AD-A118 474 OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS F/G 6/4
AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD T--ETC(U)
AUG 82 R B MCGHEE, K J WALDRON MDA903-82-K-0058

UNCLASSIFIED NL

1 of 1
AD
A118474

END
DATE
FILMED
9-82
DTIC

RF Project 762945/714250
Semi-Annual Technical Report

the
ohio
state
university
research foundation
1314 kinnear road
columbus, ohio
43212

AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION

R. B. McGhee and K. J. Waldron
College of Engineering

For the Period
October 1, 1981 - March 31, 1982

DEPARTMENT OF THE ARMY
Defense Supply Service
Washington, D.C. 20310

Contract No. MDA903-82-K-0058

August, 1982

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A118 474	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION		5. TYPE OF REPORT & PERIOD COVERED Semi-Annual Technical Report Oct. 1, 1981 - Mar. 31, 1982
		6. PERFORMING ORG. REPORT NUMBER 762945 /714250
7. AUTHOR(s) Robert B. McGhee and Kenneth J. Waldron		8. CONTRACT OR GRANT NUMBER(s) Contract No. MDA903-82-K-0058
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Ohio State University Research Foundation, 1314 Kinnear Road Columbus, Ohio 43212		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DEPT. OF THE ARMY, Defense Supply Service Room 1D-245, The Pentagon Washington, D.C. 20310		12. REPORT DATE August, 1982
		13. NUMBER OF PAGES 93
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Walking machine
Robotics
Off-road vehicles
Legged locomotion

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes research accomplished during the first six months of a project aimed at field testing of an experimental vehicle making use of an actively controlled suspension system for improved off-road mobility. The first year of this project involves component evaluation and concept verification with reduced-scale laboratory models. Preliminary test results and projections of performance to be expected in the full-scale vehicle are included in the report.

Semi-Annual Technical Report

for

DARPA Contract MDA 903-82-K-0058

AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION

prepared by

Robert B. McGhee and Kenneth J. Waldron

covering the period

October 1, 1981, through March 31, 1982

College of Engineering
The Ohio State University
Columbus, Ohio 43210

INTRODUCTION

Research under this contract began as scheduled on October 1, 1981. During the first six months, no significant deviations from the plan contained in our proposal dated May 13, 1981, occurred. Research performed relative to each of the tasks listed in our Statement of Work, dated September 21, 1981, is summarized in the following paragraphs. Further details regarding some of these topics are provided in the attached appendices and in videotapes previously furnished to DARPA.

RESEARCH ACCOMPLISHED

Task 1: Vertical Sensor Evaluation

Rough terrain locomotion on a laboratory terrain board has been accomplished by the OSU Hexapod Vehicle. A surplus vertical gyro furnished by the University of Wisconsin was used for this purpose. The results obtained are documented in a videotape entitled "The Adaptive Suspension Vehicle Concept," dated March, 1982. While the outcome of this test was generally satisfactory, a noticeable amount of gyro drift occurred during trials involving large amounts of body motion. An alternative gyro has been ordered, and further tests will be made during the coming quarter to determine if this effect can be reduced. Preliminary evaluation of pendulum sensors indicates that these devices are excessively affected by vehicle body motion during locomotion, but may be useful as an inexpensive means for verification of gyro function during automatic checkout of vehicle subsystems prior to initiation of gait.

Task 2: Control Mode Studies

A simulation study involving a CRT display of a simplified ASV-84 hexapod vehicle has been initiated for the purpose of studying alternative control modes. Research thus far has centered about a "close maneuvering" mode in which signals from a three-axis joystick are used to control fore-aft velocity, right-left velocity, and turning rate about the vehicle vertical axis. In this control mode, the vertical velocity of the body is automatically adjusted to maintain a desired altitude above the terrain. The remaining two components of body velocity, namely, pitch rate and roll rate, are also automatically controlled, either to maintain the body at an operator specified elevation and roll angle, or else to adjust these angles automatically to keep the vehicle body parallel to the average terrain slope. Examples of results obtained with this simulation are included in videotapes previously submitted to DARPA. Research on this topic during the next quarter of this contract will be aimed at definition of additional control modes, improvement of the existing simulation, and tests of the above described "terrain-following" locomotion with the OSU Hexapod.

Task 3: Insect Control Studies

A simulation program using a "free gait" algorithm has been written. This program produces nonperiodic stepping patterns and is capable of dealing with terrain possessing a significant percentage of regions unsuitable for load bearing due to the presence of holes, rocks, soft soils, etc. The results of this simulation will be compared to digital and film recordings of locust stepping patterns to be provided by the University of Alberta. Preliminary data of this type has already been obtained from Dr. Robert Franklin of the University of Oregon. The purpose of this investigation is to develop more versatile algorithms for automatic control of ASV-84 stepping patterns during complex maneuvering over rough terrain.

Task 4: Fault Tolerant Software

The definition of this task has been expanded to include automatic pre-locomotion checkout of vehicle subsystems and on-line detection of stepping pattern errors which could result in any of the following three undesirable conditions: 1) motion of a joint to an angle such that its kinematic limit is encountered; 2) collision of adjacent limbs; 3) instability of a support pattern or possible collision of the body with the ground for any other reason. With software now being written, imminent violation of any of these "safety" conditions will automatically halt the vehicle and will provide the operator with an error message. It is anticipated that this software will be completed and tested with the OSU Hexapod Vehicle during the fourth quarter of the present contract year.

Task 5: Laser Foothold Designator

One television array camera has been delivered. This camera has been successfully interfaced to the PDP-11/70 computer. Camera calibration is under way and should be completed during the next quarter of research. Delivery of the second camera, needed for triangulation on the operator controlled laser spot reflected from the terrain, is also expected in the next quarter. Software for optical control of stepping patterns for the OSU Hexapod Vehicle has been completed and tested using keyboard entry of terrain coordinates to simulate the range-finding function.

Task 6: Leg Geometry Studies

Assembly of the DUWE (Dynamic Unpowered Walking Experiment) was completed and the device walked successfully. The measured specific resistance was in the range 0.1 to 0.2 which is comparable to that of an automobile. The machine behavior also correlated well with that predicted by computer simulation. We believe that this experiment demonstrates that there is no instrinsic mechanical obstacle to the achievement of low specific resistances by fully terrain-adaptive walking machines.

Computer-aided geometric design of leg geometry continued with feedback from both the DUWE experiment and the Monopod (fully discussed in an appendix to the final report on Contract no. MDA903-81-C-0138). Dimensions were finalized for the geometry based on a four-bar linkage with a sliding shank. Parallel geometric studies of pantograph leg geometries were conducted and a set of dimensions for a leg of this type were also frozen.

Finite element studies of stress and deflection in legs of the two types mentioned above have been initiated. These studies are to be coordinated with the frame model to produce a stress and vibration model of the whole vehicle.

Task 7: Power Package Specifications

A preliminary set of pump specifications has been transmitted to the University of Wisconsin. The general configuration of the flywheel package has been agreed upon and is discussed in Appendix 1. A specification for a generator or alternator awaits estimates of computer and instrumentation power requirements.

Task 8: Power Distribution and Actuator Servo Design

A first version of the power transmission system configuration design has been completed and is discussed in Appendix 1. Simulation studies of portions of this system are in progress and representative results are discussed in Appendix 2. A kinetostatic simulation of the whole vehicle initiated to provide actuator loads, actuator velocities, and reflected inertias for input to the transmission system simulation is being further developed to simulate arbitrary paths on sloping and uneven terrain.

Component procurement for the breadboard leg system is in progress. As of the end of the time period covered by this report, the variable displacement pumps and control valves have been delivered. The actuators have been selected and studies are in progress for selection of a suitable microcomputer configuration.

Task 9: Frame Design

As of the end of the first six months of this research, a lightweight mock-up of the vehicle frame has been constructed. Field mobility tests using this frame supported by three people are scheduled to take place in the coming quarter.

A preliminary configuration for the vehicle frame structure has been selected and a structural model for weight distribution studies and for finite element input has been generated using the SUPERTAB program. Preliminary finite element analysis has been performed using the SUPERB program. Material from these studies in included in the videotape delivered as part of the final report on Contract MDA903-81-C-0138. On the basis of these preliminary studies, and as a result of experience with the OSU Hexapod and other robot structure designs, it appears that the primary design constraints will be natural frequencies and modes of structural vibration rather than load-bearing capacity. Design modelling is proceeding on this assumption.

Task 10: On-Board Computer Design

An Intel 8086/8087 single board computer has been interfaced to the electric monopod vehicle. Data conversion (A to D and D to A) is provided through an Intel 8088 computer which has also been interfaced to this machine. These two computers communicate via the Intel Multibus, which includes hardware arbitration of conflicting bus requests. A program for single axis control of the monopod is being written in PASCAL and assembly language using a Hewlett-Packard 64000 Microprocessor Development System (MDS). Although it is anticipated that this program will be completed during the next quarter of research under this contract, this approach to programming is awkward and inefficient. It has been decided that it would be much better to make use of an Intel MDS and a suitable model number has been determined. While this item is included in the budget for the second year of this project, its need is strongly felt, and permission for purchase during the current contract year has been requested from our local contract monitor.

Task 11: Software Design

Version 3.4 of the OSU Hexapod real-time software has been completed. A listing is included with this report as Appendix 3. This version has been modularized in such a way as to be conveniently adapted to a multiprocessor implementation. Among the other important features of this software are its improved on-line operator communication characteristics and the inclusion of a rough terrain locomotion capability. Test results obtained with this software in conjunction with the OSU Hexapod Vehicle are documented in the two videotapes previously referenced in this report.

Task 12: Electronic Subsystem Design

A study of sensor requirements for the ASV-84 Vehicle has been initiated. Requirements for leg servo power electronics are being developed. Design of an ultrasonic ranging system for possible use as a leg proximity sensor has been completed. Components have been ordered, and construction of a breadboard model proximity sensing subsystem is scheduled for the coming quarter.

SUMMARY

This report summarizes work accomplished during the first six months of a three-year contract leading to field testing of an experimental adaptive suspension vehicle (ASV-84). The key feature of this vehicle is that it is supported and propelled by an actively controlled linkage mechanism enabling it to adjust its configuration to ground contours in order to smooth body motion, improve slope climbing and rough terrain mobility, and potentially, to achieve better fuel economy than conventional off-road vehicles. The research is motivated by the observation that approximately one-half of the land surface of the earth is presently inaccessable to any conventional wheeled or tracked vehicle, while man and large animals are able to traverse such areas with relative ease.

The first year of this program is devoted to design of the ASV-84 machine. It now appears certain that this vehicle will be a hexapod walking machine without rolling elements in contact with the ground, although the design will be such that a retrofit of such components will be possible after initial field tests. The 1984 vehicle will be provided with advanced sensing including a terrain-scanning laser ranging device, proximity sensors and force sensors on each leg, and an inertial measurement unit. This system will carry a single operator who will be responsible for route selection, speed control, and other high level decisions. Lower levels of coordination and processing of sensor data will be handled by a multiprocessor on-board computer.

In order to conduct the design phase of this research in a realistic context, a number of experimental machines are being used. Included in these are: 1) the OSU Hexapod Vehicle, a functioning computer-controlled quarter-

scale laboratory model walking machine; 2) the Electric Monopod, a wheeled test-bed vehicle propelled and supported by a quarter-scale linkage-coordinated leg; and 3) the DUWE vehicle, a towable unpowered hexapod designed to study resonant energy storage during limb cycling in order to improve locomotion efficiency to a level comparable to that achieved by wheeled vehicles.

Due to the complexity of this project, and to the fact that the research has to do with control of motion, videotaping has been chosen as the primary means of reporting experimental results to DARPA. The present document therefore represents only a summary, and access to the videotapes is required in order to obtain a full understanding of research accomplished during this reporting period.

In conclusion, no major deviations from the original project plan have occurred during the first six months of this research. We currently anticipate that the design of the ASV-84 vehicle will be completed on schedule at the end of the first year of research. A major design review is presently scheduled for October, 1982, and it is anticipated that vehicle construction will begin immediately after this review.

Appendix 1

To be presented at 17th Intersociety Energy Conversion Engineering Conference, Los Angeles, August 8-13, 1982.

THE USE OF MECHANICAL ENERGY STORAGE IN AN UNCONVENTIONAL, ROUGH TERRAIN VEHICLE

K.J. Waldron*, A.A. Frank†, and K. Srinivasan*

* Dept. of Mechanical Engineering
Ohio State University

† Dept. of Electrical Engineering
University of Wisconsin

ABSTRACT

Vehicles designed for very rough terrain are subject to large variations in power demand and, therefore, benefit from energy storage and regeneration. This paper concerns energy management and conservation in a legged vehicle presently being designed for transportation over such terrain. Both the rough terrain requirement and the oscillatory nature of the walking action dictate careful design for energy management if the power to weight ratio is to be improved over current designs.

The arrangement of the power generation, storage and transmission system will be discussed. This comprises a gasoline engine, flywheel storage package and a hydraulic transmission and actuation system featuring computer controlled variable displacement pumps.

INTRODUCTION

The subject of this paper is the energy management system design of an unconventional vehicle intended for service in very rough terrain. An artist's conception of this vehicle is shown in Figure 1. It may be seen that the support and locomotion system consists of six legs. Legged locomotion presents a number of advantages over wheels or tracks, particularly in rough terrain (1). It comes as a surprise to most people that only about 50% of the earth's land surface is accessible to wheeled or tracked vehicles (2). At least 90% is accessible to animals using legged locomotion and, in principle, to artificial legged vehicles. Commercial machines using a combination of wheels and articulated limbs (3,4) have demonstrated ability to handle gradients, soil types and terrain features far beyond those which conventional wheeled or tracked vehicles can handle.

There have been a number of experimental walking machines built. Those described in references (5,6,7,8,9) have the capability to adapt their gaits to terrain conditions as does the machine we are designing. A severe problem with all of these machines, which has not previously been seriously addressed, is energy management. Figure 2 is a modification of Gabrielli and Von Karman's diagram (10) showing the ratio of locomotion power to weight plotted against speed for various locomotion systems. Two points have been added. These correspond to the G.E. Quadruped (5) and the OSU Hexapod (6) and are representative of fully adaptive walking machines. It may be seen that an improvement of at least one order of magnitude in power to weight ratio is necessary for these to become competitive with tracked vehicles, and two orders of magnitude improvement is needed to approach the performance of biological walking systems.

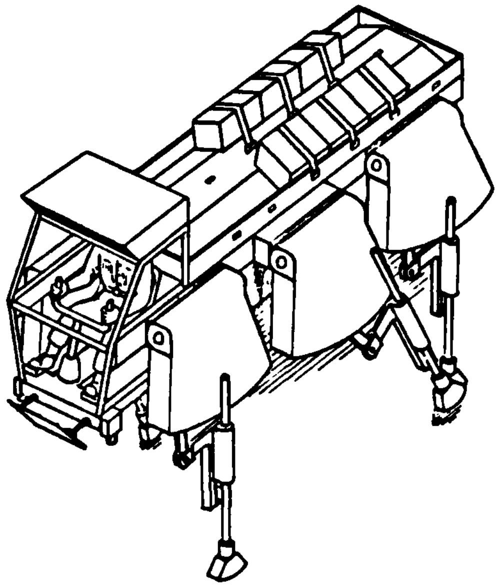
Figure 1: Concept drawing of the adaptive suspension vehicle

A discussion of the power loss mechanisms in walking machines may be found in reference (11). Both the vertical plane, "mammalian" type leg

PRECEDING PAGE BLANK-NOT FILMED

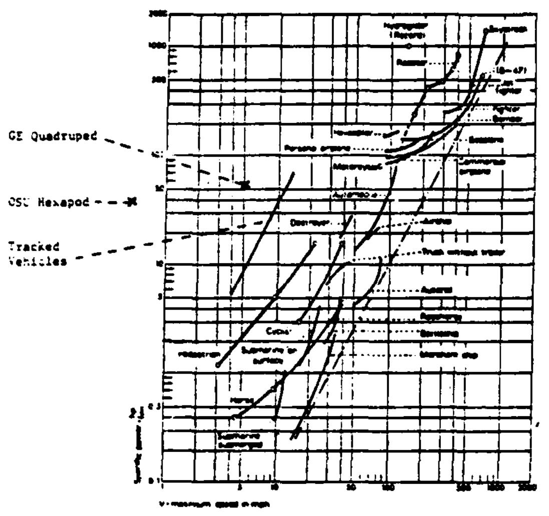

Figure 2: Specific power, defined as the ratio of the maximum power available to the gross weight of the vehicle, plotted as a function of its maximum speed (from Gabrielli and von Karman, 1954)

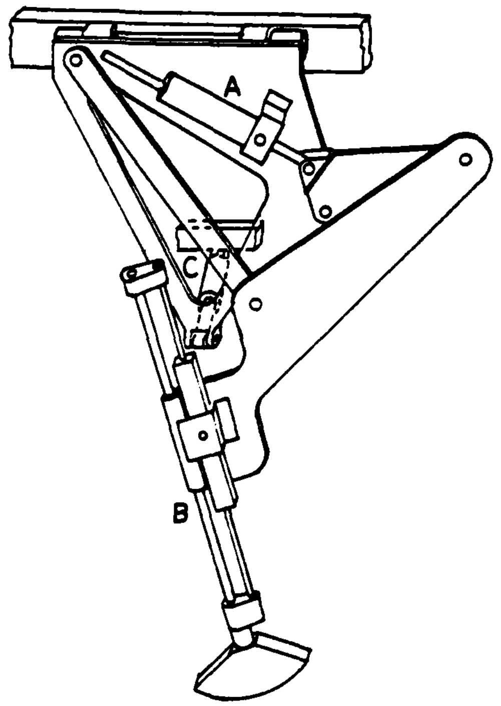

Figure 3: Concept drawing of adaptive suspension vehicle leg

geometry used, as distinct from the "arthropod" type leg geometry of the machines described in references (6,7,8,9), and the unusual linkage geometry (Figure 3) were adopted for energy management reasons. The selection of the leg geometry is further described in reference (12).

Any vehicle encounters wide variations in power demand resulting from changing terrain and speed conditions. For energy conservation purposes it is, therefore, attractive to use energy storage and regeneration on a vehicle to allow optimization of the engine operating conditions and to regeneratively recover energy absorbed when braking (13). The more severe the terrain, the more extreme the variation in power demand. The reasons for electing to make use of an energy storage device in the vehicle we are designing are, therefore:

1. The need for substantial improvement in power demand.
2. The need for spurts of very high power to overcome severe obstacles in a system which must also be highly efficient on level terrain.

For reasons of actuator force to weight ratio and of the simplicity and quick response of direct drive without gears or other speed reducers, a hydraulic power transmission and actuation system will be used. It is the configuration of this system, and of the associated prime mover and energy storage package, which forms the subject of this paper.

PRIME-MOVER AND ENERGY-STORAGE PACKAGE

Requirements

For any vehicle there are basic requirements that a power supply must meet. These are sustained expected power and peak power. The sustained or steady-state condition sets the waste-energy requirements. The peak-power requirements set the component size (i.e., torque and speed of each component).

As an example, for standard automobiles the average power required to drive the vehicle in the city is about 3 to 4 hp for a vehicle about our size. The peak power, however, is about 45 hp. The vehicle, however, may be constructed with 60 hp or more. The reason for this disparity is that the maximum available power is seldom used. The power train, however, must be designed to handle the peak, but it may not be designed to sustain this peak. We will need a peak-power capability 10 to 20 times the average. We can readily meet this requirement with a flywheel energy-storage power plant.

Energy-storage capacity is another design factor of the power supply. Standard power systems do not have large energy storage in the prime mover. The stored energy is usually in the form of liquid fuel. Fuel flow is required to increase power. These systems, however, cannot provide the same efficiency of conversion for all levels of power. In fact, the efficiency of conversion drops as the power level drops. In addition, it is not possible to regenerate energy, i.e., feed vehicle energy back to the power plant for later re-use. This implies that we waste energy every time the vehicle is stopped faster than by coasting and every time we go down a steep hill.

To provide a power supply which can provide a peak power of about 150 hp, a sustained power of 60 hp, and an energy-storage capacity of about 1/3 hp-hr, an engine-flywheel system is considered. The requirement for energy storage is determined by the amount of energy recoverable going down a grade (of 60% for 100 yards) or the time for sustained high power. In the above specification, we would be able to sustain 150 hp for about 10 seconds. This amount of power will allow the vehicle to lift itself up at 20 feet per second. Of course, to achieve this performance, all components must be sized to handle this rate of energy flow. This capacity allows the vehicle to climb vertically 200 feet in 10 seconds if so desired or required.

The power-plant package is shown in Figure 4. This package shows a design which can be relatively easily implemented. The flywheel package has two clutches. An engine clutch allows it to be decoupled from the flywheel supply and the variable-displacement drive pump. The purpose of the second clutch is to start the flywheel automatically without having to control the slip of the clutch.

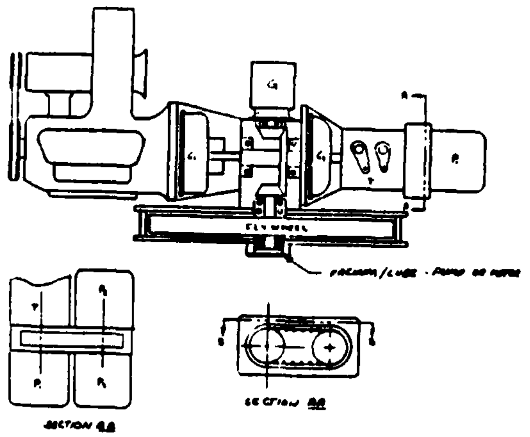

Figure 4: Power plant package

When a flywheel power plant is first started, the engine must engage the flywheel gradually to prevent too low an engine speed and consequent engine stall. There are many ways to accomplish this. One simple way which has been tried successfully is to start the engine, then slip the clutch to spin the flywheel up to a minimum speed. This method can be accomplished either by starting the engine first or electrically spinning the flywheel, then engaging the engine. The first technique can be accomplished with less electrical energy if the engine is in good tune. The second method requires gradual application of throttle during acceleration.

An alternative which requires no more components but provides a somewhat smoother starting algorithm is to start the engine normally, then hydrostatically accelerate the flywheel. This technique, though smooth, means that one hydraulic pump must be always connected to the flywheel mechanism. The consequence of this is that we would always have its associated spin loss. If we choose one of the lower-capacity pumps for this function the loss can be tolerable.

The total flywheel spin loss should be less than 1 hp at peak speed. The largest spin loss is expected to come from the hydraulic pumps. It is generally important to keep the hydraulic pumps operating at the minimum speed to supply the power required. Thus, ideally, a two- or three-speed transmission should be used between the hydrostatic pumps and the flywheel. This will allow minimization of the losses when the power requirements are low. It further gives the system more flexibility and greater dynamic range.

Design Alternatives

Engine selection is based on simplicity and power density. Simplicity would dictate an air-cooled diesel engine. However, power density would greatly favor an air-cooled gasoline engine. Simplicity and maintainability also favor a gasoline production engine. Thus, an air-cooled production engine of about 60 hp is clearly the best alternative.

The flywheel configuration is dependent upon energy-density, safety, construction-cost, reliability, and maintainability trade-offs. Two alternative designs being evaluated are shown sketched in Figures 5 and 6. These designs show different approaches to the five trade-off items above. Figure 5 shows a semiconstant-stress flywheel with an attachment boss. This sort of design requires a larger diameter for a given speed or a higher speed for a given diameter. A physical limit is approximately 24 inches diameter and a maximum comfortable operating speed of about 15,000 rpm. The second possibility is to be conservative and design a semirectangular cross section with an insert hub. This design has the advantage of providing absolute safety from overspeed; i.e., the flywheel is assembled by cooling the center to liquid-nitrogen temperature and heating the outside to a high temperature. The interference fit

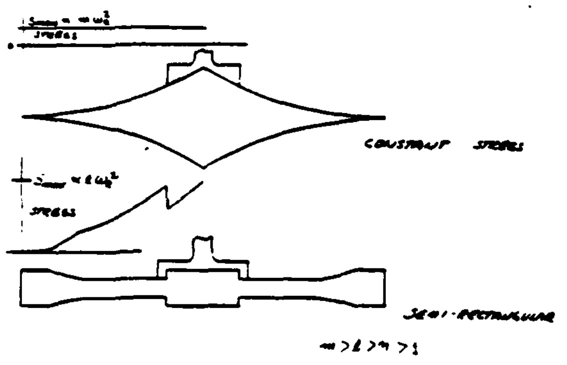

Figure 5: Alternative flywheel designs

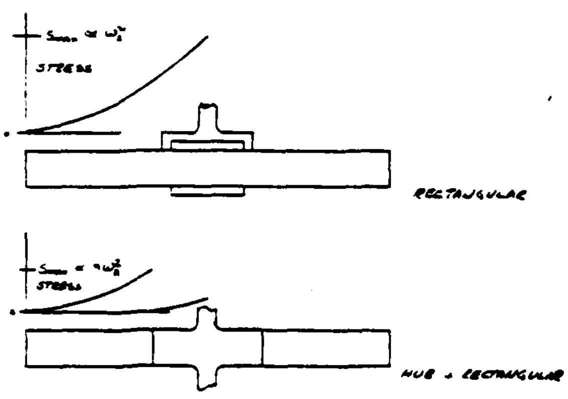

Figure 6: Alternative flywheel designs

is calculated to occur at 20% over the maximum speed. An alternative would be to assemble the system at 20% overspeed, i.e., dynamically assemble the hub and disc. Overspeed of the flywheel causes the disc to fly off the hub, thus preventing the overspeeding of the disc. This 20% margin in speed is easy to adhere to, since 20% overspeed implies a 44% increase in stored energy; i.e., with the characteristics of engines, this overspeed is difficult to achieve if all protective systems fail.

The flywheel systems discussed will have a vacuum/lubrication pump. The idea is simply to use a scavenging pump to remove oil, then to carefully use atmospheric pressure to force oil back into the flywheel case. The calibration of an orifice and the use of constant-viscosity oil will yield a simple yet effective combined lubrication-vacuum system. This pump should be incorporated around the flywheel shaft. An ordinary oil pump as used in automatic transmissions can be utilized. This pump may be used to start the flywheel; in this mode it really is a hydraulic motor.

Conclusion

The flywheel-engine power supply is a relatively straightforward design effort. The engine control is dependent on final decisions on the placement of the starter motor. It is anticipated that the engine will be bump-started by the flywheel and then the engine will spin the flywheel up at a fixed throttle until maximum speed is reached. At this speed the ignition will be automatically cut and the engine decoupled by the engine clutch; the flywheel will supply the power until its speed drops to about half the maximum, at which time the engine is re-engaged and started. This on-off operation is one mode of operation possible. A continuous-operation mode is, of course, also possible, in which the engine is throttled to maintain a given flywheel speed less than the maximum. This operational mode uses the flywheel as a speed stabilizer for the engine. It is not as efficient as the on-off operation; however, it is simpler to implement.

The flywheel power supply is a necessary part of this vehicle design, since high transient torques are expected to be applied to the hydraulic system. These transients are translated into transient or peak power. Without a flywheel for stabilization, the engine would have to be extremely responsive. The flywheel power supply allows us to use other forms of prime mover as well, if they become available. Power sources such as Stirling engines, turbines, and diesels are all viable with the flywheel as an intermediary.

HYDRAULIC SYSTEM CONFIGURATION

As is shown in Figure 3, each leg of the machine is to have three degrees of freedom each powered by a linear actuator. Hydraulic linear actuators are lighter than rotary actuators in equivalent service. The non-linearity of the relationship between actuator displacement and joint angle and the limited range of joint rotation permitted do not present serious problems in this system. The three actuators have distinctive functions. The drive actuator, A, moves the foot back and forth along a path which is, to a good approximation, a straight line parallel to the body axis. The approximate straight line is generated by the four-bar linkage configuration of the upper leg. The shank actuator, B, raises and lowers the foot at the beginning and end of the contact phase and accommodates to variations in working height due to stepping in depressions or on bumps. The abduction-adduction actuator, C, allows the leg to be tilted outward or inward to perform turning maneuvers and to accommodate to side slope conditions.

In order to eliminate valve pressure drop losses and to allow regeneration it would be attractive to use hydrostatic power transmission circuits. However, the weight and cost of providing a separate variable displacement pump for each of 18 actuators is prohibitive. Even a partial system using separate hydrostatic circuits only for the drive actuators is unattractive.

Thus, regeneration using conventional types of hydraulic circuits does not appear possible. However, the load and displacement rates of the drive actuators are related. This feature, coupled with the predictive and coordinating power of the control computer permits design of a single circuit powering all 6 drive actuators which will permit regeneration. Likewise, if all 6 shank actuators are grouped into a single circuit and all 6 abduction-adduction actuators likewise, advantage can be taken of the similarities of load and displacement histories of the actuators in each circuit.

The drive circuit has load-displacement characteristics quite similar to that of a wheeled vehicle. Displacement rate is large and varies only slowly. Load varies widely but not rapidly. Restriction of vehicle operation to wave gaits (14,15) and to those in which legs are raised and lowered at precisely the same instant, that is, those with duty factors of 5/6, 2/3, or 1/2, maintains constant circuit volume. Balanced area actuators will be used for the same reason. Average power in this circuit is high because of the large displacements.

The return portion of the leg cycle presents a problem. As far as possible one would like to convert the kinetic energy of the leg at the end of the contact phase into gravitational potential energy and re-use it to drive the leg forward during the return cycle. This requires provision for the leg to swing forward freely and for the delivery to it of a relatively small amount of energy to make up for frictional losses. This type of action is quite incompatible with the other functions of the drive circuit. Loading on the return phase is low compared to that of the contact phase and average joint rates, while the same as for the contact phase in the tripod gait, (duty factor 1/2) increase to twice the contact phase rate for 2/3 duty factor and five times it for 5/6 duty factor. Free swing of the leg can be allowed by providing a large diameter bypass across the actuator and simultaneously isolating the returning leg from the drive circuit and opening the bypass. The problem of delivering make up energy can best be handled in the way that humans and animals handle it by rapidly lifting the leg at the beginning of the return cycle higher than necessary for ground clearance. This increases the potential energy available to drive the return cycle provided the leg is allowed to drop back to ground clearance height under its own weight after passing the maximum velocity position. Thus, the energy pulse required is effectively delivered by the shank actuator circuit, which is much better suited for this purpose than the drive circuit.

It is proposed to connect the drive actuators of legs on each side of the vehicle in series. The optimum supply arrangement is a separate pump for each of these series circuits. A single pump could be used, but only at the cost of increased losses during turning movements. Control of the displacements of these pumps provides both the primary speed control and the primary steering control. Net flow rates in each of the circuits are to be sensed and their average used as a speed feedback signal and their difference as a directional feedback signal. Thus, the "drive circuit" becomes two circuits with interconnected controls.

The series connection of the drive actuators on each side of the machine, in principle, constrains the rates of the drive actuators of all legs on that side of the vehicle to be the same. In practice, bleed circuits would be used to bypass some flow around the actuators permitting speed variation as required during turning movements and in high duty factor gaits. The bleed circuits also permit actuator velocities to vary during the contact phase. This mode of operation automatically handles the problem of distributing the drive load among the legs. If a leg starts to slip, its share of the load will automatically decrease since the pressure across its actuator will drop throwing a correspondingly larger share of the supply pressure across the actuators of the other legs. The drive circuits are shown on Figure 7.

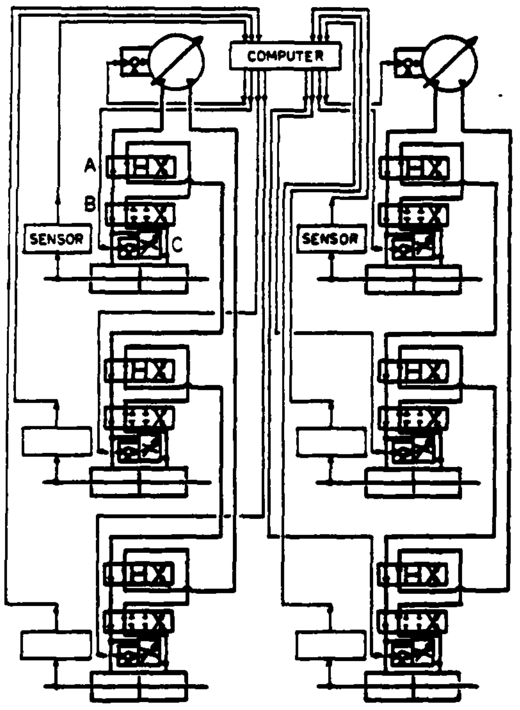

Figure 7: Drive circuit configuration. The heavy weight lines indicate hydraulic components and connections. The lighter weight lines indicate control information paths. Note that relief valve and charge pump circuits are not included on this diagram.

Since flow rate is relatively difficult to measure directly, it is probably best to measure actuator rates and infer flow rates from these. During each cycle segment the computer might select as master leg the leg with the highest rate on each side. If that leg is operated without bleed, its drive actuator velocity obtained from the joint sensors gives the flow rate for that side of the vehicle.

Each of the drive actuator circuits is a constant volume circuit in which flow rate is controlled by pump displacement. Load reversal in the actuators can be used to drive the pump as a motor and, hence, to return power to the flywheel. Although it has multiple actuators which are switched in and out of the circuit, its characteristics are those of a hydrostatic circuit.

Figure 7 is a schematic representation of the hydraulic flow paths and control information paths for this subsystem. The hydraulic elements are indicated by heavy lines and the control information paths by lighter lines. Each balanced area actuator is controlled by three valves. A and B are three-position four way valves and are used primarily as flow switches. Variation of the flow rate through the actuator is controlled by the two way flow control valve C. During the contact phase, flow through C is small and so, although it is subject to the full actuator pressure drop, the energy consumed in the valve will be small. During the return phase, valve C is fully open permitting relatively free swing of the leg. Valve A functions to switch the actuator in and out of the flow circuit. It has two functional states. In the first, the supply is connected to one side of the actuator and the return to the other. In the second, the supply is connected directly to the return, bypassing the actuator. The two ends of the actuator are also connected providing an additional actuator reverse flow path for the return phase. Valve B is provided to allow reversal of the direction in which the actuator is driven allowing positive drive of the return motion when the leg is impeded and for sprinting with increased leg return speed. Valve B is also to be used to hold the leg in position at the end of the return phase in order to synchronize leg placement. Both A and B are conventional, three position four-way control valves, although A is connected in an unusual flow configuration. Note that the flow paths are symmetric with respect to reversal of the direction of flow. This feature provides a capability for driving the machine in reverse.

The operational requirements of the shank actuator circuit are quite different from those of the drive circuit. During the contact phase the shank actuator must either hold position or execute small movements against high load. At the beginning of the return cycle it must lift rapidly against the relatively small shank and foot weight load. During the later portion of the return phase and at the end of that phase it should allow the foot to descend under its own weight.

Since there is such a disparity between load during the contact phase and during the remainder of the cycle, operation as a simple, constant supply pressure, valve controlled servo would require acceptance of severe valve losses during leg lift because of the high rate and low load relative to supply pressure which would have to be high enough to support the contact load. Control using pump displacement would probably have too narrow a bandwidth to accommodate an adequately fast leg lift. In order to overcome this disparity in operating conditions it is proposed to provide energy storage by means of accumulators placed closed to the shank actuators (figure 8). These can be used to

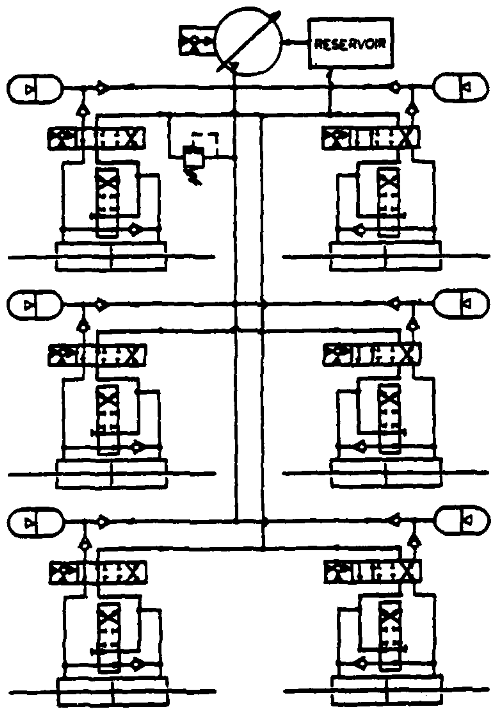

Figure 8: Shank circuit configuration

release a short pulse of energy to power the leg lift with minimal effect on the much more slowly responding supply system. Valve energy losses are minimized by keeping the valve wide open during the energy transfer. The accumulator should be sized to have sufficient stored energy for maximum leg lift. It is charged during the remainder of the swing phase while the actuator is isolated from the supply. A controlled bypass is also provided to allow controlled descent of the leg under its own weight. The shank actuator is isolated from the supply during this phase. Note that in this circuit the pump servo is set up to control supply pressure rather than flow rate as in the drive circuit. The four way control valve is a conventional, critical center valve used in a conventional manner. The bypass valve, B, is used to drop the leg under its own weight and not as the main control element as in the drive circuit. A bypass with a check valve which is also switched by valve B permits the shank to continue to rise freely after delivery of a brief pressure pulse from the accumulator. The pump displacement will be controlled to provide a supply pressure commanded by the control computer. This will allow supply pressure to be matched to weight with a relatively small valving margin.

The operational characteristics of the abduction-adduction circuit are quite different again. In straight ahead walking on level ground, these actuators simply hold position against minimal loads. They primarily come into use when turning. Except for extremely sharp turns or crab-walking, the displacements are small. Loads are highly variable. Because of the relatively small displacements expected in most operating conditions, the valve losses of a conventional valve controlled circuit should be acceptable. The primary control variable of this circuit will be joint rate although position will also be sensed and controlled. The computer will be used to regulate supply pressure in accordance with expected actuator loading. The circuit is shown in Figure 9. Shot pin location will be used on this and the drive circuit to hold position when powered down. The shank will be allowed to rest on its upper stop when powered down, effectively kneeling the vehicle.

CONCLUSIONS

It should be emphasized that the design of the system described here is still developing. Although the broad outlines of the power supply, storage and transmission system have been resolved, many questions pertaining to the details of design and operation remain to be answered. Investigations are in progress of system dynamic response, switching problems, fluid transients and dynamic optimization. These aspects of the system are being studied by a combination of computer simulation and of testing breadboarded subsystems.

Acknowledgement

The work described in this paper was supported by the Defense Advanced Research Projects Agency contract number MDA903-82-K-0058 at the Ohio State University and contract number MDA903-82-C-0028 at the University of Wisconsin.

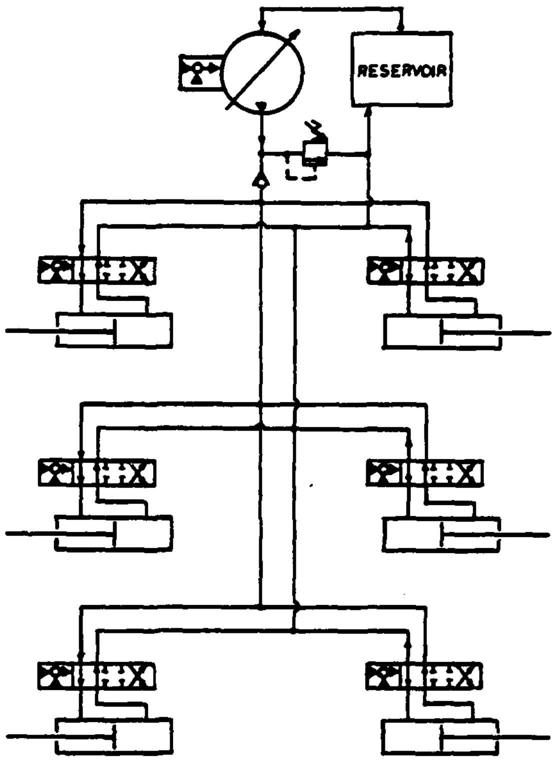

Figure 9: Abduction-Adduction circuit configuration

REFERENCES

1. Tedesco, A.N., et al., Proceedings of Off-road Mobility Research Symposium, International Society for Terrain Vehicle Systems, Washington, D.C., June 1968.

2. "Logistical Vehicle Off-road Mobility," Project TCCO 62-5, February 1967, U.S. Army Transportation Combat Developments Agency, Fort Eustis, Virginia.

3. Kaiser Spyder Specifications, Industrial and Municipal Engineering Corporation, Galva, Illinois.

4. Arola, R.A., et al., "Felling and Bunching Small Timber on Steep Slopes," U.S. Department of Agriculture, Forest Service, North Central Forest Experiment Station, Research Paper NC-203.

5. Mosher, R.S., "Test and Evaluation of a Versatile Walking Truck," Proceedings of Off-road Mobility Research Symposium, International Society for Terrain Vehicle Systems, Washington, D.C., June 1968, pp. 359-379.

6. McGhee, R.B., Olson, K.W., and Briggs, R.L., "Electronic Coordination of Joint Motions for Terrain-Adaptive Robot Vehicles," Paper No. 800382, Proceedings of 1980 SAE Automotive Engineering Congress, Detroit, Michigan, February 1980.

7. Okhotsimski, D.E., et al., "Integrated Walking Robot Simulation and Modelling," Proceedings of Seventh IFAC World Congress, Helsinki, Finland, June 1978, pp. 917-924.

8. Hirose, S. and Umetani, Y., "The Basic Motion Regulation System for a Quadruped Walking Vehicle," ASME Paper 80-DET-34, September 1980.

9. Sutherland, I.E., "A Six-legged Walking Machine," Sutherland, Sproull, and Associates, Inc., Pittsburgh.

10. Gray, Sir James, Animal Locomotion, Weidenfield and Nicholson, London 1968.

11. Waldron, K.J. and Kinzel, G.L., "The Relationship Between Actuator Geometry and Mechanical Efficiency in Robots," Proceedings of Fourth CISM-IFTOMM Symposium on Theory and Practice of Robots, Zaborow, Poland, September 1981.

12. Song, S.M., Vohnout, V.J., Waldron, K.J. and Kinzel, G.L., "Computer-aided Design of a Leg for an Energy Efficient Walking Machine," Proceedings of 7th Applied Mechanisms Conference, Kansas City, December 1981.

13. Beachley, N.H. and Frank, A.A., "Improving Vehicle Fuel Economy with Hybrid Power Systems," SAE Paper No. 780667, 1978.

14. McGhee, R.B. and Frank, A.A., "On the Stability properties of Quadruped Creeping Gaits," Mathematical Biosciences, Vol. 3, No. 3/4, October 1968.

15. Bessonov, A.P. and Umnov, N.V., "The Analysis of Gaits in Six-legged Vehicles According to Their Static Stability," Proceedings of the Symposium on Theory and Practice of Robots and Manipulators, International Centre for Mechanical Sciences, Udine, Italy, September 1973.

Appendix 2

To be presented at ASME Winter Annual Meeting, Phoenix, November 14-16, 1982.

THE DESIGN AND EVALUATION OF A HYDRAULIC ACTUATION SYSTEM FOR A LEGGED ROUGH-TERRAIN VEHICLE

by

K. Srinivasan
Assistant Professor

K.J. Waldron
Professor

and

J.A. Dworak
Graduate Research Associate

Department of Mechanical Engineering
Ohio State University
Columbus, Ohio

ABSTRACT

One of the causes of low efficiency of legged vehicles is the type of actuation system used. This paper describes the overall design and the evaluation of some aspects of a proposed hydraulic actuation system for a six-legged vehicle intended for use in rough terrain. Features of the hydraulic actuation system *designed to improve mechanical efficiency* are described. A combination of linearized dynamic system analysis and computer simulation of the nonlinear dynamic system equations is used to evaluate some aspects of the proposed design. The tradeoff between energy efficient operation and the dynamic performance of the actuation system is investigated. Some criteria for controller design are enumerated.

NOMENCLATURE

A_o	Vehicle center of mass acceleration
A_1, A_2	Polynomials in s
A_p	Piston cross sectional area
a_o	Desired actuator acceleration
B_1, B_2	Polynomials in s
B_p, B_{p1}, B_{p2}	Viscous damping coefficient of piston and load
C_1, C_2	Polynomials in s
C_d	Flow discharge coefficient for servovalve ports
C_{ep}	External leakage coefficient for actuators
C_{ip}	Internal leakage coefficient for actuators
D_1, D_2	Polynomials in s
F_{L1}, F_{L2}	Drive actuator loads
G_{11}, G_{12}, G_{21}, G_{22}	Transfer functions for series connected actuators
g	Acceleration due to gravity
K_a, K_{a1}, K_{a2}	Proportional controller gains
K_c, K_{c1}, K_{c2}	Servovalve flow-pressure coefficient
K_q, K_{q1}, K_{q2}	Servovalve flow gain
K_s	Servovalve gain
K_t	Velocity transducer gain
M	Fraction of vehicle mass assigned to a leg
M_t, M_{t1}, M_{t2}	Average reflected inertia at each actuator corresponding to vehicle mass

N_{11}, N_{12}, N_{21}, N_{22}	Transfer functions for series connected actuators
P_{11}, P_{12}, P_{21}, P_{22}	Actuator chamber pressures
P_r	Return pressure
P_s	Supply pressure
Q_1, Q_2	Flow rates through servovalve ports
s	Laplace variable
V_o	Nominal vehicle velocity
V_{11}, V_{12}	Actuator chamber volumes
V_t	Combined volume of actuator chambers
v_o, v_{o1}, v_{o2}	Desired actuator velocities
W	Area gradient for servovalve port
x_{p1}, x_{p2}	Actuator positions
x_{v1}, x_{v2}	Servovalve spool displacements
α	Angle of incline
β_e	Effective bulk modulus of hydraulic fluid
ΔF	Excess force on the drive actuator
ρ	Density of hydraulic fluid
τ	Servovalve time constant

Subscript

m	Denotes mean value

INTRODUCTION

Legged vehicles have considerable potential for improved mobility over rough terrain as compared to wheeled or tracked vehicles [1]. Some commercial machines using a combination of articulated limbs and wheels have demonstrated improved mobility characteristics over such terrain [2,3]. A number of experimental walking machines have also been built [4,5] and have demonstrated that terrain adaptive features can be included in the designs of legged vehicles.

Though legged vehicles should be more energy efficient for locomotion over soft terrain than wheeled or tracked vehicles [6], fully terrain adaptive legged vehicles have proved much less efficient in practice. Reasons for the low mechanical efficiency of terrain adaptive legged vehicles have been identified by Waldron and Kinzel [7]. These vehicles have tended to use actuator placement geometries which result in a significant amount of "back drive" work during the walking cycle. An actuator is said to be "back driven" when it acts as a brake. The mechanical design of a leg which minimizes the amount of "back drive" work has been described by Song et al. [8]. Another important effect is energy wastage due to the oscillatory nature of the leg motion. This can be reduced by operating the leg as a compound pendulum [7]. It is also desirable to use energy storage and regeneration on legged vehicles for use in very rough terrain. Prime mover operating conditions can then be optimized and energy absorbed when braking can be partially recovered.

Yet another factor in the low mechanical efficiency of some walking machines is the type of power transmission and actuator used. The General Electric Quadruped [3], for example, used a large number of valve controlled hydraulic actuators. Such a method of actuation results in good dynamic response of the actuation system but low mechanical efficiency due to high energy losses at the control valves.

This paper describes the overall design and the evaluation of some aspects of a proposed hydraulic actuation system for a six-legged vehicle for use in rough terrain. Features of the hydraulic actuation system designed to improve mechanical efficiency are described. A combination of linearized dynamic system analysis and computer simulation of the nonlinear dynamic system equations is used to evaluate the proposed design. The tradeoff between energy efficient operation and the dynamic performance of the actuation system is investigated. As a result of the study, some criteria for controller design are enumerated.

HYDRAULIC ACTUATION SYSTEM DESIGN

Hydraulic actuation was chosen for the six-legged vehicle, with an estimated weight of 1361 Kg (3000 lb), because of the higher force-to-weight ratio of a hydraulic system. Hydraulic actuation is, moreover, compatible with the low speed high load nature of the application. Regeneration of energy absorbed when braking is also possible with a hydraulic system when used in conjunction with an energy storage system.

Figure 1 shows a conceptual drawing of the vehicle leg. Each leg of the machine is to have three degrees of freedom each powered by a linear actuator. The three actuators have distinct functions. The drive actuator A moves the foot back and forth along a path which is, to a good approximation, a straight line parallel to the body axis. The approximate straight line is generated by the four-bar linkage configuration of the upper leg. Such a design minimizes the "back drive" work in walking by decoupling vertical and horizontal foot movements and, hence, improves mechanical efficiency [8]. The shank actuator B raises and lowers the foot at the beginning and end of the contact phase and accommodates to variations in working height due to stepping in depressions or on bumps. The abduction-adduction actuator C allows the leg to be tilted outward or inward to perform turning maneuvers and to accommodate to side slope conditions.

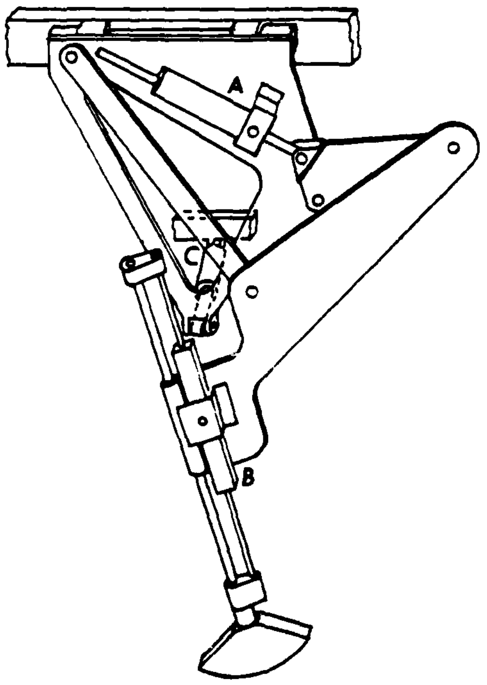

Fig. 1 Conceptual Drawing of Vehicle Leg

The design of the hydraulic actuation system takes advantage of the special nature of the actuator load and velocity requirements for legged vehicles using linkage legs of the type shown in Figure 1. The drive actuator provides most of the power in straight walking. Drive actuator load and velocity requirements are high during the contact phase of the walking cycle. Figure 2 shows drive actuator load requirement as a function of time, for a walking speed of 4.0 Km/hr (2.5 mph) over terrain with a 50% gradient, for a 1361 Kg (3000 lb) vehicle with a specific leg geometry. Figure 3 shows the drive actuator velocity requirements as a function of time for the same leg geometry. A tripod walking gait is assumed [9]. In such a gait, at any one instant, the corner legs on one side of the vehicle and the center leg on the other side of the vehicle are in contact with the ground. The remaining legs are in the return phase of the walking cycle. Though velocity requirements are identical for the drive actuators when walking in a straight line in tripod gait, the load requirements are not. The drive actuators on the corner legs in contact with the ground share half the vehicle load whereas the drive actuator on the center leg on the other side supports half the vehicle load. The drive actuator loads vary with the terrain gradient in a predictable manner. They are very low on level terrain since only frictional resistance is to be overcome. They increase with gradient when going uphill. Corresponding amounts of energy must be absorbed via the drive circuit when going downhill.

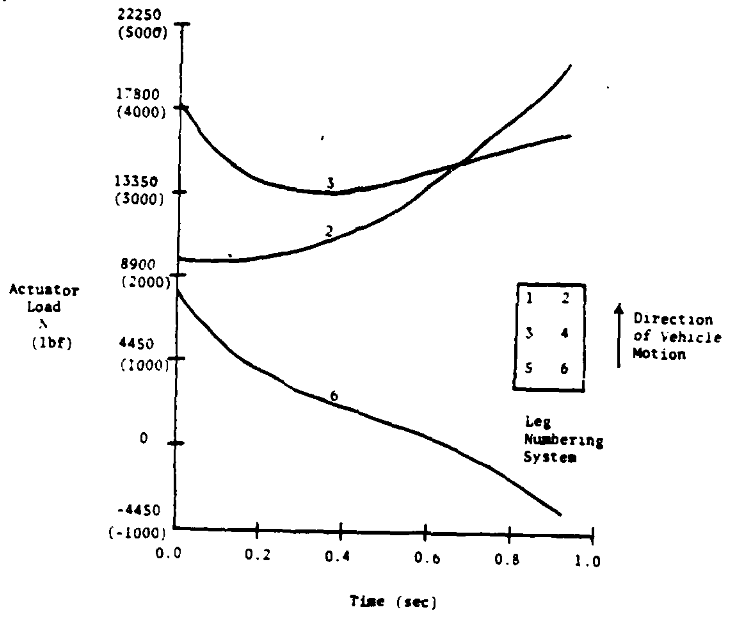

Fig.2 Drive Actuator Load Versus Time-4.0 Km/hr (2,5 mph) Walking Speed, 50% Terrain Gradient, 1361 Kg (3000 lb) Vehicle Weight.

Figure 4 is a schematic representation of one drive circuit design concept. The hydraulic connections are indicated by continuous lines and the control paths by dashed lines. The six drive actuators are grouped into a single hydraulic circuit powered by a variable delivery pump. The pressure difference across the pump is the supply pressure for the drive actuators and is controlled by a feedback loop. Each of the drive actuators is controlled by a four-way electrohydraulic servovalve. The valve actuator combination forms part of a feedback control system in which actuator velocity is the controlled variable.

The desired supply pressure is determined by the control computer as a function of the mean terrain slope and mean vehicle speed. In order to reduce energy losses at the valves, the supply pressure is chosen so as to keep the valves nearly wide open. The pressure drop across the valve, for a given flow rate, is thus reduced.

Each balanced area actuator is controlled by four valves. Valve A is an open center, three-position valve used as a flow switch. During the contact phase, it is actuated as shown on actuators 1, 4 and 5 in Figure 4. During the

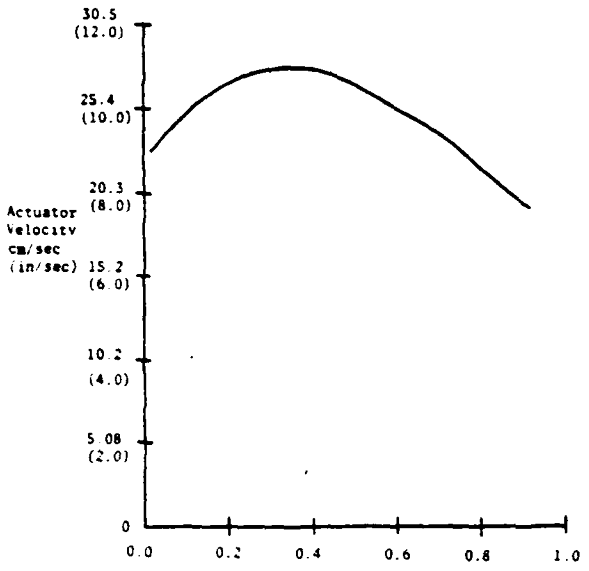

Fig. 3. Drive Actuator Velocity Versus Time - 4.0 Km/hr (2.5 mph) Walking Speed.

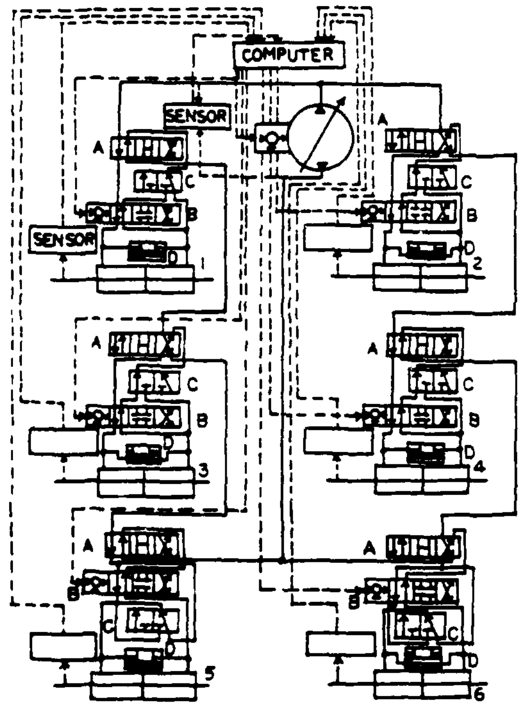

Fig. 4. Schematic Representation of Drive Circuit Concept.

return phase, valve A is actuated in the other direction as shown on actuators 2, 3 and 6 in the figure and isolates the actuators from the drive circuit. Valve B is a continuously variable, electrohydraulic four-way servovalve. With the two-position valve C actuated as shown for actuator 4, valve B controls the actuator motion in a conventional manner. Actuation of valve C as shown for actuators 1 and 5 connects the actuators in series. Valve D is actuated differently during the contact and return phases of the walking cycle as shown in Figure 4.

In tripod gait, the drive actuators on the corner legs in contact with the ground are connected in series. The series connection handles the problem of distributing the drive load among the corner legs as shown in Figure 2. The drive actuator on the center leg on the other side of the vehicle is also in contact with the ground and is controlled by the four-way servovalve B in a conventional manner. The other three actuators are on returning legs and are isolated from the drive circuit. Figure 4 shows the hydraulic circuit operation for tripod gait.

The return phase of the walking cycle requires a completely different mode of control as described by Waldron et al [10] if excessive energy losses are to be avoided. The drive actuators on the returning legs are isolated from the drive circuit and allowed to swing back freely by opening a large diameter bypass across each such actuator. The shank circuit, used to power the shank actuators, is also capable of providing make up energy to drive actuators on returning legs [10]. The control valves in the drive circuit allow implementation of the desired control action during the return phase.

Turning maneuvers are accommodated naturally by the drive circuit. The drive actuators on the inside of the turn curve would be operated at a lower speed than those on the outside of the turn curve in response to coordinated command inputs to the velocity servomechanisms.

While walking downhill at constant speed, the drive actuators experience a pressure rise instead of a pressure drop. The desired supply pressure is set by the control computer to be a negative value. The variable delivery pump is driven as a motor, in effect, and the recovered energy can be stored if there is provision to do so.

The design features and the dynamic response of drive circuit components are of primary interest in this paper. The operational requirements and the designs of the shank circuit and the abduction-adduction circuit are described elsewhere by Waldron et al [10].

DYNAMIC ANALYSIS OF DRIVE CIRCUIT COMPONENTS

The design of the drive circuit and its proposed manner of operation have been governed by the need to reduce energy losses and improve overall efficiency of operation. The resulting design features and operational characteristics affect the dynamic performance of the actuation system significantly. In this section, two components of the drive circuit are investigated. Firstly, the dynamic performance of electrohydraulic velocity servomechanisms operated with nearly wide open servovalves is examined. Secondly, the dynamic performance of servomechanisms employing drive actuators connected in series is considered. The nonlinear equations describing system behaviour are simplified to enable linear analysis of the dynamic performance. The nonlinear equations are used subsequently to simulate the dynamic behaviour of the drive circuit components on a digital computer.

Dynamic Analysis of a Balanced-Area Actuator Controlled by a Four-Way Servovalve

Consider the single balanced area actuator controlled by a four-way servovalve as shown in Figure 5. The valve-actuator combination forms part of the velocity servomechanism shown in Figure 6. The equations describing the dynamic behaviour of the system are given below and can be found in most standard textbooks on electrohydraulic servomechanisms [11]. The symbols are defined in the

Nomenclature section of the paper.

The valve flow equations for a critical center servovalve are given below assuming the valve is to be open in the direction shown in Figure 5.

$$Q_1(t) = C_d\ W\ x_{v1}(t) \sqrt{\frac{2}{\rho}\left(P_s - P_{11}(t)\right)} \tag{1}$$

$$Q_2(t) = C_d\ W\ x_{v1}(t) \sqrt{\frac{2}{\rho}\left(P_{12}(t) - P_r\right)} \tag{2}$$

The continuity equations for the two sides of the actuator are

$$Q_1(t) - C_{ip}\ (P_{11}(t) - P_{12}(t)) - C_{ep}\ P_{11}(t) = \frac{dV_{11}(t)}{dt} + \frac{V_{11}(t)}{\beta_e}\frac{dP_{11}(t)}{dt} \tag{3}$$

$$C_{ip}\ (P_{11}(t) - P_{12}(t)) - C_{ep}\ P_{12}(t) - Q_2(t) = \frac{dV_{12}(t)}{dt} + \frac{V_{12}(t)}{\beta_e}\frac{dP_{12}(t)}{dt} \tag{4}$$

The volumes $V_{11}(t)$, $V_{12}(t)$ of the actuator chambers are

$$V_{11}(t) = A_p\ x_{p1}(t) \tag{5}$$

$$V_{12}(t) = V_t - V_{11}(t) \tag{6}$$

The load equation for the actuator piston is given by

$$A_p(P_{11}(t) - P_{12}(t)) = F_{L1}(t) + B_p\ \frac{dx_{p1}(t)}{dt} + \Delta F(t) \tag{7}$$

where $F_{L1}(t)$ is the computed load on the drive actuator corresponding to a nominal vehicle motion trajectory and $\Delta F(t)$, the excess force on the drive actuator, results in deviations from the nominal trajectory.

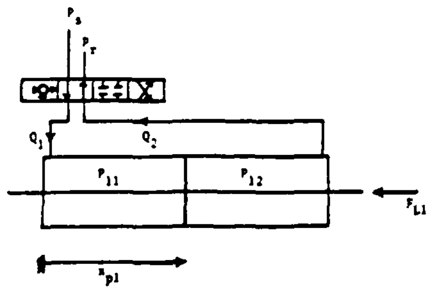

Fig. 5. Valve-Actuator Combination for Center Leg.

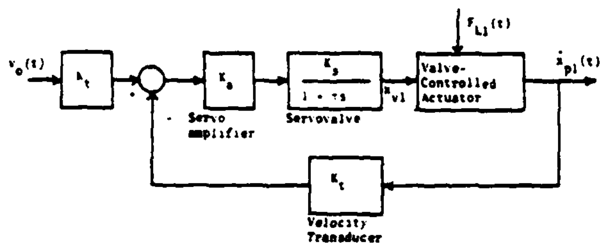

Fig. 6. Leg Velocity Servomechanism

Appendix A details the derivation of the relationship between the excess force $\Delta F(t)$ and the velocity and acceleration of the actuator and the vehicle center of mass. The resulting equation indicates that ΔF is a nonlinear and time varying function of actuator position, velocity and acceleration.

$$\Delta F(t) = \frac{MV_o}{v_o}\left(\frac{V_o}{v_o}\ddot{x}_{p1}(t) - \frac{V_o a_o}{v_o^3}\dot{x}_{p1}^2(t) + \frac{A_o}{v_o^2}\dot{x}_{p1}^2(t) - A_o\right) \tag{8}$$

V_o, A_o, v_o, a_o are computed nominal vehicle velocity and acceleration and corresponding actuator velocity and acceleration respectively. They are all functions of linkage or actuator position. M is a fraction of the vehicle mass and corresponds to the fraction of the vehicle weight supported by the leg. It is a time varying function and depends on the positions of all the legs in contact with the ground. For the actuators on the center legs in tripod gait, M is one-half the vehicle mass.

The equations listed above are adequate to define the actuator motion $x_{p1}(t)$ in terms of the servovalve spool displacement $x_{v1}(t)$ and the actuator load $F_{L1}(t)$ if the nominal vehicle and actuator velocities and accelerations and the vehicle mass fraction M are known. For purposes of a linear analysis, the equations need to be simplified considerably. Appendix A indicates that the simplest linearized form of equation (8) is

$$\begin{aligned}\Delta F(t) &= \frac{MV_o^2}{(v_o)_m^2}\ddot{x}_{p1}(t)\\ &\triangleq M_t\,\ddot{x}_{p1}(t)\end{aligned} \tag{9}$$

where M_t is the average reflected inertia at the actuator corresponding to the vehicle mass. The remaining dynamic equations can be linearized in a standard manner for small perturbations about a mean operating condition, resulting in a transfer function relationship between x_{p1}, x_{v1} and F_{L1} [11].

$$sx_{p1}(s) = \frac{\frac{K_q}{A_p}x_{v1}(s) - \frac{K_{ce}}{A_p^2}\left(1 + \frac{V_t s}{4\beta_e K_{ce}}\right)F_{L1}}{\frac{V_t M_t}{4\beta_e A_p^2}s^2 + \left(\frac{K_{ce}M_t}{A_p^2} + \frac{B_p V_t}{4\beta_e A_p^2}\right)s + \left(1 + \frac{B_p K_{ce}}{A_p^2}\right)} \tag{10}$$

where

$$K_q = \text{Servovalve flow gain, } \frac{\partial\left(\frac{Q_1+Q_2}{2}\right)}{\partial x_{v1}}$$

$$\triangleq C_d W \sqrt{\frac{1}{\rho}\,(P_s-(P_{11}-P_{12})_m)} \tag{11}$$

$$K_c = \text{Servovalve flow-pressure coefficient, } \frac{-\partial\left(\frac{Q_1+Q_2}{2}\right)}{(P_{11}-P_{12})}$$

$$\triangleq \frac{C_d W (x_{v1})_m}{2} \sqrt{\frac{1}{\rho(P_s-(P_{11}-P_{12})_m)}}, \tag{12}$$

and

$$K_{ce} \triangleq K_c + C_{ip} + \frac{C_{ep}}{2} \tag{13}$$

The servovalve dynamics are represented by a first order model [12] as shown in Figure 6.

Energy efficient operation of the servomechanism requires that P_s be adjusted for different operating conditions to keep the pressure drop across the servovalve low for a given flow rate. As a consequence, the value of the flow gain K_q is low. This fact, in itself, is not a disadvantage if the controller design can be altered, as mean operating conditions change, to maintain a high loop gain. However, the gains K_q and K_c are more sensitive to changes in the supply pressure P_s, the mean load pressure $(P_{11}-P_{12})_m$ and the mean spool displacement $(x_{v1})_m$ as a result of the low servovalve pressure drops.

$$\frac{\partial K_q}{\partial P_s} = \frac{C_d W}{2\sqrt{\rho(P_s-(P_{11}-P_{12})_m)}} \tag{14}$$

$$\frac{\partial K_q}{\partial (P_{11}-P_{12})_m} = -\frac{C_d W}{2\sqrt{\rho(P_s-(P_{11}-P_{12})_m)}} \tag{15}$$

$$\frac{\partial K_c}{\partial (x_{v1})_m} = \frac{C_d W}{2\sqrt{\rho(P_s-(P_{11}-P_{12})_m)}} \tag{16}$$

Therefore, parameters in the linear system represented by equations (10)-(13) and in the controlled system of Figure 6 change significantly during the contact phase of the walking cycle. Therefore, one objective of controller design based on the linear model of the system should be to reduce the sensitivity of the controlled system performance to changes in the parameters K_q and K_c.

Another consequence of nearly wide open servovalve operation during the contact phase is a high value of the flow-pressure coefficient K_{ce}. Higher K_{ce} results in a lower stiffness of the velocity servomechanism to the actuator load F_{L1}. Since the actuator load F_{L1} is expected to be high compared to the inertial load ΔF for a legged vehicle, another objective of controller design should be to improve the servomechanism stiffness.

Some compliance is desirable in vehicles used for locomotion over rough terrain. The achievement of active compliance for a legged vehicle, by appropriate formulation of the supervisory control algorithm, has been described by Klein and Briggs [13]. It is desirable to modify the supervisory control algorithm to utilize the velocity servomechanism compliance to the extent possible.

Let us consider the following parameter values for the servomechanism components for the purpose of conducting a linear analysis.

Servovalve

$W = 2.54$ cm (1.0 in)

$C = 0.61$

$(x_{v1})_{max} = 0.0770$ cm (0.0303 in)

$\tau = 0.00634$ sec

Actuator

$A_p = 16.1\ \text{cm}^2\ (2.5\ \text{in}^2)$

$V_t = 492\ \text{cm}^3\ (30\ \text{in}^3)$

$$C_{ip} = C_{ep} = 0\ \frac{\text{cm}^3/\text{sec}}{\text{N/cm}^2}\left(0\ \frac{\text{in}^3/\text{sec}}{\text{psi}}\right)$$

Load

$$M_t = 13847\ \frac{\text{N-sec}^2}{\text{m}}\ (949.5\ \text{slugs})$$

$$B_p = 0\ \frac{\text{Kg-sec}}{\text{m}}\left(0\ \frac{\text{lbf-sec}}{\text{in}}\right)$$

Fluid Properties

$$\rho = 833\ \frac{\text{N-sec}^2}{\text{m}^4}$$

$$\left(0.78 \times 10^{-4}\ \frac{\text{lbf-sec}^2}{\text{in}^4}\right)$$

$$\beta_e = 68975\ \frac{\text{N}}{\text{cm}^2}\left(100{,}000\ \frac{\text{lbf}}{\text{in}^2}\right)$$

The parameters correspond to a 56.7 ℓpm (15 gpm) two-stage electrohydraulic servovalve. The drive actuator is sized to provide adequate force and flow rate capability for a 1361 Kg (3000-lb) six legged vehicle using the leg design referred to in Figures 2,3 to operate at speeds of up to 8.0 Km/hr (5 mph) and over terrain with up to a 50% gradient, if a maximum supply pressure of 2069 N/cm^2 (3000 psi) is available. The leg design being considered also results in a leg cycle time of approximately 0.93 seconds for a tripod gait at 8.0 Km/hr (5 mph). The reference input to the velocity servomechanism is, therefore, a periodic input with the same period as the leg cycle time.

If a fixed controller is used in the servomechanism of Figure 6, the system dynamic response and the operating efficiency would change with the mean operating conditions. The variation in the system dynamic response is reduced if the supply pressure is maintained constant at a value of one and one-half times the maximum expected load pressure, but the system efficiency variation is greater [11]. If the supply pressure is adjusted to match the mean operating conditions, the efficiency variation is reduced but the system dynamic response varies considerably.

Table 1 indicates the closed velocity loop performance for a proportional controller with gain K_a, at different operating conditions. For each operating condition, the mean load pressure at the actuator and the mean actuator velocity are also tabulated. The controller gain K is selected for each operating condition to give a 60° phase margin based on the linearized servomechanism model for that operating condition. The results in Table 1 indicate that variation in the system dynamic performance for a wide range of operating conditions is reduced by controller adjustment. Also, for a specified operating condition, the system dynamic performance can be maintained at about the same level even if the supply pressure is reduced to improve the system operating efficiency. For instance, supply pressure reduction from 1655 N/cm^2 (2400 psi) to 1104 N/cm^2 (1600 psi) for a 4.0 Km/hr (2.5 mph) walking speed over a 50% gradient improves the mean operating efficiency from 56% to 84% while controller adjustment maintains the same dynamic performance. The closed loop bandwidth and the steady state compliance achievable also seem appropriate for the application. They should result in the actual leg velocity tracking the desired leg velocity reasonably well, for the range of walking speeds and leg loads expected.

Gradient	Mean Load Pressure $(P_{l1}-P_{l2})_m$ (N/cm^2)	Walking Speed (km/hr)	Mean Actuator Velocity $(\dot{x}_{pl})_m$ (cm/sec)	Supply Pressure P_s (N/cm^2)	Gain $K_aK_sK_t$ (sec)	3-db Closed Velocity Loop Bandwidth (Hz)	Steady State Error for Unit Velocity Step Input (cm/sec)	Steady State Compliance X 10^6 (cm/sec-N)
0%	7.2	1.6	9.9	276	0.0164	16.0	0.10	1.14
		4.0	24.9	276	0.0518	18.6	0.034	0.97
				552	0.0152	16.7	0.078	1.11
		8.0	49.5	552	0.0426	19.3	0.026	0.93
17.6%	364	1.6	9.9	552	0.031	17.0	0.066	1.06
				828	0.0063	14.0	0.181	1.20
				1104	0.0028	12.5	0.283	1.18
		4.0	24.9	552	0.098	19.9	0.022	0.89
				828	0.0201	17.3	0.065	1.07
				1104	0.0089	15.7	0.11	1.15
		8.0	49.5	828	0.0485	19.4	0.028	0.92
				1104	0.0214	18.0	0.049	1.02
50%	928	1.6	9.9	1104	0.0348	17.3	0.061	1.07
				1380	0.00658	14.1	0.177	1.19
				1655	0.00291	12.6	0.277	1.17
		4.0	24.9	1104	0.111	20.1	0.02	0.88
				1380	0.0213	17.3	0.062	1.06
				1655	0.0090	15.7	0.110	1.15

Table 1. Center Leg Velocity Servomechanism Performance Based on a Linear Model.

A proportional controller is the simplest type of controller that can be used. Alternative controllers can be designed to improve some aspects of the system dynamic performance at the expense of others. The closed velocity loop performance given in Table 1 indicates, however, that a generally satisfactory level of performance can be achieved even with proportional control, if the controller gain is adjusted as mean operating conditions change. The details of the controller gain adaptation would need to be investigated further to ensure satisfactory transient response of the controlled system as the operating conditions change.

As a first step towards a better evaluation of the closed loop system performance, its dynamic behaviour during the contact phase is simulated using the Continuous Systems Modeling Program, CSMP [14]. The nonlinear dynamic equations (1)-(8) are simulated for some of the operating conditions noted in Table 1. The load force $F_{L1}(t)$ on the drive actuator, the share of vehicle mass assigned to the leg M, the mean vehicle velocity V_o, the desired actuator velocity v_o and acceleration a_o are computed as functions of time corresponding to a constant walking speed in tripod gait and are used to characterize the actuator loading in equations (7) and (8). Because of the manner of computation of M, the simulation is valid only for small variations of the actuator trajectory about the nominal trajectory corresponding to the constant speed tripod gait assumed. A proportional controller with gain K_a and a first order representation of servovalve dynamics are assumed as in Figure 6. The servovalve and actuator parameters and the fluid properties noted in equation (17) are used in the simulation. Limits on the servovalve spool displacement and upper and lower pressure limits of P_s and 0 are also included in the simulation. The dynamics of the supply pressure control loop are neglected and P_s is assumed to be constant for each simulation run.

Figure 7 shows the simulated response of the velocity servomechanism during the contact phase of the walking cycle. The velocity error is low except when the leg contacts the ground. The stiffness of the servomechanism is acceptable but is limited by the supply pressure of 1104 N/cm^2 (1600 psi), as indicated by the wide open servovalve during the initial part of the contact phase. The average efficiency of the valve actuator combination during the contact phase is approximately 84%. It should also be noted that the proportional controller gain used for Figure 7 is a factor of 1.5 times the gain listed in Table 1 for the same operating conditions. Other simulation runs also indicated that controller gains higher than those suggested by the linear analysis can be used in the simulation without degrading system stability.

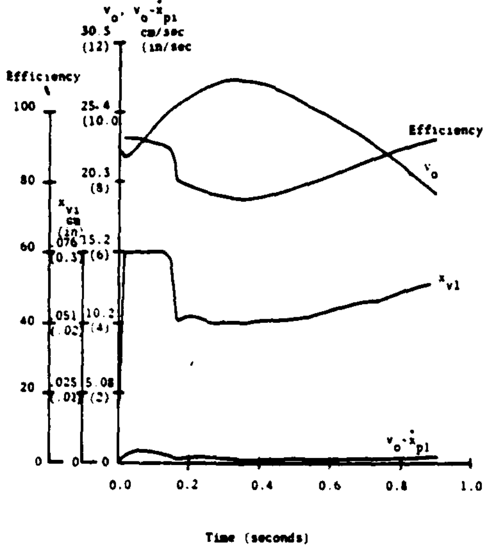

Fig. 7. Simulated Response of the Velocity Servomechanism - P_S = 1104 N/cm^2 (1600 psi), V_o = 4.0 Km/hr (2.5 mph), $K_a K_s K_t$ = 0.162 sec, 50% Terrain Gradient.

The simulation runs noted in Figure 8 indicate that average efficiencies of about 80% can be achieved together with good dynamic response at moderate and low speeds and at moderate and high load levels. Efficiencies at high speeds and low loads are limited by the relatively large valve pressure drops at high flow rates and by the fact that servovalve supply pressures for two-stage servovalves have to be maintained higher than a threshold level for proper operation, even for low load levels. Another limitation on reducing supply pressure is that the servovalve dynamic response degrades as the supply pressure decreases. There are definite efficiency gains to be achieved, however, by oversizing servovalves. For instance, if a 113 ℓpm (30 gpm) servovalve were used for the present application, the average efficiency for a 8.0 km/hr (5 mph) walking speed, 18% terrain gradient and 517 N/cm^2 (750 psi) supply pressure would be 70% instead of the 44% noted for curve (c) in Figure 8.

The analysis and simulation results indicate that the linear model of servomechanism dynamics represented by equations (8)-(9) and Figure 6 leads to the selection of lower controller gains than those suggested by the simulation. One possible refinement in the model is a describing function description of the kinematics of the linkage leg [15]. The resulting model would enable a better evaluation of proposed controller designs than the linear model and would have reduced computational requirements as compared to the computer simulation presently used.

The results cited also indicate the importance of controller adjustment as mean operating conditions change, to help achieve good servomechanism dynamic response and operational efficiency. The nonlinear and time varying nature of the servomechanism dynamics suggests that controller designs requiring an accurate model of the system dynamics would not be feasible. Since controller adaptation is considered desirable, a model reference adaptive control scheme may provide the necessary framework for controller adaptation. Such a controller has the additional advantage of not requiring a complete model of the system dynamics [16].

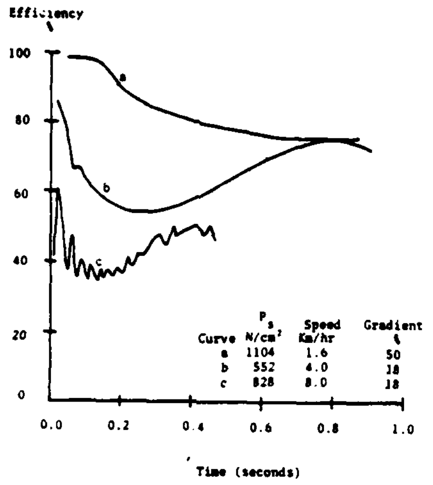

Fig. 8. Efficiencies During the Contact Phase for Different Operating Conditions.

Dynamic Analysis of Balanced - Area Actuators Connected in Series and Controlled by Four-Way Servovalves

Figure 9 shows two balanced-area actuators connected in series and controlled by servovalves. The dynamic equations describing system behaviour are straightforward extensions of equations (1)-(8) and are not listed here. The pressures P_{12} and P_{21} in the actuator chambers connected to each other are assumed to be the same. The series connection of the actuators couples their dynamic behaviour. The coupled differential equations can be linearized as for the single cylinder case. *We make the additional assumption that the actuator internal and external* leakage coefficients C_{ip} and C_{ep} are negligible since the mean servovalve spool displacements $(x_{v1})_m$ and $(x_{v2})_m$ are high during the contact phase and result in large values of the valve flow-pressure coefficients K_{c1} and K_{c2}.

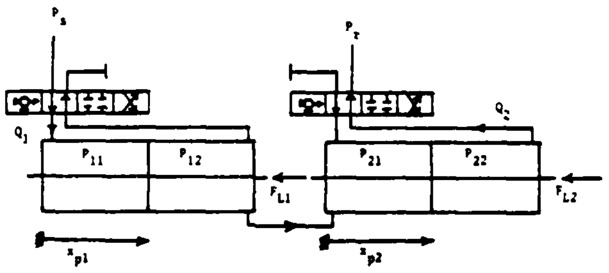

Fig. 9. Valve-Actuator Configuration for Corner Legs.

The resulting coupled fourth-order differential equations are given below, assuming the servovalves to be open in the directions shown in Figure 8.

$$A_1(s)x_{p1}(s) + B_1(s)x_{p2}(s) = C_1(s)x_{v1}(s) + D_1(s)\ F_{L1}(s) \tag{17}$$

$$A_2(s)x_{p1}(s) + B_2(s)x_{p2}(s) = C_2(s)x_{v2}(s) + D_2(s)\ F_{L2}(s) \tag{18}$$

where

$$A_1(s) \triangleq \frac{4K_{c1}K_{c2}V_t}{\beta_e}\left(1 + \frac{V_t s}{4\beta_e}\left[\frac{1}{K_{c1}} + \frac{1}{K_{c2}}\right] + \frac{V_t^{\,2}s^2}{16\beta_e^{\,2}K_{c1}K_{c2}}\right)(M_{t1}s^2+B_{p1}s)$$

$$+ A_p^{\,2}\left(2K_{c2} + \frac{V_t}{2\beta_e}s\right)\left(2K_{c1} + \frac{3V_t}{2\beta_e}s\right) \tag{19}$$

$$B_1(s) \triangleq -A_p^{\,2}\left(2K_{c1} + \frac{V_t s}{2\beta_e}\right)\left(2K_{c2} + \frac{V_t s}{2\beta_e}\right) \tag{20}$$

$$C_1(s) \triangleq \frac{A_p V_t K_{q1}}{\beta_e}\left(2K_{c2} + \frac{V_t s}{2\beta_e}\right) \tag{21}$$

$$D_1(s) \triangleq -\frac{4K_{c1}K_{c2}V_t}{\beta_e}\left(1 + \frac{V_t s}{4\beta_e}\left[\frac{1}{K_{c1}} + \frac{1}{K_{c2}}\right] + \frac{V_t^{\,2}s^2}{16\beta_e^{\,2}K_{c1}K_{c2}}\right) \tag{22}$$

$$A_2(s) \triangleq -A_p^{\,2}\left(2K_{c1} + \frac{V_t s}{2\beta_e}\right)\left(2K_{c2} + \frac{V_t s}{2\beta_e}\right) \tag{23}$$

$$B_2(s) \triangleq \frac{4K_{c1}K_{c2}V_t}{\beta_e}\left(1 + \frac{V_t s}{4\beta_e}\left[\frac{1}{K_{c1}} + \frac{1}{K_{c2}}\right] + \frac{V_t^{\,2}s^2}{16\beta_e^{\,2}K_{c1}K_{c2}}\right)(M_{t2}s^2+B_{p2}s)$$

$$+ A_p^{\,2}\left(2K_{c1} + \frac{V_t s}{2\beta_e}\right)\left(2K_{c2} + \frac{3V_t}{2\beta_e}s\right) \tag{24}$$

$$C_2(s) \triangleq \frac{K_{q2}A_p V_t}{\beta_e}\left(2K_{c1} + \frac{V_t}{2\beta_e}s\right) \tag{25}$$

$$D_2(s) \triangleq -\frac{4K_{c1}K_{c2}V_t}{\beta_e}\left(1 + \frac{V_t s}{4\beta_e}\left[\frac{1}{K_{c1}} + \frac{1}{K_{c2}}\right] + \frac{V_t^{\,2}s^2}{16\beta_e^{\,2}K_{c1}K_{c2}}\right) \tag{26}$$

The linearized valve coefficients K_{q1}, K_{q2}, K_{c1}, K_{c2} in the equations above are defined by

$$K_{q1} \triangleq \left(\frac{\partial Q_1}{\partial x_{v1}}\right)_m = C_d W\sqrt{\frac{2}{\rho}(P_s - P_{11})_m}$$

(27a)

$$K_{q2} \triangleq \left(\frac{\partial Q_2}{\partial x_{v2}}\right)_m = C_d W\sqrt{\frac{2}{\rho}(P_{22})_m}$$

$$K_{c1} \triangleq -\frac{1}{2}\left(\frac{\partial Q_1}{\partial P_{11}}\right)_m = \frac{C_d W (x_{v1})_m}{4}\sqrt{\frac{2}{\rho(P_s-(P_{11})_m)}}$$

(27b)

$$K_{c2} \triangleq \frac{1}{2}\left(\frac{\partial Q_2}{\partial P_{22}}\right)_m = \frac{C_d W (x_{v2})_m}{4}\sqrt{\frac{2}{\rho(P_{22})_m}}$$

M_{t1}, M_{t2} are average reflected inertias at the two actuators corresponding to the vehicle mass.

The dynamics of the valve-actuator combinations are therefore represented by a multivariable system with two control inputs $x_{v1}(t)$ and $x_{v2}(t)$, two disturbance inputs $F_{L1}(t)$ and $F_{L2}(t)$ and two outputs $\dot{x}_{p1}(t)$ and $\dot{x}_{p2}(t)$.

$$\begin{bmatrix} sx_{p1}(s) \\ sx_{p2}(s) \end{bmatrix} = \begin{bmatrix} G_{11}^{P}(s) & G_{12}^{P}(s) & N_{11}^{P}(s) & N_{12}^{P}(s) \\ G_{21}^{P}(s) & G_{22}^{P}(s) & N_{21}^{P}(s) & N_{22}^{P}(s) \end{bmatrix} \begin{bmatrix} x_{v1}(s) \\ x_{v2}(s) \\ F_{L1}(s) \\ F_{L2}(s) \end{bmatrix} \qquad (28)$$

The elements of the transfer function matrix are defined in terms of the polynomials in equations (19)-(26).

The same parameter values are used as in the single actuator case with the exception of the reflected inertia M_t and the servovalve time constant τ. M_{t1} and M_{t2} are each assumed to be 6924 N-sec/m² (474.7 slugs) and the servovalve dynamics are considered to be negligible. The servovalves in the two servomechanisms are assumed to operate identically at steady state resulting in the linearized valve coefficients in equation (27) being the same for the two servovalves. The coupled servomechanisms, therefore, have identical dynamic characteristics.

Table 2 indicates the closed velocity loop performance based on the linear model, for proportional controllers with identical gains K_a for the two servomechanisms. K_a is selected to be the same as for the single actuator case at the same operating conditions. The resulting values of steady state accuracy and compliance are therefore identical to those in Table 1 for similar operating conditions. Table 2 indicates, however, that the 3-db closed loop bandwidths are lower for the coupled servomechanisms. The closed loop bandwidth is evaluated assuming that the two servomechanisms have identical reference inputs v_{o1}, v_{o2}. Computation of the closed loop poles indicated that the system was stable in the three cases examined. The degree of stability of the coupled servomechanisms was not determined.

Gradient	Mean Load Pressure $(P_{11}-P_{22})_m$ (N/cm²)	Walking Speed (Km/hr)	Mean Actuator Velocity $(\dot{x}_{p1})_m, (\dot{x}_{p2})_m$ (cm/sec)	Supply Pressure P_s (N/cm²)	Gain $K_aK_sK_t$ (sec)	3-db Closed Velocity Loop Bandwidth (Hz)	Steady State Error for Unit Velocity Step Input (cm/sec)
0%	7.2	4.0	24.9	276	0.0818	14.2	0.034
17.6%	364	8.0	49.5	828	0.0485	14.5	0.028
50%	928	4.0	24.9	1104	0.111	15.1	0.020

Table 2. Corner Leg Velocity Servomechanism Performance Based on a Linear Model

A more extensive investigation of the dynamic behaviour of the series-connected actuators is called for and will be undertaken shortly. It is expected however that, as for the single actuator case, there will be a need to adjust controller parameters as a function of the mean operating conditions to achieve high operating efficiency and good dynamic response of the velocity servomechanisms.

The controller analysis and design problem is more complex for the series-connected actuators because of the multivariable nature of the system dynamics. Available frequency response techniques for multivariable control system analysis and design should however prove to be applicable here [17].

CONCLUSIONS

The above analysis is useful in evaluating the drive circuit concept described. It indicates a weakness at low load and high speed - an important condition for this application. Moreover, the series connection of the actuators in this drive circuit concept would pose problems in walking gaits other than the tripod gait. An alternative drive circuit concept described by Waldron et al. [10] uses a bleed rate control configuration and should be superior to the drive circuit described here for both these conditions of operation. However, its dynamic response is expected to be inferior to that of the drive circuit described here. The analytical and simulation study will be extended to the alternative drive circuit concept to allow a proper comparison of both designs.

The analysis and simulation results presented indicate that the controller design problem for the velocity servomechanisms becomes more complex as a result of two features of the hydraulic system designed to improve mechanical efficiency-servovalve operation at low pressure drops and the series connection of drive actuators for load distribution. However, controller adjustment as a function of the mean operating conditions does enable higher operating efficiencies without degrading system dynamic response. The study also suggests appropriate techniques for dynamic system analysis and controller design.

Because of the multiple functions required of each hydraulic circuit, the design of these circuits is exceedingly complex and very specific to the application. Although not applicable to other types of robots at present, these ideas can be expected to become important as new and more complex robot concepts are developed. Expected developments in dynamic system analysis and controller design will however be relevant to other current robotic applications.

ACKNOWLEDGEMENT

The work described in this paper was supported by the Defense Advanced Research Projects Agency Contract No. MDA903-82-K-0058 at the Ohio State University.

REFERENCES

1. Morrison, R.A., "Iron Mule Train," Proceedings of Off-Road Mobility Research Symposium, International Society for Terrain Vehicle Systems, Washington, D.C., June 1968.

2. Kaiser Spyder Specifications, Industrial and Municipal Engineering Corporation, Galva, Illinois.

3. Arola, R.A. et al., "Felling and Bunching Small Timber on Steep Slopes," U.S. Department of Agriculture, Forest Service, North Central Forest Experiment Station, Research Paper NC-203.

4. Mosher, R.S., "Test and Evaluation of a Versatile Walking Truck," Proceedings of Off-Road Mobility Research Symposium, International Society for Terrain Vehicle Systems, Washington, D.C., June 1968.

5. McGhee, R.B., Olson, K.W., and Briggs, R.L., "Electronic Coordination of Joint Motions for Terrain-Adaptive Robot Vehicles," Paper No. 800382, Proceedings of 1980 SAE Automotive Engineering Congress, Detroit, Michigan, Feb. 1980.

6. Bekker, M.G., Theory of Land Locomotion, The University of Michigan Press, Ann Arbor, Michigan, 1956.

7. Waldron, K.J., and Kinzel, G.L., "The Relationship Between Actuator Geometry and Mechanical Efficiency in Robots," Proceedings of Fourth CISM-IFTOMM Symposium on Theory and Practice of Robots, Zaborow, Poland, Sept. 1981.

8. Song, S.M., Vohnout, V.J., Waldron, K.J., and Kinzel, G.L., "Computer-aided Design of a Leg for an Energy Efficient Walking Machine," Proceedings of 7th Applied Mechanisms Conference, Kansas City, Dec. 1981.

9. Bessonov, A.P., and Umnov, N.V.,"The Analysis of Gaits in Six-Legged Vehicles According to Their Static Stability," Proceedings of the Symposium on Theory and Practice of Robots and Manipulators, International Centre for Mechanical Sciences, Udine, Italy, Sept. 1973.

10. Waldron, K.J., Frank, A.A., and Srinivasan, K., "The Use of Mechanical Energy Storage in an Unconventional, Rough Terrain Vehicle," Proceedings of the 17th Intersociety Energy Conversion Engineering Conference, Los Angeles, California, Aug. 1982.

11. Merritt, H.E., Hydraulic Control Systems, John Wiley & Sons, Inc., New York, 1967.

12. Thayer, W.J., "Transfer Functions for Moog Servovalves," Technical Bulletin 103, Moog Inc. Controls Division, East Aurora, New York, 1965.

13. Klein, C.A., and Briggs, R.L., "Use of Active Compliance in the Control of Legged Vehicles," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-10, No. 7, July 1980.

14. Speckhart, F.H., and Green, W.L., A Guide to Using CSMP, Prentice Hall, Englewood Cliffs, New Jersey, 1976.

15. Allen, R.R., and Rozelle, D.M., "A Describing Function for Dynamic Forces in Single Degree-of-Freedom Mechanisms," Transactions of ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 102, Dec. 1980.

16. Dubowsky, S., and DesForges, D.T., "The Application of Model-Referenced Adaptive Control to Robotic Manipulators," Transactions of ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 101, Sept. 1979.

17. MacFarlane, A.G., "The Development of Frequency-Response Methods in Automatic Control," IEEE Transactions on Automatic Control, Vol. AC-24, April 1979.

APPENDIX A

We wish to derive a relationship between the excess force ΔF(t) at the drive actuator and the velocity and acceleration of the actuator piston and the vehicle center of mass.

In general, the relationship between the vehicle acceleration A and the actuator velocity $\dot{x}_p$ and acceleration $\ddot{x}_p$, for a vehicle using a linkage leg of the type shown in Figure 1, is

$$A = \frac{V_o}{v_o}\ddot{x}_{p1}(t) + \frac{\dot{x}_{p1}^2(t)}{v_o^2} A_o - \frac{V_o}{v_o} a_o \qquad \text{(A-1)}$$

V_o and A_o are computed nominal vehicle velocity and acceleration and v_o and a_o are the corresponding values of the actuator velocity and acceleration, for a specified actuator position.

Assuming conservation of mechanical energy and neglecting changes in kinetic and potential energy due to changes in the linkage geometry and in the elevation of the vehicle center of mass, the force at the actuator for a vehicle moving on an incline, is

$$F(t) = F_{L1}(t) + \Delta F\ (t) = \frac{MV_o}{v_o} (A + g \sin \alpha) \tag{A-2}$$

α is the angle of the incline and M is the share of vehicle mass seen by the actuator. M depends therefore on the positions of all the legs in contact with the ground.

Combining equations (A-1) and (A-2) and noting that

$$F_{L1}(t) = \frac{MV_o}{v_o} (A_o + g \sin \alpha) \tag{A-3}$$

we get

$$\Delta F(t) = \frac{MV_o}{v_o} \left(\frac{V_o}{v_o} \ddot{x}_{p1}(t) + \frac{A_o}{v_o^2} \dot{x}_{p1}{}^2(t) - \frac{V_o a_o}{v_o^3} \dot{x}_{p1}{}^2(t) - A_o\right) \tag{A-4}$$

In general, the determination of M requires that the equations of motion of the entire vehicle be solved simultaneously to determine the leg positions and then the leg loads. For the present paper, the leg load distribution corresponding to tripod gait is used to determine M. For the center legs, M is one-half the vehicle mass.

The simplest linearized form of equation (A-4) is obtained by assuming that the vehicle moves in tripod gait at constant speed V_o and that the corresponding actuator acceleration a_o is negligible. Figure 3 indicates, however, that this is not really the case.

$$\Delta F(t) = \frac{MV_o^2}{(v_o)_m^2} \ddot{x}_{p1}(t) \tag{A-5}$$

Appendix 3

```
(*$A-*)
          /****** FILE: GBLF34.PAS ******/

/*****************************************************************/
/*                                                               */
/* FUNCTION:    THIS FILE CONTAINS GLOBAL DECLARATIONS FOR       */
/*              ROBOT 3.4. THE EXECUTABLE FILES WHICH SHARE      */
/*              THESE GLOBALS ARE:                               */
/*                                                               */
/*              ROBT34.PAS      PLAN34.PAS      FOOT34.PAS       */
/*              INIT34.PAS      LINE34.PAS      SERV34.PAS       */
/*              DLNK34.PAS      LIBR34.PAS                       */
/*                                                               */
/*****************************************************************/

CONST
        PI        = 3.14159;
        FSCALE    = -200.0;        /* FORCE SCALE FACTOR           */
        MAXSTROKE = 12.0;          /* MAXIMUM FOOT STROKE          */
        MOVE      = TRUE;          /* SWITCH TO ENABLE SERVOING    */
        NOMOVE    = FALSE;         /* SWITCH TO DISABLE SERVOING   */


TYPE
        ARRAY6   = ARRAY[1..6] OF REAL;
        ARRAY18  = ARRAY[0..17] OF REAL;
        MODETYPE = ( RANDOM, NEUTRAL, PREWALK,
                     CRUISE, SIDESTEP, TURN   );


 VAR
        MIDSTX, MIDSTY,            /* MIDSTANCE COORDINATES        */
        RPHASE,                    /* RELATIVE LEG PHASES          */
        XFA,  YFA,  ZFA,           /* ACTUAL FOOT FORCES           */
        XFD,  YFD,  ZFD,           /* DESIRED FOOT FORCES          */
        XFF,  YFF,  ZFF,           /* FILTERED FOOT FORCES         */
        XPA,  YPA,  ZPA,           /* ACTUAL FOOT POSITIONS        */
        XPD,  YPD,  ZPD,           /* DESIRED FOOT POSITIONS       */
        XRD,  YRD,  ZRD,           /* DESIRED FOOT RATES           */
        ZPTERM                     /* Z POS. TERM FOR ATT. CONTROL */

                : ARRAY6;

        ZEROFORCE                  /* TRUE FORCE OFFSETS           */

                : ARRAY18;

        FZERO                      /* FLAGS FOR FORCE ZEROING      */

                : ARRAY[1..6] OF BOOLEAN;
```

```
ATTITUDE,                   /* SWITCH FOR ATTITUDE CONTROL  */
COMPLIANCE,                 /* INDIRECT COMPLIANCE SWITCH   */
OPTIMIZATION,               /* SWITCH FOR OPTIMAL FORCE     */
PASS1,                      /* SWITCH TO INITIALIZE FILTER  */
SAVE,                       /* SWITCH FOR DATA AQUISITION   */
SPRING                      /* SWITCH FOR ACTIVE COMPLIANCE */

        : BOOLEAN;

COMMAND                     /* OPERATOR INPUT COMMAND       */

        : CHAR;

LASTCLOCK,                  /* STORAGE FOR CLOCK BUFFER     */
TOTALCLOCK                  /* CLOCK TICK ACCUMULATOR       */

        : INTEGER;

MODE                        /* HEXAPOD OPERATING MODE       */

        : MODETYPE;

BETA,                       /* LEG DUTY FACTOR              */
DPSI,                       /* FILTERED TURN RATE COMMAND   */
DT,                         /* DELTA TIME (SEC)             */
FOOTLIFT,                   /* FOOT LIFTING HEIGHT          */
MIDSTZ,                     /* MIDSTANCE Z COORDINATE       */
NVELX, NVELY,               /* OPERATOR VELOCITY COMMANDS   */
NDPSI,                      /* OPERATOR TURN RATE COMMAND   */
PERIOD,                     /* PERIOD OF KINEMATIC CYCLE    */
PHASE,                      /* KINEMATIC CYCLE PHASE        */
RADIUS,                     /* RADIUS FROM CG. TO MIDSTANCE */
SPERIOD-                    /* SUPPORT PERIOD (SEC)         */
VELX, VELY,                 /* FILTERED VELOCITY COMMANDS   */
VELMAX                      /* MAX FOOT VELOCITY COMPONENT  */

        : REAL;

SUPPORT                     /* SET OF LEGS IN SUPPORT PHASE */

        : SET OF 1..6;
```

```
                    /******  FILE: ROBT34.PAS  ******/

                     /******************************/
                     /*     PROGRAM: ROBOT 3.4     */
                     /******************************/

/*****************************************************************/
/* PROGRAMMERS: TED CHANG, DENNIS PUGH                          */
/* DATE:        30-MAR-82                                       */
/*                                                              */
/* PROGRAMMER                                                   */
/* GUIDE:       >PAS ROBT34=GBLF34,ROBT34                       */
/*              >MAC ROBT34=ROBT34                              */
/*              >TKB @ROBT34                                    */
/*                                                              */
/* LINK FILES:  LIBR34                                          */
/*              PLAN34                                          */
/*              FOOT34                                          */
/*              SERV34                                          */
/*              DLNK34                                          */
/*              INIT34                                          */
/*              LINE34                                          */
/*              KYCK34                                          */
/*              POWR34                                          */
/*                                                              */
/* USER GUIDE:  >RUN ROBT34                                     */
/*              THE PROGRAM PROMPTS FOR INSTRUCTIONS            */
/*                                                              */
/*              (THIS PROGRAM IS TO RUN ON THE PDP 11/70)       */
/*                                                              */
/*****************************************************************/

CONST   ZRMAX    = 6.0;         /* MAXIMUM  FOOT Z - VELOCITY   */

TYPE    CAPITALS = 'A'..'Z';
        CHARSET  = SET OF CAPITALS;
        SETARRAY = ARRAY[ MODETYPE ] OF CHARSET;

VAR     DPSIMAX   :REAL;        /* MAXIMUM  TURN RATE           */
        VTMAX     :REAL;        /* MAX. FOOT VELOCITY COMPONENT */

        BETAMODE  :INTEGER;     /* LEG DUTY FACTOR INDEX        */

        LETTER    :CHAR;        /* LOOP INDEX                   */
        HALTSET   :CHARSET;     /* SET OF COMMANDS FOR WHICH    */
                                /* HEXAPOD HALTS AFTERWARD      */
        NEXTCOM   :SETARRAY;    /* SET OF VALID COMMANDS FOR    */
                                /* FOR EACH MODE                */
```

```
PROCEDURE  NORMALIZE;  EXTERNAL;

PROCEDURE  INITIALIZE;  EXTERNAL;

PROCEDURE  HALT;  EXTERNAL;

PROCEDURE  UPDOWN( HEIGHT: REAL );  EXTERNAL;

PROCEDURE  PLANMOTION;  EXTERNAL;

PROCEDURE  CLOCKINIT;  EXTERNAL;

PROCEDURE  KEYINT;  FORTRAN;

PROCEDURE  KEYCK(VAR COMMAND: CHAR);  FORTRAN;

PROCEDURE  KYWAIT;  FORTRAN;

PROCEDURE  IOKILL;  FORTRAN;

PROCEDURE  INQUE;  FORTRAN;

PROCEDURE  TESTON;  EXTERNAL;

PROCEDURE  TURNON;  EXTERNAL;

PROCEDURE  TURNOF;  EXTERNAL;

FUNCTION  DELTATIME: REAL;  EXTERNAL;


BEGIN   /* START MAIN PROGRAM */

            /*****************************/
            /*  PARAMETER INITIALIZATION */
            /*****************************/

FOOTLIFT := 5.0;
BETAMODE := 4;

/***  ASSIGN MIDSTANCE POSITION ***/

MIDSTX[1] :=  22.75 + 1.436;    MIDSTX[2] :=  22.75 + 1.436;
MIDSTX[3] :=          1.436;    MIDSTX[4] :=          1.436;
MIDSTX[5] := -22.75 + 1.436;    MIDSTX[6] := -22.75 + 1.436;

MIDSTY[1] := -24.0;             MIDSTY[2] := 24.0;
MIDSTY[3] := -24.0;             MIDSTY[4] := 24.0;
MIDSTY[5] := -24.0;             MIDSTY[6] := 24.0;

MIDSTZ := 17.0;

RADIUS := SQRT( MIDSTX[1] * MIDSTX[1] + MIDSTY[1] * MIDSTY[1] );
```

```
/***  NEXTCOMMAND DECLARATION  ***/

NEXTCOM[ RANDOM ]   := ['I','M','N','X'];
NEXTCOM[ NEUTRAL ]  := ['D','I','M','N','U','X'];
NEXTCOM[ PREWALK ]  := ['C','H','I','M','N','T','X','Z'];
NEXTCOM[ CRUISE ]   := ['B','F','H','L','P','R','S','X'];
NEXTCOM[ TURN ]     := ['H','P','S','X'];
NEXTCOM[ SIDESTEP ] := ['H','L','R','X'];

HALTSET      := ['M','N','I','H','U','D','C','T','Z','W'];


WRITE( CHR(27), '[2J');          /* ERASE SCREEN */
WRITE( CHR(27), '[1;1H');        /* CURSOR TO HOME */

TESTON;    /* POWER UP THE HEXAPOD */
TURNOF;    /* SWITCH OFF MOTOR POWER UNTIL NEEDED */


           /**********************/
           /*   DISPLAY  MENU    */
           /**********************/

WRITE( CHR(27), '[2J');          /* ERASE SCREEN */
WRITE( CHR(27), '[1;1H');        /* CURSOR TO HOME */

WRITELN('ENTER N  TO NORMALIZE THE LEG POSITIONS          |   BETAMODE =   ');
WRITELN('      U  TO MOVE THE BODY UP                     |                ');
WRITELN('      D  TO MOVE THE BODY DOWN                   |   FOOTLIFT =   ');
WRITELN('      I  TO INITIALIZE THE LEG POSITIONS         |________________');
WRITELN('      M  TO MODIFY THE PARAMETERS                |                ');
WRITELN('      X  TO EXIT THE PROGRAM EXECUTION           |   COMPLIANCE   ');
WRITELN('                                                 |                ');
WRITELN('      C  TO ENTER CRUISE MODE                    |   OPTIMIZATION');
WRITELN('      T  TO ENTER TURN-IN-PLACE MODE             |                ');
WRITELN('      Z  TO ENTER SIDESTEP MODE                  |   ATTITUDE     ');
WRITELN('                                                 |________________');
WRITELN('      F  TO INCREASE FORWARD VELOCITY COMPONENT  |                ');
WRITELN('      B  TO INCREASE REARWARD VELOCITY COMPONENT |   AQUISITION   ');
WRITELN('      S  TO INCREASE CLOCKWISE TURN RATE         |      TIME      ');
WRITELN('      P  TO INCREASE COUNTERCLOCKWISE TURN RATE  |                ');
WRITELN('      R  TO INCREASE RIGHTWARD VELOCITY COMPONENT |               ');
WRITELN('      L  TO INCREASE LEFTWARD VELOCITY COMPONENT |                ');
WRITELN('      H  TO HALT MOTION                          |                ');
WRITELN;
```

```
/***  SET TERMINAL CHARACTERISTICS  ***/

WRITE( CHR(27), '[?41' );        /* SET TO NOSCROLL */
WRITE( CHR(27), '[20;22r' );     /* SET SCROLLING REGION */
WRITE( CHR(27), '[?61' );        /* SET ABSOLUTE ORIGIN MODE */
WRITE( CHR(27), '[20;1f' );      /* CURSOR TO SCROLLING REGION */

KEYINT;  { PLACE KEYBOARD INPUT REQUEST ON QUEUE }
CLOCKINIT;  { START PROGRAMMABLE CLOCK }


            /********************************/
            /*   REAL - TIME  OPERATION    */
            /********************************/

COMMAND := 'M';  { ENTER MODIFY SEQUENCE INITIALLY }
MODE := RANDOM;

REPEAT { UNTIL COMMAND = 'X' }
    WRITE( CHR(27), '[22;1H' );  { CURSOR TO SCREEN BOTTOM }
    WRITE( CHR(27), '[0m' );     { NORMAL CHARACTER ATTRIBUTES }
    WRITELN;
    WRITELN( COMMAND );

    IF NOT ( COMMAND IN NEXTCOM[ MODE ] )
        THEN { COMMAND IS INVALID }
            BEGIN
            WRITELN( CHR(13), CHR(27),'#6*** INVALID COMMAND ***');
            WRITE( 'VALID COMMANDS ARE: ' );
            FOR LETTER := 'A' TO 'Z' DO
                IF LETTER IN NEXTCOM[ MODE ] THEN WRITE(' ',LETTER);
            WRITELN;

            IF MODE IN [ RANDOM, NEUTRAL, PREWALK ]
                THEN COMMAND := 'W'   { WAIT }
                ELSE COMMAND := 'G';  { GO }
            END;

    { END IF }
```

```
IF COMMAND IN HALTSET
    THEN
        BEGIN
        CASE COMMAND OF

            'N':    BEGIN
                    NORMALIZE;
                    MODE := NEUTRAL;
                    END;

            'I':    BEGIN
                    NORMALIZE;
                    INITIALIZE;
                    MODE := PREWALK;
                    END;

            'U':    UPDOWN( 24.5 );

            'D':    UPDOWN( MIDSTZ );

            'H':    BEGIN
                    HALT;
                    MODE := PREWALK;
                    SAVE := FALSE;    { SWITCH OFF DATA AQUISITION }
                    END;

            'C':    BEGIN
                    TURNON;
                    WRITELN;
                    WRITELN( CHR(27), '$6*** CRUISE MODE ***');
                    MODE := CRUISE;
                    SPRING := COMPLIANCE;  { ENABLE ACTIVE COMPLIANCE }
                    PASS1 := TRUE;  { FLAG FOR FILTER INITIALIZATION }
                    END;

            'T':    BEGIN
                    TURNON;
                    WRITELN;
                    WRITELN( CHR(27), '$6*** TURN-IN-PLACE MODE ***');
                    MODE := TURN;
                    SPRING := COMPLIANCE;  { ENABLE ACTIVE COMPLIANCE }
                    PASS1 := TRUE;  { FLAG FOR FILTER INITIALIZATION }
                    END;

            'Z':    BEGIN
                    TURNON;
                    WRITELN;
                    WRITELN( CHR(27), '$6*** SIDESTEP MODE ***');
                    MODE := SIDESTEP;
                    SPRING := COMPLIANCE;  { ENABLE ACTIVE COMPLIANCE }
                    PASS1 := TRUE;  { FLAG FOR FILTER INITIALIZATION }
                    END;
```

```
'M':    BEGIN
        IOKILL; { CANCEL INPUT REQUEST }
        WRITELN;
        WRITELN( CHR(27), '#6PARAMETER MODIFICATION');
        WRITELN;

{***}   WRITELN( 'CHANGE BETA?');
        READLN( LETTER );
        IF LETTER = 'Y' THEN
            REPEAT
            WRITELN('PLEASE ENTER DESIRED BETAMODE:');
            WRITELN(' 1:  BETA = 5/6       2:  BETA = 3/4');
            WRITELN(' 3:  BETA = 2/3       4:  BETA = 1/2');
            READLN(BETAMODE);
            UNTIL BETAMODE IN [1,2,3,4];
        WRITE( CHR(27),'7', CHR(27),'[1;67H', CHR(27),'[1m' );
        WRITE( BETAMODE:1, CHR(27), '8' );
        WRITELN;
        WRITELN;

        CASE BETAMODE OF
            1:   BETA := 0.8333;  { 5/6 }
            2:   BETA := 0.75;    { 3/4 }
            3:   BETA := 0.6667;  { 2/3 }
            4:   BETA := 0.5      { 1/2 }
        END;/* CASE */

        /***  ASSIGN RELATIVE LEG PHASES  ***/
        RPHASE[1] := 0.0;
        RPHASE[2] := 0.5;
        RPHASE[3] := BETA;
        RPHASE[4] := BETA - 0.5;
        RPHASE[5] := 2.0 * BETA - 1.0;
        RPHASE[6] := RPHASE[5] + 0.5;
        RPHASE[6]`:= RPHASE[6] - TRUNC(RPHASE[6]);

{***}   WRITELN( 'CHANGE FOOTLIFT?' );
        READLN( LETTER );
        IF LETTER = 'Y' THEN
            REPEAT
            WRITELN( 'ENTER NEW VALUE OF FOOTLIFT' );
            READLN( FOOTLIFT );
            UNTIL ( FOOTLIFT >= 1.0 ) AND ( FOOTLIFT <= 12.0 );
        WRITE( CHR(27),'7', CHR(27),'[3;67H', CHR(27),'[1m' );
        WRITE( FOOTLIFT:4:1, CHR(27), '8' );
        WRITELN;
        WRITELN;

        VTMAX := ZRMAX * MAXSTROKE / ( FOOTLIFT * PI );
        IF VTMAX > 4.0 THEN  VTMAX := 4.0; { IN. PER SEC }
        VELMAX := VTMAX * ( 1.0 - BETA ) / BETA;
        DPSIMAX := VELMAX / RADIUS;
```

```
{***}   WRITELN( 'TURN ACTIVE COMPLIANCE ON?');
        READLN( LETTER );
        WRITE( CHR(27), '7', CHR(27), '[6;56H' );
        IF LETTER = 'Y'
            THEN
                BEGIN
                COMPLIANCE := TRUE;
                WRITE( CHR(27), '[7;5;1m' );
                END
            ELSE
                BEGIN
                COMPLIANCE := FALSE;
                WRITE( CHR(27), '[0m' );
                END;
        WRITE( 'COMPLIANCE', CHR(27), '8' );
        WRITELN;
        WRITELN;

{***}   WRITELN( 'TURN FORCE OPTIMIZATION ON?');
        READLN( LETTER );
        WRITE( CHR(27), '7', CHR(27), '[8;56H' );
        IF LETTER = 'Y'
            THEN
                BEGIN
                OPTIMIZATION := TRUE;
                WRITE( CHR(27), '[7;5;1m' );
                END
            ELSE
                BEGIN
                OPTIMIZATION := FALSE;
                WRITE( CHR(27), '[0m' );
                END;
        WRITE( 'OPTIMIZATION', CHR(27), '8' );
        WRITELN;
        WRITELN;

{***}   WRITELN( 'TURN ATTITUDE CONTROL ON?');
        READLN( LETTER );
        WRITE( CHR(27), '7', CHR(27), '[10;56H' );
        IF LETTER = 'Y'
            THEN
                BEGIN
                ATTITUDE := TRUE;
                WRITE( CHR(27), '[7;5;1m' );
                END
            ELSE
                BEGIN
                ATTITUDE := FALSE;
                WRITE( CHR(27), '[0m' );
                END;
        WRITE( 'ATTITUDE', CHR(27), '8' );
```

```
                        INQUE; { REASSERT INPUT REQUEST }
                        MODE := RANDOM;
                        WRITELN;
                        WRITELN;
                        WRITELN( CHR(27), '$6INITIALIZE HEXAPOD ');
                        END;

                'W':

                END; { CASE COMMAND }
            KYWAIT;
            KEYCK(COMMAND);
            DT := DELTATIME;  { UPDATE LASTCLOCK AFTER WAIT }
            END { THEN }

        ELSE
            BEGIN
            CASE COMMAND OF

                'F':    IF NVELX < ( 0.9 * VELMAX )
                            THEN NVELX := NVELX + ( 0.1 * VELMAX )
                            ELSE NVELX := VELMAX;

                'B':    IF NVELX > -( 0.9 * VELMAX )
                            THEN NVELX := NVELX - ( 0.1 * VELMAX )
                            ELSE NVELX := -VELMAX;

                'S':    IF NDPSI < ( 0.9 * DPSIMAX )
                            THEN NDPSI := NDPSI + ( 0.1 * DPSIMAX )
                            ELSE NDPSI := DPSIMAX;

                'P':    IF NDPSI > -( 0.9 * DPSIMAX )
                            THEN NDPSI := NDPSI - ( 0.1 * DPSIMAX )
                            ELSE NDPSI := -DPSIMAX;

                'R':    IF NVELY < ( 0.9 * VELMAX )
                            THEN NVELY := NVELY + ( 0.1 * VELMAX )
                            ELSE NVELY := VELMAX;

                'L':    IF NVELY > -( 0.9 * VELMAX )
                            THEN NVELY := NVELY - ( 0.1 * VELMAX )
                            ELSE NVELY := -VELMAX;

                'G':

                END; { CASE COMMAND }

            PLANMOTION;  { MOTION PLANNING & EXECUTION }
                         { UNTIL NEXT COMMAND INPUT    }

            END; { ELSE }
        { END IF COMMAND IN HALTSET }
    UNTIL COMMAND = 'X';
```

```
HALT;

IOKILL; { CANCEL KEYBOARD INPUT REQUEST }

WRITE( CHR(27), '[?4h');        /* SET TO SMOOTH SCROLL */
WRITE( CHR(27), '[1;22r');      /* RESTORE SCROLLING REGION */
WRITE( CHR(27), '[2J');         /* ERASE SCREEN */
WRITE( CHR(27), '[1;1H');       /* CURSOR TO HOME */
END.
```

```
(*$E+*)

          /*******    FILE:   PLAN35.PAS    *******/

                    /******************************/
                    /*   PROCEDURE:  PLANMOTION  */
                    /******************************/

/*****************************************************************/
/*                                                               */
/* PROGRAMMERS: TED CHANG, DENNIS PUGH                           */
/* DATE:        30-MAR-82                                        */
/* FUNCTION:    PLAN BODY MOTION TO IMPLEMENT OPERATOR COMMANDS */
/* USER GUIDE:  THE CALLING FORMAT IS:                           */
/*              PLANMOTION;                                      */
/*                                                               */
/* PROCEDURES CALLED:   FOOTLINE, KEYCK                          */
/*                                                               */
/* GLOBAL VARIABLES:                                             */
/*      REFERENCED:      NVELX,  NVELY,  NDPSI                   */
/*                       BETA,   VELMAX                          */
/*                       MODE                                    */
/*                                                               */
/*      MODIFIED:        VELX,   VELY,   DPSI                    */
/*                       PERIOD, SPERIOD, PHASE                  */
/*                       DT                                      */
/*                                                               */
/*****************************************************************/

PROCEDURE PLANMOTION;

CONST   TIMECONST = 0.5;        /* TIME CONST. FOR INPUT FILTER */
        MINSTROKE = 5.0;        /* MINIMUM FOOT STROKE          */
        RMIN = 60.0;            /* MIN. TURN RADIUS ( CRUISE )  */

VAR DVELX, DVELY, DDPSI,        /* VELOCITY DERIVATIVE TERMS    */
    NVEL,                       /* VELOCITY COMMAND MAGNITUDE   */
    LTIME,                      /* TEMPORARY STORAGE FOR TIME   */
    FOOTRATE,                   /* APPROX. FOOT RATE WRT BODY   */
    STROKE                      /* FOOT TRAVEL IN SUPPORT PHASE */
                    :REAL;

PROCEDURE FOOTPATH;  EXTERNAL;

PROCEDURE KEYCK( VAR COMMAND: CHAR );  FORTRAN;

FUNCTION SIGN( X: REAL ): REAL;  EXTERNAL;

FUNCTION DELTATIME: REAL;  EXTERNAL;
```

```
BEGIN  { PLANMOTION }

IF MODE = CRUISE  { SET LIMITS ON VELOCITY COMPONENTS }
    THEN
        BEGIN
        WRITELN( CHR(27), '[0m' ); { NORMAL CHARACTER ATTRIBUTES }

        /* SET LIMIT FOR CRAB ANGLE IN CRUISE MODE */
        IF ABS( NVELY ) > ABS( NVELX )
            THEN NVELY := SIGN( NVELY ) * ABS( NVELX );

        /* DETERMINE MAGNITUDE OF COMMANDED VELOCITY */
        NVEL := NVELX*NVELX + NVELY*NVELY;
        IF NVEL <> 0.0 THEN NVEL := SQRT( NVEL );

        /* LIMIT VELOCITY MAGNITUDE TO VELMAX */
        IF NVEL > VELMAX THEN
            BEGIN
            NVELX := NVELX / NVEL * VELMAX;
            NVELY := NVELY / NVEL * VELMAX;
            END;

        /* SET LIMIT FOR DPSI IN CRUISE MODE */
        IF ABS( NDPSI ) > ABS( NVEL / RMIN )
            THEN NDPSI := SIGN( NDPSI ) * ABS( NVEL / RMIN );

        WRITE( NVELX:8:3, NVELY:8:3, NDPSI:8:3 );
        IF ABS(NDPSI) > 0.00001
            THEN WRITE( NVEL / ABS( NDPSI ):8:2);

        END { THEN }

    ELSE
        WRITE( NVELX:8:3, NVELY:8:3, NDPSI:8:3 );

{ END IF MODE }

WRITE( CHR(27), '[1A' );  { ENABLE BOLD CHARACTERS }

REPEAT  { UNTIL A COMMAND IS INPUT }

    /*** GENERATE DT ***/
    DT := DELTATIME;

    /*** FILTER THE RATE COMMAND INPUTS ***/

    DVELX := (NVELX - VELX) / TIMECONST;  /* LONGITUDINAL ACCEL.  */
    DVELY := (NVELY - VELY) / TIMECONST;  /* LATERAL ACCELERATION */
    DDPSI := (NDPSI - DPSI) / TIMECONST;  /* TURNING ACCELATION   */

    VELX := DVELX*DT + VELX;              /* LONGITUDINAL VEL.    */
    VELY := DVELY*DT + VELY;              /* LATERAL VELOCITY     */
    DPSI := DDPSI*DT + DPSI;              /* TURNING RATE         */
    /*** CALCULATE STROKE & SUPPORT PERIOD ***/

    CASE MODE OF
        CRUISE:     FOOTRATE := VELX;

        SIDESTEP:   FOOTRATE := VELY;

        TURN:       FOOTRATE := RADIUS * DPSI;

        END; { CASE }

    IF ABS( FOOTRATE ) > 0.00001
        THEN
            BEGIN
            STROKE := MINSTROKE + ( MAXSTROKE - MINSTROKE )
                        * ABS( FOOTRATE ) / VELMAX;

            SPERIOD := STROKE / FOOTRATE;        { SUPPORT PERIOD }
            END
        ELSE
            SPERIOD := 10000.0;

    PERIOD := SPERIOD / BETA;           { TOTAL CYCLE PERIOD }


    /*** UPDATE PHASE VARIABLE ***/
    PHASE := PHASE + DT / PERIOD + 1;
    PHASE := PHASE - TRUNC(PHASE);

    /*** CALL FOOT TRAJECTORY GENERATION ROUTINE ***/
    FOOTPATH;

    /*** CHECK FOR OPERATOR INPUT ***/
    KEYCK(COMMAND);

UNTIL ORD( COMMAND ) <> 0;  { UNTIL A COMMAND IS INPUT }

END;
```

```
(*$E+*)

          /*******      FILE:  FOOT34.PAS     *******/

                 /******************************/
                 /*   PROCEDURE:   FOOTPATH   */
                 /******************************/

/*******************************************************************/
/* PROGRAMMERS: DENNIS PUGH, TED CHANG                             */
/* DATE:        30-MAR-82                                          */
/*                                                                 */
/* FUNCTION:    CALCULATES FOOT TRAJECTORIES TO IMPLEMENT          */
/*              THE BODY RATES COMMANDED BY BODY MOTION            */
/*              PLANNING.                                          */
/*                                                                 */
/* USER GUIDE:  THE CALLING FORMAT IS:      FOOTPATH;              */
/*                                                                 */
/*                                                                 */
/* PROCEDURES CALLED:   JSERVO, RBADC                              */
/*                                                                 */
/* GLOBAL VARIABLES                                                */
/*      REFERENCED:     VELX,    VELY,    DPSI                     */
/*                      MIDSTX, MIDSTY, MIDSTZ                     */
/*                      DT,      PERIOD, SPERIOD                   */
/*                      PHASE,  RPHASE, BETA                       */
/*                      FOOTLIFT                                   */
/*                                                                 */
/*      MODIFIED:       XPD,     YPD,     ZPD                      */
/*                      XRD,     YRD,     ZRD                      */
/*                      XFD,     YFD,     ZFD                      */
/*                      SUPPORT,   ZPTERM                          */
/*                                                                 */
/*******************************************************************/

PROCEDURE FOOTPATH;

CONST   ATTSCALE = -0.7854;      /* ATTITUDE SCALE FACTOR          */
        ATTPOLE = 8.0;           /* POLE FOR ATTITUDE CONTROL      */
        FTOTAL = 285.0;          /* TOTAL VEHICLE WEIGHT ( LBS.)  */
        PWRSCALE = -33333.3;     /* POWER SCALE FACTOR             */
```

```
VAR
        ALPHA,           /* CRAB ANGLE WRT LONGITUDINAL AXIS     */
        CGX, CGY,        /* COORDINATES OF CG. PROJECTION        */
        DT1,             /* BACKED UP TIME FROM MIDSTANCE        */
        DXB, DYB,        /* BODY DISPLACEMENTS IN BODY COORD.    */
        DXB1, DYB1,      /* TOUCHDOWN DISPL. FROM MIDSTANCE      */
        DXE, DYE,        /* BODY DISPLACEMENT IN EARTH COORD.    */
        LPHASE,          /* LEG PHASE VARIABLE                   */
        PITCH, ROLL,     /* ACTUAL ANGLES FROM GYRO              */
        PPITCH, PROLL,   /* ACTUAL ANGLES FROM PENDULUM          */
        POWER,           /* HEXAPOD INSTANTANEOUS POWER CONSUMP. */
        PSIC, PSIC1,     /* BODY ANGULAR DISPLACEMENT OVER DT    */
        Q, R, S,         /* INTERMED. FORCE OPTIMIZATION TERMS   */
        SUMX,   SUMY,    /* SUM OF FOOT DISPLACEMENTS            */
        SUMX2,  SUMY2,   /* SUM OF SQUARES OF FOOT DISPLACEMENTS */
        SUMXY,           /* SUM OF FOOT DISPL. CROSS-PRODUCTS    */
        TIMEFP,          /* REMAINING TIME IN TRANSFER PHASE     */
        TPHASE,          /* TRANSFER PHASE VARIABLE              */
        TRTIME,          /* TOTAL TIME IN TRANSFER PHASE         */
        VEL,             /* MAGNITUDE OF VELOCITY VECTOR         */
        ZOFF,            /* Z POSITION ERROR FOR ATT. CONTROL    */
        ZRTERM           /* Z RATE OFFSET FOR ATTITUDE CONTROL   */

                : REAL;

        FRSTCH,          /* FIRST A/D CHANNEL FOR DATA FETCH     */
        I,               /* LOOP COUNTER                         */
        N                /* SUPPORT PHASE LEG SET COUNTER        */

                : INTEGER;

        XFTHLD, YFTHLD  /* DESIRED FOOT TOUCHDOWN POINT          */

                : ARRAY6;


        TOPBLOCK         /* TOP A/D CHANNELS                     */

                : ARRAY18;

PROCEDURE JSERVO( MOVE: BOOLEAN );  EXTERNAL;

PROCEDURE RBADC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18 );
        EXTERNAL;

FUNCTION  ATAN2(Y, X : REAL): REAL;  EXTERNAL;
```

```
BEGIN   /* FOOT34 */

/*********************************************************************/
/*  EXPRESS INCREMENTAL BODY DISPLACEMENT IN BODY COORDINATES  */
/*********************************************************************/

/*** CALCULATE INCREMENTAL BODY DISPLACEMENT WRT GROUND ***/
PSIC    := DPSI * DT;

/*** CALCULATE MAGNITUDE OF VELOCITY WRT GROUND ***/
VEL := VELX * VELX + VELY * VELY;
IF VEL <> 0.0 THEN VEL := SQRT(VEL);

IF ABS(DPSI) > 0.00001
    THEN
        BEGIN
        DXE      := VEL/DPSI * SIN(PSIC);
        DYE      := VEL/DPSI * (1.0 - COS(PSIC));
        END
    ELSE
        BEGIN
        DXE      := VEL * DT;
        DYE      := 0.0;
        END;

/*** ROTATE DISPLACEMENT VECTORS TO BODY COORDINATES ***/
ALPHA := ATAN2( VELY, VELX );

DXB :=    DXE * COS(ALPHA)
        - DYE * SIN(ALPHA);

DYB :=    DXE * SIN(ALPHA)
        + DYE * COS(ALPHA);


/*************************************/
/*  COMPUTE FOOT TOUCHDOWN OFFSETS  */
/*************************************/

/*** CALCULATE BODY DISPLACEMENT FROM MIDSTANCE TO TOUCHDOWN ***/
DT1      := -0.5 * ABS(SPERIOD);  { BACKED UP TIME FROM MIDSTANCE }
PSIC1    := DPSI * DT1;            { BACKED UP ANGLE }

IF ABS(DPSI) > 0.00001
    THEN
        BEGIN
        DXE      := VEL / DPSI * SIN(PSIC1);
        DYE      := VEL / DPSI * ( 1.0 - COS(PSIC1) );
        END
    ELSE
        BEGIN
        DXE      := VEL * DT1;
        DYE      := 0.0;
        END;
```

```
/*** ROTATE TOUCHDOWN OFFSET TO BODY COORDINATES ***/
DXB1 :=   DXE * COS(ALPHA)
        - DYE * SIN(ALPHA);

DYB1 :=   DXE * SIN(ALPHA)
        + DYE * COS(ALPHA);



RBADC(5,58,1,TOPBLOCK); { FETCH PITCH, ROLL, & POWER FROM VEHICLE }

PPITCH := TOPBLOCK[4] * ATTSCALE;
PROLL := TOPBLOCK[5] * ATTSCALE;
POWER := TOPBLOCK[6] * PWRSCALE;
PITCH := TOPBLOCK[7] * ATTSCALE;
ROLL := TOPBLOCK[8] * ATTSCALE;


/*********************************/
/*  GENERATE FOOT TRAJECTORIES  */
/*********************************/

FOR I := 1 TO 6 DO { GENERATE FOOT COORDINATE FOR LEG I }
    BEGIN

    /*** COMPUTE LEG PHASE VARIABLE ***/
    LPHASE := PHASE + RPHASE[I] + 1.0;
    LPHASE := LPHASE - TRUNC( LPHASE );

    IF LPHASE > BETA
        THEN { LEG IN TRANSFER PHASE }
            BEGIN

            TPHASE := (LPHASE - BETA) / (1.0 - BETA);
            SUPPORT := SUPPORT - [I];  { REMOVE LEG I FROM SUPPORT SET }

            IF { LEG AT TOP OF TRANSFER PHASE AND FORCE NOT YET ZEROED }
                (     ( (PERIOD > 0.0) AND (LPHASE > (BETA+1)/2.0) )
                  OR ( (PERIOD < 0.0) AND (LPHASE < (BETA+1)/2.0) ) )
                AND (FZERO[I] = FALSE)

                THEN  { UPDATE OFFSET FORCES FOR LEG I }
                    BEGIN
                    FRSTCH := ( I - 1 ) * 3;
                    RBADC( 3, FRSTCH, FSCALE, ZEROFORCE);
                    FZERO[I] := TRUE;  { FLAG THAT LEG I IS UPDATED }
                    ZPTERM[I] := 0.0;  { INITIALIZE ATTITUDE CORRECTION TERM }
                    END;

                { END IF }
    /*** CALCULATE TIME LEFT TILL TOUCHDOWN OF THE LEG ***/
    TRTIME := ( 1 - BETA ) * ABS(PERIOD);
    IF PERIOD > 0.0
        THEN TIMEFP := TRTIME * ( 1.0 - TPHASE)
        ELSE TIMEFP := TRTIME * TPHASE;

    /*** COMPUTE THE BEST TOUCHDOWN POINT ***/
    XFTHLD[I] :=    ( MIDSTX[I] - DXB1 ) * COS(PSIC1)
                  + ( MIDSTY[I] - DYB1 ) * SIN(PSIC1);

    YFTHLD[I] := - ( MIDSTX[I] - DXB1 ) * SIN(PSIC1)
                  + ( MIDSTY[I] - DYB1 ) * COS(PSIC1);

    /*** COMPUTE DESIRED FOOT POSITION ***/
    IF DT < TIMEFP
        THEN
            BEGIN
            XPD[I] := XPD[I] + (XFTHLD[I] - XPD[I]) * DT/TIMEFP;
            YPD[I] := YPD[I] + (YFTHLD[I] - YPD[I]) * DT/TIMEFP;
            ZPD[I] := MIDSTZ - FOOTLIFT * SIN( TPHASE * PI );
            END
        ELSE
            BEGIN
            XPD[I] := XFTHLD[I];
            YPD[I] := YFTHLD[I];
            ZPD[I] := MIDSTZ;
            END;

        { END IF DT }

    /*** COMPUTE DESIRED FOOT RATE ***/
    XRD[I] := ( XFTHLD[I] - XPD[I] ) / TIMEFP;
    YRD[I] := ( YFTHLD[I] - YPD[I] ) / TIMEFP;
    ZRD[I] := - FOOTLIFT * PI * COS( PI * TPHASE )
                / ( PERIOD * ( 1 - BETA ) );

    END  { TRANSFER PHASE }

ELSE  { FOOT IN SUPPORT PHASE }
    BEGIN

    SUPPORT := SUPPORT + [I];  { INCLUDE LEG I IN SUPPORT SET }
    FZERO[I] := FALSE;  { FLAG THAT FORCE NOT ZEROED THIS CYCLE }

    /*** COMPUTE ATTITUDE CONTROL VARIABLES ***/
    ZOFF := -PITCH  * XPA[I] + ROLL * YPA[I];
```

```
                    IF ATTITUDE = TRUE
                        THEN
                            BEGIN
                            ZRTERM := ATTPOLE * ZOFF;
                            ZPTERM[I] := ZPTERM[I] + ZRTERM * DT;
                            END
                        ELSE
                            BEGIN
                            ZPTERM[I] := 0.0;
                            ZRTERM := 0.0;
                            END;

                        { END IF ATTITUDE }

                    /*** COMPUTE DESIRED FOOT POSITION ***/
                    XPD[I] :=    ( XPD[I] - DXB ) * COS(PSIC)
                              + ( YPD[I] - DYB ) * SIN(PSIC);

                    YPD[I] := - ( XPD[I] - DXB ) * SIN(PSIC)
                              + ( YPD[I] - DYB ) * COS(PSIC);

                    ZPD[I] := MIDSTZ + ZPTERM[I];

                    /*** COMPUTE DESIRED FOOT RATE ***/
                    XRD[I] := -VELX + DPSI * YPD[I];
                    YRD[I] := -VELY - DPSI * XPD[I];
                    ZRD[I] := ZRTERM;

                    END; /* SUPPORT PHASE */

                /* END IF LPHASE */

            END;  /* FOR I */


    /***************************************/
    /*  COMPUTE OPTIMAL FORCE SETPOINTS  */
    /***************************************/

    N := 0;
    SUMX := 0.0;
    SUMY := 0.0;
    SUMX2 := 0.0;
    SUMY2 := 0.0;
    SUMXY := 0.0;
```

```
FOR I := 1 TO 6 DO
    IF I IN SUPPORT THEN
        BEGIN
        SUMX := SUMX + XPA[I];
        SUMY := SUMY + YPA[I];
        SUMX2 := SUMX2 + XPA[I] * XPA[I];
        SUMY2 := SUMY2 + YPA[I] * YPA[I];
        SUMXY := SUMXY + XPA[I] * YPA[I];
        N := N + 1;
        END;  { IF I IN SUPPORT }

    { END FOR I }

R := (SUMX * SUMY2 - SUMXY * SUMY) / (SUMX2 * SUMY2 - SUMXY * SUMXY);
Q := (SUMY - SUMXY * R) / SUMY2;
S := N - ((SUMX * SUMY2 - SUMXY * SUMY) * R + SUMY * SUMY) / SUMY2;

CGX := -SIN( PITCH ) * MIDSTZ;
CGY := SIN( ROLL ) * MIDSTZ;

/*** COMPUTE FOOT FORCE SETPOINTS ***/

FOR I := 1 TO 6 DO
    BEGIN
    XFD[I] := 0.0;
    YFD[I] := 0.0;

    IF I IN SUPPORT
        THEN
            IF OPTIMIZATION = TRUE
                THEN ZFD[I] := ( 1 - Q * ( YPA[I] - CGY )
                                   - R * ( XPA[I] - CGX )  ) * FTOTAL / S
                ELSE ZFD[I] := FTOTAL / N
        ELSE
            ZFD[I] := 0.0;

    END; { FOR I }

JSERVO( MOVE );  { CALL SERVO ROUTINE }


END;  /* FOOTPATH */
```

```
(**E+*)

            /****** FILE: SERV34.PAS ******/

                  /**************************/
                  /*    PROCEDURE JSERVO    */
                  /**************************/

/******************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                       */
/* DATE:        30-MAR-82                                         */
/* FUNCTION:    ACCOMPLISHES JACOBEAN SERVO CONTROL.              */
/*              ACTIVE COMPLIANCE IS TURNED ON OR OFF             */
/*              BY THE BOOLEAN SWITCH 'SPRING'                    */
/*                                                                */
/* USER GUIDE:  THE CALLING FORMAT IS:                            */
/*              JSERVO( MOVE );                                   */
/*                                                                */
/* PROCEDURES CALLED:   RBADC, RBDAC                              */
/*                                                                */
/* GLOBAL VARIABLES                                               */
/*      REFERENCED:     XRD, YRD, ZRD                             */
/*                      XPD, YPD, ZPD                             */
/*                      XFD, YFD, ZFD                             */
/*                      ZEROFORCE, SPRING                         */
/*                                                                */
/*      MODIFIED:       XPA, YPA, ZPA                             */
/*                      XFA, YFA, ZFA                             */
/*                      XFF, YFF, ZFF                             */
/*                      PASS1                                     */
/*                                                                */
/******************************************************************/

PROCEDURE  JSERVO( MOVE: BOOLEAN );


CONST   COMPGAIN = 57.4;        /* COMPENSATOR GAIN               */

        PGAINX = 3.8;           /* RECTILINEAR POSITION GAIN      */
        PGAINY = 3.8;
        PGAINZ = 1.32;

        FGAINX = 0.0001;        /* RECTILINEAR FORCE GAIN         */
        FGAINY = 0.0001;
        FGAINZ = 0.165;

        FILTPOLE = 2.0;         /* POLE FOR FORCE TERM FILTER     */

        PSCALE = 1.571;         /* POSITION SCALE FACTOR          */
        RSCALE = 0.61;          /* RATE SCALE FACTOR              */
        ZFSCALE = -1.25;        /* Z-AXIS FORCE SCALE CHANGE      */
        VSCALE = 10.0;          /* VOLTAGE SCALE FACTOR           */
```

```
          L1      = 12.564;         /* UPPER LIMB LENGTH            */
          L2      = 17.00;          /* LOWER LIMB LENGTH            */
          L3      = -1.436;         /* AZIMUTH JOINT OFFSET         */
          L4      = 2.5;            /* ELEVATION JOINT OFFSET       */
          L5      = 2.436;          /* KNEE JOINT OFFSET            */

          YHIP    = 8.562;          /* Y DISTANCE FROM HIP TO C.G.  */


VAR       PSI,    THETA,  THETA1, THETA2, /* JOINT ANGLES           */
          CPSI,   CTH,    CTH1,   CTH2,   /* COSINES OF ANGLES      */
          SPSI,   STH,    STH1,   STH2,   /* SINES OF ANGLES        */
          TN1,    TN2,    D1,     D2,     /* INTERMEDIATE TERMS     */

          A11,    A12,                    /* INVERSE                */
          A21,    A22,    A23,            /*      JACOBEAN          */
          A31,    A32,    A33,            /*              MATRIX    */

          XRC,    YRC,    ZRC,            /* RECT. RATE COMMANDS    */
          VOUT,                           /* MOTOR OUTPUT VOLTAGE   */
          YSIGN,                          /* LEFT SIDE CORRECTION   */
          MAXTIME,                        /* MAX DT FOR FILTER      */
          DTF,                            /* FILTER DT              */
          XFE,    YFE,    ZFE,            /* RECT. FORCE ERROR      */

          XFORCE, YFORCE, ZFORCE          /* FOOT COORD. FORCES     */
                          : REAL;


          XHIP                            /* X COORD. HIP TO C.G.   */
                          : ARRAY6;


          I,      J,                      /* LOOP INDEX             */
          CHAN0,  CHAN1,  CHAN2           /* A/D CHANNEL POINTERS   */
                          : INTEGER;

          FORCE,                          /* FORCE INPUT ARRAY      */
          POSITION,                       /* ACT. JOINT POSITIONS   */
          RATE,                           /* ACTUAL JOINT RATES     */
          RATECOM,                        /* COMMAND JOINT RATES    */
          VOLT                            /* MOTOR VOLTAGE OUTPUT   */
                          : ARRAY18;


PROCEDURE  RBADC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18 );
          EXTERNAL;

PROCEDURE  RBDAC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VOLT: ARRAY18 );
          EXTERNAL;

FUNCTION SIGN( X: REAL): REAL;  EXTERNAL;
```

```
BEGIN  /* JSERVO */

XHIP[1] := 22.75;        XHIP[2] := 22.75;        XHIP[3] := 0.0;
XHIP[4] := 0.0;          XHIP[5] := -22.75;       XHIP[6] :=-22.75;

/* CONDITION DT FOR FAST FILTER */
MAXTIME := 1.0 / ( FILTPOLE * 4.0 );
IF DT > MAXTIME  { THEN FILTER DEGENERATES }
    THEN DTF := MAXTIME
    ELSE DTF := DT;

FOR I := 1 TO 6 DO
    BEGIN
    IF I IN [1,3,5]
        THEN YSIGN := -1.0
        ELSE YSIGN := 1.0;

    CHANO := (I-1) * 3;
    CHAN1 := CHANO + 1;
    CHAN2 := CHANO + 2;

    RBADC(3, 36+CHANO, PSCALE, POSITION);  { READ JOINT POSITION }
    RBADC(3, CHANO, FSCALE, FORCE);        { READ FOOT FORCES }

    THETA2 := POSITION[CHANO];
    THETA1 := POSITION[CHAN1];
    PSI := POSITION[CHAN2];

    /* COMPUTE TRUE FORCES IN LEG COORDINATES */
    XFORCE := FORCE[CHANO] - ZEROFORCE[CHANO];
    YFORCE := -(FORCE[CHAN1] - ZEROFORCE[CHAN1]);
    ZFORCE := ZFSCALE * (FORCE[CHAN2] - ZEROFORCE[CHAN2]);

    /* COMPUTE & SAVE ALL NECESSARY TRIG. FUNCTIONS */
    CPSI := COS(PSI);
    SPSI := SIN(PSI);
    CTH1 := COS(THETA1);
    STH1 := SIN(THETA1);
    CTH2 := COS(THETA2);
    STH2 := SIN(THETA2);
    THETA := THETA1+THETA2;
    STH := SIN(THETA);
    CTH := COS(THETA);
    TN1 := L1*CTH1+L5*STH1+L2*STH;
    D1 := L4+TN1;
    D2 := L2*(L1*CTH2-L5*STH2);
    TN2 := TN1/D2;

    /* COMPUTE THE ACTUAL RECTILINEAR FOOT POSITION */
    XPA[I] :=  D1*SPSI - L3*CPSI + XHIP[I];
    YPA[I] := (D1*CPSI + L3*SPSI + YHIP) * YSIGN;
    ZPA[I] := -L1*STH1 + L2*CTH + L5*CTH1;
```

```
/* ROTATE FORCES TO BODY COORDINATES */
XFA[I] :=  SPSI*CTH          * XFORCE
         - YSIGN*CPSI        * YFORCE
         + SPSI*STH          * ZFORCE;

YFA[I] :=  YSIGN*CPSI*CTH    * XFORCE
         + STH               * YFORCE
         + YSIGN*CPSI*STH    * ZFORCE;

ZFA[I] := -STH               * XFORCE
         + CTH               * ZFORCE;


IF MOVE = TRUE THEN  { SERVO THE HEXAPOD }
    BEGIN

    IF SPRING = FALSE  { THE FORCE TERM MUST BE ZEROED }
        THEN
            BEGIN
            XFF[I] := 0.0;
            YFF[I] := 0.0;
            ZFF[I] := 0.0;
            END
        ELSE
            IF  PASS1 = TRUE   { THE FILTER MUST BE INITIALIZED }
                THEN
                    BEGIN
                    XFF[I] := 0.0;
                    YFF[I] := 0.0;
                    ZFF[I] := 0.0;
                    END
                ELSE
                    BEGIN
                    /* COMPUTE THE FORCE ERRORS */
                    XFE := XFD[I] - XFA[I];
                    YFE := YFD[I] - YFA[I];
                    ZFE := ZFD[I] - ZFA[I];

                    /* LIMIT FORCE ERRORS TO GIVE 4" MAX. DEFLECTION */
                    IF ABS(XFE) > 4.0 * PGAINX/FGAINX
                        THEN XFE := 4.0 * PGAINX/FGAINX * SIGN( XFE );

                    IF ABS(YFE) > 4.0 * PGAINY/FGAINY
                        THEN YFE := 4.0 * PGAINY/FGAINY * SIGN( YFE );

                    IF ABS(ZFE) > 4.0 * PGAINZ/FGAINZ
                        THEN ZFE := 4.0 * PGAINZ/FGAINZ * SIGN( ZFE );


                    /* LOW-PASS FILTER THE FORCE ERRORS */
                    XFF[I] := XFF[I] + FILTPOLE * (XFE - XFF[I]) * DTF;
                    YFF[I] := YFF[I] + FILTPOLE * (YFE - YFF[I]) * DTF;
                    ZFF[I] := ZFF[I] + FILTPOLE * (ZFE - ZFF[I]) * DTF;
```

```
                         END;  { ELSE }

                      { END IF PASS1 }

                   { END ELSE }

                { END IF SPRING }

            /* COMPUTE RECTILINEAR FOOT RATE COMMAND */
            XRC :=  XRD[I] + PGAINX * (XPD[I] - XPA[I]) + FGAINX * XFF[I];
            YRC := (YRD[I] + PGAINY * (YPD[I] - YPA[I]) + FGAINY * YFF[I]) * YSIGN;
            ZRC :=  ZRD[I] + PGAINZ * (ZPD[I] - ZPA[I]) + FGAINZ * ZFF[I];

            /* COMPUTE INVERSE JACOBIAN MATRIX */
            A11 := CPSI/D1;
            A12 := -SPSI/D1;
            A21 := -(L2*SPSI*STH)/D2+L2*L3*STH*CPSI/D1/D2;
            A22 := -L2*STH*CPSI/D2-L2*L3*STH*SPSI/D1/D2;
            A23 := -L2*CTH/D2;
            A31 := (SPSI-L3*CPSI/D1)*TN2;
            A32 := (CPSI+L3*SPSI/D1)*TN2;
            A33 := (-L1*STH1+L2*CTH+L5*CTH1)/D2;

            /* COMPUTE JOINT RATE COMMAND */
            RATECOM[CHAN2] := A11*XRC + A12*YRC;
            RATECOM[CHAN1] := A21*XRC + A22*YRC + A23*ZRC;
            RATECOM[CHAN0] := A31*XRC + A32*YRC + A33*ZRC;


            /*** RATE SERVO SECTION ***/

            RBADC(3,18+CHAN0,RSCALE,RATE);    { FETCH ACTUAL JOINT RATES }

            FOR J := CHAN0 TO CHAN2 DO
                BEGIN
                VOUT := COMPGAIN * (RATECOM[J] - RATE[J]);
                IF VOUT > 9.8 THEN  VOUT := 9.8
                    ELSE IF VOUT < -9.8 THEN VOUT := -9.8;
                VOLT[J] := VOUT;
                RBDAC(1,J,VSCALE,VOLT);  { OUTPUT THE VOLTAGE }

                END;  { DO }

            END;  { IF MOVE }

        END;  { DO }

        PASS1 := FALSE;  { FLAG THAT FILTERS ARE INITIALIZED }

    END;    { JSERVO }
```

```
(*$E+*)

        /*******     FILE:   INIT35.PAS    *******/

                    /*****************************/
                    /*   PROCEDURE:  NORMALIZE   */
                    /*****************************/

/********************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                         */
/* DATE:        30-MAR-82                                           */
/* FUNCTION:    MOVE HEXAPOD TO NORMALIZED POSITION                 */
/* USER GUIDE:  THE CALLING FORMAT IS:                              */
/*              NORMALIZE;                                          */
/*                                                                  */
/*                                                                  */
/* PROCEDURES CALLED:   JSERVO, HALT, FOOTLINE                      */
/*                      TURNON, RBADC                               */
/*                                                                  */
/* GLOBAL VARIABLES                                                 */
/*      REFERENCED:     XPA,    YPA,    ZPA                         */
/*                      MIDSTX, MIDSTY, MIDSTZ                      */
/*                                                                  */
/*      MODIFIED:       ZEROFORCE, FZERO                            */
/*                                                                  */
/********************************************************************/

PROCEDURE  NORMALIZE;


VAR     FOOT                        : INTEGER;  { FOOT INDEX        }
        FRSTCH                      : INTEGER;  { FIRST A/D CHANNEL }
        XCOORD, YCOORD, ZCOORD      : ARRAY6;   { FOOT COORDINATES  }

PROCEDURE JSERVO( MOVE: BOOLEAN );  EXTERNAL;

PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );
        EXTERNAL;

PROCEDURE RBADC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18 );
        EXTERNAL;

PROCEDURE HALT;  EXTERNAL;

PROCEDURE TURNON;  EXTERNAL;
```

```
BEGIN
    WRITELN;
    WRITELN( CHR(27), '#6 NORMALIZING ...');

    JSERVO( NOMOVE );  { FETCH ACTUAL FOOT POSITIONS }

    TURNON;  { TURN ON MOTOR POWER }

    FOR FOOT := 1 TO 6 DO { INITIALIZE COORDINATES }
        BEGIN
        XCOORD[FOOT] := XPA[FOOT];
        YCOORD[FOOT] := YPA[FOOT];
        ZCOORD[FOOT] := MIDSTZ;
        END;
    FOOTLINE( 3.0, XCOORD, YCOORD, ZCOORD );     { MOVE TO MIDSTANCE Z }


    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 1, 4, 5 ]
            THEN ZCOORD[FOOT] := MIDSTZ - 5.0;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 1 UP }

    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 1, 4, 5 ] THEN
            BEGIN
            XCOORD[FOOT] := MIDSTX[FOOT];
            YCOORD[FOOT] := MIDSTY[FOOT];
            END;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 1 OVER }

    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 1, 4, 5 ] THEN
            BEGIN
            FRSTCH := ( FOOT - 1 ) * 3;     { COMPUTE FIRST A/D CHANNEL }
            RBADC( 3, FRSTCH, FSCALE, ZEROFORCE ); { READ FORCE OFFSET }
            FZERO[FOOT] := TRUE;          { FLAG THAT ZEROFORCE UPDATED }
            ZCOORD[FOOT] := MIDSTZ;
            END;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 1 DOWN }

    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 2, 3, 6 ]
            THEN ZCOORD[FOOT] := MIDSTZ - 5.0;
    FOOTLINE( 5.0, XCOORD, YCOORD, ZCOORD );     { MOVE LEG SET 2 UP }

    FOOTLINE( 5.0, MIDSTX, MIDSTY, ZCOORD );     { MOVE LEG SET 2 OVER }
    FOR FOOT := 1 TO 6 DO
        IF FOOT IN [ 2, 3, 6 ] THEN
            BEGIN
            FRSTCH := ( FOOT - 1 ) * 3;     { COMPUTE FIRST A/D CHANNEL }
            RBADC( 3, FRSTCH, FSCALE, ZEROFORCE ); { READ FORCE OFFSET }
            FZERO[FOOT] := TRUE;         { FLAG THAT ZEROFORCE UPDATED }
            ZCOORD[FOOT] := MIDSTZ;
            END;
    FOOTLINE( 5.0, MIDSTX, MIDSTY, ZCOORD );     { MOVE LEG SET 2 DOWN }

    HALT;

    WRITELN;
    WRITELN( CHR(27), ' NORMALIZATION COMPLETE');
    WRITELN;

END;  /* NORMALIZE */
```

```
                         /********************************/
                         /*   PROCEDURE:  INITIALIZE    */
                         /********************************/

/****************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                     */
/* DATE:        20-APR-82                                       */
/* FUNCION:     RAISE LEGS TO WALKING POSITION                  */
/*                                                              */
/* USER GUIDE:  THE CALLING FORMAT IS:                          */
/*              INITIALIZE;                                     */
/*                                                              */
/*                                                              */
/* PROCEDURES CALLED:   FOOTLINE,  HALT,  TURNON                */
/*                                                              */
/* GLOBAL VARIABLES                                             */
/*      REFERENCED:     BETA, RPHASE                            */
/*                                                              */
/*      MODIFIED:       PHASE, SUPPORT, ZPTERM                  */
/*                                                              */
/****************************************************************/

PROCEDURE  INITIALIZE;

VAR     FOOT                        : INTEGER;
        TPHASE                      : REAL;
        ZINIT                       : ARRAY6;

PROCEDURE  HALT;  EXTERNAL;

PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );  EXTERNAL;

PROCEDURE TURNON;  EXTERNAL;


BEGIN
    WRITELN;
    WRITELN( CHR(27), '#6 INITIALIZING ...');

    SUPPORT := [ 1..6 ];   { START WITH ALL FEET IN SUPPORT PHASE }

    TURNON;  { TURN ON MOTOR POWER }
    /*** COMPUTE DESIRED INITIAL FOOT HEIGHTS ***/

    FOR FOOT := 1 TO 6 DO
        BEGIN
        ZPTERM[ FOOT ] := 0.0;  { INITIALIZE ATTITUDE CORRECTION TERM }

        IF RPHASE[ FOOT ] < BETA
            THEN
                ZINIT[ FOOT ] := MIDSTZ
            ELSE
                BEGIN
                SUPPORT := SUPPORT - [FOOT]; { REMOVE FOOT FROM SUPPORT SET }
                TPHASE := ( RPHASE[ FOOT ] - BETA ) / ( 1.0 - BETA );
                ZINIT[ FOOT ] := MIDSTZ - FOOTLIFT * SIN( TPHASE * PI );
                END;

            { END IF }

        END; { FOR FOOT }

    FOOTLINE( 5.0, MIDSTX, MIDSTY, ZINIT );    { RAISE LEGS IN TRANSFER PHASE }

    HALT;

    PHASE := 0.0;  { INITIALIZE KINEMATIC CYCLE PHASE }

    WRITELN;
    WRITELN( CHR(27), '#6 INITIALIZATION COMPLETE');
    WRITELN;

END;  /* INITIALIZE */
```

```
                    /**********************/
                    /*  PROCEDURE UPDOWN  */
                    /**********************/

/*******************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                        */
/* DATE:        20-APR-82                                          */
/* FUNCTION:    CHANGE THE BODY ELEVATION                          */
/*                                                                 */
/* USER GUIDE:  THE CALLING FORMAT IS:  UPDOWN( HEIGHT );          */
/*              WHERE 'HEIGHT' IS THE DESIRED BODY ELEVATION       */
/*                                                                 */
/*                                                                 */
/* PROCEDURES CALLED:   FOOTLINE,  HALT,  TURNON                   */
/*                                                                 */
/* GLOBAL VARIABLES                                                */
/*       REFERENCED:    MIDSTX, MIDSTY                             */
/*                                                                 */
/*       MODIFIED:  NONE                                           */
/*                                                                 */
/*******************************************************************/

PROCEDURE  UPDOWN( HEIGHT : REAL );


VAR     FOOT    : INTEGER;       /* FOOT INDEX            */
        ZCOORD  : ARRAY6;        /* FOOT Z COORDINATES    */


PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );
        EXTERNAL;

PROCEDURE  HALT;  EXTERNAL;

PROCEDURE TURNON;  EXTERNAL;

BEGIN
    TURNON;  { TURN ON MOTOR POWER }

    FOR FOOT := 1 TO 6 DO ZCOORD[FOOT] := HEIGHT;

    FOOTLINE( 3.0, MIDSTX, MIDSTY, ZCOORD );   { MOVE TO DESIRED ELEVATION }

    HALT;

END;   { UPDOWN }
```

```
(*$E+*)

        /*******     FILE:   LINE34.PAS    *******/

                  /*****************************/
                  /*   PROCEDURE:  FOOTLINE   */
                  /*****************************/

/*****************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                    */
/* DATE:        20-APR-82                                      */
/*                                                             */
/* FUNCTION:    MOVE FEET FROM PRESENT POSITIONS TO            */
/*              SPECIFIED POSITIONS IN STRAIGHT LINES          */
/*              WITH ALL MOTIONS COMPLETED SIMULTANEOUSLY      */
/*                                                             */
/* USER GUIDE:  THE CALLING FORMAT IS:                         */
/*              FOOTLINE( VELMAX, XCOORD, YCOORD, ZCOORD );    */
/*                                                             */
/* PROCEDURES CALLED:   JSERVO                                 */
/*                                                             */
/* GLOBAL VARIABLES                                            */
/*      REFERENCED:     XPA,    YPA,    ZPA                    */
/*                                                             */
/*      MODIFIED:       XPD,    YPD,    ZPD                    */
/*                      XRD,    YRD,    ZRD                    */
/*                      DT,     SPRING                         */
/*                                                             */
/*****************************************************************/

PROCEDURE FOOTLINE( VELMAX: REAL;  XCOORD, YCOORD, ZCOORD: ARRAY6 );

CONST GAIN = 2.0;               /* POSITION ERROR TO RATE GAIN  */


VAR ERRORX, ERRORY, ERRORZ      /* POSITION ERROR FROM SETPOINT */
                : ARRAY6;

    PTIME, LTIME,               /* TEMPORARY STORAGE FOR TIME   */
    MAXERR                      /* MAXIMUM OF COORDINATE ERRORS */
                : REAL;

    FOOT                        /* FOOT INDEX                   */
                : INTEGER;


PROCEDURE JSERVO( MOVE: BOOLEAN );  EXTERNAL;
```

```
BEGIN
    SPRING := FALSE;    { DISABLE ACTIVE COMPLIANCE }
    JSERVO( NOMOVE );    { FETCH ACTUAL POSITIONS }

    FOR FOOT := 1 TO 6 DO { INITIALIZE DESIRED POSITIONS }
        BEGIN
        XPD[FOOT] := XPA[FOOT];
        YPD[FOOT] := YPA[FOOT];
        ZPD[FOOT] := ZPA[FOOT];
        END;

    LTIME   := TIME;
    DT      := 0.01;

    REPEAT
        MAXERR := 0.0;

        FOR FOOT := 1 TO 6 DO
            BEGIN

            ERRORX[FOOT] := XCOORD[FOOT] - XPA[FOOT];
            ERRORY[FOOT] := YCOORD[FOOT] - YPA[FOOT];
            ERRORZ[FOOT] := ZCOORD[FOOT] - ZPA[FOOT];

            IF ABS( ERRORX[FOOT] ) > MAXERR THEN MAXERR := ABS( ERRORX[FOOT] );
            IF ABS( ERRORY[FOOT] ) > MAXERR THEN MAXERR := ABS( ERRORY[FOOT] );
            IF ABS( ERRORZ[FOOT] ) > MAXERR THEN MAXERR := ABS( ERRORZ[FOOT] );

            END;  /* FOR FOOT */

        FOR FOOT := 1 TO 6 DO
            BEGIN
            IF ( GAIN * MAXERR ) < VELMAX { INCHES PER SECOND }
                THEN BEGIN
                     XRD[FOOT] := GAIN * ERRORX[FOOT];
                     YRD[FOOT] := GAIN * ERRORY[FOOT];
                     ZRD[FOOT] := GAIN * ERRORZ[FOOT];
                     END
                ELSE BEGIN
                     XRD[FOOT] := VELMAX * ERRORX[FOOT] / MAXERR;
                     YRD[FOOT] := VELMAX * ERRORY[FOOT] / MAXERR;
                     ZRD[FOOT] := VELMAX * ERRORZ[FOOT] / MAXERR;
                     END;

            XPD[FOOT] := XPD[FOOT] + XRD[FOOT] * DT;
            YPD[FOOT] := YPD[FOOT] + YRD[FOOT] * DT;
            ZPD[FOOT] := ZPD[FOOT] + ZRD[FOOT] * DT;

            END;  /* FOR FOOT */

        JSERVO( MOVE );
```

```
          PTIME   := TIME;
          DT      := (PTIME - LTIME) * 3600.0;
          LTIME   := PTIME;

     UNTIL MAXERR < 0.5;

     FOR FOOT := 1 TO 6 DO { CLEAN UP GLOBAL VARIABLES }
          BEGIN
          XRD[FOOT] := 0.0;
          YRD[FOOT] := 0.0;
          ZRD[FOOT] := 0.0;

          XPD[FOOT] := XCOORD[FOOT];
          YPD[FOOT] := YCOORD[FOOT];
          ZPD[FOOT] := ZCOORD[FOOT];
          END;

END;  /* FOOTLINE */
```

```
                    /***************************/
                    /*    PROCEDURE: HALT    */
                    /***************************/

/*****************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                      */
/* DATE:        20-APR-82                                        */
/*                                                               */
/* FUNCTION:    STOPS ALL MOTION OF HEXAPOD                      */
/* USER GUIDE:  THE CALLING FORMAT IS:                           */
/*              HALT;                                            */
/*                                                               */
/* PROCEDURES CALLED:   RBDAC                                    */
/*                                                               */
/* GLOBAL VARIABLES                                              */
/*      REFERENCED: NONE                                         */
/*                                                               */
/*      MODIFIED:       VELX,   VELY,   DPSI                     */
/*                      NVELX,  NVELY,  NDPSI                    */
/*                                                               */
/*****************************************************************/

PROCEDURE HALT;

CONST VSCALE = 10.0;

VAR     JOINT   :INTEGER;       /* JOINT INDEX          */
        VOLT    :ARRAY18;       /* OUTPUT VOLTAGES      */

PROCEDURE  RBDAC( NOCHAN, FRSTCH: INTEGER;  SCALE: REAL;  VOLT: ARRAY18 );
        EXTERNAL;

PROCEDURE TURNOF;  EXTERNAL;

BEGIN
    FOR JOINT:= 0 TO 17 DO
        VOLT[JOINT] :=0.0;

    RBDAC( 18, 0, VSCALE, VOLT);   /* TURN OFF VOLTAGE TO ALL MOTORS */

    TURNOF;    /* TURN OFF MOTOR POWER */

    /*** INITIALIZE VELOCITY VECTORS ***/
    NVELX   := 0.0;   NVELY   := 0.0;   NDPSI   := 0.0;
    VELX    := 0.0;   VELY    := 0.0;   DPSI   := 0.0;

    WRITELN;
    WRITELN( CHR(27),'#6---HEXAPOD STOPPED---');

END;  /* HALT */
```

```
(*$E+*)

              /****** FILE: LIBR34.PAS ******/

                   /*************************/
                   /*   FUNCTION:  ATAN2   */
                   /*************************/

/*****************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                      */
/* DATE:        20-APR-82.                                       */
/* FUNCTION:    IMPLEMENT THE FOUR - QUADRANT                    */
/*              INVERSE TANGENT FUNCTION                         */
/*                                                               */
/* USER GUIDE:  THE CALLING FORMAT IS:                           */
/*              ATAN2(Y, X);                                     */
/*              WHERE:  Y IS THE SIDE OPPOSED TO THE ANGLE       */
/*                      X IS THE SIDE ADJACENT TO THE ANGLE      */
/*                                                               */
/*****************************************************************/

FUNCTION  ATAN2(Y, X : REAL) : REAL;

FUNCTION SIGN( X: REAL ): REAL; EXTERNAL;

BEGIN

IF ABS(X) > 0.00001
    THEN   { X IS NON-ZERO }
        IF X > 0.0
            THEN
                ATAN2 := ARCTAN(Y/X)
            ELSE
                ATAN2 := ARCTAN(Y/X) + PI * SIGN( Y )
        { END IF X }
    ELSE
        ATAN2 := PI / 2.0 * SIGN( Y );

END;   /* ATAN2 */
```

```
                    /************************/
                    /*   FUNCTION:  SIGN   */
                    /************************/

/*****************************************************************/
/* PROGRAMMER:   DENNIS PUGH                                    */
/* DATE:         20-APR-82                                      */
/* FUNCTION:     IMPLEMENT THE SIGNUM FUNCTION                  */
/*                                                              */
/* USER GUIDE:   THE CALLING FORMAT IS:                         */
/*               SIGN( X );                                     */
/*                                                              */
/*****************************************************************/

FUNCTION  SIGN( X : REAL ) : REAL;

BEGIN
    IF X >= 0.0
        THEN SIGN := 1.0
        ELSE SIGN := -1.0
END;
```

```
(*$E+*)

            /****** FILE: DLNK34.PAS ******/

                      /************************/
                      /* PROCEDURE: RBADC  */
                      /************************/

/*****************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                     */
/* DATE:        20-APR-82                                       */
/* FUNCTION:    READ IN INPUT VARIABLES: FORCE, RATE, &         */
/*              POSITION.                                       */
/*                                                              */
/* USER GUIDE:  THE CALLING FORMAT IS:                          */
/*              RBADC( NOCHAN, FRSTCH, SCALE, INDATA );         */
/*            WHERE:                                            */
/*              NOCHAN IS THE NUMBER OF CHANNELS TO BE READ     */
/*              FRSTCH IS THE FIRST CHANNEL TO BE READ:         */
/*                0 FOR READING FORCE VARIABLES                 */
/*                18 FOR READING JOINT RATES                    */
/*                36 FOR READING JOINT ANGLES                   */
/*              SCALE IS THE INPUT SCALING FACTOR               */
/*              INDATA IS THE INPUT BUFFER                      */
/*                                                              */
/*                                                              */
/* PROCEDURES CALLED: NONE                                      */
/*                                                              */
/* GLOBAL VARIABLES: NONE                                       */
/*                                                              */
/*****************************************************************/

PROCEDURE RBADC(NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18);

VAR     I, J                    /* DATA POINTERS */
            : INTEGER;

        ADPORT  ORIGIN 166000B  /* LOCATION OF DATA LINK SHARED MEMORY */

            : ARRAY[ 0..63 ] OF INTEGER;

BEGIN  /* RBADC */

J := FRSTCH MOD 18;     /* OFFSET FROM START OF VARIABLE TYPE */

FOR  I := J TO (J + NOCHAN - 1) DO  { CONVERT & SCALE DATA }

    INDATA[I] := (777B - ADPORT[ FRSTCH-J + I ]) / 777B * SCALE;

END;   /* RBADC */
```

```
                    /************************/
                    /*  PROCEDURE:  RBDAC  */
                    /************************/

/*****************************************************************/
/* PROGRAMMER:  DENNIS PUGH                                      */
/* DATE:        20-APR-82                                        */
/* FUNCTION:    SEND OUTPUT VOLTAGES TO DATA LINK                */
/*                                                               */
/* USER GUIDE:  THE CALLING FORMAT IS:                           */
/*              RBDAC( NOCHAN, FRSTCH, SCALE, OUTDATA )          */
/*            WHERE:                                             */
/*              NOCHAN IS THE NUMBER OF OUTPUT CHANNELS          */
/*              FRSTCH IS THE FIRST CHANNEL TO BE OUTPUT         */
/*              SCALE IS THE OUTPUT SCALING FACTOR               */
/*              OUTDATA IS THE OUTPUT BUFFER                     */
/*                                                               */
/*                                                               */
/* PROCEDURES CALLED: NONE                                       */
/*                                                               */
/* GLOBAL VARIABLES: NONE                                        */
/*                                                               */
/*****************************************************************/

PROCEDURE RBDAC(NOCHAN ,FRSTCH: INTEGER; SCALE: REAL; OUTDATA: ARRAY18);

CONST   READY = 200B;

VAR     I,                         /* DATA POINTER                  */
        TEMP                       /* TEMPORARY STORAGE FOR OUTPUT */
                : INTEGER;

        STATUS  ORIGIN 166244B  /* DATA LINK STATUS WORD         */
                : INTEGER;

        OUTPORT ORIGIN 166200B  /* DATA LINK OUTPUT PORTS        */
                : ARRAY[0..17] OF INTEGER;


BEGIN  /* RBDAC */

FOR I := FRSTCH TO (FRSTCH + NOCHAN - 1) DO
    BEGIN
    TEMP := 177B - ROUND(OUTDATA[I] * 177B / SCALE);
    WHILE (STATUS AND READY) = 0 DO { WAIT UNTIL DATA LINK READY };
    OUTPORT[I] := TEMP;
    END;   /* DO */

END;  /* RBDAC */
```

```
PROCEDURE CLOCKINIT;

VAR     STATUS  ORIGIN 170404B          : INTEGER;
        PRELOAD ORIGIN 170406B          : INTEGER;
        CLOCK   ORIGIN 170416B          : INTEGER;

BEGIN
PRELOAD := 0;  { USE FULL COUNTER INTERVAL }
STATUS := 413B;  { ENABLE COUNTER AT 100 Hz }
LASTCLOCK := CLOCK;
TOTALCLOCK := 0;
END;



FUNCTION DELTATIME : REAL;

VAR     CLOCK   ORIGIN 170416B          : INTEGER;
        NOWCLOCK                        : INTEGER;
        NUMTICKS                        : INTEGER;

BEGIN

NOWCLOCK := CLOCK;
NUMTICKS := ( NOWCLOCK - LASTCLOCK ) AND 377B;
TOTALCLOCK := TOTALCLOCK + NUMTICKS;
LASTCLOCK := NOWCLOCK;

DELTATIME := NUMTICKS / 100.0;

END;
```

```
(*$E+*)

                  /******   FILE:  POWR34.PAS    ******/

/**************************************************************************/
/*                                                                        */
/*      PROGRAMMER: DENNIS PUGH                                           */
/*      DATE: 20-APR-82                                                   */
/*                                                                        */
/*      GLOBAL VARIABLES: NONE                                            */
/*                                                                        */
/*  THIS FILE CONTAINS SUBROUTINES WHICH UTILIZE THE SELF-DIAGNOSTIC      */
/*  CAPABILITIES AND THE REMOTE POWER CONTROL CAPABILITIES OF THE         */
/*  DIGITAL DATA LINK. THE EXTERNALLY USEFUL ROUTINES ARE 'TESTON',       */
/*  'TURNOF', AND 'TURNON'. NONE OF THESE REQUIRE ARGUMENTS.              */
/*                                                                        */
/**************************************************************************/




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

PROCEDURE PWAIT;   /* A ROUTINE WHICH WAITS UNTIL ALL FEEDBACK CHANNELS */
                   /* HAVE BEEN UPDATED AFTER A CHANGE IN STATUS        */
VAR NOW:REAL;

BEGIN
    NOW:=TIME;
    WHILE ((TIME-NOW)*3600.) < 0.004 (* SECONDS *) DO (* NOTHING *) ;
END;  /* PWAIT */

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

```
/*************************************************************************/
/*                                                                       */
/* TURNOF:                                                               */
/*      (TURN OFF) IS A ROUTINE WHICH TURNS OFF THE POWER TO THE         */
/*      SERVO MOTORS. IT SHOULD BE USED BEFORE EXITING THE CONTROL       */
/*      PROGRAM, AND ADDITIONALLY, IT SHOULD BE CALLED WHENEVER THE      */
/*      HEXAPOD IS NOT ACTUALLY SERVOING. THIS WILL INCREASE SAFETY      */
/*      AND REDUCE THE RISK OF DAMAGE TO THE HEXAPOD. THE EFFECTS        */
/*      OF CALLING TURNOF CAN BE REVERSED BY CALLING TURNON.             */
/*                                                                       */
/*************************************************************************/

PROCEDURE TURNOF;

(*$C    COMSTAT=^O166244        ;ADDRESS OF COMMAND & FORWARD STATUS
        POWER=^O166246          ;ADDRESS OF POWER COMMAND WORD

        .MACRO SEND Y,Z,?T
T:      TSTB    COMSTAT         ;CHECK IF FORWARD PATH BUSY
        BPL     T
        MOV     Y,Z             ;SEND DATA
        .ENDM
*)

BEGIN

(*$C    SEND    #0,COMSTAT      ;COMMAND NORMAL MODE
        SEND    #2,POWER        ;TURN OFF MOTOR POWER
*)

END;    /* PROCEDURE TURNOF */

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

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

PROCEDURE ASK;    /* A PROCEDURE WHICH GIVES THE OPTION TO ABORT */

(*$C    .MCALL  EXIT$S  *)

VAR LETTER:CHAR;

BEGIN
    WRITELN('DO YOU WISH TO ABORT?  (Y/N)');
    READLN(LETTER);
    IF LETTER = 'Y' THEN
        BEGIN
        WRITELN;
        WRITELN(' *** PROGRAM ABORTED ***');
        TURNOF;
        (*$C EXIT$S *)  /* EXIT THE PROGRAM */
        END;
END;  /* PROCEDURE ASK */

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

```
/*************************************************************************/
/*                                                                       */
/*  TURNON:                                                              */
/*      (TURN ON) IS A ROUTINE WHICH TURNS ON THE HEXAPOD MOTOR &        */
/*      ELECTRONICS POWER. IT ASSUMES THAT 'TESTON' HAS BEEN CALLED      */
/*      SUCCESSFULLY. IT SHOULD BE CALLED AT THE START OF EVERY          */
/*      REAL TIME SECTION (WHEN THE HEXAPOD IS ACTIVELY SERVOING).       */
/*                                                                       */
/*************************************************************************/

PROCEDURE TURNON;

(*$C    COMSTAT=^O166244        ;ADDRESS OF COMMAND & STATUS WORDS
        POWER=^O166246          ;ADDRESS OF POWER COMMAND WORD
        PSTAT=^O166176          ;ADDRESS OF POWER STATUS WORD

        .MACRO SEND Y,Z,?T
T:      TSTB    COMSTAT         ;CHECK IF FORWARD PATH BUSY
        BPL     T
        MOV     Y,Z             ;SEND DATA
        .ENDM
*)
```

```
BEGIN
        (*$C    BIT     #4,PSTAT                    ;TEST KILL CIRCUIT STATUS
                BEQ     KIL                         ;BRANCH IF INACTIVE
        *)

        WRITELN;
        WRITELN('THE KILL CIRCUIT IS ACTIVE, PLEASE DEACTIVATE.');

        (*$C
        WAITK:  BIT     #4,PSTAT          ;TEST KILL CIRCUIT
                BNE     WAITK             ;LOOP UNTIL DEACTIVATED
        *)

        WRITELN(' THANK YOU');

        (*$C
        KIL:    SEND    #3,POWER          ;TURN ON ELECTRONICS & MOTOR POWEK
        *)

        PWAIT;
        PWAIT;

        (*$C    MOV     @#PSTAT,R1        ;CHECK FOR MP & EP ON
                BIC     #^C0177774,R1     ;MASK OFF MP & EP STATUS BITS
                CMP     #3,R1             ;CHECK IF BOTH BITS ON
                BEQ     DONE              ;BRANCH IF CORRECT
        *)

        WRITELN;
        WRITELN(' POWER CONTROLLER MALFUNCTION');
        ASK;
        WRITELN;
        WRITE('PLEASE SWITCH TO MANUAL MODE,');
        WRITELN(' THEN TURN ON YELLOW & GREEN LIGHTS');

        (*$C
        WAITP:  MOV     @#PSTAT,R1        ;CHECK FOR MP & EP ON
                BIC     #^C0177774,R1     ;MASK OFF MP & EP STATUS BITS
                CMP     #3,R1             ;CHECK IF BOTH BITS ON
                BNE     WAITP             ;LOOP UNTIL POWER ON
        *)

        WRITELN;
        WRITELN(' THANK YOU');

        (*$C    DONE:
        *)

END;  /* TURNON */
/****************************************************************************/
```

```
/*************************************************************************/
/*                                                                       */
/*  TESTON:                                                              */
/*      (TEST & TURN ON) IS AN INTERACTIVE PROCEDURE WHICH SHOULD BE     */
/*      CALLED AT THE START OF EVERY REAL-TIME HEXAPOD CONTROL PROGRAM.  */
/*      IT AUTOMATICALLY TESTS DATA LINK OPERATION AND THEN TURNS ON     */
/*      THE POWER SYSTEMS IN THE PROPER SEQUENCE. ANY ABNORMAL           */
/*      CONDITIONS ARE REPORTED TO THE OPERATOR.                         */
/*                                                                       */
/*************************************************************************/

PROCEDURE TESTON;

VAR     I,
        N,
        DATA,
        LOC:
                INTEGER;

(*$C    ADBASE=^O166000         ;BASE ADDRESS OR A/D'S
        DABASE=^O166200         ;BASE ADDRESS OF D/A'S
        DABUF=^O166236          ;D/A CHANNEL USED FOR FORWARD TESTS
        COMSTAT=^O166244        ;ADDRESS OF FORWARD STATUS WORD
                                ;  AND MODE COMMAND WORD
        PSTAT=^O166176          ;ADDRESS OF POWER STATUS WORD
        POWER=^O166246          ;ADDRESS OF POWER COMMAND WORD

        .MACRO SEND Y,Z,?T
T:      TSTB    COMSTAT         ;CHECK IF FORWARD PATH BUSY
        BPL     T
        MOV     Y,Z             ;SEND DATA
        .ENDM
*)

BEGIN

        WRITELN;
        WRITE('PLEASE TURN ON DATA LINK POWER (RED LIGHT),');
        WRITELN(' THEN ENTER A CARRIAGE RETURN.');
        READLN;

        (*******  FEEDBACK TEST ROUTINE  *******)

        WRITELN(' PERFORMING DATA LINK FEEDBACK TEST');

        (*$C    SEND    #0,POWER        ;TURN OFF POWER SYSTEMS
                SEND    #1,COMSTAT      ;COMMAND TEST MODE #1
        *)
```

```
PWAIT;
PWAIT;
N:=0;
FOR I:=1 TO (MAXINT DIV 64) DO
        BEGIN
        (**C    MOV     #^O77,R1          ;INITIALIZE ADDRESS COUNTER
        LOOP:   ASL     R1                ;CONVERT TO WORD ADDRESS
                MOV     ADBASE(R1),R2     ;FETCH DATA
                ASR     R1                ;RESTORE COUNTER
                MOV     R1,R3             ;GENERATE EXPECTED DATA IN R3
                SWAB    R3                ;  BY DUPLICATING BITS 0-3
                ASR     R3                ;  IN BITS 6-9
                ASR     R3
                ADD     R1,R3
                BIC     #^O176000,R3      ;DATA NOW GENERATED
                CMP     R2,R3             ;CHECK IF DATA IS CORRECT
                BEQ     CONT
                MOV     R1,LOC(SP)        ;SAVE ADDRESS OF BAD DATA
        *)
                N:=N+1;
                IF N<20 THEN
                    BEGIN
                    WRITE(' ERROR DETECTED ON LOOP',I,', ADDRESS',LOC);
                    WRITELN;
                    END;

        (**C
        CONT:   DEC     R1                ;DECREMENT ADDRESS COUNTER
                BPL     LOOP              ;LOOP UNTIL R1<0
        *)

        END;  /* FOR I */

IF N <> 0 THEN
        BEGIN
        WRITELN;
        WRITELN(N,' TRANSMISSION ERRORS DETECTED IN',MAXINT,' TESTS.');
        ASK;
        END;


(*******  FEEDFORWARD TEST ROUTINE  *******)

WRITELN;
WRITELN(' PERFORMING DATA LINK FEEDFORWARD TEST');

(**C    SEND    #2,COMSTAT      ;COMMAND TEST MODE #2
*)
```

```
N:=0;
PWAIT;
FOR DATA:=0 TO 127 DO
        BEGIN
        (**C    SEND    DATA(SP),DABUF  ;SEND DATA
        *)

        PWAIT;

        (**C    MOV     @#ADBASE,R1     ;READ BACK DATA
                CMPB    R1,DATA(SP)     ;COMPARE DATA SENT & RECIEVED
                BEQ     GOOD            ;BRANCH IF DATA GOOD
                INC     N(SP)           ;INCREMENT ERROR COUNTER
        GOOD:   COM     DATA(SP)        ;FLIP BITS FOR NEXT OUTPUT
                SEND    DATA(SP),DABUF  ;SEND DATA
        *)

        PWAIT;

        (**C    MOV     @#ADBASE,R1     ;READ BACK DATA
                CMPB    R1,DATA(SP)     ;COMPARE DATA SENT & RECIEVED
                BEQ     GOOD2           ;BRANCH IF DATA GOOD
                INC     N(SP)           ;INCREMENT ERROR COUNTER
        GOOD2:  COM     DATA(SP)        ;RESTORE DATA
        *)

        END;  /* FOR DATA */

IF N <> 0 THEN
        BEGIN
        WRITELN;
        WRITELN(N,' TRANSMISSION ERRORS DETECTED IN 256 TESTS.');
        ASK;
        END;


(*******  POWER CONTROLLER MODE TEST  *******)

(**C    SEND    #0,COMSTAT      ;COMMAND NORMAL MODE
*)

PWAIT;
PWAIT;

(**C    TSTB    PSTAT   ;TEST IF POWER CONTROLLER IN COMPUTER MODE
        BMI     COMP    ;BRANCH IF IT IS
*)

WRITELN;
WRITELN('PLEASE PLACE POWER CONTROLLER IN COMPUTER MODE.');
```

```
        (*$C
        WAITC:  TSTB    PSTAT           ;TEST FOR COMPUTER MODE
                BPL     WAITC           ;LOOP UNTIL MODE CHANGED
        *)

        WRITELN(' THANK YOU');

        (*$C    COMP;
        *)


        (*******  ZERO DAC ROUTINE  *******)

        FOR I:=0 TO 17 DO
                BEGIN
                (*$C    MOV     I(SP),R1            ;MOVE I TO R1
                        ASL     R1                  ;CONVERT TO WORD ADDRESS
                        SEND    #^O177,DABASE(R1)   ;SEND ZERO VOLTS TO JOINT
                *)
                END;

        TURNON;

        WRITELN;
        WRITELN(' INITIALIZATION COMPLETE');
        WRITELN;

END;    /* TESTON */

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

```
;               /******* FILE:  KYCK34.MAC  ******/
;
; ********************************************************************
; *                                                                  *
; *  PROGRAMMERS:  CHARLES KLEIN, DENNIS PUGH                        *
; *  DATE:  20-APR-82                                                *
; *                                                                  *
; *  GLOBAL VARIABLES: NONE                                          *
; *                                                                  *
; *  PURPOSE:  ASSEMBLER LEVEL KEYBOARD INPUT FOR SPECIAL-PURPOSE    *
; *            USE WITH HIGH LEVEL LANGUAGES.                        *
; *                                                                  *
; *  SUBROUTINES CONTAINED:                                          *
; *                                                                  *
; *     KEYINT: INITIALIZES TERMINAL CHARACTERISTICS AND PLACES AN   *
; *             INPUT REQUEST ON THE SYSTEM QUEUE. KEYINT SHOULD BE  *
; *             CALLED ONLY ONCE. NO ARGUMENTS ARE REQUIRED.         *
; *                                                                  *
; *     KEYCK:  RETURNS THE CHARACTER TYPED ON THE KEYBOARD IN ITS   *
; *             ONE ARGUMENT (VARIABLE TYPE CHAR). IF NO CHARACTER HAS *
; *             BEEN INPUT SINCE THE LAST CALL, A NULL CHARACTER IS  *
; *             RETURNED (ASCII 0).  SAMPLE CALL: KEYCK(CHARACTER);  *
; *                                                                  *
; *     KYWAIT: IS TO BE CALLED WHEN IT IS DESIRED TO SUSPEND PROGRAM *
; *             EXECUTION UNTIL A CHARACTER IS INPUT, THUS FREEING THE *
; *             SYSTEM FOR OTHER TASKS. EXECUTION IS RESUMED WHEN AN *
; *             INPUT IS RECIEVED. KEYCK CAN THEN BE USED TO FETCH THE *
; *             CHARACTER. NO ARGUMENTS ARE REQUIRED.                *
; *                                                                  *
; *     IOKILL: IS USED TO CANCEL THE INPUT REQUEST WHEN A HIGH-LEVEL *
; *             READ (READLN) IS TO BE PERFORMED. NO ARGUMENTS ARE   *
; *             REQUIRED.                                            *
; *                                                                  *
; *     INQUE:  IS CALLED TO REASSERT THE INPUT REQUEST AFTER AN IOKILL *
; *             AND HIGH-LEVEL READ. NO ARGUMENTS ARE REQUIRED.      *
; *                                                                  *
; *                                                                  *
; *         *** ALL SUBROUTINES ARE IN FORTRAN FORMAT. ***           *
; *                                                                  *
; ********************************************************************
;
;
        .LIST   TTM
        .MCALL  QIO$S,WTSE$S,ALUN$S
;
; LOCAL SYMBOL DEFINITION
        LUN2    = 2
        EFN1    = 11
        EFN2    = 12
;
```

```
;
; LOCAL DATA BLOCKS:
;
        .PSECT  DATA,D,RW
MSG:    .BLKB   1                 ;CHARACTER BUFFER
        .EVEN
TCHAR:  .BYTE   TC.FDX,1          ;TERMINAL CHARACTERISTICS
IOST:   .BLKW   2                 ;I/O STATUS RETURN LOCATION
;
        .PSECT
;
KEYINT::
        ALUN$S  #LUN2,#'TI                              ;ASSIGN LUN TO TERMINAL
        QIO$S   #IO.ATT,#LUN2                           ;ATTACH TERMINAL
        QIO$S   #SF.SMC,#LUN2,,,,,<#TCHAR,#2>           ;SET TO FULL DUPLEX
        QIO$S   #IO.RNE,#LUN2,#EFN2,,#IOST,,<#MSG,#1>   ;ASSERT INPUT REQUEST
        BCS     ERROR                                   ;SIGNAL IF ERROR
        RTS     PC
;
;
KEYCK:: TST     (R5)+                                   ;POINT R5 TO ARGUMENT
        CLRB    @0(R5)                                  ;PUT NULL IN ARGUMENT
        CMPB    #IS.SUC,IOST                            ;TEST FOR SUCCESSFUL READ
        BNE     RETURN                                  ;RETURN IF NOT
        MOVB    MSG,@0(R5)                              ;PUT CHAR. IN ARGUMENT
INQUE:: QIO$S   #IO.RNE,#LUN2,#EFN2,,#IOST,,<#MSG,#1>   ;REASSERT INPUT REQUEST
        BCS     ERROR                                   ;SIGNAL IF ERROR
RETURN: RTS     PC
ERROR:  MOV #2,R0                                       ;SET ERROR FLAG
        IOT
;
;
KYWAIT::
        WTSE$S  #EFN2             ;WAIT FOR KEYBOARD INPUT WITH LITTLE OVERHEAD
        RTS     PC
;
;
IOKILL::
        QIO$S   IO.KIL,#LUN2,#EFN1          ;CANCEL I/O REQUESTS
        WTSE$S  #EFN1                       ;WAIT FOR COMPLETION
        RTS     PC
;
        .END
```

```
;          FILE: CMPL34.CMD
;
; FUNCTION: INDIRECT COMMAND FILE TO
;           COMPILE & ASSEMBLE ALL FILES
;           OF ROBOT 3.4
;
PAS ROBT34=GBLF34,ROBT34
MAC ROBT34=ROBT34
PAS PLAN34=GBLF34,PLAN34
MAC PLAN34=PLAN34
PAS FOOT34=GBLF34,FOOT34
MAC FOOT34=FOOT34
PAS SERV34=GBLF34,SERV34
MAC SERV34=SERV34
PAS INIT34=GBLF34,INIT34
MAC INIT34=INIT34
PAS LINE34=GBLF34,LINE34
MAC LINE34=LINE34
PAS LIBR34=GBLF34,LIBR34
MAC LIBR34=LIBR34
PAS DLNK34=GBLF34,DLNK34
MAC DLNK34=DLNK34
PAS POWR34=POWR34
MAC POWR34=POWR34
MAC KYCK34=KYCK34
TKB @ROBT34
```

```
;       FILE: ROBT34.CMD
;
; FUNCTION: PROVIDES LINKAGE INFORMATION
;           FOR THE TASK BUILDER
;
ROBT34/FP,ROBT34/CR/-SP=ROBT34
PLAN34
FOOT34
LIBR34
SERV34
DLNK34
INIT34
LINE34
KYCK34
POWR34
[1,1]PASLIB/LB
/
COMMON=IOPAGE:RW
extsct=$$heap:5000
UNITS=7
//
```

DATE
FILME
— 8