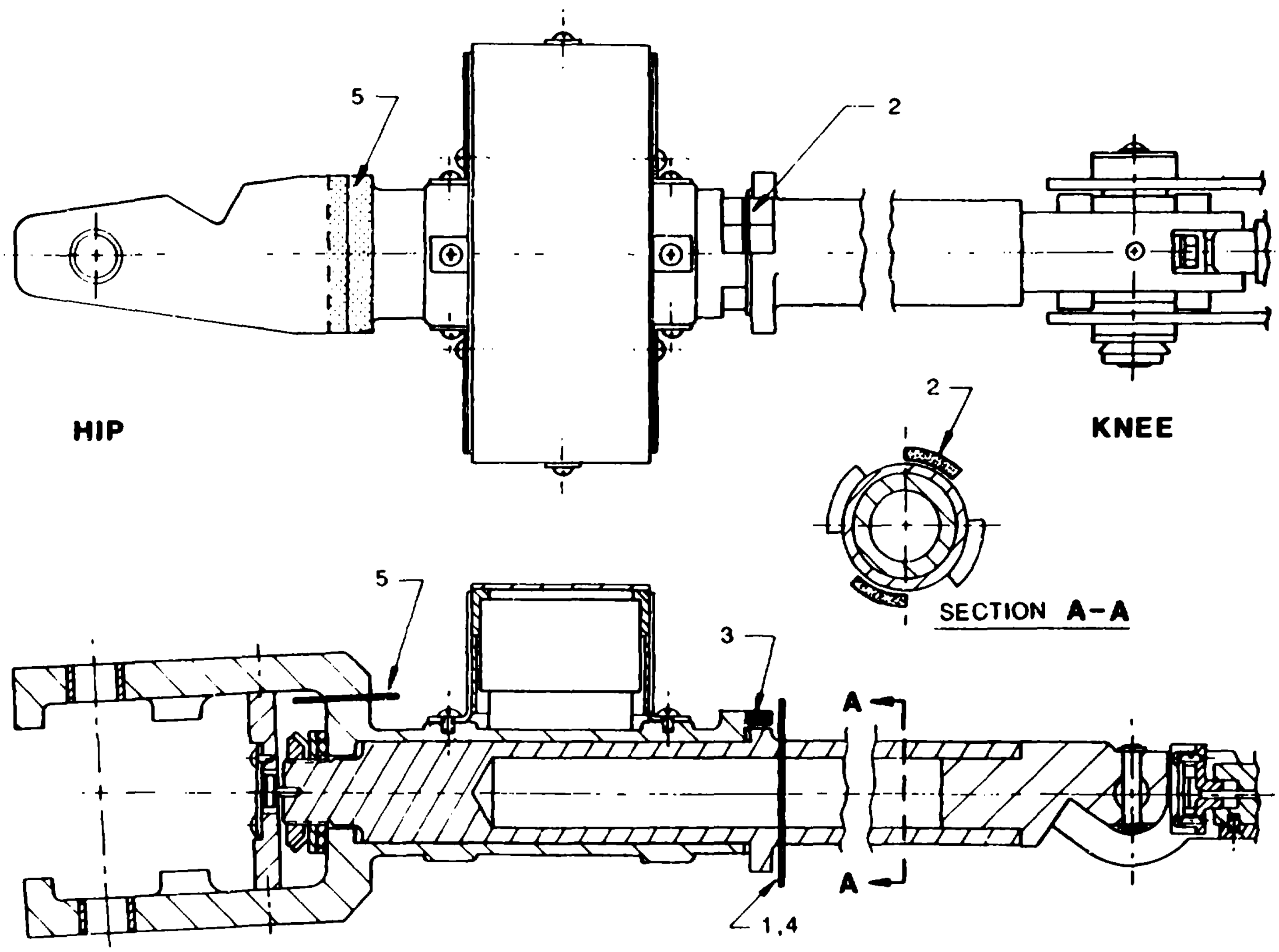
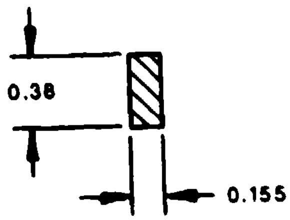



Figure 34. Small ADAM Upper Leg--Five Types of Stress Calculations

At the Area 3, the axial stress is:



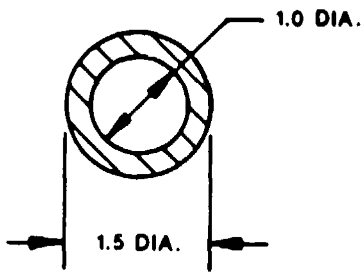

AREA 3

$$\sigma_A = \frac{P}{A} k$$

$$= \frac{6841\,(1)}{(.155)(.38)}$$

$$\sigma_A = 116 \text{ ksi}$$

The torsional shear stress in Area 4 is defined as:



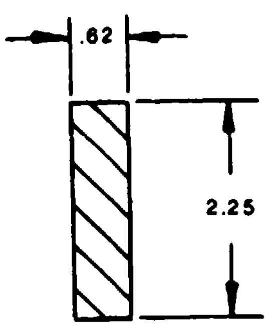

AREA 4

$$\tau_{Tc} = \frac{Tr}{J}$$

$$= \frac{(3708)\,(.75)}{\frac{\pi}{32}\left(1.5^4 - 1.0^4\right)}$$

$$\tau_{Tc} = 6973 \text{ psi}$$

At Area 5, the torsional shear stress is:




AREA 5

$$\tau_{T_R} = \frac{T}{wt^2}\left(3 + 1.8\,\frac{t}{w}\right)$$

$$= \frac{25000}{(2.25)\,(.62)^2}\left[3 + 1.8\,\frac{(.62)}{2.25}\right]$$

$$\tau_{T_R} = 101 \text{ ksi}$$

The loads used in the above examples are described in the loading section of this report.

### 2.2.4.2.2. Stress Concentration Factors

The stress concentration factors were selected based on the specific characteristics of the section under consideration and, therefore, were different for each section. The factors throughout the manikin varied in magnitude from 1.69 to 3.31 and were generally defined in Roark (1954). Figure 35 shows an area with a stress concentration factor of 3.31 and the calculations used to determine it. These factors were used to account for the steady state concentration of stresses due to abrupt changes in the material cross section.

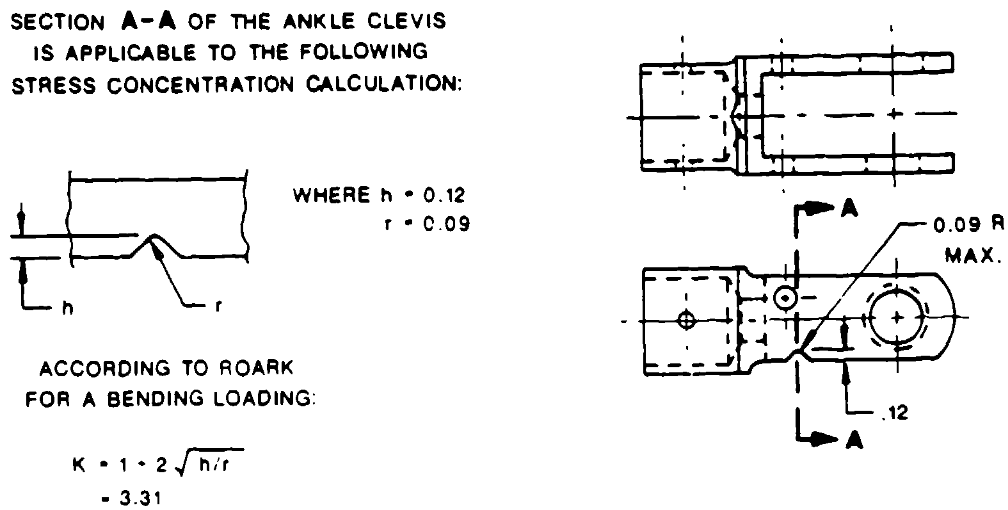



Figure 35. Stress Concentration Factor Calculation

### 2.2.4.2.3. Principal Stresses

Due to the fact that the aerodynamic loading produced a moment along the long axis of the bones, the primary loading was in bending. There was also some loading produced due to the torsion of the limbs and due to the inertial loading on the internal elements. Since the largest loading was generally in one direction, the stresses were dominated by those developed from this loading. The dominant stresses were either bending stress, as in clevis bending or shear stress, as in the joint stops. Based on the above known occurrence in the element sections, it was decided that it was not necessary to calculate principal stress.

2.2.4.2.4. Material Properties

The majority of the manikin components are composed of either aluminum 7075-T6 or 17-4 PH stainless steel, while some components consist of aluminum 6061-T6. The properties of these three materials (MIL-HDBK-5D, 1983) are as follows:

| Material | Tensile Yield Strength | Shear Strength | Ultimate Tensile Strength | Bearing Yield Strength e/D=1.5 | e/D=2.0 |
|---|---|---|---|---|---|
| Alum 7075-T66 | 66 ksi | 46 ksi | 77 ksi | 86 ksi | 92 ksi |
| Steel 17-4 PH | 170 ksi | 123 ksi | 190 ksi | 255 ksi | 280 ksi |
| Alum 6061-T6 | 35 ksi | 27 ksi | 42 ksi | 49 ksi | 56 ksi |

The e/D term for the bearing strength is the ratio of the edge distance to the diameter for the hole being analyzed. If this value is higher or lower than the ones given above, then the yield strength should be used. The tensile strengths listed were assumed to be valid for both tensile and compressive loadings. The strengths of various bolts which were analyzed were defined in MIL-HDBK-5D (1983). The specific bolt MS numbers are listed along with the results of the analysis in Tables 14 through 22.

The margins of safety resulting in the manikin components were determined using the first and second columns of data and the corresponding calculated stresses.

2.2.4.2.5. Design Criteria

The design criteria set by the Air Force for the ADAM systems was that no part or element should fail at its design load. Since the yield point is the point at which the strength of the material begins to decrease from its unloaded state, it was used in this analysis. Another widely used design criteria is the ultimate tensile strength. This is used as it is the point of fracture in the material. However, by using the yield strength, an added conservatism is inherent in the design of ADAM.

As outlined in the loading section of this report, the loads were developed through a fairly conservative process. Combined with the conservatism in the design criteria, it is believed that these two facts should ensure the safety and integrity of the mechanical system under the design loading.

TABLE 14. SMALL SHOULDER/NECK BLOCK--ANALYSIS 450 KEAS LIMB FLAIL SPEED

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Shoulder/Upper Arm Block | | | |
| ABD/ADD Stop Shear | 46 | 1.49 | |
| Shoulder Clevis | | | |
| Stop Shear | 46 | 0.48 | |
| Bending Across Pin | 66 | 0.47 | |
| Bending At Cutout | 66 | 0.38 | |
| Inner Shoulder Connector | | | |
| Bending at Stop Section | 66 | 0.21 | |
| Pronation/Retraction Stop Shear | 46 | 0.15 | |
| Bending at Pin Section | 66 | 0.06 | |
| Rotational Stop Shear | 46 | 0.41 | |
| Bending at Clevis Section | 66 | 0.13 | Bending in a horizontal plane. |
| | 66 | 0.38 | Bending in a vertical plane. |
| Inner Block | | | |
| Stop Shear No. 1 | 46 | 1.74 | |
| Stop Shear No. 2 | 46 | 0.09 | |
| Shear on Stop Pin | 46 | 1.02 | |
| Main Support Pin | | | |
| Axial Stress Due to Pin | 66 | 0.01 | |
| Pin Bearing | 66 | 0.13 | e/D = 1; therefore, the tensile yield strength is used. |
| Elevation/Depression Stop Shear | 465 | 0.06 | |
| Spine Attach Bolts--Normal Stress | 150 | 1.38 | See MS 90727 for allowable stress. |
| Compression Due to Bolt Heads | 66 | 0.16 | |
| Shear of Flange | 46 | 2.50 | |
| Compression Due to Inner Block Stops | 66 | 0.14 | Loads for abduction/ adduction were used. |
| Neck Attach Block | | | |
| Axial Stress in Neck Bolts | 155 | 1.30 | See MS 16997 for allowable stress. |
| Compression of Block Due to Bolts | 66 | 0.89 | |

TABLE 15. SMALL ARM*--ANALYSIS FOR 700 KEAS LIMB FLAIL SPEED (Unless Noted Otherwise)

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Upper Arm | | | |
| Upper Stop Shear | 123 | 0.10 | |
| Lower Stop Shear | 123 | 1.97 | |
| Bending 0.7 inch, Below Shoulder Joint Center | 170 | 0.45 | |
| Bending 1.5 inches, Below Shoulder Joint Center | 170 | 0.21 | |
| Torsion of Transition Section | 123 | 1.51 | 450 KEAS limb flail. |
| Outer Tube Bending | 170 | 1.89 | 450 KEAS limb flail. |
| Inner Tube Bending | 170 | 2.07 | 450 KEAS limb flail. |
| Elbow Inner Piece Bending | | | |
| (X-Axis) | 170 | 0.93 | 450 KEAS limb flail. |
| (Y-Axis) | 170 | 1.88 | |

* The forearm pieces are sized exactly the same for the small and large manikins.

TABLE 16. SMALL PELVIS--ANALYSIS FOR 450 KEAS LIMB FLAIL SPEED

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Main Shaft | | | |
| Bending | 66 | 0 | |
| Side Blocks | | | |
| ABD/ADD Stops | | | |
| Shear No. 1 | 46 | 0.70 | |
| Shear No. 2 | 46 | 0.24 | |
| Center Block | | | |
| Flexion/Extension Stop Shear | 46 | 0.12 | |
| Bending | 66 | 0.50 | Bending in a vertical plane. |
| | 66 | 3.00 | Bending in a horizontal plane. |
| Bearing of Main Shaft | 66 | 0.05 | Since e/D = .8, the tensile yield strength is used. |

TABLE 17. SMALL LEG*--ANALYSIS FOR 700 KEAS LIMB FLAIL (Unless Noted Otherwise)

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Battery Box | | | |
| Side Tab Stress | 123 | 0.61 | |
| Upper Clevis | | | |
| Bending at Hip Joint Center | 170 | 1.33 | 450 KEAS limb flail. |
| Bending at 1.75 inches Below Hip Joint Center | 170 | 1.53 | 450 KEAS limb flail. |
| Torsion at Transition Section | 123 | 0.22 | |
| ABD/ADD Stop Shear | 123 | 1.15 | |
| Bending at Tube Section | 170 | 0.18 | |
| Shear of Rotational Stops | 123 | 3.73 | |
| Thigh Inner Tube Assembly | | | |
| Bending 10 inches Below High Joint Center | 170 | 0.90 | |
| Bending .85 inch Above Knee Joint Center | 170 | 0.44 | |
| Rotational Joint-Upper Leg | | | |
| Stop Shear | 123 | 0.18 | |

* Remaining lower leg (calf) pieces are sized exactly the same for the small and large manikins.

TABLE 18. LARGE SHOULDER/NECK BLOCK--ANALYSIS FOR 700 KEAS LIMB FLAIL

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Shoulder/Upper Arm Block | | | |
| ABD/ADD Stop Shear | 123 | 0.78 | |
| Shoulder Clevis | | | |
| Stop Shear | 123 | 0.00 | |
| Bending Across Pin | 170 | 1.00 | |
| Bending at Cutout | 170 | 0.16 | |
| Inner Shoulder Connector | | | |
| Bending at Stop Section | 170 | 0.22 | |
| Pronation/Retraction Stop Shear | 123 | 0.09 | |
| Bending at Pin Section | 170 | 0.13 | |
| Rotational Stop Shear | 123 | 0.95 | |
| Bending at Clevis Section | 170 | 0.18 | Bending in a horizontal plane. |
| | 170 | 1.74 | Bending in a vertical plane. |
| Inner Block | | | |
| Stop Shear No. 1 | 123 | 1.62 | |
| Stop Shear No. 2 | 123 | 0.17 | |
| Shear on Stop Pin | 123 | 0.41 | |
| Main Support Block | | | |
| Axial Stress Due to Pin | 170 | 0.24 | |
| Pin Bearing | 170 | 0.50 | e/D = 1; therefore, the tensile yield strength is used. |
| Elevation/Depression Stop Shear | 123 | 0.40 | |
| Compression Due to Inner Block Shear | 170 | 0.00 | Loads for abduction/adduction were used. |
| Spine Attach Bolts--Normal Stress | 150 | 0.14 | See MS 90727 for allowable stress. |
| Compression Due to Bolt Heads | 170 | 0.26 | |
| Shear of Flange | 123 | 2.97 | |
| Neck Attach Block | | | |
| Axial Stress in Neck Bolts | 155 | 1.23 | See MS 16997 for allowable stress. |
| Compression of Block Due to Block | 66 | 0.78 | |

TABLE 19. LARGE ARM--ANALYSIS FOR 700 KEAS LIMB FLAIL

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Upper Arm | | | |
| Upper Stop Shear | 123 | 0.41 | |
| Lower Stop Shear | 123 | 1.67 | |
| Bending 1.0 inch, Below Shoulder Joint Center | 170 | 0.47 | |
| Bending 1.5 inches, Below Shoulder Joint Center | 170 | 0.24 | |
| Torsion of Transition Section | 123 | 0.15 | |
| Outer Tube Bending | 170 | 0.40 | |
| Inner Tube Bending | 170 | 0.06 | |
| Elbow Inner Piece Bending | | | |
| (X-Axis) | 170 | 1.07 | |
| (Y-Axis) | 170 | 2.70 | |
| Lower Arm | | | |
| Bending 1.5 inches Below Elbow Joint Center | 170 | 1.18 | |
| Flexion/Extension Stop Shear | 123 | 0.16 | |
| Outer Tube Bending | 170 | 0.38 | |
| Bending at Pin Section | 170 | 0.57 | |
| Torsion at Transition Section | 123 | 0.04 | |
| Inner Tube Bending | 170 | 2.47 | |

TABLE 20. LARGE PELVIS--ANALYSIS FOR 700 KEAS LIMB FLAIL

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Main Shaft | | | |
| Bending | 170 | 0.14 | |
| Side Blocks | | | |
| ABD/ADD Stop | | | |
| Shear No. 1 | 123 | 0.73 | |
| Shear No. 2 | 123 | 0.13 | |
| Center Block | | | |
| Bending | 170 | 0.08 | Bending in a vertical plane. |
| | 170 | 1.70 | Bending in a horizontal plane. |
| Bearing of Main Shaft | 170 | 0.36 | e/D = .8; therefore, the tensile yield strength is used. |

TABLE 21. LARGE LEG--ANALYSIS FOR 700 KEAS LIMB FLAIL

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Battery Box | | | |
| Side Tab Stress | 123 | 0.04 | |
| Upper Clevis | | | |
| Bending at Hip Joint Center | 170 | 0.63 | |
| Bending at 1.95 inches Below Hip Joint Center | 170 | 0.31 | |
| Torsion at Transition Section | 123 | 0.27 | |
| ABD/ADD Stop Section | 170 | 0.21 | |
| Bending at Tube Section | 170 | 0.21 | |
| Shear of Rotational Stops | 123 | 2.32 | |
| Thigh Inner Tube Assembly | | | |
| Bending 2.8 inches Above Knee Joint Center | 170 | 1.80 | |
| Bending .85 inch Above Knee Joint Center | 170 | 0.17 | |
| Shear of Rotational Stops | 123 | 3.92 | |
| Knee Clevis | | | |
| Flexion/Extension Stop Shear | 123 | 0.10 | |
| Clevis Compression Due To Stop Pin | 170 | 1.15 | |
| Bending Across Pin Hole | 170 | 1.70 | |
| Bending at 1.65 inches Below Knee Joint Center | 170 | 1.18 | |
| Bending at Transition Section | 170 | 0.62 | Obtained experimentally. |
| Bending at 2.87 inches Below Knee Joint Center | 170 | 0.53 | |
| Calf Inner Tube Assembly | | | |
| Bending at 6.9 inches Below Knee Joint Center | 170 | 0.48 | |
| Ankle Section | | | |
| Bending at Clevis Clearance Notch | 170 | 0.02 | |
| Shear of Stop Pin | 123 | 0.93 | |

TABLE 22. SMALL AND LARGE SPINE--LOADS BASED ON LARGE MANIKIN

| Segment/Element | Allowable Stress (ksi) | Margin of Safety $(M = \frac{Sy}{\sigma} - 1)$ | Comments |
|---|---|---|---|
| Top Piece | | | |
| Compression of Hole (Inner Ledge) | 170 | 5.2 | |
| Outer Tube | | | |
| Bending | 35 | 1.57 | |
| Axial Loading | 35 | 5.18 | |
| Top Screw | | | |
| Shear | 170 | 4.6 | |
| Piston Rod | | | |
| Axial Stress (at top) | 170 | 2.4 | |
| Inner Cylinder | | | |
| Bending | 66 | 0.17 | |
| Shear Due to Holes | 46 | 1.30 | |
| Spring | | | |
| Compressive Stress | 128 | 0.17 | |
| Lumbar Pin Shear | 123 | 2.62 | |
| Yoke Assembly | | | |
| Axial Stress Due to Pin | 170 | 2.5 | |
| Shear Due to Pin | 123 | 4.77 | |
| Stress on Pin Holes | 170 | .46 | |

### 2.2.4.3. Results

All components were analyzed using the above procedure. If any elements were overstressed, the design was analyzed for changes. The design was altered and the effected elements were analyzed for stress once again. This procedure was repeated until all elements were structurally sound. Since the manikin elements were much more complex than the elements used to define the equations described previously, the analytical results were not entirely accurate. At highly complex areas, assumptions were made to simplify the element creating a conservative analysis. The analysis of one particular element showed that the part would fail under the design load. Since the part was believed to be understressed, based on the results from other manikins, it was tested to determine the actual stresses developed in the element.

The tested element, the large manikin knee clevis, had a margin of safety of -0.17 from the analytical results. Strain gauges were placed in five locations to determine the bending and shearing stresses in the element. The locations of the strain gauges (Figure 36) were determined by analyzing the failure modes of similar parts in other manikins. The moment loading placed on the part was up to 28 percent of the analytical failure moment. The element was held in place using the upper leg bone and the knee clevis stops as the moment was applied at the lower leg rotational stop (Figure 37). The resulting force-stress curves from four different runs are shown in Figure 38. Notice the two distinct slopes on each curve. It is believed that these are developed due to the complexity of the element.

The analytical results are shown in Figure 39 with the data from run number 9. Notice the higher slope developed in the analytical data set. In order to assure conservatism in the results, the higher slope in the experimental data was extrapolated to 1613 pounds, the design load. The stress at this point was 105 ksi which is equivalent to a margin of safety of 0.62. This part was then determined to be understressed.

The torsional stress developed in the manikin elements was analyzed by determining the segment with the maximum torsional stress and performing stress calculations on this segment. It was assumed that, if the highest loaded segment was understressed, then all other segments would also be understressed. The upper leg was determined to develop the maximum torsional stress due to the motion of the lower leg in the 90-degree flexion position. Based on experimental data of the yaw moment of the ejection seat, it was assumed that the seat would eject at a yaw angle of 20 degrees with respect to the airstream. Using the loads developed from this analysis, the torsional stress in the manikin was determined to be much lower than the yield point. The analysis results are shown in Tables 14 through 22 for the small and large manikins, respectively. Included in these tables are the allowable stresses and margins of safety for all areas where the margin of safety is below 6.0. As shown in the tables, the elements of both the small and large manikins will withstand the design loading. The margins of safety listed are inherently conservative as they are based on the material yield strengths. Considering the conservatism in the margins of safety and the design loadings, the ADAM systems should remain intact in an ejection environment.

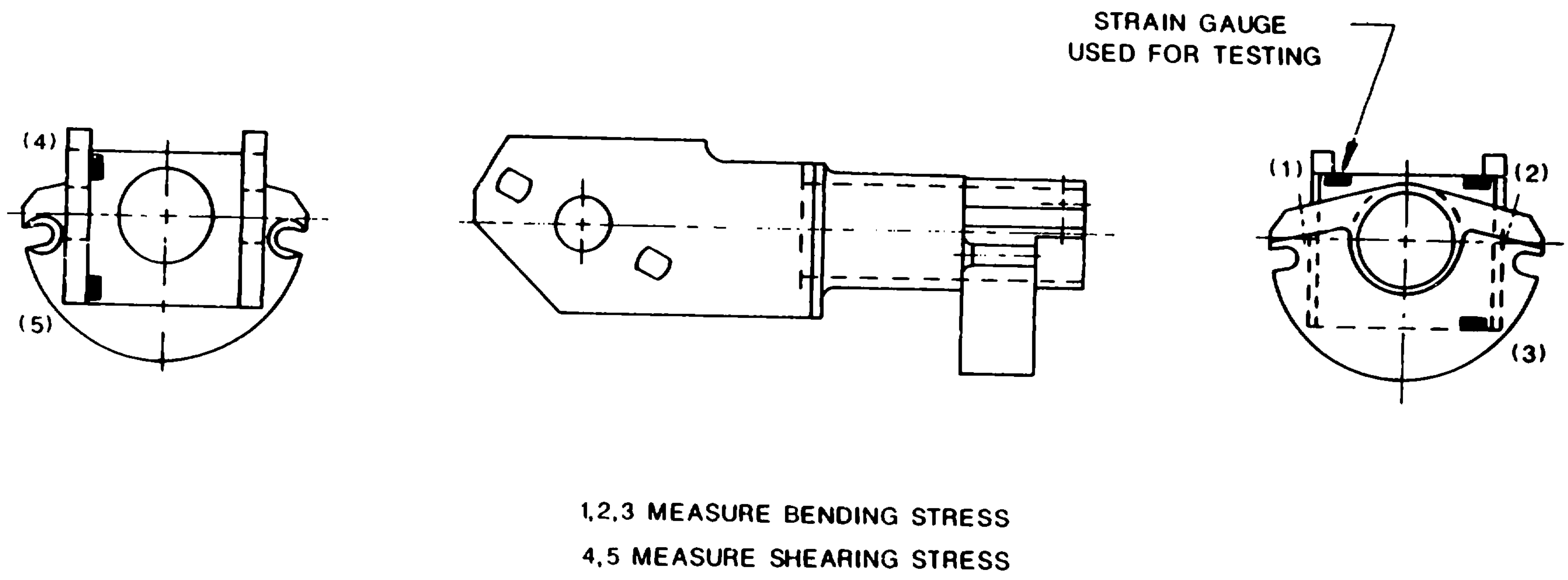



Figure 36. Strain Gauge Configuration No. 2

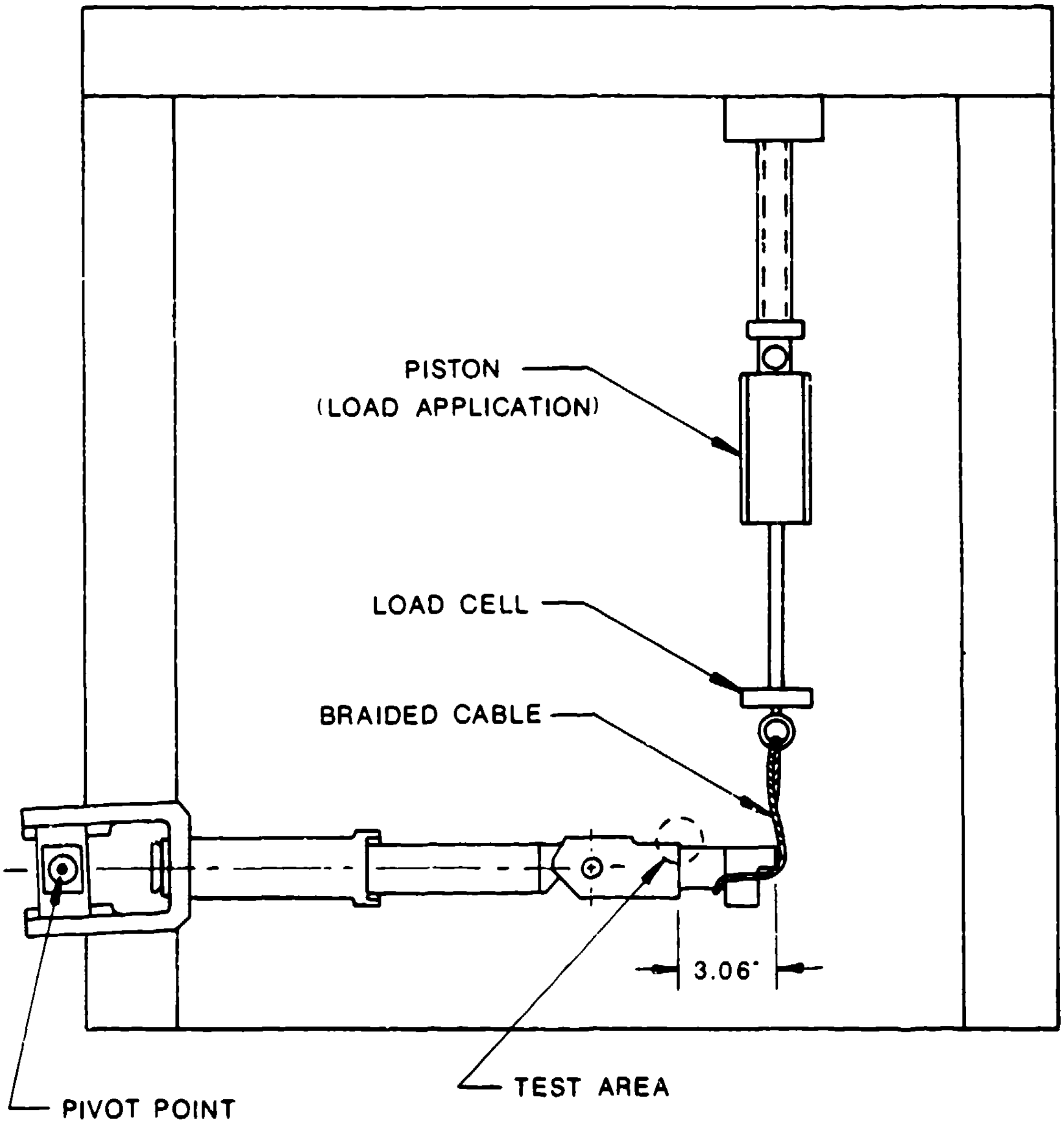



Figure 37. Knee Testing Configuration

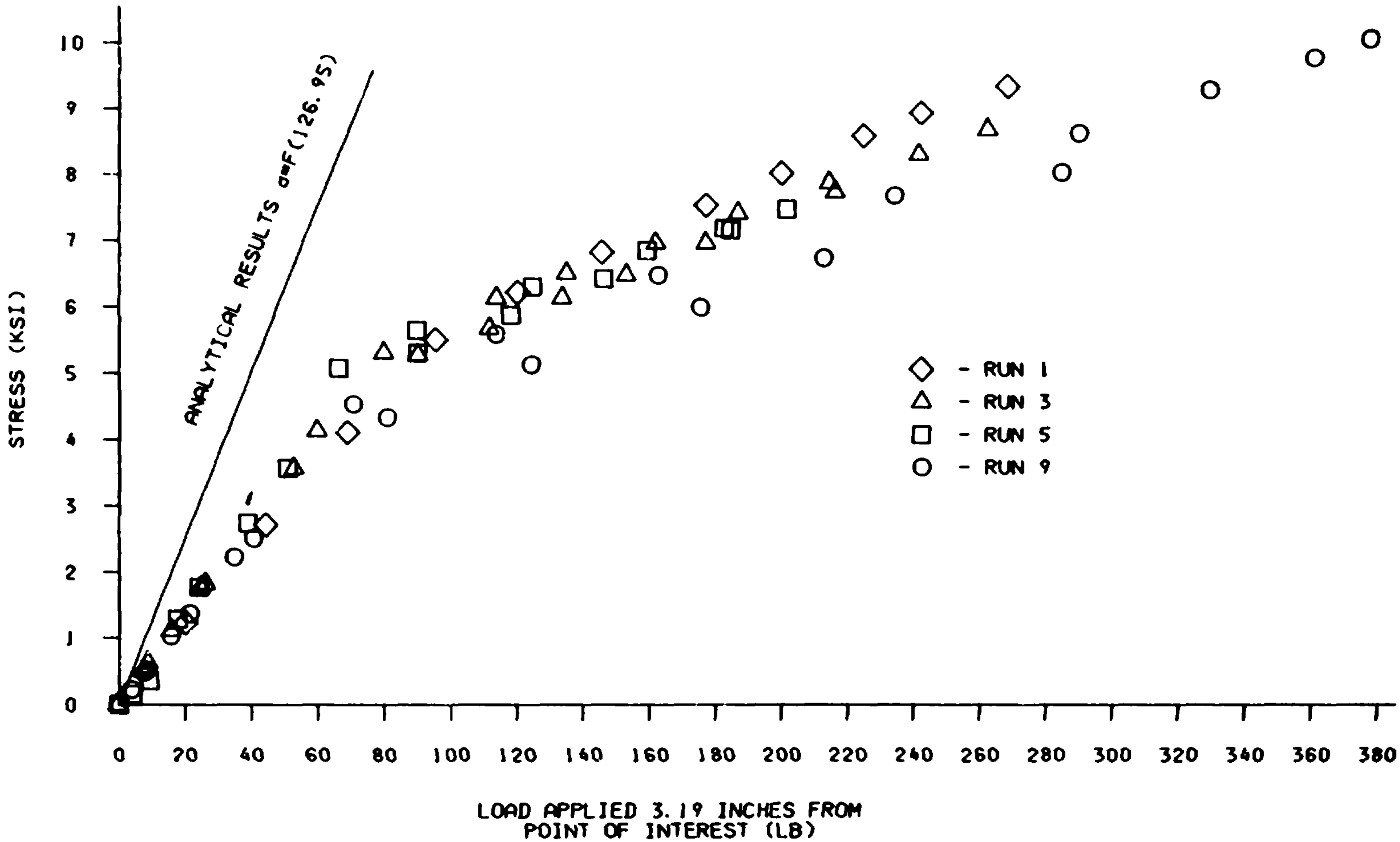



Figure 38. Knee Clevis Bending Stress Versus Load

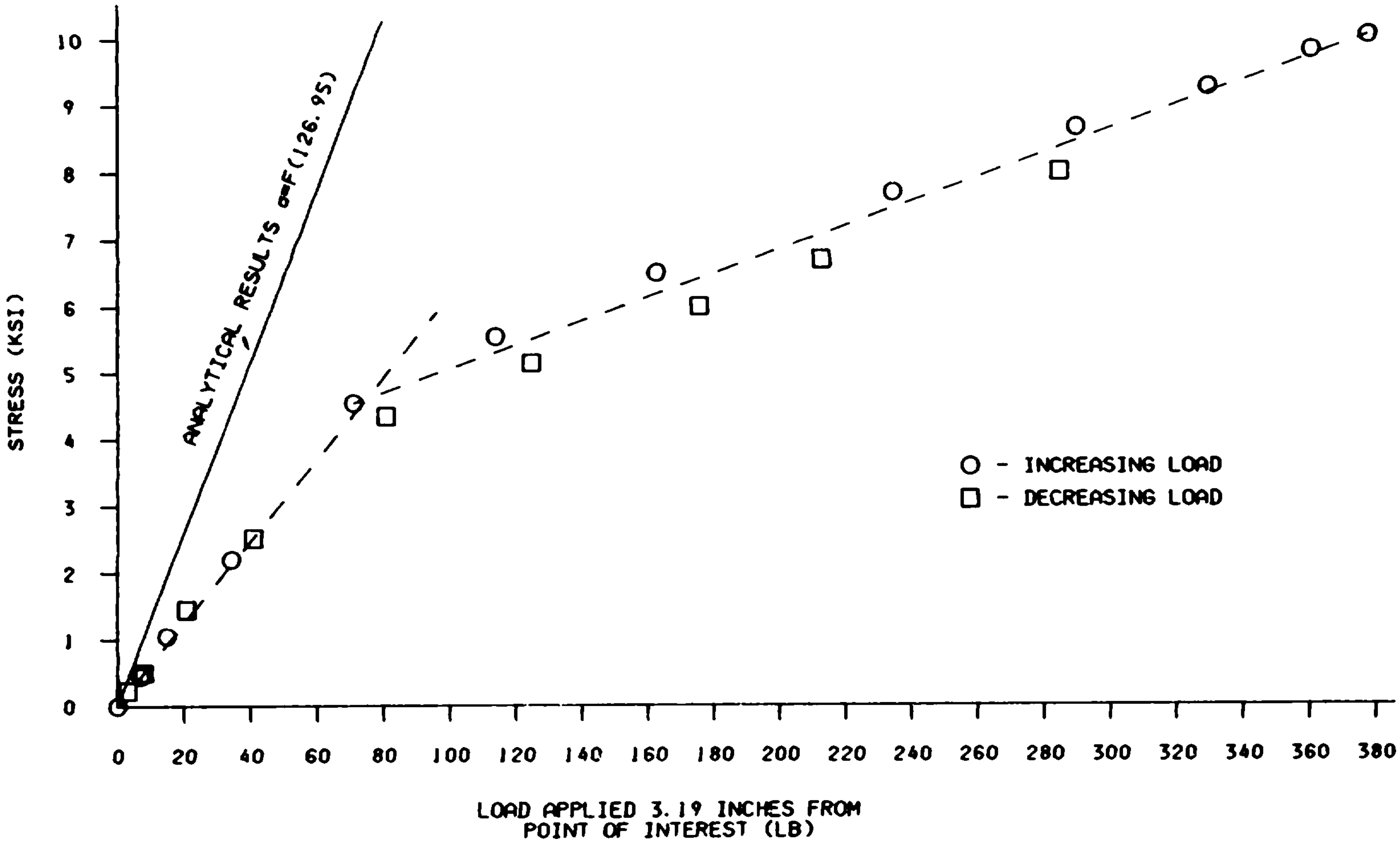



Figure 39. Stress Versus Load--Bending Strain Gauge (No. 2)

## 2.2.5. Spinal System--Research, Analysis, and Design

During an ejection sequence, the human body is subjected to numerous dynamic loadings from the catapult and sustaining rocket, as well as from wind blast. In order to fully challenge the CREST seat, it is important that the ADAM accurately model the human being in terms of dynamic response. Within this section, the development effort that was directed toward creating the final ADAM spinal design will be discussed.

### 2.2.5.1. Design Specifications and Background Data

#### 2.2.5.1.1. Impedance

The basis for determining the similarities of the seated dynamic response characteristics between the ADAM and the human being is the driving point impedance. This basis has been the standard by which the response of the human being to sinusoidal loadings has been evaluated, as it can be determined both experimentally and analytically without solving the complicated coupled equations of motion for the human body for all eigenvalues and eigenvectors (natural frequencies and mode shapes, respectively). The impedance technique also provides a ready means of defining fundamental natural frequencies and the variations of response with changes in mass, stiffness, and damping.

The specifications for the midsized manikin in general called for an impedance curve similar to the mean experimental impedance curves depicted in ISO-5982-1981 and specifically:

> "The normal gravity (0.5 Gz vibration acceleration amplitude) driving point impedance modulus of the seated 50th percentile manikin, in the 0 to 30 Hz frequency domain, should be such that a major peak of 4000(1.0 ± 0.10) Nsec/m occurs at a frequency of 5(1 ± 0.10) Hz, a lesser peak of 2500(1.0 ± 0.10) Nsec/m occurs at a frequency of 10(1.0 ± 0.10) Hz and should reflect an overall damping ratio of 0.30(1.0 ± 0.10)."

The ISO impedance standard was created by evaluating a series of ±1/2 G shake tests performed on a number of test subjects. These results, taken over a driving frequency range of 1 to 30 Hz, were combined to demonstrate an upper and lower boundary impedance curve. These results were then averaged to obtain a mean impedance curve. All three of these curves are depicted in the ISO-5982-1981 standard (Figure 40).

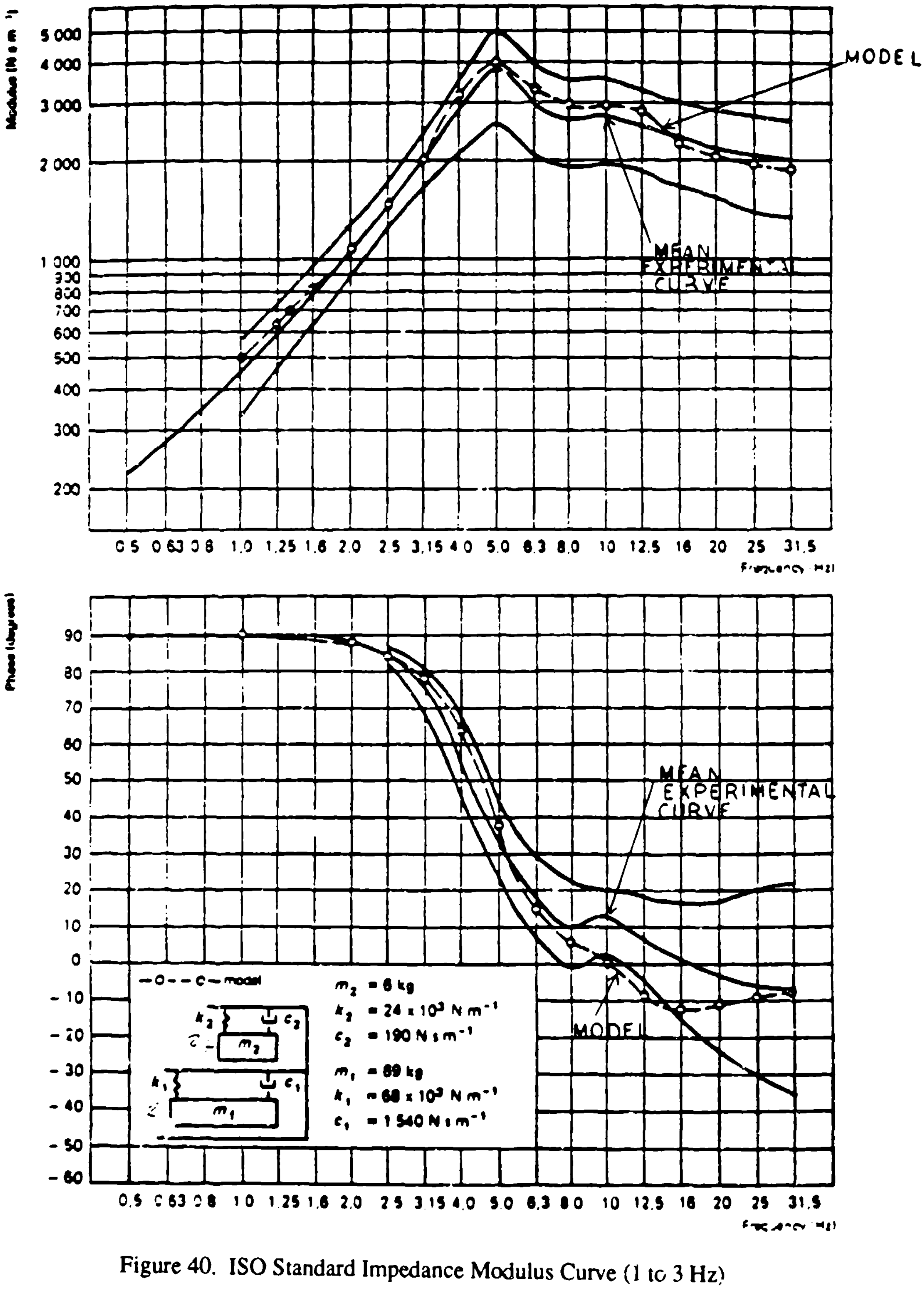



Figure 40. ISO Standard Impedance Modulus Curve (1 to 3 Hz)

The driving point impedance technique was used by Vykukal (1965); Vogt, Coermann, and Fust (1968); and Mertens (1978) to define the relationship between the force applied to the human body through the seat and the velocity of the body at this driving point.

Figure 41 presents Merten's results of the effect of the mean acceleration on the measured impedance characteristics of seated humans in the frequency range of 1 to 20 Hz. As Figure 41 indicates, the primary natural frequency of the body is approximately 5 Hz at a 1-G mean acceleration level and increases to approximately 11 Hz at a 2-G mean acceleration level. Slight increases in natural frequency were noted for larger mean accelerations. The increase in natural frequency with mean acceleration indicates a strong nonlinearity in the body's stiffness which should be accounted for in manikin design.

Figure 42 presents a comparison of the results obtained by various investigators for a 1 G steady loading. The results of two investigators indicate an impedance peak at approximately 5 Hz and a secondary peak in the range of 8 to 11 Hz. Vykukal's results, which were obtained for a human in a semisupine position, indicate that the body position relative to the mean acceleration direction has a significant effect on the response of the human body to sinusoidal dynamic loadings.

2.2.5.1.1.1. System Damping

Within the impedance specifications, a requirement that the entire manikin have a damping ratio of 0.30 (1.0 ± 0.10) was stated. This requirement appears to have originated from work done by Wittmann (1966). In his report, Wittmann used a 1 degree of freedom model consisting of a single mass, spring, and damper to match human test data obtained on a drop tower. His analysis indicated that the best results were obtained when the system was given a natural frequency of 10 Hz and a damping ratio of 0.3. The problem with this requirement is that, if the modeled system is more than a 1 degree of freedom system, the overall damping requirement becomes unclear. While it is possible to determine the damping ratio for each individual coupled mode of the system, no single damping ratio for the entire system exists. For this reason, the absolute requirement for 0.30 overall damping was relaxed.

2.2.5.1.1.2. System Elements Affecting Impedance

It was stated by Privitzer and Belytschko (1980) that the major contributors to the impedance peak at 5 Hz was the elasticity of the buttocks, spine, and viscera. In addition, the ADAM System Specification states that the nonlinear stiffening of the body to increased mean G loading has been attributed to nonlinear behavior of the spine (primarily the intervertebral discs), pelvis, buttocks,

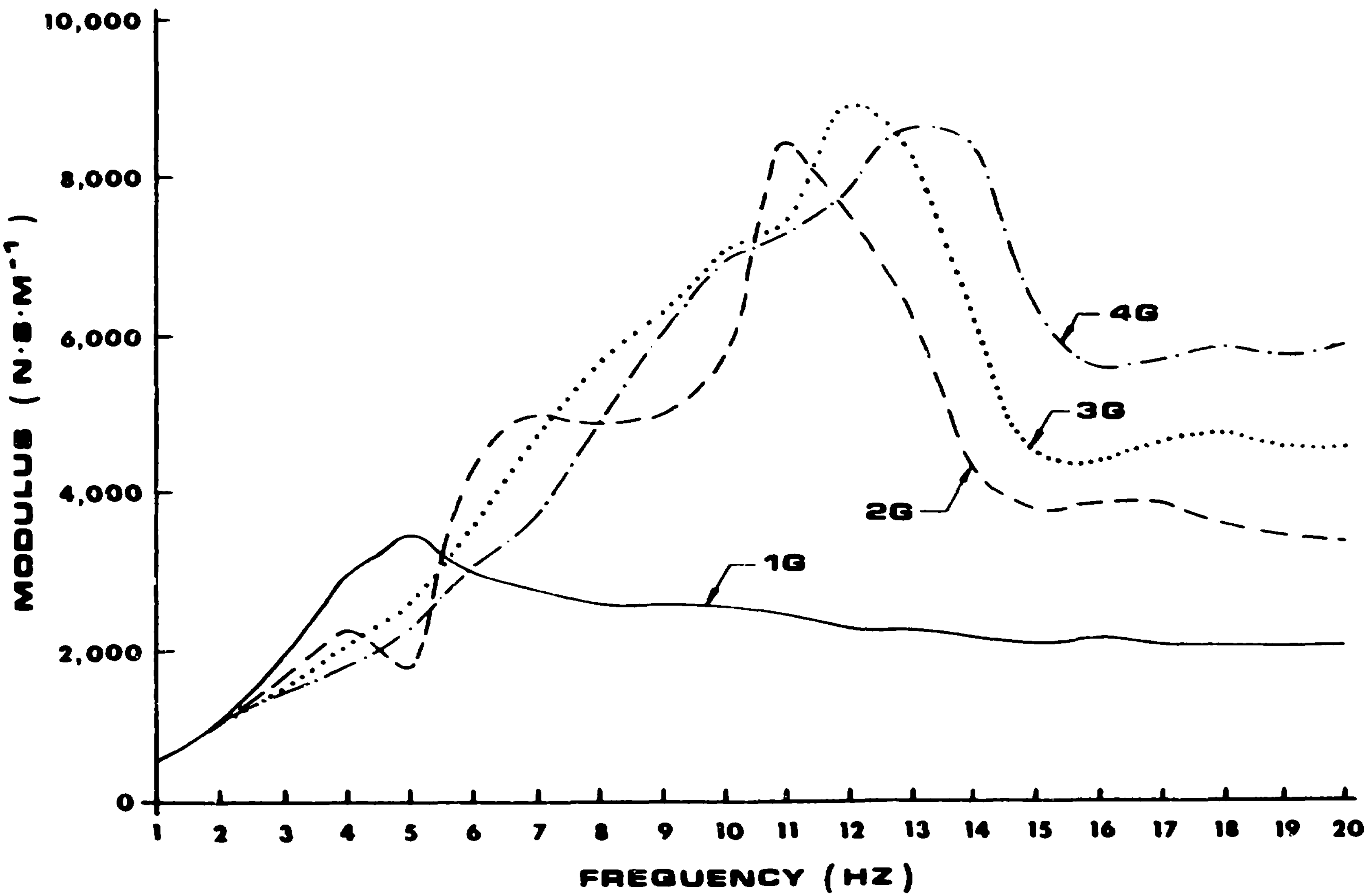



Figure 41. Impedance of the Upright Sitting Human at Various Mean Accelerations

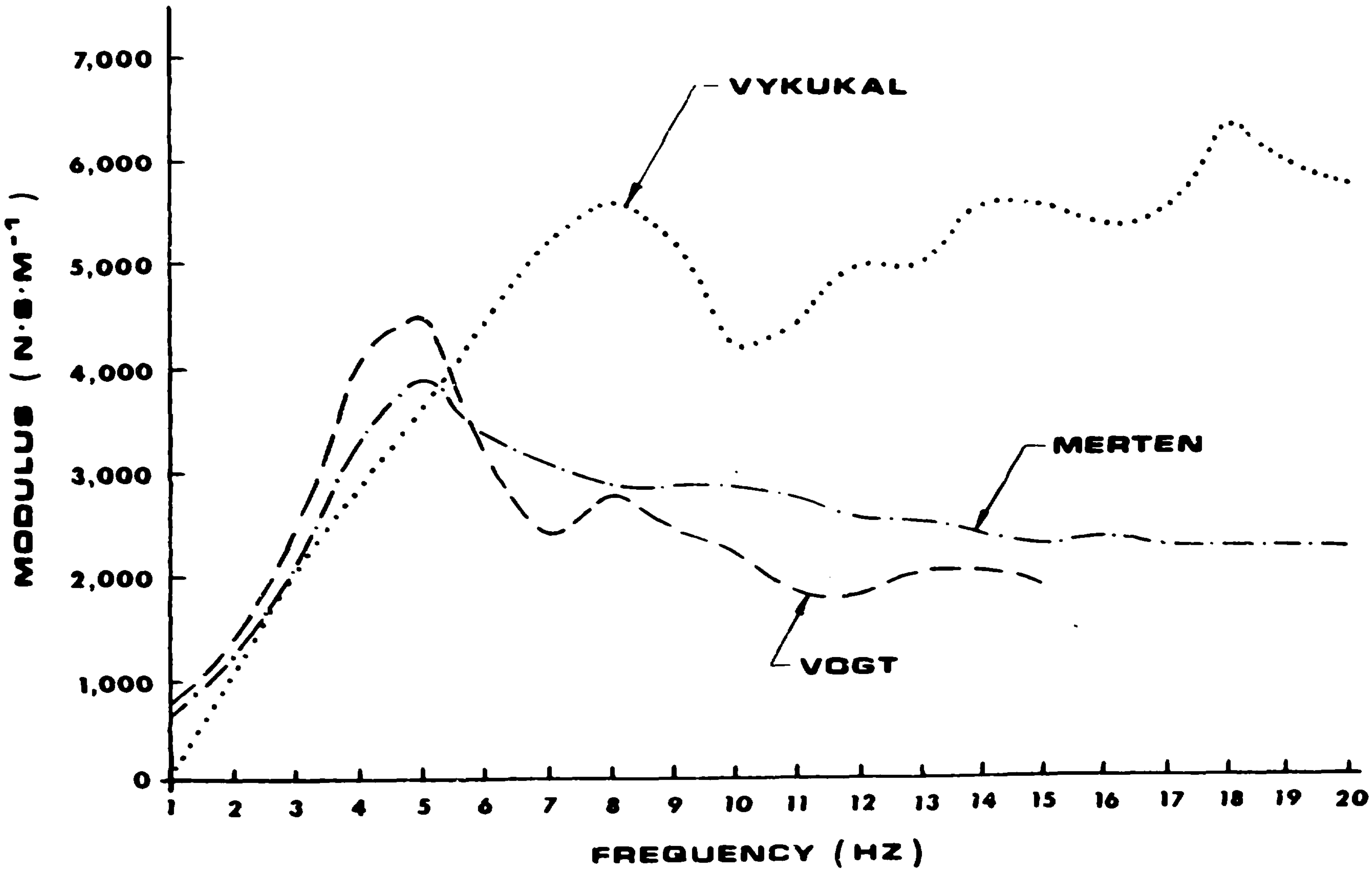



Figure 42. Impedance of the Human Body at 1 G Mean Acceleration

and abdominal viscera. For this reason, the buttocks, spine, and viscera degrees of freedom were modeled into the ADAM prototype spine.

2.2.5.1.1.3. System Level Design Criteria

Based upon an evaluation of the data presented in the referenced reports and definition of the response requirement in the ADAM System Specification, a set of criteria was defined. These criteria, given below, attempted to reduce frequency response system requirements to a tractable set of design objectives.

- Primary impedance modulus peak of 4000 at 1 G and 5 Hz for the entire system.
- Secondary peak of 2500 at 1 G and 10 Hz.
- Major contributors to impedance curve are the buttocks, spine, and viscera.
- Peak frequency shift and impedance increases for increasing mean Gz loading.

2.2.5.1.2. Dynamic Response

Since the primary objective of the ADAM program is the development of a mechanical human surrogate for use in ejection system testing, it is important that the response of the ADAM to impact loadings in the X, Y and Z directions be similar to the response of a human under the same loadings. Works by Ewing and Thomas (1972); Ewing et al. (1977); Cheng et al. (1979); and Begeman, King, and Prosad (1973) were examined. However, these reports tended to focus too tightly on quantitative experimental data to localized impact testing. While these reports were used for their qualitative results, works on the evaluation of the F/FB-111 crew seat and restraint system (Air Force Aerospace Medical Research Laboratory, 1980, 1982, and 1983) were used to aid in an analysis of the ADAM dynamic response to Gz loading.

Quantitative data for approximately 20 Air Force personnel exposed to an ejection seat environment were presented in the F/FB-111 crew seat reports. From these data, the seat input loading to the human that best represented the large and small manikins in terms of height and weight was digitized and applied to an SRL ADAM response model. This model will be described in Section 2.2.5.3.3.

2.2.5.2. Design Evolution

The human spine is divided into three basic parts: the cervical spine or neck, the thoracic spine to which the ribs are mounted, and the lumbar spine or lower back. Each of these sections has been represented in the ADAM, and various design concepts evolved during the design process. Detailed descriptions of this evolution process are presented in the following sections.

#### 2.2.5.2.1. Initial Design Concepts

##### 2.2.5.2.1.1. Cervical Spine

The major motions in the cervical spine are flexion and extension, lateral bending, and yaw rotation. The order of importance in modeling these, as described in the ADAM System Specification, was anterior and lateral bending, torsion, and posterior bending. With this in mind, the Hybrid III neck, which was developed and used extensively by General Motors, was chosen to represent the cervical spine in the ADAM. This choice was made as it was believed that the dynamic characteristics of the Hybrid III neck represented the state of the art and was suitable for use in manikin subjected to the ejection environment. The neck, illustrated in Figure 43, is a one piece, flexible neck comprised of aluminum vertebral plates molded into 75 durometer butyl elastomer. Drilled holes and saw cuts through the anterior side of the neck decrease the extension bending stiffness without affecting the flexion stiffness of the neck. Guidelines for acceptance testing of manikin necks were established by Metrz and Patrick (1971). These guidelines specify moment versus angle corridors that the midsized neck must fall within for defined flexion and extension tests. Reports by Foster, Kortge, and Walinin (1977) indicate that the responses of the Hybrid III neck generally fall within the range of the required response.

##### 2.2.5.2.1.2. Thoracic Spine/Viscera

As in a majority of the previously developed manikins, the human thoracic spine was simulated by a rigid structure in the ADAM. The rigid thoracic spine was chosen for its reliability and maintainability characteristics. Movement of a dynamic viscera in the direction was an objective in the initial concept studies in order to obtain realistic deflections and proper impedance properties for the overall ADAM system.

Three different initial design concepts for the dynamic viscera were explored. Each design concept supported the visceral weight while providing the required nonlinear stiffness and significant damping needed for the ADAM viscera. By incorporating much of the instrumentation into the visceral package, a degree of shock and vibration isolation was also provided to the instrumentation system.

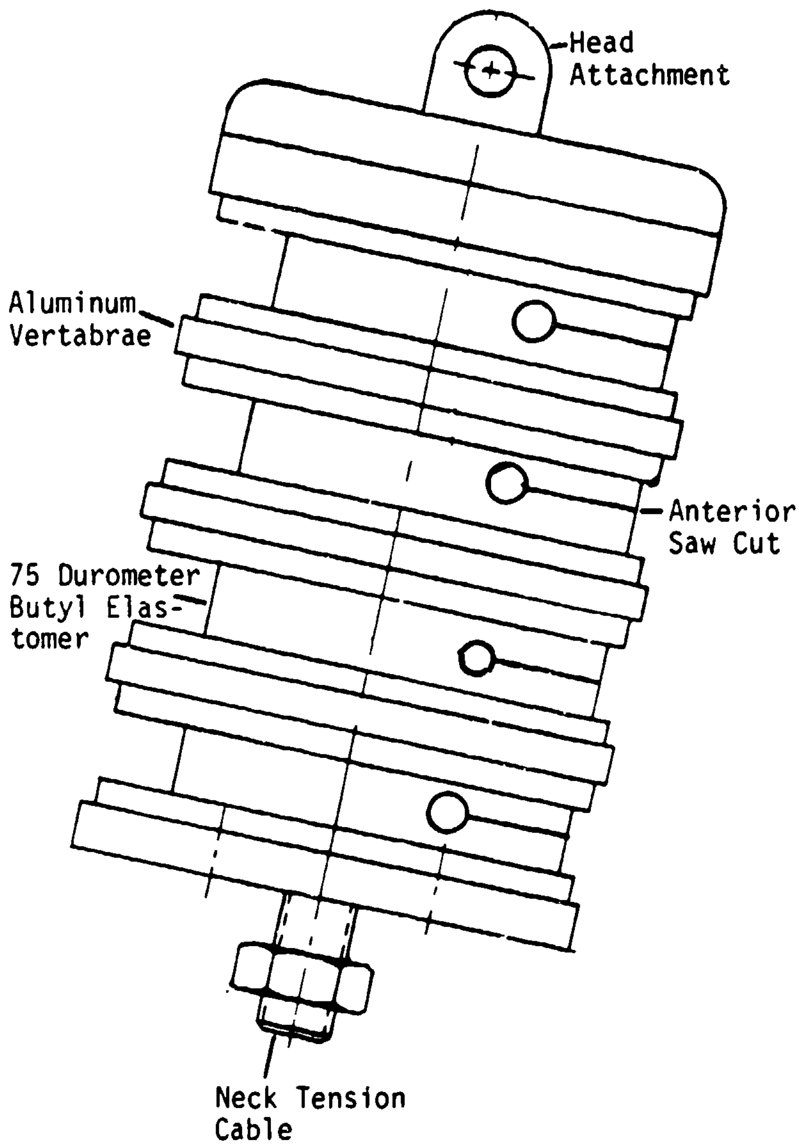



Figure 43. Hybrid III Cervical Spine

The three design concepts initially investigated in depth to obtain the system of required nonlinear springs and dampers in order provide sensitivity to mean G loading are presented in Figures 44 to 47. The concept presented in Figures 44 and 45 uses a pretensioned cable to supply the required nonlinear spring constant to the rod holding the viscera instrumentation weight. Figure 45 presents the equation describing the effective spring constant as a function of the initial deflection of the cable resulting from the effective G weight of the viscera. Sample results presented in these graphs indicate that the spring constant is highly nonlinear with respect to the initial deflection (mean G loading). System analysis of the concept revealed that, with the proper location of the cable on the viscera support arm, the desired variation of the viscera frequency could be obtained up to a level of 8 Gs.

The second concept investigated was a variable length torsion rod that supports the offset visceral weight. The concept description of this system is illustrated in Figure 46. As noted in Figure 47, the spring constant of a torsion rod is inversely proportional to the length of the rod. This characteristic was used to obtain a variation of the spring constant as a function of the initial torque developed by the effective G weight of the viscera. Figure 47 indicates that a continuous variation of the spring constant cannot be obtained by this system as it was for the braided cable system.

The third initial concept investigated was a pneumatic spring incorporated into the thoracic spine. Figure 48 illustrates that the static visceral weight is counterbalanced by greater pressure in the lower cylinder than in the upper cylinder. The equation that relates the force on the pneumatic cylinder with the deflection the piston travels is as follows:

$$F = \frac{P_L h_L A_L}{(h_L - \delta)} - \frac{P_U h_U A_U}{(h_U + \delta)}$$

where

$P_U, P_L$ = Initial pressure in the upper and lower chambers.
$h_U, h_L$ = Height of the upper and lower chambers under a 1 G condition.
$A_U, A_L$ = Surface area of the upper and lower chamber.
$\delta$ = Deflection that the piston travels under a given loading.
$F$ = Force applied to the piston by the weight of the viscera subjected to a given mean G level.

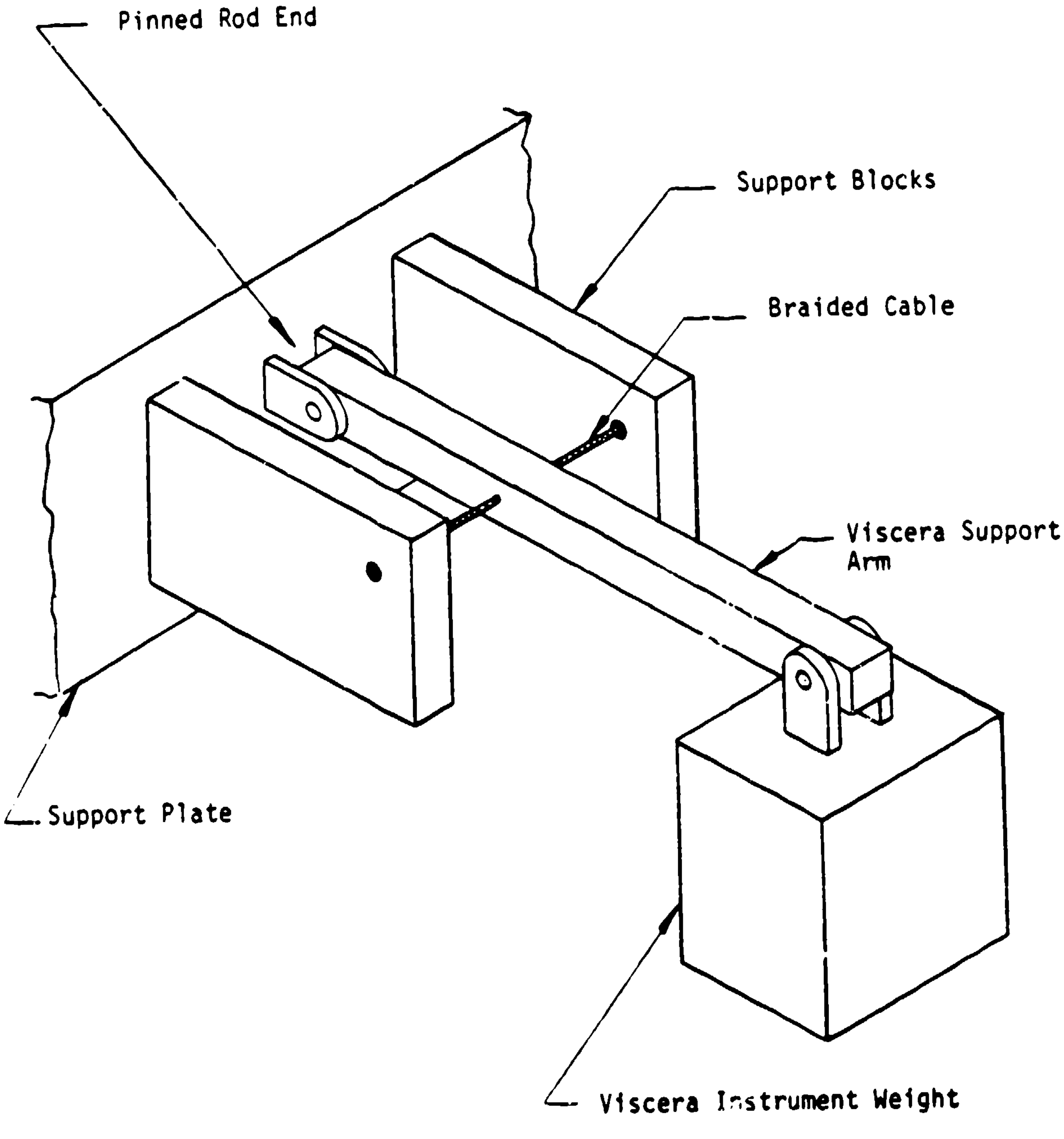
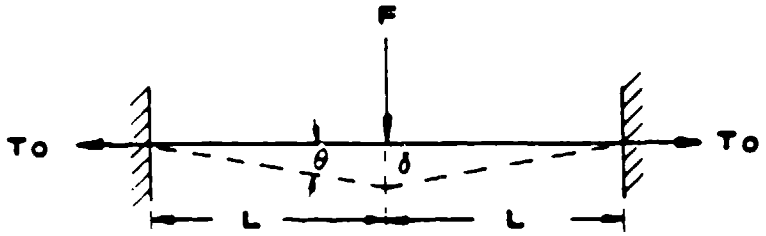



Figure 44. Pretensioned Cable as Nonlinear Spring System of Simulated Viscera




THE SPRING CONSTANT IS GIVEN BY:

$$K_s = 2\left[\frac{T_o}{L}\cos\theta + K_c(1 - \cos\theta)\right]$$

WHERE

$T_o$ : INITIAL TENSION IN CABLE

$K_s$ : SPRING CONSTANT OF SYSTEM

$K_c$ : SPRING CONSTANT OF CABLE IN TENSION

$\delta$ : DEFLECTION OF CABLE AT LOAD POINT IN INCHES

EXAMPLE:

L = 3 INCHES

CABLE : $^1/_8$ INCH CABLE WHERE $K_c \doteq 150 \times 10^3$

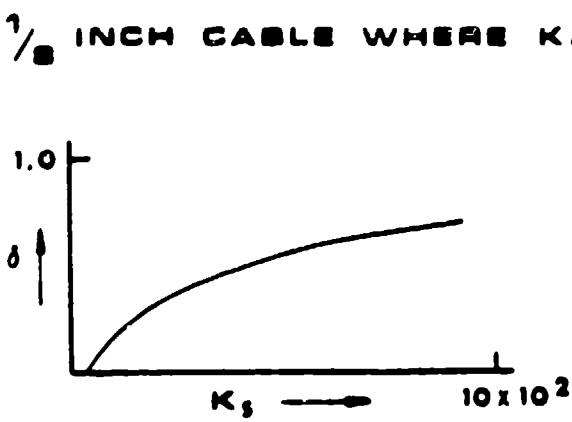



Figure 45. Nonlinear Spring Characteristics of Tension Cable

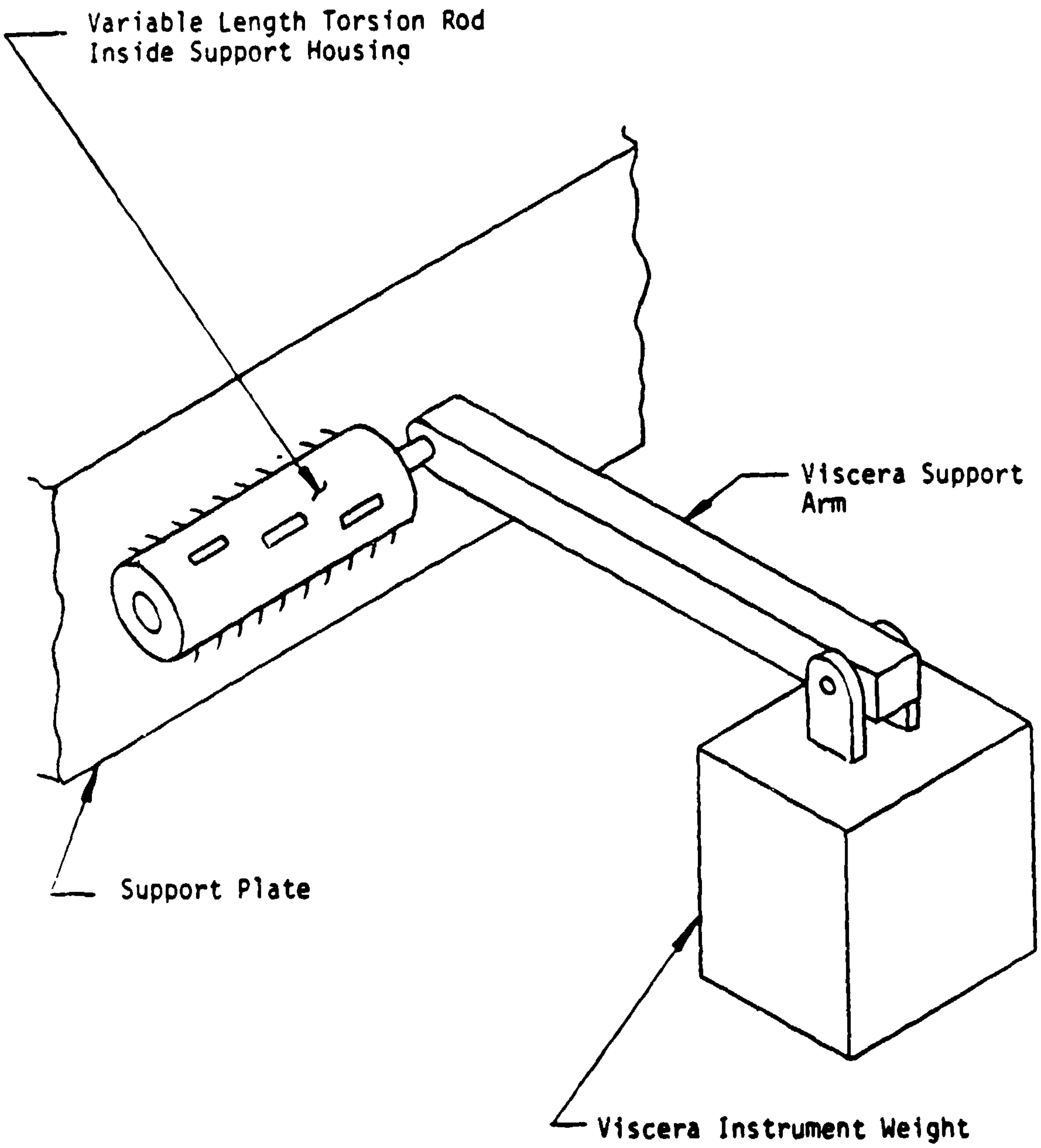



Figure 46. Variable Torsion Rod for the Nonlinear Spring System of Simulated Viscera

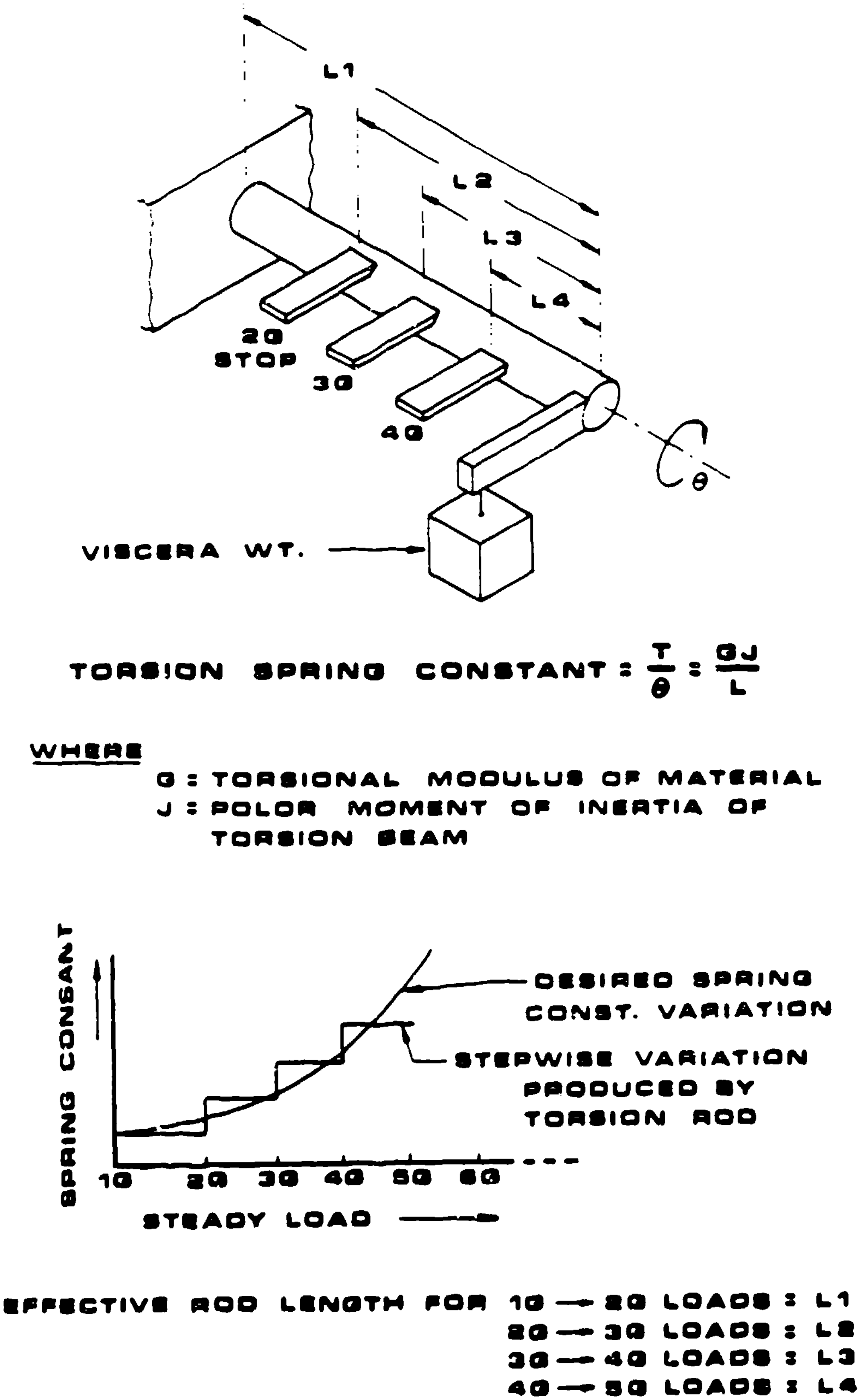



Figure 47. Nonlinear Spring Characteristics of Variable Length Torsion Rod

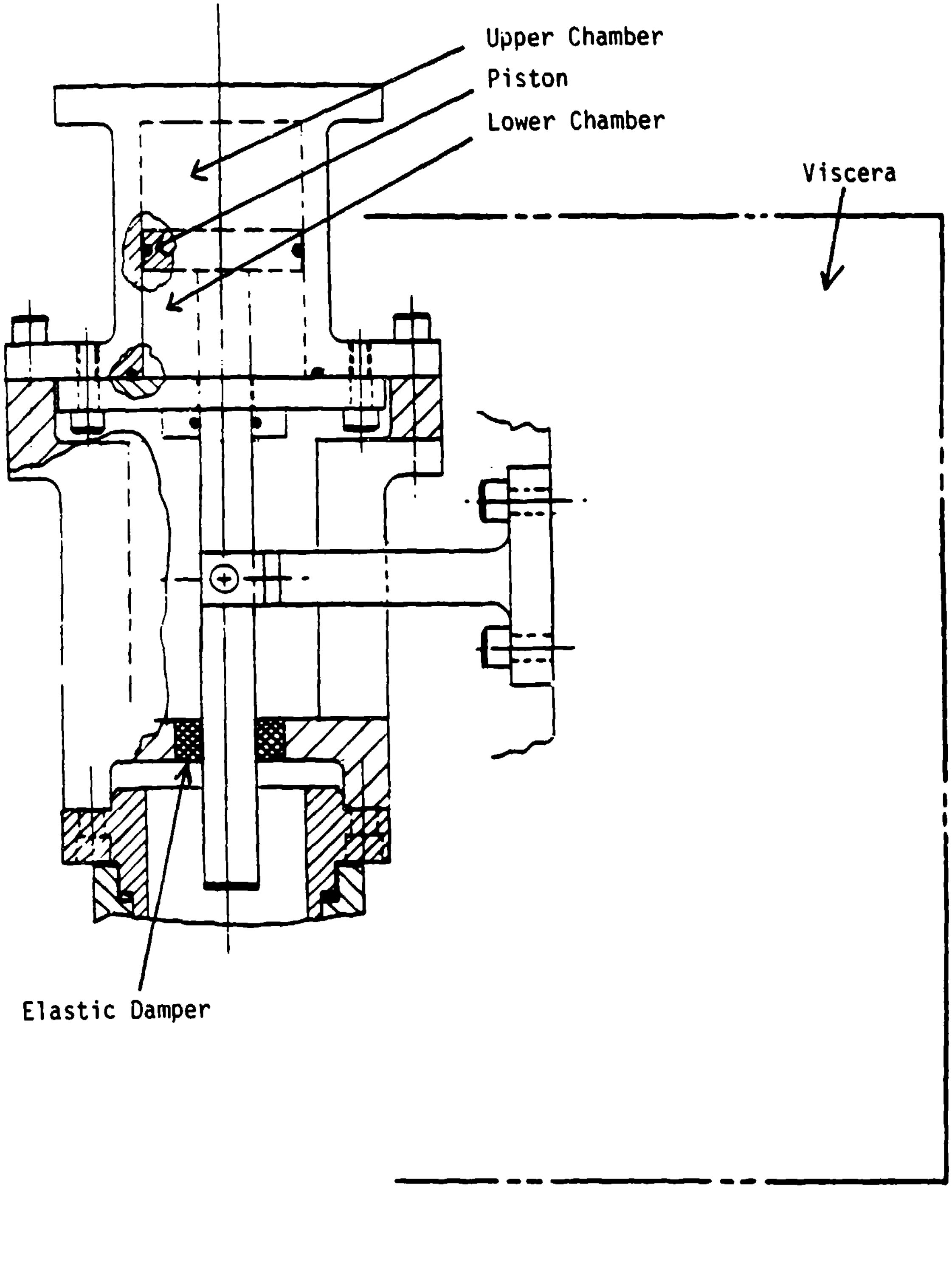



Figure 48. Pneumatic Spring Concept for Simulated Viscera

Comparison of the three initial design concepts determined that the pneumatic spring concept was better suited for the visceral spring than the other concepts for the following reasons:

- Relatively simple and maintenance free.
- Spring constant stiffens continuously with increasing mean G loading.
- System provides a snubbing action under extreme G loadings, eliminating hard stops from the visceral system.
- System allows maximum volume for the instrumentation package.

2.2.5.2.1.3. Lumbar Spine

The primary deflection of the spine in the axial direction occurs in the lumbar spine. Other motions occurring in this area are anterior and lateral bending, torsion, and posterior bending. These motions were incorporated into the lumbar spine since a majority of these motions occur in this region, and there was no ability to provide for these motions in the thoracic spine of the ADAM as it is designed to be rigid.

Evaluations of work done by Beltyschko and Privitzer (1978) to predict human dynamic response to an ejection environment initially revealed that it might be possible to generate a realistic representation of the human spine's pitch and roll motion by placing articulation points at the vertebral levels of L5 and L2. This conclusion was reached after inspecting various predicted spinal deformations and determining the best location for hinge points in the lumbar spine. Also taken into consideration was the location of the instrumentation box, which limits the maximum height at which the highest hinge point can be placed. Figure 49 illustrates the correspondence that can be achieved by using the aforementioned double articulated joint to represent the lumbar spine.

The double articulated joint concept is presented in Figure 50. Figure 50 indicates that the joint had the capability of providing motions in all axes, could be fitted with soft elastomeric stops to provide a more biofidelic nonlinear response to deflection and, most importantly, was believed to realistically represent motions obtained from the human lumbar spine. Other advantages that initially led to this design concept were measurable, repeatable values for rotation, increased strength, and the ability to use friction joints in order to establish essential "breakaway" loadings in the spine.

2.2.5.2.1.4. Buttocks

The interface between the seat and the mechanical analog of the spine in the ADAM is the buttocks. Within the manikin system, the buttocks act as a cushion between the spinal system and the

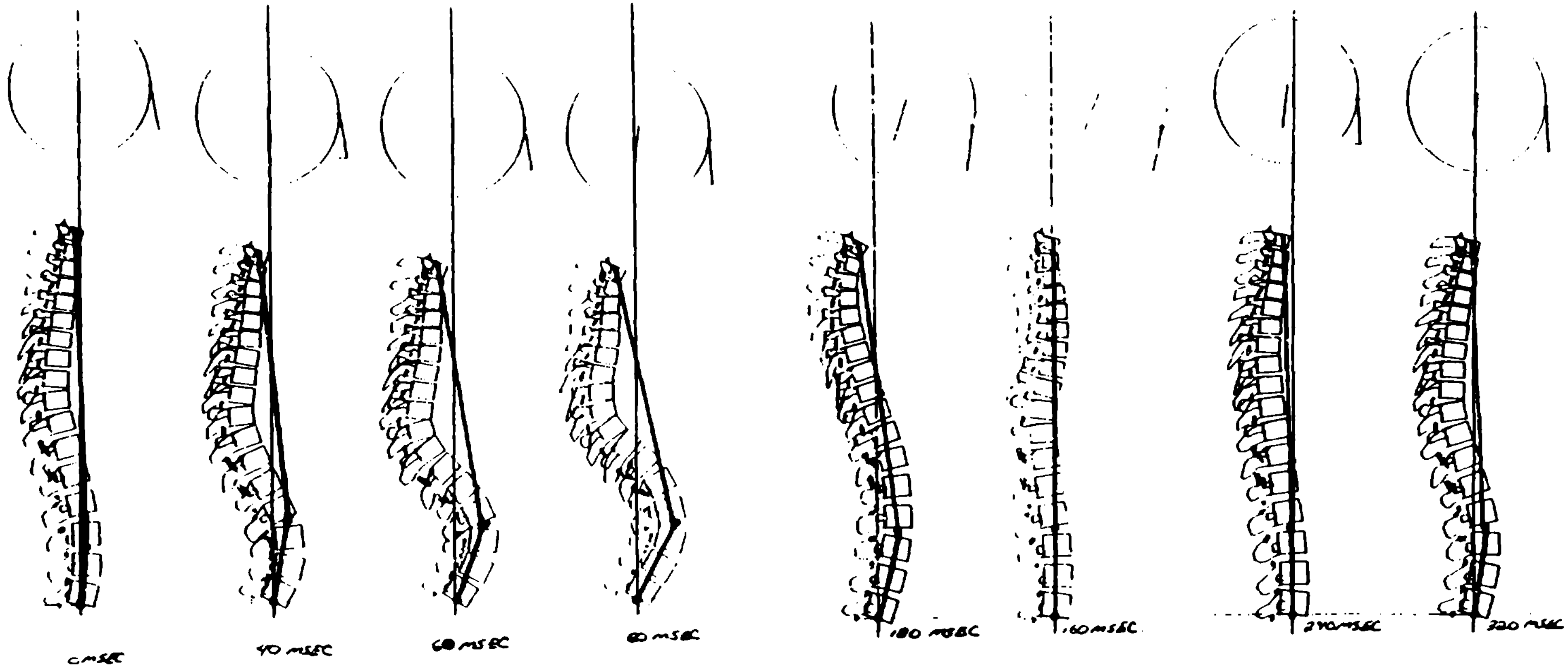



Figure 49. Spine Deformation During a 10 G Ejection as a Function of Time

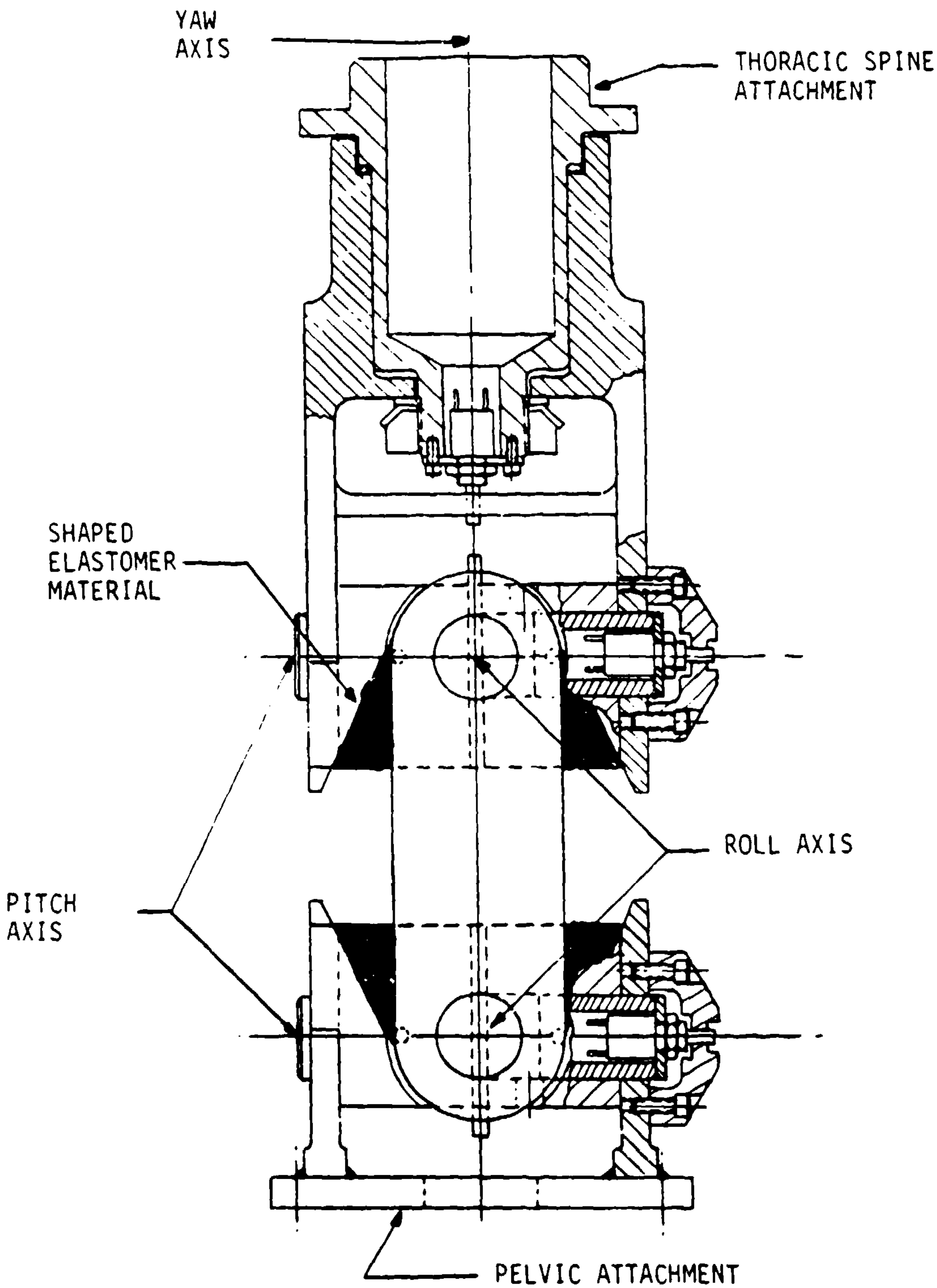



Figure 50. Double Articulated Lumbar Spine With Instrumentation

ejection seat and play a major role in the impedance characteristics of the overall manikin. The initial design of the ADAM buttocks, like that of the LRE, consisted of foam buttocks molded into the outside contours of the manikin pelvis and thighs. It was assumed that this design would have a force-deflection curve similar to that of the LRE. Figure 51 depicts the LRE foam buttocks. Differences in the foam thicknesses for the ADAM and the LRE were anticipated; however, these differences seemed minimal and could be negated by varying the foam density in the ADAM buttocks.

2.2.5.2.1.5. Spine Assembly

The initial spinal system which combines the previously mentioned initial design concepts is shown in Figure 52. It was necessary to avoid a number of complications in order to have a successful final design. One such complication was the interference of the viscera box with the articulated spine and ribs.

2.2.5.2.2. Initial Design Upgrade and Modifications

2.2.5.2.2.1. Lumbar Spine

Further exploration of the double articulated joint revealed certain conceptual flaws. First, although it was expected that the double articulated joint would provide an effective increasing stiffness vertically for the manikin with increased deflection, the joint actually worked in just the opposite manner--as a softening spring in the vertical direction. This deficiency would have had an undesirable effect on the impedance versus frequency curve for the system subjected to increasing G levels.

When the double articulated spine was designed for the small manikin, it also became evident that the spacing between the articulated joints would be severely limited by the movement of the viscera box and the manikin outer skin. An analysis determined that the upper joint for this design could be no higher than 5.9 inches above the hip-leg interface of the manikin. This limit imposed a maximum distance of approximately 2.5 inches between articulated joints in the hinge. This restriction resulted in the elimination of a major advantage of the initial design and a new design was undertaken.

The new design consisted of a single articulated joint for fore, aft, and lateral bending and a pneumatic/hydraulic cylinder system for axial motion of the upper torso. The single articulation point, pictured in Figure 53, allows for 30 degrees of flexion, 20 degrees of extension, 15 degrees

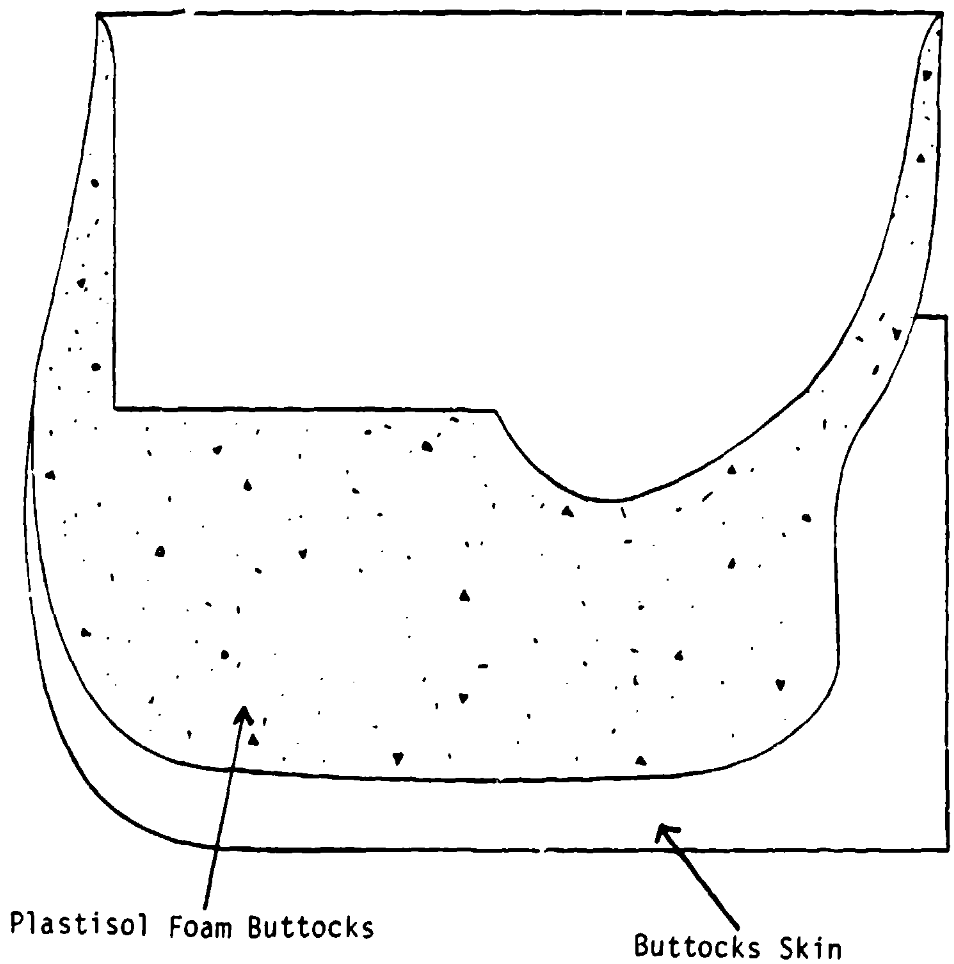



Figure 51. LRE Foam Buttocks

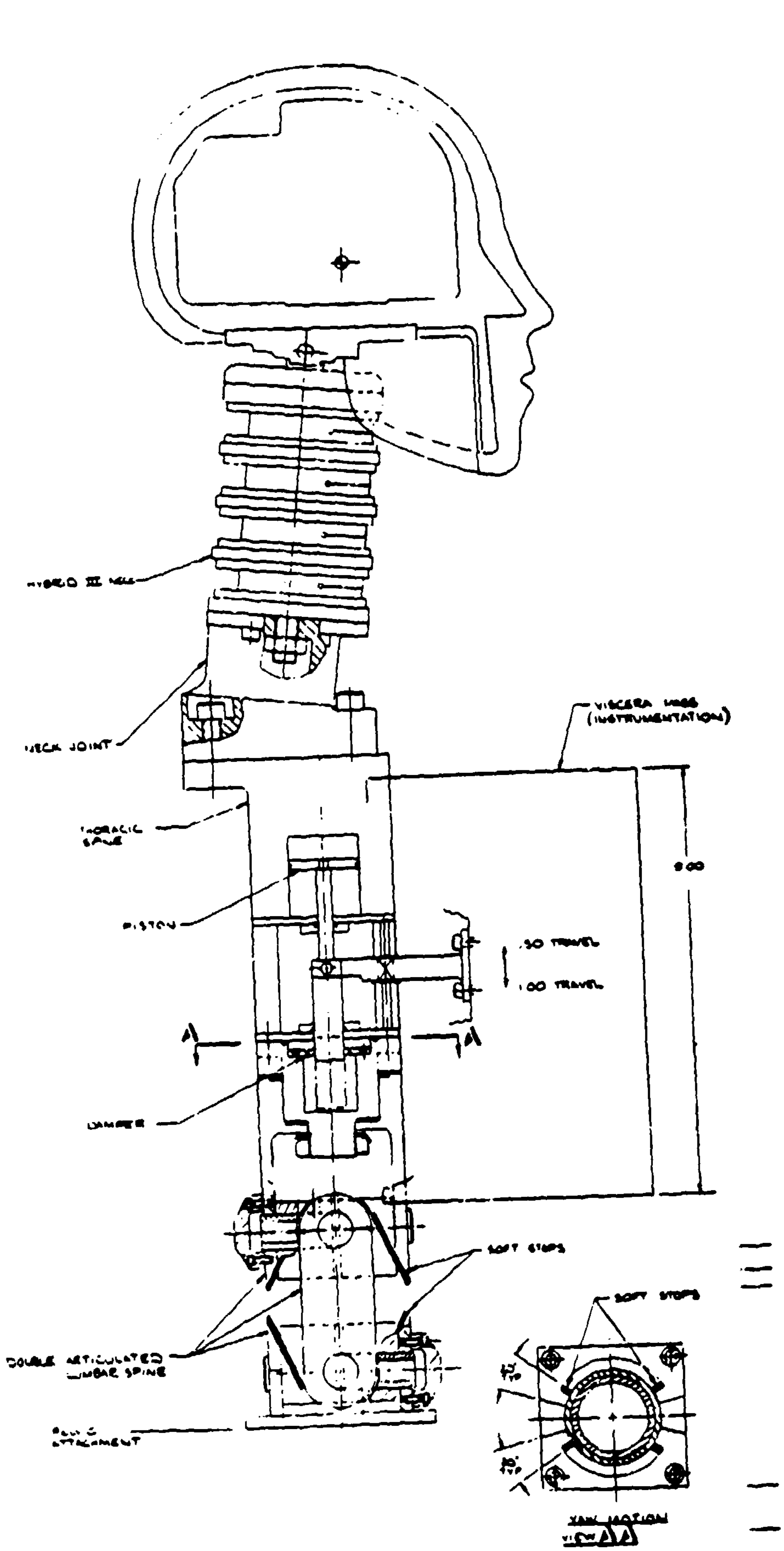



Figure 52. Initial Manikin Spine Concept

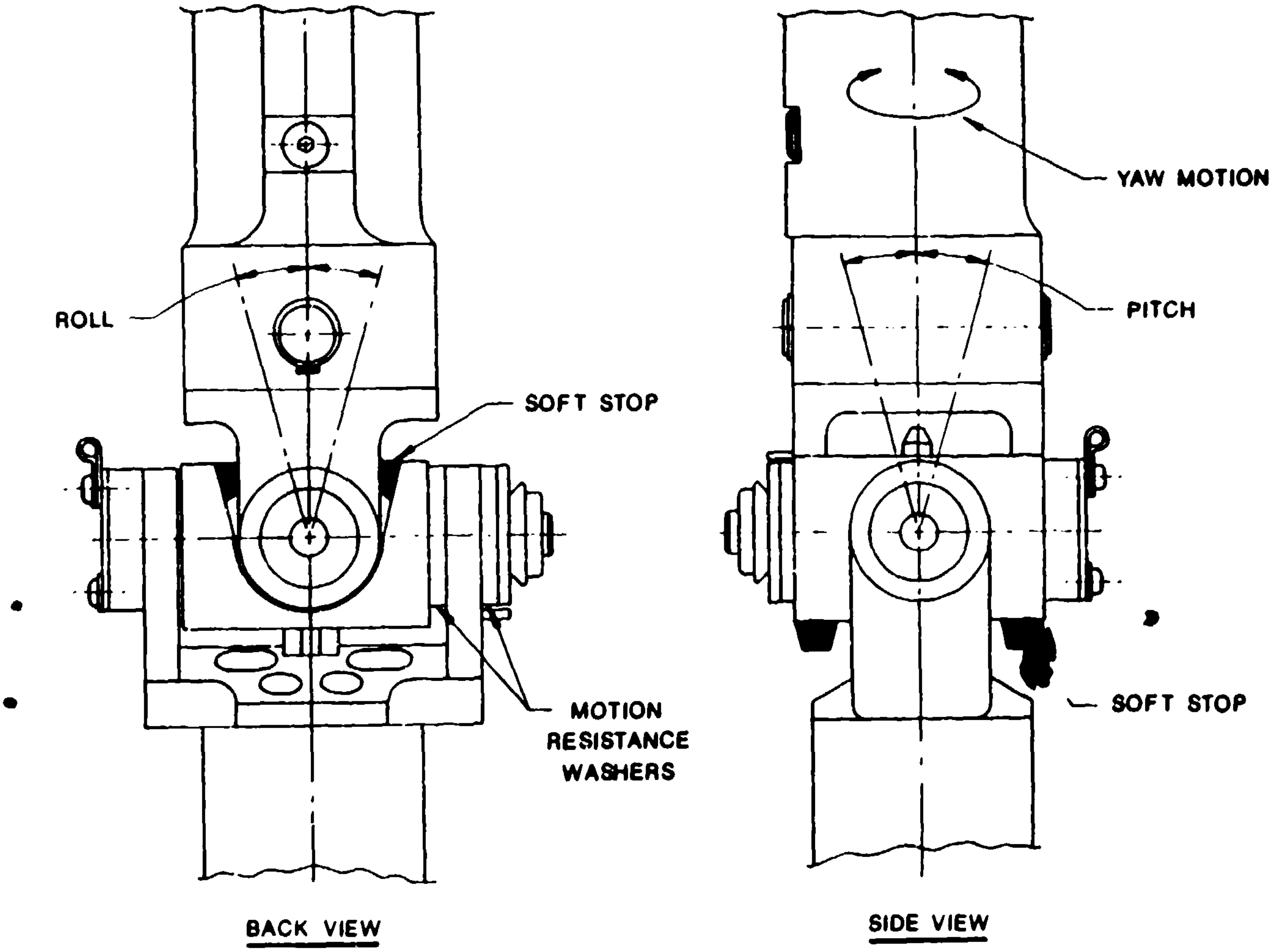



Figure 53. Single Articulation Joint for Lumbar Spine

of yaw, and 15 degrees of lateral bending. For this concept, the correct height from the hip/thigh joint was determined by evaluating the elastic response of the Hybrid III spine to applied forces. This procedure will be discussed later; however, the optimal height of the single articulated joint was found to be 3.9 inches above the hip/thigh joint. This value was used in both the small and large manikins.

In order to represent human spinal compression and nonlinear stiffness during G loading in the axial direction, the ADAM was provided with a pneumatic/hydraulic cylinder system similar to the one described previously for the viscera. A maximum vertical displacement of 7/8 inch down and 1/2 inch up was incorporated into the design.

2.2.5.2.2.2. Axial Travel for Spine and Viscera

In order to verify the effectiveness of the pneumatic/hydraulic piston concept, a mock-up of the spine/viscera was needed to test the concepts. Figure 54 depicts the main elements of the test setup. Testing the pneumatic spine/viscera system revealed that such a system would be impractical for installation into the manikin due to slow air leaks and high friction forces in the system.

The problems associated with the pneumatic/hydraulic piston design resulted in focusing the spinal design on a different type of more practical system which would not have the apparent problems associated with the initial design concepts. The new spinal system was designed to incorporate linear mechanical springs and hydraulic dampers in both the manikin spine and viscera.

2.2.5.2.2.3. Visceral Lockout

A thorough loads and stress analysis on the small manikin revealed that the torso of the small ADAM would be overweight by approximately 5 pounds. After all other weight reducing efforts had been expended, an investigation into removing the visceral degree of freedom from the manikin was under taken. It was determined that the removal of the visceral degree of freedom would allow the attainment of the desired mass characteristics although the impedance characteristics of the manikin would be altered. A decision was made in favor of the mass characteristics; thus, the visceral degree of freedom was removed. Figure 55 depicts a schematic of this modified spinal system.

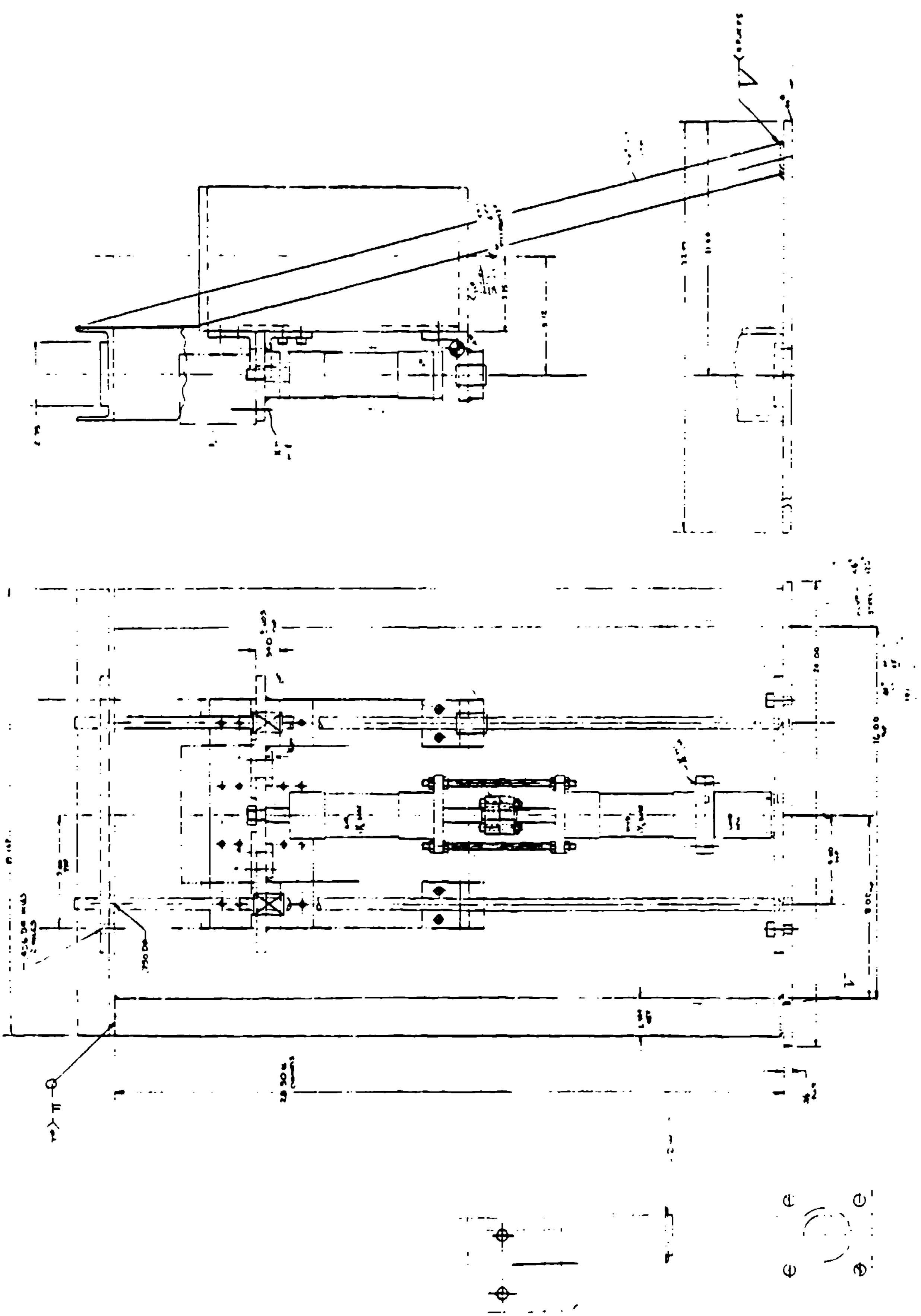


Figure 54. Schematic of Test Apparatus

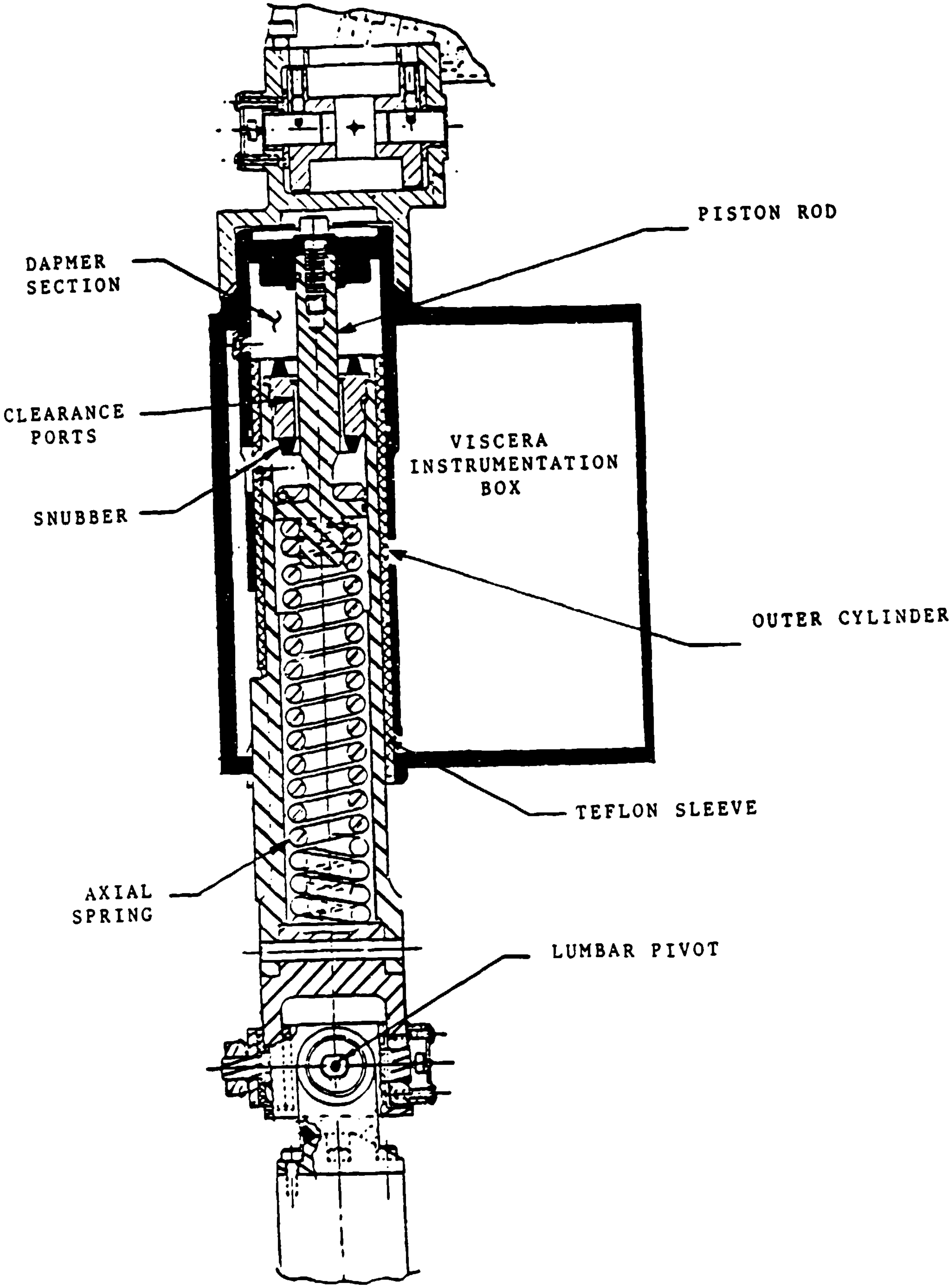



Figure 55. ADAM Mechanical Spring/Damper System--Original Design

2.2.5.2.3. Final Spine Design

2.2.5.2.3.1. Cervical Spine

As previously mentioned, the Hybrid III neck was chosen to represent the cervical spine in the ADAM. Figure 43 presented a drawing of the neck used in the large manikin. Using a shortened or three rubber disc neck in the small manikin was necessary due to height and weight requirements for the small ADAM. Acceptance of each neck was determined by how well the neck response fell within the established corridor. Figures 56 through 59 depict response curves for the accepted large and small prototype necks in both flexion and extension. Although the loading and unloading curve for the necks sometimes failed to fit within the suggested corridor boundaries, they generally met the requirements set forth by the Department of Transportation.

The Hybrid III neck is also equipped with an adjustable tension cable that runs down the center of the neck. By changing the tension in the cable, the response of the system can be altered. The manufacturer has specified a torque of 10 inch-pounds for the nut that controls the cable tension in order to achieve the dynamic responses indicated in the referenced figures.

2.2.5.2.3.2. Viscera

The final design of the ADAM spinal system includes a viscera that is rigidly attached to the thoracic spine. Both the large and small ADAM visceras are made of aluminum and house the electronics of the ADAM system.

2.2.5.2.3.3. Vertical Movement of the Lumbar Spine

The vertical movement of the lumbar spine was obtained through the use of the mechanical spring-hydraulic damper concept that has been discussed. In the final design, the mechanical spring was fabricated of spring steel and provided a spring rate of 650 pound/inch for the small spine and a spring rate of 1020 pound/inch for the large spine. These values yield a frequency of 10 Hz for both the small and large manikin upper bodies with respect to the pelvis.

The hydraulic damping system incorporated in the ADAM was comprised of a steel piston, MIL-H-5606E hydraulic fluid, and an accumulator. Figure 60 presents the thoracic/lumbar spine spring/damper system. The theory behind this system is that, as the upper torso is compressed against the mechanical spring, the displacement of the piston causes the hydraulic fluid to travel from side to side of the piston. The flow rate of the fluid is regulated by the gap width between the

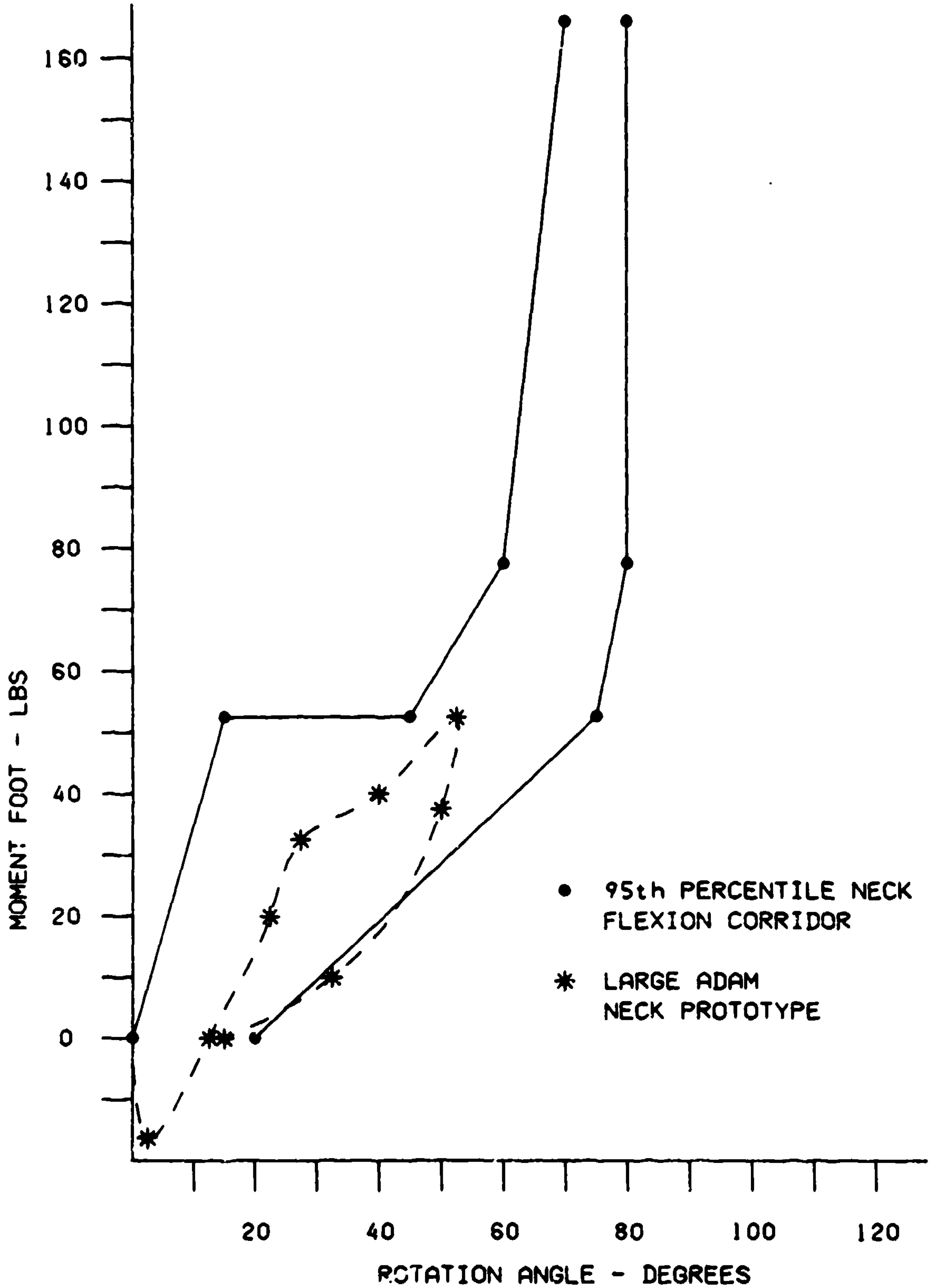



Figure 56. Large ADAM Neck--Flexion

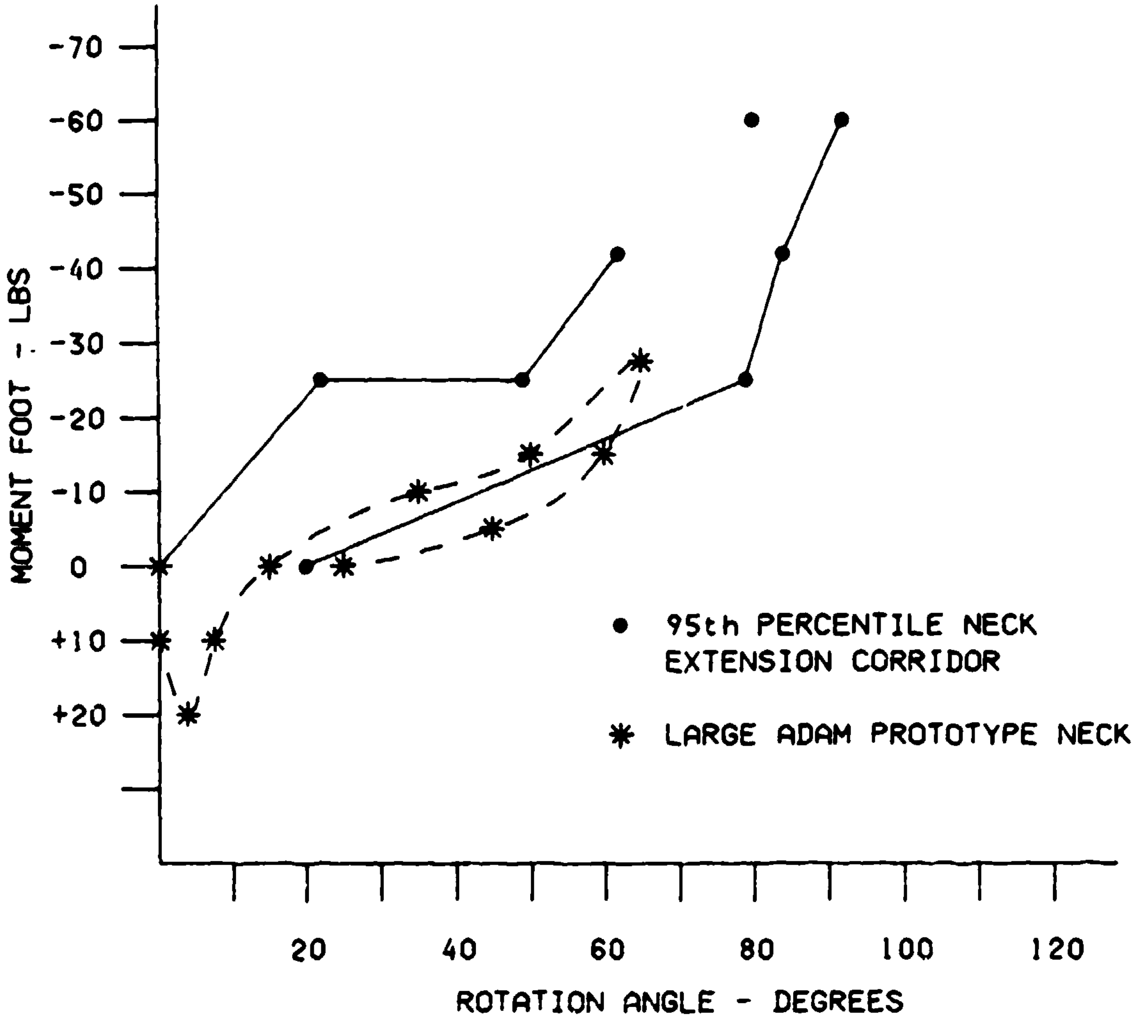



Figure 57. Large ADAM Neck--Extension

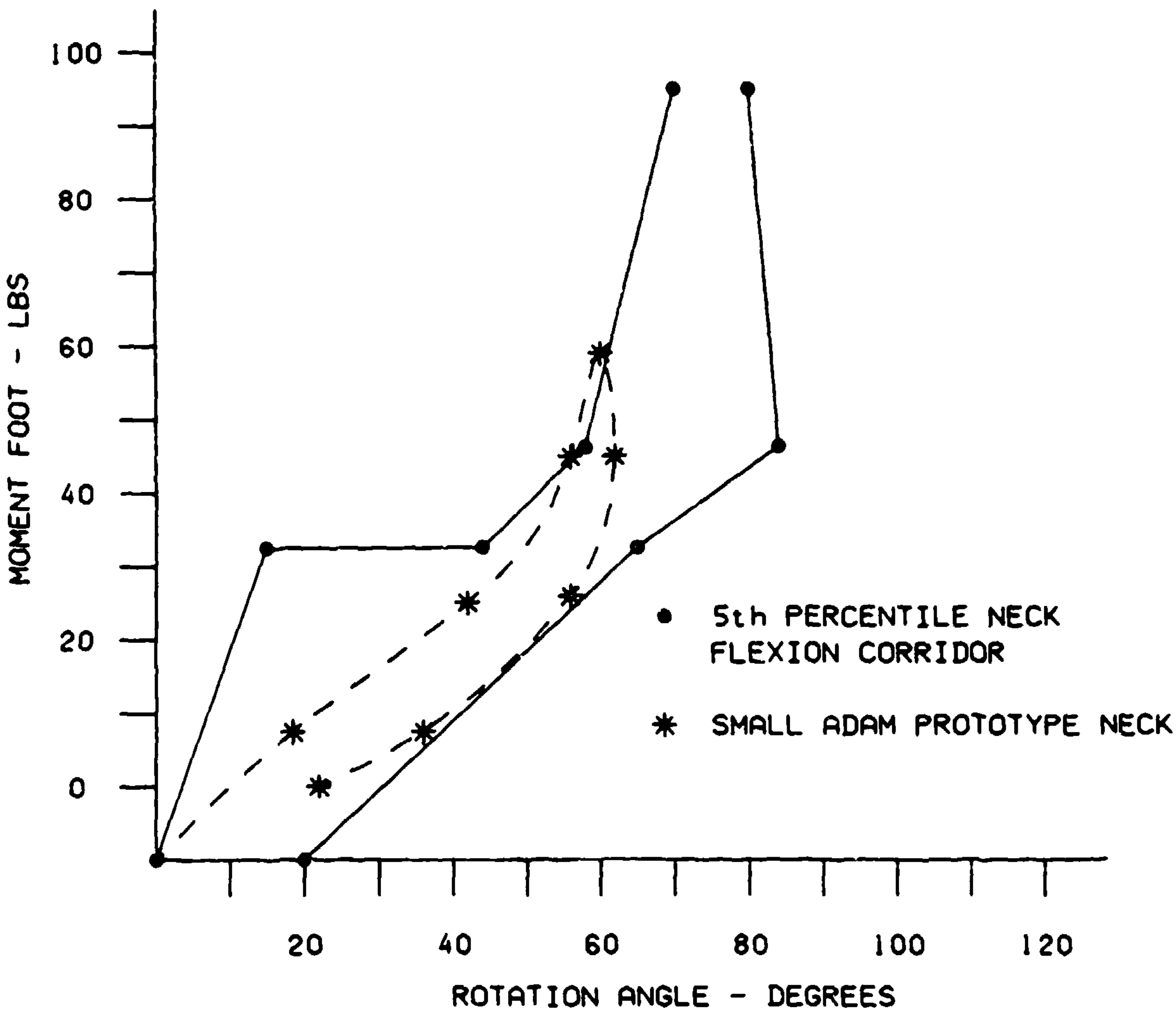



Figure 58. Small ADAM Neck--Flexion

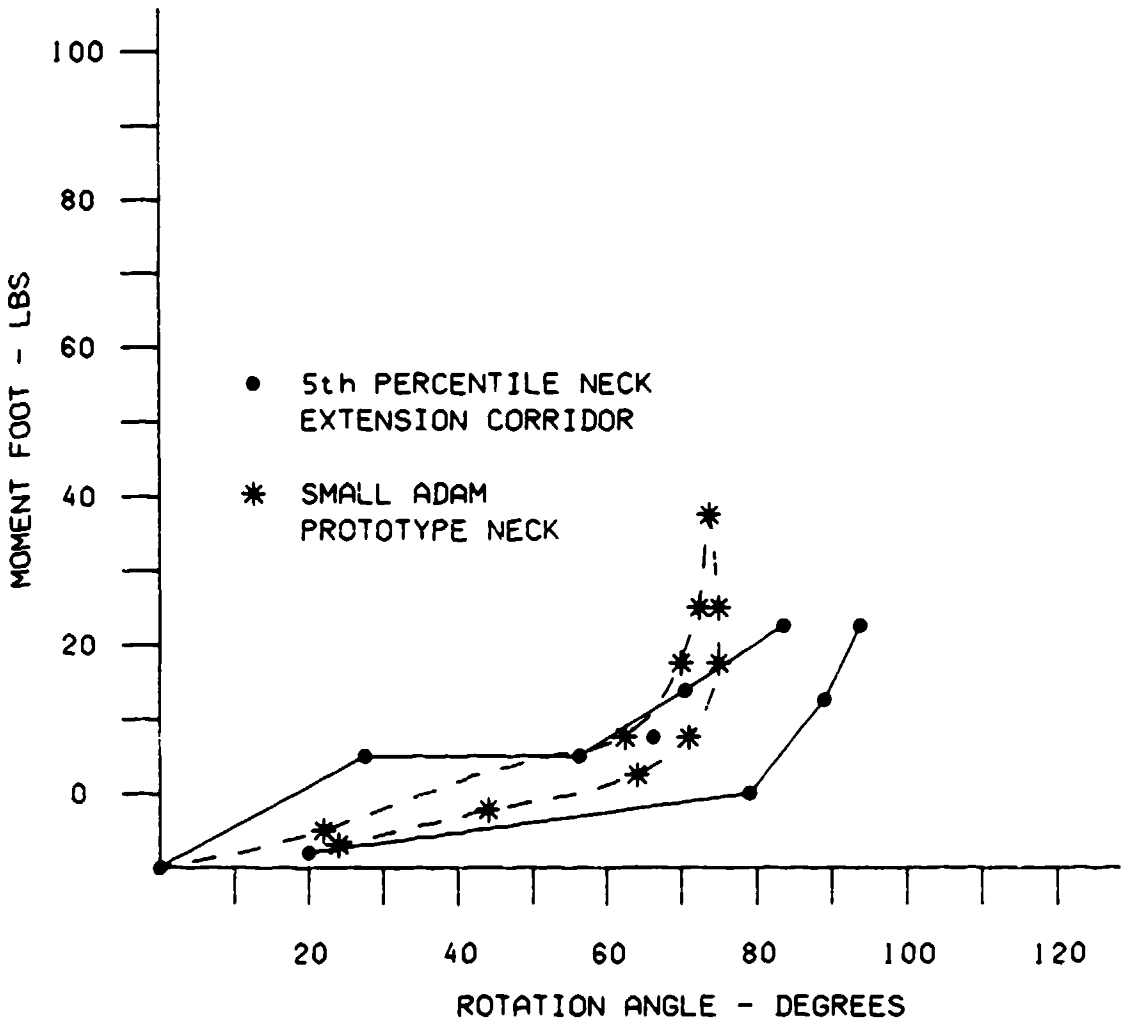



Figure 59. Small ADAM Neck--Extension

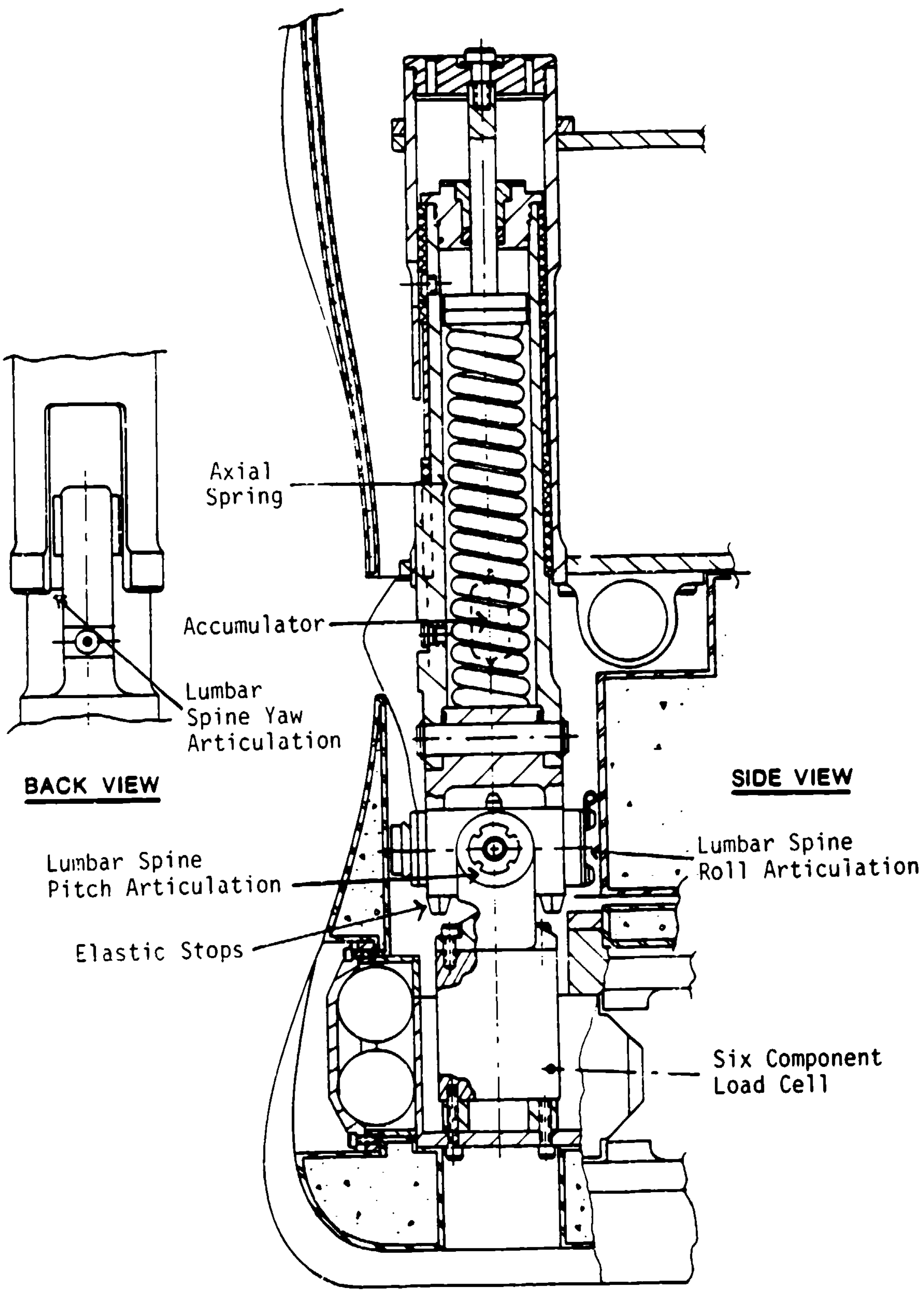



Figure 60. Thoracic/Lumbar Spine Rotational Mechanism

piston and the inner spine sleeve wall as well as the viscosity of the fluid. This limited fluid flow rate yields an effective damper for the vertical motion of the upper torso of the manikin.

An accumulator was added to the spring/damper system because the volumetric rates of change with respect to piston displacement for the upper and lower cylinder chambers were different. The accumulator, which is essentially an air pocket encased in a pliable plastic shell, will decrease in volume when more fluid is displaced from the upper chamber than the lower chamber is able to accommodate. This device will also increase in volume when fluid is passed from the bottom chamber to the upper chamber where an excess of volume is being created. By functioning in this manner, the accumulator allows the spring/damper system to operate smoothly without any system lock-ups.

In order to size the piston/cylinder gap width and accumulator, system level testing was conducted on the lumbar spine spring/damper system. The following design parameters were obtained from these tests:

| Element | Small Manikin Sizing | Large Manikin Sizing |
|---|---|---|
| Gap Width | 0.0055 inch | 0.0055 inch |
| Accumulator | 5 milliliters | 11.5 milliliters |

When incorporated into the large and small manikins, these parameters yield lumbar spine systems with damping ratios of approximately 60 percent of critical. These system tests will be discussed in more detail in Section 2.2.5.5.

Due to the lack of contact points within the ADAM spinal design, the static friction within the system was minimized. Also, rubbing of the piston against the cylinder wall did not occur due to proper alignment of the two involved parts.

2.2.5.2.3.4. Rotational Movement of the Lumbar Spine

The flexion, extension, and lateral rotation of in the ADAM manikin spines was provided through the articulated hinge joint. The joints were placed 3.9 inches above the hip/thigh interface and were made of steel. Friction devices were installed on the hinge joints to tighten the upper body of the manikin to an effective G loading. In addition to these friction devices, soft rubber stops were added to each joint stop to decrease the impact forces and increase the human-like response of the manikin near the limits of motion. The following ranges of rotational motion were incorporated into the lumbar spine of the ADAM:

- Flexion: 30 Degrees
- Extension: 20 Degrees
- Lateral Bending: 15 Degrees
- Rotation: 15 Degrees

The rotation for the lumbar spine was obtained between the inner and outer sleeve of the lumbar spine. Figure 60 presents a view of the lumbar spine rotational mechanism. As stated in the ADAM Statement of Work, USAF Contract F33615-85-C-0535, Advanced Dynamic Anthropometric Manikin (ADAM), Systems Research Laboratories, Inc., 11 September 1985, to meet the requirements, the ranges of motion in the ADAM were measured without the soft stops in place. When these stops were added to the system, the effective ranges of motion were approximately 5 degrees less than were previously indicated.

2.2.5.2.3.5. Buttocks

The buttocks in the ADAMs were constructed of a vinyl plastisol skin and foam as in the LRE system. Analysis determined that the buttocks would have a stiffness of approximately 700 pounds/inch. This analysis will be discussed in a later section. The large manikin was designed with a thicker buttock than the small manikin and, because of this, was believed to have a smaller spring constant and less damping.

2.2.5.3. Development and Verification of Prediction Programs

2.2.5.3.1. Impedance

In order to meet ISO-5982-1981, as required in the SOW, a mathematical model was needed for initial sizing of body elements. Research into impedance modeling by Mertens and others and an assumption of negligible deflections in the X and Y directions resulted in a one-dimensional impedance model. The assumption of negligible deflections in the X and Y directions was consistent with the ISO report that states, "In general, the human responses are similar to those of a single order system."

The SRL impedance predictive tool had to properly define segment mass, damping, and spring elements; at the same time, it was required to be simple enough for efficient application. Previous research revealed that the three main contributors to body vibratory response were the buttocks, spine, and viscera. Using these body segments, the four mass system, illustrated in Figure 61, was defined. In this model, $M_4$ represents the visceral mass; $M_3$ represents the remainder of the upper body mass (including the head, neck, and arms) that apply force on the lumbar spine; and $M_2$ depicts the total of the mass between the lumbar spine and the seat. The sum of these three masses equals the overall body mass without legs. $M_1$ depicts the mass of the seat. The cervical spine is not treated as a spring damper system because, based on the small mass of the head and the high stiffness of the neck, the head/neck system has little effect on system impedance. The system in Figure 61 can be transformed into the four mass system of Figure 62. The driving point impedance of the seat was not to be included in the driving point impedance calculations; therefore, this mass becomes a driving point and was assigned a mass value of zero.

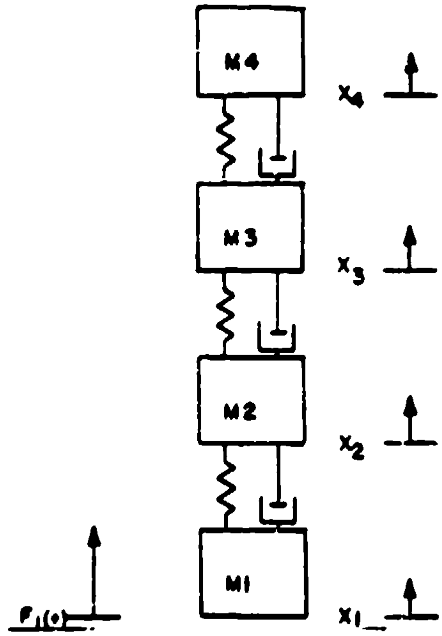



Figure 61. Mathematical Impedance Model of the Spine/Viscera

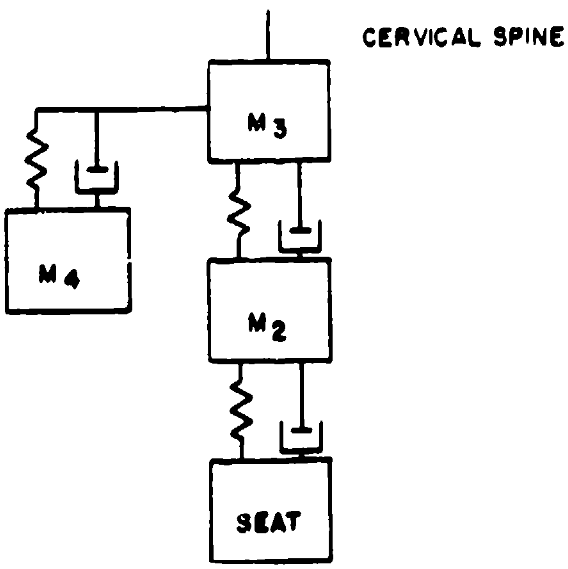



Figure 62. Equivalent Mathematical Impedance Model with Adjustable Seat Value

From the model of Figure 62, the four complex equations of motion were written. To solve for Z1, the impedance of the model taken at the seat, the following closed form solution was determined and is presented in matrix form:

$$Z_1 = \text{abs}\;\frac{\begin{vmatrix} (-m_1\omega^2+c_1 i\omega+k_1) & -(c_1 i\omega+k_1) & 0 & 0 \\ -(c_1 i\omega+k_1) & [-m_2\omega^2+(c_1+c_2)i\omega+(k_1+k_2)] & -(c_2 i\omega+k_2) & 0 \\ 0 & -(c_2 i\omega+k_2) & [-m_3\omega^2+(c_2+c_3)i\omega+(k_2+k_3)] & -(c_3 i\omega+k_3) \\ 0 & 0 & -(c_3 i\omega+k_3) & (-m_4\omega^2+c_3 i\omega+k_3) \end{vmatrix}}{\begin{vmatrix} 1 & -(c_1 i\omega+k_1) & 0 & 0 \\ 0 & [-m_2\omega^2+(c_1+c_2)i\omega+(k_1+k_2)] & -(c_2 i\omega+k_2) & 0 \\ 0 & -(c_2 i\omega+k_2) & [-m_3\omega^2+(c_2+c_3)i\omega+(k_2+k_3)] & -(c_3 i\omega+k_3) \\ 0 & 0 & -(c_3 i\omega+k_3) & (-m_4\omega^2+c_3 i\omega+k_3) \end{vmatrix}}$$

where

| | | |
|---|---|---|
| $\omega$ | = | Excitation frequency. |
| $m_1, m_2, m_3, m_4$ | = | Masses of the individual elements from the seat to the viscera (m1 was given an initial value of O). |
| $c_1, c_2, c_3$ | = | Damping constant of damper above m1, m2, and m3, respectively. |
| $k_1, k_2, k_3$ | = | Spring constant of spring above m1, m2, and m3, respectively. |
| 1 | = | Imaginary part of each term. |
| abs | = | Absolute value of the entire equation. |

A computer program was written to permit systematic and efficient evaluation of the effects of various spring, mass, damper combinations. This evaluation was used to size elements that result in an impedance curve that would correlate with the ISO standard.

Research revealed that the weight of the visceral mass was approximately 28 pounds for the 50th percentile human. This value is consistent with the value used for the visceral mass in works by Belytschko and Privitzer (1978). The center of gravity of the viscera located as illustrated in Figure 63, just forward of the second lumbar vertebrae. Values for $m_2$ and $m_3$ were obtained using the visceral weight and Table 23; $m_2$ was composed of segments 4 and 5 (abdomen and pelvis); $m_3$ was comprised of segments 1 through 11, minus $m_2$, minus the visceral mass. Final numerical values for a 50th percentile human were as follows:

- $m_2 = 31.3$ pounds
- $m_3 = 55.6$ pounds
- $m_4 = 28.0$ pounds

2.2.5.3.2. Verification of the Impedance Program

The computer program was verified by effectively eliminating two of the masses ($m_3$ and $m_4$) so that a one degree of freedom system remained. By allowing the mass spring damper system to first have the values of $m_1$, $k_1$, and $c_1$ specified in ISO-5982-1981, Annex B, then have the values of $m_2$, $k_2$, and $c_2$ in the same work, the overall impedance of the system was obtained by linear superposition of the two sets of results. This method was successful because the model presented in the ISO standard is comprised of two one degree of freedom systems connected in parallel. Verification of the program can be seen in Figure 64, where SRL computer results of the impedance modulus are plotted against the ISO report mechanical model results.

In addition to calculating the impedance modulus, the computer program determined the phase angle of the system. Figure 65 illustrates acceptable correspondence between the ISO standard data and data calculated by the impedance program.

Although matching the one-dimensional ISO standard experimental data did not guarantee correct experimental impedance characteristics for the three-dimensional manikin, it did represent a viable starting point from which the three-dimensional manikin could be developed. By incorporating the one-dimensional springs and dampers of the impedance model into ADAM, the multidegree of

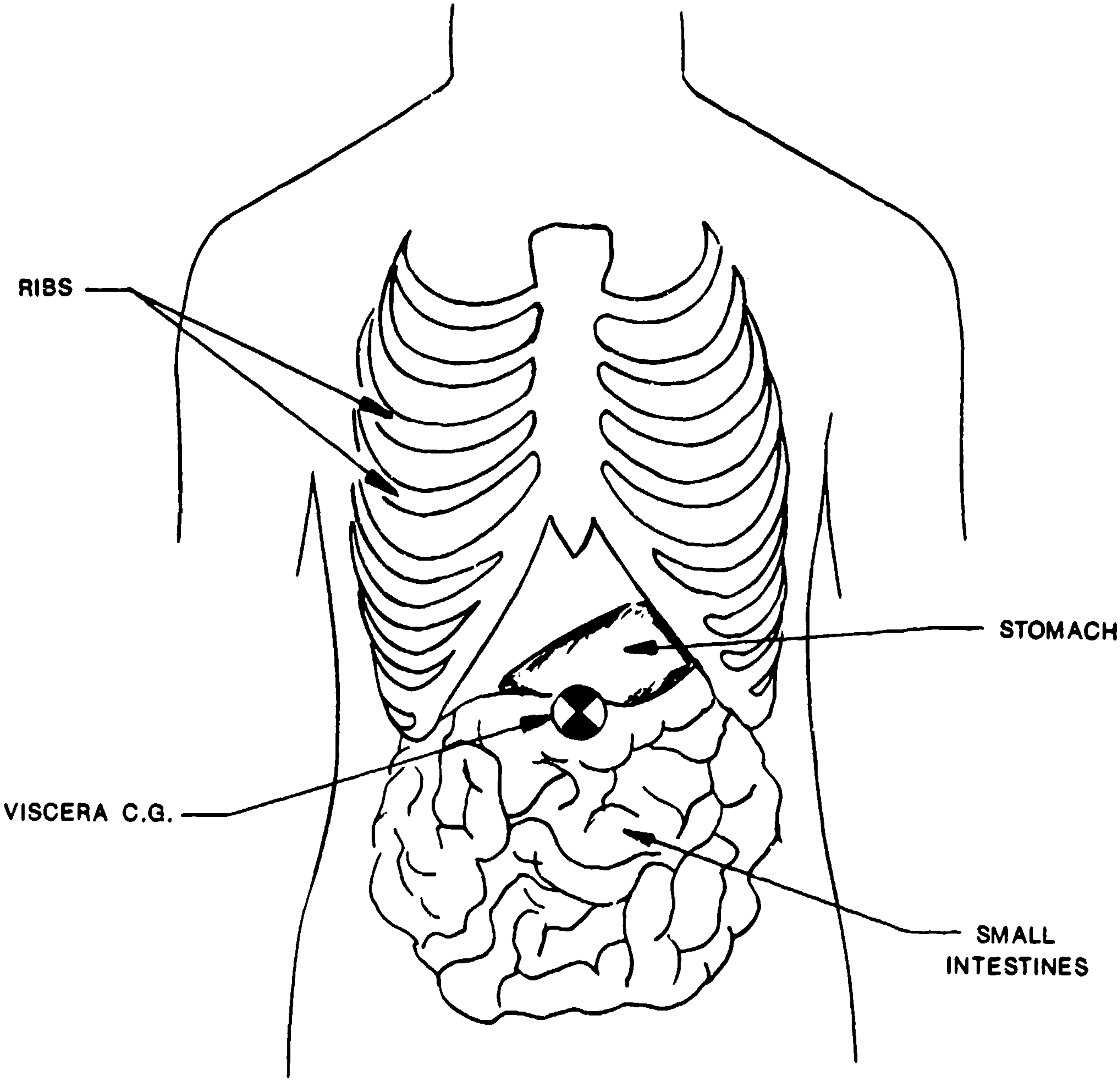



Figure 63. Abdominal and Thoracic Viscera

TABLE 23. SPECIFICATION WEIGHTS FOR BODY SEGMENTS AND THE WHOLE BODY FOR SMALL, MIDSIZE, AND LARGE ADAM

| Number | Segment | Small | Midsize | Large |
|---|---|---|---|---|
| 1 | Head | 9.2 | 9.4 | 9.8 |
| 2 | Neck | 2.0 | 2.3 | 2.8 |
| 3 | Thorax * | 39.6 | 54.9 | 66.2 |
| 4 | Abdomen * | 4.2 | 5.3 | 6.4 |
| 5 | Pelvis * | 19.5 | 26.0 | 32.6 |
| 6 | Rt. Upper Arm | 3.4 | 4.4 | 5.4 |
| 7 | Rt. Forearm | 2.5 | 3.0 | 3.7 |
| 8 | Rt. Hand | 1.0 | 1.1 | 1.3 |
| 9 | Lt. Upper Arm | 3.4 | 4.4 | 5.4 |
| 10 | Lt. Forearm | 2 5 | 3.0 | 3.7 |
| 11 | Lt. Hand | 1.0 | 1.1 | 1.3 |
| 12 | Rt. Thigh | 17.1 | 21.7 | 25.9 |
| 13 | Rt. Calf | 6.8 | 8.5 | 10.0 |
| 14 | Rt. Foot | 1.7 | 2.1 | 2.5 |
| 15 | Lt. Thigh | 17.1 | 21.7 | 25.9 |
| 16 | Lt. Calf | 6.8 | 8.5 | 10.0 |
| 17 | Lt. Foot | 1.7 | 2.1 | 2.5 |
| | TOTALS | 139.5 | 179.5 | 215.4 |
| *Torso | | 63.3 | 86.2 | 105.2 |

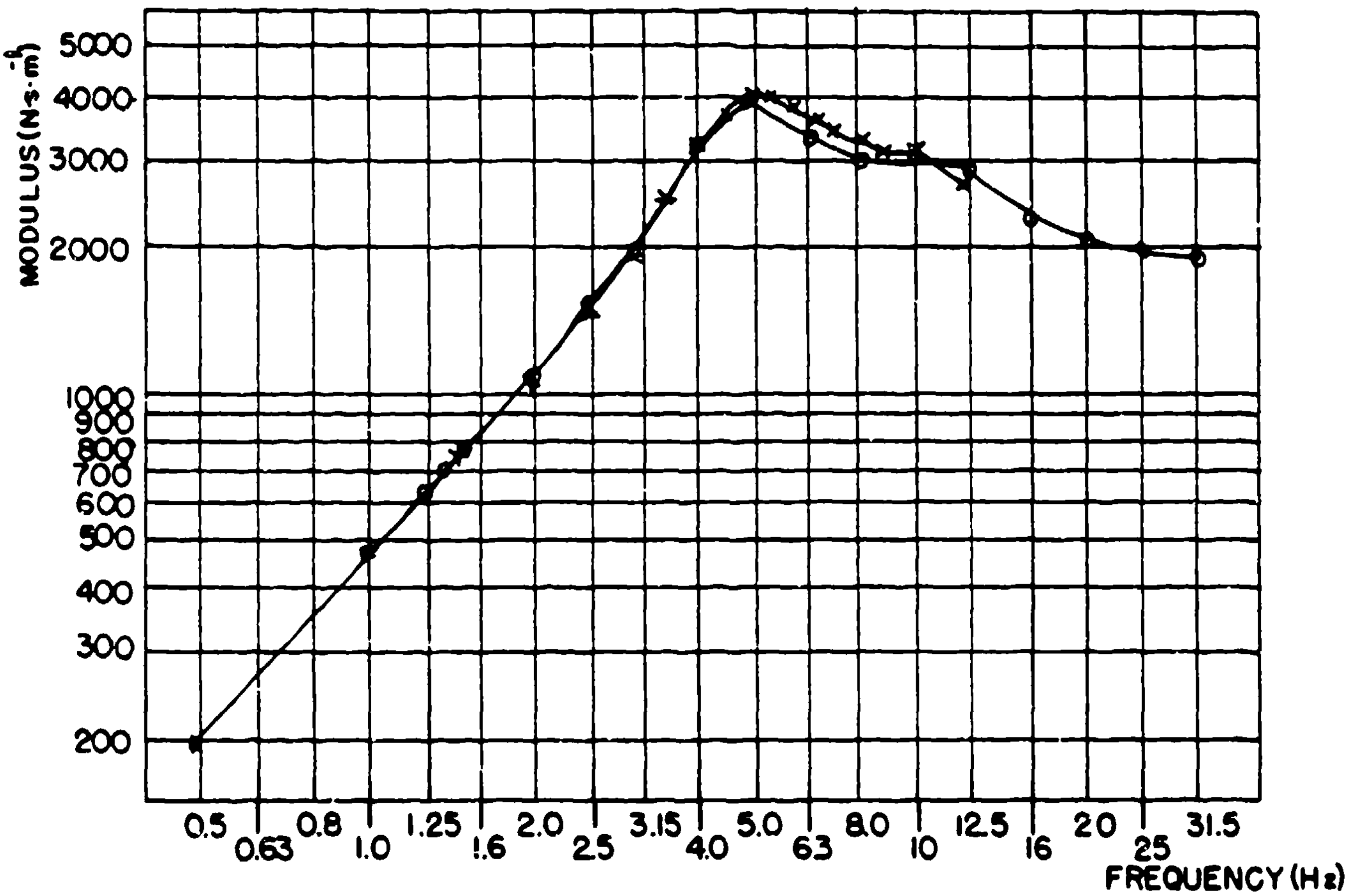



Figure 64. Impedance Modulus Versus Frequency Verification for SRL Computer Program (X)

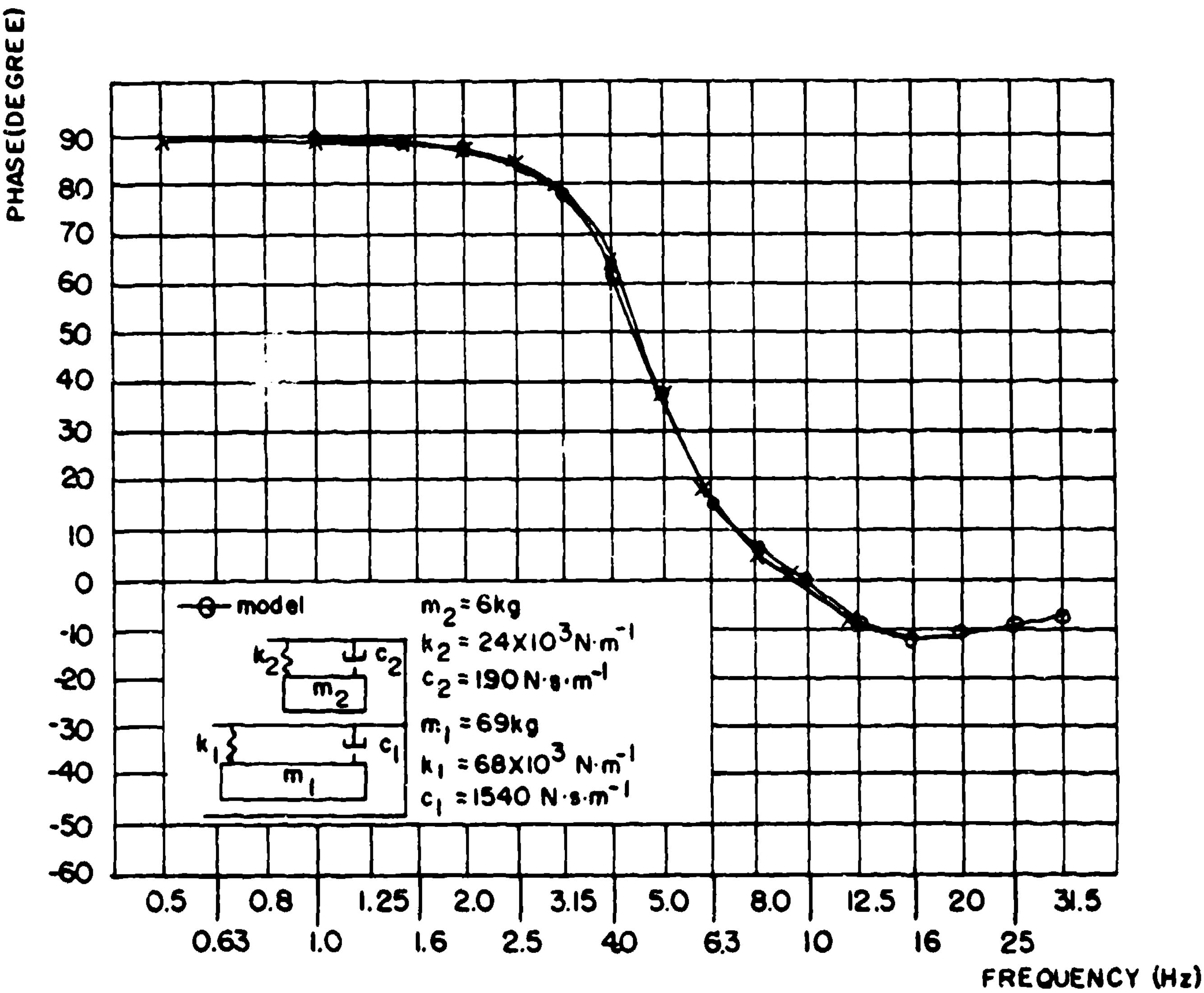



Figure 65. Phase Angle Verification for SRL Computer Program

freedom manikin could be refined by fine tuning each of the spring/damper units to obtain correct response to vibratory loading.

By accounting for $m_1$ in the computer program, provision was made for adding seat mass at the driving point. By changing $m_1$ from zero to a value representing the seat mass, the analysis accounts for a situation in which the sinusoidal forcing function is applied to the manikin through the seat. This provides ready capability to calculate the change in impedance attributable to the presence of the seat and a means of eliminating it for comparison of the experimental system with the ISO requirements.

#### 2.2.5.3.3. Dynamic Response Program

In order to fully understand both the internal loadings and the dynamic response of the ADAM manikins, a dynamic response analysis was developed. The response analysis was confined to the Z direction for two reasons. First, the amount of restraint the CREST seat provides would tend to limit X and Y response of the manikin to a negligible amount. Second, available data for Z direction drop tower acceleration tests of humans were separated into X, Y, and Z responses to input Z loading. This tended to demonstrate that a majority of human response to the drop tower tests came in the Z direction.

The X and Y direction responses were not analyzed for ADAM as the response to G-loadings in these directions are controlled by the elastic characteristics of the restraint harnesses. Since the mass distribution and the articulations of humans were duplicated in ADAM, it can be assumed that the dynamic response characteristics, controlled by restraint system, would be correct.

Like the impedance program, the dynamic response program is based on a multidegree of freedom one-dimensional model in the Z direction. However, unlike the impedance model, the response analysis is easily modified for up to 50 masses connected in series. Figure 66 presents a symbolic representation of the dynamic response model. Each mass in the dynamic response analysis uses the following one-dimensional equation of motion:

$$M_i\ddot{X} + C_i(\dot{X}_i-\dot{X}_{i-1}) + C_{i+1}(\dot{X} -\dot{X}_{i+1}) + K_i(\dot{X}_i-X_{i-1}) + K_{i+1}(X_i-X_{i+1}) = F_i$$

This equation indicates that each mass in the system is capable of having an external force applied to it. The ability to model nonlinear springs and dampers is also present in this computer program, as nonlinear equations based on displacements and velocities can be written and incorporated for

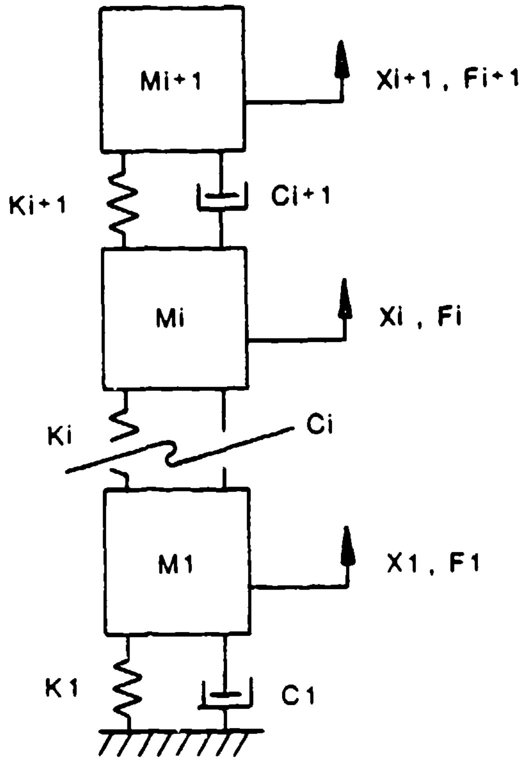



Figure 66. Dynamic Response Model

each Ki and Ci, respectively. A fourth order Runge-Kutta method was used to integrate the entire system of equations as a function of time. Therefore, unlike the impedance program which solves the system using a closed form solution, the response analyses results are time dependent.

The output of the program includes displacements and forces, driving point impedances for sinusoidal motions, and G level accelerations on each element.

2.2.5.3.3.1. Verification of Dynamic Response

Because of the inherent similarities between the SRL impedance program and the dynamic response analysis in terms of system elements, a verification procedure using the impedance program was developed. To verify the dynamic response analysis, the impedance program was run at ±1 G oscillatory acceleration. Next, the dynamic response analysis was run for frequencies of 1 to 15 Hz at ±1 G oscillatory acceleration. The results of both analyses are plotted and compared in Figure 67. Evaluation of the plotted results indicates excellent correlation between impedance

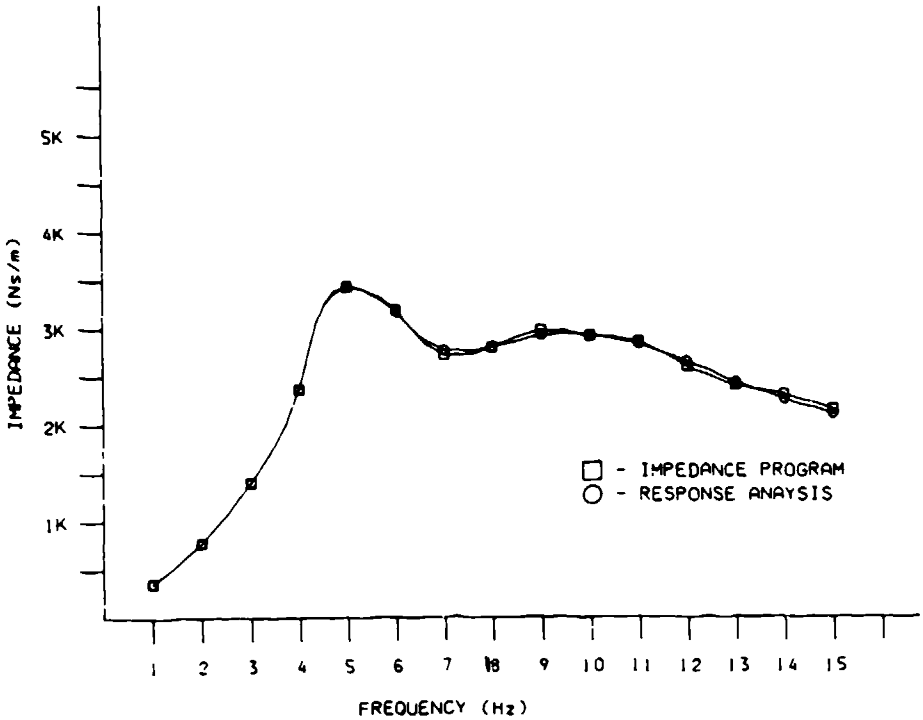



Figure 67. Linear 1 G Validation for Response Analysis Program

values obtained using the dynamic response analysis and those obtained using the impedance program. This led to the conclusion that the dynamic response analysis was technically sound.

The evaluation of Gz response was based on correlation of theoretical results with results of drop tests of an F-111 seat system which were obtained by AAMRL (1980, 1983). The data shown in Figures 68 and 69 were selected from AAMRL (1983) and represent the seat acceleration input and torso response (acceleration), respectively, of the human that best represents the small manikin based on an evaluation of manikin height and weight.

In order to establish a correlation between test data and model results, the experimental seat acceleration input was broken down into a set of linear segments and used as input in the dynamic response analysis program. The output of the predictor program, acceleration of the model torso element, was plotted versus time, and a correlation between the experimental and analytical results was established.

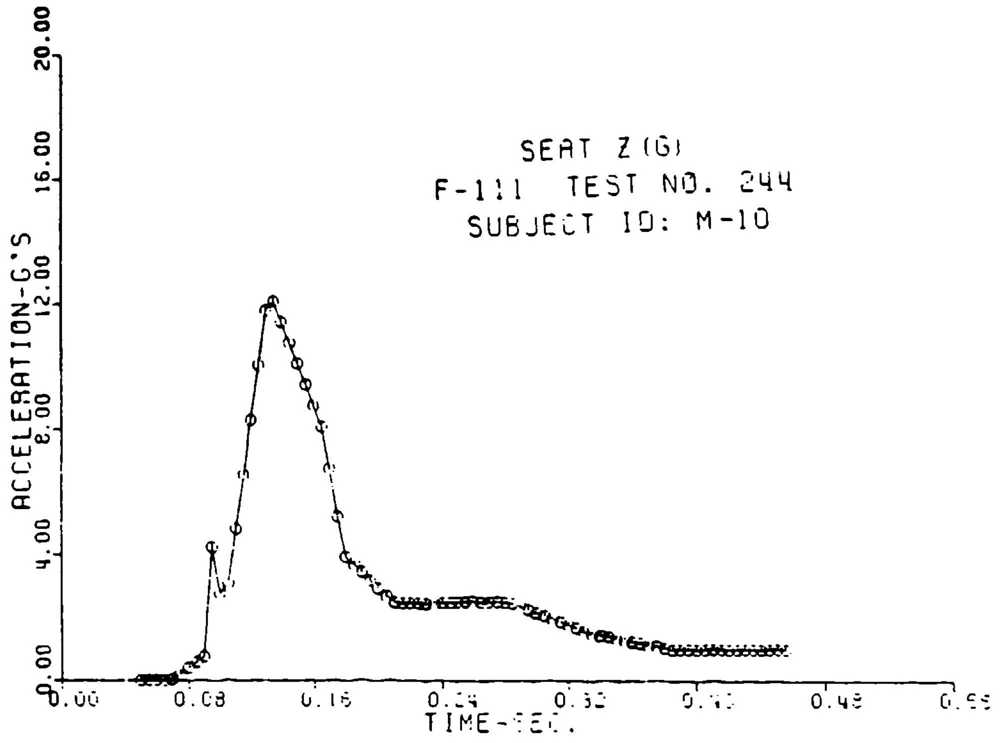



Figure 68. Experimentally Measured F-111 Seat Acceleration Used in the Small ADAM Correlation Effort

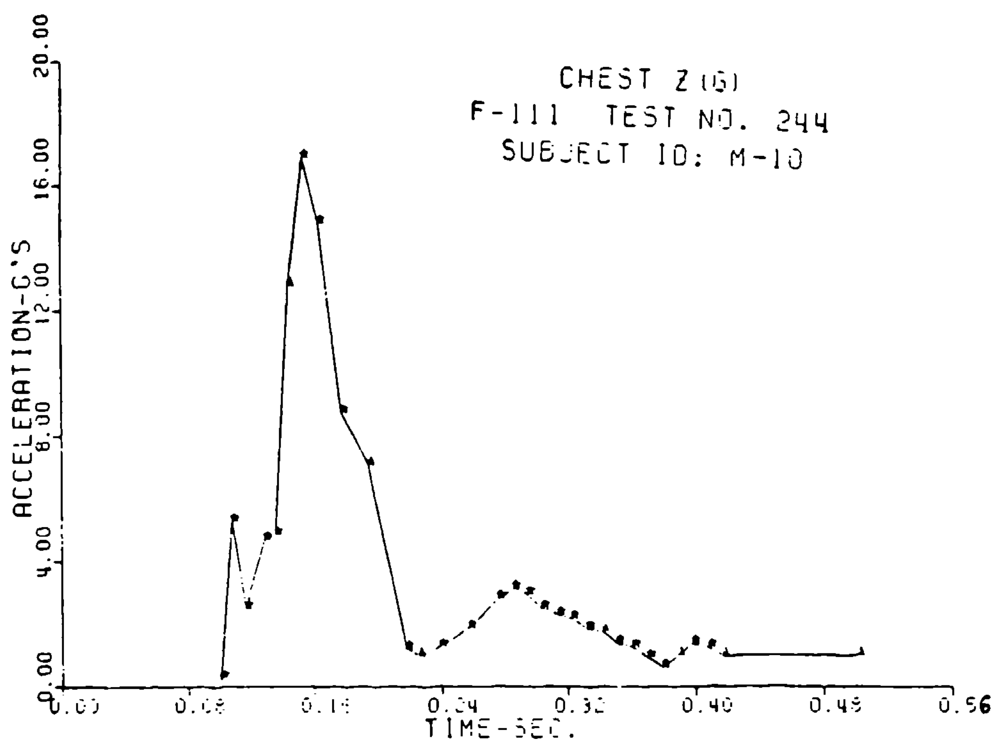



Figure 69. Experimentally Measured F-111 Torso Acceleration Data Used in the Small ADAM Correlation Efforts

2.2.5.4. Determination of Desired Parameters

2.2.5.4.1. Ranges of Motion

Accurately depicting the human ranges of motion in the ADAM spine was necessary in order for the ADAM system to exhibit dynamic response and static positioning characteristics representative of a human being. By studying the works of Mertz and Patrick (1971) and Schneider et al. (1976) for human neck response, and Nyquist and Murton (1975) and Cheng et al., (1979) for lower back response, realistic ranges of motion for the human spine were determined.

2.2.5.4.1.1. Cervical Spine Motion

From Mertz and Patrick (1971) and Schneider et al. (1976), human neck free ranges of motion for flexion and extension, right and left lateral bending, and right and left rotation were compiled. The values of these free ranges of motion, which are measures of the travel that an average human can move through freely, are as follows:

- Flexion: 56 Degrees
- Extension: 74 Degrees
- R + L Rotation (Yaw): 72.5 Degrees
- R + L Lateral Bending: 45 Degrees

Values for neck ranges of motion showed no variation with respect to the size of the human being measured.

The Hybrid III neck, which was chosen for the ADAM systems, has no free range of motion. In other words, force must be applied for neck deflection to occur. The requirements that must be met in terms of moment versus angle corridors for both flexion and extension were developed by R. F. Neathery for 5th and 95th percentile manikins and were discussed earlier.

2.2.5.4.1.2. Lumbar Spine Motion

Data on lower back flexion and extension were provided exclusively by Nyquist and Cheng. A rough average of the values presented in these reports are as follows:

- Flexion: 50 Degrees
- Extension: 55 Degrees

While both of these motions could be realized in the ADAM, lower back flexion and extension motions were decreased to approximately 30 degrees and 20 degrees, respectively, due to possible interference problems between the ADAM viscera and pelvis skins. This restriction in motion will cause no biofidelity problems in the seated, restrained manikin, as it has been found that both the seat and the restraint harness tend to limit the motions of the human in an ejection environment.

Data relative to lateral bending and rotation of the lower back could not be located in available references. However, it was concluded that the sides of the CREST seat would limit the lateral movement of the ADAM torso. Buttocks roll, plus 15 degrees of lateral bending in the lower spine, will result in an overall shoulder lateral deflection of considerably more than 4 inches. Since travel of 4 inches is very likely to exceed the range of motion allowed by the seat harness or the sides of the seat, lateral roll of 15 degrees for the lower back, which yields a lateral displacement of 4 inches at the shoulder, was used as the design parameter.

Finally, upper torso yaw (i.e., rotation) was assigned a value of 15 degrees because it is believed that the ejection seat restraint harness will limit yaw motion to less than 15 degrees.

#### 2.2.5.4.2. ADAM Buttock Stiffness

Early in the ADAM program, an estimate of buttocks force-deflection characteristics was determined to be essential for correct modeling of the ADAM system. Like the LRE, the buttocks for the ADAM used a plastisol foam to provide the spring and damping characteristics of the buttocks. However, due to possible density variations in the foam, as well as its complex molded geometry, a force-deflection calculation would have been extremely difficult to develop.

In order to estimate the static force-deflection curve for the ADAM buttocks, the measured force-deflection curve of the LRE buttocks was used initially. This curve, presented in Figure 70, was considered representative of the ADAM buttocks due to similar foam thickness and density between the two parts. A third-order equation was then fitted to the curve. This equation was used in the dynamic response computer program to yield loading results to the drop tower input and impedance results using a sinusoidal forcing function as input.

Although this estimate of buttocks stiffness was adequate for early analysis, measurement of the force-deflection curves for both the large and small buttocks was necessary on arrival of the prototypes (Figures 71 and 72). It is noted that the prototype buttock stiffnesses are considerably less

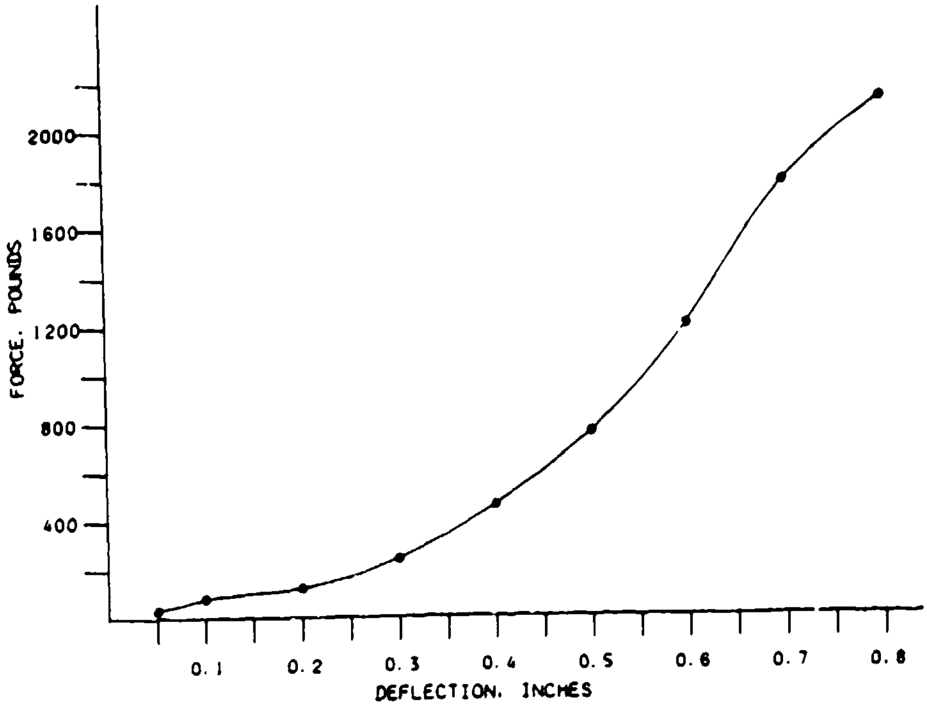



Figure 70. Force Deflection Curve for the LRE Buttocks

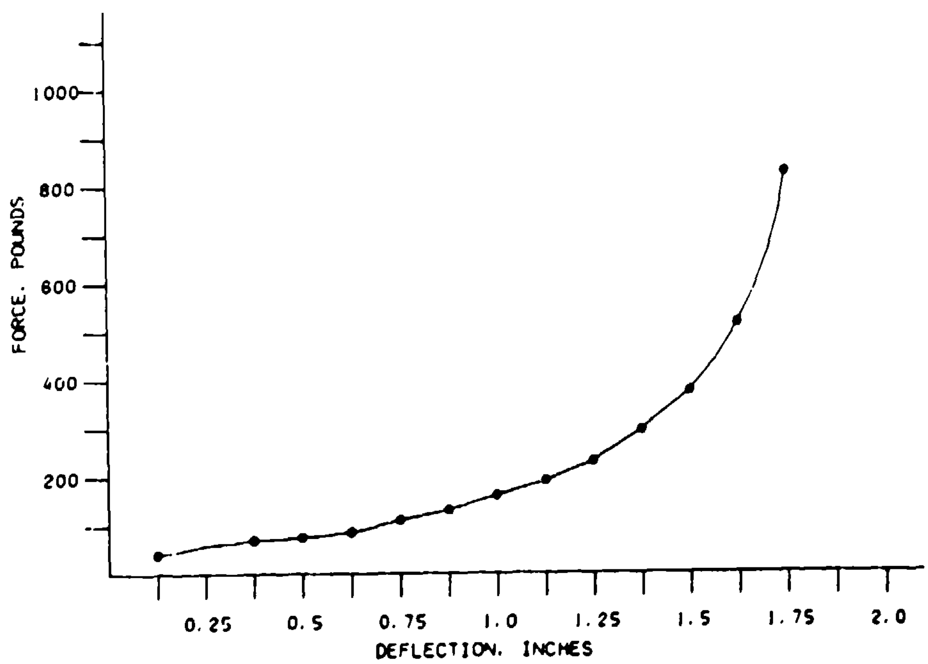



Figure 71. Force Deflection Curve for the Large ADAM Buttocks

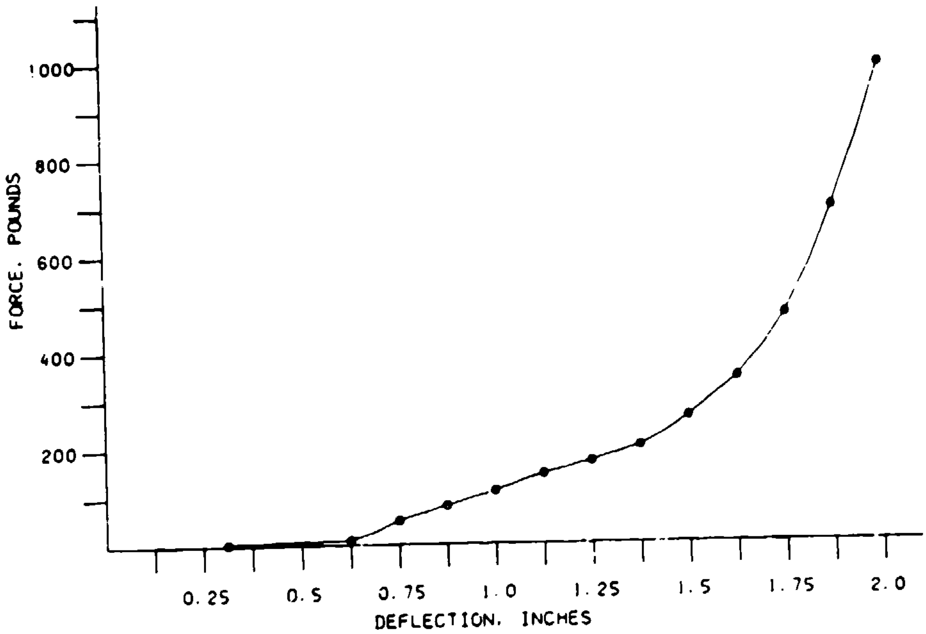


Figure 72. Force Deflection Curve for the Small ADAM Buttocks

than the LRE. These data were then entered into the impedance and dynamic analyses along with prototype weights and other parameters to obtain final estimates of the dynamic properties of the small and large manikins.

2.2.5.4.3. Effective Hinge Point Placement

In order to meet the requirements for ranges of motion in the ADAM torso, axial deflection, torsion, and anterior, posterior, and lateral bending must be incorporated into the ADAM spine. The lumbar spine provides these articulations because evidence indicates that a majority of these motions occur in this region, and the thoracic spine of the ADAM did not provide this ability due to the presence of the viscera box.

Various concepts were reviewed. Finally, a single articulated joint at the base of the spine was chosen to provide the anterior, posterior, and lateral bending motions for the upper torso. In addition, an analysis was undertaken to determine where the single articulated joint should be located to most effectively represent the primary (fore and aft) bending mode of the lumbar spine.

After reviewing spinal motion data received at the outset of the program, no clear cut spinal landmark for placement of a single articulated joint was evident. However, due to possible interference and a need for movement of the complete upper torso, the articulation point was placed in the lower lumbar spine of ADAM.

In order to determine an exact placement point in the lumbar spine, a mathematical analysis of the effective joint height of the Hybrid III butyl rubber lumbar spine was undertaken. The analysis took into account the following system characteristics: the modulus of elasticity, moment of inertia, and length of the butyl rubber spine, as well as the overall length of the spine and the position and weight of the manikin center of gravity.

Figure 73 presents a schematic of the Hybrid III model with the various model parameters pointed out. The representation of the restraint system in the analysis was addressed next. At first, restraint forces were placed at the locations of vertebrates T1, T10, and L3 as in Williams and Belytschko (1981). However, when the model was restrained, upper body displacement was minimized and the placement of the spinal joint became an irrelevant issue. Therefore, the Hybrid III spine was modeled in an unrestrained environment to establish the effects of extreme bending on the effective hinge point of the system.

The analyses of the effective hinge point of the Hybrid III butyl rubber lumbar spine was carried out as follows. First, a known horizontal load was applied to the CG of the model. With this loading in place, the bending of the butyl rubber spine was calculated in terms of linear and angular displacement at the top of the butyl rubber column. Next, these angular and linear displacements were used to calculate the angular and linear displacements at the top of the rigid thoracic spine. Using these values, the rigid part of the spine was then extended back to the vertical intersection point. This intersection point was the theoretical hinge point of the system under the given loading. Figure 73 depicts the initial loading on the spine including angular and linear deflection of the various elements, as well as the extension of the rigid portion of the spine back to the vertical intersection point.

The process for determining intersection points for the model system was carried out for loadings which yielded horizontal deflections to approximately 7 inches. Loadings that caused greater deflections than this at the top of the spine were not examined because of the belief that the seated human would not see deflections beyond this amount in a functioning ejection seat. After the effective hinge points for the manikin model had been determined for a variety of loads, the average hinge point was chosen. In order for the hinge point to be acceptable, it was necessary that the displacement of the system with the hinge point vary by no more than 5 percent from the displacement of the actual butyl rubber system for each of the loadings at the top of the spine.

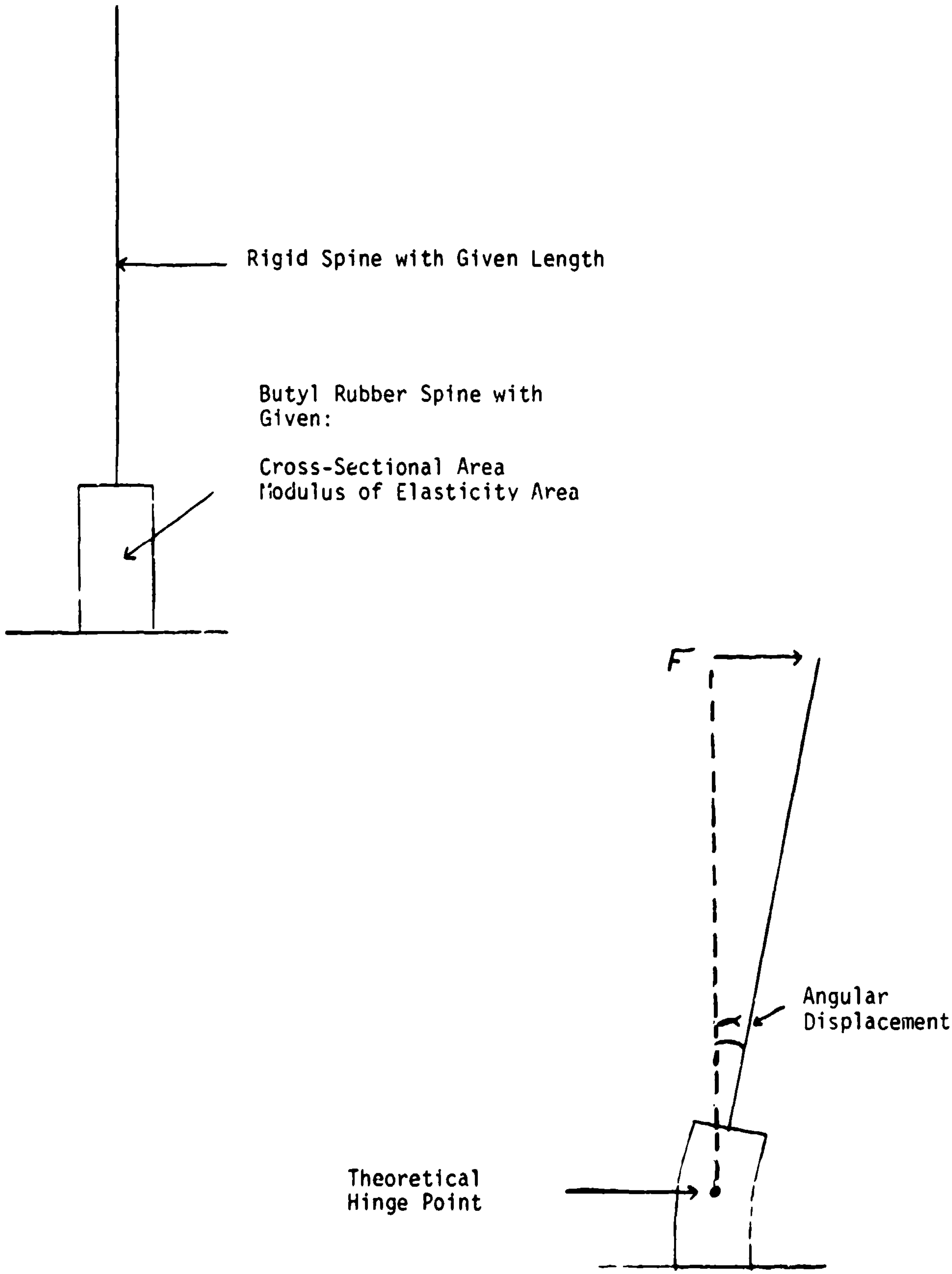



Figure 73. Hybrid III Model

The actual analyses identified that the best point for the lumbar spine articulation point was 4.4 inches above the hip/leg pivot point for both the small and large ADAM. When this positioning of the single articulation joint was carried over to the small manikin, the hinge point was restric: :d to 3.9 inches. Even with the hinge moved to 3.9 inches above the hip/leg pivot point, the largest variation between the model and the Hybrid III was less than 5 percent for both manikins. Because of this, the single articulation point was placed 3.9 inches above the hip/leg pivot point in both the large and small manikins.

2.2.5.4.4. Impedance Sensitivity Analysis

The search for a set of design parameters for the large and small ADAM began by focusing on finding a set of mid-sized parameters which corresponded reasonably well with the ISO standard mean experimental data. This search was carried out through the one dimensional, 3 degree of freedom impedance analysis. The initial studies included the viscera, spine, and the buttocks as degrees of freedom.

Figures 74 and 75 show two analytical computer results that compare relatively well with the ISO mean experimental impedance modulus curve. Also demonstrated is an acceptable correlation of the ISO phase angle data with the mean experimental phase angle data. At the time, the best impedance match came with the following 1 G natural frequencies:

- 6 Hz for the Dynamic Viscera
- 10 to 12 Hz for the Spine
- 8 Hz for the Buttocks

Damping ratios for all three elements were 0.3 $c/_{cc}$.

Since the upper thigh mass tended to affect the response of the system, two different configurations were analyzed to account for the addition of 55 percent of the leg mass to the system. First, the leg mass was added directly to the buttocks and the impedance was calculated. Next, the same amount of leg mass was given its own spring damper system, independent of the buttocks. Both these systems are shown in Figure 76. The figure on the left represents the model system where the additional leg mass is added to the buttocks mass, and the figure on the right represents the model system where the additional upper leg mass is considered as a separate degree of freedom. Figure 77 presents a graph of the impedance modulus curves for the two systems. The curve designated by the triangles represents the system with all of the mass in the buttocks and a natural frequency of 8 Hz. The curve defined by the squares refers to the system where the mass of the legs and

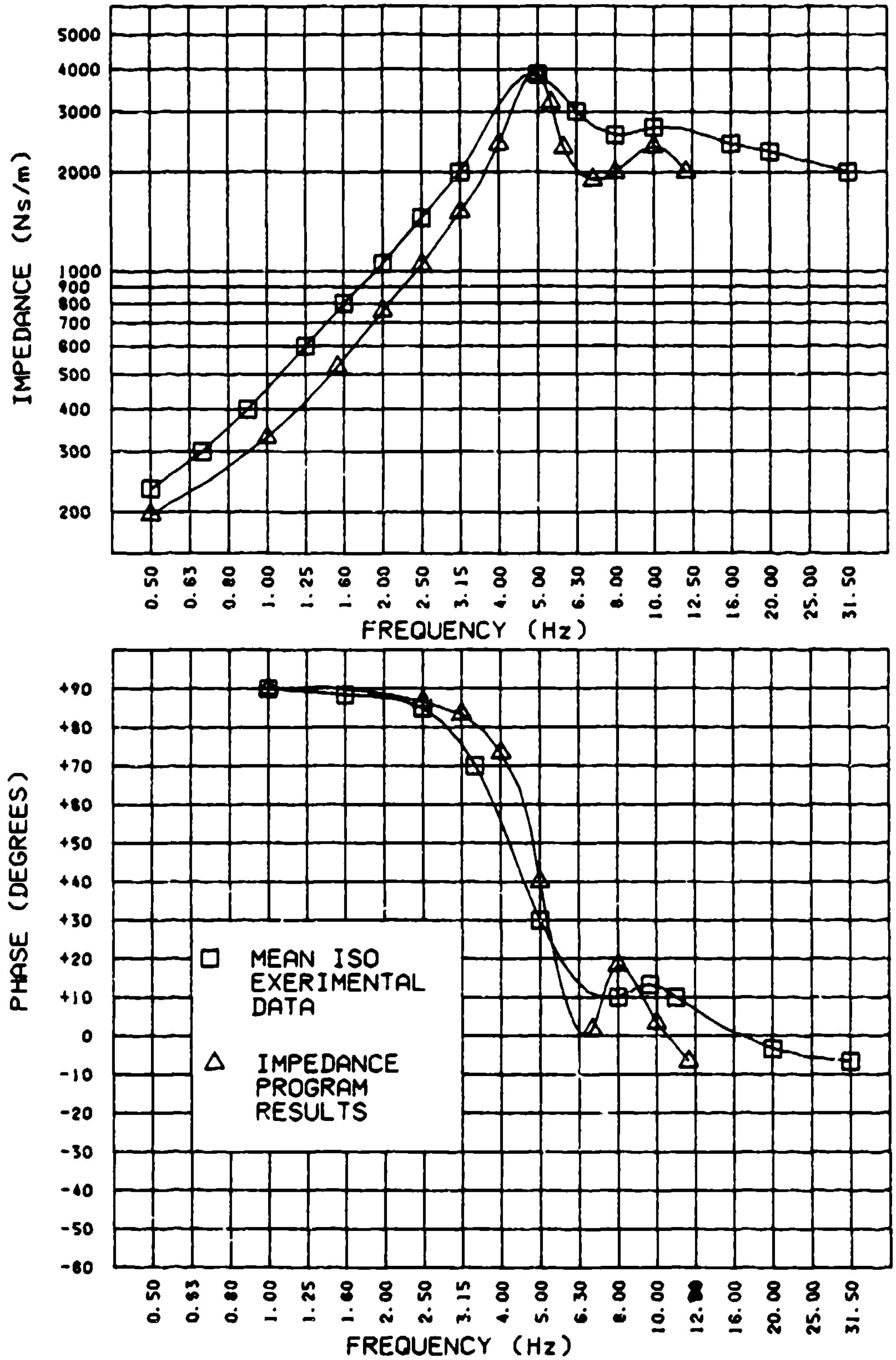



Figure 74. Computer Results Versus ISO Mean Experimental Data (Spine Frequency 10 Hz)

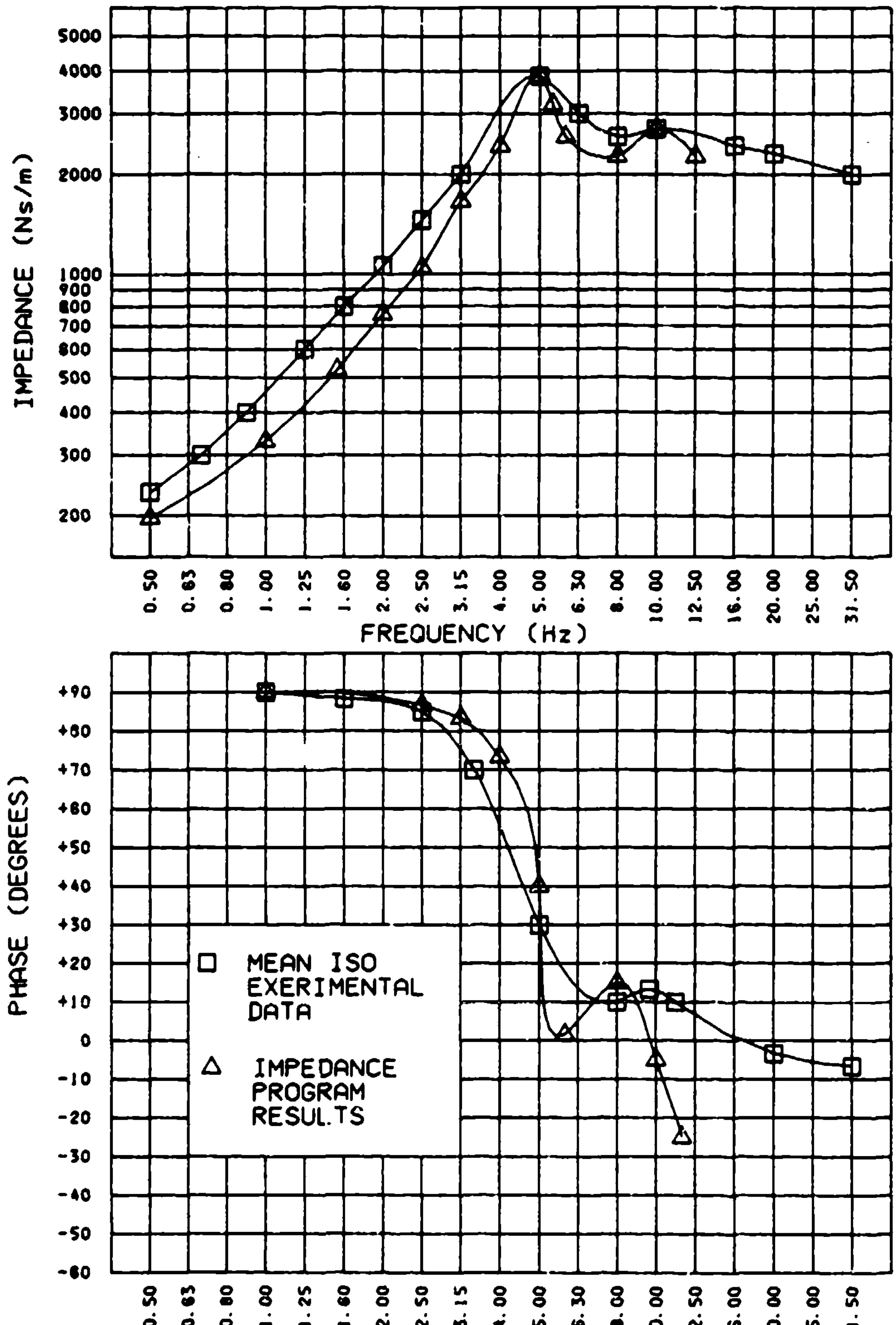



Figure 75. Computer Results Versus ISO Mean Experimental Data (Spine Frequency 12 Hz)

Buttocks and Upper Leg Added Directly

$M_1$ = Seat

$M_2$ = Buttocks + Upper Leg

$M_3$ = Upper Torso - Viscera

$M_4$ = Viscera

Buttocks and Upper Leg in Parallel

$M_1$ = Seat

$M_2$ = Buttocks

$M_3$ = Upper Torso - Viscera

$M_4$ = Viscera

$M_5$ = Upper Leg

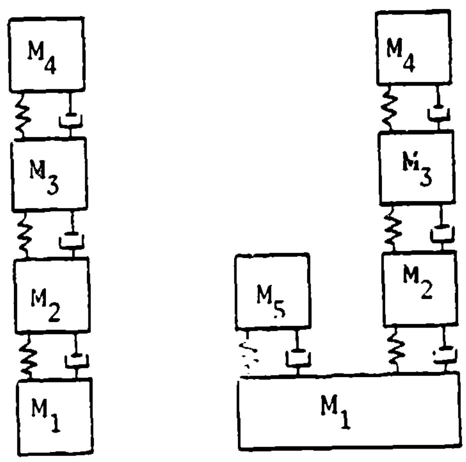



Figure 76. Leg Mass

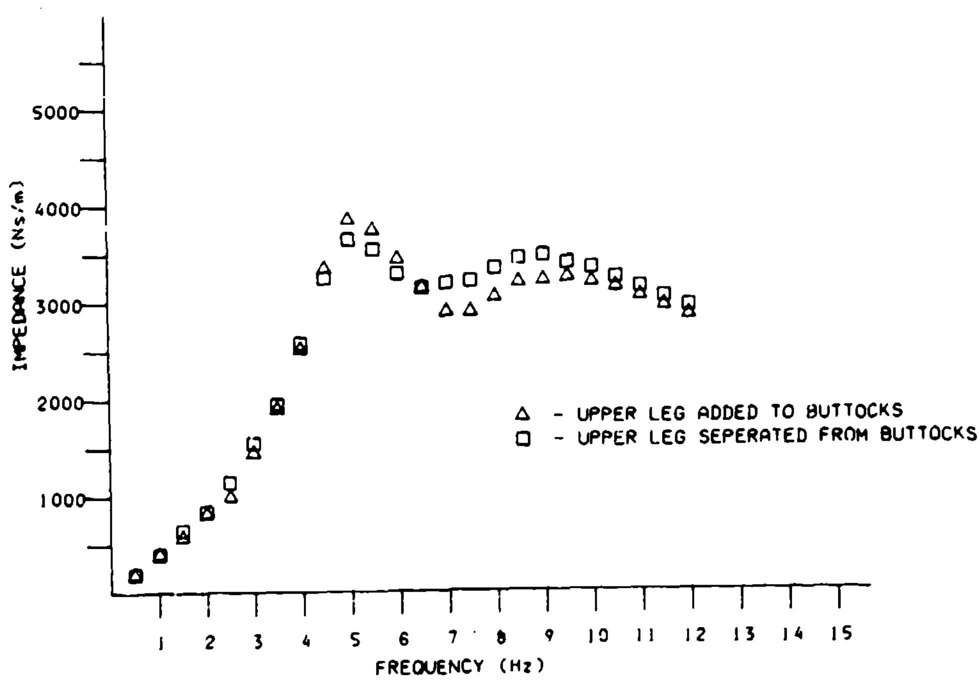



Figure 77. Impedance Modulus Curves for the Addition of Leg Mass to the Impedance Program

buttocks are separated yet still have natural frequencies of 8 Hz and a damping factor of 0.3. The results presented in Figure 77 indicate that both systems are relatively equivalent in their peak frequencies and moduli.

This limited variation between the system with the leg mass incorporated directly into the buttocks and the leg mass separated from the buttocks, as well as the belief that both the buttocks and legs would possess similar stiffness and damping characteristics, led to the conclusion that the leg mass could be added directly to the buttocks in impedance analyses for the manikins.

The inherent nonlinearity in the buttocks led to an exploration of how this nonlinearity would affect the impedance curves of the system. When the nonlinear force-deflection curve of the buttocks was incorporated into the impedance program, the system, under a vibratory loading of ±1/2 G, was never excited enough to cause a large change in buttocks stiffness. Thus, the nonlinear buttocks did not affect the impedance modulus or phase angle. This led to the decision that the entire ADAM impedance model could be simplified by the use of linear springs and dampers. This simplification allowed for the use of the closed form solution for the impedance analysis.

A sensitivity analysis was also performed on the parameters describing the lumbar spine and buttocks. These analyses were performed to ensure that the impedance requirements could be met even if a moderate variation in any of the spinal elements occurred.

As stated earlier in this section, the stiffness and damping characteristics of the buttocks are obtained from the plastisol foam used to fill the buttocks skin shell. These characteristics are inherent to the solidified foam, and no adjustments can be made after filling. Therefore, any deviations in design of the buttocks had to be made up by changes in the lumbar spine parameter or viscera so that the ISO standard would still be met.

The sensitivity analysis determined that the viscera, spine, and buttocks interact to a large extent at both the primary and secondary impedance modulus peaks. The graph in Figure 78 shows the sensitivity of the lumbar spine stiffness in the Gz direction. The results presented in Figure 78 demonstrate that decreasing the stiffness of the lumbar spine lowers the modulus and moves the first peak's frequency lower.

In Figure 79, the sensitivity of the impedance characteristics to damping in the buttock is presented. Both peak's tend to increase when the buttocks damping is decreased. Finally, Figure 80 shows the effect of increasing the buttocks spring constant. The results presented in Figure 80 show little

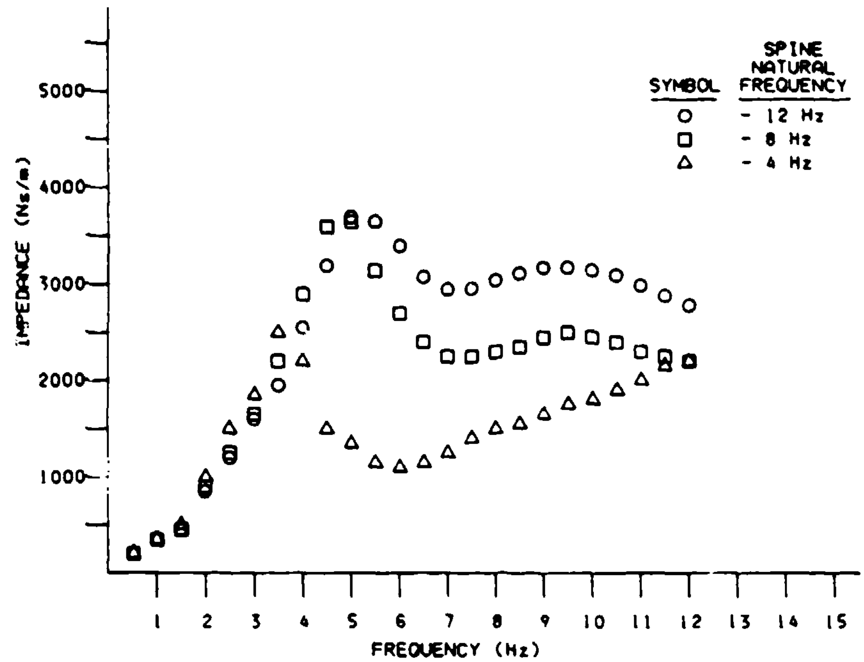



Figure 78. Spine Sensitivity to Spring Rate in Terms of Impedance Modulus

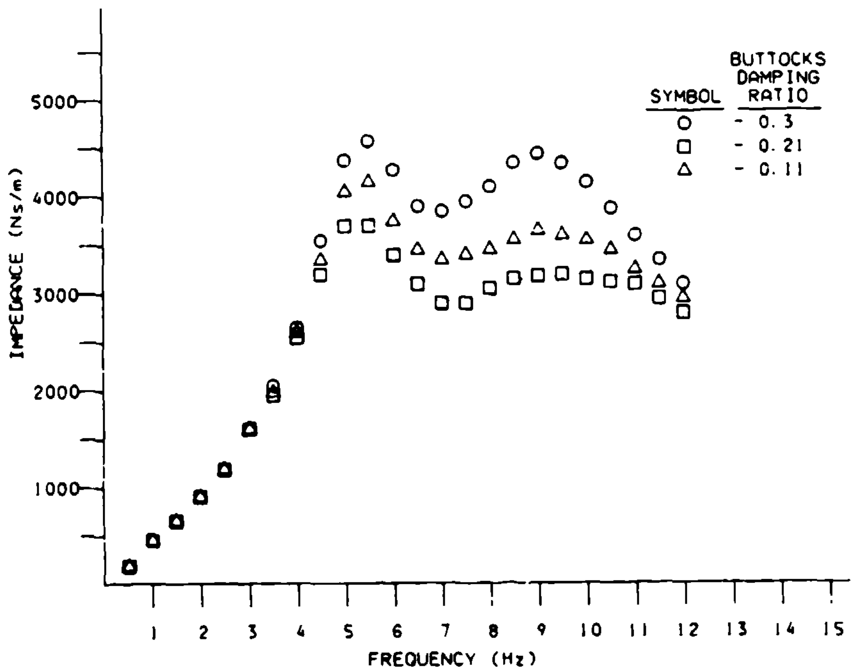



Figure 79. Buttocks Sensitivity to Damping in Terms of Impedance Modulus

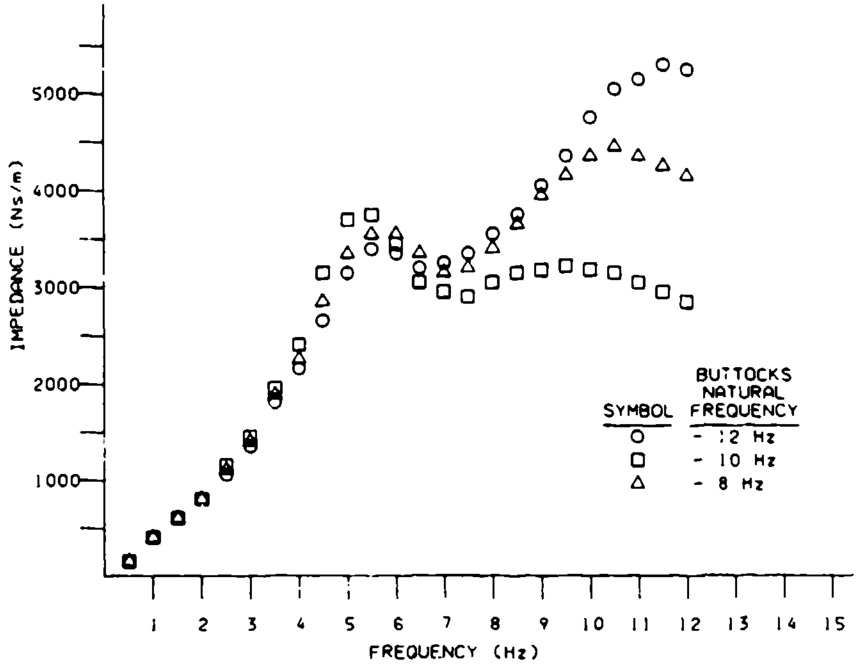



Figure 80. Buttocks Sensitivity to Spring Rate in Terms of Impedance

modification of the initial peak but large changes in the second modulus peak with an increase in the buttocks spring stiffness.

Since changes in the lumbar spine parameters had a significant effort on the secondary peak of the modulus curve and that the buttock parameters did not affect the first peak to any great extent, the main conclusion obtained from the sensitivity analyses was that the impedance requirements could be met even with a moderate deviation of the buttocks nominal stiffness from the estimated value. Any such deviation, however, must be compensated for by the other elements of the system.

Table 24 indicates the parameters for a 50th percentile manikin that yield proper impedance characteristics as set forth in the statement of work. These values were obtained through the sensitivity analyses completed for the system elements.

TABLE 24. MIDSIZE MANIKIN DESIGN DATA

| Element | Spring (N/m) | Damping (NS/m) | $W_o$ (Hz) | $\xi$ |
|---|---|---|---|---|
| Viscera | 18000 | 200 | 6.0 | 0.21 |
| Lumbar Spine | 215000 | 2000 | 12.0 | 0.35 |
| Buttocks | 121900 | 2000 | 7.0 | 0.36 |

The graph in Figure 81 indicates that using these manikin parameters results in an impedance modulus versus frequency curve that closely resembles the mean experimental data published in ISO-5982-1981. Also illustrated in Figure 81 is the correlation of the phase angle versus driving frequency. Although the phase angle is not as accurate as the modulus, it does exhibit the overall trend of the experimental data.

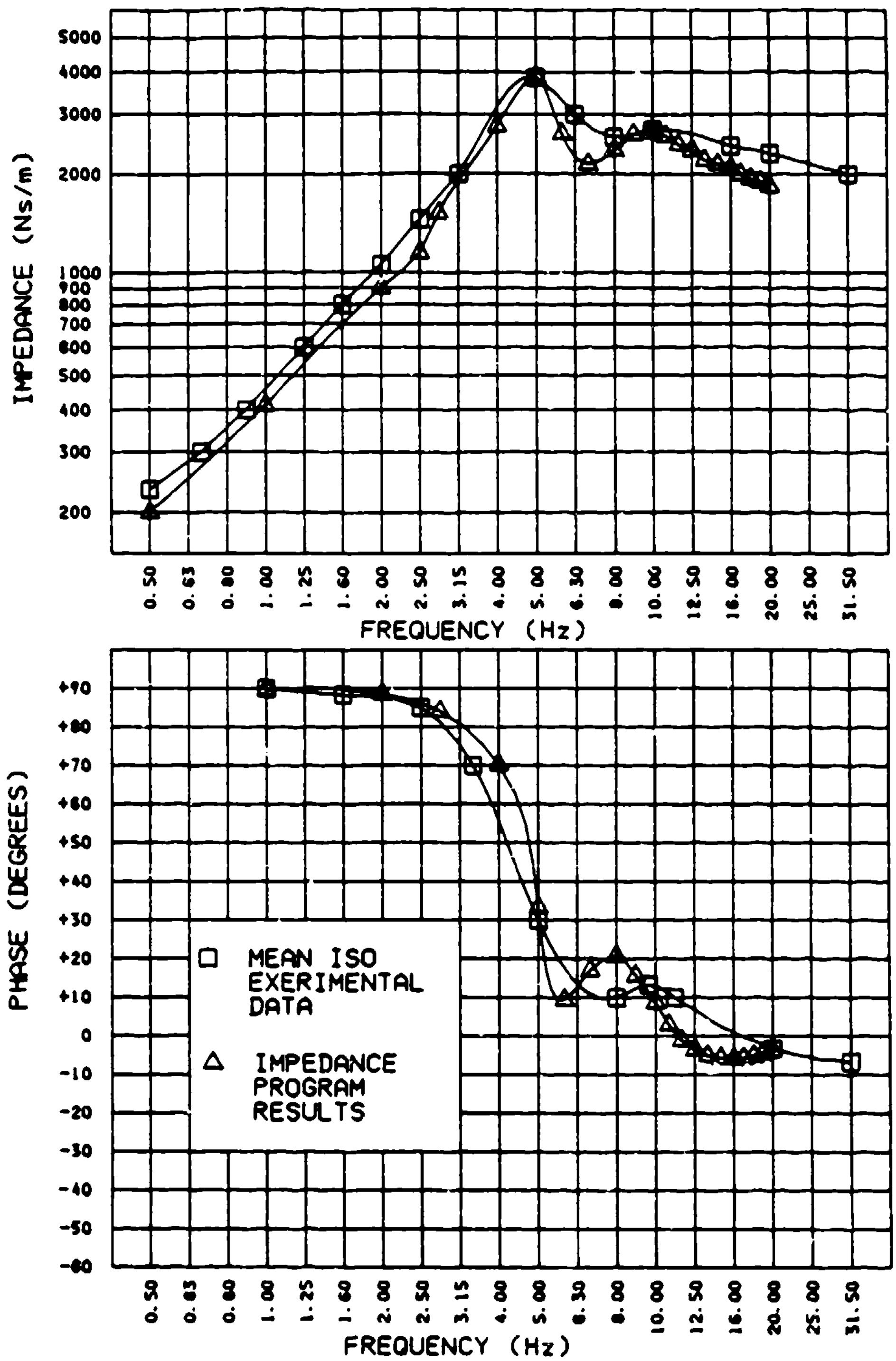


Figure 81. Computer Results Versus ISO Mean Experimental Data

The corresponding parameters for the large and small manikins were derived from direct mass scaling of the midsized manikin. Leaving the natural frequencies and damping ratios of the individual elements intact enabled the derivation of spring and damping constants for the individual elements of both the large and small manikins. The theoretical design parameters for the 3 degrees of freedom small and large manikins are presented in Tables 25 and 26, respectively.

TABLE 25. SMALL MANIKIN DESIGN DATA FOR 3 DEGREES OF FREEDOM

| Element | Spring (N/m) | Damping (NS/m) | $W_o$ (Hz) | $\xi$ |
|---|---|---|---|---|
| Viscera | 12900 | 140 | 6.0 | 0.21 |
| Lumbar Spine | 166600 | 1550 | 12.0 | 0.35 |
| Buttocks | 94000 | 1550 | 7.0 | 0.36 |

TABLE 26. LARGE MANIKIN DESIGN DATA FOR 3 DEGREES OF FREEDOM

| Element | Spring (N/m) | Damping (NS/m) | $W_o$ (Hz) | $\xi$ |
|---|---|---|---|---|
| Viscera | 23200 | 260 | 6.0 | 0.21 |
| Lumbar Spine | 256700 | 2400 | 12.0 | 0.35 |
| Buttocks | 146600 | 2400 | 7.0 | 0.36 |

Although the parameters listed in Tables 25 and 26 should yield impedance versus driving point frequency curves that will meet the requirements established by ISO-5982-1981, the ability to realize these parameters in a functional configuration proved to be an even more formidable task. As was stated earlier, the viscera degree of freedom was dropped from the manikin design. After additional analyses were conducted, the conclusion was drawn that the ISO standard could not be met completely with the viscera degree of freedom removed. The impedance characteristics of the final manikin design which did not include the viscera degree of freedom are presented and discussed later in this section.

2.2.5.4.5. Dynamic Response Sensitivity

Analyses were performed on the small manikin data to define a nominal set of design parameters that would produce appropriate correlation with experimental human response data similar to that presented in Figure 82. The parameters given in Table 27 are the results of these analyses and were the target values for the small ADAM detail design. The nonlinear buttocks stiffness that was used initially was developed from compression test data of the LRE buttocks (see Figure 70).

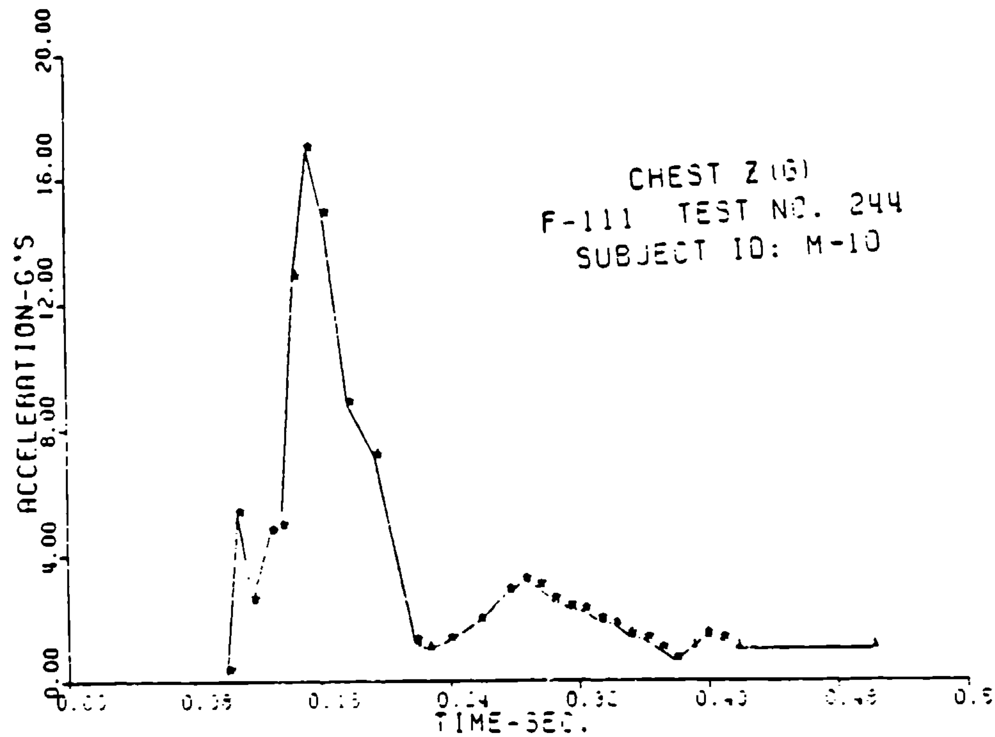
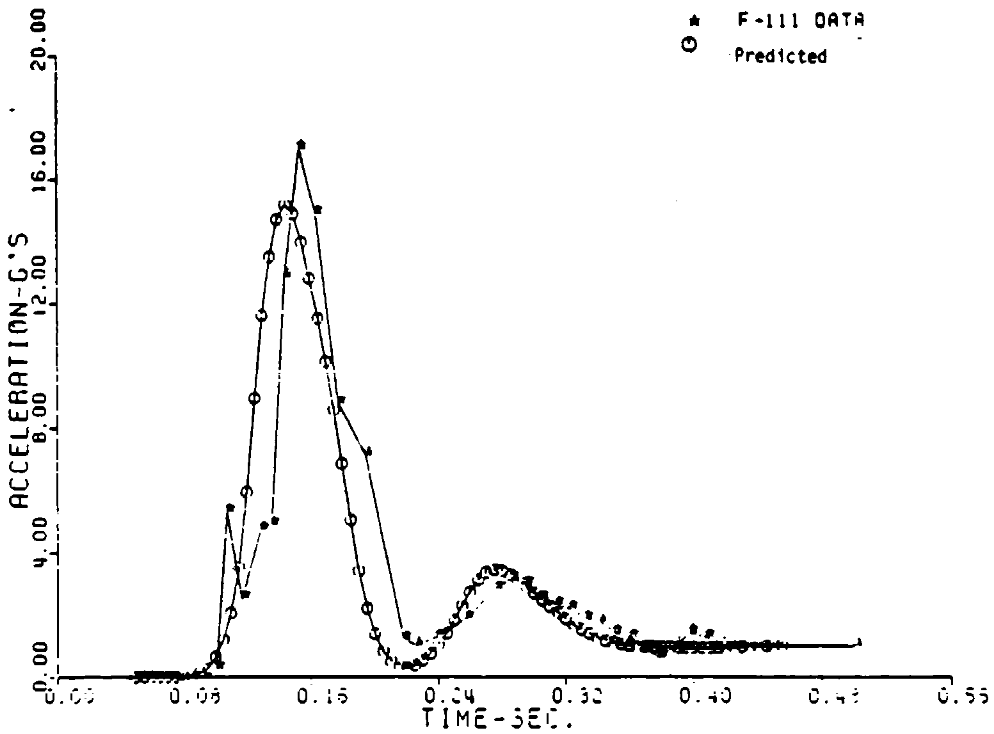



Figure 82. F-111 Torso Acceleration Data Used in Small ADAM Correlation Efforts

TABLE 27. NOMINAL VALUES OF MASS, DAMPING, AND STIFFNESS FOR SMALL ADAM DESIGN

| Element | Mass (kg) | Damping (NS/m) | Spring (N/M) |
|---|---|---|---|
| Viscera | 9.07 | 205 | Locked Out |
| Lumbar Spine | 20.20 | 2212 | 116000 |
| Buttocks | 19.20 | 920 | Nonlinear |

The calculated response for this set of nominal values was compared with experimental data taken from Brinkley, Raddin, Hearon, McGowan, and Powers (1980). In this report, a series of drop tower tests were run on 22 test subjects. Each subject had time history acceleration data measured at numerous points on his body, and these graphs were presented within the report. For comparison purposes, the data of the test subject that most closely resembled the small ADAM in overall height and weight were used. Figure 83 shows the comparison between the test subject chest acceleration data and the calculated response analysis for the ADAM torso. The correlation was acceptable except for discrete perturbations in the F-111 data. The belief was that these perturbations are tied to carriage characteristics or limb motions that were not modeled in the response analysis.




| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 920. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 116000. |
| M (3) = 9.07 | C (3) = 205. | K (3) = Locked Out |

Figure 83. Comparison of F-111 Torso Response with Calculated Response for Nominal Design

In order to make use of the data in Table 27 for prototype design, it was necessary to determine how accurately these parameters must be modeled. Therefore, a sensitivity analysis was performed for each variable independently about the nominal case. The results of these sensitivity analyses are as follows:

- <u>Spine Spring Stiffness</u>: The spine spring stiffness was varied by a value of ±10 percent. This value was selected because of the relative ease with which a mechanical spring can be designed and manufactured to produce the stiffness required for this variation. The results of these variations are given in Figures 84 through 86. As can be seen from the results, variation in spring stiffness does not significantly alter the system response.

- Spine Damping Coefficient: The spine damping coefficient was varied from 40 percent critical damping to 80 percent critical damping. The results of this analysis are shown in Figures 87 through 89. These figures indicate that a damping coefficient in the range of 60 percent to 80 percent critical damping should provide an acceptable system response characteristic.

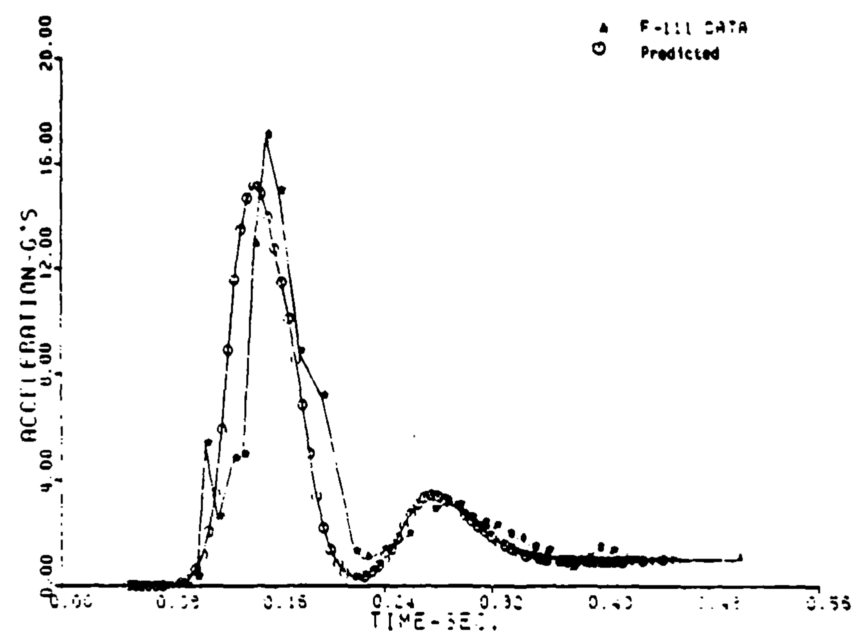
- Buttocks Spring Stiffness: The buttocks spring stiffness was varied from 0.6 times the nominal stiffness to 5.0 times the nominal stiffness. The reason for investigating this large variation in the stiffness was the uncertainty with regard to the buttocks foam stiffness. As can be seen in Figures 90 through 92 in which the variations in buttock response are compared with the nominal case, the sensitivity to buttocks spring stiffness is negligible.

- Buttocks Damping Coefficient: The buttocks damping coefficient was varied over a range of 10 percent critical damping to 50 percent critical damping. The results of this analysis are shown in Figures 93 through 95 and indicate no change in response with a change in buttocks damping. This result is to be expected because of the small displacement and velocity associated with the buttocks degree of freedom.

An overview of the dynamic response sensitivity analyses shows that none of the parameters, over the ranges evaluated, had any significant effect on dynamic response analysis of the small ADAM system. Thus, the small and large manikins should successfully replicate the human response to impact loadings.

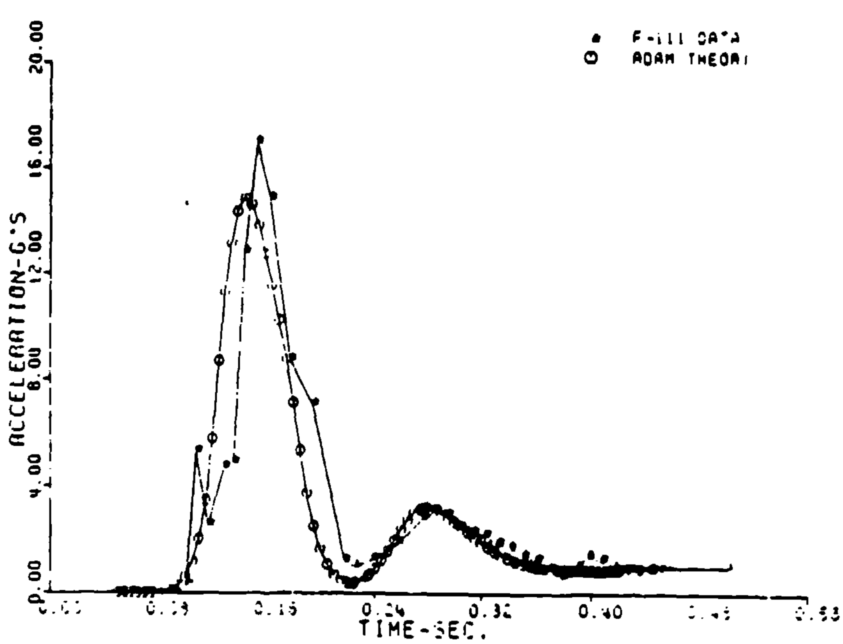
In addition to the quantitative analysis of spring and damper sensitivity, a qualitative evaluation was undertaken to determine the effect of system friction. Friction will truncate the amplitude of response but will not shift the frequency of the response waveform. Based on this determination, it is assumed that friction values of less than 5 percent of the dynamic loading will not significantly impact the system response. In order to maintain friction values below this percentage, emphasis was placed on designing a lumbar spine spring damper system that minimized potential sources of friction.

The final design of the small and large ADAM systems is presented in Section 2.2.5.6. Impedance and response analysis for the actual system (i.e., using the actual ADAM buttocks parameters) is also presented.



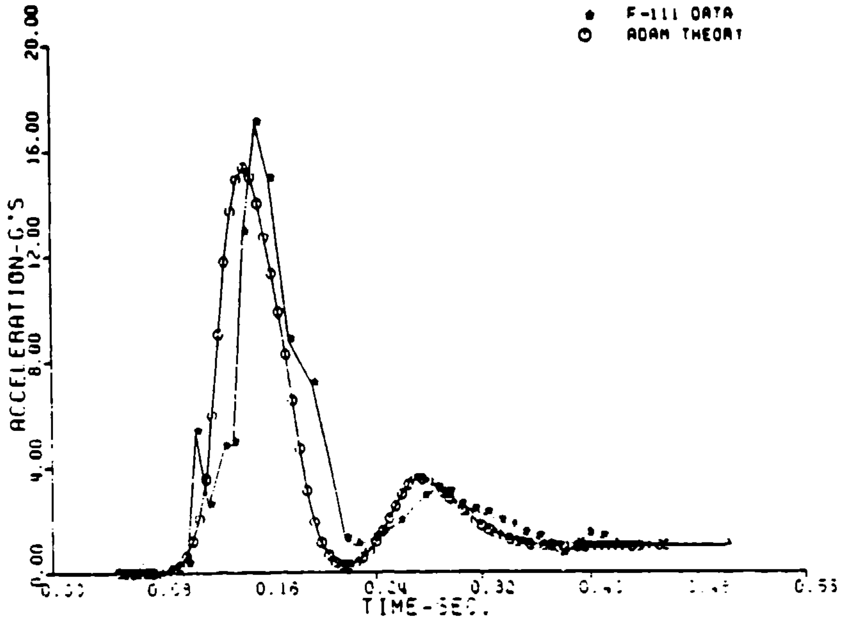

| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 920. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 116000. |
| M (3) = 9.07 | C (3) = 205. | K (3) = Locked Out |

Figure 84. Spine Stiffness Sensitivity Analysis, Nominal Value



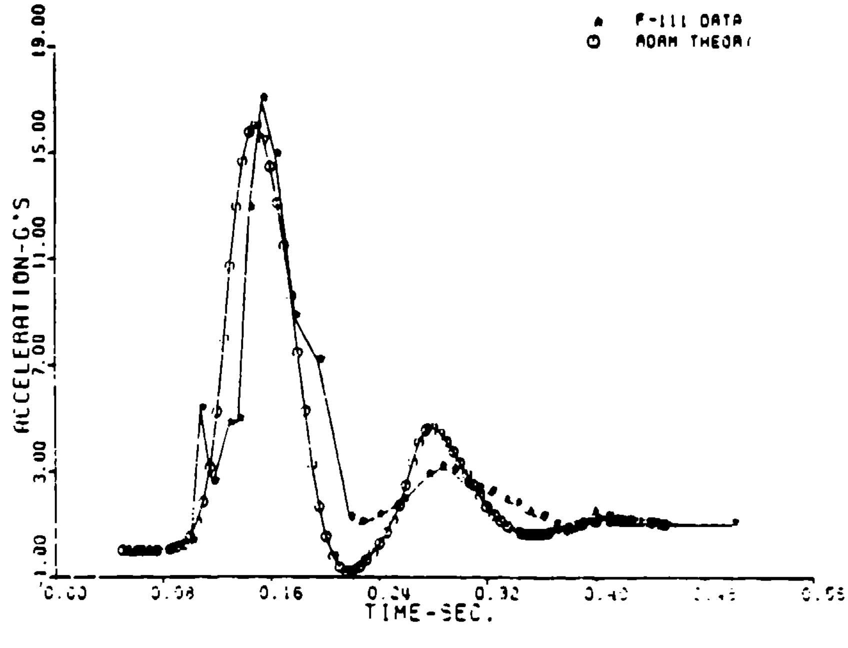

| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 920. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 104000 |
| M (3) = 9.07 | C (3) = 205. | K (3) = Locked Out |

Figure 85. Spine Stiffness Sensitivity Analysis, Nominal Case Reduced 10 Percent



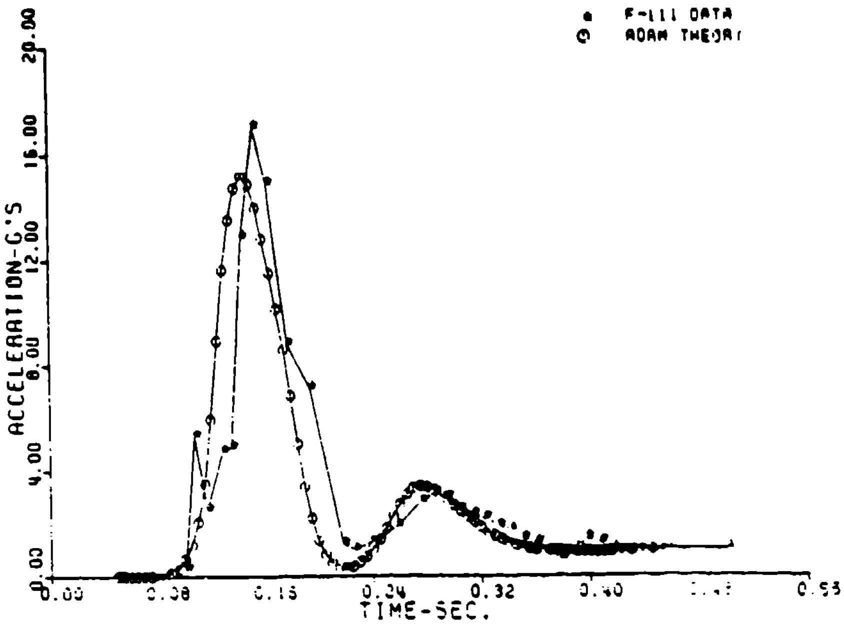

| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 920. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 127000. |
| M (3) = 9.07 | C (3) = 205. | K (3) = Locked Out |

Figure 86. Spine Stiffness Sensitivity Analysis, Nominal Case Increased 10 Percent



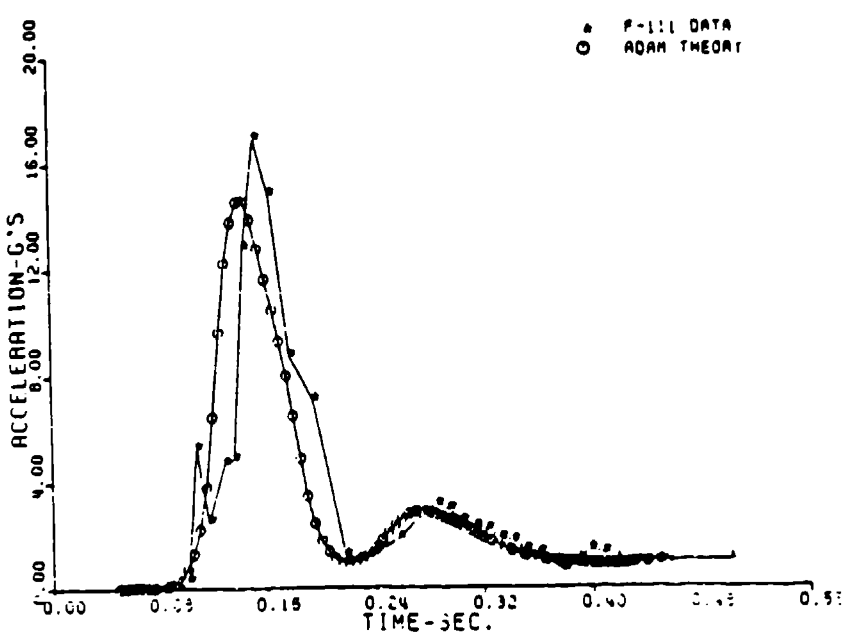

| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 920. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 1475. | K (2) = 116000 |
| M (3) = 9.07 | C (3) = 205. | K (3) = Locked Out |

Figure 87. Spine Damping Sensitivity Analysis, 40 Percent Nominal Case Critical Damping



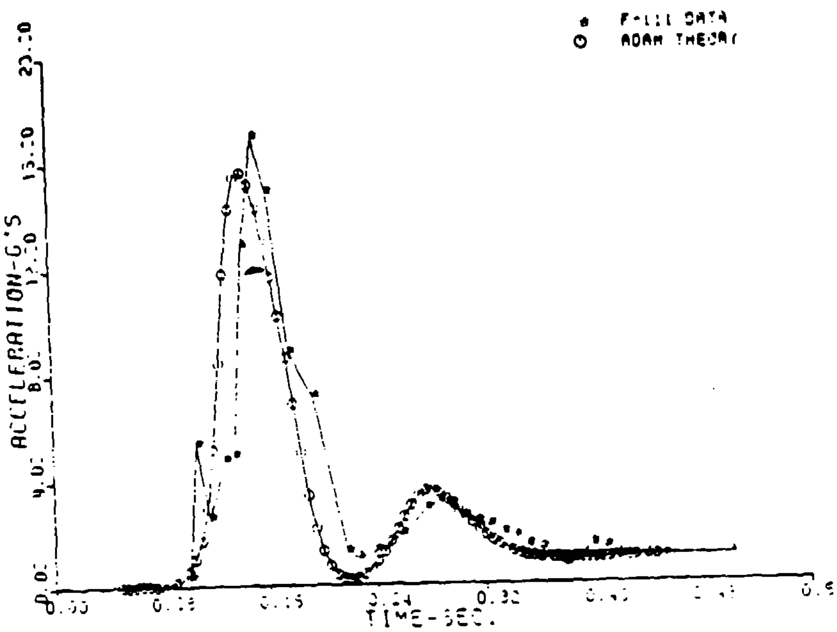

M (1) = 19.28 C (1) = 920. K (1) = Nonlinear
M (2) = 20.23 C (2) = 2212. K (2) = 116000
M (3) = 9.07 C (3) = 206. K (3) = Locked Out

Figure 88. Spine Damping Sensitivity Analysis, 60 Percent Nominal Case Critical Damping




M (1) = 19.28 C (1) = 920. K (1) = Nonlinear
M (2) = 20.23 C (2) = 2960. K (2) = 116000
M (3) = 9.07 C (3) = 206. K (3) = Locked Out

Figure 89. Spine Damping Sensitivity Analysis, 20 Percent Nominal Case Critical Damping

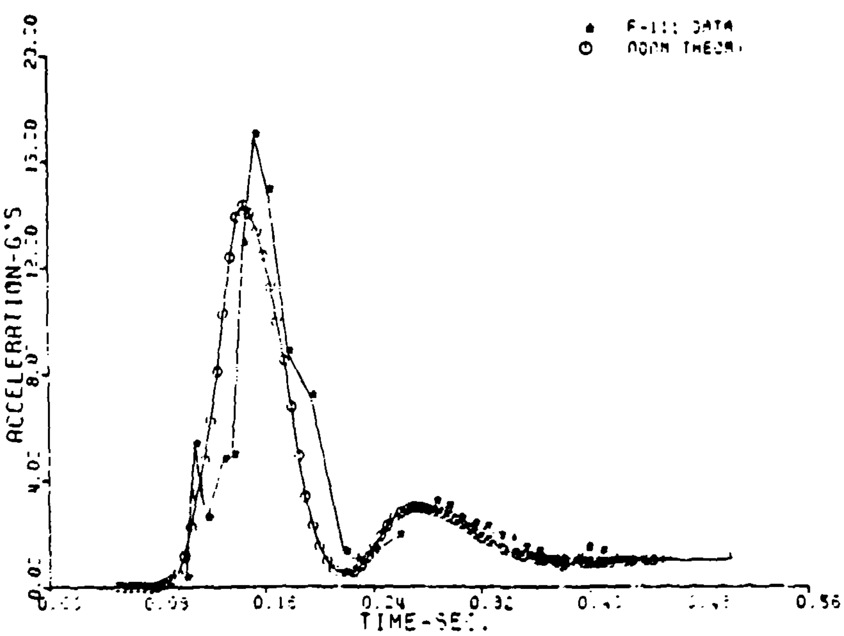



M (1) = 19.28
M (2) = 20.23
M (3) = 9.07

C (1) = 920.
C (2) = 2212.
C (3) = 205.

K (1) = Nonlinear
K (2) = 116000
K (3) = Locked Out

Figure 90. Buttocks Stiffness Sensitivity Analysis, 0.6 Times Nominal Case

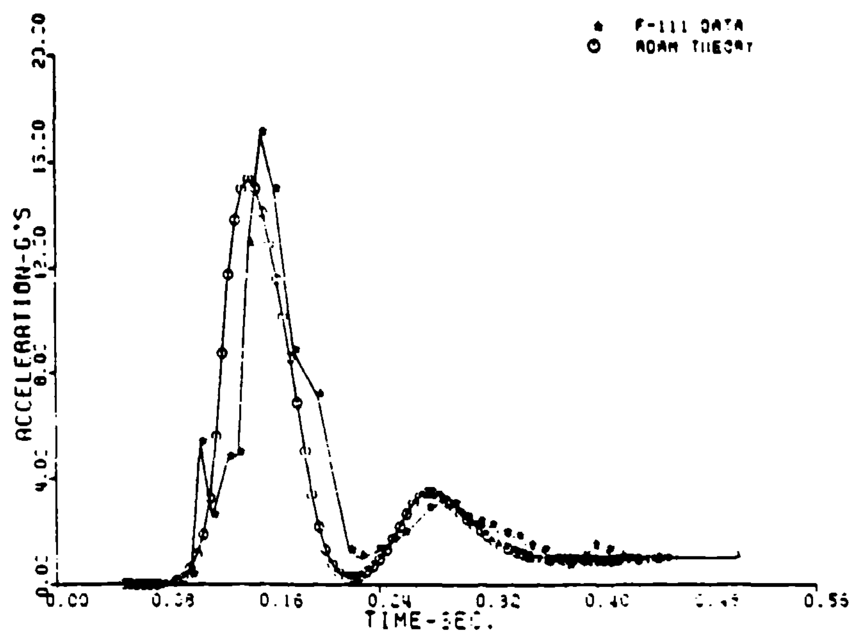



M (1) = 19.28
M (2) = 20.23
M (3) = 9.07

C (1) = 920.
C (2) = 2212.
C (3) = 205.

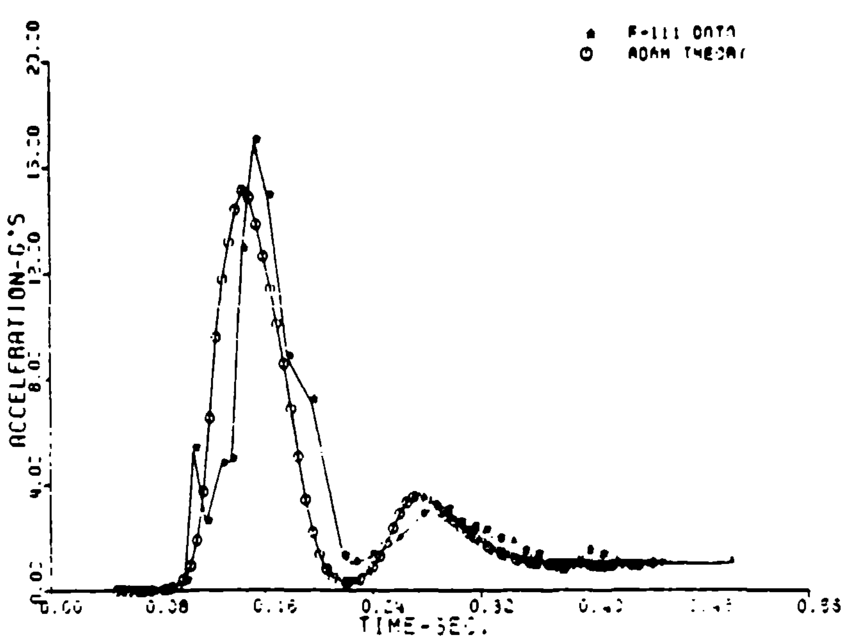
K (1) = Nonlinear
K (2) = 116000
K (3) = Locked Out

Figure 91. Buttocks Stiffness Sensitivity Analysis, Nominal Case



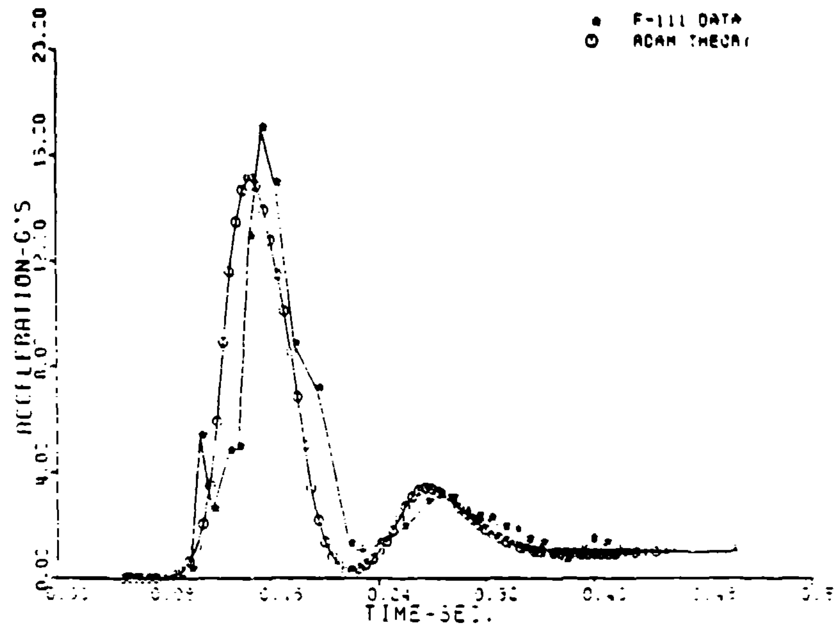

| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 920. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 116000 |
| M (3) = 9.07 | C (3) = 206. | K (3) = Locked Out |

Figure 92. Buttocks Stiffness Sensitivity Analysis, 5.0 Times Nominal Case



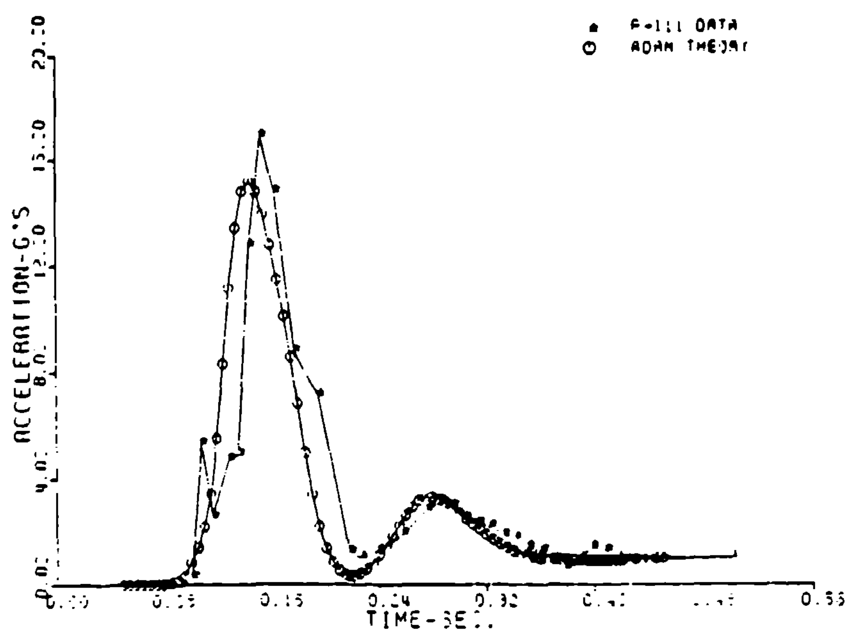

| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 308. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 116000 |
| M (3) = 9.07 | C (3) = 206. | K (3) = Locked Out |

Figure 93. Buttocks Damping Sensitivity Analysis, 10 Percent Critical Damping




| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 920. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 116000 |
| M (3) = 9.07 | C (3) = 205. | K (3) = Locked Out |

Figure 94. Buttocks Damping Sensitivity Analysis, 30 Percent Critical Damping




| | | |
|---|---|---|
| M (1) = 19.28 | C (1) = 1580. | K (1) = Nonlinear |
| M (2) = 20.23 | C (2) = 2212. | K (2) = 116000 |
| M (3) = 9.07 | C (3) = 205. | K (3) = Locked Out |

Figure 95. Buttocks Damping Sensitivity Analysis, 50 Percent Critical Damping

### 2.2.5.5. Spinal System Testing

Completion of a number of tests was necessary to develop and validate the final ADAM spinal system. The main objectives of these test were as follows:

- To obtain both a force-deflection curve of the spinal system and the natural frequency of the system with the upper torso weight.
- To determine the amount of coulomb damping in the unfilled system.
- To determine the effect of an offset load on the spinal system force deflection characteristics, response, and natural frequency.
- To investigate the effects of varying both the accumulator size and orifice (gap width) in the filled system.
- To identify the viscosity range needed for proper operation of the system as well as specific fluids for a given temperature range.

This section presents the test apparatus, procedures, and results associated with these spine system tests.

#### 2.2.5.5.1. Pretest Analyses

Prior to spinal testing, a series of pretest analyses were run to determine the design of the test system apparatus, the type of instrumentation needed, and initial sizing of various test system parameters. Specific analyses that were carried out included spinal system friction estimations, both vertical and lateral natural frequency analyses, computer generated damping decay curves for a one degree of freedom system, and sizing of the initial spine piston and accumulator.

In order to keep friction of the spinal system to a minimum, it was necessary to eliminate as many moving contact points as possible. Figure 96 illustrates a total of three moving contact points within the ADAM spine: one where the piston rod passes through the cap O-ring, and two where the outer spine sleeve passes over teflon bearings.

In the friction analysis, the piston rod was assumed to have a finish of 15 micro inches, the total friction of the system was taken as the sum of the frictions from all three contact points, and both a

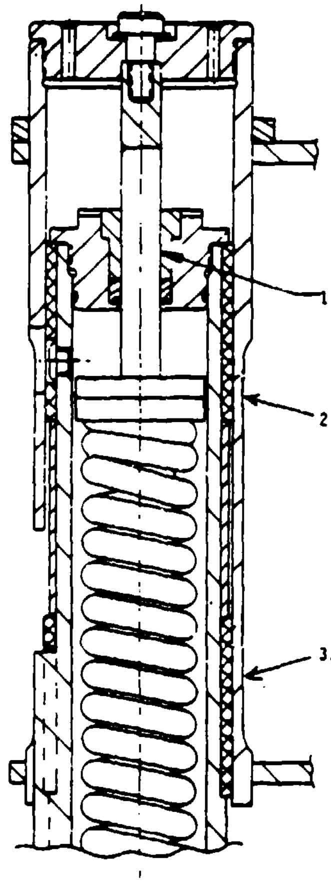



Figure 96. Three Moving Contract Points within the ADAM Spine

best and worst case estimate were made for the overall system. For the worst case, the static friction (the force to start the spine moving) was 6 pounds; and the dynamic friction (which measures the force required to keep the system moving) was approximately 4 pounds. For the best case, the static friction was 3 pounds, and the dynamic friction was 2 pounds. It was determined that friction less then 15 pounds should be adequate to allow relatively free movement of the entire spinal system.

The frequency analyses were broken into two parts: vertical and lateral analyses. The dynamic response analysis determined that a 10 Hz vertical spinal frequency was desired; however, frequencies up to 13 Hz could be tolerated. From a simple analysis, the natural frequency of the spinal system in the Z direction was determined to be 10 Hz.

Two cases were run for the lateral natural frequency of the spinal test system. In the first case, the system was assumed to be a cantilevered hollow rod with a point mass of 64.6 pounds acting on the end of the rod. From this case, a natural frequency of 27.5 Hz was derived for the unloaded system.

Next, the analysis was repeated, replacing the 64.6-pound point force with one of 464.6 pounds, the weight capacity of the test system. This case yielded a lateral natural frequency of 6.3 Hz. This analysis determined that, at higher loadings, lateral motion could tend to interfere with the spine test system primary mode. Therefore, in order to assure clean vertical response data, the system should not be fully loaded when system damping is being tested.

As a visual referencc for use during the dynamic drop tests, computer generated graphs for a one degree of freedom system were developed for theoretical system dampings from 10 percent to 70 percent of critical. These graphs were used to visually estimate the system damping of the spine as various fluids were used in the spring damper unit and to obtain an under standing of how the damping signature should appear. Figures 97 to 99 illustrate three of these curves.

A viscous damping analysis was undertaken to obtain a rough estimate of the gap width, inner cylinder diameter, and piston diameter needed to obtain a 60 percent critically damped spinal system for the small manikin. Within this analysis, the following assumptions were made: MIL-H-5606E hydraulic fluid was used, there was a laminar flow around the piston, the fluid was incompressible, and the inner cylinder diameter was held constant at 1.500 inches. Using these assumptions, the following equation for determining the gap width and piston diameter of the system was developed:

$$C = \frac{6\mu L\, F R_i}{(R_O - R_i)^3\, \Delta P}$$

where

$\mu$ = Fluid Dynamic Viscosity
$L$ = Thickness of the Piston
$F$ = Force Applied to the Piston
$R_i$ = Piston Radius
$R_O$ = Cylinder Inside Radius
$\Delta P$ = Difference in Fluid Pressues in Upper and Lower Cylinder Chambers
$C$ = Damping Value
$R_O - R_i$ = Gap Width of the Cylinder System

This equation gave a gap width of 0.0015 inch and a piston diameter of 1.497 inches for the small spinal system. These values were used as initial design values for the experimental spring/damper spine.

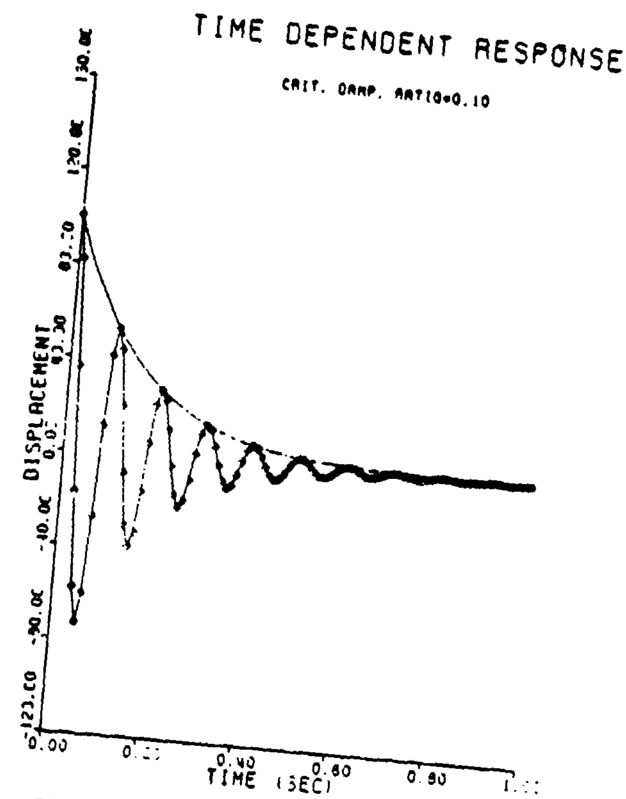



Figure 97. Time Dependent Response

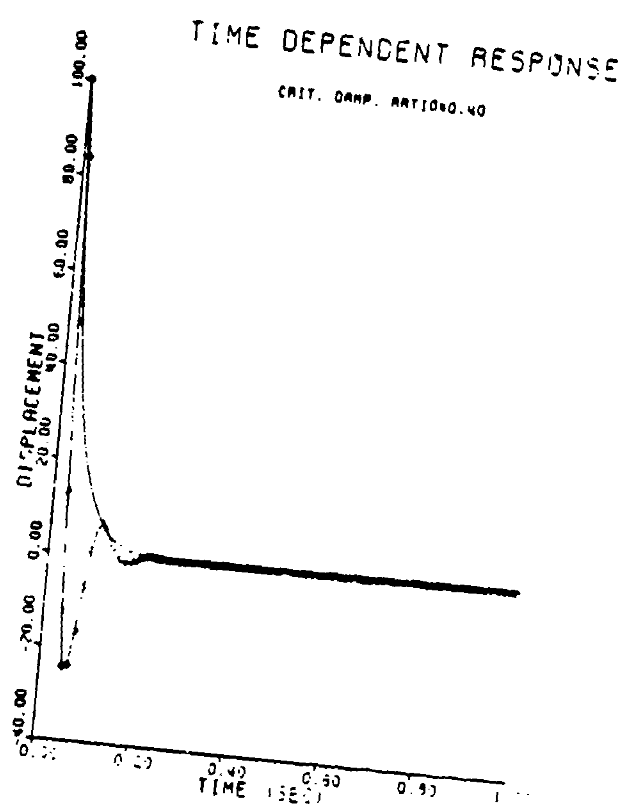



Figure 98. Time Dependent Response

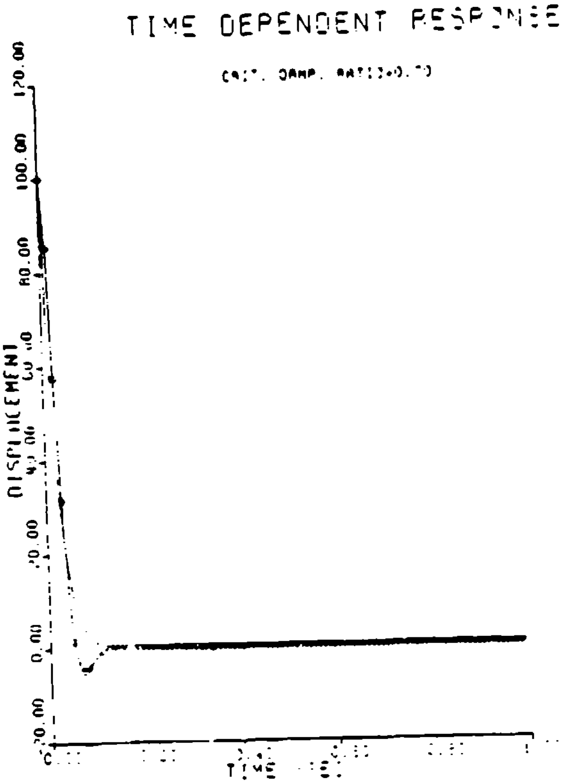



Figure 99. Time Dependent Response

The initial sizing of the accumulator was based upon a 30 percent decrease in the volume of air of the accumulator in its worst case. Earlier work determined that, at its worst case, the piston rod would use a total of 0.18 cubic inches of additional volume. From these numbers, an initial volume of 0.60 cubic inches was derived for the accumulator.

2.2.5.5.2. Mechanical Test Apparatus

The mechanical spine test system is depicted in Figure 100. The four basic components of this system are: the base support system, the prototype spinal assembly, the weight support system, and the A frame winch system (not shown).

The base support system for testing in the Z direction is a system designed to both support the prototype spinal assembly and limit lateral motion of the test assembly. This system consisted of the support frame welded to a base plate with guide rods and base block which was bolted together and pinned to the lower half of the prototype spine assembly. By specifying close tolerances between the prototype spinal system and the base block, system slop was kept to a minimum, allowing travel in only the vertical direction. The support frame was bolted to the floor to prevent

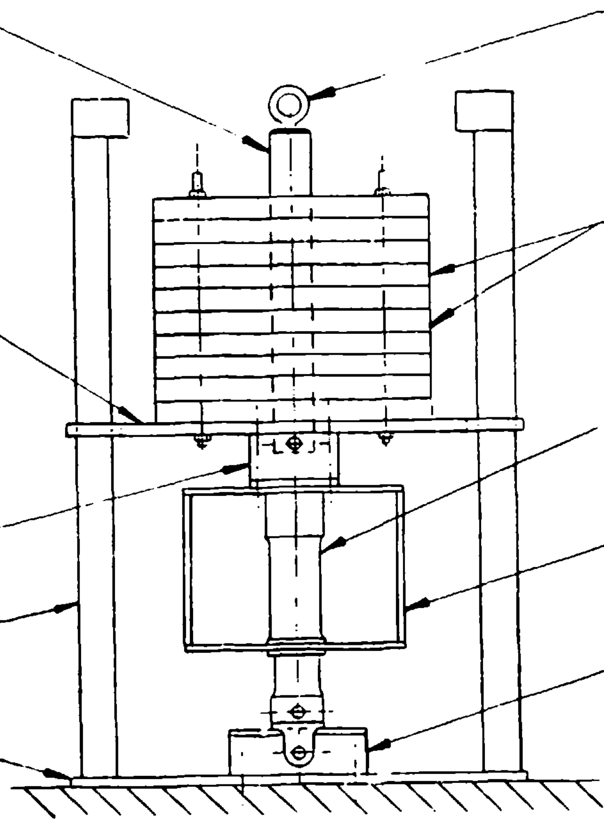



Figure 100. Mechanical Spine Test System

extraneous acceleration readings and was equipped with vertical stanchions to protect the spinal system from breaking should the system begin to resonate laterally.

The prototype spinal assembly consisted of the entire spine from the base pivot to the outer cylinder cap including the top, bottom, and side plates of the viscera box. The mechanical spring and hydraulic damper system were situated inside the inner spine cylinder.

The weight support system sat on top of the prototype spinal assembly and was made up of the adapter block, the weight plate, and the support rod, which were all bolted or pinned together to the upper half of the prototype spine assembly. The weight plate provided a base where 40-pound weights could be placed and a lateral support that would be restrained by the vertical stanchions should the spinal system fail during testing. The support rod provided a centering mechanism to ensure that the center of gravity of the weights added to the system were placed directly over the spine center line. A removable eye bolt that connected the entire test system to the winch was mounted in the top of the support rod.

The A frame winch system was comprised of the A frame, the winched cable system, and a quick release mechanism. The A frame itself was approximately 6 feet high and 8 feet wide and was used as an outer support frame for the winch system that sat inside it. The winch contained 1/4-inch steel braided cable and raised or lowered the spinal system in approximately 3/8-inch increments. Finally, a mechanical quick release mechanism was added in series with the steel cable and the eye bolt.

Other hardware used during the spinal system tests included a 2-foot long beam for offset loading tests and ten 40-pound weights for static deflection tests and the dynamic response tests.

2.2.5.5.3. Test Instrumentation

2.2.5.5.3.1. Static Tests

The instrumentation for the static spine tests included a vertical displacement gauge with accuracy to 0.001 inch for measuring the travel of the spinal system, a 1000-pound load cell, and a digital voltmeter to measure the output of the load cell.

2.2.5.5.3.2. Dynamic Tests

The instrumentation used in the dynamic spine tests included either a 50 G or 10 G accelerometer, an adjustable filter, a Dynascan 1650 tri-output power supply, a Nicolet 3091 storage oscilloscope, and a Hewlett-Packard (HP) 7004B X-Y recorder. In early testing, data were taken with a 200 Hz filter and a 30 Hz filter in order to determine the effect of the filter on the recorder signature.

It was found that the 200 Hz filter was not suitable for the measurements being taken; therefore, a 30 Hz filter was used in a majority of the tests. Figure 101 illustrates the schematic for the dynamic response test electronics. The accelerometer signal was passed through the filter to the Nicolet storage scope. From here, the data could be passed to the HP X-Y recorder and printed.

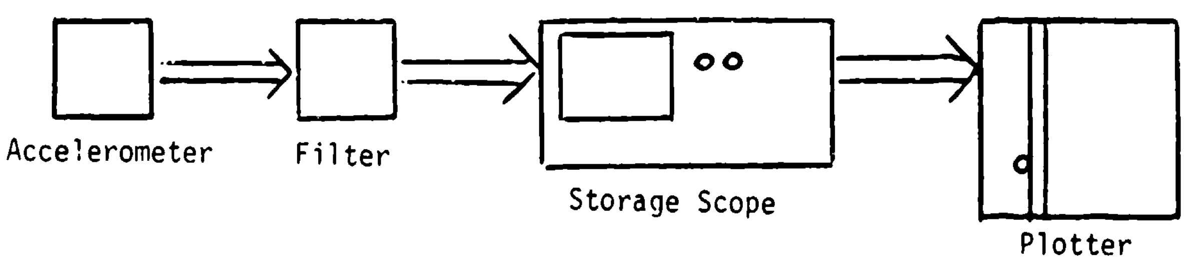



Figure 101. Dynamic Response Test Electronics

2.2.5.5.4. Test Procedures and Results

This section covers the spinal test procedures. Examples of data gathered during these tests will not be presented in this section; however, the results will be discussed in the section describing test results.

2.2.5.5.4.1. Static Calibration

Static calibration in the extension and compression direction was conducted for the unfilled spine. First, the system was calibrated in the compression direction by adding ten 40-pound weights, one at a time, and taking readings with the vertical deflection gauge. After the test system was fully loaded, the weights were removed and measurements were again taken after removal of each plate. This procedure was completed three times.

Next, force deflection curves for spinal extension were developed using the displacement gauge and the 1000-pound load cell in series with the winch cable system. This setup is shown in Figure 102. Displacement measurements and load readings were taken with each complete incremental setting of the winch. When the system had traveled the maximum distance upward, it was unloaded and the deflection measured, one setting at a time, back to its equilibrium position.

The data were then plotted on individual graphs. From these graphs, an effective spring rate for the spinal system was established and an estimation of static friction was accomplished by comparing the loading curves and unloading curves of the system.

For the small ADAM spine, an average spring rate of 650.5 pounds/inch with a standard deviation of 28.2 pounds/inch was obtained. This value compares favorably with the design value of 662 pounds/inch.

For the large manikin, the spring rate was found to be 1482.2 pounds/inch with a standard deviation of 117.1 pounds/inch. Although this test value does not compare well with the design value of 1020 pounds/inch for the large spine, it is believed that the extension tests completed on this system yielded erroneous data. The large spine compression test determined a value of 1077 pounds/inch for the spring rate of the large spine, while a rate of 1887 pounds/inch was derived from the extension tests. Since only two data points could be obtained during each extension test, only one of which is within the calibrated range of the load cell, the reliability of this data is questionable. By using ten 40-pound weights, 10 data points were taken during each compression test. It is believed

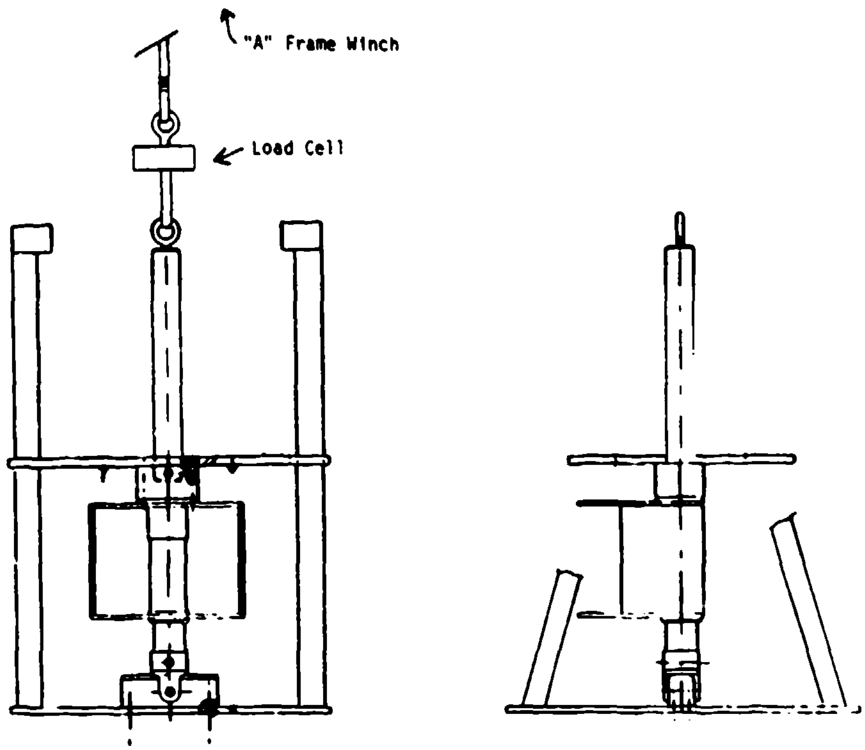



Figure 102. Spinal Test System Setup for Extension Tests with Load Cell

then, that the value of 1077 pounds/inch is more representative of the large spine spring rate than the value obtained during the extension tests.

The spinal system static friction was estimated from the force-deflection curves as one half the maximum difference between the loading and unloading curves. Figure 103 gives an example of how this calculation was derived.

The average maximum static friction in the small spinal system was determined to be 8.3 pounds. For the large spinal system, the static friction was found to be approximately 11.7 pounds. Previous analysis determined that spinal system friction would not significantly hinder the manikin response if kept under 15 pounds.

2.2.5.5.4.2. Hydraulic Fluid Tests

The damping of the spinal system is dependent on the viscosity of the fluid used in the system. However, the viscosity of a given fluid is not constant with respect to temperature. Therefore, environmental conditions during the system testing could greatly affect the damping characteristics of the spine. In order to minimize this effect, a fluid with a viscosity that is only slightly affected by temperature was sought. The proper fluid was determined by performing viscosity tests of 21 different types of fluids, ranging in viscosity at room temperature from 15 to 850 centipoises (cp).

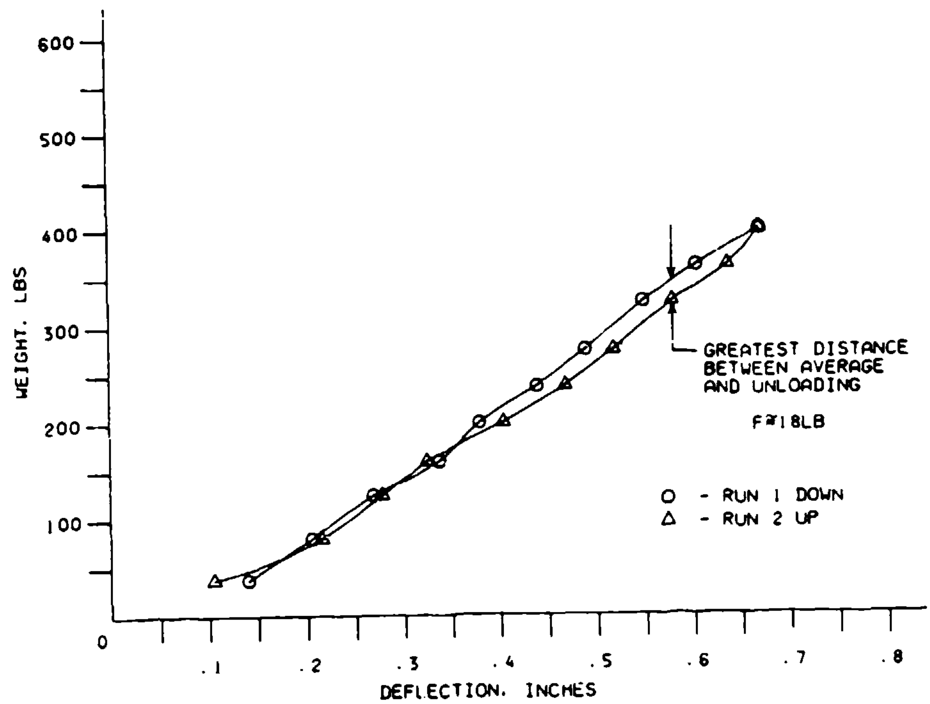



Figure 103. Example of How Manikin Static Friction is Determined from Force-Deflection Test Data

The candidate fluids were selected for their commercial availability, low cost, and nontoxic characteristics. These fluids were originally selected based on the anticipated damping characteristics of the system. As the system was tested and certain parameters were not met, the size of the piston varied and different fluids were required to produce the correct amount of damping.

The Gilmont Falling Ball Viscosimeter, consisting of a glass tube and a steel or glass ball, was used. Using two tube sizes and two different balls, four configurations, depending on fluid viscosity, were available and accurate readings for viscosities from 2 to 1000 cp were possible.

The first step in the test procedure was to determine the density of the fluid at room temperature. Next, the viscosimeter with either the glass or steel ball was filled with the fluid and sealed. The tube was then placed in a room temperature bath along with a thermometer. After 5 minutes, the system was assumed to be at thermal equilibrium and the ball was dropped. The time the ball took to travel between the two sets of fiduciary lines was recorded, and the viscosity of the fluid was obtained from the following equation:

$$u = K\ (f_1 - f_2)\ t$$

where

$u$ = viscosity in cp

$f_1$ = density of the ball (gm/m$^3$); 2.53 for glass, 8.02 for stainless steel

$f_2$ = density of liquid (gm/m$^3$)

$t$ = time of descent (minutes)

$K$ = viscosimeter constant; 3.3 for the smaller tube, 35 for the larger tube

Following the room temperature measurement, the temperature bath was raised to approximately 130°F. Tests were run at approximately every 10°F drop in temperature until the bath was back at ambient temperature. For each test run, the beginning and ending temperature and the time of descent were recorded. The room temperature density was used to calculate all viscosities. Although the room temperature density was not the same at higher temperatures, its effect on the overall viscosity of the fluids was expected to be minimal.

In order to compare viscosity characteristics, a series of viscosity versus temperature curves was generated for the 21 fluids. Comparison of the viscosities of all the tested fluids at both 75°F and 130°F is presented in Table 28. This table also presents a percent change in viscosity for each fluid from 130°F to 75°F.

To assure that the damping of the ADAM spinal system will remain between 50 and 70 percent critical for 75°F to 130°F, hydraulic fluid with a viscosity relatively insensitive to temperature was needed. Table 28 indicates that the two fluids least sensitive to temperature are the MIL-H-5606E and the Supreme Silicone. Due to its prior acceptability and accessibility by the government, MIL-H-5606E was chosen as the damping fluid for the large and small manikins.

### 2.2.5.5.4.3. Offset Loading Tests

Within the actual ADAM ejection environment, the vertical loading on the spine acts through the manikin CG which is offset from the spine centerline. Therefore, it is important to understand how this offset loading affects the free movement of the spine. A high degree of coulomb damping could not be tolerated in the spinal system. Because of this, a series of static deflection tests with an offset load was conducted, and quantitative estimates of system friction due to the offset moment were made.

TABLE 28. COMPARISON OF HYDRAULIC FLUID VISCOSITIES AT 75°F AND 130°F

| Fluid | Viscosity at 75°F (cp) | Viscosity at 130°F (cp) | Change in Viscosity from 130°F to 75°F (Percent) |
|---|---|---|---|
| Antifreeze | 15 | 6 | 150 |
| Hyjet IV | 15 | 6.5 | 131 |
| Supreme Glycol | 18 | 6.5 | 177 |
| 5606E | 18.5 | 9.5 | 95 |
| Jack Oil | 24.5 | 7 | 250 |
| Supreme Silicone | 24.5 | 13.5 | 81 |
| 10 Wt | 60 | 17 | 253 |
| Transmission | 62 | 21 | 195 |
| 10 W-30/10 Wt | 92 | 23 | 300 |
| 10 W-30 | 120 | 35 | 243 |
| 5 W-30 | 120 | 32.5 | 300 |
| 20 W-20 WD | 180 | 30 | 333 |
| 10 W-40 | 162 | 41 | 295 |
| 15 W-40 | 165 | 40 | 313 |
| 30 Wt | 205 | 35 | 486 |
| 40 Wt | 208 | 75 | 211 |
| 60 Wt | 505 | 100 | 405 |
| 50 Wt | 315 | 110 | 368 |
| 70 Wt Aircraft Oil | 760 | 125 | 508 |
| 35 Wt | 775 | 125 | 520 |
| Nitro 70 | 650 | 130 | 554 |

For this series of tests, a 2-foot long extension piece was attached to the bottom of the viscera box. It was then loaded with weights to produce a moment of 20 foot-pounds. With this offset loading in place, the system was then vertically loaded with the ten 40-pound weights. Once again, force-deflection curves were made and friction was estimated. The system was deflected only in the compression direction and the test was conducted a total of three times.

For the small manikin, the friction measured approximately 15 pounds when the spine was loaded up to 400 pounds vertically with an offset moment of 20 foot-pounds. For the large manikin, the friction measured 16 pounds under a maximum load of 400 pounds and moment of 20 foot-pounds. On the basis of engineering judgement it was concluded that values were satisfactory for the spinal system and no modifications were needed to reduce the friction forces.

2.2.5.5.4.4. Dynamic Damping Tests

The final series of tests was completed to size both the piston and the accumulator for the small and large sized manikin spines. In these tests, the 100-pound load cell was replaced with the quick release mechanism and the spine support plate was equipped with the 10 G accelerometer placed as close to the spine centerline as possible.

The testing sequence that followed consisted of filling the spine with hydraulic fluid, assembling the test system, lifting the spinal setup to its maximum height, and releasing the system. The spinal decay pattern was recorded and stored on the recording oscilloscope. The pattern was plotted and the system damping was calculated using a logarithmic decrement equation.

The first test was conducted using the unfilled spine to obtain the undamped natural frequency and to further investigate the friction of the system. After this test was conducted, the spine was filled with hydraulic fluid. The test was usually conducted eight times for each fluid in order to determine the repeatability of the test data. When a number of different fluids had been tested, the spinal system was disassembled and inspected. Modifications were made to either the piston or the accumulator and, again, various fluids were tested.

The process of modifying the piston and accumulator was continued until a large percentage of the lower viscosity fluids produced system damping between 50 and 70 percent at room temperature.

The final spine system for the small manikin consisted of MIL-H-5606E fluid, an accumulator of volume 5 ml, and a gap width of 0.0055 inch. This system had a damped frequency of approximately 10 Hz and damping of approximately 57 percent critical at room temperature, well within the established design criteria.

The testing of the large manikin spinal system was completed next with an emphasis on using MIL-H-5606E hydraulic fluid and a gap width of 0.0055 inch so that both manikins would be as similar as possible. After the testing was completed, the spinal system that was chosen for the large manikin consisted of MIL-H-5606E hydraulic fluid, an accumulator of volume 11.5 ml, and a gap width of 0.0055 inch. This system yielded a damped natural frequency of 10 Hz and damping of 59 percent critical.

## 2.2.5.6. Analysis of Final System

Within this section, the final masses, springs, and dampers of the large and small prototype manikins are presented. Results from the impedance and dynamic response analysis programs are presented using the final design parameters for both the large and small manikins.

### 2.2.5.6.1. Final System Parameters

The final measured parameters for both the small and large manikins are presented in Tables 29 and 30. All of the mass data were obtained by weighing the individual segments that make up a specific model element. The spine spring and damper values for the large and small manikins were obtained during the spine evaluation testing. Nonlinear spring rate equations for the large and small ADAM buttocks were obtained from force deflection curves of the manikins plastiscol buttocks (see Figures 71 and 72). The values for the buttocks damping were estimated at 30 percent critical for both the large and small ADAM buttocks.

TABLE 29. FINAL SMALL ADAM DESIGN PARAMETERS

| Element | Mass (kg) | Spring (N/m) | Damping (NS/m) |
|---|---|---|---|
| Viscera | -- | Locked Out | Locked Out |
| Lumbar Spine | 29.16 | 118720 | 2233 |
| Buttocks | 34.11 | 31500 (NL) | 850* |

*Estimated Value

TABLE 30. FINAL LARGE ADAM DESIGN PARAMETERS

| Element | Mass (kg) | Spring (N/m) | Damping (NS/m) |
|---|---|---|---|
| Viscera | -- | Locked Out | Locked Out |
| Lumbar Spine | 41.94 | 188600 | 3375 |
| Buttocks | 55.75 | 83700 (NL) | 1716* |

*Estimated Value

2.2.5.6.2. System Impedance and Dynamic Response

Figures 104 and 105 present plots of the small manikin design impedance and dynamic response analysis. The impedance analysis, indicates a single peak of approximately 2700 NS/m at a frequency of 3.5 Hz. This shape of impedance curve was expected for the spinal system with the viscera locked out. The dynamic response analysis in Figure 97 corresponds well with the test subject that best matches the manikin size and weight.

The large ADAM final design impedance and response results are presented in Figures 106 and 107. The large manikin exhibits the same impedance curve trend as the small manikin with a peak of 5500 NS/m at a frequency of 4.5 Hz.

The large manikin response analysis presented in Figure 107 also bears resemblance to the small manikin plotted data. However, unlike the small dynamic response analysis data, the large data could not be plotted against mean experimental data as none of the Air Force test subjects resembled the large manikin in overall height and weight. Nonetheless, it is believed that, like the small manikin, the large manikin will behave like an equally sized human in an ejection environment.

2.2.6. <u>Joint Design</u>

A main characteristic of a manikin is its ability for movement. This is achieved in the ADAM through its joints which allow 38 degrees of freedom and the elasticity of the neck which allows 3 degrees of freedom. These degrees of freedom not only allow movement at the correct locations but they also represent some special features of a human. A human joint is highly complex in that it has a muscular system (to power the movement of the joint) integrated with tendons (to transfer power to the joint and connect several elements) and bones (to transfer all movement along the segment). The ADAM joints have represented the combined effort of these major groups through possessing the following characteristics of a human joint: the degrees of freedom, the range of motion, increasing torque resistance near the limits of the range of motion, and constant resistance throughout the range of motion.

2.2.6.1. Ranges of Motion

The ranges of motion designed into the ADAM joints are listed in Table 31. As these specified ranges of motion and degrees of freedom are similar to those found in the human, they contribute to a proper whole body dynamic response for the manikin.

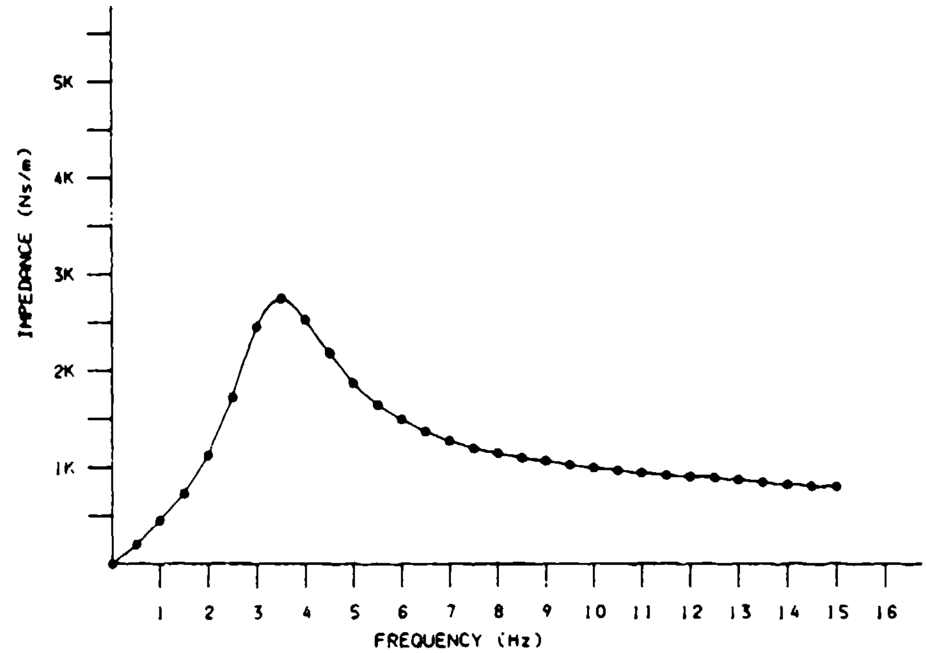



Figure 104. Final Small Prototype Impedance Curve Estimation

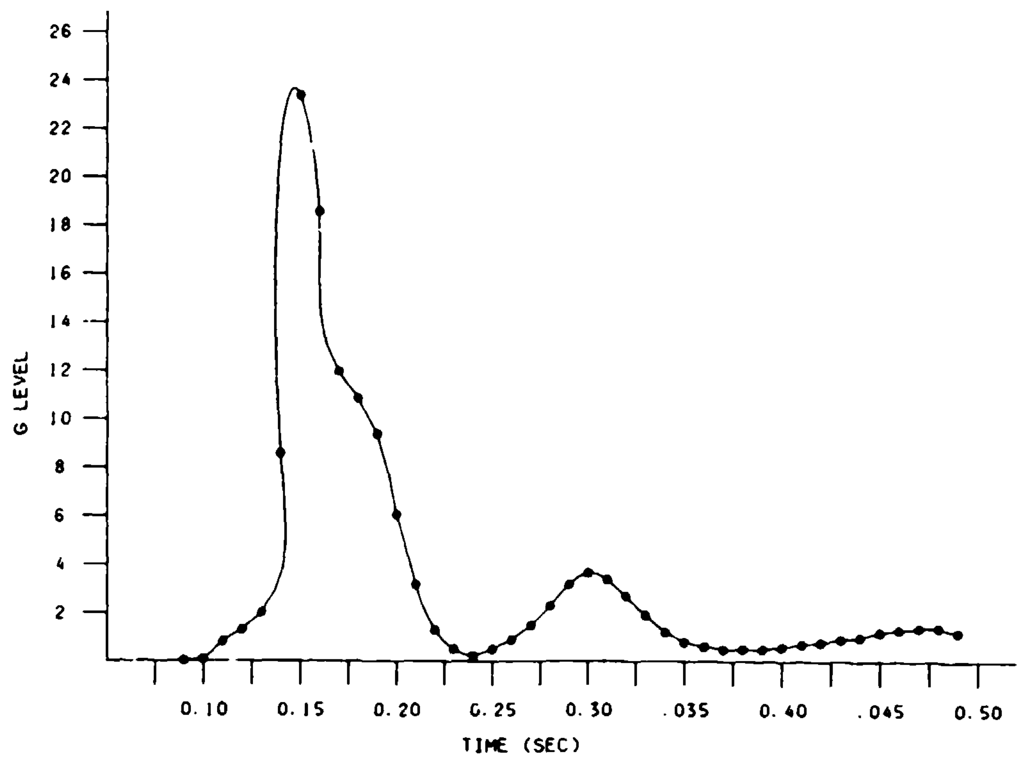



Figure 105. Final Small Prototype Dynamic Response Estimation

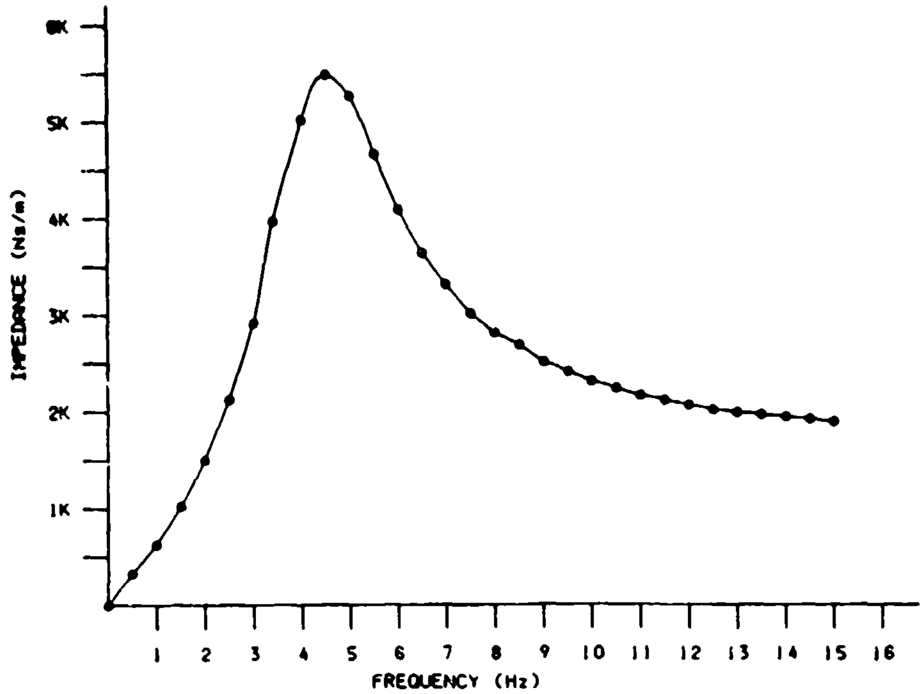



Figure 106. Final Large Prototype Impedance Estimation

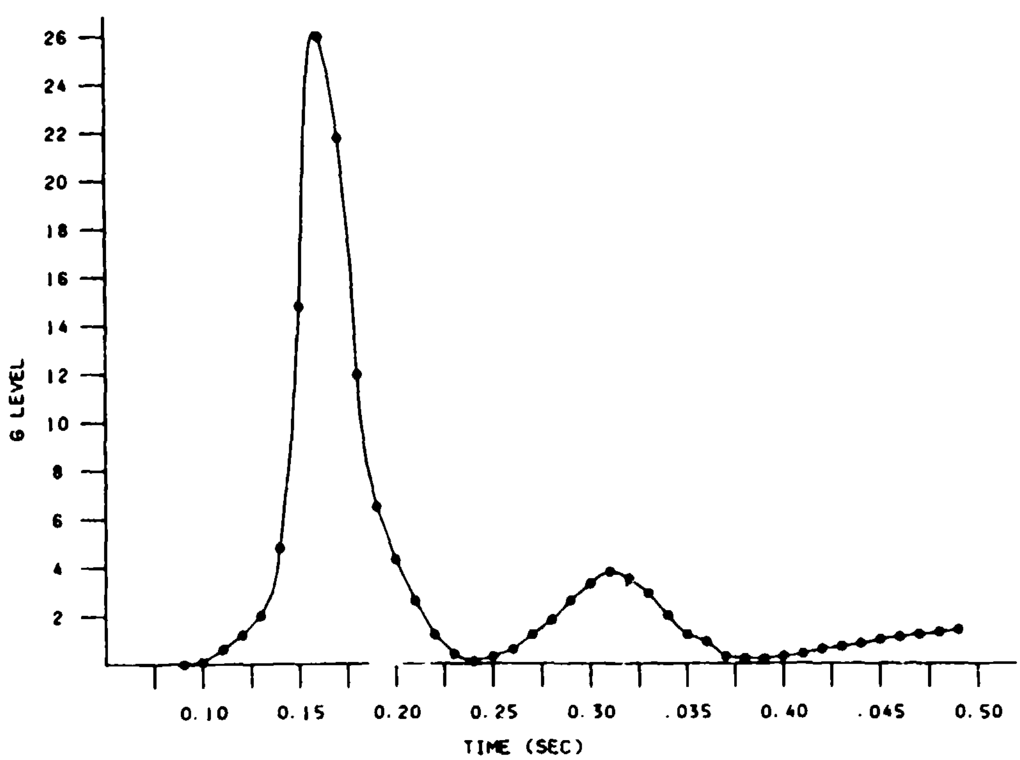



Figure 107. Final Large Prototype Dynamic Response Estimation

TABLE 31. JOINT DEGREE OF FREEDOM AND ROTATIONAL LIMITS

| Joint | Description of Motion | ADAM Requirement (Degrees) |
|---|---|---|
| Wrist | Flexion | 85 |
| | Extension | 85 |
| | Abduction | 45 |
| | Adduction | 25 |
| Elbow | Flexion | 140 |
| Forearm | Supination | 95 |
| | Pronation | 75 |
| Shoulder | Flexion | 178 |
| | Extension | 57 |
| | Traverse Abduction | 134 |
| | Traverse Adduction | 48 |
| | Coronal Abduction | 170 |
| Upper Arm Rotations | | 115 |
| | | 15 |
| Sternoclavicular Joint | Pronation | 10 |
| | Retraction | 10 |
| | Elevation | 10 |
| | Depression | 10 |
| Lumbar Pivot Point | Yaw | 15 |
| | | 15 |
| | Pitch | 30 |
| | | 20 |
| | Roll | 15 |
| | | 15 |
| Ankle | Flexion | 45 |
| | Extension | 25 |
| | Inversion | 34 |
| | Eversion | 18 |
| Knee | Standing Flexion | 125 |
| | Tibial Rotation at 90° Flexion | |
| | Internal | 35 |
| | External | 45 |
| | Tibial Rotation at 0° Flexion | |
| | Internal | 0 |
| | External | 0 |
| Hip | Flexion | 115 |
| | Extension | 0 |
| | Supine Abduction | 60 |
| | Supine Adduction | 30 |

TABLE 31. JOINT DEGREE OF FREEDOM AND ROTATIONAL LIMITS (continued)

| Joint | Description of Motion | ADAM Requirement (Degrees) |
|---|---|---|
| Hip (continued) | 90° Flexion Abduction | 50 |
| | 90° Flexion Adduction | 30 |
| | Rotation 90° Hip Flexion | 40 |
| | | 40 |
| | Rotation Full Extension | 40 |
| | | 40 |
| | Rotation Prone 90° Knee | 40 |
| | | 40 |

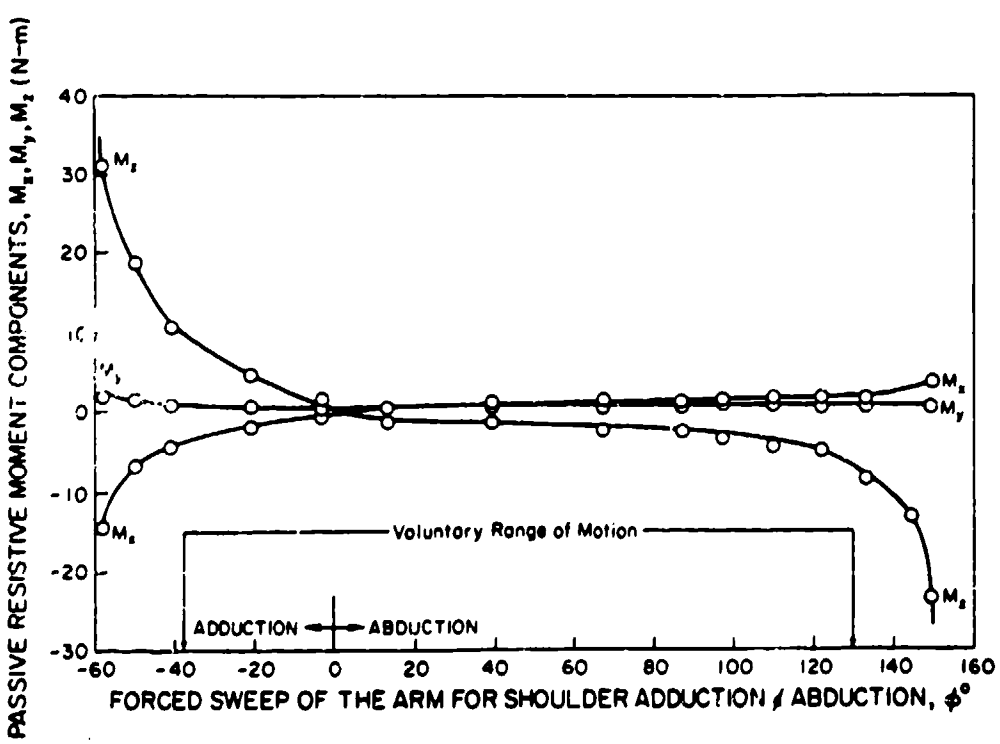



Figure 108. Components of the Passive Resistive Moment Vector at the Shoulder Joint During Forced Sweep of the Arm for Shoulder Abduction and Adduction

The manikin joints modeled the human degrees of freedom through the use of two types of basic devices; clevis, rotational, or a combination of these two types of devices. The details of the different types will be discussed later.

Modeling a characteristic found in humans, joint increasing resistance mechanisms have been designed within these ranges of motion. These mechanisms create an increasing resistance to motion near the limits of the joint rotation. They are discussed in detail in the following section.

### 2.2.6.2. Joint Increasing Resistance Mechanisms (Soft Stops)

#### 2.2.6.2.1. Design Parameters

Engin (1979) found that human joints exhibit a nonlinear increase in resistance as a joint flexes to the end of its range of motion. Figure 108 shows this trend in a typical curve for shoulder adduction/abduction.

This nonlinear behavior found in human joints is believed to be caused by two actions. The first is muscle and tendon tissue coming into play. This action was modeled in the manikins by soft stops located within the ranges of motion. The second cause of increasing resistance is skin-to-skin contact which is generally modeled in the manikins by skin-to-skin contact.

The increasing resistance mechanisms located throughout the manikin actually serve two purposes: (1) to represent the characteristic described above as these characteristics are critical to the dynamic response of the manikin, and (2) to absorb some of the dynamic loading from limb motion.

#### 2.2.6.2.2. Design Concepts

The basic concept developed to achieve increasing resistance to motion of the joints at the ends of the ranges of motion was to place an elastic material on the fixed metal stop at the end of the joint range of motion.

This material would interact with the stops over the last 10 to 15 degrees of motion and would deflect in the desired nonlinear manner based on the material shape and force deflection characteristics.

This concept required that the material be compatible with the other materials that made up the joint, be readily attachable and detachable for ease of repair, exhibit the required deflection/stiffness characteristics, and have sufficient toughness and durability. Since elastomer materials exhibit nonlinear deflection characteristics, they were investigated thoroughly to develop a material with the above characteristics.

A testing procedure was developed to define the material and shape which best met the requirements.

2.2.6.2.3. Testing Procedure for Elastomer Stops

The testing involved two procedures--the preliminary testing which narrowed down the number of elastomers to be tested, and the combined dynamic/static testing which determined the force versus deflection and stability characteristics of the elastomer.

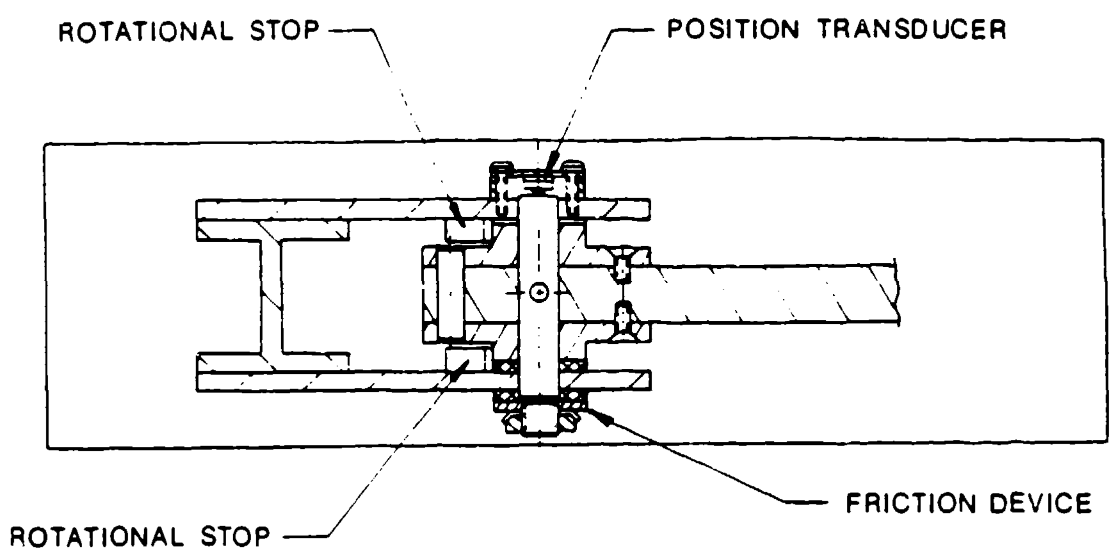
The preliminary testing involved cutting the seven samples listed in Table 32 to representative sizes and then measuring the force versus deflection characteristics with a test fixture. The samples were subjected to approximately 1600 psi using an arm which was rotated to create contact with the stop. From these tests, it was concluded that most of the materials could meet the force-deflection requirements by varying the shape of the stop. From the durability standpoint, the 75 and 90 durometer polyurethane samples performed the best. The polyurethane samples showed no signs of surface cuts or internal cracks. All other material showed significant damage after being highly stressed.

TABLE 32. SOFT STOP MATERIALS TESTED

| | |
|---|---|
| 1/2-inch thick | 55-60 Durometer Neoprene |
| 1/8-inch thick | 75-80 Durometer Neoprene |
| 1/4-inch thick | 35-50 Durometer Gum Rubber |
| 1/2-inch thick | Sponge Rubber |
| 1/4-inch thick | 75 Durometer Polyurethane |
| 1/4-inch thick | 90 Durometer Polyurethane |

These tests were only preliminary as the elastomer sections were being deflected by metal pieces which were much larger than the stops being designed in the manikin. After selecting the polyurethane as the best material, a more realistic testing environment was designed to determine the stop shapes to be used and the durability of the material in a dynamic environment. A testing device was designed based on the same ideas as the preliminary device; however, it was a closer representation of the actual manikin joints.

This device consisted of an arm connected to a base via a model joint (Figure 109). A drop tower (Figure 110) was designed to be used in combination with the second test device. Using the drop tower to simulate the joint acceleration in an ejection test, the characteristics of the elastomer stops



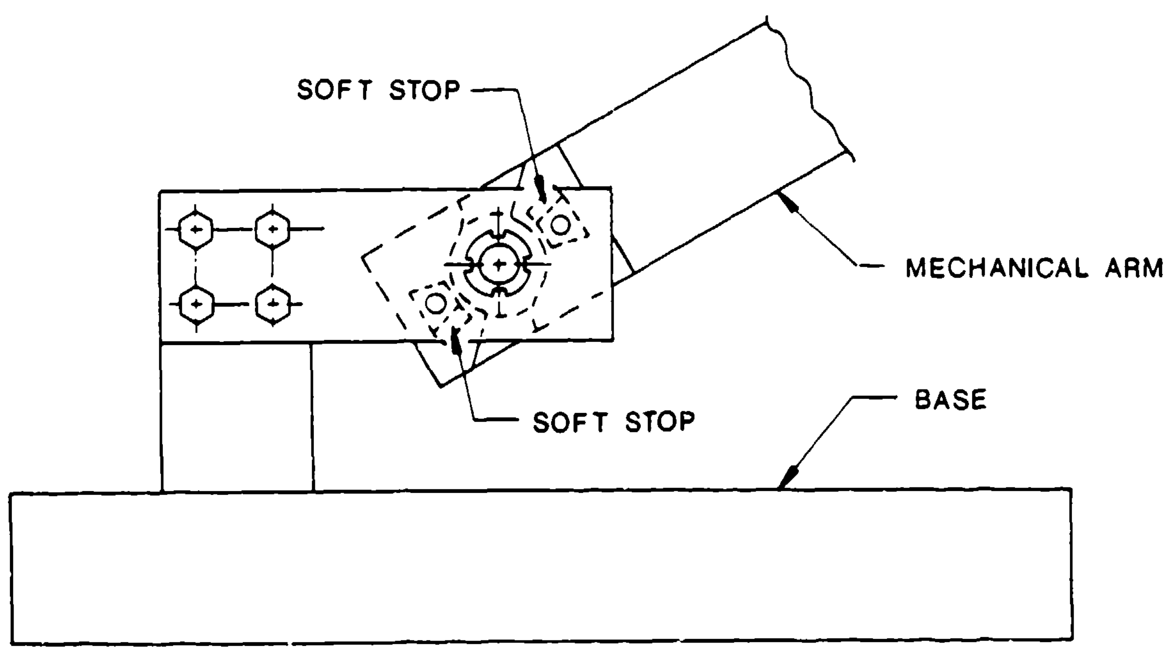




Figure 109. Joint Test Fixture

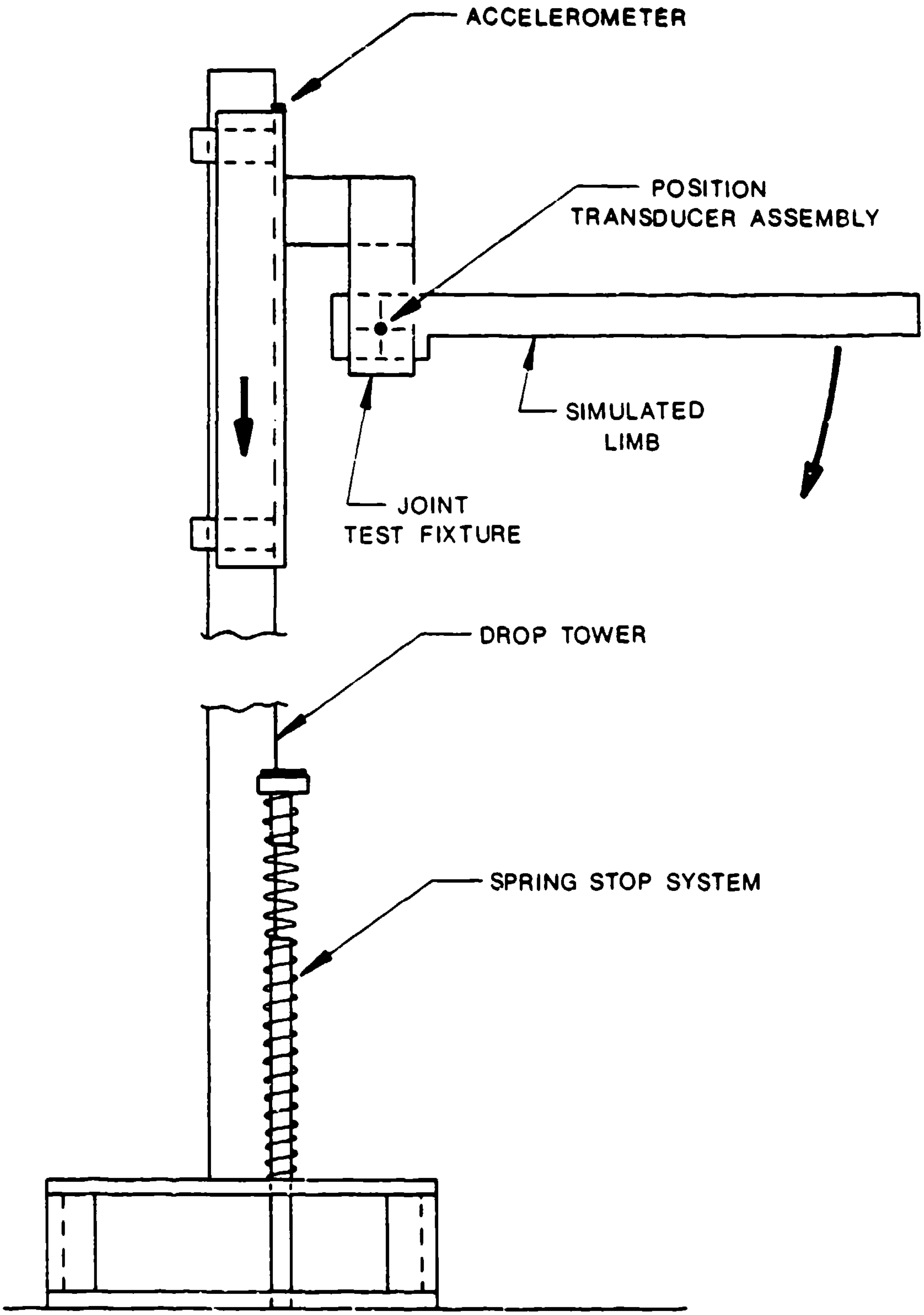



Figure 110. Schematic of Drop Tower and Test Fixture

in a dynamic testing environment could be determined. Static tests were also performed using this same device to determine the force-deflection curves of each soft stop. The static loading produced loads up to 6400 in-lb of torque on three different shapes. Each shape was compressed four times and inspected during the loading and unloading process.

Drops were made from heights of up to 11 feet with resultant fixture decelerations of 75 Gs and arm velocities of 52 rad/sec. Figure 111 gives an indication of the maximum average arm velocities generated by the test device.

The test procedure consisted of the following steps:

- Measurement of force versus deflection of the arm prior to each drop.
- Recording of damage that occurs to the soft stop as a result of high dynamic and static loading.
- Observation of how well the fastening method performed.
- Measurement of force versus deflection of the arm for a static loading after each drop test.

After several drops were made and the static testing was completed, the following criteria were used to evaluate the results:

- Ability of the specimen to exhibit the proper nonlinear force versus deflection characteristics (Engin, 1979).
- Ability of the material to withstand high static loading with no permanent deformation.
- Ability of the fastener to hold the soft stop in place.

The polyurethane specimen met the first two criteria and was, therefore, a good material for soft stops. The material developed the correct force-deflection curve when in a trapezoidal shape. The attachment method selected was bonding the stops to the hard stops using "super glue." A mechanical attachment was investigated; however, due to strength and space limitations, it was eliminated. The bonding method worked well in testing and was, therefore, used in the manikin.

2.2.6.3. Joint Resistance Mechanisms

2.2.6.3.1. Design Requirements

It was required to design a mechanism into the manikin joints which would create a constant torque resistance to movement. This resistance level had to be variable with the ability to resist at least the

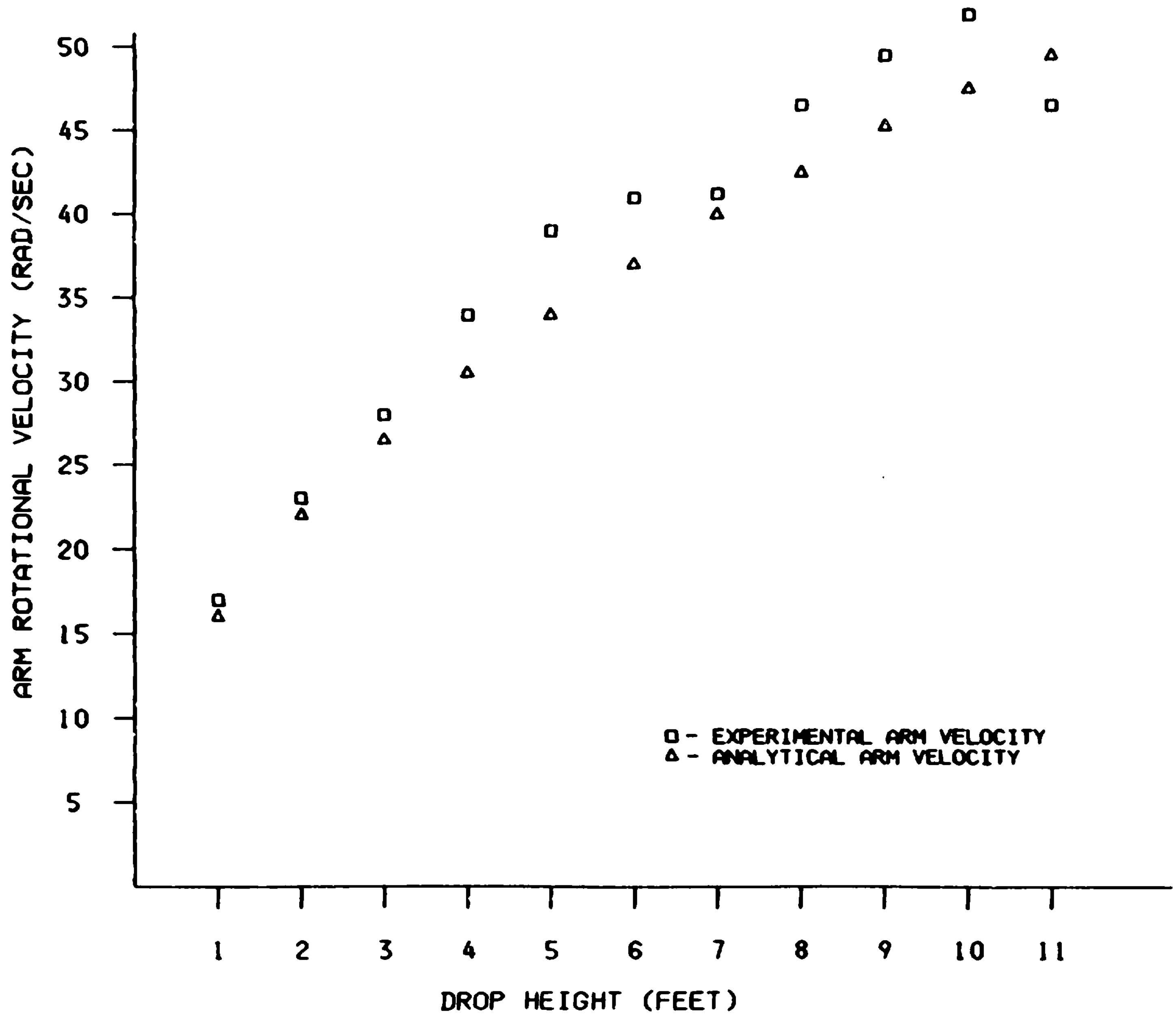



Figure 111. Drop Height Versus Arm Velocity

pull of gravity on the attaching limb. Once set, this resistance should remain constant after several tests such that an additional torque need not be applied.

The primary purpose of this mechanism was to simulate muscle resistance. As shown previously in Figure 108, there is a constant resistance throughout a range of motion of a human joint. A secondary purpose of this ability is to aid in the handling and initial setup of the manikin.

2.2.6.3.2. Design Concepts

The Hybrid II and III and the GARD CG manikins employ friction mechanisms consisting of a set of delrin disks, a sliding nut, and a bolt. The bolt applied pressure on the disks by pulling the nut inward. Joint resistance was developed by friction between the two disks. Operational experience with these joints had indicated that their resistance development was somewhat temperature sensitive. As this was the only concept in operation and not acceptable for ADAM, four new concepts were developed.

The first concept, as shown in Figure 112, developed pressure on the friction disks by applying an outward load on the clevis forks. The clevis pin was fixed such that it provided the necessary relative motion at the external surface of the clevis for the potentiometer. Figure 113 shows the second concept. This was composed of a bolt and a nut which applied an inward pressure on the clevis forks which, in turn, applied pressure on the friction disks. The pin of the clevis was again fixed such that it functioned similar to the first concept. The concept shown in Figure 114 involved using a tapered brass bushing for the development of resistive torque instead of a friction disk. The fourth concept, as shown in Figure 115, again employed friction disks. In this case, they were squeezed against one side of the clevis fork. The clevis pin was allowed to float freely in the other side of the clevis fork and, thus, provided a place for the potentiometer to be mounted. Prototype hardware of each concept was built and tests were performed on each with various friction materials.

2.2.6.3.3 Testing Procedure

The selection process of the design concept and friction material to be used for the manikin involved a testing procedure which simulated the ejection environment. Initially, there were 11 combinations of materials and concepts (Table 33). Before extensive testing commenced, a series of preliminary tests were performed to narrow down this number. The test device was used for both the preliminary and detailed friction mechanism testing.

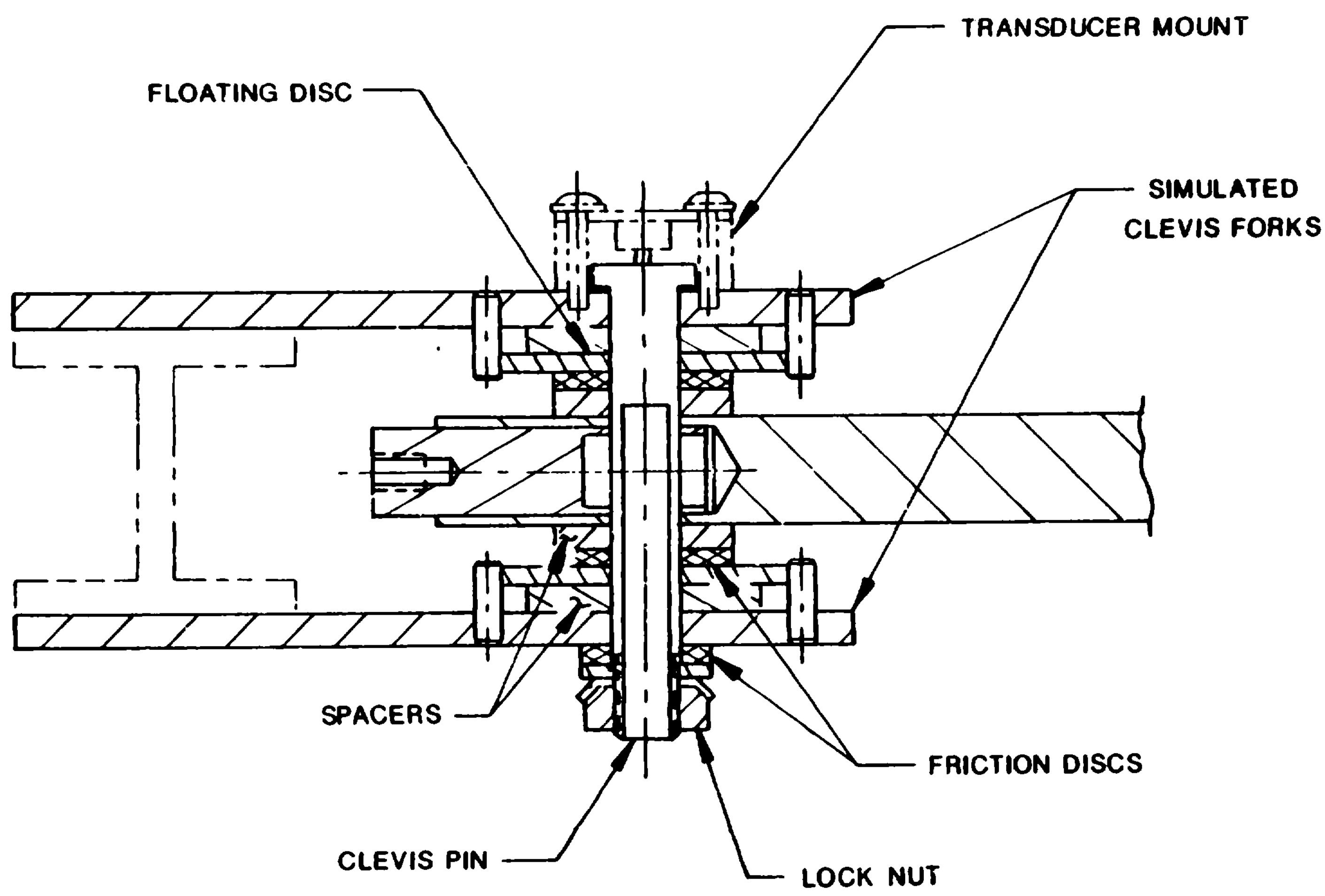



Figure 112. Friction Test Fixture, Concept No. 1

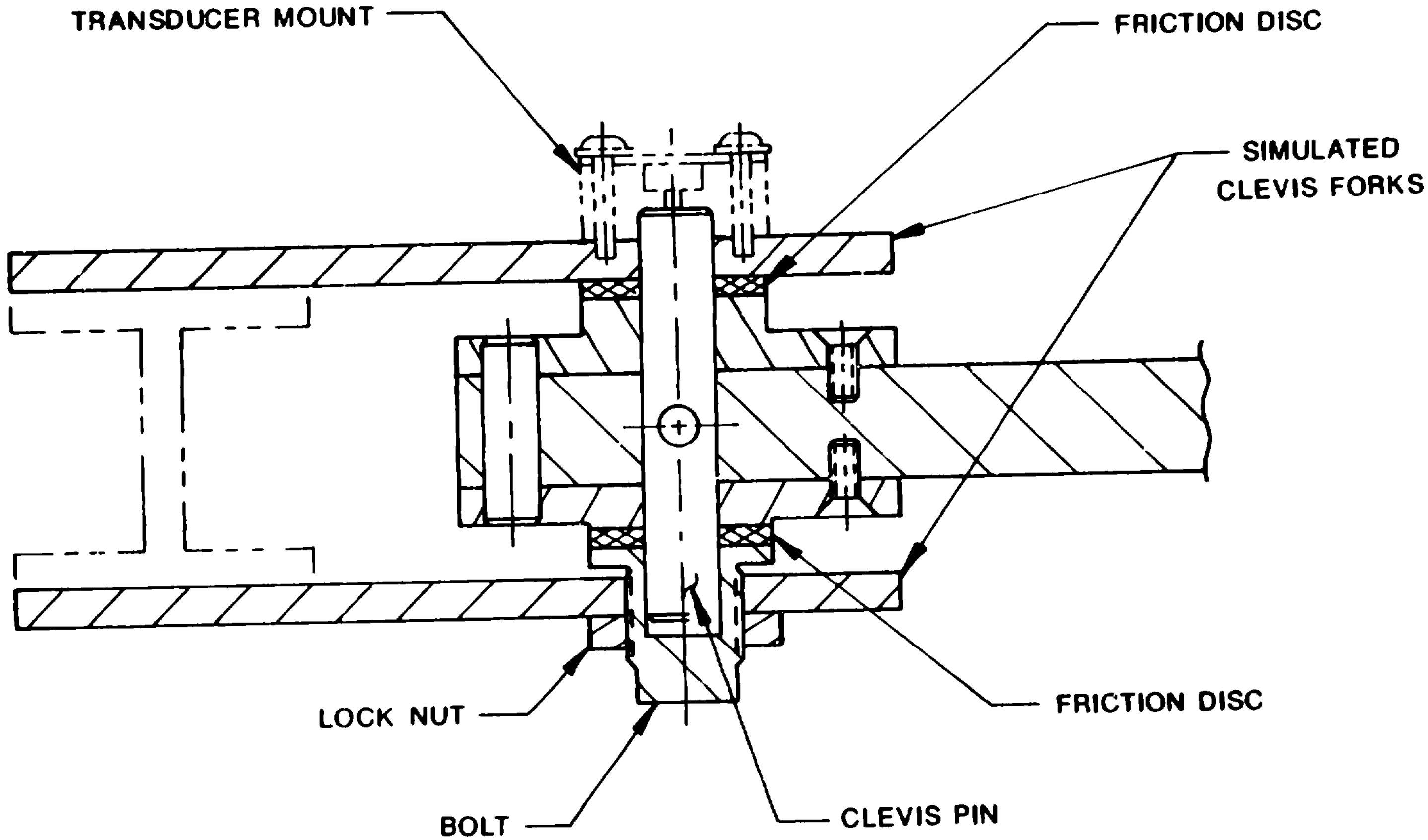



Figure 113. Friction Test Fixture, Concept No. 2

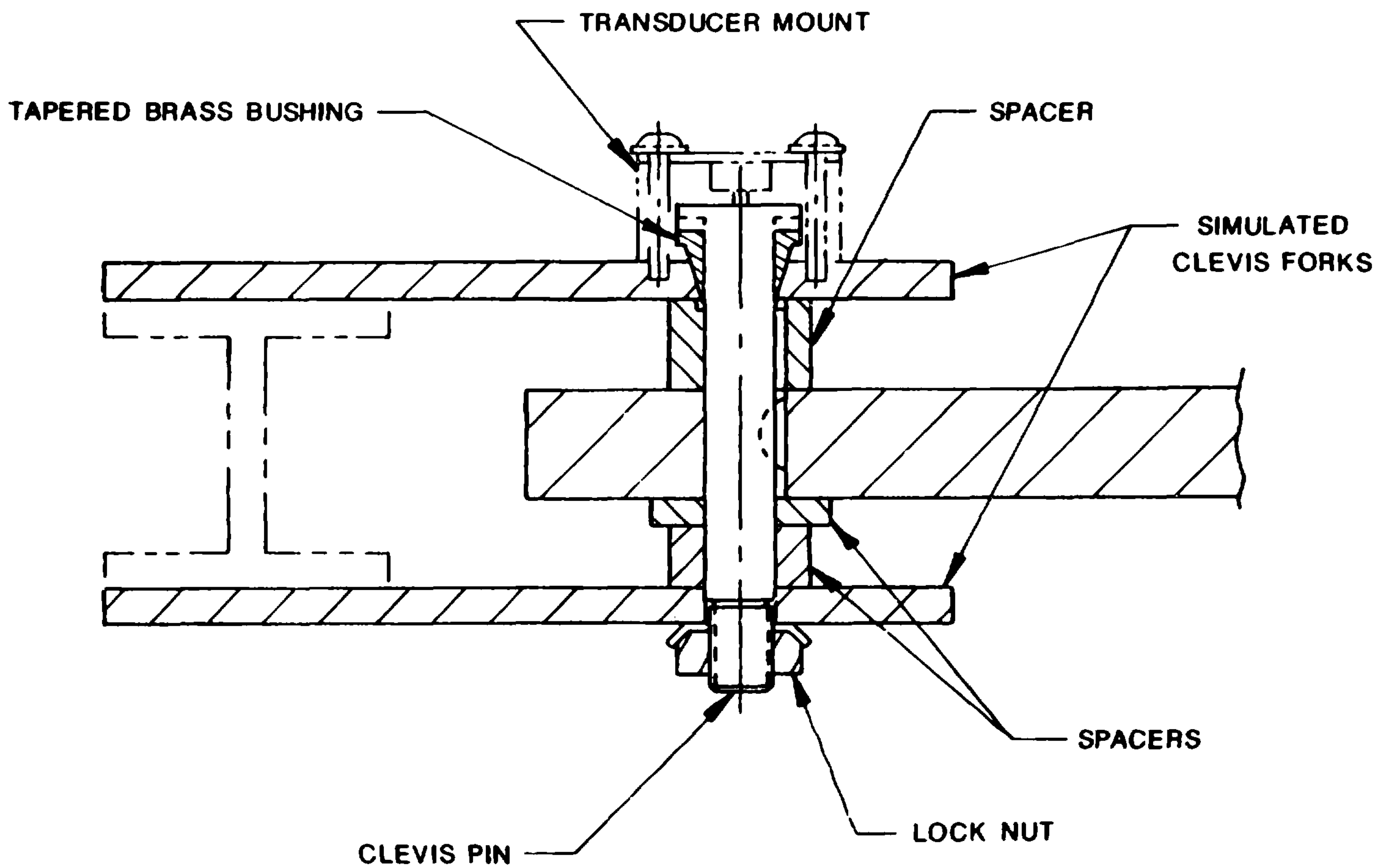



Figure 114. Friction Test Fixture, Concept No. 3

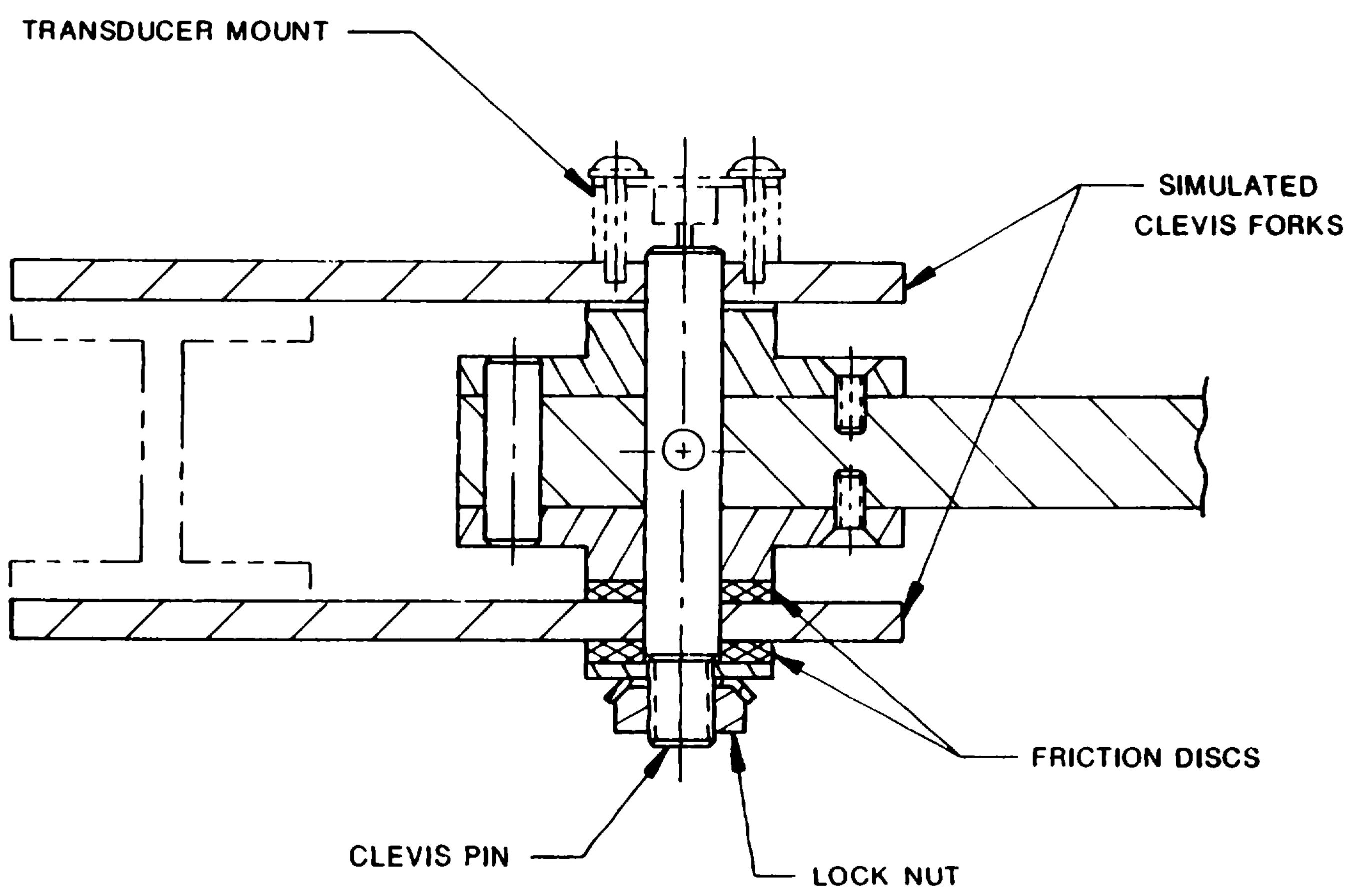



Figure 115. Friction Test Fixture, Concept No. 4

TABLE 33. MATRIX OF TESTED FRICTION MECHANISMS AND MATERIALS

| | Design Concepts | | | |
|---|---|---|---|---|
| Friction Materials | No. 1 | No. 2 | No. 3 | No. 4 |
| Cork/Neoprene Disk Backed with Steel | X | X | O | X |
| Leather Disk | X | X | O | X |
| Nonasbestos Brake Material | X | X | O | X |
| Solid Brass Bushing | O | O | X | O |
| Split Brass Bushing | O | O | X | O |

X = Tested Combination
O = Nontested Combination

The preliminary test procedure involved adjusting the tension on the resistance mechanism to a predetermined level with a torque wrench and then measuring the resistive torque developed by the joint. The test fixture was then heated to approximately 160°F and the joint resistive torque was rechecked. This procedure checked each concept for the ability to develop resistance and sensitivity to temperature.

The nonasbestos brake material was consistently found to exhibit superior traits regarding joint design. It was less temperature sensitive than the others and developed a higher resistive torque than the others due to its higher coefficient of friction. Of the design concepts, the fourth proved to be the best. The first and second concepts relied upon pressure on the clevis forks, inward for the first and outward for the second, to develop resistive torque. This turned out to be significant in that some relatively large deflections were created when trying to develop high resistive torques. A deflection of a few thousandths of an inch created an unacceptably high bending stress in the clevis forks. The third concept was found to be very temperature sensitive. When heated to 160°F the tapered bushing bound up and created resistances that were orders of magnitude greater than at room temperature. The fourth concept proved to be essentially insensitive to temperature changes, provided a relatively easy method of adjustment, and developed a repeatable resistive torque quite well.

After determining that this fourth concept had superior qualities when compared to the other concepts, the test fixture was modified so that it represented the large ADAM elbow and forearm with respect to the size and range of motion properties and employed the fourth concept for a friction mechanism. Drop tower and resistance repeatability tests were then conducted.

The drop tower tests were broken into both experimental and analytical work. The analytical work conducted prior to the tests involved solving the differential equations of motion on a digital computer using the Runge-Kutta technique. The test parameters were set using the results from this work. The experimental work involved the following procedure:

1. The fixture was lifted to and fixed at a specific height.
2. The arm was tied in a horizontal position with a thin wire.
3. The release mechanism was tripped, allowing the fixture to drop.
4. The base of the fixture struck a set of springs, which caused it to decelerate.
5. The sharp deceleration of the fixture base caused the arm to rotate.

The test fixture was dropped from heights up to 3 feet which produced fixture decelerations of 40 Gs and arm velocities of 28 rad/sec. The joint resistance was set at three basic levels of 1, 2, and 5 Gs and was dropped nine times at each level. It was then removed from the drop tower and the static resistance was rechecked.

Through the use of a position transducer at the joint and an accelerometer on the moving test fixture, the results consisted of the deceleration pulse and the position of the arm recorded with a digital oscilloscope. Through this information, the velocity of the arm was known. The resistance developed was found to be very consistent. While it did vary slightly from the original setting, its overall standard deviation was only 0.1 G.

Using these results, the fourth concept with the brake material was accepted for use in the manikin joints. This configuration met all temperature, durability, and resistance capability requirements. Each joint was then sized separately using the above configuration.

2.2.6.4. Joint Instrumentation

2.2.6.4.1. Requirements

In order to determine the response of the manikin, the joint rotations must be monitored. Since ADAM must not only react like a human but show some insight into the motions of a human in different loading situations, the ability to determine the response of the manikin is essential to this effort. The method chosen for measuring the rotations must not interfere with the rotation of the joint or segment. It also must be adaptable to all of the joint designs.

2.2.6.4.2. Design

The selection process for the joint measuring devices began with the LRE as it was the only manikin designed with joint measuring devices for each range of motion. This manikin employed position transducers with large cylindrical bodies which were buried in a hollow joint pin or within the hollow long bones (Figure 116). Since the wiring of these transducers was difficult, the use of a flat transducer mounted external to the joint was investigated.

These externally mounted transducers were also easily accessible for calibration and could be used with the joint resistance mechanisms mentioned earlier. Figure 117 shows this type of transducer mounted in the elbow joint. To test these position transducers in a typical joint under representative loadings seen by the manikin, two variations were incorporated into the joint friction test device. The two variations were different in that the mating hole was either hexagonal (with a hexagonal mating shaft) or cross-shaped (with a blade shaped shaft). It was found that there was approximately six degrees of mechanical hysteresis between the hexagonal hole and interfacing shaft even when the shaft was carefully selected. The cross-shaped hole, however, had no such hysteresis.

Except for the spine yaw and the wrist and ankle joints, which are not measured, each articulation possesses the transducer with the cross-shaped mating hole mounted external to the joint. The mating pin in each case is concentric to the center of rotation of the joint.

2.2.6.5. Final Design

Joints in the ADAM consist of two general types: the clevis (such as elbow and knee), and the sleeve (such as forearm rotation). Some joints (such as the shoulder and hip) combine these to form a joint with several degrees of freedom. A detailed description of one of each type of joint will be presented in the following paragraphs. Those not presented are simply variations of the same concepts with respect to the dimensions, degrees of freedom, and ranges of motion.

2.2.6.5.1 Clevis Joint

The elbow is an example of a clevis joint. As shown in Figure 117, this joint type consists of two major parts--the clevis, and the inner piece. They are connected by the clevis pin which is fixed to

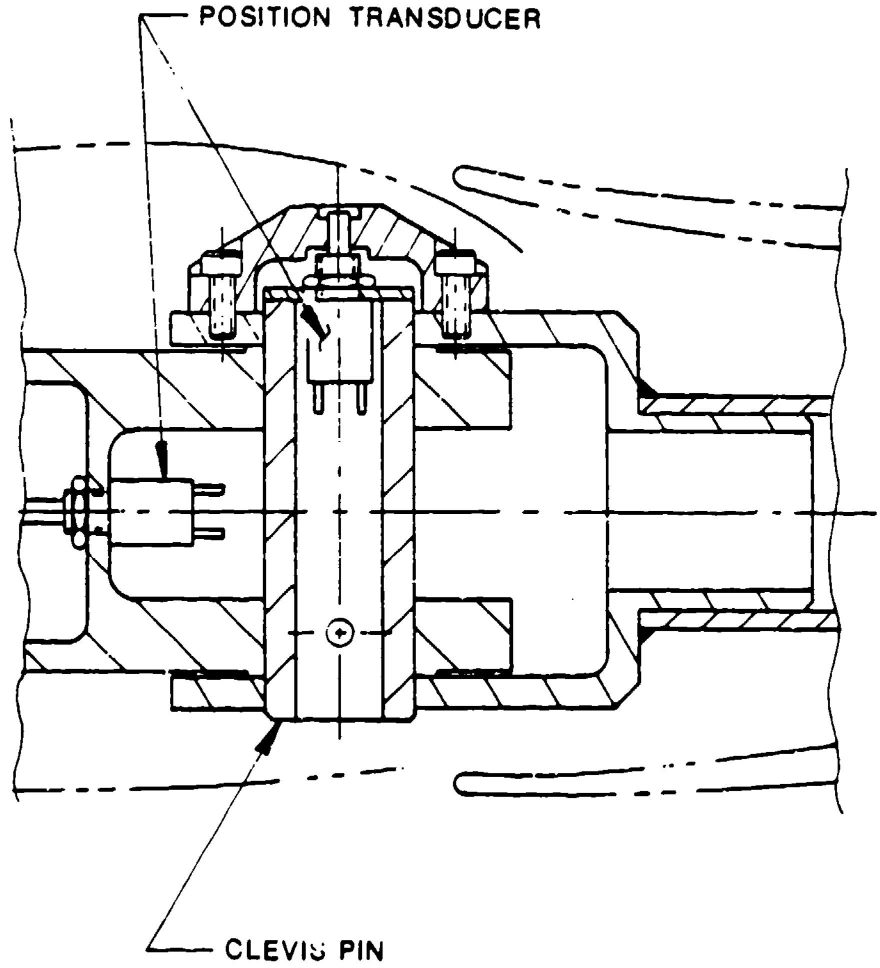



Figure 116. LRE Knee Joint Instrumentation

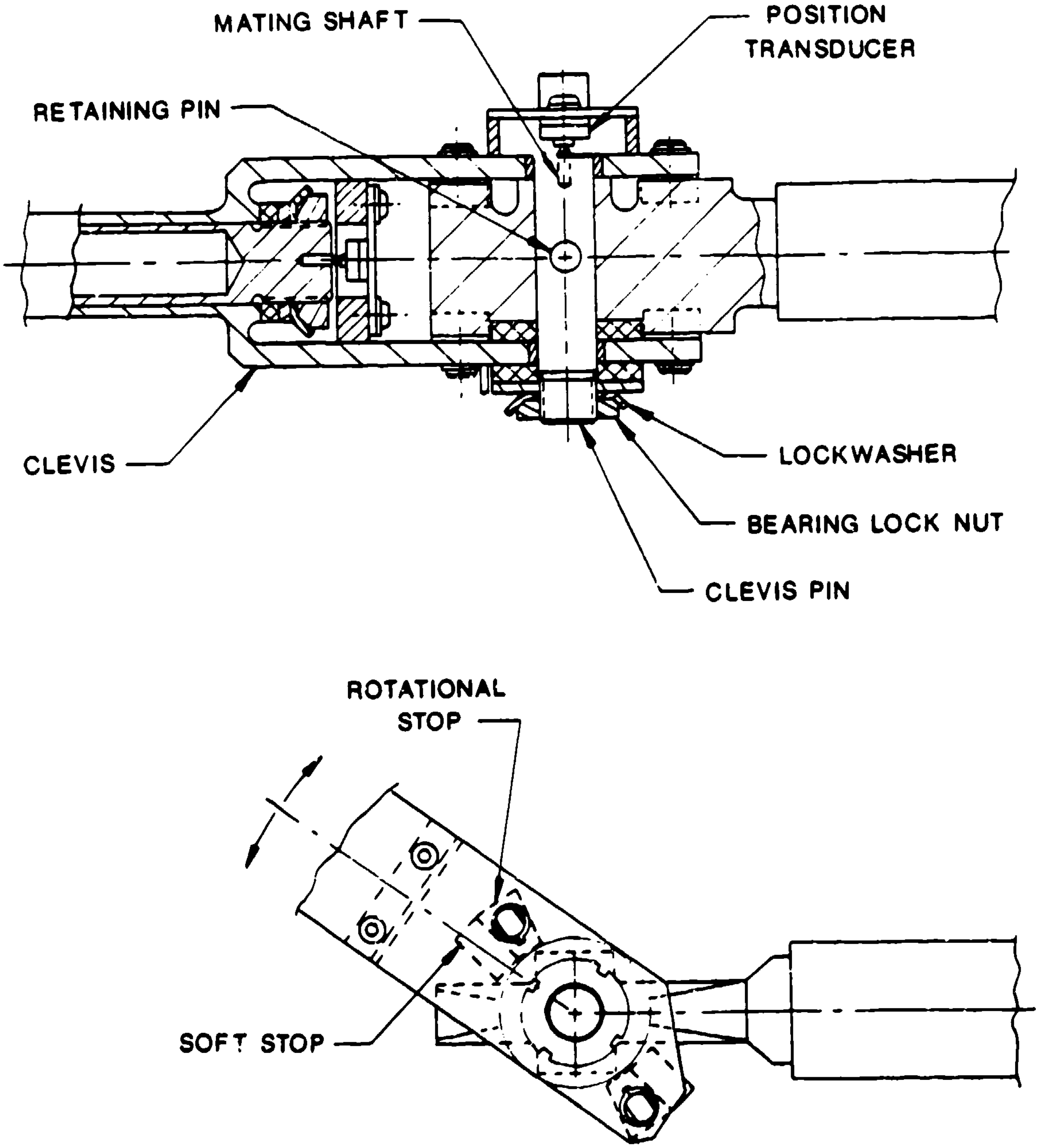



Figure 117. Elbow Joint

the inner part of the joint with a retaining pin. The retaining pin is held in place with small clips at either end. The resistance is adjusted by tightening the bearing lock nut. Once the tension is adjusted to the desired level, the nut is held in place by bending a tang on the lock washer into a groove on the nut. The required range of motion is obtained by proper placement of four stops, two on each side of the clevis.

#### 2.2.6.5.1. Sleeve Joint

Forearm rotation is achieved through the use of a sleeve joint. ADAM sleeve joints, as shown in Figure 118, consist of two concentric tubes. The inner tube is able to rotate with respect to the outer one about their common axis. The resistance mechanism contains only one friction washer and a lock nut. The lock nut is attached on the end of the inner tube and pushes the washer down onto the end of the outer tube. The range of motion is determined by stops located at the other end of the outer tube.

#### 2.2.6.6. Conclusions

The joints present in the ADAM are representative of a human in that they exhibit a variable resistance to motion within the range of motion. They also have ranges of motion and degrees of freedom similar to a human along with the ability to resist motion throughout the range of motion. Unlike a human, ADAM can measure its own response. With these features, the response of the manikin is likely to simulate that of a human. Being able to measure its response, the manikin could be used to predict human responses to high loadings and possibly used to develop injury criteria.

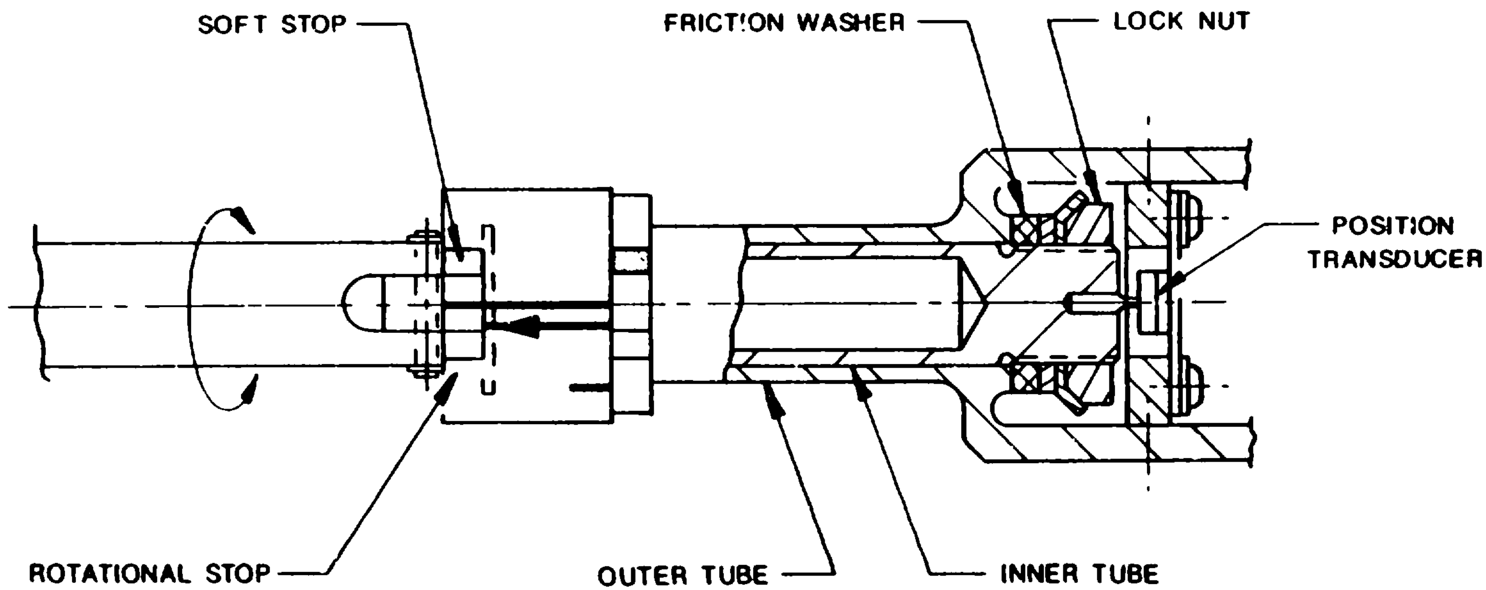



Figure 118. Small Forearm Sleeve Joint

## Section 3
## INSTRUMENTATION DESIGN

### 3.1. INTRODUCTION

This section describes the ADAM instrumentation design and the functional operation of the ADAM instrumentation system. ADAM instrumentation is used to measure various manikin articulations and loadings, and includes signal conditioning, telemetry, and onboard data storage. The instrumentation system can be divided into two distinct subsystems--the signal (analog) conditioning system and the microprocessor (digital) system.

The Instrumentation System Block Diagram (Figure 119) illustrates these two subsystems. The two analog interface boards and CREST interface board make up the signal conditioning system. The processor board, the memory board, and the digital I/O board make up the microprocessor system. The analog-to-digital conversion board provides the interface between the two systems.

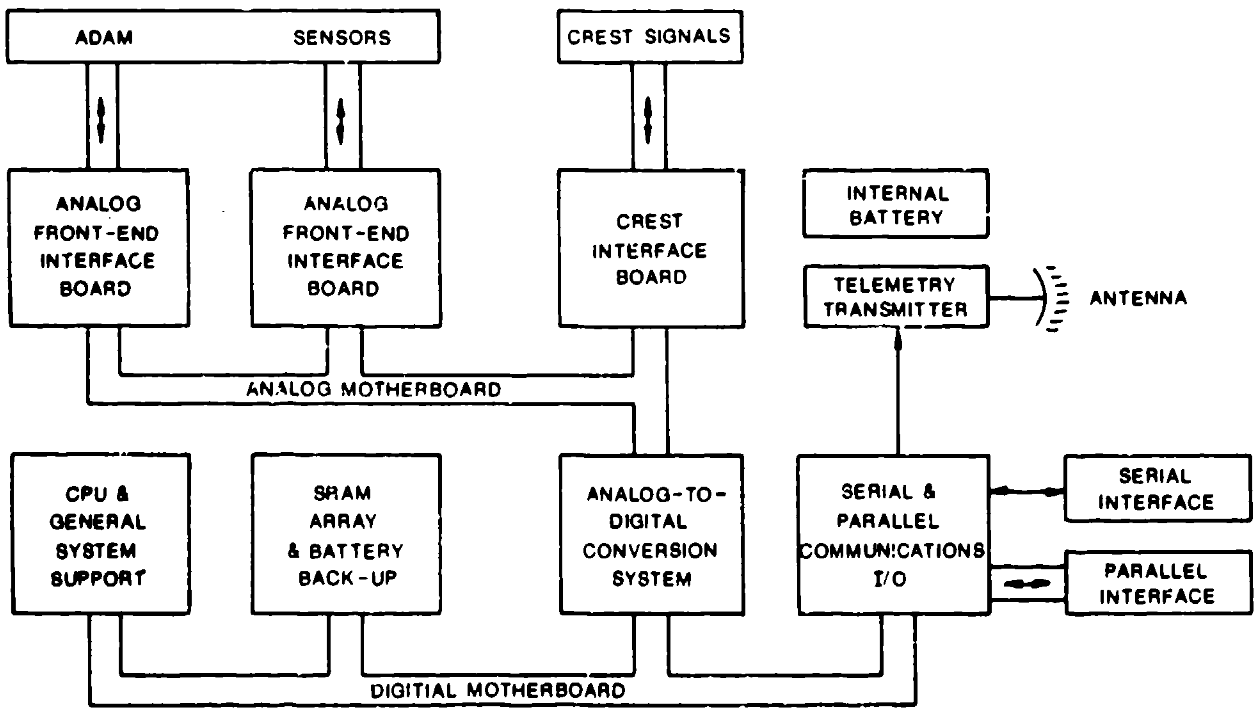



Figure 119. ADAM Instrumentation System Block Diagram

## 3.2. SIGNAL CONDITIONING CIRCUITRY

This section describes the signal conditioning circuitry in the ADAM instrumentation system. The instrumentation is capable of digitizing 128 channels. Seventy-two of these channels are manikin channels in which signal conditioning circuits are incorporated in the instrumentation design. The remaining 56 channels do not have any signal conditioning circuitry and are designed to receive analog signals conditioned externally to the manikin. The signal conditioning circuitry also serves to provide computer controlled calibration information on all the manikin sensors.

### 3.2.1. Functional Description

A block diagram of the signal conditioning circuitry is presented in Figure 120. It shows that there are 72 manikin channels available for use. There are 36 channels available to measure low level signals (those that require an amplifier) and 36 channels available for measurement of high level signals (those that do not need an amplifier). Each signal has a provision for a computer controlled shunt calibration ($R_{cal}$) in order to verify the proper calibration of the channel and verify that the sensor is attached.

Each of the sensor signals is passed through an eight-pole Butterworth low-pass filter. The low-pass filter is used to prevent aliasing errors from being introduced in the digitized data. Aliasing occurs when a time sampled signal has a frequency content greater than half the sampling frequency because those higher frequencies are "aliased" or folded back to appear as a lower frequency. This introduces an "aliasing" error in the sampled data. The Butterworth filter was chosen since there was a minimal gain error in the pass band of the filter. These filters have a cut-off frequency that is computer controllable from 0.1 Hz to 10 kHz.

The outputs of the filters are channeled into analog multiplexers. These are electronic multiposition switches that are computer controlled. The multiplexers are arranged such that only one channel of 32 is output to the analog-to-digital (A/D) converter. Four banks of 32-channel multiplexers are used to channel the signals to the A/D conversion system.

Each of these areas of the signal conditioning circuitry will be discussed in detail in the following sections.

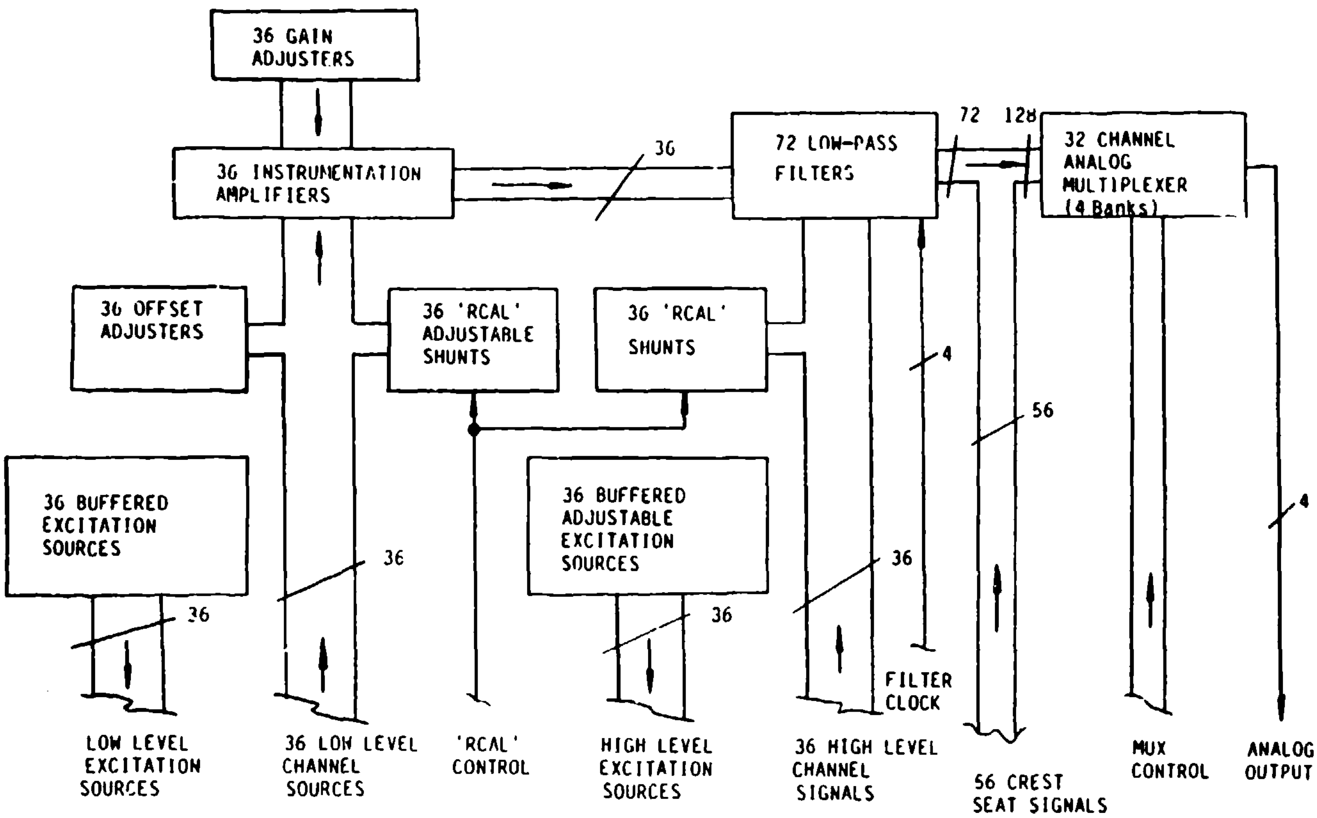



Figure 120. ADAM Signal Conditioning Block Diagram

3.2.1.1. Low Level Circuitry

The final design for the low level circuitry is shown in Figure 121. This circuitry is used to amplify sensor signal levels that are too small to be used without amplification. The low level circuit could be divided into the following sections.

- Buffered Excitation Source
- Offset Adjustment
- Instrumentation Amplifier
- Shunt Calibration Circuit

The design of each of the above circuits will be discussed separately.

3.2.1.1.1. Buffered Excitation Source

The buffered excitation source uses two resistors, HBUFF and LBUFF (Figure 121), to provide a buffered, or protected, source of excitation of any four-arm bridge piezoresistive sensor. The

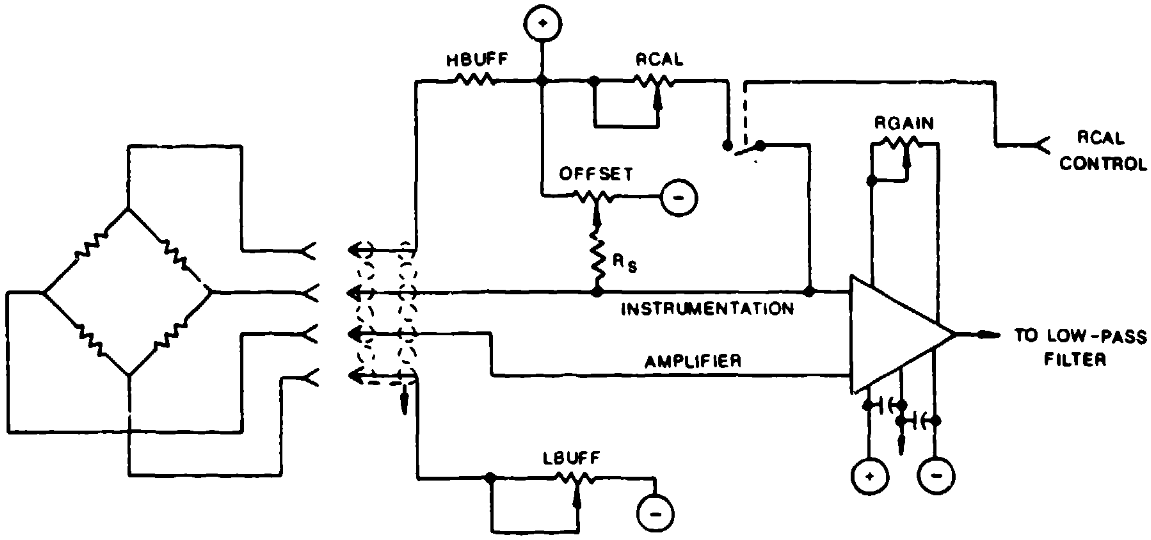


Figure 121. ADAM Low Level Signal Conditioning Schematic

positive excitation source is 10 VDC and the negative excitation source is -10 VDC. The values for HBUFF and LBUFF were selected to provide a range of excitation voltages for a variety of common sensors.

HBUFF was selected as a fixed 332 ohm resistor in order to save room on the circuit card assemblies. The LBUFF potentiometer value is 2 kohm. This value was selected to provide a range of excitation voltages in which all of the sensors in ADAM fell.

The circuit implemented will use both HBUFF and LBUFF as dropping resistors in a voltage divider circuit as shown in Figure 122. In order to get the proper excitation voltage across the sensor ($R_{sens}$ in Figure 122), the voltage drop across LBUFF is varied until the sensor voltage is correct. As Figure 122 implies, the actual voltage range that can be achieved is dependent on the input impedance of the sensor in use.

For a given sensor input impedance, the minimum and maximum sensor excitations may be calculated using the following formulas:

$$E_{min} = \frac{20\ R_{sens}}{2232 + R_{sens}}$$

and

$$E_{max} = \frac{20\ R_{sens}}{332 + R_{sens}}$$

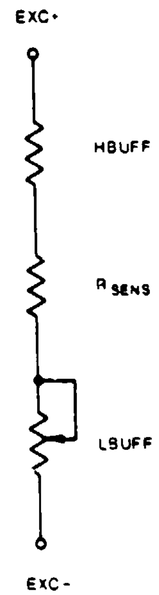



Figure 122. Low Level Channel Excitation Model

where $E_{min}$ is the minimum excitation voltage and $E_{max}$ is the maximum excitation voltage. These limits are considered to be theoretical limits since there is no safety factor included for cases where the sensor shorts one of its excitation lines to ground or to another excitation source.

The protection of the excitation sources from shorts to ground or the other excitation source is a second purpose of the HBUFF and LBUFF resistors. HBUFF is a 1/3 watt resistor that dissipates 0.30 watt when shorted to ground and 1.2 watts until failure when it is directly shorted to the negative supply. HBUFF will fail if it is shorted directly to the negative supply since it is unable to dissipate the 1.2 watts adequately, but this will prevent the positive supply from shorting to the negative supply. Since LBUFF serves to protect the negative excitation source, it is possible for the potentiometer to dissipate from 0.050 to 5 watts when shorted to ground and 0.200 to 20 watts when shorted to the positive supply. LBUFF is a 0.25 watt potentiometer that may fail due to excessive power dissipation. The possible failure of these devices is deemed an acceptable trade-off for the savings in space required by these components as opposed to the larger units required to survive any short condition.

3.2.1.1.2. Offset Adjustment

As shown in Figure 121, the offset adjustment is made using the offset adjust potentiometer. The offset adjustment will null offset errors by sinking or sourcing a small current into one arm of the sensor bridge to change the sensor output. This change is reflected at the amplifier output by a change in the DC offset. The offset adjustment potentiometer value is 500 kohm. The series resistor, $R_s$, is used to limit the current that the offset circuit may sink or source to the sensor to prevent sensor damage.

3.2.1.1.3. Instrumentation Amplifier

The instrumentation amplifier used in the system is a Burr Brown INA101 monolithic instrumentation amplifier. Table 34 compares the specifications of this amplifier to those specified in the ADAM SOW. As can be seen from the data presented in the table, the selected amplifier exceeds the one required by the SOW. The amplifier selection process is outlined in Section 3.3 of SRL Document 6885-01-86 and will not be repeated. The amplifier is a true differential input amplifier requiring a single resistor to program its gain. The equation used to determine the gain of the amplifier is:

$$\text{Gain} = (1 + \frac{40{,}000}{R_{gain}})$$

where $R_{gain}$ is the value of the gain setting resistor in ohms. Gains from 1 to 1000 may be programmed easily using this amplifier. Based on information available regarding the gains necessary for different sensors, a range of gains from 2 to 100 was established as satisfactory for most sensors. A 50 kohm potentiometer was selected for this purpose for most channels; however, some sensors require gains as high as 600. Those channels were provided with 2 kohm potentiometers to supply a gain from about 21 to 700. Two gain potentiometer values are used because the high gain channels would require a resistance value less than 1 percent of the total resistance of the 50 kohm potentiometers. This is a very difficult resistance value to hold so the smaller 2 kohm potentiometer values were used for high gain channels. The instrumentation amplifier is capable of supply output voltages in excess of ±5 volts with an excitation of ±10 volts, and the unit's power dissipation is a mere 135 milliwatts.

3.2.1.1.4. Shunt Calibration Circuit

The low level circuit also has provisions for a computer controlled shunt calibration signal. This shunt calibration signal (or $R_{cal}$ signal as it is more commonly called) provides a step change in the output signal level of the amplifier. This step change occurs because a resistor ($R_{cal}$) is shunted from the negative output of the sensor to the positive excitation sources. This resistor supplies a minute current to the sensor which causes the sensor bridge to become unbalanced. This produces a step change in the output of the amplifier. The size of the step change is dependent on the gain of

TABLE 34. COMPARISON OF INA101 SPECIFICATIONS TO ADAM SOW REQUIREMENTS

| Parameter | INA101 | ADAM SOW REQUIREMENT |
|---|---|---|
| Nonlinearity | 0.002 Percent FS | 0.39 Percent FS |
| Input Offset | 2 μV/°C | 15 μV/°C |
| Common Mode Rejection | 80 dB (Gain = 100) | 80 dB (Gain = 100) |
| Dynamic Response | 10 kHz (Gain = 10) | 10 kHz (Gain = 10) |
| Noise with Respect to Output | 0.8 μV (0.1 to 10 Hz)<br>18 μV (.01 to 1 kHz) | 230 μV (0.1 to 10 Hz)<br>130 μV (.01 to 1 kHz) |

the amplifier, impedance of the sensor, and the value of $R_{cal}$. By setting the gain of the amplifier and adjusting the $R_{cal}$ potentiometer, a known step change can be established. This will aid in the calibration of the sensor channel. If the gain of the channel and the sensor are the same, then the step change ("shunt calibration value") will be the same. If the gain has changed or the sensor is damaged or not connected, the shunt calibration value will change. This indicates that a problem exists with that channel.

The shunt calibration circuit is implemented with a Precision Monolithics, Inc., SW-05 Field Effect Transistor (FET) analog switch and a 200 kohm $R_{cal}$ potentiometer. The SW-05 was selected for its low power dissipation, and the 200 kohm potentiometer was selected to provide a wide range of resistance values for most sensors used in the ADAM system. The SW-05 switch control accepts a logic level signal input to control whether the switch is on or off. This allows computer control of the shunt calibration switching of low level circuitry.

3.2.1.2. High Level Circuitry

The high level circuitry is used for sensors whose outputs do not require an amplifier. There are two basic circuits that comprise the high level circuitry--the excitation circuitry and the shunt calibration circuitry (Figure 123). These circuits will be discussed in the following paragraphs.

3.2.1.2.1. Excitation Circuitry

The excitation circuitry for the high level circuit is very similar in function to the low level circuitry. Figure 123 shows that two potentiometers, HBUFF and LBUFF, provide a buffered

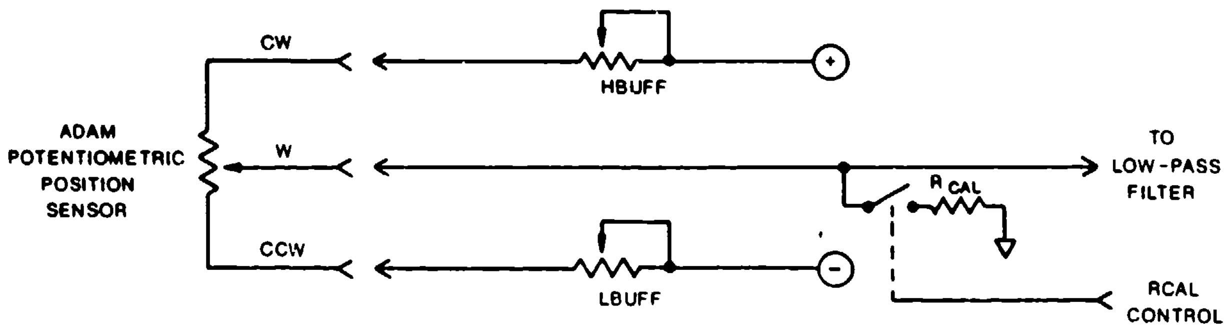



Figure 123. ADAM High Level Signal Conditioning Circuit

excitation source for the sensor. The sensor shown in Figure 123 is a potentiometric position sensor used in ADAM.

HBUFF and LBUFF are both 10 kohm potentiometers. The 10 kohm value was selected to provide the best adjustment for proper excitation of the high level sensors. This adjustment allows a maximization of the sensitivity of the position sensor while trimming offset errors at the same time.

The HBUFF and LBUFF resistors also serve to protect the ±10 volt power supplies from being shorted together if a sensor cable shorts a sensor excitation line to ground or to the opposite excitation supply. In these cases, it is possible for either potentiometer to dissipate from 10 milliwatts to 360 milliwatts when the excitation line is shorted to ground, and 40 milliwatts to 1.44 watts when shorted to the opposite supply voltage. Since these potentiometers are 250 milliwatt devices, some resistance settings will cause the component to fail if it dissipates too much power. Because the size trade-off involved with 1.5 watt potentiometers would require more printed circuit board space than is available, this was deemed an acceptable risk to allow a reasonable means of excitation adjustment. If a potentiometer does fail, it fails open, so it still serves to protect the excitation sources from shorting to each other.

3.2.1.2.2. Shunt Calibration Circuitry

Each high level circuit includes a shunt calibration circuit. The circuit uses a Precision Monolithics, Inc., SW-05 FET switch and a 3.4 kohm resistor. This shunt calibration circuit will switch the $R_{cal}$ resistor to the signal line. Since the $R_{cal}$ resistor is grounded (Figure 123), this provides a change in the sensor output impedance and results in a step change in the sensor output. This step change is a result of the output impedance and output voltage of the sensor. If the output voltage is near ground potential, then there will not be a sufficient impedance change to produce a step change. In the case of potentiometric position sensors, the value of the step change is dependent upon the output impedance of the sensor and it will change as the output impedance changes. While this shunt calibration measurement does not indicate the exact operational status of the high level circuit, it does indicate whether the sensor is attached and the sensor wiring is correct.

### 3.2.1.3. Antialiasing Filters

As Figure 120 indicates, the outputs of all the high level and low level channels are channelled into antialiasing filters. The low-pass filter is used to prevent aliasing errors from being introduced into a time sampled signal. Aliasing occurs when a time sampled signal has a frequency content greater than half the sampling frequency because those higher frequencies are "aliased" or folded back to appear as a lower frequency. This introduces an "aliasing" error in the sampled data. By filtering the high frequencies out of a signal prior to it being sampled, the aliasing errors can be reduced to an insignificant amount. The error cannot be completely eliminated in a signal since a low pass filters with infinite attenuation in the stop band cannot be implemented.

The low-pass filter in the ADAM instrumentation is a low pass filter implemented using CMOS switched capacitor technology. The device selected is a National Semiconductor MF-4 four pole Butterworth filter. A Butterworth filter was chosen for this application since the ripple (gain error) was the smallest of all the filter responses available. The ADAM SOW specifies a gain error in the pass band of no more than ±3 percent of the actual value. This corresponds to a value of +0.257 dB and -0.265 dB for the maximum allowable error in the pass band. The MF-4 filter has a maximum gain error of ± 0.15 dB so it exceeds the ADAM SOW requirement for gain error.

The switched capacitor technology was selected over other filter circuits because the cutoff frequency is set by a single clock driving the filters. The cutoff frequency of the filter is set by driving the filter with a clock having a frequency 50 times the desired cutoff frequency. The response of the switched capacitor filter is the same regardless of the cutoff frequency selected. The cutoff frequency can be set from 0.1 Hz to 10 kHz using a clock frequency from 5 Hz to 500 kHz.

The ADAM SOW requires that the aliasing error be 1.5 percent or less when the sampling frequency is five times the filter cutoff frequency. This requirement dictates that the filters used have a minimum of five poles in order to achieve the required degree of attenuation in the stop band. The method selected to implement the antialiasing filter in the ADAM instrumentation is shown in Figure 124.

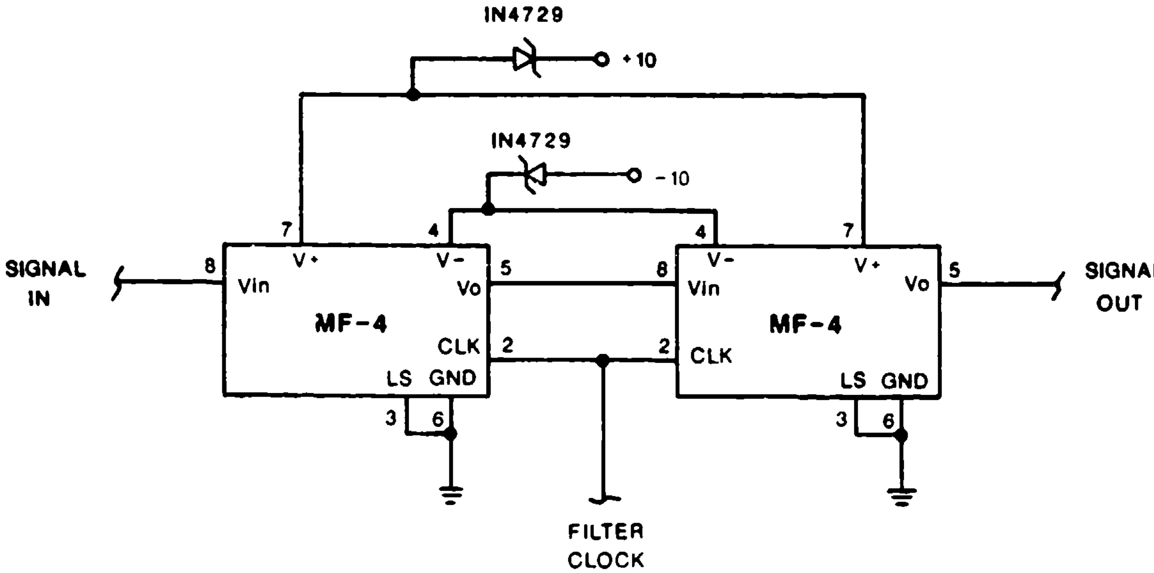



Figure 124. Antialiasing Filter Schematic

Two MF-4 filters are concatenated together to form an eight-pole, unity gain, Butterworth filter. A standard 0 to 5 volt clock is used to drive the filters and set the cutoff frequency. The clock frequency is computer controlled and is discussed in Section 3.2.3. The choice of the National Semiconductor MF-4 was made due to its small size, simple implementation, and Butterworth filter response. A problem encountered with the MF-4 is output offset voltages in the range of -200 mV to -600 mV. When two filters are concatenated together, the offset voltage of the two filters can be as high as 1.2 volts. The method that is used to provide offset adjustment for both the high level and the low level channels can be used to eliminate this offset error. Another problem with the use of these filters is the power supply requirements in order for the filters to provide a 5 volt output for a 5 volt input. The filters require a 6.1 volt to 7 volt supply range to have an output voltage swing of 5 volts. This required extra circuitry to generate the filter power supply. Figure 124 shows the zener diode supply circuit used to provide 6.4 volt supply. The power supply circuit description to discussed in-depth in Section 3.2.2.

3.2.1.4. Hybrid Microcircuit Development

A major design requirement for the ADAM instrumentation is that it must be completely contained within the manikin. The instrumentation design contains a large amount of circuitry that must fit within the space allotted in the viscera of the manikin. In order to accomplish this task, the miniaturization of these circuits is required.

A study was made which concluded that the signal conditioning circuitry required too much space to allow a standard circuit design using discrete components. A decision was made, therefore, to develop a hybrid microcircuit to replace the majority of the discrete components in the signal conditioning circuitry. A hybrid microcircuit is a miniaturization of a group of integrated circuits and components into a single, compact integrated circuit package.

A study was performed to determine the best mixture of high level and low level circuits required to optimize the space required by the hybrid microcircuit. Table 35 shows the number of pins required for power and control lines, a high level circuit, and a low level circuit. The study estimated the space required for different combinations of high level and low level channels, and determined which combination would likely yield the smallest package. The estimates were based on the size of the integrated circuit die (a die is an integrated circuit that is not mounted in a package), the number of pins required, and the size of standard integrated circuit packages. The result of the study indicated that the optimum hybrid design should contain four low level circuits and four high level circuits for a total of 40 pins required as a minimum. This package was estimated to require 2.2 square inches of printed circuit board space.

TABLE 35. PINS REQUIRED FOR EACH OF THE DIFFERENT CIRCUITS IN A HYBRID MICROCIRCUIT

| Circuit Description | Pins Required |
|---|---|
| Power and Control Lines | 8 |
| Low Level Circuit | 6 |
| High Level Circuit | 2 |

The final hybrid microcircuit layout and fabrication was done by Cincinnati Electronics based on the schematic shown in Figure 125. The package that Cincinnati Electronics used for the hybrid is shown in Figure126. Nine hybrids are required to provide 72 channels (36 high level channels and 36 low level channels) of signal conditioning. The power consumption of this device is 1.5 watts. Table 36 shows the typical quiescent currents for each power supply required by the hybrid. The use of this hybrid circuit reduced the packaging problem in such a way that it was possible for the signal conditioning circuitry to be designed on three circuit card assemblies.

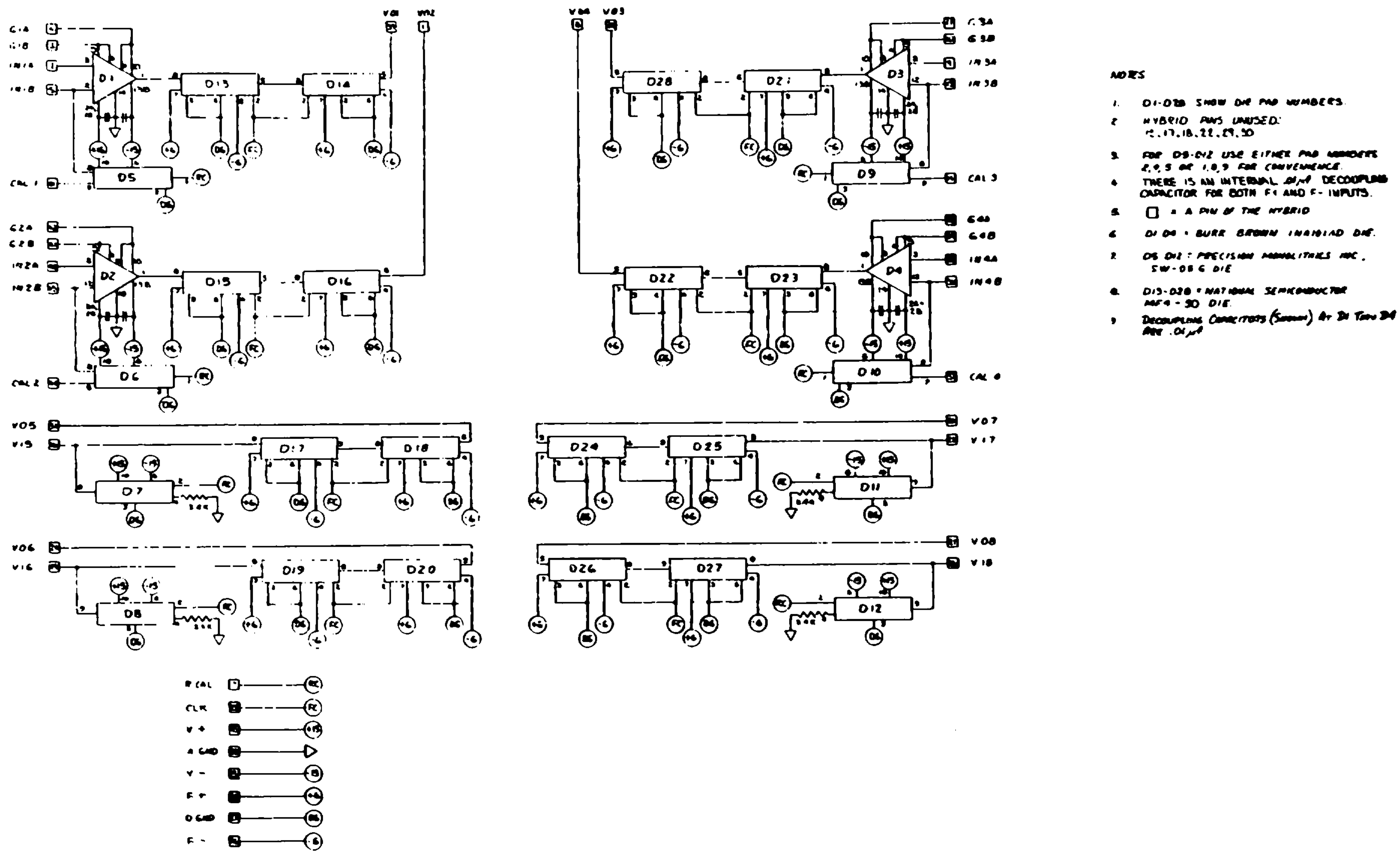



Figure 125. Hybrid Microcircuit Schematic

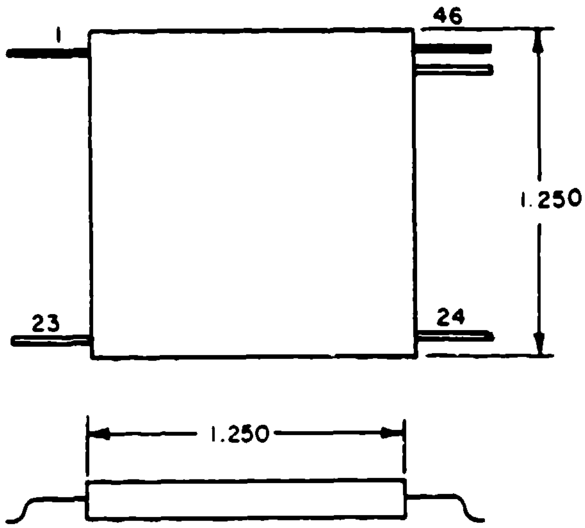



Figure 126. Hybrid Package Layout

TABLE 36. HYBRID MICROCIRCUIT QUIESCENT CURRENT

| Power Supply | Quiescent Current |
|---|---|
| +6.3 | 40 mA |
| -6.3 | -40 mA |
| +15 | 34 mA |
| -15 | -34 mA |

3.2.1.5. Analog Multiplexers

The last functional block shown in Figure 120 is the 32-channel multiplexer. A multiplexer is an electronic switch that is used to output one of several signals input to the device. The purpose of the multiplexers in the signal conditioning circuitry is to select one of 32 channels for digitization by the A/D converter. By using a multiplexer to channel one of 32 inputs to an A/D converter, the need for an A/D converter for each channel is eliminated. This reduces the space required by the system electronics and reduces the power consumption.

The multiplexers chosen for the ADAM instrumentation were the Burr Brown MPC800KG 16 to 1 high speed analog multiplexers. These devices were selected due to their low power consumption (500 mW), high speed (250 nanosecond settling time), and expandability to 32 channels. A representative schematic of the multiplexer circuit implemented in the ADAM instrumentation is shown in Figure 127.

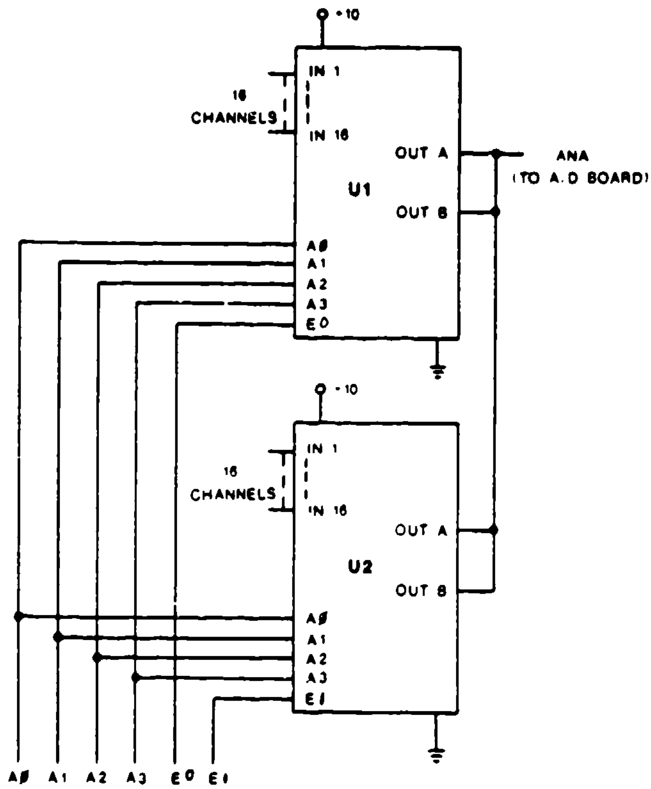



Figure 127. 32-Bit Multiplexer Schematic

A 32-channel multiplexer is developed from two 16-channel multiplexers with their outputs tied together. The four address select lines, A0 to A3, are used to select the channel that is to be directed to the output and are controlled from the A/D conversion board. The enable lines, E0 and E1, are used to select which multiplexer, U1 or U2, has its output enabled and is active on the ANA line that goes to the A/D board. The multiplexer that is not enabled is in a tri-state mode and will not affect the signal that is currently active on the ANA line. Table 37 shows the actual decoding combinations to select each channel based on the address select lines and the enable lines. The speed of the multiplexers was desired to be high so that the minimum setup time was available before the next channel of data could be sampled. This setup time is the time required for the multiplexer to change its output to that programmed by the address select lines and the enable lines. This setup time is defined as the sum of the access time and the settling time of the multiplexer. The access time is the time required for the multiplexer to turn a new channel ON after a new address has been applied to the address inputs, and settling time is the time required for the output of the multiplexer to reach and maintain of specified value within an error band in response to a step input. For the MPC800KG the access time is 200 nanoseconds (maximum), and the settling time to an error of 0.1 percent is 250 nanoseconds. The total setup time for a newly selected channel is 450 nanoseconds. After a new multiplexer channel is selected, 450 nanoseconds must pass before the output of the multiplexer represents the actual signal value of the channel selected.

TABLE 37. MULTIPLEXER CHANNEL FOR EACH ENABLE LINE

| E0 | E1 | A3 | A2 | A1 | A0 | Multiplexer Channel (HEX) |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 2 |
| 1 | 0 | 0 | 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 1 | 0 | 0 | 4 |
| 1 | 0 | 0 | 1 | 0 | 1 | 5 |
| 1 | 0 | 0 | 1 | 1 | 0 | 6 |
| 1 | 0 | 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 1 | 0 | A |
| 1 | 0 | 1 | 0 | 1 | 1 | B |
| 1 | 0 | 1 | 1 | 0 | 0 | C |
| 1 | 0 | 1 | 1 | 0 | 1 | D |
| 1 | 0 | 1 | 1 | 1 | 0 | E |
| 1 | 0 | 1 | 1 | 1 | 1 | F |
| 0 | 1 | 0 | 0 | 0 | 0 | 10 |
| 0 | 1 | 0 | 0 | 0 | 1 | 11 |
| 0 | 1 | 0 | 0 | 1 | 0 | 12 |
| 0 | 1 | 0 | 0 | 1 | 1 | 13 |
| 0 | 1 | 0 | 1 | 0 | 0 | 14 |
| 0 | 1 | 0 | 1 | 0 | 1 | 15 |
| 0 | 1 | 0 | 1 | 1 | 0 | 16 |
| 0 | 1 | 0 | 1 | 1 | 1 | 17 |
| 0 | 1 | 1 | 0 | 0 | 0 | 18 |
| 0 | 1 | 1 | 0 | 0 | 1 | 19 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1A |
| 0 | 1 | 1 | 0 | 1 | 1 | 1B |
| 0 | 1 | 1 | 1 | 0 | 0 | 1C |
| 0 | 1 | 1 | 1 | 0 | 1 | 1D |
| 0 | 1 | 1 | 1 | 1 | 0 | 1E |
| 0 | 1 | 1 | 1 | 1 | 1 | 1F |

Four circuits like the one shown in Figure 127 are used to provide a bank of four 32-channel multiplexers. This provides for 128 channels of data to be multiplexed into four lines of data. These four lines are then channelled to its own A/D converter for processing by the digital subsystem. These four 32-channel multiplexers are sufficient to manage all 22 manikin channels and 56 external channels of the ADAM system.

### 3.2.2. Signal Conditioning Boards

The circuits and descriptions in Sections 3.2.1.1 through 3.2.1.5 were used as basic building blocks for the circuit card assemblies that are the signal conditioning circuitry for the ADAM instrumentation. The first assembly that will be described is the analog front-end interface board (AFIB) which contains 32 channels on board. Sixteen channels are high level channels, and 16 channels are low level channels. The second board to be discussed is the CREST interface board (CRIB) which contains 64 channels. Four channels are high level channels, four channels are low level channels, and 56 channels are for external signals.

#### 3.2.2.1. Analog Front-End Interface Board (AFIB)

The AFIB circuit and assembly contains signal conditioning circuitry for 32 manikin data channels. Sixteen of these channels are high level channels, and 16 channels are low level channels. There is a single 32-channel multiplexer circuit on board the AFIB. Figure 128 is a block diagram of the AFIB board showing this detail plus the power supply block and the control signals that come from the A/D board. Figures 129 and 130 are the two sheets that comprise the AFIB schematic. While the majority of the schematic has been discussed in detail in previous sections, this discussion focuses on the system level design concepts of the AFIB.

Figure 130 shows the schematic of the power supply section of the AFIB. The +10 volt positive excitation voltage is generated from U8 and its associated circuitry. Q1 acts as a high current pass transistor under the control of U8. The purpose of Q1 is to allow the voltage regulator integrated circuit to regulate a voltage at a much higher current output than is possible for the integrated circuit itself. By using Q1, the voltage regulation on this line is as good as the voltage regulator itself, but the current carrying capacity increases from 40 mA to 800 mA. The output voltage is adjusted using R131, and the excitation voltage is established at 10 volts. The excitation voltage for the positive filter supply (F+) is established by the zener diode VR1. The drop in voltage is 3.7 volts across VR1 which produces a voltage of 6.3 volts for the filters. Since the four hybrids require

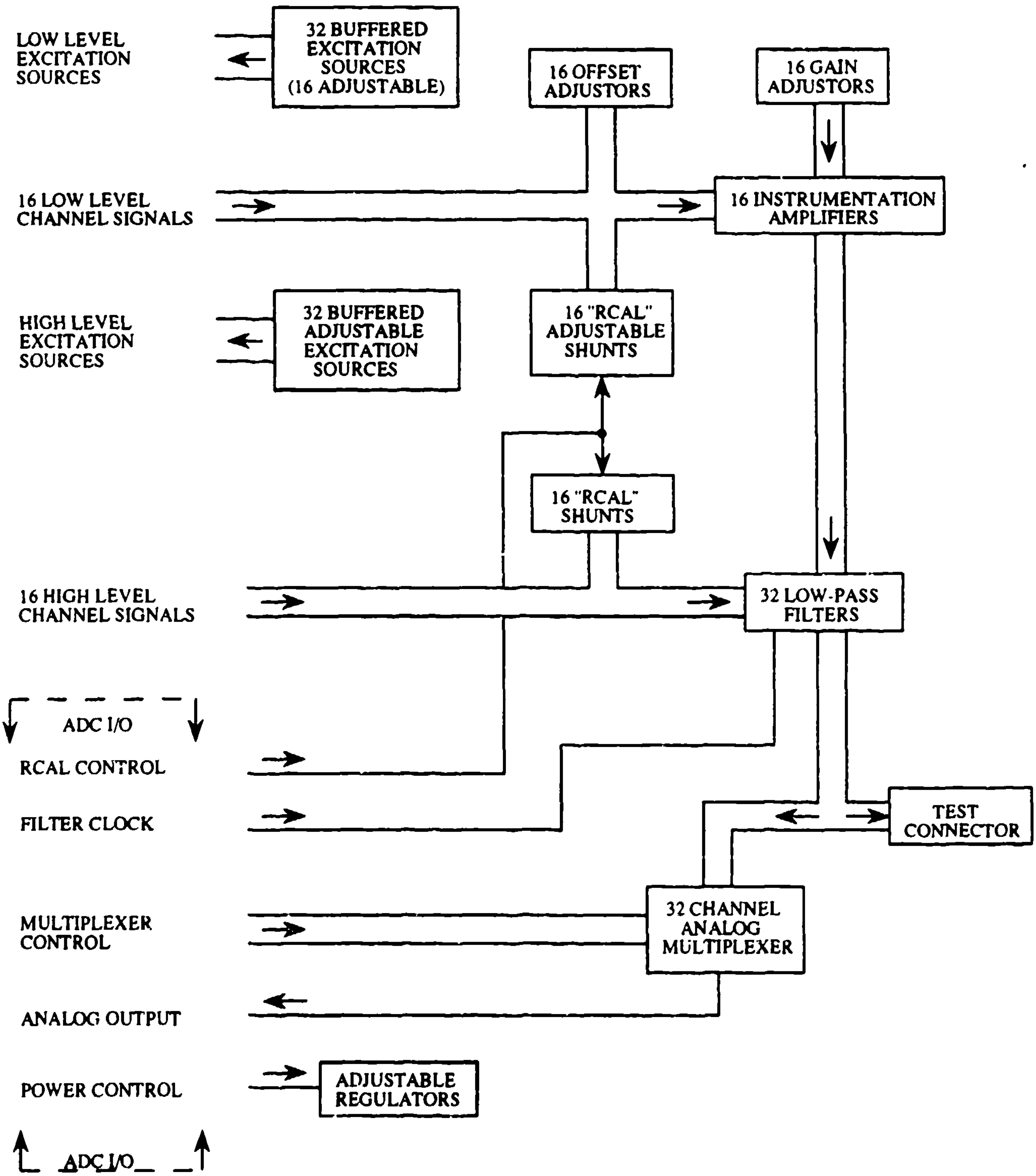



Figure 128. AFIB Block Diagram

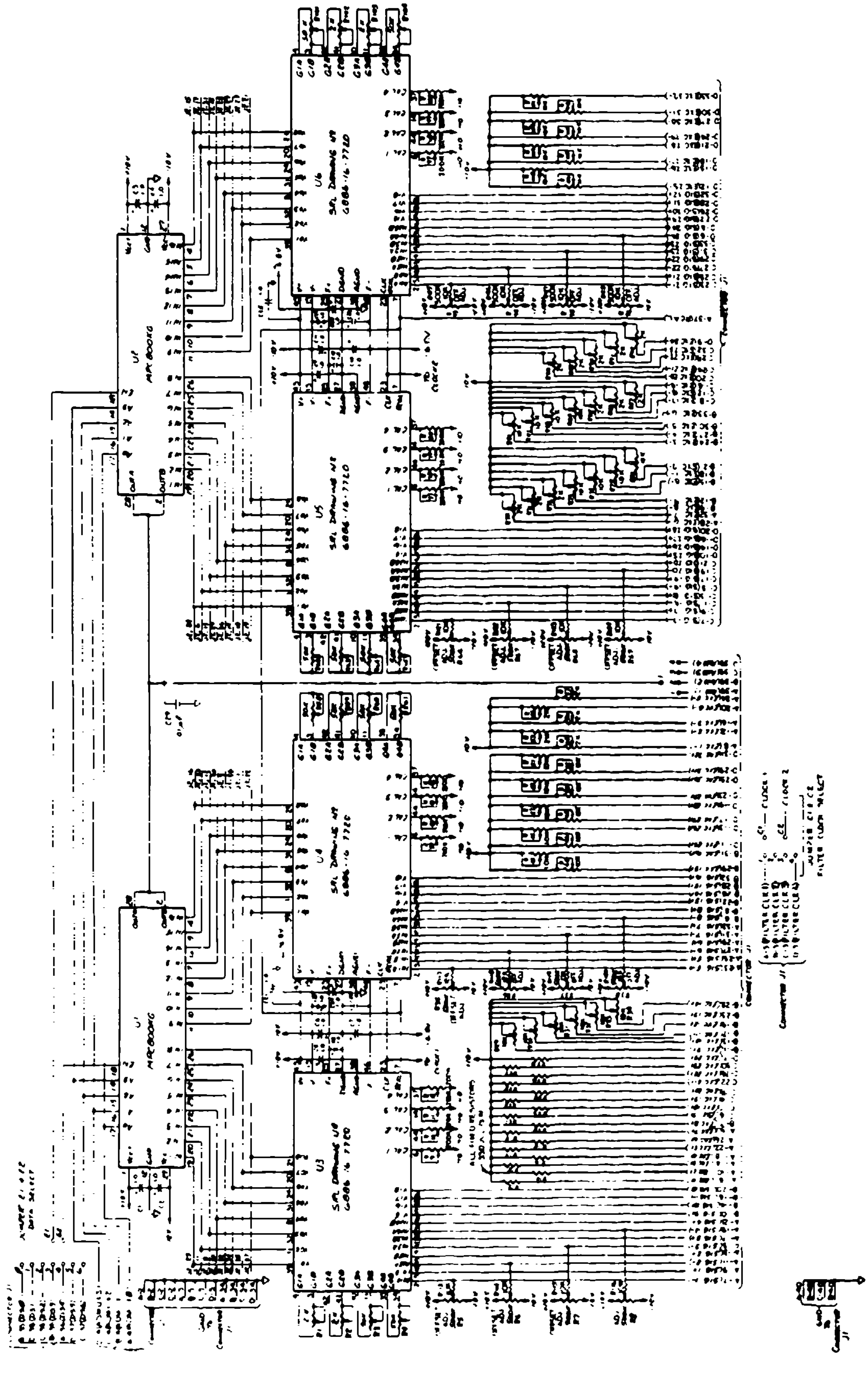


Figure 129. AFIB Schematic

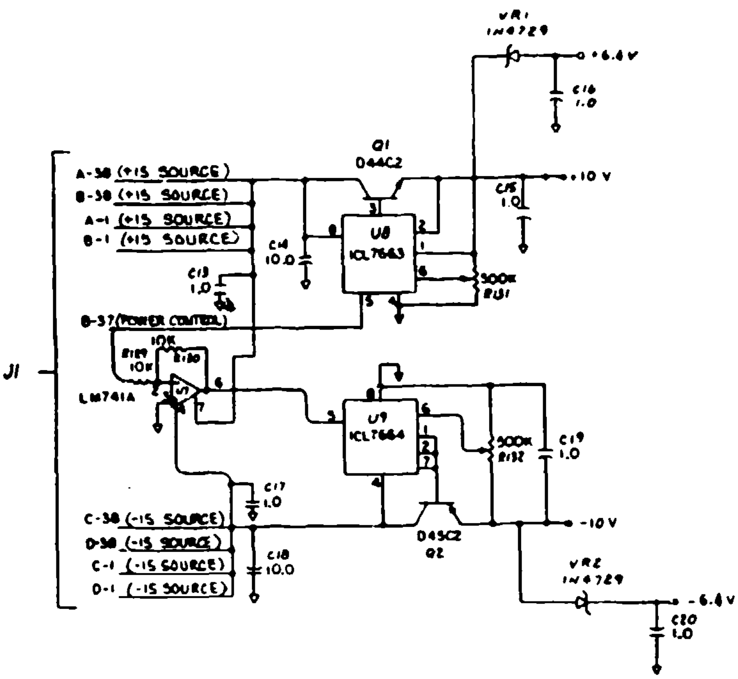



Figure 130. AFIB Schematic

approximately 250 mA to operate the filters, the load of the filters is sufficient to keep the zener diode conducting in the zener mode to maintain regulation.

The negative excitation (-10 volts) and the negative filter supply (-6.3 volts) are implemented in the same manner as the positive supply. The negative voltage regulator, U9, uses a pass transistor, Q2, to bring the power supply current carrying capacity up to 800 mA from 40 mA, and a zener diode VR2 is used to provide the negative power supply for the filters used in the hybrids (U3-U6, Figure 129). Both regulators may be shut down to reduce the system power consumption. The power control line is used to shut the regulators off. When the power control line is in a logic high level (5 volts), then U9 shuts down the positive supplies. The negative voltage regulator, U9, requires -5 volts to shut it down so U7 is an LM741 op amp acting as a simple inverter to invert the power control line and shut down the negative supply. Since the analog circuitry dissipates a large amount of heat, this feature is desirable in order to reduce the heat buildup from the instrumentation.

The majority of the AFIB circuitry is built into the hybrid microcircuits (U3-U6, Figure 129), and the circuitry on the AFIB consists mostly of the hybrids, the multiplexers, and the excitation adjustment potentiometers. Figure 129 shows the outputs of two hybrid microcircuits are directed to the 16 inputs of their respective multiplexer. The sensor input to the hybrids come from the connector J1. This connector mates with the analog mother board connector which is discussed in

Section 3.2.2.3. The excitation potentiometers for each channel have their outputs channelled to connector J1, where the analog mother board distributes the sensor excitation voltages and returns the sensor signals to the AFIB.

The outputs of the two multiplexers, U1 and U2, are tied to a single line to be sent to the A/D board for digitization. A jumper, labeled 1, is available to jumper the output of the multiplexers to one of four lines (ANA1-ANA4) leading to the A/D board. These lines are part of the ADC I/O lines shown in Figure 128.

The multiplexer address select lines, ADMUX0-ADMUX3 are on the ADC I/O lines and are used to select the proper multiplexer address. The data select lines, DSO-DS6 are jumpered to the enable lines E1 and E2. Table 37 shows the multiplexer addresses that will be selected for each data select line.

The clock that is used to drive the switched capacitor filters on the hybrids is jumper selectable also. Four filter clocks are on the ADC I/O lines, and the clocks may be jumpered to either C1 or C2. Line C1 drives the filters off the hybrids U3 and U4. Line C2 drives the filters off the hybrids U5 and U6.

The last line of the ADC I/O lines is the Rcal control line. When this line is in a logic high (5 volts), the hybrids are in standard output (non-$R_{cal}$ mode). When the $R_{cal}$ control line is a logic low (0 volts), the hybrids are in the shunt calibration mode ($R_{cal}$ mode) and the real resistor is shunting the sensor as described earlier.

The final detail concerning the AFIB is the connector J2. The inputs of the multiplexers, U1 and U2, are also connected to the test connector J2. This connector allows the measurement of the output signals from the hybrids and was placed on the circuit card to allow simple calibration and troubleshooting of the sensor channels on the board.

Figure 131 shows the component side assembly drawing for the AFIB. The hybrid microcircuits occupy the majority of the board space (U3-U6), and the test connector, J2, and multiplexers, U1 and U2, occupy the center space of the AFIB. The positive voltage regulator circuit is located on the bottom right side of the board. The board dimensions are also outlined on the drawing.

Figure 132 shows the assembly drawing for the solder side of the AFIB. This is the side where the majority of the 98 potentiometers in the AFIB design are located.

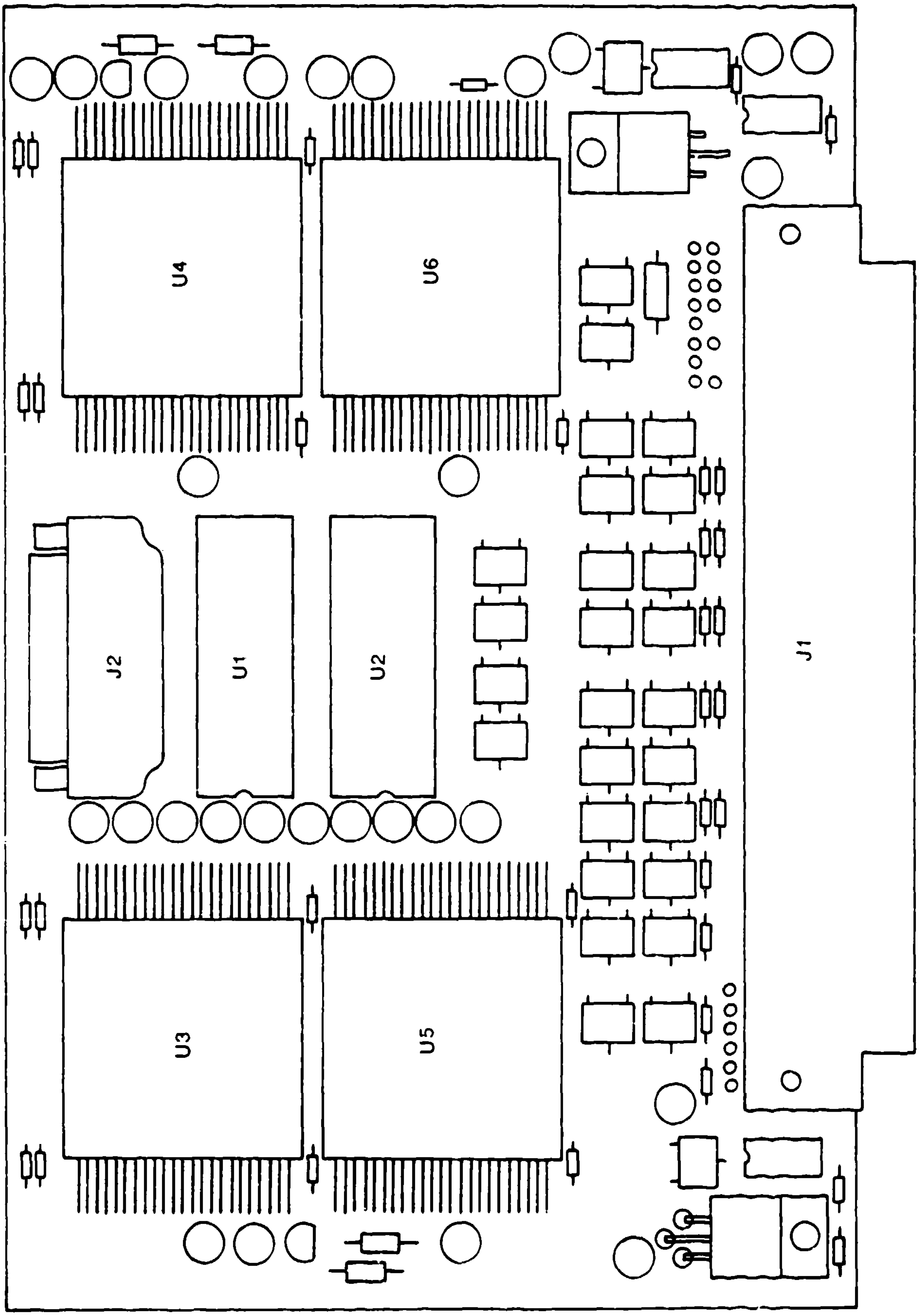



Figure 131. AFIB Assembly--Component Side

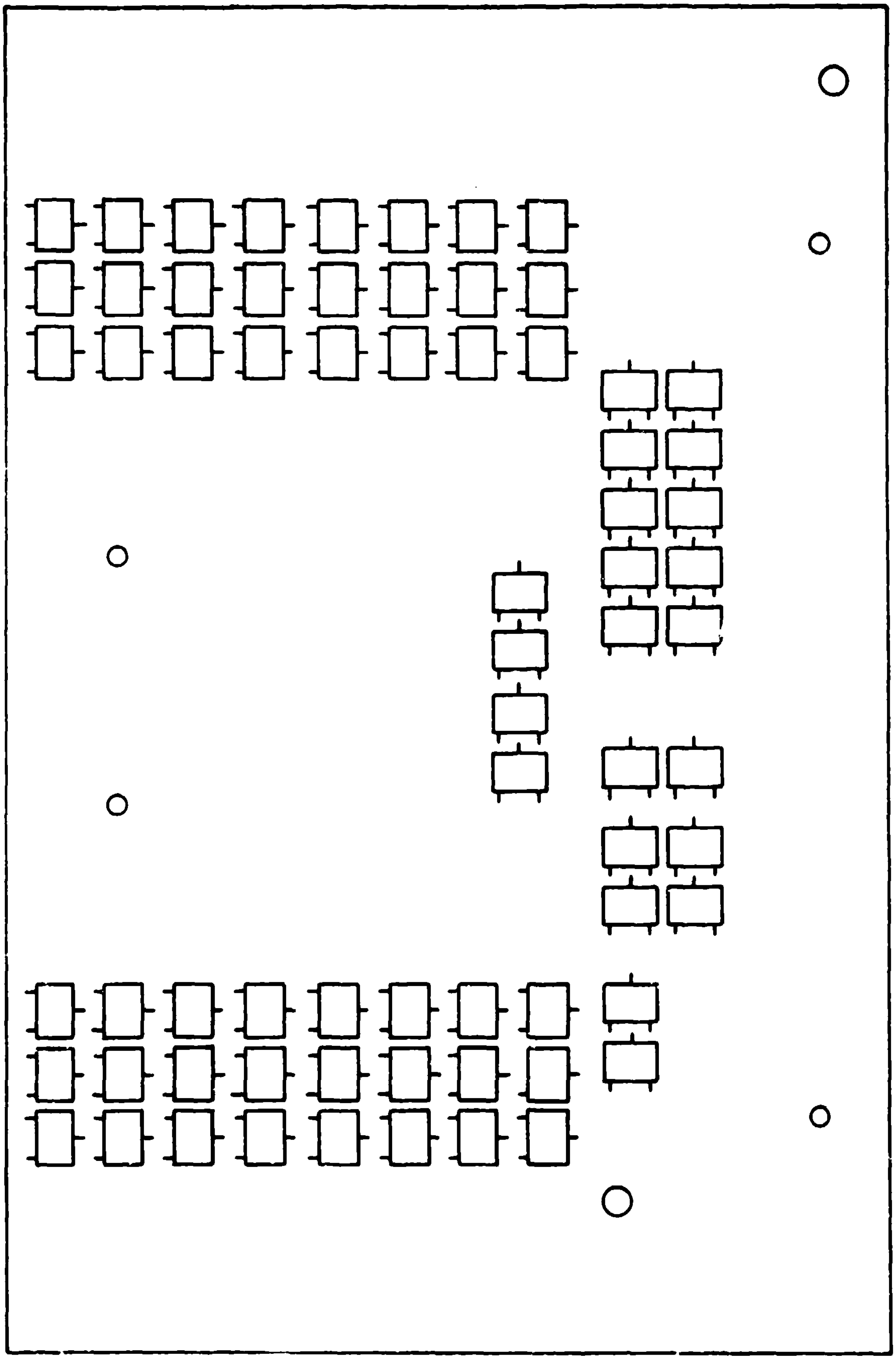


Figure 132. AFIB Assembly--Solder Side

Because the manikin requires a great deal of signal conditioning circuitry, two AFIBs are used to provide most of the signal conditioning for the ADAM system. By design, the jumpers for the ADC I/O lines allow the two AFIBs to function in the system without interference. Table 38 shows the jumper assignment for AFIB No. 1, and Table 39 shows the jumper assignments for AFIB No. 2. These two boards provide 64 channels of signal conditioning. The remaining eight signal conditioned channels and the 56 channels of external data are on the third and last analog signal conditioning board--the CREST interface board.

3.2.2.2. CREST Interface Board (CRIB)

The CRIB contains eight channels of signal conditioning, 48 channels of data that have no signal conditioning, and eight channels of data that have either no signal conditioning or signal conditioning that is the equivalent of one hybrid microcircuit. As intended for its original purpose, the CRIB contains eight channels with signal conditioning and 56 channels for externally conditioned data. An optional hybrid may be installed to convert eight of the 56 externally conditioned channels to internally signal conditioned channels. A block diagram of the CRIB illustrating this fact is shown in Figure 133. The block diagram shows that there is circuitry on the CRIB to provide four high level channels and four low level channels plus an additional four high level channels and four level channels when the optional hybrid microcircuit is installed. If the optional microcircuit is not installed, then the appropriate jumpers are installed to provide circuitry for eight externally conditioned channels.

The schematic for the CRIB is shown in Figures 134 and 135. Figure 134 shows the schematic for the first set of 32 channels and the power supply circuitry. The power supply for the CRIB is the same in composition and function as the power supply for the AFIB and it will not be repeated in this discussion. The hybrid, U3, contains the signal conditioning circuitry for eight manikin channels. These eight plus the 64 from the two AFIBs complete the complement of 72 manikin channels in the ADAM system. The optional hybrid, U4, can provide eight channels of signal conditioning when the jumpers J4 and J5 are installed, or U4 can be jumpered completely, out of the circuit by making the appropriate connections from jumper J4 to jumper J5. The latter mode is the manner in which the CRIB is implemented in the ADAM system. In order to reduce the power consumption of the circuitry, the hybrid U4 is not installed on the CRIB in the ADAM system.

TABLE 38. AFIB#1 JUMPER ASSIGNMENTS

| Signal Label (J1's Row-Pin) | Connected To |
|---|---|
| DSO (A-35) | E2 |
| DS1 (D-36) | E1 |
| DS2 (C-36) | No Connection |
| DS3 (B-36) | No Connection |
| DS4 (A-36) | No Connection |
| DS5 (D-37) | No Connection |
| DS6 (C-37) | No Connection |
| FILTER CLK1 (A-3) | C1 and C2 |
| FILTER CLK2 (B-3) | No Connection |
| FILTER CLK3 (C-3) | No Connection |
| FILTER CLK4 (D-3) | No Connection |
| ANA1 (A-39) | 1 |
| ANA2 (B-39) | No Connection |
| ANA3 (-39) | No Connection |
| ANA4 (D-39) | No Connection |

TABLE 39. AFIB#2 JUMPER ASSIGNMENTS

| Signal Label (J1's Row-Pin) | Connected To |
|---|---|
| DSO (A-35) | E2 |
| DS1 (D-36) | E1 |
| DS2 (C-36) | No Connection |
| DS3 (B-36) | No Connection |
| DS4 (A-36) | No Connection |
| DS5 (D-37) | No Connection |
| DS6 (C-37) | No Connection |
| FILTER CLK1 (A-3) | No Connection |
| FILTER CLK2 (B-3) | C1 AND C2 |
| FILTER CLK3 (C-3) | No Connection |
| FILTER CLK4 (D-3) | No Connection |

TABLE 39. AFIB#2 JUMPER ASSIGNMENTS (continued)

| Signal Label (J1's Row-Pin) | Connected To |
|---|---|
| ANA1 (A-39) | No Connection |
| ANA2 (B-39) | 1 |
| ANA3 (C-39) | No Connection |
| ANA4 (D-39) | No Connection |

The outputs of U3 and the eight external signals are fed to the multiplexer U1, and another 16 external signals are fed to the multiplexer U2. The output of the multiplexers are connected together and wired to the jumper J2A. Jumper J2A allows the output of this 32 channel multiplexer circuit to be wired to one of the ADC I/O lines ANA1-ANA4 which take the signal to the A/D board for digitization.

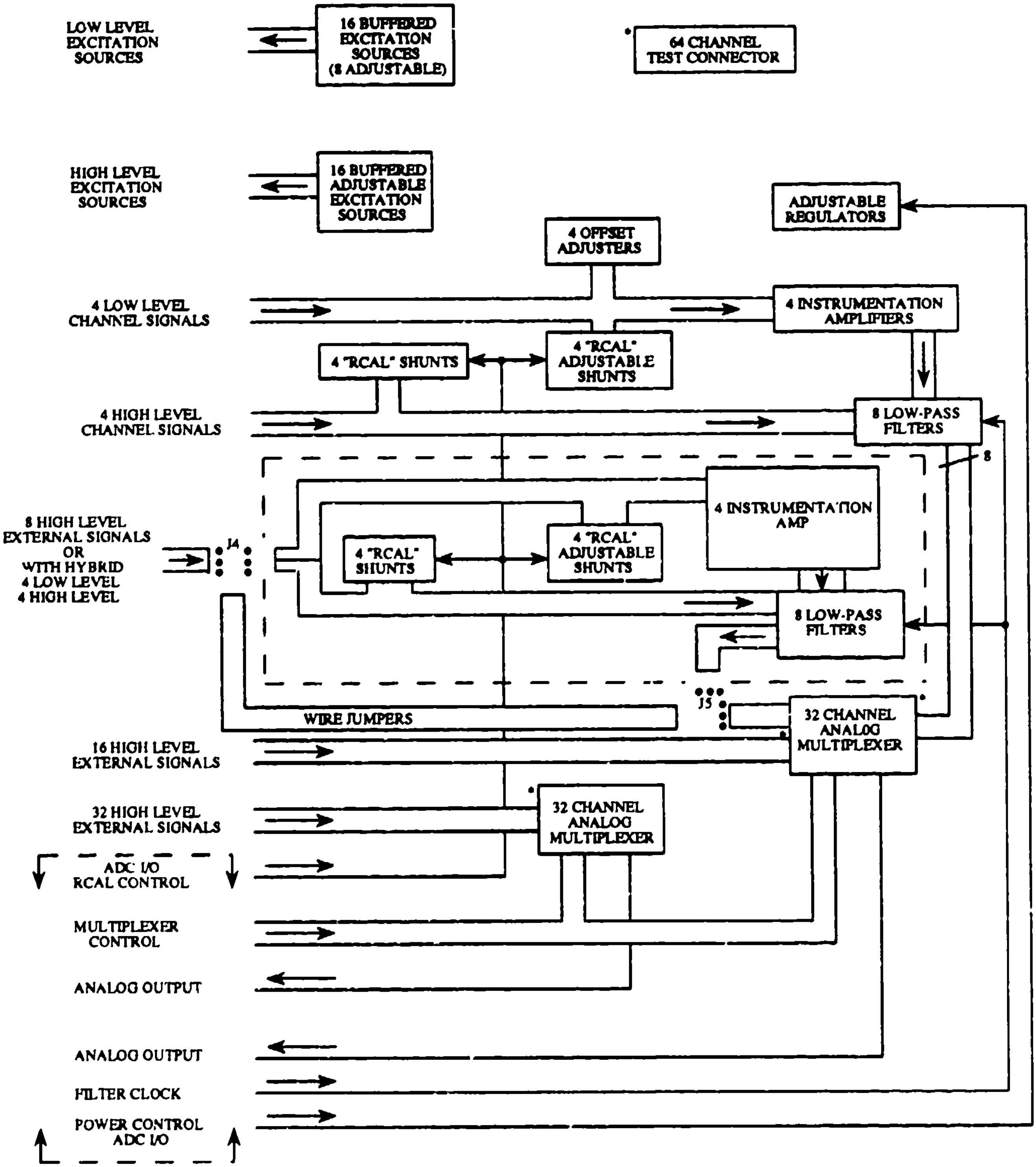
Figure 135 shows the schematic for the second 32 channel multiplexer circuit. This schematic shows that 32 external channels are brought to the inputs of the multiplexers U5 and U6. The outputs of U5 and U6 are wired together to go to jumper J2B. Jumper J2B allows the multiplexer output to be jumpered to one of the ADC I/O lines ANA1-ANA4. The ANA line will connect the multiplexer output to the A/D board for digitization.

Jumper J1 in the top left corner of Figure 134 is used to establish the address range of the multiplexers by jumping the data select lines, DS0-DS6, to the multiplexer enable pins. The appropriate multiplexer address for each data select line is outlined in Table 37.

The multiplexer channel is selected by the ADC I/O lines ADMUX0-ADMUX3 and are shown in the top left corner of Figure 134.

The hybrids, U3 and U4, receive the filter clock from jumper J3. This jumper allows any of the four filter clocks to be selected to drive the filters for U3 and U4.

The $R_{cal}$ control line is also used for the circuits contained in U3 and U4 to switch the channels into the shunt calibration mode when the line is in the logic low (0 volts) state. The channels are in their standard output mode when the $R_{cal}$ control line is in the logic high (5 volts) state.




• HIGH LEVEL INPUTS TO 32 CHANNEL ANALOG MULTIPLEXERS ARE PARALLEL CONNECTED TO 64 CHANNEL TEST CONNECTOR J2.

Figure 133. CRIB Block Diagram

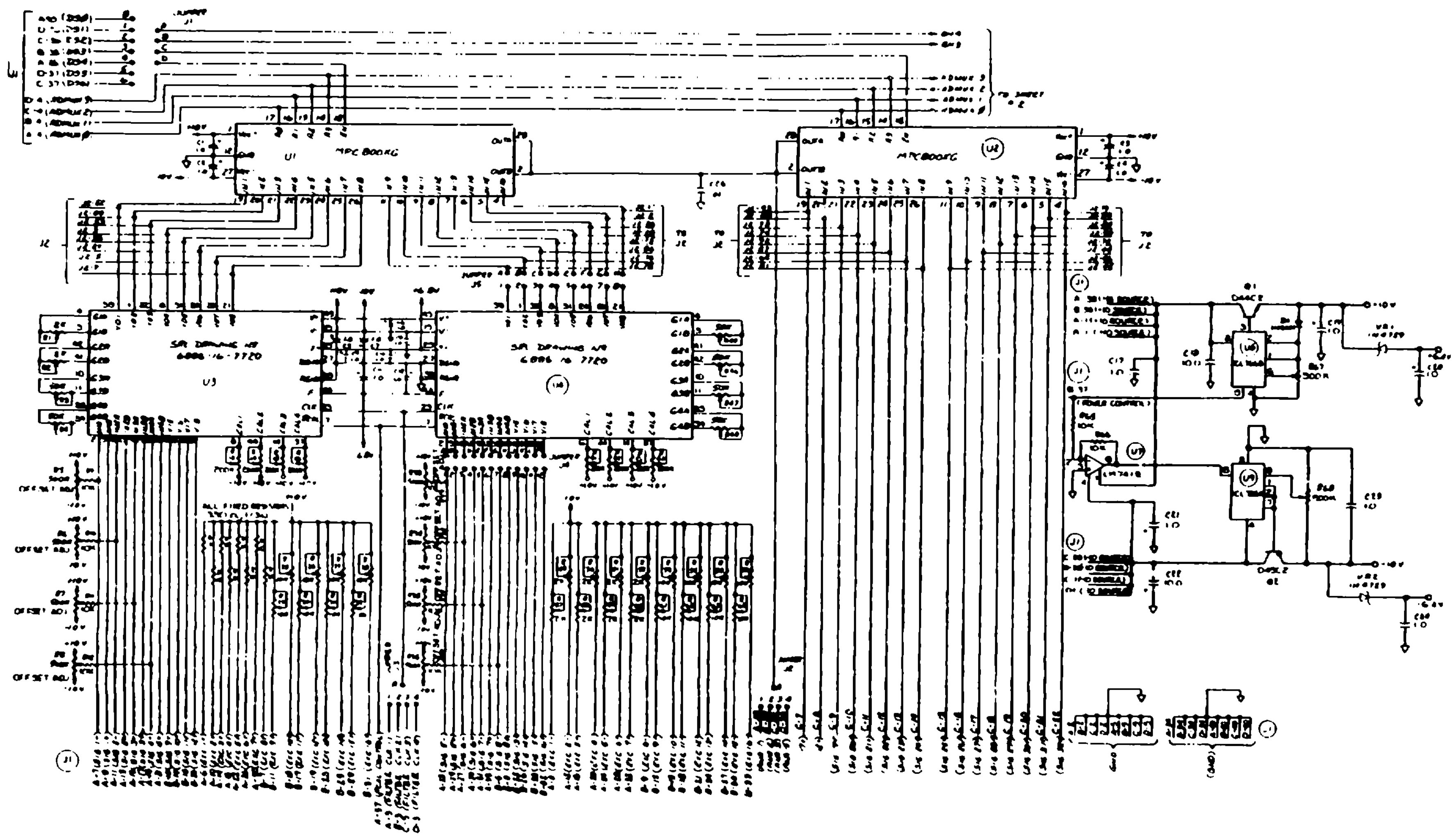



Figure 134. CRIB Schematic

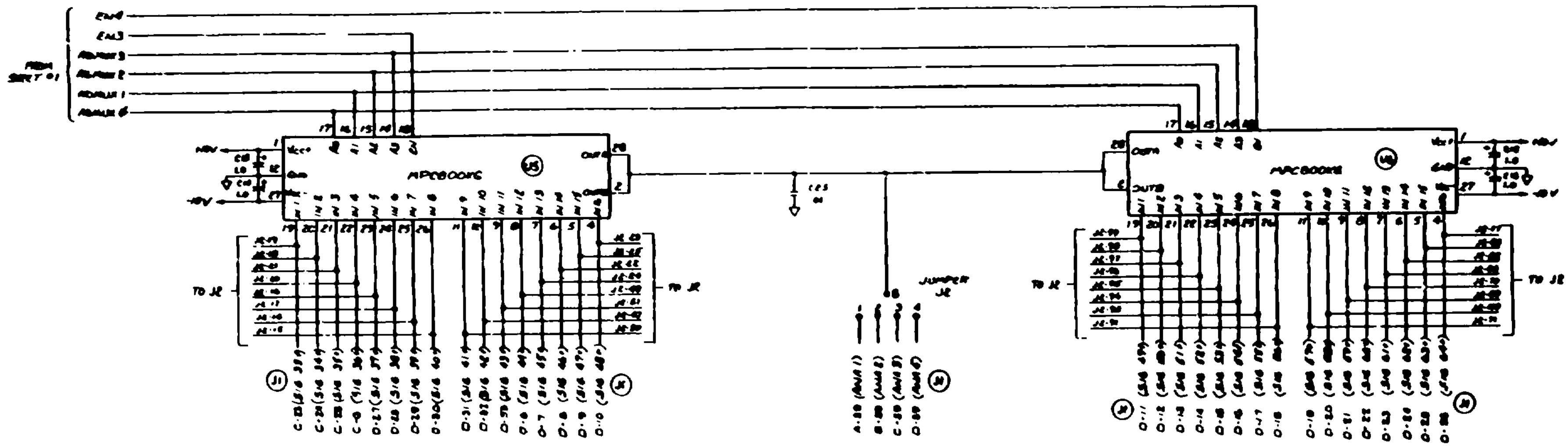


Figure 135. CRIB Schematic

A test connector, J2, is provided to allow measurement of the voltage at the inputs to the multiplexers U1, U2, U5, and U6. This was done to provide a means to assist in the trouble-shooting and test of the circuit card, analog mother board, and sensor wiring. The connector chosen was a 100 pin microminiature "D" type connector.

Figure 136 is an assembly drawing of the CRIB. The test connector, J2, is in the upper left corner and the hybrids U3 and U4 are right of centers. All of the potentiometers for the high level and low level channels are located on the component side of the board. The positive regulator is located in the lower left corner and the negative power supply in the lower right. The multiplexers U1, U2, U5, and U6 are located near the center and left centers regions of the board. The dimensions of the board are in Figure 136 and are the same as the AFIB. Figure 136 also shows the mother board connector, J1, that is used to provide the connection to the analog mother board so that the sensor signals, ADC I/O lines, buffered excitation, and power signals may be connected to the board.

### 3.2.2.3. Analog Mother Board

The analog mother board provides all of the interconnections between the two AFIBs, the CRIB, the sensors, the power supply sources, and the digital subsystem. Figure 137 is an assembly drawing of the analog mother board. On the component side of the board, there are three 160 pin connectors that mate with the connectors on the AFIBs and the CRIB. The solder side of the mother board has 10 cable assemblies soldered to the board. Seven cable assemblies lead to the limb connectors for the arms, legs, pelvis, chest, and head. Two cable assemblies are for external channels, and the last cable is a ribbon cable assembly that carries all of the ADC I/O signals to and from the A/D board.

Figure 138 shows the layout of the analog mother board. It shows that the signals that are common to all three boards are located on the top and bottom sections of the board and the signals that are unique to each board, the sensor excitation and signal lines, are in the center section of the board. The 160 pin connectors are mounted in the slots marked J1, J2, and J3. The sensor cable interconnections are made in the areas marked SC1 for J1, SC2 for J2, and SC3 for J3. The pin assignments for J1 (the CRIB) are shown in Table 40, and the pin assignments for J2 and J3 (the AFIB) are shown in Table 41.

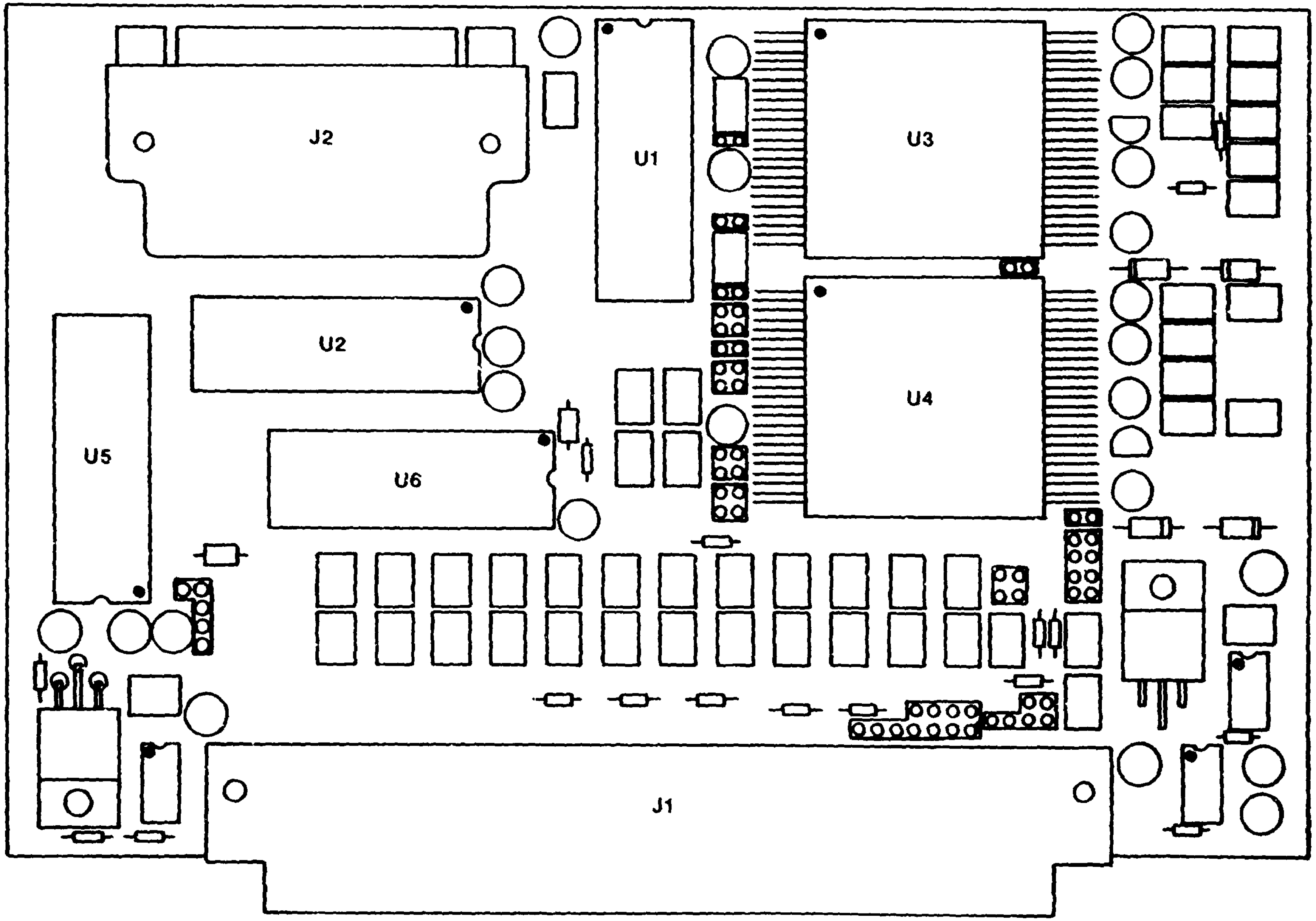



Figure 136. CRIB Assembly

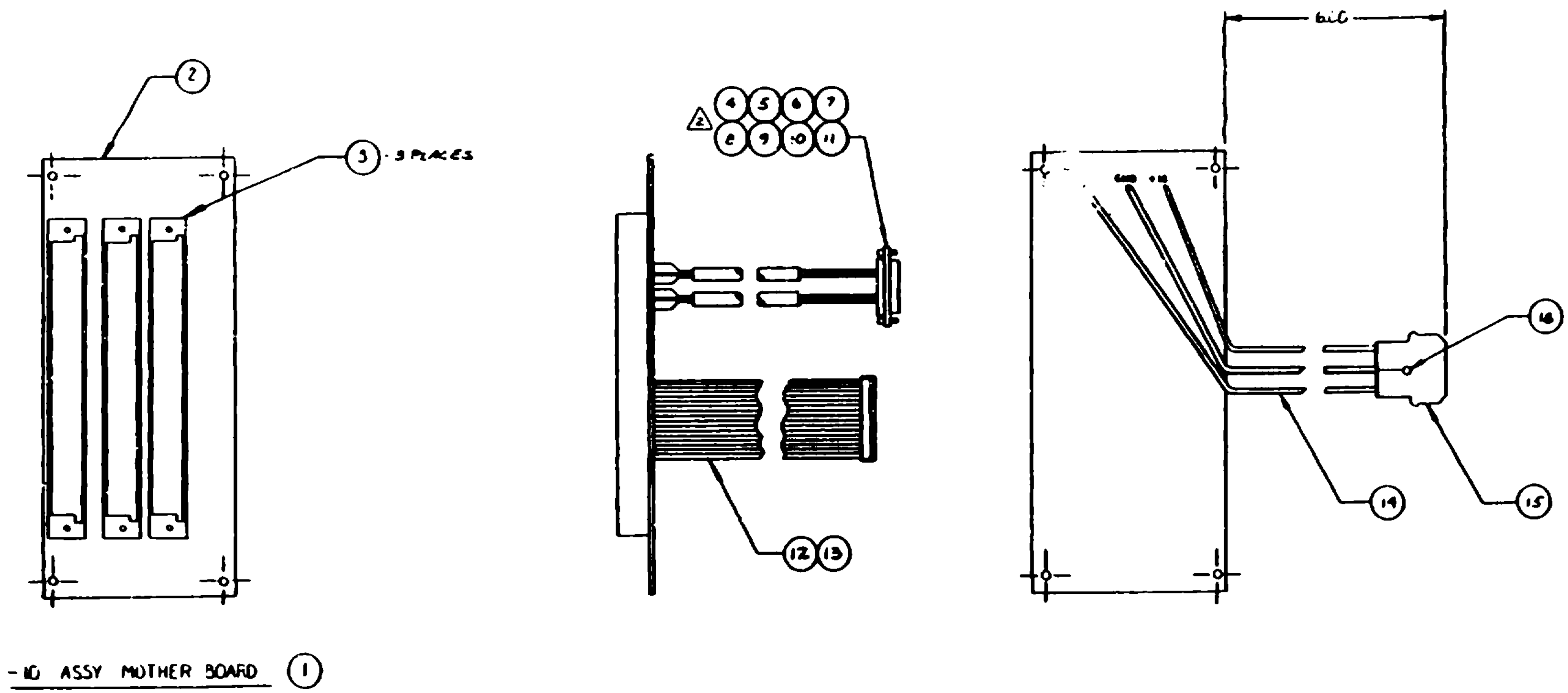



Figure 137. Analog Mother Board Assembly

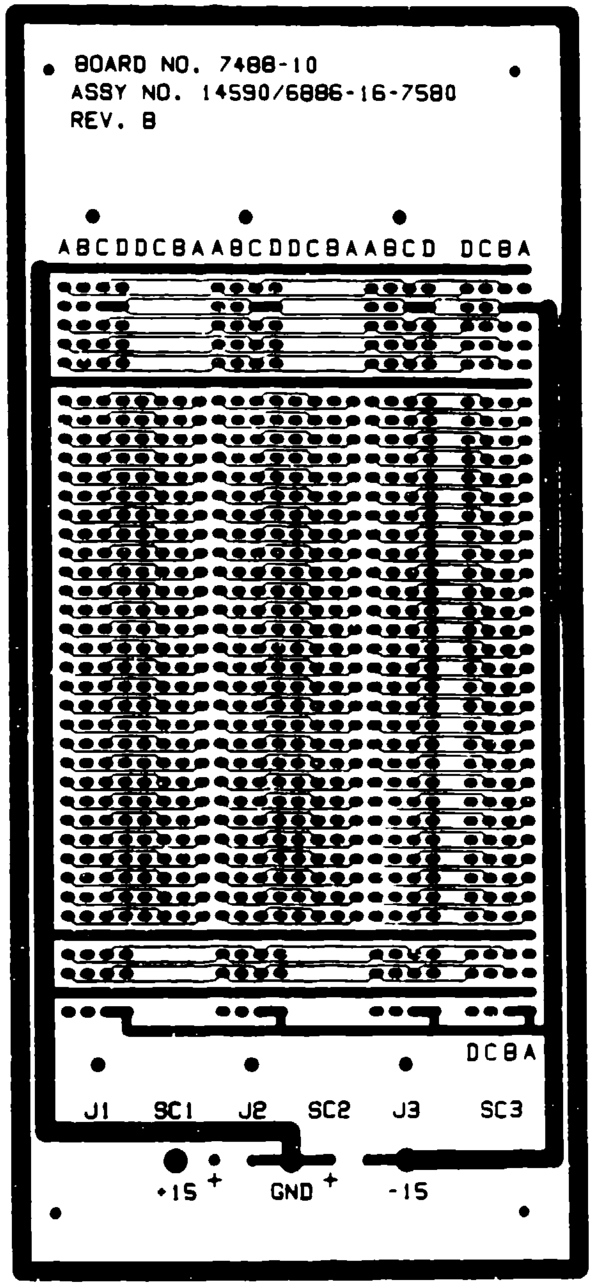



Figure 138. Analog Mother Board Layout

TABLE 40. CRIB MOTHER BOARD ASSIGNMENTS

| Row A | Function | Row B | Function | Row C | Function | Row D | Function |
|---|---|---|---|---|---|---|---|
| 1 | +15 SOURCE | 1 | +15 | 1 | -15 SOURCE | 1 | -15 SOURCE |
| 2 | GRD | 2 | GRD | 2 | GRD | 2 | GRD |
| 3 | FILTER CLK1 | 3 | FILTER CLK2 | 3 | FILTER CLK3 | 3 | FILTER CLK4 |
| 4 | ADMUXO | 4 | ADMUX1 | 4 | ADMUX2 | 4 | ADMUX3 |
| 5 | GRD | 5 | GRD | 5 | GRD | 5 | GRD |
| 6 | Exc 1 + | 6 | Sig 8 + | 6 | Sig 36 | 6 | Sig 44 |
| 7 | Sig 1 + | 7 | Exc 8 + | 7 | Sig 17 | 7 | Sig 45 |
| 8 | Exc 1 + | 8 | Sig 8 - | 8 | Sig 18 | 8 | Sig 46 |
| 9 | Sig 1 - | 9 | Exc 8 - | 9 | Sig 19 | 9 | Sig 47 |
| 10 | Exc 2 + | 10 | Sig 9 | 10 | Sig 20 | 10 | Sig 48 |
| 11 | Sig 2 + | 11 | Exc 9 + | 11 | Sig 21 | 11 | Sig 49 |
| 12 | Exc 2 - | 12 | Exc 9 - | 12 | Sig 22 | 12 | Sig 50 |
| 13 | Sig 2 - | 13 | Exc 10 + | 13 | Sig 23 | 13 | Sig 51 |
| 14 | Exc 3 - | 14 | Sig 10 | 14 | Sig 24 | 14 | Sig 52 |
| 15 | Sig 3 + | 15 | Exc 10 - | 15 | Sig 25 | 15 | Sig 53 |
| 16 | Exc 3 - | 16 | Sig 11 | 16 | Sig 26 | 16 | Sig 54 |
| 17 | Sig 3 - | 17 | Exc 11 + | 17 | Sig 27 | 17 | Sig 55 |
| 18 | Exc 4 + | 18 | Exc 11 - | 18 | Sig 28 | 18 | Sig 56 |
| 19 | Sig 4 + | 19 | Exc 12 + | 19 | Sig 29 | 19 | Sig 57 |
| 20 | Exc 4 - | 20 | Sig 12 | 20 | Sig 30 | 20 | Sig 58 |
| 21 | Sig 4 - | 21 | Exc 12 - | 21 | Sig 31 | 21 | Sig 59 |
| 22 | Exc 5 + | 22 | Sig 13 | 22 | Sig 32 | 22 | Sig 60 |
| 23 | Sig 5 + | 23 | Exc 13 | 23 | Sig 33 | 23 | Sig 61 |
| 24 | Exc 5 - | 24 | Exc 13 - | 24 | Sig 34 | 24 | Sig 62 |
| 25 | Sig 5 - | 25 | Exc 14 + | 25 | Sig 35 | 25 | Sig 63 |
| 26 | Exc 6 + | 26 | Sig 14 | 26 | spare | 26 | Sig 64 |
| 27 | Sig 6 + | 27 | Exc 14 + | 27 | spare | 27 | Sig 37 |
| 28 | Exc 6 - | 28 | Sig 15 | 28 | spare | 28 | Sig 38 |
| 29 | Sig 6 - | 29 | Exc 15 + | 29 | spare | 29 | Sig 39 |
| 30 | Exc 7 + | 30 | Exc 15 - | 30 | spare | 30 | Sig 40 |
| 31 | Sig 7 + | 31 | Exc 16 + | 31 | spare | 31 | Sig 41 |
| 32 | Exc 7 - | 32 | Sig 16 | 32 | spare | 32 | Sig 42 |
| 33 | Sig 7 - | 33 | Exc 16 - | 33 | spare | 33 | Exc 43 |
| 34 | GRD | 34 | GRD | 34 | GRD | 34 | GRD |
| 35 | DSO | 35 | +5 SOURCE | 35 | +5 SOURCE | 35 | spare |
| 36 | DS4 | 36 | DS3 | 36 | DS2 | 36 | DS1 |
| 37 | RCAL CONTROL | 37 | PWR CONTROL | 37 | DS6 | 37 | DS5 |
| 38 | +15 SOURCE | 38 | +15 | 38 | -15 SOURCE | 38 | -15 SOURCE |
| 39 | ANA1 | 39 | ANA2 | 39 | ANA3 | 39 | ANA4 |
| 40 | GRD | 40 | GRD | 40 | GRD | 40 | GRD |

## TABLE 41. AFIB MOTHER BOARD ASSIGNMENTS

| Row A | Function | Row B | Function | Row C | Function | Row D | Function |
|---|---|---|---|---|---|---|---|
| 1 | +15 SOURCE | 1 | +15 SOURCE | 1 | -15 SOURCE | 1 | -15 SOURCE |
| 2 | GRD | 2 | GRD | 2 | GRD | 2 | GRD |
| 3 | FILTER CLK1 | 3 | FILTER CLK2 | 3 | FILTER CLK3 | 3 | FILTER CLK4 |
| 4 | ADMUXO | 4 | ADMUX1 | 4 | ADMUX2 | 4 | ADMUX3 |
| 5 | GRD | 5 | GRD | 5 | GRD | 5 | GRD |
| 6 | Exc 1 + | 6 | Sig 8 + | 6 | Exc 17 + | 6 | Sig 24 + |
| 7 | Sig 1 + | 7 | Exc 8 + | 7 | Sig 17 + | 7 | Exc 24 + |
| 8 | Exc 1 - | 8 | Sig 8 - | 8 | Exc 17 - | 8 | Sig 24 - |
| 9 | Sig 1 - | 9 | Exc 8 - | 9 | Sig 17 - | 9 | Exc 24 - |
| 10 | Exc 2 + | 10 | Sig 9 | 10 | Exc 18 + | 10 | Sig 25 |
| 11 | Sig 2 + | 11 | Exc 9 + | 11 | Sig 18 + | 11 | Exc 25 + |
| 12 | Exc 2 - | 12 | Exc 9 - | 12 | Exc 18 - | 12 | Exc 25 - |
| 13 | Sig 2 - | 13 | Exc 10 + | 13 | Sig 18 - | 13 | Exc 26 + |
| 14 | Exc 3 + | 14 | Sig 10 | 14 | Exc 19+ | 14 | Sig 26 |
| 15 | Sig 3 + | 15 | Exc 10 - | 15 | Sig 19 + | 15 | Exc 26 - |
| 16 | Exc 3 - | 16 | Sig 11 | 16 | Exc 19 - | 16 | Sig 27 |
| 17 | Sig 3 - | 17 | Exc 11 + | 17 | Sig 19 - | 17 | Exc 27 + |
| 18 | Exc 4 + | 18 | Exc 11 - | 18 | Exc 20 + | 18 | Exc 27 - |
| 19 | Sig 4 + | 19 | Exc 12 + | 19 | Sig 20 + | 19 | Exc 28 + |
| 20 | Exc 4 - | 20 | Sig 12 | 20 | Exc 20 - | 20 | Sig 28 |
| 21 | Sig 4 - | 21 | Exc 12 | 21 | Sig 20 - | 21 | Exc 28 - |
| 22 | Exc 5 + | 22 | Sig 13 | 22 | Exc 21 + | 22 | Sig 29 |
| 23 | Sig 5+ | 23 | Exc 13 - | 23 | Sig 21 + | 23 | Exc 29 + |
| 24 | Exc 5 - | 24 | Exc 13 - | 24 | Exc 21 - | 24 | Exc 29 - |
| 25 | Sig 5 - | 25 | Exc 14 + | 25 | Sig 21 - | 25 | Exc 30 + |
| 26 | Exc 6 + | 26 | Sig 14 + | 26 | Sig 22 + | 26 | Sig 30 |
| 27 | Sig 6 + | 27 | Exc 14 - | 27 | Sig 22 + | 27 | Exc 30 - |
| 28 | Exc 6 - | 28 | Sig 15 | 28 | Exc 22 - | 28 | Sig 31 |
| 29 | Sig 6 - | 29 | Exc 15 + | 29 | Sig 22 - | 29 | Exc 31 + |
| 30 | Exc 7 + | 30 | Exc 15 - | 30 | Exc 23 + | 30 | Exc 31 - |
| 31 | Sig 7 + | 31 | Exc 16 + | 31 | Sig 23 - | 31 | Exc 32 + |
| 32 | Exc 7 - | 32 | Sig 16 | 32 | Exc 23 - | 32 | Sig 32 |
| 33 | Sig 7 - | 33 | Exc 16 - | 33 | Sig 23 - | 33 | Exc 32 - |
| 34 | GRD | 34 | GRD | 34 | GRD | 34 | GRD |
| 35 | DSO | 35 | +5 SOURCE | 35 | +5 SOURCE | 35 | spare |
| 36 | DS4 | 36 | DS3 | 36 | DS2 | 36 | DS1 |
| 37 | RCAL CONTROL | 37 | PWR CONTROL | 37 | DS6 | 37 | DS5 |
| 38 | +15 SOURCE | 38 | +15 SOURCE | 38 | -15 SOURCE | 38 | -15 SOURCE |
| 39 | ANA1 | 39 | ANA2 | 39 | ANA3 | 39 | ANA4 |
| 40 | GRD | 40 | GRD | 40 | GRD | 40 | GRD |

### 3.2.3. Manikin Sensors

This section describes the sensors that have been selected for use within the ADAM. These sensors were selected for their ability to measure the necessary physical phenomena and fit into the manikin with a minimal impact on manikin biofidelity. Table 42 outlines the manikin measurement and the sensor associated with the measurement.

TABLE 42. ADAM MEASUREMENTS

| No. | Type of Measurement | Sensor Type |
|---|---|---|
| 1 | Left Hip Abduction/Adduction Position | Potentiometer |
| 2 | Right Hip Abduction/Adduction Position | Potentiometer |
| 3 | Left Hip Flexion Position | Potentiometer |
| 4 | Right Hip Flexion Position | Potentiometer |
| 5 | Left Hip Medial/Lateral Position | Potentiometer |
| 6 | Right Hip Medial/Lateral Position | Potentiometer |
| 7 | Left Knee Flexion Position | Potentiometer |
| 8 | Right Knee Flexion Position | Potentiometer |
| 9 | Left Knee Medial/Lateral Position | Potentiometer |
| 10 | Right Knee Medial/Lateral Position | Potentiometer |
| 11 | Left Shoulder Arm-Joint Abduction/Adduction Position | Potentiometer |
| 12 | Right Shoulder Arm-Joint Abduction/Adduction Position | Potentiometer |
| 15 | Left Shoulder Flexion/Extension Position | Potentiometer |
| 16 | Right Shoulder Flexion/Extension Position | Potentiometer |
| 17 | Left Shoulder Medial/Lateral Position | Potentiometer |
| 18 | Right Shoulder Medial/Lateral Position | Potentiometer |
| 19 | Left Arm Raising/Lowering Position | Potentiometer |
| 20 | Right Arm Raising/Lowering Position | Potentiometer |
| 21 | Left Elbow Flexion Position | Potentiometer |
| 22 | Right Elbow Flexion Position | Potentiometer |
| 23 | Left Forearm Supination/Pronation Position | Potentiometer |
| 24 | Right Forearm Supination/Pronation Position | Potentiometer |
| 25-26 | Left Lower Leg Torque (Positive and Negative) | Load Cell |
| 27-28 | Right Lower Leg Torque (Positive and Negative) | Load Cell |
| 29-34 | Neck Forces and Moments (6 axis) | Denton Force Balance |
| 35-40 | Lumbar Forces and Moments (6 axis) | Denton Force Balance |
| 41-43 | Head Acceleration (triaxial) | Accelerometer |
| 47-49 | Chest Acceleration (triaxial) | Accelerometer |
| 50-52 | Pelvis Acceleration (triaxial) | Accelerometer |
| 53-54 (GFE) | Parachute Loads, Right and Left Risers | Loads Cells |
| 55 | Temperature Measurement | IC Sensor |
| 56-57 | Lumbar Position | Potentiometer |

TABLE 42. ADAM MEASUREMENTS (continued)

| No. | Type of Measurement | Sensor Type |
|---|---|---|
| 58 | Sternoclavicular Elevation/Depression Position | Potentiometer |
| 60-61 | Sternoclavicular Pronation/Retraction Position | Potentiometer |
| 62-63 | Hand Breakaway Signal | Normally Closed State Sensor |

3.2.3.1. Manikin Sensor Usage

Several different sensor types are being used within the ADAM for phenomena measurement. The breakdown of the sensors in use is:

- Position Potentiometers (28 per manikin)
- Piezoresistive Accelerometers (9 per manikin)
- Six Component Force Balances (2 per manikin)
- Load Cells (4 per manikin)
- Miscellaneous (3 per manikin)

The following is a description of the sensors in use in the manikin.

3.2.3.2. Position Potentiometer

The position potentiometers are trimmer potentiometers manufactured by Preh Electronic Industries. These pots are a cermet design with a temperature coefficient of 100 ppm/°C and are 0.4 x 0.46 x 0.25 inches in size. The pots are mounted on a printed circuit board to facilitate mechanical mounting of the pot and allow various mounting positions. The pot is actuated by a 2 mm diameter screwdriver type of blade that is mounted on the moving portion of the joint (Figure 139). Vibration tests were conducted on these potentiometers to ensure the integrity of the data. No problems with noise or discontinuities were noted.

A problem was discovered with the mechanical interface between the actuator blade and the potentiometer. Mechanical hysteresis between the blade and the potentiometer allow a variation in the output of the potentiometer as it is rotated first in one direction and then in the opposite direction. The hysteresis was tracked to the fit between the actuator blade and the cross keyway of the

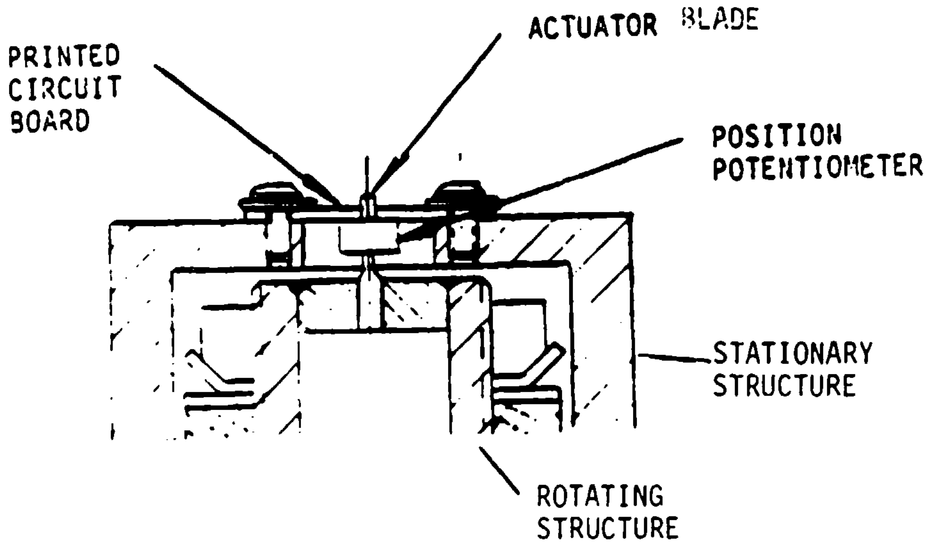



Figure 139. Sample of a Position Potentiometer Mounting

potentiometer. A study was conducted to determine the best solution to the problem of eliminating the free play between the potentiometers and the blades. The solution resulting from this study was to use hot melt glue between the potentiometers and actuator blade. The glue is melted in a glue gun and has a set time of about 30 seconds in this application. The glue is placed in the keyway of the potentiometers, and the potentiometer is placed in position over the actuator blade. The hot glue surrounds the blade and cools in a position to eliminate the free play between the potentiometer and blade.

The hot melt glue is used as a gap filler in this application, and it has the added benefit that the adhesive property of the glue is weak in shear. This benefit was important because the potentiometer could be removed without being damaged.

3.2.3.3. Accelerometers

The accelerometers selected for use in the manikin are Entran EGA-125 miniature piezoresistive accelerometers. All accelerometers are a single axis style with a 100 g acceleration range with 0.7 critical damping and 100 percent over range. The single axis style was selected to minimize the replacement cost if a unit used in a triaxial configuration failed.

### 3.2.3.4. Force Balances

Two Force Balances have been provided with the ADAM. Both balances are manufactured by the Robert A. Denton Company and resolve the forces acting on the lumbar spine and neck into three orthogonal forces and their corresponding first moment. One force balance, Denton Model 1914, is used in the lumbar spine and the other balance, Denton Model 1716, is used at the junction of the head and neck.

### 3.2.3.5. Load Cells

The load cells are used to measure the left and right lower leg torque. Two load cells are used per leg for the torque measurement. One load cell is used to measure the torque in each direction. The load cell selected is specially built by Hitec Corporation for SRL. It is a 3000-pound ring load cell with 25 percent over range that is 0.5-inch in diameter and approximately 0.375-inch wide.

### 3.2.3.6. Miscellaneous Sensors

The miscellaneous sensors are the temperature sensor and the hand breakaway mechanism. The temperature transducer is an integrated circuit type LM235A from National Semiconductor. The circuit will give a direct output of the ambient temperature in degrees Kelvin. The hand breakaway sensor is a simple logic state sensor that will be low while the manikin hand is in the proper position, but when the hand breaks away, the sensor output will go high to indicate the hand is free. One sensor will be used for each hand.

## 3.3. DIGITAL SUBSYSTEM

The digital subsystem of the ADAM instrumentation contains the circuitry to control the data acquisition process, communicate with the user, and store the test data. The main controller is the central processing unit (CPU) that executes the software commands used to control the instrumentation. The data acquisition process begins with the analog-to-digital (A/D) board, which converts the analog signals to digital data. This conversion process is controlled by the CPU which is located on the processor board. The processor board contains the CPU, central support circuitry, filter clock generator, and serial communications interface. The data memory and system firmware are located on the memory board. The final board to be discussed is the digital I/O board. This board contains the high speed parallel port, telemetry interface, and CPU status/control port.

These four boards mate to the digital mother board where the digital system control and bus signals are routed to every board.

### 3.3.1. Analog-to-Digital Conversion Board

#### 3.3.1.1. Functional Description

Figure 140 illustrates the block diagram of the ADC system. The system uses four A/D converters on a 32-bit data bus to perform the actual digitization of the data. Four A/Ds are used to increase the system conversion throughput by four. The A/D converters used have 12-bit resolution with 11-bit accuracy, and by placing each A/D's output on one quarter of the 32-bit data bus, four conversions can be done in the time it would require a single A/D to perform a single conversion. This technique quadruples the conversion throughput. The A/D converters selected for use in the ADAM instrumentation are Burr Brown ADC803 A/D converters. These are 12-bit A/Ds with 11-bit accuracy in which only the eight most significant bits are recorded by the system. The digital data are latched in the data latches so that it may be read by the CPU.

The outputs from the analog multiplexers are fed into high speed buffers that drive the inputs to the A/Ds. These buffers are used to match the impedance characteristics of the A/Ds to the analog multiplexers.

The ADC control logic controls the start conversion command of the A/Ds and the load command of the programmable counters used to generate the analog multiplexer address and control lines. The programmable counters are designed such that they can be sequenced automatically by the ADC control logic, or they can be loaded with an individual multiplexer address. This allows a large amount of flexibility in the design of the software data acquisition routine. The ADC control logic also ensures that the CPU will not try to read the data latches while any of the A/Ds are making a conversion. This protects the integrity of the data read by the microprocessor. A safety backup allows the data to be read if an A/D fails. A detailed description of these function blocks will follow.

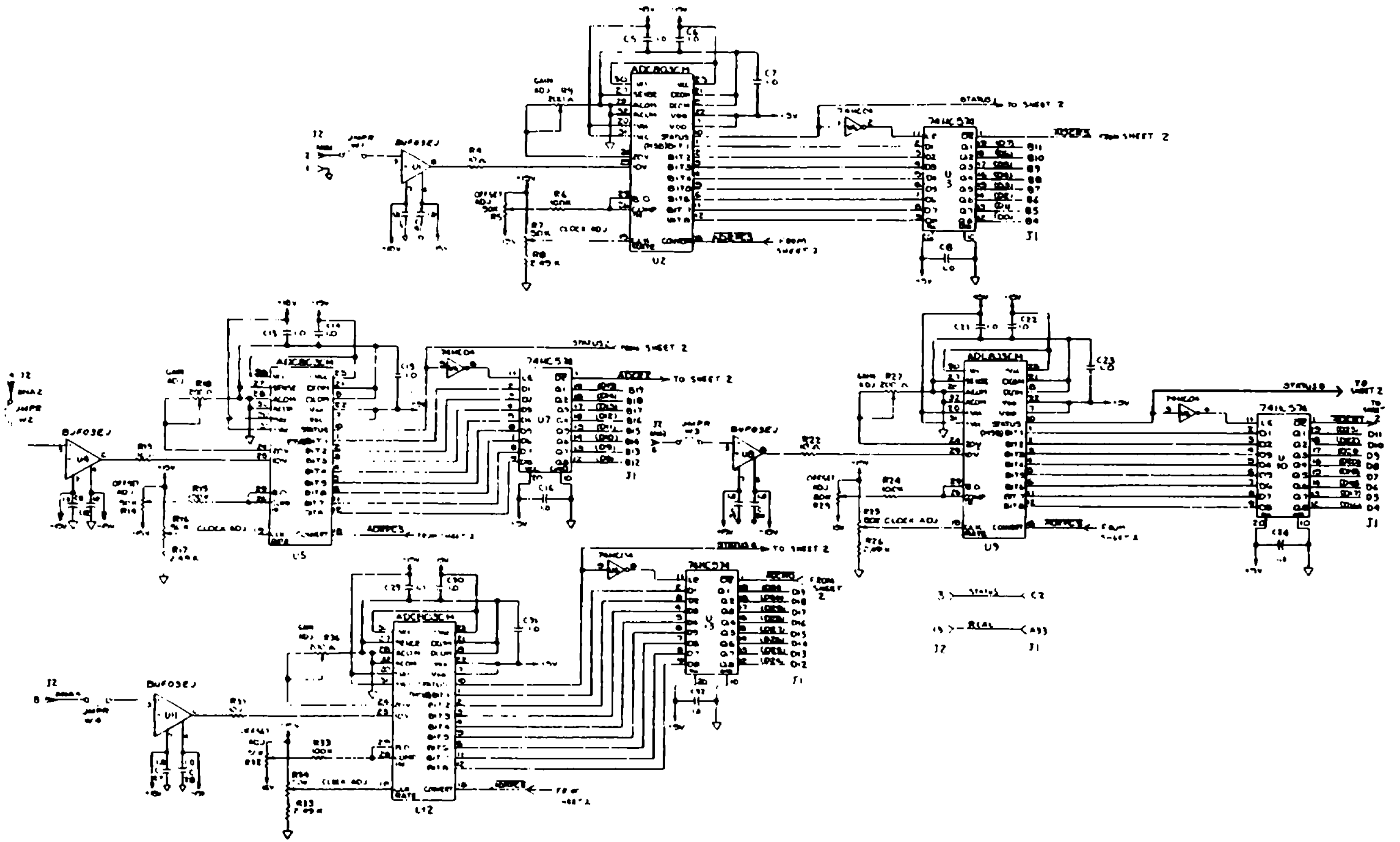


Figure 140. Analog-to-Digital Conversion Board Schematic

3.3.1.2. High Speed Buffer

The high speed buffers chosen for the input to the A/D were Precision Monolithics, Inc., BUF-03 unity gain buffers. These buffers are used to match the impedance of the analog signal from the multiplexers to the input impedance of the A/D converter. The signal from the multiplexers requires a high input impedance to prevent excessive loading of the signal, and these buffers have an input impedance of 400,000 megohm. The output impedance of the buffer must be very low to prevent the current transients that typically occur with the A/Ds from affecting the A/D input voltage. For this same reason, the buffer is required to have a settling time of less than 100 nanoseconds. The BUFO3 has a typical output impedance of 2 ohms and a settling time of 100 nanoseconds. In order to maintain the die temperature below 105°C, heat sinks are used on the buffers to maintain temperatures in the vicinity of 75°C to maintain operation within specifications. If the temperature of the die does exceed 105°C, the settling time and output offset voltage increases by 10 percent and the output offset voltage increases by a similar quantity.

3.3.1.3. A/D Converter

Figure 140 shows the schematic of the four A/Ds, BUF-O3 high speed buffers, and the data buffers. There are four identical circuits shown in Figure 140. These four A/D circuits are used to digitize the analog data from the analog signal conditioning circuitry. Four circuits were used because the four A/Ds will digitize four times as much data in the same time that one A/D will digitize one set of data. This speeds the data conversion process by four fold. The A/D in use is the Burr Brown ADC803. This is a very high speed successive approximation 12-bit A/D. This A/D has a 12-bit resolution with an accuracy of 11 bits. Because an 8-bit resolution and accuracy is all that is necessary in a system where 1 percent nonlinearity is required, only the eight most significant data bits of the A/D are latched and used in the system. Each of the four A/D data latches are assigned as different 8-bit segment of the 32-bit data bus so that the data from the four circuits may be read by the CPU at the same time. This also serves to increase the conversion speed of the system.

The top quarter of the schematic in Figure 140 is a circuit that is typical of the four circuits used on this board. U1 is the BUF-03 high speed buffer, and it is tied to the analog signal line ANA1. Each of the other three circuits are tied to the remaining analog signal lines, ANA2-ANA4. The output of U1 (pin 6) is connected through the 10 ohm series resistor, R4, to the input of the A/D.

On the ADC803, U2, pin 29 is connected to pin 26, and pin 24 is tied to ground through the gain adjust potentiometers, R9, in order to program the input for a 5 volt input range with a binary-offset-binary data output. Binary-offset-binary output uses a logic "1" true. The ADC803 has an internal clock that is used to control the conversion process. The speed of this conversion process is determined by this clock frequency. The potentiometer, R7, and resistor, R8, are used to adjust this clock frequency to 13 Mhz. This generates an A/D conversion time that is under 1 microsecond.

The data from the A/D is latched into the data latch, U3, when the STATUS line (U2, pin 10), inverted by U6, makes a low-to-high transition at pin 11 of U3. The STATUS line makes a low-to-high transition when the A/D conversion cycle is started, and a high-to-low transition when the twelfth data bit has been determined. The logic required to latch the data into U3 is the opposite of the STATUS line; hence, the inverter, U6, is used to generate the correct control signal to U3.

The last line of the A/D to be discussed is the convert pin, U2-pin 18. This input is used to start the conversion process of the A/D. When the convert pin is brought to a logic low level for a minimum of 50 nanoseconds, the conversion process starts. This conversion command comes from the ADC control logic. The remaining connections to the A/D are multiple power and ground pins.

The output pins of U3 (pins 12 to 19) are tri-state devices since they are connected to the system data bus. The data in the latch is active on the data bus when the control signal $\overline{\text{ADCR3}}$ ($\overline{\text{ADCR0}}$-$\overline{\text{ADCR2}}$ for the other three circuits) is active. The signals $\overline{\text{ADCR0}}$-$\overline{\text{ADCR3}}$ also are generated by the ADC control logic. These signals are active when the microprocessor reads the data from the A/D port.

### 3.3.1.4. ADC Control Logic

Figure 141 represents the circuitry for the ADC control logic and the programmable counters used to generate the multiplexer address signals. The discussion will first deal with the ADC control logic and then the programmable counters.

The ADC control logic consists of the three Programmable Array Logic (PAL) ICs (U14, U15, and U19). These devices are programmable sums of products logic ICs that implement boolean equations to generate the required control signals from the correct combination of inputs. The PAL, U19, is the partial address chip select generator. The address of the A/D port is 800000 (hex), and U19 generates the control signal PADCS (pin 23) for addresses in the range of 800000

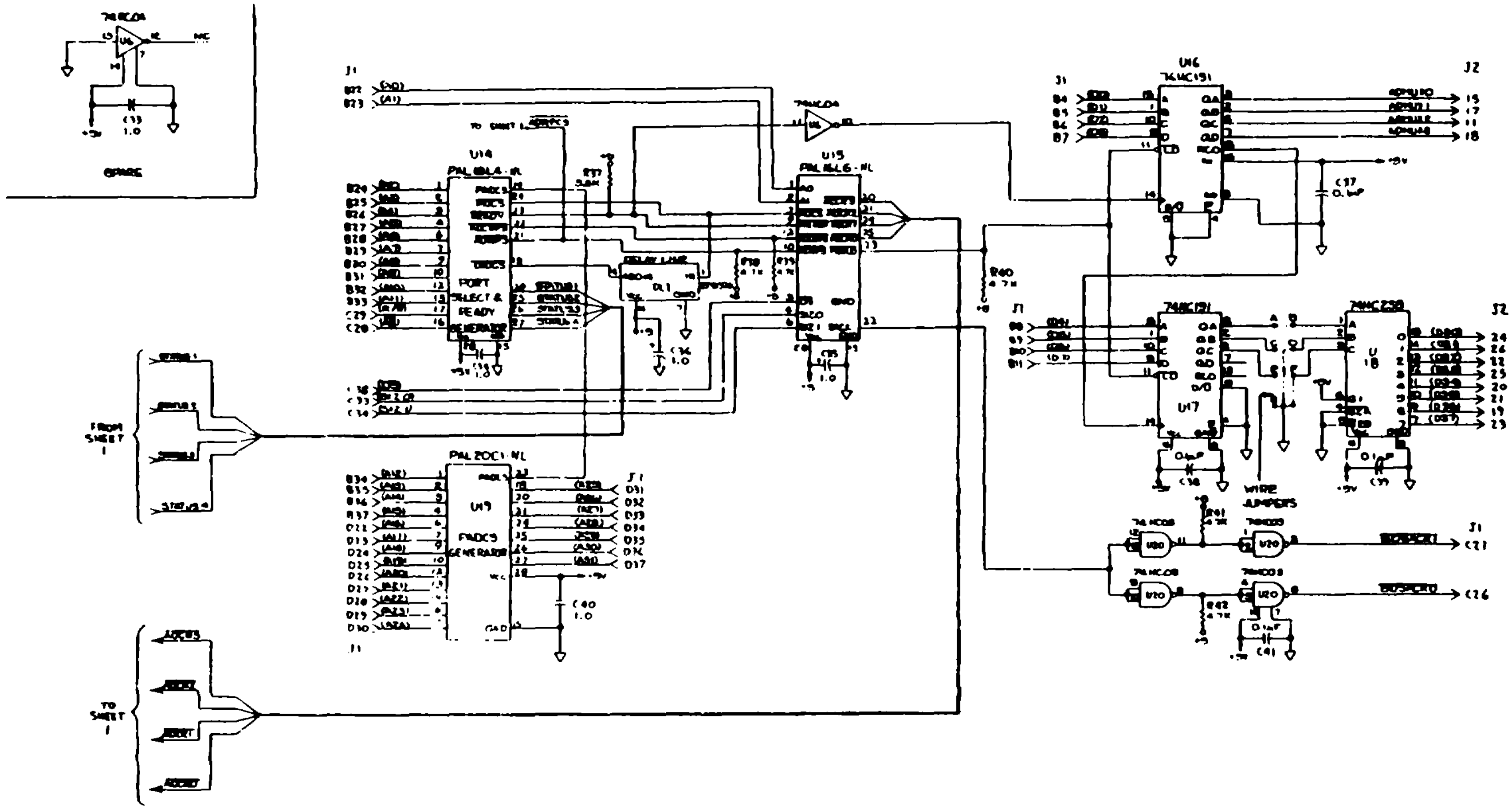


Figure 141. Analog-to-Digital Conversion Board Schematic

where the PAL generators the read and write chip selects for the A/Ds. The chip select $\overline{\text{ADRPCS}}$ is connected to the convert pin of each A/D (Figure 141) and is active whenever the processor reads the A/D port. An A/D conversion begins on the rising edge of the signal. $\overline{\text{ADRPCS}}$ and $\overline{\text{ADRPS}}$ are the same signal, but $\overline{\text{ADRPCS}}$ is defined as the signal that interfaces with the A/Ds to start a data conversion cycle. ADRPS is also connected as an input to U15, a general control signal generator.

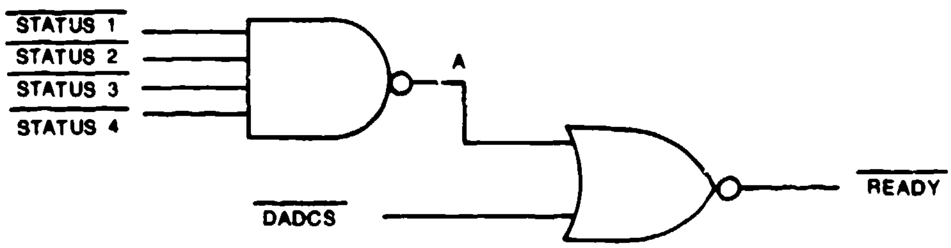
The output $\overline{\text{ADCS}}$ is active whenever the processor is reading or writing to the A/D port. $\overline{\text{ADCS}}$ is generated as an input for U15 and an input to U14 delayed by 480 nanoseconds through the delay line DL1. This delayed $\overline{\text{ADCS}}$ (called $\overline{\text{DADCS}}$) is used in the ready generator. The $\overline{\text{READY}}$ control line is generated from the four status lines from the A/Ds and the $\overline{\text{DADCS}}$ line. The ready line is used to prevent the processor from reading the A/D's data latch before it is through with its conversion. READY is generated by "ANDing" the four status lines together so that READY is low when all four STATUS lines are low. The $\overline{\text{DADCS}}$ line is "ORed" with the "ANDed" and STATUS lines so that in cases where an A/D or several A/Ds fail, the A/D data may still be read from the A/D port.

Figure 142 shows the graphic representation of how the $\overline{\text{READY}}$ line is implemented and a table that shows how the $\overline{\text{READY}}$ line is generated. The $\overline{\text{READY}}$ line is connected as an input to U15, and the $\overline{\text{READY}}$ line is also inverted through U6 and connected as a clock input to U16, a programmable multiplexer address counter.

The last PAL in the ADC control logic section is U15. This PAL is used for many purposes. The $\overline{\text{ADRPS}}$ input along with the SIZO, SIZI, and $\overline{\text{DS}}$ (data strobe) inputs from the microprocessor control bus are used to generate the $\overline{\text{ADCR0}}$-$\overline{\text{ADCR3}}$ A/D converter READ lines. These lines are the lines that enable the outputs of the data latches when the microprocessor is reading the A/D data.

The $\overline{\text{ADCWPS}}$, A/D converter write port select, is used with $\overline{\text{DS}}$ to generate the $\overline{\text{MUXLD}}$ (multiplexer load) signal that allows the multiplexer address to be loaded into the programmable counters when the microprocessor executes a write to the A/D port.

The ready line is currently unused in U15, and the last output of U15, SACK, is the inversion of the input $\overline{\text{ADCS}}$. This output is inverted twice through U20, using two pair of NAND gates to generate the control signals $\overline{\text{BDSACK0}}$ and $\overline{\text{BDSACK1}}$, buffered data transfer and size




| STATUS | | | | A | DADCS | READY |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | | | |
| 1 | d | d | d | 0 | 0 | 1 |
| d | 1 | d | d | 0 | 0 | 1 |
| d | d | 1 | d | 0 | 0 | 1 |
| d | d | d | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | d | 0 |
| 1 | d | d | d | 0 | 1 | 0 |
| d | 1 | d | d | 0 | 1 | 0 |
| d | d | 1 | d | 0 | 1 | 0 |
| d | d | d | 1 | 0 | 1 | 0 |

Figure 142. Ready Generator Logic

ackno wledge 0 and 1, which are used to tell the processor that the A/D port is arranged as a 32-bit (long word) wide port. The operation of these signals is described in detail in the section describing the board.

### 3.3.1.5. Programmable Counters

The programmable counters section of Figure 141 is used to generate the multiplexer address lines ADMUX0-ADMUX3 and the data select lines DS0-DS6 that are used on the CRIB and AFIB. U16 and U17 are 74HC191s, 4-bit binary up/down counters with an asynchronous load capability, that generate the multiplexer address. This address has the range of 00 (hex) to 7F. This first four bits are loaded into the counter, U16, whose ouptuts generate the ADMUX0-ADMUX3 lines directly.

The data select lines DS0-DS6 are generated by feeding the outputs of U17 into the inputs of U18, a 74HC238 three to eight line decoder/demultiplexer, through the jumper pads A-G. U18 uses the binary input of pins 1 to 3 to activate a single output. For the ADAM design, A is jumpered to B, ADMUX3 lines to select a multiplexer (data select line) and a multiplexer address (ADMUX0-ADMUX3 lines).

There are two different techniques that may be used to generate the multiplexer address required. The first is an automatic sequence that is incremented by the ADC control logic, and the second is an individual write of the multiplexer address to the ADC port. The automatic sequencing occurs at the end of each data conversion cycle. The $\overline{\text{READY}}$ line clocks U16, which causes the multiplexer address to increment by one. This automatic sequencing occurs after every data conversion cycle regardless of the address selection method used. Individual multiplexer addresses may be selected by writing the multiplexer address to the ADC port. This will cause the $\overline{\text{MUXLD}}$ line to go active, and the data that appears on D0-D7 is written to U16 and U17. The ADC control is designed to allow a long word write only to this port.

3.3.1.6. Other A/D Circuitry

The remaining circuitry on the A/D board is shown in Figure 143. The power supply for the A/D board is shown on the left side of Figure 143. It is the same functional circuit as those discussed previously. These regulators are not set up to be shut down electronically as was the case on the CRIB and AFIB. Two regulators are required for the +5 volt supply to prevent the maximum current rating of the regulators and pass transistor from being exceeded. Table 43 below outlines the current requirements of this board.

TABLE 43. A/D BOARD CURRENT REQUIREMENTS

| Power Supply | Current Consumption |
|---|---|
| +5 | 1 A |
| -15 | 200 mA |
| +15 | 200 mA |

The remaining circuit is the circuit used to measure the internal viscera temperature. The sensor, E1, is an LM235A precision temperatures sensor, and was located on the A/D board since this board produces the most heat in the system. The sensor circuit is designed so that the temperature output varies 10 mV/°K. R50 has been provided to trim the sensor output to a precise value. The

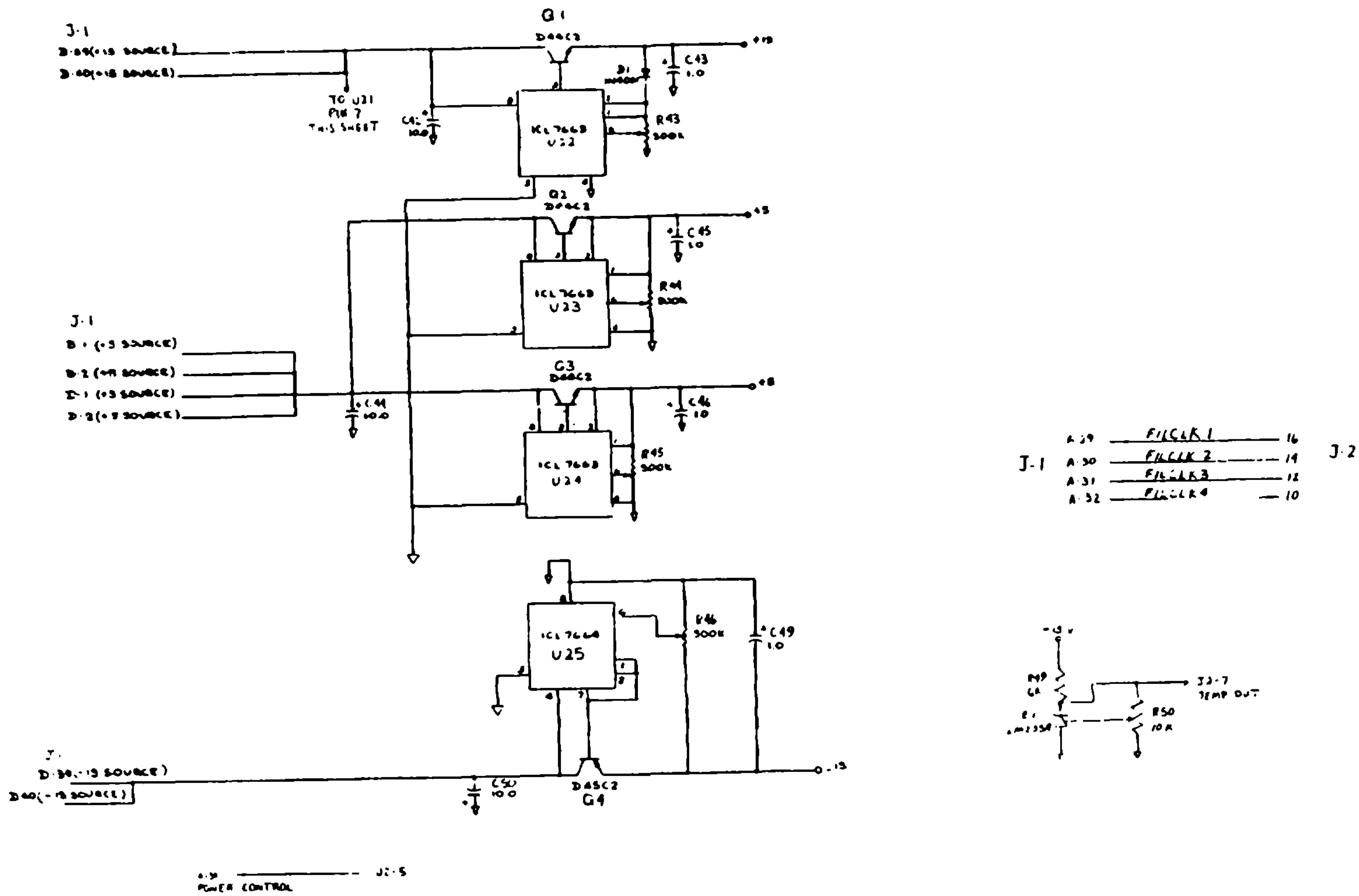



Figure 143. Analog-to-Digital Conversion Board Schematic

sensor is completely linear and requires only a single point calibration throughout its temperature range of -40°C to 125°C with a 200°C overrange.

Figure 144 shows the A/D board assembly drawing. The dimensions of the board are 6.35 inches x 4.43 inches. The connector, J1, is the connector that mates with the digital mother board. It passes all the necessary control signals, data lines, address lines, and functional signals the A/D circuitry. The connector, J2, is a 26 pin ribbon cable connector that is used to pass all of the ADC I/O Lines from the A/D board to the analog mother board. Table 44 lists the connector pin and the signal that is carried on that pin as well as the origin of the signal. This is the only link between the digital subsystem and the analog subsystem, and flow of information is controlled by the CPU on the processor board.

TABLE 44. ADC BOARD--ANALOG MOTHER BOARD CONNECTOR

| Signal Name | J2 Pin | Signal Source | Signal Destination |
|---|---|---|---|
| GROUND | 1 | J1-A3, B3, C3, D3, A20, B20, C30, D20, A21, B21, C21, D21, B38, D38 | Analog Mother Board Interconnect Cable |
| ANA1 | 2 | Analog Mother Board | R1 at U1 |
| STATUS | 3 | J1-C2 | Analog Mother Board |
| ANA2 | 4 | Analog Mother Board | R10 at U4 |
| PWR CONTROL5 | J1-A34 | Analog Mother Board | |
| ANA3 | 6 | Analog Mother Board | R19 at U8 |
| TEMP(erature) | 7 | E1 | Analog Mother Board |
| ANA4 | 8 | Analog Mother Board | R28 at U11 |
| FILTER CLK4 | 10 | J1-A32 | Analog Mother Board |
| ADMUX2 | 11 | U16-6 | Analog Mother Board |
| FILTER CLK3 | 12 | J1-A31 | Analog Mother Board |
| FILTER CLK2 | 14 | J1-A30 | Analog Mother Board |
| ADMUX0 | 15 | U16-3 | Analog Mother Board |
| FILTER CLK1 | 16 | J-A29 | Analog Mother Board |
| ADMUX1 | 17 | U16-2 | Analog Mother Board |
| ADMUX3 | 18 | U6-7 | Analog Mother Board |
| DS6 | 19 | U18-9 | Analog Mother Board |
| DS4 | 20 | U18-11 | Analog Mother Board |
| DS5 | 21 | U18-10 | Analog Mother Board |
| DS2 | 22 | U18-13 | Analog Mother Board |
| DS7 | 23 | U18-7 | No Connection |
| DS0 | 24 | U18-15 | Analog Mother Board |
| DS3 | 25 | U18-12 | Analog Mother Board |
| DS1 | 26 | U18-14 | Analog Mother Board |
| Spare | 9 | | |
| Spare | 13 | | |

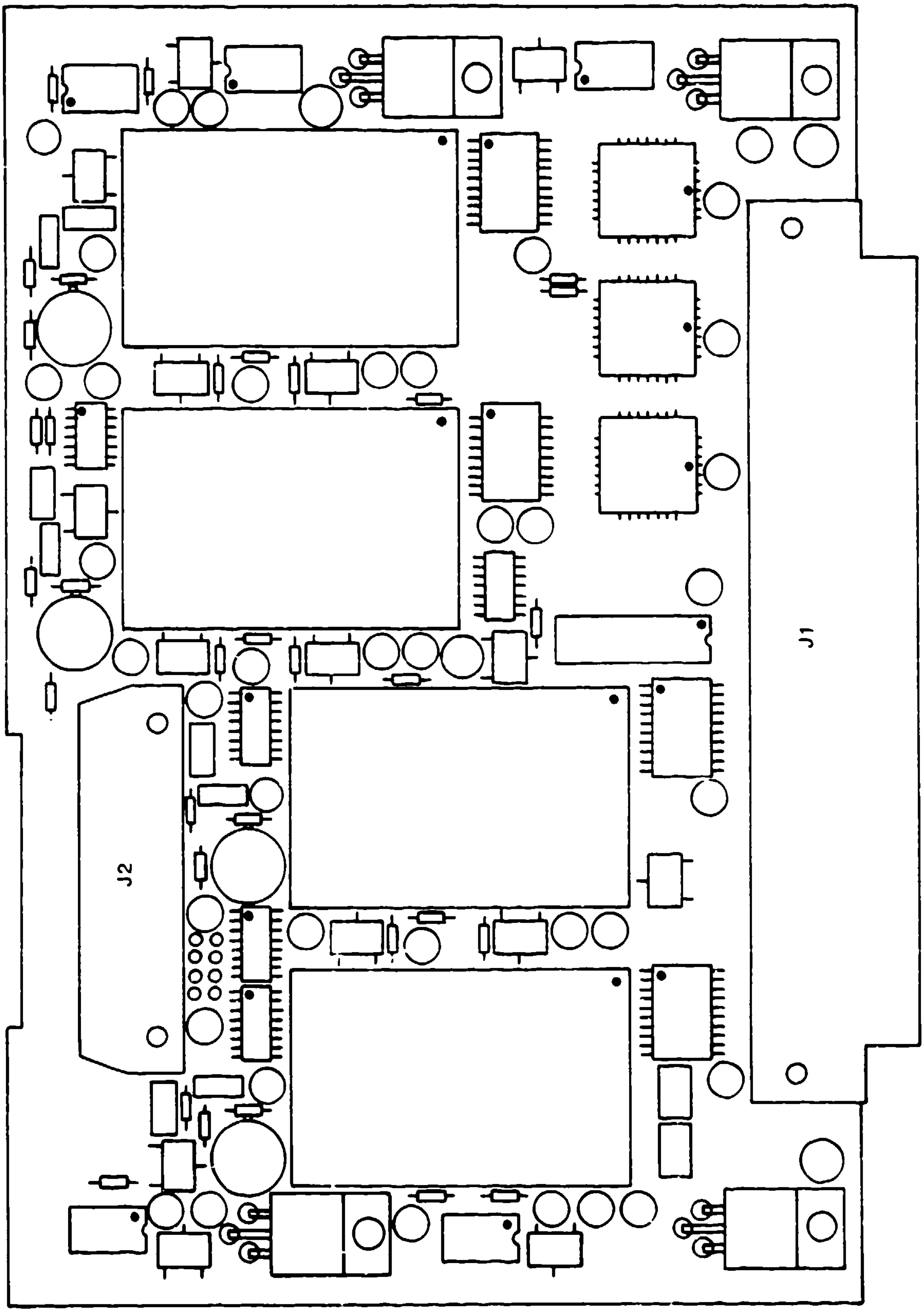



Figure 144. Analog-to-Digital Board Assembly

### 3.3.2.1. Functional Description

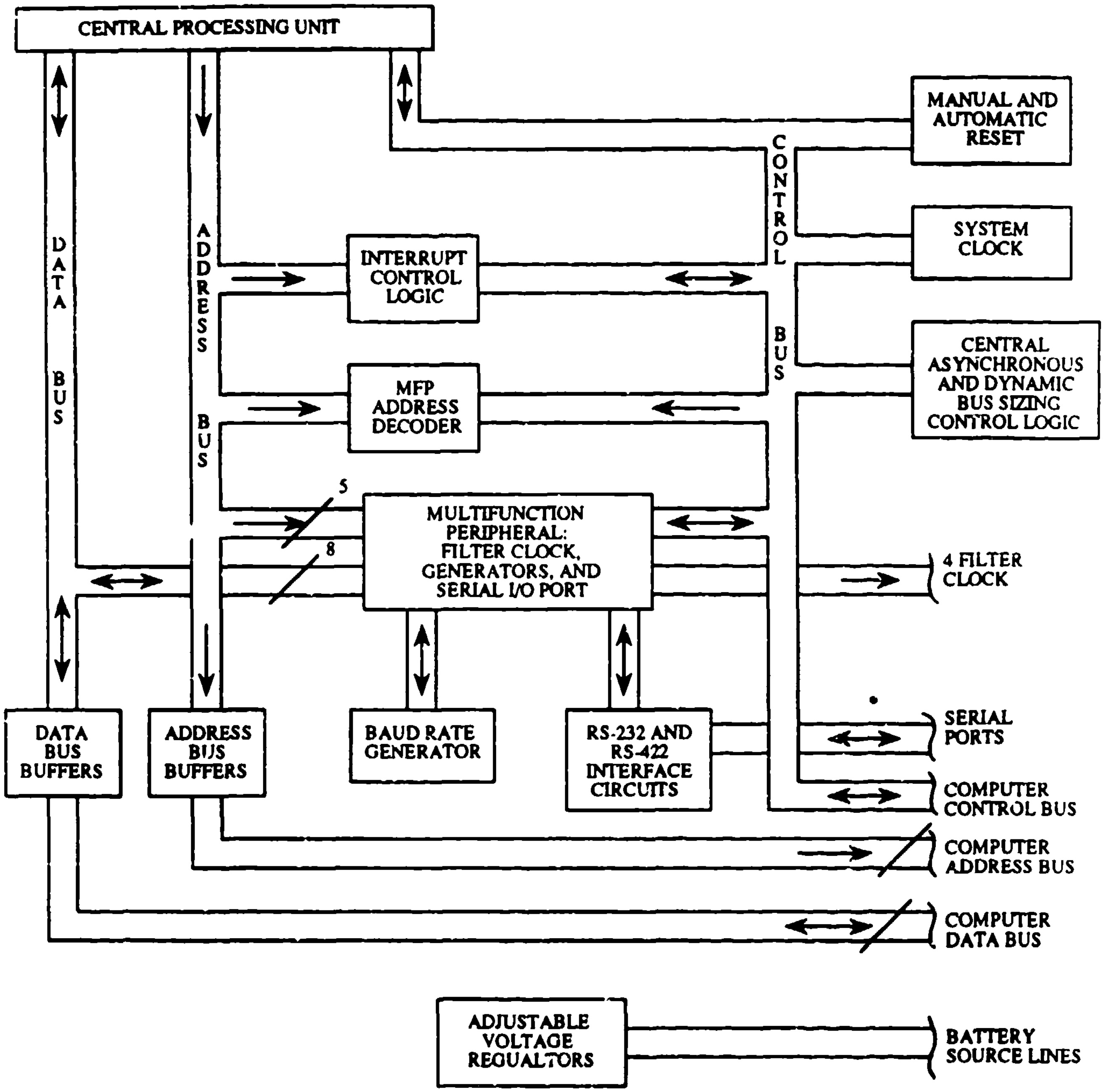
The processor board contains the CPU, address and data bus drivers, the main system clock, power-on and manual system reset circuitry, central interrupt control logic, central DSACK control logic, and an onboard +5V regulator. This board also contains RS-232 and RS-422 serial communications and the filter clock generator. The processor board block diagram is presented in Figure 145.

The CPU is the central core of the system and controls all system activities, either directly or indirectly. The CPU is the Motorola MC68020, a 32-bit virtual memory microprocessor. It is implemented using HCMOS technology, providing maximum computing power for the energy consumed. Its internal registers, data paths, and its nonmultiplexed asynchronous external data and address paths are 32 bits in width. It has a rich basic instruction set, 18 versatile addressing modes, object code compatibility with MC68000 family processors, and an architecture that easily supports high level languages. This CPU has a 64 long word on-chip instruction cache and a parallel internal structure that allows multiple instructions to be executed concurrently. The processor supports a dynamic sizing mechanism that allows the CPU to transfer information to or from external devices while automatically determining device port size on a cycle-by-cycle basis, which eliminates all data alignment restrictions. It has 16 32-bit general purpose data and address registers, a 32-bit program counter for a 4 gigabyte direct addressing range, memory mapped I/O, operations on seven data types, and many special internal registers to enhance program execution.

The processor board contains a multifunction peripheral IC that is used to generate an RS-232 and RS-422 standard serial output and the circuitry used to generate four separate filter clocks for the signal conditioning circuitry. Each of these systems will be described in detail.

### 3.3.2.2. Central Processing Unit and Support Circuitry

The microprocessor that forms the heart of the ADAM system control functions is the Motorola 68020. The schematic for the CPU and its support circuitry is shown in Figure 146. The circuitry contained in Figure 146 includes the CPU, address and data bus buffers, interrupt control logic, main system clock reset circuitry, and central data transfer and size acknowledge (DSACK) circuitry.




• VIA DEDICATED CONNECTOR; NOT ON DIGITAL MOTHER BOARD

Figure 145. Processor Board Block Diagram

MC68020RC12A which is a 12 megahertz version of the processor. The system clock circuitry consists of a 16 MHz clock oscillator, U17, and "D" flip-flop connected as a divide-by-two circuit to provide an 8 MHz system clock. This clock is also connected to the digital mother board for use by other circuits on other boards. An 8 MHz clock was chosen to drive this circuitry because it provides adequate speed for the ADAM system, and at 8 MHz, the processor consumes less power than the processor does at 12 MHz.

The 68020 has a full 32-bit data bus and address bus that is available to the user. An analysis of the current and future drive requirements of the address and data bus, along with experience gained in the design of other HCMOS high speed microprocessor systems, indicated that the address and data buses need to be buffered. This provides a high speed, high current drive capability to allow many more ICs to be on the address and data bus without excessive loading.

The address bus buffers selected are 74HC244 octal buffers. These devices are U7-U10 in Figure 146. The address buffers are unidirectional since the address bus is an output-only function on the 68020. The data buffers, designated U1-U4 on Figure 146, are 74HC245 devices. These devices are bidirectional tri-state octal buffers. The tri-state control comes from the CPU control line $\overline{\text{DBEN}}$ (date buffer enable) which activates the buffers when the CPU is making a data bus access. Since the data bus is bidirectional to handle both reads and writes of data, the R/W (read/write) control line is used to select the direction of the data. When the CPU is reading data, pins 2-9 of U1-U4 are inputs and pins 11-19 are outputs. The R/W line is connected to pin 1 of U1-U4 and is high during a read. During a CPU write cycle, the R/W line is low and this reverses the input and output pins on the data bus buffers.

The system reset circuitry consists of two separate systems, an automatic power-up reset circuit and a manual reset. The automatic power-up reset uses a low power XR555 (U20) timer IC to produce a reset signal for approximately 50 milliseconds. The XR555 is activated only once during power-up and cannot be activated again unless the system is turned off. In order to allow the system to be reset while system power is on, a manual reset circuit was also included. The manual reset uses three NAND gates from the ICU21. Two NAND gates are connected as a set-reset flipflop and the third NAND gate is used to hold the flip-flop in the set mode until the reset command is given. The manual reset is accomplished by an external switch that will connect J2-14 to J2-13. When the connection is broken, the $\overline{\text{RESET}}$ line goes inactive and the processors begins its execution. Most of the CPU control bus signals have been connected to the digital mother board for use by circuits on other boards. Figure 147 outlines the functional signal groups

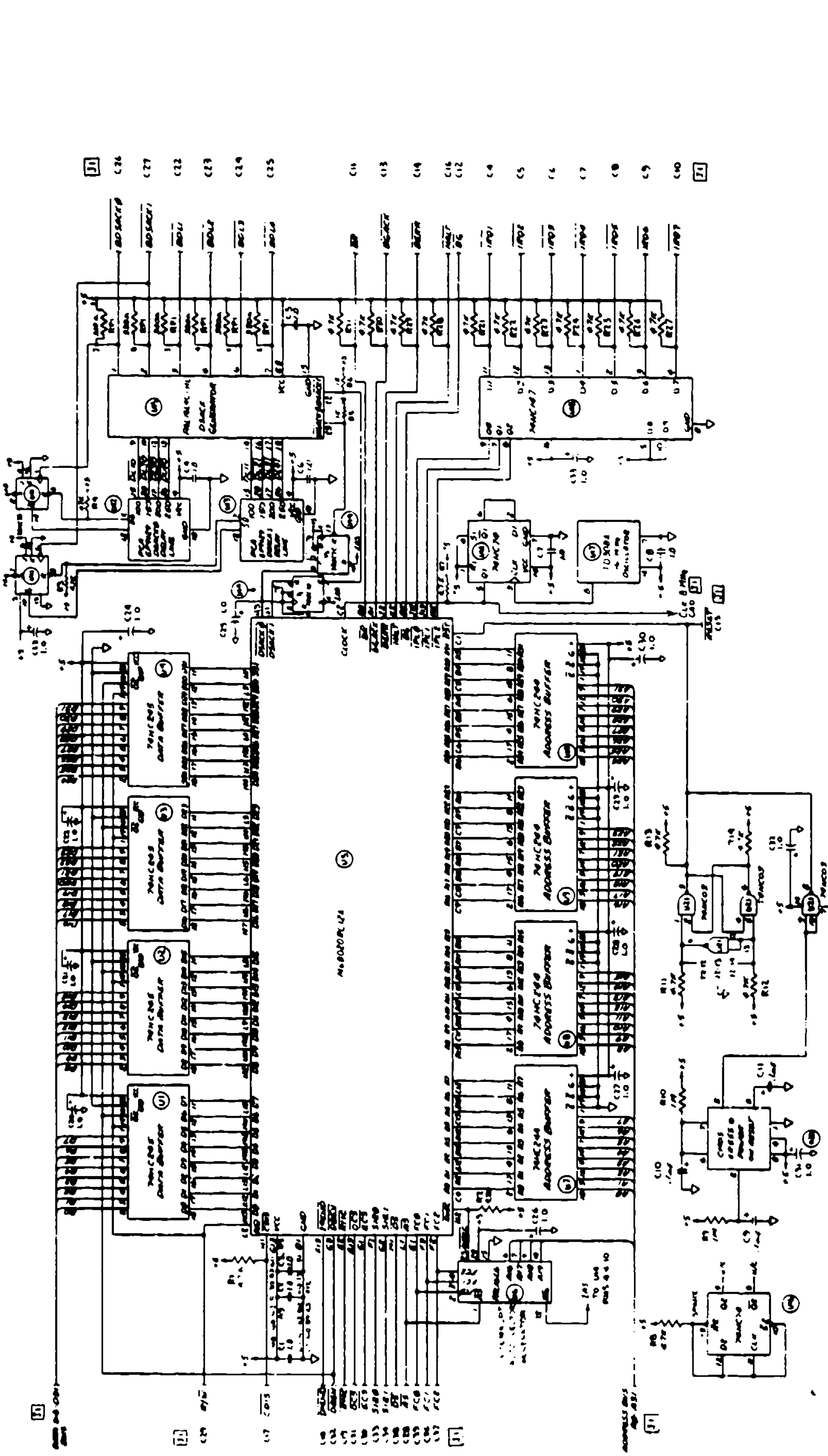


Figure 146. Microprocessor and Support Circuitry Schematic

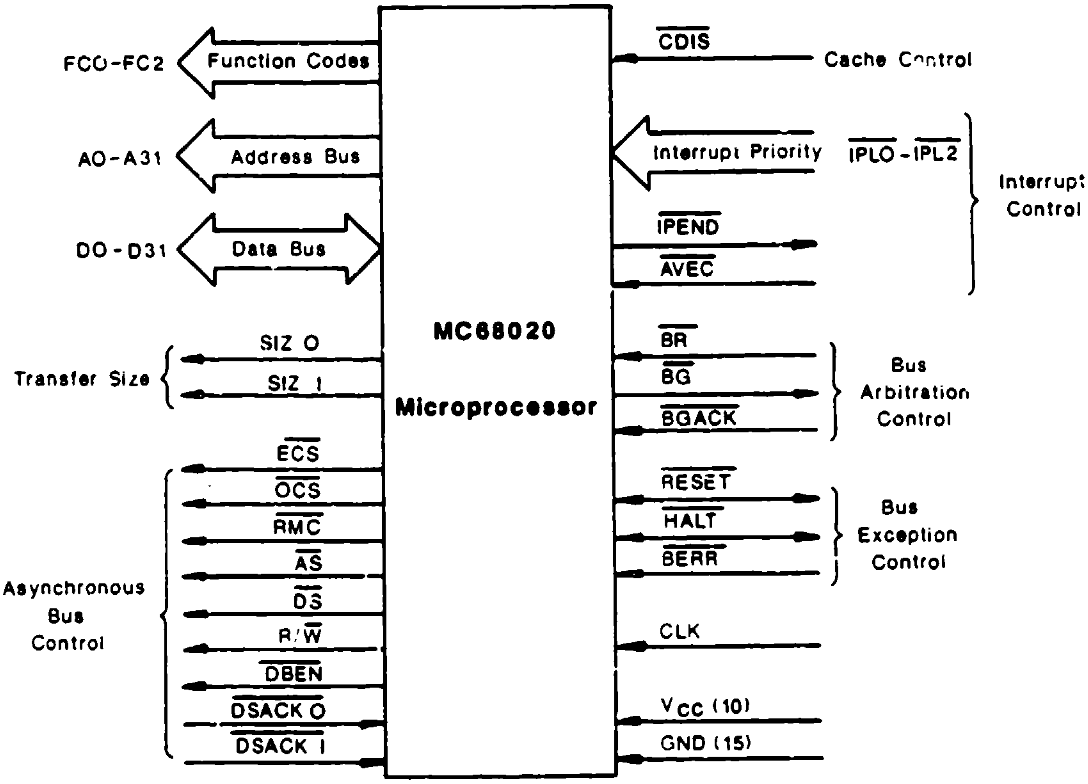



Figure 147. CPU Functional Signal Group Block Diagram

of the CPU. Some signals shown in Figure 147 are not connected to the digital mother board. $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ have been replaced by $\overline{\text{BDSACK0}}$ and $\overline{\text{BSACK1}}$, respectively. The interrupt control lines $\overline{\text{IPL0}}$-$\overline{\text{IPL2}}$ have been replaced with the lines $\overline{\text{IRQ1}}$-$\overline{\text{IRQ7}}$ on the digital mother board, and the $\overline{\text{AVEC}}$ line is used only by the interrupt vector generator on the CPU board, so it is not on the digital mother board. These signal substitutions on the digital mother board were made as a system level design consideration for the central interrupt control logic and the DSACK circuitry. As each of these systems is explained, the reasons for these substitutions will become evident.

The interrupt control logic performs two functions. It provides the processor with a signal to indicate there is an interrupt pending, and it generates an automatic jump vector for the processor to begin execution of the interrupt service routine. The 68020 has three control inputs, $\overline{\text{IPL0}}$-$\overline{\text{IPL2}}$, that are used to indicate to the processor that a device is requesting an interrupt. The 68020 has seven levels of interrupts that are prioritized in order of importance. Interrupt request level 7 is the highest priority and interrupt request level 1 is the lowest priority interrupt as shown below.

| /IPL2-0 | INTERRUPT PRIORITY |
|---|---|
| 111 (7) | 1 (Highest) |
| 110 (6) | 2 |
| 101 (5) | 3 |
| 100 (4) | 4 |
| 011 (3) | 5 |
| 010 (2) | 6 |
| 001 (1) | 7 (Lowest) |
| 000 (0) | No Int. Pending |

If two different levels of interrupt requests occur at the same time, then the higher priority interrupt will be serviced first. Since most peripheral ICs for microprocessors use a single line to request an interrupt from the processor, the seven interrupt levels, $\overline{IRQ1}$ to $\overline{IRQ7}$ were placed on the digital mother board and a decimal to binary coded decimal encoder, U18, is used to encode each of the processor's $\overline{IPL}$ lines according to the $\overline{IRQ}$ line that is asserted. U18 is a 74HC147 and is designed to encode the highest decimal value that is asserted to its binary outputs. This encoding technique ensures that the $\overline{IPL}$ lines are always encoded with the highest interrupt level requested.

The auto vector generator is the second part of the interrupt control logic on the processor board. The sequence of events in this system design when an interrupt occurs is:

- An interrupt request occurs.
- The processor recognizes the interrupt and finishes the instruction it is currently executing.
- The processor compares the interrupt request with the interrupt mask level to see if the interrupt level is enabled.
- If the interrupt mask level is higher than the requested level, the interrupt is ignored.
- If the interrupt level is above the mask level, the processor continues with the interrupt service.
- The R/W line is set to read.
- Set function code to CPU space (FC0-FC2 set to 111).
- Place interrupt level on A1, A2, and A3 and set A16-A19 to 1111.
- Set size to BYTE.
- Assert $\overline{AS}$ and $\overline{DS}$.
- Assert $\overline{AVEC}$.
- Latch jump vector (generated by CPU).
- Start processing interrupt ser[illegible] routine.

The auto vector generator circuitry on the processor board decodes the special outputs on the function codes signal lines and the address bus that indicates that the processor is processing an interrupt. This is then used to assert the $\overline{AVEC}$ control line of the processor. This entire function is handled by a single PAL IC, U6. The PAL asserts the $\overline{AVEC}$ line when $\overline{AS}$ is asserted, A16- A19 are all a logic one, and $\overline{FC0}$-$\overline{FC2}$ are all a logic one. Asserting $\overline{AVEC}$ informs the processor that it must generate the address vector where the interrupt service routine is located.

The last function block of CPU support circuitry to be discussed is the central DSACK control logic. Before this can be discussed, more detail regarding the operation of the processor must be given.

As stated earlier, the 68020 has a dynamic bus sizing feature that allows the processor to read and write to byte, word, and long word ports on any address boundary. During an operant transfer to a port, the port signals the processor the port size and transfer status (complete or not complete). The port size and transfer status is accomplished through the use of the data transfer and size acknowledge ($\overline{DSACK0}$ and $\overline{DSACK1}$) inputs to the processor. For example, if the processor attempts to write a 32-bit operant to a port, and if the port responds that it is a 32-bit port, the processor writes the 32 bits of data and continues. If the port responds that it is a 16-bit port, the processor writes the first 16 bits to the port and runs another write cycle to write the remaining 16-bits to the port. A similar action takes place with an 8-bit port, but there are four write cycles. The advantage of this circuitry is that any size port may exist in a 68020 system with an efficient use of the address space.

The port responds to the processor with its size using the system level interface lines $\overline{BDSACK0}$ and $\overline{BDSACK1}$. The BDSACK lines are inputs to the DSACK control logic that generates the $\overline{DSACK0}$ and $\overline{DSACK1}$ lines for the processor. The BDSACK lines are open collector lines that are used by all of the ports in the digital system to respond with their port size. There are four other system level open collector lines, $\overline{BDL1}$-$\overline{BDL4}$ (buffered delay line) that are used to signal any increase in the access time required of a port that is slower than the processor. This function is also handled by the DSACK control logic. The lines $\overline{BDSACK0}$, $\overline{BDSACK1}$, and $\overline{BDL1}$-$\overline{BDL4}$ are on the digital mother board and available to all circuit boards in the system. Table 45 lists the combinations of $\overline{DSACK0}$ and $\overline{DSACK1}$ that determines the size of the port.

TABLE 45. /DSACK0-1 VALUES FOR DIFFERENT PORT SIZES

| /DSACK0-1 | Port Size |
|---|---|
| 00 | Inactive |
| 01 | 8 Bits |
| 10 | 16 Bits |
| 11 | 32 Bits |

Figure 148 shows the timing relationship between the assertion of the BDSACK lines and the DSACK inputs to the processor when no delays are requested ($\overline{BDL1}$-$\overline{BDL4}$ are all high). As soon as the BDSACK lines are asserted, the propagation delay in the DSACK control logic is the only delay in the assertion of the DSACK lines. Figure 149 shows the timing relationship between the BDSACK, BDL, and DSACK lines when a delay is requested ($\overline{BDL2}$ is asserted). As Figure 149 shows, when $\overline{BDL2}$ is asserted, a 150 nanosecond delay occurs between the assertion of the BDSACK lines and the assertion of the DSACK lines. This delay occurs while both the address and data bus have valid information allowing slower devices requiring longer setup times to work correctly.

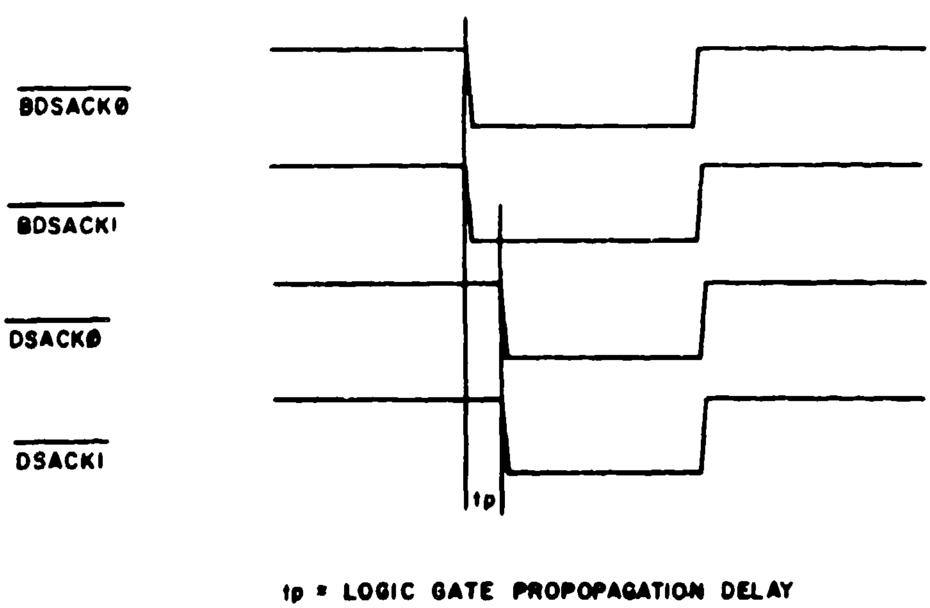



Figure 148. DSACK Timing Diagram - No Delays (32-Bit Port)

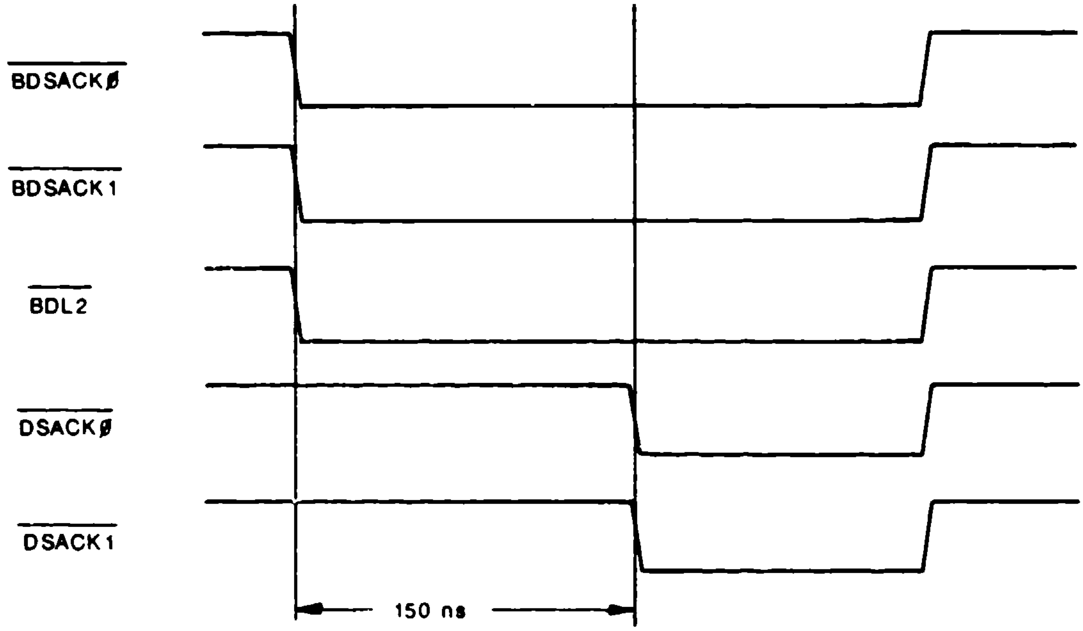



Figure 149. DSACK Timing Diagram with BDL2 Asserted

The DSACK control logic uses the PAL U15, the two delay lines U12 and U13, the DSACK The latches U14, and the J- flip-flops, U11 (Figure 146). The inputs from the digital mother board, $\overline{\text{BDSACK0}}$, $\overline{\text{BDSACK1}}$, and $\overline{\text{BDL1}}$-$\overline{\text{BDL4}}$ are input to the DSACK generator, PAL U18, and the BDSACK lines are tied to the clock inputs of the J-K flip-flops of U11.

As the BSDACK lines are asserted, the falling edge of the BDSACK line will toggle the appropriate J-K flip-flop of U11. The output of the flip-flop drives the input to the delay modules, U12 and U13. The DSACK generator uses the $\overline{\text{BDSACKX}}$, $\overline{\text{BDLX}}$, and $\overline{\text{DLXX}}$ lines to generate an intermediate $\overline{\text{DSACKX}}$ signal called $\overline{\text{IDSACKX}}$. The $\overline{\text{IDSACKX}}$ signal resets the "D" flip-flop U145, and it generates a $\overline{\text{DSACKX}}$ signal. At the completion of the bus cycle, the line IAS (the inversion of $\overline{\text{AS}}$) sets the flip-flop and the DSACK circuitry is ready for the next bus cycle. Figure 150 illustrates the timing of these signals to help clarify the operation of this circuitry. A central $\overline{\text{DSACK}}$ control circuit was used to minimize the space required on the other circuit cards. Six lines of the digital mother board bus are all that was required to implement this circuitry on the processor board.

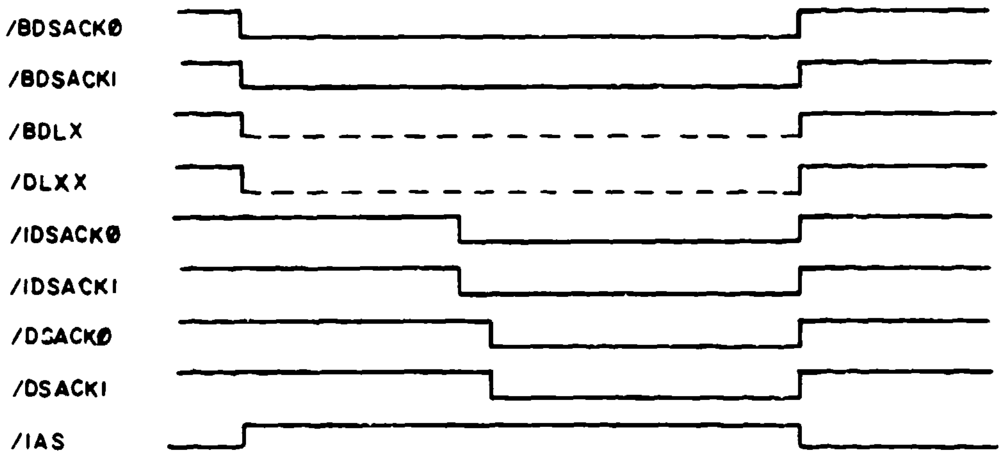



Figure 150. /BDSACKX, /IDSACKX, and /DSACKX Timing

This discussion of the operation of the processor and its support circuitry has been brief to emphasize only those points that were necessary for the previous discussion. For more information on the 68020 and its operation, see MC68020 Bit Microprocessor User's Manual, 2nd Edition, which is published by Prentice-Hall, Englewood Cliffs, New Jersey.

3.3.2.3. Multifunction Peripheral

The multifunction peripheral (MFP) is an IC that contains a universal synchronous/asynchronous receiver transmitter (USART) for serial communication, four programmable timers, an 8-bit parallel port, and interrupt request circuitry. This multifunction peripheral is a Motorola MC68901 IC that is designed to interface with Motorola's 68000 series family of processors. The ADAM instrumentation uses the MFP for serial communications using the USART and the four programmable timers to generate the four filter clocks used in the signal conditioning circuitry.

Figure 151 shows the schematic of the MFP and all the support circuitry associated with it. The MFP, designated U26, is an 8-bit wide port. It is placed on the data bus from D24 to D31 because the processor's dynamic bus sizing circuitry requires that 8-bit ports be located on that portion of the data bus. The MFP has 24 separately addressable registers to control the various functions of the IC, so five address lines (A0-44) are connected to the MFP to select these registers. The base

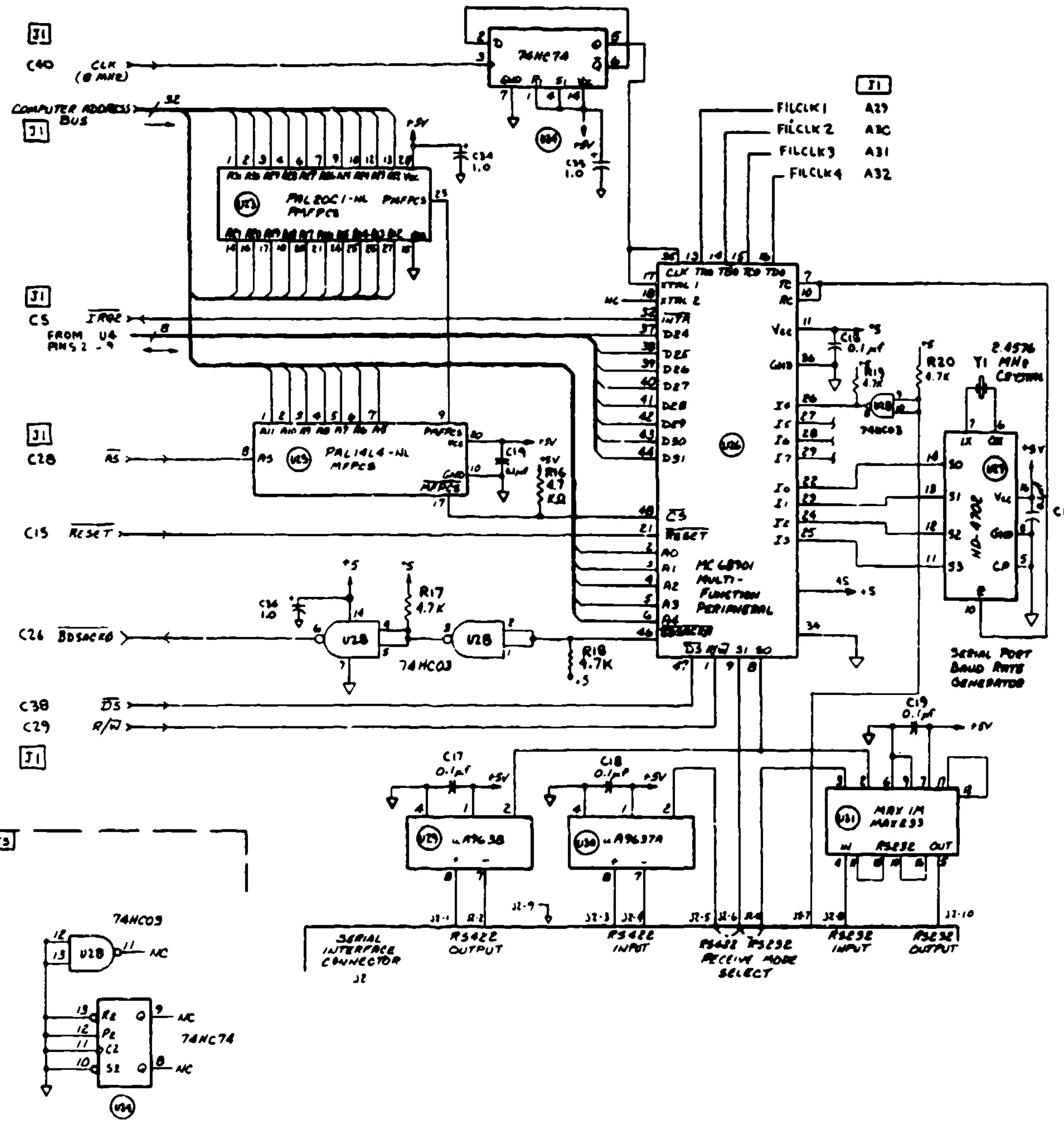



Figure 151. ADAM Multifunction Peripheral (MFP) Schematic

address of the MFP is 800040 (HEX) and the address decoding is performed by the PALs U23 and U25, and the MFP chip select, $\overline{\text{MFPCS}}$, is connected to the $\overline{\text{CS}}$ line of the FP (pin 43). The $\overline{\text{BDSACK0}}$ line is asserted through circuitry onboard the MFP (pin 46) and is buffered by two open collector NAND gates connected as inverters to provide the proper interfacing to the /BDSACKX bus. The MFP requires a clock to maintain internal timing operations. The system clock is used with a "D" flip-flop (U24) connected as a divide-by-two circuit to provide a 4 MHz clock to the MFP. The MFP maximum clock frequency is 4 MHz so the divide-by-two circuit was installed to provide the proper frequency from the system clock.

The USART is a single full duplex serial channel that utilizes a double buffered receiver and transmitter with interrupt capabilities. The USART requires an external clock for both the receiver and transmitter in order to provide the proper baud rate. The USART is used in the asynchronous mode for the ADAM instrumentation which allows the baud rate clock to be either the same frequency as the baud rate or 16 times the baud rate. The 16 times clock mode offers better noise immunity than the times 1 clock and is used in the ADAM system.

The selection of the clock rate, data format, and data parity is done in software by writing to the USART Control Register [address 800055 (hex)]. The clock modes available have been discussed previously, and Table 46 outlines the possible data and parity formats. There is a receiver status register [address 800056 (hex)] that enables the receiver and provides information on errors in the data (such as parity error) and the status of the receiver buffer (full or empty). There is a corresponding transmitter status register [address 800067 (hex)] that provides error flags, transmitter buffer status, and enables the transmitter. A USART data register [address 800058 (hex)] writes data to the transmitter buffer and reads data from the receiver register.

The clocks for the receiver and transmitter are the same and are generated by the programmable baud rate generator U27. The inputs S0-S3 (pin 14-11, respectively) of U27 are tied to the lower nibble of the 8-bit parallel port on MFP. These four bits are programmed in MFP as outputs and allow a software selectable baud rate to be generated. Table 47 shows the different baud rates available for different inputs of S0-S23 of U27. The high nibble bits of the MPF parallel port are programmed as inputs.

TABLE 46. USART PARAMETER SUMMARY

| Parameter | Selection |
|---|---|
| Word Length | 5 |
| | 6 |
| | 7 |
| | 8 |
| Parity | Even |
| | Odd |
| | None |
| Stop Bits | 1 |
| | 2 |
| Baud Rate | 50 |
| | 75 |
| | 110 |
| | 134 |
| | 150 |
| | 200 |
| | 300 |
| | 600 |
| | 1200 |
| | 1800 |
| | 4800 |
| | 9600 |

TABLE 47. USART BAUD RATE SELECTIONS

| $S_0$ - $S_3$ | Baud Rate |
|---|---|
| 0000 | N/A |
| 0001 | N/A |
| 0010 | 50 |
| 0011 | 75 |
| 0100 | 134.5 |
| 0101 | 200 |
| 0110 | 600 |
| 0111 | 2400 |
| 1000 | 9600 |
| 1001 | 4800 |
| 1010 | 1800 |
| 1011 | 1200 |
| 1100 | 2400 |
| 1101 | 300 |
| 1110 | 150 |
| 1111 | 110 |

Bit I4 is used as a terminal connected status bit. The input is buffered through the open collector NAND gate of U28 to prevent damage to the MFP. If bit I4 is low, then a receiver for the serial is not connected. If I5 is high, then a receiver is attached to receive serial data.

The serial output is provided in two formats, RS-232 standard and RS-422 standard. The line drivers and receivers are located on U31 for the R2-232 data, and the RS-422 receiver is in U30 while the RS-422 driver is in U29. The serial output line, S0 (pin 8) of the MFP, is connected to both drivers, but the serial input line, SI (pin 9) of the MFP, is jumper selectable from the RS-232 receiver or the RS-422 receiver.

The MFP is also used to generate the four filter clocks used in the signal conditioning circuitry. The MFP has four programmable timers that may be programmed with a clock frequency from DC (clock stopped) to 500 kHz.

Each timer is designed with an 8-bit binary down counter and a prescaler register. The prescale register controls the prescaling of the count from a divide-by-four to divide-by-200. Table 48 outlines the possible modes the prescalers may be programmed to enter. In the delay mode, the prescaler specifies the number of counts that must pass before a count pulse is sent to the main counter. When the main counter has been decremented to 01 (hex), the next count pulse causes the main counter to send out a time out pulse to reload from the timer data register, and the output of the timer will toggle. The timer output remains in this state until the next time-out pulse occurs. This is the mode that is used to generate the filter clocks. The four timer outputs (pins 13-16) are connected directly to the digital bus where they are routed to the analog mother board by way of the A/D board.

TABLE 48. PRESCALER MODES FOR TIMERS

| Control Register Bits 2-0 | Timer Mode |
|---|---|
| 000 | Stopped |
| 001 | Divide by 4 |
| 010 | Divide by 10 |
| 011 | Divide by 16 |
| 100 | Divide by 50 |
| 101 | Divide by 64 |
| 110 | Divide by 100 |
| 111 | Divide by 200 |

### 3.3.2.4. Miscellaneous Processor Board Circuitry

Figure 152 shows the assembly drawing of the processor board. The mother board connector, J1, is a 160 pin connector that joins the processor board to the digital mother board. The address bus, data bus, and control bus are all placed on the digital mother board by the processor board. The connector J2 is a serial interface connector that is provided for the serial communications link. The connector provides the RS-232, RS-422, and reset lines for interfacing with the user.

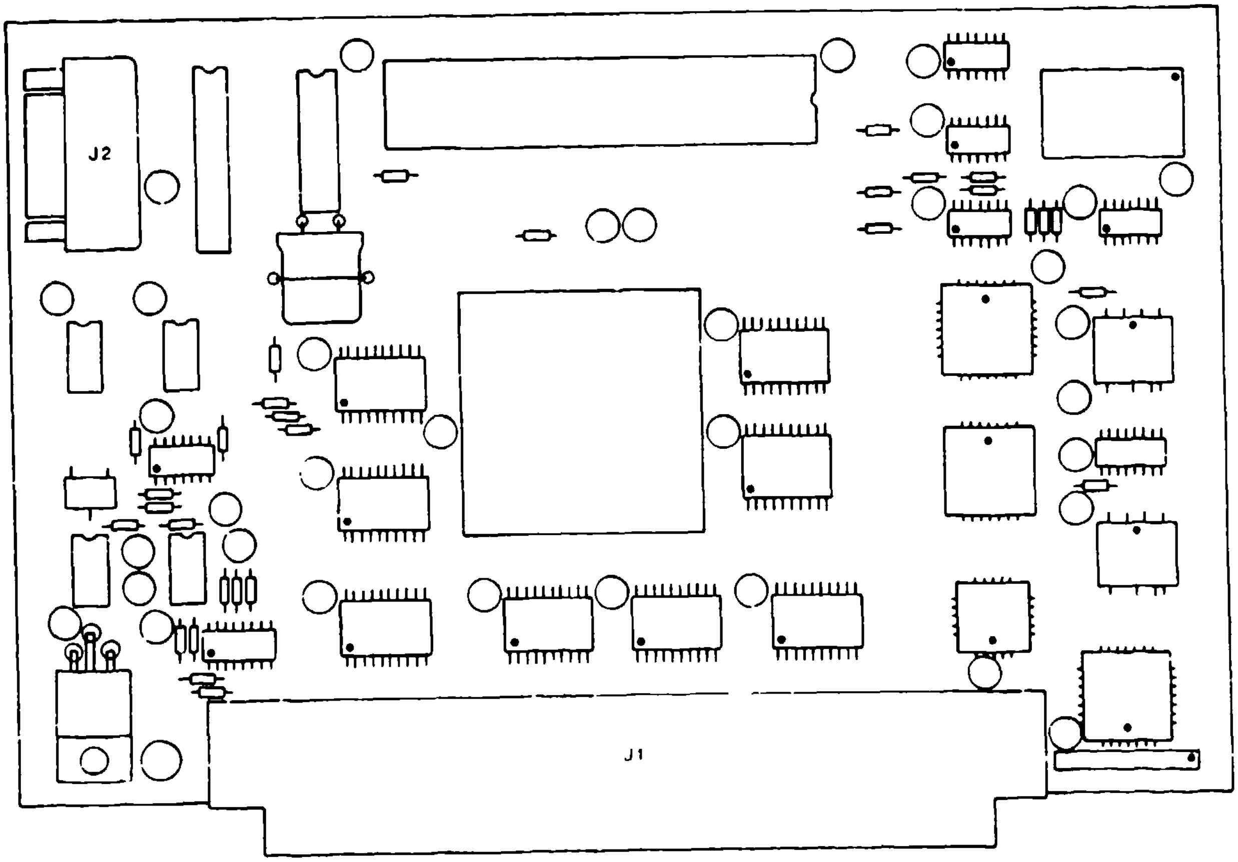



Figure 152. Processor Board Assembly

Figure 153 shows the power supply's schematic for the processor board. A single 5V regulator is used for the entire processor board. The circuit is functionally the same as the power supply circuits discussed earlier. This regulator is connected so that it cannot be shut down. The circuitry on this board requires approximately 400 milliamperes of current to operate.

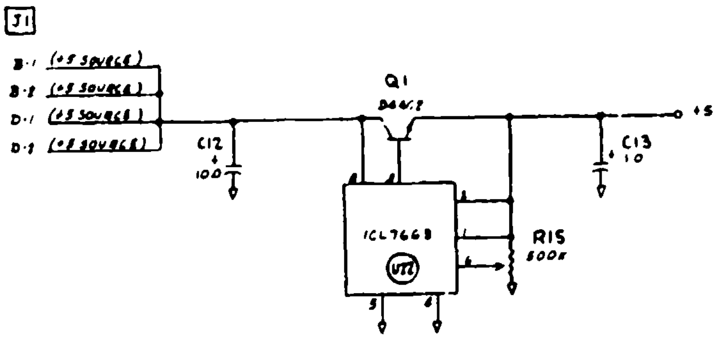



Figure 153. Processor Board Power Supply Schematic

### 3.3.3. Memory Board

The memory board contains two separate circuits--the data memory for storing the test data collected by ADAM, and the program memory used to store the software that ADAM executes. Figure 154 is the block diagram of the memory board. It shows that the data random access memory (RAM) is organized as a 128 Kbyte by 32-bit wide memory bank and the program memory is organized as a 32 Kbyte by 16-bit wide array using erasable programmable read only (EPROMs). The memory board address decoding and control circuitry for each memory array is separate and will be discussed with each individual circuit.

#### 3.3.3.1. Static RAM Array

The static RAM (SRAM) array is the memory used to store the ADAM test data. It also is used by the processor to store temporary data generated during the execution of the software and to store the pretest and posttest calibration data. The SRAM also is battery backed-up by two lithium cells to prevent the loss of data during a catastrophic failure of the power supply.

Figure 155 is the schematic for the SRAM array. The SRAM array is made up of the two SRAM array assembly boards (SAAB) that contain two 128 Kbyte by 8-bit SRAM single in-line packages (SIPs) that are made by Advanced Electronic Packaging. The SAAB is an assembly that was used to mount the SRAM SIPs horizontally instead of the standard vertical mounting technique.

Figure 156 shows the assembly for the SAAB. The only circuitry onboard the SAAB are two Advanced Electronic Packaging AEPSS128K8A SRAM SIPs. The printed circuit board is used to separate the data lines for each SIP and the chip select line, since these are unique connections for each SIP. The address lines and the remainder of the control lines are connected in parallel. Figure 157 is the schematic for the SAAB that illustrates these connections.

The SRAM is arranged as a 128K x 32-bit wide array in order to allow the processor to write long words (4 bytes) of data at a time. The access time of the SRAM is 120 nanoseconds so there is no need for any delay in the reads or writes to the port.

The address bus and the data bus of the memory board are buffered to provide the guaranteed drive for all of the devices on the memory board. The data buffers U9-U12 are the wired the same functionally as the data buffers on the processor. The address lines A2-A17 are buffered using 74HC244 (U13-U14) in the same manner as the processor boards. The address lines A18-A20 are buffered using U2 and U3, 74HC03 open collector NAND gates, with two NAND gates

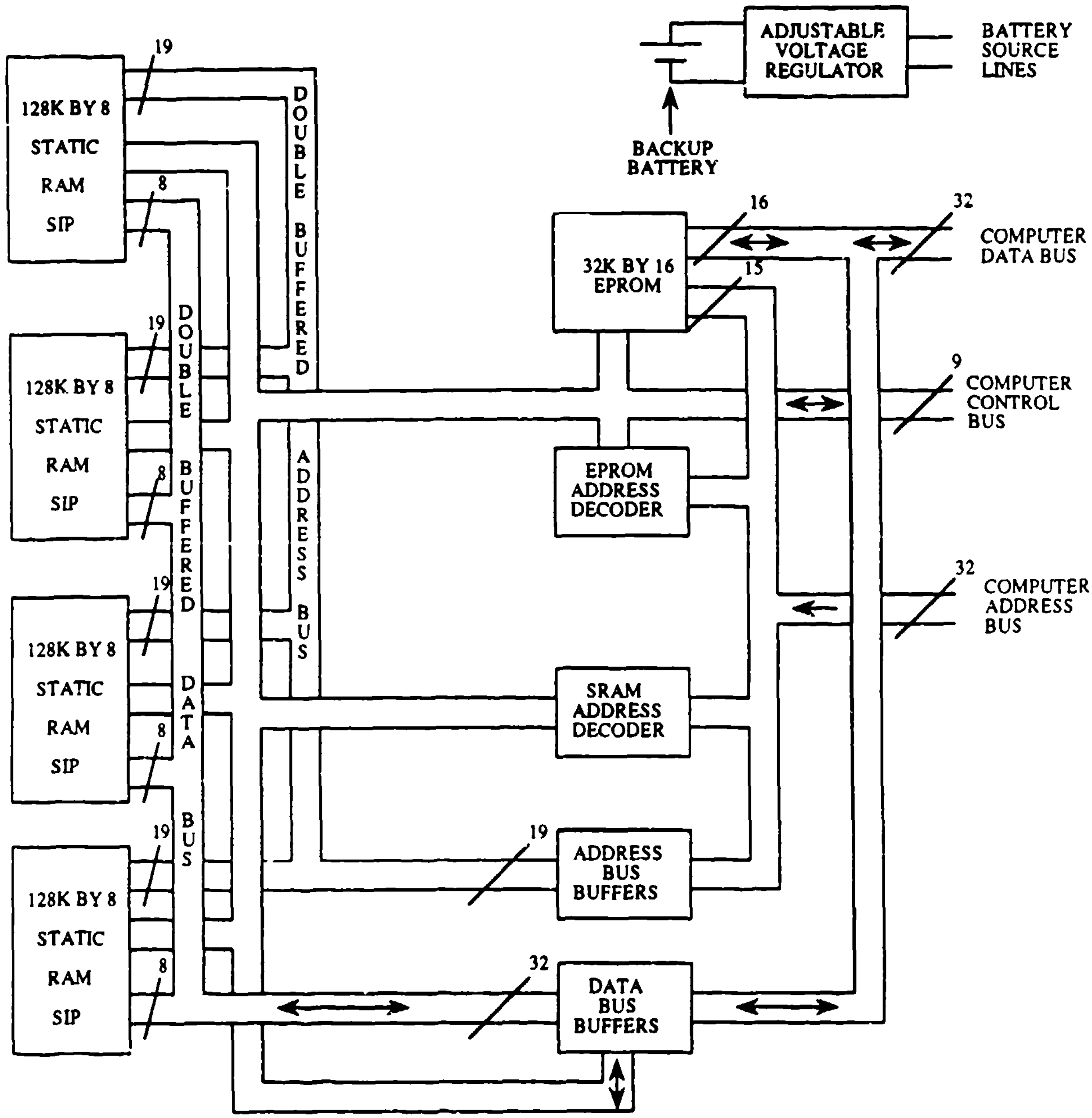



Figure 154. Memory Board Block Diagram

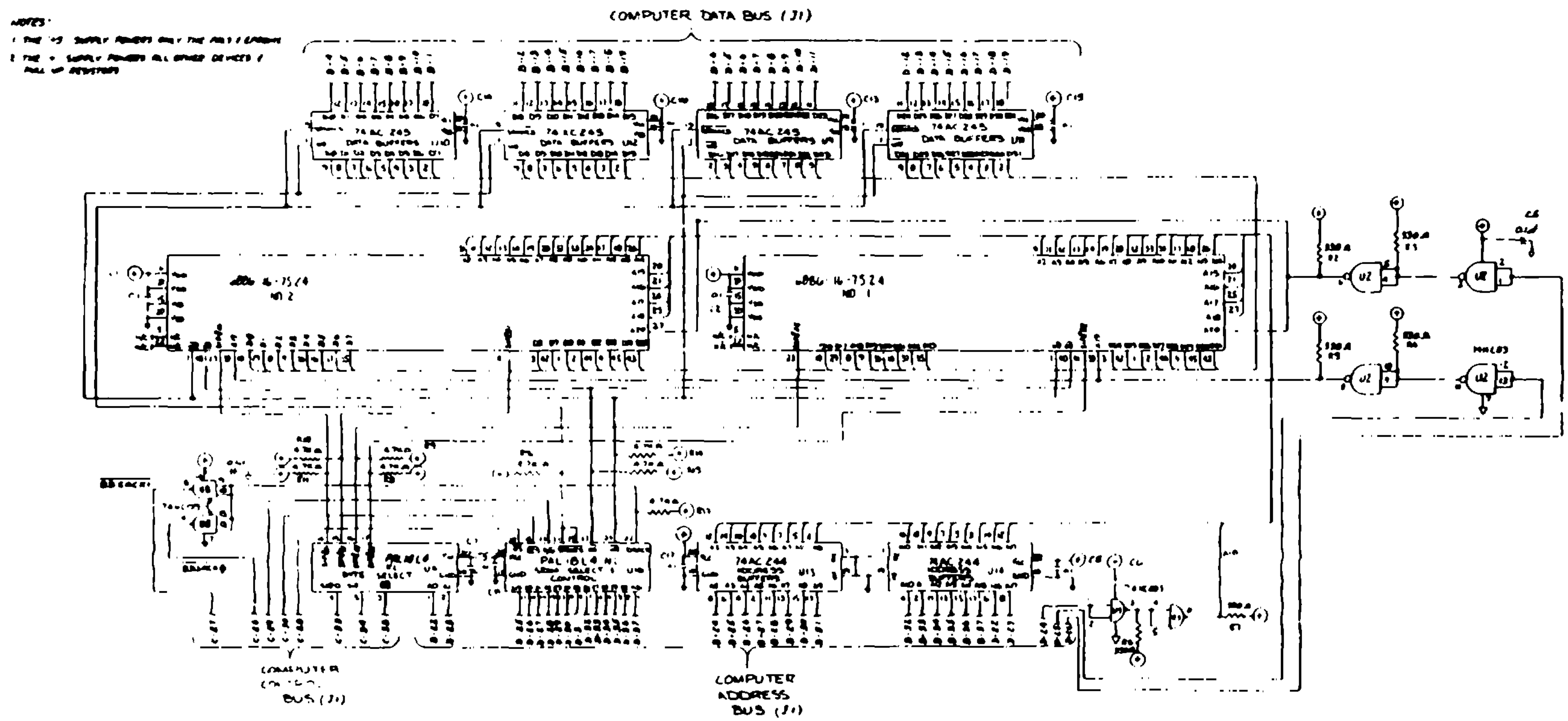



Figure 155. Memory Board Schematic-SRAM Array

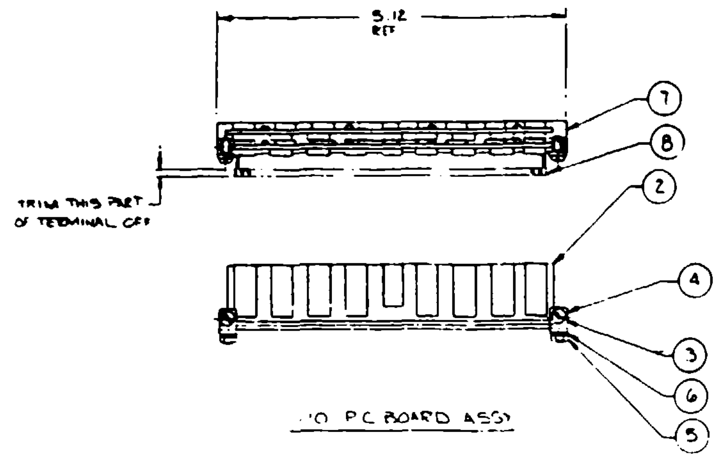


Figure 156. SRAM Array Assembly Board (SAAB)

connected as inverters in series. This was done to conserve space and to provide for some expansion in the system.

The expansion capability of the system is in the size of the SRAM array. The current SRAM array requires address lines up to A18. The lines A19 and A20 are not used and U2 is not installed in the system. For a 512K by 32-bit SRAM array (four times the current memory size), U2 is installed to provide address decoding to the SRAM array, and the 128 Kbyte by 8-bit SRAM SIPs with 512 Kbyte by 2-bit SRAM SIPs that are available from Advanced Electronic Packaging (P/N AEPSS512K8-12). No change in the SAAB design is needed nor is any change required of the memory board. The only change in the system would be the address decoding circuitry.

The address decoding circuitry consists of two PALs, U4 and U16, which generate all the address selects and chip selects necessary. U16 uses the combination of A19-A31, $\overline{AS}$, and $\overline{ECS}$ to generate the control input $\overline{SRAMCS}$ to the two SAABs. U16 also uses the combination of A19-A31 , $\overline{AS}$, and R/W to generate the $\overline{OE}$ and $\overline{WE}$ control lines. $\overline{SRAMCS}$ is a control line that activates the address decoding circuitry on each of the SRAM SIPs. The $\overline{OE}$ (output enable) control line is used to read data from the SRAM SIPs and the $\overline{WE}$ (write enable) control line is used to write data to the SRAM SIPs.

The PAL U4 is used to generate the individual SIP enable lines. Full dynamic bus sizing has been implemented on this 32-bit port, so each SIP's eight data lines appear on an 8-bit boundary of the data bus. U4 uses the different combinations of A0-A1, SIZ0, SIZ1, and $\overline{DS}$ to generate the individual SIP enables. Table 49 illustrates which parts of the data bus are active during byte, word (2 bytes), 3 byte, and long word transfers. It also illustrates which SIP enable line is active during

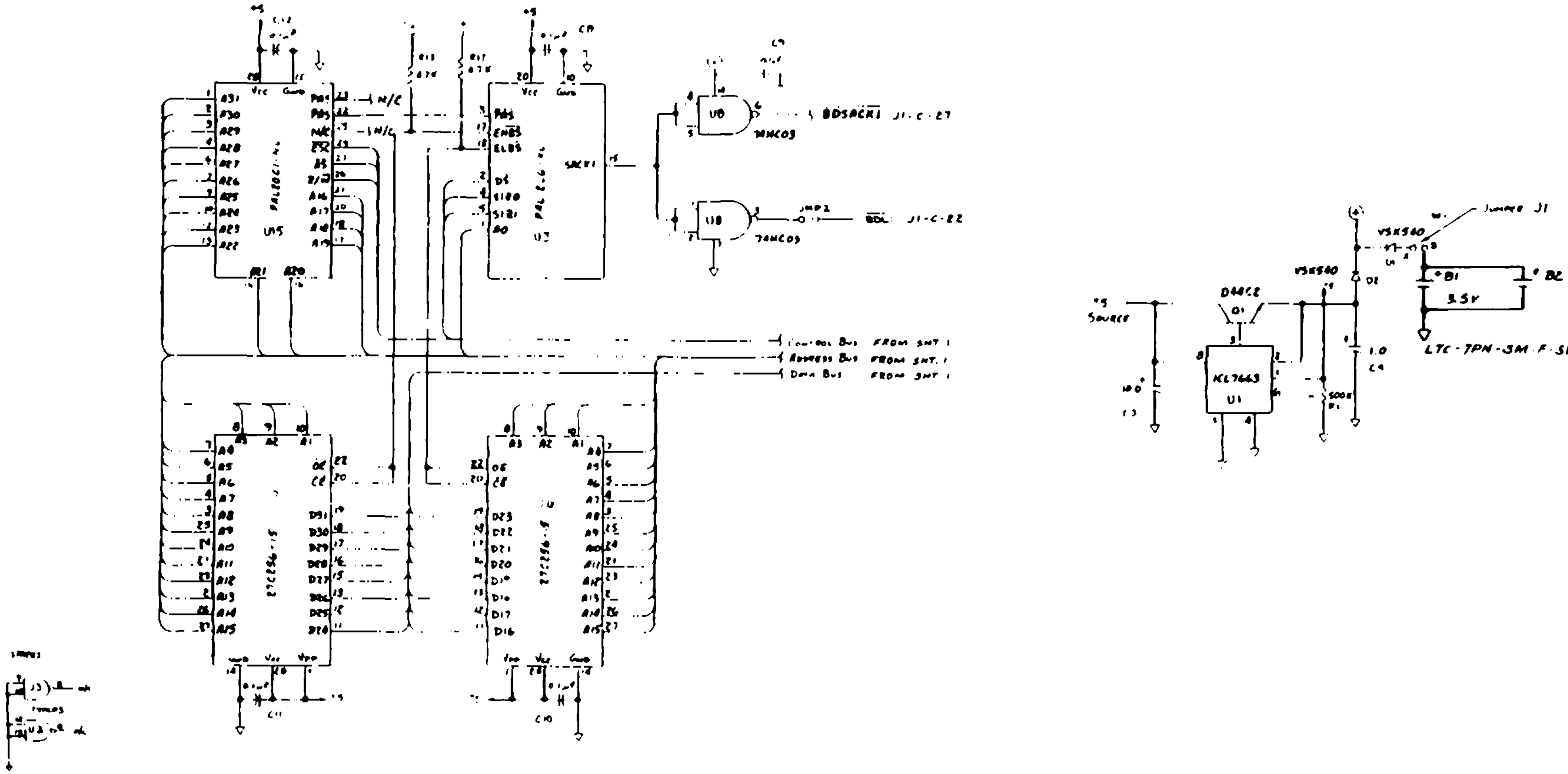



Figure 157. Memory Board

TABLE 49. DATA BUS ACTIVITY FOR BYTE, WORD, AND LONG WORD PORTS

| Transfer Size | SIZ1 | SIZ0 | A1 | A0 | Date Bus Active Sections Byte (B) - Word (W) - Long Word (L) Ports D31-D24 | D23-D16 | D15-D8 | D7-D0 |
|---|---|---|---|---|---|---|---|---|
| Byte | 0 | 1 | 0 | 0 | B W L | - | - | - |
| | 0 | 1 | 0 | 1 | B | W L | - | - |
| | 0 | 1 | 1 | 0 | B W | - | L | - |
| | 0 | 1 | 1 | 1 | B | W | - | L |
| Word | 1 | 0 | 0 | 0 | B W L | W L | - | - |
| | 1 | 0 | 0 | 1 | B | W L | L | - |
| | 1 | 0 | 1 | 0 | B W | W | L | L |
| | 1 | 0 | 1 | 1 | B | W | - | L |
| Three-Byte | 1 | 1 | 0 | 0 | B W L | W L | L | L |
| | 1 | 1 | 0 | 1 | B | W L | L | L |
| | 1 | 1 | 1 | 0 | B W | W | L | L |
| | 1 | 1 | 1 | 1 | B | W | - | L |
| Long Word | 0 | 0 | 0 | 0 | B W L | W L | L | L |
| | 0 | 0 | 0 | 1 | B | W L | L | L |
| | 0 | 0 | 1 | 0 | B W | W | L | L |
| | 0 | 0 | 1 | 1 | B | W | - | L |

the same bus cycle. This type of addressing was designed into this port to allow the processor to transfer any size data on any address boundary.

The DSACK response of the memory port is generated by U16 whenever the $\overline{\text{SRAMCS}}$ line is asserted. The base address for the memory is 1000000 (hex) and extends to 107FFFF (hex) for the 128K by 32-bit array. The ADAM SOW specifies 512,000 samples of data (plus calibration data) be available for data storage in the memory array. The ADAM system has 524,288 bytes of memory available. The 12,288 bytes that are not used for data storage are used to store pretest and posttest calibration data, and the processor uses it as a stack for temporary data storage.

3.3.3.2. EPROM Memory Array

The EPROM array is used to hold the machine code for the microprocessor to execute the data acquisition programs. This array is organized in a 32K by 16-bit wide configuration using two 32 Kbyte by 8-bit wide EPROMs. The schematic for the EPROM port is shown in Figure 157. The two EPROMs, U6 and U7, are 27C256-15 CMOS EPROMs with a maximum access time of 150 nanoseconds. U6 is on the data bus from D16 to D23, and U7 is on the data bus from D24 to D31. This configuration is standard for 16-bit ports on the 68020 data bus. The EPROM signal lines for the data bus and the address bus are connected directly to the digital mother board. Since the EPROMs did not represent the large capacitive load to the address and data bus, the signals were not buffered on the memory board.

The address decoder logic is accomplished through a single PAL, U15. The base address of the EPROMs is 0 (hex) and the memory extends to FFFF (hex). U15 is programmed to assert the partial address select ($\overline{PAS}$) line when the correct combination of A16-A31, R/$\overline{W}$, and $\overline{AS}$ is present. The PAL U5 uses the line to generate the EPROM chip select lines. U5 uses the combination of ($\overline{PAS}$), ($\overline{DS}$), S1Z0, S1Z1, and A0 to generate the EPROM high byte select ($\overline{EHBS}$) and the EPROM low byte select ($\overline{ELBS}$). ($\overline{EHBS}$) is used to select U7 which is on the upper byte of the data bus, and ($\overline{ELBS}$) is used to select U6 which is on the lower byte of the data bus. U5 will also generate the $\overline{BDSACK1}$ response using U8 to drive the bus. The EPROMs selected do not require delays in order for the processor to access the data, but in the case where slower devices are used in future applications, a 100 nanosecond delay may be requested by using jumper JMP2 to assert $\overline{BDL1}$. This allows a 100 nanosecond delay in the processor receiving a DSACK response as described earlier.

### 3.3.3.3. Miscellaneous Memory Board Circuitry

Figure 157 also shows the schematic for the battery backed power supply for the memory board. The voltage regulator circuit, using Q1 and U1, is the same circuit that has been described earlier. There is no provision for this regulator to be shut down by the processor, but a provision was made to keep the SRAM array powered in case there was a loss of primary power. The backup power source are two 3.5V lithium cells. The diode D2 blocks the battery voltage from feeding into the regulator circuit and damaging it, and the diode D1 is used to block the voltage produced by the regulator. The lithium cells are not rechargeable and should never be placed in a reverse-current mode where they may be charged. In Figure 157, the voltage output marked with a "+" powers all CMOS devices that are used with the SRAM array. The EPROMs and PALs are not powered by the batteries backup system so the voltage marked "+5" is used to power these ICs. They are not backed up by the batteries due to the large current requirements of these devices. Current indications from tests is that the lithium battery can power the system for approximately 6 days. A wire jumper J1 is on the board to allow the user to use the battery backup only when the jumper is installed.

Figure 158 shows the memory board assembly. The dimensions of the board are 6.85 inches x 4.43 inches. The connector J1 is a 160 pin connector that mates with the digital mother board to bring all of the bus signals required by the memory board from the digital mother board.

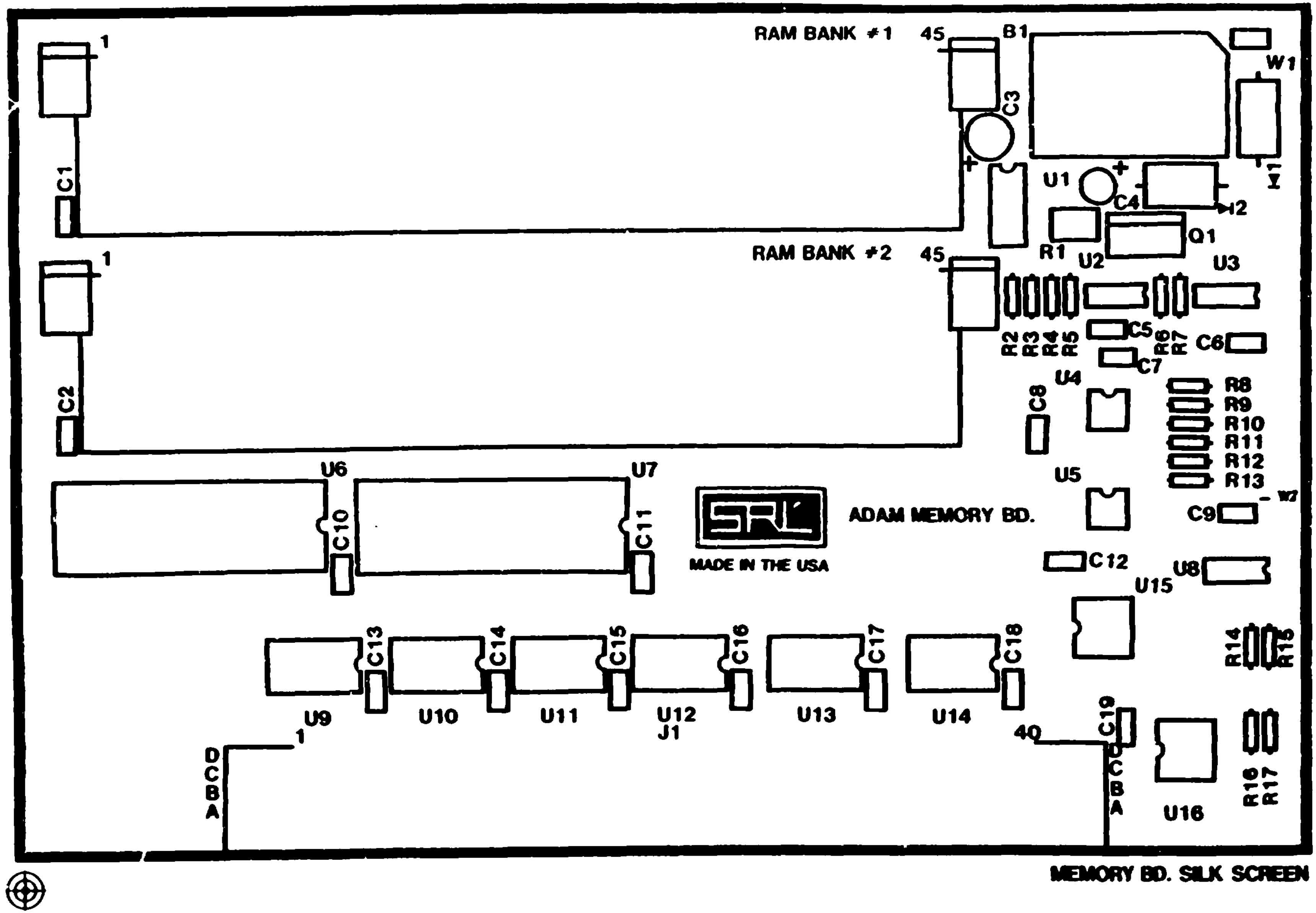



Figure 158. Memory Board Assembly

#### 3.3.4. Digital I/O Board

The last daughter board of the digital subsystem to be discussed is the digital I/O board. This circuit card assembly contains three major function blocks. The first is the telemetry port which is used to provide the data collected to a transmitter using a pulse code modulation (PCM) technique. The PCM data may also be provided over a hard wire link in cases where such a link may be used. The second is a 32-bit high speed parallel port that is used to transfer data from the ADAM SRAM array to the data retrieval and storage system (DRASS). This parallel port is able to swiftly offload the test data to the DRASS after a test event. The third circuit is a computer status and control (STAT/CONTROL) port. This port is used by the processor to control certain instrumentation functions such as power control, $R_{cal}$ control, and parallel port handshaking. The STATUS/ CONTROL port also is used by the processor to report the status of the manikin such as diagnostics complete, $R_{cal}$ mode, test complete, and others.

The discussion of this board will begin with a functional description of the board combined with detailed descriptions of each of the circuits on the board.

##### 3.3.4.1. Functional Description

Figure 159 is a block diagram of the digital I/O board. This diagram shows that the three circuits, the telemetry port, the parallel port, and the STAT/CONTROL port, have separate address decoder and control logic blocks and the circuits function separately from each other but are resident on the same circuit card. The separate address decoder and control logic blocks were implemented to facilitate the case of expansion or upgrades of the circuit functions.

As can be seen in Figure 159, the telemetry port consists of a buffered parallel-to-serial converter, telemetry encoding circuitry, a frame synchronization pulse, and output low pass filter. This circuitry provides an interrupt-driven pulse code modulation (PCM) output that provides a nonreturn to zero-level (NRZ-L) and biphase-level (BIØ-L) outputs.

The parallel port provides a high speed 32-bit half-duplex configuration that will interface with 8-, 16-, and 32-bit parallel ports meeting the port's interface requirements. The parallel port consists of 32 data lines plus the required handshaking lines, input and output latches, and control circuitry.

The purpose of the parallel port is to provide an interface to the data retrieval system that allows a fast upload of the test data to the data retrieval system.

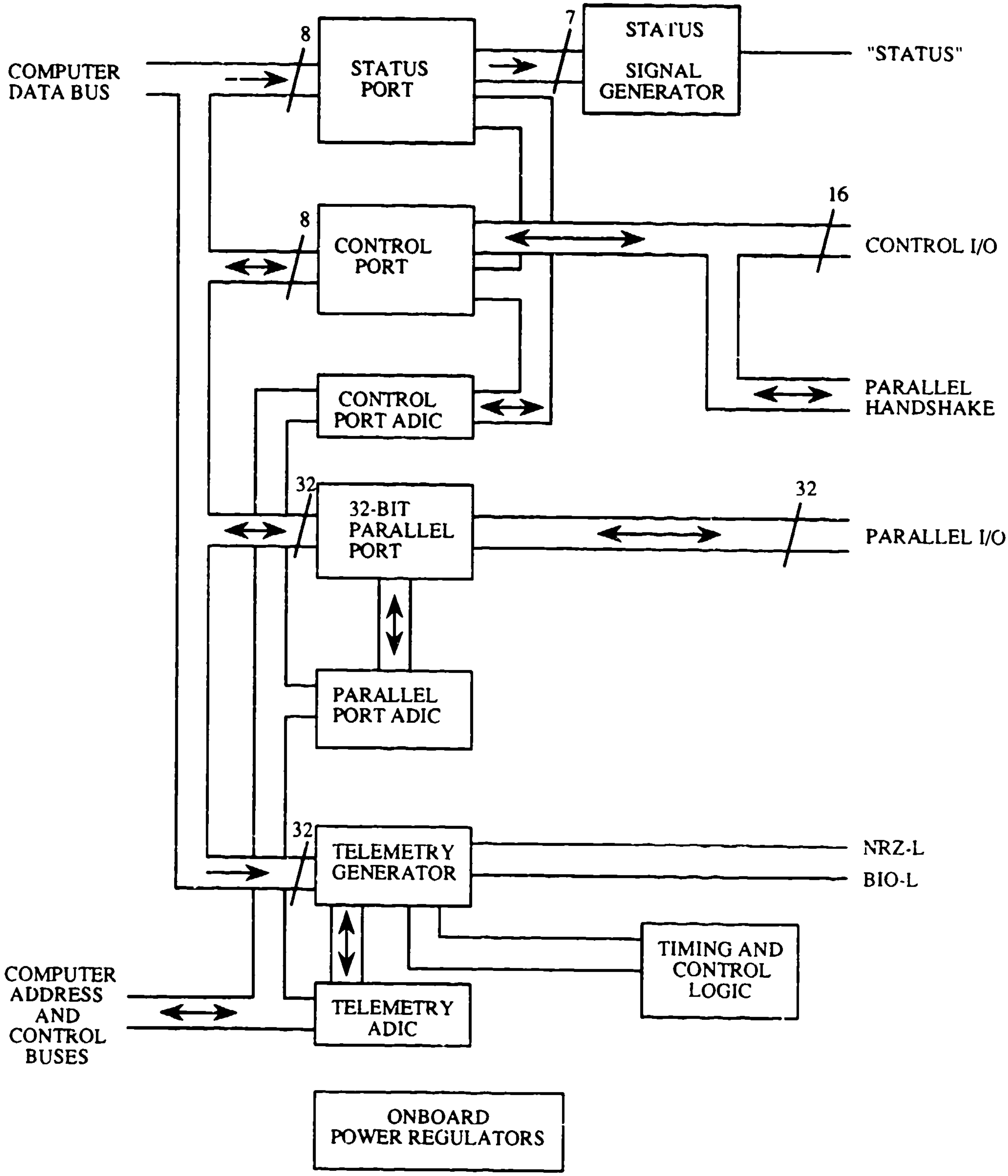



Figure 159. Digital Input/Output Board Block Diagram

The STATUS/CONTROL port consists of two separate blocks. The status block consists of an 8-bit latch and an 8-bit digital-to-analog converter (DAC). This circuit provides a single analog output that is digitized by the processor. This is used by the processor to report its status as it executes the ADAM software. Note only seven lines are used from the latch to the DAC. This is done to account for a 2-bit minimum signal swing on the changes from the DAC to give the STATUS signal a better signal-to-noise ratio. The second circuit in the STATUS/CONTROL port is the control port which consists of the control input latch and control output data latch. These latches are used to read and write signals that control different operations within the instrumentation including parallel port handshake signals, interrupt request levels, power control, $R_{cal}$ control, and other functions. The STATUS/CONTROL port has a single address decoder for both the status and the control ports. The first circuit to be discussed in detail will be the STATUS/CONTROL port.

### 3.3.4.2. Status and Control Port

The status and control port consists of two separate circuits. The status port is used to provide an analog output signal that corresponds to the current state of the ADAM system, and the control port is used to allow the processor to input and output control signals for various functions throughout the instrumentation system. The first circuit to be discussed is the status port.

#### 3.3.4.2.1. Status Port

Figure 160 is a schematic of the STATUS/CONTROL port. The status port consists of the data latch, digital-to-analog converter, and scaling circuitry. The status port is a byte wide write-only port located at address 800030 (hex). The most significant bit of the port is used to enable the telemetry interrupt and will be discussed under the telemetry port discussion (see Section 3.3.4.3). The remaining seven bits are used to report the status of the instrumentation.

The status port data latch, U3 (Figure 160), is a 74HC374 which is located on the data bus from D24-D31. This is the location required by the MC68020 microprocessor for 8-bit ports. The data on the data bus is latched into U3 on the rising edge of the control signal $\overline{\text{STATCS}}$. $\overline{\text{STATCS}}$ is generated by U2 when the correct combination of address lines (A11-A0), $\overline{\text{AS}}$, $\overline{\text{DS}}$, R/W, and PSCS (partial STATUS/CONTROL chip select) are present. PSCS is generated by U1 when the correct combination of A31-A12 is present.

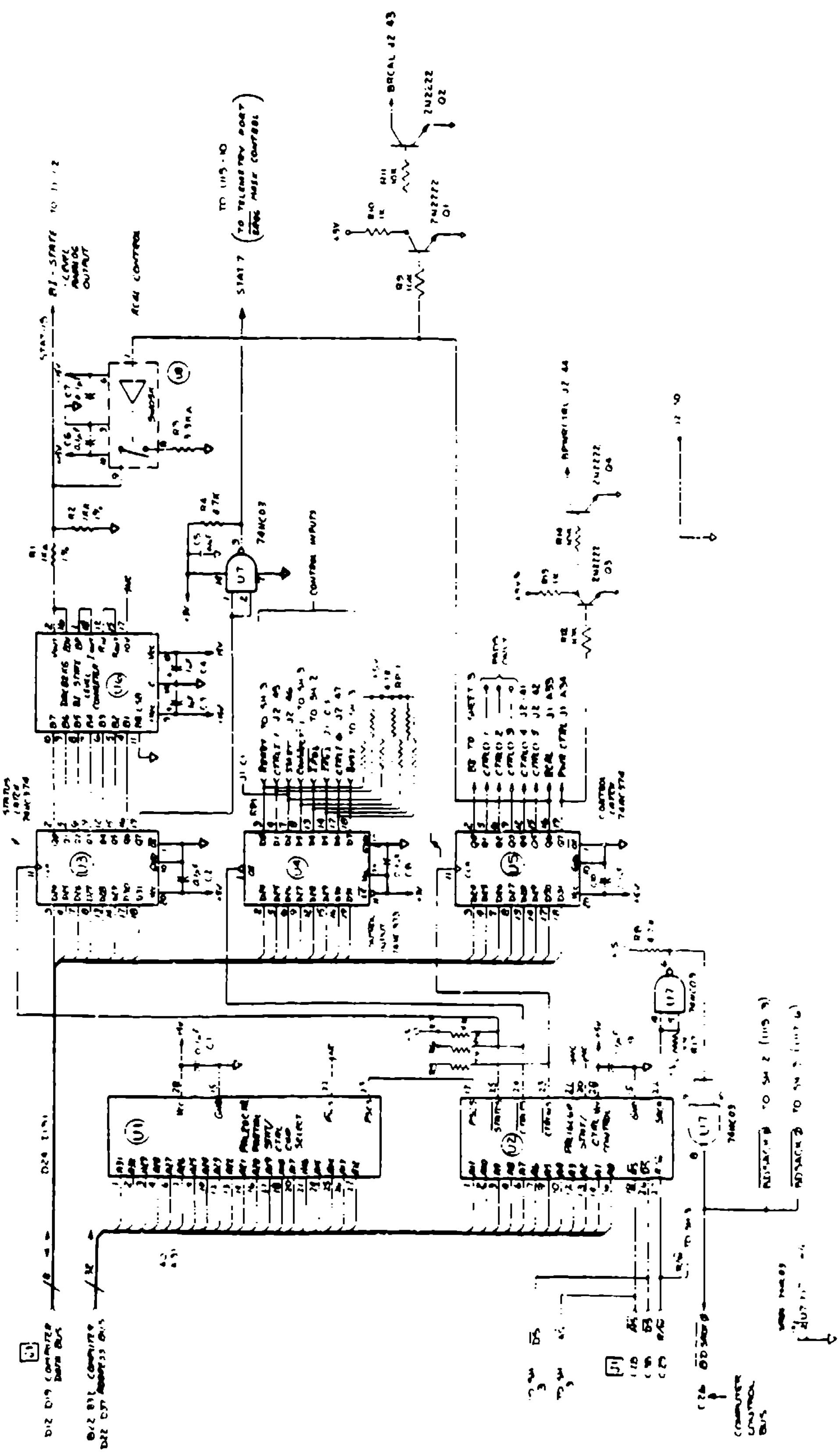


Figure 160. ADAM Status/Control Port Schematic

The outputs of the status latch, U3, are enabled at all times and drive the inputs to the DAC, U6. U6 is an 8-bit DAC that is used to convert the seven digital status lines into a single unique analog voltage based on the combination of inputs from the status latch. U6 is a Burr Brown DAC82KG digital-to-analog converter that is configured to provide a ±10V signal out. The voltage divider circuit of R1 and R2 divides the ±10V output voltage range into a ±5V output voltage range.

The last significant bit input (pin 11 of U6) is tied low, and the seven status latch outputs are connected to the seven most significant bits of U6. This combination will provide at least a 2-bit change in the output voltage level when a bit in the status latch changes states. This 2-bit minimum change in the analog output offers additional noise immunity to the status signal when it is digitized by the A/D board. Table 50 shows the output voltage levels of the DAC for each bit that is set in the status latch.

TABLE 50. STATUS PORT VOLTAGE ASSIGNMENTS

| Bit | Voltage Assignment* |
|---|---|
| 0 | Set - 5.00 Volts<br>Clear - 0.00 Volts |
| 1 | Set - 2.50 Volts<br>Clear - 0.00 Volts |
| 2 | Set - 1.25 Volts<br>Clear - -0.00 Volts |
| 3 | Set - 0.625 Volts<br>Clear - 0.00 |
| 4 | Set - 0.313 Volts<br>Clear - 0.00 Volts |
| 5 | Set - 0.156 Volts<br>Clear - 0.00 Volts |
| 6 | Set - 0.078 Volts<br>Clear - 0.00 |
| 7 | N/a |

*To determine the composite voltage, add voltages for each of the bits together and subtract the sum from 5.00 volts.

In order to provide a step change to the status line when the system is switched into the $R_{cal}$ mode, an FET switch (U8) and $R_{cal}$ resistor (R3) are provided at the status port. When the $R_{cal}$ control line is high, the FET switch is open and the status output is not affected. When the $R_{cal}$ control line is low, the FET switch is closed and the $R_{cal}$ resistor, R3, is switched is parallel with R2 of the output voltage divider and it causes a step change in the output voltage level. The step change will always be in a direction that brings it closer to ground potential since one side of R2 and R3 are both grounded. Table 51 outlines the current assignments for each bit of the status port.

TABLE 51. STATUS PORT BIT ASSIGNMENTS

| Bit | Functional Assignment |
|---|---|
| 0 | Set - Diagnostics to be Run or Running<br>Clear - Diagnostics Passed |
| 1 | Set - In $R_{cal}$ Mode<br>Clear - In Normal Signal Mode |
| 2 | Set - Memory Board Full<br>Clear - No Data in Memory |
| 3 | Set - Start Signal Has Been Received<br>Clear - Waiting for Valid Start Signal |
| 4 | Set - Storing Data in Memory<br>Clear - Waiting to or Finished Filling Memory |
| 5 | Set - System armed and Waiting for a Start<br>Clear - System in Pre or Post Test activities |
| 6 | Set - Not Defined<br>Clear - Not Defined |
| 7 | Set - Mask/IRQ6 Telemetry Interrupt<br>Clear - Enable /IRQ6 Telemetry Interrupt |

3.3.4.2.2. Control Port

The control port consists of two data latches--an input data latch and an output data latch. These latches are used by the processor to read and write different control inputs that affect the operation of the instrumentation.

The output data latch is U5, a 74HC374. The address of this latch is 800031, and it is a byte wide write-only port. The data from the data bus are latched into U5 on the rising edge of the control line $\overline{ACTRLWS}$. $\overline{CTRLWS}$ is generated by U2, a PAL16L6, when the correct combination of A11-A0, $\overline{AS}$, $\overline{DS}$, R/W, and PSCS is present. Table 52 lists the bit definitions for the control output data latch.

TABLE 52. CONTROL OUTPUT PORT BIT ASSIGNMENTS

| Bit | Functional Assignment |
|---|---|
| 0 | Set - Parallel Port in Input Mode<br>Clear - Parallel Port in Output Mode |
| 1 | Set - Not Defined<br>Clear - Not Defined |
| 2 | Set - Not Defined<br>Clear - Not Defined |
| 3 | Set - Not Defined<br>Clear - Not Defined |
| 4 | Set - Not Defined<br>Clear - Not Defined |
| 5 | Set - Not Defined<br>Clear - Not Defined |
| 6 | Set - Normal Signal Mode<br>Clear - $R_{cal}$ Calibration Mode |
| 7 | Set - Analog System Power Off<br>Clear - Analog System Power Off |

The $R_{cal}$ control line (bit 6) also drives the transistors Q1 and Q2 to provide a buffered open collector output, BRCAL, that may be used for external circuitry that requires ADAM to control its $R_{cal}$ status. The open collector circuit of Q3 and Q4, BPWR CTRL is provided to supply external circuitry the power control line. Both open collector circuit designs were provided to protect the ADAM circuitry from excessive current drain and provide the external circuitry with a high current drive capability.

The control input latch, U4, is a 74HC373. It is a byte wide port that is a read-only device at address 800031 (hex). The data on the inputs of U4 are active on the data bus when the control line $\overline{CTRLRS}$, control read asserted, is low. $\overline{CTRLRS}$ is asserted when the PAL16L6, U2, has the correct combination of A11-A0, $\overline{AS}$, $\overline{DS}$, R/W, and PSCS present. U4 is wired to act as a transparent latch which means that the outputs change with the inputs and the data are never "latched" into U4. Because U4 is a CMOS device that requires a voltage at each input to function properly, pullup resistors are provided for each input of U4 in cases where there is no signal present on the inputs. Table 53 lists the bit assignments for the control input port.

TABLE 53. CONTROL INPUT PORT BIT ASSIGNMENTS

| Bit | Functional Assignment |
|---|---|
| 0 | Set - Parallel Port Data Ready to be Read<br>Clear -Parallel Port Buffer Empty |
| 1 | Set - Not Defined<br>Clear - Not Defined |
| 2 | Set - Valid Start Signal<br>Clear - Start Signal Not active |
| 3 | Set - DRASS Not Connected<br>Clear - DRASS Connected and Ready |
| 4 | Set - /IRQ6 Telemetry Interrupt Not Pending<br>Clear - /IRQ6 Telemetry Interrupt Pending |
| 5 | Set - /IRQ2 MFP Interrupt Not Pending<br>Clear - /IRQ2 MFP Interrupt Pending |
| 6 | Set - Not Defined<br>Clear - Not Defined |
| 7 | Set - DRASS Busy (Buffer Full)<br>Clear - DRASS Not Busy (Buffer Empty) |

The STATUS/CONTROL port has one DSACK circuit. It is generated by the PAL16L6, U2, whenever the valid address 800030 or 800031 (hex) is on the address bus. Two NAND gates from a 74HC031C are used to provide an open collector driver for the BDSACK0 bus.

### 3.3.4.3. Telemetry Port

The telemetry port is used to provide a means to transmit the test data by wire or RF link to data acquisition equipment at the test facility. The telemetry data can be transmitted before, during, and after a test to allow remote monitoring and collection of the test data.

#### 3.3.4.3.1. Definition of Pulse Code Modulation Technique

The PCM techniques that are implemented in ADAM are the NRZ-L and the BIø-L techniques. The BIø-L signal can be derived from the NRZ-L signal, so this section will deal with the definition of the NRZ-L signal. A PCM signal is a set of binary data that is time multiplexed into a single serial bit stream. The binary data comes from the ADAM ADC and the telemetry interface would work in conjunction with the system microprocessor to gene rate the time multiplexed serial bit stream. The serial bit stream that is produced for NRZ-L data is one in which, if it is a logic low, then the telemetry signal is a logic low. If it is a logic high, then the telemetry signal is a logic high.

Figure 161 illustrates the format that the PCM signal follows for the telemetry interface. Each frame of the data stream will consist of 3 bytes of synchronization code, an 8-bit frame count, and the remaining bytes are the actual data for that frame. The synchronization code indicates the beginning of a new frame, and the frame count is immediately after the synchronization code. The data are organized in multiples of 4 bytes due to the design of the telemetry interface. The frame size is 4 bytes longer than the number of data samples.

#### 3.3.4.3.2. Functional Description

Figure 162 is a block diagram of the telemetry interface. The diagram shows the telemetry port consists of a input data latch, a parallel to serial converter, NRZ-L and BIø-L generators, timing and interface controller, and output low-pass filter.

The input data latch is 32 bits wide. When the parallel-to-serial converter completes the operation on its current set of data, the timing and interface controller circuitry issues a load command to the parallel-to-serial converter, and the data from the latches are loaded into the converter. The parallel-to-serial converter is a shift register that shifts the data out at a rate specified by the frequency of the bit clock. The telemetry format is generated in the NRZ-L and BIø-L generators, and the telemetry output is fed into the low-pass filter. The low-pass filter is a six pole bessel design which is used to limit the harmonic content of the telemetry signal being fed to the telemetry

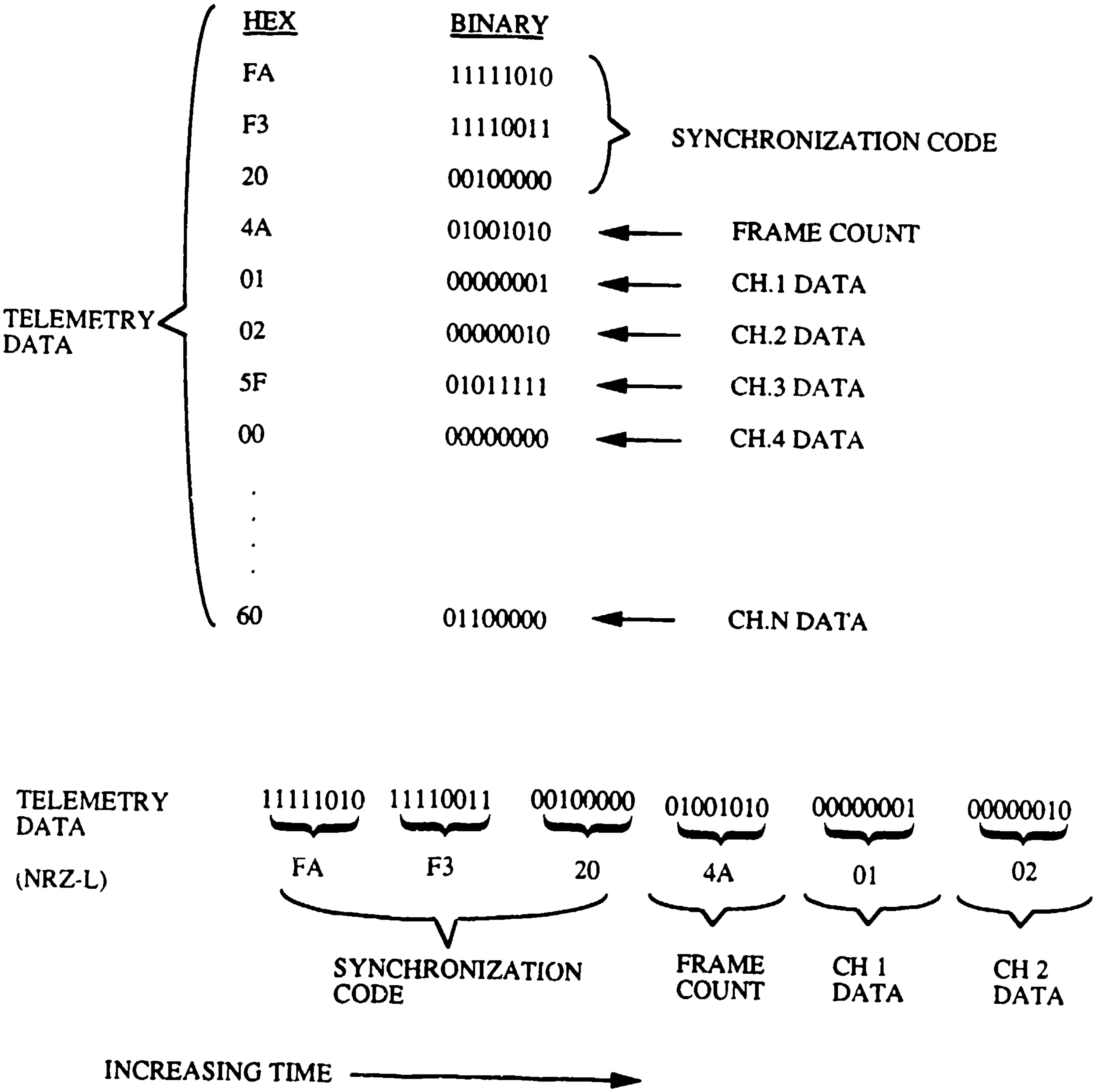



Figure 161. Data Format for Telemetry Output

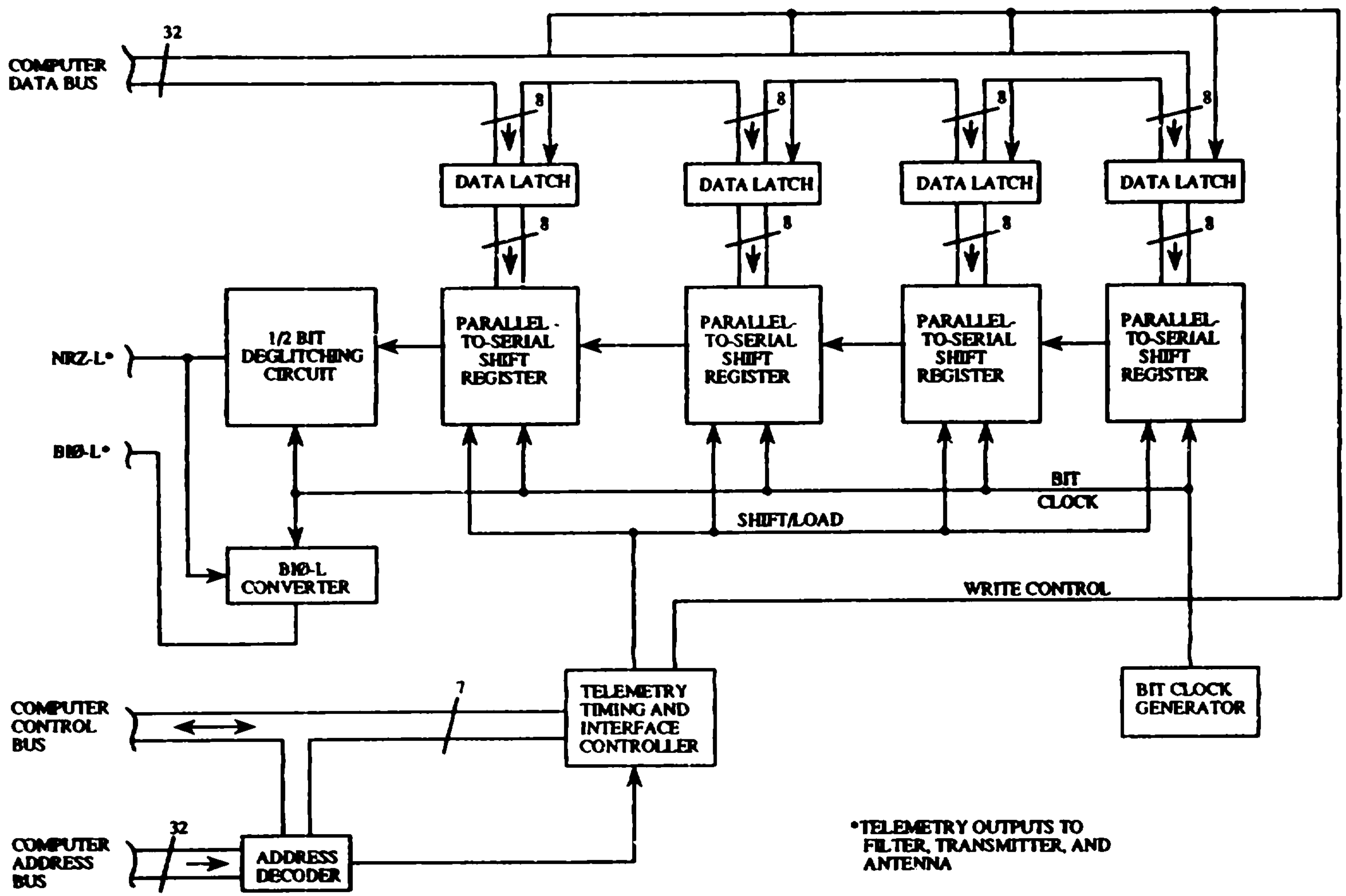



Figure 162. Telemetry Generator Block Diagram

transmitter. The timing and interface controller also produces a frame synchronization pulse that marks the start of a new frame. This pulse is available for use be external devices that may need this information.

Figure 163 shows the schematic of the telemetry circuit except for the frame synchronization pulse and the low-pass filter. The input data latches and parallel-to-serial shift registers are contained in the circuits U9-U12, which are 74HC589 devices. The input data are received from the data bus and stored in the input latch of U9-U12. The 74HC589 is an 8-bit parallel load shift register with an internal data latch. The data are latched on the rising edge of the telemetry port write chip select ($\overline{\text{TPWCS}}$). $\overline{\text{TPWCS}}$ is generated by the PAL U21, when the telemetry port partial address select (TPPADS) from the PAL U20, $\overline{\text{AS}}$, and A11-A0 are present in the correct combination. The telemetry port is a long word write-only port at address 800010 (hex). Data written to the telemetry port are stored in the input data latch until being loaded into the shift register. The shift register is asynchronously loaded when the S/L line is a logic low.

The S/L line is generated by the combination of the counters U14 and U16. U14 is a 74HC4040, a 12 stage binary counter, that is reset to zero when the shift registers are loaded with new data. When 32-bit clocks have passed, the Q6 output of U14 goes high. U16 is a 74HC4017, a decade counter, that is held inactive by the state the Q6 output of U14. The NAND gate of U15 inverts the level of Q6 to provide the proper logic level for the enable input of U16.

When Q6 of U14 goes high, the enable pin of U16 goes low and the counter, U16, is able to count. The counter is clocked by the 8 MHz system clock. The first rising edge of the system clock causes the Q1 output of U16 to go high. This line is the load/shift line. a NAND gate is connected as an inverter to this line to provide a shift/load line to U9-U12. The high on Q1 of U16 causes a low on the shift/load line and the data in the data latches is loaded into the shift registers. The second clock into U16 causes Q1 to be reset and Q2 to go high. The Q2 output of U16 is connected to the RESET inputs of U14 and U16. When Q2 goes high, the RESET lines of both U14 and U16 are asserted and Q2 is reset as well as U14-Q6, which disables the U16 ENABLE until 32-bit clocks have again passed. The entire operation takes less time than 1/2-bit clock period, which is the limiting factor in determining the maximum bit rate of the telemetry port.

The upper limit of the bit rate may be calculated as:

$$\text{Bit Clock Period (MAX)} = 2 * (\text{System Clock Period}) + 60 \text{ ns}$$

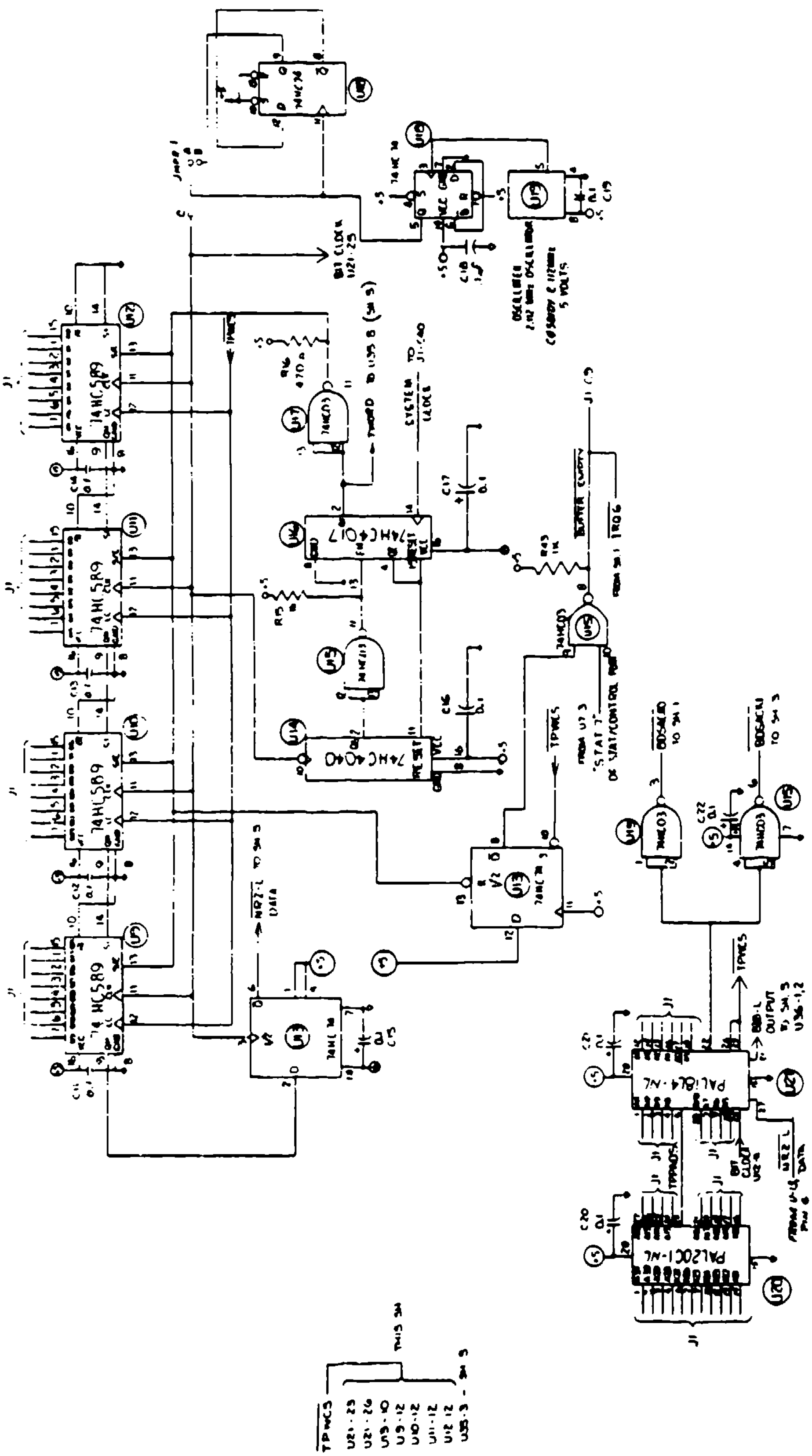


Figure 163. ADAM Telemetry Port Schematic

where the two system clock periods are from the two clock cycles required by U16, and the 60 nanoseconds is the time for the reset line of U16 to go high, reset the circuit, and go low. With an 8 MHz clock, the hardware limit of the bit clock is approximately 3.2 megabytes per second. This is a theoretical limit, and the actual bit rate is limited by the software overhead required to service the telemetry port.

The serial output of U9 (pin 9) is fed into the input of the "D" flip-flop of U13. This flip-flop is used to produce a stable NRZ-L output from the serial output of pin 9. This is required since the value of the data from U9-pin 9 may change during the load operation from the internal data latch. The flip-flop is used to provide inverted NRZ-L ($\overline{\text{NRZ-L}}$) data so that the input to the telemetry filter is the correct polarity.

The combination of the NAND gate from U15 and the other flip-flop from U13 generates the interrupts for the processor. The interrupt is generated at U13 pin 8 whenever the S/L line goes low. The interrupt request is passed on to the processor only when the STAT 7 line at pin 10 of U15 is high. This circuit is used as an interrupt enable so that the lower priority interrupts can be used more efficiently.

The bit clock generator consists of a crystal clock oscillator and a pair of "D" flip-flops. The clock oscillator is twice the required frequency of 1.056 MHz. The first flip-flop of U18 is connected as a divide-by-two unit to generate a 1.056 MHz, 50 percent duty cycle clock. This clock output is fed to jumper 1, pin A, and to the second flip-flop of U18. This flip-flop is connected as a divide-by-two device, and it generates a 528 KHz clock that goes to jumper 1, pin B. Jumper 1, pin C, is the bit clock leading to the shift registers. By selecting the jumper A-C the bit clock is 1.056 MHz; with jumper B-C, the bit clock is 528 KHz. This selection provides the option to use two different clock rates without having to change the clock oscillator.

The programmable logic devices, U20 and U21, are used to generate the $\overline{\text{TPWCS}}$ control signal, the BDSACK signals, and the BIØ-L telemetry data. The BIø-L data are generated from the EXCLUSIVE O-ring of the $\overline{\text{NRZ-L}}$ data and the bit clock.

Figure 64 shows the schematic of the frame synchronization pulse and the output low-pass filter circuitry. The frame synchronization pulse circuitry consists of programmable logic devices U34 and U35 and the driver transistors Q8 and Q9. The synch output of U35 is normally in a logic high state. The output goes low when the frame synchronization code FAF320 (hex) is written to the data latches of the shift registers, and the logic programmed into U35 resets the synch output to

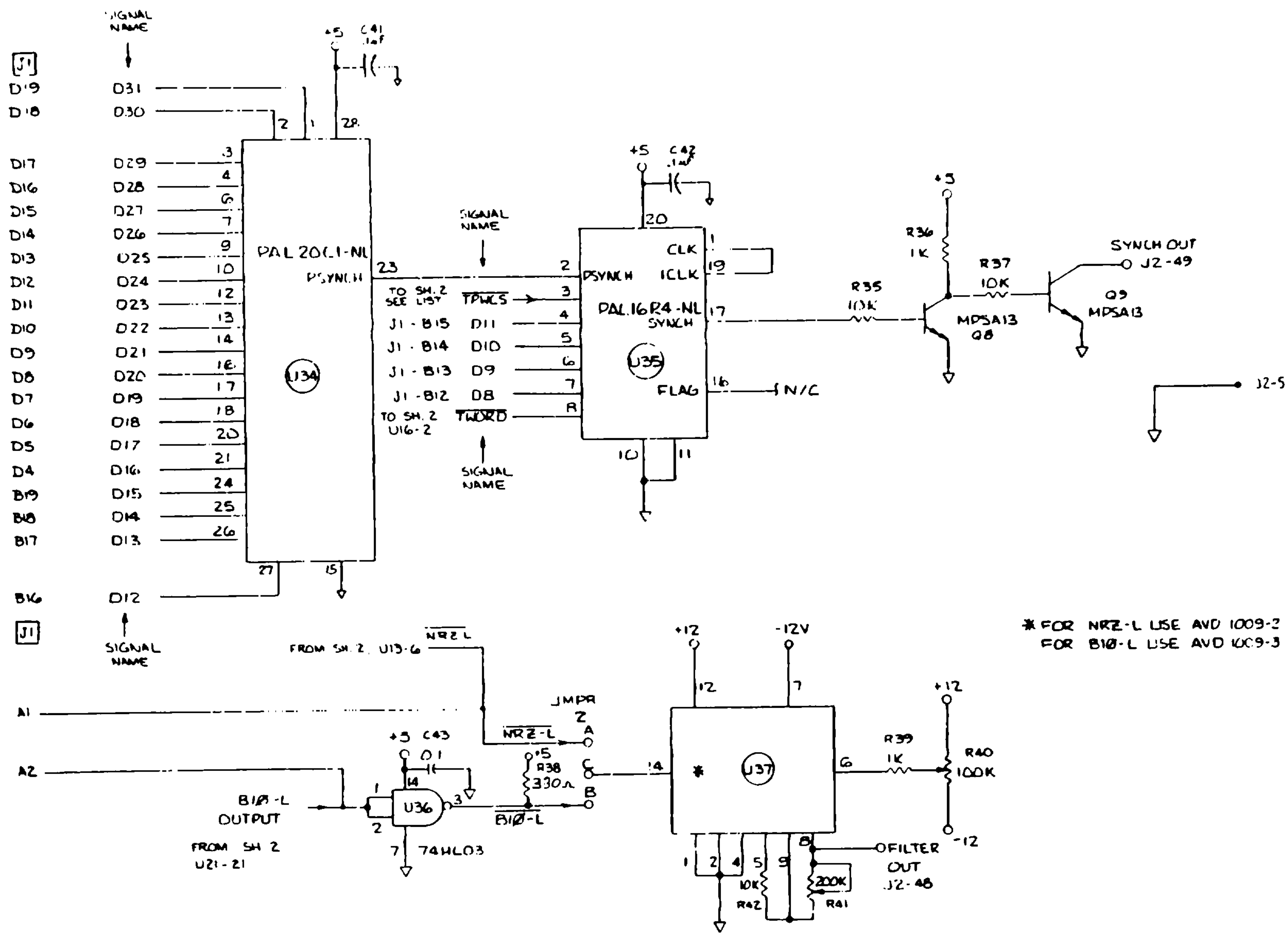



Figure 164. ADAM Telemetry Port Schematic

a high state when the synch code is loaded into the shift registers. Because the time that is required for the processor to load the synchronization code into the data latches is dependent on software considerations, the synchronization pulse is guaranteed low for a minimum of 2 microseconds only. The rising edge of the pulse is guaranteed to occur at the beginning of the frame synchronization code, but the falling edge is only guaranteed to occur at least 2 microseconds before the synchronization code is loaded into the shift registers (the S/L line goes low).

The output circuitry of the frame synchronization pulse consists of two transistors that act as an open collector buffer from the synchronization pulse. This buffer is provided for the protection of the circuitry of U35 in case the output is shorted to ground or a power supply. The open collector output was used to allow the external device using this signal to provides its own voltage level for the proper interface to that system.

The low-pass filter circuitry is also shown on Figure 164. The input to the low-pass filter, U37, is jumper selectable between NRZ-L and BIØ-L data using jumper 2. The low-pass filter is manufactured by the Aydin Vector Company specifically for use in PCM systems. Two filters are provided with the instrumentation. One has a cutoff frequency of 746.6 kHz and is used for NRZ-L data, and the second has a cutoff frequency of 1,493 MHz for use with BIØ-L data. The filter IC is placed in a socket on the digital I/O board to facilitate changing the filter circuit. Individual gain and offset adjustments are provided for the filter so that bipolar or unipolar data may be used on this system.

#### 3.3.4.4. Parallel Port

The parallel port is a 32-bit, high speed, bidirectional port intended to transfer the data stored in the SRAM array to the DRASS. A block diagram is shown in Figure 165. The 32-bit data lines are shared bidirectionally, and the transfer of data to and from the digital subsystem is handled by a common configuration of pairs of handshake lines. The input and output data latches are used as a buffer when the data are read or written by the processor, and the timing for the read and write instructions is managed by the interface and control logic. The interface and control logic also handles the operation of the handshake lines.

A schematic of the parallel port is shown in Figure 166. The input and output latches are contained within ICs U22-U25, which are 8-bit 74ACT547 bidirectional latch transceivers. One side of the transceiver is connected to the data bus and the other side is connected to the parallel port I/O lines. The output lines are enabled through a control line originating from the STATUS/CONTROL port,

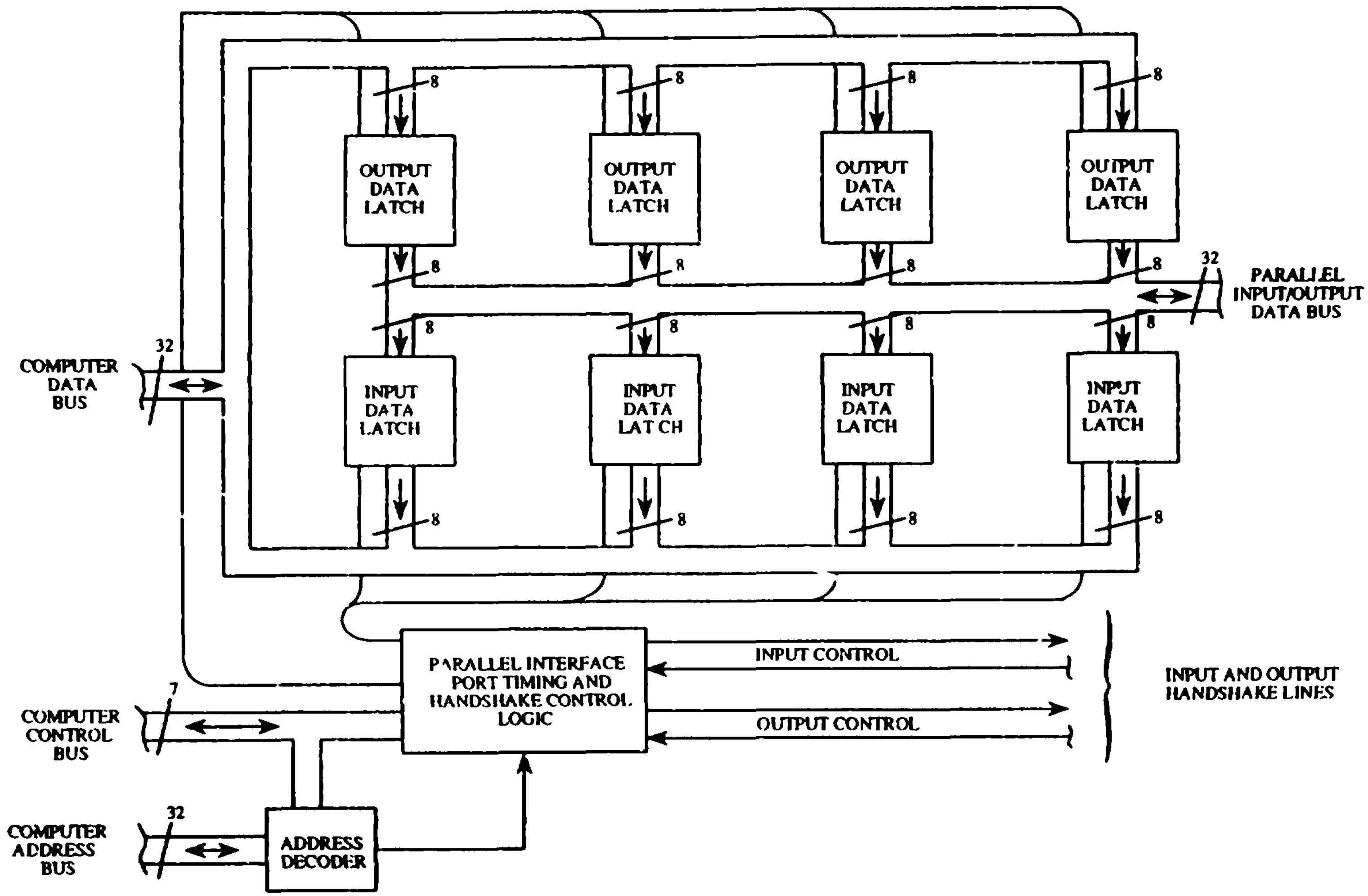



Figure 165. Parallel I/O Port Block Diagram

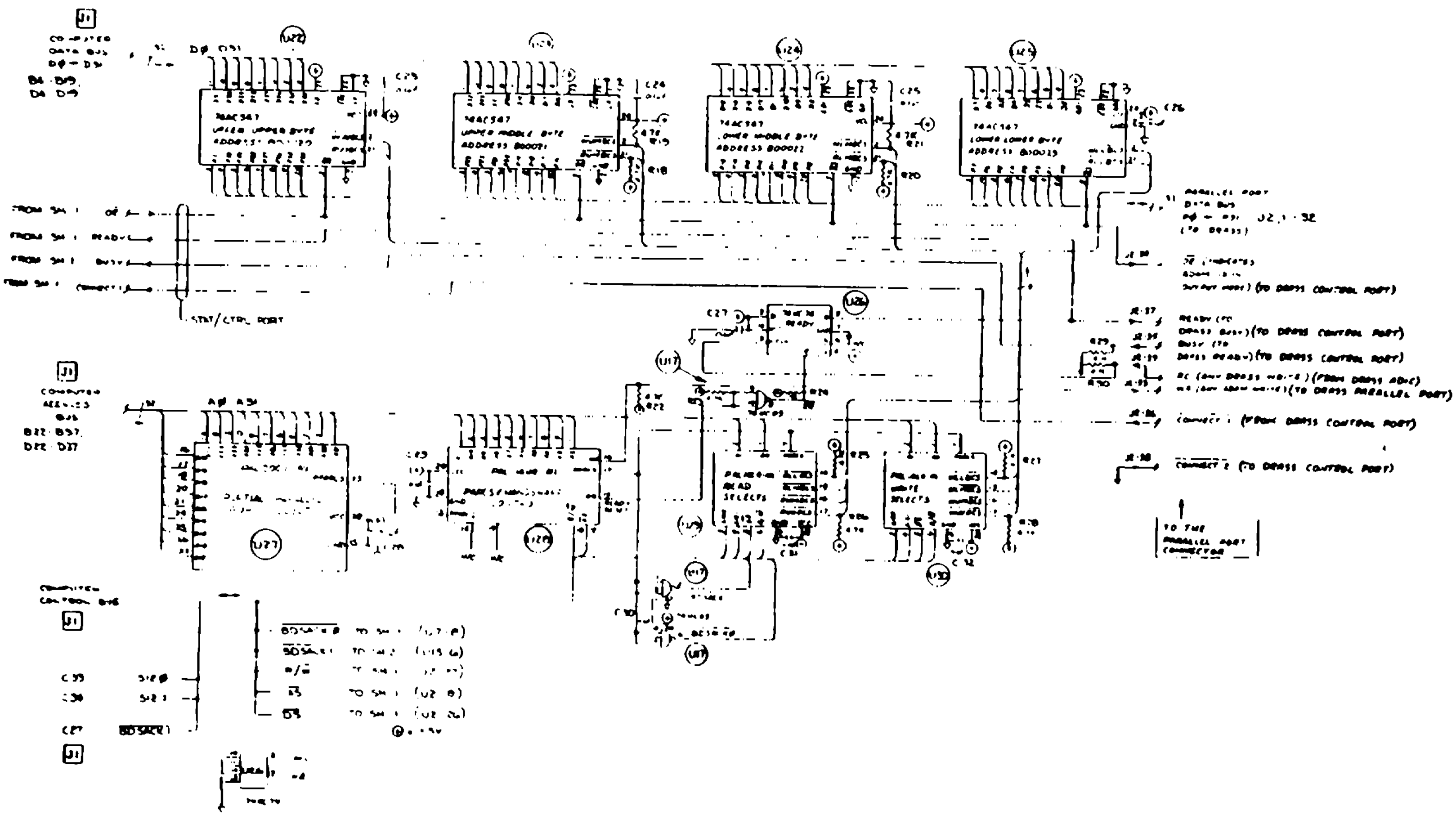


Figure 166. Digital Input/Output Board Parallel Port Schematic

the $\overline{OE}$ line. This line is also provided to external devices to indicate if ADAM is in the input or output mode. The parallel port is a 32-bit wide port located at address 800020 (hex). Full address select decoding in used in this scheme to allow 8-, 16-, and 32-bit words to be written and read by the parallel port. This scheme was selected to provide circuitry able to support systems with 8- and 16-bit parallel ports. The write chip select lines are connected to pin 2 of U22- U25 and are generated by the programmable logic device U30. The read select lines are generated by the programmable logic device U29. The read selects are connected to pin 21 of U22-U25.

The address decoding logic to contained within the two PAL devices, U27 and U28. U27 generates the parallel port partial chip select PPARCS and parallel port chip select PARCS is completely decoded in U28. The PARCS is used to generate the $\overline{BDSACKX}$ response using the open collector NAND gates of U17 connected to act as inverters.

In order to generate the correct read and write chip selects, U29 and U30 use S1Z0, S1Z1, $\overline{DS}$, R/W, and PARCS in their logic implementation. This decoding generates the individual byte chip selects required for the dynamic bus sizing features of the 68020.

The handshake control for the parallel port is managed through U28. This chip generates the ADAM write (WA) line that is used to indicate that ADAM has written data to the parallel port. The rising edge of this signal may be used by the external device to latch the data output by ADAM. The second handshake line generated by U28 is the ready reset (RR) line. This line is active whenever ADAM reads data from the parallel port. This line is inverted through a NAND gate and connected to the RESET line of the flip-flop U26.

This flip-flop is used to generate the ready handshake line. The ready line indicates that the ADAM parallel port is empty and data may be written to the parallel port. When data are written, the DRASS will generate a signal, called RC, that provides the same function as the ADAM WA signal described earlier. The rising edge of the RC signal will clock a high into the ready output of U26. This ready line is made available to the DRASS and to the ADAM STATUS/CONTROL port to indicate that data are present in the ADAM parallel port.

A similar circuit to the ready circuit just described provides the same information to ADAM regarding the DRASS. When ADAM writes data to the DRASS parallel port, the busy handshake line is used by ADAM to determine if the DRASS parallel port contains data. The busy line is connected directly to the ADAM STATUS/CONTROL port. By polling this line, ADAM is able to determine when the DRASS parallel port is ready for more data.

There are two more handshake lines on the parallel port. These lines are $\overline{\text{CONNECT1}}$ and $\overline{\text{CONNECT2}}$. $\overline{\text{CONNECT2}}$ is used by ADAM to indicate to the DRASS that ADAM is on-line and ready to send data. $\overline{\text{CONNECT2}}$ grounds the input on the DRASS which indicates that ADAM is on-line. $\overline{\text{CONNECT1}}$ is from the DRASS and is connected directly to the ADAM STAT/CTRL port. When $\overline{\text{CONNECT1}}$ is low, it indicates that the DRASS is connected and on-line. When $\overline{\text{CONNECT1}}$ is high, the DRASS off-line or not connected.

3.3.4.5. Miscellaneous Digital I/O Circuitry

Figure 167 shows the schematic of the remaining circuitry on the digital I/O board. It is the schematic of the power supply circuitry. This circuitry functions the same as the power supply circuitry described previously, so it will not be repeated. The additional voltage regulators, VR1 and VR2, are used to supply ±12 VDC to the telemetry port low pass filter. The telemetry filter operates at a ±12V supply instead of the ±15V supply used in the system.

Figure 168 shows the digital I/O board assembly. The connector J2 is a 51 pin connector that is used to provide access to the 32 data lines and handshake lines of the parallel port, and the control lines of the STATUS/CONTROL port. The connector J1 is used to mate the daughter board with the digital mother board. This connector provides access to all of the systems level signals available on the digital mother board.

The dimensions of this board are 6.95 inches by 4.425 inches. The power requirements of this board are approximately 800 mA for the +5 VDC supply, and 100 mA for each of the ±15 VDC supplies.

3.3.5. <u>Digital Mother Board</u>

The ADAM computer system consists of four daughter boards residing on the digital mother board. All four daughter boards share a common 160-line bus consisting mostly of computer address, data, and control signal lines. The bus also carries special system-level signals, as well as distributes power to the entire digital system. The only wire connections to the digital mother board come from the power distribution board. These are the plus and minus source lines and the system ground return line. All other digital system interconnections are made via individual daughter board connectors. The digital mother board is situated in the ADAM viscera instrumentation box so that the four digital daughter boards are physically oriented vertically with the connectors toward the front of the chest so that $G_x$ (eyeballs out) forces the pins into the sockets. The back

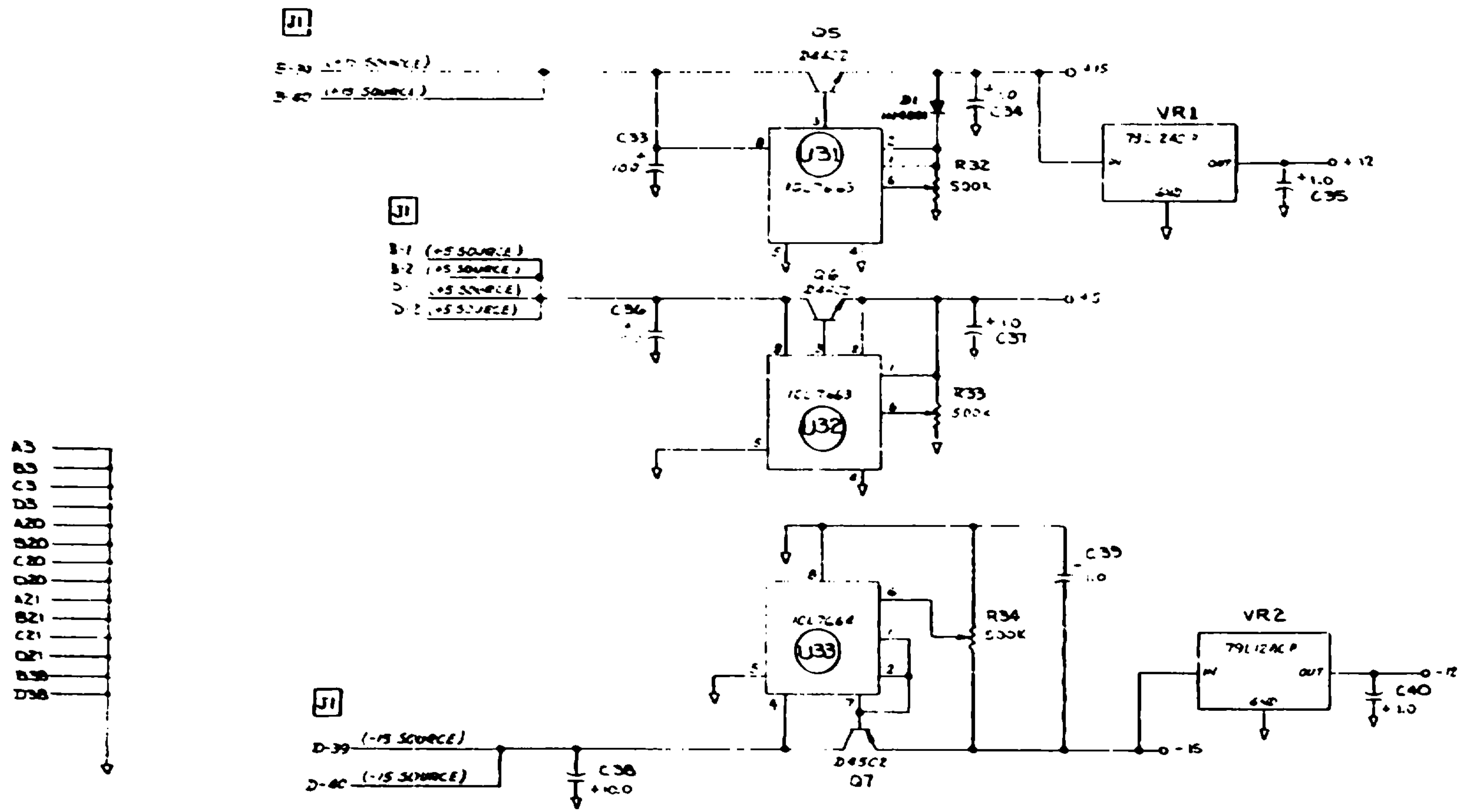



Figure 167. Digital Input/Output Board Power Supply Schematic

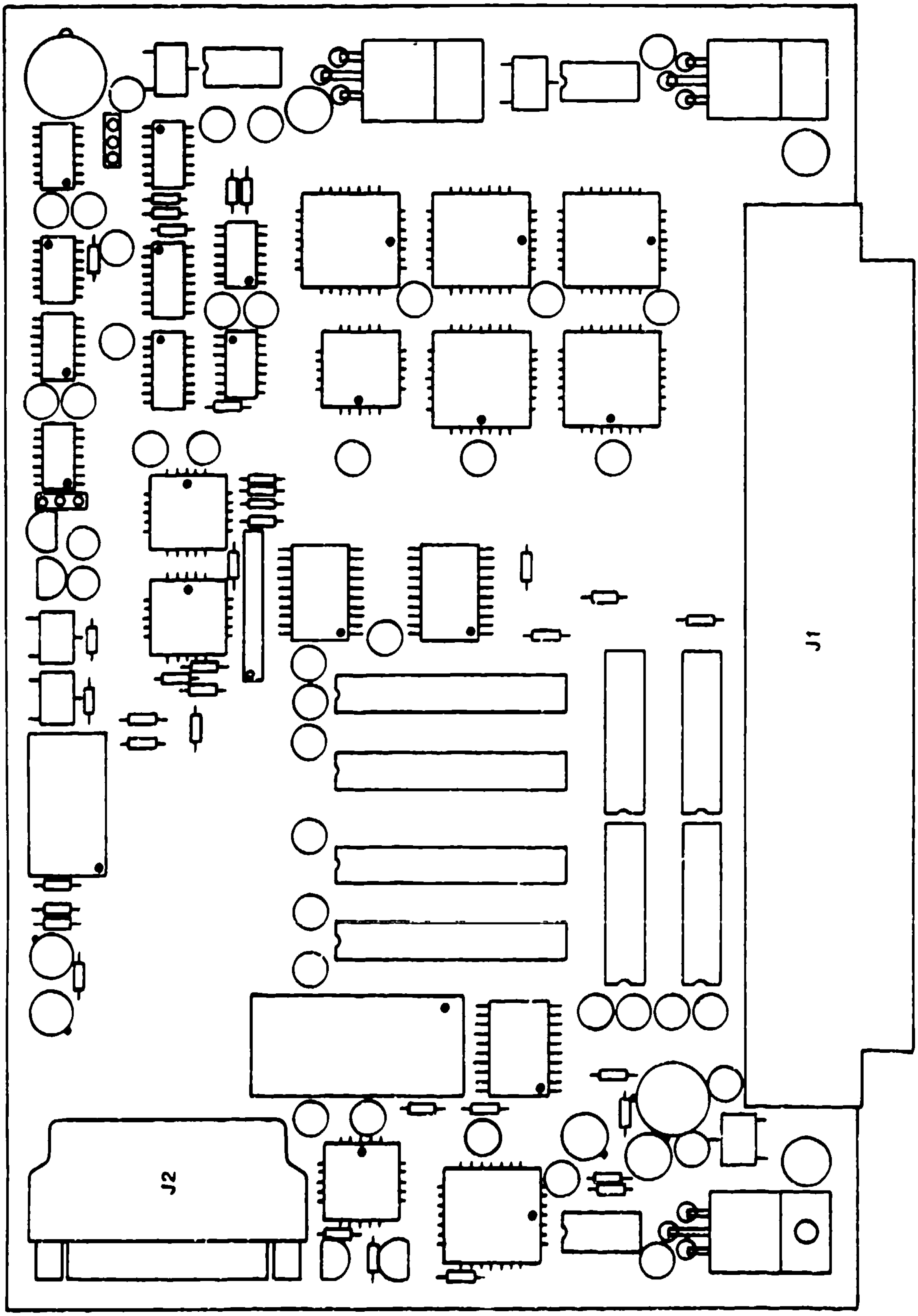



Figure 168. Digital I/O Circuit Card Assembly

plate of the instrumentation box can be removed so that the digital daughter boards can be removed or extended without removing the digital mother board. (Also see Section 3.8.2 as reference.)

Table 54 presents the digital system connector list which is identical for all four 160 pin connectors on the digital mother board. Signals in rows B, C, and D, between pins 4 through 19 and pins 22 through 37 are computer signals. The designations of D** in rows B and D between pins 4 and 19 are the 32 data lines, where ** is the data line number. The designation of A** in rows B and D between pins 22 through 37 are the 32 address lines, where ** is the address line number. Other lines in the previously defined computer region of the digital mother board are computer control lines. The remainder of the mother board signals are power lines and system control and status lines.

Table 55 presents a signal list for the digital mother board. This list identifies users of signals on the digital mother board by signal name, indicating for each daughter board whether they are an output source (O), a user input only (I), or a bidirectional user using both input and output activities (B).

## 3.4. POWER DISTRIBUTION SYSTEM

The power distribution system within the ADAM is intended to provide power to each of the the mother boards and the telemetry transmitter. The power distribution system components include the internal battery assemblies, external field power supply, and power distribution board. The manikin is capable of operating from the internal batteries or the external field power supply. If both the internal batteries and the field power supply are in use, then the power distribution board provides the circuitry to automatically switch from the batteries to the field power supply to conserve the battery strength. This section discusses the internal battery design, the external field power supply, and the power distribution board.

### 3.4.1. Internal Batteries

The internal battery system is used to power the ADAM instrumentation when external power is not available. There are four separate locations in the manikin where the cells have been located. The batteries are located in the right and left legs, the lumbar region of the buttocks, and the bottom of the viscera. The cells are connected in three series configurations.

## TABLE 54. DIGITAL SYSTEM CONNECTOR LIST

| Pin # | Row A | Row B | Row C | Row D |
|---|---|---|---|---|
| 1 | NRZ-L OUT | +5 SOURCE | START | +5 SOURCE |
| 2 | BIØ-L OUT | +5 SOURCE | STATUS | +5 SOURCE |
| 3 | GROUND | GROUND | GROUND | GROUND |
| 4 | | D00 | /IRQ1 | D16 |
| 5 | | D01 | /IRQ2 | D17 |
| 6 | | D02 | /IRQ3 | D18 |
| 7 | | D03 | /IRQ4 | D19 |
| 8 | | D04 | /IRQ5 | D20 |
| 9 | | D05 | /IRQ6 | D21 |
| 10 | | D06 | /IRQ7 | D22 |
| 11 | | D07 | /BR | D23 |
| 12 | | D08 | /BG | D24 |
| 13 | | D09 | /BGACK | D25 |
| 14 | | D10 | /BERR | D26 |
| 15 | | D11 | /RESET | D27 |
| 16 | | D12 | /HALT | D28 |
| 17 | | D13 | /CDIS | D29 |
| 18 | | D14 | /IPEND | D30 |
| 19 | | D15 | /RMC | D31 |
| 20 | GROUND | GROUND | GROUND | GROUND |
| 21 | GROUND | GROUND | GROUND | GROUND |
| 22 | | A00 | /BDL1 | A16 |
| 23 | | A01 | /BDL2 | A17 |
| 24 | | A02 | /BDL3 | A18 |
| 25 | | A03 | /BDL4 | A19 |
| 26 | | A04 | /BDSACK0 | A20 |
| 27 | | A05 | /BDSACK1 | A21 |
| 28 | | A06 | /AS | A22 |
| 29 | FILTER CLK1 | A07 | R/W | A23 |
| 30 | FILTER CLK2 | A08 | /ECS | A24 |
| 31 | FILTER CLK3 | A09 | /OCS | A25 |
| 32 | FILTER CLK4 | A10 | /DBEN | A26 |
| 33 | RCAL CONTROL | A11 | SIZ0 | A27 |
| 34 | PWR CONTROL | A12 | SIZ1 | A28 |
| 35 | | A13 | FC0 | A29 |
| 36 | | A14 | FC1 | A30 |
| 37 | | A15 | FC2 | A31 |
| 38 | | GROUND | /DS | GROUND |
| 39 | | +15 SOURCE | -15 SOURCE | |
| 40 | TEMPERATURE | +15 SOURCE | CLK | -15 SOURCE |

TABLE 55. DIGITAL MOTHER BOARD SIGNAL LIST

| Signal | J1:Row-Pin | CPU Board | Memory Board | I/O Board | ADC Board |
|---|---|---|---|---|---|
| A00 | B22 | 0 | I | I | I |
| A01 | B23 | 0 | I | I | I |
| A02 | B24 | 0 | I | I | I |
| A03 | B25 | 0 | I | I | I |
| A04 | B26 | 0 | I | I | I |
| A05 | B27 | 0 | I | I | I |
| A06 | B28 | 0 | I | I | I |
| A07 | B29 | 0 | I | I | I |
| A08 | B30 | 0 | I | I | I |
| A09 | B31 | 0 | I | I | I |
| A10 | B32 | 0 | I | I | I |
| A11 | B33 | 0 | I | I | I |
| A12 | B34 | 0 | I | I | I |
| A13 | B35 | 0 | I | I | I |
| A14 | B36 | 0 | I | I | I |
| A15 | B37 | 0 | I | I | I |
| A16 | D22 | 0 | I | I | I |
| A17 | D23 | 0 | I | I | I |
| A18 | D24 | 0 | I | I | I |
| A19 | D25 | 0 | I | I | I |
| A20 | D26 | 0 | I | I | I |
| A21 | D27 | 0 | I | I | I |
| A22 | D28 | 0 | I | I | I |
| A23 | D29 | 0 | I | I | I |
| A24 | D30 | 0 | I | I | I |
| A26 | D32 | 0 | I | I | I |
| A27 | D33 | 0 | I | I | I |
| A28 | D34 | 0 | I | I | I |
| A29 | D35 | 0 | I | I | I |
| A30 | D36 | 0 | I | I | I |
| A31 | D37 | 0 | I | I | I |
| /AS | C28 | 0 | I | I | I |
| /BDL1 | C22 | I | 0 | | |
| /BDL2 | C23 | I | | | |
| /BDL3 | C24 | I | | | |
| /BDSACK0 | C26 | I | | 0 | 0 |
| /BDSACK1 | C27 | I | 0 | 0 | 0 |
| /BERR | C14 | I | | | |
| /BG | C12 | 0 | | | |
| /BGACK | C13 | I | | | |
| BIØ-L 0UT | A2 | 0 | | | |
| /BR | C11 | I | | | |
| /CDIS | C17 | I | | | |
| CLK | C40 | 0 | I | | |
| D00 | B4 | B | B | B | B |
| D02 | B6 | B | B | B | B |
| D03 | B7 | B | B | B | B |

TABLE 55. DIGITAL MOTHER BOARD SIGNAL LIST (continued)

| Signal | J1:Row-Pin | CPU Board | Memory Board | I/O Board | ADC Board |
|---|---|---|---|---|---|
| D04 | B8 | B | B | B | B |
| D05 | B9 | B | B | B | B |
| D06 | B10 | B | B | B | B |
| D07 | B11 | B | B | B | B |
| D08 | B12 | B | B | B | B |
| D09 | B13 | B | B | B | B |
| D10 | B14 | B | B | B | B |
| D11 | B15 | B | B | B | B |
| D12 | B16 | B | B | B | B |
| D13 | B17 | B | B | B | B |
| D14 | B18 | B | B | B | B |
| D15 | B19 | B | B | B | B |
| D16 | D4 | B | B | B | B |
| D17 | D5 | B | B | B | B |
| D18 | D6 | B | B | B | B |
| D19 | D7 | B | B | B | B |
| D20 | D8 | B | B | B | B |
| D21 | D9 | B | B | B | B |
| D22 | D10 | B | B | B | B |
| D23 | D11 | B | B | B | B |
| D25 | D13 | B | B | B | B |
| D26 | D14 | B | B | B | B |
| D27 | D15 | B | B | B | B |
| D28 | D16 | B | B | B | B |
| D29 | D17 | B | B | B | B |
| D30 | D18 | B | B | B | B |
| /DBEN | C32 | 0 | | | |
| /DS | C38 | 0 | I | I | I |
| /ECS | C30 | 0 | I | | |
| FCO | C35 | 0 | | | |
| FC1 | C36 | 0 | | | |
| FC2 | C37 | 0 | | | |
| FILTER CLK1 | A29 | 0 | | | I |
| FILTER CLK2 | A30 | 0 | | | I |
| FILTER CLK3 | A31 | 0 | | | I |
| FILTER CLK4 | A32 | 0 | | | I |
| /HALT | C16 | B | | | |
| /IPEND | C18 | 0 | | | |
| /IRQ1 | C4 | I | | | |
| /IRQ2 | C5 | 0 | | I | |
| /IRQ3 | C7 | I | | | |
| /IRQ6 | C9 | I | | 0 | |
| /IRQ7 | C10 | I | | | |
| NRZ-L 0UT | A1 | 0 | | | |
| /OCS | C31 | 0 | | | |
| PWR CONTROL | A34 | | | 0 | I |
| $R_{CAL}$ CONTROL | A33 | | | 0 | I |
| /RESET | C15 | B | | I | |
| /RMC | C29 | 0 | | | |

TABLE 55. DIGITAL MOTHER BOARD SIGNAL LIST (continued)

| Signal | J1:Row-Pin | CPU Board | Memory Board | I/O Board | ADC Board |
|---|---|---|---|---|---|
| R/W | C29 | 0 | I | I | I |
| SIZ0 | C33 | 0 | I | I | I |
| SIZ1 | C34 | 0 | I | I | I |
| START | C1 | | | 0 | |
| STATUS | C2 | | | 0 | I |
| TEMPERATURE | A40 | | | 0 | |

*0 = Output Only, I = Input Only, B = Bidirectional
Ground: A3, B3, C3, D3
A20, B20, C20, D20
A21, B21, C21, D21
B38, D38
+5 Source: B1, D1, B2, D2
+15 Source: B39, B40
-15 Source: D39, D40

The selection of the type of cell for use in the instrumentation system was dictated by the line current requirements of the instrumentation and the space constraints within the manikin. a survey of available battery technologies indicated that lithium cells were the only cells available to meet the space requirements and sufficient current to operate the instrumentation system. The lithium cells offered a 3 volt cell voltage and the capability to source better than 3 amperes (A). These two properties combined to reduce the number of cells required (nickel cadmium batteries, in contrast, have a 1.2 volt cell voltage and source about 1 to 2 A) and help to alleviate the space problem.

The lithium cells selected are manufactured by Electrochem Industries and are not rechargeable. Two different size cells have been used in the manikin. The D-size lithium cell is a 13 ampere-hour (Ah) cell with a maximum current of 4 A. The D cell is 1.32 inches in diameter and 2.33 inches long. The DD-size cell is a 26-Ah cell with a maximum current of 7 A. It is 1.32 inches in diameter and 4.38 inches long. Both cells are equipped with an internal fuse that will open if the cell current exceeds the maximum discharge current of the cell. This fuse protects the cell from excessive discharge.

Figure 169 is a schematic of the pelvis battery. The cell B1 is located on the bottom of the viscera, and cells B2 and B3 are located in the lumbar area of the buttocks. This battery is connected to ADAM system ground on one end, and the +5 source voltage (labelled "BDIG") is the output of B1. BDIG is nominally 9 VDC and must not drop below 7 VDC.

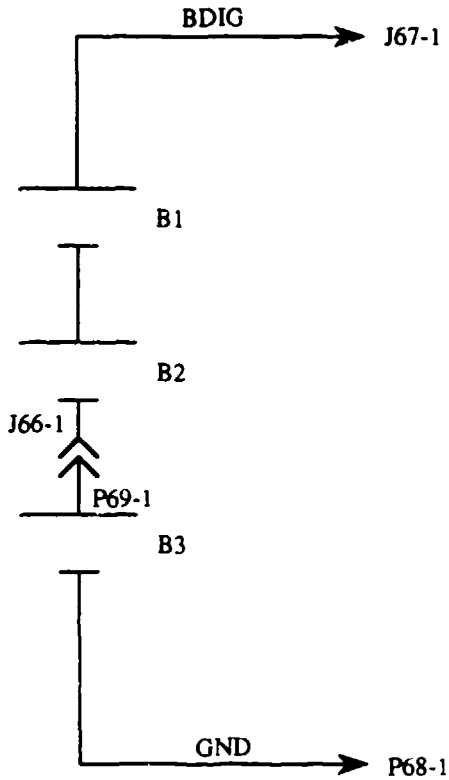



Figure 169. ADAM Pelvis Battery Schematic

Figure 170 is a schematic of the left leg battery. The six D cells, B1-B6, provide the positive voltage BVCC which is nominally 18 VDC and must not fall below 16 VDC. BVCC goes directly to the power distribution board and provides power for +VCC, XMIT, and +EXC at an amperage level of approximately 2.9 A.

Figure 171 is a schematic of the right leg battery. The cells, B1-B6, are D-size lithium cells. The positive terminal of B1 is connected to system ground, and the remaining cells are connected in series to create the negative supply required to operate the system. The negative voltage (-BVCC) provided by the right leg is nominally -18 VDC and must not rise above -16 VDC. The voltage -BVCC goes directly to the power distribution board and provides power for -VCC and -EXC at an amperage level of approximately 2.1 A.

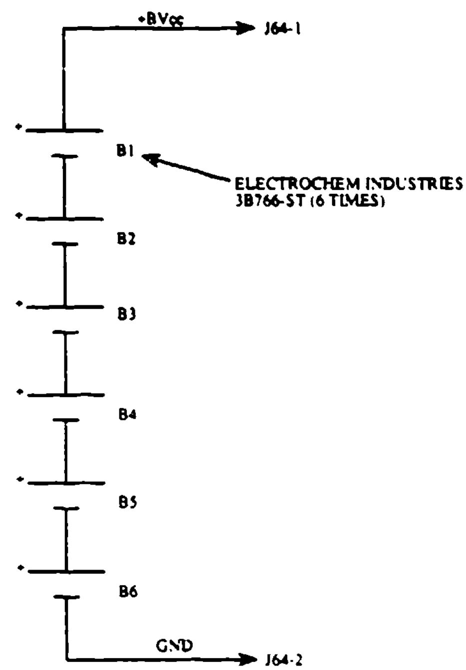



Figure 170. ADAM Left Leg Battery Schematic

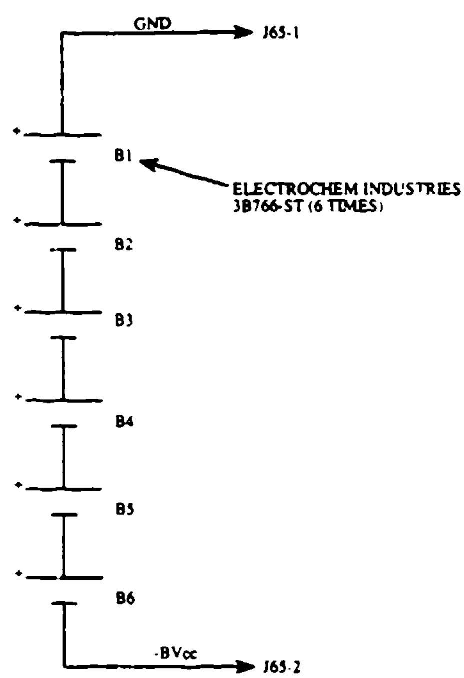



Figure 171. ADAM Right Leg Battery Schematic

The interconnections between the battery packs is made by an interconnect cable that is routed from each leg through the pelvis and to a mating connector on the power distribution board. The wire used for all of the battery interconnections is 16 gauge wire to reduce the voltage drops due to the high currents it must carry.

The capacity of the cells selected should provide full power operation of the instrumentation for 1 to 2 hours. If the analog circuitry has been shut down, the batteries supplying the digital 5 VDC source voltage should last even longer. This allows the transfer of data out of the manikin several hours after the data has been collected.

### 3.4.2. Field Power Supply

The field power supply is used to power the instrumentation system when batteries are not used or to prevent discharge of the batteries prior to a test. The field power supply (FPS) is designed to operate on 115 VAC, its own internal batteries, or an external 28 VDC supply. The FPS provides all of the voltages required to operate the instrumentation using DC to DC converters to develop the proper voltages for the manikin.

Figure 172 is a schematic of the FPS. The 115 VAC power is provide at connector J1 and fed through the EMI filter F2. The main power switch, S2, controls the 115 VAC power and the battery power at batteries B1 and B2. The 115 VAC is used to power the switching power supply, PS1.

PS1 is a 28 V, 266 watt power supply used to supply the required 28 V excitation for the DC to DC converters PS2-PS5; the output of PS1 is connected to the plugs BP-3. To operate the FPS on 115 VAC, the plug BP-3 must be jumpered to provide 28 VDC to BP2. If the FPS was operated with an external 28 VDC supply, the power is supplied to the FPS at BP-2.

From BP-2, there is an EMI filter for noise suppression, and then the voltage is supplied to a dual rate, constant voltage punch/float charger to keep the internal batteries charged. This charger uses the circuitry out lined in detail A to provide a high charge rate when the batteries are low and automatically switch to a trickle charge when the batteries are almost completely charged.

The diodes D2 and D3 make up a matrix to prevent the external 28 VDC from feeding into the batteries and the battery voltage from feeding back into the battery charger.

The 28 volts powers the DC to DC converters, PS3-PS5, through the solid state relay, K1. The switch S1 will activate this relay and power PS3-PS5 or BP-1 provides a hook-up for a remote activation of the FPS.

PS2-PS5 are 100 watt DC to DC converters that are used to generate the necessary voltages for the instrumentation system. The resistors R14, R17, and R19 are used to adjust the voltage out

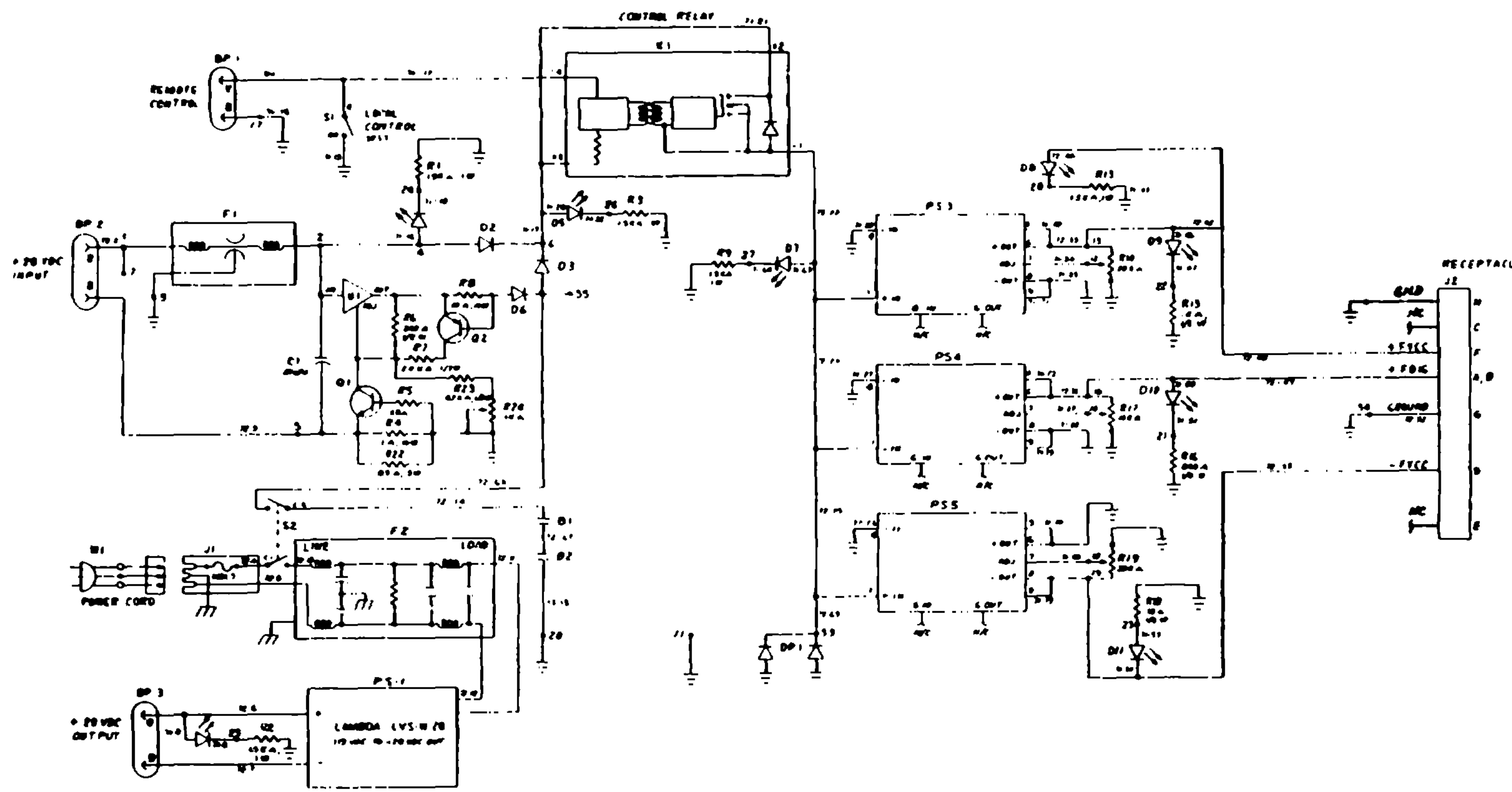


Figure 172. Field Power Supply Schematic

of the supplies. PS4 provides the +5 source (FDIG) for the digital circuitry. PS3 provides the +15 source voltage (FVCC), and PS5 generates the -15 source voltage (-FVCC).

The FPS circuitry is mounted in a stainless steel NEMA 12 weathertight enclosure to prevent dust and foreign material from attacking the circuitry. Stainless steel was chosen to limit the effects of any corrosion on the case.

The power supply can operate the ADAM for greater than 30 minutes on its internal batteries, and the charger will replenish the charge on the batteries within 24 hours. Whenever the FPS supplies power to the manikin, the manikin internal batteries are automatically switched out of the circuit by the power distribution board.

### 3.4.3. Power Distribution Board

The purpose of the power distribution board is to provide power to the telemetry transmitter, digital mother board, and analog mother board from either the internal batteries or the FPS. The power distribution board is designed to automatically channel the power from the FPS to the manikin whenever FPS power is present. This is true whether the internal batteries are installed in the manikin or not. If manikin batteries are installed in the manikin and the main battery chest connector is connected, then the power distribution board will route this power to the instrumentation system only when the FPS voltage are not present. The power distribution board has a magnetic latching relay to turn on or off total manikin power for safety reasons and an electronic switch to turn the telemetry transmitter power on or off. This circuitry was included to conserve battery power once the test data have been collected.

Figure 173 is the schematic for the power distribution board. The diode switching matrix of CR5 and CR6 is used to switch the BDIG and FDIG voltages. The maximum voltage at the BDIG terminal is 11.6 volts with no load on the batteries, and FDIG is 12 VDC. Whenever FDIG is present, CR5 is reverse biased and the FPS supplies the source voltage to U1 and K2. If the FDIG voltage is missing, then the batteries will supply the source voltage.

A wire jumper is used in place of K1 (used in remote turn-on configuration only) to apply power to the 5V regulator (U1) which will turn ont he solid state relays K2 through K4. If both BDIG and FDIG are missing, then all power to the analog and digital boards will be cut off. With K2 switched on, the output of the relay goes through a power resistor, R1, to reduce the DIG voltage by approximately 0.75 VDC prior to the voltage being applied to the digital mother board.

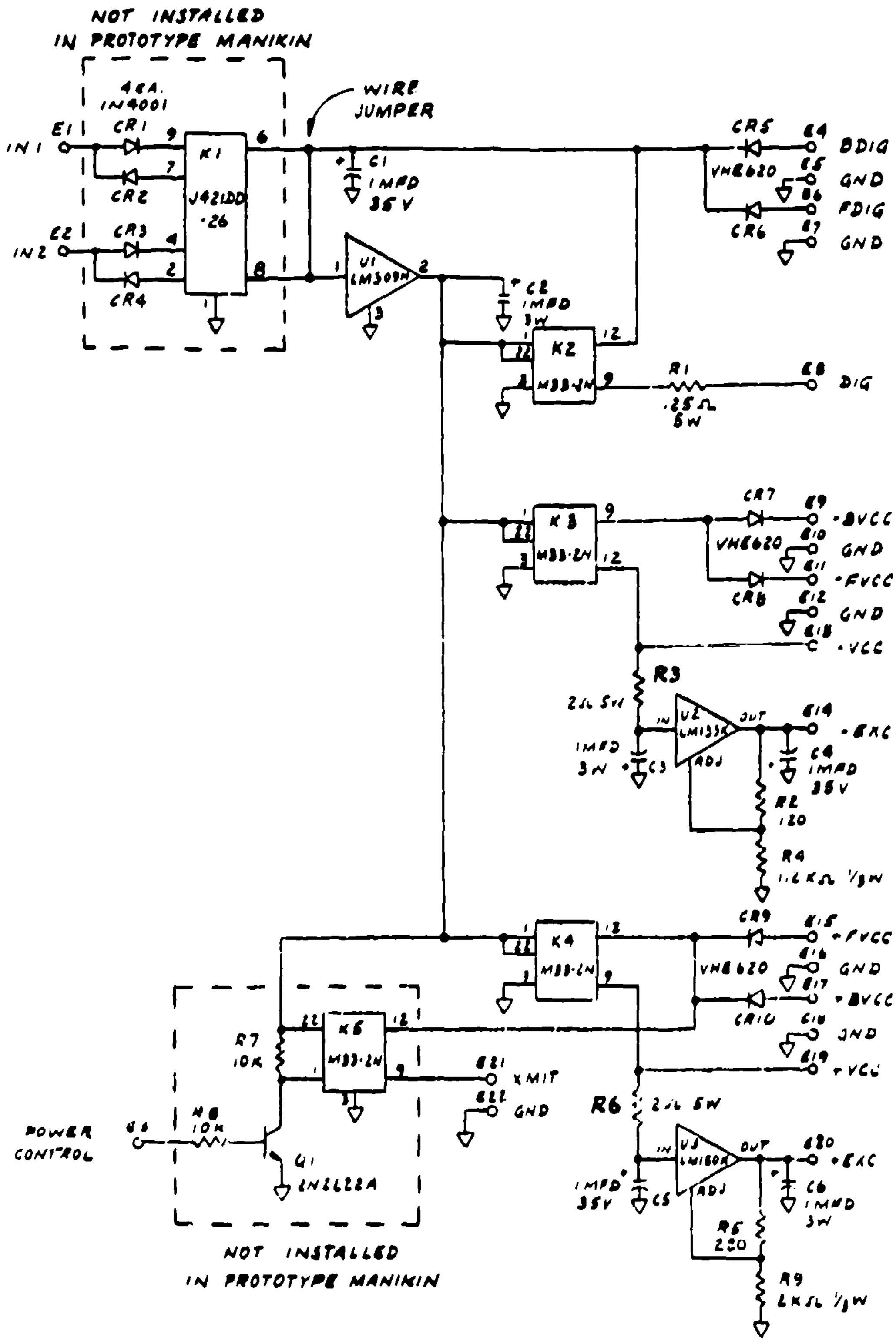



Figure 173. ADAM Power Distribution Board Schematic

The diodes CR7 and CR8 are used in a switching matrix for -BVCC and -FVCC. If -FVCC is present, then CR7 is reverse biased and the batteries are not used. If -FVCC is removed, then CR7 will be forward biased and the batteries will be supplying power to the switched input of K3. The relay (K3) output supplies the -VCC voltage which goes to the digital mother boards and to the voltage regulator, U2, through a power resistor, R3. The power resistor, R3, is a dropping resistor to reduce the voltage entering the regulator U2 by approximately 2 to 3 volts. The output of UE supplies the -EXC voltage (-12.6 VDC) and it is applied to the analog mother board.

The voltages +FVCC and +BVCC go through a switching diode matrix of CR9 and CR10. If +FVCC is present, then power from the battery is not consumed because CR10 will be reverse biased. Source voltage coming out of the diode matrix goes to the inputs of the solid state relays K4 and K5. The output of relay, K4, supplies the +VCC voltage, which goes to the digital mother board and to the power resistor, R6. Used as a dropping resistor, R6 drops the voltage 2-3 volts before entering the voltage regulator, U3, which outputs the +EXC voltage (+12.6 VDC) and goes to the analog mother board. The solid state relay K5 is used as a switch to turn the transmitter power on or off. For K5 to be activated (transmitter on), U1 must have a 5 VDC output and the power control line is a logic low. To turn the transmitter power off, the power control line goes to a logic high thereby turning Q1 on, which lowers the logic high on pin 1 of K5 to near ground potential. This shuts K5 off. The output of K5 is the XMIT voltage that goes to the transmitter.

The power distribution board will manage the power switching tasks between the manikin batteries and the FPS as long as the FDIG or BDIG voltage is present. If the FDIG voltage is absent, the LED for FDIG on the FPS will not light. However, if the other FPS voltages are present, the manikin will be powered by both the BDIG batteries and the remaining FPS voltages. Therefore, the manikin should not be allowed to operate with the FDIG voltages absent because to do so will shorten the life of the BDIG batteries.

## 3.5. ADAM RESIDENT SOFTWARE

### 3.5.1. Introduction

The purpose of this section is to present a description of the software menu trees and task descriptions which are used while operating, verifying, calibrating, and troubleshooting the ADAM instrumentation system with the pocket Video Display Unit (VDU) shown in Figure 174. Figure 175 presents the menu tree for the ADAM system. Two system-level checkout software packages are resident in ADAM: software for the system self-test executed when the system is reset, and software for the interactive checkout of the manikin. A complete description of the software routines and flow charts is located in Appendix F.

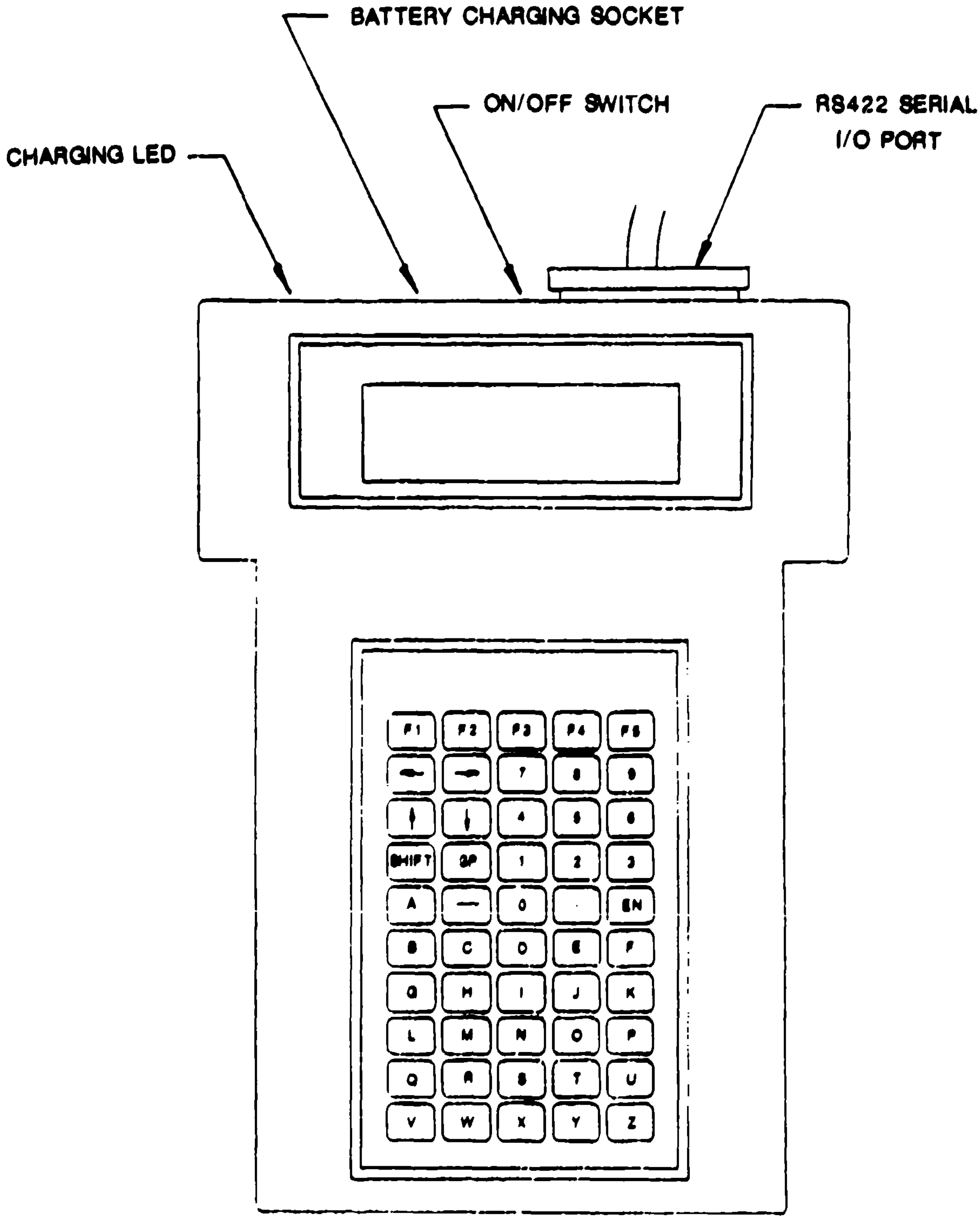



Figure 174. Pocket Video Display Unit

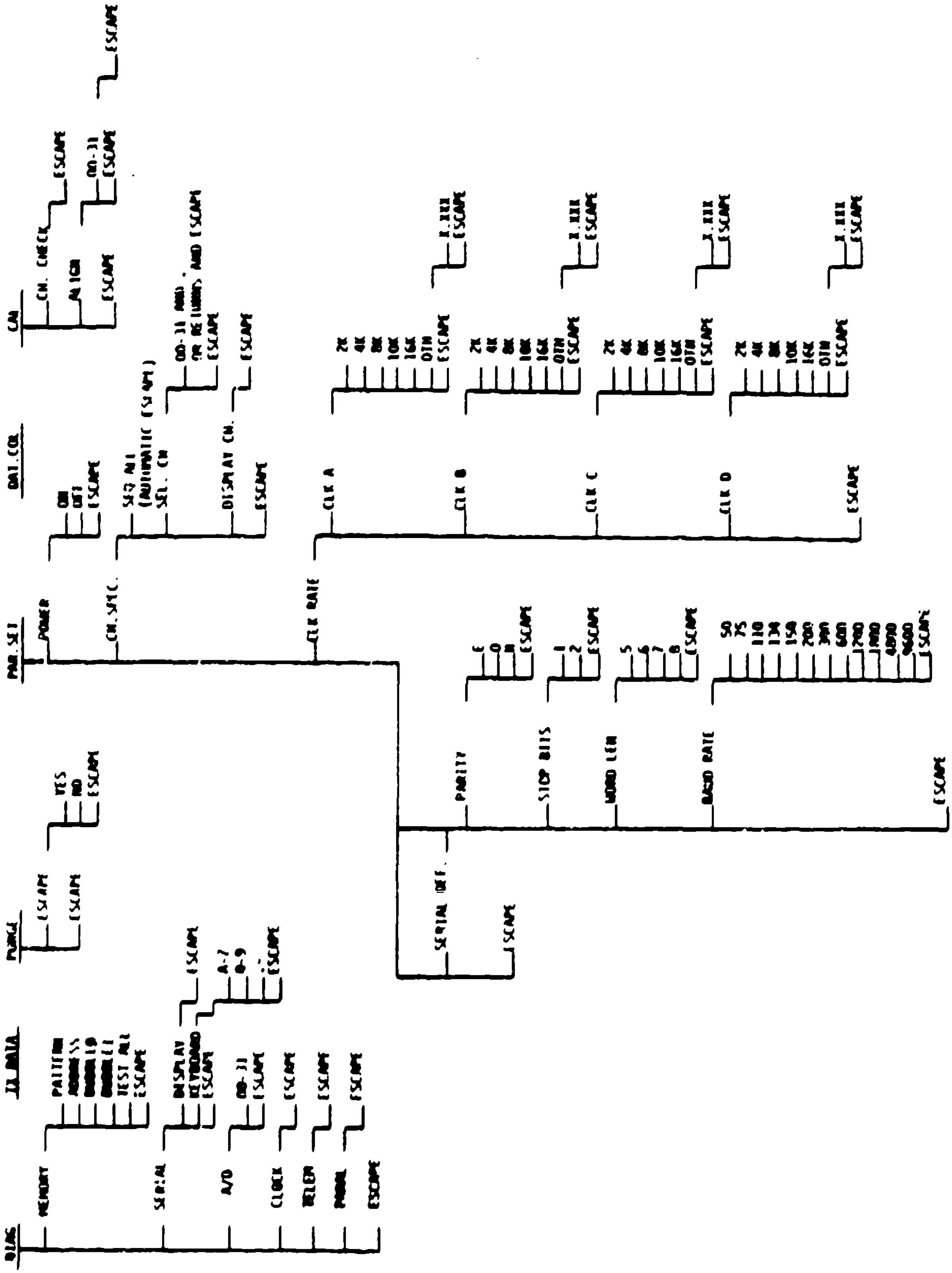



Figure 175. ADAM Menu Tree

3.5.2. Main Menu

The MAIN MENU is presented below with a list of menu selection descriptions:

| | |
|---|---|
| 1. DIAG | 2. CAL |
| 3. PAR.SET | 4. TX DATA |
| 5. DAT.COL | 6. PURGE |

DIAG is the abbreviation for diagnostics.
CAL is the abbreviation for calibrate mode.
PAR.SET is the abbreviation for parameter setting.
TX DATA is the abbreviation for the parallel transmission of data to the DRASS.
DAT.COL is the abbreviation for data collection mode.
PURGE is the abbreviation for the clearing of data from the memory.

The MAIN MENU is the menu which appears on the pocket VDU following a successful system self-test. This menu provides the first level major branches for interactive activities with ADAM.

For all menu selections, entering F4 [ASCII "escape" character - 1B (hex)] on the pocket VDU returns the user to the previous level menu. Entering EN closes the keyboard entry process and sends a specific menu request to ADAM.

3.5.2.1. Diagnostics

When diagnostic operations are selected from the MAIN MENU, the following DIAGNOSTICS Menu will appear, presenting the user options for interactive diagnostics.

| | |
|---|---|
| 1. MEMORY | 2. SERIAL |
| 3. A/D | 4. CLOCK |
| 5. TELEM | 6. PARAL |

MEMORY is the SRAM diagnostic that writes and reads back expected data patterns into the ADAM memory. This used to check the SRAM integrity.

SERIAL is the I/O port diagnostic that sends expected test patterns to the VDU display.

A/D is the analog-to-digital conversion system diagnostic. It will display 1 mux channels hexadecimal output.

CLOCK is the filter clock generation circuitry diagnostic that resets the filter clock through five frequencies.

TELEM is the telemetry generator circuitry diagnostic that generates a set pattern of data out of the telemetry port.

PARAL is the parallel port circuitry diagnostic used to verify the integrity of data transmission between the ADAM and DRASS.

3.5.2.1.1. MEMORY Diagnostics

When memory diagnostics are selected, the following MEMORY Diagnostics Menu will appear:

| | |
|---|---|
| 1. PATTERN | 2. ADDRESS |
| 3. BUBBLE0 | 4. BUBBLE1 |
| 5. TEST ALL | |

PATTERN is a quadruple fixed data pattern test.
ADDRESS is an address-in-address test.
BUBBLE0 is a floating zero in a field of ones test.
BUBBLE1 is a floating one in a field of zeroes test.
TEST ALL is a sequence of all four of the above tests.

If there are test data present in memory, the following prompt will be displayed:

VALID DATA IN MEMORY

This prompt is provided as a safeguard to protect any test data in the memory from inadvertently being destroyed. Before memory diagnostics can be performed, the test data must be purged from the SRAM array using the PURGE selection on the Main Menu.

- PATTERN Test: The PATTERN test fills RAM with all 5s using byte accesses, followed by a read of all RAM locations by byte accesses with data verification taking place at each location. The entire process is repeated three more times using As, Fs, and Os as subsequent test patterns. "PATTERN TEST RUNNING" is displayed during the diagnostic, and "MEMORY TEST PASSED" is displayed following a successful completion of the PATTERN test. A failure will be indicated on the display with the value read, value expected, and the address where the error occurred.

- <u>ADDRESS Test</u>: The ADDRESS test consists of a writing to each long word location (on long word boundaries) the address of each location to its own memory location, reading it back, and verifying a valid transaction at each location. An example of valid data would be the writing and subsequent reading of the data value of 010006EC (hex) to/from address location 010006EC (hex). "ADDRESS TEST RUNNING" is displayed during the diagnostic, and "MEMORY TEST PASSED" is displayed following a successful completion of the ADDRESS test. A failure will be indicated on the display with the value read, the value expected, and the address where the error occurred.

- <u>BUBBLE0</u>: The BUBBLE0 tests the RAM by floating a zero through each word length location through 16 groups of write/read/verify operations with a background pattern of 15 ones. This test has the effective result of moving a zero from the least significant bit of the lowest RAM location through the most significant bit of the highest tested RAM location. This test is effective in identifying any address location that has a bit location stuck in the high state (stuck one), whether due to a bad memory device or a malfunctioning data bus buffer. "BUBBLE0 TEST RUNNING" is displayed during the diagnostic, and "MEMORY TEST PASSED" is displayed following a successful completion of the BUBBLE0 test. A failure will be indicated on the display with the value read, the value expected, and the address where the error occurred.

- <u>BUBBLE1 Test</u>: The BUBBLE1 tests the RAM by floating a one through each word length location through 16 groups of write/read/verify operations with a background pattern of 15 zeroes. This test has the effective result of moving a one from the least significant bit of the lowest RAM location through the most significant bit of the highest tested RAM location. This test is effective in identifying any address location that has a bit location stuck in the low state (stuck zero), whether due to a bad memory device or a malfunctioning data bus buffer. "BUBBLE1 TEST RUNNING" is displayed during the diagnostic, and "MEMORY TEST PASSED" is displayed following a successful completion of the BUBBLE1 test. A failure will be indicated on the display with the value read, the value expected, and the address where the error occurred as follows:

  MEMORY ERROR
  VAL EXP: XXXX

VAL READ: XXXX
ADDRESS: XXXXXXXX

- TEST ALL Test: The TEST ALL selection activates a test sequence which runs all four of the memory diagnostics in the sequence presented above. "TEST ALL RUNNING" is displayed during the diagnostic, and "MEMORY TEST PASSED" is displayed following a successful completion of the TEST ALL series of memory diagnostics. A failure will be indicated on the display with the value read, value expected, and the address where the error occurred. Since there are four distinct tests with different types of test data patterns used, the expected data value displayed indicates which memory diagnostic routine was running when the failure occurred.

3.5.2.1.2. SERIAL Diagnostics

The following SERIAL Diagnostics Menu may be selected:

1. DISPLAY
2. KEYBOARD

DISPLAY is an ADAM transmit only serial diagnostic requiring a visual inspection of the terminal data for test pass/fail criteria.

KEYBOARD is an echo test where ADAM places on the terminal display data that are entered by the user on the keyboard.

- DISPLAY Test: The DISPLAY test is a unidirectional serial communications test conducted from ADAM to the serial terminal connected. It can be operated with either the RS-232C or the RS-422 serial interface circuits, whichever is appropriate for the terminal in use. ADAM continuously writes to the terminal all of the capital letters from A through Z, the 10 numbers from 0 through 9, and the four punctuation marks . / ? and -. There are approximately 3-second intervals between write activities which allow user to visually inspect the terminal display for valid character printing.

- KEYBOARD Test: The KEYBOARD test is a bidirectional serial communications test conducted in an echo manner. The ADAM writes to the terminal display valid entry characters sent to it as the user makes entries on the keyboard. The valid keypad entries include all of the letters from A through Z, all ten digits from 0 through 9, and the two punctuation marks . and -. The KEYBOARD test requires a visual pass/fail determination from the user.

3.5.2.1.3. A/D Diagnostics

When the A/D diagnostic routine is selected, the following prompt will appear:

ENTER MUX CH:

The user must enter the desired multiplexer channel. Valid entries are any one or double combinations of the numbers from 0 through 31 decimal. ADAM will display the four channels of data continuously for the selected multiplexer channel number. The continuous display of the four channels of information is terminated by an entry of F4, which places the DIAG Menu back on the display. an example of the interaction follows:

| | |
|---|---|
| ENTER MUX CH: | message sent by ADAM |
| 22 | MUX channel selected for example |
| EN | entry of selected channel |
| | |
| A/D TEST CH: 22 | message displayed during test |
| 83 83 83 83 | example data for one sample |
| | |
| F4 | user entry terminates test and |
| | |
| 1. MEMORY 2. SERIAL | ADAM displays Diagnostics Menu again |
| 3. A/D 4. CLOCK | |
| 5. TELEM 6. PARAL | |

A/D Diagnostics help determine the ADAM system's capacity to acquire and convert analog information to a digital format. The user must know the sensor information for the chosen MUX channel to be able to visually verify that the data displayed is correct for that MUX channel.

3.5.2.1.4. CLOCK Diagnostics

When the clock diagnostic has been selected, the terminal will display:

CLOCK TEST RUNNING

This free-running test requires no user selectable inputs. All four filter clocks generate the same output frequency simultaneously. A reset condition and each of the following five frequencies presented in loopback fashion are: 2.000 KHz, 4.000 KHz, 8.000 KHz, 10.000 KHz, and 16.128 KHz. An input of F4 returns the user to the DIAG Menu.

### 3.5.2.1.5. TELEMETRY Diagnostics

When the telemetry diagnostic has been selected, the terminal will display:

TELEMETRY TEST RUNNING

This free-running test has no user selectable inputs. The test involves visual verification of telemetry port operations using the telemetry display. The telemetry generator develops a signal which conforms to IRIG 106-80 telemetry standards.

The following data values will appear on the telemetry word selector display during this test and should be visually verified.

| CHANNEL NUMBER | DATA VALUE | CHANNEL NUMBER | DATA VALUE | CHANNEL NUMBER | DATA VALUE |
|---|---|---|---|---|---|
| 1 | 250 | 45 | 40 | 89 | 84 |
| 2 | 243 | 46 | 41 | 90 | 85 |
| 3 | 32 | 47 | 42 | 91 | 86 |
| 4 | 0 | 48 | 43 | 92 | 87 |
| 5 | 0 | 49 | 44 | 93 | 88 |
| 6 | 1 | 50 | 45 | 94 | 89 |
| 7 | 2 | 51 | 46 | 95 | 90 |
| 8 | 3 | 52 | 47 | 96 | 91 |
| 9 | 4 | 53 | 48 | 97 | 92 |
| 10 | 5 | 54 | 49 | 98 | 93 |
| 11 | 6 | 55 | 50 | 99 | 94 |
| 12 | 7 | 56 | 51 | 100 | 95 |
| 13 | 8 | 57 | 52 | 101 | 96 |
| 14 | 9 | 58 | 53 | 102 | 97 |
| 15 | 10 | 59 | 54 | 103 | 98 |
| 16 | 11 | 60 | 55 | 104 | 99 |
| 17 | 12 | 61 | 56 | 105 | 100 |
| 18 | 13 | 62 | 57 | 106 | 101 |
| 19 | 14 | 63 | 58 | 107 | 102 |
| 20 | 15 | 64 | 59 | 108 | 103 |
| 21 | 16 | 65 | 60 | 109 | 104 |
| 22 | 17 | 66 | 61 | 110 | 105 |
| 23 | 18 | 67 | 62 | 111 | 106 |
| 24 | 19 | 68 | 63 | 112 | 107 |
| 25 | 20 | 69 | 64 | 113 | 108 |
| 26 | 21 | 70 | 65 | 114 | 109 |
| 27 | 22 | 71 | 66 | 115 | 110 |
| 28 | 23 | 72 | 67 | 116 | 111 |

| CHANNEL NUMBER | DATA VALUE | CHANNEL NUMBER | DATA VALUE | CHANNEL NUMBER | DATA VALUE |
|---|---|---|---|---|---|
| 29 | 24 | 73 | 68 | 117 | 112 |
| 30 | 25 | 74 | 69 | 118 | 113 |
| 31 | 26 | 75 | 70 | 119 | 114 |
| 32 | 27 | 76 | 71 | 120 | 115 |
| 33 | 28 | 77 | 72 | 121 | 116 |
| 34 | 29 | 78 | 73 | 122 | 117 |
| 35 | 30 | 79 | 74 | 123 | 118 |
| 36 | 31 | 80 | 75 | 124 | 119 |
| 37 | 32 | 81 | 76 | 125 | 120 |
| 38 | 33 | 82 | 77 | 126 | 121 |
| 39 | 34 | 83 | 78 | 127 | 122 |
| 40 | 35 | 84 | 79 | 128 | 123 |
| 41 | 36 | 85 | 80 | 129 | 124 |
| 42 | 37 | 86 | 81 | 131 | 126 |
| 44 | 39 | 88 | 83 | 132 | 127 |

3.5.2.1.6. PARALLEL Diagnostics

To execute the parallel port diagnostics, the DRASS must be connected first to ADAM and the DRASS parallel test initiated. This test will verify the functionality of the parallel ports on both ADAM and DRASS.

When the parallel test is selected, the following will appear on the display:

PARALLEL TEST RUNNING

This test performs long word writes followed by long word reads of the following test data in a loop list fashion until diagnostic is terminated or an error occurs:

00000000, 01010101, 02020202, ..., FFFFFFFF

The error messages encountered during the parallel test are:

| | |
|---|---|
| DRASS NOT CONNECTED | The DRASS is physically not connected or the connect status is not being detected by ADAM. |
| TRANSMIT TIME-OUT ERROR | ADAM has timed out waiting for the DRASS write status to go not busy. |

| | |
|---|---|
| RECEIVE TIME-OUT ERROR | ADAM has timed out waiting for the DRASS ready-to-read status to be set. |
| RECEIVE DATA ERROR | The data read from the DRASS is not correct. |

Once the DRASS is connected to the ADAM and is placed in parallel test mode, the parallel test on ADAM can be selected repeatedly. If an error does occur, select the test again to see if symptoms change. F4 returns the user to the to DIAG Menu.

3.5.2.2. Calibration

When CAL operations are selected from the MAIN MENU, the following CALIBRATION Menu will appear:

1. CH.CHECK
2. ALIGN

CH.CHECK is an automatic channel check.
ALIGN is an interactive channel alignment check.

3.5.2.2.1. CH.CHECK Operations

To enter CH.CHECK, select 1 and EN. The terminal will display:

BAD CHANNELS

and a list of MUX channels containing faulty sensor signals. The Channel Check automatically samples all channels in NON-RCAL mode, and then samples all channels in RCAL mode. It checks each channel for a delta spread of 20 bits. If the delta is large enough, the channel is considered operational. If a channel does not report the delta difference, then the MUX channel containing that apparently bad sensor channel is reported as a bad channel. A report of a bad channel during this test does not necessarily confirm a bad channel. The rotational sensors are also checked in RCAL mode; if they are within 20 bits to the center of their rotation, they will be reported as a bad channel. The list of reported bad channels should be recorded prior to leaving this test. The following ALIGN procedure will confirm actual bad channels. F4 returns the user to the CALIBRATION Menu.

3.5.2.2.2. ALIGN Operations

When the ALIGN mode is selected, the terminal will display the following prompt:

ENTER MUX CH:

Select any valid MUX channel number from 0 through 31 (decimal). An example of the interaction follows, with the subsequent display of channel information:

| | |
|---|---|
| ENTER MUX CH: | message sent by ADAM |
| 5 | MUX channel selected for example |
| EN | entry of selected channel |

ADAM clears the display of the above information and displays the following:

| | |
|---|---|
| NON-RCAL CH: 5 | message displayed during test |
| 64 65 65 64 | example data for one sample |
| RCAL READINGS | message displayed during test |
| 40 40 41 42 | example data for one sample |

The data, both NON-RCAL and RCAL, is continuously updated and displayed for the four sensor channels defined by the selected MUX channel. This example shows there is sufficient delta between all four sensor channels NON-RCAL and RCAL readings for the channel to not have been reported as bad during the CH.CHECK test. This free-running test is beneficial in alignment procedures as the display can be used as the real-time reporter during the adjustment of excitation buffer potentiometers, full range rotation sensor swing tests, and other analog alignment procedures. If a channel was reported bad during CH.CHECK, this test will indicate which channel does not have sufficient delta difference between NON-RCAL and RCAL values. If the channel is a rotational sensor channel, then rotate that joint during this test to verify its operational status.

3.5.2.3. Parameter Setting

When PAR.SET (parameter setting) operations are selected, the following PARAMETER SETTING Menu will appear:

| | |
|---|---|
| 1. CH.SPEC. | 2. CLK RATE |
| 3. SERIAL DEF. | 4. POWER |

CH.SPEC. is the channel specification setting procedure.
CLK RATE is the filter clock frequency setting procedure.
SERIAL DEF. is the serial port parameter setting procedure.
POWER is the mode for toggling ON and OFF system analog power.

3.5.2.3.1. CHANNEL SPECIFICATIONS Operations

When CH.SPEC. (channel specifications) is selected, the following menu will appear:

1. SEQ.ALL 2. SEL.CH
3. DISPLAY CH.

SEQ.ALL is the selection of the default acquisition list.
SEL.CH is the procedure for user selection of channels to be sampled.
DISPLAY CH. is the output fiom ADAM of the current list of channels to be sampled whether by default or choice.

SEQ.ALL Operations: This default procedure samples all ADAM channels sequentially. A user selection of SEQ.ALL (sequence all channels) requires no further entries following its selection. ADAM will automatically keep the CHANNEL SPECIFICATIONS Menu on the terminal display.

SEL.CH: When this options is selected, the following prompt will appear:

ENTER MUX CH:

Enter the desired sampling list using periods or the EN key as delimiters. The maximum number of multiplexer values that can be entered is 512. Any time the number of multiplexer values is different from 32 the system sampling rate is affected if a change is not made in the oscillator frequency for the telemetry generator. The terminal display will scroll automatically as entries are made. Enter F4 to close list and return to CHANNEL SPECIFICATIONS Menu.

The following example demonstrates the SEL.CH function as it would be used from the MAIN MENU:

| | |
|---|---|
| 3 | to select PAR.SET mode |
| EN | to enter the selected PAR.SET mode |
| 1 | to select CH.SPEC. mode |
| EN | to enter the selected CH.SPEC. mode |
| 2 | to select SEL.CH mode |
| EN | to enter the selected SEL.CH mode |
| | |
| ENTER MUX CH: | prompt message sent by ADAM |

At this time, the user is to enter the desired multiplexer channel sampling order with periods or entry keystrokes used as value delimiters, pressing F4 to close the list.

| | |
|---|---|
| 20.1.3.5.2.4.7.8.9 | Note that there are just 20 characters per line shown, |
| .10.15.11.16.12.17.2 | as presented by the terminal. |
| 0.13.18.14.19.21.22EN | |
| 31.30.28.26.24.23EN | Less than 20 characters can be placed on a line for |
| 25.27.29F4 | better readability. |

DISPLAY. CH. Operations: This mode of operation inspects the sampling order list currently logged in ADAM. During this operation, the entire multiplexer value list will be written to the terminal. Enter F4 to clear display and return to PARAMETER SETTING Menu.

3.5.2.3.2. CLOCK RATE Operations

When the user selects clock rate operations, the following CLOCK RATE Menu will appear on the display:

| | |
|---|---|
| 1. CLK A | 2. CLK B |
| 3. CLK C | 4. CLK D |

CLK A through CLK D are the four filter clocks, each individually settable. Selecting any one of the above generates the next level menu:

| | | |
|---|---|---|
| 1. 2K | 2. 4K | 3. 8K |
| 4. 10K | 5. 16K | 6. OTH |

2K is the preset selection of a 2.000 KHz clock.
4K is the preset selection of a 4.000 KHz clock.
8K is the preset selection of a 8.000 KHz clock.
10K is the preset selection of a 10.000 KHz clock.
16K is the preset selection of a 16.125 KHz clock.
OTH allows user to manually define any other clock frequency.

For example, to set CLK C to a frequency rate of 10000 Hz, the user selects 4 and EN. CLK C is now set at 10000 Hz.

To set CLK C to any other frequency besides those listed in one through five, the user selects 6 and EN. The following prompt will appear:

ENTER PRESCALE
AND COUNT (X.XXX)

where (X.XXX) is defined as (PRESCALE.COUNT)

| Prescale Value | Function |
|---|---|
| 0 | TIMER STOPPED |
| 1 | DIVIDE-BY-FOUR (4) PRESCALER |
| 2 | DIVIDE-BY-TEN (10) PRESCALER |
| 3 | DIVIDE-BY-SIXTEEN (16) PRESCALER |
| 4 | DIVIDE-BY-FIFTY (50) PRESCALER |
| 5 | DIVIDE-BY-SIXTY-FOUR (64) PRESCALER |
| 6 | DIVIDE-BY-ONE-HUNDRED (100) PRESCALER |
| 7 | DIVIDE-BY-TWO-THOUSAND (2000) PRESCALER |
| 8 | invalid entry |
| 9 | invalid entry |

MAIN COUNTER VALID DECIMAL NUMBERS RANGE: 0 through 255 (inclusive)

SYSTEM CLOCK FREQUENCY: 8.000 MHz (which runs the prescalers at 4.000 MHz). All preset clock rate values, as well as the discussion of OTHer clock rates are based upon the computer system clock rate of 8.000 MHz.

To find and enter valid OTHER CLOCK RATES, the user first selects the desired cutoff frequency. The desired clock frequency is found by multiplying the cutoff frequency by 50. The total number of counts (between the main counter and the prescaler) is found by multiplying the clock frequency by two. The total count is then divided by one of the prescaler values, with a remainder within the range of the main counter. The equation for this operation is:

MAIN COUNTER VALUE = [40000/(PRESCALE*DESIRED CUTOFF FREQUENCY)]

It is possible to have several different value combinations of prescale and main counter numbers that work for a target cutoff frequency. It does not matter which combination is used. There is also a possibility that an exact desired cutoff frequency can not be achieved because the prescale values available do not permit whole number values to work in the main counter. In this case, the combination that provides the frequency that best matches the desired cutoff frequency.

Table 56 presents cutoff frequencies and their responses for OTHer clock frequency definitions. The entry process is identical to that presented above.

3.5.2.3.3. SERIAL DEFINITIONS Operations

When the SERIAL DEF. operation has been selected, the user is able to change the serial communications parameters on the manikin. These changes are made using an interactive procedure that allows the user to change the parity, stop bits, word length, and baud rate. after all of the selections have been made, the change is made to the serial port on ADAM. Table 57 outlines the selections the user may make when changing the serial communications parameters.

3.5.2.3.4. Power

The POWER mode allows the user to turn off the voltage regulators on the analog signal conditioning circuit cards to reduce power consumption after data have been taken. The user enters the POWER mode from the MAIN MENU by selecting:

3 to select PAR.SET mode
EN to enter the selected PAR.SET mode
4 to select POWER mode
EN to enter the selected POWER mode

The terminal will display the prompts:

1. POWER ON to turn power back on
2. POWER OFF to turn power off

## TABLE 56. TYPICAL OTHER CLOCK FREQUENCIES

| Filter Frequency | Filter Cutoff Frequency | Prescale and Count (X.XXX) Entry(s) |
|---|---|---|
| * 9.0625 | 0.78125 Hz | 7.000 (minimum possible) |
| 50 | 1 | 7.200 |
| 100 | 2 | 7.100, 6.200 |
| 200 | 4 | 7.050, 6.100, 4.200 |
| 250 | 5 | 7.040, 6.100, 5.125, 4.160 |
| 400 | 8 | 7.025, 6.050, 4.100 |
| 1000 | 20 | 7.010, 6.020, 4.040, 3.125, 2.200 |
| 1250 | 5 | 7.008, 6.016, 5.025, 4.032, 3.100, 2.160 |
| 1373.5 | 27.47 | 3.091 |
| 1388.9 | 27.78 | 3.090 |
| 1405 | 28.1 | 3.089 |
| * | 40 | (2000 Hz preset selection #1) |
| 2500 | 50 | 7.004, 6.008, 4.016, 3.050, 2.080, 1.200 |
| 2840 | 56.8 | 5.033, 3.132 |
| 3333.34 | 66.67 | 7.003, 6.006, 4.012, 2.060, 1.150 |
| 3863.6 | 72.7 | 4.011, 2.055 |
| * | 80 | (4000 Hz preset selection #2) |
| 4505 | 90.1 | 1.111 |
| 4807.7 | 96.15 | 3.026, 1.104 |
| 5000 | 100 | 7.002, 6.004, 4.008, 3.025, 2.040, 1.100 |
| 5208.34 | 104.17 | 5.006, 3.024, 1.096 |
| 6250 | 125 | 5.005, 3.020, 2.032, 1.080 |
| 6666.67 | 133.34 | 6.003, 4.006, 2.030, 1.075 |
| 7812.5 | 156.25 | 5.004, 3.016, 1.064 |
| * | 160 | (8000 Hz preset selection #3) |
| 8333.34 | 166.67 | 3.015, 2.024, 1.060 |
| | 200 | (10000 Hz preset selection #4) |
| 12500 | 250 | 3.010, 2.016, 1.040 |
| | 322.56 | (16128 Hz preset selection #5) |
| 25000 | 500 | 3.005, 2.008, 1.020 |
| 50000 | 1000 | 2.004, 1.010 |
| * 500000 | 10000 | 1.001 (maximum possible) |

*Preset or text precalculated settings.

TABLE 57. SERIAL PORT PARAMETER SUMMARY

| Parameter | Selection |
|---|---|
| Word Length | 5<br>6<br>7<br>8 |
| Parity | Even<br>Odd<br>None |
| Stop Bits | 1<br>2 |
| Baud Rate | 50<br>75<br>110<br>134<br>150<br>200<br>300<br>600<br>1200<br>1800<br>4800<br>9600 |

Enter 1 to turn power on and return to MAIN MENU. Enter 2 to turn the computer controlled power off and return to the MAIN MENU.

3.5.2.4. Data Collection Operations

DATA COLLECTION operations are usually entered following the successful checkout of ADAM prior to an actual test. This mode's data acquisition operations sample data and send it out the telemetry port as specified by the parameter setting sampling list. The DAT.COL mode is also actively monitoring the START signal line. When the START signal is received, ADAM will (in addition to telemetry) save the sampled data in the ADAM resident nonvolatile memory until full. After memory is full, ADAM will return to sampling and sending data out the telemetry port for a fixed period of time. Then ADAM will power down all nonessential systems for power conservation purposes and wait.

The user enters DAT.COL from the Main Menu selection number 5. If previous test data are in ADAM, the terminal will display:

MEMORY FULL

If ADAM is able to enter the data collection mode, the terminal will display:

COLLECTING DATA

3.5.2.4.1. Transmit Data Operations

TRANSMIT DATA operation is done after a successful test when data have been collected and stored in ADAM. The user would connect the DRASS to the ADAM parallel port and enter the TXDATA from the Main Menu selection number 4. When the transfer to data is done, the terminal will display:

TRANSFER COMPLETE

3.5.2.5. Purge

The purge routine is used to clean data from a memory module. This operation should only be done following a data transfer to the DRASS. Allow data within ADAM to remain intact until transfer from the DRASS to a more permanent storage medium and spot checked for validity. The PURGE process does not remove the system operating parameters, but it does destroy all test data, including pretest and posttest calibration samples. To select PURGE, enter 6 and EN.

The terminal will display the following prompt:

DO YOU WANT THE DATA
MODULES ERASED (Y/N)?

Any response other than Y will be interpreted as no and the terminal display will repost the MAIN MENU. When memory is clear, the terminal will display the MAIN MENU.

3.6. SUPPORT EQUIPMENT

This section describes two pieces of support equipment designed for use with the ADAM instrumentation. The first is the DRASS which is used as a nonvolatile mass storage device for the

ADAM test data, and the second is the lithium battery conditioner which is used to exercise the internal lithium batteries to eliminate an oxide build-up in the batteries prior to their use. A third piece of support equipment, the FPS, was previously described in Section 3.4.2.

### 3.6.1. Data Retrieval and Storage System

The DRASS is a microprocessor controlled piece of equipment designed to receive and store the test data from the ADAM instrumentation. The DRASS is intended to off-load the test data from the manikin quickly by way of the high speed parallel port and store the data temporarily in non-volatile memory until it is transferred to a permanent storage media. The DRASS contains 512 kilobytes of static RAM using the same memory board that is used in the ADAM instrumentation. The DRASS uses a microprocessor to control the upload and download procedures. The DRASS has been designed with RS-232 and RS-422 serial ports to upload the test data stored in its memory array to a host computer at baud rates up to 19200 baud.

The DRASS has the ability to operate from 115 VAC power or its own internal batteries. This provides a degree of flexibility in the DRASS operation. In a constrained laboratory environment, the DRASS is powered by the 115 VAC line; in field testing the internal batteries allow the DRASS to operate for up to 8 hours between charges.

The CPU in the DRASS is, like ADAM, the 68020. The schematic for the CPU and the central support circuitry is found in Figure 176. All 32 data lines between the CPU (U24) and the rest of the DRASS are buffered by U15, U23, U30, and U34. All 32 address lines from the CPU are buffered by U17, U18, U25, and U31. The reset circuitry is primarily located at and around U6 and U7. The main system clock and divider/driver circuit is at U27 and U26, respectively. The central interrupt controller and autovector generator devices are located at U16 and U41, respectively.

Figure 177 presents the schematic for the centralized DRASS ADIC logic. The DRASS internal addressing scheme is a combination of partial and total address decoding. The total address decoding is complete for any address ranges the removable memory boards may be able to use. Partial address decoding is used for the DRASS I/O ports in an area that is not usable by the removable memory boards. The PAS detector is located at U32. The I/O port select (IOPS) address detector is located at U20. In addition to generating an enable signal for the display, U11 is the core of the centralized DSACK signal generator with support from U1, U2, and parts of U3, U4, and U12. The multifunction peripheral (MFP) receives its chip select signal (MFPCS) from U10. This integrated circuit is also responsible for generating the R/W control line (CTRLR and

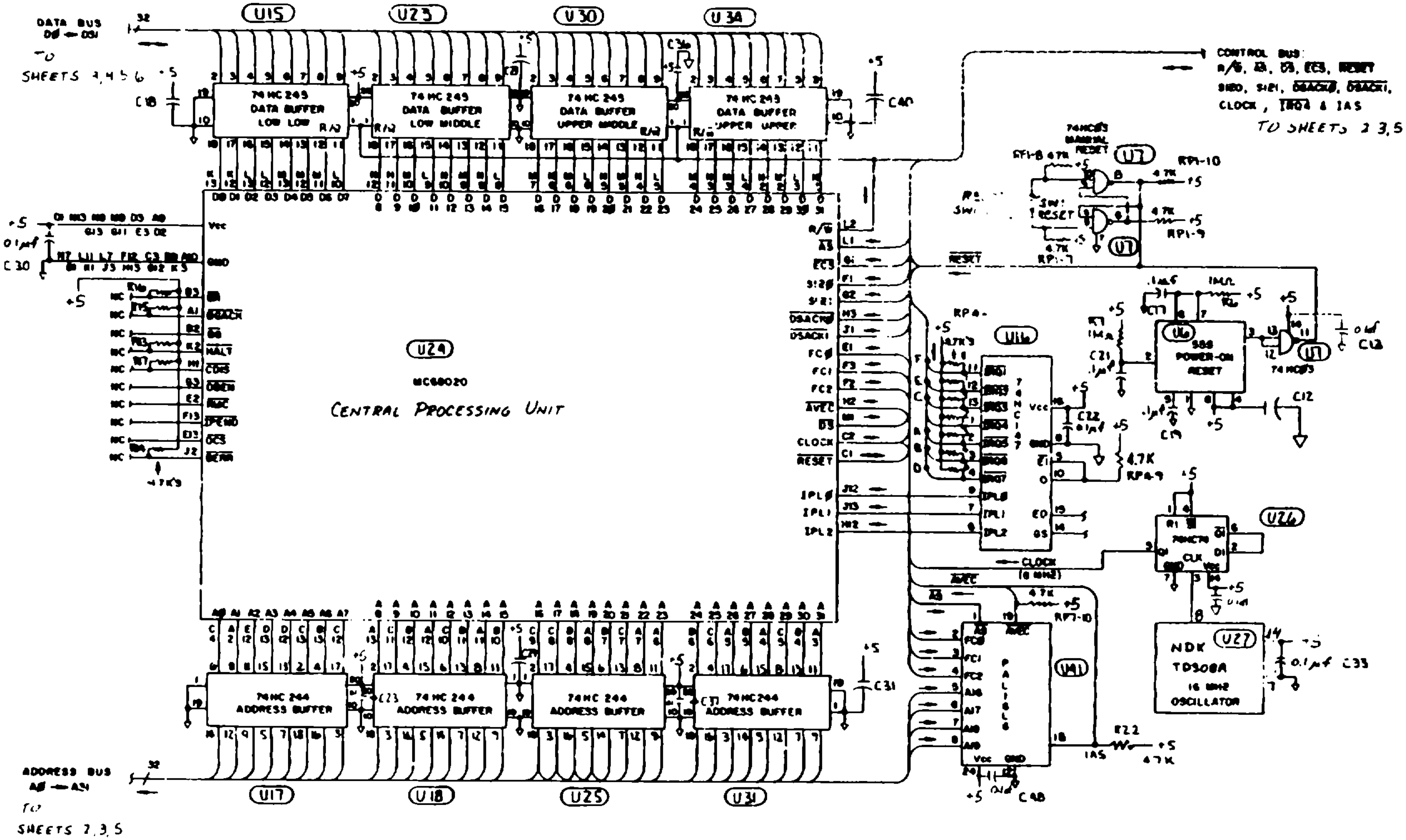



Figure 176. DRASS Microprocessor and Support Schematic

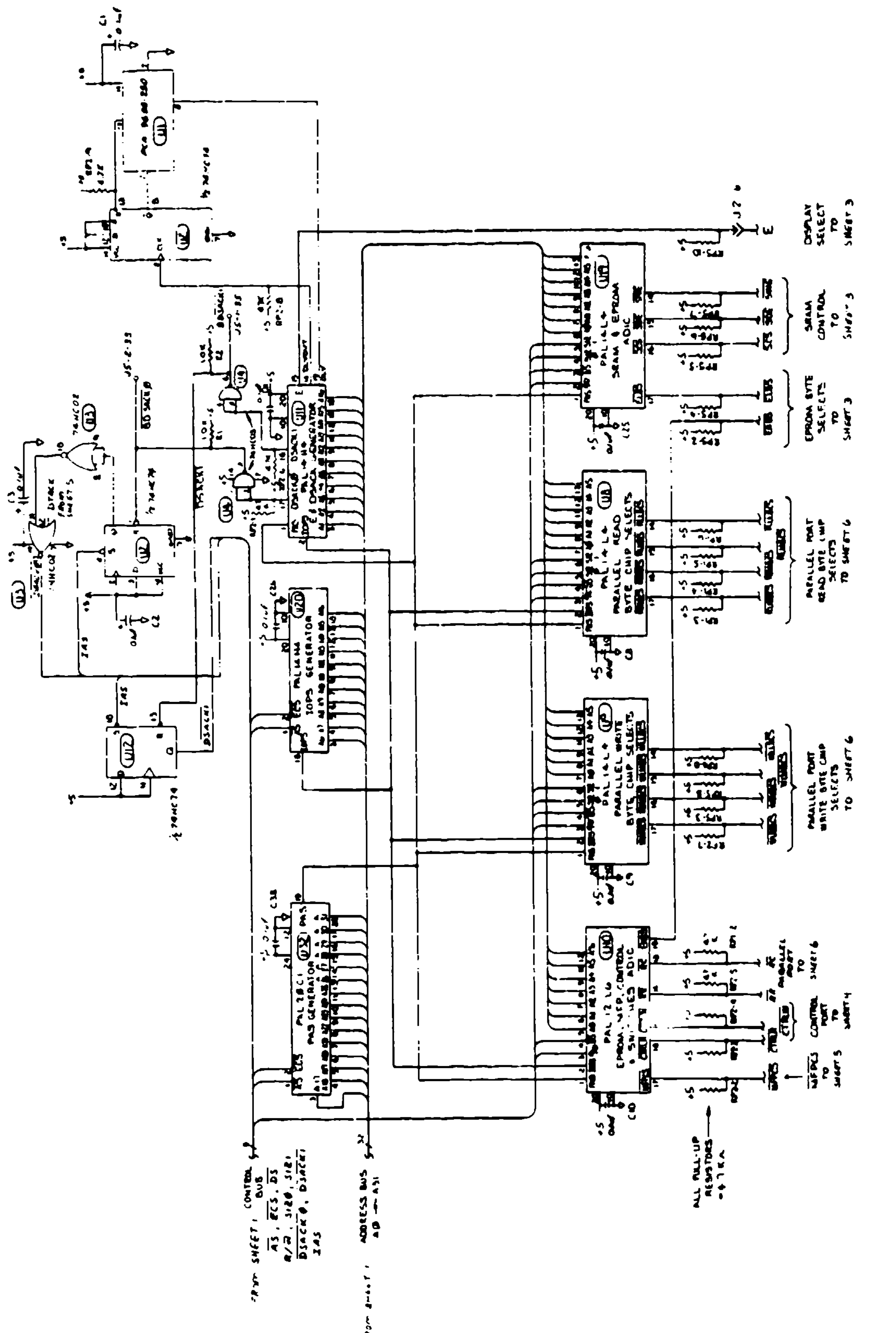


Figure 177. DRASS Address Decoder and Interface Controller (ADIC) Schematic

CTRLW) for the control port and two handshake related control lines (RR and RC) for the parallel port, and the high byte select for the EPROM (EHBS).

The parallel port write byte chip select lines (WXXBCS) are generated by U9. The parallel port read byte chip select lines (RXXBCS) are generated by U8. The last integrated circuit on this drawing is U19. This device generates the EPROM low byte select signal (ELBS) and three control lines for the CPU cache SRAM.

Figure 178 presents the DRASS EPROM schematic, cache SRAM, and the LCD display subsystem. This display is mounted on the DRASS front panel and connected to the DRASS PC board through J2. The loosely coupled cache SRAM at U37 is used by the CPU as a scratchpad and extended register memory so that operations in the removable memory board are not necessary. The DRASS EPROM is composed of two devices at U35 and U36 sectored into bytes. These devices house the DRASS resident software routines for all diagnostics and interface modes of operations with both ADAM and external data upload target computers.

Figure 179 presents the DRASS parallel port but does not include the origin and termination points for some handshake lines controlled by other elements of the DRASS hardware. U14, U22, U29, and U33 are the bidirectional data latching ports which can operate on any byte boundary as needed. U12 is part of the handshake circuitry that supports the high speed data transfers between itself and ADAM, and between itself and any other device conforming to the parallel port hardware and software protocols.

Figure 180 presents the DRASS control port, the primary human interface I/O port for mode selection, and operational status during transfer activities. This port also generates and receives most handshake line signals for the high speed parallel port. There are four LEDs, four individual switches, and four thumbwheel switches which interface to the control port. They are located on the front panel of the DRASS with the LCD display and are connected to the DRASS PC board through J2.

An output port byte latch is at U5. Two lines, form pins 17 and 18, are used for parallel port handshake operations. Four lines, pins 13, 14, 15, and 16 are controlled output signals for the status LEDs. The status LEDs receive their sinks through U40 as determined by U5.

There are three control port input byte gates as read by U13, U21, and U28. The lowest byte position reads two thumbwheel switch assemblies. The lowest nibble on the lowest byte reads the 16 position hexadecimal code for the serial port baud rate selection, while the highest nibble on the

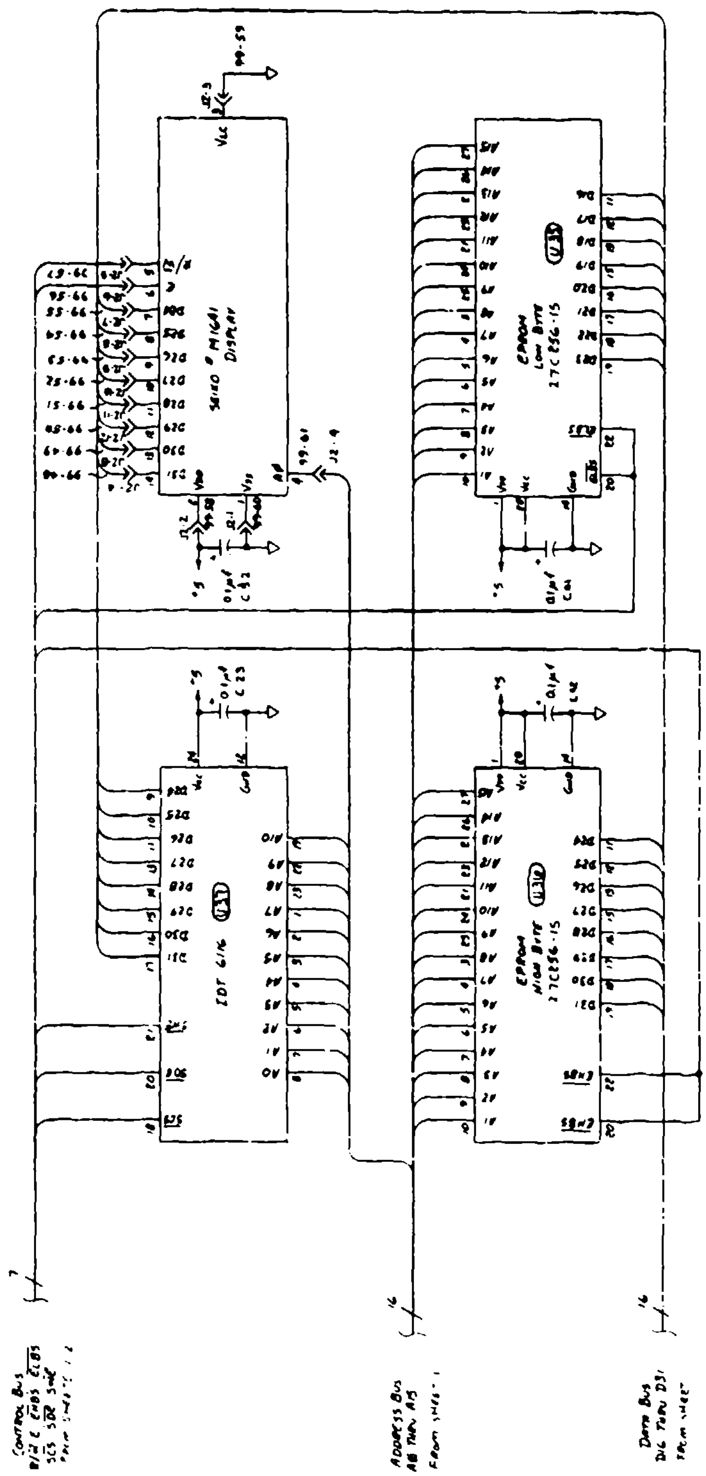


Figure 178. DRASS Eprom, SRAM, and Display Schematic

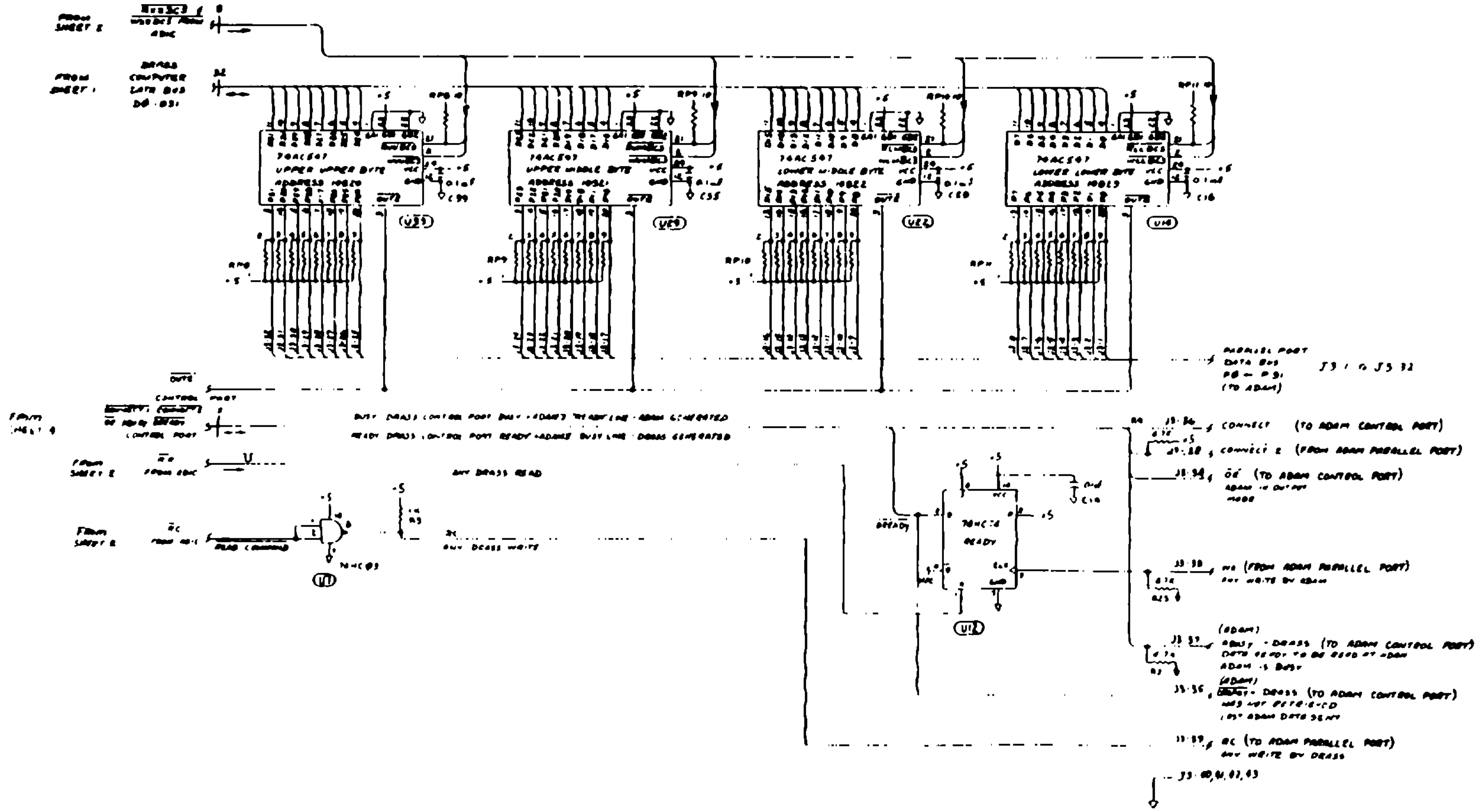


Figure 179. DRASS Parallel Port Schematic

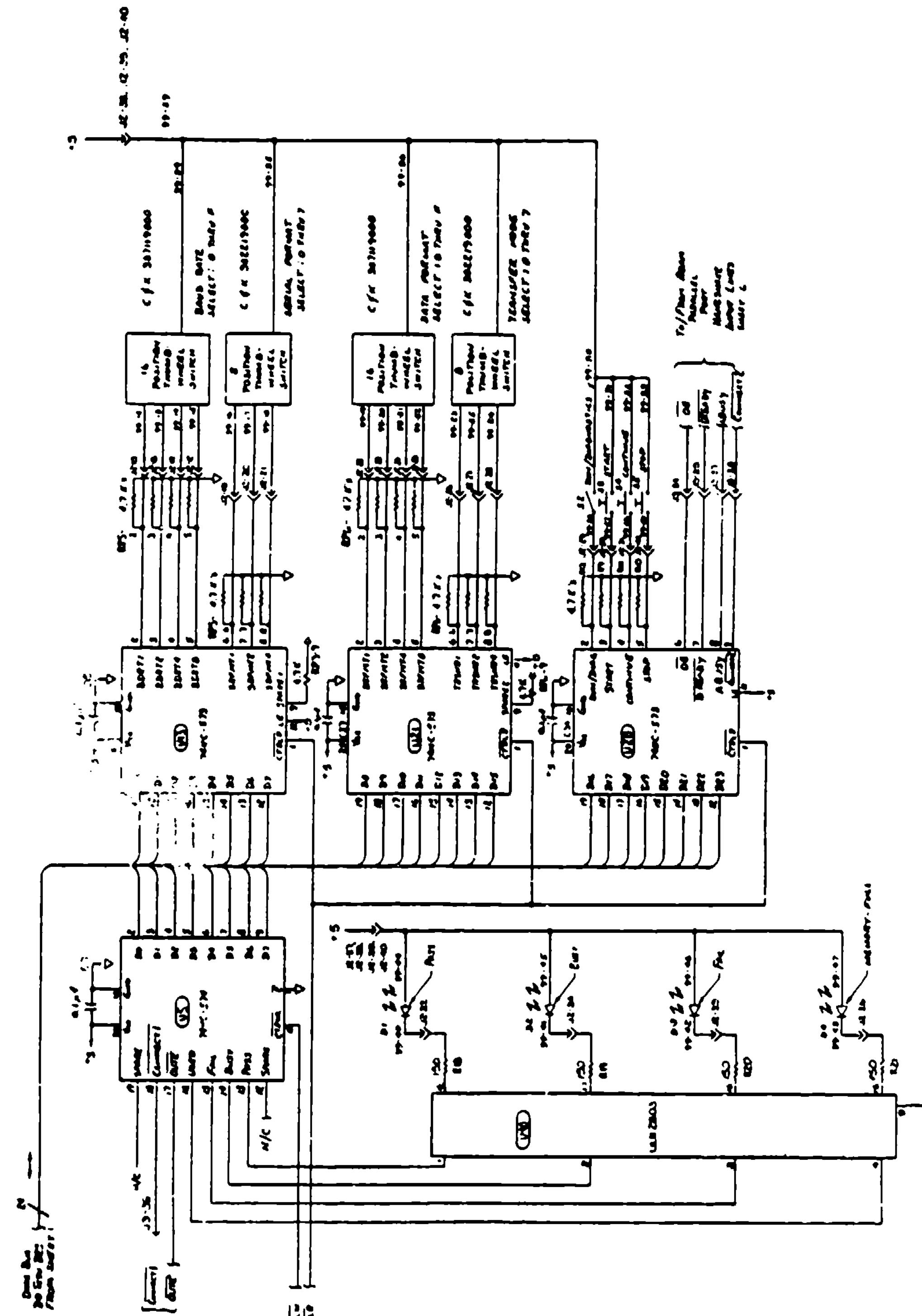


Figure 180. DRASS Control Port Schematic

lowest byte reads the eight position octal code for the serial port word format select. The middle byte reads two thumbwheel switch assemblies. The lowest nibble on the middle byte reads the 16 position hexadecimal code for the serial port data format select, while the highest nibble on the middle byte reads the eight-position octal code for the transfer mode selection. The high byte reads the state of four individual switch control inputs and four parallel port handshake input lines. The lowest nibble on the high byte reads the diagnostics/run mode toggle switch, the start control push button switch, the continue control pushbutton switch, and the stop control pushbutton switch, while the highest nibble of the high byte monitors the four output lines form the ADAM parallel port entering U28 at pins 6, 7, 8, and 9.

Figure 181 presents the schematic for the DRASS MFP port, which provides half-duplex serial communications between the DRASS and any other compatible device. The MFP not only provides the UART operations but also serves as a baud rate generator for the transmit and receive times-16 clocks.

Most activities for this port occur within the MFP, which is U38. The clock for the MPF operations is supplied by a divide-by-two circuit running off the main system clock at U26. The base frequency for the MFP baud rate generating function is controlled by a 2.4576 MHz crystal for the proper frequency divisions needed for standard communication baud rates. Communication handshake lines are buffered by U39, with biasing resistors inserted as needed for proper operations if the lines are not used by some serial devices. U44 provides the RS-232 electrical interface circuitry for both transmission and reception functions, U42 provides the RS-232 electrical interface circuitry for transmission operations, and U43 provides reception operations. Connector J1 provides interface between the I/O port and its panel connector (J3).

The power supply and power distribution schematic for the DRASS is presented on Figure 182. Power (115 VAC) is applied to the DRASS through a power cord connected to the female power connector in the top right corner of the DRASS front panel. When the main system power switch is placed in the EXT.AC position, power is simultaneously applied to the integral battery charger and to the linear power supply. The latter device provides power to both the DRASS PC board and to the memory modules plugged into the DRASS for data retention when the DRASS is operating on 115 VAC power. When the main system power switch is placed in battery position, the battery provides power to both the DRASS PC board and the memory modules. When the main power switch is in neither position, power is completely removed from the DRASS.

The DRASS is provided with a resident software routine that provides diagnostics of the major function blocks of the DRASS, and it also provides the run-time routines for the DRASS. The

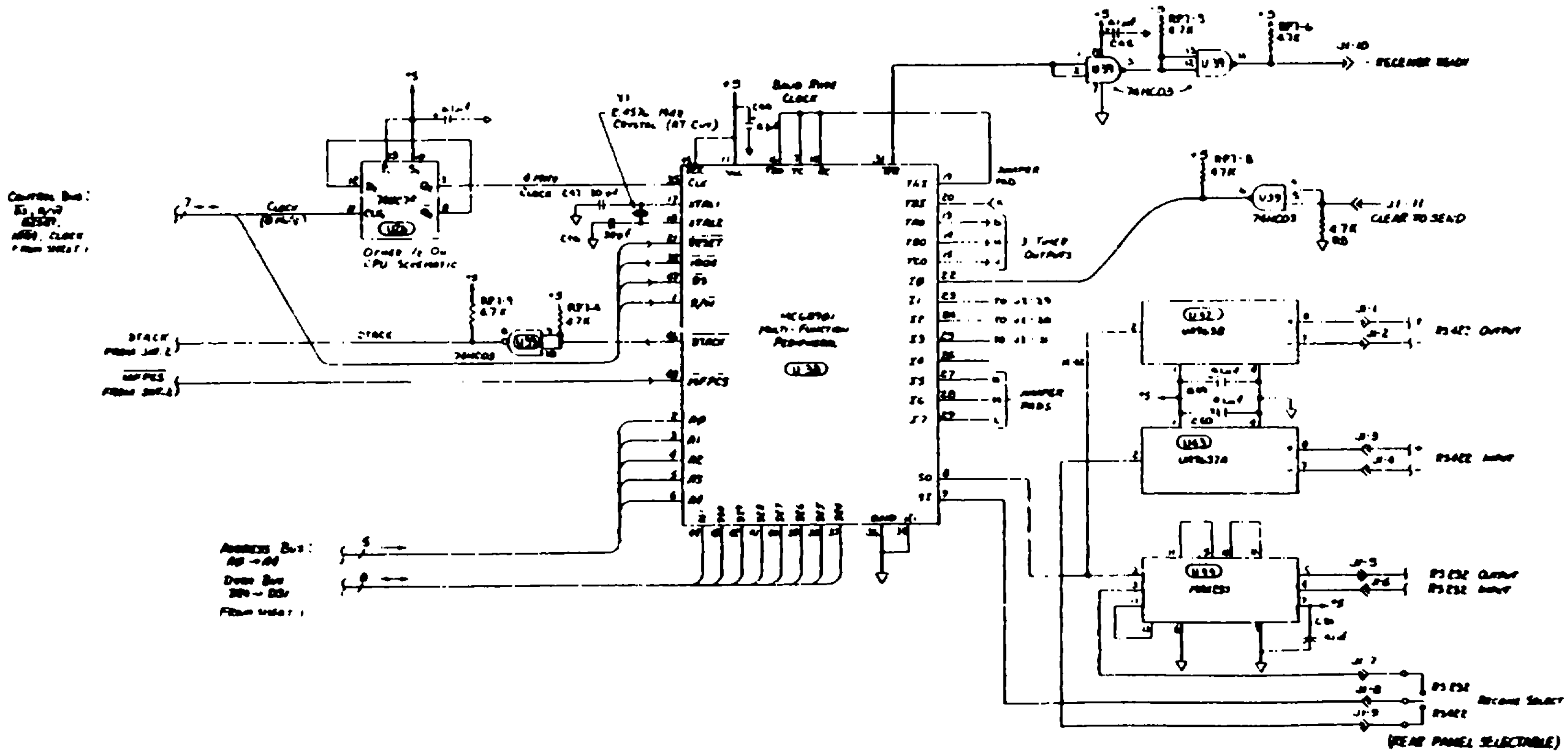



Figure i81. DRASS Multifunction Peripheral Port Schematic

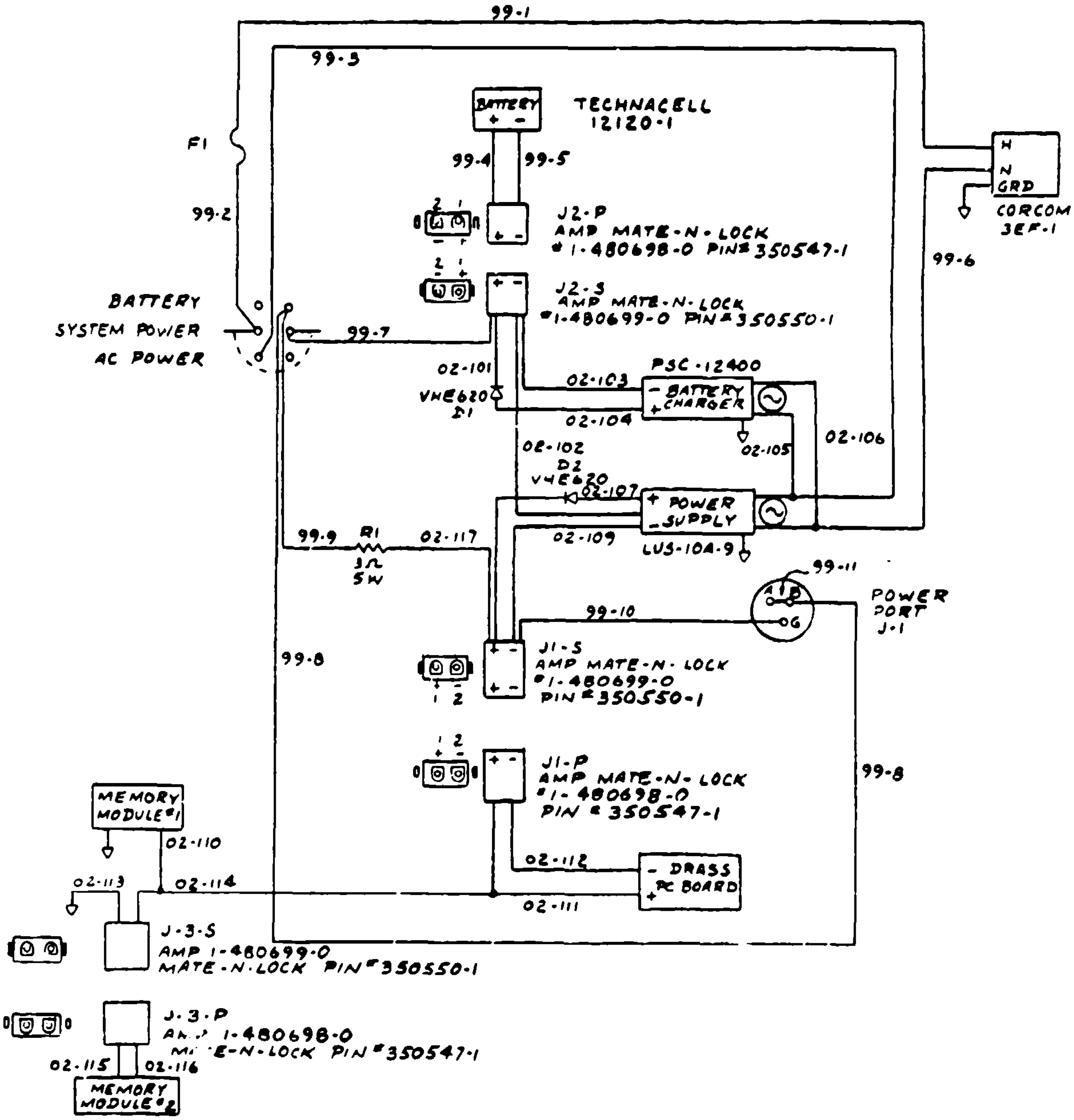



Figure 182. DRASS Power Schematic Distribution

diagnostic routines check the SRAM array, parallel port, serial port, LCD display, and panel switches. The two run-time routines included are the DOWNLOAD ADAM routine used to receive data from the manikin, and the SERIAL TRANSFER routine is used to upload the data in the DRASS to another computer via the serial port. Appendix G presents information on the serial drives routine of the DRASS.

The transfer of data out of ADAM to the DRASS is accomplished using the high speed parallel port and takes four seconds to complete, which includes time for error checking of each block of data as it is transferred. The serial transfer routine allows the data to be transferred at several baud rates and serial data formats, but requires much more time than the transfers using the high speed parallel port. The time required is dependent upon the baud rate selected and the speed at which the receiving device is able to read the data transmitted.

### 3.6.2. Lithium Battery Conditioner

The lithium battery conditioner is a piece of support equipment used to exercise the manikin batteries prior to use. This conditioning procedure is necessary to eliminate a characteristic of lithium batteries known as "voltage delay." This phenomenon occurs when the lithium batteries experience a heavy rate of discharge after a period of inactivity. The oxide that forms on the anode of the lithium cell during periods of inactivity, which provides the long shelf life for the lithium cell, increases the internal resistance of the cell. The increased internal resistance of the cell will cause the cell voltage to drop under load until the oxide is burned away. This process will require several seconds to several minutes depending on the type of lithium cell and the rate of discharge. The purpose of the lithium battery conditioner is to discharge the lithium cells for a period of time to eliminate this oxide buildup before the batteries are used to power the instrumentation.

The schematic for the lithium battery conditioner is shown in Figure 183. The lithium battery conditioner consists of a timer circuit and a relay driver circuit. It is designed to provide a discharge rate of approximately 50 percent of the discharge rate of each battery pack for a period of 10 minutes.

The timer circuit consists of the binary counters U1 and U4 and the counter decoding gate U2. As U1 is clocked at pin 10, U1 pin 1 clocks U4 at a rate of one pulse for every 4096 clock pulses at U1 pin 10. The eight input NAND gate, U2 is used to decode selected outputs of the binary counters U1 and U4. After 35,106 clock pulses have occurred, the output of U2 activates the relay control circuit and turns the relay circuit off.

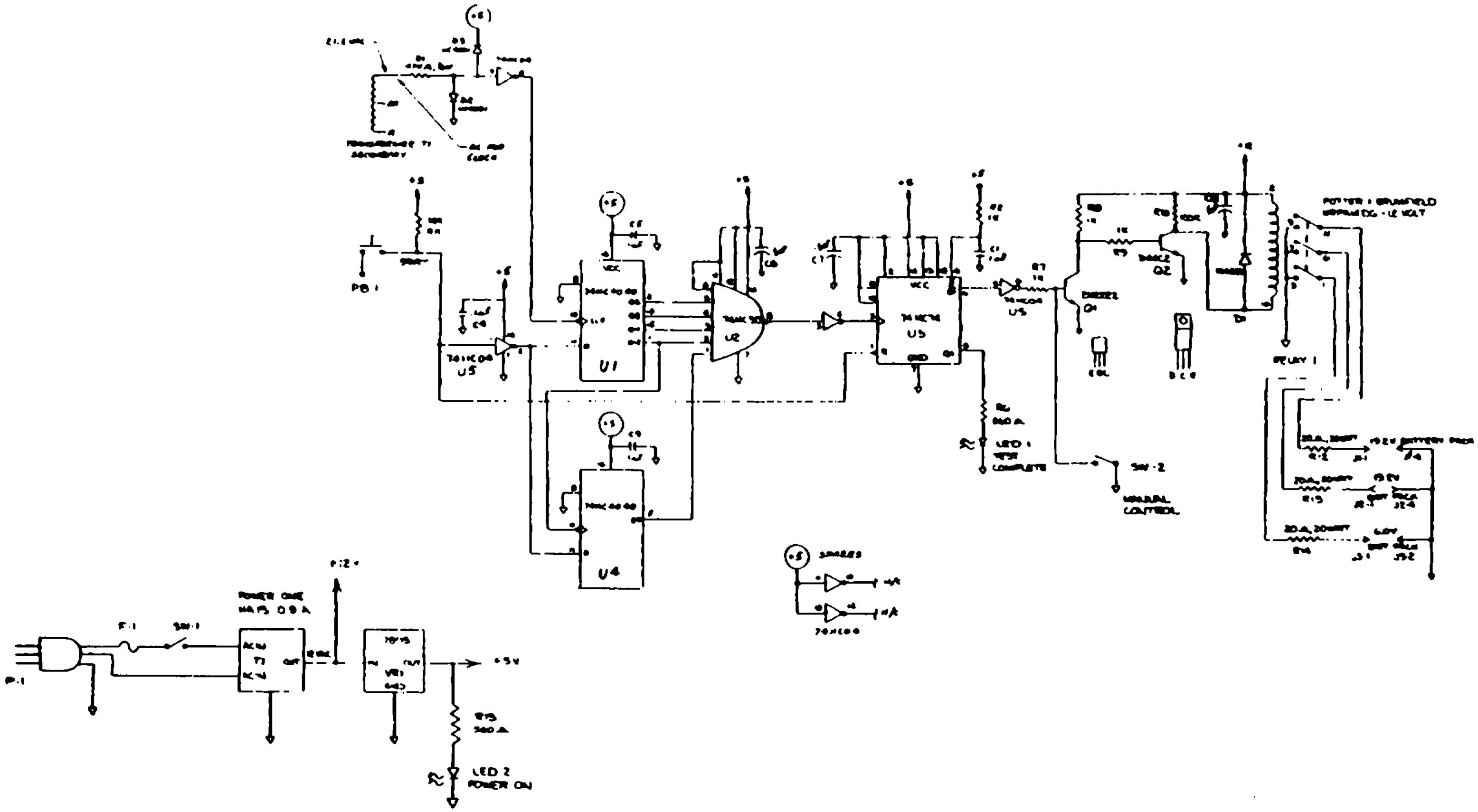



Figure 183. Lithium Battery Conditioner Schematic

The clock for this circuit is derived from the 60 Hz AC signal available from the power supply in the circuit. The diodes D2 and D3 clamp the voltage to the logic level range used by these circuits and the resistor R1 is used to limit the current through D2 and D3.

The relay control logic is activated by momentarily depressing the switch PB-1. This provides a reset signal to pin 1 of U5 and resets the counters U1 and U4 to zero. When pin 1 of U5 is low, it resets the output $\overline{Q}$ of U5 which turns Q1 off and Q2 on. With Q2 on the relay coil is energized, and the battery current is allowed to discharge through the 20 ohm, 20 watt resistors R12, R13, and R14. After approximately 9 minutes, 45 seconds, the counter circuit clocks the flip-flop of U5, and the $\overline{Q}$ output is set. This turns Q1 on and Q2 off, deenergizing the relay coil.

The R-C network of R2 and C1 is used in the relay control logic to act as a power-on reset. This ensures the relay control circuit will power-up with the relay coil deenergized.

Experimental data have shown that 10 minutes of conditioning eliminates the majority of problems of voltage delay with the lithium cells used in the ADAM. Even after conditioning of the cells, if several days pass between the time the batteries are conditioned and their time of use, some amount of voltage delay occurs. This problem is easily circumvented by operating the manikin using batteries for 5 to 10 to minutes before the actual test. This allows the cell voltages to increase and stabilize before any test data are taken.

## 3.7. DATA RETRIEVAL TECHNIQUES

The following is a brief presentation of some possible test configurations that demonstrate different techniques in capturing data with the ADAM instrumentation system. The two classes of test are constrained and unconstrained tests. A constrained test is one where it is possible to have an attached umbilical cable throughout the entire test for the purpose of transfer of power, data, and control/status information. An unconstrained test is one where, due to the nature of the test and/or its environment, it is not possible to have an umbilical attached for the duration of the pretest, test, and posttest sequence.

The ADAM instrumentation system has redundant data capturing techniques to increase the confidence level of obtaining all of the test data. For unconstrained tests, the typical technique used for capturing data is via some form of radio telemetry link. The biggest advantage of this technique, beyond being the well established standard, is the ability to handle almost any kind of test situation and environment possible, except for tests that are swamped with large radio frequency interference levels. Unfortunately, many test situations do exist, including those in which ADAM most typically will be working, where a good radio link is difficult to establish and maintain for the duration of the test. When the radio reception is degraded due to atmospheric noise or other radio interference or antenna misalignment, valuable (and often costly) data is either distorted or totally lost. This problem spawned the use of onboard memory for redundant data capture. If the telemetry data are lost, in whole or part, it can be retrieved from the onboard memory. If the onboard memory is physically destroyed by a test failure, at least some, if not all, of the data will be captured by the telemetry link. Under ideal conditions, two full sets of data will be obtained for full cross-verification purposes.

The most commonly used configuration of the ADAM instrumentation system will be that of the unconstrained test mode shown in Figure 184. The two primary data capture techniques, telemetry and onboard storage, are implemented. For nearly all test situations, it will be possible to have some form of hardwire hookup to the ADAM instrumentation system during both pretest and posttest modes. For those test environments where hardwire hookup is possible, a telemetry decommutation system is typically hookup to the ADAM telemetry port, and a serial communications terminal is hooked up to the ADAM serial port. Then the operator/user can interactively perform system diagnostics and calibration checks using both pieces of equipment. During an actual test event, the telemetry port sends data real-time to the receiving station, decommutation system, and a host computer for data storage. The onboard storage system also captures the same real-time data. Following a test, the DRASS extracts the data from the ADAM onboard memory, and then later uploads it into a host computer for permanent storage and posttest data analysis. The extraction of data from ADAM by the DRASS is performed via the high speed parallel port, and the subsequent upload of data to a host computer is performed via the RS-232 or RS-422 serial link using one of several different baud rates. While this will be the most commonly used unconstrained test configuration for ADAM, it will not be the only one. There are several test sites that either do not support radio telemetry, or they are not able to support the high modulation and data rates. In those instances, it becomes necessary to rely on the data retrieved by the onboard memory alone.

The DRASS undergoes a complete diagnostic checkout prior to the storage of any test data in the DRASS. These diagnostics are identical to those performed by ADAM on its own memory. Once it is determined that the DRASS target memory module is fully operational, a complete transfer of data is initiated from the ADAM memory to the DRASS. During the transfer, a check sum error checking routine is used to verify that the block of data written to the DRASS was not received in error.

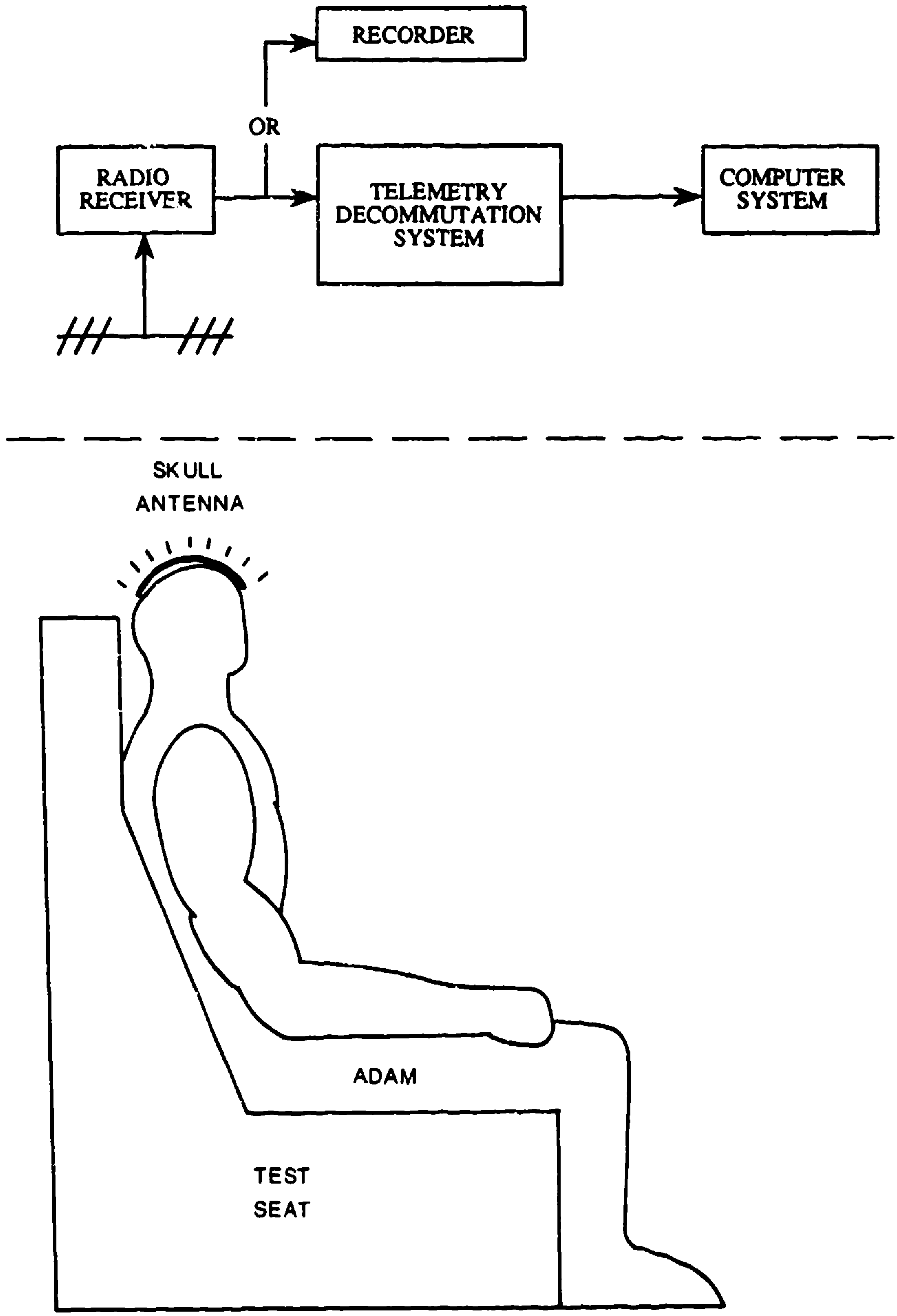



Figure 184. Totally Unconstrained Test Configuration

The other class of tests, constrained test configurations, allow more versatility in data capture techniques, and often more data sources, for a higher data capture confidence level. An ultimate configuration might be at a test facility that supports radio and hardwire landline telemetry links, with multiple parallel port storage capacity on its host computer for permanent data storage at real-time data rates. The following system configuration would yield four data sources: three capture real-time, and one in posttest mode. The first link would be identical to the radio telemetry link used in the unconstrained test configuration. A second decommutation system could capture data from the ADAM telemetry port via hardwire/landline in the test umbilical cable, with its output feeding the host computer. The third real-time data capture path could be via a direct link between ADAM's high speed parallel port and the host computer via the umbilical cable. The fourth data source could be a posttest dump of ADAM's memory to the host computer, with or without the DRASS as an intermediate device. The ADAM computer system would save each data set (consisting of four sensor data values) in each of three locations: the telemetry port, the parallel port, and the next available location in its onboard memory system.

## 3.8. PACKAGING AND INTERCONNECTIONS

The ADAM instrumentation design packages all of the instrumentation system within the manikin. The instrumentation fits in both the small and large manikin. Since the small manikin represented the smallest space in which the instrumentation must fit, the design of the circuit card assemblies and wiring harnesses was based on the units fitting within the small manikin. The packaging design was then adapted to the large manikin.

### 3.8.1. Circuit Card Assembly Design

The circuit card assemblies for the instrumentation are designed to fit inside the manikin viscera. The dimensional constraints of the viscera design allowed a maximum printed circuit (PC) board size of 4.426 inches deep by 6.850 inches high. Because of the large quantity of integrated circuits that were required by the design, several techniques were employed to fit the instrumentation electronics on boards of this size.

By analyzing the height of components when mounted on a PC board, a determination was made that there could be a maximum of seven circuit card assemblies (CCA) within the viscera. In order to design all of the circuitry on these boards, certain decisions were made regarding the IC selection and placement. Four PC boards were required to hold all of the digital circuitry, and three PC boards contain all of the analog signal conditioning circuitry. The power distribution board is not included as one of the seven CCAs since it is mounted on the front plate of the viscera.

The high density design of the CCAs was accomplished through the use of the SAFE hybrid micro-circuit, and the use of surface mount components wherever possible to minimize the space required by the components. The PC boards were also designed as multilayer boards, using four or six layers of conductors to make all of the interconnections between components.

Surface-mount components require one-half to two-thirds of the space that a standard integrated circuit package requires on the board. Surface-mount packages also have better lead inductance and capacitance characteristics than standard IC packages. The SAFE hybrid is also a surface-mount device that provided the necessary room savings to allow the analog circuits to fit on only three CCAs.

The IC count was also reduced on the board by the use of Programmable Array Logic (PAL) devices in surface mount packages. These PALs are programmed with the boolean logic equations necessary to generate the required control signals on the CCA. The use of PALs reduced the number of ICs required to generate these control signals.

The use of multiplayer PC boards and surface mount components provided the means to allow high density PC board design to provide a considerable amount of electronics in a very small space. The PC boards are the same size and quantity for both the large and small.

### 3.8.2. Viscera Packaging

The manikin viscera houses the seven CCAs, the power distribution board, and the two mother boards. It is also the focal point for all of the sensor excitation and signal lines. Figure 185 shows the layout of the CCA in the viscera. The seven CCAs of the digital and analog subsystem plug into their respective mother boards. The four digital subsystem CCAs are located on the left side of the manikin spine, and the three analog signal conditioning boards are on the right side of the manikin spine.

The outermost digital board is the memory board. It was located in this position to provide sufficient clearance for all of the components on the top of the PC board and clearance for the backup battery mounted on the solder side of the board. The digital I/O board is located next to the memory board. It was located in this position so that the digital I/O board interface cable could be inserted in the board through a slot in the bottom of the viscera. The A/D board is the innermost board and was located in this position so that the ribbon cable carrying the ADC I/O lines could be connected to the board between the digital I/O board. The board adjacent to the A/D board is the processor board which was placed in the only remaining slot. The right ankle cable for the

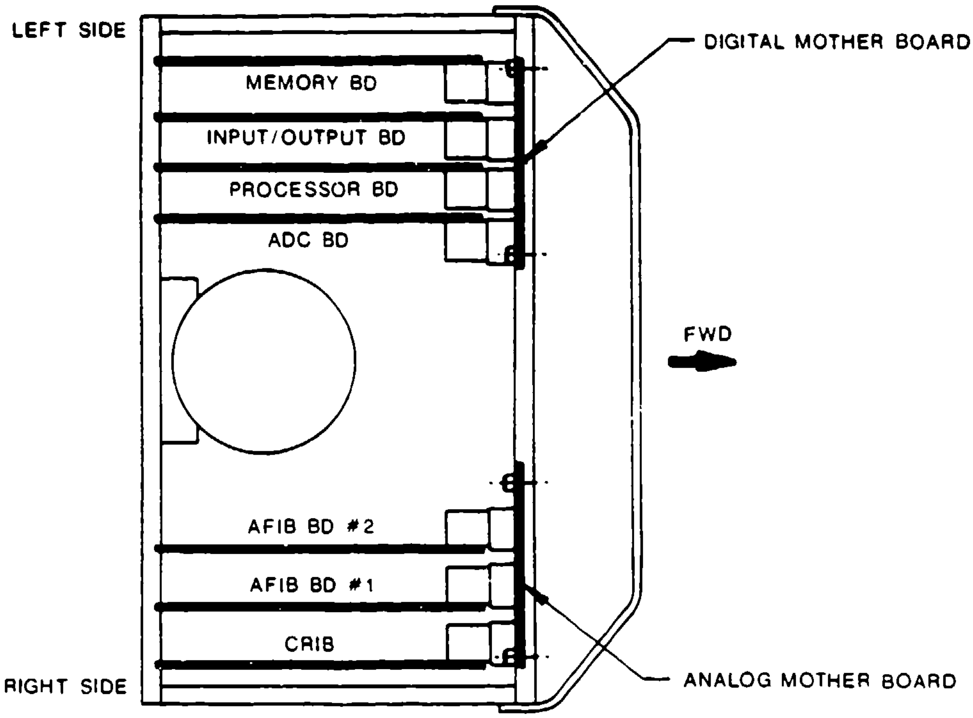



Figure 185. Viscera Packaging (Top View)

manikin which carries the communications and control signals to the processor board plugs into this board through a slot in the bottom of the viscera.

The CRIB is the outermost CCA of the signal conditioning boards. The CRIB was located in this position because it has no components on the solder side of the board. This allowed the CRIB to be located closer to the side of the viscera than the AFIB. AFIB #1 is located next to the CRIB, and AFIB #2 is located next to AFIB #1. These boards were located 0.8 inch from the next board to provide the proper amount of spacing for the components that are mounted on each side of the AFIB PC board. although AFIB #1 and AFIB #2 will fit in either slot, the boards are not interchangeable once they have been calibrated. The reason that the boards cannot be interchanged is that each channel on the boards has been calibrated to a specific sensor. When the boards are swapped, the sensors associated with each channel are different, and the boards would require a complete calibration to be used in the other board's slot.

The orientation of the CCAs was chosen in such a manner as to reduce the exposure of the CCAs to excessive acceleration in their most vulnerable axis. The CCAs were mounted in the viscera with the lane of the CCAs perpendicular to the manikin Y-axis. This exposes the edge of the CCAs to the higher X- and Z-axis accelerations. The CCAs are held in place on three sides with card guides, and on the fourth side by the mother board/daughter board connector pair.
The CCAs and the mother boards were oriented in such a way that a deceleration in the X-axis

would hold the mother board/daughter board connectors together. This would provide an additional margin of safety so there would be no discontinuities between the mother board and daughter board.

The power distribution board is mounted to the front plate of the viscera using eleven 0.25 inch long standoffs. Since the plane of this CCA is perpendicular to the X-axis, a number of standoffs were used to prevent excessive bending of the CCA during high accelerations along the X-axis. Two large cables extend from the power distribution board and exit the viscera. One cable goes to the top of the viscera and has a connector that mates with the battery interconnect cable. This provides a battery "chest connector" to disconnect the batteries from the instrumentation and prevent unnecessary discharge of the batteries. The second cable exits the bottom left of the viscera and connects with the left ankle connector to provide external field power to the instrumentation. The power distribution board also has power connectors that supply the source voltages to the analog and digital mother boards.

The analog and digital mother boards are mounted near the front of the viscera with the planes of the CCAs perpendicular to the X-axis. The boards are mounted at the top and bottom of the viscera. This mounting scheme was deemed adequate since the 160 pin mother board connectors add a significant amount of rigidity to the boards. The component side of the mother board contains the mother board connectors. The solder side of the digital mother board contains the power connector for the digital subsystem, and the solder side of the analog mother board contains the power connector for the analog signal conditioning circuitry. The analog mother board has all of the excitation and signal lines for the manikin sensors and external channels soldered in place on the solder side of the mother board.

The sensor excitation and signal lines were soldered into the mother board because studies indicated there was not enough room in the small viscera to provide connectors on the mother board for these lines. If connectors were used, the wires coming off the connectors were required to make very sharp bends in order to exit the top and bottom of the viscera. Sharp bends in these wires increase the stress on the wires, and the wires are more likely to break. Soldering the wires directly to the mother board allowed the sensor wires to make a gentle bend that put less stress on the wires.

### 3.8.3. Sensor Wiring and Interconnections

The sensor wiring for the manikin was designed with the following goals:

- Connectors that allow removal of the sensor.
- Connectors that allow removal of major manikin subassemblies.
- Wiring that minimized the space required.

The sensor wiring design achieved these goals. There is a connector provided for every sensor on the manikin, and there is a connector provided for each wiring harness on a major limb (head, arm, leg, etc.). These connectors provide a means to maintain the manikin, both electrically and mechanically, without removing major portions of the sensor wiring or having to unsolder the wires from the analog mother board. Because of the space constraints that exist throughout the manikin, connectors and wiring harness designs that minimized the space required were used.

The sensor connectors are four pin strip connectors with a center jackscrew and jackpost. The connector pins are spaced 0.050 inch apart so that the sensor connectors are only 0.30 inch wide, 0.1 inch high, and 0.31 inch long. These connectors have been tested in shock and vibration environments similar to that to which the ADAM is exposed. In-line receptacles were used for most low level sensors (the Denton head/neck load cells have their own connector built in), and right angle PC mount receptacles were used for the position sensors. The mating plug for these connectors was designed into the wiring harness to provide the excitation for the sensor and carry the sensor signal lines to the analog mother board.

These small connectors provide an excellent means to remove the sensor from the manikin for repair, replacement, or recalibration. In most cases, the sensor may be removed by disconnecting the sensor connector and removing the sensor from its mount. The sensor wiring harness does not need to be disturbed.

The connectors that are used at the major limbs of the manikin were required to be as small as possible and provide continuity between connectors in the shock and vibration environment in which the manikin is to be used. a study of all connector systems available was done to find a connector design that would be small, come in a variety of sizes, and function in the environmental conditions to which the manikin is subjected. The study showed only one connector design that would meet these criteria. These connectors were the MDM series by ITT/Cannon. These subminiature D connectors come in standard sizes up to 100 pins and meet the shock and vibration requirements of the manikins since they are commonly used in aircraft and missile systems. These connectors are very small. The pins are located on 0.050 inch centers, and the connectors are purchased with pigtail leads color coded to MIL-STD-681.

The connectors had four conductor, 100 percent, wire braid shield spliced onto these pigtail leads and the four pin strip connector spliced onto the end of the cable. The shields are connected to their assigned pins on the MDM connector and left open at the sensor connector. The mating wiring harnesses for the major limb harnesses are constructed in the same manner, but the cable spliced on these connectors is soldered directly to the analog mother board. The shields are carried through the major limb connectors and grounded on the analog mother board.

There are seven major limb connectors. Four are located on the top of the viscera, and three are located below the viscera in the pelvic region. The four above the viscera are used for the chest harness, head harness, left arm harness, and the right arm harness. The three below the viscera are used to join the pelvic harness, left leg harness, and the right leg harness.

The wiring harnesses are routed along the manikin bones to the sensors from the top and bottom of the viscera. The harnesses are kept away from pinch points on the manikin bones. When a joint or pinch point must be crossed by a harness, it is routed in a way that minimizes the opportunity for the harness to be pinched or cut. The harness is also wrapped in cable dressing where needed to protect the harness when it is inadvertently caught in a pinch point.

The cable dressing serves to protect the harnesses in two ways. The first is to prevent any sharp edges on the bones from cutting directly into the wires, and the second is to provide a slick surface that tends to push the harness out of a pinch point instead of allowing the harness to be cut. Because it is likely in handling and service that the cable dressing will wear, frequent inspection of the cable dressing is necessary to prevent harness damage. The harnesses are tied to the bones using plastic tie wraps or lacing cord. These devices were selected because of their ease of use and low cost. The wire lengths were chosen to minimize the extra cable since the excess cable is difficult to tie down and control the location of the wire.

### 3.8.4. Packaging Achievements

The goal of the instrumentation design was to provide the most advanced and largest manikin instrumentation system in one of the most human-like and strongest physical packages ever designed. The instrumentation was required to fit in both the small and large size manikins without affecting their strength and human-like characteristics. The use of programmable logic integrated circuits, surface mount integrated circuits, a custom hybrid microcircuit, and multilayer PC boards provided the miniaturization required for the PC board packaging. The use of microminiature connectors and the best wire routing techniques possible have provided the ability to sense the reactions of the manikin without impacting the strength and human-like qualities of the manikin.

## Section 4
## CONCLUSIONS AND RECOMMENDATIONS

Increases in high speed and altitude performance of current and planned high performance aircraft and the persistent high rate of fatality and injury associated with the operation of current aircraft are driving research programs to develop better restraint and escape systems. The development of the ADAM was initiated to effectively test and evaluate these systems.

This effort has resulted in the design of two prototype instrumented, anthropomorphic manikins for testing, evaluating, and qualifying high performance aircraft escape systems. The manikins were designed to provide a humanlike reactive live load into the ejection seat and possess realistic dynamics and kinematics due to windblast, impact, vibration, and acceleration forces representative of those encountered during ejection from aircraft. In addition to improved biomechanical response properties, the manikins were designed to have a data acquisition system to measure and record its responses and the data from the escape system.

On the basis of successful analyses and design, it is recommended that a small and large prototype manikin be fabricated for testing and evaluation.

## Appendix A

## ROTRANS

```
C        This program will rotate and translate anatomical axis systems
C          so that one global axis system is formed for the entire body.
C

C        The input data consists of the rotation matrices of each
C          anatomical axis, the joint locations in the respective
C          anatomical axes, and the CG locations.

         INTEGER ZZ,K,COUNT,AB
         REAL M(3,3,20),BB(3,6),POINT3(3,20),POINT(3,20),START(3),AV(3,12)
         DIMENSION A(3,20),B(3,20),CG(3,20),C(3,20),D(3,20),X(3,20)
         DIMENSION POINT2(3,20)
         CHARACTER HEAD*40

         WRITE(5,51)
  51     FORMAT(19X,'***** PROGRAM ROTRANS *****'/,9X,'ARE YOU WORKING
     1   WITH THE SMALL, MEDIUM OR LARGE?'/'    Type a "1" for the small,
     1   "2" for medium and "3" for large')
         ACCEPT*, AB
         WRITE(5,52)
  52     FORMAT(9X,'TYPE IN THE HEADING YOU WISH TO BE SHOWN IN THE
     1   OUTPUT ',/,17x,'(make sure you use single quotes)')
         ACCEPT*, HEAD

         IF (AB.EQ.1) GO TO 6
         IF (AB.EQ.2) GO TO 7
         CALL ASSIGN(4,'INPUT95A.DAT')
         CALL ASSIGN(3,'OUTPUT95A.DAT')
         GO TO 8
   6     CALL ASSIGN(4,'INPUT5A.DAT')
         CALL ASSIGN(3,'OUTPUT5A.DAT')
         GO TO 8
   7     CALL ASSIGN(4,'INPUT50A.DAT')
         CALL ASSIGN(3,'OUTPUT50A.DAT')

   8     READ(4,9)
   9     FORMAT(////)

         READ(4,*) (((M(I,J,K),I=1,3),J=1,3),K=1,20)
         READ(4,*) ((A(I,J),I=1,3),J=1,20)
         READ(4,*) ((B(I,J),I=1,3),J=1,12)
         READ(4,*) ((C(I,J),I=1,3),J=1,6)
         READ(4,*) ((D(I,J),I=1,3),J=1,6)
         READ(4,*) ((CG(I,J),I=1,3),J=1,20)


         COUNT=0
         DO 10 K=1,6
           CALL ROTRAN(K,M,A,B,C,D,CG,COUNT,X)
  10     CONTINUE
         COUNT=1
         DO 20 K=7,13
           CALL ROTRAN(K,M,A,B,C,D,CG,COUNT,X)
  20     CONTINUE
```

```
      DO 30 I=1,3
        X(I,7)=C(I,5)-A(I,7)
        B(I,7)=X(I,7)+B(I,7)
        CG(I,7)=X(I,7)+CG(I,7)
        X(I,9)=D(I,5)-A(I,9)
        B(I,9)=X(I,9)+B(I,9)
        CG(I,9)=X(I,9)+CG(I,9)
        X(I,11)=C(I,3)-A(I,11)
        B(I,11)=X(I,11)+B(I,11)
        CG(I,11)=X(I,11)+CG(I,11)
 30   CONTINUE

      COUNT=2
      DO 40 K=8,12,2
        CALL ROTRAN(K,M,A,B,C,D,CG,COUNT,X)
 40   CONTINUE
      COUNT=3
      DO 50 K=14,20,2
        CALL ROTRAN(K,M,A,B,C,D,CG,COUNT,X)
 50   CONTINUE

      WRITE(3,150) HEAD
150   FORMAT(10X,A40)

      DO 60 I=1,3
        X(I,13)=A(I,1)-A(I,13)
        CG(I,13)=CG(I,13)+X(I,13)
 60   CONTINUE


      WRITE(3,100)
100   FORMAT(20X,'SEGMENT CG',21X,'JOINT LOC',/)
      WRITE(3,200) CG(1,14),CG(2,14),CG(3,14),B(1,6),B(2,6),B(3,6)
200   FORMAT(2X,'1',3X,'HEAD',5X,'(',3F6.1,')',3X,'2',7X,'(',3F6.1,')'
     1 )
      WRITE(3,300) (CG(I,6),I=1,3),(B(I,5),I=1,3)
300   FORMAT(2X,'2',3X,'NECK',5X,'(',3F6.1,')',3X,'3',7X,'(',3F6.1,')'
     1 )
      WRITE(3,400) (CG(I,5),I=1,3),(B(I,4),I=1,3)
400   FORMAT(2X,'3',3X,'THORAX',3X,'(',3F6.1,')',3X,'4',7X,'(',3F6.1,
     1 ')')
      WRITE(3,500) (CG(I,4),I=1,3),(B(I,3),I=1,3)
500   FORMAT(2X,'4',3X,'ABDOMEN',2X,'(',3F6.1,')',3X,'5',7X,'(',3F6.1,
     1 ')')
      WRITE(3,600) (CG(I,3),I=1,3)
600   FORMAT(2X,'5',3X,'PELVIS',3X,'(',3F6.1,')')
      IF(COUNT.EQ.4) GOTO 90
      WRITE(3,700) (CG(I,7),I=1,3),(C(I,5),I=1,3)
700   FORMAT(2X,'6',3X,'RUARM',4X,'(',3F6.1,')',3X,'RSHLDR',2X,'(',3F6
     1 .1,')')
      WRITE(3,800) (CG(I,8),I=1,3),(B(I,7),I=1,3)
800   FORMAT(2X,'7',3X,'RFARM',4X,'(',3F6.1,')',3X,'RELBOW',2X,'(',3F6
     1 .1,')')
      WRITE(3,900) (CG(I,16),I=1,3),(B(I,8),I=1,3)
900   FORMAT(2X,'8',3X,'RHAND',4X,'(',3F6.1,')',3X,'RWRIST',2X,'(',3F6
     1 .1,')')
      WRITE(3,1000) (CG(I,2),I=1,3),(B(I,2),I=1,3)
```

```
1000   FORMAT(2X,'9',3X,'RTHIGH',3X,'(',3F6.1,')',3X,'RHIP',4X,'(',3F6.
     1 1,')')
       WRITE(3,1100) (CG(I,1),I=1,3),(B(I,1),I=1,3)
1100   FORMAT(2X,'10',2X,'RCALF',4X,'(',3F6.1,')',3X,'RKNEE',3X,'(',3F6
     1 .1,')')
       WRITE(3,1200) (CG(I,13),I=1,3),(A(I,1),I=1,3)
1200   FORMAT(2X,'11',2X,'RFOOT',4X,'(',3F6.1,')',3X,'RANKLE',2X,'(',3F
     1 6.1,')')
       WRITE(3,1300) (CG(I,9),I=1,3),(D(I,5),I=1,3)
1300   FORMAT(2X,'12',2X,'LUARM',4X,'(',3F6.1,')',3X,'LSHLDR',2X,'(',3F
     1 6.1,')')
       WRITE(3,1400) (CG(I,10),I=1,3),(B(I,9),I=1,3)
1400   FORMAT(2X,'13',2X,'LFARM',4X,'(',3F6.1,')',3X,'LELBOW',2X,'(',
     1 3F6.1,')')
       WRITE(3,1500) (CG(I,18),I=1,3),(B(I,10),I=1,3)
1500   FORMAT(2X,'14',2X,'LHAND',4X,'(',3F6.1,')',3X,'LWRIST',2X,'(',
     1 3F6.1,')')
       WRITE(3,1600) (CG(I,11),I=1,3),(C(I,3),I=1,3)
1600   FORMAT(2X,'15',2X,'LTHIGH',3X,'(',3F6.1,')',3X,'LHIP',4X,'(',
     1 3F6.1,')')
       WRITE(3,1700) (CG(I,12),I=1,3),(B(I,11),I=1,3)
1700   FORMAT(2X,'16',2X,'LCALF',4X,'(',3F6.1,')',3X,'LKNEE',3X,'(',
     1 3F6.1,')')
       WRITE(3,1800) (CG(I,20),I=1,3),(B(I,12),I=1,3)
1800   FORMAT(2X,'17',2X,'LFOOT',4X,'(',3F6.1,')',3X,'LANKLE',2X,'(',
     1 3F6.1,')',/,/,/)


       DO 70 I=3,6
         CG(2,I)=0
         B(2,I)=0
           DO 70 J=1,3
         BB(J,I)=B(J,I)
 70    CONTINUE

       CG(2,14)=0

       DO 80 I=1,3
         AV(I,1)=(CG(I,7)+CG(I,9))/2
         AV(2,1)=(ABS(CG(2,7))+ABS(CG(2,9)))/2
         AV(I,3)=(CG(I,8)+CG(I,10))/2
         AV(2,3)=(ABS(CG(2,8))+ABS(CG(2,10)))/2
         AV(I,5)=(CG(I,16)+CG(I,18))/2
         AV(2,5)=(ABS(CG(2,16))+ABS(CG(2,18)))/2
         AV(I,7)=(ABS(CG(I,2))+ABS(CG(I,11)))/2
         AV(I,9)=(CG(I,1)+CG(I,12))/2
         AV(2,9)=(ABS(CG(2,1))+ABS(CG(2,12)))/2
         AV(I,11)=(CG(I,13)+CG(I,20))/2
         AV(2,11)=(ABS(CG(2,13))+ABS(CG(2,20)))/2
         AV(I,2)=(D(I,5)+C(I,5))/2
         AV(2,2)=(ABS(D(2,5))+ABS(C(2,5)))/2
         AV(I,4)=(B(I,7)+B(I,9))/2
         AV(2,4)=(ABS(B(2,7))+ABS(B(2,9)))/2
         AV(I,6)=(B(I,8)+B(I,10))/2
         AV(2,6)=(ABS(B(2,8))+ABS(B(2,10)))/2
         AV(I,8)=(ABS(B(I,2))+ABS(C(I,3)))/2
         AV(I,10)=(B(I,11)+B(I,1))/2
         AV(2,10)=(ABS(B(2,11))+ABS(B(2,1)))/2
         AV(I,12)=(A(I,1)+B(I,12))/2
         AV(2,12)=(ABS(A(2,1))+ABS(B(2,12)))/2
 80    CONTINUE
```

```
         COUNT=4
        WRITE(3,2000)
 2000   FORMAT(20X,'FOR A SYMMETRICAL MANIKIN...',/)
         GOTO 60
  90    DO 89 I=1,12
         AV(2,I)=-AV(2,I)
  89    CONTINUE
        WRITE(3,700) (AV(I,1),I=1,3),(AV(I,2),I=1,3)
        WRITE(3,800) (AV(I,3),I=1,3),(AV(I,4),I=1,3)
        WRITE(3,900) (AV(I,5),I=1,3),(AV(I,6),I=1,3)
        WRITE(3,1000) (AV(I,7),I=1,3),(AV(I,8),I=1,3)
        WRITE(3,1100) (AV(I,9),I=1,3),(AV(I,10),I=1,3)
        WRITE(3,1200) (AV(I,11),I=1,3),(AV(I,12),I=1,3)
        IF(COUNT.EQ.4)  THEN
        DO 99 I=1,12
         AV(2,I)=-AV(2,I)
  99    CONTINUE
        END IF
        WRITE(3,1300) (AV(I,1),I=1,3),(AV(I,2),I=1,3)
        WRITE(3,1400) (AV(I,3),I=1,3),(AV(I,4),I=1,3)
        WRITE(3,1500) (AV(I,5),I=1,3),(AV(I,6),I=1,3)
        WRITE(3,1600) (AV(I,7),I=1,3),(AV(I,8),I=1,3)
        WRITE(3,1700) (AV(I,9),I=1,3),(AV(I,10),I=1,3)
        WRITE(3,1800) (AV(I,11),I=1,3),(AV(I,12),I=1,3)


C    ****** This section  utilizes the  DOIT  subroutine ******
C    *** This subroutine will take a point  in  the  global ***
C    *** axis system and transform it back into the segment ***
C    *** anatomical axis system.                            ***


C       POINT3 IS THE VARIABLE THAT CARRIES THE POINTS
C         THAT ARE ORIGINALLY IN A GLOBAL AXIS
C         POINT3(X,3) IS THE ORIGIN FOR TORSO,ABD,PELVIS
C         POINT3(X,4) IS THE Z-AXIS FOR TORSO,ABD,PELVIS
C         POINT3(X,5) IS THE Y-AXIS FOR TORSO,ABD,PELVIS

C         POINT3(X,13) IS THE ORIGIN FOR THE RT FOOT
C         POINT3(X,14) IS THE Z-AXIS FOR THE RT FOOT
C         POINT3(X,15) IS THE Y-AXIS FOR THE RT FOOT

C         POINT3(X,18) IS THE ORIGIN FOR THE LEFT FOOT
C         POINT3(X,19) IS THE Z-AXIS FOR THE LEFT FOOT
C         POINT3(X,20) IS THE Y-AXIS FOR THE LEFT FOOT
```

```
      IF (AB.EQ.2) GO TO 3000
      IF (AB.EQ.3) GO TO 4000
C   THE NEXT 5 DATA LINES ARE DATA FOR THE SMALL MANIKIN.  THESE POINTS
C   WERE TRANSLATED BY THE VECTOR (-.9i + 0j - 2.3k) TO ADJUST FOR THE
C   ANKLE POSITIONING.

      DATA POINT3(1,3)/1.25/,POINT3(2,3)/0.0/,POINT3(3,3)/51.1/
      DATA POINT3(1,4)/1.25/,POINT3(2,4)/0.0/,POINT3(3,4)/36.4/
      DATA POINT3(1,5)/1.25/,POINT3(2,5)/6.88/,POINT3(3,5)/51.1/

      DATA POINT3(1,13)/-0.9/,POINT3(2,13)/-5.2/,POINT3(3,13)/0.1/
      DATA POINT3(1,14)/-0.9/,POINT3(2,14)/-5.2/,POINT3(3,14)/-4.9/
      DATA POINT3(1,15)/-0.9/,POINT3(2,15)/-0.2/,POINT3(3,15)/0.1/

      DATA POINT3(1,18)/-0.9/,POINT3(2,18)/3.1/,POINT3(3,18)/0.1/
      DATA POINT3(1,19)/-0.9/,POINT3(2,19)/3.1/,POINT3(3,19)/-4.9/
      DATA POINT3(1,20)/-0.9/,POINT3(2,20)/8.1/,POINT3(3,20)/0.1/

C     DATA POINT3(1,6)/3.2/,POINT3(2,6)/0.7/,POINT3(3,6)/60.6/
      GO TO 5000

 4000 CONTINUE

C   THE NEXT THREE GROUPS OF DATA LINES FOLLOWING ARE FOR THE LARGE
C     MANIKIN.  THESE POINTS WERE TRANSLATED BY (-.8i + 0j - 2.7k)
C     TO ADJUST FOR THE ANKLE POSITION.

      POINT3(1,3)=-0.5
      POINT3(2,3)=0
      POINT3(3,3)=56.2
      POINT3(1,4)=-0.5
      POINT3(2,4)=0
      POINT3(3,4)=40.7
      POINT3(1,5)=-0.5
      POINT3(2,5)=6.8
      POINT3(3,5)=56.2

      POINT3(1,13)=-0.8
      POINT3(2,13)=-5.3
      POINT3(3,13)=0.1
      POINT3(1,14)=-0.8
      POINT3(2,14)=-5.3
      POINT3(3,14)=-4.9
      POINT3(1,15)=-0.8
      POINT3(2,15)=-0.3
      POINT3(3,15)=0.1

      POINT3(1,18)=-0.8
      POINT3(2,18)=4.8
      POINT3(3,18)=0.1
      POINT3(1,19)=-0.8
      POINT3(2,19)=4.8
      POINT3(3,19)=-4.9
      POINT3(1,20)=-0.8
      POINT3(2,20)=9.8
      POINT3(3,20)=0.1
```

```
C    THESE CALLS ARE USED FOR THE DEFINITION OF THE MECHANICAL AXIS SYSTEMS.
C         CALL DOIT(M,14,BB,5,X,POINT)
C         CALL DOIT(M,6,POINT3,6,X,POINT)
C         CALL DOIT(M,9,C,5,X,POINT)
C         CALL DOIT(M,7,D,5,X,POINT)
C         CALL DOIT(M,10,D,5,X,POINT)
C         CALL DOIT(M,8,C,5,X,POINT)
C         CALL DOIT(M,18,B,9,X,POINT)
C         CALL DOIT(M,16,B,7,X,POINT)
C         CALL DOIT(M,4,POINT3,4,X,POINT)
C         CALL DOIT(M,11,B,2,X,POINT)
C         CALL DOIT(M,2,C,3,X,POINT)
C         CALL DOIT(M,12,C,3,X,POINT)
C         CALL DOIT(M,1,B,2,X,POINT)
C         CALL DOIT(M,20,B,11,X,POINT)
C         CALL DOIT(M,13,B,1,X,POINT)


 5000     CONTINUE
C     THESE CALLS WILL PUT THE REQUIRED POINT3'S INTO THE RESPECTIVE
C           ANATOMICAL AXIS SYSTEMS.


          WRITE(3,2600)
 2600     FORMAT(///5X,'The ORIGIN, Z and Y axis points in the RT FOOT
     1    anat. axis system are:')
          CALL DOIT(M,13,POINT3,13,X,POINT)
          CALL DOIT(M,13,POINT3,14,X,POINT)
          CALL DOIT(M,13,POINT3,15,X,POINT)
          WRITE(3,2700)
 2700     FORMAT(5X,'The ORIGIN, Z and Y axis points in the LEFT FOOT anat.
     1    axis system are:')
          CALL DOIT(M,20,POINT3,18,X,POINT)
          CALL DOIT(M,20,POINT3,19,X,POINT)
          CALL DOIT(M,20,POINT3,20,X,POINT)
          WRITE(3,2100)
 2100     FORMAT(5X,'The ORIGIN, Z and Y axis points in the TORSO anat.
     1    axis system are:')
          CALL DOIT(M,5,POINT3,3,X,POINT)
          CALL DOIT(M,5,POINT3,4,X,POINT)
          CALL DOIT(M,5,POINT3,5,X,POINT)
          WRITE(3,2200)
 2200     FORMAT(5X,'The ORIGIN, Z and Y axis points in the ABDOMEN anat.
     1    axis system are:')
          CALL DOIT(M,4,POINT3,3,X,POINT)
          CALL DOIT(M,4,POINT3,4,X,POINT)
          CALL DOIT(M,4,POINT3,5,X,POINT)
          WRITE(3,2300)
 2300     FORMAT(5X,'The ORIGIN, Z and Y axis points in the PELVIS anat.
     1    axis system are:')
          CALL DOIT(M,3,POINT3,3,X,POINT)
          CALL DOIT(M,3,POINT3,4,X,POINT)
          CALL DOIT(M,3,POINT3,5,X,POINT)
```

```
C         DATA POINT2(1,5)/0.4/,POINT2(2,5)/7.875/,POINT2(3,5)/56.2/
C         DATA POINT2(1,3)/1.5/,POINT2(2,3)/4.0/,POINT2(3,3)/37.2/
C         CALL DOIT(M,5,POINT2,5,X,POINT)
C         CALL DOIT(M,3,POINT2,3,X,POINT)
C         POINT2(1,1)=0.4
C         POINT2(2,1)=6.9
C         POINT2(3,1)=56.1
C         CALL DOIT(M,5,POINT2,1,X,POINT)

C         THE NEXT DATA POINT WILL BE THE CG FOR THE SMALL PELVIS.
C         THIS POINT IS A POINT SHIFTED FROM THE GLOBAL CG DUE TO
C         A SHIFT IN THE HIP FOR A REALISTIC MANIKIN.  IN ORDER TO
C         GET THIS POINT INTO THE ANATOMICAL AXES, IT MUST BE RUN
C         THROUGH ROTRANS.
C         WRITE(3,2400)
C2400     FORMAT(/5X,' The CG for the PELVIS in its anat. axes is')
C         DATA POINT3(1,9)/2.17/,POINT3(2,9)/0.0/,POINT3(3,9)/34.2/
C         CALL DOIT(M,3,POINT3,9,X,POINT)
C         A CHECK IS NOW USED TO ASSURE THE ACCURACY OF THE CG
C         THE SMALL ABDOMEN CG IS TRANSFORMED FROM THE GLOBAL TO THE
C         ANATOMICAL AXIS SYSTEM OF THE ABDOMEN
C         WRITE(3,2500)
C2500     FORMAT(5X,' The point used for checking is the abdomen cg
C    1    and it follows here')
C         DATA POINT3(1,8)/3.2/,POINT3(2,8)/0.0/,POINT3(3,8)/38.3/
C         CALL DOIT(M,4,POINT3,8,X,POINT)
 3000     CONTINUE

          STOP
          END
```

```
      SUBROUTINE ROTRAN(K,ROT,PTA,PTB,PTC,PTD,PCG,KOUNT,X)
      DIMENSION ROTPTA(3,20),ROTPTB(3,12),ROTPTC(3,6),ROTPTD(3,6),
     1     ROT(3,3,20),PTA(3,20),PTB(3,20),PTC(3,20),PTD(3,20),
     2     PCG(3,20),X(3,20),START(3),RTPTCG(3,20)
      INTEGER ZZ,KOUNT
      ZZ=1

      IF(KOUNT.EQ.2)GO TO 55
        GO TO 65
55      DO 60 I=1,3
           ROTPTA(I,K)=PTA(I,K)
           ROTPTB(I,K)=PTB(I,K)
           RTPTCG(I,K)=PCG(I,K)
60      CONTINUE
        GO TO 40
65        CONTINUE

      DO 20 I=1,3
        DO 10 J=1,3
          ROTPTA(I,K)=ROTPTA(I,K)+ROT(I,J,K)*PTA(J,K)
          IF (KOUNT.EQ.3)GO TO 15
          ROTPTB(I,K)=ROTPTB(I,K)+ROT(I,J,K)*PTB(J,K)
          IF (KOUNT.NE.0) GO TO 15
          ROTPTC(I,K)=ROTPTC(I,K)+ROT(I,J,K)*PTC(J,K)
          ROTPTD(I,K)=ROTPTD(I,K)+ROT(I,J,K)*PTD(J,K)
15        CONTINUE
          RTPTCG(I,K)=RTPTCG(I,K)+ROT(I,J,K)*PCG(J,K)
10      CONTINUE
20    CONTINUE


      IF(KOUNT.EQ.1) GO TO 50
      IF(KOUNT.EQ.3) ZZ=8


40    DO 30 I=1,3
        IF(K.EQ.1) GO TO 80
        X(I,K)=PTB(I,K-ZZ)-ROTPTA(I,K)
        IF(KOUNT.EQ.3) GO TO 25
45      PTB(I,K)=ROTPTB(I,K)+X(I,K)
        IF(KOUNT.NE.0) GO TO 25
        PTC(I,K)=ROTPTC(I,K)+X(I,K)
        PTD(I,K)=ROTPTD(I,K)+X(I,K)
25      CONTINUE
        PCG(I,K)=RTPTCG(I,K)+X(I,K)
30    CONTINUE
      RETURN
```

```
50    DO 70 I=1,3
        PTA(I,K)=ROTPTA(I,K)
        IF(K.GE.13) GOTO 75
        PTB(I,K)=ROTPTB(I,K)
75      CONTINUE
        PCG(I,K)=RTPTCG(I,K)
70    CONTINUE
      RETURN

      DATA START(1)/-0.96/,START(2)/-5.29/,START(3)/0.17/
80    X(I,K)=START(I)-ROTPTA(I,K)
      PTA(I,K)=START(I)
      GO TO 45
      END


      SUBROUTINE DOIT(ROTM,SEGNUM,POINT1,PNTNUM,TRANS,POINT3)

C   THIS SUBROUTINE WILL TAKE THE INPUT POINT (POINT1) WHICH IS IN
C   THE "GLOBAL" AXIS SYSTEM AND WILL ROTATE THROUGH ROTM AND TRANS-
C   LATE BY TRANS TO OBTAIN THE POINT (POINT3) IN THE ORIGINAL
C   COORDINATE SYSTEM ( I.E. THE ANATOMICAL AXES. ).

      INTEGER SEGNUM,PNTNUM
      DIMENSION ROTM(3,3,20),POINT1(3,20),TRANS(3,20),POINT3(3,20),
     1 POINT2(3,20),ROTMT(3,3,20)
      K=SEGNUM
      L=PNTNUM

      DO 10 I=1,3
        DO 10 J=1,3
10        ROTMT(I,J,K)=ROTM(J,I,K)

      DO 15 I=1,3
15      POINT2(I,K)=POINT1(I,L)-TRANS(I,K)

      DO 20 I=1,3
20      POINT3(I,K)=ROTMT(I,1,K)*POINT2(1,K)+ROTMT(I,2,K)*POINT2(2,K)
     1  +ROTMT(I,3,K)*POINT2(3,K)

      WRITE(3,100) SEGNUM,POINT3(1,K),POINT3(2,K),POINT3(3,K)
100   FORMAT(10X,'In the anat. system for segment #',I3,
     1 '     =',F6.2,4X,F6.2,5X,F6.2)

      RETURN
      END
```

# Appendix B

## TOTAL2

```
   TOTAL2 WILL DEFINE A MECHANICAL AXIS SYSTEM USING THREE
POINTS KNOWN IN AN ANATOMICAL AXIS SYSTEM.  THESE THREE
POINTS MUST DEFINE THE MECHANICAL ORIGIN, Z-AXIS, AND Y-
AXIS.  THE PROGRAM WILL FIND THE RELATIONSHIP BETWEEN THE
TWO AXES IN THE FORM OF A DISPLACEMENT MATRIX.
   AFTER DEFINING THE TWO AXES, TOTAL2 WILL TRANSFORM ALL
DATA POINTS AND THE PRINCIPAL MOMENTS OF INERTIA FROM THE
ANATOMICAL TO THE MECHANICAL AXES SYSTEM.
```

```
      CHARACTER NAME*41(35),HEADER2*80
      INTEGER NUMLNMK,PL,SEGMENT,A,B,NSUBJ,T,NUMPOINT(1),SETS
      REAL LNDMARK(35,3),COORDS(4),Z,MATRIXT(4,4)
      REAL ATOMMTX(4,4),PTOAMTX(4,4),INERTIAP(4,4),INERTIAA(4,4),
     *INERTIAM(4,4),SEGCGM(3),MASS,INERTIAF(4,4),ANGLE,RAD,NWCRDS(4)
      CHARACTER STUFF*80,HEADER*80
      DATA COORDS(4)/1.0/

5     CONTINUE
      READ (3,190) HEADER
      WRITE (7,191) HEADER
      WRITE (8,191) HEADER
      READ(3,121) NUMPOINT(1)
      IF (NUMPOINT(1).EQ.0) GOTO 1000
      SETS=1
      WRITE (7,19) NUMPOINT(1)
      READ(3,26) (NAME(I),LNDMARK(I,1),LNDMARK(I,2),LNDMARK(I,3),
     &  I=1,NUMPOINT(1))
      WRITE(7,25) (NAME(I),(LNDMARK(I,J),J=1,3),I=1,NUMPOINT(1))
      WRITE (5,191) HEADER
      WRITE (5,19) NUMPOINT(1)
      WRITE (5,1)
      WRITE (7,1)
      WRITE (5,31)
      WRITE(5,26) (NAME(I),(LNDMARK(I,J),J=1,3),I=1,NUMPOINT(1))


     INTSAXIS WILL FIND THE DISPLACEMENT MATRIX--
      CALL INTSAXIS (LNDMARK,MATRIXT)


      WRITE (5,2)
      WRITE(5,32)
```

```
--THE DISPLACEMENT MATRIX IS USED TO TRANSFORM THE DATA POINTS--
   DO 10 M=1,NUMPOINT(1)
      DO 20 K=1,3
20       COORDS(K)=LNDMARK(M,K)
      CALL DISPMULT (MATRIXT,COORDS,NWCRDS)
      CALL WRTCOOR (NAME,NWCRDS,M)
      DO 10 K=1,3
10       LNDMARK(M,K)=NWCRDS(K)
       CALL WRTMATR (MATRIXT)


      READ(3,100) HEADER2
      DO 30 I=1,3
        DO 30 J=1,3
30        ATOMMTX(I,J)=MATRIXT(I,J)
 ----PL IS THE LOCATION OF THE CG IN THE INPUT DATA----
      PL=NUMPOINT(1)

      DO 40 I=1,3
        SEGCGM(I)=LNDMARK(PL,I)
 --INERTIAP IS THE PRINCIPAL MOMENT OF INERTIA TENSOR--
          READ(3,200) (INERTIAP(I,J),J=1,3)
        DO 40 J=1,3
40        INERTIAP(I,J)=INERTIAP(I,J)*12*32.2
      READ(3,400) MASS
      READ(3,400) ANGLE

 ----USING THE SPECIFIED ROTATION ABOUT THE Y-AXIS, THE--
 --ROTATION MATRIX A(AP)(IE PTOAMTX) IS DETERMINED.------
      DO 50 I=1,3
        DO 50 J=1,3
50        PTOAMTX(I,J)=0
      RAD=ANGLE*3.1415927/180
      PTOAMTX(1,1)=COS(RAD)
      PTOAMTX(3,3)=COS(RAD)
      PTOAMTX(1,3)=-SIN(RAD)
      PTOAMTX(3,1)=SIN(RAD)
      PTOAMTX(2,2)=1


 ----ROTATE WILL PERFORM A SIMILARITY TRANSFORMATION ON--
 --AN INERTIAL TENSOR.  HERE IT CALCULATES THE TENSORS---
 --ALONG THE ANATOMICAL AND MECHANICAL AXES (RESP).-------
 ----TRANSLATE WILL TRANSLATE THE TENSOR TO ANOTHER------
 --LOCATION FROM THE SEGMENT CENTER OF GRAVITY.----------

      CALL ROTATE(PTOAMTX,INERTIAP,INERTIAA)
      CALL ROTATE(ATOMMTX,INERTIAA,INERTIAM)
      CALL TRANSLATE(SEGCGM,INERTIAM,MASS,INERTIAF)
```

```
      WRITE(5,450) MASS
      WRITE(5,700)
      DO 60 K=1,3
 60     WRITE(5,500) (INERTIAP(K,J),J=1,3)
      WRITE(5,800)
      DO 70 K=1,3
 70     WRITE(5,500) (INERTIAA(K,J),J=1,3)
      WRITE(5,900)
      DO 80 K=1,3
 80     WRITE(5,500) (INERTIAM(K,J),J=1,3)
      WRITE(5,600)
      DO 90 K=1,3
 90     WRITE(5,500) (INERTIAF(K,J),J=1,3)


190   FORMAT(A80)
121   FORMAT(I3)
19    FORMAT(I3,A77)
25    FORMAT(1X,A41,3F8.2)
26    FORMAT(A41,3F8.2)
1     FORMAT(' POINTS BEFORE TRANSFORMATION ',20X,'INCHES')
2     FORMAT(' POINTS AFTER TRANSFORMATION ',20X,'INCHES')
31    FORMAT(5X,'(IN ANAT. AXIS SYSTEM)',20X,'X',7X,'Y',7X,'Z')
32    FORMAT(5X,'(IN MECH. AXIS SYSTEM)',20X,'X',7X,'Y',7X,'Z')
191   FORMAT(1X,A80)
100   FORMAT(A80)
200   FORMAT(5X,3F10.5)
300   FORMAT(41X,3F8.2).
400   FORMAT(F6.2)
450   FORMAT(/,1X,'THE MASS IN POUNDS =',F6.2)
600   FORMAT(5X,'THE INERTIAL TENSOR AFTER TRANSFORMATION',/,
     * 'AND CENTERED AT THE ORIGIN OF THE MECHANICAL AXES IS')
500   FORMAT(5X,3F12.5)
700   FORMAT(5X,'THE PRINCIPAL INERTIAL TENSOR IS')
800   FORMAT(5X,'THE INERTIAL TENSOR ALONG THE ANATOMICAL AXES IS')
900   FORMAT(5X,'THE INERTIAL TENSOR ALONG THE MECHANICAL AXES IS')

      GO TO 5
1000  CONTINUE
      STOP
      END


      SUBROUTINE TRANSP(A,B)

   --THIS SUBROUTINE WILL RETURN THE TRANSPOSE(B) OF MATRIX A--

      REAL A(4,4),B(4,4)
      DO 10 I=1,4
        DO 10 J=1,4
 10       B(I,J)=A(J,I)
      RETURN
      END
```



BEST AVAILABLE COPY

```
      SUBROUTINE ROTATE(M,ITNSR,RSNTI)

C----THIS SUBROUTINE WILL,THROUGH A SIMILARITY TRANSFORM------
C--ATION, TRANSFORM AN INERTIAL TENSOR INTO ANOTHER AXIS SYSTEM--

      REAL ITNSR(4,4),RSNTI(4,4),M(4,4),MT(4,4),FRST(4,4)
      CALL TRANSP(M,MT)
      CALL MULTMAT(M,ITNSR,FRST)
      CALL MULTMAT(FRST,MT,RSNTI)
      RETURN
      END


      SUBROUTINE TRANSLATE(CG,INERTIAL,MASS,TIALINER)

C----THIS SUBROUTINE WILL TRANSLATE AN INERTIAL TENSOR( IE)--
C--CONSISTING OF MASS MOMENTS OF INERTIA)------------------

      REAL INERTIAL(4,4),CG(3),TIALINER(4,4),MASS,T(3)
      DO 10 I=1,3
        DO 10 J=1,3
10        TIALINER(I,J)=INERTIAL(I,J)

      DO 20 I=1,3
20      T(I)=-CG(I)
      DO 30 I=1,3
        DO 30 J=1,3
          TIALINER(I,J)=INERTIAL(I,J)+MASS*(T(I)*T(J))
30    CONTINUE

      TIALINER(1,1)=INERTIAL(1,1)+MASS*(T(2)**2+T(3)**2)
      TIALINER(2,2)=INERTIAL(2,2)+MASS*(T(3)**2+T(1)**2)
      TIALINER(3,3)=INERTIAL(3,3)+MASS*(T(1)**2+T(2)**2)

      RETURN
      END


      SUBROUTINE MULTMAT(A,B,C)

C---- THIS SUBROUTINE WILL MULTIPLY TWO MATRICES--
C--( A & B ) AND IT WILL RETURN THE RESULT ( C )---

      REAL A(4,4),B(4,4),C(4,4)
      DO 10 I=1,3
        DO 10 J=1,3
10          C(I,J)=A(I,1)*B(1,J)+A(I,2)*B(2,J)+A(I,3)*B(3,J)+A(I,4)*B(4
     * ,J)
      RETURN
      END
```

```
      SUBROUTINE WRTMATR (OUTMATR)

   --THIS SUBROUTINE WILL WRITE OUT A MATRIX--

      REAL OUTMATR(4,4)

      WRITE(5,35)
      WRITE(8,35)
      WRITE(7,35)
      DO 400 I=1,4
         WRITE(5,45) (OUTMATR(I,J),J=1,4)
         WRITE(8,45) (OUTMATR(I,J),J=1,4)
400      WRITE(7,45) (OUTMATR(I,J),J=1,4)
35    FORMAT('     DISPLACEMENT MATRIX  (ANAT. => MECH.)')
45    FORMAT(T5,F10.5,T15,F10.5,T25,F10.5,T35,F10.5)
      RETURN
      END


      SUBROUTINE WRTCOOR (NAME,OUTCOOR,J)

   --THIS SUBROUTINE WILL WRITE OUT COORDINATE LOCATIONS--

      CHARACTER NAME*41(35)
      REAL OUTCOOR(4)
      WRITE (5,66) NAME(J),OUTCOOR(1),OUTCOOR(2),OUTCOOR(3)
      WRITE (7,65) NAME(J),OUTCOOR(1),OUTCOOR(2),OUTCOOR(3)
65    FORMAT (1X,A41,3F8.2)
66    FORMAT(A41,3F8.2)
      RETURN
      END


      SUBROUTINE DISPMULT(A,B,C)

   ----DISPMULT MULTIPLIES THE MATRIX A BY THE VECTOR--
   --B AND RETURNS THE VECTOR C.--

      REAL A(4,4),B(4),C(4)
      C(4)=1.0
      DO 10 I=1,3
10       C(I)=A(I,1)*B(1)+A(I,2)*B(2)+A(I,3)*B(3)+A(I,4)
      RETURN
      END


      SUBROUTINE CROSS (A,B,C)

   ----CROSS COMPUTES THE CROSS PRODUCT OF PARAMETERS A AND B--
   --AND RETURNS THE RESULT IN PARAMETER C.-------------------

      DIMENSION A(3),B(3),C(3)

      C(1) = A(2)*B(3) - A(3)*B(2)
      C(2) = B(1)*A(3) - B(3)*A(1)
      C(3) = A(1)*B(2) - A(2)*B(1)
      RETURN
      END
```

```
      SUBR... INE INTSAXIS(MRK,DISP)

C ----INT... IS WILL PRODUCE THE DISPLACEMENT MATRIX---
C --BETWE.. TWO AXIS SYSTEMS GIVEN THREE POINTS THAT--
C --DEFINE THE NEW AXES IN THE OLD AXIS SYSTEM.-------

      REAL A(3),B(3),C(3),D(3),E(3),F(3),G(3),H,Z(3),MRK(35,3),
     * FDZ(4,4),ND(3),NF(3),NZ(3),ZDF(4,4),DISP(4,4)
     + ,ORIG(3)
      INTEGER ZERO(3)

      DO 101 I=1,4
         DO 101 J=1,4
         S=0.
         IF(I.EQ.J) S=1.
101      FDZ(I,J)=S

      DO 100 I=1,3
         D(I)=MRK(2,I)-MRK(1,I)
100      E(I)=MRK(3,I)-MRK(1,I)

      CALL NORM (D,ND)

      H=DOT(E,ND)

      DO 200 I=1,3
200      G(I)=H*ND(I)

      DO 300 I=1,3
300      F(I)=E(I)-G(I)

      CALL NORM (F,NF)

      CALL CROSS (NF,ND,Z)

      CALL NORM (Z,NZ)


      DO 500 I=1,3
         ORIG(I)=MRK(1,I)
         ZERO(I)=0
         FDZ(I,1)=NZ(I)
         FDZ(I,2)=NF(I)
500      FDZ(I,3)=ND(I)

      CALL TRANSP (FDZ,ZDF)
      CALL DISPMAT (ZDF,ORIG,ZERO,DISP)
      RETURN
      END
```

```
      SUBROUTINE DISPMAT (R,P0,P1,D)

    --COMPUTES DISPLACEMENT MATRIX D FOR A ROTATION R--
    ----AND A TRANSLATION FROM P0 TO P1.--------------

        ARGUMENTS:
          D(4,4): DISPLACEMENT MATRIX TO BE COMPUTED.
          R(4,4): ROTATION MATRIX
          P0(3): VECTOR
          P1(3): VECTOR

          R, P0 AND P1 ARE ALL IN THE GLOBAL COORDINATE SYSTEM

      DIMENSION  D(4,4),R(4,4),P0(3),P1(3)
      DO 1 I=1,3
      DO 2 J=1,3
      D(I,J)=R(I,J)
      D(I,4)=P1(I)-(R(I,1)*P0(1)+R(I,2)*P0(2)+R(I,3)*P0(3))
      D(4,I)=0.0
      D(4,4)=1.0
      RETURN
      END


      FUNCTION DOT (A,B)

   ----FUNCTION DOT RETURNS THE DOT PRODUCT OF THE--
   --TWO THREE DIMENSIONAL VECTORS A AND B.---------

      REAL A(3),B(3)
      DOT = 0.
      DO 100 I=1,3
100      DOT = DOT + A(I)*B(I)
      RETURN
      END


      SUBROUTINE NORM (A,B)

   --NORM NORMALIZES THE THREE DIMENSIONAL VECTOR C.--

      REAL A(3),B(3)
      SIZE = 0.
      DO 50 I=1,3
50       SIZE = SIZE + A(I)**2
      SIZE = SQRT(SIZE)
      DO 100 I=1,3
100      B(I) = A(I)/SIZE
      RETURN
      END
```

# Appendix C

## MASSPR

```
      REAL M(3,40),LX(3,40),LY(3,40),LZ(3,40),L(3,40),ID(3,40),IX(3,40)
     /,IY(3,40),IZ(3,40),D(3,40),OD(3,40),AX(3,40),CGX(3,40),CGY(3,40)
      INTEGER DP,DSS,S,SS,P,QQ,X,FLAG,GSS,GP(3),ST(3,40)
      CHARACTER SEGNAM*9,SUBNAM(3)*10
      COMMON PI,D,M,LX,LY,LZ,L,OD,ID,AX,CGX,CGY,CGZ(3,40),IX,IY,IZ
     /,SSIX(3),SSIY(3),SSIZ(3),SSM(3),SSCGX(3),SSCGY(3),SSCGZ(3)
     /,SIX,SIY,SIZ,SM,SCGX,SCGY,SCGZ,S,SS,P,FLAG,X,GSS,GP,ST,DSS,DP
      PI=3.1415927
      READ (4,175)
175   FORMAT(//////////////)
150 READ (4,200)S,SEGNAM,GSS
350 DO 445 I=1,GSS
      READ (4,400)SS,SUBNAM(SS),GP(SS)
      DO 405 J=1,GP(SS)
      READ (4,410)P,ST(SS,P),D(SS,P),LX(SS,P),LY(SS,P),LZ(SS,P)
      READ (4,420) L(SS,P),OD(SS,P),ID(SS,P),AX(SS,P)
      READ (4,430) CGX(SS,P),CGY(SS,P),CGZ(SS,P)
      READ (4,440) M(SS,P),IX(SS,P),IY(SS,P),IZ(SS,P)
405 CONTINUE
      READ(4,*)
      READ (4,430)SSCGX(SS),SSCGY(SS),SSCGZ(SS)
      READ (4,440)SSM(SS),SSIX(SS),SSIY(SS),SSIZ(SS)
445 CONTINUE
245 WRITE (5,250)SEGNAM
250 FORMAT (' THE SEGMENT YOU WILL BE EDITING IS THE ',A9)
275 WRITE(5,300)
300 FORMAT (' WHAT SUBSEGMENT WOULD YOU LIKE TO USE?')
      ACCEPT*,DSS
      IF (DSS.LE.GSS)GOTO 450
      WRITE (5,325)
325 FORMAT (' THERE ARE NOT THAT MANY SUBSEGMENTS AVAILABLE')
      GOTO 275
450 WRITE (5,460)SUBNAME(DSS)
460 FORMAT (' THE SUBSEGMENT YOU WILL BE WORKING WITH IS THE ',A10)
500 WRITE (5,550)
550 FORMAT (' WHICH PART NUMBER WOULD YOU LIKE TO ENTER DATA FOR?')
      ACCEPT*,DP
      IF (DP.LE.GP(DSS))GOTO 800
      WRITE (5,600)
600 FORMAT (' THIS IS A NEW PART' )
      GP(DSS)=DP
800 WRITE (5,850)DP,SUBNAME(DSS),SEGNAM
850 FORMAT (' IS PART ',I2,' OF THE ',A10,' IN THE ',A9/,' A BOX(1)
     /, CYLINDER(2), OR OTHER(3)?')
      ACCEPT*,ST(DSS,DP)
      IF (ST(DSS,DP).EQ.1)GOTO 1000
      IF (ST(DSS,DP).EQ.2)GOTO 2000
      IF (ST(DSS,DP).EQ.3)GOTO 3000
      WRITE(5,900)
900 FORMAT (' INVALID PART TYPE')
      GOTO 800
1000 WRITE (5,1050)
1050 FORMAT (' WHAT IS THE LENGTH IN THE "X" DIRECTION? (INCHES)')
      ACCEPT*,LX(DSS,DP)
      WRITE (5,1100)
```

```
1100 FORMAT (' WHAT IS THE LENGTH IN THE "Y" DIRECTION? (INCHES)')
     ACCEPT*, LY(DSS,DP)
     WRITE (5,1150)
1150 FORMAT (' WHAT IS THE LENGTH IN THE "Z" DIRECTION? (INCHES)')
     ACCEPT*,LZ(DSS,DP)
     L(DSS,DP)=0
     ID(DSS,DP)=0
     OD(DSS,DP)=0
     AX(DSS,DP)=0
1175 WRITE (5,1200)
1200 FORMAT (' WHAT IS THE DENSITY OF THE PART?')
     ACCEPT*,D(DSS,DP)
     WRITE (5,1250)
1250 FORMAT (' WHAT IS THE CENTER OF GRAVITY IN THE MECHANICAL AXIS
    /SYSTEM? (X,Y,Z)')
     ACCEPT*, CGX(DSS,DP),CGY(DSS,DP),CGZ(DSS,DP)
1300 WRITE(5,1400)
1400 FORMAT (' IS THIS INFORMATION CORRECT? YES(1),NO(2) ')
     ACCEPT*,X
     IF (X.EQ.2)GOTO 800
     IF (ST(DSS,DP).EQ.3) GOTO 4000
     CALL MASS
     CALL MOI
     IF (S.EQ.-1)GOTO 5000
     GOTO 4000
2000 WRITE (5,2050)
2050 FORMAT (' WHAT IS THE CYLINDER LENGTH?')
     ACCEPT*, L(DSS,DP)
     WRITE (5,2100)
2100 FORMAT (' WHAT IS THE OUTER DIAMETER?')
     ACCEPT*, OD(DSS,DP)
     WRITE (5,2150)
2150 FORMAT (' WHAT IS THE INNER DIAMETER?')
     ACCEPT*, ID(DSS,DP)
     WRITE (5,2200)
2200 FORMAT (' WHAT IS THE CENTROID AXIS? X(1),Y(2),Z(3)')
     ACCEPT*,AX(DSS,DP)
     LX(DSS,DP)=0
     LY(DSS,DP)=0
     LZ(DSS,DP)=0
     GOTO 1175
3000 WRITE (5,3150)
3150 FORMAT (' WHAT ARE THE MOIs - Ix,Iy,Iz?')
     ACCEPT*,IX(DSS,DP),IY(DSS,DP),IZ(DSS,DP)
     WRITE (5,3200)
3200 FORMAT (' WHAT IS THE MASS OF THE PART? (POUNDS)')
     ACCEPT*,M(DSS,DP)
     LX(DSS,DP)=0
     LY(DSS,DP)=0
     LZ(DSS,DP)=0
     L(DSS,DP)=0
     OD(DSS,DP)=0
     ID(DSS,DP)=0
     AX(DSS,DP,=0
     GOTO 1175
```

```
4000 WRITE (5,4250)
4250 FORMAT (4X,'PART TOTALS')
       WRITE (5,4300) CGX(DSS,DP),CGY(DSS,DP),CGZ(DSS,DP)
       WRITE (5,4350) M(DSS,DP)
       WRITE (5,4400) IX(DSS,DP),IY(DSS,DP),IZ(DSS,DP)
       WRITE (5,4275)

4275 FORMAT(' ANY MORE PARTS TO CHANGE IN THIS SEGMENT? YES(1),NO(2)')
       ACCEPT*,X
       IF (X.EQ.1) GOTO 500
4200 FLAG=0
       CALL MASTOT
       CALL CGTOT
       CALL MOITOT
       WRITE (5,4450)SUBNAM(DSS)
4450 FORMAT (4X,A10, 'TOTALS')
       WRITE (5,4300) SSCGX(DSS),SSCGY(DSS),SSCGZ(DSS)
       WRITE (5,4350) SSM(DSS)
       WRITE (5,4400) SSIX(DSS),SSIY(DSS),SSIZ(DSS)
       WRITE(5,4475)
4475 FORMAT(' ARE YOU FINISHED? YES(1),NO(2)')
       ACCEPT*,X
       IF(X.EQ.2) GOTO 275
4100 FLAG=1
       CALL MASTOT
       CALL CGTOT
       CALL MOITOT
       WRITE (5,4500)SEGNAM
       WRITE (5,4300)SCGX,SCGY,SCGZ
4300 FORMAT (5X,'THE CG IS',F6.2,4X,F6.2,4X,F6.2,2X,'INCHES')
       WRITE (5,4350)SM
4350 FORMAT (5X,'THE MASS IS ',F5.2,' LBS')
       WRITE (5,4400)SIX,SIY,SIZ
4400 FORMAT (5X,'THE MOIS ARE  ',F9.3,2X,F9.3,2X,F9.3,'  LB-INSQ')
4500 FORMAT (4X,A9, 'TOTALS')
       READ (2,175)
     WRITE (2,200)S,SEGNAM,QSS
     DO 4511 SS=1,QSS
       WRITE (2,400)SS,SUBNAM(SS),QP(SS)
       DO 4522 P=1,QP(SS)
       WRITE (2,410)P,ST(SS,P),D(SS,P),LX(SS,P),LY(SS,P),LZ(SS,P)
       WRITE (2,420) L(SS,P),OD(SS,P),ID(SS,P),AX(SS,P)
       WRITE (2,430) CGX(SS,P),CGY(SS,P),CGZ(SS,P)
       WRITE (2,440) M(SS,P),IX(SS,P),IY(SS,P),IZ(SS,P)
4522 CONTINUE
       WRITE(2,4150)
       WRITE (2,430)SSCGX(SS),SSCGY(SS),SSCGZ(SS)
       WRITE (2,440)SSM(SS),SSIX(SS),SSIY(SS),SSIZ(SS)
4511 CONTINUE
       WRITE (2,4225)
4225 FORMAT(4X,'SEGMENT TOTALS')
       WRITE (2,430) SCGX,SCGY,SCGZ
       WRITE (2,440) SM,SIX,SIY,SIZ
       WRITE (2,*)
4150 FORMAT (19X,'SUBSEGMENT TOTALS')
```

```
 200 FORMAT (1X, I2, 1X, A9, 1X, I2)
 400 FORMAT (16X, I2, 1X, A10, 1X, I2)
 410 FORMAT (33X, I2, 1X, I2, 1X, F6. 4, 3X, F5. 2, 5X, F5. 2, 5X, F5. 2)
 420 FORMAT (48X, F5. 2, 5X, F5. 2, 5X, F5. 2, 2X, F3. 1)
 430 FORMAT (47X, F6. 2, 4X, F6. 2, 4X, F6. 2)
 440 FORMAT (39X, F5. 2, 1X, F9. 3, 1X, F9. 3, 1X, F9. 3/)
5000 STOP
      END
      SUBROUTINE MASS
      REAL M(3, 40), LX(3, 40), LY(3, 40), LZ(3, 40), L(3, 40), ID(3, 40), IX(3, 40)
    /, IY(3, 40), IZ(3, 40), D(3, 40), OD(3, 40), AX(3, 40), CGX(3, 40), CGY(3, 40)
      INTEGER DP, DSS, S, SS, P, GG, X, FLAG, GSS, GP(3), ST(3, 40)
      CHARACTER SEGNAM*9, SUBNAM(3)*10
      COMMON PI, D, M, LX, LY, LZ, L, OD, ID, AX, CGX, CGY, CGZ(3, 40), IX, IY, IZ
    /, SSIX(3), SSIY(3), SSIZ(3), SSM(3), SSCGX(3), SSCGY(3), SSCGZ(3)
    /, SIX, SIY, SIZ, SM, SCGX, SCGY, SCGZ, S, SS, P, FLAG, X, GSS, GP, ST, DSS, DP
      IF (ST(DSS, DP). EQ. 2) GO TO 500
      M(DSS, DP)=LX(DSS, DP)*LY(DSS, DP)*LZ(DSS, DP)*D(DSS, DP)
      RETURN
 500 M(DSS, DP)=L(DSS, DP)*PI*D(DSS, DP)/4*(OD(DSS, DP)**2-ID(DSS, DP)**2)
     RETURN
      END
      SUBROUTINE MOI
      REAL M(3, 40), LX(3, 40), LY(3, 40), LZ(3, 40), L(3, 40), ID(3, 40), IX(3, 40)
    /, IY(3, 40), IZ(3, 40), D(3, 40), OD(3, 40), AX(3, 40), CGX(3, 40), CGY(3, 40)
      INTEGER DP, DSS, S, SS, P, GG, X, FLAG, GSS, GP(3), ST(3, 40)
      CHARACTER SEGNAM*9, SUBNAM(3)*10
      COMMON PI, D, M, LX, LY, LZ, L, OD, ID, AX, CGX, CGY, CGZ(3, 40), IX, IY, IZ
    /, SSIX(3), SSIY(3), SSIZ(3), SSM(3), SSCGX(3), SSCGY(3), SSCGZ(3)
    /, SIX, SIY, SIZ, SM, SCGX, SCGY, SCGZ, S, SS, P, FLAG, X, GSS, GP, ST, DSS, DP
      IF (ST(DSS, DP). EQ. 2) GOTO 500
      IX(DSS, DP)=M(DSS, DP)*(LY(DSS, DP)**2+LZ(DSS, DP)**2)/12
      IY(DSS, DP)=M(DSS, DP)*(LX(DSS, DP)**2+LZ(DSS, DP)**2)/12
      IZ(DSS, DP)=M(DSS, DP)*(LX(DSS, DP)**2+LY(DSS, DP)**2)/12
      RETURN
 500 IF (ID(DSS, DP). GT. 0)GOTO 600
      XX=M(DSS, DP)*OD(DSS, DP)**2/8
      YY=M(DSS, DP)*(3*OD(DSS, DP)**2+4*L(DSS, DP)**2)/48
      GOTO 700
 600 XX=M(DSS, DP)*(OD(DSS, DP)**2+ID(DSS, DP)**2)/8
     YY=M(DSS, DP)*(3*OD(DSS, DP)**2+3*ID(DSS, DP)**2+4*L(DSS, DP)**2)/48
 700 IF (AX(DSS, DP). EQ. 1)GOTO 710
      IF (AX(DSS, DP). EQ. 2)GOTO 720
      IF (AX(DSS, DP). EQ. 3)GOTO 730
      WRITE (5, 705)
 705 FORMAT ('INVALID AXIS NUMBER FOR CYLINDER')
      S=-1
      RETURN
 710 IX(DSS, DP)=XX
      IY(DSS, DP)=YY
      IZ(DSS, DP)=YY
      RETURN
 720 IX(DSS, DP)=YY
      IY(DSS, DP)=XX
      IZ(DSS, DP)=YY
      RETURN
```

```
730 IX(DSS,DP)=YY
      IY(DSS,DP)=YY
      IZ(DSS,DP)=XX
      RETURN
      END
      SUBROUTINE MASTOT
      REAL M(3,40),LX(3,40),LY(3,40),LZ(3,40),L(3,40),ID(3,40),IX(3,40)
     /,IY(3,40),IZ(3,40),D(3,40),OD(3,40),AX(3,40),CGX(3,40),CGY(3,40)
      INTEGER DP,DSS,S,SS,P,GG,X,FLAG,GSS,GP(3),ST(3,40)
      CHARACTER SEGNAM*9,SUBNAM(3)*10
      COMMON PI,D,M,LX,LY,LZ,L,OD,ID,AX,CGX,CGY,CGZ(3,40),IX,IY,IZ
     /,SSIX(3),SSIY(3),SSIZ(3),SSM(3),SSCGX(3),SSCGY(3),SSCGZ(3)
     /,SIX,SIY,SIZ,SM,SCGX,SCGY,SCGZ,S,SS,P,FLAG,X,GSS,GP,ST,DSS,DP
      IF (FLAG.EQ.1) GOTO 200
      SSM(DSS)=0
      DO 100 X=1,GP(DSS)
      SSM(DSS)=SSM(DSS)+M(DSS,X)
100 CONTINUE
      RETURN
200 SM=0
      DO 300 X=1,GSS
      SM=SM+SSM(X)
300 CONTINUE
      RETURN
      END
      SUBROUTINE CGTOT
      REAL M(3,40),LX(3,40),LY(3,40),LZ(3,40),L(3,40),ID(3,40),IX(3,40)
     /,IY(3,40),IZ(3,40),D(3,40),OD(3,40),AX(3,40),CGX(3,40),CGY(3,40)
      INTEGER DP,DSS,S,SS,P,GG,X,FLAG,GSS,GP(3),ST(3,40)
      CHARACTER SEGNAM*9,SUBNAM(3)*10
      COMMON PI,D,M,LX,LY,LZ,L,OD,ID,AX,CGX,CGY,CGZ(3,40),IX,IY,IZ
     /,SSIX(3),SSIY(3),SSIZ(3),SSM(3),SSCGX(3),SSCGY(3),SSCGZ(3)
     /,SIX,SIY,SIZ,SM,SCGX,SCGY,SCGZ,S,SS,P,FLAG,X,GSS,GP,ST,DSS,DP
      IF (FLAG.EQ.1)GOTO 200
      SSCGX(DSS)=0
      SSCGY(DSS)=0
      SSCGZ(DSS)=0
      DO 100 X=1,GP(DSS)
      SSCGX(DSS)=SSCGX(DSS)+M(DSS,X)*CGX(DSS,X)
      SSCGY(DSS)=SSCGY(DSS)+M(DSS,X)*CGY(DSS,X)
      SSCGZ(DSS)=SSCGZ(DSS)+M(DSS,X)*CGZ(DSS,X)
100 CONTINUE
      SSCGX(DSS)=SSCGX(DSS)/SSM(DSS)
      SSCGY(DSS)=SSCGY(DSS)/SSM(DSS)
      SSCGZ(DSS)=SSCGZ(DSS)/SSM(DSS)
      RETURN
200 SCGX=0
      SCGY=0
      SCGZ=0
      DO 300 X=1,GSS
      SCGX=SCGX+SSM(X)*SSCGX(X)
      SCGY=SCGY+SSM(X)*SSCGY(X)
      SCGZ=SCGZ+SSM(X)*SSCGZ(X)
300 CONTINUE
      SCGX=SCGX/SM
```

```
      SCGY=SCGY/SM
      SCGZ=SCGZ/SM
      RETURN
      END
      SUBROUTINE MOITOT
      REAL M(3,40),LX(3,40),LY(3,40),LZ(3,40),L(3,40),ID(3,40),IX(3,40)
     /,IY(3,40),IZ(3,40),D(3,40),OD(3,40),AX(3,40),CGX(3,40),CGY(3,40)
      INTEGER DP,DSS,S,SS,P,GG,X,FLAG,GSS,GP(3),ST(3,40)
      CHARACTER SEGNAM*9,SUBNAM(3)*10
      COMMON PI,D,M,LX,LY,LZ,L,OD,ID,AX,CGX,CGY,CGZ(3,40),IX,IY,IZ
     /,SSIX(3),SSIY(3),SSIZ(3),SSM(3),SSCGX(3),SSCGY(3),SSCGZ(3)
     /,SIX,SIY,SIZ,SM,SCGX,SCGY,SCGZ,S,SS,P,FLAG,X,GSS,GP,ST,DSS,DP
      IF (FLAG.EQ.1)GOTO 200
      SSIX(DSS)=0
      SSIY(DSS)=0
      SSIZ(DSS)=0
      DO 100 X=1,GP(DSS)
      SSIX(DSS)=SSIX(DSS)+IX(DSS,X)+M(DSS,X)*((CGY(DSS,X)-SSCGY
     /(DSS))**2+(CGZ(DSS,X)-SSCGZ(DSS))**2)
      SSIY(DSS)=SSIY(DSS)+IY(DSS,X)+M(DSS,X)*((CGX(DSS,X)
     /-SSCGX(DSS))**2+(CGZ(DSS,X)-SSCGZ(DSS))**2)
      SSIZ(DSS)=SSIZ(DSS)+IZ(DSS,X)+M(DSS,X)*((CGX(DSS,X)-SSCGX(DSS))**2
     /+(CGY(DSS,X)-SSCGY(DSS))**2)
100   CONTINUE
      RETURN
200   DO 300 X=1,GSS
      SIX=SIX+SSIX(X)+SSM(X)*((SSCGY(X)-SCGY)**2+(SSCGZ(X)-SCGZ)**2)
      SIY=SIY+SSIY(X)+SSM(X)*((SSCGX(X)-SCGX)**2+(SSCGZ(X)-SCGZ)**2)
      SIZ=SIZ+SSIZ(X)+SSM(X)*((SSCGX(X)-SCGX)**2+(SSCGY(X)-SCGY)**2)
300   CONTINUE
      RETURN
      END
```

# Appendix D
## BACK5

```
C        BACK5 WILL USE THE COSINE MATRIX, FOUND IN TOTAL2, BETWEEN
C     THE ANATOMICAL AND MECHANICAL AXIS SYSTEMS TO TRANSFORM THE
C     DATA FOR THE SMALL AND LARGE ADAM DESIGNS.  THE DISPLACEMENTS
C     ARE UNIQUE TO THE SIZE AND WILL BE DEVELOPED WITHIN THIS
C     PROGRAM FOR EACH SIZE.  THE CALC MECH DATA WILL BE TRANSFORMED
C     INTO THE ANALYTICAL ANAT AXIS SYSTEMS.
C        AFTER TRANSFORMING THE DATA, BACK5 WILL TRANSFORM THE
C     MOMENTS OF INERTIA FROM THE MECHANICAL TO THE
C     ANATOMICAL AXES SYSTEM.  IT WILL NOT TRANSLATE THE DATA.
C
C
      CHARACTER NAME*41(35),HEADER2*80
      INTEGER NUMLNMK,PL,SEGMENT,A,B,NSUBJ,T,NUMPOINT,SETS
      REAL LNDMARK(35,3),COORDS(4),Z,M3(4,4),MATRIXT(4,4),NWCRDS(4),
     +     ATOMMTX(4,4),PTOAMTX(4,4),INERTIAP(4,4),INERTIAA(4,4),
     +     INERTIAM(4,4),SEGCGM(3),MASS,INERTIAF(4,4),ANGLE,RAD,
     +     POINT1(3),ZERO(3),NEGMAS,MTOAMTX(4,4),ATOPMTX(4,4),
     +     MATRIT(4,4)
      CHARACTER STUFF*80,HEADER*80
      DATA COORDS(4)/1.0/

 5    CONTINUE
      WRITE(5,6)
      READ (3,190) HEADER
      WRITE (7,191) HEADER
      READ(3,121) NUMPOINT
      IF (NUMPOINT.EQ.0) GOTO 1000
      SETS=1
      WRITE (7,19) NUMPOINT
      READ(3,26) (NAME(I),LNDMARK(I,1),LNDMARK(I,2),LNDMARK(I,3),
     +           I=1,NUMPOINT)
      WRITE(7,25) (NAME(I),(LNDMARK(I,J),J=1,3),I=1,NUMPOINT)
      WRITE (5,191) HEADER
      WRITE (5,19) NUMPOINT
      WRITE (5,1)
      WRITE (7,1)
      WRITE (5,32)
      WRITE(5,25) (NAME(I),(LNDMARK(I,J),J=1,3),I=1,NUMPOINT)



      DO 15 I=1,4
 15     READ(3,250) (MATRIXT(I,J),J=1,4)
        CALL TRANSP(MATRIXT,MATRIT)
      WRITE (5,2)
      WRITE(5,31)

      DO 17 I=1,3
        POINT1(I)=LNDMARK(1,I)
 17     ZERO(I)=0

      CALL DISPMAT(MATRIT,POINT1,ZERO,MATRIT)
```

```
C   --THE DISPLACEMENT MATRIX IS USED TO TRANSFORM THE DATA POINTS--
C   --FROM THE MECH TO THE ANAT AXIS SYSTEMS-----------------------
        DO 10 M=1,NUMPOINT
           DO 20 K=1,3
 20           COORDS(K)=LNDMARK(M,K)
           CALL DISPMULT (MATRIT,COORDS,NWCRDS)
           CALL WRTCOOR (NAME,NWCRDS,M)
           DO 10 K=1,3
 10           LNDMARK(M,K)=NWCRDS(K)
            CALL WRTMATR (MATRIT)



       READ(3,100) HEADER2
       DO 30 I=1,3
         DO 30 J=1,3
   30      MTOAMTX(I,J)=MATRIT(I,J)
C   ----PL IS THE LOCATION OF THE CG IN THE INPUT DATA----
       PL=NUMPOINT

       DO 40 I=1,3
         SEGCGM(I)=LNDMARK(PL,I)
C   --INERTIAF IS THE CAL'D OR SPEC MOI TENSOR------------
   40      READ(3,200) (INERTIAM(I,J),J=1,3)
       READ(3,400) MASS
       READ(3,400) ANGLE

C   ----USING THE SPECIFIED ROTATION ABOUT THE Y-AXIS, THE--
C   --ROTATION MATRIX A(AP)(IE PTOAMTX) IS DETERMINED.------
       DO 50 I=1,3
         DO 50 J=1,3
   50      PTOAMTX(I,J)=0
       RAD=ANGLE*3.1415927/180
       PTOAMTX(1,1)=COS(RAD)
       PTOAMTX(3,3)=COS(RAD)
       PTOAMTX(1,3)=-SIN(RAD)
       PTOAMTX(3,1)=SIN(RAD)
       PTOAMTX(2,2)=1

       NEGMAS=-1*MASS
       CALL TRANSP(PTOAMTX,ATOPMTX)

C   ----ROTATE WILL PERFORM A SIMILARITY TRANSFORMATION ON--
C   --AN INERTIAL TENSOR.  HERE IT CALCULATES THE TENSORS--
C   --ALONG THE ANATOMICAL AND MECHANICAL AXES (RESP).------
C   ----TRANSLATE WILL TRANSLATE THE TENSOR TO ANOTHER------
C   --LOCATION FROM THE SEGMENT CENTER OF GRAVITY.----------
C
       CALL ROTATE(MTOAMTX,INERTIAM,INERTIAA)
       DO 55 I=1,3
          DO 55 J=1,3
             INERTIAA(I,J)=INERTIAA(I,J)/(12*32.2)
  55   CONTINUE
       CALL ROTATE(ATOPMTX,INERTIAA,INERTIAP)

       WRITE(5,450) MASS
       WRITE(5,900)
       DO 70 K=1,3
   70    WRITE(5,500) (INERTIAM(K,J),J=1,3)
       WRITE(5,800)
       DO 80 K=1,3
   80    WRITE(5,500) (INERTIAA(K,J),J=1,3)
       WRITE(5,700)
       DO 90 K=1,3
```

```
190   FORMAT(A80)
121   FORMAT(I3)
19    FORMAT(I3,A77)
25    FORMAT(2X,A41,3F8.2)
26    FORMAT(A41,3F8.2)
1     FORMAT(' POINTS BEFORE TRANSFORMATION ',20X,'INCHES')
2     FORMAT(' POINTS AFTER TRANSFORMATION ',20X,'INCHES')
6     FORMAT('1  ',/,/,/,/,/)
31    FORMAT(5X,'(IN ANAT. AXIS SYSTEM)',20X,'X',7X,'Y',7X,'Z')
32    FORMAT(5X,'(IN MECH. AXIS SYSTEM)',20X,'X',7X,'Y',7X,'Z')
191   FORMAT(1X,A80)
100   FORMAT(A80)
200   FORMAT(5X,3F10.5)
250   FORMAT(5X,4F10.5)
300   FORMAT(41X,3F8.2)
400   FORMAT(F6.2)
450   FORMAT(/,1X,'THE WEIGHT IN POUNDS =',F6.2)
600   FORMAT(5X,'THE INERTIAL TENSOR AFTER TRANSFORMATION',/,
   +          ' AND CENTERED AT THE ORIGIN OF THE MECHANICAL AXES IS')
500   FORMAT(5X,3F12.5)
700   FORMAT(5X,'THE PRINCIPAL INERTIAL TENSOR IS')
800   FORMAT(5X,'THE INERTIAL TENSOR ALONG THE ANATOMICAL AXES IS')
900   FORMAT(5X,'THE INERTIAL TENSOR ALONG THE MECHANICAL AXES IS')

      GO TO 5
1000  CONTINUE
      STOP
      END

      SUBROUTINE TRANSP(A,B)
C
C  --THIS SUBROUTINE WILL RETURN THE TRANSPOSE(B) OF MATRIX A--
C
      REAL A(4,4),B(4,4)
      DO 10 I=1,4
        DO 10 J=1,4
10        B(I,J)=A(J,I)
      RETURN
      END


      SUBROUTINE ROTATE(M,ITNSR,RSNTI)
C
C  ----THIS SUBROUTINE WILL,THROUGH A SIMILARITY TRANSFORM---------
C  --ATION, TRANSFORM AN INERTIAL TENSOR INTO ANOTHER AXIS SYSTEM--

      REAL ITNSR(4,4),RSNTI(4,4),M(4,4),MT(4,4),FRST(4,4)
      CALL TRANSP(M,MT)
      CALL MULTMAT(M,ITNSR,FRST)
      CALL MULTMAT(FRST,MT,RSNTI)
      RETURN
      END


      SUBROUTINE TRANSLATE(CG,INERTIAL,MASS,TIALINER)

C  ----THIS SUBROUTINE WILL TRANSLATE AN INERTIAL TENSOR( IE,--
C  --CONSISTING OF MASS MOMENTS OF INERTIA)-------------------
C
      REAL INERTIAL(4,4),CG(3),TIALINER(4,4),MASS,T(3)
      DO 10 I=1,3
        DO 10 J=1,3
10        TIALINER(I,J)=INERTIAL(I,J)
```

```
      DO 20 I=1,3
 20     T(I)=-CG(I)
      DO 30 I=1,3
        DO 30 J=1,3
          TIALINER(I,J)=INERTIAL(I,J)+MASS*(T(I)*T(J))
 30   CONTINUE

      TIALINER(1,1)=INERTIAL(1,1)+MASS*(T(2)**2+T(3)**2)
      TIALINER(2,2)=INERTIAL(2,2)+MASS*(T(3)**2+T(1)**2)
      TIALINER(3,3)=INERTIAL(3,3)+MASS*(T(1)**2+T(2)**2)

      RETURN
      END


      SUBROUTINE MULTMAT(A,B,C)

C  ---- THIS SUBROUTINE WILL MULTIPLY TWO MATRICES--
C  --( A & B ) AND IT WILL RETURN THE RESULT ( C )--
C
      REAL A(4,4),B(4,4),C(4,4)
      DO 10 I=1,3
        DO 10 J=1,3
10        C(I,J)=A(I,1)*B(1,J)+A(I,2)*B(2,J)+A(I,3)*B(3,J)+A(I,4)*B(4
    +            ,J)
      RETURN
      END
C
C
      SUBROUTINE WRTMATR (OUTMATR)

C  --THIS SUBROUTINE WILL WRITE OUT A MATRIX--
C
      REAL OUTMATR(4,4)

      WRITE(5,35)
      WRITE(7,35)
      DO 400 I=1,4
        WRITE(5,45) (OUTMATR(I,J),J=1,4)
400     WRITE(7,45) (OUTMATR(I,J),J=1,4)
35    FORMAT('     DISPLACEMENT MATRIX  (MECH. =) ANAT.)')
45    FORMAT(T5,F10.5,T15,F10.5,T25,F10.5,T35,F10.5)
      RETURN
      END

      SUBROUTINE WRTCOOR (NAME,OUTCOOR,J)

C  --THIS SUBROUTINE WILL WRITE OUT COORDINATE LOCATIONS--
C
      CHARACTER NAME*41(35)
      REAL OUTCOOR(4)
      WRITE (5,66) NAME(J),OUTCOOR(1),OUTCOOR(2),OUTCOOR(3)
      WRITE (7,65) NAME(J),OUTCOOR(1),OUTCOOR(2),OUTCOOR(3)
65    FORMAT (2X,A41,3F8.2)
66    FORMAT(2X,A41,3F8.2)
      RETURN
      END

      SUBROUTINE DISPMULT (A,B,C)

C  ----DISPMULT MULTIPLIES THE MATRIX A BY THE VECTOR--
C  --B AND RETURNS THE VECTOR C.--
C
```

```
      ...
      DO 10 I=1,3
10       C(I)=A(I,1)*B(1)+A(I,2)*B(2)+A(I,3)*B(3)+A(I,4)
      RETURN
      END


      SUBROUTINE DISPMAT (R,P0,P1,D)
C
C   ----COMPUTES DISPLACEMENT MATRIX D FOR A ROTATION--
C   --R AND A TRANSLATION FROM P0 TO P1.---------------
C
C        ARGUMENTS:
C          D(4,4): DISPLACEMENT MATRIX TO BE COMPUTED.
C          R(4,4): ROTATION MATRIX
C          P0(3): VECTOR
C          P1(3): VECTOR
C
C          R, P0 AND P1 ARE ALL IN THE GLOBAL COORDINATE SYSTEM
C

      DIMENSION  D(4,4),R(4,4),P0(3),P1(3)
      DO 1,I=1,3
      DO 2 J=1,3
2     D(I,J)=R(I,J)
      D(I,4)=P1(I)-(R(I,1)*P0(1)+R(I,2)*P0(2)+R(I,3)*P0(3))
1     D(4,I)=0.0
      D(4,4)=1.0
      RETURN
      END

      SUBROUTINE INTSAXIS(MRK,FDZ)

C  ----INTSAXIS WILL PRODUCE THE DISPLACEMENT MATRIX---
C  --BETWEEN TWO AXIS SYSTEMS GIVEN THREE POINTS THAT--
C  --DEFINE THE NEW AXES IN THE OLD AXIS SYSTEM.-------
C
      REAL A(3),B(3),C(3),D(3),E(3),F(3),G(3),H,Z(3),MRK(35,3),
     +     FDZ(4,4),ND(3),NF(3),NZ(3)

      DO 101 I=1,4
         DO 101 J=1,4
         S=0.
         IF(I.EQ.J) S=1.
 101     FDZ(I,J)=S

      DO 100 I=1,3
         D(I)=MRK(2,I)-MRK(1,I)
100      E(I)=MRK(3,I)-MRK(1,I)

      CALL NORM (D,ND)

      H=DOT(E,ND)

      DO 200 I=1,3
200      G(I)=H*ND(I)

      DO 300 I=1,3
300      F(I)=E(I)-G(I)

      CALL NORM (F,NF)

      CALL CROSS (NF,ND,Z)

      CALL NORM (Z,NZ)
```

```
      DO 500 I=1,3
         FDZ(4,I)=-MRK(1,I)
         FDZ(I,1)=NZ(I)
         FDZ(I,2)=NF(I)
500      FDZ(I,3)=ND(I)

      RETURN
      END

C  ----FUNCTION DOT RETURNS THE DOT PRODUCT OF THE--
C  --TWO THREE DIMENSIONAL VECTORS A AND B.---------
C
C
      FUNCTION DOT (A,B)
      REAL A(3),B(3)
      DOT = 0.
      DO 100 I=1,3
100      DOT = DOT + A(I)*B(I)
      RETURN
      END


C  ----SUBROUTINE CROSS COMPUTES THE CROSS PRODUCT OF PARAMETERS--
C  --A AND B AND RETURNS THE RESULT IN PARAMETER C.---------------
C
C
      SUBROUTINE CROSS (A,B,C)
      DIMENSION A(3),B(3),C(3)

      C(1) = A(2)*B(3) - A(3)*B(2)
      C(2) = B(1)*A(3) - B(3)*A(1)
      C(3) = A(1)*B(2) - A(2)*B(1)
      RETURN
      END


C  --SUBROUTINE NORM NORMALIZES THE THREE DIMENSIONAL VECTOR C.--
C
C
      SUBROUTINE NORM (A,B)
      REAL A(3),B(3)

      SIZE = 0.
      DO 50 I=1,3
50       SIZE = SIZE + A(I)**2
      SIZE = SQRT(SIZE)
      DO 100 I=1,3
100      B(I) = A(I)/SIZE
      RETURN
      END
```

Appendix E

ADAMLD3

```
C       RUNGE-KUTTA SOLUTION TO ADAM LIMB
C       LOADING PROBLEM BY BILL NETTLES
C         CHANGED 9/8/87
C
     DIMENSION AARM(6), ARM(6), CD(6),CF(6),
    1VEL(6), FARM(6)
       CHARACTER HEAD*15
     REAL MBOD,MLMB,LX(5),KX(5),KTHT(5),LTHT(5),MARM
       OPEN (2,FILE='LOADS.OUT',STATUS='NEW')
  10 FORMAT(//' PROGRAM: ADAMLD3.FTN'///' ENTER THE FREESTREAM VEL(FPS) AND
    1 AIR DENSITY(LB/FT3')
       WRITE(5,5)
  5  FORMAT(/' ENTER THE HEADING ')
  4    FORMAT(15A)
       READ(5,4) HEAD
       WRITE(2,4) HEAD
     WRITE(5,10)
       WRITE(2,10)
     ACCEPT*,VO,RHO
       WRITE(2,*)VO,RHO
  20 FORMAT(' ENTER INITIAL TORSO X DISP (FT), INITIAL VEL (FPS)
    1AND INITIAL ACCEL (FPS2)')
     WRITE(5,20)
       WRITE(2,20)
     ACCEPT*,XX,XXDOT,XXDDT
       WRITE(2,*) XX,XXDOT,XXDDT
  25 FORMAT(///' ENTER INITIAL LIMB THETA DISP (RAD), INITIAL ANG
    1 VEL (RPS),AND ANG ACCEL (RDPS2)')
     WRITE(5,25)
       WRITE(2,25)
     ACCEPT*,THTA,THTDT, THTDD
       WRITE(2,*) THTA,THTDT,THTDD
  30 FORMAT(//3X' XDISP',8X,' VEL',8X,' ACCL',3X,' THETA',5X' ANG V',
    14X,' ANG ACCEL'4X,'T'/)
  35 FORMAT(6X,'FT',10X,'FPS',9X,'FPS2',5X,'RAD',7X,'RDPSC',7X,'RDPS2',
    15X,'SEC')
  40 FORMAT(//' ENTER DELTA TIME STEP (SEC)')
     T=0
     WRITE(5,40)
       WRITE(2,40)
     ACCEPT*,DELT
       WRITE(2,*) DELT
  41 FORMAT(' ENTER RANGE OF MOTION WO SOFT STOP PRESENT (RAD)')
     WRITE(5,41)
       WRITE(2,41)
     ACCEPT *,ANG
       WRITE(2,*) ANG
  42 FORMAT(' ENTER SOFT STOP DEPTH AND RADIUS IN INCHES')
     WRITE(5,42)
       WRITE(2,42)
     ACCEPT*,STDEP,STRD
       WRITE(2,*) STDEP,STRD
  43 FORMAT(' ENTER NO OF STOPS IN SERIES AND STOP PAIRS')
     WRITE(5,43)
```

```
      WRITE(2,43)
    ACCEPT*,STSR,STPR
      WRITE(2,*) STSR,STPR
    THSS=STDEP*STSR/STRD
    THLM=ANG-THSS
 45 FORMAT(//' ENTER TORSO-SEAT AREA AND DRAG COEF')
    WRITE(5,45)
      WRITE(2,45)
    ACCEPT*, ATOR,CDTOR
      WRITE(2,*) ATOR,CDTOR
 50 FORMAT(//' ENTER TORSO-SEAT AND LIMB MASS')
    WRITE (5,50)
      WRITE(2,50)
    ACCEPT*,MBOD,MLMB
      WRITE(2,*) MBOD,MLMB
 55 FORMAT(//' ENTER NUMBER OF LIMB SECTIONS-INTEGER')
    WRITE(5,55)
      WRITE(2,55)
    ACCEPT*, NN
      WRITE(2,*) NN
    DO 65 J=1,NN
 60 FORMAT(//' ENTER AERO AREA, MOM ARM, CD AND CLOTH FAC FOR ')
 61 FORMAT(5X,I2,' TH LIMB SECTION')
    WRITE (5,60)
      WRITE(2,60)
    WRITE (5,61)J
      WRITE(2,61)J
    ACCEPT*,AARM(J),ARM(J),CD(J),CF(J)
      WRITE(2,*) AARM(J),ARM(J),CD(J),CF(J)
 65 CONTINUE
 70 FORMAT(//' ENTER LIMB CG AND MOM OF INERTIA')
    WRITE(5,70)
      WRITE(2,70)
    ACCEPT*,RCG,AI
      WRITE(2,*) RCG,AI
    WRITE(5,30)
      WRITE(2,30)
    WRITE(5,35)
      WRITE(2,35)
 80 CONTINUE
 85 FORMAT(F9.3,3X,F10.3,3X,F10.2,2X,F6.3,3X,F9.3,3X,F9.3,3X,F5.3)
    WRITE(5,85) XX,XXDOT,XXDDT,THTA,THTDT,THTDD,T
      WRITE(2,85) XX,XXDOT,XXDDT,THTA,THTDT,THTDD,T
    T = T+DELT
    IF(THTA.GT.THLM) GO TO 210
    XUDOT=XXDOT
    THTU=THTA
    THTDU=THTDT
    DO 200 N=1,4
    FACT=1.0
    IF (N.LT.2) GO TO 110
    IF (N.GT.3) GO TO 100
    FACT=0.5
100 XXDDT1=LX(N-1)/DELT*FACT
    XUDOT=XXDOT+LX(N-1)*FACT
    THTU=THTA+KTHT(N-1)*FACT
```

```
      THTDU=THTDT+LTHT(N-1)*FACT
      THTDD1=LTHT(N-1)/DELT*FACT
110   FTOR=(RHO/2.)*((VO-XUDOT)**2.)*ATOR*CDTOR
      KX(N)=DELT*XUDOT
      LX(N)=DELT*(FTOR-MLMB*RCG*SIN(THTU)*THTDD1-MLMB*RCG*COS(THTU)
     1*THTDU**2)/(MBOD+MLMB)
      MARM=0.0
      DO 120 J=1,NN
      VEL(J)=(VO-XUDOT)*SIN(THTU)-ARM(J)*THTDU
      FARM(J)=(RHO*0.5)*VEL(J)**2.*AARM(J)*CD(J)*CF(J)
      MARM=FARM(J)*ARM(J)+MARM
120   CONTINUE
      KTHT(N)=DELT*THTDU
      LTHT(N)=DELT*(MARM-(MLMB*RCG*SIN(THTU)*XXDDT1))/AI
200   CONTINUE
      XXOLD=XX
      XDTOLD=XXDOT
      XDDOLD=XXDDT
      THTOLD=THTA
      THDOLD=THTDT
      TDDOLD=THTDD
      XX=XX+(KX(1)+(KX(2)+KX(3))*2.0+KX(4))/6.0
      XDTNW=XXDOT+(LX(1)+(LX(2)+LX(3))*2.0+LX(4))/6.0
      XXDDT=(XDTNW-XXDOT)/DELT
      XXDOT=XDTNW
      THTA=THTA+(KTHT(1)+(KTHT(2)+KTHT(3))*2.0+KTHT(4))/6.0
      THDNW=THTDT+(LTHT(1)+(LTHT(2)+LTHT(3))*2.0+LTHT(4))/6.0
      THTDD=(THDNW-THTDT)/DELT
      THTDT=THDNW
      GO TO 80
210   RAT=(THLM-THTOLD)/(THTA-THTOLD)
      XX=XXOLD +RAT*(XX-XXOLD)
      XXDOT=XDTOLD+RAT*(XXDOT-XDTOLD)
      XXDDT=XDDOLD+RAT*(XXDDT-XDDOLD)
      THTA=THTOLD+RAT*(THTA-THTOLD)
      THTDT=THDOLD+RAT*(THTDT-THDOLD)
      THTDD=TDDOLD+RAT*(THTDD-TDDOLD)
      FTOR=(RHO/2)*((VO-XXDOT)**2)*ATOR*CDTOR
      MARM=0.
      DO 215 I=1,NN
      VEL(I)=(VO-XXDOT)*SIN(THTA) - ARM(I)*THTDT
      FARM(I)=0.5*RHO*VEL(I)**2*AARM(I)*CD(I)*CF(I)
      MARM=FARM(I)*ARM(I) + MARM
215   CONTINUE
      KE=(AI*THTDT**2)/2
      FORCE=(KE*12)/(STDEP*STPR*STSR)
      DYNMO=FORCE*STRD/12.0
      AMFAC=(DYNMO+MARM)/MARM
220   FORMAT(///' VALUES AT STOP CONTACT')
      WRITE(5,220)
        WRITE(2,220)
      WRITE(5,30)
        WRITE(2,30)
      WRITE(5,35)
        WRITE(2,35)
```

```
      WRITE(5,85) XX,XXDOT,XXDDT,THTA,THTDT,THTDD
        WRITE(2,85) XX,XXDOT,XXDDT,THTA,THTDT,THTDD
230 FORMAT(//,5X,' SEAT X FORCE',5X,' LIMB MOMENT',5X,' AMP FACT')
      WRITE(5,230)
        WRITE(2,230)
240 FORMAT(5X,F12.6,5X,F12.6,5X,F12.6)
      WRITE(5,240) FTOR,MARM,AMFAC
        WRITE(2,240) FTOR,MARM,AMFAC
      STCP
      END
```

Appendix F

## SOFTWARE ROUTINES AND FLOW CHARTS

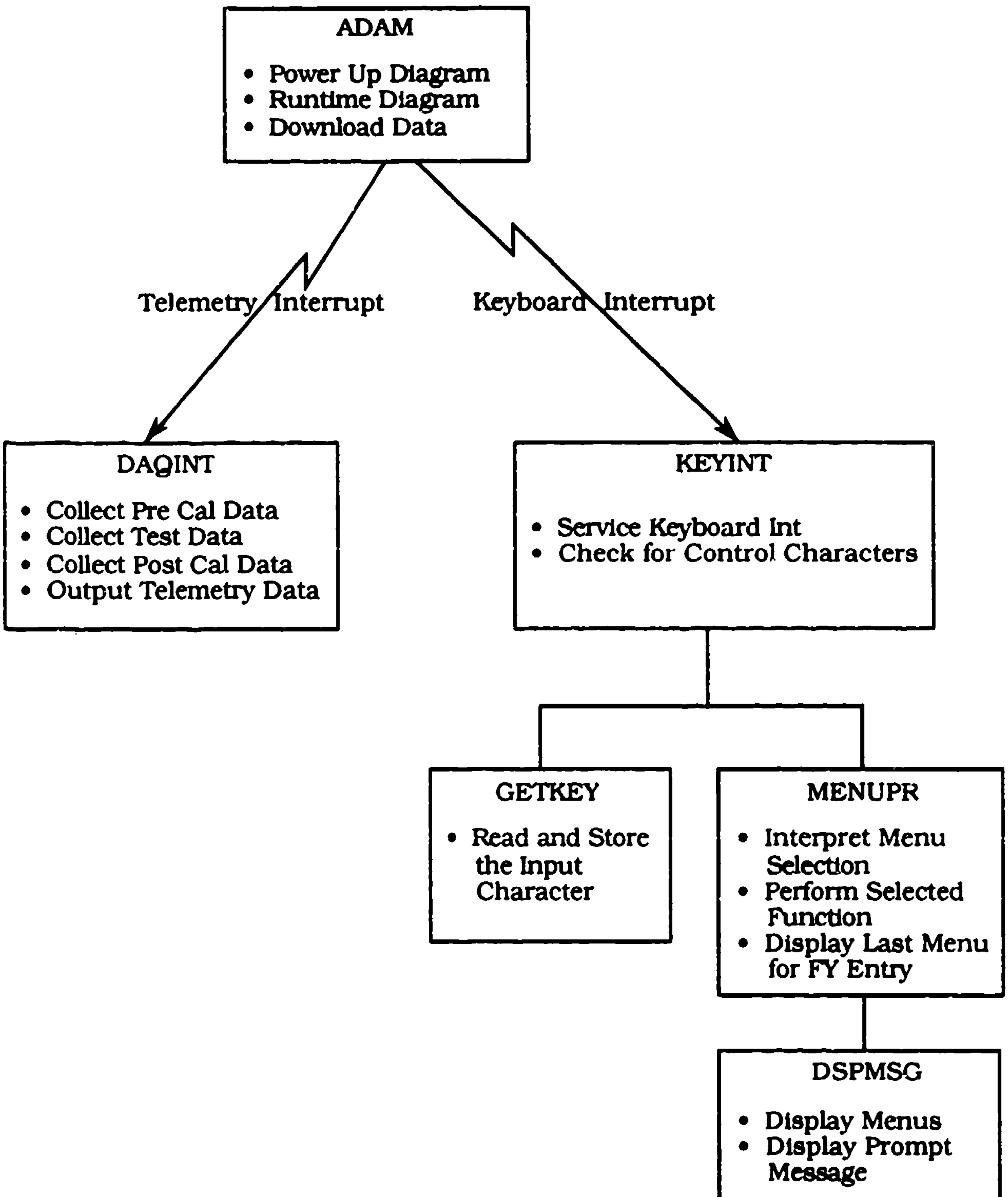



Figure 186. ADAM System (Top Level) Flow Chart

The ADAM program is divided into two main functional sections: Data Acquisition and Menu Processing. Each function is driven by an interrupt. The Data Acquisition (DAQINT) function is driven by the telemetry interrupt #6, and the Menu Processing (KEYINT) function is driven by the keyboard interrupt #2. When an interrupt occurs, the respective handler is executed. DAQINT is serviced by IDLE, PRECAL, DATCOL, or POSTCAL depending on the stage of data acquisition. KEYINT is serviced by MENUPR. While the system is in data acquisition mode, the keyboard interrupts are disabled until the data collection is complete and then reenabled if the terminal is connected.

## MODULE HIERARCHY

ADAM (Initialization)

| | |
|---|---|
| SERINIT | Initialize serial port |
| ROMTST | Power up ROM test |
| SERTST | Power up serial test |
| CLRSC | Clear screen |
| TMRTST | Power up timer test |
| ADTST | Power up A/D test |
| PARLTST | Power up parallel test |
| RAMTST | Power up RAM test |
| DSPMSG | Display message |

WAITLP (Process Server)

| | |
|---|---|
| DMPDAT | Download test data |
| IDLEJMP | Idle routine |
| PRLDIAG | Parallel diagnostic |
| CLKTST | Filter clock diagnostic |
| TELMTST | Telemetry port diagnostic |
| DISPTST | Display diagnostic |
| ADDIAG | A/D diagnostic |
| ALIGN | A/D alignment test |
| DSPLMU | Display last menu |

DAQINT (Telemetry Interrupt Handler)

| | |
|---|---|
| IDLE | Data collection but not storage |
| PRECAL | Precalibration data storage |
| DATCOL | Test data storage |
| POSTCAL | Postcalibration data storage |

KEYINT (Keyboard Interrupt Handler)

| | |
|---|---|
| GETKEY | Read keyboard entry |
| MENUPR | Process keyboard entry |

## REGISTER ASSIGNMENT

| | |
|---|---|
| A0 | General purpose |
| A1 | General purpose |
| A2 | Display pointer |
| A3 | DAQINT jump address |
| A4 | Scan table pointer |
| A5 | Data buffer pointer |
| A6 | Keyboard buffer pointer |
| A7 | Stack pointer |
| D0 | General purpose |
| D1 | General purpose |
| D2 | Display counter |
| D3 | General purpose |
| D4 | General purpose |
| D5 | A/D data |
| D6 | Key input buffer index |
| D7 | General purpose |

## STATUS BYTE DEFINITION

DIAGNOSTIC STATUS (DSTAT):

Bit set = 1 = error, Bit clear = 0 = passed

| | |
|---|---|
| Bit 0 | ROM error |
| Bit 1 | Serial error |
| Bit 2 | Filter clock error |
| Bit 3 | A/D error |
| Bit 4 | Parallel port error |
| Bit 5 | RAM error |
| Bit 6 | Not used |
| Bit 7 | Not used |

TEST STATUS (TSTST):

| | |
|---|---|
| Bit 0 | Precalibration mode (when set) |
| Bit 1 | Data collection mode (when set) |
| Bit 2 | Postcalibration mode (when set) |
| Bit 3 | Memory full (when set) |
| Bit 4 | Terminal connected (when cleared) |
| Bit 5 | RCAL mode (when set) |
| Bit 6 | Not used |
| Bit 7 | Start storing data (when set) |

The remainder of this appendix presents the next several levels of flow charts for all operations within ADAM. Following the opening top level ADAM system flow chart, there are more than 50 more flow charts with introductory text for them. Additional information about the events taking place in these routines can be found in the comments (right column) on the 68020 assembly source code listings of this ADAM resident software.

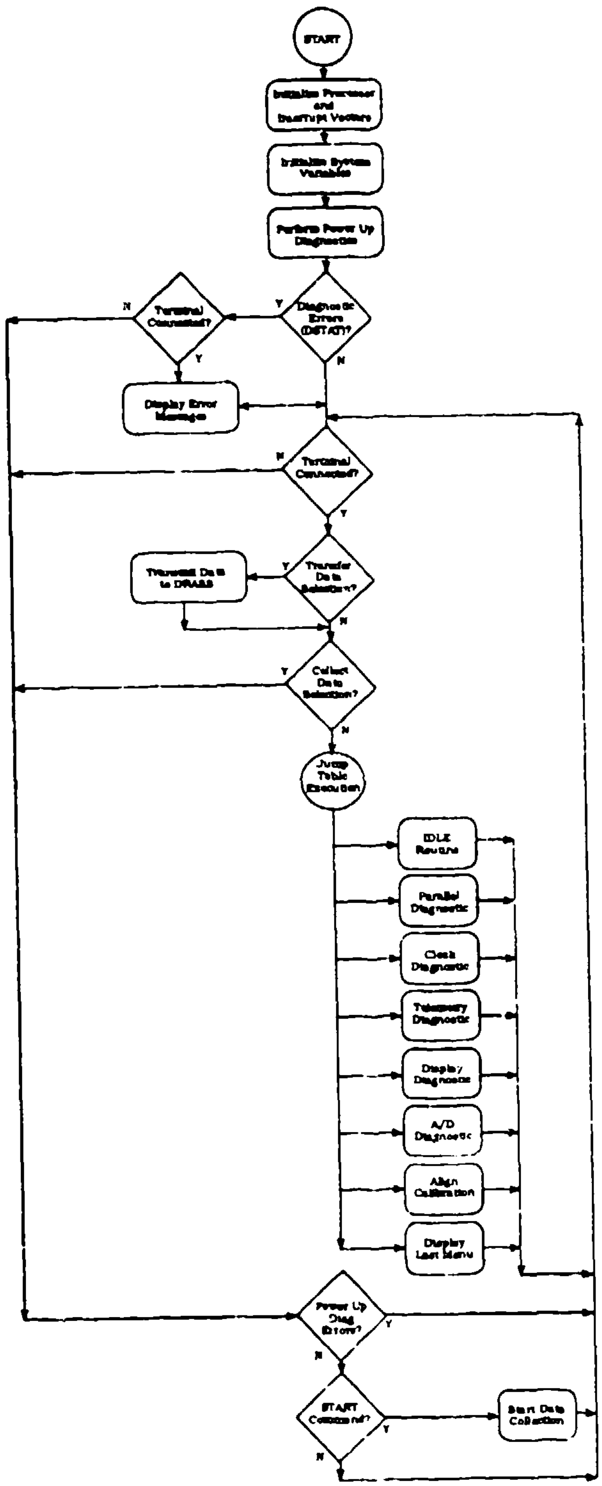



Figure 187. ADAM Initialization Flow Chart

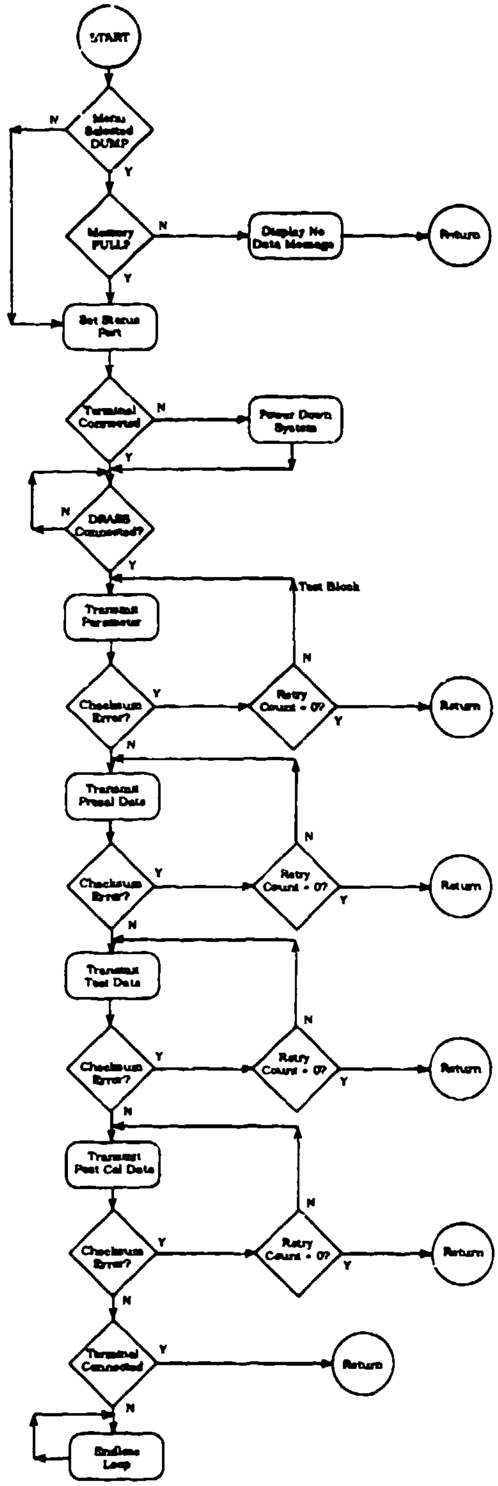



Figure 188. DMPDAT Download Test Data Flow Chart

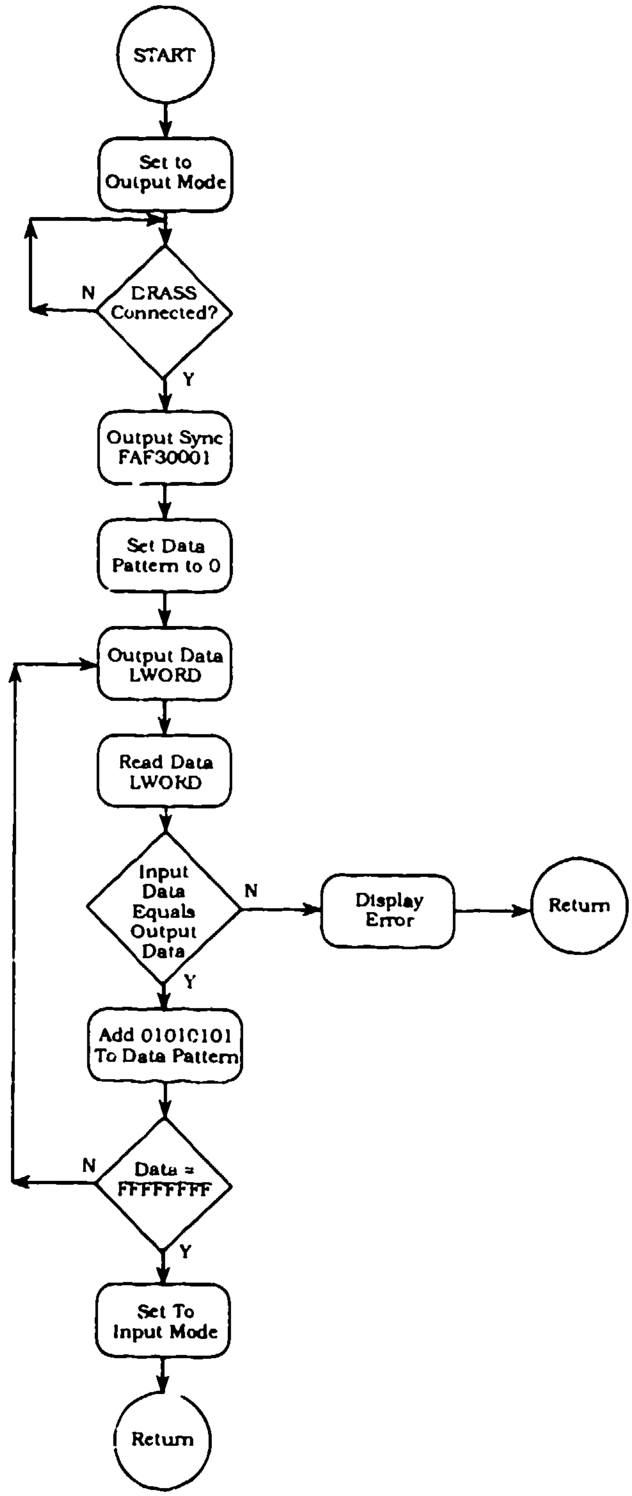



Figure 189. PRLDIAG Parallel Diagnostic Flow Chart

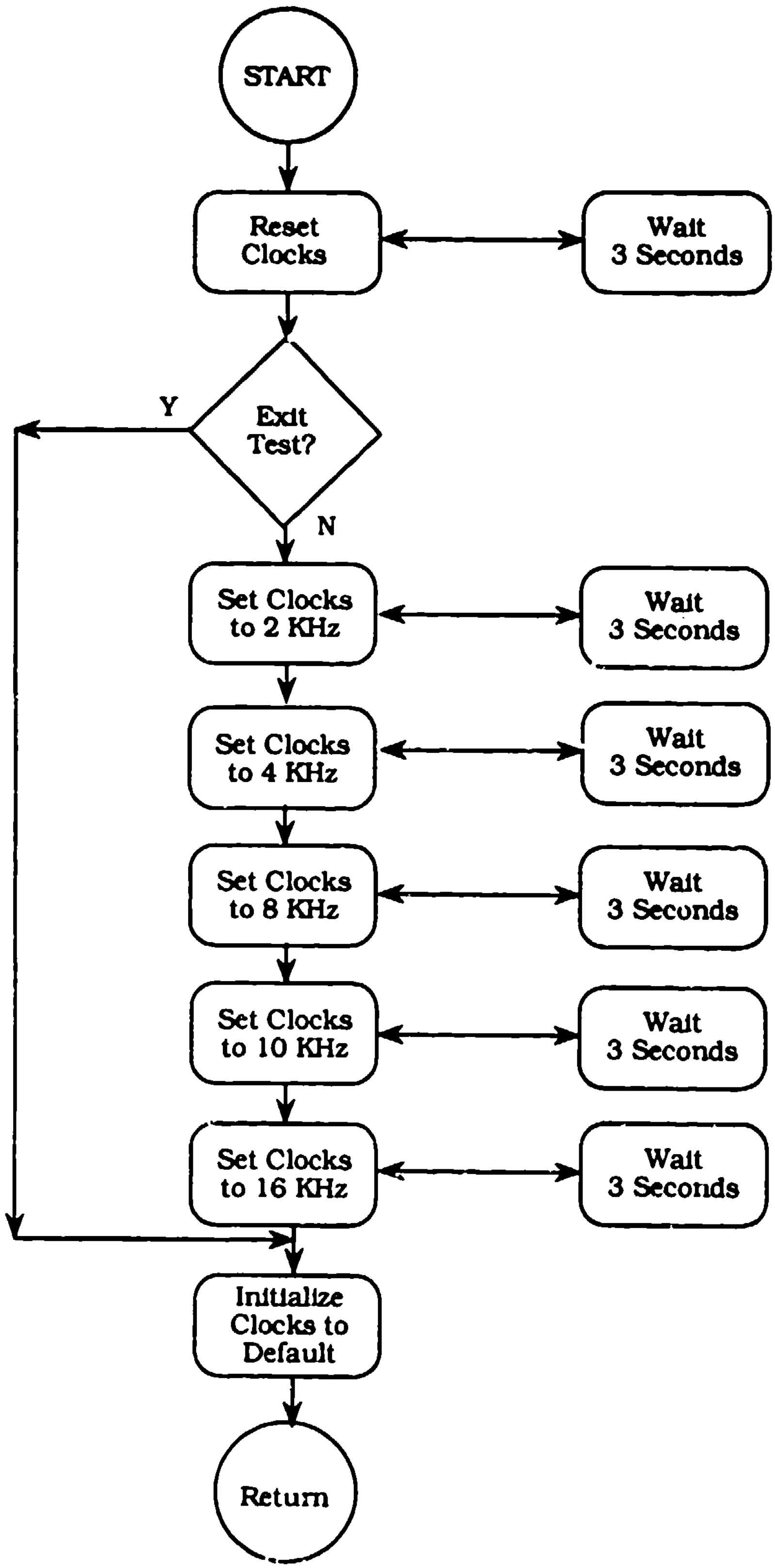



Figure 190. CLKTST Filter Clock Diagnostic Flow Chart

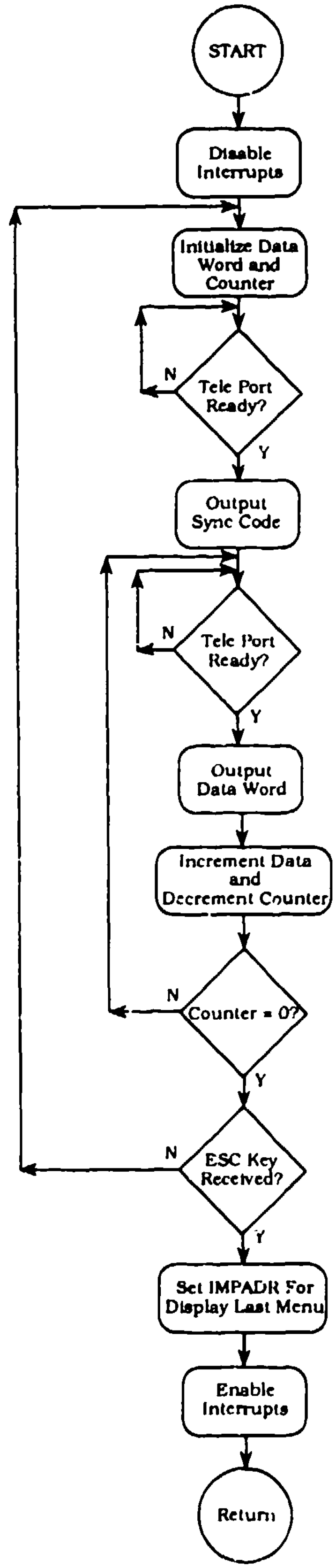



Figure 191. TELMTST Telemetry Port Diagnostic Flow Chart

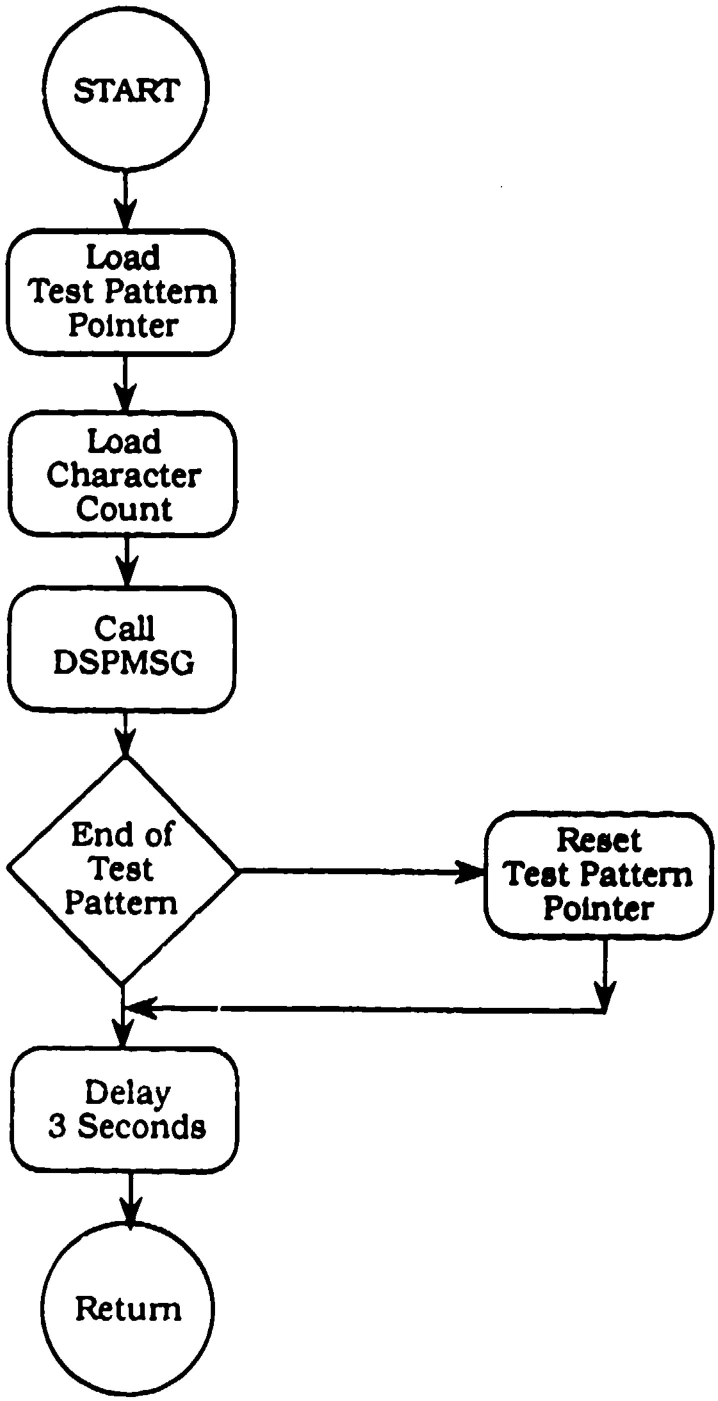



Figure 192. DISPTST Display Diagnostic Flow Chart

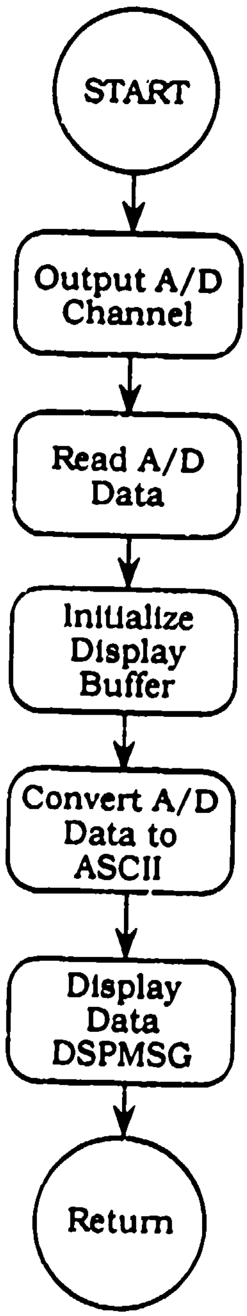



Figure 193. ADDIAG A/D Diagnostic Flow Chart

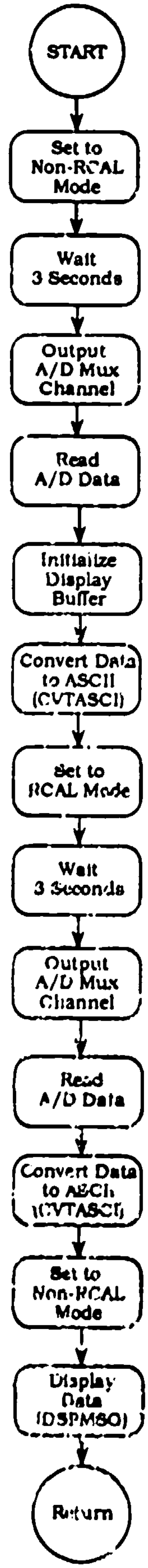



Figure 194. ALIGN A/D Alignment Test Flow Chart

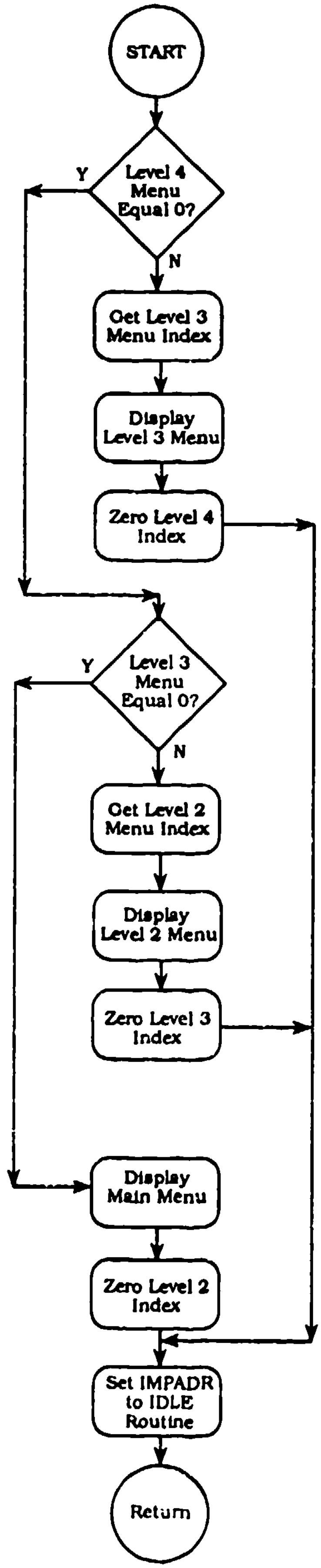



Figure 195. DSPLMU Display Last Menu Flow Chart

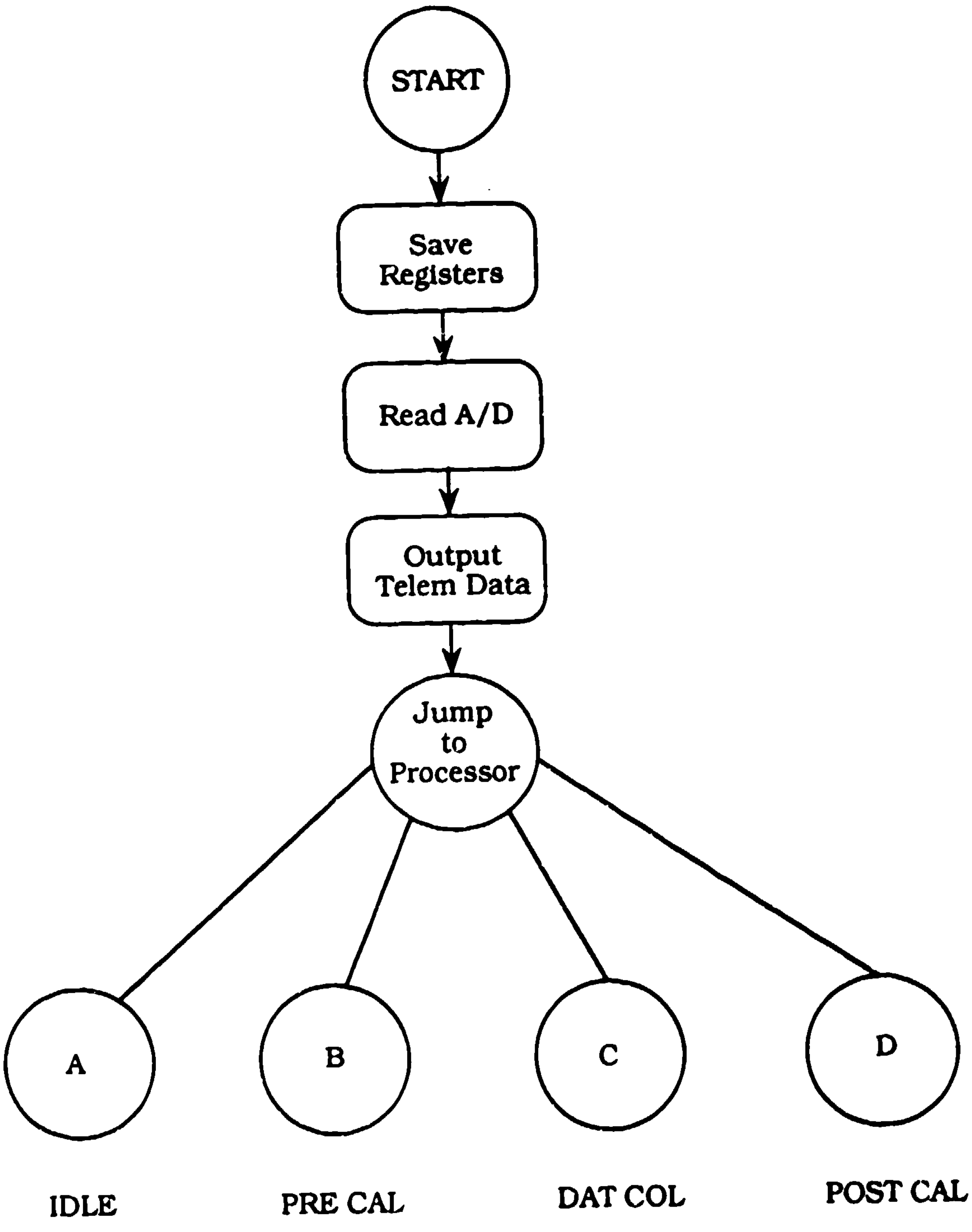



Figure 196. DAQINT Telemetry Interrupt Handler Flow Chart

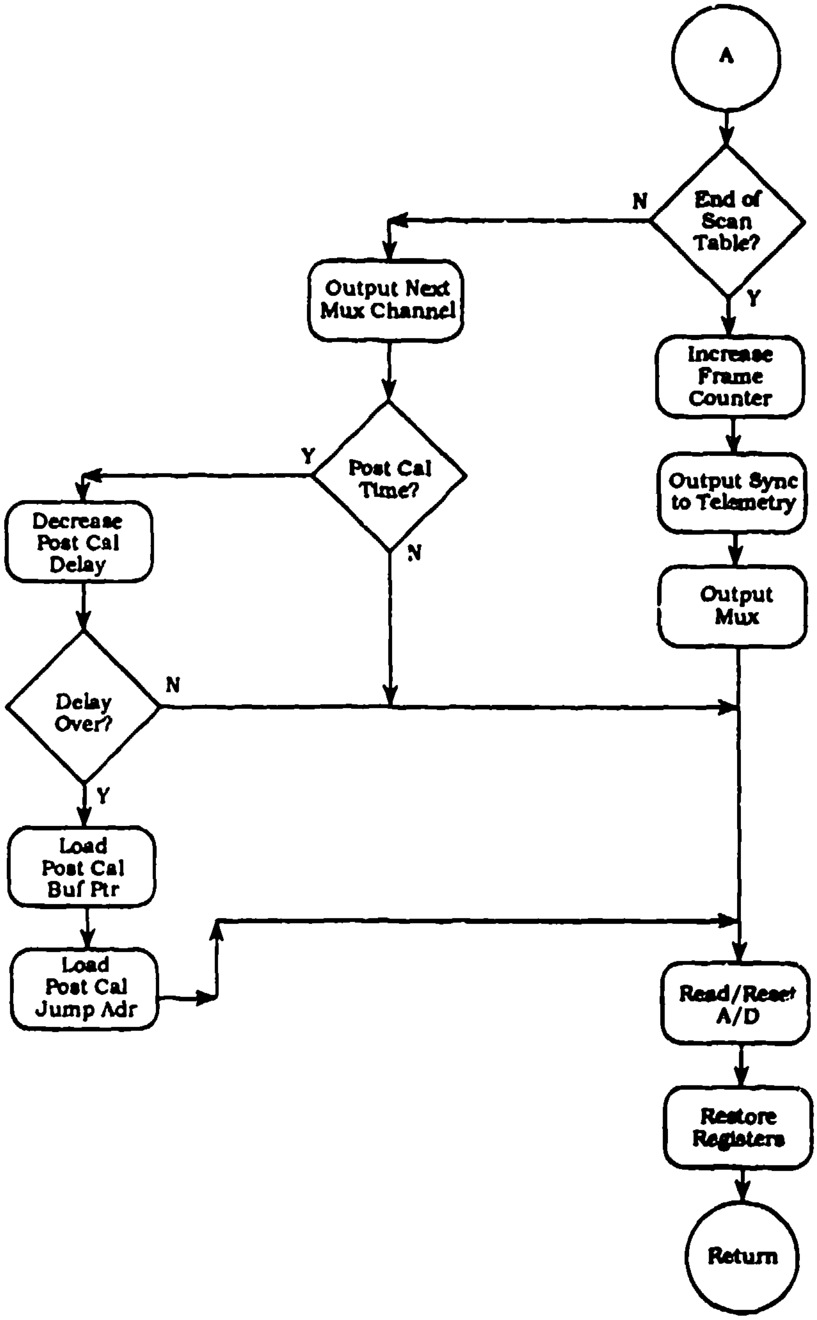



Figure 197. IDLE Nonstorage Data Collection Flow Chart

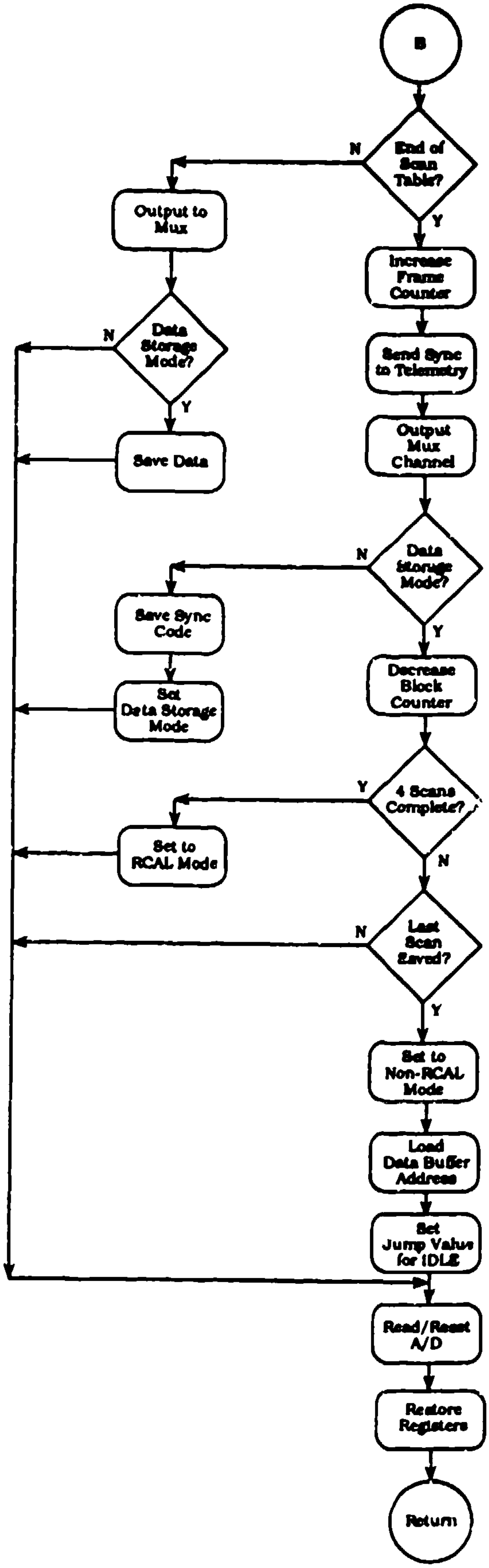



Figure 198. PRECAL Precalibration Data Storage Flow Chart

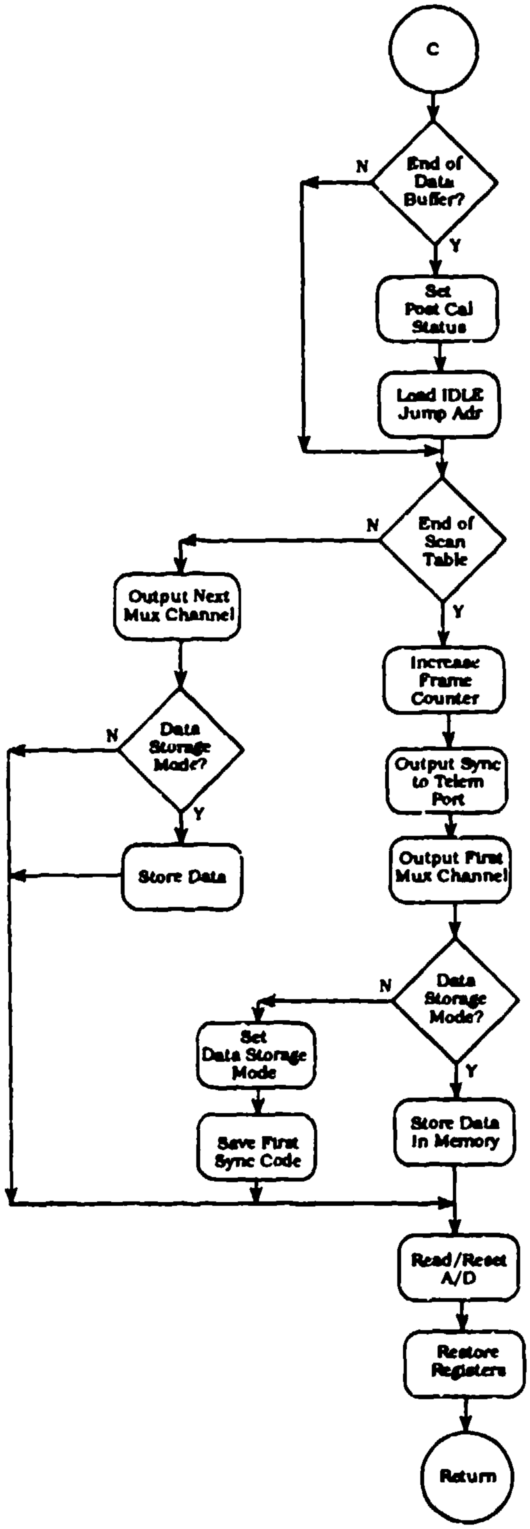



Figure 199. DATCOL Test Data Storage Flow Chart

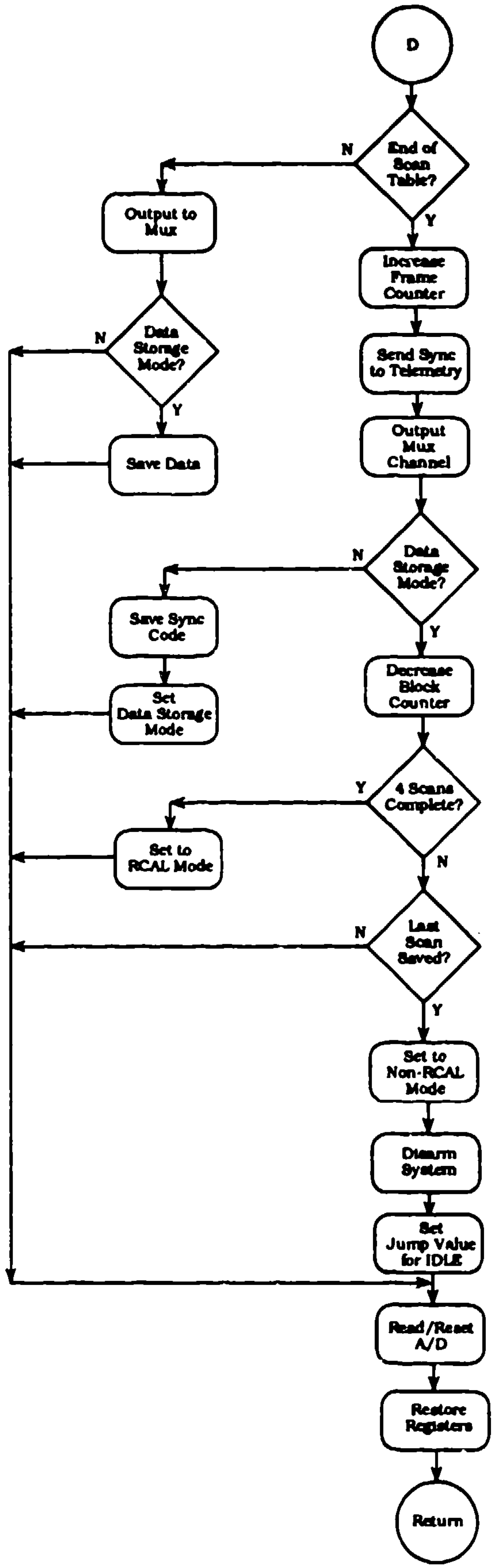



Figure 200. POSTCAL Postcalibration Data Storage Flow Chart

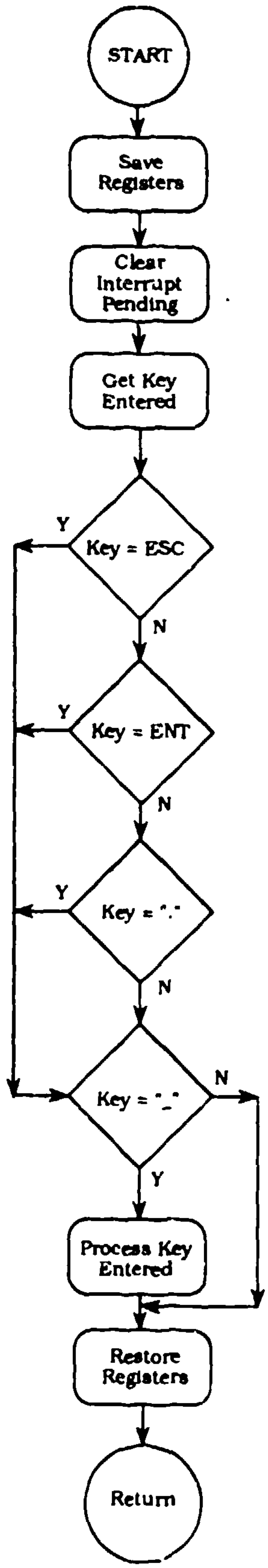



Figure 201. KEYINT Keyboard Interrupt Handler Flow Chart

## WDTST

This routine is used by RAMTST, RAMDIAG, and the routine to clear memory to write a data pattern to memory and check it. WDTST does a 16 bit word write and read of the data in register D0 to the address in register A2 up to the address in register A1. If there are any errors in the compare operations, register D7 is set to FF, and register A2 contains the error address.

| REGISTER: | CONTENTS: |
|---|---|
| **INPUT PARAMETERS** | |
| A0 | Start of test memory |
| A1 | End of test memory |
| D0 | Test data pattern (word) |
| **RETURNED PARAMETERS** | |
| A2 | Memory error address |
| D0 | Test pattern written |
| D1 | Data actually read from memory |
| D7 | Error flag (set to FF if failed) |

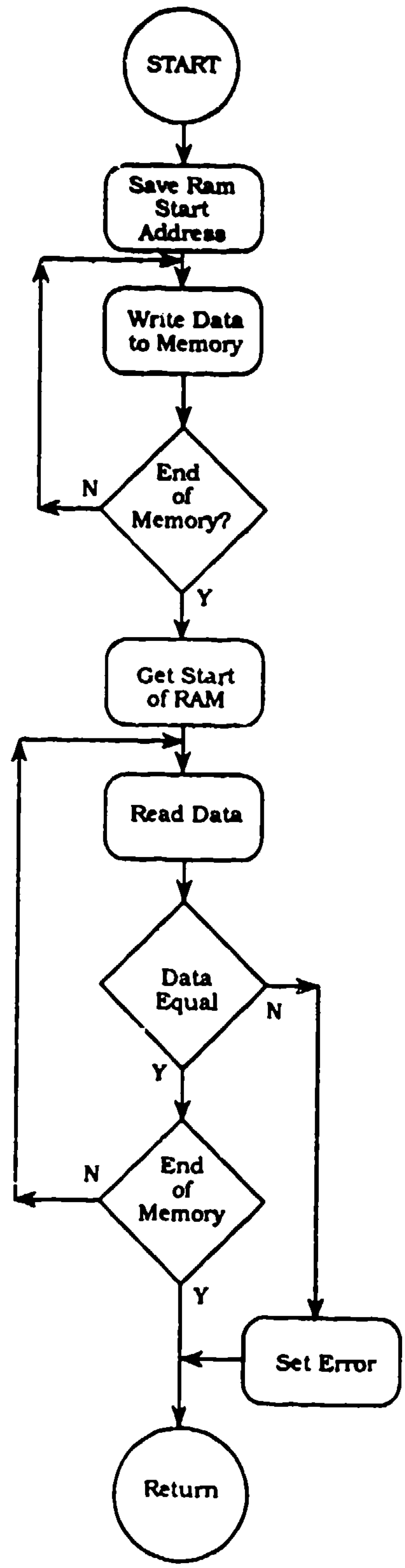



Figure 202. WDTST Word Test Flow Chart

## ROMTST

This routine calculates a sumcheck of the PROM and compares that value with the sumcheck stored in address FFFF. If the test fails, bit 0 in DSTAT is set to be checked after power up diagnostics is complete.

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | DSTAT: Contains the pass fail results of the test |
| REGISTER: | CONTENTS: |

REGISTERS USED

| | |
|---|---|
| A0 | Running index through PROM |
| D0 | End address of PROM |
| D1 | Running checksum total |

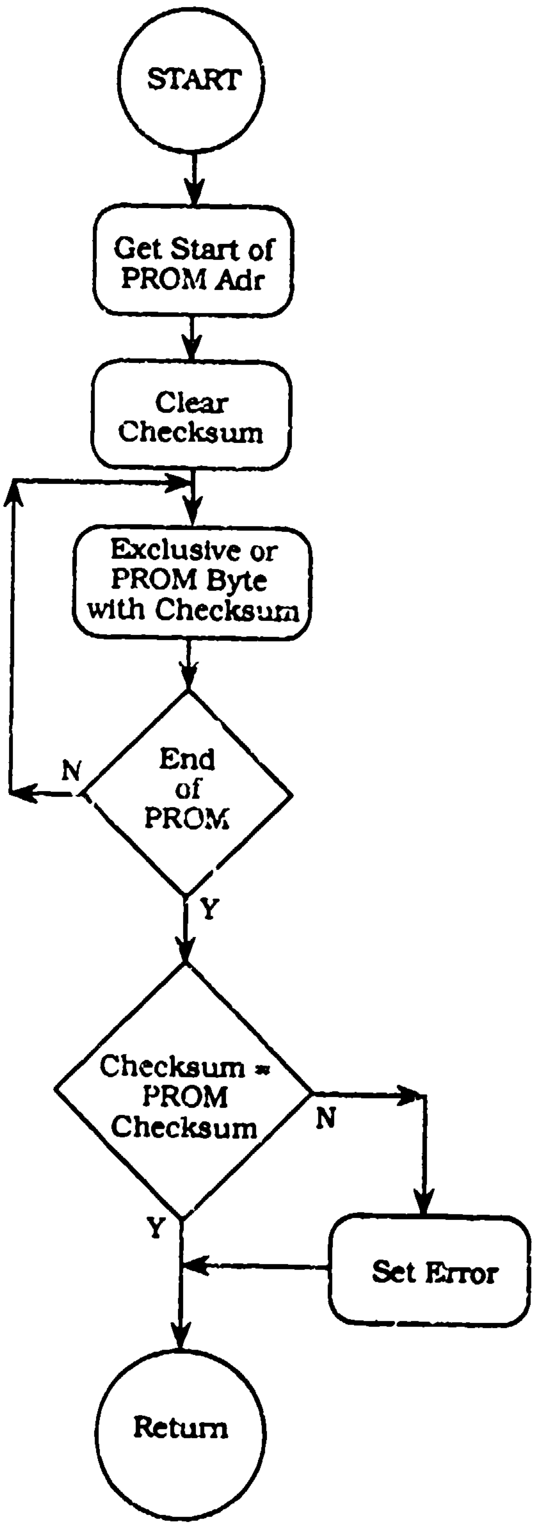



Figure 203. ROMTST PROM Checksum Test Flow Chart

## SERTST

This routine performs the power up diagnostic on the UART in the MFP. It performs an internal loop back test with a canned message. SERTST does not utilize the receive interrupt but it does test the receive status. The data format and baud rate used during the test (as established by the SERIAL DEFinitions mode) is the same that is used for this test.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: DSTAT: Power up diagnostics status

SERST: 1 - Frame error
2 - Parity error
3 - Overrun error

REGISTERS USED:

| REGISTER: | CONTENTS: |
|---|---|
| A0 | Index to test pattern |
| D0 | Test data character |
| D1 | Temporary UART status |
| D7 | Character count |

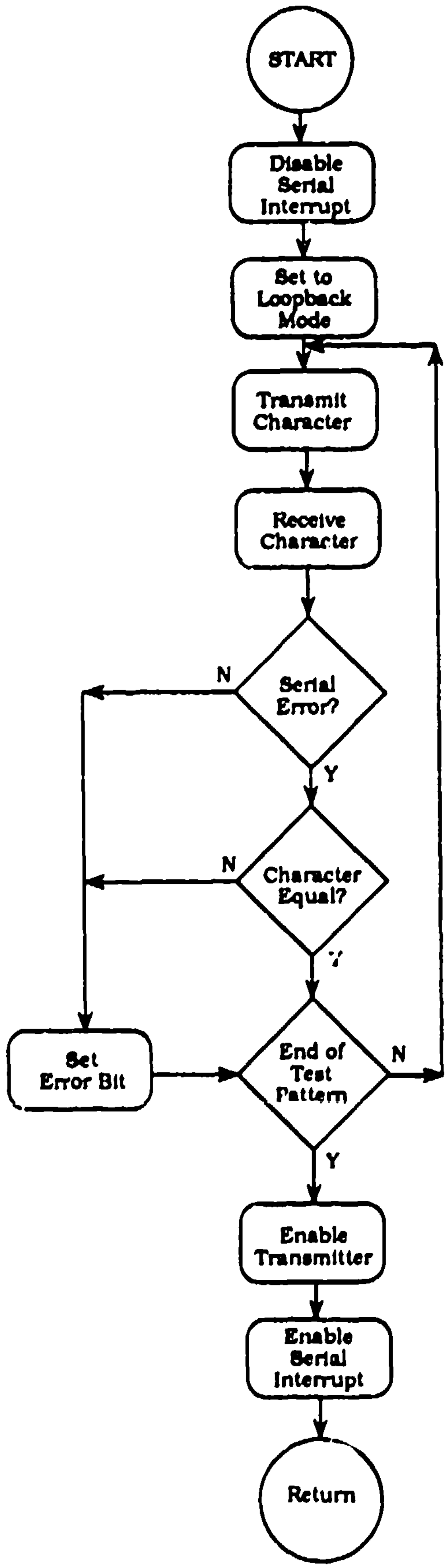



Figure 204. SERTST Serial Port Diagnostic Flow Chart

## TMRTST

This routine checks the functionality of the four filter clock timers during power up diagnostics. The timers are set to the prescale values that are established during the power up sequence (as previously established by the CLK RATE mode) but the timers' count values are set at 255. Then the count values are read from the timers and a delay is initiated. After the delay times out, the count values are read again. If the count has changed, then the timers are said to be operational. If no change was noted, then an error status is set in DSTAT.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: DSTAT: Power up diagnostic status

| REGISTERS: | CONTENTS: |
|---|---|
| **REGISTERS USED:** | |
| A1 | Index to the timer counters |
| D0 | First timer reading |
| D1 | Second timer reading |
| D2 | Temporary counter |
| D7 | Delay counter |

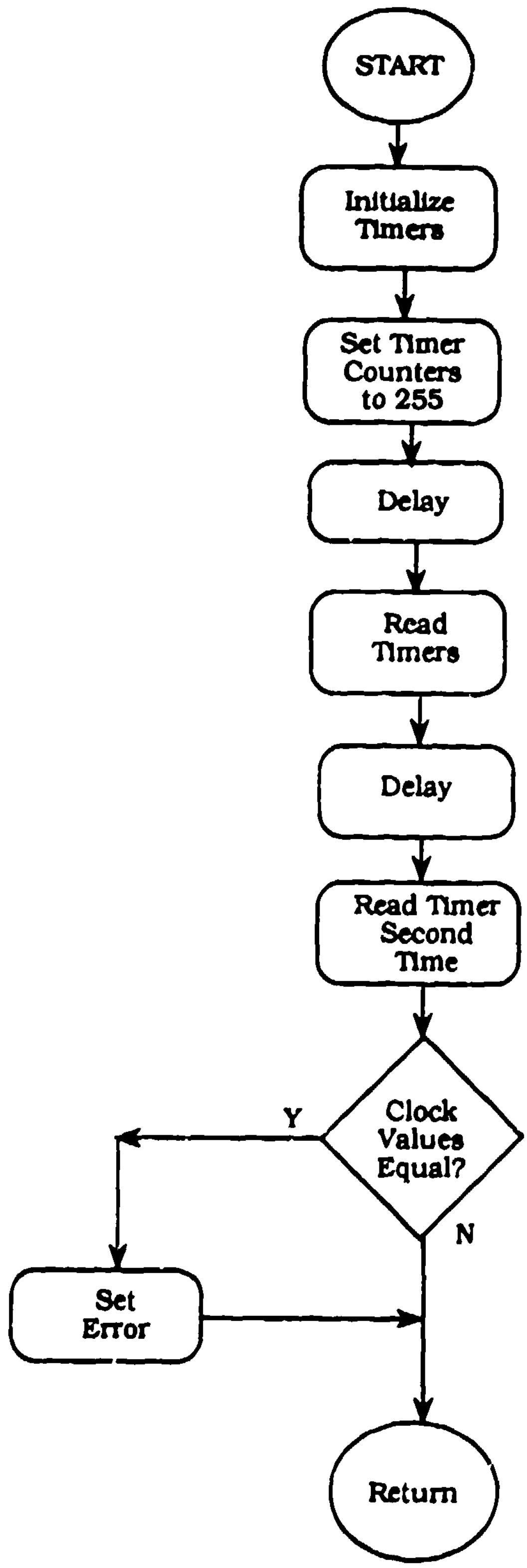



Figure 205. TMRTST Filter Clock Timers Diagnostic Flow Chart

## ADTST

This routine performs a check of the four A/Ds during power up. The test is performed on mux channel 01. ADTST first sets the system to RCAL mode and takes a reading. Then it sets the system to non-RCAL mode and takes another reading. It then compares the two readings; if they are the same value, DSTAT is flagged with an A/D error. If the values differ, then the A/Ds are tagged as operational. This doe snot test for A/D calibration, just functionality.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: DSTAT: Power up diagnostic status

| REGISTERS: | CONTENTS: |
|---|---|

REGISTERS USED:

| | |
|---|---|
| D1 | Sample counter |
| D2 | First reading |
| D3 | Second reading |

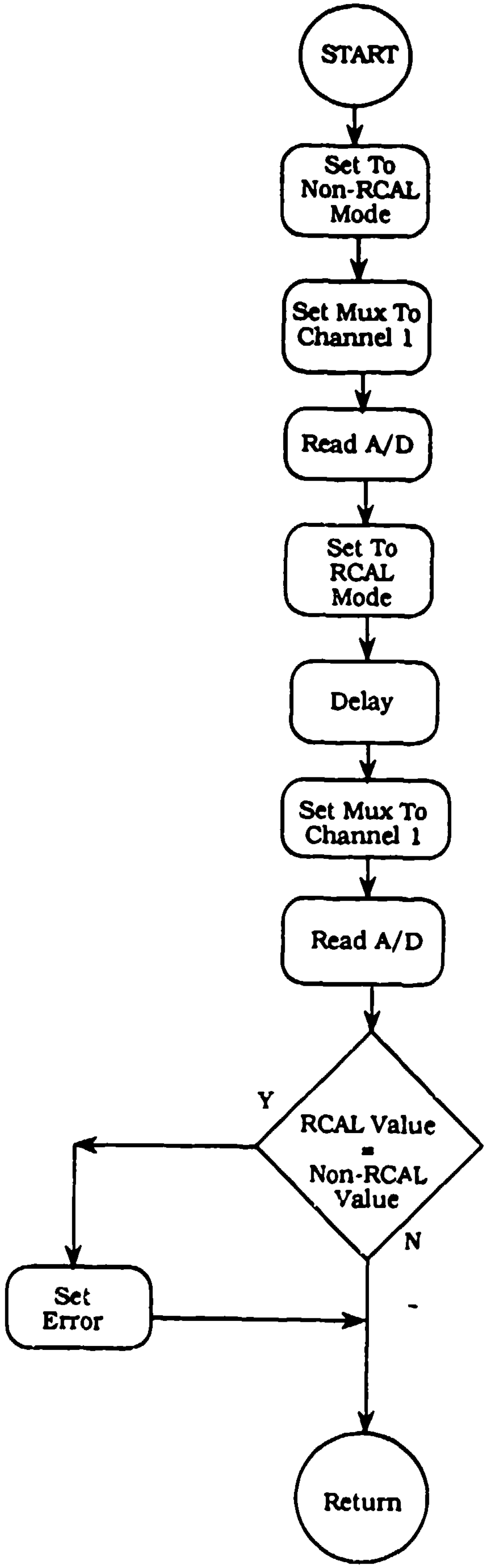



Figure 206. ADTST A/D Diagnostic Flow Chart

## RAMTST

This routine tests the SRAM in the system during power up diagnostics. It performs a byte write and read of memory with the data patterns AA, 55, FF, and 00. The memory that is tested is from 10000000 through 107ERE0. This prevents the destruction of system parameters and system stack. RAMTST uses the routine WDTST to do the actual memory accesses.

| | |
|---|---|
| INPUT PARAMETERS: | D7: Test status returned by WDTST |
| OUTPUT PARAMETERS: | DSTAT: Test status for power up diagnostics |
| REGISTERS: | CONTENT: |

REGISTERS USED:

| | |
|---|---|
| A0 | Start of memory test |
| A1 | End of memory test |
| D0 | Test data pattern |
| D7 | Test status from WDTST |

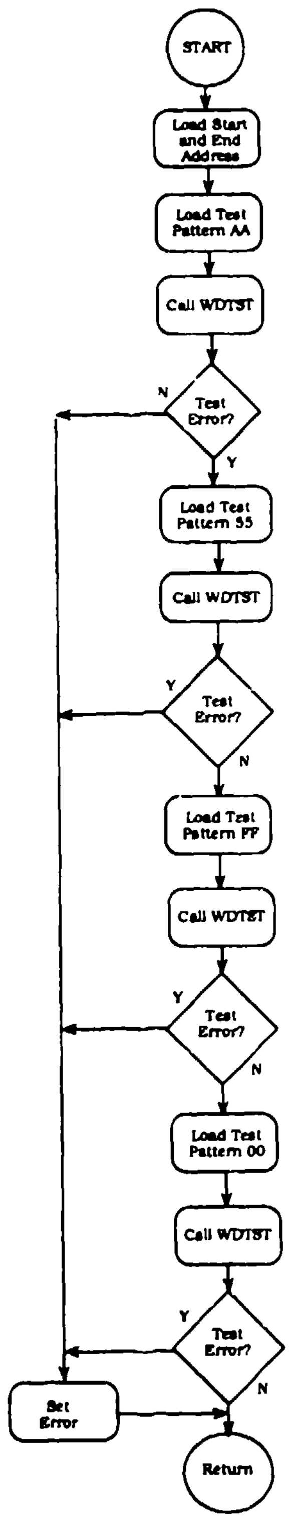



Figure 207. RAMTST SRAM Diagnostic Flow Chart

## MENUPR

The MENU PRocessor routine processes the menu entries entered on the handheld terminal. It is called by KEYINT when a delimiter (ENT, F4, .) is entered. MENUPR determines where to go to process the entry by the values contained in the menu level variables (MU1SL, MU2SL, MU3SL, and MUI4SL). The delimiter entry is fetched from the variable KEY, and the parameter entry is read from KEYBUF which is indexed by the register A6. Register D6 contains the number of characters entered. When a selection requires a new menu to be displayed, MENUPR calls DSPMSG to display it and then updates the manu level variables so that the next entry will be processed by the appropriate routine. When an ESC key is detected, LASTMU is executed, which looks at the manu levei variables to determine the previous menu for updating the display.

INPUT PARAMETERS:

KEY: Last key entered

KEYBUF: Characters entered less the delimiter

MU1SL: Value of the level 1 menu; always equal to 1 except during power up when it equals 0

MU2SL: Value of the level 2 menu

MU3SL: Value of the level 3 menu

MU4SL: Value of the level 4 menu (either 0 or 1)

OUTPUT PARAMETERS:

TXFLG: Flag for transmitting data to the DRASS

CLCTD: Flag to trigger data collection

JMPADR: Jump table value for the main loop execution

Figure 208 presents the flow charts for the MENU PRocessor.

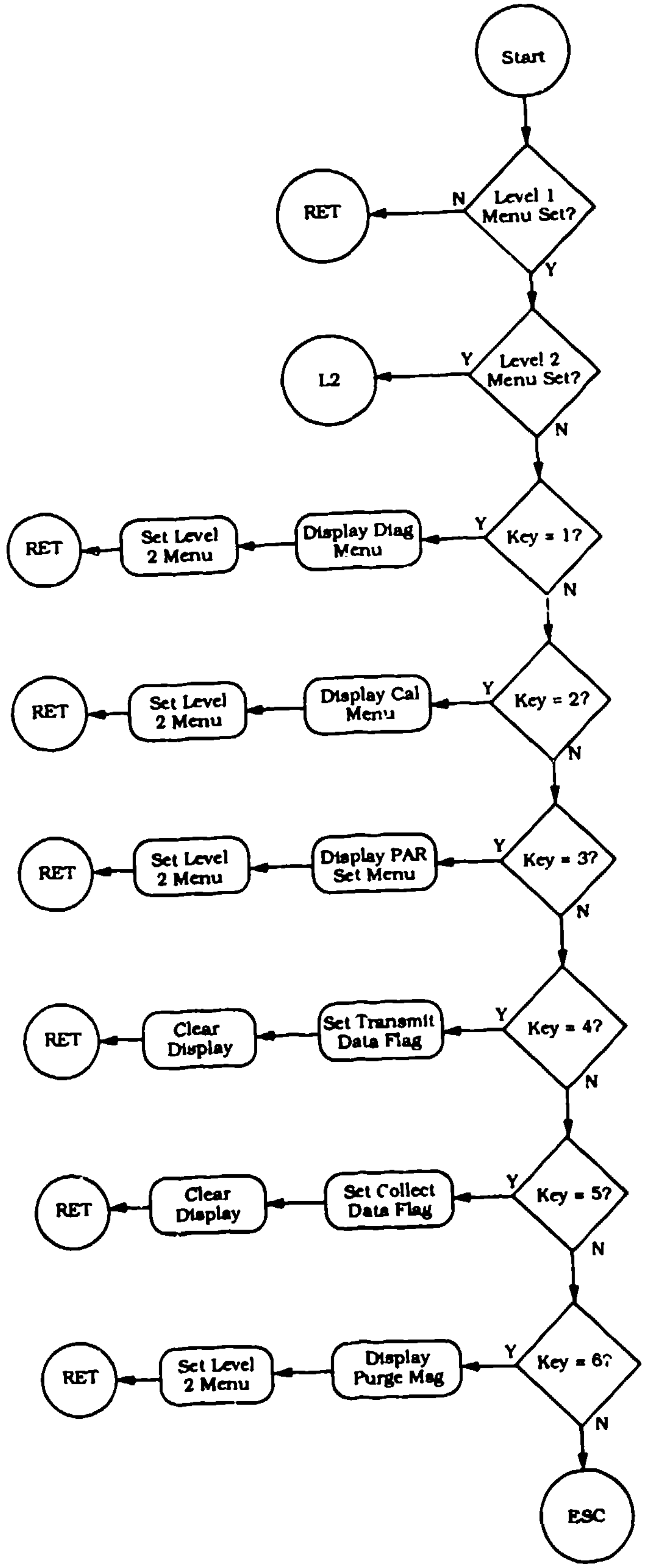



Figure 208. MENUPR Menu Processor Flow Chart (1 of 16)

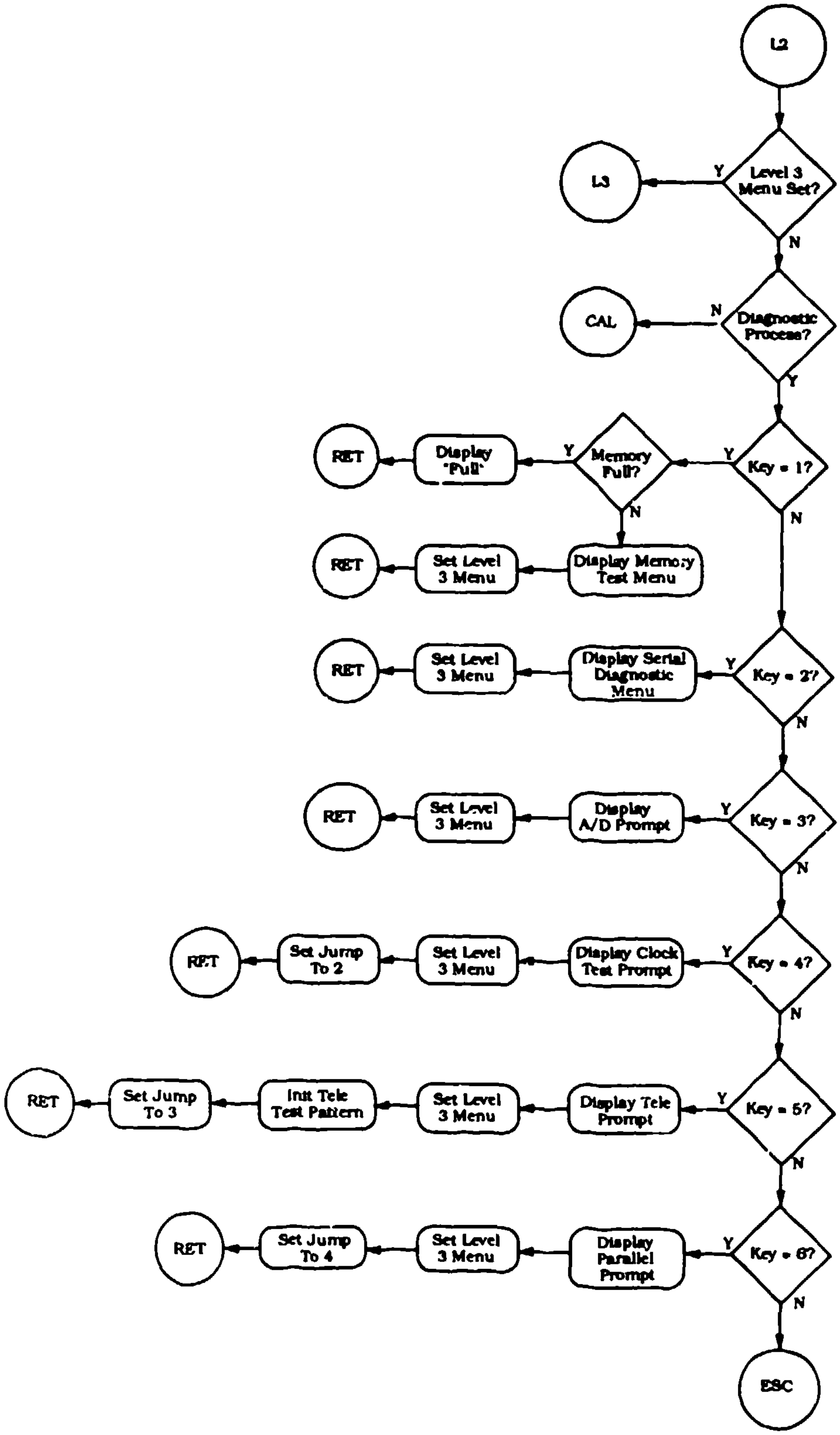



Figure 208. MENUPR Menu Processor Flow Chart (continued) (2 of 16)

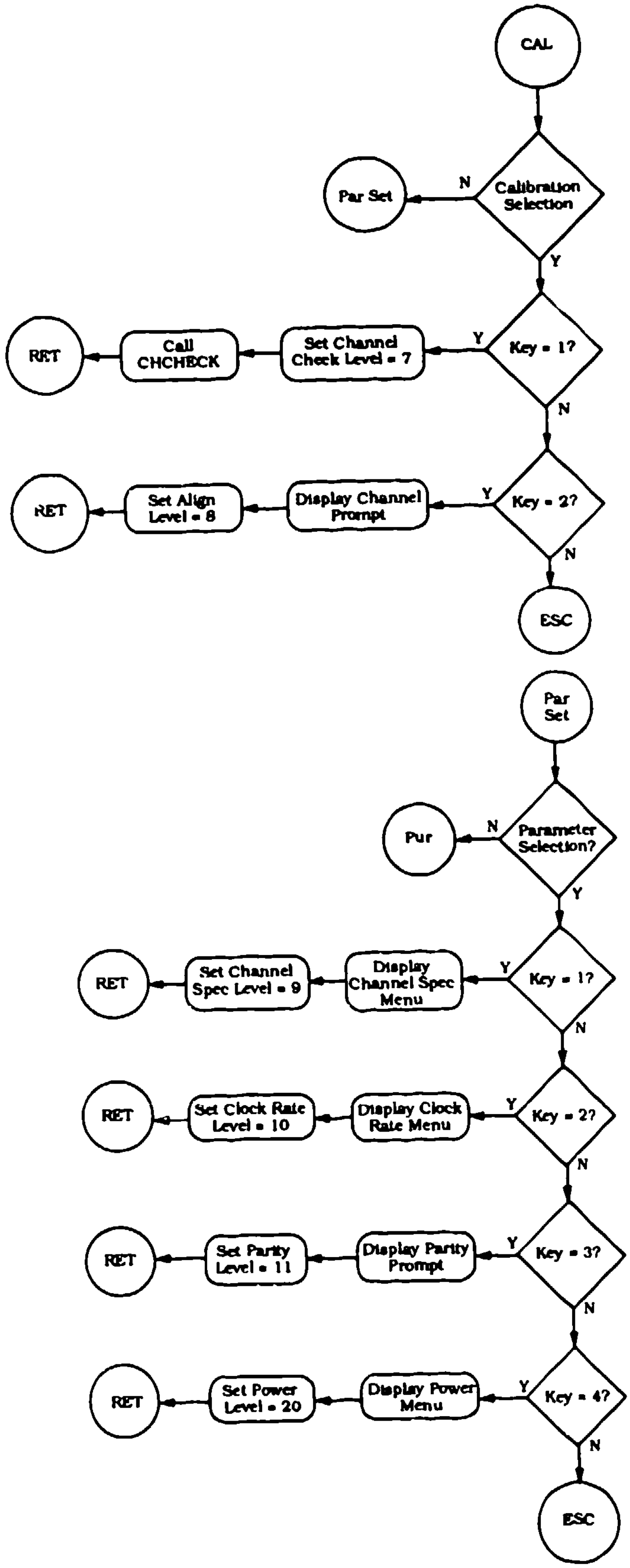



Figure 208. MENUPR Menu Processor Flow Chart (continued) (3 of 16)

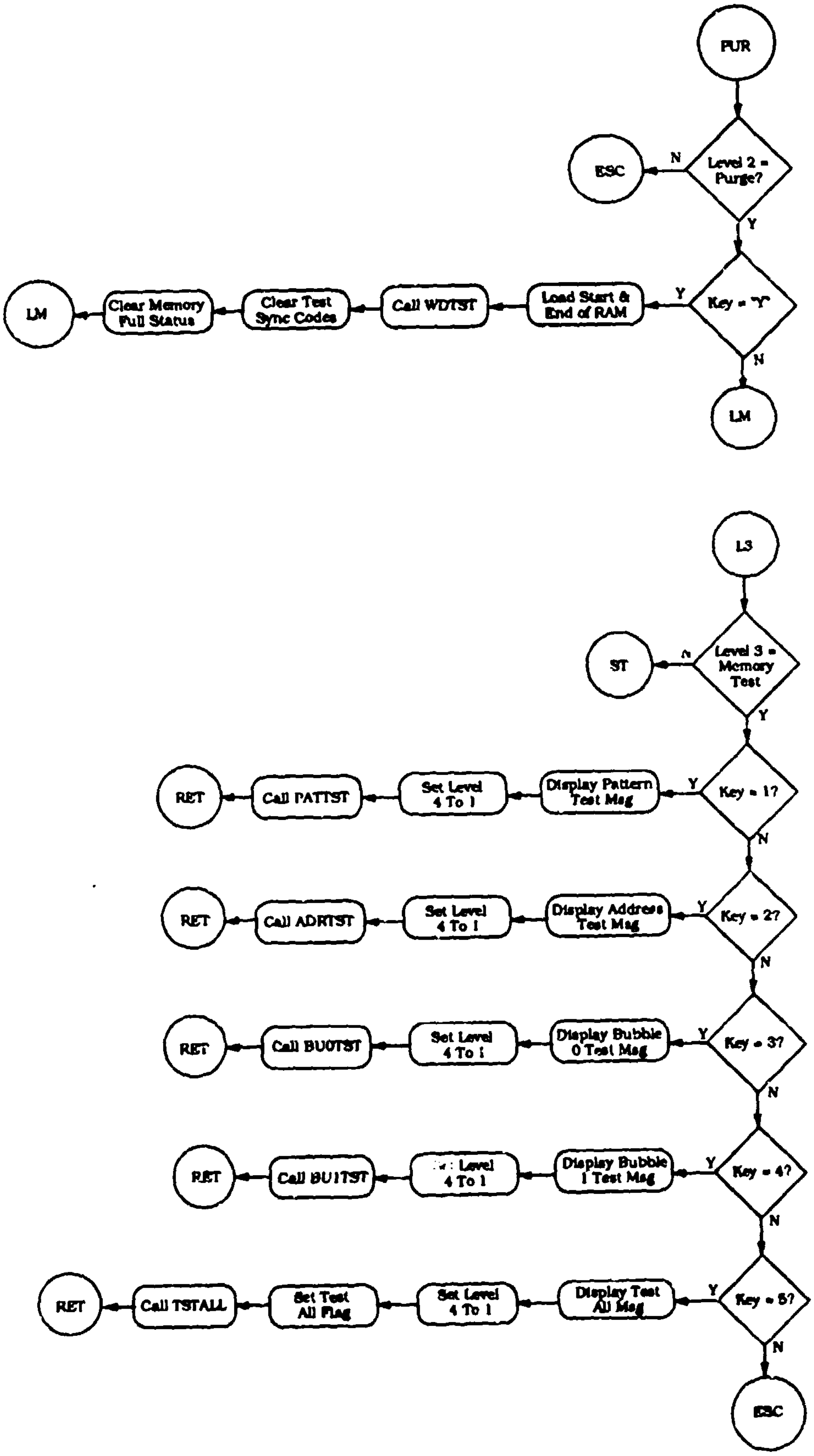



Figure 208. MENUPR Menu Processor Flow Chart (continued) (4 of 16)

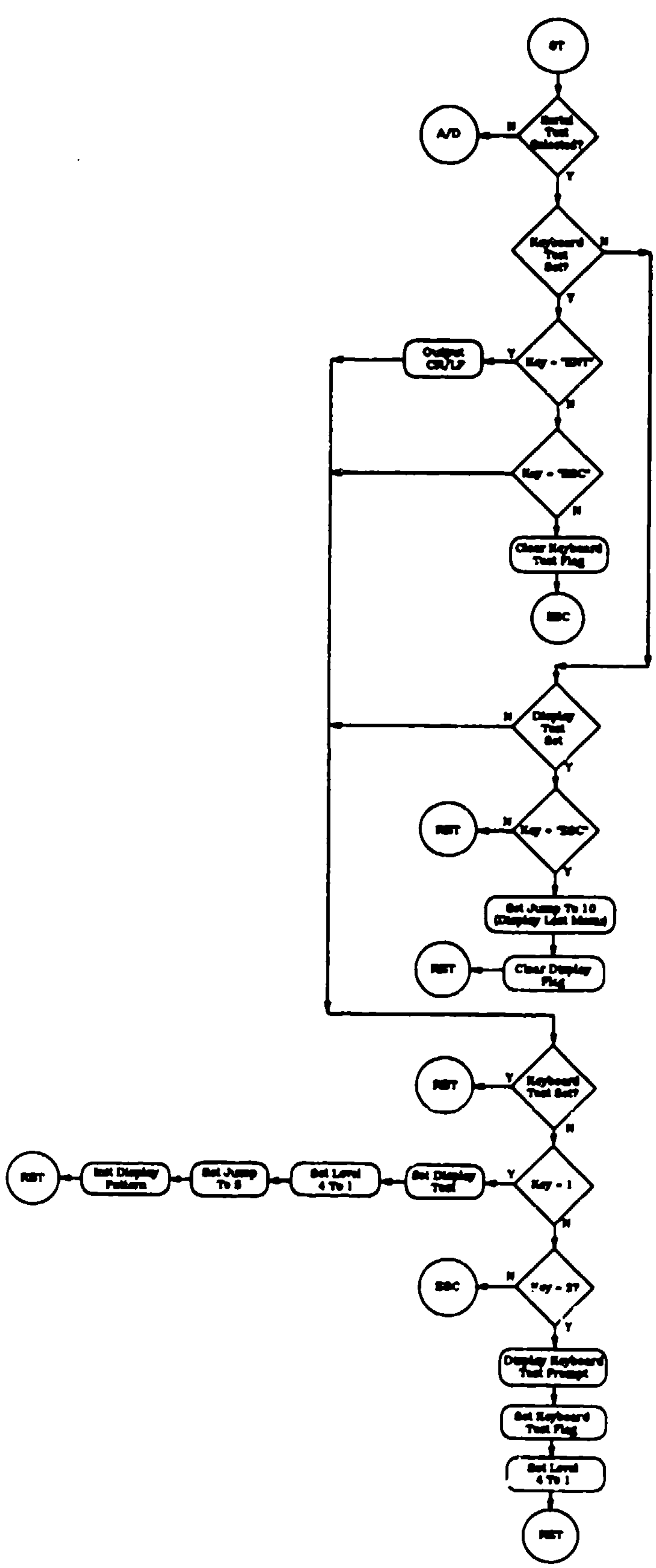



Figure 208. MENUPR Menu Processor Flow Chart (continued) (5 of 16)

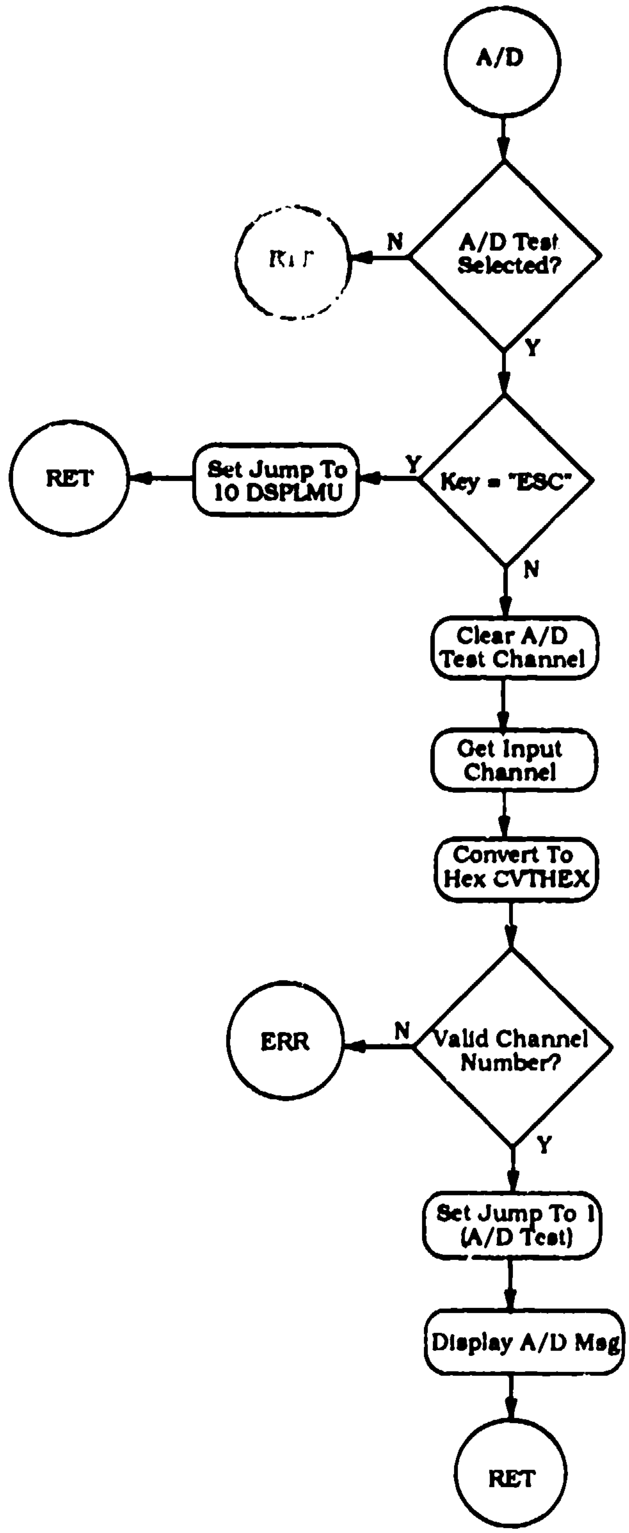



Figure 208. MENUPR Menu Processor Flow Chart (continued) (6 of 16)

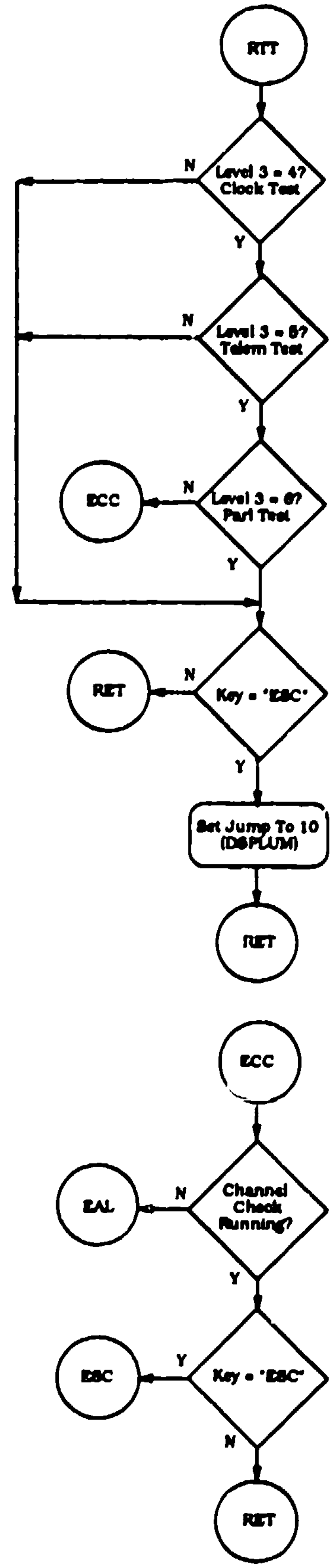



Figure 208. MENUPR Menu Processor Flow Chart (continued) (7 of 16)

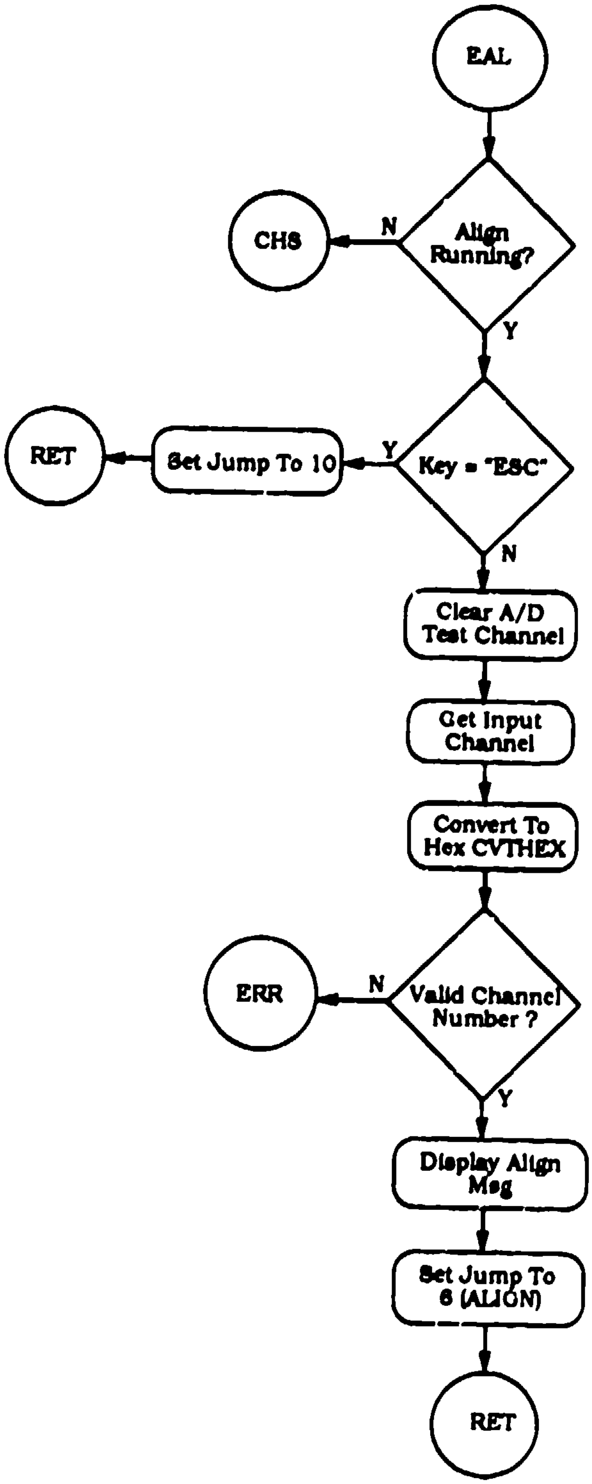



Figure 208. MENUPR Menu Processor Flow Chart (continued) (8 of 16)

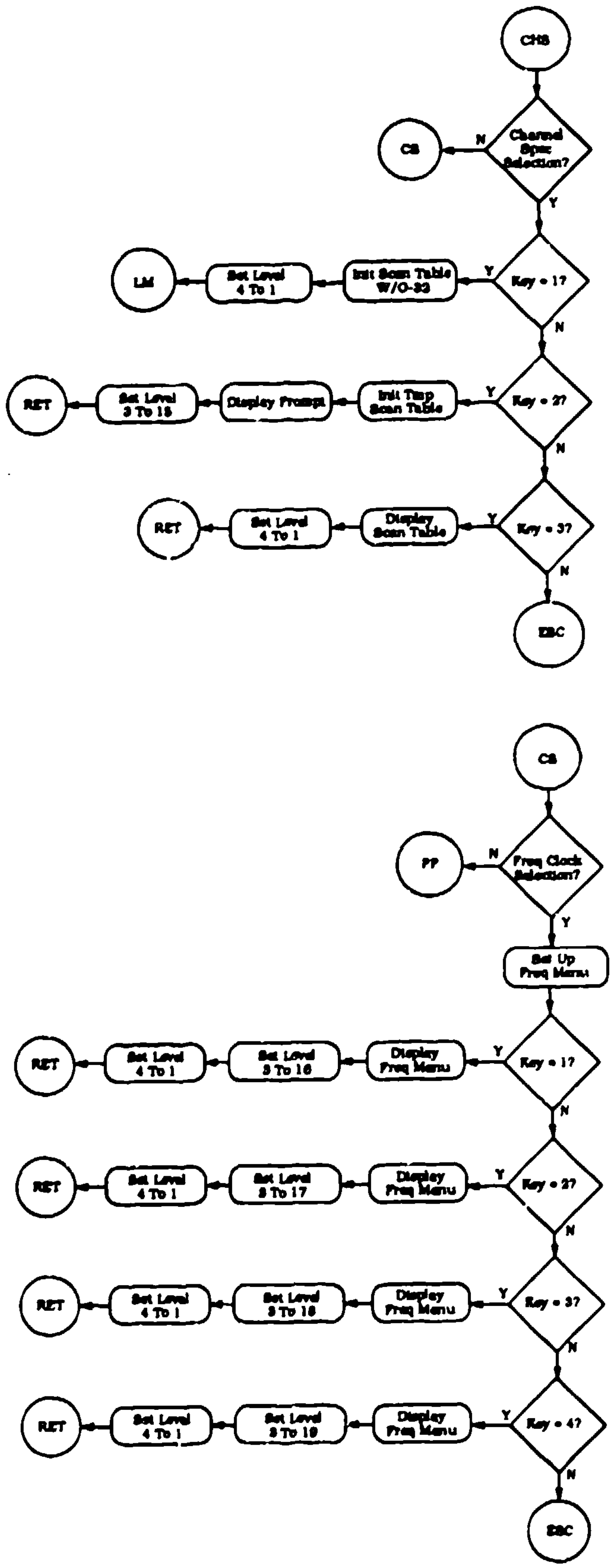



Figure 208. MENUPR Menu Processor Flow Chart (continued) (9 of 16)

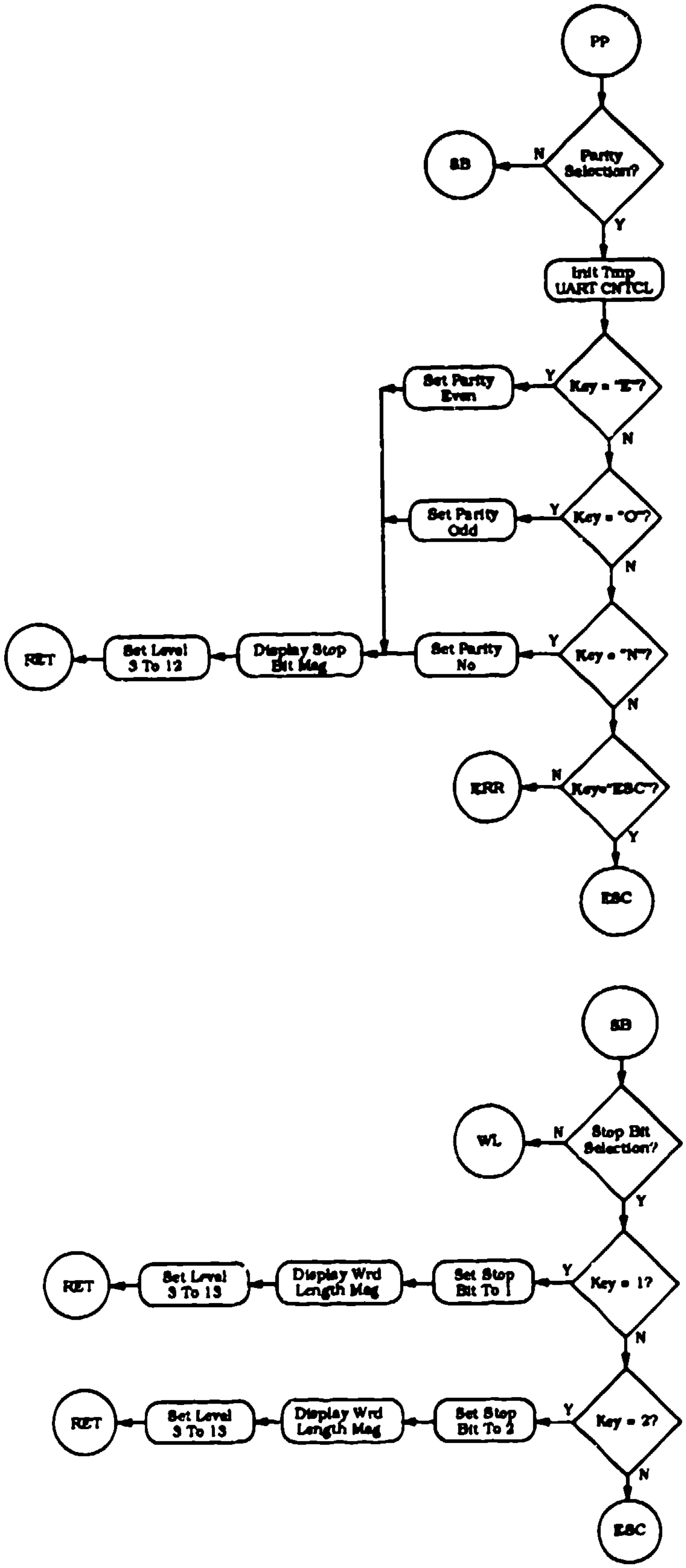



Figure 208. MENUPR Menu Processor Flow Chart (continued) (10 of 16)

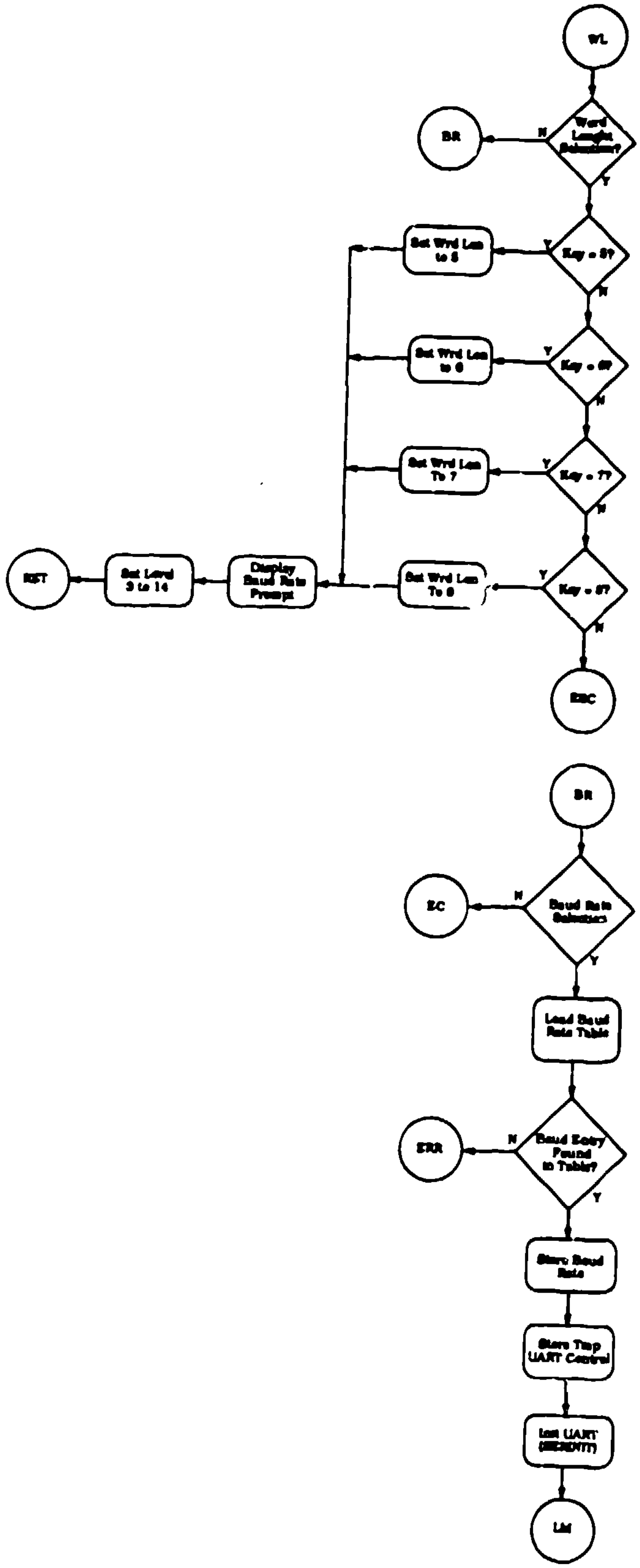



Figure 208. MENUPR Menu Processor Flow Chart (continued) (11 of 16)

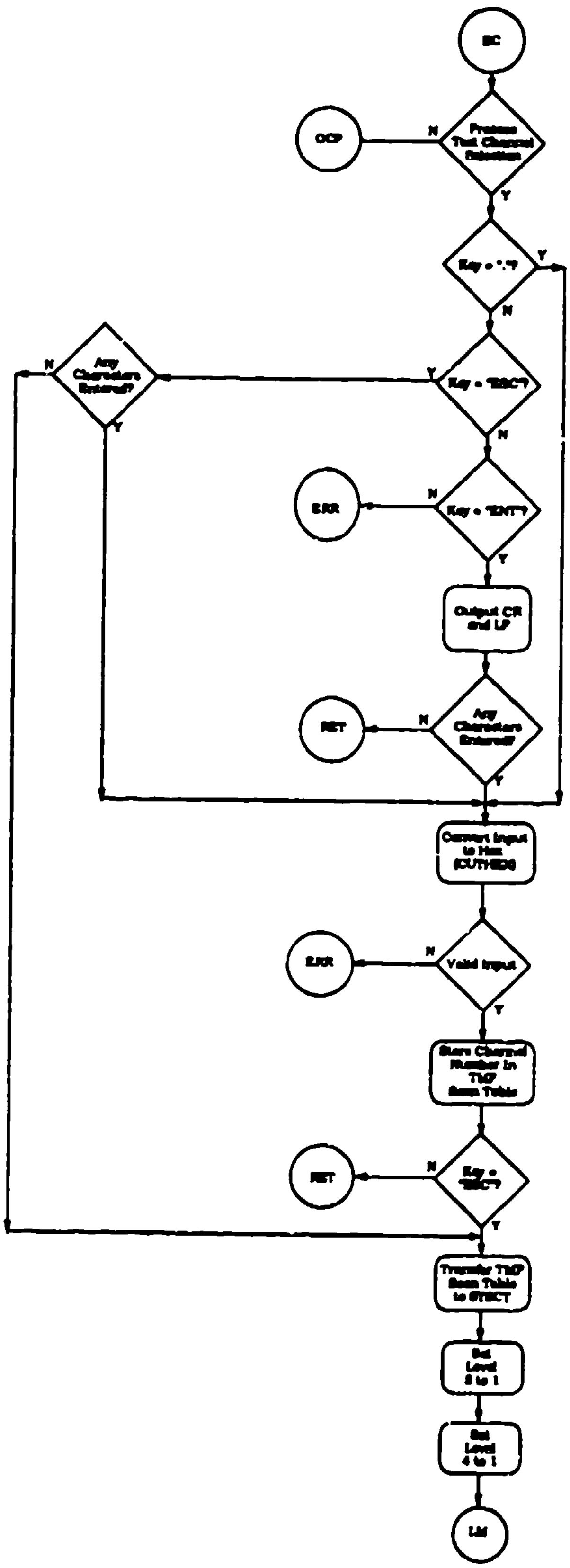



Figure 208. MENUPR Menu Processor Flow Chart (continued) (12 of 16)

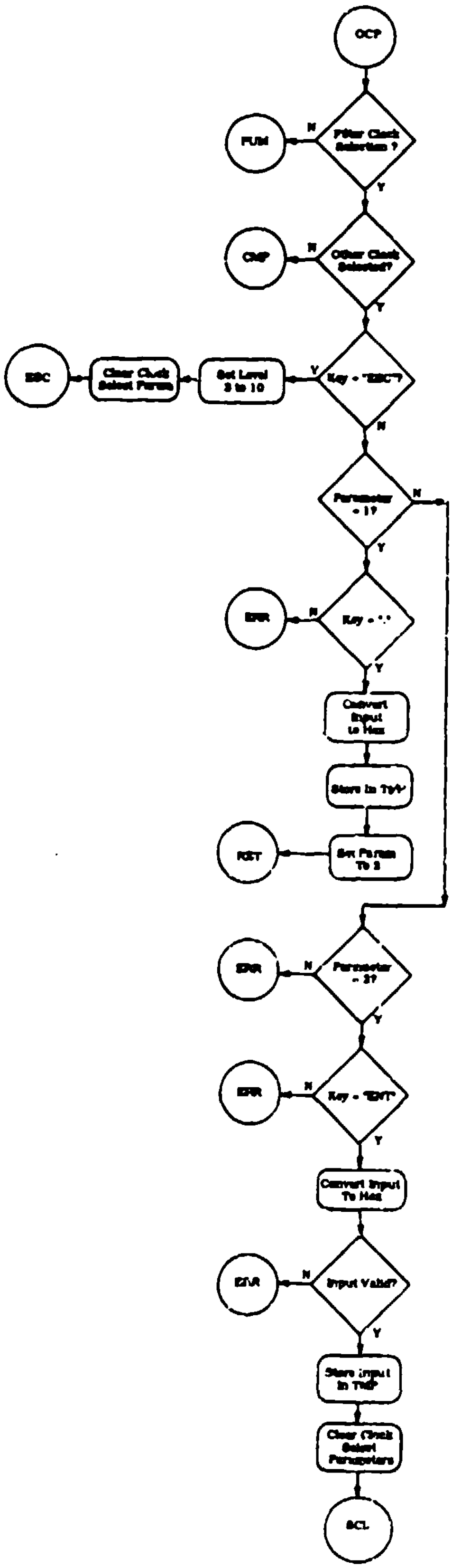



Figure 208. MENUPR Menu Processor Flow Chart (continued) (13 of 16)

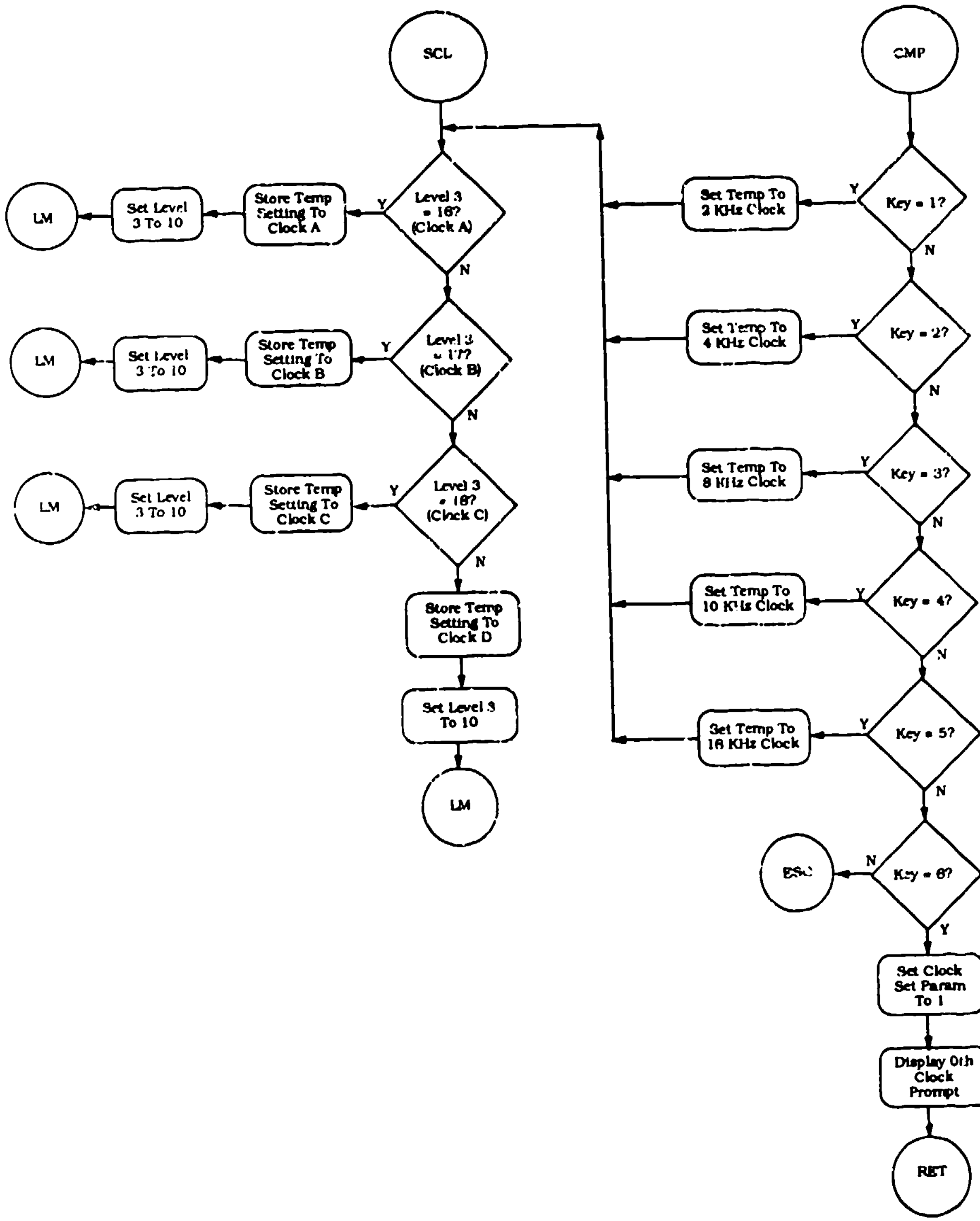



Figure 208. MENUPR Menu Processor Flow Chart (continued) (14 of 16)

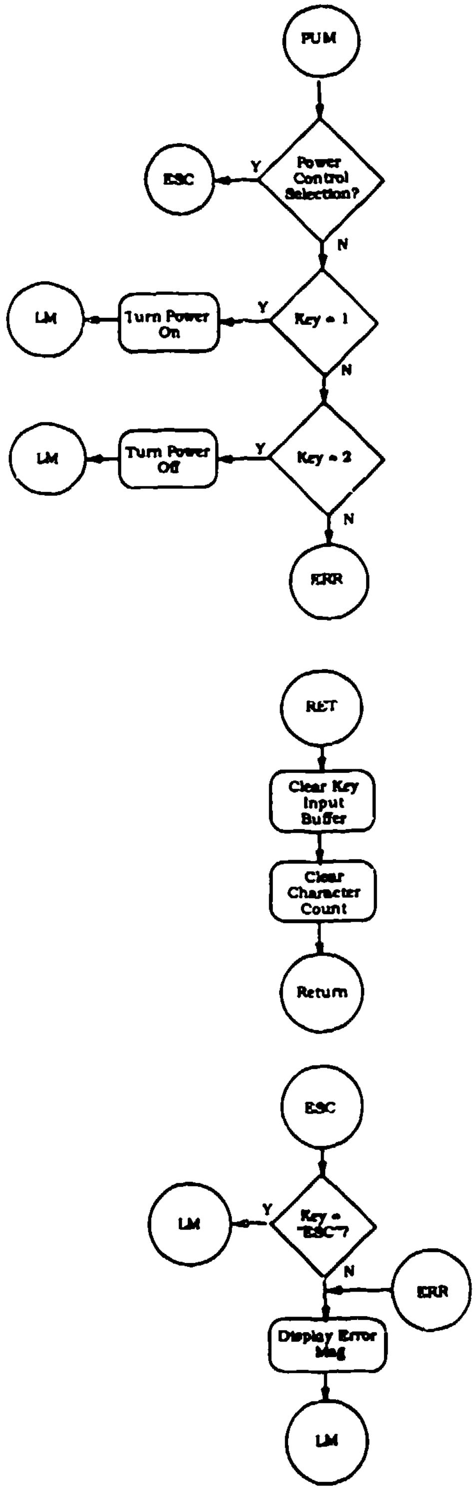



Figure 208. MENUPR Menu Processor Flow Chart (continued) (15 of 16)

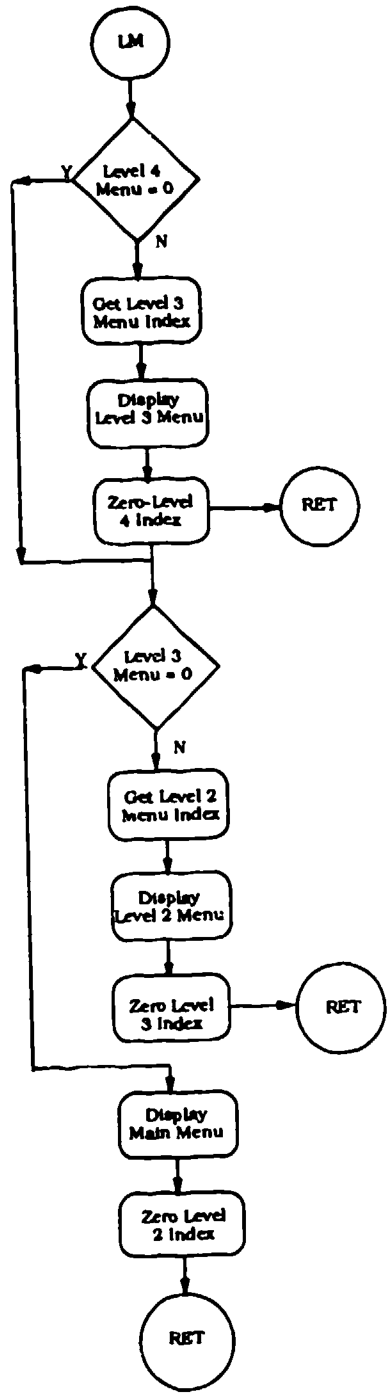



Figure 208. MENUPR Menu Processor Flow Chart (continued) (16 of 16)

# DSPSCT

This routine displays the scan table that is currently residing in the array STSCT. The scan table contains the mux channel numbers that are to be used for a test. These values are converted to decimal, then to ASCII before they are displayed. The call to DSPMSG performs the actual display.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: None

| REGISTERS: | CONTENTS: |
|---|---|
| REGISTERS USED: | |
| A0 | Index to DSPBUF |
| A1 | Index to STSCT |
| A2 | Pointer to the message to display |
| D0 | General purpose |
| D1 | General purpose |
| D2 | Number of characters to display |
| D3 | Word size for CVTDEC |
| D7 | Value to convert for CVTDEC and CVTASCI |

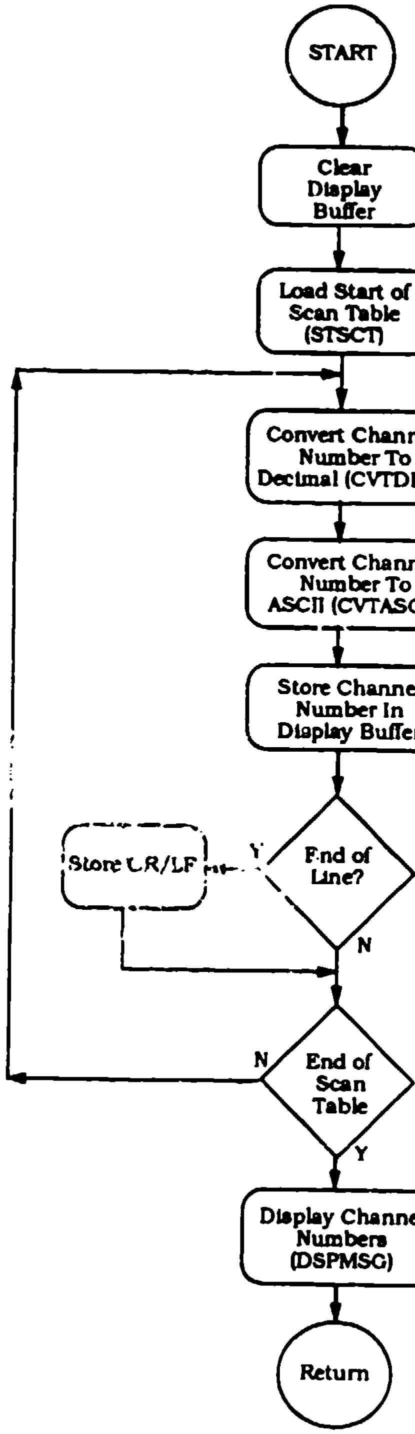



Figure 209. DSPSCT Display Scan Table Flow Chart

## CHCHECK

This routine performs a channel check of all 128 A/D channels. The check consists of reading the data from each channel in RCAL mode and saving it is CCBUF1. Then the routine reads the data from all channels and saves it in CCBUF2. Finally, the two arrays are compared and any two data values that show less than 10 counts difference in the positive or negative ranges are reported as errors by mux channel numbers. For each mux channel, there are four A/D channels used in the comparison. The bad channel numbers are converted to decimal (CTVDEC) and then to ASCII (CVTASCI) and finally displayed by DSPMSG.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: None

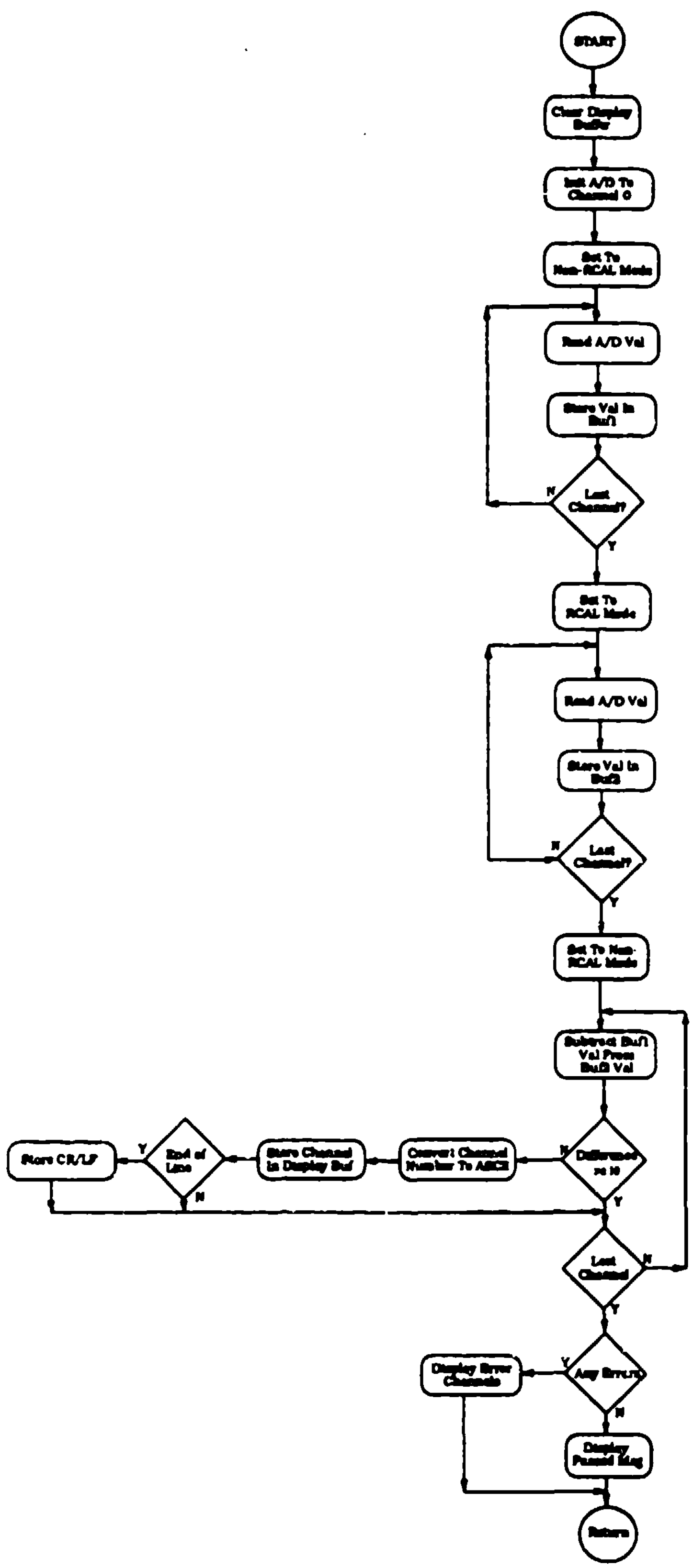



Figure 210. CHCHECK Channel Check Flow Chart

## CVTASCI

This routine converts the value in register D7 (1, 2, 3, or 4 bytes) to ASCII format and stores the converted value into the buffer pointed to by register A0. The value in register D3 indicates how many bytes to convert. The ASCII characters are stored into the buffer left justified. Register A0 is restored to its original value before it returns.

| | |
|---|---|
| INPUT PARAMETERS: | A0: Pointer to output buffer<br>D3: Number of bytes to convert<br>D7: Value to convert |
| OUTPUT PARAMETERS: | A0: Pointer to converted characters |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| D4 | Temporary storage (saved and restored) and all others listed above |

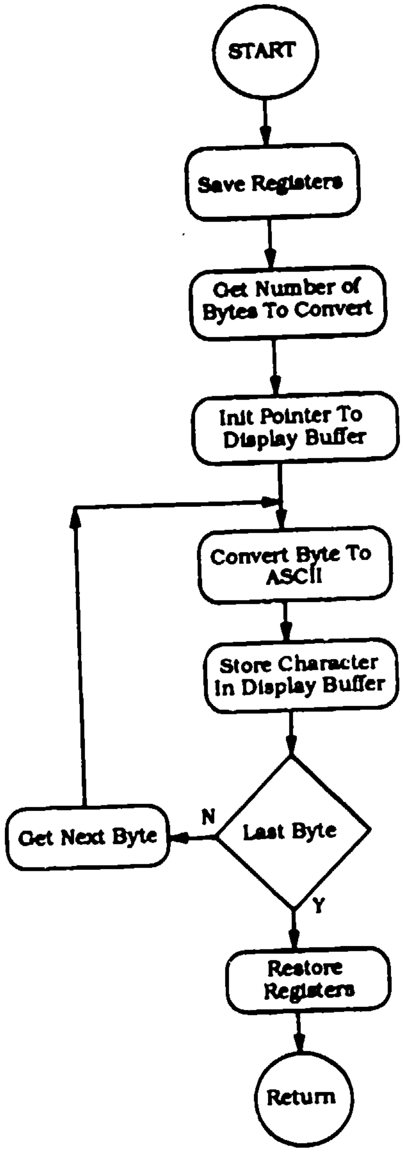



Figure 211. CVTASCI Convert to ASCII Flow Chart

## CVTHEX

This routine converts the ASCII characters in KEYBUF (pointed to by A6) first to an unpacked decimal number and then to a packed hexadecimal number. The result is stored in the word INVAL. CVTHEX will convert any number from 0 through 999 to 0 through 3E7. The input value must be pointed to by register A6 with the number of characters in D6.

| | |
|---|---|
| INPUT PARAMETERS: | A6: Pointer to input buffer<br>D6: Number of characters to convert |
| OUTPUT PARAMETERS: | INVAL: Converted hex value |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| A0 | Temporary intermediate value pointer |
| D0 | Temporary character count |
| D1 | General purpose |

NOTE: All registers are saved and restored by CVTHEX.

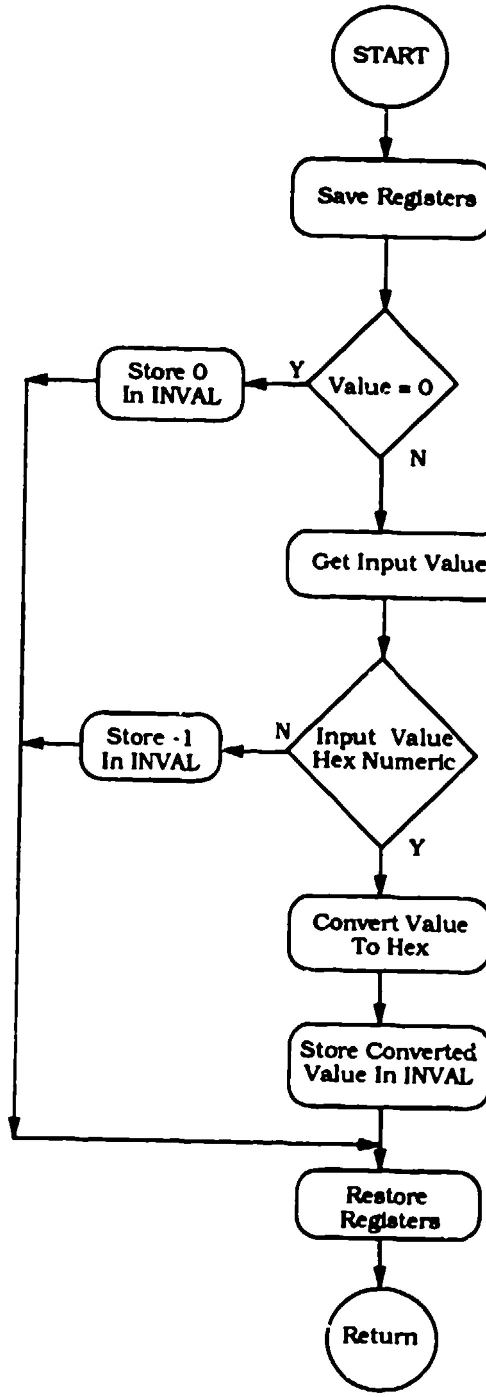



Figure 212. CVTHEX Convert to Hexadecimal Flow Chart

## TSTALL

This routine is called by MENUPR when the TEST ALL selection is made on the Memory Diagnostic menu. TSTALL calls all of the memory test routines and checks the error status (MEMFAIL). The routines that are called are PATTST, ADRTST, BU0TST, and BUATST. If the tests passed, this routine calls DSPMSG to report a passed status.

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | MEMFAIL: Test failed status |
| REGISTERS: | CONTENTS: |

REGISTERS USED:

| | |
|---|---|
| A2 | Pointer to display message |
| D2 | Display count |

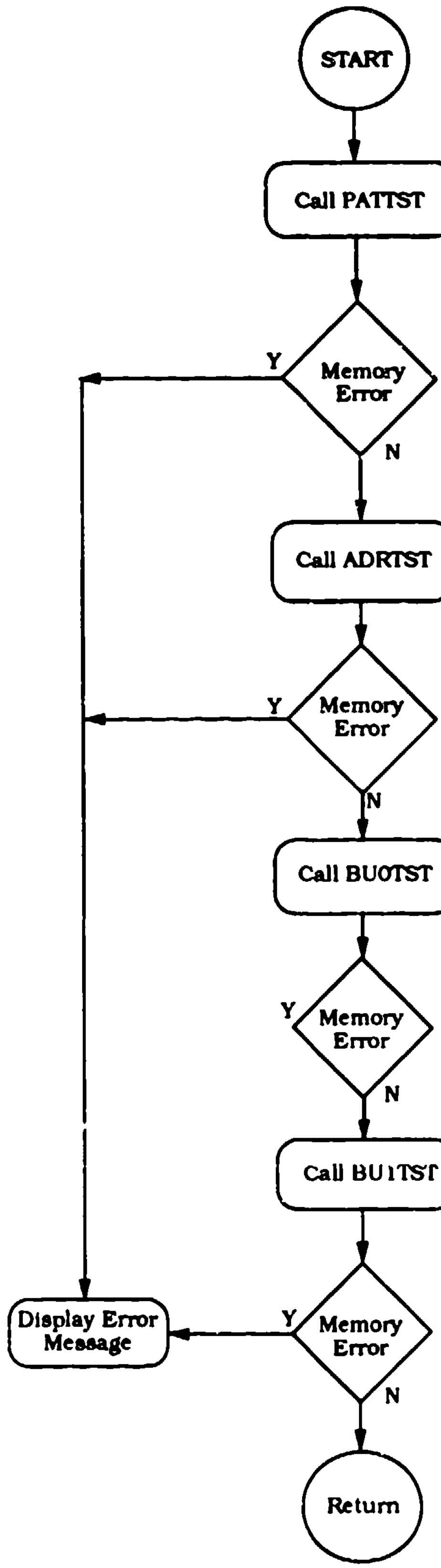



Figure 213. TSTALL Run All Memory Tests Flow Chart

# BU0TST

BU0TST performs a bubble zero test on memory. This is a word test that shifts a zero bit through each of the 16 bits of each word in memory. That memory that is tested is in the address range of 10000000 through 107EFF0 (STRAM, ENRAM).

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | MEMFAIL: Indicates that the test failed |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| A0 | Start of memory |
| A1 | End of memory |
| A2 | Current memory pointer |
| D0 | Test pattern |
| D3 | Word length for CVTASCI |

NOTE: All registers are saved and restored by BU0TST.

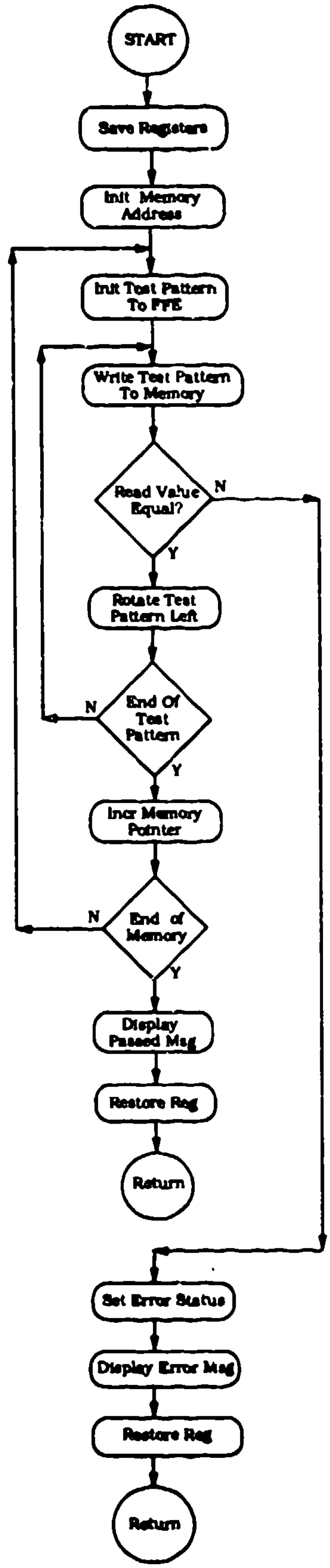



Figure 214. BU0TST Bubble Zero Memory Test Flow Chart

# BU1TST

BU1TST performs a bubble one test on memory. This is a word test that shifts a one bit through each of the 16 bits of each word in memory. The memory that is tested ranges from addresses 10000000 through 107EFF0 (STRAM, ENRAM).

INPUT PARAMETERS: None

OUTPUT PARAMETERS: MEMFAIL: Indicates that test failed

| REGISTERS: | CONTENTS: |
|---|---|

REGISTERS USED:

| | |
|---|---|
| A0 | Start of memory |
| A1 | End of memory |
| A2 | Current memory pointer |
| D0 | Test pattern |
| D3 | Word length for CVTASCI |

NOTE: All registers are saved and restored by BU1TST.

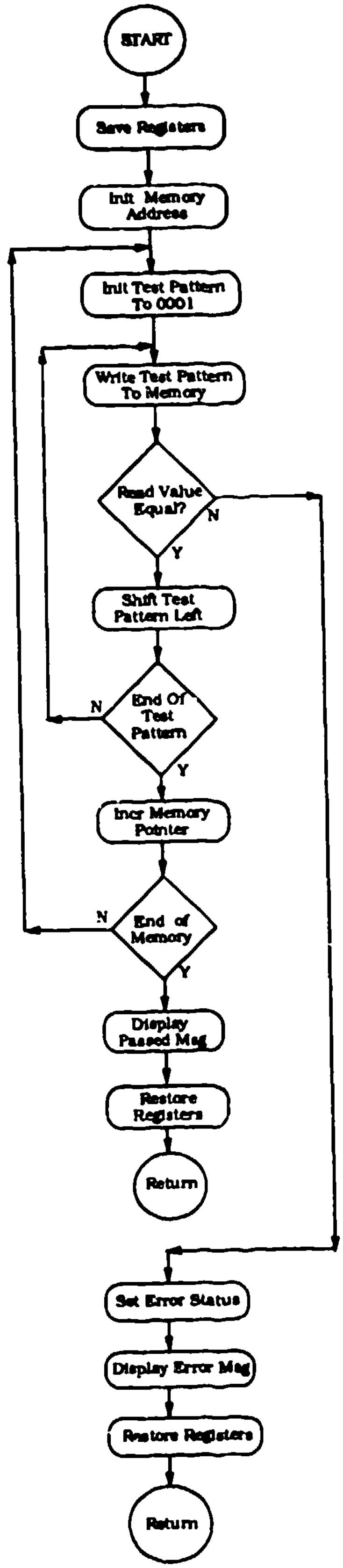



Figure 215. BU1TST Bubble One Memory Test Flow Chart

## ADRTST

ADRTST performs an address test on memory. This tests consists of writing the long word address of a memory location into its own memory location. The memory that is tested ranges from addresses 10000000 through 107EFF0 (STRAM, ENRAM).

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | MEMFAIL: Indicates that the test failed |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| A0 | Start of memory |
| A1 | End of test memory |
| A2 | Current memory pointer |
| D0 | Address value |
| D3 | Word length for CVTASCI |

NOTE: All registers are saved and restored by ADRTST.

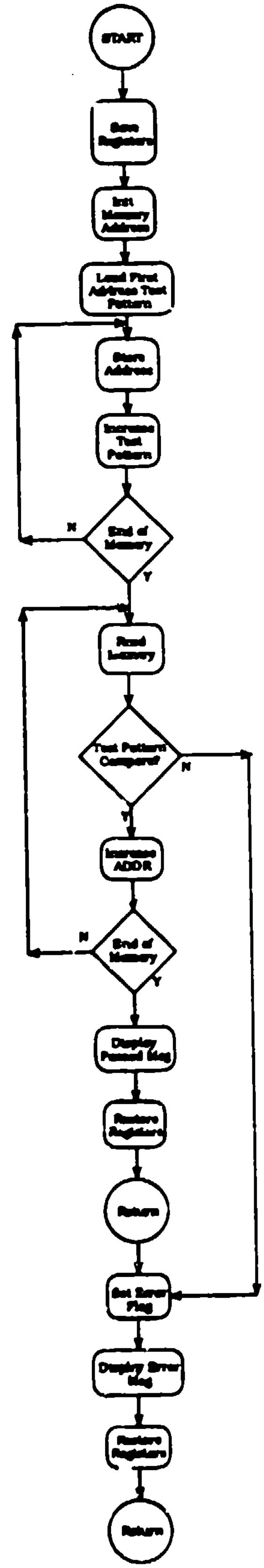


Figure 216. ADRTST Address (In Address) Memory Flow Chart

# PATTST

PATTST performs a pattern test on memory. This test consists of writing several byte data patterns to memory and checking the results. The data patterns include: AA, 55, FF, and 00. The memory that is tested ranges from address 10000000 through 10FEFF0 (STRAM, ENRAM).

INPUT PARAMETERS: None

OUTPUT PARAMETERS: MEMFAIL: Indicates that the test failed

| REGISTERS: | CONTENTS: |
|---|---|

REGISTERS USED:

| | |
|---|---|
| A0 | Start of memory |
| A1 | End of memory |
| A2 | Current memory pointer |
| D0 | Data pattern value |
| D3 | Word length for CVTASCI |

Note: All registers are saved and restored by PATTST.

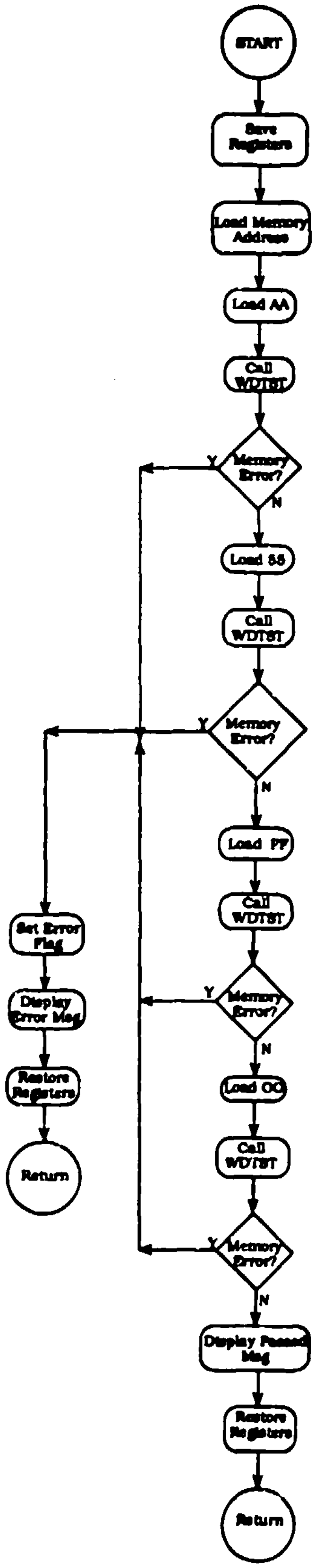



Figure 217. PATTST Pattern Memory Test Flow Chart

# GETKEY

GETKEY reads the MFP serial port to fetch the character that was entered on the handheld terminal. This routine is called by KEYINT when a keyboard interrupt is received. This ASCII character red from the data port (UDR) is stored into KEY. If the character is an alphabetic or numeric character (A-Z, 0-9), it is stored into the input buffer indexed by register A6. The character count in register D6 is also updated. If the input is a control character (scroll up, scroll down, ENT, ESC, dot, hyphen), it is just returned in KEY and not stored. If it was the delete key, GETKEY deletes the last character on the display.

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | KEY: Input character<br>A6: Input character buffer<br>D6: Character count |

No additional registers used.

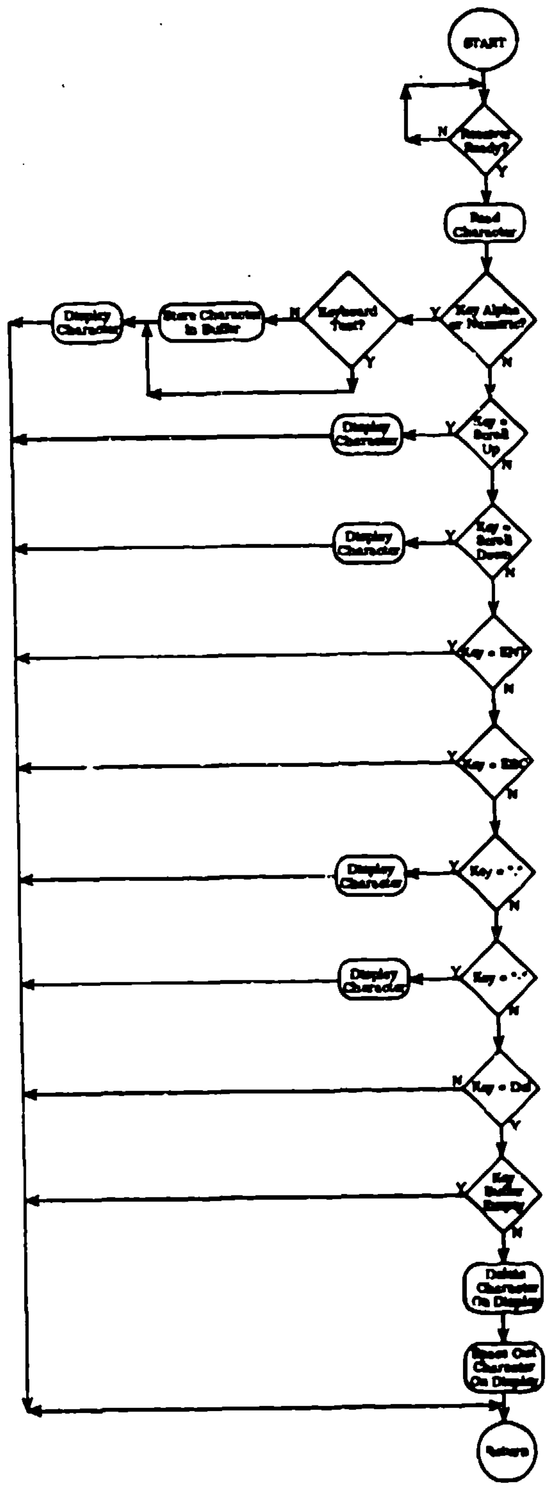



Figure 218. GETKEY Input Character Key Fetch Flow Chart

## DSPMSG

DSPMSG displays the message on the handheld terminal that is contained in the buffer pointed to by registers A2 for the number of characters in D2. The data must already be in ASCII format (if any additional messages are added to the system message bank in the future).

| | |
|---|---|
| INPUT PARAMETERS: | A2: Pointer to output buffer (message start)<br>D2: Number of output characters from message bank |
| OUTPUT PARAMETERS: | None |
| REGISTERS: | CONTENTS: |
| REGISTER USED: | |
| D0 | Temporary character count |

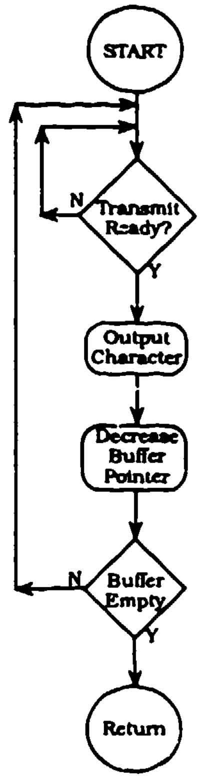



Figure 219. DSPMSG Display Message Flow Chart

## CLRSC

This routine clears the screen and rests the cursor and internal address counters to the beginning of the display on the handheld screen. The code that is sent to the terminal for this activity is 0C. The delay shown in Figure 220 is required for the terminal to clear its memory before the system tries to write to it.

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | None |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| D0 | Temporary storage |
| D1 | Temporary storage |

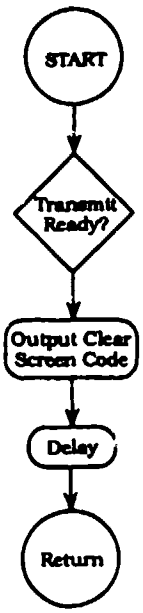



Figure 220. CLRSC Clear Screen of Handheld Terminal Flow Chart

## CVTHEX

This routine converts a hexadecimal number from 0 through 7EFF to a decimal number.

| | |
|---|---|
| INPUT PARAMETERS: | D7: Hex value to be converted (word) |
| OUTPUT PARAMETERS: | D7: Decimal converted value |
| REGISTERS: | CONTENTS: |
| REGISTER USED: | |
| D0 | Temporary storage |

NOTE: D0 is saved and restored by CVTHEX.

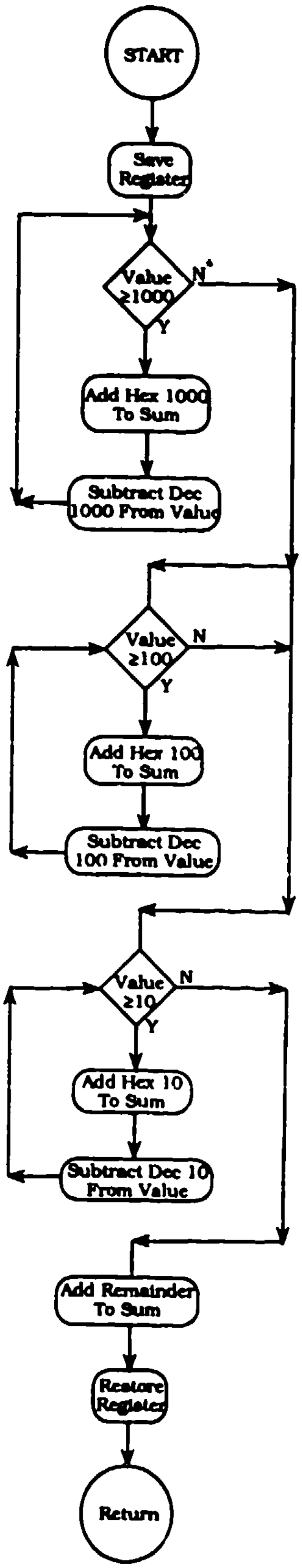



Figure 221. CVTDEC Convert to Decimal Flow Chart

# CRLFO

CRLFO outputs a carriage return character and a line feed character to the serial port of the MFP. This controls the cursor position on the handheld terminal.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: None

REGISTERS USED: None

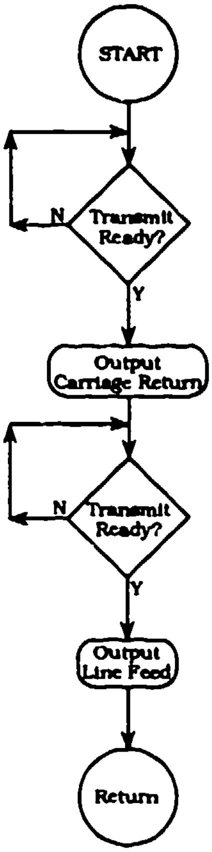



Figure 222. CRLFO Carriage Return/Line Feed Output Flow Chart

## TMRINIT

TMRINIT initializes the four filter clocks to their predetermined frequency values. The values of the prescale and count values are determined during the power-up initialization or during the Clock Setting menu processing. The prescale values are in PRSCA, PRSCB, and PRSCCD for clocks A, B, and CD. The count values are in FLCNTA, FLCNTB, FLCNTC, and FLCNTD. Before the clocks are initialized, they are reset.

| | |
|---|---|
| INPUT PARAMETERS: | FLCNTA: Count values for the four filter clocks<br>FLCNTB<br>FLCNTC<br>FLCNTD |
| OUTPUT PARAMETERS: | None |
| REGISTERS USED: | None |

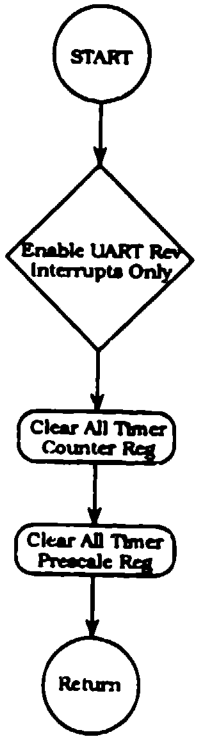



Figure 223. TMRINIT Filter Clock Initialization Flow Chart

## SERINIT

SERINIT initializes the UART on the MFP for the serial communications to the handheld terminal. The values for the UART control (UCNTRL) and baud rate (BAUD) are determined during the power-up initialization or during the SERIAL DEF menu processing.

INPUT PARAMETERS: UCNTRL: Control value for the UART
BAUD: Baud rate code

OUTPUT PARAMETERS: None

REGISTERS USED: None

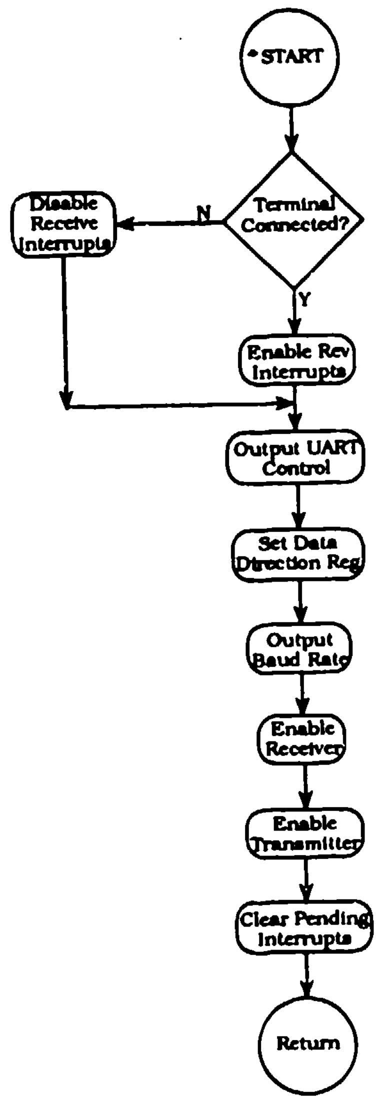



Figure 224. SERINIT Serial Port Initialization Flow Chart

## PARLTST

PARLTST performs the power-up diagnostic on the parallel port. It outputs a pattern to the port and reads and compares that value. Since this is an internal-only test, the port is set to output mode during the entire test. The synchronization code FAF30000 is output first so that the DRASS will not attempt to process the data. The data pattern that is used in 0000000, 01010101, ... 0F0F0F0F.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: DSTAT: Bit 4 is set is there is a test failure

| REGISTERS: | CONTENTS: |
|---|---|
| REGISTERS USED: | |
| D0 | Temporary register |
| D1 | Contains the data test pattern |
| D2 | Test counter |
| D7 | Input value |

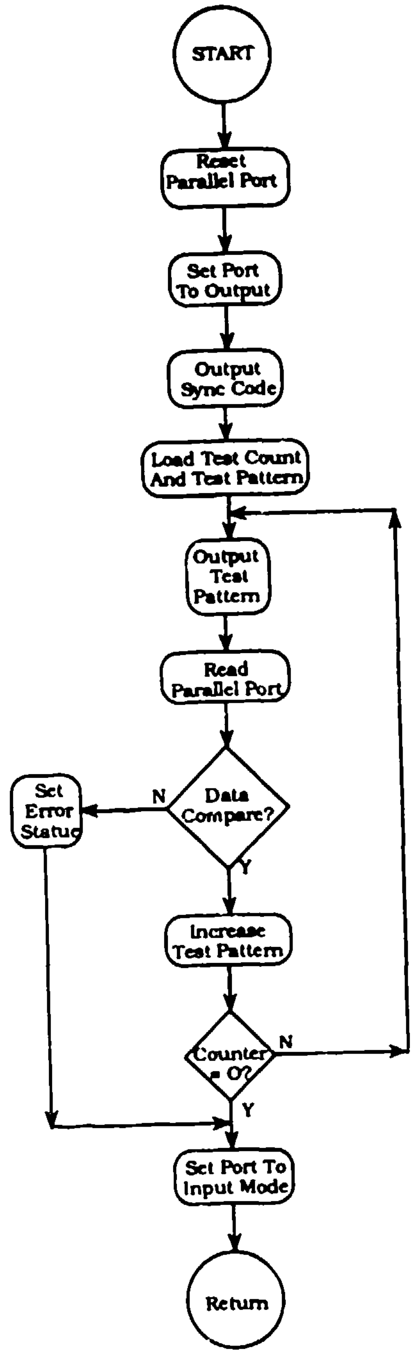



Figure 225. PARLTST Parallel Port Loopback Test Flow Chart

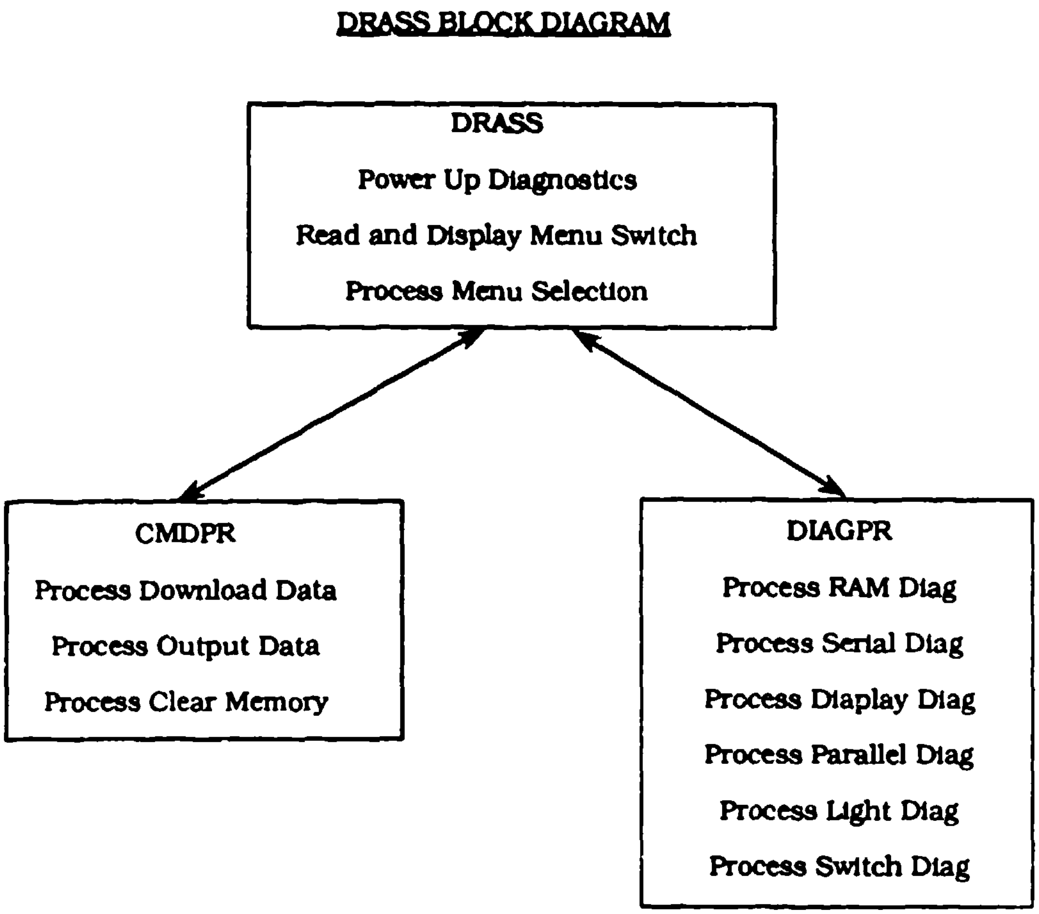



Figure 226. DRASS System (Top Level) Flow Chart

The DRASS program is structured as a command processor. The commands are received by BUTINT, which services the interrupt generated by the front panel buttons and passes the command to CMDRPR or DIAGPR for processing. The position of the RUN/TEST toggle switch determines which routine services the command. The position of the thumbwheel switches, and the RUN/TEST toggle switch, when the START button is pressed, determines what command will be performed. When a command is selected, it is processed until it is completed, or the HALT button is pressed.

## MODULE HIERARCHY

DRASS (Initialization)

| | |
|---|---|
| ROMTST | Power up ROM test |
| SERTST | Power up serial test |
| PARLTST | Power up parallel test |
| LITTST | Power up light test |
| RAMTST | Power up RAM test |
| DISPLAY | Display message |

MAINLP (Process Server)

| | |
|---|---|
| CMDPR | Check RUN/TEST switch and perform ADAM data transfer (DLDATA) |
| CMDPR2 | Perform serial output of data (OUTDATA) |
| CMDPR3 | Erase data memory (CLRMEM) |
| DIAGPR | Perform RAM diagnostic (RAMDIAG) |
| DIAGPR1 | Perform serial diagnostic (SERDIAG) |
| DIAGPR2 | Perform display diagnostic (DSPDIAG) |
| DIAGPR3 | Perform parallel diagnostic (PARDIAG) |
| DIAGPR4 | Perform light diagnostic (LITDIAG) |
| DIAGPR5 | Perform switch diagnostic (SWTDIAG) |

BUTINT (Button Interrupt Service Routine)

Reads the button pressed. If it was the START button, store the thumbwheel setting in CMD. If the HALT button was pressed, set the halt flag (HLTFLG). If the CONTINUE button was pressed, set the continue flag (CONTFL).

## REGISTER ASSIGNMENTS

| | |
|---|---|
| A0 | Temporary index register |
| A7 | Stack pointer |
| D0 | Temporary data register |

## STATUS BYTE DEFINITION

### POWER UP DIAGNOSTICS STATUS (DSTAT)

| | |
|---|---|
| Bit 0 | ROM error |
| Bit 1 | Serial error |
| Bit 2 | Parallel error |
| Bit 3 | RAM error |

### SYSTEM STATUS (STAT)

| | |
|---|---|
| Bit 1 | DRASS on-line/off-line |
| Bit 2 | DRASS read/write mode |
| Bit 3 | Memory full status |
| Bit 4 | Test fail status |
| Bit 5 | Busy status |
| Bit 6 | Test passed status |

The remainder of this appendix presents the next several levels of flow charts for all operations within the DRASS. Following the opening top level DRASS system flow chart, there are 21 more flow charts with introductory test for them. Additional information about the events taking place in these routines can be found in the comments (right column) on the 68020 assembly source code listings of this DRASS resident software.

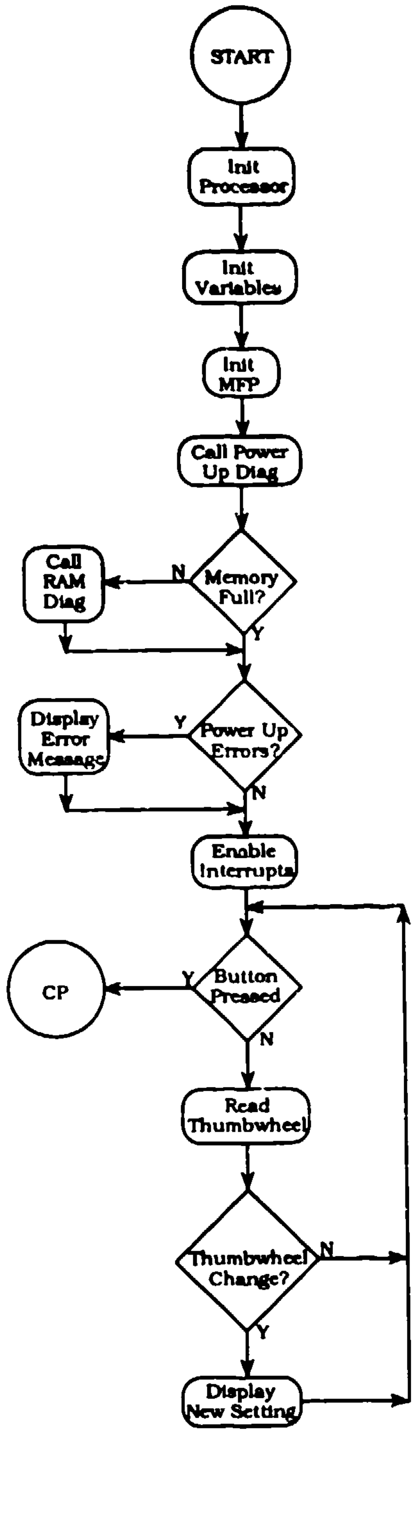



Figure 227. DRASS Initialization Flow Chart (1 of 2)

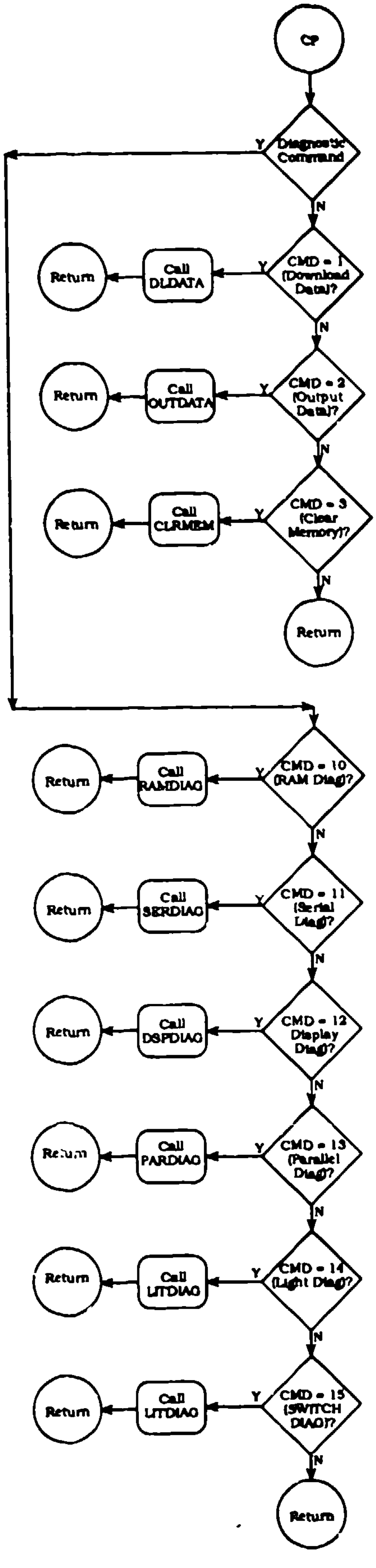



Figure 227. DRASS Initialization Flow Chart (continued) (2 of 2)

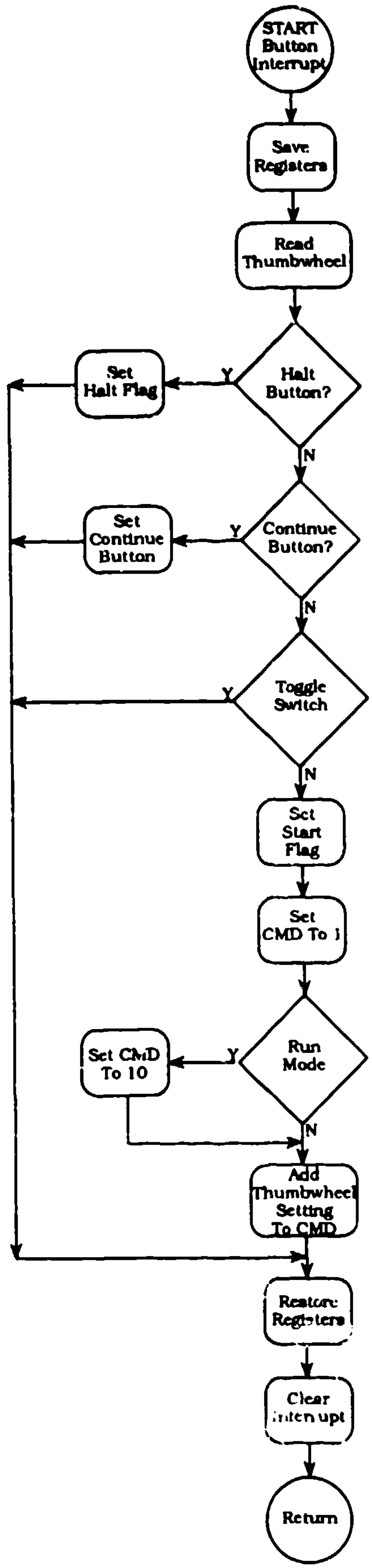



Figure 228. BUTINT Button Interrupt Flow Chart

## CLRMEM

CLRMEM purges the data from the RAM modules in the DRASS. The data are erased from addresses 1000000 (STRAM) through 107EEFE (ENRAM) and the synchronization codes for precalibration mode data, posttest calibration data, and test data. Before exiting, the memory full status is reset.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: None

| REGISTERS: | CONTENTS: |
|---|---|

REGISTERS USED:

| | |
|---|---|
| A0 | Start of memory |
| A1 | End of memory |
| D0 | Value that is written to memory |

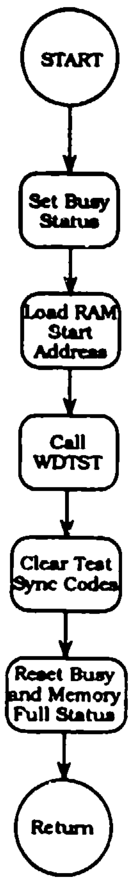



Figure 229. CLRMEM Clear Memory Flow Chart

## DISPLAY

DISPLAY updates the 16 character display on the front panel of the DRASS. The message that is output t it is pointed to by register A0. Sixteen characters arc a[illegible] o.ıtput. Each time DISPLAY is called, the display is initialized before the characters are outpu'.

| | |
|---|---|
| INPUT PARAMETERS: | Register A0: Points to the message buffer |
| OUTPUT PARAMETERS: | None |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| D0 | Temporary delay counter |
| D6 | Character counter |

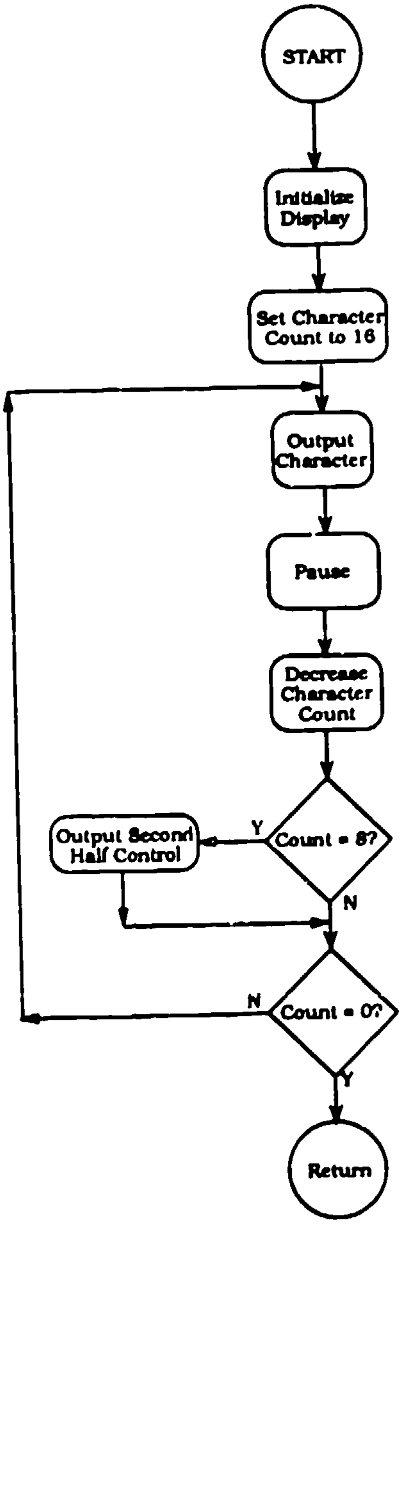



Figure 230. DISPLAY Update Front Panel Display Flow Chart

# PARDIAG

PARDIAG performs the DRASS parallel diagnostic. It executes in conjunction with the ADAM parallel diagnostic. When ADAM sends a data pattern over the parallel port to the DRASS, it echoes that data back. The test starts when the Sync Code FAF30001 is received. The test is exited when the HALT button is detected (HLTFLG). Since timeouts are built into the ADAM parallel diagnostic, the DRASS diagnostic must be activated first.

| | |
|---|---|
| INPUT PARAMETERS: | HLTFLG: Halt the test when set |
| OUTPUT PARAMETERS: | STAT: Bit 5 - busy status |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| D5 | Parallel port handshake status |
| D6 | Input/output data |

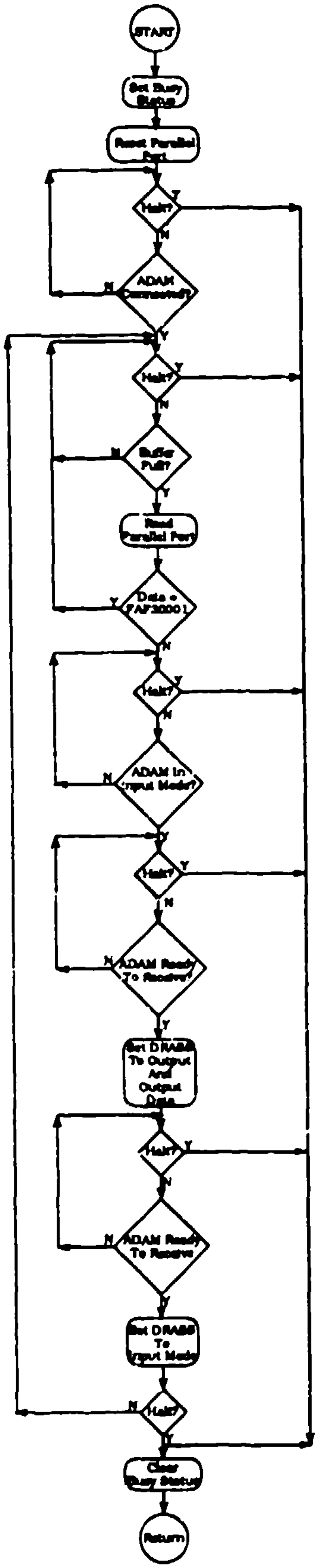



Figure 231. PARDIAG Parallel Port Diagnostic Flow Chart

## DSPDIAG

DSPDIAG performs the diagnostic to the front panel display. It outputs a series of test patterns (TEXT1 through TEXT 7) to the display until the HALT button is detected (HLTFLG). There is a 3-second delay between outputs of each test pattern. DISPLAY performs the actual display.

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | STATUS: Bit 5 - busy status |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| A0 | Pointer to the test pattern |
| D0 | Temporary delay counter |
| D2 | Saved pointer to the test pattern |

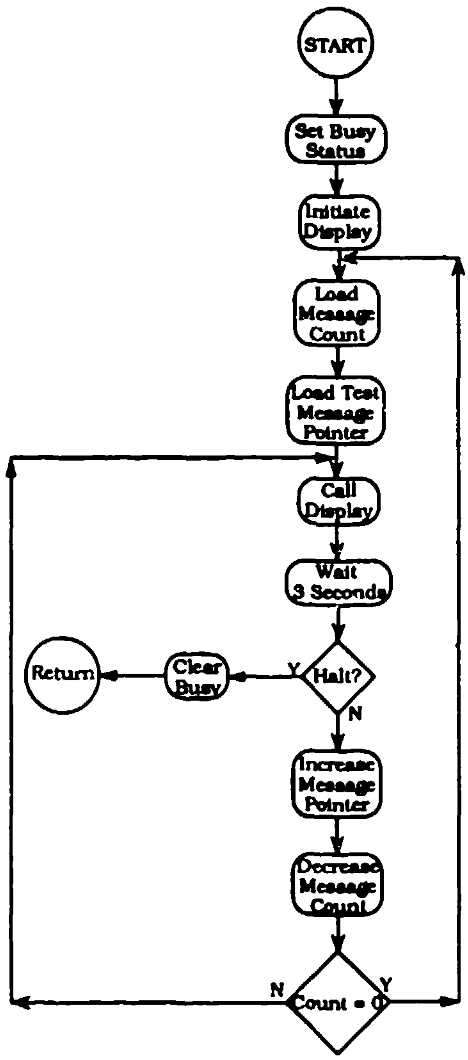



Figure 232. DSPDIAG Display Diagnostic Flow Chart

## LITTST

LITTST performs a test of the front panel lights during power-up diagnostics. The tests consists of sequencing each of the status lights. Since there is no status from the lights, DSTAT is not updated.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: None

| REGISTERS: | CONTENTS: |
|---|---|

REGISTERS USED:

| | |
|---|---|
| D0 | Temporary delay counter |
| D1 | Output control byte |
| D2 | Bit set value for the control byte |

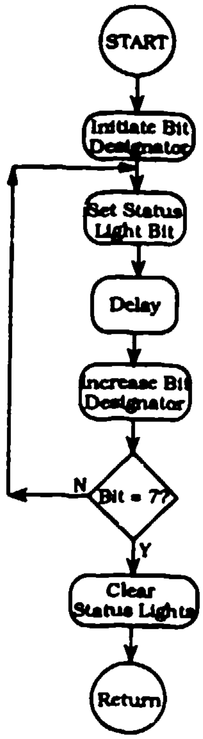



Figure 233. LITTST Panel Lights Diagnostic Flow Chart

## ROMTST

This routine calculates a sumcheck of the PROM and compares that value with the sumcheck stored in PROM at address FFFF. If the test fails, bit 0 in DSTAT is set to be checked after power-up diagnostics is complete.

| | |
|---|---|
| INPUT PARAMETERS: | Sumcheck contained at address FFFF |
| OUTPUT PARAMETERS: | DSTAT: Contains the pass/fail result of this test |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| A0 | Running index through PROM |
| D0 | End address of PROM |
| D1 | Running checksum total |

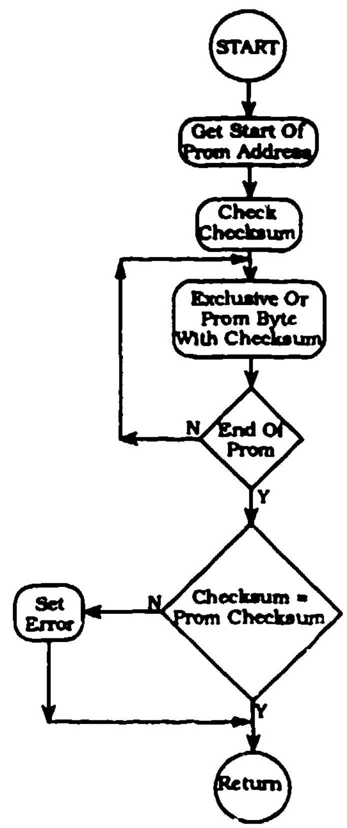



Figure 234. ROMTST PROM Checksum Diagnostic Flow Chart

## SERTST

This routine performs the power-up diagnostic on the UART in the MFP. It performs an internal loop back test with a canned message. SERTST does not utilize the receive interrupt but it does test the receive status. The data format and baud rate used during the test is the same as that which is set up during the system initialization.

| | |
|---|---|
| INPUT PARAMETERS: | None |
| OUTPUT PARAMETERS: | DSTAT: Power-up diagnostic status<br>SERTST: 1 - Frame error<br>2 - Parity error<br>3 - Overrun error |

| REGISTERS: | CONTENTS: |
|---|---|

REGISTERS USED:

| | |
|---|---|
| A0 | Index to test pattern |
| D0 | Test data character |
| D1 | Temporary UART status |
| D7 | Character count |

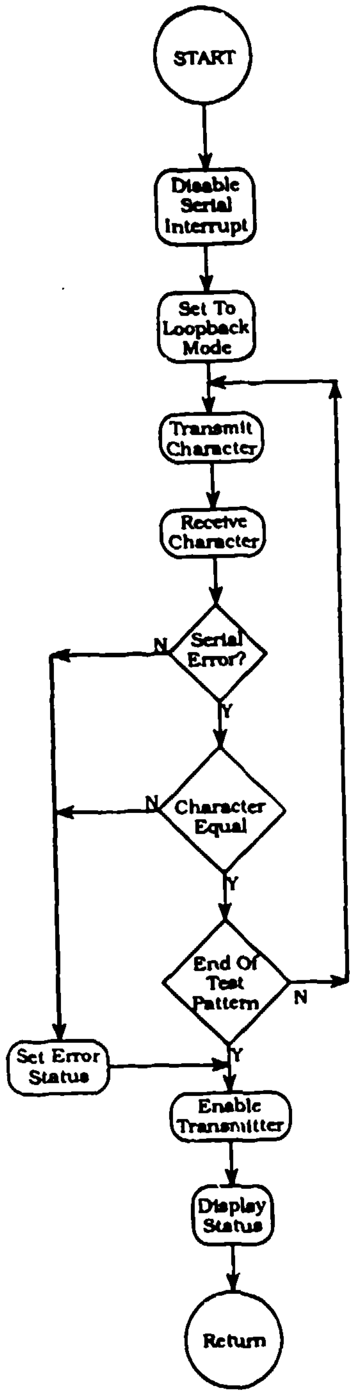



Figure 235. SERTST Serial Port Diagnostic Flow Chart

## WDTST

This routine is used by RAMTST, RAMDIAG, and CLRMEM to write a data pattern to memory and check it. WDTST does a byte write and read of the data in register D0 to the address in register A2 up to the address in A1. If there are any errors in the compare, register D7 is set to FF, and register A2 contains the error address.

INPUT PARAMETERS:

| | |
|---|---|
| A0 | Start of test memory |
| A1 | End of test memory |
| D0 | Test data pattern (byte) |

OUTPUT PARAMETERS:

| | |
|---|---|
| A2 | Memory error address |
| D0 | Test pattern written |
| D1 | Data read from memory |
| D7 | Error flag (set to FF if failed) |

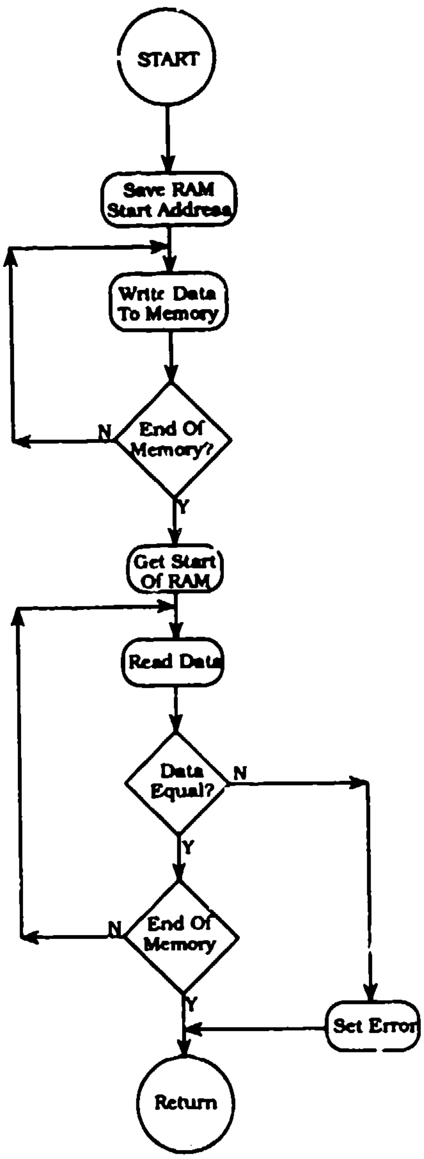



Figure 236. WDTST Memory Word Test Subroutine Flow Chart

## PARLTST

PARLTST performs the power-up diagnostic on the parallel port. It outputs a data pattern to the port and reads and compares that value. Since this is an internal test only, the port is set to output mode during the entire test. The data pattern that is used is : 00000000, 01010101, ... 0F0F0F0F.

INPUT PARAMETERS: None

OUTPUT PARAMETERS: DSTAT: Bit 2 is set if there is a test failure

REGISTERS USED:

| REGISTERS: | CONTENTS: |
|---|---|
| D0 | Temporary register |
| D1 | Contains the data pattern |
| D2 | Test counter |
| D7 | Input value |

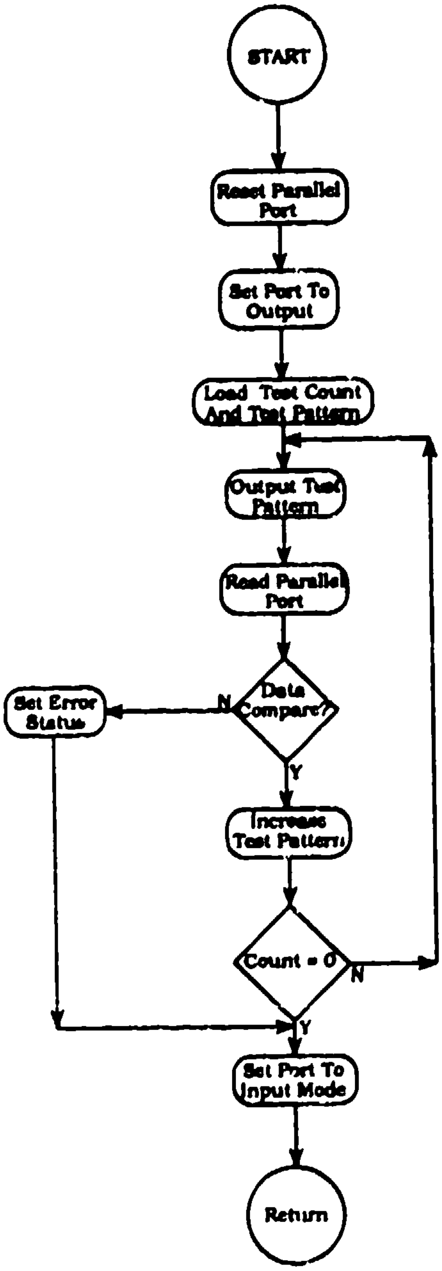



Figure 237. PARLTST Parallel Port Self-Diagnostic Flow Chart

# RAMTST

This routine tests both the SRAM and the Cache RAM in the system during power-up diagnostics. It performs a byte write and read of the memory with the data patterns AA, 55, FF, and 00. The memory that is tested is from 1000000 (STRAM) through 107FFFE (ENRAM), and from 10100 (STCACH) through 10700 (ENCACH). The abbreviated regions prevent the destruction of system parameters and the system stack. RAMTST uses the routine WDTST to do the actual memory accesses.

INPUT PARAMETERS: D7: Test status returned by WDTST

OUTPUT PARAMETERS: DSTAT: Test status for power-up diagnostics

| REGISTERS: | CONTENTS: |
|---|---|

REGISTERS USED:

| REGISTERS | CONTENTS |
|---|---|
| A0 | Start of memory to test |
| A1 | End of memory to tests |
| D0 | Test data pattern |
| D7 | Test status from WDTST |

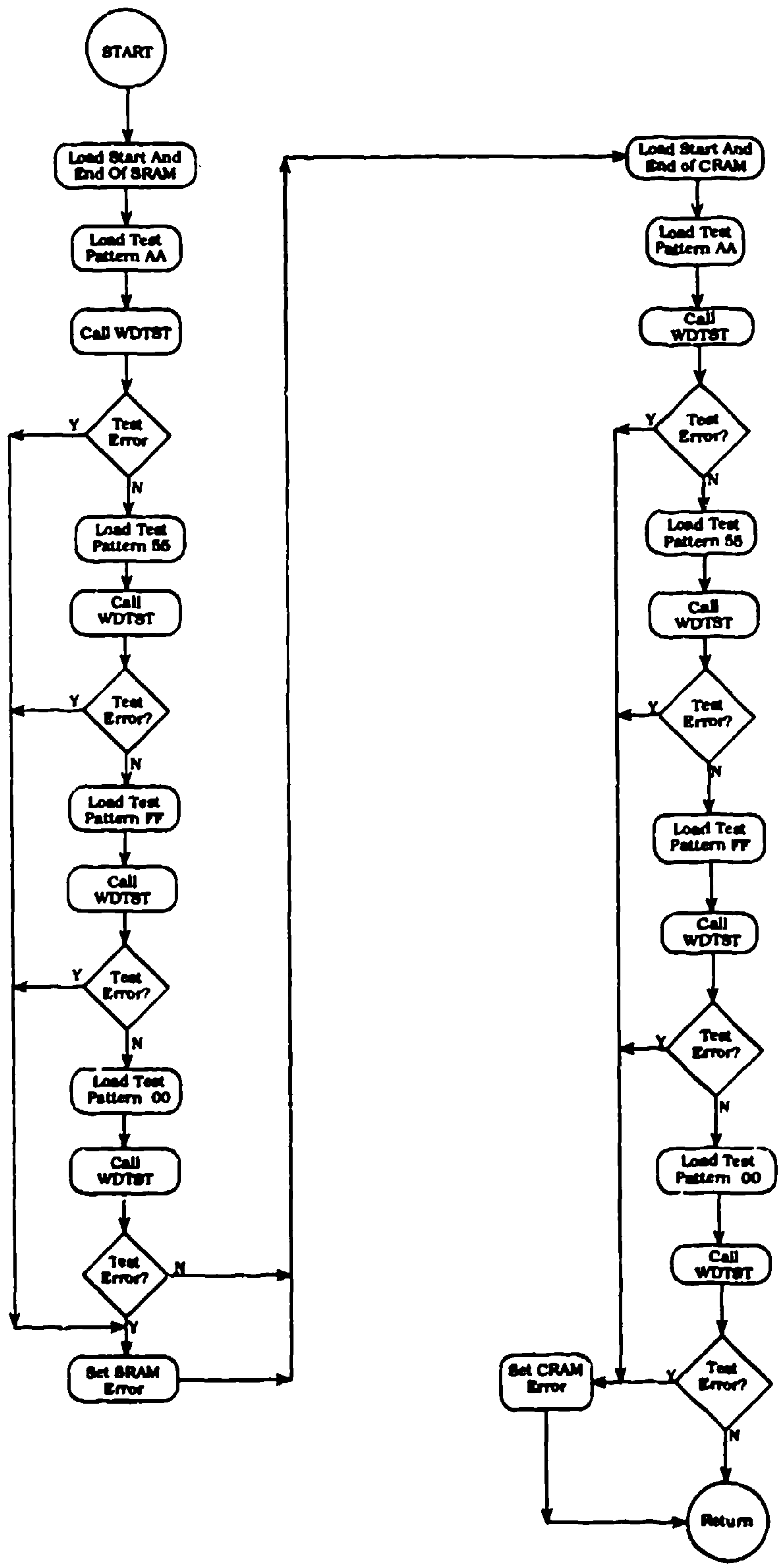



Figure 238. RAMTST Memory Diagnostic Flow Chart

# SWTDIAG

SWTDIAG performs the diagnostic for exercising the front panel switches. This tests the toggle switch, pushbutton switches, and the thumbwheel switches. The toggle and pushbutton switches light the LEDs when the state of the switch changes. The thumbwheel switches display their values on the front panel display. This test is free running until the halt switch is detected. The pushbuttons and toggle switch utilize the interrupts to detect state changes. BUTINT indicates this change through CONTFL, HLTFLG, and STRTFL.

| | |
|---|---|
| INPUT PARAMETERS: | CONTFL: Set by BUTINT when the CONTINUE button is pressed |
| | HLTFLG: Set by BUTINT when the HALT button is pressed |
| | STRTFL: Set by BUTINT when the START button is pressed |
| OUTPUT PARAMETERS: | None |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| A0 | Pointer to DSPBUF for displaying thumbwheel switches |
| D1 | LED status |
| D2 | Switch input state |
| D3 | Display character for thumbwheel switches |

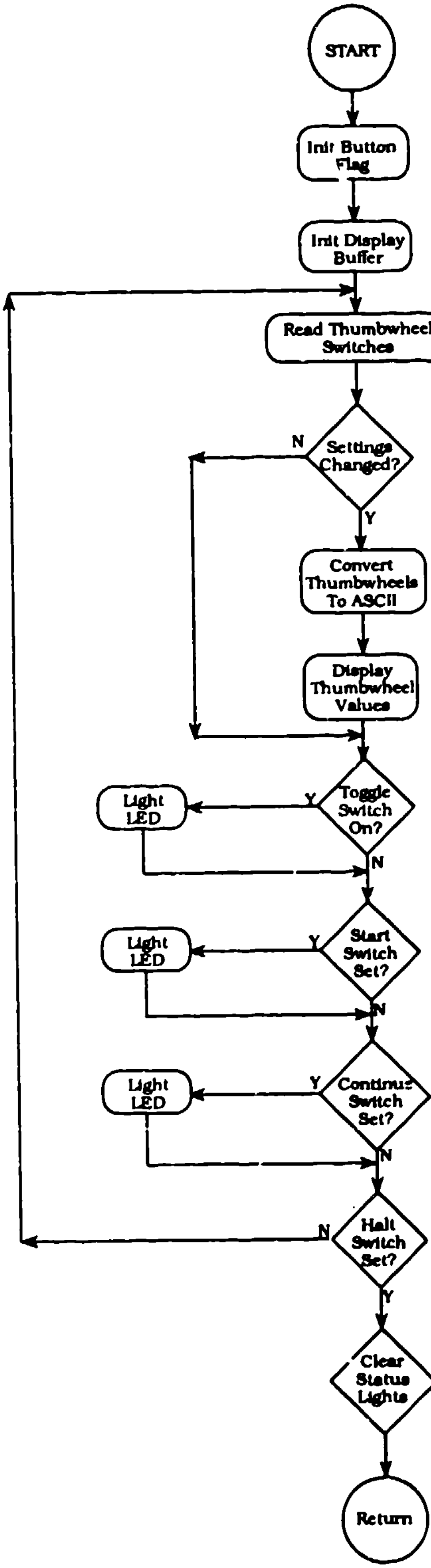



Figure 239. SWTDIAG Control Switches Diagnostic Flow Chart

## LITDIAG

LITDIAG performs a test of the front panel lights when the light diagnostic is manually selected. The test consists of continuously sequencing the lights on the front panel until the HALT button (HLTFLG) is detected.

| | |
|---|---|
| INPUT PARAMETERS: | HLTFLG: Set by BUTINT when the HALT button is pressed |
| OUTPUT PARAMETERS: | None |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| D0 | Delay counter |
| D1 | Output to the lights |
| D2 | Bit set value for the control byte |

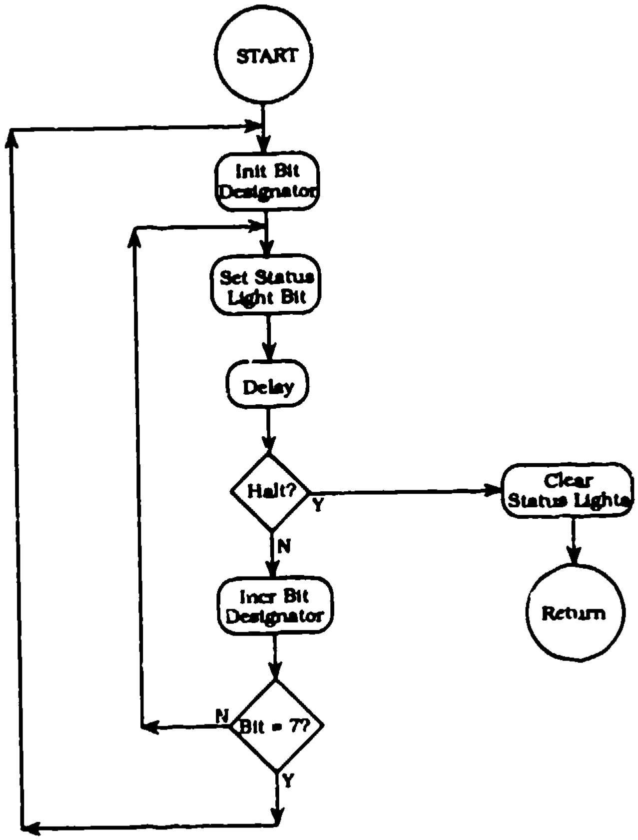



Figure 240. LITDIAG Selected Lights Diagnostic Flow Chart

## DLDATA

DLDATA transfers the test data from ADAM to the DRASS memory through the parallel port. The data are transferred in a predefined sequence: first the parameters, then precal data, then test data, and finally the postcal data. The data are transferred in blocks starting with a sync code, followed by data, and then a checksum. The size of the data blocks is determined by the number of A/D channels specified for the test. The test parameters are always transferred as one block. DLDATA calculates the checksum of the data as it receives it and if it matches the expected checksum, it responds to ADAM with FAF3FF00. If there is a checksum error, it will respond to ADAM with FAF3FFXX, where XX is any value other than 00. DLDATA will allow five attempts at receiving a data block before it errors. The sync code at the beginning of each data block indicates what kind of data it is as follows:

| | |
|---|---|
| FAF30000 | End of transmission |
| FAF31000 | Parameter block |
| FAF32000 | Precal data block |
| FAF33000 | Test data block |
| FAF34000 | Postcal data block |
| FAF3FF00 | Checksum match message |
| FAF3FFXX | Checksum value |

DLDATA saves the sync code of the first block of new type of data to determine the starting frame counter for those data.

| | |
|---|---|
| PRESYNC | Precal sync code |
| DATSYNC | Test data sync code |
| POSSYNC | Postcal sync code |

INPUT PARAMETERS: HLTFLG: Set when the halt button is pressed and will cause transfer to exit

OUTPUT PARAMETERS: None

REGISTERS: CONTENTS:

REGISTERS USED:

| | |
|---|---|
| A0 | Display message pointer |
| A1 | Start of data buffer |
| A2 | End of data buffer |
| D1 | Hand shaking status |
| D2 | Input data |
| D3 | Temporary data storage |

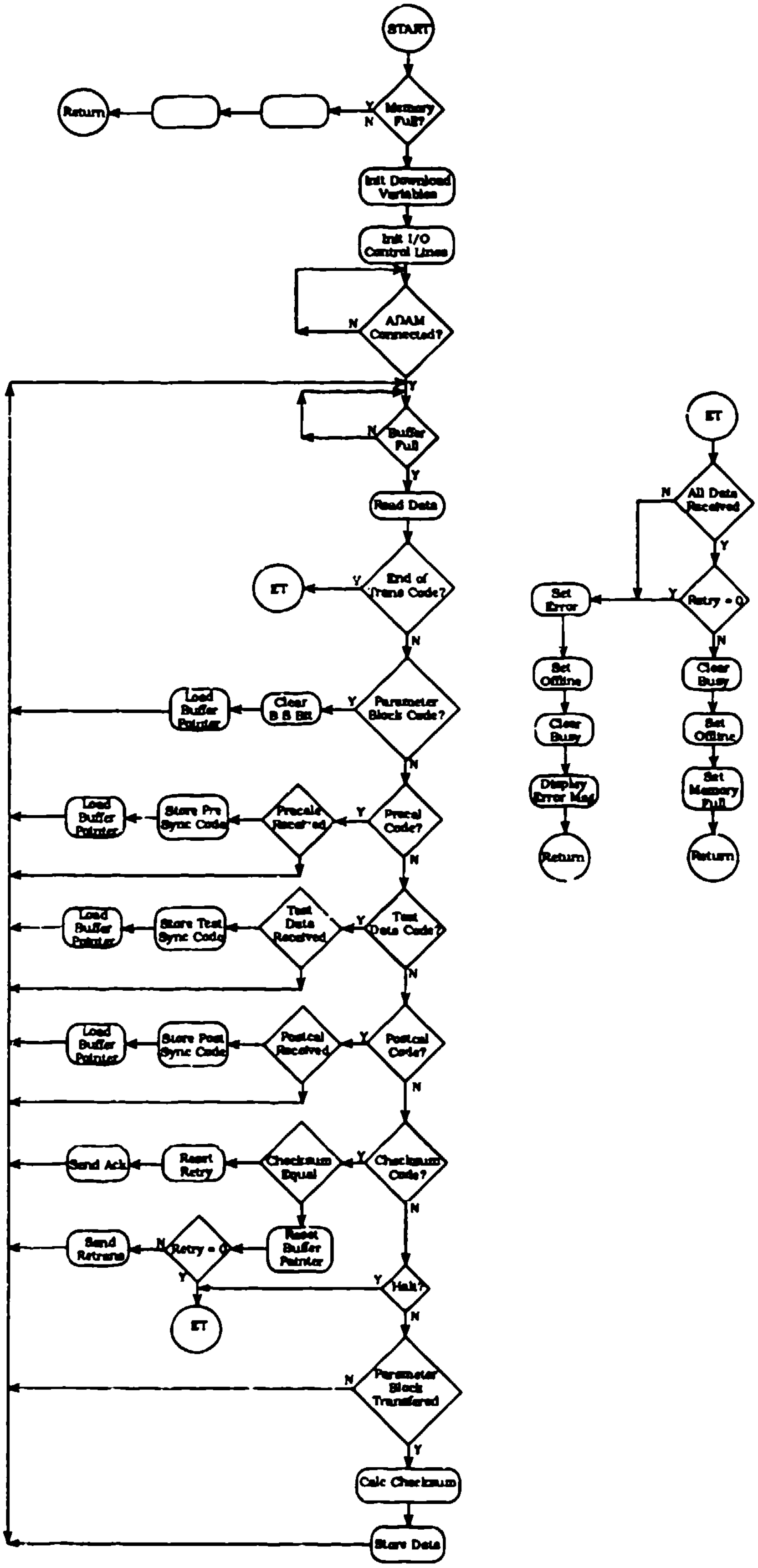



Figure 241. DLDATA ADAM to DRASS Data Transfer Flow Chart

## OUTDATA

OUTDATA outputs the test data contained in the DRASS RAM to the serial port. Before the transfer is initiated, it reads the configuration for the port from the thumbwheel switches. The selections are for baud rate, character size, number of stop bits, and parity. Then it reads the selection for data format: ASCII format or binary format. In binary format mode, the binary data are just output to the serial port as read from memory. In ASCII format mode, the data are converted to ASCII, spaces are inserted between each channel data, the frame counter is inserted in front of each data block, and the data are displayed in blocks for easier readability. Once this function is initiated, it will continue to output data until the end of data is reached or the HALT button is detected.

| | |
|---|---|
| INPUT PARAMETERS: | HLTFLG: Set when the HALT button is pressed<br>CONTFL: Set when the CONTINUE button is pressed<br>PREBUF: Precal data<br>DATBUF: Test data buffer<br>POSTBU: Postcal buffer |
| OUTPUT PARAMETERS: | None |
| REGISTERS: | CONTENTS: |
| REGISTERS USED: | |
| A0 | Message pointer |
| A1 | Buffer pointer to transmit data |
| D1 | Sync code |
| D2 | Block length |
| D7 | Temporary storage |

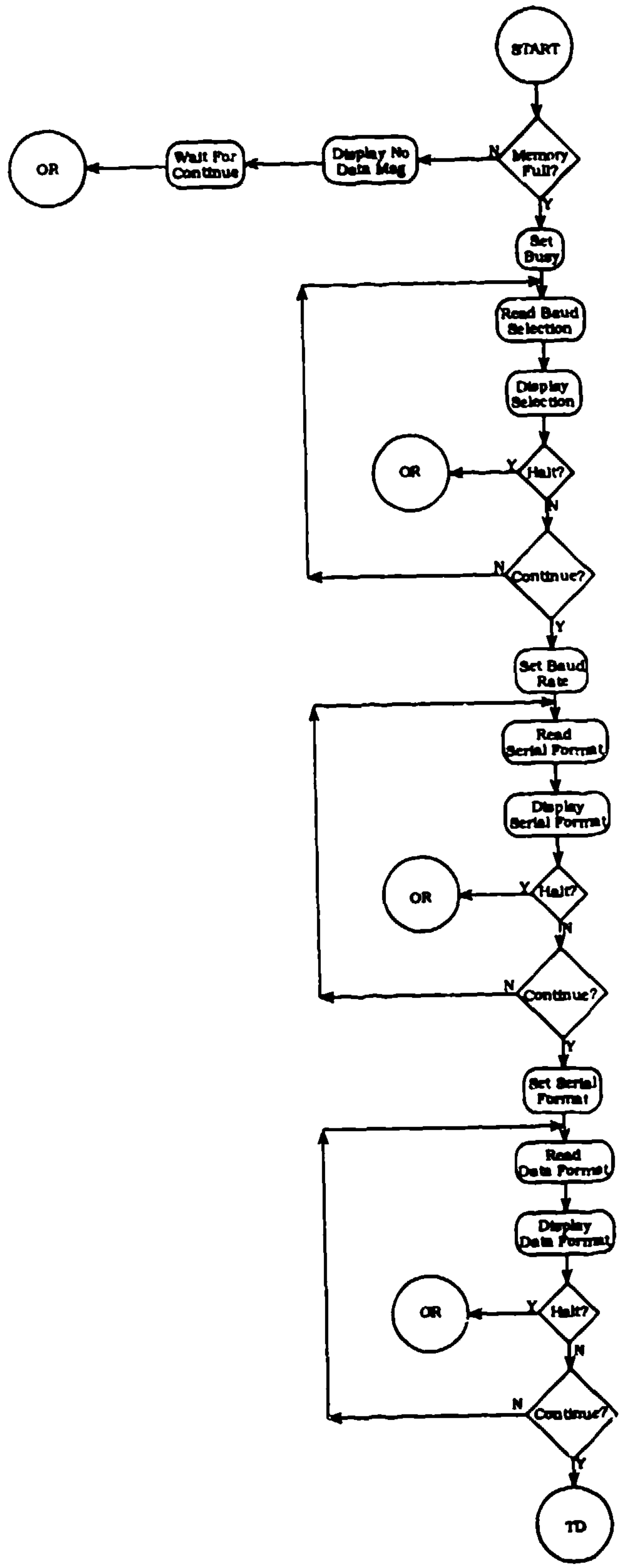



Figure 242. OUTDATA DRASS Output Data Flow Chart (1 of 3)

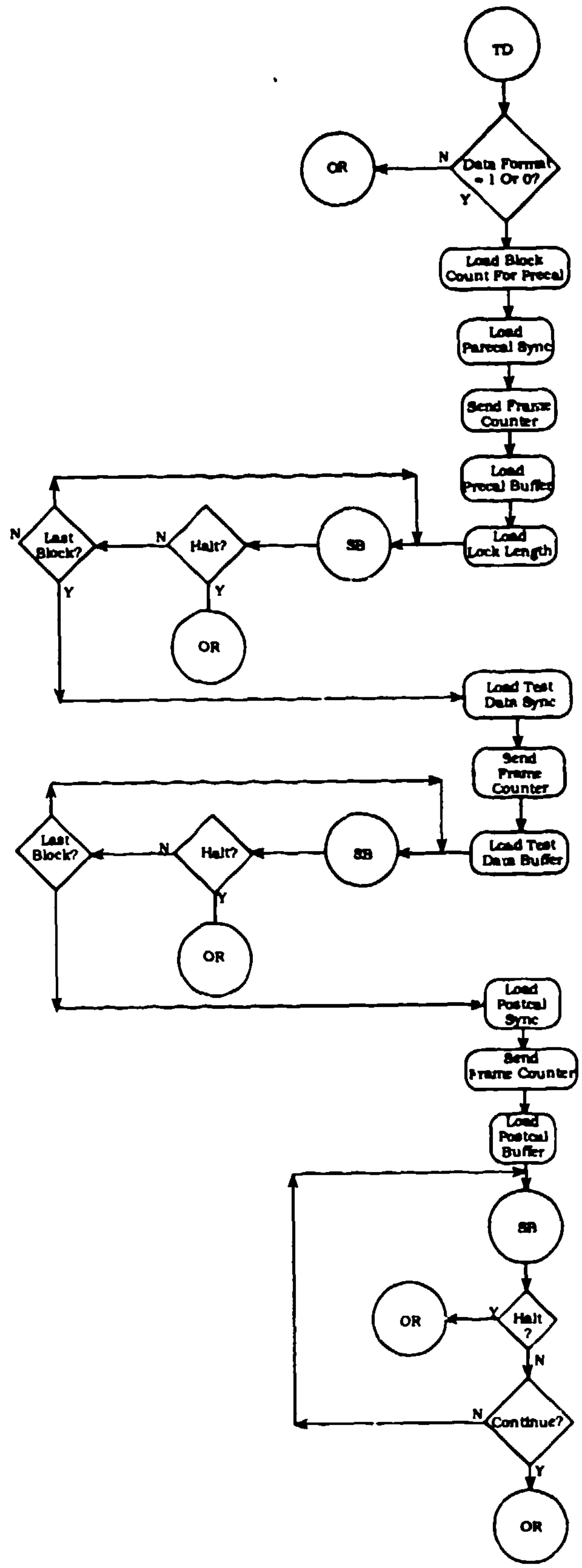



Figure 242. OUTDATA DRASS Output Data Flow Chart (continued) (2 of 3)

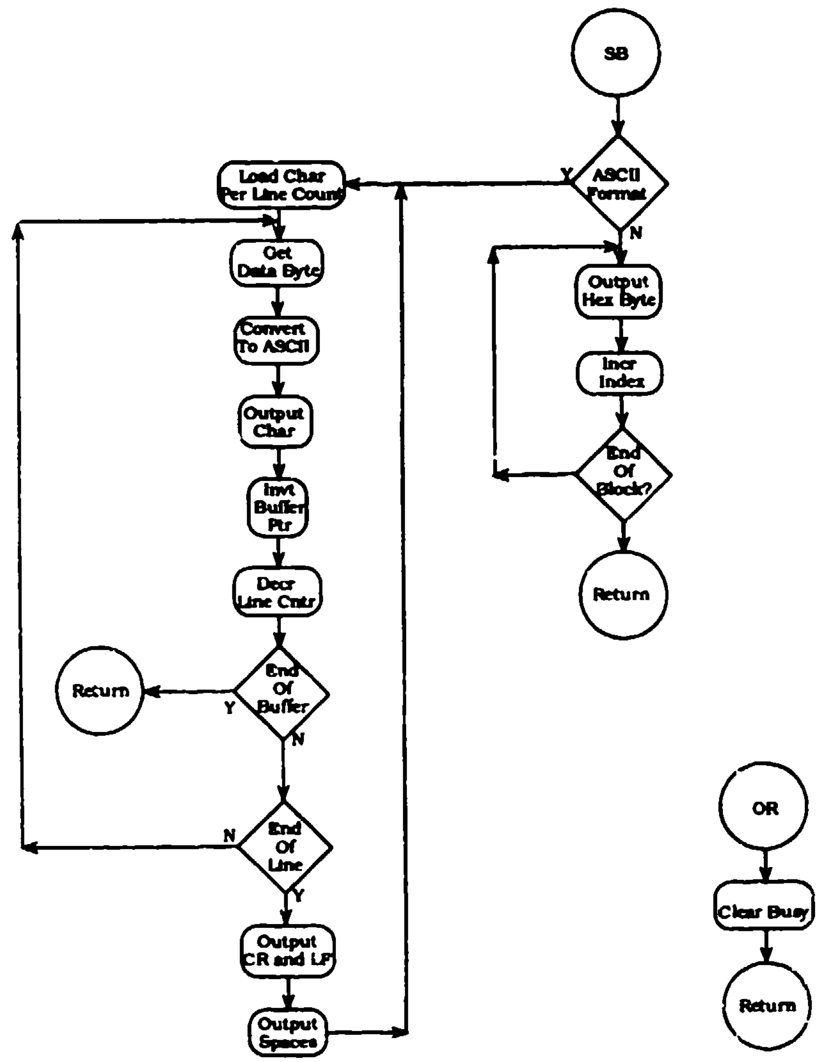



Figure 242. OUTDATA DRASS Output Data Flow Chart (continued) (3 of 3)

## SERDIAG

SERDIAG performs the serial diagnostic test on the UART in the MFP when selected. It executes an internal loop back test with a canned message. SERDIAG does not utilize interrupts but it does test the transmit and receive status words. The data format and baud rate used during the test are defaulted to 1200 baud, seven bit words, no parity, and two stop bits. SERDIAG will execute continuously until the HALT button is detected (HLTFLG).

| | |
|---|---|
| INPUT PARAMETERS: | HLTFLG: Set when the HALT button is pressed |
| OUTPUT PARAMETERS: | SERST: \$00: Data error<br>\$10: Frame error<br>\$20: Parity error<br>\$30: Overrun error<br>\$40: Transmit time out<br>\$50: Receive error |

| REGISTERS: | CONTENTS: |
|---|---|
| REGISTERS USED: | |
| A0 | Index to test pattern |
| D0 | Test data character |
| D1 | Temporary UART status |
| D7 | Character count |

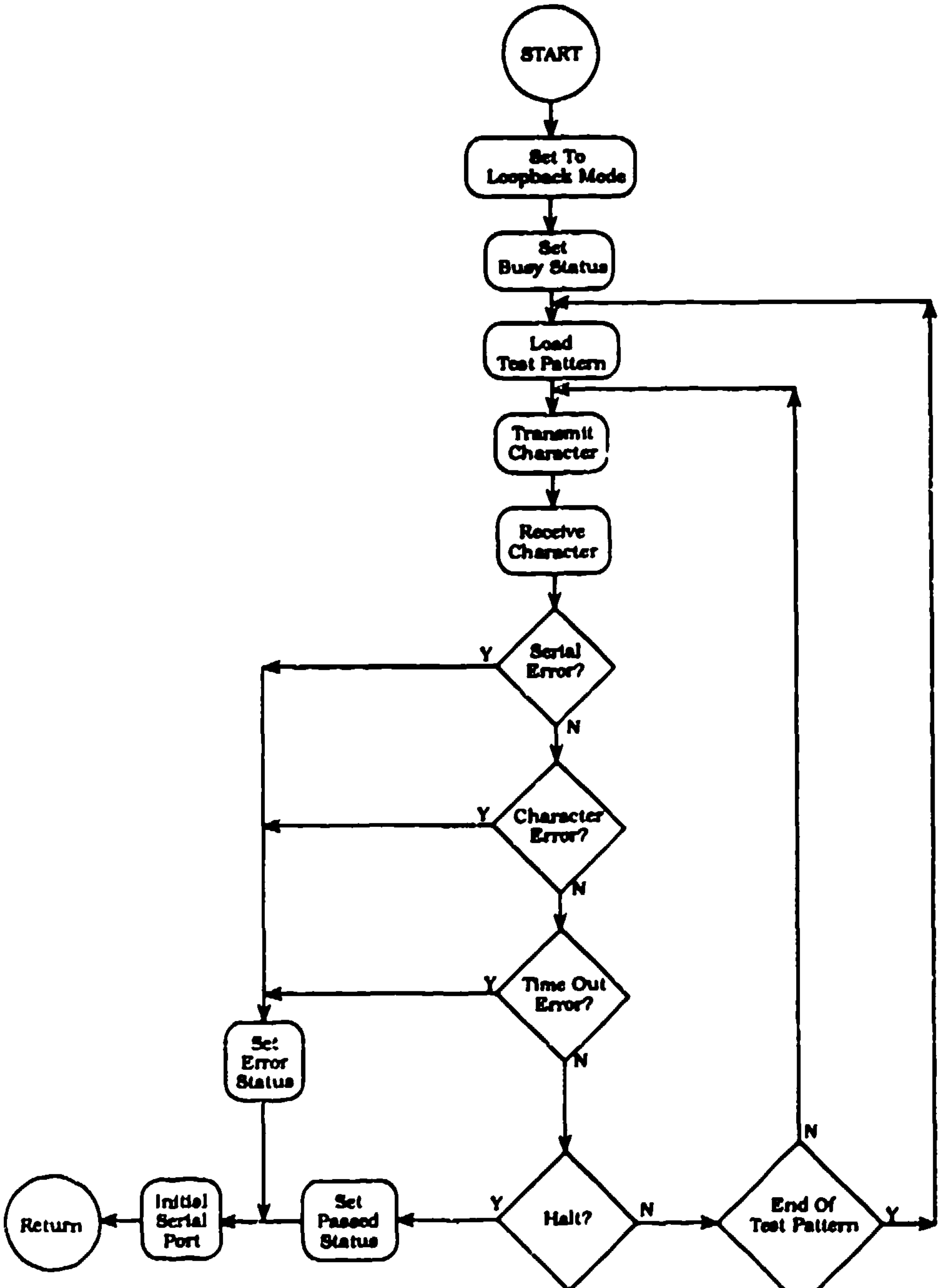



Figure 243. SERDIAG Selected Serial Port Diagnostics Flow Chart

# RAMDIAG

RAMDIAG tests both the SRAM and Cache RAM in the system when the memory diagnostics are selected. It performs a byte write and read of memory with the data patterns of AA, 55, FF, and 00. The memory that is tested is from 1000000 (STRAM) through 107FFFE (ENRAM), and 10100 (STCACH) through 10700 (ENCACH). Partial range testing prevents the destruction of system parameters and the system stack. RAMDIAG uses the routine WDTST to do the actual memory accesses. If a failure is detected, an error message is displayed.

INPUT PARAMETERS: D7: Test status from WDTST

OUTPUT PARAMETER: None

| REGISTERS: | CONTENTS: |
|---|---|
| **REGISTERS USED:** | |
| A0 | Start of memory to test |
| A1 | End of memory to test |
| D0 | Test data pattern |
| D7 | Test status from WDTST |

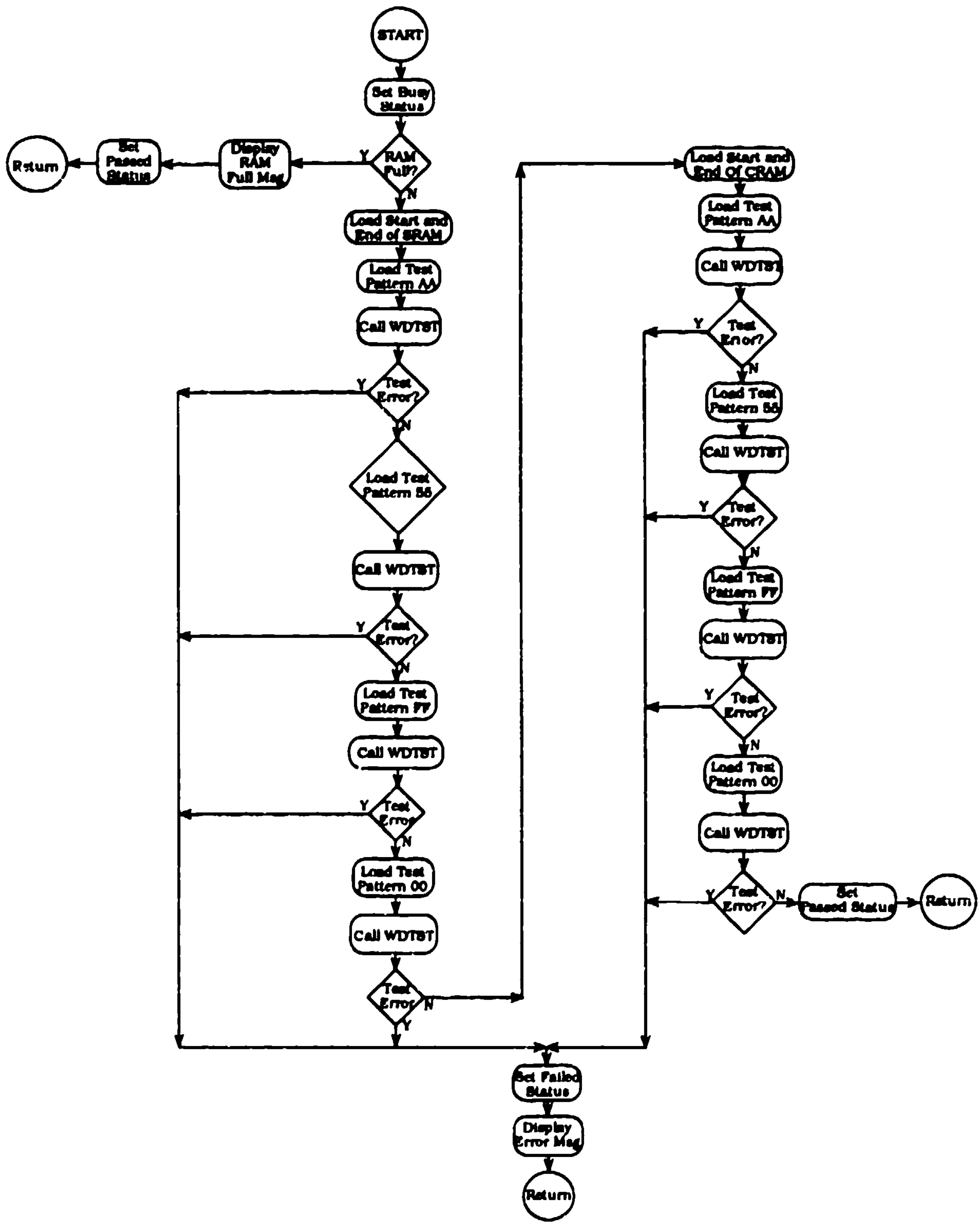



Figure 244. RAMDIAG Selected RAM Diagnostics Flow Chart

```
      *          ADAM
        OPT     P=68020
*
*
**********************************************************************
*
*    The ADAM program is divided into two main functional
*    sections : Data Acquistion and Menu Processing. Each
*    function is driven by an interrupt, Data Acquisition
*    (DAQINT) is the telemetry interrupt 6 and Menu Processing
*    (KEYINT) ic the keyboard interrupt 2. When an interrupt
*    occurs, the respective handler is executed. DAQINT is
*    serviced by IDLE,or DATCOL depending
*    on the stage of data acquisition. KEYINT is serviced by
*    MENUPR. While the system is in data acquisition mode,
*    the keyboard interrupts are disabled until the data
*    collection is complete and then re-enabled if the
*    terminal is connected.
*
*
*                        Module Hierarchy
*
*   ADAM (Initialization) : SERINIT     - Initialize serial port
*                           ROMTST      - Power up ROM test
*                           SERTST      - Power up serial test
*                           CLRSC       - Clear screen
*                           TMRTST      - Power up timer test
*                           ADTST       - Power up A/D test
*                           PARLTST     - Power up parallel test
*                           RAMTST      - Power up RAM test
*                           DSPMSG      - Display message
*
*   WAITLP (Process Server): DMPDAT     - Down load test data
*                            IDLEJMP    - Idle routine
*                            PRLDIAG    - Parallel diagnostic
*                            CLKTST     - Filter clock diagnostic
*                            TELMTST    - Telemetry port diagnostic
*                            DISPTST    - Display diagnostic
*                            ADDIAG     - A/D diagnostic
*                            ALIGN      - A/D alignment test
*                            DSPLMU     - Display last menu
*   DAQINT (Telemetry interrupt handler)
*
*                            IDLE       - Data collection but not storage
*                            DATCOL     - Test data storage
*
*   KEYINT (Keyboard interrupt handler)
*
*                            GETKEY     - Read keyboard entry
*                            MENUPR     - Process keyboard entry
```

```
*
*                         Register Assignment
*
*          A0 - General purpose
*          A1 - General purpose
*          A2 - Display pointer
*          A3 - DAQINT jump address
*          A4 - Scan table pointer
*          A5 - Data buffer pointer
*          A6 - Keyboard buffer pointer
*          A7 - Stack pointer
*
*          D0 - General purpose
*          D1 -    "        "
*          D2 - Display counter
*          D3 - General purpose
*          D4 -    "        "
*          D5 - A/D data
*          D6 - Key input buffer index
*          D7 - General purpose
*
*  Status Byte Definition
*  ----------------------
*  Diagnostic Status (DSTAT) - Bit set = error / zero = passed
*
*          Bit 0 - ROM error
*              1 - Serial error
*              2 - Filter clock error
*              3 - A/D error
*              4 - Parellel port error
*              5 - RAM error
*              6 - Not used
*              7 -  "   "
*
*  Test Status (TSTST)
*
*          Bit 0 - Pre cal. stage (when set)
*              1 - Data collection stage ( " )
*              2 - Post cal. stage (  "   )
*              3 - Memory full     (  "   )
*              4 - Terminal connected (when 0)
*              5 - RCAL mode (when set)
*              6 - Not used
*              7 - Start storing data (when set)
*
*
```

```
*
***************************************************************
*
*          EXTERNAL REFERENCES
*
*          DEFINE STORAGE PARAMETERS
*
ADAM       IDNT 1,1
           XDEF    ROMPRM,PRMEND,PRMSG1,PRMCNT1,PRMSG2,PRMCNT2
           XDEF    ROMER,SERER,FLCERR,PRLERR,ADER
           XDEF    RAMER,ERRCNT,MENU1,MEN1CT,PREBUF,POSTBU
           XDEF    STSCT,ENDSCT,UCNTRL,BAUD,PRSCA,PRSCB,PRSCCD
           XDEF    SYNC,FCNTR,TSTST,FLCNTA,FLCNTB,FLCNTC,FLCNTD
           XDEF    DSTAT,MU1SL,MU2SL,MU3SL,MU4SL,KEYBUF,PARCHK
           XDEF    KEY,CALCNT,SERST,ENDROM,START,STATM,CNTRLM
           XDEF    MU2SAD,MU2SCT,MU3SAD,MU3SCT,BDTBL,CLK2K
           XDEF    CLK4K,CLK8K,CLK10K,CLK16K,PRE2K,PRE4K,PRE8K
           XDEF    PRE10K,PRE16K,JMPADR,TXFLG,CLCTD,TELMTX
           XDEF    TELMWRD,KBFLG,DSPTST,DSPTPTR,ADCHD1,ADCHD2
           XDEF    INVAL,ADCH,ALNCH,ALNCHD1,ALNCHD2,TMPST
           XDEF    ENTMPST,TSERCTL,TMPBD,OTHCLK,PARNO,TMPPRE
           XDEF    TMPCLK,DSPBUF,CCBUF1,CCBUF2,TMPCHR,TMPCVT
           XDEF    MEMFAIL,JMPTBL,TSTAFL,STPCNT,PRESYNC,DAQSYNC
           XDEF    POSSYNC
*
           XDEF    MENU2,MEN2CT,MENU3,MEN3CT,MENU4,MEN4CT
           XDEF    MENU5,MEN5CT,MENU6,MEN6CT,MENU7,MEN7CT
           XDEF    MENU8,MEN8CT,MENU9,MEN9CT,ERRMSG,ERRMCT
           XDEF    ADPRMT,ADPRCT,CLKTSTM,CLKTCT,TELTSTM,TELTCT
           XDEF    PARTSTM,PARTCT,MEMMSG1,MEMCT1,MEMMSG2,MEMCT2
           XDEF    MEMMSG3,MEMCT3,MEMMSG4,MEMCT4,MEMMSG5,MEMCT5
           XDEF    KBPRMT,KBCT,FCLKMSG,FCLKCT,PARMSG,PARCT
           XDEF    STBMSG,STBCT,WRDLMSG,WRDLCT,BAUDMSG,BAUDCT
           XDEF    CLRENT,CLRECT,MEMPAS,MEMPCT,MEMERM,MEMERC
           XDEF    MEMERM1,MEMERC1,MEMERM2,MEMERC2,CCPASS
           XDEF    CCPCT,CCMSG,CCMCT,DSCTMSG,DSCTCT,ALNHDR1
           XDEF    ALNCT1,ALNHDR2,ALNCT2,ADHDR,ADCT1,ADHDR2
           XDEF    ADCT2,DSPTPAT,ENDSPT
           XDEF    DRACON,DRASCC,TDATMSG,TDATC,TXERM,TXERC
           XDEF    RCVERM,RCVERC,CONERM,CONERC,RDATER,RDATC
           XDEF    TDCOMP,TDCOMPC,CLCTDA,CLCTDC,PURPRMT,PURPRC
           XDEF    TDATER,TDATRC,MEMFLM,MEMFLC,NDATMSG,NDATMCT
           XDEF    DATFLM,DATFLC,PWRMU,PWRCT,SYSPAS,SYSPASC
           XDEF    VERNUM,VERCNT
*
*          SUBROUTINE DEFINITIONS
*
           XREF    ROMTST,SERTST,TMRTST,ADTST,DSPMSG,RAMTST
           XREF    CRLFO,GETKEY,TMRINIT,SERINIT,CLRSC,CVTASCI
           XREF    MENUPR,PARLTST,WDTST
*
*
```



BEST AVAILABLE COPY

```
         SECTION 0
SYSTP    DC.L    $107FFF0          System stack pointer
INITPC   DC.L    START             Starting address
         DC.L    BUSERR            Buss error vector
         DC.L    ADDERR            Address error vector
         DC.L    ERRVEC
         DC.L    ERRVEC
         DC.L    0,0
         DC.L    PRIVER            Privelege error vector
         DC.L    0,0,0,0
         DC.L    0,0,0
         DC.L    0,0,0,0
         DC.L    0,0,0,0
         DC.L    SPURINT           Spurious interrupt vector
         DC.L    AVEC              Auto vector 1
         DC.L    KEYINT            Keyboard interrupt 2
         DC.L    AVEC              Auto vector 3
         DC.L    AVEC              Auto vector 4
         DC.L    AVEC              Auto vector 6
         DC.L    DAQINT            Telemetry interrupt 6
         DC.L    AVEC              Auto vector 7
SWORD0   EQU     $2700             Disable interrupt mask
SWORD1   EQU     $2000             Enable interrupt mask
VBASE    EQU     0                 Vector base address
CEN      EQU     1                 Cache enable
ISTACK   EQU     $107FFF0          Interrupt stack pointer
PROM     EQU     0                 Start PROM address
STRAM    EQU     $1000000          Start of RAM
ENRAM    EQU     $107EFFE          End of RAM
DATBUF   EQU     $1000000          Start of test data buffer
ENDBUF   EQU     $107CFFF          End of test data buffer
ADC      EQU     $800000           A/D address
TELE     EQU     $800010           Telemetry port address
PPRT     EQU     $800020           Parallel port address
STAT     EQU     $800030           Status port
CNTRL    EQU     $800031           Control port
GPIP     EQU     $800040           GPIP address
DDR      EQU     $800042           Data direction address
IERA     EQU     $800043           Interrupt enable A
IERB     EQU     $800044           Interrupt enable B
IPRA     EQU     $800045           Interrupt pending A
IMRA     EQU     $800049           Interrupt mask A
TACR     EQU     $80004C           Timer A control
TBCR     EQU     $80004D           Timer B control
TCDCR    EQU     $80004E           Timer C&D control
TADR     EQU     $80004F           Timer A data
TBDR     EQU     $800050           Timer B data
TCDR     EQU     $800051           Timer C data
```

```
TDDR      EQU   $800052      Time- D data
UCR       EQU   $800054      UART control
RSR       EQU   $800055      Receive status
TSR       EQU   $800056      Transmit status
UDR       EQU   $800057      Serial data register
RCAL      EQU   06           RCAL bit
PWR       EQU   07           Power bit
TYCON     EQU   4            Terminal connected bit
STCOL     EQU   2            Start collection bit
DIAGPAS   EQU   0            Diagnostic status bit
CALMOD    EQU   1            Calibration mode bit
MEMFUL    EQU   2            Memory full bit
RUNMOD    EQU   3            Test running bit
SAVDAT    EQU   4            Saving data bit
ARMED     EQU   5            System armed bit
TELEMSK   EQU   7            Telemetry staus bit
LPBAK     EQU   $0007        UART loopback control
TXENA     EQU   $0025        Transmit enable control
TXDIS     EQU   $24          Transmit disable control
RSVEN     EQU   01           Receive enable control
ENT       EQU   $0D          ENT key
ESC       EQU   $1B          F4 key
DOT       EQU   $2E          Period key
HYFN      EQU   $2D          Hyfen key
SCRLUP    EQU   $84          Scroll up key
SCRLDN    EQU   $83          Scroll down key
DEL       EQU   $08          Delete key
CARET     EQU   $0D          Carrage return
LINFED    EQU   $0A          Line feed
RSTALL    EQU   $10100000    Reset timer control
CNTDT1    EQU   $05050505    2KHz counter
CNTDT2    EQU   $0A0A0A0A    4KHz counter
CNTDT3    EQU   $05050505    8KHz counter
CNTDT4    EQU   $01010101    10KHz counter
CNTDT5    EQU   $1F1F1F1F    16KHz counter
CLKPRE1   EQU   $07077700    2KHz prescale
CLKPRE2   EQU   $04044400    4KHz prescale
CLKPRE3   EQU   $04044400    8KHz prescale
CLKPRE4   EQU   $07077700    10KHz prescale
CLKPRE5   EQU   $01011100    16KHz prescale
*
```

```
*
         SECTION 1
*
*        INITIALIZE PROCESSOR PARAMETERS
*
START    MOVE.L #ISTACK,A7           Initialize stack
         MOVE.W #SWORD0,SR           Initialize status reg.
         MOVE.L #VBASE,A0            Initialize vector base reg.
         MOVEC  A0,VBR
         MOVE.L #ISTACK,A7
         MOVEC  A7,ISP               Initialize stack reg.
         MOVE.L #CEN,A0              Initialize cache enable
         MOVEC  A0,CACR
*
*        INITIALIZE SYSTEM PARAMETERS
*
         CLR.L  D6
         MOVE.W D6,KBFLG             Clear keyboard test flag
         MOVE.W D6,DSPTST              "   display   "    "
         MOVE.B D6,TSTST             Clear test status
         MOVE.B D6,DSTAT             Clear pwr up diagnostic status
         MOVE.W D6,MU1SL             Clear menu level values
         MOVE.W D6,MU2SL
         MOVE.W D6,MU3SL
         MOVE.W D6,MU4SL
         MOVE.W D6,JMPADR            Set jump index to IDLE
         MOVE.W D6,TELMFL            Clear telemetry diag. flag
         MOVE.W D6,TXFLG             Clear transmit data flag
         MOVE.W D6,CLCTD             Clear collect data flag
         LEA.L  DSPBUF,A6            Fill display buffer
         MOVE.L #$20202020,D6          with spaces
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         MOVE.L D6,(A6)+
         LEA.L  KEYBUF,A6            load keyboard input buf. pointer
         MOVE.B #$80,STATM           Set power on bit
         MOVE.B STATM,STAT
         MOVE.B #$41,CNTRLM          Set to Non-RCAL mode
         MOVE.B CNTRLM,CNTRL
         CLR.L  D6                   Clear keyboard input buffer
```

```
            MOVE.L  #0,(A6)+
            MOVE.L  #0,(A6)+
            MOVE.L  #0,(A6)+
            MOVE.L  #0,(A6)+
            MOVE.L  #0,(A6)+
            LEA.L   KEYBUF,A6
            BTST    #TYCON,GPIP          Is terminal connected?
            BNE.S   TERCON               Yes
            BSET    #4,TSTST             Set not connected bit
*
*           INIT PARAMETER TABLE
*
TERCON      CMPI.L  #$ABCD1234,PARCHK    See if RAM is valid
            BEQ.S   PAROK                Yes
            LEA.L   ROMPRM,A0            Transfer default parmaters
            LEA.L   UCNTRL,A1            From ROM to RAM
TRNPRM      MOVE.B  (A0)+,(A1)+
            CMPA.L  #PRMEND,A0
            BLE     TRNPRM
*
*           INIT SCAN TABLE
*
            LEA.L   STSCT,A0             Set index to start of scan table
            CLR.L   D0                   set first mux channel
INIST       MOVE.B  D0,(A0)+             Store mux channels
            ADDI.W  #1,D0                Incr. mux channel no.
            CMPI.W  #33,D0               Check for end
            BLT.S   INIST
            SUB.L   #1,A0
            MOVE.L  A0,ENDSCT            Save end of scan table
            MOVE.L  #$ABCD1234,PARCHK    Set valid data mask
*
*           START POWER UP DIAGNOSTICS
*
PAROK       BSR     SERINIT              Serial initialization
            BSR     ROMTST               ROM diag.
            BSR     SERTST               Serial diag.
            BSR     TMRTST               Timer diag.
            BSR     ADTST                A/D diag
            BSR     PARLTST              Parallel diag.
*
*           CHECK FOR VALID DATA IN MEMORY
*
            MOVE.L  DAQSYNC,D0           Check test data sync code
            ANDI.L  #$FFFFFF00,D0        Mask off frame counter
            CMPI.L  #$FAF32000,D0        Check code
            BNE     NODATA               No data branch
            BSET    #MEMFUL,STATM        Set memory full status
            CMPI.B  #0,DSTAT             Pwr. up diag. errors?
            BNE.S   PAROK1               Yes - dont set diag passed
```

```
         BSET    #DIAGPAS,STATM      Else set diag passed
PAROK1   MOVE.B  STATM,STAT          Output status
         MOVE.B  #00,IERA            Disable key int.
         BSET    #3,TSTST            Set memory full status
         BTST    #4,TSTST            Check for terminal connected
         BNE     DAQST               No
*
*        INIT SERIAL PORT & INTERUPT
*
         BSR     SERINIT             Initialize serial port
         BSR     CLRSC               Clear terminal screen
         MOVE.W  #SWORD1,SR          Enable interrupts
         LEA.L   PRMSG1,A2           Display data present prompt
         MOVE.W  PRMCNT1,D2
         BSR     DSPMSG
         MOVE.B  #0,IPRA             Clear keyboard interrupt pending
         MOVE.B  #0,KEY              Clear input key
YLOOP    CMPI.B  #ENT,KEY            Wait for ENTER key
         BNE     YLOOP
         BSR     CLRSC               Clear terminal screen
         CMPI.B  #$59,(A6)           Check for "Y"
         BEQ.S   NODATA              If "Y" clear memory with RAM diag.
         BSET    #3,TSTST            Else set memory full bit in status
         BRA.S   REPERR              Go report pwr. up diag. errors
*
*        CLEAR RAM WITH RAM DIAG.
*
NODATA   BSR     RAMTST              Perform RAM diag.
         MOVE.L  #0,PRESYNC          Clear Pre. Cal. sync
         MOVE.L  #0,DAQSYNC          Clear Test Data sync
         MOVE.L  #0,POSSYNC          Clear Post Cal. sync
         BCLR    #MEMFUL,STATM       Clear memory full bit
         CMPI.B  #0,DSTAT            Pwr up diag errors?
         BNE.S   NODATA1             Yes - continue
         BSET    #DIAGPAS,STATM      Else set diag passed status
NODATA1  MOVE.B  STATM,STAT          Output status
         BCLR    #3,TSTST            Set memory empty status
         BTST    #4,TSTST            Terminal connected?
         BNE     DAQST               No - go start data collection
*
*        REPORT DIAG ERRORS
*
REPERR   BSR     SERINIT             Initialize serial port
         BSR     CLRSC               Clear screen
         MOVE.W  #SWORD1,SR          Enable interrupts
         CMPI.B  #0,DSTAT            Check for pwr up diag errors
         BNE.S   DSPERR              Go display errors
         BSET    #DIAGPAS,STATM      Set diag passed status
         MOVE.B  STATM,STAT
         MOVE.B  #0,KEY              Clear input key
         LEA.L   SYSPAS,A2           Display System ready prompt
         MOVE.W  SYSPASC,D2
         BSR     DSPMSG
         BRA.S   NOERR1
```

```
DSPERR      MOVE.B #0,KEY              Clear input key
            LEA.L  PRMSG2,A2           Display pwr up diag err msg
            MOVE.W PRMCNT2,D2
            BSR    DSPMSG
            MOVE.W #4,D3               Init. err msg.per line
            MOVE.W ERRCNT,D2           Init. characters per msg
            LEA.L  ROMER,A2            Point to start of err msgs
            MOVE.W #6,D1               Set limit count
            MOVE.B DSTAT,D7            Get pwr up diag status
NXTERR      BTST   #0,D7               Test error bit
            BEQ.S  NOERR               Jmp if no error
            BSR    DSPMSG              Display error msg for that bit
            MOVE.W ERRCNT,D2           Reset char. cont per error
            SUBI.W #1,D3               Decr. msgs. per line
            BNE.S  NOERR               If < 4 continue
            BSR    CRLFO               Else output CR & LF
            MOVE.W #4,D3               Reset msg per line count
NOERR       ADDA.L #4,A2               Incr. error msg pointer
            LSR    #1,D7               Shift in next diag error bit
            SUBI.W #1,D1               If more bits to check
            BNE.S  NXTERR              Do it again
            LEA.L  VERNUM,A2           Display version number msg
            MOVE.W VERCNT,D2
            BSR    DSPMSG
NOERR1      CMPI.B #ESC,KEY            Wait for ESC key entry (F4)
            BNE    NOERR1
            CLR    D6                  Clear key input counter
*
*           DISPLAY MAIN MENU
*
MAINMU      LEA.L  MENU1,A2            Load main menu pointer
            MOVE.W MEN1CT,D2
            MOVE.W #1,MU1SL            Set level 1 menu index
            BSR    DSPMSG
*
*           GET READY TO COLLECT TEST DATA
*
DAQST       BTST   #3,TSTST            RAM full?
            BNE    WAITLP              Yes wait for dump data
            CMPI.B #0,DSTAT            Pwr up diag. errors?
            BNE    WAITLP              Yes Process menu selections only
            MOVE.L #2000000,STPCNT     Load 1 min test stop counter
            BTST   #4,TSTST            Is terminal connected
            BEQ    WAITLP              Yes-process menus
PRECOLC     BSET   #0,TSTST
            CLR.L  D0
            LEA.L  PREBUF,A5
            MOVE.B #01,FCNTR
            MOVE.L SYNC,PRESYNC
            BSET   #RCAL,CNTRLM
            MOVE.B CNTRLM,CNTRL
            LEA.L  STSCT,A4
            MOVE.B (A4),D0
            MOVE.L D0,ADC
            MULU   D5,D5
            MULU   D5,D5
            MOVE.L ADC,D5
```

```
         MOVE.W  #8,CALCNT
         BCLR    #7,TSTST
PRELOP   MOVE.L  ADC,D5
         CMPA.L  ENDSCT,A4
         BLE     PRELOP1
         LEA.L   STSCT,A4
         MOVE.B  (A4)+,D0
         MOVE.L  D0,ADC
         MULU    D5,D5
         MULU    D5,D5
         MOVE.L  ADC,D5
         BTST    #7,TSTST
         BNE.S   PRECON
         BSET    #7,TSTST
         BRA.S   PRELOP
PRECON   SUBI.W  #1,CALCNT
         CMPI.W  #5,CALCNT
         BNE.S   PRENDCH
         BCLR    #RCAL,CNTRLM
         MOVE.B  CNTRLM,CNTRL
         MOVE.W  #4000,D1
PREDEL   NOP
         MULU    D5,D5
         SUBI.W  #1,D1
         BNE.S   PREDEL
         BRA     PRELOP
PRENDCH  CMPI.W  #0,CALCNT
         BNE.S   PRELOP
         BSET    #RCAL,CNTRLM
         MOVE.B  CNTRLM,CNTRL
         BCLR    #0,TSTST
         LEA.L   DATBUF,A5
         BSET    #ARMED,STATM
         MOVE.B  STATM,STAT
         BCLR    #7,TSTST
         LEA.L   IDLE,A3
         MOVE.B  #0,FCNTR
         BRA.S   PREEND
PRELOP1  MOVE.B  (A4)+,D0
         MOVE.L  D0,ADC
         BTST    #7,TSTST
         BEQ.S   PRELOP2
         MOVE.L  D5,(A5)+
PRELOP2  MULU    D5,D5
         MULU    D5,D5
         MOVE.L  ADC,D5
         BRA     PRELOP
PREEND   BCLR    #TELEMSK,STATM        Enable telemetry interrupts
         BCLR    #7,TSTST
         MOVE.B  STATM,STAT
         MOVE.W  #SWORD1,SR            Enable interrupts
*
```

```
*         WAIT FOR START DATA COLLECTION COMMAND
*
WAITLP    BTST    #4,TSTST              Is terminal connected?
          BNE.S   WAITLP1               NO - skip menu processing
          CMPI.W  #0,TXFLG              Is transmit data flag set?
          BNE     DMPDAT                Yes - dump data to DRASS
          CMPI.W  #0,CLCTD              Is collect data flag set?
          BNE     STRTDAQ               Yes - start data acquisition
          CLR.L   D0
          MOVE.W  JMPADR,D0             Fetch JUMP table index
          LSL.L   #2,D0                 Multiply by 4
          LEA.L   JMPTBL,A0             Get start of jump table
          MOVE.L  (A0,D0),A1            Fetch JUMP address
          JMP     (A1)                  GO THERE!
WAITLP1   BTST    #3,TSTST              Memory full?
          BNE     DMPDAT                Yes - go dump data
          CMPI.B  #0,DSTAT              Pwr. up diag. errors?
          BNE     WAITLP                Yes - skip data processing
          BTST    #4,TSTST              Terminal connected?
          BEQ.S   WAITLP                Yes - go process menus
          BTST    #2,CNTRL              START bit present?
          BEQ     WAITLP                No - go wait
WAITLP2   BTST    #0,TSTST              Wait for Pre.Cal to finish
          BNE     WAITLP2
          BSET    #RUNMOD,STATM         Then put in run mode
          BSET    #SAVDAT,STATM         Set saving data
          MOVE.B  STATM,STAT
          BSET    #1,TSTST              Set test data collection status
          LEA.L   DATCOL,A3             Set test data collection jump
*                                       address
*         WAIT FOR BUFFUER FULL
*
WTLP2     BTST    #3,TSTST              Check for memory full
          BEQ     WTLP2                 If not full - wait
          BCLR    #SAVDAT,STATM         Clear saving data
          BSET    #MEMFUL,STATM         Set memory full
          BSET    #TELEMSK,STATM        Turn off telemetry
          MOVE.B  STATM,STAT
POSCOLC   LEA.L   POSTBU,A5
          CLR.L   D0
          MOVE.B  #01,FCNTR
          MOVE.L  SYNC,POSSYNC
          BSET    #RCAL,CNTRLM
          MOVE.B  CNTRLM,CNTRL
          LEA.L   STSCT,A4
          MOVE.B  (A4),D0
          MOVE.L  D0,ADC
          MULU    D5,D5
          MULU    D5,D5
          MOVE.L  ADC,D5
          MOVE.W  #8,CALCNT
          BCLR    #7,TSTST
```

```
POSLOP   MOVE.L  ADC,D5
         CMPA.L  ENDSCT,A4
         BLE     POSLOP1
         LEA.L   STSCT,A4
         MOVE.B  (A4)+,D0
         MOVE.L  D0,ADC
         MULU    D5,D5
         MULU    D5,D5
         MOVE.L  ADC,D5
         BTST    #7,TSTST
         BNE.S   POSCON
         BSET    #7,TSTST
         BRA.S   POSLOP
POSCON   SUBI.W  #1,CALCNT
         CMPI.W  #5,CALCNT
         BNE.S   POSENCH
         BCLR    #RCAL,CNTRLM
         MOVE.B  CNTRLM,CNTRL
         MOVE.W  #4000,D1
POSDEL   NOP
         MULU    D5,D5
         SUBI.W  #1,D1
         BNE.S   POSDEL
         BRA.S   POSLOP
POSENCH  CMPI.W  #0,CALCNT
         BNE.S   POSLOP
         BSET    #RCAL,CNTRLM
         MOVE.B  CNTRLM,CNTRL
         BCLR    #7,TSTST
         BCLR    #2,TSTST
         BRA.S   POSEND
POSLOP1  MOVE.B  (A4)+,D0
         MOVE.L  D0,ADC
         BTST    #7,TSTST
         BEQ.S   POSLOP2
         MOVE.L  D5,(A5)+
POSLOP2  MULU    D5,D5
         MOVE.L  ADC,D5
         BRA     POSLOP
POSEND   BCLR    #7,TSTST
         BTST    #TYCON,GPIP       Is terminal connected
         BEQ     DMPDAT            No - go wait to dump data
         MOVE.B  #$10,IERA         Enable key int.
         MOVE.B  #0,IPRA           Clear int. pending status
         BCLR    #4,TSTST          Set terminal present status
         MOVE.W  #0,CLCTD          Clear collecting data flag
         BRA     MAINMU            Go process menus
*
```

```
*          START OF DATA ACQUISITION WHEN ACTIVATED
*          FROM THE TERMINAL
*
STRTDAQ    BTST   #3,TSTST               Is memory full?
           BEQ.S  STRTDQ1                No - go start
           LEA.L  MEMFLM,A2              Else display memory full msg
           MOVE.W MEMFLC,D2
           BSR    DSPMSG
           MOVE.W #0,CLCTD               Clear data collection flag
           BRA    WAITLP                 Go wait some more
STRTDQ1    CMPI.B #0,DSTAT               Were ther pwr up diag errors
           BEQ.S  CONSDAQ                No - continue daq
           LEA.L  PUERM,A2               Else display pwr up error prompt
           MOVE.W PUERC,D2
           BSR    DSPMSG
           MOVE.W #0,CLCTD               Clear data collection flag
           BRA    WAITLP                 Go process menues
CONSDAQ    BSET   #4,TSTST               Set terminal not connected
           BCLR   #7,CNTRLM              Power system on
           MOVE.B CNTRLM,CNTRL
           MOVE.B #$00,IERA              Mask off MFP interrupts
           LEA.L  CLCTDA,A2              Display collecting data prompt
           MOVE.W CLCTDC,D2
           BSR    DSPMSG
           BRA    DAQST                  Start collecting data
*
*          POWER DOWN SYS. AND WAIT TO DUMP DATA
*
**********************************************************************
*
*       DMPDAT will transmit the test data (including test parameters,
*       pre. cal data and post cal. data) to the DRASS over the parallel
*       port. The test parameter data block consists of the Test channels,
*       length of a data block, control value for the serial port, Baud
*       rate, prescale for clocks A,B,C,and D and filter clock counters
*       for clocks A,B,C, and D. The order of transmission is: test
*       parameter data, precal data, test data, and finally post cal
*       data. To distinguish the different types of data, a unique
*       SYNC code is transmitted with each block:
*                 FAF31000  -  test parameters
*                 FAF320xx  -  precal data
*                 FAF330xx  -  test data
*                 FAF340xx  -  postcal data
*                   xx - is the frame counter determined at the time
*                        of data collection
```

```
*
*       The length of a data block ( except the test parameters ) is
*       determined by the number of channels of data collected in one
*       scan. After each block transfer, a checksum is transmited to
*       the DRASS for verification. The format of the checksum is :
*       FAF3FFxx where xx is the checksum for that data block. If
*       the DRASS verifies the checksum, it will respond with FAF3FF00
*       and the next block will be sent. If a checksum error is
*       detected, it will respond with FAF3FFxx and ADAM will retransmit
*       the data block. Up to five retries will be attempted by ADAM
*       before it errors out.
*
*       Data transmission can be initiated in one of two ways; either
*       through a menu selection on the ADAM terminal or by default
*       after a test is completed. The default method will only be
*       performed if the hand held terminal is not connected to ADAM.
*
***********************************************************************
*
DMPDAT    CMPI.W #0,TXFLG          Menu selected dump?
          BEQ.S  DMPCON            No - go dump data
          BTST   #3,TSTST          Else is memory full?
          BNE.S  DMPCON            Yes - go dump data
          LEA.L  NDATMSG,A2        Else display no data present msg
          MOVE.W NDATMCT,D2
          BSR    DSPMSG
          MOVE.W #0,TXFLG          Clear dump data flag
          BRA    WAITLP            Go process menues
DMPCON    BCLR   #SAVDAT,STATM     Clear saving data status
          BSET   #MEMFUL,STATM     Set memory full status
          BSET   #TELEMSK,STATM    Mask telemetry port
          MOVE.B STATM,STAT
          BTST   #TYCON,GPIP       Is terminal connected?
          BNE.S  DMPDAT1           Yes - then leave power on
          BSET   #PWR,CNTRLM       Else power down system
          MOVE.B CNTRLM,CNTRL
DMPDAT1   MOVE.L PPRT,D0           Initialize parallel port
          BSET   #0,CNTRLM         Set to input mode
          MOVE.B CNTRLM,CNTRL
          BTST   #TYCON,GPIP       Is terminal connected?
          BEQ.S  DRASCON           No - skip display
          LEA.L  DRACON,A2         Display DRASS not connected
          MOVE.W DRASCC,D2
          BSR    DSPMSG
DRASCON   MOVE.B CNTRL,D1          Wait for DRASS connect bit
          BTST   #3,D1
          BNE.S  DRASCON
          BTST   #TYCON,GPIP       Is terminal connected?
          BEQ.S  DRASCN1           No - skip display
          LEA.L  TDATMSG,A2        Display transmitting data msg
          MOVE.W TDATC,D2
          BSR    DSPMSG
```

```
DRASCN1  MOVE.W  #5,RETRY          Set error retry to five
         LEA.L   STSCT,A0          Determine the size of a data
         MOVE.L  ENDSCT,D7         block by the size of the scan
         SUBI.L  #1,D7             table
         SUB.L   A0,D7
         LSL.L   #2,D7
         ADD.L   #4,D7
         MOVE.L  D7,BLKLEN         Save the data block size
         CLR.L   D0                Initialize checksum to zero
TXDRAS0  MOVE.L  #$FAF31000,D7     Send parameter block code
         BSR     TXCMD
         CMPI.W  #0,D5             Check for time out
         BEQ     TXERR             Yes - display error
         CLR.L   D0                Init. checksum
         LEA.L   STSCT,A0          Load start of parameter block
         LEA.L   FLCNTD,A1         Load end of parameter block
         BSR     TXDATA            Transmit parameter block
         CMPI.W  #0,D5             Time out?
         BEQ     TXERR             Yes - display error
         MOVE.L  #$FAF3FF00,D7     Get checksum code
         OR.L    D0,D7             Insert checksum
         BSR     TXCMD             Send checksum
         CMPI.W  #0,D5             Time out?
         BEQ     TXERR             Yes - display error
         BSR     RCVCMD            Read checksum response
         CMPI.W  #0,D5             Time out?
         BEQ     INPERR            Yes - display error
         CMPI.L  #$FAF3FF00,D7     Checksum error?
         BEQ.S   TXPRE             No - go transmit pre cal data
         MOVE.W  RETRY,D7          decrement retry counter
         SUBI.W  #1,D7
         BEQ     TIMOUT            If 0 - display error
         MOVE.W  D7,RETRY          Else go retransmit parameter
         BRA     TXDRAS0           block
TXPRE    MOVE.W  #8,D1             Load scan count for precal
         MOVE.W  #5,RETRY          Load retry counter
         MOVE.L  PRESYNC,D7        Fetch precal sync code
         ANDI.W  #$00FF,D7         Mask in precal buffer code
         ORI.W   #$2000,D7
         BSR     TXCMD             Transmit precal code
         CMPI.W  #0,D5             Time out error?
         BEQ     TXERR             Yes - display error
         CLR.L   D0                Zero checksum
         LEA.L   PREBUF,A0         Load precal buffer pointer
         MOVE.L  A0,A1
         ADDA.L  BLKLEN,A1         Calculate end of block
```

```
TXPRE2     BSR     TXDATA            Transmit precal data block
           CMPI.W  #0,D5             Timeout error?
           BEQ     TXERR             Yes - display error
           MOVE.L  #$FAF3FF00,D7     Get checksum code
           OR.L    D0,D7             Insert checksum
           BSR     TXCMD             Transmit checksum
           CMPI.W  #0,D5             Timeout?
           BEQ     TXERR             Yes - display error
           BSR     RCVCMD            read checksum response
           CMPI.W  #0,D5             Timeout error
           BEQ     INPERR            Yes - display error
           CMPI.L  #$FAF3FF00,D7     Checksum error?
           BEQ.S   TXPRE3            No - go transmit next block
           MOVE.W  RETRY,D7          Else decr. retry counter
           SUBI.W  #1,D7
           BEQ     TIMOUT            If retry = 0 display error
           MOVE.W  D7,RETRY
           MOVE.L  A1,A0             Else reset buffer pointer
           SUBA.L  BLKLEN,A0
           CLR.L   D0                Zero checksum
           BRA.S   TXPRE2            Go retransmit
TXPRE3     SUBI.W  #1,D1             Decr. precal scan counter
           BEQ.S   TXDBUF            If 0 - go transmit test data
           MOVE.W  #5,RETRY          Else transmit next
           MOVE.L  A1,A0             Precal data block
           ADDA.L  BLKLEN,A1
           CLR.L   D0
           BRA     TXPRE2
*
TXDBUF     MOVE.W  #5,RETRY          Set retry counter
           MOVE.L  DAQSYNC,D7        Fetch test data sync code
           ANDI.W  #$00FF,D7         Mask in test data code
           ORI.W   #$3000,D7
           BSR     TXCMD             Send test data code
           CMPI.W  #0,D5             Timeout?
           BEQ     TXERR             Yes - display error
           CLR.L   D0                Zero checksum
           LEA.L   DATBUF,A0         load data buffer pointer
           MOVE.L  A0,A1             Calculate end of data block
           ADD.L   BLKLEN,A1
TXDBUF1    BSR     TXDATA            Send test data block
           CMPI.W  #0,D5             Timeout?
           BEQ     TXERR             Yes - display error
           MOVE.L  #$FAF3FF00,D7     Get checksum msg
           OR.L    D0,D7             Insert checksum
           BSR     TXCMD             Send checksum
           CMPI.W  #0,D5             Timeout?
           BEQ     TXERR             Yes - display error
           BSR     RCVCMD            Read checksum response
           CMPI.W  #0,D5             Timeout?
           BEQ     INPERR            yes - display error
```

```
          CMPI.L  #$FAF3FF00,D7      Checksum error
          BEQ.S   TXDBUF2            No - transmit next data block
          MOVE.W  RETRY,D7           Decr. retry counter
          SUBI.W  #1,D7
          BEQ     TIMOUT             If 0 - display error
          MOVE.W  D7,RETRY
          MOVE.L  A1,A0              Else reset block pointers
          SUBA.L  BLKLEN,A0
          CLR.L   D0                 clear checksum
          BRA.S   TXDBUF1            Go transmit block again
TXDBUF2   MOVE.W  #5,RETRY           Reset retry counter
          MOVE.L  A1,A0              Set pointers to next data block
          ADDA.L  BLKLEN,A1
          CMPA.L  #ENDBUF,A1         Is end of data reached
          BGT.S   TXPOST             Yes - go transmit post cal data
          CLR.L   D0                 Else transmit next test data
          BRA     TXDBUF1            block

TXPOST    CMP.W   #$FAF3,POSSYNC     Was post cal data collected
          BEQ.S   TXPCONT            If yes - go transmit it
          MOVE.L  #$FAF34000,D7      Else just transmit postcal code
          BSR     TXCMD
          BRA     FINDMP             and end transmission
TXPCONT   MOVE.W  #5,RETRY           Init. retry counter
          MOVE.L  POSSYNC,D7         Get post cal sync
          ANDI.W  #$00FF,D7          Mask in post cal code
          ORI.W   #$4000,D7
          BSR     TXCMD              Transmit post cal code
          CMPI.W  #0,D5              Timeout?
          BEQ     TXERR              Yes - display error
          CLR.L   D0                 Clear checksum
          LEA.L   POSTBU,A0          Load postcal buffer pointer
          MOVE.L  A0,A1
          ADDA.L  BLKLEN,A1          Calculate end of block
          MOVE.W  #8,D1              Set postcal scan counter
TXPOST1   BSR     TXDATA             Transmit postcal block
          CMPI.W  #0,D5              Timeout?
          BEQ     TXERR              Display error
          MOVE.L  #$FAF3FF00,D7      Get checksum command
          OR.L    D0,D7              Insert checksum
          BSR     TXCMD              Send checksum
          CMPI.W  #0,D5              Timeout?
          BEQ     TXERR              Yes - display error
          BSR     RCVCMD             Read checksum response
          CMPI.W  #0,D5              Timeout?
          BEQ     INPERR             Yes - display error
          CMPI.L  #$FAF3FF00,D7      Checksum error?
          BEQ.S   TXPOST2            No - tgransmit next block
          MOVE.W  RETRY,D7           Else decr. retry counter
          SUBI.W  #1,D7
```

```
         BEQ      TIMOUT              Retry = 0 display error
         MOVE.W   D7,RETRY
         MOVE.L   A1,A0               Reset block pointer
         SUBA.L   BLKLEN,A0
         CLR.L    D0                  Clear checksum
         BRA.S    TXPOST1             Retransmit postcal block
TXPOST2  SUBI.W   #1,D1               Decr. scan counter
         BEQ.S    FINDMP              If 0 - go end transmission
         MOVE.W   #5,RETRY            Reset retry counter
         MOVE.L   A1,A0               Load block pointers
         ADDA.L   BLKLEN,A1
         CLR.L    D0                  Clear checksum
         BRA      TXPOST1             Transmit next block
*
FINDMP   MOVE.L   #$FAF30000,D7       Send end of transmission command
         BSR      TXCMD
         BSET     #0,CNTRLM           Set parallel to input
         MOVE.B   CNTRLM,CNTRL
         BTST     #TYCON,GPIP         Is terminal connected?
         BEQ.S    FINDMP1             No - skip display
         LEA.L    TDCOMP,A2           Display transmission complete
         MOVE.W   TDCOMPC,D2          message
         BSR      DSPMSG
FINDMP1  MOVE.W   #0,TXFLG            Clear transmit data flag
         BCLR     #4,TSTST
         BTST     #TYCON,GPIP         Terminal connected?
         BNE      MAINMU              Yes - go process menues
         BSET     #4,TSTST            Else set terminal not connected
         BSET     #7,CNTRLM           Else power down
         MOVE.B   CNTRLM,CNTRL
DMPLOP   NOP                          Wait forever
         BRA.S    DMPLOP
*
*
TXERR    BTST     #TYCON,GPIP         Is terminal connected?
         BEQ      WTLP2               No - go to wait
         LEA.L    TXERM,A2            Display transmit data error
         MOVE.W   TXERC,D2            due to a checksum error
         BSR      DSPMSG
         BCLR     #4,TSTST
         MOVE.W   #0,TXFLG            go process menues
         BRA      MAINMU
*
INPERR   BTST     #TYCON,GPIP         Is terminal connected
         BEQ      WTLP2               No - go wait
         LEA.L    RCVERM,A2           Display input error
         MOVE.W   RCVERC,D2           due to a receive timeout
         BSR      DSPMSG
         BCLR     #4,TSTST
         MOVE.W   #0,TXFLG            go process menues
         BRA      MAINMU
*
```

```
TIMOUT   BTST    #TYCON,GPIP        Is terminal connected?
         BEQ     WTLP2              No - go wait
         LEA.L   TDATER,A2          Display transmit timeout error
         MOVE.W  TDATRC,D2
         BSR     DSPMSG
         MOVE.W  #0,TXFLG           Go process menues
         BCLR    #4,TSTST
         BRA     MAINMU
*
RCVCMD   MOVE.W  #1000,D5           Init. timeout counter
RCVCMD1  BTST    #7,CNTRL           DRASS busy?
         BEQ.S   RCVCMD2            No - go read
         SUBI.W  #1,D5              Else decr. timeout
         BNE.S   RCVCMD1            If not zero try again
         BTST    #TYCON,GPIP        Is terminal connected?
         BEQ.S   RCVCMD1            No - continue check
         RTS
RCVCMD2  BSET    #0,CNTRLM          Set to inout mode
         MOVE.B  CNTRLM,CNTRL
RCMDRDY  BTST    #0,CNTRL           Data ready to be read?
         BNE.S   RCMRDY1            Yes - go read data
         SUBI.W  #1,D5              Else decr. timeout
         BNE.S   RCMDRDY            If not zer try again
         BTST    #TYCON,GPIP        Is terminal connected?
         BEQ.S   RCMDRDY            No - continue check
         RTS
RCMRDY1  MOVE.L  PPRT,D7            Read data from parallel port
         RTS
*
TXCMD    BCLR    #0,CNTRLM          Set to output mode
         MOVE.B  CNTRLM,CNTRL
         MOVE.W  #1000,D5           Set timeout counter
TXRDY    BTST    #7,CNTRL           DRASS busy?
         BEQ.S   TXRDY1             No - go transmit data
         SUBI.W  #1,D5              Else decr. timeout counter
         BNE.S   TXRDY
         BTST    #TYCON,GPIP        Is terminal connected?
         BEQ.S   TXRDY              No - continue check
         RTS
TXRDY1   MOVE.L  D7,PPRT            Output data to parallel port
         RTS
*
TXDATA   BCLR    #0,CNTRLM          Set to output mode
         MOVE.B  CNTRLM,CNTRL
         MOVE.W  #1000,D5           set timeout counter
TXDAT0   MOVE.L  (A0),D7            fetch data lword
         ADD.B   D7,D0              calculate checksum
         LSR.L   #8,D7
         ADD.B   D7,D0
         LSR.L   #8,D7
         ADD.B   D7,D0
         LSR.L   #8,D7
         ADD.B   D7,D0
```

```
TXDAT1     BTST    #7,CNTRL            DRASS busy?
           BEQ.S   TXDAT2              No - go transmit data
           SUBI.W  #1,D5               Decr. timeout counter
           BNE.S   TXDAT1
           BTST    #TYCON,GPIP         Is terminal connected?
           BEQ.S   TXDAT1              No - continue check
           RTS
TXDAT2     MOVE.L  (A0)+,PPRT          Output data to parallel port
           CMPA.L  A0,A1               End of data block
           BGT.S   TXDAT0              No - go transmit next word
           RTS
*
************************************************************************
*
*          REAL TIME DIAGNOSTIC ROUTINES
*
*          IDLE LOOP
*
************************************************************************
*
IDLEJMP    NOP
           BRA     WAITLP
*
************************************************************************
*
*          PARALLEL TEST
*
*     PRLDIAG performs a free running test of the parallel port.
*     This test transmits a data pattern to the DRASS and the
*     DRASS echoes it back. For this reason, the DRASS must be
*     connected to ADAM and in Parallel diagnostic mode before the
*     test is run. ADAM signals the DRASS that it is performing a
*     parallel diagnostic test by first sending it the code FAF30001.
*     Then any data that is sent to the DRASS is echoed back. The
*     ADAM does all of the comparison of the data. The data patterns
*     used for the test are long word 00000000,01010101,02020202,...
*     FFFFFFFF. The errors detected by PRLDIAG are data errors,
*     transmit timeouts, and receive timeouts.
*
************************************************************************
```

```
*
PRLDIAG   MOVE.L  PPRT,D0          reset parallel port
          BSET    #0,CNTRLM        Put in input mode
          MOVE.B  CNTRLM,CNTRL
          MOVE.W  #$FF,D7          Set timeout counter
DRCON     BTST    #3,CNTRL         Is DRASS connected?
          BEQ.S   DRCON1           Yes - send command
          SUBI.W  #1,D7            Else decr. timeout
          BNE.S   DRCON
          BRA     CONERR           Display error if timeout
DRCON1    MOVE.L  #$FAF30001,D0    Send diag. command to DRASS
          BSR.S   TXDRAS
          CLR.L   D0               Init data pattern to 0
PDLOOP    BSR.S   TXDRAS           Send test data word
          CMPI.W  #0,D7            Timeout?
          BEQ     TXTMO            Yes - display error
          CMPI.L  #$FFFFFFFF,D0    End of test?
          BEQ.S   PRLRCV           Yes- receive status
          ADDI.L  #$01010101,D0    Else incr test pattern
          BRA.S   PDLOOP           And do again
PRLRCV    BSR.S   RCVDRAS          Else read data from DRASS
          CMPI.W  #0,D7            Timeout?
          BEQ     RCVTMO           Yes - display error
          CMPI.L  #$FAF30001,D3    Receive status good?
          BNE     TXDERR           No - display error
          BRA.S   PRLEXT           Yes - exit
*
*
TXDRAS    MOVE.W  #3000,D7          Set timeout counter
          BCLR    #0,CNTRLM        Set to output mode
          MOVE.B  CNTRLM,CNTRL
TXLOOP    BTST    #7,CNTRL         DRASS busy?
          BEQ.S   TXDCON           No - transmit data
          SUBI.W  #1,D7            Decr. timeout
          BNE.S   TXLOOP           Return if 0
          RTS
TXDCON    MOVE.L  D0,PPRT          Transmit data word
          RTS
*
*
RCVDRAS   MOVE.W  #3000,D7          Set timeout counter
RCVBSY    BTST    #7,CNTRL         DRASS busy?
          BEQ.S   RCVNBSY          No - go read data
          SUBI.W  #$1,D7           Else decr. timeout
          BNE.S   RCVBSY           Return if timeout
          RTS
```

```
RCVNBSY   MOVE.W #3000,D7          Set timeout counter
          BSET   #0,CNTRLM         Set to receive mode
          MOVE.B CNTRLM,CNTRL
RCVRDY    BTST   #0,CNTRL          Data ready?
          BNE.S  RCVDAT            Yes - go read data
          SUBI.W #1,D7             Else decr. timeout
          BNE.S  RCVRDY            Return if timeout
          RTS
RCVDAT    MOVE.L PPRT,D3           Read data from parallel port
          RTS
*
*
PRLEXT    BSET   #0,CNTRLM         Set parl. port to input mode
          MOVE.B CNTRLM,CNTRL
          BRA    WAITLP            Return to main loop
*
*
TXTMO     LEA.L  TXERM,A2          Display transmit timeout
          MOVE.W TXERC,D2           error massage
          BSR    DSPMSG
          MOVE.W #0,JMPADR         Set jump address to Idle loop
          BRA.S  PRLEXT
*
*
RCVTMO    LEA.L  RCVERM,A2         Display receive timeout
          MOVE.W RCVERC,D2          erro message
          BSR    DSPMSG
          MOVE.W #0,JMPADR         Set jump address to IDLE loop
          BRA    PRLEXT
*
*
CONERR    LEA.L  CONERM,A2         Display DRASS not connected
          MOVE.W CONERC,D2          error message
          BSR    DSPMSG
          MOVE.W #0,JMPADR         Set jump address to IDLE loop
          BRA    PRLEXT
*
*
TXDERR    LEA.L  TDATER,A2         Display receive data error
          MOVE.W TDATRC,D2          message (does not compare)
          BSR    DSPMSG
          MOVE.W #0,JMPADR         Set jump address to IDLE loop
          BRA    PRLEXT
*
*
```

```
*************************************************************************
*
*          FILTER CLOCK TEST
*
*      CLKTST is a free running test of the filter clocks. All four
*      clocks are first reset then set to 2KHz, 4KHz, 8KHz, 10KHz,
*      and 16KHz frequencies with a 3 second delay between each
*      setting. The test will continue to run until it is aborted
*      with the ESC (F4) key.
*
*************************************************************************
*
CLKTST     LEA.L   TACR,A0              Load pointer to prescale addr.
           LEA.L   TADR,A1              Load pointer to counter addr.
CLKRST     MOVE.L  #RSTALL,(A0)         Reset all clocks
           BSR     CLKWAIT              Wait 3 sec.
           CMPI.W  #10,JMPADR           Was F4 key entered
           BEQ.S   CLKTFIN              Yes - return
           MOVE.L  #CLKPRE1,(A0)        Output 2KHz prescale
           MOVE.L  #CNTDT1,(A1)         Output 2KHz counter
           BSR.S   CLKWAIT              Wait 3 sec.
           CMPI.W  #10,JMPADR           Was F4 key entered?
           BEQ.S   CLKTFIN              Yes - return
           MOVE.L  #CLKPRE2,(A0)        Output 4KHz orescale
           MOVE.L  #CNTDT2,(A1)         Output 4KHz counter
           BSR.S   CLKWAIT              Wait 3 sec.
           CMPI.W  #10,JMPADR           Was F4 key entered?
           BEQ.S   CLKTFIN              Yes - return
           MOVE.L  #CLKPRE3,(A0)        Output 8KHz prescale
           MOVE.L  #CNTDT3,(A1)         Output 8KHz counter
           BSR.S   CLKWAIT              Wait 3 sec.
           CMPI.W  #10,JMPADR           Was F4 key entered?
           BEQ.S   CLKTFIN              Yes - return
           MOVE.L  #CLKPRE4,(A0)        Output 10KHz prescale
           MOVE.L  #CNTDT4,(A1)         Output 10KHz counter
           BSR.S   CLKWAIT              Wait 3 sec.
           CMPI.W  #10,JMPADR           Was F4 key entered?
           BEQ.S   CLKTFIN              Yes - return
           MOVE.L  #CLKPRE5,(A0)        Output 16KHz prescale
           MOVE.L  #CNTDT5,(A1)         Output 16KHz counter
           BSR.S   CLKWAIT              Wait 3 sec.
           BSR     TMRINIT              Initialize timers
           BRA     WAITLP               Return to main loop
CLKTFIN    BSR     TMRINIT              Initialize timers
           BRA     WAITLP               Return to main loop
CLKWAIT    MOVE.L  #$80000,D0           3 sec. delay routine
CLKDLP     CLR.L   D1
           MULU    D1,D1
           SUBQ.L  #1,D0
           BNE.S   CLKDLP
           RTS
*
```

```
**********************************************************************
*
*          TELEMETRY TEST
*
*     TELMTST is a free running test of the telemetry port. This
*     routine is not interrupt driven but it does use the telemetry
*     port status bit to determine when to output the data word.
*     The data pattern that is transmitted is a sync code FAF32000
*     followed by incremental data 00010203 through FCFDFEFF. When
*     the test is aborted by the F4 key on the keypad, the interrupts
*     are re-enabled and the test is exited.
*
**********************************************************************
*
TELMTST    MOVE.W #SWORD0,SR          Disable interrupts
           MOVE.W #33,D7              Load test word counter
           MOVE.L #$00010203,D1       Load initial data pattern
           BCLR   #7,STATM            Enable telemetry port
           MOVE.B STATM,STAT
           MOVE.B #0,IPRA             Clear interrupt pending
           MOVE.W #$FF,TELMFL         Set telem. test running flg.
TELMTS1    MOVE.B CNTRL,D0            Telemetry active status?
           BTST   #4,D0
           BNE.S  TELMTS1             No - then check again
           CMPI.W #33,D7              Frame complete?
           BNE.S  TELMCON             No - go output next data wrd.
           MOVE.L #$FAF32000,TELE     Else output sync code
           SUBI.W #1,D7               Decr. word counter
           BRA.S  TELMTS1             Go to next word
TELMCON    MOVE.L D1,TELE             Output test data word
           ADDI.L #$04040404,D1       increment test pattern
           SUBI.W #1,D7               Decr. word counter
           CMPI.W #0,D7               End of frame?
           BNE.S  TELMTS1             No - go to next word
           MOVE.L #$00010203,D1       Reset data pattern
           MOVE.W #33,D7              Reset word counter
TELMRET    BTST   #5,D0               Keyboard input pending?
           BNE.S  TELMTS1             No - continue test
           MOVE.B UDR,KEY             Else read keyboard input
           MOVE.B #0,IPRA             Clear pending status
           CMPI.B #ESC,KEY            Esc key?
           BNE    TELMTS1             No - continue test
           MOVE.W #10,JMPADR          Else set jump addr. to IDLE
           BSET   #7,STATM            Mask off telemetry
           MOVE.B STATM,STAT
           MOVE.W #SWORD1,SR          Re-enable interrupts
           MOVE.W #0,TELMFL           Clear telem. test flag
           BRA    WAITLP              Go to main loop
*
```

```
*
******************************************************************************
*
*         DISPLAY TEST
*
*     DISPTST is a continuous running test of the display on the hand
*     held terminal. On each pass, 20 characters are sent to the display
*     and a 3 sec delay is performed. This will continue until the F4
*     key is detected.
*
******************************************************************************
*
*
DISPTST   MOVE.L DSPTPTR,A2          Load test pattern pointer
          MOVE.W #20,D2              Load character counter
          BSR    DSPMSG              Display test pattern
          ADDI.L #20,DSPTPTR         Incr. display pattern pointer
          CMPI.L #ENDSPT,DSPTPTR     End of display pattern?
          BLT.S  DISPCON             No - continue test
          MOVE.L #DSPTPAT,DSPTPTR    Else reset test pattern pointer
DISPCON   MOVE.L #$80000,D0          Delay 3 sec.
          MOVE.W #2,D1
DSPDLP    MULU   D1,D1
          SUBQ.L #1,D0
          BNE.S  DSPDLP
          BRA    WAITLP              Return to main loop
*
******************************************************************************
*
*         A/D TEST
*
*      ADDIAG is a continuous running test of the A/D converter and
*      filter clock for the channel that has been selected. The
*      channel that is used for the test is entered by the operator
*      and is passed to ADDIAG by MENUPR through ADCH. ADCH contains
*      the actual mux channel which consists of 4 A/D channels.
*      These are read, converted to ASCII, and displayed on each
*      pass through ADDIAG. When the F4 key is detected by MENUPR,
*      this routine is no longer executed.
*
***************************************************************************
*
ADDIAG    MOVE.L ADCH,ADC            Output channel to the mux
          MULU   D7,D7
          MULU   D7,D7
          MULU   D7,D7
          MOVE.L ADC,D0              Read A/D value
          CLR.L  D7                  Clear ASCII convert reg.
          LEA.L  DSPBUF,A0           Space out display buffer
          MOVE.L #$20202020,D3
          MOVE.L D3,(A0)
          MOVE.L D3,1(A0)
          MOVE.L D3,2(A0)
          MOVE.L D3,3(A0)
          MOVE.L D3,4(A0)
```

```
         MOVE.W #1,D3              Set to convert 1 byte
         ROL.L  #8,D0              Shift in msb of A/D input
         MOVE.B D0,D7              Store in byte to convert
         BSR    CVTASCI            Convert to ASCII
         ADD.L  #2,A0              Move display pntr past data
         MOVE.W #$2020,(A0)+       Store spaces in display
         ROL.L  #8,D0              Shift in next A/D value
         MOVE.B D0,D7              Store in byte to convert reg.
         MOVE.W #1,D3              Set to convert 1 byte
         BSR    CVTASCI            Convert to ASCII
         ADD.L  #2,A0              Bump pointer past data
         MOVE.W #$2020,(A0)+       Store in spaces
         ROL.L  #8,D0              Shift in next A/D value
         MOVE.B D0,D7              Store into byte to convert reg
         MOVE.W #1,D3              Set to convert 1 byte
         BSR    CVTASCI            Convert to ASCII
         ADD.L  #2,A0              Bump pointer past data
         MOVE.W #$2020,(A0)+       Store inb spaces
         ROL.L  #8,D0              Shift in last A/D value
         MOVE.B D0,D7              Store in byte to convert reg
         MOVE.W #1,D3              Set to convert 1 byte
         BSR    CVTASCI            Convert to ASCII
         ADD.L  #2,A0              Bump pointer past data
         MOVE.W #$010A,(A0)        Store in Home & Line feed
         LEA.L  DSPBUF,A2          Display A/D message
         MOVE.W #16,D2
         BSR    DSPMSG
         BRA    WAITLP             Go to main loop
*
***********************************************************************
*
*        CALABRATION ALIGNMENT OF A/D
*
*     ALIGN is similar to ADDIAG except that it samples the channel
*     in ALNCH in both RCAL mode and Non-RCAL mode. Both values for
*     each of the four A/D channels is then converted to ASCII and
*     displayed. This process is performed on each pass through
*     ALIGN until the MENUPR detects the F4 key.
*
***********************************************************************
*
ALIGN    BSET   #6,CNTRLM          Set to non-RCAL mode
         MOVE.B CNTRLM,CNTRL
         MOVE.W #$8000,D0          Load delay counter
ALNDEL1  NOP                       Delay
         MULU   D7,D7
         SUBI.W #1,D0
         BNE.S  ALNDEL1
         MOVE.W #20,D7             Set counter for 20 samples
ALIGN1   MOVE.L ALNCH,ADC          Set mux channel
         MULU   D0,D0
         MULU   D0,D0
         MULU   D0,D0
         MOVE.L ADC,D0             Read A/D values
         SUBI.W #1,D7
```

```
         BNE.S    ALIGN1                  Do again 20 times
         CLR.L    D7                      Clear ASCII convert reg.
         LEA.L    DSPBUF,A0               Space out display buffer
         MOVE.L   #$20202020,D3
         MOVE.L   D3,(A0)
         MOVE.L   D3,1(A0)
         MOVE.L   D3,2(A0)
         MOVE.L   D3,3(A0)
         MOVE.L   D3,4(A0)
         MOVE.L   D3,5(A0)
         MOVE.L   D3,6(A0)
         MOVE.L   D3,7(A0)
         MOVE.L   D3,8(A0)
         MOVE.W   #1,D3                   Convert one byte
         ROL.L    #8,D0                   Shift in first A/D value
         MOVE.B   D0,D7                   Store in ASCII convert reg.
         BSR      CVTASCI                 Convert to ASCII
         ADD.L    #2,A0                   Incr. past data value
         MOVE.W   #$2020,(A0)+            Store spaces
         ROL.L    #8,D0                   Shift in next A/D value
         MOVE.B   D0,D7                   Store in next byte to convert
         MOVE.W   #1,D3                   Convert 1 byte
         BSR      CVTASCI                 Convert to ASCII
         ADD.L    #2,A0                   Bump past data value
         MOVE.W   #$2020,(A0)+            Store in spaces
         ROL.L    #8,D0                   Shift in next A/D value
         MOVE.B   D0,D7                   Store in Convert reg.
         MOVE.W   #1,D3                   Set to convert 1 byte
         BSR      CVTASCI                 Convert to ASCII
         ADD.L    #2,A0                   Bump pass data
         MOVE.W   #$2020,(A0)+            Store in spaces
         ROL.L    #8,D0                   Shift in next A/D value
         MOVE.B   D0,D7                   Store in convert reg.
         MOVE.W   #1,D3                   Set to convert 1 byte
         BSR      CVTASCI                 Convert to ASCII
         ADD.L    #2,A0                   Bump pass data
         MOVE.W   #$0D0A,(A0)+            Store carriage return,line feed
         MOVE.B   #$0A,(A0)+              Store another linefeed
         BCLR     #6,CNTRLM               Set to RCAL mode
         MOVE.B   CNTRLM,CNTRL
         MOVE.W   #$8000,D0               Set delay counter
ALNDEL   NOP                              Delay
         MULU     D7,D7
         SUBI.W   #1,D0
         BNE.S    ALNDEL
         MOVE.W   #20,D7                  Set sample counter to 20
ALIGN2   MOVE.L   ALNCH,ADC               Output mux channel
         MULU     D0,D0
         MULU     D0,D0
         MULU     D0,D0
         MOVE.L   ADC,D0                  Read A/D value
         SUBI.W   #1,D7
```

```
BNE.S   ALIGN2              Do again 20 times
CLR.L   D7                  Clear convert reg.
MOVE.W  #1,D3               set to convert 1 byte
ROL.L   #8,D0               shift in high value
MOVE.B  D0,D7               Store in convert reg.
BSR     CVTASCI             Convert to ASCII
ADD.L   #2,A0               Bump past data
MOVE.W  #$2020,(A0)+        Store in spaces
ROL.L   #8,D0               Shift in next data byte
MOVE.B  D0,D7               Store in convert reg.
MOVE.W  #1,D3               Set to convert 1 byte
BSR     CVTASCI             Convert to ASCII
ADD.L   #2,A0               Bump past data
MOVE.W  #$2020,(A0)+        Store in spaces
ROL.L   #8,D0               Shift in next data byte
MOVE.B  D0,D7               Store value in convert reg.
MOVE.W  #1,D3               Convert 1 byte
BSR     CVTASCI             Convert to ASCII
ADD.L   #2,A0               Bump past data
MOVE.W  #$2020,(A0)+        Store in spaces
ROL.L   #8,D0               Shift in last data byte
MOVE.B  D0,D7               Store in convert reg.
MOVE.W  #1,D3               Set to convert 1 byte
BSR     CVTASCI             Convert to ASCII
ADD.L   #2,A0               Bump past data
MOVE.W  #$010A,(A0)         Store Home & linefeed
BSET    #6,CNTRLM           Set to non-RCAL mode
MOVE.B  CNTRLM,CNTRL
LEA.L   DSPBUF,A2           Display results
MOVE.W  #33,D2
BSR     DSPMSG
BRA     WAITLP              Return to main loop
```

```
*********************************************************************
*
*           DISPLAY LAST MENU JUMP ROUTINE
*
*       DSPLMU is activated by MENUPR when the F4 key is detected.
*       It's pupose is to deactivet the current menu and display
*       the previous level menu. If the main menu is current, then
*       it is redisplayed. The variables that determine the menu
*       level and operation are:
*                     MU1SL - Level 1 (main menu)
*                     MU2SL - Level 2
*                     MU3SL - Level 3
*                     MU4SL - Level 4
*
*       The values in these variables provide an index to the tables
*       that contain the menu addresses and charactercounts. These
*       tables are:
*                     MU2SAD - Level 2 menu addresses
*                     MU2SCT - Level 2 character counts
*                     MU3SAD - Level 3 menu addresses
*                     MU3SCT - Level 3 character counts
*
*       Level 1 menu is always the main menu. Level 4 menues are
*       never a previous menu. After the previous menu has been
*       determined and displayed, DSPLMU sets JMPADR to 0 which
*       causes the system to execute the IDLE loop.
*
*********************************************************************
*
DSPLMU    CMPI.W #0,MU4SL          Is level 4 menu set?
          BEQ.S  LMU3              No - then check level 3
          CLR.W  D6
          MOVE.W D6,MU4SL          Clear level 4 index
          MOVE.W MU3SL,D1          Get level 3 index
          SUBI.W #1,D1             Calc. menu pointer
          LSL.W  #2,D1
          LEA.L  MU3SAD,A0         Get level 3 menu table
          MOVE.L (A0,D1),A2        Get level 3 menu address
          LEA.L  MU3SCT,A0         Get level 3 menu count
          MOVE.L (A0,D1),A1
          MOVE.W (A1),D2
          BSR    DSPMSG            Display level 3 menu
          BRA.S  DLMURET           Return
LMU3      CMPI.W #0,MU3SL          Level 3 menu set
          BEQ.S  LMU2              No - check level 2
          CLR.W  D6
          MOVE.W D6,MU3SL          Clear level 3 menu index
          MOVE.W MU2SL,D1          Get level 2 menu index
          SUBI.W #1,D1
          LSL.W  #2,D1
          LEA.L  MU2SAD,A0
          MOVE.L (A0,D1),A2        Get levl 2 menu address
          LEA.L  MU2SCT,A0
          MOVE.L (A0,D1),A1
```

```
         MOVE.w  (A1),D2                Get level 2 menu count
         BSR     DSPMSG                 Display level 2 menu
         BRA.S   DLMURET                Return
LMU2     CLR.W   D6
         MOVE.W  D6,MU2SL               Clear leve 2 menu index
         LEA.L   MENU1,A2               Display level 2 menu
         MOVE.W  MEN1CT,D2                (main menu)
         BSR     DSPMSG
DLMURET  MOVE.W  #0,JMPADR              Set jump address to IDLE
         BRA     WAITLP
*
***********************************************************************
*
*          TELEMETRY INTERRUPT SERVICE ROUTINE
*
*    DAQINT is the interrupt service routine for the telemetry
*    interrupt (interrupt 6). This is the routine that reads the
*    A/D data, and outputs the data to the telemetry port. Associated
*    with DAQINT are IDLE which outputs the next channel number and
*    updates the sync code; PRECAL which also stores the pre-calibration
*    data; DATCOL which also stores the test data; and POSTCAL which
*    also stores the post calibration data. DAQINT is what performs the
*    indirect jump to one of these other routines depending upon what
*    stage the data acquisition is in. The register A3 contains the
*    the jump address for the data acquisition routine.
*
***********************************************************************
*
DAQINT   MOVEM.L D0/D5,-(A7)            Save registers
         CLR.L   D0
         MOVE.L  ADC,D5                 Read A/D value
         MOVE.L  D5,TELE                Output value to telemetry port
         JMP     (A3)                   Go to processing routine
*
*
***********************************************************************
*
*          DATA ACQUISITION DURING IDLE STATE
*
*    IDLE is executed after the precal data has been collected and
*    before the start for data collection has been received. It is
*    also executed for 3 min. after the post cal. data has been
*    collected and before the system is powered down. IDLE will
*    update the A/D channel number for the next scan and also update
*    the frame count and output the sync code when the end of the
*    scan table has been reached.
*
***********************************************************************
*
```

```
IDLE      CMPA.L ENDSCT,A4         End of scan table?
          BLE.S  IDLE1             No - continue with next ch.
          ADDI.B #1,FCNTR          Else incr. frame counter
          MOVE.L SYNC,TELE         Output sync to telemetry port
          LEA.L  STSCT,A4          Set to start of scan table
          MOVE.B (A4)+,D0          Get first mux channel
          MOVE.L D0,ADC            Output mux channel
          BRA.S  IDLERET           Go to return
IDLE1     MOVE.B (A4)+,D0          Get next mux channel
          MOVE.L D0,ADC            Output mux channel
IDLERET   BTST   #2,TSTST          Is it post cal time?
          BEQ.S  IDLRET1           If not go return
          SUB.L  #1,STPCNT         Else decr. 3 min counter
          BNE.S  IDLRET1           if not 0 return
          BSET   #3,TSTST
IDLRET1   MOVE.L ADC,D5            Start A/D
          MOVEM.L (A7)+,D0/D5      Restore registers
          RTE
*
*********************************************************************
*
*         SAVING DATA ROUTINE
*
*    DATCOL collects the test data from the time the start signal
*    is received until the memory is full. The data is collected from
*    the channels specified in the scan table (STSCT) and stored
*    in DATBUF which is indexed by register A5. A5 is initialized
*    by PRECAL upon it's exit. DATCOL also increments the frame
*    counter and outputs the sync to the telemetry port array
*    each time the end of the channel array (STSCT) is reached.
*    Once DATBUF is full, DATCOL Returns control back to the
*    IDLE routine to wait for the 3 min delay before the post
*    cal. data can be collected.
*
*********************************************************************
*
DATCOL    CMPA.L #ENDBUF,A5        End of data buffer reached?
          BLE.S  DATSET            No - continue
          BSET   #2,TSTST          Set post cal status
          BCLR   #1,TSTST          Clear Test data collect status
          BCLR   #7,TSTST          Clear data storage status
          LEA.L  IDLE,A3           Set for Idle routine
DATSET    CMPA.L ENDSCT,A4         End of scan table reached?
          BLE.S  DATSET1           No - continue
          ADDI.B #1,FCNTR          Else incr. frame counter
          MOVE.L SYNC,TELE         Output sync code
          LEA.L  STSCT,A4          Reset scan table pointer
          MOVE.B (A4)+,D0          Get first mux channel
          MOVE.L D0,ADC            Output channel to mux
          BTST   #1,TSTST          Data collection mode?
          BEQ.S  DATRET            No - return
          BTST   #7,TSTST          Data storage mode?
          BNE.S  DATRET            No - return
          BSET   #7,TSTST          Else set data storage mode
          MOVE.L SYNC,DAQSYNC      Save first sync code
          BRA.S  DATRET            Return
```

```
DATSET1   MOVE.B  (A4)+,D0              Get next mux channel
          MOVE.L  D0,ADC                Output channel to mux
          BTST    #7,TSTST              Data storage mode?
          BEQ.S   DATRET                No - return
          MOVE.L  D5,(A5)+              Else store data in memory
DATRET    MOVE.L  ADC,D5                Start A/D for next time
          MOVEM.L (A7)+,D0/D5           Restore registers
          RTE
*
***********************************************************************
*
*         KEY PAD INTERRUPT SERVICE ROUTINE
*
*    KEYINT services the receive interrupt (interrupt 2) generated
*    by the hand held terminal through the serial port on the MFP.
*    It first calls GETKEY which actually reads the input character.
*    Then it checks for certain control characters (ENT,ESC,.,-). If
*    it is one of those characters, then it calls MENUPR to parse the
*    message in KEYBUF and KEYINT returns. All registers used by
*    KEYINT, GETKEY, and MENUPR are saved by KEYINT, these include
*    registers D0,D1,D2,D3,D4,D7,A0,A1,and A3.
*
***********************************************************************
*
KEYINT    NOP
          MOVEM.L D0-D4/D7/A0-A1/A3,-(A7)    Save registers used
          MOVE.B  #0,IPRA               Clead interrupt pending
          BSR     GETKEY                Go read the input key
          CMPI.B  #ENT,KEY              Is it the ENT key?
          BEQ.S   KEYPR                 Yes - go process input buffer
          CMPI.B  #ESC,KEY              Is it the F4 key
          BEQ.S   KEYPR                 Yes - go process ESC key
          CMPI.B  #DOT,KEY              Is it the "." key?
          BEQ.S   KEYPR                 Yes - go process input buffer
          CMPI.B  #HYFN,KEY             Is it the "-" key
          BEQ.S   KEYPR                 Yes - go process input buffer
          BRA.S   KEYRET                Else return
KEYPR     BSR     MENUPR                Execute the menu processor
KEYRET    MOVEM.L (A7)+,D0-D4/D7/A0-A1/A3    Restore registers
          RTE                           Return
*
*
BUSERR    NOP                           Buss error trap
          RTE
ADDERR    NOP                           Address error trap
          RTE
ERRVEC    NOP                           Misc. error trap
          RTE
PRIVER    NOP                           Priveledge error trap
          RTE
SPURINT   NOP                           Spurious interrupt trap
          RTE
AVEC      NOP                           Misc. auto vector trap
          RTE
*
```

```
*****************************************************************
*
*          ROM CONSTANTS DEFINITION
*
*****************************************************************
*
ROMPRM     DC.B    $BE                     UCNTRL
           DC.B    $0B                     BAUD
           DC.B    $07                     PRSCA
           DC.B    $07                     PRSCB
           DC.B    $77                     PRSCCD
           DC.B    01                      FLCNTA
           DC.B    01                      FLCNTB
           DC.B    01                      FLCNTC
           DC.B    01                      FLCNTD
           DC.B    $FA                     SYNC
           DC.B    $F3
           DC.B    $20
PRMEND     DC.B    0                       FCNTR
PRMSG1     DC.B    $0C
           DC.B    'VALID DATA PRESENT'
           DC.B    $0D,$0A
           DC.B    'ERASE DATA(Y/N)?'
           DC.B    $0D,$0A
PRMCNT1    DC.W    39
PRMSG2     DC.B    'DIAGNOSTIC ERRORS:'
           DC.B    $0D,$0A
PRMCNT2    DC.W    20
ROMER      DC.B    'ROM '
SERER      DC.B    'SER '
FLCERR     DC.B    'FCL '
ADER       DC.B    'A/D '
PRLERR     DC.B    'PRL '
RAMER      DC.B    'RAM '
ERRCNT     DC.W    4
MENU1      DC.B    $0C
           DC.B    '1.DIAG     2.CAL'
           DC.B    $0D,$0A
           DC.B    '3.PAR.SET 4.TX DATA'
           DC.B    $0D,$0A
           DC.B    '5.DAT.COL 6.PURGE'
           DC.B    $0D,$0A
MEN1CT     DC.W    58
MENU2      DC.B    $0C
           DC.B    '1.MEMORY  2.SERIAL'
           DC.B    $0D,$0A
           DC.B    '3.A/D     4.CLOCK'
           DC.B    $0D,$0A
           DC.B    '5.TELEM.  6.PARAL.'
           DC.B    $0D,$0A
MEN2CT     DC.W    60
MENU3      DC.B    $0C
           DC.B    '1.CH. CHECK'
           DC.B    $0D,$0A
           DC.B    '2.ALIGN'
           DC.B    $0D,$0A
```

```
MEN3CT    DC.W    23
MENU4     DC.B    $0C
          DC.B    '1.CH.SPEC 2.CLK RATE'
          DC.B    $00
          DC.B    '3.SER.DEF. 4.POWER'
          DC.B    $0D,$0A
MEN4CT    DC.W    42
MENU5     DC.B    $0C
          DC.B    '1.PATTERN 2.ADDRESS'
          DC.B    $0D,$0A
          DC.B    '3.BUBBLE0 4.BUBBLE1'
          DC.B    $0D,$0A
          DC.B    '5.TEST ALL'
          DC.B    $0D,$0A
MEN5CT    DC.W    55
MENU6     DC.B    $0C
          DC.B    '1.DISPLAY'
          DC.B    $0D,$0A
          DC.B    '2.KEYBOARD'
          DC.B    $0D,$0A
MEN6CT    DC.W    24
MENU7     DC.B    $0C
          DC.B    '1.SEQ.ALL 2.SEL.CH'
          DC.B    $0D,$0A
          DC.B    '3.DISPLAY CH.'
          DC.B    $0D,$0A
MEN7CT    DC.W    36
MENU8     DC.B    $0C
          DC.B    '1.CLK A  2.CLK B'
          DC.B    $0D,$0A
          DC.B    '3.CLK C  4.CLK D'
          DC.B    $0D,$0A
MEN8CT    DC.W    37
MENU9     DC.B    $0C
          DC.B    '1.2K  2.4K  3.8K'
          DC.B    $0D,$0A
          DC.B    '4.10K 5.16K 6.OTH'
          DC.B    $0D,$0A
MEN9CT    DC.W    38
PWRMU     DC.B    $0C
          DC.B    '1.POWER ON'
          DC.B    $0D,$0A
          DC.B    '2.POWER OFF'
          DC.B    $0D,$0A
PWRCT     DC.W    26
ERRMSG    DC.B    '     ERROR'
          DC.B    $0D
ERRMCT    DC.W    10
ADPRMT    DC.B    $0C
          DC.B    'ENTER MUX CH:'
          DC.B    $0D,$0A
ADPRCT    DC.W    16
CLKTSTM   DC.B    $0C
```

```
         DC.B    $0D,$0A
         DC.B    '    CLOCK TEST    '
         DC.B    $0D,$0A
         DC.B    '    RUNNING'
CLKTCT   DC.W    34
TELTSTM  DC.B    $0C
         DC.B    $0D,$0A
         DC.B    '  TELEMETRY TEST   '
         DC.B    $0D,$0A
         DC.B    '       RUNNING   '
TELTCT   DC.W    40
PARTSTM  DC.B    $0C
         DC.B    $0D,$0C
         DC.B    '  PARALLEL TEST   '
         DC.B    $0D,$0A
         DC.B    '       RUNNING  '
PARTCT   DC.W    39
MEMMSG1  DC.B    $0C
         DC.B    $0D,$0A
         DC.B    '   PATTERN TEST'
         DC.B    $0D,$0A
         DC.B    '       RUNNING'
         DC.B    $0D,$0A
MEMCT1   DC.W    35
MEMMSG2  DC.B    $0C
         DC.B    $0D,$0A
         DC.B    '   ADDRESS TEST'
         DC.B    $0D,$0A
         DC.B    '       RUNNING'
         DC.B    $0D,$0A
MEMCT2   DC.W    35
MEMMSG3  DC.B    $0C
         DC.B    $0D,$0A
         DC.B    '   BUBBLE 0 TEST'
         DC.B    $0D,$0A
         DC.B    '       RUNNING'
         DC.B    $0D,$0A
MEMCT3   DC.W    36
MEMMSG4  DC.B    $0C
         DC.B    $0D,$0A
         DC.B    '   BUBBLE 1 TEST'
         DC.B    $0D,$0A
         DC.B    '       RUNNING'
         DC.B    $0D,$0A
MEMCT4   DC.W    36
MEMMSG5  DC.B    $0C
         DC.B    '   TEST ALL'
         DC.B    $0D,$0A
         DC.B    '   RUNNING'
         DC.B    $0D,$0A
MEMCT5   DC.W    29
KBPRMT   DC.B    $0C
```

```
              DC.B    'ENTER KEYS:'
KBCT          DC.W    12
FCLKMSG       DC.B    $0C
              DC.B    ' ENTER PRESCALE'
              DC.B    $0D,$0A
              DC.B    '  AND COUNT(X.XXX)'
              DC.B    $0D,$0A
FCLKCT        DC.W    39
PARMSG        DC.B    $0C
              DC.B    'PARITY(E/O/N):'
              DC.B    $0D,$0A
PARCT         DC.W    17
STBMSG        DC.B    $0D,$0A
              DC.B    'STOP BITS(1/2):'
              DC.B    $0D,$0A
STBCT         DC.W    19
WRDLMSG       DC.B    $0D,$0A
              DC.B    'WORD LEN(5/6/7/8):'
              DC.B    $0D $0A
WRDLCT        DC.W    22
BAUDMSG       DC.B    $0D,$0A
              DC.B    'BAUD RATE(XXXX):'
              DC.B    $0D,$0A
BAUDCT        DC.W    20
CLRENT        DC.B    '                                  '
              DC.B    $0D
CLRECT        DC.W    21
MEMPAS        DC.B    $0C
              DC.B    '      MEMORY TEST'
              DC.B    $0D,$0A
              DC.B    '          PASSED'
              DC.B    $0D,$0A
MEMPCT        DC.W    34
MEMERM        DC.B    $0C
              DC.B    '      MEMORY ERROR'
              DC.B    $0D,$0A
              DC.B    'VAL EXP:'
MEMERC        DC.W    28
MEMERM1       DC.B    $0D,$0A
              DC.B    'VAL READ:'
MEMERC1       DC.W    11
MEMERM2       DC.B    $0D,$0A
              DC.B    'ADDRESS:'
MEMERC2       DC.W    10
CCPASS        DC.B    $0C
              DC.B    $0D,$0A
              DC.B    '      CHANNEL CHECK'
              DC.B    $0D,$0A
              DC.B    '      TEST PASSED'
CCPCT         DC.W    39
CCMSG         DC.B    $0C
              DC.B    'BAD CHANNELS:'
              DC.B    $0D,$0A
```

```
CCMCT     DC.W    16
DSCTMSG   DC.B    $0C
          DC.B    'MUX CHANNELS:'
          DC.B    $0D,$0A
DSCTCT    DC.W    16
ALNHDR1   DC.B    $0C
          DC.B    'NON-RCAL   CH:'
ALNCT1    DC.W    14
ALNHDR2   DC.B    $0D,$0A,$0A
          DC.B    'RCAL READINGS'
          DC.B    $01,$0A
ALNCT2    DC.W    18
ADHDR     DC.B    $0C
          DC.B    'A/D TEST CH: '
ADCT1     DC.W    14
ADHDR2    DC.B    $0D,$0A
ADCT2     DC.W    2
TXERM     DC.B    $0C,$0A
          DC.B    'TRANSMIT TIME-OUT'
          DC.B    $0D,$0A
          DC.B    '       ERROR'
          DC.B    $0D,$0A
TXERC     DC.W    33
RCVERM    DC.B    $0C,$0A
          DC.B    'RECEIVE TIME-OUT'
          DC.B    $0D,$0A
          DC.B    '       ERROR'
          DC.B    $0D,$0A
RCVERC    DC.W    32
CONERM    DC.B    $0C,$0A
          DC.B    'DRASS CONNECT'
          DC.B    $0D,$0A
          DC.B    '    TIME-OUT'
          DC.B    $0D,$0A
CONERC    DC.W    30
RDATER    DC.B    $0C,$0A
          DC.B    'RECEIVE  DATA'
          DC.B    $0D,$0A
          DC.B    '       ERROR'
          DC.B    $0D,$0A
RDATC     DC.W    28
TDATER    DC.B    $0C,$0A
          DC.B    '    TRANSMIT DATA'
          DC.B    $0D,$0A
          DC.B    '         ERROR'
          DC.B    $0D,$0A
TDATRC    DC.W    34
MEMFLM    DC.B    $0C,$0A,$0A
          DC.B    '    MEMORY FULL'
          DC.B    $0D,$0A
MEMFLC    DC.W    19
NDATMSG   DC.B    $0C,$0A,$0A
          DC.B    '    NO DATA PRESENT'
          DC.B    $0D,$0A
```

```
NDATMCT    DC.W    23
PUERM      DC.B    $0C,$0A
           DC.B    'POWER-UP DIAGNOSTIC'
           DC.B    $0D,$0A
           DC.B    '        FAILURE'
           DC.B    $0D,$0A
PUERC      DC.W    38
DATFLM     DC.B    $0C,$0A
           DC.B    '    VALID DATA IN'
           DC.B    $0D,$0A
           DC.B    '          MEMORY'
           DC.B    $0D,$0A
DATFLC     DC.W    35
DRACON     DC.B    $0C,$0A
           DC.B    '    DRASS NOT'
           DC.B    $0D,$0A
           DC.B    '    CONNECTED'
           DC.B    $0D,$0A
DRASCC     DC.W    30
TDATMSG    DC.B    $0C,$0A,$0A
           DC.B    '   TRANSFERING DATA'
           DC.B    $0D,$0A
TDATC      DC.W    23
TDCOMP     DC.B    $0C,$0A
           DC.B    '    DATA TRANSFER'
           DC.B    $0D,$0A
           DC.B    '       COMPLETE'
           DC.B    $0D,$0A
TDCOMPC    DC.W    35
CLCTDA     DC.B    $0C,$0A
           DC.B    'COLLECTING DATA'
           DC.B    $0D,$0A
CLCTDC     DC.W    19
SYSPAS     DC.B    $0C,$0D,$0A
           DC.B    ' ADAM SYSTEM READY'
VERNUM     DC.B    $0D,$0A
           DC.B    ' VERSION 1.1'
           DC.B    $0D,$0A
SYSPASC    DC.W    37
VERCNT     DC.W    16
PURPRMT    DC.B    $0C
           DC.B    '    DO YOU WANT THE'
           DC.B    $0D,$0A
           DC.B    '    DATA MODULES'
           DC.B    $0D,$0A
           DC.B    '    ERASED (Y/N)?'
           DC.B    $0D,$0A
PURPRC     DC.W    56
DSPTPAT    DC.B    'ABCDEFGHIJKLMNOPQRST'
           DC.B    'UVWXYZ0123456789./?-'
           DC.B    'ZYXWVUTSRQPONMLKJIHG'
           DC.B    'FEDCBA'
           DC.B    '9876543210./'
ENDSPT     DC.L    *
```

```
         DC.B    '      '
CLK2K    DC.B    05
CLK4K    DC.B    10
CLK8K    DC.B    05
CLK10K   DC.B    01
CLK16K   DC.B    31
PRE2K    DC.B    07
PRE4K    DC.B    04
PRE8K    DC.B    04
PRE10K   DC.B    07
PRE16K   DC.B    01
JMPTBL   DC.L    IDLEJMP
         DC.L    ADDIAG
         DC.L    CLKTST
         DC.L    TELMTST
         DC.L    PRLDIAG
         DC.L    DISPTST
         DC.L    ALIGN
         DC.L    IDLEJMP
         DC.L    IDLEJMP
         DC.L    IDLEJMP
         DC.L    DSPLMU
*
MU2SAD   DC.L    MENU2
         DC.L    MENU3
         DC.L    MENU4
MU2SCT   DC.L    MEN2CT
         DC.L    MEN3CT
         DC.L    MEN4CT
*
MU3SAD   DC.L    MENU5
         DC.L    MENU6
         DC.L    ADPRMT
         DC.L    CLKTSTM
         DC.L    TELTSTM
         DC.L    PARTSTM
         DC.L    ERRMSG
         DC.L    ADPRMT
         DC.L    MENU7
         DC.L    MENU8
         DC.L    PARMSG
         DC.L    STBMSG
         DC.L    WRDLMSG
         DC.L    BAUDMSG
         DC.L    ADPRMT
         DC.L    MENU9
         DC.L    MENU9
         DC.L    MENU9
         DC.L    MENU9
```

```
MU3SCT     DC.L    MEN5CT
           DC.L    MEN6CT
           DC.L    ADPRCT
           DC.L    CLKTCT
           DC.L    TELTCT
           DC.L    PARTCT
           DC.L    ERRMCT
           DC.L    ADPRMT
           DC.L    MEN7CT
           DC.L    MEN8CT
           DC.L    PARCT
           DC.L    STBCT
           DC.L    WRDLCT
           DC.L    BAUDCT
           DC.L    ADPRCT
           DC.L    MEN9CT
           DC.L    MEN7CT
           DC.L    MEN9CT
           DC.L    MEN7CT
*
BDTBL      DC.L    0
           DC.L    0
           DC.B    $35,$30,0,0
           DC.B    $37,$35,0,0
           DC.B    $31,$33,$34,0
           DC.B    $32,$30,$30,0
           DC.B    $36,$30,$30,0
           DC.B    '2400'
           DC.B    '9600'
           DC.B    '4800'
           DC.B    '1800'
           DC.B    '1200'
           DC.B    '2400'
           DC.B    $33,$30,$30,0
           DC.B    $31,$35,$30,0
           DC.B    $31,$31,$30,0
ENDROM     DS.W    1
*
*****************************************************
*
*          DATA STORAGE DEFINITION
*
*****************************************************
*
           OFFSET $107D000
PREBUF     DS.B    4096
POSTBU     DS.B    4096
STSCT      DS.B    512
ENDSCT     DS.L    1
BLKLEN     DS.L    1
UCNTRL     DS.B    1
BAUD       DS.B    1
PRSCA      DS.B    1
PRSCB      DS.B    1
PRSCCD     DS.B    1
FLCNTA     DS.B    1
FLCNTB     DS.B    1
```

```
FLCNTC     DS.B   1
FLCNTD     DS.B   1
SYNC       DS.B   3
FCNTR      DS.B   1
TSTST      DS.B   1
DSTAT      DS.B   1
STATM      DS.B   1
CNTRLM     DS.B   1
MU1SL      DS.W   1
MU2SL      DS.W   1
MU3SL      DS.W   1
MU4SL      DS.W   1
KEYBUF     DS.B   20
PARCHK     DS.L   1
KEY        DS.B   1
CALCNT     DS.W   1
SERST      DS.W   1
PRESYNC    DS.L   1
DAQSYNC    DS.L   1
POSSYNC    DS.L   1
JMPADR     DS.W   1
TXFLG      DS.W   1
CLCTD      DS.W   1
TELMTX     DS.W   1
TELMWRD    DS.L   1
KBFLG      DS.W   1
DSPTST     DS.W   1
DSPTPTR    DS.L   1
ADCHD1     DS.B   1
ADCHD2     DS.B   1
INVAL      DS.W   1
ADCH       DS.L   1
ALNCH      DS.L   1
ALNCHD1    DS.B   1
ALNCHD2    DS.B   2
TMPST      DS.B   512
ENTMPST    DS.L   1
TSERCTL    DS.B   1
TMPBD      DS.L   1
OTHCLK     DS.W   1
PARNO      DS.W   1
TMPPRE     DS.B   1
TMPCLK     DS.B   1
DSPBUF     DS.B   400
CCBUF1     DS.L   33
CCBUF2     DS.L   33
TMPCHR     DS.B   1
TMPCVT     DS.L   1
MEMFAIL    DS.W   1
TSTAFL     DS.W   1
TELMFL     DS.W   1
RETRY      DS.W   1
SYNCTMP    DS.L   1
STPCNT     DS.L   1
           END
```

```
*
******************************************************************************
*
*                ROMTST
*
*       This calculates a sumcheck of the PROM and compares that value
*       with the sumcheck stored in address FFFF. If the test fails,
*       bit 0 in DSTAT is set to be checked after power up diagnostics
*       is complete.
*
*            Input parameters : None
*
*            Output parameters : DSTAT - contains the pass/fail results
*                                        of the test
*
*            Registers used : A0 - running index through PROM
*                             D0 - end address of PROM
*                             D1 - running checksum total
*
******************************************************************************
*
ROMTST     IDNT    1,1
PROM       EQU     0
           INCLUDE ADAMDEF
           XDEF    ROMTST
*
ROMTST     LEA.L   PROM,A0              Load PROM start address
           MOVE.L  #$FFFF,D0            Get address of ROM
           CLR.L   D1                   Clear checksum
           CLR.L   D7
ROMLP      MOVE.B  (A0)+,D7             Accumulate checksum
           EOR.B   D7,D1
           CMPA.L  D0,A0                End of PROM?
           BNE     ROMLP                No - continue
           MOVE.W  #0,CCR               Clear carry bit
           CMP.B   (A0),D1              Compare checksum with ENDROM
           BEQ.S   ROMPAS               Passed - return
           BSET    #0,DSTAT             Else set diag failure
ROMPAS     RTS
           END
```

```
*
*****************************************************************:
*
*             SERTST
*
*      This routine performs the power up diagnostic on the UART
*      in the MFP. It performs an internal loop back test with a
*      canned message. SERTST does not utilize the receive interrupt
*      but it does test the receive status. The data format and
*      baud rate used during the test is the same that's set up
*      during system initialization.
*
*          Input parameters : None
*
*          Output parameters : DSTAT - power up diagnostic status
*                              SERST - 1- frame error
*                                      2- parity error
*                                      4- overrun error
*
*          Registers used : A0 - index to test pattern
*                           D0 - test data character
*                           D1 - Temp UART status
*                           D7 - character count
*
*****************************************************************:
*
SERTST     IDNT    1,1
IMRA       EQU     $800049             Interrupt mask reg.
RSR        EQU     $800055             Receive status
TSR        EQU     $800056             Transmit status
UDR        EQU     $800057             Serial data reg.
LPBAK      EQU     07                  Loopback command
TXENA      EQU     $05                 Transmit enable command
           INCLUDE ADAMDEF
           XDEF    SERTST
*
```

```
SERTST   BCLR    #4,IMRA          Disable serial interrupt
         MOVE.B  #LPBAK,TSR       Set to loopback mode
         LEA.L   TSTPAT,A0        Set pointer to test pattern
         MOVE.L  #31,D7           Set character count
SERLP    MOVE.B  (A0),D0          Get test character
         BSR.S   XMIT             Transmit it
         BSR.S   RECV             Go read character
         CMP.B   (A0)+,D0         Compare input character
         BNE.S   SERERR           If not = set error
         CMPI.W  #0,SERST         Any other errors?
         BNE.S   SERERR           Yes - set error bit
         DBNE    D7,SERLP         Else decr character count & do again
         MOVE.B  #TXENA,TSR       Else re init transmitter
         BSET    #4,IMRA          Re enable interrupt
         RTS                      Return
*
SERERR   MOVE.B  #TXENA,TSR       Enable transmitter
         BSET    #4,IMRA          Re enable interrupt
         BSET    #1,DSTAT         Set pwr up diag error
         RTS
*
XMIT     BTST    #7,TSR           Transmit ready?
         BEQ     XMIT             No - then wait
         MOVE.B  D0,UDR           Else output data
         RTS
*
RECV     CLR.W   SERST            Clear serial status
RECLP    MOVE.B  RSR,D1           Get receive status
         BTST    #7,D1            Receive buffer full?
         BEQ     RECLP            No - check again
         MOVE.B  UDR,D0           Else read data byte
FRME     BTST    #4,D1            Frame error?
         BEQ.S   PARE             No - check parity
         BSET    #0,SERST         Else set error status
PARE     BTST    #5,D1            Parity error?
         BEQ.S   OVRE             No - check overrun
         BSET    #1,SERST         Else set error status
OVRE     BTST    #6,D1            Overrun error?
         BEQ.S   RECRET           No - then return
         BSET    #2,SERST         Else set error status
RECRET   RTS
TSTPAT   DC.B    $0C
         DC.B    '     SYSTEM TEST'
         DC.B    $0D,$0A
         DC.B    '        RUNNING'
         END
```

```
**************************************************************************
*
*           TMRTST
*
*     This routine checks the functionality of the four filter clock
*     timers during power up diagnostics. The timers are set to the
*     prescale values that are established during power up but the
*     count values are set to 255. Then the count values are read from
*     the timers and a delay is initiated. After the delay times out,
*     the count values are read again. If the count has changed, then
*     the timers are said to be operational else an error status is
*     set in DSTAT.
*
*          Input parameters : None
*
*          Output parameters : DSTAT - power up diagnostic status
*
*          Registers used : A1 - index to the timer counters
*                           D0 - First timer reading
*                           D1 - second timer reading
*                           D2 - temporary counter
*                           D7 - delay counter
*
**************************************************************************
*
TMRTST    IDNT    1,1
          INCLUDE ADAMDEF
TADR      EQU     $80004F              Timer counter address
          XREF    TMRINIT
          XDEF    TMRTST
*
TMRTST    BSR     TMRINIT              Initialize timers to default
          LEA.L   TADR,A1              Get timer address
          MOVE.L  #$FFFFFFFF,(A1)      Set counters to 255
          BSR.S   TMRWAT               Delay
          MOVE.L  (A1),D0              Read timer counters
          BSR.S   TMRWAT               Delay
          MOVE.L  (A1),D1              Read timer counters again
          MOVE.B  #4,D2                Set counter number
TMCHK     CMP.B   D0,D1                Compare first & sec. reading
          BEQ.S   TMRERR               If equal then error
          LSR.L   #8,D0                Get next timer value
          LSR.L   #8,D1
          SUBI.W  #1,D2                Decr number of timers
          BNE.S   TMCHK                Check again
          BRA.S   TMREXT               Else exit
TMRERR    BSET    #2,DSTAT             Set timer error status
TMREXT    NOP
          BSR     TMRINIT              Re-initialize timers
          RTS                          Return
*
TMRWAT    MOVE.L  #$1000,D7            Delay
TMLP      NOP
          SUBI.L  #1,D7
          BNE     TMLP
          RTS
          END
```

```
*******************************************************************************:
*
*          ADTST
*
*     This routine performs a check of the four A/D's during power
*     up. The test is performed on mux channel 1. ADTST first sets
*     the A/D to RCAL mode and takes a reading. Then it sets the
*     A/D to non-RCAL mode and takes a reading. It then compares the
*     two readings and if they are the same value, DSTAT is
*     flagged with an A/D error. If the values differ, then the A/D
*     is O.K.
*
*          Input parameters : None
*
*          Output parameters : DSTAT - power up diagnostic status
*
*          Registers used : D1 - sample counter
*                           D2 - first reading
*                           D3 - second reading
*
*******************************************************************************
*
ADTST      IDNT    1,1
           INCLUDE ADAMDEF
           XDEF    ADTST
ADC        EQU     $800000                 A/D address
CNTRL      EQU     $800031                 Control port
RCAL       EQU     6                       RCAL bit
*
ADTST      MOVE.W #4,D1                    Set no. of samples to 4
           BSET    #RCAL,CNTRLM            Set to non-RCAL mode
           MOVE.B CNTRLM,CNTRL
           MOVE.L #3,ADC                   Set mux to channel 3
ADLP1      MOVE.L ADC,D2                   Read A/D
           MOVE.L #3,ADC                   Set mux to 3
           DBNE    D1,ADLP1                Go read again
           BCLR    #RCAL,CNTRLM            Set to RCAL mode
           MOVE.B CNTRLM,CNTRL
           MOVE.W #$8000,D1                Settle time delay
ADDEL      NOP
           MULU    D3,D3
           SUBI.W #1,D1
           BNE.S   ADDEL
           MOVE.W #4,D1                    Set no. of samples to 4
           MOVE.L #3,ADC                   Set mux to channel 3
ADLP2      MOVE.L ADC,D3                   Read A/D
           MOVE.L #3,ADC                   Set mux to 3
           DBNE    D1,ADLP2                Go read again
           BSET    #RCAL,CNTRLM            Set to non-RCAL mode
           MOVE.B CNTRLM,CNTRL
           MOVE.W #4,D1                    Load A/D counter
ADCHK      CMP.B   D2,D3                   Check each A/D seperatly
           BEQ.S   ADERR                   If equal then error
           LSR.L   #8,D2                   Shift in next A/D
           LSR.L   #8,D3
           DBNE    D1,ADCHK                Check it
ADRET      RTS                             Else return
ADERR      BSET    #3,DSTAT                Set A/D error status
           RTS
           END
```

```
*************************************************************************
*
*           RAMTST
*
*       This routine test the SRAM in the system during power up
*       diagnostic. It performs a byte write and read of memory with
*       the data patterns AA,55,FF, and 00. The memory that is tested
*       is from 1000000 through 107EFF0. This prevents the destruction
*       of system parameters and system stack. RAMTST uses the routine
*       WDTST to do the actual memory accesses.
*
*           Input parameters : D7 - test status returned by WDTST
*
*           Output parameters : DSTAT - test status for power up
*                                       diagnostics
*
*           Registers used : A0 - start of memory test
*                            A1 - end of memory test
*                            D0 - test data pattern
*                            D7 - test status from WDTST
*
*************************************************************************
*
RAMTST      IDNT    1,1
            INCLUDE ADAMDEF
            XREF    WDTST
            XDEF    RAMTST
STRAM       EQU     $1000000            Start of memory
ENRAM       EQU     $107EFF0            End of tested memory
*
RAMTST      CLR.L   D7                  Clear test status
            LEA.L   STRAM,A0            Load start of memory
            LEA.L   ENRAM,A1            Load end of memory
            MOVE.B  #$AA,D0             Load first test pattern
            BSR     WDTST               Call memory test rountine
            CMP.W   #0,D7               Error?
            BNE.S   RAMERR              Yes - set error bit
            MOVE.B  #$55,D0             Load next test pattern
            BSR     WDTST               Call memory test
            CMP.W   #0,D7               Error?
            BNE.S   RAMERR              Yes - go set error bit
            MOVE.B  #$FF,D0             Load next test pattern
            BSR     WDTST               Call memory test
            CMP.W   #0,D7               Error?
            BNE.S   RAMERR              Yes - go set error bit
            CLR.W   D0                  Load test pattern
            BSR     WDTST               Call memory test
            CMP.W   #0,D7               Error?
            BEQ.S   RAMEXT              No - return
RAMERR      BSET    #5,DSTAT            Else set error status
RAMEXT      RTS
            END
```

```
************************************************************************
*
*       GETKEY
*
*   GETKEY reads the MFP serial port to fetch the character that
*   was entered on the hand held terminal. This routine is called
*   by KEYINT when a keyboard interrupt is received. The ASCII
*   character read from the data port (UDR) is stored into KEY.
*   If the character is an alpha or numeric character (A-Z or 0-9),
*   it is stored into the input buffer indexed by register A6.
*   The character count in register D6 is also updated. If the
*   input is a control character (Scroll up, Scroll down, ENT,
*   ESC, Dot, Hyphen) it is just returned in KEY and not stored.
*   If it was the Delete key, GETKEY deletes the last character
*   in the key input buffer and spaces out the last character
*   on the display.
*
*       Input parameters : None
*       Output parameters : KEY - input character
*                           A6 - input character buffer
*                           D6 - character count
*
*       Registers used : A6, D6
*
************************************************************************
*
GETKEY    IDNT    1,1
          INCLUDE ADAMDEF
          XDEF    GETKEY
          XREF    CRLFO
RSR       EQU     $800055          UART receive status
TSR       EQU     $800056          UART transmit status
UDR       EQU     $800057          Data register
SCRLUP    EQU     $84              Scroll up code
SCRLDN    EQU     $83              Scroll down code
ENT       EQU     $0D              Enter code
ESC       EQU     $1B              Escape code
DOT       EQU     $2E              Dot code
HYFN      EQU     $2D              Hyphen code
DEL       EQU     08               Delete code
*
GETKEY    BTST    #7,RSR           Receive buffer full
          BEQ     GETKEY           No - wait
          MOVE.B  UDR,KEY          Read key entered
          CMPI.B  #$30,KEY         Numeric entry ?
          BLT.S   CHKCC
          CMPI.B  #$39,KEY
          BLE.S   INSCHR           Yes - go insert char. into buf.
          CMPI.B  #$41,KEY         Alpha character?
          BLT     GETRET
          CMPI.B  #$5A,KEY
          BGT.S   CHKCC            No - check for control characrter
```

```
INSCHR    CMPI.W #0,KBFLG          Keyboard test flag set ?
          BNE.S  INSCHR1           Yes - skip character storage
          CMPI.W #20,D6            If not valid character
          BGT    GETRET               return
          MOVE.B KEY,0(A6,D6.W)    Else store character in KEYBUF
          ADDI.W #1,D6             Incr character count
INSCHR1   BSR    KXMIT             Display the character entered
          BRA    GETRET            Return
CHKCC     CMPI.B #SCRLUP,KEY       If scroll up
          BNE.S  CHKCC1
          BSR.S  KXMIT              Send to terminal
          BRA    GETRET
CHKCC1    CMPI.B #SCRLDN,KEY       If scroll down
          BNE.S  CHKCC2
          BSR.S  KXMIT               Send to terminal
          BRA.S  GETRET
CHKCC2    CMPI.B #ENT,KEY          If enter key
          BNE.S  CHKCC3
          BRA.S  GETRET             Return
CHKCC3    CMPI.B #ESC,KEY          If escape key
          BNE.S  CHKCC4
          BRA.S  GETRET              Return
CHKCC4    CMPI.B #DOT,KEY          If "."
          BNE.S  CHKCC5
          BSR.S  KXMIT              Send to terminal
          BRA.S  GETRET
CHKCC5    CMPI.B #HYFN,KEY         If "-"
          BNE.S  CHKCC6
          BSR.S  KXMIT               Send to terminal
          BRA.S  GETRET
CHKCC6    CMPI.B #DEL,KEY          If delete key
          BNE.S  GETRET
          CMPI.W #0,D6             If no characters in KEYBUF
          BEQ.S  GETRET             Return
          SUBI.W #1,D6             Else decr character count
          BSR.S  KXMIT             Send DEL to terminal
          MOVE.B #$20,KEY          Space out character on the
          BSR.S  KXMIT                display
          MOVE.B #DEL,KEY
          BSR.S  KXMIT
          BRA.S  GETRET            Return

KXMIT     BTST   #7,TSR            Transmit ready?
          BEQ    KXMIT
          MOVE.B KEY,UDR           Output input character to display
          RTS

GETRET    RTS                      Return
          END
```

```
*
*
*****************************************************************
*
*       DSPMSG
*
*   DSPMSG displays the message on the hand held terminal thats
*   contained in the buffer pointed to by registers A2 for the
*   number of characters in D2. The data must already be in ASCII
*   format.
*
*       Input parameters : A2 - pointer to output buffer
*                          D2 - number of output characters
*
*       Output parameters : None
*       Registers used : D0 - temporary character count
*
*****************************************************************
*
DSPMSG    IDNT    1,1
          INCLUDE ADAMDEF
          XDEF    DSPMSG
TSR       EQU     $800056              UART transmit status
UDR       EQU     $800057              UART data register
*
DSPMSG    CLR.W   D0                   Reset DSPBUF index
DXMIT     BTST    #7,TSR               Transmit ready
          BEQ     DXMIT
          MOVE.B  0(A2,D0),UDR         Output character
          ADDI.W  #1,D0                Incr. DSPBUF index
          CMP.W   D2,D0                Check for end of message
          BLT     DXMIT
          CLR.W   D2
          RTS
*
          END
```

```
*********************************************************************
*
*          CRLFO
*
*     CRLFO outputs a carrage return and line feed character to the
*     serial port of the MFP. This controls the cursor position on
*     the hand held terminal.
*
*          Input parameters : None
*          Output parameters : None
*          Registers used : none
*
*********************************************************************
*
CRLFO      IDNT    1,1
           INCLUDE ADAMDEF
           XDEF    CRLFO
TSR        EQU     $800056             Transmit status register
UDR        EQU     $800057             UART data register
CARET      EQU     $0D                 Carrage return code
LINFED     EQU     $0A                 Line feed code
*
CRLFO      BTST    #7,TSR              Transmit ready?
           BEQ     CRLFO               No - wait
           MOVE.B #CARET,UDR           Output carrage return
CRLF1      BTST    #7,TSR              Transmit ready?
           BEQ     CRLF1               No - wait
           MOVE.B #LINFED,UDR          Output linefeed
           RTS                         Return
           END
```

```
*
*        EXTERNAL REFERENCES
*
*        DEFINE STORAGE PARAMETERS
*
         XREF    ROMPRM,PRMEND,PRMSG1,PRMCNT1,PRMSG2,PRMCNT2
         XREF    ROMER,SERER,FLCERR,PRLERR,ADER
         XREF    RAMER,ERRCNT,MENU1,MEN1CT,PREBUF,POSTBU
         XREF    STSCT,ENDSCT,UCNTRL,BAUD,PRSCA,PRSCB,PRSCCD
         XREF    FLCNTA,FLCNTB,FLCNTC,FLCNTD,SYNC,FCNTR,TSTST
         XREF    DSTAT,MU1SL,MU2SL,MU3SL,MU4SL,KEYBUF,PARCHK
         XREF    KEY,CALCNT,SERST,ENDROM,STATM,CNTRLM,JMPADR
         XREF    TXFLG,CLCTD,TELMTX,TELMWRD,KBFLG,DSPTST,DSPTPTR
         XREF    ADCHD1,ADCHD2,INVAL,ADCH,ALNCH,ALNCHD1,ALNCHD2
         XREF    TMPST,ENTMPST,TSERCTL,TMPSD,OTHCLK,PARNO,TMPPRE
         XREF    TMPCLK,DSPBUF,CCBUF1,CCBUF2,TMPCHR,TMPCVT
         XREF    MEMFAIL,JMPTBL,MU2SAD,MU2SCT,MU3SAD,MU3SCT
         XREF    BDTBL,MENU2,MEN2CT,MENU3,MEN3CT,MENU4,MEN4CT
         XREF    MENU5,MEN5CT,MENU6,MEN6CT,MENU7,MEN7CT,MENU8
         XREF    MEN8CT,MENU9,MEN9CT,ERRMSG,ERRMCT,ADPRMT,ADPRCT
         XREF    CLKTSTM,CLKTCT,TELTSTM,TELTCT,PARTSTM,PARTCT
         XREF    MEMMSG1,MEMCT1,MEMMSG2,MEMCT2,MEMMSG3,MEMCT3
         XREF    MEMMSG4,MEMCT4,MEMMSG5,MEMCT5,KBPRMT,KBCT
         XREF    FCLKMSG,FCLKCT,PARMSG,PARCT,STBMSG,STBCT,WRDLMSG
         XREF    WRDLCT,BAUDMSG,BAUDCT,CLRENT,CLRECT,MEMPAS,MEMPCT
         XREF    MEMERM,MEMERC,CCPASS,CCPCT,CCMSG,CCMCT,DSCTMSG
         XREF    DSCTCT,ALNHDR1,ALNCT1,ALNHDR2,ALNCT2,ADHDR,ADCT1
         XREF    ADHDR2,ADCT2,DSPTPAT,ENDSPT,TSTAFL,STPCNT
         XREF    PRE2K,CLK2K,PRE4K,CLK4K,PRE8K,CLK8K,PRE10K,CLK10K
         XREF    PRE16K,CLK16K,MEMERM1,MEMERM2,MEMERC1,MEMERC2
         XREF    DRACON,DRASCC,TDATMSG,TDATC,TDCOMP,TDCOMPC
         XREF    TXERM,TXERC,RCVERM,RCVERC,CONERM,CONERC
         XREF    RDATER,RDATC,CLCTDA,CLCTDC,PURPRMT,PURPRC
         XREF    TDATER,TDATRC,MEMFLM,MEMFLC,NDATMSG,NDATMCT
         XREF    DATFLM,DATFLC,PWRMU,PWRCT,SYSPAS,SYSPASC
         XREF    VERNUM,VERCNT,PRESYNC,DAQSYNC,POSSYNC
```

```
*
************************************************************************
*
*           TMRTST
*
*     This routine checks the functionality of the four filter clock
*     timers during power up diagnostics. The timers are set to the
*     prescale values that are established during power up but the
*     count values are set to 255. Then the count values are read from
*     the timers and a delay is initiated. After the delay times out,
*     the count values are.read again. If the count has changed, then
*     the timers are said to be operational else an error status is
*     set in DSTAT.
*
*         Input parameters : None
*
*         Output parameters : DSTAT - power up diagnostic status
*
*         Registers used : A1 - index to the timer counters
*                          D0 - First timer reading
*                          D1 - second timer reading
*                          D2 - temporary counter
*                          D7 - delay counter
*
************************************************************************
*
TMRTST    IDNT    1,1
          INCLUDE ADAMDEF
TADR      EQU     $80004F              Timer counter address
          XREF    TMRINIT
          XDEF    TMRTST
*
TMRTST    BSR     TMRINIT              Initialize timers to default
          LEA.L   TADR,A1              Get timer address
          MOVE.L  #$FFFFFFFF,(A1)      Set counters to 255
          BSR.S   TMRWAT               Delay
          MOVE.L  (A1),D0              Read timer counters
          BSR.S   TMRWAT               Delay
          MOVE.L  (A1),D1              Read timer counters again
          MOVE.B  #4,D2                Set counter number
TMCHK     CMP.B   D0,D1                Compare first & sec. reading
          BEQ.S   TMRERR               If equal then error
          LSR.L   #8,D0                Get next timer value
          LSR.L   #8,D1
          SUBI.W  #1,D2                Decr number of timers
          BNE.S   TMCHK                Check again
          BRA.S   TMREXT               Else exit
TMRERR    BSET    #2,DSTAT             Set timer error status
TMREXT    NOP
          BSR     TMRINIT              Re-initialize timers
          RTS                          Return
*
TMRWAT    MOVE.L  #$1000,D7            Delay
TMLP      NOP
          SUBI.L  #1,D7
          BNE     TMLP
          RTS
          END
```

```
*           SERINIT
*           INITIALIZE THE SERIAL PORT
*
SERINIT     IDNT    1,1
GPIP        EQU     $800040
DDR         EQU     $800042
IERA        EQU     $800043
IMRA        EQU     $800049
UCR         EQU     $800054
RSR         EQU     $800055
TSR         EQU     $800056
RSVEN       EQU     01
TXENA       EQU     $05
*
URTST       MOVE.L  #ISTACK,A7
            MOVE.W  #SWORD0,SR
            MOVE.L  #VBASE,A0
            MOVEC   A0,VBR
            MOVE.L  A7,ISP
            MOVE.L  #CEN,A0
            MOVEC   A0,CACR
            MOVE.B  #$FF,STAT
SERINIT     MOVE.W  #$1000,IERA
            MOVE.W  #$1000,IMRA
            MOVE.B  UCNTRL,UCR
            MOVE.B  #$0F,DDR
            MOVE.B  BAUD,GPIP
            MOVE.B  #RSVEN,RSR
            MOVE.B  #TXENA,TSR

            END
```

```
*          SERINIT
*          INITIALIZE THE SERIAL PORT
*
SERINIT    IDNT     1,1
GPIP       EQU      $800040
DDR        EQU      $800042
IERA       EQU      $800043
IMRA       EQU      $800049
UCR        EQU      $800054
RSR        EQU      $800055
TSR        EQU      $800056
RSVEN      EQU      01
TXENA      EQU      $05
*
URTST      MOVE.L  #ISTACK,A7
           MOVE.W  #SWORD0,SR
           MOVE.L  #VBASE,A0
           MOVEC   A0,VBR
           MOVE.L  A7,ISP
           MOVE.L  #CEN,A0
           MOVEC   A0,CACR
           MOVE.B  #$FF,STAT
SERINIT    MOVE.W  #$1000,IERA
           MOVE.W  #$1000,IMRA
           MOVE.B  UCNTRL,UCR
           MOVE.B  #$0F,DDR
           MOVE.B  BAUD,GPIP
           MOVE.B  #RSVEN,RSR
           MOVE.B  #TXENA,TSR

           END
```

```
*********************************************************************
*
*           MENUPR
*
*         MENUE PROCESSOR
*
*    This routine processes the menu entries entered on the hand
*    held terminal. It is called by KEYINT when a delimitor
*    (ENT,F4,.) is entered. MENUPR determines where to go to process
*    the entry by the values contained in the menu level variables
*    (MU1SL, MU2SL, MU3SL,and MU4SL). The delimitor entry is fetched
*    from the variable KEY and the parametor entry is read from
*    KEYBUF which is indexed by register A6. Register D6 contains
*    the number of characters entered. When a selection requires a
*    new menu to be displayed, MENUPR calls DSPMSG to dispplay it
*    and then updates the menu level variables so that the next
*    entry will be processed by the appropriate routine. When
*    an ESC key is detected, LASTMU is executed, which looks at
*    the menu level variables to determine the previous menu
*    for updating the display.
*
*         Input parameters : KEY - last key entered
*                            KEYBUF - characters entered less the
*                                     delimiter
*                            MU1SL - value of the level 1 menu
*                                    (always equal 1 except during
*                                     power up when it equals 0)
*                            MU2SL - value of the level 2 menu
*                            MU3SL - value of the level 3 menu
*                            MU4SL - value of the level 4 menu
*                                    (either equals 0 or 1)
*
*         Output parameters : TXFLG - flag for transmitting data to
*                                     the DRASS
*                             CLCTD - flag to trigger data collection
*                             JMPADR - jump table value for the main
*                                      loop execution.
*
*********************************************************************
*
MENUPR      IDNT     1,1
            INCLUDE  ADAMDEF
            XREF     DSPMSG,CHCHECK,PATTST,ADRTST,BU0TST,BU1TST
            XREF     TSTALL,CVTHEX,DSPSCT,SERINIT,CVTDEC,CRLFO,WDTST
            XDEF     MENUPR
*
CNTRL       EQU      $800031               Control port address
GPIP        EQU      $800040               GPIP address on the MFP
TACR        EQU      $80004C               Timer A prescale addr.
TBCR        EQU      $80004D               Timer B prescale addr
TCDCR       EQU      $80004E               Timers C & D prescale addr.
TADR        EQU      $80004F               Timer A counter addr.
TBDR        EQU      $800050               Timer B counter addr.
TCDR        EQU      $800051               Timer C counter addr.
TDDR        EQU      $800052               Timer D counter addr.
```

```
UCR         EQU    $800354            UART control addr.
STRAM       EQU    $1000000           Start of memory
ENRAM       EQU    $107EFFE           End of memory
ENT         EQU    $0D                Enter key
ESC         EQU    $1B                ESC key
DOT         EQU    $2E                "." key
*
*        Process the main menu selection
*
MENUPR      CMPI.W #0,MU1SL           Level 1 menu set?
            BEQ    MURET1             No - return
            CMPI.W #0,MU2SL           Level 2 menu set?
            BNE    CHKMU2             Yes - go to level 2 processor
            CMPI.W #1,D6              More than 1 key entered?
            BGT    MUERR              Yes - display error
            CMPI.B #$31,(A6)          Menu selection = 1
            BNE.S  MU1S2              No - check for 2
            LEA.L  MENU2,A2           Else display diagnostic menu
            MOVE.W MEN2CT,D2
            BSR    DSPMSG
            MOVE.W #1,MU2SL           Set level 2 to diag
            CLR.W  D6                 Clear key entry
            BRA    MURET              Return
*
* Process calibration selection
*
MU1S2       CMPI.B #$32,(A6)          Selection = 2
            BNE.S  MU1S3              No - check for 3
            LEA.L  MENU3,A2           Display calibration menu
            MOVE.W MEN3CT,D2
            BSR    DSPMSG
            MOVE.W #2,MU2SL           Set to cal processing
            CLR.W  D6
            BRA    MURET
*
*      Process PAR.SET selection
*
MU1S3       CMPI.B #$33,(A6)          Selection = 3
            BNE.S  MU1S4              No - check for 4
            LEA.L  MENU4,A2           Display cal. menu
            MOVE.W MEN4CT,D2
            BSR    DSPMSG
            CLR.W  D6
            MOVE.W #3,MU2SL           Sey to cal processing
            BRA    MURET
*
```

```
*       Process TXDATA selection
*
MU1S4       CMPI.B  #$34,(A6)          Selection = 4
            BNE.S   MU1S5              No - go check for 5
            MOVE.W  $$FF,TXFLG         Else set transmit data flag
            LEA.L   CLRENT,A2          Clear display
            MOVE.W  CLRECT,D2
            BSR     DSPMSG
            CLR.W   D6
            BRA     MURET
*
*       Process the DATA. COL. selection
*
MU1S5       CMPI.B  #$35,(A6)          Selection = 5
            BNE.S   MU1S6              No - go check for 6
            MOVE.W  #$FF,CLCTD         Else set the collect data flag
            LEA.L   CLRENT,A2          Clear display
            MOVE.W  CLRECT,D2
            BSR     DSPMSG
            CLR.W   D6
            BRA     MURET
*
*       Process the PURGE selection
*
MU1S6       CMPI.B  #$36,(A6)          Selection = 6?
            BNE     MUESC              No - check for ESC key
            LEA.L   PURPRMT,A2         Else display pruge msg
            MOVE.W  PURPRC,D2
            BSR     DSPMSG
            MOVE.W  #4,MU2SL           Set to the purge processor
            CLR.W   D6
            BRA     MURET
*
*       Process the level 2 menu selections
*       This includes : Diagnostic Menu, Cal Menu, and Parameter
*                       Set Menu
*
CHKMU2      CMPI.W  #0,MU3SL           Level 3 menu set?
            BNE     CHKMU3             Yes - go process level 3
            CMPI.W  #1,D6              More than 1 key entered?
            BGT     MUERR              Yes - display error
*
*      Process diagnostic level 2
*
            CMPI.W  #1,MU2SL           Diag. process?
            BNE     CHKMU22            No - go check next
*
*    Process memory diagnostics
*
            CMPI.B  #$31,(A6)          Selection = 1
            BNE.S   MU2S12             No - go check for  2
            BTST    #3,TSTST           Memory full?
            BEQ.S   MU2CON             No - continue test
            LEA.L   DATFLM,A2          Else display memory full msg.
            MOVE.W  DATFLC,D2
            BSR     DSPMSG
            CLR.W   D6
            BRA     MURET
```

```
MU2CON   LEA.L   MENU5,A2            Display memory diag menu
         MOVE.W  MEN5CT,D2
         BSR     DSPMSG
         CLR.W   D6
         MOVE.W  #1,MU3SL            Set to memory diag
         BRA     MURET
*
*    Process the serial diagnostic menu selection
*
MU2S12   CMPI.B  #$32,(A6)           Selection = 2?
         BNE.S   MU2S13              No - go check for 3
         LEA.L   MENU6,A2            Else display serial diag menu
         MOVE.W  MEN6CT,D2
         BSR     DSPMSG
         MOVE.W  #2,MU3SL            Set to serial diag processor
         CLR.W   D6
         MOVE.W  D6,KBFLG            Clear keyboard test flag
         MOVE.W  D6,DSPTST           Clear display test flag
         BRA     MURET
*
*    Process the A/D diagnostic selection
*
MU2S13   CMPI.B  #$33,(A6)           Selection = 3?
         BNE.S   MU2S14              No - check 4
         LEA.L   ADPRMT,A2           Else display A/D prompt
         MOVE.W  ADPRCT,D2
         BSR     DSPMSG
         MOVE.W  #3,MU3SL            Set to A/D diag. processor
         CLR.W   D6
         BRA     MURET
*
*    Process the Clock test selection
*
MU2S14   CMPI.B  #$34,(A6)           Selection = 4?
         BNE.S   MU2S15              No - go check 5
         LEA.L   CLKTSTM,A2          Else display clock test menu
         MOVE.W  CLKTCT,D2
         BSR     DSPMSG
         MOVE.W  #4,MU3SL            Set to clock processor
         MOVE.W  #2,JMPADR           Set to CLKTST routine
         CLR.W   D6
         BRA     MURET
*
*    Process the Telemetry test selection
*
MU2S15   CMPI.B  #$35,(A6)           Selection = 5?
         BNE.S   MU2S16              No - check 6
         LEA.L   TELTSTM,A2          Else display Clck test msg
         MOVE.W  TELTCT,D2
         BSR     DSPMSG
         MOVE.W  #5,MU3SL            Set to clck test processor
         MOVE.W  #33,TELMTX          Init. frame size
         MOVE.L  #$00010203,TELMWRD   Init test pattern
         MOVE.W  #3,JMPADR           Set to TELMTST
         CLR.W   D6
         BRA     MURET
```

```
*
*     Process the Parallel test selection
*
MU2S16     CMPI.B #$36,(A6)          Selection = 6
           BNE    MUESC              No - go check ESC key
           LEA.L  PARTSTM,A2         Else display parallel test msg
           MOVE.W PARTCT,D2
           BSR    DSPMSG
           MOVE.W #6,MU3SL           Set to parallel test processor
           MOVE.W #4,JMPADR          Set to PARLTST routine
           CLR.W  D6
           BRA    MURET
*
*     Process calibration menu
*
CHKMU22    CMPI.W #2,MU2SL           Calibration selection?
           BNE.S  CHKMU23            No - check Parameter set
*
*    Process channel check selection
*
           CMPI.B #$31,(A6)          Selection = 1?
           BNE.S  MU2S22             No - check 2
           CLR.W  D6
           MOVE.W #7,MU3SL           Set to channel check
           BSR    CHCHECK            Call channel check routine
           BRA    MURET
*
*    Process the Align selection
*
MU2S22     CMPI.B #$32,(A6)          Selection = 2?
           BNE    MUESC              No - check for ESC
           LEA.L  ADPRMT,A2          Display align ch. prompt
           MOVE.W ADPRCT,D2
           BSR    DSPMSG
           CLR.W  D6
           MOVE.W #8,MU3SL           Set to Align processor
           BRA    MURET
*
*    Process parameter set selection
*
CHKMU23    CMPI.W #3,MU2SL           Param. set level
           BNE    CHKMU24            No - go check purge selection
*
*    Process channel specification selection
*
           CMPI.B #$31,(A6)          Selection = 1?
           BNE.S  MU2S32             No - go check 2
           LEA.L  MENU7,A2           Else display channel spec. menu
           MOVE.W MEN7CT,D2
           BSR    DSPMSG
           CLR.W  D6
           MOVE.W #9,MU3SL           Set to ch. spec. processor
           BRA    MURET
*
```

```
*    Process clock rate selection
*
MU2S32      CMPI.B #$32,(A6)          Selection = 2?
            BNE.S  MU2S33             No - go check for 3
            LEA.L  MENU8,A2           Else display clock rate menu
            MOVE.W MENBCT,D2
            BSR    DSPMSG
            CLR.W  D6
            MOVE.W #10,MU3SL          Set to clock rate process
            BRA    MURET
*
*    Process the serial definition selection
*
MU2S33      CMPI.B #$33,(A6)          Selection = 3?
            BNE.S  MU2S34             No - check for 4
            LEA.L  PARMSG,A2          Else display Parity prompt
            MOVE.W PARCT,D2
            BSR    DSPMSG
            CLR.W  D6
            MOVE.W #11,MU3SL          Set to parity processor
            BRA    MURET
*
*    Power control selection
*
MU2S34      CMPI.B #$34,(A6)          Selection = 4
            BNE    MUESC              No - go check for ESC key
            LEA.L  PWRMU,A2           Display power menu
            MOVE.W PWRCT,D2
            BSR    DSPMSG
            CLR.W  D6
            MOVE.W #20,MU3SL          Set to power processor
            BRA    MURET
*
*    Purge selection processor
*
CHKMU24     CMPI.W #4,MU2SL           Purge level?
            BNE    MUESC              No - exit
            CMPI.B #$59,(A6)          Selection = "Y" ?
            BNE    LASTMU             No - go display last menu
            LEA.L  STRAM,A0           Else erase the data RAM
            LEA.L  ENRAM,A1
            CLR.W  D0
            BSR    WDTST
            MOVE.L #0,PRESYNC         Clear the pre cal sync code
            MOVE.L #0,DAQSYNC         Clear the test data sync code
            MOVE.L #0,POSSYNC         Clear the post cal sync code
            BCLR   #3,TSTST           Set memory empty
            BCLR   #7,TSTST           Clear data storage mode
            BCLR   #2,TSTST           Clear post cal collect stat
            BCLR   #1,TSTST           Clear test data collect stat
            BRA    LASTMU             Go display last menu
```

```
*
*    Memory diagnostic selection processor
*
CHKMU3     CMPI.W #1,MU3SL              Memory test selected
           BNE    CHMU32                No check next keyboard test
           CMPI.W #1,D6
           BGT    MUERR
*
*    Pattern test processor
*
           CMPI.B #$31,(A6)             Selection = 1?
           BNE.S  MU3S12                No - check for 2
           LEA.L  MEMMSG1,A2            Display pattern test msg
           MOVE.W MEMCT1,D2
           BSR    DSPMSG
           CLR.W  D6
           MOVE.W D6,TSTAFL             Clear test all flag
           MOVE.W #1,MU4SL              Set level 4 menu
           BSR    PATTST                Call pattern test
           BRA    MURET
*
*    Perform Address memory test
*
MU3S12     CMPI.B #$32,(A6)             Selection = 3?
           BNE.S  MU3S13                No - check 4
           LEA.L  MEMMSG2,A2            Else display address test msg
           MOVE.W MEMCT2,D2
           BSR    DSPMSG
           CLR.W  D6
           MOVE.W D6,TSTAFL             Clear test all flag
           MOVE.W #1,MU4SL              Set level 4 menu
           BSR    ADRTST                Call Address test
           BRA    MURET
*
*     Perform Bubble 0 test
*
MU3S13     CMPI.B #$33,(A6)             Selection = 3
           BNE.S  MU3S14                No - check 4
           LEA.L  MEMMSG3,A2            Else display Bubble 0 test msg
           MOVE.W MEMCT3,D2
           BSR    DSPMSG
           CLR.W  D6
           MOVE.W D6,TSTAFL             Clear test all flag
           MOVE.W #1,MU4SL              Set level 4 menu
           BSR    BU0TST                Call Bubble 0 test
           BRA    MURET
*
*     Perform Bubble 1 test
```

```
*
MU3S14    CMPI.B  #$34,(A6)       Selection = 4
          BNE.S   MU3S15          No - check for 5
          LEA.L   MEMMSG4,A2      Display Bubble 1 test msg
          MOVE.W  MEMCT4,D2
          BSR     DSPMSG
          CLR.W   D6
          MOVE.W  D6,TSTAFL       Clear test all flag
          MOVE.W  #1,MU4SL        Set level 4 menu
          BSR     BU1TST          Call Bubble 1 test
          BRA     MURET
*
*    Perform test all
*
MU3S15    CMPI.B  #$35,(A6)       Selection = 5?
          BNE     MUESC           No - process ESC
          LEA.L   MEMMSG5,A2      Else display test all msg
          MOVE.W  MEMCT5,D2
          BSR     DSPMSG
          CLR.W   D6
          MOVE.W  #$FF,TSTAFL     Set test all flag
          MOVE.W  #1,MU4SL        Set level 4
          BSR     TSTALL          Call Test all
          BRA     MURET
*
*    Perfor serial diagnostic selections
*
CHMU32    CMPI.W  #2,MU3SL        Serial diag selected?
          BNE     CHMU33          No - check A/D diag
          CMPI.W  #0,KBFLG        Keyboard test flag set?
          BEQ.S   TST322          No - check for display test
          CLR.W   D6
          CMPI.B  #ENT,KEY        Else ENT key entered
          BNE.S   CONCH32         No - continue
          BSR     CRLFO           Else output car.ret.lin.feed
          BRA.S   CON322
CONCH32   CMPI.B  #ESC,KEY        ESC key entered?
          BNE.S   CON322          No - continue
          MOVE.W  #0,KBFLG        Else clear keyboard test flag
          BRA     MUESC           Process ESC key
*
*    Perform display test
*
TST322    CMPI.W  #0,DSPTST       Display test active?
          BEQ.S   CON322          No - continue
          CMPI.B  #ESC,KEY        ESC key entered
          BNE     MURET           No - return
          MOVE.W  #30,JMPADR      Else set to display last menu
          CLR.W   D6
          MOVE.W  D6,DSPTST       Clear display test flag
          BRA     MURET
CON322    CMPI.W  #0,KBFLG        Keyboard test active?
```

```
         BNE     MURET                 Yes - return
         CMPI.W  #1,D6
         BGT     MUERR
         CMPI.B  #$31,(A6)             Selection = 1?
         BNE.S   MU3S22                No - check for 2
         MOVE.W  #$FF,DSPTST           Set display test active
         MOVE.W  #1,MU4SL              Set level 4 menu
         MOVE.W  #5,JMPADR             Activate DSPTST
         MOVE.L  #DSPTPAT,DSPTPTR      Init display test pattern pntr.
         CLR.W   D6
         BRA     MURET
MU3S22   CMPI.B  #$32,(A6)             Selection = 2?
         BNE     MUESC                 No - go process ESC
         LEA.L   KBPRMT,A2             Display keyboard prompt
         MOVE.W  KBCT,D2
         BSR     DSPMSG
         MOVE.W  #$FF,KBFLG            Set keyboard test flag
         MOVE.W  #1,MU4SL              Set level 4 menu
         CLR.W   D6
         BRA     MURET
*
*    A/D diagnostic processor
*
CHMU33   CMPI.W  #3,MU3SL              A/D diag active?
         BNE     CHMU34                No - check next test
         CMPI.B  #ESC,KEY              F4 key entered?
         BNE.S   CHK3IN                No - check A/D diag input
         CLR.W   D6
         MOVE.W  #10,JMPADR            Set to display last menu
         BRA     MURET
CHK3IN   CMPI.W  #2,D6                 More than 2 keys entered?
         BGT     MUERR                 Yes - display error
         MOVE.B  #$20,ADCHD1           Space out channel no.
         MOVE.B  #$20,ADCHD2
         MOVE.B  (A6),ADCHD1           get A/D channel no. entered
         CMPI.W  #2,D6
         BNE.S   CONT33
         MOVE.B  1(A6),ADCHD2          Convert channel to HEX
CONT33   BSR     CVTHEX
         CMPI.W  #0,INVAL              Invalid entry?
         BLT     MUERR                 Yes - display error
         CMPI.W  #31,INVAL
         BGT     MUERR
         MOVE.W  INVAL,D1              Store channel no. for A/D routine
         MOVE.L  #0,ADCH
         MOVE.B  D1,ADCH+3
         MOVE.W  #1,JMPADR             Set to A/D diag.
         LEA.L   ADHDR,A2              Display A/D msg
         MOVE.W  ADCT1,D2
         BSR     DSPMSG
         LEA.L   ADCHD1,A2
```

```
        MOVE.W #2,D2
        BSR    DSPMSG
        LEA.L  ADHDR2,A2
        MOVE.W ADCT2,D2
        BSR    DSPMSG
        CLR.W  D6
        BRA    MURET
*
*    Process the exiting of the Clock test, Telemetry test,
*    and Parellel test
*
CHMU34  CMPI.W #4,MU3SL         Check for any of those tests
        BEQ.S  EX456
        CMPI.W #5,MU3SL
        BEQ    MURET
        CMPI.W #6,MU3SL
        BNE.S  CHMU37           If none then check for Ch check
EX456   CLR.W  D6
        CMPI.B #ESC,KEY         If not ESC return
        BNE    MURET
        MOVE.W #10,JMPADR       Else set for display last menu
        BRA    MURET
*
*    Process the exiting of the Align routine
*
CHMU37  CMPI.W #7,MU3SL         Ch.check running?
        BNE.S  CHMU38           No - check for Align sel.
        CLR.W  D6
        CMPI.B #ESC,KEY         ESC entered?
        BEQ    MUESC            Yes - go exit test
        BRA    MURET            Else return
*
*    Process the exiting of Align
*
CHMU38  CMPI.W #8,MU3SL         Align test running?
        BNE    CHMU39           No - go check Ch. spec.
        CMPI.B #ESC,KEY         ESC entered?
        BNE.S  CHK38IN          No - go check for ch.no. entered
        CLR.W  D6
        MOVE.W #10,JMPADR       Set for display last menu
        BRA    MURET
CHK38IN CMPI.W #2,D6            Too many characters entered?
        BGT    MUERR            Yes - display error
        MOVE.B #$20,ALNCHD1     Init. display msg. with spaces
        MOVE.B #$20,ALNCHD2
        MOVE.B (A6),ALNCHD1
        CMPI.W #2,D6
        BNE.S  CONT38
        MOVE.B 1(A6),ALNCHD2
CONT38  BSR    CVTHEX           Convert channels entered to hex
        CMPI.W #0,INVAL         Invalid entry?
        BLT    MUERR            Yes - display error
        CMPI.W #31,INVAL
        BGT    MUERR
```

```
         MOVE.W INVAL,D1          Store mux channel for test
         MOVE.L #0,ALNCH
         MOVE.B D1,ALNCH+3
         LEA.L  ALNHDR1,A2
         MOVE.W ALNCT1,D2         Display Align test msg.
         BSR    DSPMSG
         LEA.L  ALNCHD1,A2
         MOVE.W #2,D2
         BSR    DSPMSG
         LEA.L  ALNHDR2,A2
         MOVE.W ALNCT2,D2
         BSR    DSPMSG
         MOVE.W #6,JMPADR         Set for align test
         BRA    MURET
*
*    Process the channel specification selection
*
CHMU39   CMPI.W #9,MU3SL          Ch. spec menu operation?
         BNE    CHMU310           No - go check freq. setting
         CMPI.W #1,D6             Too many characters entered?
         BGT    MUERR             Yes - display error
         CMPI.B #$31,(A6)         Selection = 1?
         BNE.S  MU3S92            No - go check sel. ch.
*
*    Perform sequence all
*
         LEA.L  STSCT,A0          Else do sequence all
         CLR.L  D0                Init scan table to 0-32 hex
LOOP91   MOVE.B D0,(A0)+
         ADDI.W #1,D0
         CMPI.W #33,D0
         BLT.S  LOOP91
         SUBQ.L #1,A0
         MOVE.L A0,ENDSCT         Save end of scan table
         CLR.W  D6
         MOVE.W #1,MU4SL          Go display last menu
         BRA    LASTMU
*
*    Process channel selection
*
MU3S92   CMPI.B #$32,(A6)         Channel selection process
         BNE.S  MU3S93            No - go process display channels
         LEA.L  TMPST,A2          Store ch. nos. in temporary array
         MOVE.L A2,ENTMPST
         LEA.L  ADPRMT,A2         Display prompt
         MOVE.W ADPRCT,D2
         BSR    DSPMSG
         CLR.W  D6
         MOVE.W #15,MU3SL         Set to get channel numbers
         BRA    MURET
*
```

```
*     Process display channels menu selection
*
MU3S93      CMPI.B #$33,(A6)           Display ch. selected
            BNE    MUESC               No - return
            BSR    DSPSCT              Call display scan table
            MOVE.W #1,MU4SL            Set for display last menu
            BRA    MURET
*
*     Process the clock frequency menu for clock A
*
CHMU310     CMPI.W #10,MU3SL           Clock setting process?
            BNE    CHMU311             No - go check parity process
            CMPI.W #1,D6               Too many characters entered?
            BGT    MUERR               Yes - display error
            LEA.L  MENU9,A2            Set for freq. menu
            MOVE.W MEN9CT,D2
            CMPI.B #$31,(A6)           Clock A selected?
            BNE.S  MU3S102             No - go check clock B
            BSR    DSPMSG              Display fraq. menu
            CLR.W  D6
            MOVE.W #16,MU3SL           Set to process freq select
            MOVE.W #1,MU4SL
            BRA    MURET
*
*     Process the clock frequency menu for clock B
*
MU3S102     CMPI.B #$32,(A6)
            BNE.S  MU3S103
            BSR    DSPMSG
            CLR.W  D6
            MOVE.W #17,MU3SL
            MOVE.W #1,MU4SL
            BRA    MURET
*
*     Process the clock frequency menu for clock C
*
MU3S103     CMPI.B #$33,(A6)
            BNE.S  MU3S104
            BSR    DSPMSG
            CLR.W  D6
            MOVE.W #18,MU3SL
            MOVE.W #1,MU4SL
            BRA    MURET
*
*     Process the clock frequency menu for clock D
*
MU3S104     CMPI.B #$34,(A6)
            BNE    MUESC
            BSR    DSPMSG
            CLR.W  D6
            MOVE.W #19,MU3SL
            MOVE.W #1,MU4SL
            BRA    MURET
```

```
*
*    Process the parity selection for the UART in the MFP
*
CHMU311     CMPI.W #11,MU3SL           Check for parity processing
            BNE.S  CHMU312
            CMPI.W #1,D6
            BGT    MUERR
            CLR.W  D6
            MOVE.B D6,TSERCTL          Clear UART control byte
            CMPI.B #$45,(A6)           Parity = E ?
            BNE.S  MU3SO               No - next check
            BSET   #1,TSERCTL          Set parity even
            BSET   #2,TSERCTL
            BRA.S  CONT311
MU3SO       CMPI.B #$4F,(A6)           Parity = O ?
            BNE.S  MU3SN               No - next check
            BSET   #2,TSERCTL          Set odd parity
            BRA.S  CONT311
MU3SN       CMPI.B #$4E,(A6)           Parity = N?
            BNE.S  ESC311              No - return
CONT311     LEA.L  STBMSG,A2           Display stop bit prompt
            MOVE.W STBCT,D2
            BSR    DSPMSG
            MOVE.W #12,MU3SL           Set for stop bit processing
            BRA    MURET
ESC311      CMPI.B #ESC,KEY
            BNE    MUERR
            BRA    MUESC
*
*    Process the stop bit selection
*
CHMU312     CMPI.W #12,MU3SL           Stop bit processing?
            BNE.S  CHMU313
            CMPI.W #1,D6               More than 1 character entered?
            BGT    MUERR               Yes - display error
            CMPI.B #$31,(A6)           Stop bit = 1?
            BNE.S  MU3S122             No - check 2
            BSET   #3,TSERCTL          Set 1 stop bit
            CLR.W  D6
            LEA.L  WRDLMSG,A2          Display word length msg.
            MOVE.W WRDLCT,D2
            BSR    DSPMSG
            MOVE.W #13,MU3SL           Set for word length processing
            BRA    MURET
MU3S122     CMPI.B #$32,(A6)           2 stop bits selected?
            BNE    MUESC               No - error
            BSET   #3,TSERCTL          Set 2 stop bits
            BSET   #4,TSERCTL
            CLR.W  D6
            LEA.L  WRDLMSG,A2          Display wordlength prompt
            MOVE.W WRDLCT,D2
            BSR    DSPMSG
            MOVE.W #13,MU3SL           Set for word length processing
            BRA    MURET
*
```

```
*       Process the word length menu selection
*
CHMU313     CMPI.W #13,MU3SL             Word length processing?
            BNE.S  CHMU314               No - check baud rate
            CMPI.W #1,D6                 More than 1 character entered?
            BGT    MUERR                 Yes - error
            LEA.L  BAUDMSG,A2            Set up baud rate msg
            MOVE.W BAUDCT,D2
            CMPI.B #$35,(A6)             Word length = 5 ?
            BNE.S  MU3136                No - check 6
            BSET   #5,TSERCTL            Set word length at 5
            BSET   #6,TSERCTL
            BRA.S  CONT313
MU3136      CMPI.B #$36,(A6)             Word length = 6?
            BNE.S  MU3137                No - check 7
            BSET   #6,TSERCTL            Set word length at 6
            BRA.S  CONT313
MU3137      CMPI.B #$37,(A6)             Word length = 7 ?
            BNE.S  MU3138                No check 8
            BSET   #5,TSERCTL            Set word length at 7
            BRA.S  CONT313
MU3138      CMPI.B #$38,(A6)             Word length = 8?
            BNE    MUESC                 No - reurn
CONT313     CLR.W  D6
            MOVE.W #14,MU3SL             Set for baud rate selection
            BSR    DSPMSG                Display baud rate prompt
            BRA    MURET
*
*       Process Baud rate selection
*
CHMU314     CMPI.W #14,MU3SL             Baud rate process?
            BNE.S  CHMU315               No - che ch. selection
            CMPI.W #0,D6                 Invalid number of characters entered?
            BEQ    MUESC                 Yes - return
            CMPI.W #4,D6
            BGT    MUERR
            CLR.L  D0
            LEA.L  TMPBD,A1              Store input to temp. buffer
            MOVE.L D0,TMPBD
TRBDLP      MOVE.B (A6,D0),(A1,D0)
            ADDI.W #1,D0
            CMP.W  D0,D6
            BEQ.S  TBLCHK
            BRA.S  TRBDLP
TBLCHK      MOVE.W #$3C,D0               Set baud rate table size
            LEA.L  BDTBL,A0              Load baud rate table
            MOVE.L TMPBD,D1              Store inputed baud rate
BUDSRCH     CMP.L  (A0,D0),D1            Search for match
            BEQ.S  BAUDFND
            SUBI.W #4,D0
            BPL.S  BUDSRCH
            CLR.W  D6
```

```
         BRA      MUERR                       No match - error
BAUDFND  LSR.W    #2,D0                       Divide index by 4
         MOVE.B   D0,BAUD                     Store into baud select
         BSET     #7,TSERCTL
         MOVE.B   TSERCTL,UCNTRL              Store new UART control byte
         BSR      SERINIT                     Init serial port
         BRA      LASTMU                      Go display last menu
*
*     Process the entry of mux channel numbers for
*     the select channel menu
*
CHMU315  CMPI.W   #15,MU3SL                   Process select channels?
         BNE      CHMU316                     No - go check oth clock process
         CMPI.B   #$2E,KEY                    "." entered?
         BEQ.S    CONT315                     Yes - go process entry
         CMPI.B   #ESC,KEY                    ESC entered?
         BNE.S    ENT315                      No - check for ENT
         CMPI.W   #0,D6                       Any characters in input buffer?
         BNE.S    CONT315                     Yes - go process input
         BRA.S    TRNST                       Else go build scan table
ENT315   CMPI.B   #ENT,KEY                    ENT entered?
         BNE      MUERR                       No - display error
         BSR      CRLFO                       Else outplut carrage ret. linefeed
         CMPI.W   #0,D6
         BEQ      MURET                       Return
CONT315  BSR      CVTHEX                      Convert input to hex
         CLR.W    D6
         CMPI.W   #0,INVAL                    Input < 0
         BLT      MUERR
         CMPI.W   #31,INVAL                   Or > 31
         BGT      MUERR                       Display error
         MOVE.L   ENTMPST,A0                  Get last pntr to temp scan table
         CLR.L    D1
         MOVE.W   INVAL,D1                    Store entered mux channel
         MOVE.B   D1,(A0)+
         MOVE.L   A0,ENTMPST                  Restore tmp scan table pointer
         CMPI.B   #ESC,KEY                    Was ESC entered
         BEQ.S    TRNST                       Yes - go build scan table
         BRA      MURET                       Else return
TRNST    LEA.L    STSCT,A0                    Transfer temp scan table to
         LEA.L    TMPST,A1                    real scan table
TRNLP    MOVE.B   (A1)+,(A0)+
         CMPA.L   ENTMPST,A1
         BLE.S    TRNLP
         SUBQ.W   #1,A0
         MOVE.L   A0,ENDSCT
         CLR.W    D6
         MOVE.W   #9,MU3SL
         MOVE.W   #1,MU4SL
         BRA      LASTMU                      Go display last menu
*
*     Process Other Clock menu selection
*
```

```
CHMU316   CMPI.W #19,MU3SL       Oth. clock process?
          BGT    CHMU320         No - go check power menu process
          CMPI.W #0,OTHCLK       No selection made?
          BEQ    ESC316          Yes - go check ESC
          CMPI.B #ESC,KEY        ESC entered anyway?
          BNE.S  PAR1CHK         No - go process 1st param.
          MOVE.W #10,MU3SL       Else go display last menu
          MOVE.W #0,OTHCLK
          MOVE.W #0,PARNO
          BRA    MUESC
PAR1CHK   CMPI.W #1,PARNO        Parameter = 1?
          BNE.S  PAR2CHK         No - go check param. 2
          CMPI.B #DOT,KEY        Input = "."?
          BNE    MUERR           No - display error
          MOVE.B (A6),D0         Get input key
          ANDI.B #$07,D0         Convert to decimal
          MOVE.B D0,TMPPRE       Save in temp variable
          MOVE.W #2,PARNO        Set to parameter 2
          CLR.W  D6
          BRA    MURET           Return
PAR2CHK   CMPI.W #2,PARNO        Parameter = 2?
          BNE    MUERR           No - display error
          CMPI.B #ENT,KEY        ENT entered?
          BNE    MUERR           No - display error
          BSR    CVTHEX          Else convert input to hex
          CMPI.W #0,INVAL        Valid input?
          BLT    MUERR           No display error
          CMPI.W #$FF,INVAL
          BGT    MUERR
          MOVE.W INVAL,D1
          MOVE.B D1,TMPCLK       Save input in temp variable
          CLR.W  D6
          MOVE.W D6,OTHCLK       Reset control valiables
          MOVE.W D6,PARNO
          BRA    SETCLK          Initialize clock
ESC316    CMPI.B #ESC,KEY        Process ESC key
          BNE.S  MU3S161
          CLR.W  D6
          MOVE.W D6,OTHCLK       Reset control variables
          MOVE.W D6,PARNO
          MOVE.W #10,MU3SL       Set to display last menu
          BRA    MUESC
*
*   Process 2KHz menu selection
*
MU3S161   CMPI.W #1,D6
          BGT    MUERR
          CMPI.B #$31,(A6)       Key = 1?
          BNE.S  MU3S162         No - check for 2
          MOVE.B PRE2K,TMPPRE    Set clock to 2 KHz
          MOVE.B CLK2K,TMPCLK
          BRA    SETCLK
```

```
*
*   Process 4KHz menu selection
*
MU3S162    CMPI.B #$32,(A6)          Key = 2?
           BNE.S  MU3S163            No - check for 3
           MOVE.B PRE4K,TMPPRE       Set clock to 4KHZ
           MOVE.B CLK4K,TMPCLK
           BRA    SETCLK
*
*    Process 8KHz menu selection
*
MU3S163    CMPI.B #$33,(A6)
           BNE.S  MU3S164
           MOVE.B PRE8K,TMPPRE
           MOVE.B CLK8K,TMPCLK
           BRA.S  SETCLK
*
*   Process 10KHz menu selection
*
MU3S164    CMPI.B #$34,(A6)
           BNE.S  MU3S165
           MOVE.B PRE10K,TMPPRE
           MOVE.B CLK10K,TMPCLK
           BRA.S  SETCLK
*
*    Process 16KHz menuselection
*
MU3S165    CMPI.B #$35,(A6)
           BNE.S  MU3S166
           MOVE.B PRE16K,TMPPRE
           MOVE.B CLK16K,TMPCLK
           BRA.S  SETCLK
*
*    Process Other Clock menu selection
*
MU3S166    CMPI.B #$36,(A6)
           BNE    MUESC
           MOVE.W #1,PARNO
           MOVE.W #$FF,OTHCLK
           LEA.L  PCLKMSG,A2          Display Oth. clock prompt
           MOVE.W PCLKCT,D2
           BSR    DSPMSG
           CLR.W  D6
           BRA    MURET
*
*   Set filter clock A with the new parameters
*
```

```
SETCLK    CMPI.W #16,MU3SL         Clock A set?
          BNE.S  SETCLKB           No - check B
          MOVE.B TMPPRE,PRSCA
          MOVE.B TMPCLK,FLCNTA
          MOVE.B TMPPRE,TACR       Set prescale & counter
          MOVE.B TMPCLK,TADR       for clock A
          MOVE.W #10,MU3SL         Display last mwnu
          BRA    LASTMU
SETCLKB   CMPI.W #17,MU3SL         Clock B set?
          BNE.S  SETCLKC           No - check C
          MOVE.B TMPPRE,PRSCB
          MOVE.B TMPCLK,FLCNTB
          MOVE.B TMPPRE,TBCR       Set prescale & counter
          MOVE.B TMPCLK,TBDR       for clock B
          MOVE.W #10,MU3SL         Display last menu
          BRA    LASTMU
SETCLKC   CMPI.W #18,MU3SL         Clock C set?
          BNE.S  SETCLKD           No - check D
          CLR.W  D1
          MOVE.B TMPPRE,D1
          LSL.B  #4,D1
          ANDI.B #$0F,PRSCCD
          OR.B   D1,PRSCCD
          MOVE.B TMPCLK,FLCNTC
          MOVE.B PRSCCD,TCDCR      Set prescale & counter
          MOVE.B FLCNTC,TCDR       for clock C
          MOVE.W #10,MU3SL         Display last menu
          BRA    LASTMU
SETCLKD   CLR.W  D1
          MOVE.B TMPPRE,D1
          ANDI.B #$F0,PRSCCD
          OR.B   D1,PRSCCD
          MOVE.B TMPCLK,FLCNTD
          MOVE.B PRSCCD,TCDCR      Set prescale & counter
          MOVE.B FLCNTD,TDDR       for clock D
          MOVE.W #10,MU3SL         Display last menu
          BRA.S  LASTMU
*
*   Process Power control menu
*
CHMU320   CMPI.W #20,MU3SL         Pwr control ?
          BGT.S  MUESC             No - return
          CMPI.B #$31,(A6)         Key = 1?
          BNE.S  MU3S202           No - check 2
          BCLR   #7,CNTRLM         Turn power on
          MOVE.B CNTRLM,CNTRL
          BRA.S  LASTMU            Display last menu
MU3S202   CMPI.B #$32,(A6)         Key = 2?
          BNE.S  MUERR             No - display error
          BSET   #7,CNTRLM         Turn power off
          MOVE.B CNTRLM,CNTRL
          BRA.S  LASTMU            Display last menu
*
*    Menu processor return
*
MURET     MOVE.L #0,(A6)
MURET1    CLR.W  D6
          RTS
```

```
*
*    Processor ESC (F4) key
*
MUESC      CMPI.B #ESC,KEY         Key = ESC?
           BNE.S  MUERR            No - display error
           BRA.S  LASTMU           Else display last menu
MUERR      LEA.L  ERRMSG,A2
           MOVE.W ERRMCT,D2
           BSR    DSPMSG
           CLR.W  D6
           BRA.S  MURET
*
*   If the ESC key was detected, this routine will display
*   the last menu in thew hierarchy
*
LASTMU     CMPI.W #0,MU4SL         Level 4 active?
           BEQ.S  LSTMU3           No check level 3
           CLR.W  D6
           MOVE.W D6,MU4SL         Else clear level 4
           MOVE.W MU3SL,D1         Get level 3 index
           SUBI.W #1,D1
           LSL.W  #2,D1
           LEA.L  MU3SAD,A0        Get level 3 menu
           MOVE.L (A0,D1),A2
           LEA.L  MU3SCT,A0        Get level 3 menu count
           MOVE.L (A0,D1),A1
           MOVE.W (A1),D2
           BSR    DSPMSG           Display level 3 menu
           BRA    MURET            Return
LSTMU3     CMPI.W #0,MU3SL         Level 3 active?
           BEQ.S  LSTMU2           No - check level 2
           CLR.W  D6
           MOVE.W D6,MU3SL         Else clear level 3
           MOVE.W MU2SL,D1         Get level 2 index
           SUBI.W #1,D1
           LSL.W  #2,D1
           LEA.L  MU2SAD,A0
           MOVE.L (A0,D1),A2       Get level 2 menu
           LEA.L  MU2SCT,A0
           MOVE.L (A0,D1),A1       Get level 2 menu counter
           MOVE.W (A1),D2
           BSR    DSPMSG           Display level 2 menu
           BRA    MURET            Return
LSTMU2     CLR.W  D6               Display main menu
           MOVE.W D6,MU2SL         Clear level 2 menu
           LEA.L  MENU1,A2
           MOVE.W MEN1CT,D2
           BSR    DSPMSG           Display main menu
           BRA    MURET            Return
           END
```

```
*
*************************************************************************
*
*           DSPSCT
*
*      This routine displays the scan table that is currently residing
*      in the array STSCT. The scan table contains the mux channel
*      numbers that are to be used for a test. These values are
*      converted to decimal then to ASCII before they are displayed.
*      The call to DSPMSG performs the actual display.
*
*           Input parameters : None
*           Output parameters : None
*           Registers used : A0 - index to DSPBUF
*                            A1 - index to STSCT
*                            A2 - pointer to the message to display
*                            D0 - general purpose
*                            D1 - general purpose
*                            D2 - Number of characters to display
*                            D3 - word size for CVTDEC
*                            D7 - value to convert for CVTDEC & CVTASCI
*
*************************************************************************
*
DSPSCT     IDNT    1,1
           INCLUDE ADAMDEF
           XREF    CVTASCI,DSPMSG,CVTDEC
           XDEF    DSPSCT
*
DSPSCT     LEA.L   DSPBUF,A0           Space out display buffer
           MOVE.L  #$20202020,D0
           MOVE.W  #50,D1
DSPCLLP    MOVE.L  D0,(A0)+
           SUBI.W  #1,D1
           BNE.S   DSPCLLP
           CLR.L   D0
           CLR.L   D1
           CLR.L   D7
           LEA.L   STSCT,A1            Get scan table pointer
           LEA.L   DSPBUF,A0           Get display buffer pointer
DSCTLP     MOVE.W  #1,D3
           MOVE.B  (A1),D7             Get mux channel no.
           BSR     CVTDEC              Convert to decimal
           BSR     CVTASCI             Convert to hex
           ADDI.W  #2,D1
           ADDA.L  #2,A0               Bump past channel in dsp buf
           MOVE.B  #$2C,(A0)+          Insert ","
           ADDI.W  #1,D1
```

```
         ADDI.W #1,D0
         CMPI.W #6,D0              If end of line
         BLT.S  DSPCCNT
         MOVE.W #$0D0A,(A0)+       Store CR & LF
         ADDI.W #2,D1
         CLR.W  D0
DSPCCNT  ADDA.L #1,A1              Bump scan table pointer
         CMP.L  ENDSCT,A1          End of scan table?
         BLT.S  DSCTLP             No - do it again
         LEA.L  DSCTMSG,A2         Else display it
         MOVE.W DSCTCT,D2
         BSR    DSPMSG
         LEA.L  DSPBUF,A2
         MOVE.W D1,D2
         SUBI.W #1,D2
         BSR    DSPMSG
         RTS
         END
```

```
**********************************************************************
*
*        CHCHECK
*
*    This routine performs a channel check of all 128 A/D channels.
*    The check consists of reading the data from each channel in
*    RCAL mode and saving it in CCBUF1. Then reading the data in
*    Non-RCAL mode and saving it in CCBUF2. Finally the two arrays
*    are compared and any two data values that show less than 10
*    counts difference in the positive or negitive range, the mux
*    channel is reported in error. For each mux channel, there are
*    four A/D channels used in the comparison. The bad channel
*    numbers are converted to decimal (CVTDEC) then to ASCII
*    (CVTASCI) and finally displayed by DSPMSG.
*
*        Input parameters : none
*        Output parameters : none
*
**********************************************************************
*
CHCHECK   IDNT    1,1
ADC       EQU     $800000              Address of A/D
CNTRL     EQU     $800031              Address of control port
*
          INCLUDE ADAMDEF
          XREF    DSPMSG,CVTASCI,CVTDEC
          XDEF    CHCHECK
*
*
CHCHECK   MOVE.W #50,D0                Space out display message
          MOVE.L #$20202020,D1
          LEA.L   DSPBUF,A0
CCCLRLP   MOVE.L D1,(A0)+
          SUBI.W #1,D0
          BNE.S   CCCLRLP
          CLR.L   D0                   Set mux to channel 0
          MOVE.L D0,ADC
          MOVE.L ADC,D1                Reset A/D
          LEA.L   CCBUF1,A0
          BSET    #6,CNTRLM            Set to Non-RCAL mode
          MOVE.B CNTRLM,CNTRL
          MOVE.W #18000,D1
CCDEL1    NOP                          Delay
          MULU    D0,D0
          SUBI.W #1,D1
          BNE.S   CCDEL1
          CLR.L   D0
          MOVE.W #10,D1                Set counter for 10 scans
CCONE     MOVE.L ADC,(A0,D0)           Read and store A/D value
          ADDI.W #4,D0                 Incr. buffer pointer
          CMPI.W #128,D0               Check for end
          BNE.S   CCONE                No - continue
          CLR.W   D0                   Else do again
          SUBI.W #1,D1                 10 times
```

```
          BNE.S   CCONE
          BCLR    #6,CNTRLM          Set to RCAL mode
          MOVE.B  CNTRLM,CNTRL
          MOVE.W  #$8000,D1
CCDEL2    NOP                        Delay
          MULU    D0,D0
          SUBI.W  #1,D1
          BNE.S   CCDEL2
          CLR.L   D0                 Set mux to 0
          LEA.L   CCBUF2,A0
          MOVE.W  #10,D1             Set for 10 scans
CCTWO     MOVE.L  ADC,(A0,D0)        Read and store A/D value
          ADDI.W  #4,D0              Incr. buffer pointer
          CMPI.W  #128,D0            End of scan?
          BNE.S   CCTWO              No - continue
          CLR.W   D0                 Else reset mux to 0
          SUBI.W  #1,D1              Do again 10 times
          BNE.S   CCTWO
          BSET    #6,CNTRLM          Set to Non-RCAL mode
          MOVE.B  CNTRLM,CNTRL
          LEA.L   DSPBUF,A0          Get display buffer
          MOVE.W  #1,D3
          CLR.L   D7
          CLR.L   D4
          LEA.L   CCBUF1,A1          Load non-RCAL buffer pntr
          LEA.L   CCBUF2,A3          Load RCAL buffer ptr
CDATLP    MOVE.B  (A3,D4),D7         Get RCAL value
          SUB.B   (A1,D4),D7         Subtract from Non-RCAL value
          CMPI.B  #0,D7              If > 0 go check pos. range
          BGE.S   CDAPOS
          CMPI.B  #-10,D7            If dif. is < 10 in neg range
          BGT.S   CCERR              Display error
          BRA.S   CDATLP1            Else check next channel
CDAPOS    CMPI.B  #10,D7             If dif < 10 in pos range
          BLT.S   CCERR              Display error
CDATLP1   ADDI.W  #1,D4              Incr buffer pointer
          CMPI.W  #128,D4            If not end
          BLT.S   CDATLP             Check next channel
          CMP.L   #DSPBUF,A0         If no errors
          BEQ.S   NOCCERR            Return
          LEA.L   CCMSG,A2           Else display bad channels
          MOVE.W  CCMCT,D2
          BSR     DSPMSG
          LEA.L   DSPBUF,A2
          MOVE.L  A0,D2
          SUB.L   #DSPBUF,D2
          SUBI.W  #1,D2
          BSR     DSPMSG
          RTS
```

```
*
NOCCERR   LEA.L   CCPASS,A2         Display channel check passed
          MOVE.W  CCPCT,D2
          BSR     DSPMSG
          RTS
*
CCERR     CLR.L   D7
          MOVE.W  D4,D7             Get bad channel number
          LSR.L   #2,D7             Convert channel number to
          MOVE.W  D7,D4               mux channel
          LSL.L   #2,D4
          ADDI.W  #3,D4
          MOVE.W  #1,D3
          BSR     CVTDEC            Convert to decimal
          BSR     CVTASCI           Convert to ASCII and store in
          ADDA.L  #2,A0                DSPBUF
          MOVE.B  #$2C,(A0)+        Store ","
          ADDI.W  #1,D0
          CMPI.W  #6,D0             If end of line
          BLT.S   CDATLP1
          MOVE.W  #$0D0A,(A0)+      Store carrage return line feed
          CLR.W   D0
          BRA     CDATLP1
          END
```

```
*
*************************************************************************
*
*          CVTHEX
*
*     This routine converts the ASCII characters in KEYBUF (pointed to
*     by A6) first to an unpacked decimal number and then to a packed
*     hex number. The result is stored in the word INVAL. CVTHEX will
*     convert any number from 0 to 999 to 0 through 3E7. The input
*     value must be pointed to by register A6 with the number of
*     characters in D6.
*
*          Input parameters : Register A6 - pointer to input buffer
*                                 "      D6 - number of characters to
*                                              convert
*
*          Output parameters : INVAL - converted hex value
*
*          Registers used : A0 - temporary intermediate value pointer
*                           D0 - temporary character count
*                           D1 - general purpose
*                           Note : all registers are saved and restored
*                                  by CVTHEX
*
*************************************************************************
*
CVTHEX    IDNT    1,1
*
          INCLUDE ADAMDEF
          XDEF    CVTHEX
*
*
CVTHEX    MOVEM.L D0-D1/A0,-(A7)     Save registers
          MOVE.W D6,D0              Get number of characters
          CMPI.W #0,D0              If zero - return
          BNE.S  ASCITRN
          MOVE.W #0,INVAL
          BRA    CVTRET
ASCITRN   MOVE.L #0,TMPCVT          Clear temp val
          LEA.L  TMPCVT+3,A0
          SUBI.W #1,D0
ASCILP    MOVE.B (A6,D0),D1         Get input character
          CMPI.B #$30,D1            Check for valid numeric value
          BLT.S  CVTERR
          CMPI.B #$39,D1
          BGT.S  CVTERR
          ANDI.B #$0F,D1            get low nibble
          MOVE.B D1,(A0)            And store in temp buffer
          SUBI.W #1,D0
          CMPI.W #0,D0
```

```
         BLT.S   HEXTRN               If last character - go convert
         SUBA.L  #1,A0
         BRA     ASCILP
HEXTRN   CLR.W   D0
         MOVE.W  D0,INVAL             Clear result
         CLR.L   D1
         LEA.L   TMPCVT,A0            Get temp buffer
         MOVE.B  (A0,D0),D1           Get high digit
         MULU.W  #$3E8,D1             Multiply by 1000
         ADD.W   D1,INVAL             Add to result
         CLR.L   D1
         ADDI.W  #1,D0
         MOVE.B  (A0,D0),D1           Get next digit
         MULU.W  #$64,D1              Multiply by 100
         ADD.W   D1,INVAL             Add to result
         CLR.L   D1
         ADDI.W  #1,D0
         MOVE.B  (A0,D0),D1           Get next digit
         MULU.W  #$0A,D1              Multiply by 10
         ADD.W   D1,INVAL             Add to result
         CLR.L   D1
         ADDI.W  #1,D0
         MOVE.B  (A0,D0),D1           Get last digit
         ADD.W   D1,INVAL             Add to result
CVTRET   MOVEM.L (A7)+,D0-D1/A0       Restore registers
         RTS
CVTERR   MOVE.W  #-1,INVAL            Set error status
         BRA.S   CVTRET
         END
```

```
*
**********************************************************************
*
*          TSTALL
*
*     This routine is called by MENUPR when the Test All selection
*     is made on the Memory Diagnostic menu. TSTALL calls all of
*     the memory test routines and checks the error status
*     (MEMFAIL). The routines that are called are PATTST,
*     ADRTST, BU0TST( and BU1TST. If the tests passed, this
*     routine calls DSPMSG to report a passed status.
*
*          Input parameters : none
*
*          Output parameters : MEMFAIL - test failed status
*
*          Registers used : A2 - pointer to display message
*                           D2 - display count
*
**********************************************************************
*
TSTALL     IDNT     1,1
*
           INCLUDE ADAMDEF
           XREF     PATTST,ADRTST,BU0TST,BU1TST,DSPMSG
           XDEF     TSTALL
*
TSTALL     BSR      PATTST               Perform pattern test
           CMPI.W   #0,MEMFAIL           Check for error
           BNE.S    TSTARET
           BSR      ADRTST               Perform address test
           CMPI.W   #0,MEMFAIL           Check for error
           BNE.S    TSTARET
           BSR      BU0TST               Perform bubble 0 test
           CMPI.W   #0,MEMFAIL           Check for error
           BNE.S    TSTARET
           BSR      BU1TST               Perform bubble 1 test
           CMPI.W   #0,MEMFAIL           Check for error
           BNE.S    TSTARET
           LEA.L    MEMPAS,A2            If no errors display passed msg.
           MOVE.W   MEMPCT,D2
           BSR      DSPMSG
TSTARET    RTS
           END
```

```
*
**********************************************************************
*
*          BU1TST
*
*     BU1TST performs a bubble one test on memory. This is a word test
*     that shifts a zero bit through each of the 16 bits of each word
*     in memory. The memory that is tested is 1000000 through 107EFF0
*     ( STRAM,ENRAM).
*
*          Input parameters : None
*          Output parameters : MEMFAIL - indicates that test failed
*
*          Registers used : A0 - start of memory
*                           A1 - end of memory
*                           A2 - current memory pointer
*                           D0 - test pattern
*                           D3 - word length for CVTASCI
*                           *Note : all registers are saved
*
**********************************************************************
*
BU1TST     IDNT    1,1
*
STRAM      EQU     $1000000             Start of memory
ENRAM      EQU     $107EFF0             End of tested memory
           INCLUDE ADAMDEF
           XREF    DSPMSG,CVTASCI
           XDEF    BU1TST
*
BU1TST     MOVEM.L D0-D3/D7/A0-A2,-(A7)      Save registers
           LEA.L   STRAM,A0             Get start of memory
           LEA.L   ENRAM,A1             Get end of memory
           MOVE.L  A0,A2                Save start
SETONE     MOVE.W  #1,D0                Initialize test mask
ONEWR      MOVE.W  D0,(A2)              Store test pattern
           CMP.W   (A2),D0              Read and compare
           BNE.S   ONEERR               If not = set error
           ASL.W   #1,D0                Shift test pattern
           BCC.S   ONEWR
           ADDQ.L  #2,A2                Incr. memory address
           CMP.L   A2,A1                Check for end
           BCC.S   SETONE
           MOVE.W  #0,MEMFAIL           Clear error status
           CMPI.W  #0,TSTALL            If test all
           BNE.S   BU1EXT               Exit
           LEA.L   MEMPAS,A2            Else display passed msg.
           MOVE.W  MEMPCT,D2
           BSR     DSPMSG
BU1EXT     MOVEM.L (A7)+,D0-D3/D7/A0-A2   Restore registers
           RTS
```

```
*
ONEERR    MOVE.W  #$FF,MEMFAIL          Set error status
          MOVE.L  A2,A1                 Get error address
          LEA.L   DSPBUF,A0             Build error display msg
          CLR.L   D7
          MOVE.W  D0,D7
          MOVE.W  #2,D3
          BSR     CVTASCI               Store test pattern written
          LEA.L   MEMERM,A2                into display msg
          MOVE.W  MEMERC,D2
          BSR     DSPMSG
          LEA.L   DSPBUF,A2
          MOVE.W  #4,D2
          BSR     DSPMSG
          LEA.L   DSPBUF,A0
          MOVE.W  (A1),D7
          MOVE.W  #2,D3
          BSR     CVTASCI               Store test pattern read into
          LEA.L   MEMERM1,A2               display msg
          MOVE.W  MEMERC1,D2
          BSR     DSPMSG
          LEA.L   DSPBUF,A2
          MOVE.W  #4,D2
          BSR     DSPMSG
          LEA.L   DSPBUF,A0
          MOVE.L  A1,D7
          MOVE.W  #4,D3
          BSR     CVTASCI               Store error address into display
          LEA.L   MEMERM2,A2               msg
          MOVE.W  MEMERC2,D2
          BSR     DSPMSG
          LEA.L   DSPBUF,A2
          MOVE.W  #8,D2
          BSR     DSPMSG
          MOVEM.L (A7)+,D0-D3/D7/A0-A2     Restore registers
          RTS
          END
```

```
*
*************************************************************************
*
*          BU0TST
*
*     BU0TST performs a bubble zero test on memory. This is a word
*     test that shifts a zero bit through each of the 16 bits of
*     each word in memory. That memory that is tested is 1000000
*     through 107EFF0. (STRAM,ENRAM)
*
*          Input parameters : none
*
*          Output parameters : MEMFAIL - indicates that the test failed
*
*          Registers used : A0 - start of memory
*                           A1 - end of memory
*                           D0 - test pattern
*                           A2 - current memory pointer
*                           D3 - word length for CVTASCI
*                                note : all registers are saved
*
*************************************************************************
*
BU0TST     IDNT    1,1
*
STRAM      EQU     $1000000            Start of memory
ENRAM      EQU     $107EFF0            End of tested memory
           INCLUDE ADAMDEF
           XREF    DSPMSG,CVTASCI
           XDEF    BU0TST
*
BU0TST     MOVEM.L D0-D3/D7/A0-A2,-(A7)      Save registers
           LEA.L   STRAM,A0
           LEA.L   ENRAM,A1
           MOVE.L  A0,A2               Save start of memory
SETZER     MOVE.W  #$FFFE,D0           Set initial test pattern
ZERWR      MOVE.W  D0,(A2)             Write bubble 0 word
           CMP.W   (A2),D0             Read & compare
           BNE.S   BUBZERR             Go if error
           ROL.W   #1,D0               Shift 0 to next bit
           BCS.S   ZERWR               Continue
           ADDQ.L  #2,A2               Go to next memory location
           CMP.L   A2,A1
           BCC.S   SETZER
           MOVE.W  #0,MEMFAIL          Clear error status
           CMPI.W  #0,TSTAFL           Test all?
           BNE.S   BU0EXT              Yes - skip msg display
           LEA.L   MEMPAS,A2           Else display passed
           MOVE.W  MEMPCT,D2
           BSR     DSPMSG
BU0EXT     MOVEM.L (A7)+,D0-D3/D7/A0-A2      Restore registers
           RTS
```

```
*
BUBZERR    MOVE.W  #$FF,MEMFAIL          Set error flag
           MOVE.L  A2,A1                 Build error message
           LEA.L   DSPBUF,A0
           CLR.L   D7
           MOVE.W  D0,D7
           MOVE.W  #2,D3                 Display test pattern written
           BSR     CVTASCI
           LEA.L   MEMERM,A2
           MOVE.W  MEMERC,D2
           BSR     DSPMSG
           LEA.L   DSPBUF,A2
           MOVE.W  #4,D2
           BSR     DSPMSG
           LEA.L   DSPBUF,A0
           MOVE.W  (A1),D7
           MOVE.W  #2,D3                 Display test pattern read
           BSR     CVTASCI
           LEA.L   MEMERM1,A2
           MOVE.W  MEMERC1,D2
           BSR     DSPMSG
           LEA.L   DSPBUF,A2
           MOVE.W  #4,D2
           BSR     DSPMSG
           LEA.L   DSPBUF,A0
           MOVE.L  A1,D7
           MOVE.W  #4,D3                 Display error address
           BSR     CVTASCI
           LEA.L   MEMERM2,A2
           MOVE.W  MEMERC2,D2
           BSR     DSPMSG
           LEA.L   DSPBUF,A2
           MOVE.W  #8,D2
           BSR     DSPMSG
           MOVEM.L (A7)+,D0-D3/D7/A0-A2  Restore registers
           RTS
           END
```

```
*
************************************************************************
*
*         ADRTST
*
*    ADRTST performs an address test on memory. This test consists
*    of writting the long word address of a memory location into
*    memory. The memory that is tested is from address 1000000
*    through 107EFF0 (STRAM,ENRAM).
*
*         Input parameters : none
*         Output parameters : MEMFAIL - indicates that the test failed
*
*         Registers used : A0 - start of memory
*                          A1 - end of tested memory
*                          A2 - current memory pointer
*                          D0 - address value
*                          D3 - word length for CVTASCI
*                          * note : all registers are saved
*
************************************************************************
*
*
ADRTST    IDNT    1,1
*
STRAM     EQU     $1000000          Start of memory
ENRAM     EQU     $107EFF0          End of tested memory
          INCLUDE ADAMDEF
          XREF    DSPMSG,CVTASCI
          XDEF    ADRTST
*
ADRTST    MOVEM.L D0-D3/D7/A0-A2,-(A7)    Svae registers
          LEA.L   STRAM,A0          Get start of memory
          LEA.L   ENRAM,A1          Get end of memory
          CLR.W   D7
          MOVE.L  A0,A2             Save start
          MOVE.L  A0,D0             Get first addres test pattern
ADRWRT    MOVE.L  D0.(A2)+          Store test address
          ADDQ.L  #4,D0             Incr. test pattern
          CMP.L   A2,A1             Check for end of write
          BCC.S   ADRWRT
          MOVE.L  A0,A2             Get start of memory
          MOVE.L  A0,D0             Reset test pattern
ADRRD     CMP.L   (A2)+,D0          Read and compare data
          BNE.S   ADRERR            If not = set error
          ADDQ.L  #4,D0             Incr test pattern
          CMP.L   A2,A1             Check for end
          BCC.S   ADRRD
          MOVE.W  #0,MEMFAIL        Clear error status
          CMPI.W  #0,TSTAFL         If test all
          BNE.S   ADREXT            Exit
          LEA.L   MEMPAS,A2         Else display passed message
          MOVE.W  MEMPCT,D2
          BSR     DSPMSG
ADREXT    MOVEM.L (A7)+,D0-D3/D7/A0-A2  Restore registers
```

```
         RTS
*
ADRERR   SUBA.L  $4,A2              Get errored address
         MOVE.L  A2,A1
         MOVE.W  #$FF,MEMFAIL       Set error status
         LEA.L   DSPBUF,A0          Set up display message
         CLR.L   D7
         MOVE.L  D0,D7
         MOVE.W  #4,D3
         BSR     CVTASCI            Store data written into display
         LEA.L   MEMERM,A2            msg
         MOVE.W  MEMERC,D2
         BSR     DSPMSG
         LEA.L   DSPBUF,A2
         MOVE.W  #8,D2
         BSR     DSPMSG
         LEA.L   DSPBUF,A0
         MOVE.L  (A1),D7            Store data read into display msg
         MOVE.W  #4,D3
         BSR     CVTASCI
         LEA.L   MEMERM1,A2
         MOVE.W  MEMERC1,D2
         BSR     DSPMSG
         LEA.L   DSPBUF,A2
         MOVE.W  #8,D2
         BSR     DSPMSG
         MOVE.L  A1,D7
         MOVE.W  #4,D3
         LEA.L   DSPBUF,A0
         BSR     CVTASCI            Store error address into display
         LEA.L   MEMERM2,A2           msg
         MOVE.W  MEMERC2,D2
         BSR     DSPMSG
         LEA.L   DSPBUF,A2
         MOVE.W  #8,D2
         BSR     DSPMSG
         MOVEM.L (A7)+,D0-D3/D7/A0-A2    Restore registers
         RTS
         END
```

```
*
*************************************************************************
*
*                        WDTST
*
*    This routine is used by RAMTST, RAMDIAG, and clear memory to
*    write a data pattern to memory and check it. WDTST does a
*    16 bit word write and read of the data in register D0 to the
*    address in register A2 up to the address in register A1. If
*    there are any errors in the compare, register D7 is set to FF,
*    and register A2 contains the error address.
*
*       Input parameters :
*                 Reg. A0 - start of test memory
*                 Reg. A1 - end of test memory
*                 Reg. D0 - test data pattern (word)
*
*       Returned parameters :
*
*                 Reg. A2 - memory error address
*                 Reg. D0 - test pattern written
*                 Reg. D1 - data read from memory
*                 Reg. D7 - error flag (set to FF if failed)
*
**************************************************************************
*
WDTST     IDNT    1,1
          XDEF    WDTST
*
WDTST     MOVE.L A0,A2               Get start of memory
WDWRT     MOVE.B D0,(A2)+            Store test pattern in memory
          CMPA.L A2,A1               End of memory?
          BCC.S   WDWRT              No - continue writting
          MOVE.L A0,A2               Else reset start of memory
WDRD      MOVE.B (A2)+,D1            Read memory
          CMP.B   D0,D1              Input data = test pattern?
          BNE.S   WDERR              No - set error status
          CMPA.L A1,A2               End of memory?
          BLE.S   WDRD               No - continue check
          RTS                        Else return
WDERR     SUBA.L #1,A2               Set error address
          MOVE.B #$FF,D7             Set error status
          RTS                        Return
          END
```

```
*
************************************************************************
*
*         PATTST
*
*   PATTST performs a pattern test on memory. This test consists
*   of writting several byte data patterns to memory and checking
*   the results. The data patterns include : AA,55,FF, and 00.
*   The memory tthat is tested is from address 1000000 through
*   107EFF0 (STRAM,ENRAM).
*
*         Input parameters : none
*         Output parameters : MEMFAIL - indicates that the test failed
*
*         Registers used : A0 - start of memory
*                          A1 - end of memory
*                          A2 - current memory pointer
*                          D0 - data pattern value
*                          D3 - word length for CVTASCI
*                          * note: all registers are saved
*
************************************************************************
*
PATTST    IDNT    1,1
*
STRAM     EQU     $1000000             Start of memory
ENRAM     EQU     $107EFF0             End of tested memory
          INCLUDE ADAMDEF
          XREF    WDTST,DSPMSG,CVTASCI
          XDEF    PATTST
*
PATTST    MOVEM.L D0-D1/D3/D7/A0-A2,-(A7)    Save registers
          LEA.L   STRAM,A0             Get start of memory
          LEA.L   ENRAM,A1             Get end of memory
          MOVE.B  #$AA,D0              Load first test pattern
          CLR.W   D7                   Clear status
          BSR     WDTST                Perform test
          CMPI.W  #0,D7                Check status
          BNE.S   PATERR
          MOVE.B  #$55,D0              Load second test pattern
          BSR     WDTST                Perform test
          CMPI.W  #0,D7                Check status
          BNE.S   PATERR
          MOVE.B  #$FF,D0              load third test pattern
          BSR     WDTST                Perform test
          CMPI.W  #0,D7                Check status
          BNE.S   PATERR
          CLR.W   D0                   Load final test pattern
          BSR     WDTST                Perform test
          CMPI.W  #0,D7                Check status
          BNE.S   PATERR
          CMPI.W  #0,TSTAFL            If test all
          BNE.S   PATEXT               Exit
          LEA.L   MEMPAS,A2            Else display test passed msg
          MOVE.W  MEMPCT,D3
          BSR     DSPMSG
```

```
PATEXT   MOVE.W D7,MEMFAIL           Clear test status
         MOVEM.L (A7)+,D0-D1/D3/D7/A0-A2   Restore registers
         RTS
*
PATERR   MOVE.W D7,MEMFAIL           Set error status
         MOVE.L A2,A1                Get error address
         LEA.L  DSPBUF,A0            Build error message
         CLR.L  D7
         MOVE.W D0,D7
         MOVE.W #1,D3
         BSR    CVTASCI              Store pattern written into
         LEA.L  MEMERM,A2               display msg
         MOVE.W MEMERC,D2
         BSR    DSPMSG
         LEA.L  DSPBUF,A2
         MOVE.W #4,D2
         BSR    DSPMSG
         MOVE.W D1,D7
         MOVE.W #1,D3
         LEA.L  DSPBUF,A0
         BSR    CVTASCI              Store pattern read into
         LEA.L  MEMERM1,A2              display msg
         MOVE.W MEMERC1,D2
         BSR    DSPMSG
         LEA.L  DSPBUF,A2
         MOVE.W #4,D2
         BSR    DSPMSG
         MOVE.L A1,D7
         MOVE.W #4,D3
         LEA.L  DSPBUF,A0
         BSR    CVTASCI              Store error address into display
         LEA.L  MEMERM2,A2                msg
         MOVE.W MEMERC2,D2
         BSR    DSPMSG
         LEA.L  DSPBUF,A2
         MOVE.W #8,D2
         BSR    DSPMSG
         MOVEM.L (A7)+,D0-D1/D3/D7/A0-A2   Restore registers
         RTS
         END
```

```
*********************************************************************
*
*          DSPSCT
*
*     This routine displaya the scan table that is currently residing
*     in the array STSCT. The scan table contains the mux channel
*     numbers that are to be used for a test. These values are
*     converted to decimal then to ASCII before they are displayed.
*     The call to DSPMSG performs the actual display.
*
*          Input parameters : None
*          Output parameters : None
*          Registers used : A0 - index to DSPBUF
*                           A1 - index to STSCT
*                           A2 - pointer to the message to display
*                           D0 - general purpose
*                           D1 - general purpose
*                           D2 - Number of characters to display
*                           D3 - word size for CVTDEC
*                           D7 - value to convert for CVTDEC & CVTASCI
*
*********************************************************************
*
DSPSCT    IDNT    1,1
          INCLUDE ADAMDEF
          XREF    CVTASCI,DSPMSG,CVTDEC
          XDEF    DSPSCT
*
DSPSCT    LEA.L   DSPBUF,A0           Space out display buffer
          MOVE.L  #$20202020,D0
          MOVE.W  #50,D1
DSPCLLP   MOVE.L  D0,(A0)+
          SUBI.W  #1,D1
          BNE.S   DSPCLLP
          CLR.L   D0
          CLR.L   D1
          CLR.L   D7
          LEA.L   STSCT,A1            Get scan table pointer
          LEA.L   DSPBUF,A0           Get display buffer pointer
DSCTLP    MOVE.W  #1,D3
          MOVE.B  (A1),D7             Get mux channel no.
          BSR     CVTDEC              Convert to decimal
          BSR     CVTASCI             Convert to hex
          ADDI.W  #2,D1
          ADDA.L  #2,A0               Bump past channel in dsp buf
          MOVE.B  #$2C,(A0)+          Insert ","
          ADDI.W  #1,D1
          ADDI.W  #1,D0
          CMPI.W  #6,D0               If end of line
          BLT.S   DSPCCNT
          MOVE.W  #$0D0A,(A0)+        Store CR & LF
          ADDI.W  #2,D1
          CLR.W   D0
```

```
DSPCCNT    ADDA.L  #1,A1           Bump scan table pointer
           CMP.L   ENDSCT,A1       End of scan table?
           BLT.S   DSCTLP          No - do it again
           LEA.L   DSCTMSG,A2      Else display it
           MOVE.W  DSCTCT,D2
           BSR     DSPMSG
           LEA.L   DSPBUF,A2
           MOVE.W  D1,D2
           SUBI.W  #1,D2
           BSR     DSPMSG
           RTS
           END
```

```
*************************************************************************
*
*          PARLTST
*
*   PARLTST performs the power up diagnostic on the parallel port.
*   It outputs a pattern to the port and reads and compares that
*   value. Since this is an internal test, the port is set to
*   output mode during the entire test. The sync code FAF30000 is
*   output first so that the DRASS wont try to process the data.
*   The data pattern that is used is : 00000000, 01010101,...
*   0F0F0F0F.
*
*          Input parameters : None
*          Output parameters : DSTAT - bit 4 is set if there is a
*                                        test failure
*
*          Registers used : D1 - contains the data pattern
*                           D2 - test counter
*                           D7 - input value
*                           D0 - temporary register
*
*************************************************************************
*
PARLTST    IDNT    1,1
*
           INCLUDE ADAMDEF
           XDEF    PARLTST
*
PPRT       EQU     $800020              Parallel port address
CNTRL      EQU     $800031              Control port address
*
PARLTST    MOVE.L  PPRT,D0              Reset parallel port
           BCLR    #0,CNTRLM            Set to output mode
           MOVE.B  CNTRLM,CNTRL
           MOVE.L  #$FAF30000,PPRT      Send DRASS sync code for power up test
           CLR.L   D1                   Initialize test pattern
           MOVE.W  #16,D2               Test test loop counter
PARLOP     MOVE.L  D1,PPRT              Output test pattern
           MOVE.L  PPRT,D7              Read test pattern
           CMP.L   D1,D7                Compare in & out values
           BNE.S   PARERR               Display error if not =
           ADDI.L  #$01010101,D1        Increment test pattern
           SUBI.W  #1,D2                Check for end of test
           BNE.S   PARLOP
           BSET    #0,CNTRLM            Set parallel port to input mode
           MOVE.B  CNTRLM,CNTRL
           RTS                          Return
PARERR     BSET    #4,DSTAT             Set parallel port error status
           BSET    #0,CNTRLM            Set port to input mode
           MOVE.B  CNTRLM,CNTRL
           RTS                          Return
           END
```

```
*
            OPT     P=68020
*           DRASS SYSTEM
*
PPRT        EQU     $10020
CNTRL       EQU     $10824
VECTOR      EQU     0
IRQMSK      EQU     $2700
SYSTK       EQU     $107C0
CACHEN      EQU     1
*
            ORG     0
INTSTK      DC.L    $107C0
PRGST       DC.L    DRASS
*
            ORG     $100
DRASS       MOVE.W  #IRQMSK,SR
            MOVE.L  #VECTOR,A0
            MOVEC   A0,VBR
            MOVE.L  #SYSTK,A7
            MOVEC   A7,ISP
            MOVE.L  #CACHEN,A0
            MOVEC   A0,CACR
            CLR.L   D0
            CLR.L   D2
            MOVE.B  #04,CNTRL
            MOVE.L  PPRT,D1
ADAMCON     MOVE.L  CNTRL,D1
            ANDI.L  #$800000,D1
*           BNE.S   ADAMCON
ADMRDY      MOVE.L  CNTRL,D1
            ANDI.L  #$200000,D1
            BEQ.S   ADMRDY
            MOVE.L  PPRT,D0
            CMPI.L  #$FAF30001,D0
            BEQ.S   ADMDIAG
            MOVE.L  D0,D1
            ANDI.L  #$FFFFFF00,D0
            CMPI.L  #$FAF3FF00,D0
            BEQ.S   SNDACK
            CMPI.L  #$FAF30000,D0
            BEQ.S   RSTDIAG
            CMPI.L  #$FAF31000,D0
```

```
        BEQ.S   RSTDIAG
        CMPI.L  #$FAF32000,D0
        BNE.S   DIAGCK
RSTDIAG MOVE.W  #0,D2
        BRA.S   ADMRDY
DIAGCK  CMPI.W  #0,D2
        BEQ.S   ADMRDY
        MOVE.L  D1,D0
        BRA.S   SNDACK
ADMDIAG MOVE.W  #$FF,D2
        BRA.S   ADMRDY
*
SNDACK  MOVE.L  CNTRL,D1
        ANDI.L  #$100000,D1
        BEQ.S   SNDACK
ADMBSY  MOVE.L  CNTRL,D1
        ANDI.L  #$400000,D1
        BNE.S   ADMBSY
        MOVE.B  #0,CNTRL
        MOVE.L  D0,PPRT
ADMBSY1 MOVE.L  CNTRL,D1
        ANDI.L  #$400000,D1
        BNE.S   ADMBSY1
        MOVE.B  #04,CNTRL
        BRA     ADMRDY
        END
```

```
*****************************************************************************
*
*                                DRASS
*
*   The DRASS program is structured as a command processor. The commands
*   are received by BUTINT, which services the interrupt generated by
*   the front panel buttons and passes the command to CMDRPR or DIAGPR
*   for processing. The position of the Run/Test switch determines
*   which routine sevices the command. The position of the thumbwheel
*   switches, and Run/Test toggle switch when the start button is
*   pressed determines what command will be performed. When a command
*   is selected, it is processed until it is completed or the Halt
*   button is pressed.
*
*                         Module Hierarchy
*
*    DRASS (Initialization) : ROMTST - Power up ROM test
*                             SERTST - Power up serial test
*                             PARLTST - Power up parallel test
*                             LITTST - Power up light test
*                             RAMTST - Power up RAM test
*                             DISPLAY - Display message
*
*    MAINLP (Process server) : CMDPR  - Check Run/Test switch and perform
*                                        ADAM data transfer (DLDATA)
*                              CMDPR1 - DCOM data transfer (DLDCOM)
*                              CMDPR2 - Perform serial output of
*                                       data (OUTDATA)
*                              CMDPR3 - Erase data memory (CLRMEM)
*                              DIAGPR  - Perform RAM diagnostic (RAMDIAG)
*                              DIAGPR1 - Perform serial diag. (SERDIAG)
*                              DIAGPR2 - Perform display diag. (DSPDIAG)
*                              DIAGPR3 - Perform parallel diag. (PARDIAG)
*                              DIAGPR4 - Perform light diag. (LITDIAG)
*                              DIAGPR5 - Perform switch diag. (SWTDIAG)
*
*    BUTINT (Button interrupt service routine)
*                              Reads the button pressed. If it was the
*                              Start button, store the thumbwheel
*                              setting in CMD. If the Halt button was
*                              pressed, set the halt flag (HLTFLG). If
*                              the Continue button was pressed, set the
*                              continue flag (CONTFL).
*
*    Registers used : A0 - temporary index register
*                     A7 - stack pointer
*                     D0 - temporary data register
*
*                     Status Byte Definition
```

```
*
*     Power up diagnostic status (DSTAT)
*          Bit 0 - ROM error
*          Bit 1 - Serial error
*          Bit 2 - Parallel error
*          Bit 3 - RAM error
*
*     System status (STAT)
*          Bit 1 - DRASS online/offline
*          Bit 2 - DRASS read/write mode
*          Bit 3 - Memory full status
*          Bit 4 - Test fail status
*          Bit 5 - Busy status
*          Bit 6 - Test passed status
*
*****************************************************************
*
          OPT      P=68020
DRASS     IDNT     1,1
*
          XDEF     PREBUF,PREEND,POSTBU,POSTEND,STSCT,ENDSCT,BLKLEN
          XDEF     UCNTRL,BAUD,PRSCA,PRSCB,PRSCCD,FLCNTA,FLCNTB
          XDEF     FLCNTC,FLCNTD
          XDEF     DSTAT,STAT,DSPBUF,LSTCMD,CONTFL,HLTFLG,CMD,LSTSWT
          XDEF     STRTFL,SUMCHK,PRESYNC,DATSYNC,POSSYNC,LSTPTR
          XDEF     DATBUF,DATEND,BLKSTAT,LSTSEL,TCNT,SERST,ENDBUF
          XDEF     DMSG,PRMPT,TEXT1,TEXT2,TEXT3,TEXT4,TEXT5,TEXT6
          XDEF     TEXT7,TERRMSG,MFMSG,ERRMSG,BAUDMSG,SERFMT,DFMT
          XDEF     PRESC,CNTTBL,FMTTBL,ENDROM,SERERM,DRAMEM,CRAMEM
          XDEF     NDATMSG,RETRY,DATCNT
*
          XREF     ROMTST,SERTST,PARLTST,LITTST,RAMTST,DISPLAY
          XREF     DLDATA,OUTDATA,CLRMEM,RAMDIAG,SERDIAG,DSPDIAG
          XREF     PARDIAG,LITDIAG,SWTDIAG,DLDCOM
*
*
          SECTION  0
SYSTP     DC.L     $107F0              System stack
INITPC    DC.L     START               Starting address of program
          DC.L     BUSERR              Buss error addr.
          DC.L     ADDERR              Address error addr.
          DC.L     ERRVEC
          DC.L     ERRVEC
          DC.L     0,0
          DC.L     PRIVER              Privelege error addr.
          DC.L     0,0,0,0
          DC.L     0,0,0
          DC.L     0,0,0,0
          DC.L     0,0,0,0
          DC.L     SPURINT             Spurious interrupt vector
          DC.L     AVEC                Auto vector 1
          DC.L     AVEC                Auto vector 2
          DC.L     AVEC                Auto vector 3
          DC.L     BUTINT              Button interrupt vector
          DC.L     AVEC                Auto vector 5
          DC.L     AVEC                Auto vector 6
          DC.L     AVEC                Auto vector 7
```

```
*
SWORD0    EQU   $2700        Disable int. status wrd
SWORD1    EQU   $2000        Enable int. status wrd
VBASE     EQU   0            Vector base addr.
CEN       EQU   1            Cache enable
ISTACK    EQU   $107F0       Interrupt stack addr.
PROM      EQU   0            Start of PROM
STRAM     EQU   $1000000     Start of static RAM
ENRAM     EQU   $107FFFE     End of static RAM
DATBUF    EQU   $1000000     Start of data buffer
DATEND    EQU   $107CFFF     End of data buffer
STCACH    EQU   $10000       Start of cache RAM
ENCACH    EQU   $107FF       End of cache RAM
GPIP      EQU   $10800       General purpose interface port
AER       EQU   $10801       Active edge register
DDR       EQU   $10802       Data direction register
IERA      EQU   $10803       Int. enable A reg.
IERB      EQU   $10804       Int. enable B reg.
IPRA      EQU   $10805       Int. pending A reg.
IPRB      EQU   $10806       Int. pending B reg.
IMRA      EQU   $10809       Int. mask A reg.
IMRB      EQU   $1080A       Int. mask B reg.
TCDCR     EQU   $1080E       Timer C & D cntrl reg.
TDDR      EQU   $10812       Timer D data reg.
USR       EQU   $10814       UART status reg.
RSR       EQU   $10815       Receive status
TSR       EQU   $10816       Transmit status
UDR       EQU   $10817       UART data reg.
PPRT      EQU   $10820       Parallel port addr.
DPORT     EQU   $10824       DRASS status/control port
INSTR     EQU   $10830       Display instruction port
DSPREG    EQU   $10831       Display data port
LPBAK     EQU   $07          Serial loopback command
*
*
          SECTION 1
*
*         INITIALIZE PROCESSOR PARAMETERS
*
START     MOVE.L #ISTACK,A7     Init. stack pointer
          MOVE.W #SWORD0,SR     Init. system status word
          MOVE.L #VBASE,A0
          MOVEC  A0,VBR         Init. vector base reg.
          MOVE.L #ISTACK,A7
          MOVEC  A7,ISP         Init. stack reg.
          MOVE.L #CEN,A0        Enable cache
          MOVEC  A0,CACR
          CLR.L  D0
          MOVE.W D0,DSTAT       Initialize system variables
          MOVE.W D0,LSTCMD
          MOVE.W D0,CONTFL
```

```
         MOVE.W  D0,CMD
         MOVE.W  D0,STRTFL
         MOVE.W  D0,LSTSEL
         MOVE.W  D0,TCNT
         MOVE.W  D0,SERST
         MOVE.B  #0,IMRA          Clear int. A mask
         MOVE.B  #$4E,IMRB        Enable button interrupt mask
         MOVE.B  #0,IERA          Disable A interrupts
         MOVE.B  #$4E,IERB        Enable button interrupts
         MOVE.B  #$1E,AER         Set active edge rising
         MOVE.B  #$E0,DDR         Set data direction on input
         MOVE.B  #$FF,IPRA        Set int. A pending
         MOVE.B  #$B1,IPRB        Reset button int. pending
         MOVE.B  #$22,STAT        Initialize status port
         MOVE.B  STAT,DPORT
         MOVE.B  #05,TCDCR        Initialize baud rate to 1200
         MOVE.B  #01,TDDR
         MOVE.B  #$BE,USR         Initialize UART
         MOVE.B  #01,RSR          Set receiver ready
*
*        PERFORM POWER UP DIAGNOSTICS
*
         BSR     ROMTST           Perform ROM test
         BSR     SERTST           Perform serial test
         BSR     PARLTST          Perform parallel test
         BSR     LITTST           Perform light test
         MOVE.L  PRESYNC,D0       Check for memory full
         ANDI.L  #$FFFFFF00,D0
         CMPI.L  #$FAF35000,D0
         BEQ.S   CONTST1
         CMPI.L  #$FAF32000,D0
         BNE.S   CONTST
CONTST1  BSET    #3,STAT          If full- set status
         BRA.S   DSPERR           And skip memory test
CONTST   BSR     RAMTST           Perform Ram test
DSPERR   BSET    #6,STAT          Set passed status
         CMPI.W  #0,DSTAT         If diag. passed
         BEQ.S   ENADRAS          Skip error display
         BCLR    #6,STAT          Set failed status
         BSET    #4,STAT
         LEA.L   DMSG,A1          Build and display error msg
         LEA.L   DSPBUF,A0
         MOVE.L  (A1)+,(A0)+
         MOVE.L  (A1)+,(A0)+
         MOVE.L  (A1)+,(A0)+
         MOVE.L  (A1),(A0)
         LEA.L   DSPBUF+9,A0
         BTST    #0,DSTAT         Check for ROM error
         BEQ.S   ERR1
         MOVE.B  #$52,(A0)+       Move R into msg
         ADDQ.L  #1,A0
ERR1     BTST    #1,DSTAT         Check for serial error
         BEQ.S   ERR2
         MOVE.B  #$53,(A0)+       Move S into msg
         ADDQ.L  #1,A0
ERR2     BTST    #2,DSTAT         Check for parallel error
```

```
            BEQ.S   ERR3
            MOVE.B  #$50,(A0)+          Move P into msg
            ADDQ.L  #1,A0
ERR3        BTST    #3,DSTAT            Check for memory error
            BEQ.S   ERRCON
            MOVE.B  #$4D,(A0)+          Move M into msg
            ADDQ.L  #1,A0
ERRCON      LEA.L   DSPBUF,A0           Display error msg
            BSR     DISPLAY
ENADRAS     MOVE.B  STAT,DPORT          Output status word to lights
            MOVE.W  #SWORD1,SR          Enable interrupts
            MOVE.W  #$FF,LSTCMD         Init. last command flag
            CMPI.B  #0,DSTAT            If no power up errors
            BEQ.S   MAINLP                continue
CONTWAT     CMPI.B  #0,CONTFL           Else wait for continue button
            BEQ.S   CONTWAT               to be pressed
            BCLR    #4,STAT             Clear failed status
            MOVE.B  STAT,DPORT
            MOVE.W  #0,CONTFL           Init. button flags
            MOVE.W  #0,CMD
MAINLP      CMP.B   #0,HLTFLG           Halt button pressed?
            BEQ.S   MAINLP1             No - continue
            MOVE.W  #0,CONTFL           Else clear continue flag
            MOVE.W  #$FF,LSTCMD         Reset last cmd flg
            MOVE.W  #0,CMD              Clear last command
            BCLR    #5,STAT             Clear busy
            MOVE.B  STAT,DPORT
MAINLP1     CMPI.W  #0,CMD              Is there a command pending?
            BNE.S   CMDPR               Yes - go process it
            MOVE.L  DPORT,D0            Else read switches
            ANDI.L  #$00017000,D0       Get thumbwheel and Run/test bits
            LSR.L   #8,D0
            BTST    #8,D0               Check state of Run/test switch
            BEQ.S   DSPSEL
            BCLR    #8,D0               Set bit accordingly
            BSET    #7,D0
DSPSEL      CMP.W   LSTCMD,D0           setting changed?
            BEQ     MAINLP1             No - continue
            MOVE.W  D0,LSTCMD           Else save state
            LEA.L   PRMPT,A0            Get display msg table
            ADDA.L  D0,A0
            BSR     DISPLAY             Display current thumbwheel setting
            BRA     MAINLP1             Continue
CMDPR       CMPI.W  #9,CMD              Command set for diagnostics
            BGT     DIAGPR              Yes - go process diag setting
            CMPI.W  #1,CMD              Command set for download data?
            BNE.S   CMDPR1              no - go check next command
            BSR     DLDATA              Else download data
            MOVE.W  #0,CMD              Reset command
            MOVE.W  #$FF,LSTCMD
            BRA     MAINLP              Return
CMDPR1      CMPI.W  #2,CMD              Check for down load DCOM data
```

```
         BNE.S   CMDPR2          If not check next command
         BSR     DLDCOM          Else perform download
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
CMDPR2   CMPI.W  #3,CMD          Command set for serial output?
         BNE.S   CMDPR3          No - go check next command
         BSR     OUTDATA         Else go output data serially
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
CMDPR3   CMPI.W  #4,CMD          Command set for clear memory?
         BNE     INVLCMD         No - display error
         BSR     CLRMEM          Else go clear memory
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
DIAGPR   CMPI.W  #10,CMD         Command = RAM diag.?
         BNE.S   DIAGPR1         No - check next command
         BSR     RAMDIAG         Else perform RAM diag.
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
DIAGPR1  CMPI.W  #11,CMD         Command = serial diag.?
         BNE.S   DIAGPR2         No - check next command
         BSR     SERDIAG         Else perform serial diag.
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
DIAGPR2  CMPI.W  #12,CMD         Command = display diag.?
         BNE.S   DIAGPR3         No - check next command
         BSR     DSPDIAG         Else perform display diag
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
DIAGPR3  CMPI.W  #13,CMD         Command = parallel diag.?
         BNE.S   DIAGPR4         No - check next command
         BSR     PARDIAG         Else perform parallel diag
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
DIAGPR4  CMPI.W  #14,CMD         Command = Light diag?
         BNE.S   DIAGPR5         No - check next command
         BSR     LITDIAG         Else perform light diag.
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
DIAGPR5  CMPI.W  #15,CMD         Command = switch diag.?
         BNE.S   INVLCMD         No - display error
         BSR     SWTDIAG         Else perform switch diag
         MOVE.W  #0,CMD          Reset command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP          Return
INVLCMD  MOVE.W  #0,CMD          Clear invalid command
         MOVE.W  #$FF,LSTCMD
         BRA     MAINLP
```

```
*
*************************************************************************
*
*          Button interrupt service routine
*
*************************************************************************
*
BUTINT     MOVEM.L D0-D3,-(A7)        Save register
           MOVE.L #2000,D3
BUDELAY    MULU   D2,D2
           SUB.L  #1,D3
           BNE.S  BUDELAY
           MOVE.W #10,D3              Set loop counter
           MOVE.L DPORT,D0            Read switches
           MOVE.L D0,D1               Save switch settings
READBUT    MOVE.L DPORT,D2            Read switches
           AND.L  D2,D1               Mask off false settings
           SUB.W  #1,D3               Do 10 times
           BNE.S  READBUT
RSTWAT     MOVE.L DPORT,D2            Read switches
           ANDI.L #$000E0000,D2       Mask off all but buttons
           AND.L  D1,D2               Wait till all buttons are reset
           BNE.S  RSTWAT
           MOVE.L #5000,D3            Set delay counter
BUTWAT     MULU   D2,D2               Wait for things to settle down
           SUB.L  #1,D3
           BNE.S  BUTWAT
           BTST   #19,D1              Check for halt button
           BNE.S  HLTBUT              Yes - go process it
           BTST   #18,D1              Check for continue button
           BNE.S  CONTBUT             Yes - continue
           BTST   #17,D1              Check for toggle switch
           BEQ.S  BUTRET              Yes - return
           MOVE.B #$FF,STRTFL         Else must be start button
           MOVE.W #1,CMD              Set start command
           BTST   #16,D0              Toggle switch in Run mode?
           BEQ.S  THMCHK              Yes - go build command
           MOVE.W #10,CMD             Else init for test mode
THMCHK     ANDI.L #$00007000,D0       mask off thumbwheel setting
           LSR.L  #8,D0
           LSR.L  #4,D0
           ADD.W  D0,CMD              Add setting to command
           MOVE.W #0,CONTFL           Clear continue flag
           BRA.S  BUTRET              Return
HLTBUT     MOVE.B #$FF,HLTFLG         Set halt flag
           BRA.S  BUTRET
CONTBUT    MOVE.B #$FF,CONTFL         Set continue flag
BUTRET     MOVEM.L (A7)+,D0-D3        Restore register
           MOVE.B #$B1,IPRB           Clear interrupt pending
           RTE                        Return
*
```

```
BUSERR   NOP                                 Buss error processor
         RTE
ADDERR   NOP                                 Address error processor
         RTE
ERRVEC   NOP                                 Error interrupt processor
         RTE
PRIVER   NOP                                 Privelege error processor
         RTE
SPURINT  NOP                                 Spurious interrupt processor
         RTE
AVEC     NOP                                 Misc. auto vector interrupt
         RTE
*
*
*        CONSTANTS DEFINITION
*
*
DMSG     DC.B    'DIAG ERR:         '
PRMPT    DC.B    'DOWN LOAD ADAM    '
         DC.B    'DOWN LOAD DCOM    '
         DC.B    'SERIAL TRANSFER   '
         DC.B    'CLEAR MEMORY      '
         DC.B    '      ERROR       '
         DC.B    '      ERROR       '
         DC.B    '      ERROR       '
         DC.B    '      ERROR       '
         DC.B    '   MEMORY TEST    '
         DC.B    '   SERIAL TEST    '
         DC.B    '   DISPLAY TEST   '
         DC.B    '  PARALLEL TEST   '
         DC.B    '    LIGHT TEST    '
         DC.B    '   SWITCH TEST    '
         DC.B    '      ERROR       '
         DC.B    '      ERROR       '
TEXT1    DC.B    '@ABCDEFGHIJKLMNO'
TEXT2    DC.B    'PQRSTUVWXYZ['
         DC.B    $5C
         DC.B    ']^_'
TEXT3    DC.B    '`abcdefghijklmno'
TEXT4    DC.B    'qrstuvwxyz[;]'
         DC.B    $7E,$7F
TEXT5    DC.B    ' !"#$%'
         DC.B    $26,$27
         DC.B    '()*+,-./'
TEXT6    DC.B    '0123456789:;<=>?'
TEXT7    DC.L    $FFFFFFFF
         DC.L    $FFFFFFFF
         DC.L    $FFFFFFFF
         DC.L    $FFFFFFFF
TERRMSG  DC.B    ' TRANSFER ERROR '
NDATMSG  DC.B    'NO DATA PRESENT '
SERERM   DC.B    'SERIAL DATA ERR '
```

```
         DC.B    ' FRAME ERROR     '
         DC.B    '  PARITY ERROR   '
         DC.B    'OVERRUN ERROR    '
         DC.B    'TRANSMIT TIMEOUT'
         DC.B    'RECEIVE TIMEOUT '
DRAMEM   DC.B    'DRAM ADR:        '
CRAMEM   DC.B    'CRAM ADR:        '
MFMSG    DC.B    ' DATA RAM FULL   '
ERRMSG   DC.B    '      ERROR      '
BAUDMSG  DC.B    '       50        '
         DC.B    '       75        '
         DC.B    '       150       '
         DC.B    '       200       '
         DC.B    '       300       '
         DC.B    '       600       '
         DC.B    '      1200       '
         DC.B    '      2400       '
         DC.B    '      4800       '
         DC.B    '      9600       '
         DC.B    '      19200      '
         DC.B    '      ERROR      '
         DC.B    '      ERROR      '
         DC.B    '      ERROR      '
         DC.B    '      ERROR      '
         DC.B    '      ERROR      '
SERFMT   DC.B    ' 7 BIT / NO PAR '
         DC.B    '7 BIT / EVEN PAR'
         DC.B    ' 7 BIT / ODD PAR'
         DC.B    ' 8 BIT / NO PAR '
         DC.B    '8 BIT / EVEN PAR'
         DC.B    ' 8 BIT / ODD PAR'
         DC.B    '      ERROR     '
         DC.B    '      ERROR     '
DFMT     DC.B    ' BINARY TRANSFER'
         DC.B    ' ASCII TRANSFER '
PRESC    DC.B    5
         DC.B    5
         DC.B    5
         DC.B    5
         DC.B    5
         DC.B    5
         DC.B    5
         DC.B    3
         DC.B    3
         DC.B    1
         DC.B    1
         DC.B    0
         DC.B    0
         DC.B    0
         DC.B    0
         DC.B    0
CNTTBL   DC.B    $18
         DC.B    $10
         DC.B    8
         DC.B    6
```

```
         DC.B    4
         DC.B    2
         DC.B    1
         DC.B    2
         DC.B    1
         DC.B    2
         DC.B    1
         DC.B    0
         DC.B    0
         DC.B    0
         DC.B    0
         DC.B    0
FMTTBL   DC.B    $A8
         DC.B    $AE
         DC.B    $AC
         DC.B    $88
         DC.B    $8E
         DC.B    $8C
         DC.B    0
         DC.B    0
ENDROM   DS.W    1
*
*        DRASS STORAGE DEFINITION
*
         OFFSET $107D000
*
PREBUF   DS.B    4096          Pre cal buffer
PREEND   EQU     *
POSTBU   DS.B    4096          Post cal buffer
POSTEND  EQU     *
STSCT    DS.B    512           Scan table
ENDSCT   DS.L    1
BLKLEN   DS.L    1             Block length
UCNTRL   DS.B    1             ADAM UART control
BAUD     DS.B    1             UART Baud
PRSCA    DS.B    1             ADAM Prescale A
PRSCB    DS.B    1               "      "     B
PRSCCD   DS.B    1               "      "     C & D
FLCNTA   DS.B    1             ADAM filter clock A counter
FLCNTB   DS.B    1               "     "      "   B    "
FLCNTC   DS.B    1               "     "      "   C    "
FLCNTD   DS.B    1               "     "      "   D    "
PRESYNC  DS.L    1
DATSYNC  DS.L    1
POSSYNC  DS.L    1
LSTPTR   DS.L    1
ENDBUF   DS.L    1
*
         OFFSET $10000
```

```
*
DSTAT     DS.W   1          Diagnostic status
STAT      DS.B   1          Test status
DSPBUF    DS.L   4          Display buffer
LSTCMD    DS.W   1          Last command
CONTFL    DS.B   1          Continue flag
HLTFLG    DS.B   1          Halt flag
CMD       DS.W   1          Current command
STRTFL    DS.B   1          Start flag
SUMCHK    DS.B   1          Sum check for down load data
BLKSTAT   DS.B   1          Block status for download
LSTSEL    DS.W   1          Last thumbwheel selection
LSTSWT    DS.L   1          Last switch selected
TCNT      DS.W   1          Test counter
RETRY     DS.W   1          Retry counter for download
DATCNT    DS.W   1
SERST     DS.W   1          Serial status
          END
```

```
*
*****************************************************************************
*
*                                      CLRMEM
*
*    CLRMEM purges the data from the RAM modules in DRASS. The data is
*    erased from address 1000000 (STRAM) through 107EFFE (ENRAM) and
*    the sync code for precal, postcal, and test data. Before exiting,
*    the memory full status is reset.
*
*         Input parameter : None
*         Output parameter : None
*         Registers used : A0 - start of memory
*                          A1 - end of memory
*                          D0 - value that is written to memory (0)
*
*****************************************************************************
*
CLRMEM    IDNT    1,1
STRAM     EQU     $1000000          Start of memory
ENRAM     EQU     $107EFFE          End of test memory
DPORT     EQU     $10824            Status port
*
          INCLUDE DRASDEF
          XREF    WDTST
          XDEF    CLRMEM
*
*
CLRMEM    BCLR    #4,STAT           Clear pass
          BCLR    #6,STAT           Clear fail
          BSET    #5,STAT           Set busy status
          MOVE.B  STAT,DPORT
          CLR.L   D0                Init to 0 for writting to memory
          LEA.L   STRAM,A0          Load start addr. of memory
          LEA.L   ENRAM,A1          Load end address of memory
          BSR     WDTST             Clear memory
          MOVE.L  D0,PRESYNC        Clear precal sync code
          MOVE.L  D0,DATSYNC        Clear test data sync code
          MOVE.L  D0,POSSYNC        Clear post cal sync code
          BCLR    #3,STAT
          BCLR    #5,STAT
          BSET    #6,STAT           Set passed status
          MOVE.B  STAT,DPORT
          RTS
          END
```

```
************************************************************************
*
*                               PARDIAG
*
*    PARDIAG performs the DRASS parallel diagnostic. It executes in
*    conjunction with the ADAM parallel diagnostic. When ADAM sends
*    a data pattern over the parallel port to the DRASS, it echoes
*    that data back. The test starts when the sync code FAF30001 is
*    received. The test is exited when the Halt button is detected
*    (HLTFLG). Since timeouts are built into the ADAM parallel
*    diagnostic, DRASS's diagnostic must be activated first.
*
*          Input parameters : HLTFLG - halt the test when set
*
*          Output parameters : STAT - bit 5 - busy status
*
*          Registers used : D5 - parallel port handshake status
*                           D6 - input/output data
*
************************************************************************
*
PARDIAG    IDNT    1,1
DPORT      EQU     $10824              DRASS status/control port
PPRT       EQU     $10820              Parallel port address
*
           INCLUDE DRASDEF
           XDEF    PARDIAG
*
PARDIAG    BCLR    #1,STAT
           BSET    #2,STAT             Set parallel port to input
           BSET    #5,STAT             Set busy light
           BCLR    #4,STAT
           BCLR    #6,STAT
           MOVE.B  STAT,DPORT
           MOVE.B  #0,HLTFLG           Clear halt flag
           MOVE.L  PPRT,D6             Reset parallel port
CONCHK     CMP.B   #0,HLTFLG           Halt pressed?
           BNE     PARDEXT             Yes - exit
           MOVE.L  DPORT,D5
           BTST    #23,D5              ADAM connected?
           BNE.S   CONCHK              No - go wait
ADMRDY     CMP.B   #0,HLTFLG           Halt pressed?
           BNE     PARDEXT             Yes - exit
           MOVE.L  DPORT,D6
           BTST    #21,D6              Buffer full set?
           BEQ.S   ADMRDY              No - go wait
           MOVE.L  PPRT,D5             Else read parallel data
           CMP.L   #$FAF30001,D5       Sync code ?
```

```
         BEQ.S   TSTRST              Yes - reset data pattern
         CMP.L   D0,D5               Data error?
         BNE.S   DATERR              Yes - go set error
         CMP.L   #$FFFFFFFF,D0       End of data pattern?
         BEQ.S   SNDBAK              Yes - sen ack back
         ADD.L   #$01010101,D0       Incr data pattern
         BRA     ADMRDY              Go get next transmission
TSTRST   CLR.L   D0                  Zero data pattern
         BRA     ADMRDY              Get data from adam
DATERR   BSET    #4,STAT             Set error status
         BRA     ADMRDY              Get rest of data
SNDBAK   CMP.B   #0,HLTFLG           Halt pressed ?
         BNE     PARDEXT             Yes - exit
         MOVE.L  DPORT,D6
         BTST    #20,D6              ADAM in input mode?
         BEQ.S   SNDBAK              No - wait for mode change
ADMBSY   CMP.B   #0,HLTFLG           Halt pressed?
         BNE.S   PARDEXT             Yes - exit
         MOVE.L  DPORT,D6
         BTST    #22,D6              ADAM ready to receive?
         BNE.S   ADMBSY              No - go wait
         BCLR    #2,STAT             Set DRASS to output mode
         MOVE.B  STAT,DPORT
         BTST    #4,STAT             Test error?
         BEQ.S   PRLOK               No - send ack
         MOVE.L  #$FAF30002,PPRT     Else sen nak
         BRA.S   ADMBSY1             And exit
PRLOK    MOVE.L  #$FAF30001,PPRT     Send ack
ADMBSY1  CMP.B   #0,HLTFLG           Halt pressed?
         BNE.S   PARDEXT             Yes - exit
         MOVE.L  DPORT,D6
         BTST    #22,D6              ADAM ready to receive?
         BNE.S   ADMBSY1             No go wait
         BSET    #2,STAT             Set DRASS to input mode
         MOVE.B  STAT,DPORT
         BTST    #4,STAT             Test error?
         BNE.S   PARDEXT             Yes - exit
         CMP.B   #0,HLTFLG           Halt pressed ?
         BEQ     ADMRDY              No - continue test
PARDEXT  BCLR    #5,STAT             Clear busy light
         BSET    #1,STAT             Set DRASS offline
         BSET    #2,STAT             Set DRASS to input mode
         BTST    #4,STAT             Test error set?
         BNE.S   PARDX1              Yes - return
         BSET    #6,STAT             Else set passed
PARDX1   MOVE.B  STAT,DPORT
         RTS                         Return
         END
```

```
*************************************************************************
*
*                          DISPLAY
*
*    DISPLAY updates the 16 character display on the front panel of
*    the DRASS. The message that is output is pointed to by register
*    A0. 16 characters are always output. Each time DISPLAY is called,
*    the display is initialized before the characters are output.
*
*         Input parameters : Register A0 - points to the message buffer
*
*         Output parameters : None
*
*         Registers used : D0 - temporary delay counter
*                          D6 - character counter
*
*************************************************************************
*
DISPLAY   IDNT    1,1
*
INSTR     EQU     $10830             Display instruction register
DSPREG    EQU     $10831             Display data register
*
          XDEF    DISPLAY
*
DISPLAY   MOVE.B #$38,INSTR          Initialize display
          BSR.S   PAUSE1
          MOVE.B #6,INSTR
          BSR.S   PAUSE1
          MOVE.B #$0E,INSTR
          BSR.S   PAUSE1
          MOVE.B #01,INSTR
          BSR.S   PAUSE2
          MOVE.B #02,INSTR
          BSR.S   PAUSE2
DSPMES    MOVE.L #16,D6              Set character counter
SNDTXT    MOVE.B (A0)+,DSPREG        Output ASCII character
          BSR.S   PAUSE1             Delay
          SUB.L   #1,D6              Decr. character counter
          CMP.L   #8,D6              If not 8
          BNE.S   ENDOFTXT           Continue
          MOVE.B #$C0,INSTR          Else set display to second half
          BSR.S   PAUSE1             Delay
ENDOFTXT  CMP.L   #0,D6              If last character
          BNE.S   SNDTXT
          RTS                        Return
*
*
PAUSE1    MOVE.L #$130,D0
          BRA.S   LOOP
PAUSE2    MOVE.L #$3000,D0
LOOP      SUBQ.L #1,D0
          BNE.S   LOOP
          RTS
          END
```

```
***********************************************************************
*
*                              DSPDIAG
*
*    DSPDIAG performs the diagnostic to the front panel display. It
*    outputs a series of test patterns (TEXT1 thru TEXT7) to the
*    display until the halt button is detected (HLTFLG). There is a
*    3 second delay between outputs of each test pattern. DISPLAY
*    performs the actual display.
*
*         Input parameters : none
*         Output parameters : STATUS - bit 5 - busy status
*
*         Registers used : A0 - pointer to the test pattern
*                          D0 - temporary delay counter
*                          D2 - saved pointer to the test pattern
*
***********************************************************************
*
DSPDIAG   IDNT    1,1
*
INSTR     EQU     $10830             Instruction reg. for the display
DSPREG    EQU     $10831             Display data reg.
DPORT     EQU     $10824             Control/status port
*
          INCLUDE DRASDEF
          XDEF    DSPDIAG
          XREF    DISPLAY
*
DSPDIAG   BSET    #5,STAT            Set busy light
          BCLR    #4,STAT
          BCLR    #6,STAT
          MOVE.B  STAT,DPORT
          MOVE.B  #$38,INSTR         Initialize display
          BSR.S   PAUSE1
          MOVE.B  #$38,INSTR
          BSR.S   PAUSE1
          MOVE.B  #6,INSTR
          BSR.S   PAUSE1
          MOVE.B  #$0E,INSTR
          BSR.S   PAUSE1
DSPST     MOVE.W  #7,D1             Set message counter
          LEA.L   TEXT1,A0           Load first test pattern
          MOVE.L  A0,D2              Save start of test patterns
DSPLOP    BSR     DISPLAY            Dislay test message
          BSR.S   PAUSE3             Wait 3 sec.
          CMP.B   #0,HLTFLG          Halt pressed?
          BNE.S   DSPEXT             Yes - exit
          ADD.L   #16,D2             Incr. test pattern pointer
          MOVE.L  D2,A0              Store pointer
          SUB.W   #1,D1              Decr. message count
          BNE.S   DSPLOP             Continue if not last message
          BRA.S   DSPST              Else start over again
DSPEXT    BCLR    #5,STAT            Clear busy light
          MOVE.B  STAT,DPORT
          RTS
```

```
*
*
PAUSE1      MOVE.L #$130,D0
            BRA.S  LOOP
PAUSE2      MOVE.L #$3000,D0
            BRA.S  LOOP
PAUSE3      MOVE.L #$1FFFFF,D0
LOOP        SUBQ.L #1,D0
            BNE.S  LOOP
            RTS
            END
```

```
***************************************************************************
*
*                              LITTST
*
*    LITTST performs a test of the front panel lights during power
*    up diagnostics. The test consists of sequencing each of the status
*    lights. Since there is no status from the lights, DSTAT is
*    not updated.
*
*         Input parameters : None
*         Output parameters : None
*         Registers used : D1 - output control byte
*                          D2 - bit set value for the control byte
*                          D0 - Temporary delay counter
*
***************************************************************************
*
LITTST    IDNT    1,1
*
DPORT     EQU     $10824               DRASS status/control port
*
          INCLUDE DRASDEF
          XDEF    LITTST
*
LITTST    MOVE.L  #2,D2                Initialize for setting bit 2
          CLR.L   D1                   Clear save bit reg.
LITLP     BSET    D2,D1                Set bit contained in D2
          MOVE.L  D1,DPORT             Turn on light
          BSR.S   DELAY                Delay 1 sec
          ADD.L   #1,D2                Incr bit designator
          CMP.L   #7,D2                End of test?
          BLT.S   LITLP                No - continue
LITEXT    BCLR    #4,STAT              Clear status
          BCLR    #5,STAT
          MOVE.B  STAT,DPORT
          RTS                          Return
*
*
DELAY     MOVE.L  #$1FFFF,D0
DELOP     SUBQ.L  #1,D0
          BNE.S   DELOP
          RTS
          END
```

```
************************************************************************
*
*                               ROMTST
*
*    This routine calculates a sumcheck of the PROM and compares
*    that value with the sumcheck stored in PROM at address FFFF.
*    If the test fails,bit 0 in DSTAT is set to be checked after
*    power up diagnostics is complete.
*
*         Input parameters : sumcheck contained at address FFFF
*
*         Output parameters : DSTAT - contains the pass/fail
*                                     result of the test
*
*         Registers used : A0 - running index through PROM
*                          D0 - end address of PROM
*                          D1 - running checksum total
*
************************************************************************
*
ROMTST     IDNT     1,1
PROM       EQU      0                  Starting address of PROM
           INCLUDE  DRASDEF
           XDEF     ROMTST
*
ROMTST     LEA.L    PROM,A0            Load starting address
           MOVE.L   #$FFFF,D0          Load ending address
           CLR.L    D1                 Clear checksum
           CLR.L    D7
ROMLP      MOVE.B   (A0)+,D7           Get byte from PROM
           EOR.B    D7,D1              Exclusive or it to checksum
           CMPA.L   D0,A0              If not end of PROM
           BNE.S    ROMLP              Continue
           MOVE.W   #0,CCR             Else clear carry
           CMP.B    (A0),D1            Compare checksums
           BEQ.S    ROMPAS             If = exit
           BSET     #0,DSTAT           Else set error bit
ROMPAS     RTS
           END
```

```
*******************************************************************************
*
*                              SERTST
*
*    This routine performs the power up diagnostic on the UART in the
*    MFP. It performs an internal loop back test with a canned message.
*    SERTST does not utilize the receive interrupt but it does test
*    the receive status. The data format and Baud rate used during
*    the test is the same that's set up during system initialization.
*
*          Input parameters : none
*
*          Output parameters : DSTAT - power up diagnostic status
*                              SERST - 1 - frame error
*                                      2 - parity error
*                                      4 - overrun error
*
*          Registers used : A0 - index to test pattern
*                           D0 - Test data character
*                           D1 - Temporary UART status
*                           D7 - Character count
*
*******************************************************************************
*
SERTST   IDNT    1,1
IMRA     EQU     $10809              Interrupt mask reg.
RSR      EQU     $10815              Receiver status
TSR      EQU     $10816              Transmit status
UDR      EQU     $10817              UART  data reg.
LPBAK    EQU     07                  Loopback command
TXENA    EQU     $25                 Transmit enable command
         INCLUDE DRASDEF
         XDEF    SERTST
*
SERTST   NOP
         MOVE.B #LPBAK,TSR           Set UART to loopback mode
         LEA.L   TSTPAT,A0           Load test pattern
         MOVE.L #$0A,D7              Set character count
SERLP    MOVE.B (A0),D0              Get character
         BSR.S   XMIT                Output it
         BSR.S   RECV                Read it back
         CMP.B   (A0)+,D0            Compare the two
         BNE.S   SERERR              Branch if not equal
         CMPI.W #0,SERST             If error status set
         BNE.S   SERERR              Go to error exit
         DBNE    D7,SERLP            Else decr. character count
         MOVE.B #TXENA,TSR           If done - enable transmitter
         RTS                         And return
SERERR   MOVE.B #TXENA,TSR           Else enable transmitter
         BSET    #1,DSTAT            Set error status
         RTS
```

```
*
XMIT     BTST    #7,TSR       Transmit ready
         BEQ     XMIT         No - wait
         MOVE.B  D0,UDR       Else output character
         RTS
*
RECV     MOVE.W  #0,SERST     Clear status
RECLP    MOVE.B  RSR,D1       Buffer full ?
         BTST    #7,D1
         BEQ     RECLP        No - wait
         MOVE.B  UDR,D0       Else read character
FRME     BTST    #4,D1        Frame error set?
         BEQ.S   PARE         No - check parity
         BSET    #0,SERST     Else set frame error stat
PARE     BTST    #5,D1        Parity error?
         BEQ.S   OVRE         No - check overrun
         BSET    #1,SERST     Else set error
OVRE     BTST    #6,D1        Overrun set?
         BEQ.S   RECRET       No - go return
         BSET    #2,SERST     Else set overrun
RECRET   RTS
TSTPAT   DC.B    $30          Test pattern
         DC.B    $35
         DC.B    $40
         DC.B    $45
         DC.B    $3A
         DC.B    $4F
         DC.B    $55
         DC.B    $11
         DC.B    $22
         DC.B    $0C
         END
```

```
***********************************************************************
*
*                              PARLTST
*
*    arltst performs the power up diagnostic on the parallel port.
*    It outputs a data pattern to the port and reads and compares
*    that value. Since this is an internal test, the port is set
*    to output mode during the entire test. The data pattern that
*    is used is : 00000000, 01010101,... 0F0F0F0F.
*
*         Input parameters : none
*
*         Output parameters : DSTAT - bit 2 is set if there is a test
*                                     failure
*
*         Registers used : D1 - contains the data pattern
*                          D2 - test counter
*                          D7 - input value
*                          D0 - temporary register
*
***********************************************************************
*
PARLTST  IDNT    1,1
*
         INCLUDE DRASDEF
         XDEF    PARLTST
*
PPRT     EQU     $10820                  Parallel port address
DPORT    EQU     $10824                  DRASS status/control port
*
PARLTST  MOVE.L  PPRT,D0                 Reset parallel port
         BCLR    #2,STAT                 Set to output mode
         MOVE.B  STAT,DPORT
         CLR.L   D1                      Initialize data pattern
         MOVE.W  #16,D2                  Set test counter
PARLOP   MOVE.L  D1,PPRT                 Output test pattern
         MOVE.L  PPRT,D7                 Read parallel port
         CMP.L   D1,D7                   Compare data
         BNE.S   PARERR                  Set error if not equal
         ADDI.L  #$01010101,D1           Else incr. test pattern
         SUBI.W  #1,D2                   Decr test counter
         BNE.S   PARLOP                  If not finished continue
         BSET    #2,STAT                 Set to input mode
         MOVE.B  STAT,DPORT
         RTS                             Else return
PARERR   BSET    #2,DSTAT                Set port to input mode
         BSET    #2,STAT                 Set to input mode
         MOVE.B  STAT,DPORT
         RTS
         END
```

```
**********************************************************************
*
*                          RAMTST
*
*    This routine test both the SRAM and Cache RAM in the system
*    during power up diagnostic. It performs a byte write and read
*    of the momory with the data patterns AA,55,FF, and 00. The
*    memory that is tested is from 1000000 (STRAM) through 107FFFE
*    (ENRAM) and 10100 (STCACH) through 10700 (ENCACH). This
*    prevents the destruction of system parameters and the system
*    stack. RAMTST uses the routine WDTST to do the actual memory
*    accesses.
*
*         Input parameters : D7 - test status returned by WDTST
*
*         Output parameters : DSTAT - test status for power up diag.
*
*         Registers used : A0 - start of memory to test
*                          A1 - end of memory to test
*                          D0 - test data pattern
*                          D7 - test status from WDTST
*
**********************************************************************
*
RAMTST    IDNT    1,1
          INCLUDE DRASDEF
          XREF    WDTST
          XDEF    RAMTST
STRAM     EQU     $1000000          Start of static RAM
ENRAM     EQU     $107FFFE          End of RAM
STCACH    EQU     $10100            Start of Cache RAM
ENCACH    EQU     $10700            End of tested Cache RAM
*
RAMTST    CLR.L   D7                Clear status
          LEA.L   STRAM,A0          Load start of memory address
          LEA.L   ENRAM,A1          Load end of memory addr.
          MOVE.B  #$AA,D0           Load test pattern
          BSR     WDTST             Call test
          CMP.W   #0,D7             Test failed?
          BNE.S   RAMERR            Yes - set error
          MOVE.B  #$55,D0           Load test pattern
          BSR     WDTST             Call test
          CMP.W   #0,D7             Test failed?
          BNE.S   RAMERR            Yes - set error
          MOVE.B  #$FF,D0           Load test pattern
          BSR     WDTST             Call test
          CMP.W   #0,D7             Test failed?
          BNE.S   RAMERR            Yes - set error
          CLR.W   D0                Load test pattern
          BSR     WDTST             Call test
          CMP.W   #0.D7             Test failed?
```

```
         BEQ.S   CACHTST      No - do Cache test
RAMERR   BSET    #3,DSTAT     Else set error bit
CACHTST  CLR.L   D7           Clear error status
         LEA.L   STCACH,A0    Load start of Cache addr.
         LEA.L   ENCACH,A1    Load end of Cache addr
         MOVE.B  #$AA,D0      Load test pattern
         BSR     WDTST        Call test
         CMP.W   #0,D7        Test failed?
         BNE.S   CACHERR      Yes - set error
         MOVE.B  #$55,D0      Load test pattern
         BSR     WDTST        Call test
         CMP.W   #0,D7        Tets failed?
         BNE.S   CACHERR      Yes - set error
         MOVE.B  #$FF,D0      Load test pattern
         BSR     WDTST        Call test
         CMP.W   #0,D7        Test failed?
         BNE.S   CACHERR      Yes - set error
         CLR.L   D0           Load test pattern
         BSR     WDTST        Call test
         CMP.W   #0,D7        Test failed?
         BEQ.S   RAMEXT       No - exit
CACHERR  BSET    #4,DSTAT     Else set error
RAMEXT   RTS
         END
```

```
**********************************************************************
*
*                                WDTST
*
*     This routine is used by RAMTST, RAMDIAG, and CLRMEM to write
*     a data pattern to memory and check it. WDTST does a byte write
*     and read of the data in register D0 to the address in register
*     A2 up to the address in A1. If there are any errors in the
*     compare, register D7 is set to FF, and register A2 contains
*     the error address.
*
*          Input parameters : Register A0 - start of test memory
*                                 "     A1 - end of test memory
*                                 "     D0 - test data pattern (byte)
*
*          Output parameters : Register A2 - memory error address
*                                  "     D0 - test pattern written
*                                  "     D1 - data read from memory
*                                  "     D7 - error flag (set to FF
*                                              if failed)
*******************************************************************
*
WDTST      IDNT    1,1
           XDEF    WDTST
*
WDTST      MOVE.L  A0,A2               Save start address
WDWRT      MOVE.B  D0,(A2)+            Store data pattern
           CMPA.L  A2,A1               Check for end of memory
           BCC.S   WDWRT               Loop if no
           MOVE.L  A0,A2               Get start fo memory
WDRD       MOVE.B  (A2)+,D1            Read data from memory
           CMP.B   D0,D1               Compare data
           BNE.S   WDERR               Branch if error
           CMPA.L  A1,A2               Check for end of memory
           BLE.S   WDRD                Loop if not end
           RTS
WDERR      SUBA.L  #1,A2               Adjust to error address
           MOVE.B  #$FF,D7             Set error flag
           RTS
           END
```

```
**************************************************************************
*
*                               SWTDIAG
*
*    SWTDIAG performs the diagnostic for exercising the front panel
*    switches. This tests the toggle switch, push button switches,
*    and the thumbwheel switches. The toggle and push button switches
*    light the LED's when the state of the switch changes. The
*    thumbwheel switches display it's value on the front panel display.
*    This test is free running until the halt switch is detected. The
*    push buttons and toggle switch utilizes the interrupts to detect
*    state changes. BUTINT indicates this change through CONTFL,
*    HLTFLG, and STRTFL.
*
*          Input parameters : CONTFL - set by BUTINT when the continue
*                                      button is pressed
*                             HLTFLG - set by BUTINT when the halt
*                                      button is pressed
*                             STRTFL - set by BUTINT when the start
*                                      button is pressed
*
*          Output parameters : None
*
*          Registers used : A0 - pointer to DSPBUF for displaying
*                                thumbwheel switches
*                           D1 - LED status
*                           D2 - switch input state
*                           D3 - display character for thumbwheel
*                                swithches
*
**************************************************************************
*
SWTDIAG   IDNT    1,1
*
DPORT     EQU     $10824               DRASS status/control port
*
          INCLUDE DRASDEF
          XDEF    SWTDIAG
          XREF    DISPLAY
*
SWTDIAG   MOVE.B #0,CONTFL             Clear all switch flags
          MOVE.B #0,HLTFLG
          MOVE.B #0,STRTFL
          MOVE.L #$20202020,D1         Initialize display buffer with spaces
          LEA.L   DSPBUF,A0
          MOVE.L  D1,(A0)+
          MOVE.L  D1,(A0)+
          MOVE.L  D1,(A0)+
          MOVE.L  D1,(A0)
          CLR.L   D1
          MOVE.L #0,LSTSWT             Clear last switch variable
          BSET    #1,D1                Initialize LED output byte
SWTLOOP   LEA.L   DSPBUF,A0
          ADDQ.L #2,A0
          MOVE.L  DPORT,D2             Read thumbwheel switches
          AND.L   #$17FFF,D2           Mask off other input bits
```

```
        CMP.L   LSTSWT,D2         Same as last time ?
        BEQ.S   NXTBIT1           Yes - check other switches
        MOVE.L  D2,LSTSWT         Update last setting
        ROL.W   #4,D2             Shift in each thumbwheel switch
        BCLR    #3,D2
        MOVE.B  D2,D3
        ANDI.B  #$0F,D3           Convert to ASCII
        ORI.B   #$30,D3
        MOVE.B  D3,(A0)+          Store in display buffer
        ADDQ.L  #2,A0
        ROL.W   #4,D2             Get next thumbwheel switch
        MOVE.B  D2,D3
        ANDI.B  #$0F,D3           Convert to ASCII
        ADDI.B  #$30,D3
        CMPI.B  #$39,D3
        BLE.S   NUM1
        ADDI.B  #7,D3
NUM1    MOVE.B  D3,(A0)+          Store in display buffer
        ADDQ.L  #2,A0
        ROL.W   #4,D2             Get next switch
        BCLR    #3,D2
        MOVE.B  D2,D3
        ANDI.B  #$0F,D3           Convert to ASCII
        ORI.B   #$30,D3
        MOVE.B  D3,(A0)+          Store in display buffer
        ADDQ.L  #2,A0
        ROL.W   #4,D2             Get last switch
        MOVE.B  D2,D3
        ANDI.B  #$0F,D3           Convert to ASCII
        ADDI.B  #$30,D3
        CMPI.B  #$39,D3
        BLE.S   NUM3
        ADDI.B  #7,D3
NUM3    MOVE.B  D3,(A0)           Store in display buffer
        LEA.L   DSPBUF,A0
        BSR     DISPLAY           Display thumbwheel switches
        BCLR    #3,D1
        BTST    #16,D2            Check toggle switch
        BEQ.S   NXTBIT1           If not set continue
        BSET    #3,D1             Else turn on LED
NXTBIT1 CMPI.B  #0,STRTFL         Start button pressed?
        BEQ.S   NXTBIT2           No - check next button
        MOVE.B  #0,STRTFL         Else Clear start flag
        BCHG    #4,D1             Alternate LED
NXTBIT2 CMPI.B  #0,CONTFL         Continue button pressed?
        BEQ.S   NXTBIT3           No - check next button
        MOVE.B  #0,CONTFL         Else clear contue flag
        BCHG    #5,D1             Alternate LED
NXTBIT3 MOVE.B  D1,DPORT          Output to LED's
        CMPI.B  #0,HLTFLG         Halt button pressed
        BEQ     SWTLOOP           No - continue test
        BCLR    #4,STAT           Else exit test
        BCLR    #5,STAT
        MOVE.B  STAT,DPORT
        RTS
        END
```

```
***************************************************************************
*
*                                   DLDATA
*
*    DLDATA transfers the test data from ADAM to the DRASS memory
*    through the parallel port. The data is transfered in a predefined
*    sequence, first the test parameters, then precal data, then test
*    data, and finally the post cal data. The data is transfered in
*    blocks starting with a sync code then the data followed by a
*    checksum. Th e size of the data blocks are determined by the
*    number of A/D channels specified for the test. The test parameters
*    though are received in one block. DLDATA calculates the checksum
*    of the data as it receives it and if it matches the expected
*    checksum, it responds with FAF3FF00. If there is a checksum
*    error, it will respond with FAF3FFXX. DLDATA will allow five
*    attempts at receiving a data block before it errors. The sync
*    code at the beginning of each data block indicates what kind
*    of data it is :
*                        FAF31000 - parameter block
*                        FAF32000 - precal data block
*                        FAF33000 - test data block
*                        FAF34000 - post cal data block
*                        FAF30000 - end of transmission
*                        FAF3FFXX - checksum
*
*    DLDATA saves the sync code of the first block of a new type of
*    data to determine the starting frame counter for that data.
*                        PRESYNC - precal sync code
*                        DATSYNC - test data sync code
*                        POSSYNC - post cal sync code
*
*        Input parameters : HLTFLG - set when the halt button
*                                    is pressed and will cause
*                                    transfer to exit
*
*        Output parameters : None
*
*        Registers used : A1 - start of data buffer
*                         A2 - end of data buffer
*                         A0 - display message pointer
*                         D1 - hand shaking status
*                         D2 - input data
*                         D3 - temporary data storage
*
***************************************************************************
*
DLDATA    IDNT    1,1
*
DPORT     EQU     $10824          DRASS status/control port
PPRT      EQU     $10820          Parallel port addr.
DATBUF    EQU     $1000000        Start of test data buffer
DATEND    EQU     $107CFFF        End of test data buffer
ENRAM     EQU     $107FFFE        End of static RAM
*
          INCLUDE DRASDEF
          XREF    DISPLAY
          XDEF    DLDATA
```

```
DLDATA   BTST    #3,STAT          Memory full?
         BNE     DATEND           Yes - exit
         MOVE.W  #0,CONTFL        Clear halt/continue flags
         MOVE.W  #5,RETRY         Init retry counter
         MOVE.L  #0,ENDBUF        init buffer pointer
         MOVE.B  #$0F,BLKSTAT     Init block transfer status
         MOVE.B  #0,SUMCHK        Init checksum
         BSET    #2,STAT          Set parallel port to input
         BCLR    #1,STAT          Set DRASS online
         BCLR    #4,STAT
         BCLR    #6,STAT
         BSET    #5,STAT          Set busy light on
         MOVE.B  STAT,DPORT
         MOVE.L  PPRT,D1          Reset parallel port
CONCHK   MOVE.L  DPORT,D1
         BTST    #23,D1           ADAM connected?
         BNE.S   CONCHK           No - wait
RDYCHK   MOVE.L  DPORT,D1
         BTST    #21,D1           Parallel buffer full?
         BEQ.S   RDYCHK           No - wait
         MOVE.L  PPRT,D2          Read data
         MOVE.L  D2,D3            Save data word
         ANDI.L  #$FFFFFF00,D3
         CMPI.L  #$FAF30000,D3    End of transmission code?
         BEQ     ENDTRN           Yes - go exit
         CMPI.L  #$FAF31000,D3    Parameter block code?
         BEQ.S   GETPAR           Yes - go read parameter data
         CMPI.L  #$FAF32000,D3    Precal data block code?
         BEQ.S   GETPRE           Yes - go read precal data
         CMPI.L  #$FAF33000,D3    Test data code?
         BEQ     GETDAT           Yes - go read test data
         CMPI.L  #$FAF34000,D3    Post cal block code?
         BEQ     GETPOST          Yes - go read post cal data
         CMPI.L  #$FAF3FF00,D3    Checksum code?
         BEQ     GETSUMC          Yes - go read & compare checksum
         CMP.B   #0,HLTFLG        Halt button pressed?
         BNE     ENDTRN           Yes - go exit
         BTST    #0,BLKSTAT       Paramaters transfered yet?
         BNE     RDYCHK           No - go wait for parameters
         MOVE.L  D2,D3            Else it must be data
         ADD.B   D3,SUMCHK        Calculate checksum
         LSR.L   #8,D3
         ADD.B   D3,SUMCHK
         LSR.L   #8,D3
         ADD.B   D3,SUMCHK
         LSR.L   #8,D3
         ADD.B   D3,SUMCHK
         CMPA.L  A1,A2            End of data buffer?
         BLT     RDYCHK           Yes - go get next transmission
         MOVE.L  D2,(A1)+         Else store data in buffer
         BRA     RDYCHK           Go get next transmission
```

```
*
GETPAR   BCLR    #0,BLKSTAT        Clear parameters received stat
         LEA.L   STSCT,A1          Get start of transfer buffer
         LEA.L   ENRAM,A2          Get end of buffer pointer
         MOVE.L  A1,LSTPTR         Save running pointer for start
         BRA     RDYCHK            Go get next transmission
*
GETPRE   BTST    #1,BLKSTAT        Pre cal data received yet
         BEQ     RDYCHK            Yes - ignore sync code
         BCLR    #1,BLKSTAT        Else clear precal received stat
         MOVE.L  D2,PRESYNC        Save first precal sync code
         LEA.L   PREBUF,A1         Get start of precal pointer
         LEA.L   PREEND,A2         Get end of precal buffer
         MOVE.L  A1,LSTPTR         Save running pointer
         BRA     RDYCHK            Go get next transmission
*
GETDAT   BTST    #2,BLKSTAT        Test data block receive yet?
         BEQ     RDYCHK            Yes - go get rest of data
         BCLR    #2,BLKSTAT        Else clear test data recv'd stat
         MOVE.L  D2,DATSYNC        Save first test data sync code
         LEA.L   DATBUF,A1         Get start of test data buffer
         LEA.L   DATEND,A2         Get end of test data buffer
         MOVE.L  A1,LSTPTR         Save running pointer
         BRA     RDYCHK            Go get next transmission
*
GETPOST  BTST    #3,BLKSTAT        Post cal data received yet?
         BEQ     RDYCHK            Yes - go get rest of data
         MOVE.L  A1,ENDBUF         Save the end of the test data buffer
         BCLR    #3,BLKSTAT        Clear post cal received status
         MOVE.L  D2,POSSYNC        Save first post cal sync code
         LEA.L   POSTBU,A1         Get start of post cal buffer
         LEA.L   POSTEND,A2        Get end of post cal buffer
         MOVE.L  A1,LSTPTR         Save running pointer
         BRA     RDYCHK            Go get next transmission
*
GETSUMC  MOVE.L  #$FAF3FF00,D3     Get checksum sync code
         MOVE.B  SUMCHK,D3         Store calculate checksum
         EOR.B   D2,D3             Exclusive OR with received checksum
         CMPI.B  #0,D3             Are the the same?
         BNE.S   SUMERR            No - process error
         MOVE.W  #5,RETRY          Else reset retry counter
         MOVE.L  A1,LSTPTR         Update running buffer pointer
         BRA.S   SNDACK            Send ack to ADAM
SUMERR   MOVE.L  LSTPTR,A1         Else restore buffer pntr to block
         MOVE.W  RETRY,D1          Decr. retry counter
         SUB.W   #1,D1             If retransmitted 5 times
         BEQ.S   ENDTRN            Exit
         MOVE.W  D1,RETRY          Else send checksum error to ADAM
SNDACK   MOVE.L  DPORT,D1
         BTST    #20,D1            ADAM in input mode?
         BEQ.S   SNDACK            No - then wait
```

```
ADMBSY    MOVE.L  DPORT,D1
          BTST    #22,D1         ADAM ready to receive?
          BNE.S   ADMBSY         No - then wait
          BCLR    #2,STAT        Set port to output mode
          MOVE.B  STAT,DPORT
          MOVE.L  D3,PPRT        Send checksum to ADAM
ADMBSY1   MOVE.L  DPORT,D1
          BTST    #22,D1         ADAM ready to receive?
          BNE.S   ADMBSY1        No - then wait
          BSET    #2,STAT        Set port to input mode
          MOVE.B  STAT,DPORT
          MOVE.B  #0,SUMCHK      Clear checksum
          BRA     RDYCHK         Go get next transmission
ENDTRN    CMPI.B  #0,BLKSTAT     All of the data received?
          BNE.S   TRNERR         No - display error
          CMP.W   #0,RETRY       Retransmission counter expired?
          BEQ.S   TRNERR         Yes - display error
          BCLR    #5,STAT        Clear busy status
          BSET    #1,STAT        Set DRASS offline
          BSET    #6,STAT        Set passed light
          BSET    #3,STAT        Set memory full light
          MOVE.B  STAT,DPORT
          RTS                    Return
*
TRNERR    BSET    #4,STAT        Set failed light
          BCLR    #5,STAT        Clear busy light
          BSET    #1,STAT        Set DRASS offline
          MOVE.B  STAT,DPORT
          MOVE.W  #0,CONTFL      Clear cont/halt flag
          LEA.L   TERRMSG,A0     Display error message
          BSR     DISPLAY
          BRA.S   WAITRSP
*
DATFND    LEA.L   MFMSG,A0       Display data present message
          BSR     DISPLAY
WAITRSP   CMPI.W  #0,CONTFL      Wait for continue button
          BEQ.S   WAITRSP
          BCLR    #4,STAT        Clear failed light
          MOVE.B  STAT,DPORT
WAITCMP   RTS                    Return
          END
```

```
************************************************************************
*
*                              SERDIAG
*
*   SERDIAG performs the serial diagnostic test on the UART in
*   the MFP. It executes an internal loop back test with a canned
*   message. SERDIAG does not utilize interrupts but it does test
*   the transmit and receive status. The data format and baud rate
*   used during the test are defaulted to 1200 baud, 7 bit, no
*   parity, and 2 stop bits. SERDIAG will execute continuously
*   until the halt button is detected (HLTFLG).
*
*         Input parameters : HLTFLG - set when the halt button
*                                     is pressed
*
*         Output parameters : SERST - 0 - data error
*                                    $10 - frame error
*                                    $20 - parity error
*                                    $30 - overrun error
*                                    $40 - Transmit time out
*                                    $50 - receive error
*
*         Registers used : A0 - index to test pattern
*                          D0 - test data character
*                          D1 - temp. UART status
*                          D7 - character count
*
************************************************************************
*
SERDIAG   IDNT    1,1
IMRA      EQU     $10809          Interrupt mask register
RSR       EQU     $10815          Receive status
TSR       EQU     $10816          Transmit status
UDR       EQU     $10817          UART data register
DPORT     EQU     $10824          DRASS status/control port
LPBAK     EQU     07              Loopback command
TXENA     EQU     $25             Transmit enable command
          INCLUDE DRAGDEF
          XREF    DISPLAY
          XDEF    SERDIAG
*
SERDIAG   BCLR    #4,STAT
          BCLR    #6,STAT
          BSET    #5,STAT         Set busy status
          MOVE.B  STAT,DPORT
          MOVE.B  #LPBAK,TSR      Output loopback command
          LEA.L   TSTPAT,A0       Get test pattern array
          MOVE.L  #$0A,D7         Load character count
SERLP     MOVE.B  (A0),D0         Get test character
          BSR     XMIT            Transmit character
          CMP.W   #0,SERST        Transmit error?
          BNE.S   SERERR          Yes - go set error status
```

```
         BSR     RECV              Read character back
         CMP.W   #0,SERST          Serial error?
         BNE.S   SERERR            Yes - go set error status
         CMP.B   (A0)+,D0          Compare in & out character
         BNE.S   SERERR            Set erro if not equal
         DBNE    D7,SERLP          Go send next character
         CMP.B   #0,HLTFLG         Halt pressed?
         BEQ     SERDIAG           No - go start test again
         BCLR    #5,STAT
         BSET    #6,STAT           Set passed light
         MOVE.B  STAT,DPORT
         MOVE.W  #0,CONTFL         Clear continue/halt flag
         MOVE.B  #TXENA,TSR        Enable transmitter
         RTS                       Return
*
SERERR   LEA.L   SERERM,A0         Display apropriate error message
         CLR.L   D1
         MOVE.W  SERST,D1
         ADDA.L  D1,A0
         BSR     DISPLAY
         BCLR    #5,STAT
         BSET    #4,STAT           Set error light
         MOVE.B  STAT,DPORT
SERWAT   CMP.B   #0,HLTFLG         Halt pressed?
         BNE.S   SERFXT            Yes - exit
         CMP.B   #0,CONTFL         Continue pressed?
         BEQ.S   SERWAT            No - wait
SEREXT   BCLR    #4,STAT           Clear busy
         MOVE.B  STAT,DPORT
         MOVE.W  #0,CONTFL         Clear continue/halt flag
         MOVE.B  #TXENA,TSR        Enable transmitter
         RTS                       Return
*
XMIT     MOVE.W  #$1000,D1         Load timeout counter
         MOVE.W  #0,SERST          Clear serial status
XMIT1    BTST    #7,TSR            Transmit ready
         BNE.S   XMITC             Yes - go output character
         SUB.W   #1,D1             Else decr. timeout counter
         BNE.S   XMIT1             Timeout?
         MOVE.W  #$40,SERST        Yes - set error
         BRA.S   RECRET            Return
XMITC    MOVE.B  D0,UDR            Output character
         RTS
*
RECV     MOVE.W  #0,SERST          Clear serial status
         MOVE.W  #$1000,D1         Load timeout counter
RECLP    MOVE.B  RSR,D1            Receive buffer full?
         BTST    #7,D1
         BNE.S   RECCNT            Yes - go read character
         SUB.W   #1,D1             Else decr timeout counter
         BNE.S   RECLP             Timeout?
         MOVE.W  #$50,SERST        Yes - set error
         BRA.S   RECRET            Return
```

```
RECCNT      MOVE.B UDR,D0             Read character
FRME        BTST   #4,D1              Frame error?
            BEQ.S  PARE               No - check parity
            MOVE.W #$10,SERST         Else set error
            BRA.S  RECRET
PARE        BTST   #5,D1              Parity error?
            BEQ.S  OVRE               No - check overrun
            MOVE.W #$20,SERST         Else set error
            BRA.S  RECRET
OVRE        BTST   #6,D1              Overrun error?
            BEQ.S  RECRET             No - return
            MOVE.W #$30,SERST         Else set error
RECRET      RTS
TSTPAT      DC.B   $30
            DC.B   $35
            DC.B   $40
            DC.B   $45
            DC.B   $3A
            DC.B   $4F
            DC.B   $55
            DC.B   $11
            DC.B   $22
            DC.B   $0C
            END
```

```
**********************************************************************
*
*                        RAMDIAG
*
*    RAMDIAG tests both the SRAM and Cache RAM in the system when the
*    memory diagnostic is selected. It performs a byte write and read
*    of memory with the data pattern AA,55,FF, and 00. The memory that
*    is tested is from 1000000 (STRAM) through 107FFFE (ENRAM) and
*    10100 (STCACH) through 10700 (ENCACH). This prevents the
*    destruction of system parameters and the system stack. RAMDIAG
*    uses the routine WDTST to do the actual memory accesses. If
*    a failure is detected, an error message is displayed.
*
*         Input parameters : D7 - test status from WDTST
*
*         Output parameters : None
*
*         Registers used : A0 - start of memory to test
*                          A1 - end of memory to test
*                          D0 - test data pattern
*                          D7 - test status from WDTST
*
**********************************************************************
*
RAMDIAG   IDNT    1,1
          INCLUDE DRASDEF
          XREF    WDTST,DISPLAY
          XDEF    RAMDIAG
STRAM     EQU     $1000000            Start of memory
ENRAM     EQU     $107FFFE            End of memory
STCACH    EQU     $10000              Start of Cache
ENCACH    EQU     $10700              End of Cache
DPORT     EQU     $10824              DRASS status/control port
*
RAMDIAG   BCLR    #4,STAT
          BCLR    #6,STAT
          BSET    #5,STAT             Set busy light on
          MOVE.B  STAT,DPORT
          BTST    #3,STAT             Ram full?
          BNE     RAMFULL             Yes - exit test
          CLR.L   D7                  Clear test status
          LEA.L   STRAM,A0            Get start of ram
          LEA.L   ENRAM,A1            Get end of ram
          MOVE.B  #$AA,D0             Load test pattern
          BSR     WDTST               Call test
          CMP.W   #0,D7               Test failed?
          BNE.S   RAMERR              Yes - set error
          MOVE.B  #$55,D0             Load next test pattern
          BSR     WDTST               Call test
          CMP.W   #0,D7               Test failed?
          BNE.S   RAMERR              Yes - set error
          MOVE.B  #$FF,D0             Load next test pattern
          BSR     WDTST               Call test
          CMP.W   #0,D7               Test failed?
```

```
         BNE.S    RAMERR              Yes - set error
         CLR.W    D0                  Set test pattern
         BSR      WDTST               Call test
         CMP.W    #0,D7               Test failed?
         BEQ.S    CACHTST             No - test Cache
RAMERR   LEA.L    DSPBUF,A5           Else display RAM error msg
         LEA.L    DRAMEM,A6
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6),(A5)
         LEA.L    DSPBUF+10,A5
         BSR      DSPRERR
CACHTST  CLR.L    D7                  Clear test status
         LEA.L    STCACH,A0           Get start of Cache
         LEA.L    ENCACH,A1           Get end of Cache
         MOVE.B   #$AA,D0             Get test pattern
         BSR      WDTST               Call test
         CMP.W    #0,D7               Test failed ?
         BNE.S    CACHERR             Yes - set error
         MOVE.B   #$55,D0             load test pattern
         BSR      WDTST               Call test
         CMP.W    #0,D7               Test failed
         BNE.S    CACHERR             Yes - set error
         MOVE.B   #$FF,D0             Load test pattern
         BSR      WDTST               Call test
         CMP.W    #0,D7               Test failed ?
         BNE.S    CACHERR             Yes - set error
         CLR.L    D0                  load test pattern
         BSR      WDTST               Call test
         CMP.W    #0,D7               Test failed?
         BEQ.S    RAMEXT              No - exit
CACHERR  LEA.L    DSPBUF,A5           Else display Cache error msg
         LEA.L    CRAMEM,A6
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6)+,(A5)+
         MOVE.L   (A6),(A5)
         LEA.L    DSPBUF+10,A5
         BSR.S    DSPRERR
RAMEXT   BCLR     #5,STAT             Clear Busy light
         BTST     #4,STAT             Failure set?
         BNE.S    RAMEXT1             Yes - exit
         BSET     #6,STAT             Else set passed light
RAMEXT1  MOVE.B   STAT,DPORT
         RTS
*
DSPRERR  SUBQ.L   #2,A0
         MOVE.L   A0,D5
         LSL.L    #4,D5
```

```
         MOVE.W  #7,D6
RDLP     LSL.L   #4,D5
         MOVE.B  D5,D7
         AND.B   #$0F,D7
         ADD.B   #$30,D7
         CMP.B   #$39,D7
         BLE.S   STCHR
         ADD.B   #$07,D7
STCHR    MOVE.B  D7,(A5)+
         SUB.W   #1,D6
         BNE.S   RDLP
         LEA.L   DSPBUF,A0
         BSR     DISPLAY
         BCLR    #5,STAT
         BSET    #4,STAT
         MOVE.B  STAT,DPORT
         MOVE.W  #0,CONTFL
WAITRSP  CMP.B   #0,CONTFL
         BNE.S   RSPRET
         CMP.B   #0,HLTFLG
         BEQ.S   WAITRSP
RSPRET   MOVE.W  #0,CONTFL
         BCLR    #4,STAT
         BSET    #5,STAT
         MOVE.B  STAT,DPORT
         RTS
RAMFULL  LEA.L   MFMSG,A0
         BSR     DISPLAY
         MOVE.W  #0,CONTFL
         BRA     WAITRSP
         END
```

```
************************************************************************
*
*                              DLDCOM
*
*     DLDCOM transfers the test data from DCOM to the DRASS memory
*     through the parallel port. All data that is read from the DCOM
*     system is stored into the DRASS memory until the memory is full.
*     The sync code FAF35000 is stored in PRESYNC to designate DCOM
*     data is present.
*
*          Input parameters : HLTFLG - set when the halt button
*                                      is pressed and will cause
*                                      transfer to exit
*
*          Output parameters : None
*
*          Registers used : A1 - start of data buffer
*                           A2 - end of data buffer
*                           A0 - display message pointer
*                           D1 - hand shaking status
*                           D2 - input data
*                           D3 - temporary data storage
*
************************************************************************
*
DLDCOM     IDNT     1,1
*
DPORT      EQU      $10824              DRASS status/control port
PPRT       EQU      $10820              Parallel port addr.
DATBUF     EQU      $1000000            Start of test data buffer
DATEND     EQU      $107CFFF            End of test data buffer
ENRAM      EQU      $107FFFE            End of static RAM
*
           INCLUDE  DRASDEF
           XREF     DISPLAY
           XDEF     DLDCOM
*
*
DLDCOM     BTST     #3,STAT             Memory full?
           BNE      DATFND              Yes - exit
           MOVE.W   #0,CONTFL           Clear halt/continue flags
           MOVE.L   #0,ENDBUF           Init buffer pointer
           LEA.L    DATBUF,A1           load start of memory
           LEA.L    DATEND,A2           Load end of memory
           BSET     #2,STAT             Set parallel port to input
           BCLR     #1,STAT             Set DRASS online
```

```
         BCLR    #4,STAT
         BCLR    #6,STAT
         MOVE.B  STAT,DPORT
         MOVE.L  PPRT,D1                   Reset parallel port
CONCHK   CMP.B   #0,HLTFLG                 Check for halt button
         BNE.S   ENDTRN                    Exit if pressed
         MOVE.L  DPORT,D1
         BTST    #23,D1                    ADAM connected?
         BNE.S   CONCHK                    No - wait
         BSET    #5,STAT                   Set busy light
         MOVE.B  STAT,DPORT
RDYCHK   CMP.B   #0,HLTFLG                 Check for halt button
         BNE.S   ENDTRN                    Exit if found
         MOVE.L  DPORT,D1
         BTST    #21,D1                    Parallel buffer full?
         BEQ.S   RDYCHK                    No - wait
         MOVE.L  PPRT,D2                   Read data
         MOVE.L  D2,(A1)+                  Store data in buffer
         CMPA.L  A1,A2                     Check for end of mem
         BGT.S   RDYCHK                    Go get more data
         MOVE.L  #$FAF35000,PRESYNC Store DCOM sync code
ENDTRN   CMP.L   #$FAF35000,PRESYNC All of the data received?
         BNE.S   TRNERR                    No - display error
         BCLR    #5,STAT                   Clear busy status
         BSET    #1,STAT                   Set DRASS offline
         BSET    #6,STAT                   Set passed light
         BSET    #3,STAT                   Set memory full light
         MOVE.B  STAT,DPORT
         RTS                               Return
*
TRNERR   BSET    #4,STAT                   Set failed light
         BCLR    #5,STAT                   Clear busy light
         BSET    #1,STAT                   Set DRASS offline
         MOVE.B  STAT,DPORT
         RTS
*
DATFND   LEA.L   MFMSG,A0                  Display data present message
         BSR     DISPLAY
WAITRSP  CMPI.W  #0,CONTFL                 Wait for continue button
         BEQ.S   WAITRSP
         BCLR    #4,STAT                   Clear failed light
         MOVE.B  STAT,DPORT
WAITCMP  RTS                               Return
         END
```

## Appendix G
## DRASS SERIAL DRIVERS ROUTINE

```
*******************************************************************
*
*                          OUTDATA
*
*    OUTDATA outputs the test data contained in the DRASS RAM to the
*    serial port. Before the transfer is initiated, it reads the
*    configuration for the port from the thumbwheel switches. The
*    selections are for : Baud rate, character size, number of
*    stop bits, and parity. Then it reads the selection for data
*    format : ASCII format or Binary format. In binary format
*    mode, the binary data is just output to the serial port as it
*    is read from memory. In ASCII format mode, the data is
*    converted to ASCII, spaces are inserted between each channel
*    data, the frame counter is inserted in front of each data
*    block, and the data is displayed in bloc s. Once this function
*    is initiated, it will continue to otput data until the end of
*    of data is reached or the halt button is detected.
*
*         Input parameters : HLTFLG - set when the halt button is
*                                     pressed
*                            CONTFL - set when the continue button
*                                     is pressed
*                            PREBUF - precal data
*                            DATBUF - test data buffer
*                            POSTBU - post cal buffer
*
*         Output parameters : None
*
*         Registers used : A0 - message pointer
*                          A1 - buffer pointer to transmit data
*                          D1 - sync code
*                          D2 - block length
*                          D7 - temporary storage
*
*******************************************************************
*
DPORT       EQU     $10824              DRASS status/control port
GPIP        EQU     $10800              GPIP for DTR status
TCDCR       EQU     $1080E              Baud rate control reg.
TDDR        EQU     $10812              Baud reat counter reg.
UCR         EQU     $10814              UART control port
RSR         EQU     $10815              Receive status
TSR         EQU     $10816              Transmit status reg.
UDR         EQU     $10817              UART data reg.
DATEND      EQU     $107CFFF            End of data buffer
*
OUTDATA     IDNT    1,1
            INCLUDE DRASDEF
            XREF    DISPLAY
            XDEF    OUTDATA
```

```
*
*
OUTDATA   BTST    #3,STAT            Memory full of data?
          BEQ     NODATA             No - display no data msg
          MOVE.W  #$FF,LSTSEL        Set last selection flag
          MOVE.W  #0,CONTFL          Clear continue/halt flag
          BCLR    #4,STAT
          BCLR    #6,STAT
          MOVE.B  STAT,DPORT
CHK1      MOVE.L  DPORT,D1           Read baud rate selection
          ANDI.L  #$0F,D1             from thumbwheel
          CMP.W   LSTSEL,D1
          BEQ.S   CHK1CON
          MOVE.W  D1,LSTSEL
          LEA.L   BAUDMSG,A0         Display current baud rate selection
          LSL.L   #4,D1
          ADDA.L  D1,A0
          BSR     DISPLAY
CHK1CON   CMP.B   #0,CONTFL          Continue button pressed?
          BNE.S   SETBAUD            Yes - baud rate
          CMP.B   #0,HLTFLG          Halt button pressed?
          BEQ.S   CHK1               No - go read thumbwheel
          BRA     OUTDRET            Else return
SETBAUD   CLR.L   D1
          MOVE.W  LSTSEL,D1          Get thumbwheel selection
          LEA.L   PRESC,A1
          MOVE.B  (A1,D1),TCDCR      Output prescale value
          LEA.L   CNTTBL,A1
          MOVE.B  (A1,D1),TDDR       Output count for baud rate
          MOVE.W  #$FF,LSTSEL        Clear last selection
          MOVE.W  #0,CONTFL          Clear continue/halt flag
CHK2      MOVE.L  DPORT,D1           Read serial format thumbwheel
          LSR.L   #4,D1
          AND.L   #$07,D1
          CMP.W   LSTSEL,D1          If change go display selection
          BEQ.S   CHK2CON
          MOVE.W  D1,LSTSEL
          LEA.L   SERFMT,A0          Get serial format msg.
          LSL.L   #4,D1
          ADD.L   D1,A0
          BSR     DISPLAY            And display it
CHK2CON   CMP.B   #0,CONTFL          Continue button pressed?
          BNE.S   SETFMT             Yes - go set serial format
          CMP.B   #0,HLTFLG          Halt pressed?
          BEQ.S   CHK2               No - go check thumbwheels
          BRA     OUTDRET            Else return
SETFMT    CLR.L   D1
          MOVE.W  LSTSEL,D1          Get last selection
          LEA.L   FMTTBL,A1
          MOVE.B  (A1,D1),UCR        Output serial format to UART
          MOVE.B  #$05,TSR           Enable transmitter
          MOVE.B  #$01,RSR           Enable receiver
          MOVE.W  #$FF,LSTSEL        Clear last selection
          MOVE.W  #0,CONTFL          Clear continue/halt flag
CHK3      MOVE.L  DPORT,D1           Get data format selection
```

```
          LSR.L    #8,D1
          AND.L    #$07,D1
          CMP.W    LSTSEL,D1           If changed display it
          BEQ.S    CHK3CON
          MOVE.W   D1,LSTSEL
          CMP.W    #2,D1
          BLT.S    DSPSEL
          LEA.L    ERRMSG,A0           If not 0 or 1 display error
          BSR      DISPLAY
          BRA.S    CHK3CON
DSPSEL    LSL.L    #4,D1
          LEA.L    DFMT,A0
          ADDA.L   D1,A0               Get data format msg
          BSR      DISPLAY             And display it
CHK3CON   CMP.B    #0,CONTFL           Continue button pressed?
          BNE.S    TXDATA              Yes - transmit data
          CMP.B    #0,HLTFLG           Halt pressed?
          BEQ      CHK3                no - go check thumbwheel
          BRA.S    OUTDRET             Else return
TXDATA    CMP.W    #0,LSTSEL           If thumbwheel= 0 or 1
          BEQ.S    SNDDAT              Transmit data
          CMP.W    #1,LSTSEL
          BEQ.S    SNDDAT
          BRA.S    OUTDRET             Else return
NODATA    LEA.L    NDATMSG,A0          Display no data present msg
          BSR      DISPLAY
          MOVE.W   #0,CONTFL           Clear continue/halt flag
NODATW    CMP.B    #0,CONTFL           If continue or halt pressed
          BNE.S    OUTDRET             Return
          CMP.B    #0,HLTFLG
          BEQ.S    NODATW
OUTDRET   BCLR     #5,STAT             Clear busy status
ODRETC    MOVE.B   STAT,DPORT
          RTS
*
SNDDAT    CMP.W    #1,LSTSEL           If in binary dump mode
          BEQ      SNDDAT0
          MOVE.W   #$2000,DATCNT
          MOVE.B   #0,SUMCHK
STRTWT    CMP.B    #0,HLTFLG
          BNE      OUTDRET
          BTST     #7,RSR
          BEQ.S    STRTWT
          MOVE.B   UDR,D0
          CMP.B    #5,D0
          BNE.S    STRTWT
          BSET     #5,STAT
          MOVE.B   STAT,DPORT
          CMP.L    #$FAF35000,PRESYNC
          BEQ      SNDDCOM
          MOVE.L   #$FAF31000,D1       Send par. block sync code
          BSR      SNDCMD
          CMP.B    #0,HLTFLG
          BNE      OUTDRET
          LEA.L    STSCT,A1            Get parameter block address
```

```
         MOVE.L #$210,D2         Get block size
         BSR    SNDBLK           Send data block
         CMP.B  #0,HLTFLG
         BNE    OUTDRET
SNDDAT0  CMP.L  #$FAF35000,PRESYNC
         BEQ    SNDDCOM
         MOVE.W #8,TCNT          Load block count for precal data
         MOVE.L PRESYNC,D1       Get precal sync
         BSR    SNDCMD           Send it
         CMP.B  #0,HLTFLG
         BNE    OUTDRET
         LEA.L  PREBUF,A1        Get precal data
SNDDAT1  MOVE.L BLKLEN,D2        Get data block length
         BSR    SNDBLK           Send data
         CMP.B  #0,HLTFLG
         BNE    OUTDRET
         SUBI.W #1,TCNT          Decr block count
         BEQ.S  SNDDAT2          If finished go transmit test data
         ADD.B  #1,D1
         BSR    SNDCMD           Else send frame counter
         CMP.B  #0,HLTFLG        If halt pressed
         BNE    OUTDRET          Return
         BRA.S  SNDDAT1          Else send next precal block
SNDDAT2  MOVE.L DATSYNC,D1       Get test data sync code
         BSR    SNDCMD           Output it
         CMP.B  #0,HLTFLG
         BNE    OUTDRET
         LEA.L  DATBUF,A1        Get test data buffer
SNDDAT3  MOVE.L BLKLEN,D2        Get block length
         BSR    SNDBLK           Send data
         CMP.B  #0,HLTFLG
         BNE    OUTDRET
         CMPA.L ENDBUF,A1        End of buffer?
         BGT.S  SNDDAT4          If end of data send post cal
         ADDI.B #1,D1            Else incr frame count
         BSR    SNDCMD           Send frame count
         CMP.B  #0,HLTFLG        Halt pressed?
         BNE    OUTDRET          Yes - return
         BRA.S  SNDDAT3          Else send next block
SNDDAT4  MOVE.W #8,TCNT          Get block count for postcal
         MOVE.L POSSYNC,D1       Get post cal sync code
         BSR    SNDCMD           Send frame count
         CMP.B  #0,HLTFLG
         BNE    OUTDRET
         LEA.L  POSTBU,A1        Get post cal buffer
SNDDAT5  MOVE.L BLKLEN,D2        Get block length
         BSR    SNDBLK           Send data block
```

```
         CMP.B    #0,HLTFLG
         BNE      OUTDRET
         SUB.W    #1,TCNT              Decr block counter
         BEQ.S    SNDEXT               If finished - exit
         ADD.B    #1,D1                Incr frame counter
         BSR.S    SNDCMD               And send it
         CMP.B    #0,HLTFLG            Halt pressed?
         BNE      OUTDRET              Yes - exit
         BRA.S    SNDDATS              Else send next block
*
SNDDCOM  MOVE.L   PRESYNC,D1
         BSR.S    SNDCMD
         CMP.B    #0,HLTFLG
         BNE      OUTDRET
         LEA.L    DATBUF,A1
SNDCOM1  MOVE.L   #$400,D2
         BSR      SNDBLK
         CMP.B    #0,HLTFLG
         BNE      OUTDRET
         CMPA.L   #DATEND,A1
         BLT.S    SNDCOM1
*
SNDEXT   CMP.W    #1,LSTSEL            If in binary dump mode
         BEQ.S    SNDEXT0
         MOVE.L   #$FAF30000,D1        Send end of transmission sync
         BSR.S    SNDCMD
SNDEXT1  CMP.B    #0,HLTFLG
         BNE      OUTDRET
         BTST     #7,TSR
         BEQ.S    SNDEXT1
         MOVE.B   SUMCHK,UDR
SNDEXT0  BRA      OUTDRET              Return
*
SNDCMD   CMP.W    #1,LSTSEL            ASCII data transfer?
         BEQ      ASCICMD              Yes - convert to ASCII first
         MOVE.L   D1,D7                Else send the data in hex
         ROL.L    #8,D7
         ADD.B    D7,SUMCHK
         BSR.S    BINXMIT
         ROL.L    #8,D7
         ADD.B    D7,SUMCHK
         BSR.S    BINXMIT
         ROL.L    #8,D7
         ADD.B    D7,SUMCHK
         BSR.S    BINXMIT
         ROL.L    #8,D7
         ADD.B    D7,SUMCHK
         BSR.S    BINXMIT
         RTS
*
SNDBLK   CMP.W    #1,LSTSEL            ASCII data transfer?
         BEQ      ASCIBLK              Yes - convert to ASCII first
         MOVE.B   (A1)+,D7             Else send data in hex
         ADD.B    D7,SUMCHK
         BSR.S    BINXMIT
         SUB.L    #1,D2
         BNE.S    SNDBLK
         RTS
```

```
*
BINXMIT   CMP.B   #0,HLTFLG          Halt pressed?
          BNE     SERXRET            Yes - return
          BTST    #7,TSR             Transmit ready?
          BEQ     BINXMIT            No - wait
          MOVE.B  D7,UDR             Else send data
          SUB.W   #1,DATCNT
          BNE.S   BINXRET
BINXMT2   BTST    #7,TSR
          BEQ.S   BINXMT2
          MOVE.B  SUMCHK,UDR
          MOVE.B  #0,SUMCHK
          MOVE.W  #$2000,DATCNT
BINXMT3   CMP.B   #0,HLTFLG
          BNE.S   BINXRET
          BTST    #7,RSR
          BEQ.S   BINXMT3
          MOVE.B  UDR,D0
          CMP.B   #6,D0
          BEQ.S   BINXRET
          BSET    #4,STAT
          MOVE.B  #$FF,HLTFLG
BINXRET   RTS
*
SERXMIT   CMP.B   #0,HLTFLG
          BNE.S   SERXRET
          BTST    #7,TSR
          BEQ.S   SERXMIT
          MOVE.B  D7,UDR
SERXRET   RTS
*
ASCICMD   MOVE.B  #$0D,D7            Send carrage return
          BSR     SERXMIT
          MOVE.B  #$0A,D7            Send line feed
          BSR     SERXMIT
          MOVE.B  #$20,D7            Insert a space
          BSR     SERXMIT
          MOVE.B  D1,D7              Convert frame count to ASCII
          LSR.B   #4,D7
          AND.B   #$0F,D7
          ADD.B   #$30,D7
          CMP.B   #$39,D7
          BLE.S   ACMD1
          ADD.B   #$07,D7
ACMD1     BSR     SERXMIT            And send it
```

```
         MOVE.B  D1,D7
         AND.B   #$0F,D7
         ADD.B   #$30,D7
         CMP.B   #$39,D7
         BLE.S   ACMD2
         ADD.B   #$07,D7
ACMD2    BSR     SERXMIT
         MOVE.B  #$20,D7             Send three spaces
         BSR     SERXMIT
         BSR     SERXMIT
         RTS
*
ASCIBLK  MOVE.W  #23,D6              Set for 23 characters per line
ASCBLK0  MOVE.B  (A1),D7             Get data byte
         LSR.B   #4,D7               Convert it to ASCII
         AND.B   #$0F,D7
         ADD.B   #$30,D7
         CMP.B   #$39,D7
         BLE.S   ASCBLK1
         ADD.B   #$07,D7
ASCBLK1  BSR     SERXMIT             And send it
         MOVE.B  (A1)+,D7
         AND.B   #$0F,D7
         ADD.B   #$30,D7
         CMP.B   #$39,D7
         BLE.S   ASCBLK2
         ADD.B   #$07,D7
ASCBLK2  BSR     SERXMIT             Send second character
         MOVE.B  #$20,D7
         BSR     SERXMIT             Insert a space
         SUB.L   #1,D2
         BEQ.S   ASCRET              Return if end of block
         SUB.W   #1,D6
         BNE     ASCBLK0             If end of line
         MOVE.B  #$0D,D7             Send carrage return
         BSR     SERXMIT
         MOVE.B  #$0A,D7             Send line feed
         BSR     SERXMIT
         MOVE.B  #$20,D7             Send 5 spaces
         BSR     SERXMIT
         BSR     SERXMIT
         BSR     SERXMIT
         BSR     SERXMIT
         BSR     SERXMIT
         BRA     ASCIBLK             Continue transmitting block
ASCRET   RTS                         Return
         END
```

```
*            DRASS EXTERNAL REFERENCES
*                STORAGE PARAMETERS
*
             XREF      PREBUF,PREEND,POSTBU,POSTEND,STSCT,ENDSCT,BLKLEN
             XREF      UCNTRL,BAUD,PRSCA,PRSCB,PRSCCD,FLCNTA,FLCNTB,FLCNTC
             XREF      FLCNTD
*
             XREF      DSTAT,STAT,DSPBUF,LSTCMD,CONTFL,HLTFLG,CMD,STRTFL
             XREF      SUMCHK,PRESYNC,DATSYNC,POSSYNC,BLKSTAT,LSTSEL,RETRY
             XREF      LSTPTR,ENDBUF,TCNT,SERST,DATBUF,LSTSWT,DATCNT
*
             XREF      DMSG,PRMPT,TEXT1,TEXT2,TEXT3,TEXT4,TEXT5,TEXT6
             XREF      TEXT7,TERRMSG,MFMSG,ERRMSG,BAUDMSG,SERFMT,DFMT
             XREF      PRESC,CNTTBL,FMTTBL,ENDROM,SERERM,CRAMEM,DRAMEM
             XREF      NDATMSG
```

## REFERENCES


Armstrong Aerospace Medical Research Laboratory, 1988, Anthropometry and Mass Distribution for Human Analogs, Volume I: Military Male Aviators, March 1988, AAMRL-TR-88-010.

Air Force Aerospace Medical Research Laboratory, 1983, Effects of Negative Strap on Restraint Dynamics and Human Impact Response, AFAMRL-TR-83-083, Wright-Patterson Air Force Base, Ohio.

Air Force Aerospace Medical Research Laboratory, 1980, Evaluation of a Proposed Modified F/FB-111 Crew Seat and Restraint System, AFAMRL-TR-80-50, Wright-Patterson Air Force Base, Ohio.

Air Force Aerospace Medical Research Laboratory, 1982, Vertical Impact Tests of a Modified F/FB -111 Crew Seat to Evaluate Headset Position and Restraint Configuration Effect, AFAMRL-TR-82-51, Wright-Patterson Air Force Base, Ohio.

Bartol, Aileen M. and Kaleps, Ints, 1987, The Development of Segment Based Axis Systems for the Air Force Advanced Dynamic Anthropomorphic Manikin (ADAM), Proceedings of the Eleventh International Conference on Experimental Safety Vehicles.

Bateman, R. P., Bressler, J. R., Gustin, T. W., Riegler, J. T., Tieber, J. A., and White, R. P., May 1984, The State of the Art of Anthropomorphic Surrogate and Requirements for the Evaluation of Advanced Aircraft Ejection Systems , AFAMRL-TR-84-XXX, Air Force Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio (in process).

Begeman, P. C., King, A. I., and Prosad, P., 1973, Spinal Loads Resulting from $-G_x$ Acceleration, Proceedings of the 17th Stapp Car Crash Conference.

Belytschko, T., and Privitzer, E., 1978, Refinement and Validation of a Three Dimensional Head-Spine Model, AMRL-TR-78-7, Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

Budynas, Richard G., 1977, Advanced Strength and Applied Stress Analysis, McGraw-Hill Company, New York.

Cheng, R., Mital, N. K., Levine, R. S., and King, A. I., 1979, Biodynamics of the Living Human Spine During $-G_x$ Input Acceleration, Proceedings of the 23rd Stapp Car Crash Conference.

Churchill, Edmund, Churchill, Thomas, Downing, Kay, Erskine, Peggy, Lauback, Lloyd L., and McConville, John T., 1978, Anthropometric Source Book, Volume II: A Handbook of Anthropometric Data, NASA Reference Publication No. 1024, National Aeronautics and Space Administration, Scientific and Technical Information Office.

Engin, A. E., 1979, Measurement of Resistive torques in Major Human Joints, AFAMRL-TR-79-4, Air Force Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

Ewing, C. L. and Thomas, D. J., August 1972, Human Head and Neck Response to Impact Acceleration, USAARL 73-1 and NARML Monograph 21.

Ewing, C. L., Thomas, D. J., Lustik, L., Muzzy, W. H., III, Willems, G. C., and Majewski, P., 1977, Dynamic Response of the Human Head and Neck to $G_y$ Impact Acceleration, Proceedings of the 21st Stapp Car Crash Conference.

Foster, J. K., Kortge, J. O., and Waianin, J. J., 1977, Hybrid III-A Biomechanically Based Crash Test Dummy, Proceedings of the 21st Stapp Car Crash Conference.

Halfman, Robert L., 1962, Dynamics, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Hoerner, S. F., 1965, Fluid Dynamic Drag, Hoerner Fluid Dynamics, New Jersey.

McConville, J. T., Churchill, T. D., Kaleps, I., Clauser, C. E., and Cuzzi, J., 1980, Anthropometric Relationships of Body and Body Segment Moments of Inertia, AFAMRL-TR-80-119, Air Force Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

McConville, John T. and Laubach, Lloyd L., 1978, Anthropometry, Chapter III in Anthropometric Source Book, Volume I: Anthropometry for Designers, NASA 1024, National Aeronautics and Space Administration, Scientific and Technical Information Office.

Mertens, H., January 1978, Nonlinear Behavior of the Sitting Human Under Increasing Gravity,
Mertz, H. J. and Patrick, L. M., 1971, Strength and Response of the Human Neck, Proceedings of the 19th Stapp Car Crash Conference.

MIL-HDBK-5D, 1 June 1983.

Nyquist, G. W. and Murton, C. M., 1975, Static Body Response of the Human Lower Torso, Proceedings of the 19th Stapp Crash Conference.

Payne, P. R., 1975, Low Speed Aerodynamic Forces and Moments Acting on the Human Body, AMRL-TR-75-6, Air Force Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

Payne, P. R., 1974, Some Studies Relating to "Limb Flailing" After an Emergency Escape from an Aircraft, AMRL-TR-73-24, Air Force Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

Privitzer, E., and Belytschko, T., January 1980, Impedance of a Three-Dimensional Head-Spine Model, Mathematical Modelling.

Roark, Raymond J., 1954, Formulas for Stress and Strain, Third Edition, McGraw-Hi'l Company, New York.

Schneider, L. W., Bowna, B. M., Snyder, R. G., and Peck, L. S., May 1976, A Prediction of Response of the Head and Neck of the U.S. Adult Military.

Population Dynamic Impact Acceleration from Selected Dynamic Test Subjects, UM-HSRI-76-10.

Shigley, Joseph E. and Larry D. Mitchell, 1983, Mechanical Engineering Design, Fourth Edition, McGraw-Hill Company, New York.

Specker, Lawrence J., 1985, Flow Stagnation as an Advanced Windblast Protection Technique, Presented at the 1985 SAFE Symposium.

Systems Research Laboratories, Inc., Revised April 1987, System Specification, Advanced Dynamic Anthropomorphic Manikin (ADAM), U.S. Air Force Contract No. F33615-85-C-0535, Dayton, Ohio.

U.S. Air Force, 1985, ADAM Statement of Work, Contract No. F33615-85-C-0535, Issued by the Department of the Air Force, Air Force Systems Command, Aeronautical Systems Division/ PMRSB, Wright-Patterson Air Force Base, Ohio.

Vogel, M. G., April 1986, Progress Report on Testing of Friction Concepts for Manikin Joints, Systems Research Laboratories, Inc.

Vogel, M. G., June 1986, Design Study of the Manikin's Joint Articulation Mechanisms, SRL Report No. 6995-05-86, Systems Research Laboratories, Inc.

Vogt, H. L., Coermann, R. R., and Fust, H. D., July 1968, Mechanical Impedance of the Sitting Human Under Sustained Acceleration, Aerospace Medicine, Volume 39, Number 7.

Vykukal, H. C., November 1965, Dynamic Response of the Human Body to Vibration When Combined With Various Magnitudes of Linear Acceleration, Aerospace Medicine.

White, R. P., Jr., Gustin, T. W., and Tyler, M. C., December 1984, Preliminary Design of a Limb Restraint Evaluator, AFAMRL-TR-84-042, Air Force Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

Williams, J. and Belytschko, T., 1981, A Dynamic Model of the Cervical Spine and Head, AFAMRL-TR-81-5, Air Force Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

Wittman, Thomas J., 1966, An Analytical Model to Duplicate Human Dynamic Force Response to Impact, AMRL-TR-66-126, Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio.

 *U.S.Government Printing Office: 1990—548-018/20210

END