DESIGN AND IMPLEMENTATION OF A MULTILEGGED WALKING ROBOT

A Senior Honors Thesis Presented
by

Willard S. MacDonald

Submitted May 1994

APPROVED

____
Roderic Grupen, Computer Science

____
Theodore Djaferis, Electrical and Computer Engineering

____
Robert Jackson, Electrical and Computer Engineering

Abstract

An autonomous four legged 12 degree-of-freedom walking robot was designed, constructed and successfully tested by a single undergraduate student in less than eleven months for less than $1000. The project was undertaken to provide a platform for the study of sensor-based legged robots in the Laboratory for Perceptual Robotics at The University of Massachusetts–Amherst.

Contents

List of Figures

List of Tables

Acknowledgements

The story all started when I walked into the LPR and asked Professor Rod Grupen if I could build a robot that would be useful to the work in the lab. His response was enthusiastic and welcoming. I appreciate his openness to outsiders and to new ideas, and I am grateful for his acceptance of me into the lab.

Thank you to Professor Robert Jackson and to Professor Ted Djaferis for their service on my advisory committee and for their enthusiasm and interest in the project. And a special thank you to Professor Jackson for the the use of the milling machine in his lab–it was an invaluable asset to the project.

Thank you for the interest and support of Rebekah B., Pierre T., and my family, Mom, Dad, Bob, and Nate. And thank you to T. Baird Soules for all his HC11 techno-gossip and references.

Chapter 1

Introduction

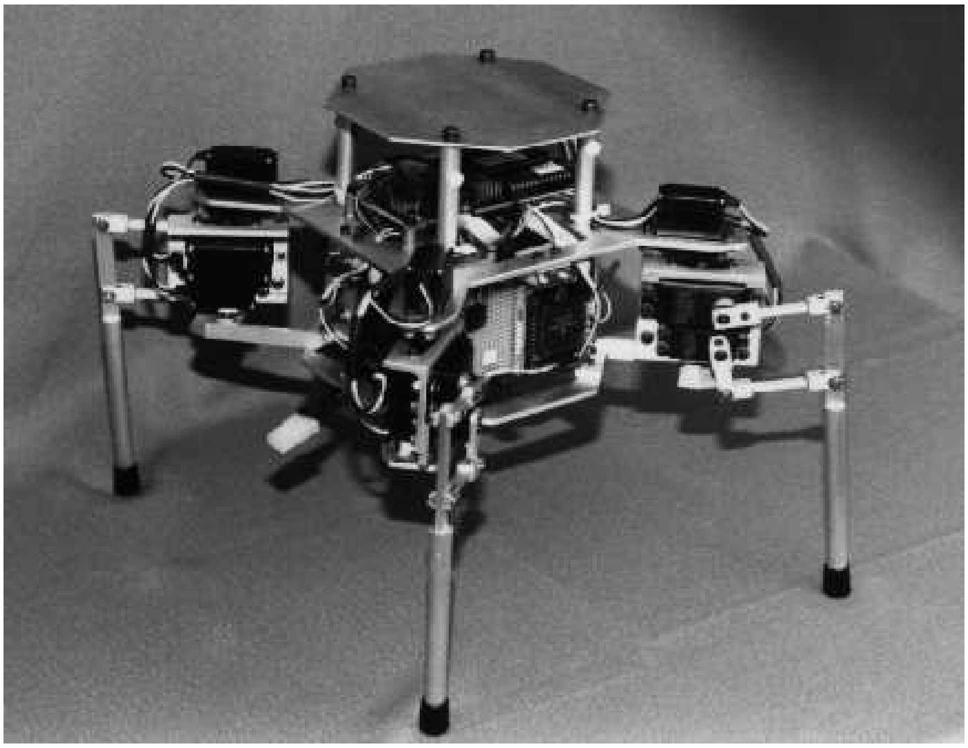


Figure 1.1: Thing

1.1 Legged Robots

The importance of legged robots is widely recognized (Bekker 69), (Waldron et. al. 84), (Hirose 84), (Hodgins 88), (Angle and Brooks 90). As a means of transportation and exploration, legs are more flexible than wheels over soft or irregular terrains. Most of the earth's surface is, in fact, untraversable by wheeled vehicles.

As a control problem, legged robotics takes on more importance in its expansive relevance to the development of AI (artificial intelligence). While there are many issues relevant specifically to legged robotics (and to robotics as a whole), various subgroups in AI share many of the same goals and problems as the autonomous control of legged robots. In fact, robotics is generally thought of as a sub-group of AI.

This thesis describes an approach to the design and construction of a legged robot. It does not focus on the high-level control of such a robot; although, inevitably hardware design has an important effect on high-level control.

1.2 Motivation

The LPR (Laboratory for Perceptual Robotics) at the University of Massachusetts (department of Computer Science) is investigating the design of intelligent systems. The lab has been successful in controlling a 25 degree-of-freedom robotic hand to do reaching and grasping tasks from on-line visual and haptic sensor information.

It is believed that the sensor-based controllers developed for the robotic hand have general applicability. One test of this conjecture will be on a legged robot. Legs are very similar to fingers, and a legged robot could even pass as a hand if the legs were oriented at a reasonable distance from one another. It was desired, in fact, that the robot under design be able to implement such a paradigm and hence demonstrate the controllers' generality. It is based on this desire that it was named "Thing" after the disembodied hand from the TV show The Addams Family. However, Thing will accomplish more than simply demonstrating the flexibility of the hand controllers. It will serve as a platform for the general study of sensor-based walking robots in the lab.

1.3 Organization of Thesis

Chapter 2 lays out the goals of the project. These goals drive most of the decisions in the chapters that follow. The goals are broken into two types: *fixed goals* and *flexible goals*.

Chapter 3 discusses design decisions about the overall robot design. Decisions about the scale of the robot, number of legs, and materials are discussed.

Chapters 4, 5, and 6 discuss design issues specifically relevant to the leg, body, and electronics designs respectively. Chapter 7 solves for the resulting forward and inverse kinematics of the legs.

Chapter 8 discusses distribution of the control problem in software and gives the algorithms used for the final demo.

Chapter 9 gives a summary of the project and discusses plans for future work.

Chapter 2

Design Goals

The overall goal of the project was to design and construct a platform suitable for the study of coordinated multi-legged locomotion.

In determining the specific goals for this project, it was found that there were certain aspects that were imperative to the usefulness of the platform and others that were flexible and could be set arbitrarily for convenience. These are grouped below as *fixed goals* and *flexible goals*.

In setting goals for any project, one must have a point of reference from which to judge reality. There have been many legged robots built in the last 20 years of many different configurations and sizes, and it would be foolish not to use the trials and tribulations of these projects to qualify the goals of a new design. Many of the *flexible goals*, especially, were based on the accomplishments of other designers: (Angle 89), (Angle 91), and (Hirose 84).

Fixed Goals

These are goals that must be met for the platform to be useful and for the project to fit into the undergraduate research guidelines.

- **The total cost must be less than $1000.** This is an important design constraint that was determined by the fellowship support provided for the project.
- **The design and construction of the robot must be completed in less than twelve months**–another important design constraint this time determined by the designer's graduation date.
- **The platform must be physically capable of traversing uneven terrain while maintaining its stability.**
- **The robot must be capable of arbitrary navigation.** That is, it should be able to walk to any arbitrary location in the plane by turning and walking forward or by side-stepping.
- **The robot must have a computer system that is capable of running the sensor-based control algorithms.**
- **The robot must be expandable to include sensors capable of sensing the environment well enough to form a local (to each leg) model of the terrain** The local sensing will allow the robot to choose the best foot placements.

Flexible Goals

These are goals that are not critical to the usefulness of the platform and are set based on reasonable guesses and translations from existing designs. These were used as starting points for the design of numerous parts but were reduced, increased, or discarded when necessary.

- **The platform should be primarily a walking robot, but should also function successfully as a hand.** If the robot can function in both configurations with the same controller, it will most dramatically demonstrate the versatility of the sensor-based control theory.
- **The robot should be capable of functioning completely autonomously.**
- **The robot should have the following locomotive abilities (speed and acceleration specifications are assumed to be for level ground):**

$$\text{max step height} = 4 \text{ in.}$$
$$\text{steepest traversable slope} = 30 \text{ degrees}$$
$$\text{max walking speed} = 0.75 \text{ in/sec}$$
$$\text{max acceleration} = 0.6 \text{ in/sec}^2$$

- **The robot should be expandable to include sensors capable of some form of high-level navigational sensing.** This navigational sensing would allow the platform to have a more interesting existence as an autonomous robot.
- **The robot should be able to function for at least 30 minutes un-tethered on one charging of the batteries.**
- **The robot should be a distributed system.** This is to ease the trouble-shooting of the system and to promote incremental development as well as to simplify the control problem.
- **The robot should be a modular system.** This is so that new systems can be added and old ones can be replaced quickly and easily.

2.1 Completion Criteria

While the ultimate behavior of the robot will be to intelligently traverse rough terrain, utilizing sensor-based leg controllers, to reach this stage requires too much work for one person (working part-time) to complete in one year. For this reason, completion of the first phase of the project, in fulfillment of the honors research requirements, was to be marked by the demonstration of very simple coordinated walking–as that which could be produced by a finite state machine controller. It was not to utilize any sensors or carry its own power supply. The remaining work was to be continued after the allotted twelve month design and construction period.

Chapter 3

Robot Design

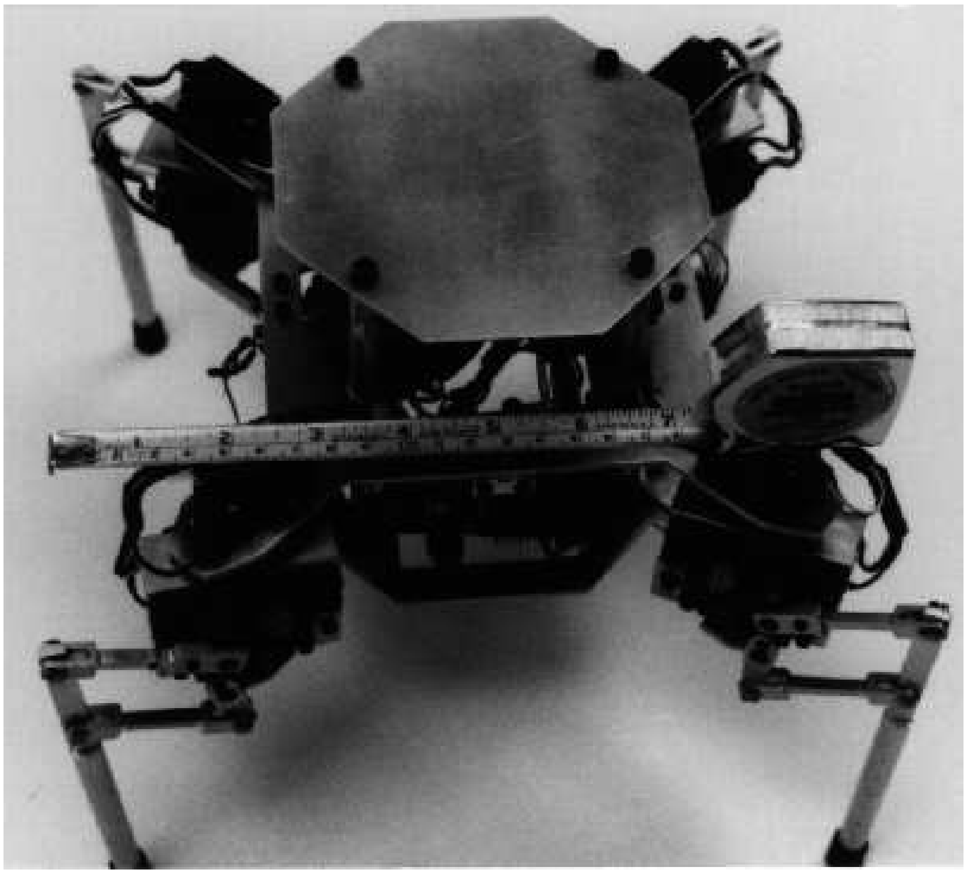


Figure 3.1: Thing is approximately 10 inches square and 9 inches tall.

3.1 Existing Designs

The task of designing such a robot is rather prodigious when taken as a whole. Meeting all of the goals in a single working design was a challenge.

With the many legged robot designs in existence, it is possible to take the existing work and utilize it to the benefit of a new design. The reason for a new design in the first place is that the characteristics of the old designs do not meet the new requirements. But these old designs represent years of good work and can, therefore, be expanded, dissected, and grouped eclectically to utilize their good points. Such designs include the Adaptive Suspension Vehicle (Pugh et al), Ambler (Bares 89), Genghis (Angle 89), Attila (Angle 91), PVII (Hirose and Umetani 84), and Titan III (Hirose et al 84).

The successful design of the two different hexapods, Genghis and Attila, were particularly inspiring. Their small scales were unique characteristics that provided certain advantages over prior legged robot designs. Two advantages were particularly applicable to the constraints on the design of Thing: low cost and quick design/implementation time. This is discussed in section 3.3.

The energy efficiency of the leg design in the PVII quadruped provided insight into ways to limit energy consumption and hence increase battery life. This issue is discussed in section 4.3.

3.2 Design Procedure

Typically, when a new system is to be designed, one accepted method of proceeding is to design the system completely–possibly incorporating some modeling or simulation–essentially trying to eliminate as much ambiguity as possible, then completely implement the system and see if it works. The amount of time spent in the theoretical design stages is usually very large; but the results are not always representative of this time since everything was done without experimental verification. Often a design in real life is different than it is on paper or in a simulation. After such a system has been constructed, so much time and money has been put into the design that it is not feasible to redesign it. If the result is not what was intended, then the system remains suboptimal. This is not to say that theoretical analysis should be avoided, on the contrary, it is an important stage in design. However, in certain cases, the system can be incrementally designed, alternating theoretical analysis with experimental analysis, to increase the efficiency of the design process.

The type of loose design and construction described above was used for this project. Much of the design occurred "on the fly." The design process followed is depicted in fig 3.2. An initial design of a part was made based on intuition, theoretical analysis, and educated guessing; an attempt was made to fabricate the part; and the design was modified based on the limitations of the fabrication equipment and materials. Once a fabricatable part was accomplished, the part was analyzed for functionality. If it was not functionally acceptable, the process was repeated. The initial design was based on both the *fixed goals* and the *flexible goals*. During the redesign stages, the *flexible goals* were compromised if necessary. The flexibility in the goals was important to the success of the project.

By incrementally re-designing during implementation, the whole cycle of design and construction was shortened. This method required, of course, that the designer also be the machinist, as was the case with this project.

It was found that by using commercially available components like model airplane servos, highly integrated microprocessors, and prefabricated circuit boards, it was possible to create a quick design that was relatively inexpensive and could meet all of the important design criteria.

TECHNIC Legos were used to build prototypes of the legs and robot and to help visualize the designs. This turned out to be invaluable, because it was possible to discover many mechanical nuances in the prototypes that led to better initial designs.

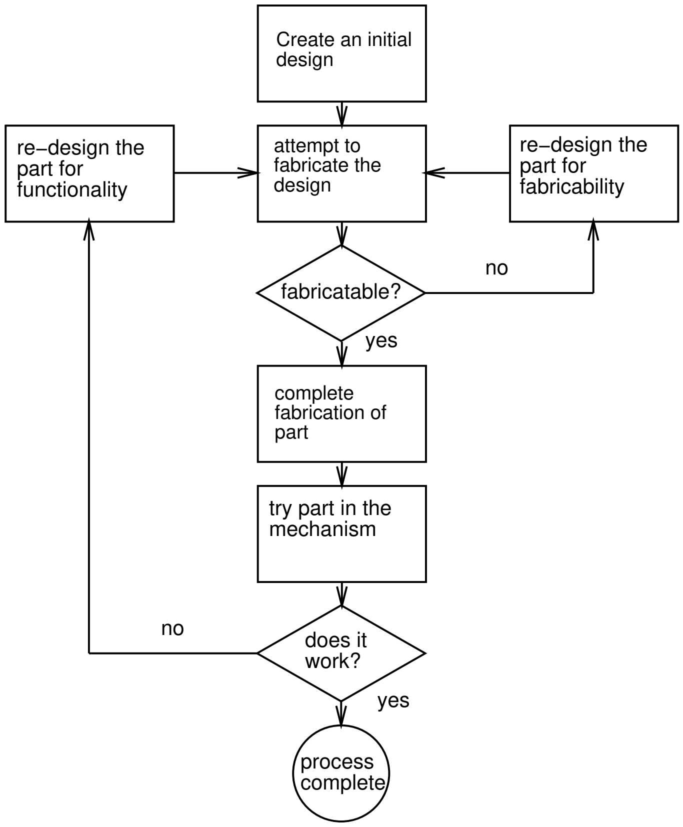


Figure 3.2: The design procedure

3.3 Scale

The cost and time constraints were deciding factors throughout most of the design process. The robot had to be built quickly and inexpensively. This led to a natural choice of scale: *small.* The benefits of small robots over larger ones are many, including a favorable strength-to-weight ratio, reduced dynamic effects, and lower power requirements (Angle 91). Even more importantly, they are cheaper and able to be manufactured with standard equipment in a short amount of time.

The resources immediately available were light aluminum stock and a small milling machine. To go beyond these resources would have required contracting a machine shop and ordering large amounts of metal. Both are expensive time-wise and dollar-wise. Additionally, by conducting the machining in-house, the designer was in closer contact with the limitations involved with milling aluminum and with the capabilities of the milling machine itself. This experience provided for better first-time designs as the project progressed and intuition grew.

3.4 Four Legs

At least one leg must be free to move when the robot is in a particular posture, or it will not be able to progress in a gait. Since Thing is to walk in statically stable gaits only, there must be redundancy in the number of legs. Four legs is the minimum number to still have redundancy in a stable stance.

The problem of maintaining a stable platform is considerably more complex with four legs than it is with five, six or more since to maintain a statically stable platform there must always be at least three legs on the ground at any given time. Hence, with only four legs, a shift in the center of mass is required to take a step. A six legged robot, on the other hand, can always have a stable triangle (one that strictly contains the center of mass). In figure 3.3 two successive postures, or steps, are shown for a four and for a six legged robot. In fig 3.3a the triangle for the first posture is stable because it contains the center of mass, but for the second posture, the center of mass must be shifted in order for the triangle to be stable. In contrast, for the six legged robot in fig 3.3b, the center of mass can remain the same for successive postures.

By choosing four legs, the control problem is preserved in software. The goal of the LPR is to study feedback stabilized multi-manipulator systems as opposed to passively stable hardware design. Therefore, four legs were chosen for Thing. For similar reasons, the LPR uses the four fingered Utah/MIT Dexterous hand for much of its work.

Additionally, a four legged design is in favor of the two main design criterion: low cost and short construction time. One or two fewer legs means less hardware to buy, build and debug.

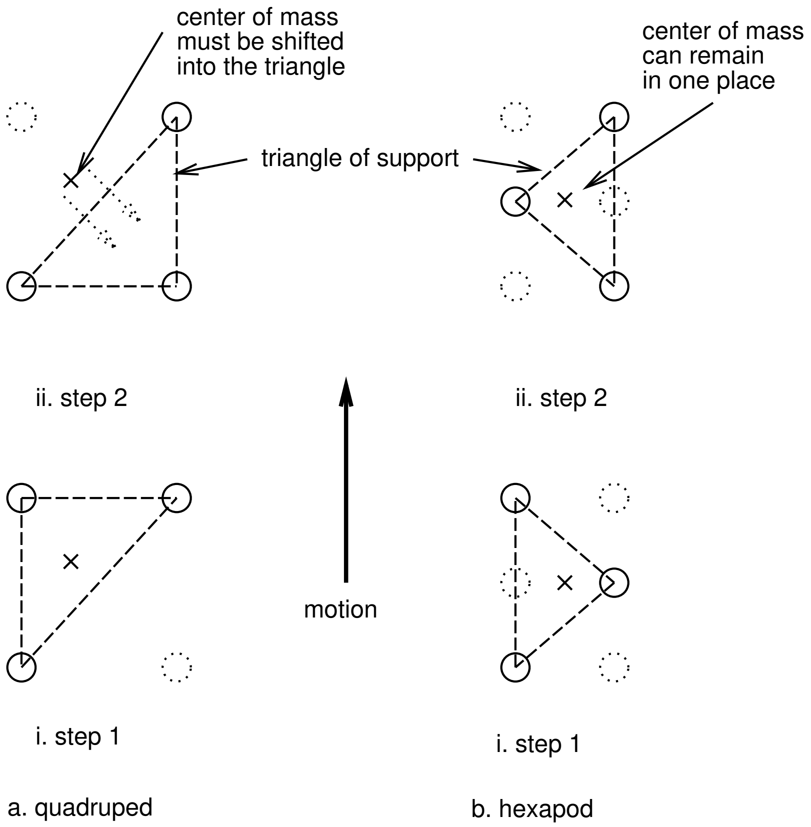


Figure 3.3: Center of mass in quadrupedal and hexapodal walking.

3.5 Weight

Weight assumptions and goals were based on the total robot weight and the distribution of that weight in existing legged robot designs, particularly in a high performance hexapod robot developed at MIT named Attila (Angle 91). A first approximation was that Thing would weight 2/3 the weight of Attila since Thing would have 2/3 as many 3 DOF (degree of freedom) legs. Attila is a more complex robot with more sensors and electronics, but it was also highly optimized for weight minimization. These two opposing characteristics imply that 2/3 is a good first approximation and most likely has a generous safety factor because the additional electronics may be more significant.

This assumption gave an approximate weight for the motors. When the motors had been selected, the total weight approximation was modified to include the known weight of the motors and a change in battery weight. The change in battery weight was based on the assumption that a heavier battery is required to run a heavier robot. The weights of the other parts of the robot were less easily estimated. As progress was made, the weights of these other components were checked and maintained below the approximations, but the weight profile was not modified. The weight distribution assumptions are given in table 3.1.

comparison	Attila		Thing	
robot part	weight	% total weight	weight	% total weight
structure	45.2 oz	44 %	30.1 oz	42 %
electronics	23.0 oz	22 %	15.3 oz	22 %
motors	20.8 oz	20 %	15.6 oz	22 %
batteries	15 oz	14 %	10.3 oz	14 %
total	104 oz	100 %	71.3 oz	100 %

Table 3.1: Weight distribution

3.6 Material

The choice of material must be based on a number of factors:

- strength to weight ratio
- workability (and availability of the proper machinery)
- durability

A qualitative comparison between four common materials is made in table 3.2. These generalizations are based on experimentation with the different materials. The workability was based on the available equipment which included: hobby-size lathe, milling machine, drill press, tap and die equipment, assorted saws and tools. Aluminum was chosen as the best material for the project because it scored the highest in all categories.

material	strength/weight	work-ability	durability
aluminum	good	good	good
steel	medium	medium	good
plexi-glass	bad	good	medium
balsa wood	good	good	bad

Table 3.2: Comparison of available materials

3.7 Budget

The final budget of the project is given in table 3.3. This budget was limited by a $1000 fellowship from the Honors Program at UMass and a small-purchases fund provided by the LPR. The total given does not include miscellaneous purchases or mishap compensation (e.g. wasted aluminum due to milling mistakes).

Qty.	Material	Vendor	Price each	Total
8	Futaba S9601 Servos	Tower Hobbies	$42.00	$336.00
4	Futaba S9201 Servos	Tower Hobbies	$41.00	$164.00
4	CGN1001 HC11 module	CGN	$19.00	$76.00
4	Motorola MC68HC811E2	CGN	$20.00	$80.00
1	HC11 Bootloader	CGN	$10.00	$10.00
1	NMIS-0024 HC11 board	New Micros, Inc.	$113.00	$113.00
-	bearings, shaft, etc	Stock Drive/Sterling	–	$54.00
-	aluminum stock	Sunderland Metal	–	$50.00
-	elec. connectors, etc	Newark	–	$50.00
-	nuts, screws, etc	Serv-U Hardware	–	$30.00
-	Total	–	–	$963.00

Table 3.3: Summary of budget

Chapter 4

Leg Design

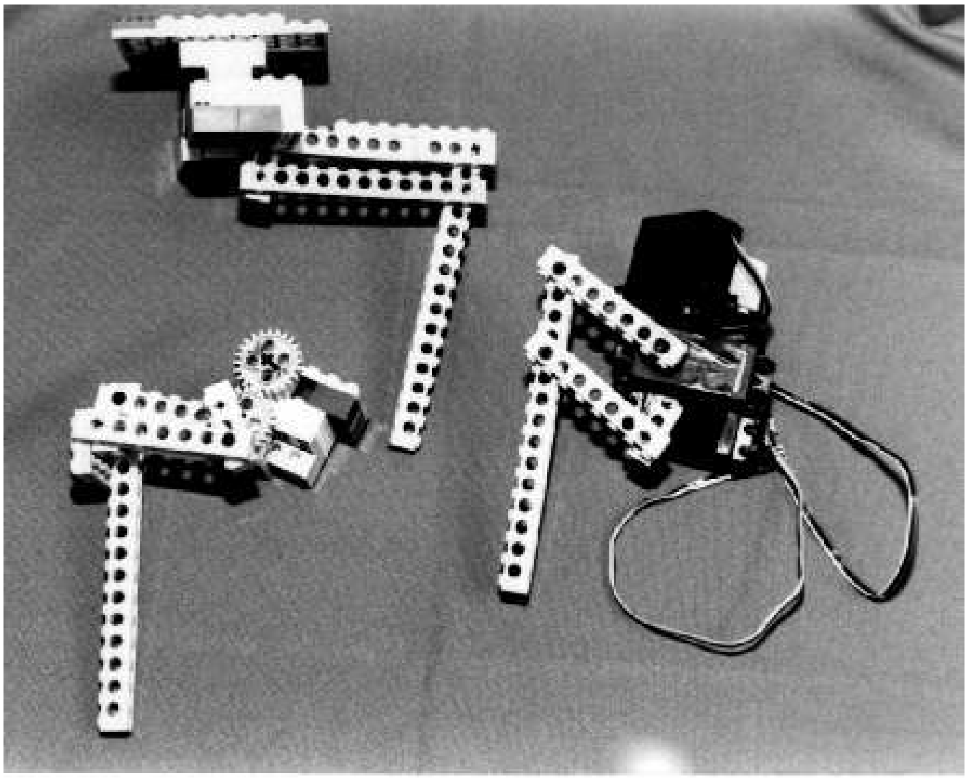


Figure 4.1: Three Lego prototypes. Leg on right (equipped with model airplane servos) shows the chosen geometry.

The design of robotic legs has been well studied (Hirose 84), (Raibert 86), (Angle 91). There are many issues that must be given attention. The following subsections will deal with: degrees-of-freedom per leg, actuator type, efficiency, workspace, kinematics, and manufacturability. These will be followed by a discussion of how the link lengths and actuators were chosen. The complete set of technical drawings for the legs can be found in appendix E.

4.1 Degrees-of-Freedom per Leg

In choosing the number of DOF (degrees of freedom) per leg for Thing, there was an apparent trade-off between the increased kinematic range of motion obtained with more DOF and the decreased cost and weight obtained with less. The advantages of having more than three DOF do not out-weight the additional hardware because any position in space can be reached by a three DOF robot (except for some special cases). What about two DOF? It is worthwhile to investigate whether the particular application of a walking robot can utilize two DOF as opposed to three DOF.

With two DOF, there is potentially less hardware to worry about and one less actuator that has to be brought into the control loop. This is a simplification with obvious design and fabrication advantages. With two DOF, however, it becomes less straight forward how to reach the *fixed goal* of arbitrary navigation because in some configurations turning and walking straight become an exclusive-OR pair (depending on the configuration, it can either turn or go straight, but not both). In the remaining two DOF configurations, the feet must be allowed to slip on the ground. This is unacceptable if position control is being used because the position of the robot becomes unknown when the feet slip. One example of the geometric problem is illustrated in fig 4.2. Part "a" shows one of numerous possible 2 DOF leg configurations, and part "b" shows a typical four legged robot configuration. The large arrow indicates the movement of the body, while the small arrows show the relative movement of the legs to the body that might occur as a follow-up to a step. With this configuration, the motion shown in fig 4.2c as two successive snap-shots in time is geometrically impossible without either the front or the back feet slipping.

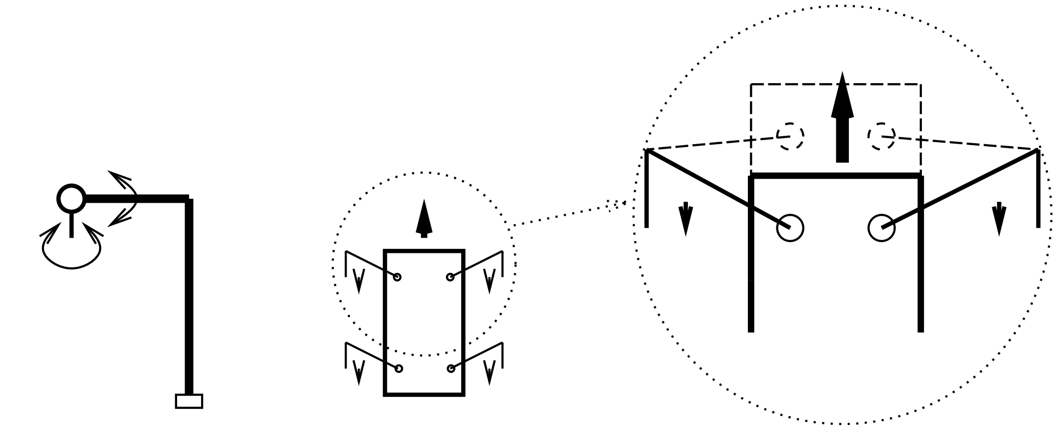


Figure 4.2: A problem with a two DOF leg.

There are many different leg configurations possible, and an exhaustive search was not conducted. It is postulated, however, that the "exclusive-OR" principle outlined above holds, and therefore, a simple 2 DOF leg is not acceptable for this design.

Conceptualizing the problem another way, it would seem necessary for a leg to have three orthogonal planes of motion for the foot to have arbitrary placement in three-space. Since such arbitrary foot placement is necessary for traversing rough terrain, two DOF is again deemed unacceptable.

There are some creative ways to try to get around the problems associated with 2 DOF legs such as having a joint in the body of the robot or having additional passive spring-loaded joints in the legs. The most straightforward configuration, however, is with three DOF per leg. The protracted design and fabrication times for the more esoteric designs out-weigh their benefits.

The whole reason for trying to eliminate one of the degrees of freedom in the first place was to simplify the control hardware. Having chosen to use the third actuator, other decisions were made to alleviate the control problems like the decision to use easily controllable servo packages and sophisticated microcontroller electronics.

4.2 Actuators

The three most common types of actuators in robotics are pneumatics, hydraulics and electric motors. While the first two types have some advantages like the backdrivability and speed of pneumatics and the strength obtained from hydraulics, both suffer from one big disadvantage–size. The actuators themselves can be made small, however the servo-driven valves tend to be large and heavy. Usually, the valves and compressor or storage tank are remotely located from the actuators. These characteristics limit the miniaturization of an autonomous system.

There has not been a demanding market for complete hydraulic or pneumatic systems that are small and light weight. In contrast, there is a large market for small DC motor systems due to consumer products like portable tape and CD players and remote control models that has created a relatively large and inexpensive selection. By choosing the correct combination of transmission and motor, acceptable speed and torque characteristics can be obtained. It was decided early in the project to use model airplane servos for the design. They are inexpensive, compact and come in complete packages. The decision to use such packages was made prior to the geometric design of the leg, while the choice of which type of model airplane servo was left until after link lengths had been determined and the required torques could be calculated. The torques and servo choices are discussed in sections 4.8 and 4.9.

4.3 Efficiency

One measure of the efficiency of a robot's legs is the amount of energy required to stand and walk. There are two important determinants of a leg's efficiency: (1) the axis in which each actuator functions, and (2) the lengths and inertias of the leg links.

4.3.1 Actuator Axis Planes

There are two key considerations in the determination of the axis planes (Hirose 84): (1) the isolation of the motions of each actuator from one another, and (2) the support of the robot against gravity in its isometric and active modes. Fig 4.3a shows a particular leg configuration. Here, the axis of rotation for both joints is horizontal (out of the plane of the paper). With this configuration joint 2 must actively support the lower link against the motion of the the upper link in order for the leg to push the robot forward. Additionally, in this posture both joints must be actively supplied with torque in order for the robot to remain standing (assuming neither link is perpendicular to the ground). The efficiency of this leg is poor as it requires supporting torques in multiple joints to make a step and continuous torques to remain standing. In an electrical system torque requires electrical current which drains the batteries. Hence, minimizing torque requirements lengthens the life-per-charge of the batteries. Note, high gear ratios in the servos at each joint can reduce the power consumption. This solution reduces the speed of the robot and may limit the backdriveability.

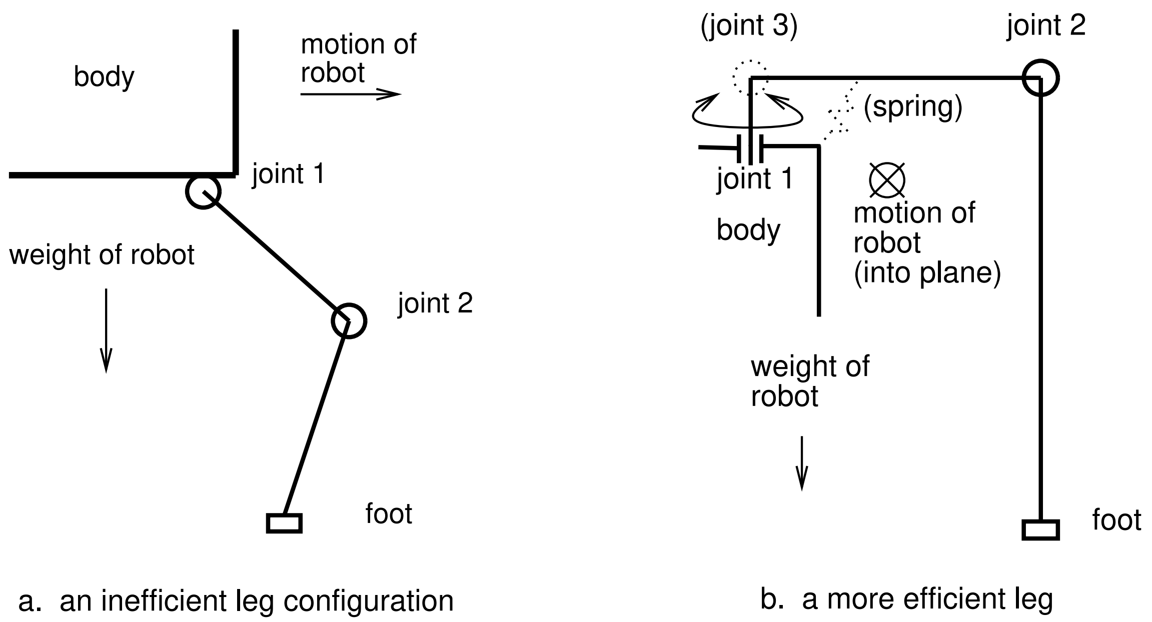


Figure 4.3: Comparing actuator axis planes in leg configurations.

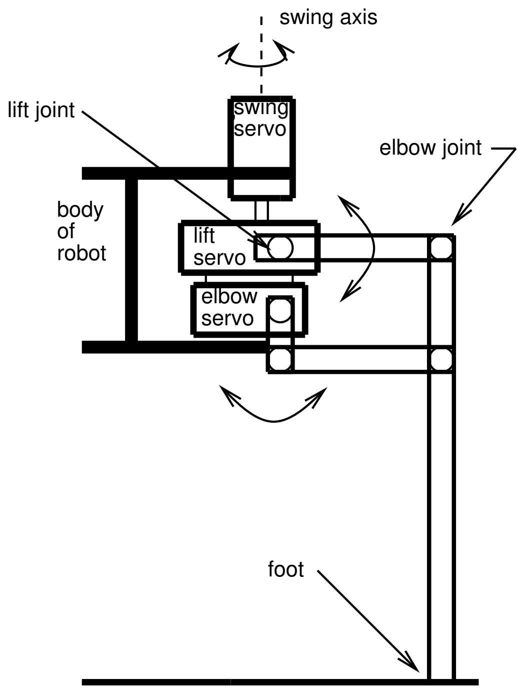
Fig 4.3b shows another possible configuration. Here the first joint is vertical, and the second is horizontal. The motion of the robot is into the plane of the paper. With this configuration, joint 2 is isolated from joint 1 when the robot is pushed forward. The bearings in joint 2 passively support the lower link, and no motor torque is required. Even more dramatically, neither joint is actively supporting the weight of the robot. The bearings are passively holding up the body. (Note, it is assumed that joint 2 is at a 90^o angle, as shown in fig 4.3b, most of the time when standing and walking.) This is clearly an improvement over fig 4.3a.

The configuration used for Thing was that of fig 4.3b with an additional joint added as shown by the dotted circle. The addition of a third joint was necessary to meet the 3 DOF requirement, however it decreases the efficiency of the leg. Now, joint 2 and joint 3 (dotted) are coupled kinematically, and joint 3 must support the weight of the robot. A passive solution exists, however, to eliminate some of the second inefficiency (Angle 91). Fixed torsion springs can be added at joint three to compensate for the weight of the body.

If each spring holds up one fourth of the weight, then, theoretically, joint three requires no torque to hold the robot up. This will be beneficial if most of the legs spend most of their time on the ground since now the joint requires additional torque to lift the leg off of the ground. One could visualize a gallop in which such spring usage would be detrimental; however, Thing is designed to walk in a statically stable gait, which means three legs will be on the ground at all times.

4.3.2 Link Lengths and Inertias

The leg efficiency is improved with shorter moment arms (which lead to lower torque requirements) and lower inertia in the moving parts of the leg. An example of legs with good and poor inertias is shown in fig 4.4. The leg in fig 4.4b has poor inertia because the elbow servo, a heavy component, is removed from the swing axis. The leg in fig 4.4a has all of its heavy components centered around the swing axis and therefore has better inertia characteristics. Note, this figure also defines the various parts of the leg. These terms will be used throughout this thesis.



a. Good inertia

Figure 4.4: A comparison of leg inertias.

4.4 Workspace

It is desirable to maximize the reachable workspace of the foot. This requires maximizing the legs range of motion while minimizing the self collisions among actuators and structure.

The range of motion of the leg is generally increased by larger allowable joint angles and longer link lengths. The required space for the motors, gears, and moving linkages is minimized by increasing the density of all these parts (ie. making them all fit closer together). A high parts density also supports a low leg inertia for better efficiency. (Longer link lengths, however, decrease efficiency.) A clear example of improved workspace is seen by the simple inversion shown in fig 4.5. By putting the swing servo on top, out of the way of the leg, the clearance below the robot is dramatically improved. Also, notice that in the second configuration, the elbow joint is lowered, resulting in a shorter radius of the vertical link as it rotates about the elbow joint. This in turn results in a shorter arc swept by the foot for a given elbow movement. These are examples of the numerous decisions that were made to improve the workspace. The leg geometry shown in fig 4.4a and 4.5a is that which was chosen for Thing. It has a good parts density with all of its servos mounted tightly around the swing axis, and it has a good reachable workspace. The specific link lengths will be discussed in section 4.7.

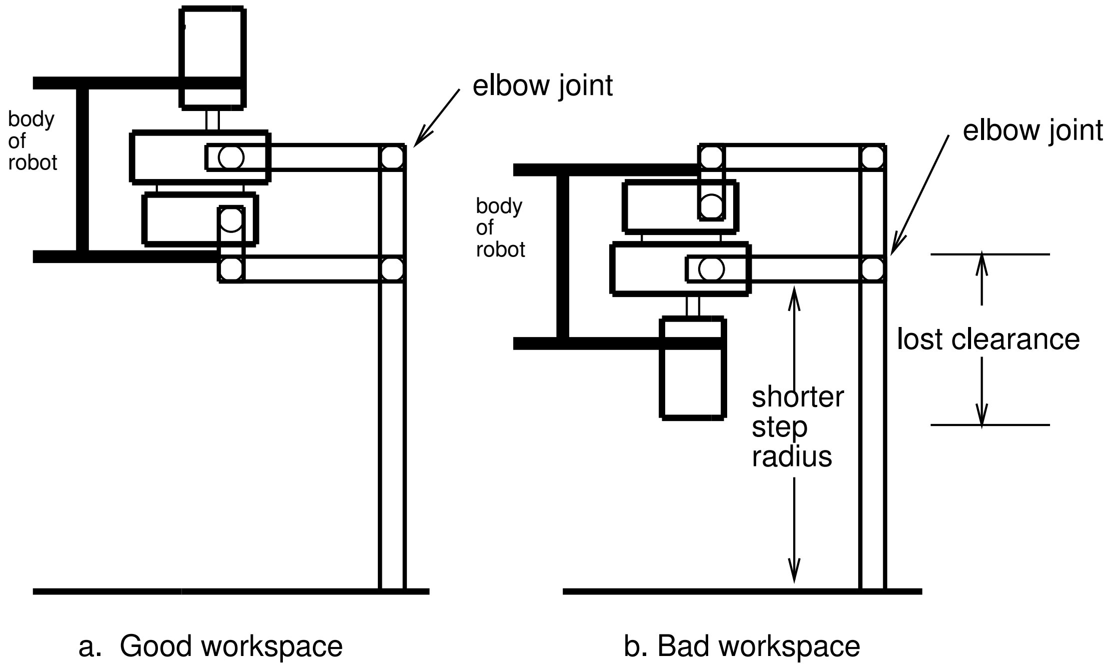


Figure 4.5: An example of improved workspace

4.5 Kinematic Considerations

The leg kinematics are the translations and rotations that describe the motion of the leg through three-space with respect to known joint angles. In order to move the foot to a particular (x,y,z) location, the relationship between a position of the servos and the corresponding position of the foot must be known. This is a typical problem in robotics. It can be very complicated (or impossible) to find a closed-form solution (to the inverse kinematics in particular) in many cases. Once the kinematics of the robot have been determined, if they are overly complicated then the robot can get bogged down solving complex equations just to make a simple movement.

The issue of concern here is the location of the joint angle sensors on the leg. Each servo has a built in position sensor that is part of the feedback loop for the servo. It would be foolish not to make use of such a convenience. The problem is that the best location for the servo is not necessarily the best location for the position sensor. For example, if the leg had the configuration of fig 4.4b, then the kinematics of the leg would be fairly simple because the angles at the lift and elbow joints could be read directly from the servo's position sensor. However, in the case of fig 4.4a, it is not at all clear how to derive the elbow joint angle from the elbow servo's position sensor since the leg is a parallel bar mechanism only when the elbow servo link is vertical. As the elbow servo rotates in either direction, the end of the lower link that is closest to the elbow servo rises faster than the other end while the upper link remains level, causing the two links to fall out of parallel. A non-parallel four-bar mechanism is harder to solve than a parallel one.

Despite the above discussion, the benefits of this later design outweighed its more complicated kinematics, and it was used. Attempts to ease the complexity with variations in the design inevitably led to fabrication complications. One example of an alternative design was to use a gear link that made the axis of the elbow servo coincide with the axis of the lift servo, making the linkage a parallel bar mechanism. This design maintained the advantageously short link length of the elbow servo by allowing for a gear reduction. The design and construction, however, of such a leg is much more involved than the chosen design, and therefore, this alternative (and others) was dismissed. The forward and inverse kinematics for Thing's legs are derived in chapter 7.

4.6 Manufacturability

Manufacturability is the ease with which the leg can be fabricated. This is partially dependent on the available equipment and can affect the construction time and cost of the robot. In the case of Thing, it was determined that all parts must be able to be cut on a milling machine from aluminum stock or, in the case of bearings, gears or axle shafts, ordered from a distributer.

This issue is not as straightforward as the others, because often one does not know how difficult it is to make something until it has been designed and an attempt has been made to create it. Complicated linkages like the gear mechanism described in section 4.5 were avoided. Intuitive decisions and educated guesses were made for initial designs, and redesign in the machine shop was a common occurrence with Thing.

4.7 Link Lengths

Starting with the geometric design determined in sections 4.1-4.6, reasonable guesses for the extreme angles of each joint were made. The lift joint has a maximum down angle of $\alpha = 40^o$ and a maximum up angle of $\alpha' = 60^o$. The elbow joint has a maximum down angle of $\beta = 30^o$ and a maximum up angle of $\beta' = 20^o$. These extreme angles are determined by the geometry of each leg link, the joints, and the rotational capabilities of the servos and are shown in fig 4.7. The main limit in the joint angles is due to the elbow link. There is an important reason, however, for why the rotation of the elbow servo was limited to that shown in fig 4.7. A geometric singularity occurs when the leg is down and the elbow is open as shown in fig 4.6. A singularity complicates the kinematics of the leg and is generally avoided. To avoid the occurrence, the servo is set so that it cannot go past the point shown in figure 4.6. This, unfortunately, limits the range of the elbow in other (non-singularity) situations as shown by the dotted lines.

The link lengths directly affect the torque requirements of the motors. Since it was determined that model airplane servos would be used, the torque range available was limited. Initial link lengths were chosen and then modified to accommodate the actual servos used. By making the initial leg design flexible in this way, the required torque specifications were flexible. This made it easier to choose the best servos in a number of different categories (like weight, price, quality, etc.) rather than settle on lesser servos that fit the exact torque requirements.

The first-guess link lengths were base on the desired step height, walking speed, workspace,

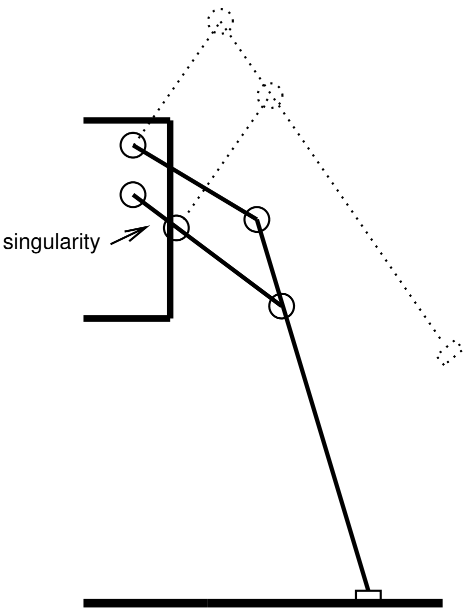


Figure 4.6: Occurrence of a singularity in the chosen leg design.

and approximate range of servo torques. The final link lengths (after the servos were picked) were chosen to be:

$$\begin{aligned} \text{L1} &= 2.0 \ \text{ in} \\ \text{L2} &= 6.0 \ \text{ in} \end{aligned}$$

One of the *flexible goals* was for the robot to have a 4 in maximum step height. Fig 4.7 shows how this is measured and calculated based on the joint angle limits. One leg is up, making a high step, while the other three legs lift the body as high as possible.

$$\begin{aligned} \textit{step height} &= 2.0\sin(\alpha) + 6.0 - 6.0\cos(90 - (90 - \alpha') - \beta') + 2.0\sin(\alpha') \qquad (4.1) \\ &= 4.4 \ \textit{in} \end{aligned}$$

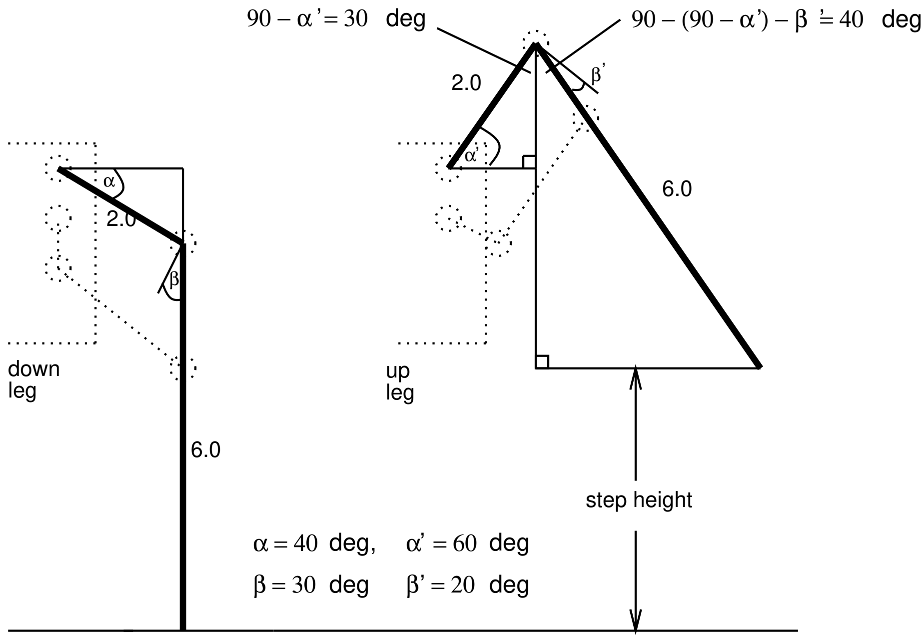


Figure 4.7: Calculating the step height.

4.8 Servo Design

Before motors can be selected the torque and weight requirements must be determined. The initial goal for the combined weight of the motors (slightly different than table 3.1) was 13.9 oz, which corresponds to a weight of approximately 1.2 oz each (since there will be twelve servos).

There are three actuated degrees of freedom. The servos that drive each of these were defined in fig 4.4 as the *swing*, *lift*, and *elbow* servos. The swing servo propels the robot forward by rotating the leg backward and then swinging it forward for another step. The lift servo supports the weight of the robot. And the elbow servo moves the elbow joint of the leg. The torque requirement for each must be calculated. The following is a summary of the goals and assumptions pertinent to these calculations. The assumptions were made to simplify the design analysis.

Pertinent goals from chapter 2

- robot able to climb 30^o slope
- top walking speed is 0.75 in/s
- top acceleration is 0.6 in/s^2 (on flat ground)

Results from chapters 3 and 4 and assumptions

- total weight of robot is 71.3 oz
- upper crossbar length (L1) is 2.0 in
- torsion springs are used on the lift joints
- at least 3 legs are on the ground at all times
- upper crossbar is horizontal most of the time during operation
- vertical link is perpendicular to the ground most of the time during operation

4.8.1 Swing Servo

When the robot is ascending a slope, its weight is partially transferred from the lift servos to the swing servos. The amount of weight transferred is determined as shown in fig 4.8. In most cases a safety factor is used in the torque calculations. This is embodied in assuming only two legs are on the ground at a time instead of three. The result is a safety factor of 50%.

$$\begin{aligned} W' &= W \sin(slope) \\ &= 35.7\ oz \end{aligned} \tag{4.2}$$

$$\begin{aligned} \text{torque to support weight} &= \frac{(crossbar\ length)(W')}{\#\ \text{legs on ground}} \\ &= 35.7\ in - oz \end{aligned} \tag{4.3}$$

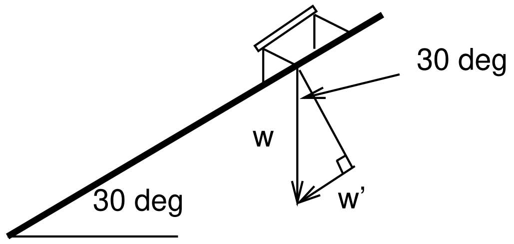


Figure 4.8: Transfer of weight during slope ascension

The necessary torque to meet the acceleration requirement can be found as follows:

$$\begin{aligned} total\ mass &= \frac{total\ weight}{acceleration\ of\ gravity} \\ &= 0.185\ oz\ s^2/in \end{aligned} \tag{4.4}$$

$$\begin{aligned} \text{torque for acceleration} &= \frac{(crossbar\ length)(acceleration)(total\ mass)}{\#\text{ legs on ground}} \\ &= 0.111\ oz\ in \end{aligned} \tag{4.5}$$

The acceleration torque is apparently an easy specification to meet and can be ignored if the weight support torque is met.

The speed in the servo is determined by first calculating the distance traveled in one step which is actually defined to be the distance traveled by the robot after every leg has taken a step. The step length is determined as shown in fig 4.9. The swing angle is assumed to be 45^o forward and 45^o back, and the vertical leg link is assumed to be perpendicular to the ground (as in fig 4.5a). The step length is

$$d = 2L1\sin(45) = 2.8\ in$$

A step corresponds to 1/4 revolution (90^o) of the swing servo, and each leg must complete its respective step for the robot body to have traveled the step distance, d. Hence, the required servo speed (in rev/sec) is found as follows:

$$
\begin{aligned}
time\ for\ step &= \frac{d}{4(desired\ speed)} \\
&= 0.93\ sec
\end{aligned}
\tag{4.6}
$$

$$
\begin{aligned}
servo\ speed &= \frac{servo\ revolutions\ per\ step}{time\ for\ step} \\
&= 0.27\ rev/s
\end{aligned}
\tag{4.7}
$$

Note, this is an approximation. A full analysis of the speed of the robot and the required servo speeds would require further consideration of the kinematic configuration of the leg. Since the speed of the robot was a $flexiblegoal$, further analysis was not conducted.

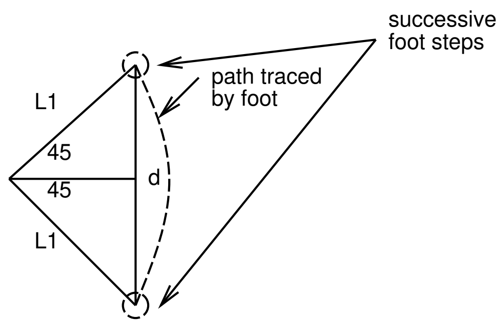


Figure 4.9: Determining the step length

4.8.2 Lift Servo

The most torque will be required when the leg is fully extended and in the down position as shown in figure 4.10. Due to the singularity avoidance (discussed in section 4.7), the elbow joint is limited to approximately 30^o. A torsion spring of 40 oz-in is assumed to be in place at the lift joint, and no safety factor is incorporated since a larger spring can be used if necessary and since the actual torque will be less due to the geometry of the elbow linkage. (Note, a spring of approximately 48 oz-in would completely neutralize the weight of the robot when the vertical link is perpendicular to the ground.) The moment arm and required torque are

$$M = 2.0 + 6.0\sin(20) = 4.0\ in$$

$$\begin{aligned} lift\ torque &= \frac{(M)(total\ weight)}{\#\ legs\ on\ ground} - spring\ torque \\ &= 55.1\ oz\ in \end{aligned} \tag{4.8}$$

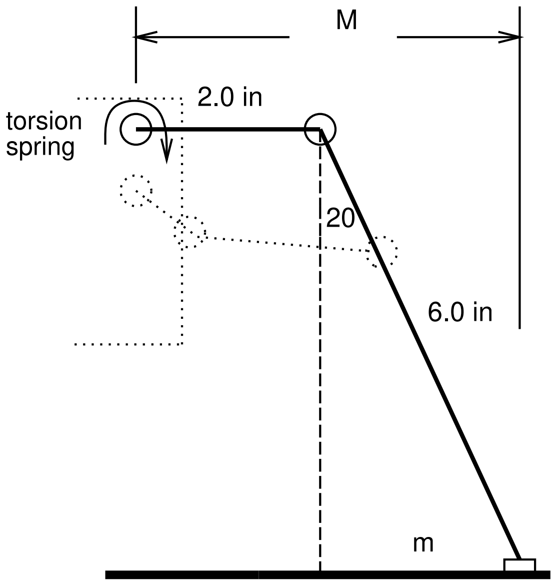


Figure 4.10: Determining the torque in the lift servo

4.8.3 Elbow Servo

The elbow torque can be calculated by finding the torque at the actual elbow joint. The safety factor *is* included in this calculation. From fig 4.10, the moment arm, m, and the torque can be found to be

$$m = 6.0\sin(20) = 2.0\ in \tag{4.9}$$

joint	torque	speed	weight
lift	55.1 oz-in	–	1.2 oz
swing	35.7 oz-in	0.27 rev/s	1.2 oz
elbow	35.7 oz-in	–	1.2 oz

Table 4.1: Summary of required servo specifications.

$$\begin{aligned} elbow\ torque &= \frac{(m)(total\ weight)}{\#\ legs\ on\ ground} \\ &= 71.3\ oz\ in \end{aligned} \tag{4.10}$$

This is a relatively large torque for a secondary servo. There are numerous ways of dealing with this large torque at the elbow joint including using a heavy gear reduction in the servo (at the expense of speed) or changing the geometry or actuator type. The method used was to give the elbow servo an advantageous moment arm with which to compensate for a lower torque rating. This is shown in fig 4.11. It is a simple kinematic deduction similar to that obtained with a gear reduction. The following analysis assumes that the vertical link is perpendicular to the ground most of the time during operation, as shown by the solid line. With this assumption, the new required torque rating for the lift servo is easy to approximate.

The mechanism is such that the elbow servo drives a link of half the length of the distance from the elbow joint to the lower cross link, cutting the required torque in half. Hence, the elbow servo must be rated to $71.3/2 = 35.7$ oz-in. A summary of the required servo specifications is given in table 4.1.

4.9 Servo Selection

There is a large selection of motors and gear-heads available on the market, and there are many different types of motor/gear-head combinations that meet the required torque ratings. However, when other requirements, such as light weight, small size, and low cost, are included, the selection becomes almost zero. The biggest factor is the cost. As motors

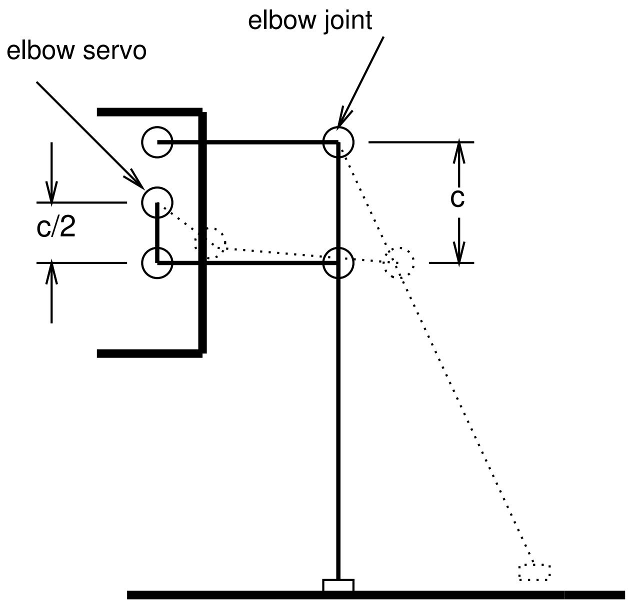


Figure 4.11: The elbow servo mechanism

are miniaturized, their cost almost invariably goes up.

Fortunately, a convenient product exists that has been around for many years. It is the model airplane servo, a completely closed-loop mechanism with gear train, position sensor, PWM (pulse width modulation) referencing, and motor all contained in a single small inexpensive package. These servos come in a variety of configurations, a number of which fit the requirements set out for Thing. The servos used are given in table 4.2 along with their pertinent specifications. Note, all of the required ratings are met except the weight of the lift servo. As discussed in section 3.5, the weight profile was modified to include the heavier servos. The last entry in the table gives an example of a motor/gear-head combination from MicroMo Inc. that meets all of the requirements except the cost which is more than two times higher and is therefore unacceptable. This last entry is representative of the available motors outside the model airplane market. They surely are of better quality, which is the main compromise; however, they require separate position sensing and servo control thereby adding design and construction time to the overall project.

The use of model airplane servos in miniature robots has been popular among hobbyists

joint	type	torque	speed	weight	cost
lift	Futaba S9201	69.5 oz-in	0.76 rev/s	1.7 oz	$41.00
swing	Futaba S9601	36.1 oz-in	0.98 rev/s	1.1 oz	$42.00
elbow	Futaba S9601	36.1 oz-in	0.98 rev/s	1.1 oz	$42.00
—	MicroMo 1319-15/5	42.4 oz-in	0.88 rev/s	1.18 oz	$125.00

Table 4.2: Summary of servos

and hackers for many years and was taken a step beyond into a serious research application in the implementation of the hexapod robot named Genghis (Angle 89) that is used as a research platform at MIT.

The Futaba servos that were used for Thing have ball bearings, nylon gears (the S9601 has one metal gear), and coreless motors. Their PWM input reference makes them directly interfacable to a microcontroller equipped with a programmable timer.

Chapter 5

Body

The important considerations in body design were:

- Stability: the legs must be configured so that the platform can stand stably.
- Symmetry: if the robot body is geometrically symmetric in the plane of motion, then it can walk in any direction with an identical gait.
- Repairability: the robot should be easy to work on.
- Rigidity: the structure should not sag or twist under its own weight.
- Overlap of leg workspace: consideration should be given to the interference of one leg with another.

An early issue was whether or not the swing servo could support the leg with its own bearings as shown in fig 5.1a. It was found empirically that it could not. The nylon shaft and the plastic servo housing flexed as shown by the dashed lines in fig 5.1a. For this reason, a lower support bearing was used to eliminate the movement perpendicular to the axis of rotation as shown in fig 5.1b. The required additional support was unfortunate because it added bulk and weight to the robot, although it did make the entire robot more rigid.

The body design that was chosen is shown in fig 5.2. The dotted lines show the workspace of each leg. Overlap of the workspaces was intentionally allowed. It was not in the intentions

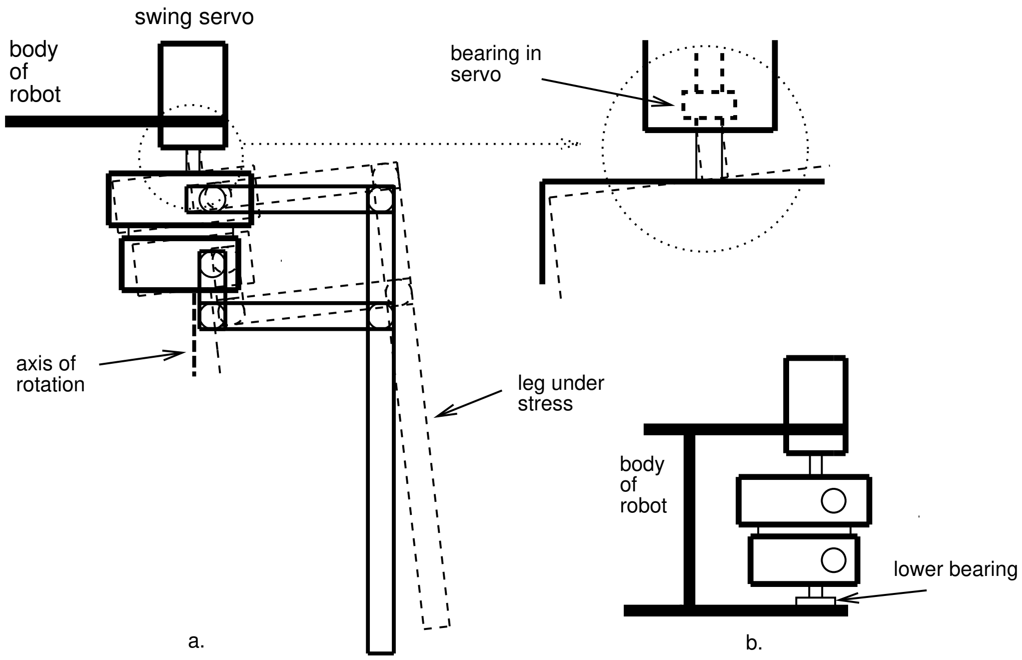


Figure 5.1: Swing servo supporting leg

of this project to commit to hardware solutions to control problems, which is one of the things that keeping the workspaces of the legs separate would accomplish. We elected instead to preserve the flexibility of the hardware and to study methods of coordinating the behavior of multiple legs. The legs were situated as close as possible to one another to allow the robot to act as a simple hand which was one of the design goals. In the final design, the feet come to a point approximately two inches from one another, allowing it to easily grasp an object the size of a grapefruit.

The horizontal view in fig 5.2b shows the structure which is made up of three parallel planes linked by four vertical columns. This configuration is very rigid. All parts are milled aluminum. To lighten the structure, the four columns were bored out axially, the bottom two 1/8 inch thick planes were milled out inside as shown by the dashed lines in fig 5.2a, and the top plane was made 1/16 inch thick. The complete technical drawings can be found in appendix E.

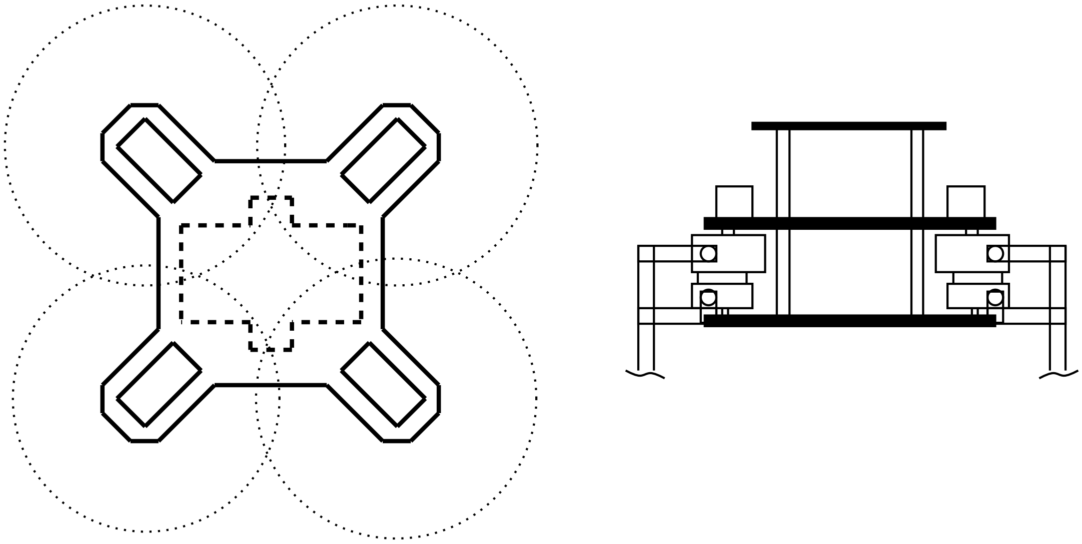


Figure 5.2: Thing's body

The circuit boards for each of the satellite processors were mounted on hinges so that they could be easily serviced as shown in fig 5.3. All of the legs and boards can be removed in less than twenty minutes with an allen wrench and screw driver.

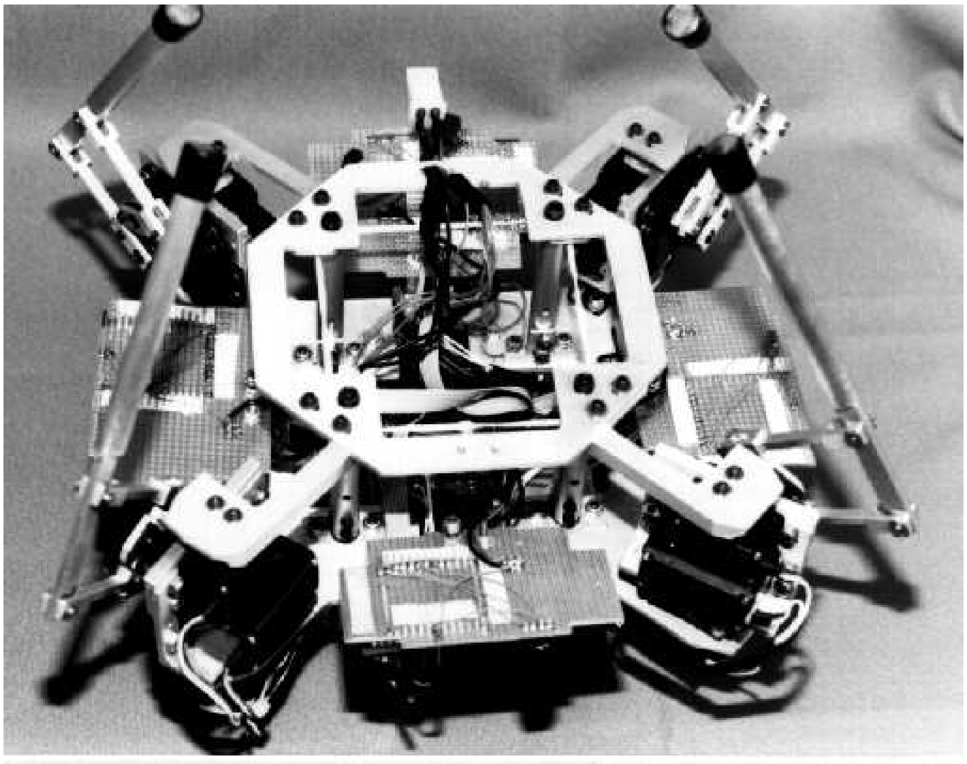


Figure 5.3: Hinged leg processor boards ease servicing the robot.

Chapter 6

Electronics

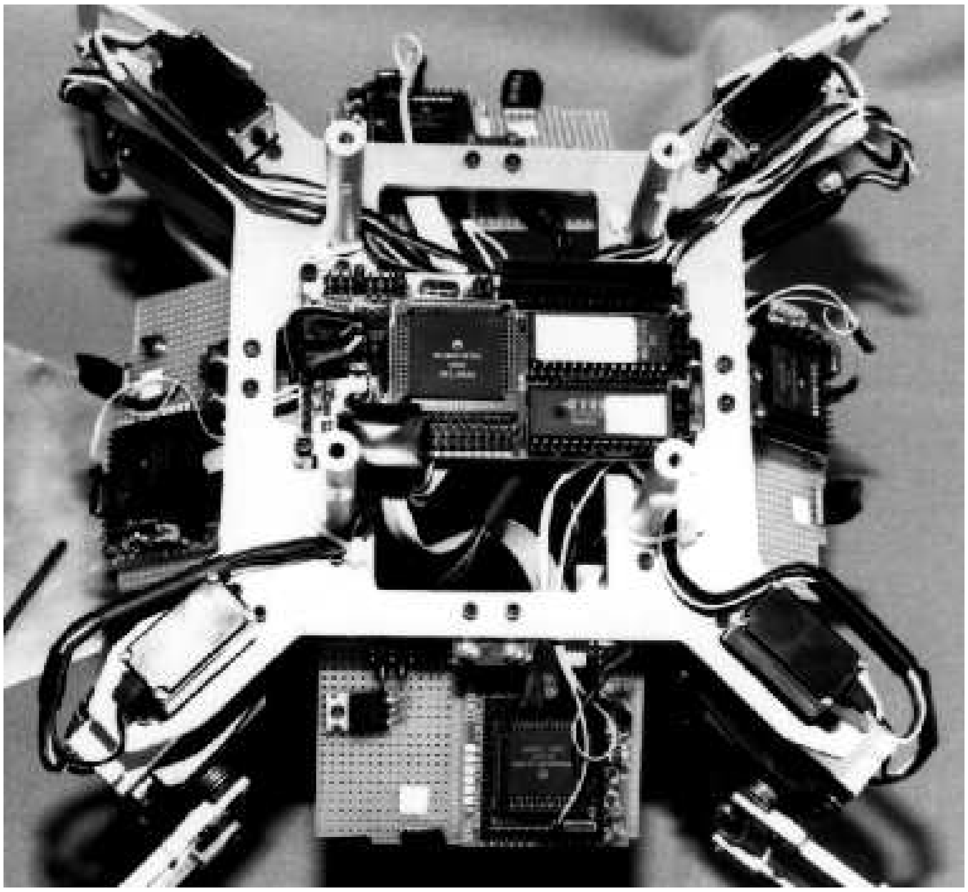


Figure 6.1: Thing runs with five HC11 microcontrollers.

In determining the type of electronics to use, the following issues were important:

- cost of hardware
- time required for design, construction, debug, and, coding
- computational ability to handle the control problem
- physical size of hardware

6.1 Processing Electronics

Having chosen to use a leg design that has fairly complicated kinematics as outlined in section 4.5, the question remained of how to deal with this complexity.

6.1.1 Real-time vs Pre-calculation

One important computation issue was whether or not the forward and inverse kinematics of the robot would be calculated in real-time or taken from "lookup tables" of pre-calculated data. Real-time calculations take up processor time, leaving less for other important tasks like global navigation or sensor analysis, but are very accurate. Lookup tables take up large amounts of memory and are not precise representations, but they require relatively less computation and hence less processor time.

If a look up table is used, then accuracy of the data is directly affected by the iteration step size in the data. Decreasing the step size (to improve accuracy), however, increases the size of the table. Since the table must be a three dimensional array (one dimension for each joint), a linear decrease in the step size translates to a cubic increase in the memory required to store the table.

The accuracy that would be required if Thing were to use such a lookup table is unknown until further work is conducted with the robot. For this reason, an attempt was made both to provide enough memory to implement a table and, at the same time, prepare for real-time computation.

6.1.2 Distributed System

A common way to control a legged robot is to tell each leg how to move in terms of joint references. With a single processor the serial controller that sequentially controls each leg might look like that shown in fig 6.2a. Another way–one which is particularly proficient at dealing with large amounts of task-oriented computation–is to distribute the tasks to different processors and then fuse the results together when the tasks are complete. With multiple processors, the legs could be moved simultaneously with each leg controlled by its own processor as shown in fig 6.2b. If such an architecture can be achieved within a reasonable price range, this would clearly be a superior method of control since the central processor, #5, could spend its time with global calculations, and the other satellite processors could compute movements of the legs. Moreover, it is generally accepted that humans and other animals have distributed intelligence and controllers.

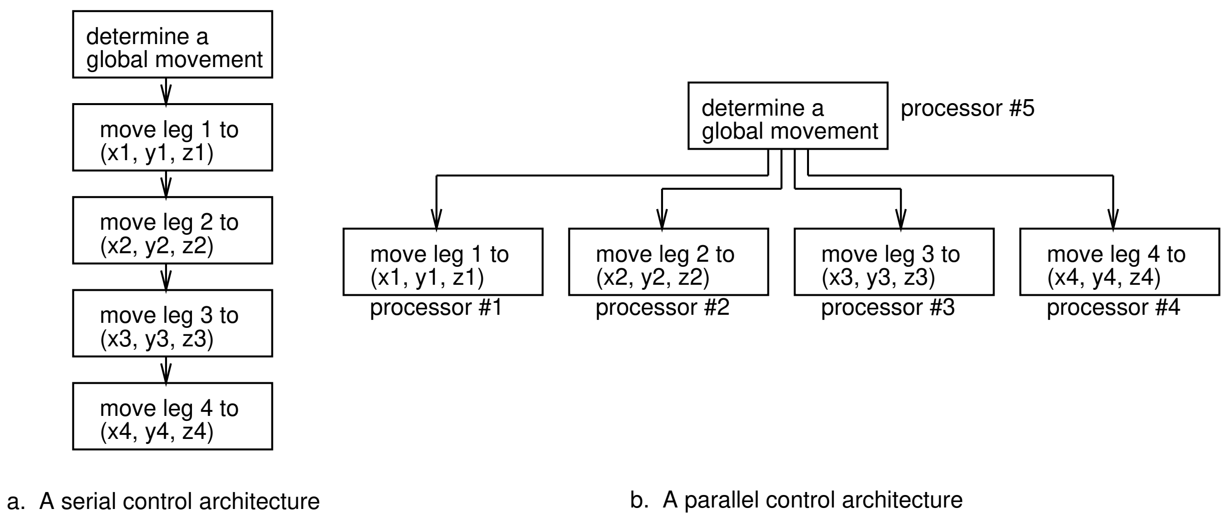


Figure 6.2: Serial vs parallel control architectures

6.1.3 Processors

As with the high production model airplane servo, there exist microcontrollers that are functionally complete and are inexpensive. Also developing popularity among hobbyists and hackers, the Motorola MC68HC11 family of microcontrollers (see appendix C for specifications) is the perfect solution. Its low price and built-in serial communication ports and memory make it ideal for a distributed system.

The microcontroller used for the legs was the MC68HC811E2. Each processor has 2k bytes of EEPROM, 256 bytes of RAM, 8 A/D converters, programable timer, and synchronous and asynchronous serial ports. These functions allow the single microcontroller to process resident code, control the servos in a leg, input and process sensor data for a leg, and communicate with the central processor via a serial link–all without any exterior hardware.

The microcontroller used for the central processor was the MC68HC11F1. This processor is essentially the same as the MC68HC811E2, but it has an expanded address bus, allowing it to be easily interfaced with external memory. Utilizing this feature, the central processor has 32K bytes of RAM and 32K bytes of EPROM at its disposal. This allows the central processor to run the high-level navigational code.

The complete system is shown in fig 6.3, and a detailed schematic of Thing's entire electrical system is given in appendix A.

6.2 Trouble-Shooting Considerations

The task of coordinating five different processors can be complicated. Designing and constructing a computer system (let alone a system of computer systems) can require a lot of trouble-shooting time. Hours can be spent searching for a weak solder connection on a homemade circuit board. It is for this reason that a rule of using prefabricated (and pre-tested) circuit boards was adhered to whenever possible within the limits of the budget.

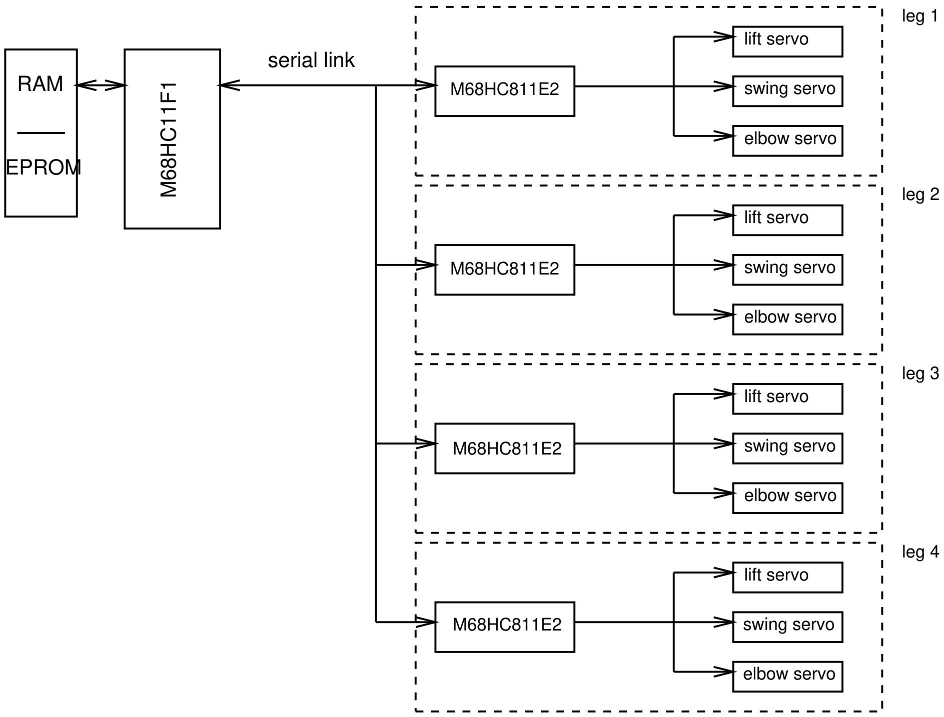


Figure 6.3: Schematic of Thing's processor system

For the central processor, a New Micros NMIS-0024 v3.0 board was used. It is small and simple, supplying 64K bytes of external RAM/ROM, reset circuitry, and pin-outs to connectors. This board has all of the functions necessary (except an RS-232 serial port) to stand alone. The central board is shown in fig 6.4, and the schematic is given in appendix A.

For the leg processors, CGN 1001 wire-wrap modules were used. The module has reset circuitry and pin-outs to connectors. It was soldered onto a larger perforated board to allow for local sensor conditioning circuitry and future expansion. A leg processor board is shown in fig 6.5, and a schematic is given in appendix A.

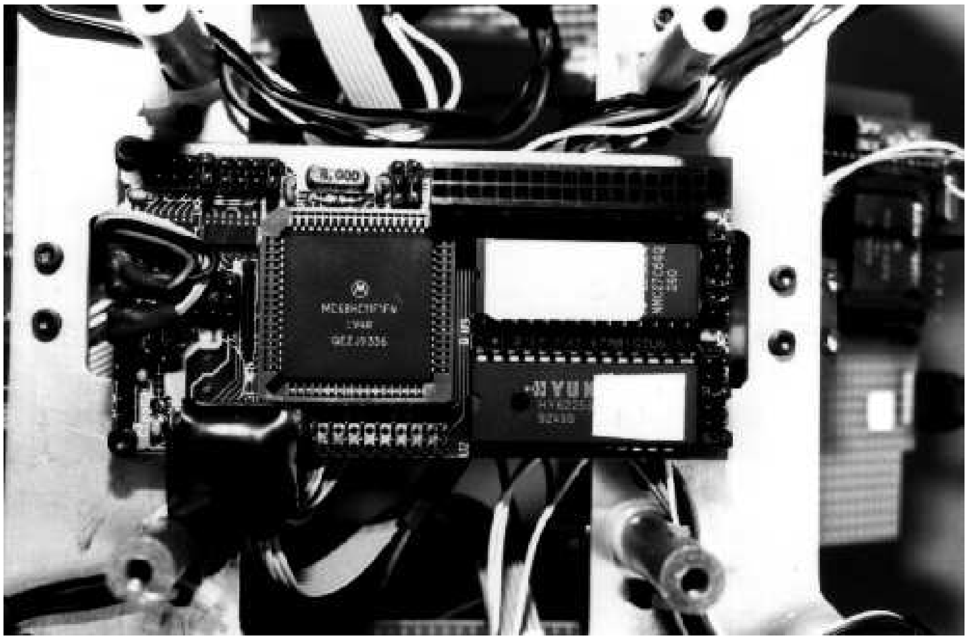


Figure 6.4: The New Micros board mounted on Thing.

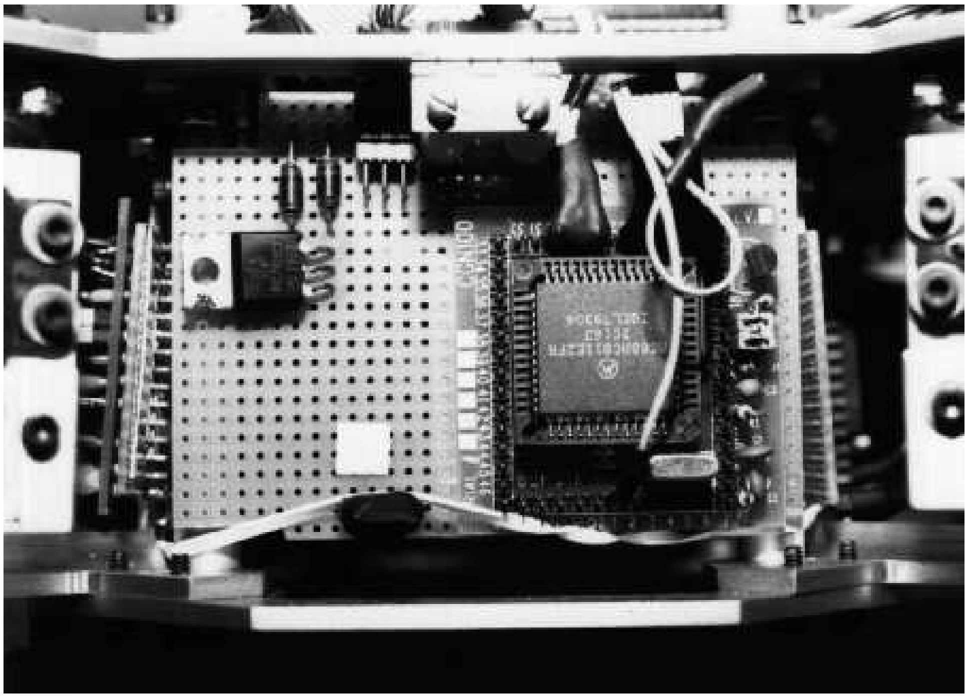


Figure 6.5: The CGN board mounted on Thing.

6.3 Servo Control

The PWM servo reference signals are generated using the programmable timer on each leg processor. The signal is a pulse repeated every 20ms. The neutral pulse width, corresponding to a central positioning of the servo, is approximately 1.3ms, while the extreme rotations of the servo are obtained by pulse widths of 0.7ms and 1.7ms. The code for configuring the timer can be found in appendix B or (Motorola 91).

6.4 Power

The power supplies for the electronics and for the servo motors must be isolated from one another. The reason is two-fold: 1) the reset circuitry on the processor boards is very sensitive to the supply voltage. If the voltage drops even slightly bellow 5 volts, it will reset. Since the motors create a very large current draw (up to 2 amps), the supply voltage cannot be expected to remain stable. 2) The motors create inductance spikes that can not only trigger the reset circuitry, but burn out the CMOS electronics.

One solution is to use a single source with voltage regulators and isolation capacitors. While the capacitors are a good idea, large voltage regulators eat up too much power.

A simple solution is to use two separate supplies (battery packs). The grounds still have to be common since the servos require a common ground, and so isolation capacitors will still be necessary. However, a large voltage regulator would not be necessary (a small one may still be used for the electronics battery pack).

6.4.1 Batteries

Based on the total design weight of 71.3 oz, the battery requirements were found empirically. The lift servo requires approximately 0.25 amps to lift 24 oz (1/3 of 71.3) with a 2 inch moment arm. And, the swing servo requires approximately 0.25 amps to move a mass that simulates 1/3 of the total inertia of the robot forward as would be required for a typical step. The total instantaneous current draw is then

$$I = 3(0.25\ amps + 0.25\ amps) = 1.5\ amps$$

If the robot is to walk autonomously for 30 minutes, as was the goal, then the battery must be rated for

$$E = (1500\ mA)(0.5\ hrs) = 750\ mAhs$$

There are many batteries available that meet this rating and additionally have a weight acceptable to the design specifications of 14.0 oz total. One example is the Hydramax GMH-1700A, which is a 1.2 volt, 1700 mAh cell that weighs 1.16 oz. The voltage required by the servos is 4.8 volts, hence at least four cells would be required, giving a total weight of 4.64 oz. This leaves 9.63 oz for the electronics battery pack which is far more than enough. Final selection of batteries is left for future work.

6.5 Sensors

There are two natural ways for a walking robot to navigate (Klein, Olson, and Pugh 83). The first is by means of high level sensing such as that obtained by a vision system. This is similar to how humans navigate but tends to be very computationally expensive as analyzing a video image for recognizable objects and choosing a suitable path requires complex algorithms. Such vision processing has become a concentrated area of study in and of itself.

The second way is for the robot to blindly feel its way over and around obstacles by means of contact sensors on the legs and body of the robot. This is more like the way a spider navigates. The scale difference in nature between humans and spiders and their respective navigation systems should lend insight into the type of system suitable for a small scale robot. It has been argued (Angle 91), in fact, that tactile sensing is the human being's most fundamental sensing mechanism. Thing was designed to implement this lower-level sensor scheme. Its size and computational ability were the deciding factors.

Chapter 7

Kinematics

The *forward* kinematics are the translations from the known angles of the servos to the (x,y,z) coordinates of the foot. The *inverse* kinematics are the opposite–the translations from the known (or desired) (x,y,z) coordinates of the foot to the (required) angles of the servos.

7.1 Forward Kinematics

The coordinate transformation from the frame of reference of the leg's origin on the robot to the frame of reference of the foot is described by the following homogeneous transform. The angle conventions and variables are defined in fig 7.1. Note, χ is in the xy plane, and the lift axis has an offset indicated by the fixed lengths "d", "m", and "h".

$$
{}_O T_F = \begin{vmatrix} \mathrm{C}(\chi) & -\mathrm{S}(\chi)\mathrm{C}(\theta_1-\gamma) & \mathrm{S}(\chi)\mathrm{S}(\theta_1-\gamma) & -\mathrm{S}(\chi)[l_2\mathrm{S}(\theta_1-\gamma)+l_1\mathrm{S}(\theta_1)+m]-d\mathrm{C}(\chi) \\ \mathrm{S}(\chi) & \mathrm{C}(\chi)\mathrm{C}(\theta_1-\gamma) & -\mathrm{C}(\chi)\mathrm{S}(\theta_1-\gamma) & \mathrm{C}(\chi)[l_2\mathrm{S}(\theta_1-\gamma)+l_1\mathrm{S}(\theta_1)+m]-d\mathrm{S}(\chi) \\ 0 & \mathrm{S}(\theta_1-\gamma) & \mathrm{C}(\theta_1-\gamma) & -l_2\mathrm{C}(\theta_1-\gamma)-l_1\mathrm{C}(\theta_1)-h \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{7.1}
$$

where "S( )" means sin( ), and "C( )" means cos( ).

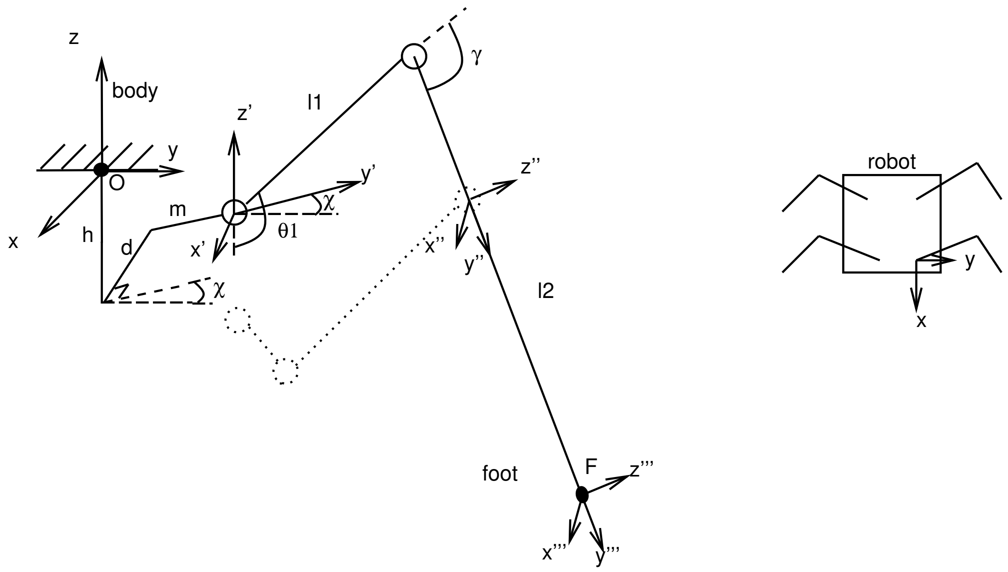


Figure 7.1: 3 DOF leg and body orientation

The complete homogeneous transform is useful in high-level control, but for the basic movement of the leg's joints, it is more convenient to break-up the transformation into step-by-step calculations that can be easily adapted to low-level subroutines. Also, the rotational translations may not be needed in the low-level control. The steps are as follows. Fig. 7.2 shows the angles and lengths used in the calculations. Note, the coordinate system (y',z') has its origin at the lift joint and therefore rotates with χ. From the law of cosines and sines,

$$
\begin{aligned}
r_1 &= \sqrt{a^2 + d^2 - 2ad\cos(\theta_1)} && (7.2)\\
\alpha_1 &= \sin^{-1}(\frac{d\sin(\theta_1)}{r_1}) && (7.3)\\
\alpha_3 &= \theta_1 - \theta_2 + \alpha_1 \\
r_2 &= \sqrt{r_1^2 + e^2 - 2er_1\cos(\alpha_3)} && (7.4)\\
\alpha_4 &= \sin^{-1}(\frac{e\sin(\alpha_3)}{r_2}) && (7.5)\\
\alpha_5 &= \cos^{-1}(\frac{r_2^2 + c^2 - a^2}{2r_2c})
\end{aligned}
$$

$$
\begin{aligned}
180 - \gamma &= \alpha_1 + \alpha_4 + \alpha_5 \\
r_3 &= \sqrt{a^2 + (b+c)^2 - 2a(b+c)\cos(180-\gamma)} && (7.6) \\
\alpha_6 &= \sin^{-1}\left(\frac{(b+c)\sin(180-\gamma)}{r_3}\right) && (7.7) \\
\alpha_7 &= \theta_1 - \alpha_6 \\
y' &= r_3 \sin(\alpha_7) \\
z' &= r_3 \cos(\alpha_7)
\end{aligned}
$$

Note that the pair of equations 7.2 and 7.3 is analogous to the pairs 7.4, 7.5 and 7.6, 7.7. These equations can therefore be combined into a single subroutine in code, minimizing the required memory.

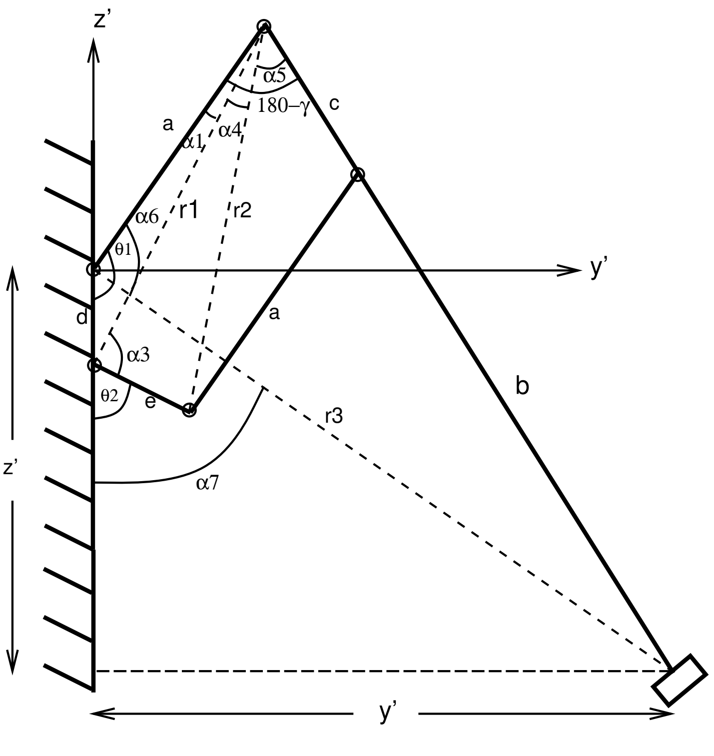


Figure 7.2: Looking down the -x' axis at the leg.

The angles and lengths used in the final steps of the forward kinematics are defined in fig 7.3 which shows a view of the leg looking down the -z axis (ie. looking down at the top of the robot). The thick line is the leg, and the short hash marks that cross it show the locations of the joints. The dot is the foot. Since there is an offset, "d," in the swing axis, the angle that the upper link is at will be different than χ by an amount δ. From the figure,

$$\begin{aligned} r &= \sqrt{(y' + m)^2 + d^2} \\ \delta &= \sin^{-1}(\frac{d}{r}) \\ x &= r\sin(\chi + \delta) \\ y &= r\cos(\chi + \delta) \end{aligned}$$

and from fig 7.1,

$$z = -z' - h$$

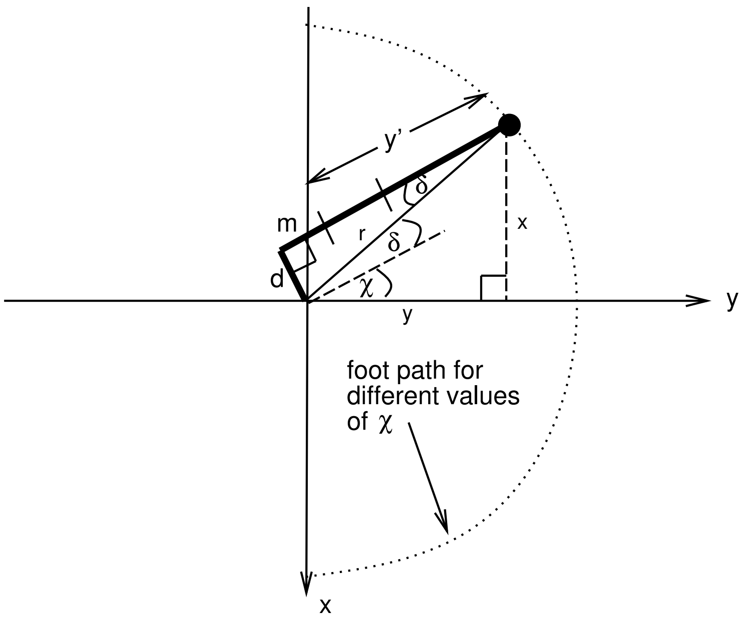


Figure 7.3: Looking down the -z axis at the leg.

A look-up table (given in appendix D) was constructed, using the above calculations, in which (x,y,z) coordinates of the foot are given for combinations of $\theta 1$, $\theta 2$ and χ.

7.2 Inverse Kinematics

A simple closed form of the inverse kinematics can be found since there is a one-to-one relationship between the servo angles and the coordinates of the foot (a result of eschewing singularities in the geometry). First find the swing angle, χ, as follows. From fig 7.3,

$$r = \sqrt{x^2 + y^2} \tag{7.8}$$

$$\chi = \sin^{-1}(\frac{-x}{r}) - \delta \tag{7.9}$$

where

$$\delta = \sin^{-1}(\frac{d}{r})$$

additionally,

$$y' = r\cos(\delta) - m$$

$$z' = -z - h$$

In fig 7.4, the upper cross link and the vertical link form a simple two link manipulator that is analogous to the two link manipulator formed by the elbow servo link and the lower cross link. This insight leads to the possibility of using a single subroutine to calculate the two angles, θ_1 and θ_2. First, the angles are found individually. Fig 7.4 defines the variables used in the calculations. To find the lift servo angle,

$$r_1 = \sqrt{z'^2 + y'^2} \tag{7.10}$$

$$\theta_1 = \cos^{-1}(\frac{a^2 + r_1^2 - (b+c)^2}{2ar_1}) + \cos^{-1}(\frac{z'}{r_1}) \tag{7.11}$$

Next, find the point (z",y"):

$$180 - \gamma = \cos^{-1}(\frac{a^2 + (b+c)^2 - r_1^2}{2a(b+c)})$$

$$
\begin{aligned}
z'' &= -a\cos(\theta_1) - c\cos(\gamma - \theta_1) \\
y'' &= a\sin(\theta_1) - c\sin(\gamma - \theta_1)
\end{aligned}
$$

Then, repeat the two-link manipulator calculation to find the elbow servo angle:

$$
r_2 = \sqrt{(z''+f)^2 + y''^2} \tag{7.12}
$$

$$
180 - \theta_2 = \cos^{-1}\left(\frac{e^2 + r_2^2 - a^2}{2er_2}\right) + \cos^{-1}\left(\frac{z''+f}{r_2}\right) \tag{7.13}
$$

Equations 7.10 and 7.11 can be made into a subroutine that will provide the same calculation as equations 7.12 and 7.13, decreasing the memory requirement for the code in the leg processors.

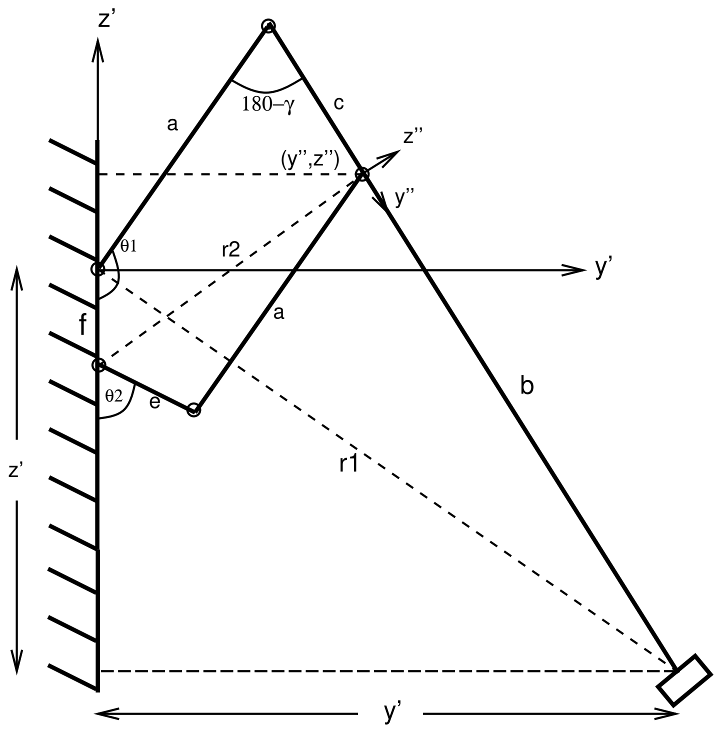


Figure 7.4: Looking down the -z' axis at the leg.

Alternatively, an inverse look-up table can be pre-calculated or constructed from the forward kinematics look-up table. Since there is a one-to-one relationship between the servo angles and the coordinates of the foot, the table can be easily constructed by traversing the existing table, looking for consecutive coordinates in each dimension.

Chapter 8

Robot Software

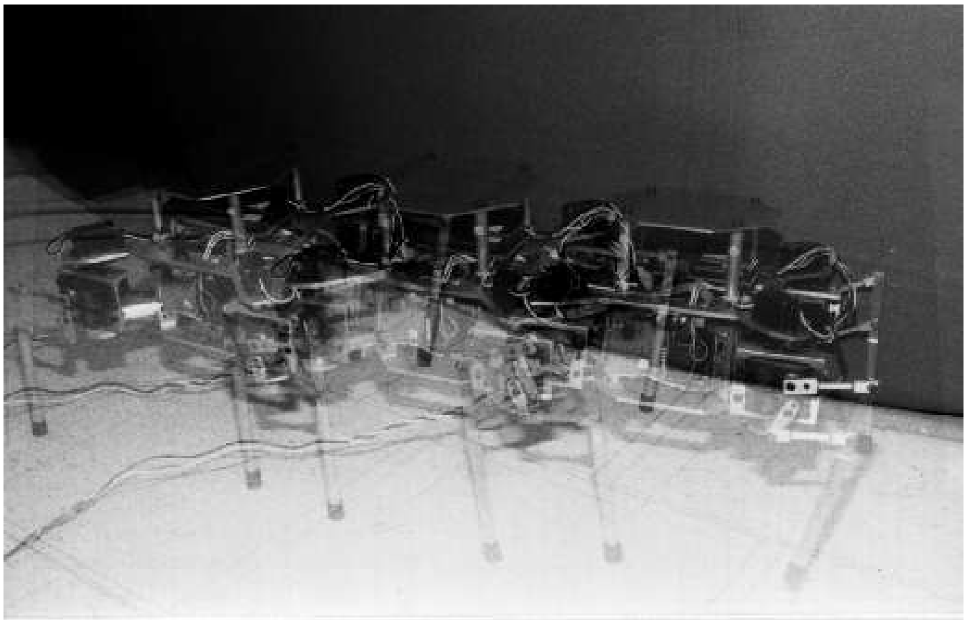


Figure 8.1: The demo displayed Thing walking.

8.1 A Distributed Control Architecture

The hardware is physically distributed, however it must be determined how the software will be distributed. There were three possibilities considered in the control of the robot. (1) The central processor could send each leg processor a reference joint configuration that specifies the position of each servo. This would require the central processor to solve the kinematics or use a look-up table for each leg. The leg processors would then compute the PWM servo control references and read the sensors. (2) The central processor could tell each leg processor where to locate its respective *foot*. This requires each leg processor to compute the joint angles given an (x,y,z) coordinate for its respective foot. It relieves the central processor of the task of computing the inverse kinematics; however, the code or look-up table for the kinematics has to be available to each leg processor and therefore must be redundantly resident in each processor's limited memory. (3) The central processor could tell each leg processor where it would like the *entire robot* to go. This would require each leg processor to determine where its respective foot should go in order to move the robot as a whole. It would then, of course, have to solve the kinematics (et cetera) to get the foot to that position. The central processor is left only with the task of determining the direction of locomotion.

Initially, the second control architecture was selected. It breaks up the control into reasonable tasks for each processor; although, it requires that the leg processors have enough memory to hold the code or look-up table for the inverse kinematics. The MC68HC811E2 has a total of 2.5K bytes of resident memory, which is marginally sufficient.

8.1.1 Leg Processor

A flow chart of the algorithm that the leg processors will run is shown in fig 8.2a. The MC68HC811E2 provides interrupt-driven timer and serial communication control. So, the PWM generation and the communication with the central processor are done in the background. The code can reside in non volatile EEPROM so that upon power up, the processor can begin execution completely separate from the central processor. It positions its leg in an initial posture and then waits for the central processor to give it further command. Alternatively, the central processor can down load a program to the leg processors at power up. This allows for easier code modification.

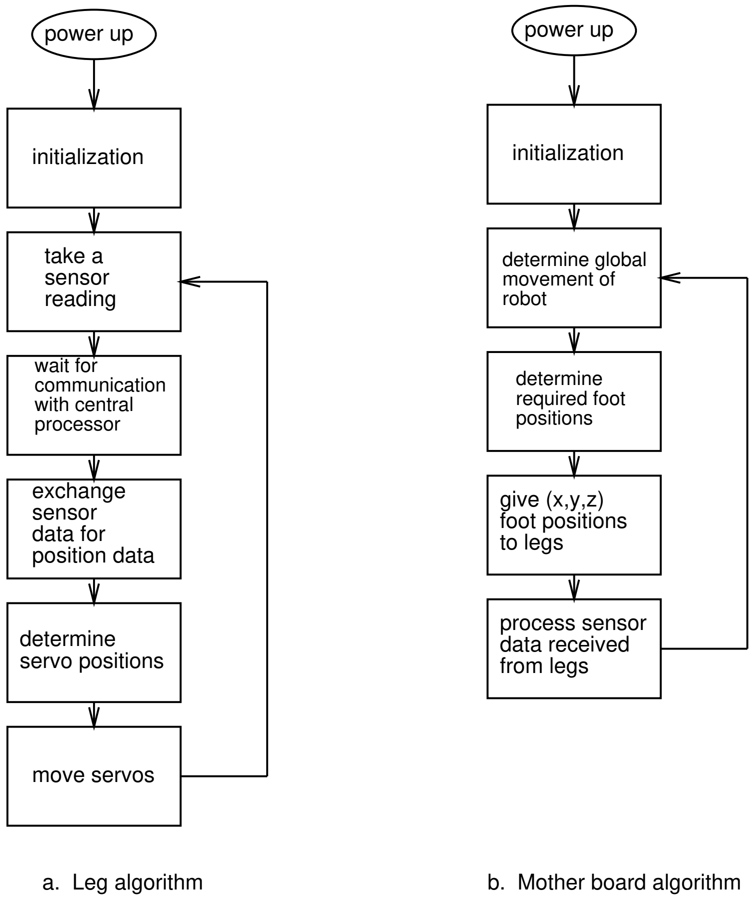


Figure 8.2: Processor algorithms

8.1.2 Central Processor

The central processor determines the direction that the robot should go and determines the necessary foot placements to get there. It then tells each leg where to put its respective foot and receives sensor data from the legs. The central board will run the algorithm shown in fig 8.2b. Note, the active stabilization and navigation of the robot based on sensor data will be studied in future work.

8.2 Solutions for Limited Memory Mathematics

The limited memory space on the leg processors disallows the residence of a complete math library on the chip. This poses a problem for calculating the kinematics on-line. A miniature math library, however, can be constructed as needed. For example, the determination of the swing angle, χ requires the following calculation (eqn 7.9):

$$\chi = \sin^{-1}(\frac{x}{r}) - \delta$$

While the MC68HC811E2 provides some complex functions like multiplication and division, the instruction set does not include an inverse sine function. But a power series approximation can be made and written as an assembly language subroutine. The power series for the inverse sine function is

$$\sin^{-1}(x) = x + \frac{x^3}{6} + \frac{3x^5}{40} + ...$$

Which can be easily written as an assembly language subroutine. Similar methods can be used for the sine, cosine, and inverse cosine functions as well.

A different method must be used to find the square root, say in solving for r (eqn 7.8), where

$$r = \sqrt{x^2 + y^2}$$

The square root function can not be expanded in a power series; however, Newton's method can be used to find an iterative solution. The $n + 1^{th}$ iteration for the solution to $y = \sqrt{x}$ can be expressed as

$$y_{n+1} = y_n - \frac{y_n^2 - (x)}{2y_n}$$

where the initial value, y_o can be assumed to be x. This method can also be easily written as an assembly language subroutine.

It may be advantageous to implement some or all of the math functions as look-up tables instead. The main issue is the amount of memory required.

8.3 A Demonstration of Functionality

In the final demonstration of Thing, there were no sensors in place, and the central processor simply followed a sequence of 12 different postures that, when put in sequence, resulted in a forward progression. The foot positions were sent to the correct leg processors which then positioned the servos. The purpose of the exhibit was to demonstrate the basic functioning of the hardware. The code used for the demo is given in appendix B.

Chapter 9

Conclusion

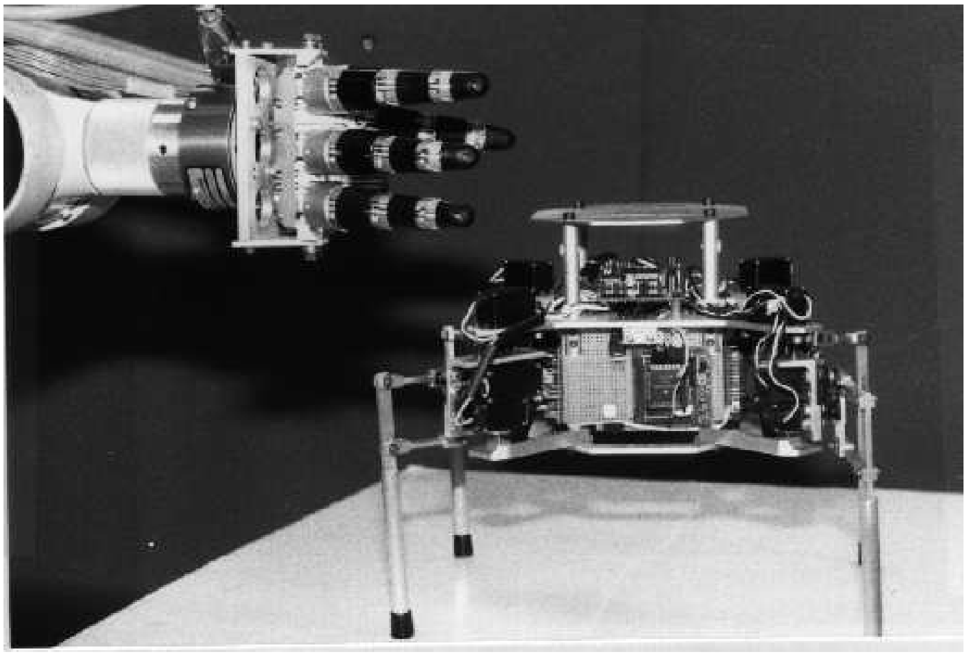


Figure 9.1: Thing with the Utah/MIT dexterous hand.

9.1 Summary

Thing was successfully demonstrated executing a four legged walking gait on May 16, 1994. This marked the completion of the hardware phase of the project and the work for the undergraduate research requirement. All of the *fixed goals* were met.

The cost and time constraints were heavy factors in guiding the project. They brought about a method of design and construction that worked well and is applicable to the design of any robot and to engineering in general. Both requirements led towards an ideally small size for the robot and worked in harmony with many of the other design goals. For example, having to keep things cheap and small lead to the use of packaged high production servos that made the design mechanically modular. The low cost of highly sophisticated microcontrollers led itself to a distributed as well as modular electronic system. The modularity allowed for an incremental construction of the system. Each leg could be constructed, wired, and tested separately. The robot can stand up on its own with each leg and its respective controller functioning independently. Coordinated movement is achieved by merely tying each leg to a serial network monitored by the central processor that gives positioning information to the leg processors.

A loose method of design and construction was used for Thing. The scale of Thing led itself to be constructed completely by the designer which in turn allowed for incremental design. Its small scale combined with its modularity and inexpensive materials (electronics, servos and metal) such that mistakes were acceptable and repairable, resulted in a solid working design in a very short time period.

9.2 Future Work

In the immediate future, the kinematics will be placed on-board the robot, and the sensor and battery systems will be put in place. Some developmental work has already begun on piezoelectric touch sensors that can be mounted on Thing's legs. The batteries will consist of two separate NiCd packs, one for the motors and one for the electronics and will allow the robot to walk autonomously.

Thing will act as a platform for expanding the application of the sensor-based controllers developed in the LPR and will serve as a starting point for research in legged robots in general.

References

[**Angle 1989**] "Genghis, a Six Legged Autonomous Walking Robot," Colin M. Angle. MIT S.B. Thesis in Electrical Engineering and Computer Science, March 1989.

[**Angle 1991**] "Design of an Artificial Creature," Colin M. Angle. Master's Thesis, MIT Electrical Engineering and Computer Science Department. June 1991.

[**Angle and Brooks 90**] "Small Planetary Rovers." Colin M. Angle and Rodney A. Brooks, MIT Artificial Intelligence Lab, April, 1990.

[**Bares 89**] "Ambler, an autonomous Rover for Planetary Exploration," John Bares, Martial Hebert, Takeo Kanade, Eric Krotkov, Tom Mitchell, Reid Simmons, and William Whittaker, *IEEE Computer*, June 1989.

[**Bekker 1969**] "Introduction to Terrain-Vehicle Systems," M. G. Bekker. Ann Arbor: University of Michigan Press, 1969.

[**Hirose 1984**] "A study of Design and Control of a Quadruped Walking Vehicle," Shigeo Hirose. *International Journal of Robotics Research*, 3:2, Summer 1984.

[**Hirose et al 84**] "Titan III: A Quadruped Walking Vehicle," S. Hirose, T. Masuri, and H. Kikchi, *Proc. 2nd ISRR Symp. Kyoto, Japan*, 247-253.

[**Hirose and Umetani 80**] "The Basic Motion Regulation System for a Quadruped Walking Vehicle," S. Hirose, and Y. Umetani. *American Society of Mechanical Engineers*, 80-DET-34.

[**Hodgins 88**] "Adjusting Step Length for Rough Terrain Locomotion," Jessica Hodgins, *Proceedings of IEEE conference on Robotics and Automation*, Philadelphia, 1988.

[**Klein, Olson and Pugh 1983**] "Use of Force and Attitude Sensors for Locomotion of a Legged Vehicle over Irregular Terrain," Charles A. Klein, Karl W. Olson, and Dennis R.

Pugh. *The International Journal of Robotics Research*, 2:2, summer 1983.

[**Motorola 91**] "M68HC11 Reference Manual," Motorola Inc., Phoenix, Arizona, 1991.

[**Paul 1979**] "Kinematics and Dynamics of Planar Machinery," Burton Paul. Prentice-Hall, Inc, Englewood Cliffs, N.J., 1979.

[**Pugh et al 90**] "Technical Description of the Adaptive Suspension Vehicle," D. Pugh, E. Ribble, V. Vohnout, T. Bihari, T. Walliser, M. Patterson, and K. Waldron. *International Journal of Robotics Research*, 9:2, April, 24-42.

[**Raibert 86**] "Legged Robots that Balance," Marc H. Raibert. MIT Press, Cambridge, MA, 1986.

[**Waldron et. al. 84**] "Configuration Design of the Adaptive Suspension Vehicle," Kenneth J. Waldron, Vincent J. Vohnout, Arrie Pery, and Robert B. McGhee, *The International Journal of Robotics Research*, 3:2, Summer 1984.

References Not Cited

Alexander, R. McN., "The Gaits of Bipedal and Quadrupedal Animals," The International Journal of Robotics Research, 3:2, Summer 1984.

Chapman, David, "How to do Research at the MIT AI Lab," MIT AI lab, 1988.

Ferrell, Cynthia, "Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators," M.S. Thesis, MIT Department of Electrical Engineering and Computer Science, 1993.

Gerald, Curtis F., and Wheatly, Patrick O., "Applied Numerical Analysis." Addison-Wesley, Reading, MA, 1994.

Jones, Joseph L., and Flynn, Anita M., "Mobile Robots: Inspiration to implementation."

A.K. Peters, Ltd., Wellesley, MA, 1993.

Lipovski, G.L., "Single- and Multiple-Chip Microcomputer Interfacing," Prentice-Hall, Inc, Englewood Cliffs, NJ, 1988.

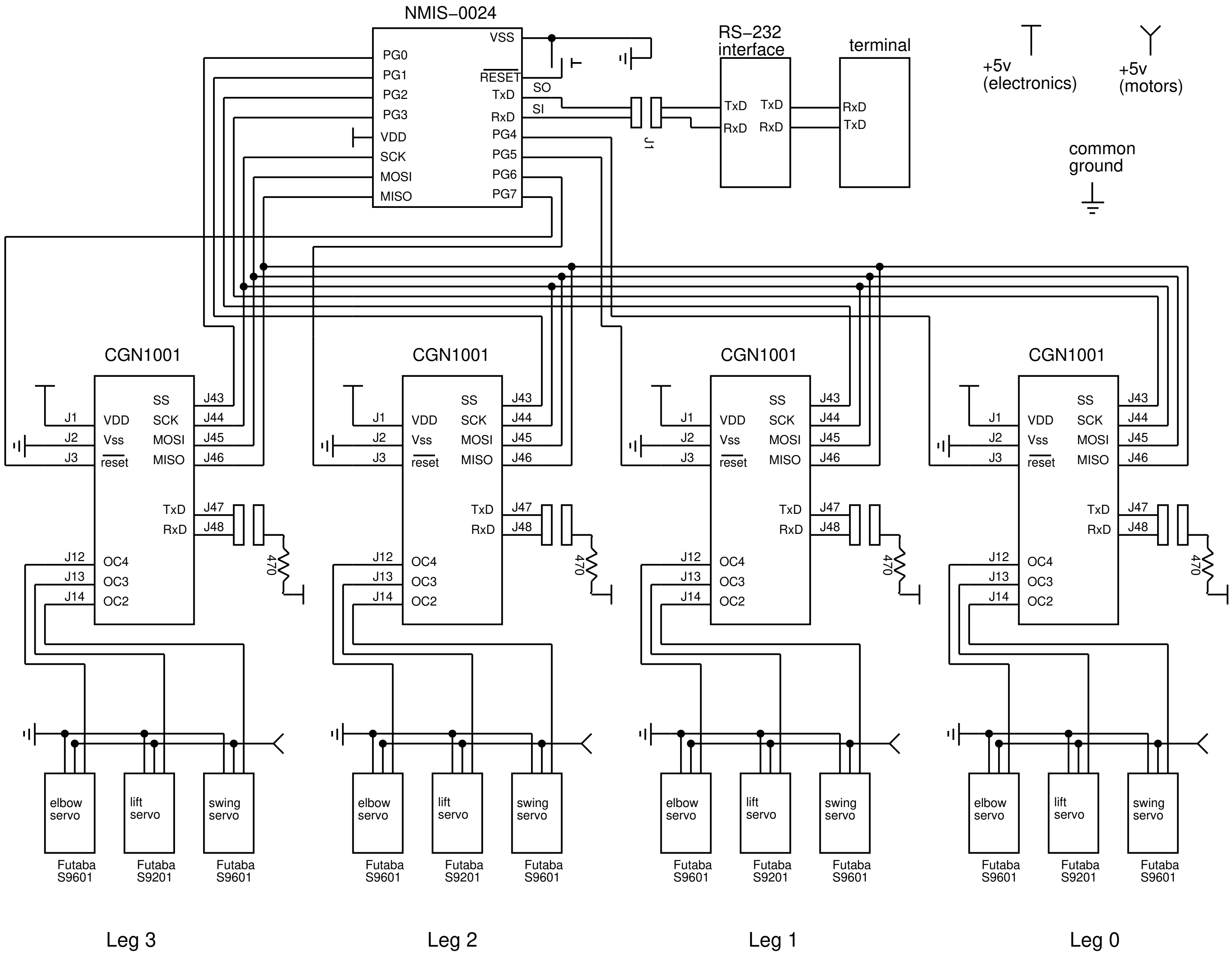
Spasov, Peter, "Microcontroller Technology," Regents/Prentice Hall, Englewood Cliffs, NJ, 1993.

Stallings, William, "Data and Computer Communications," Macmillan Publishing Co., New York, NY, 1985.

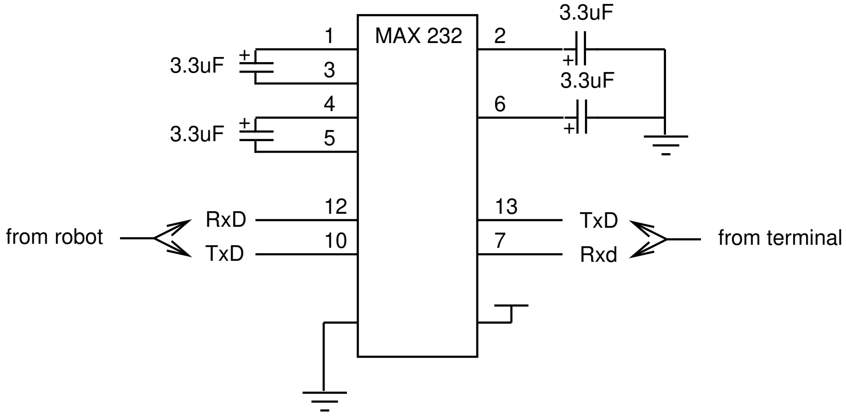
Appendix A

Electrical Schematics

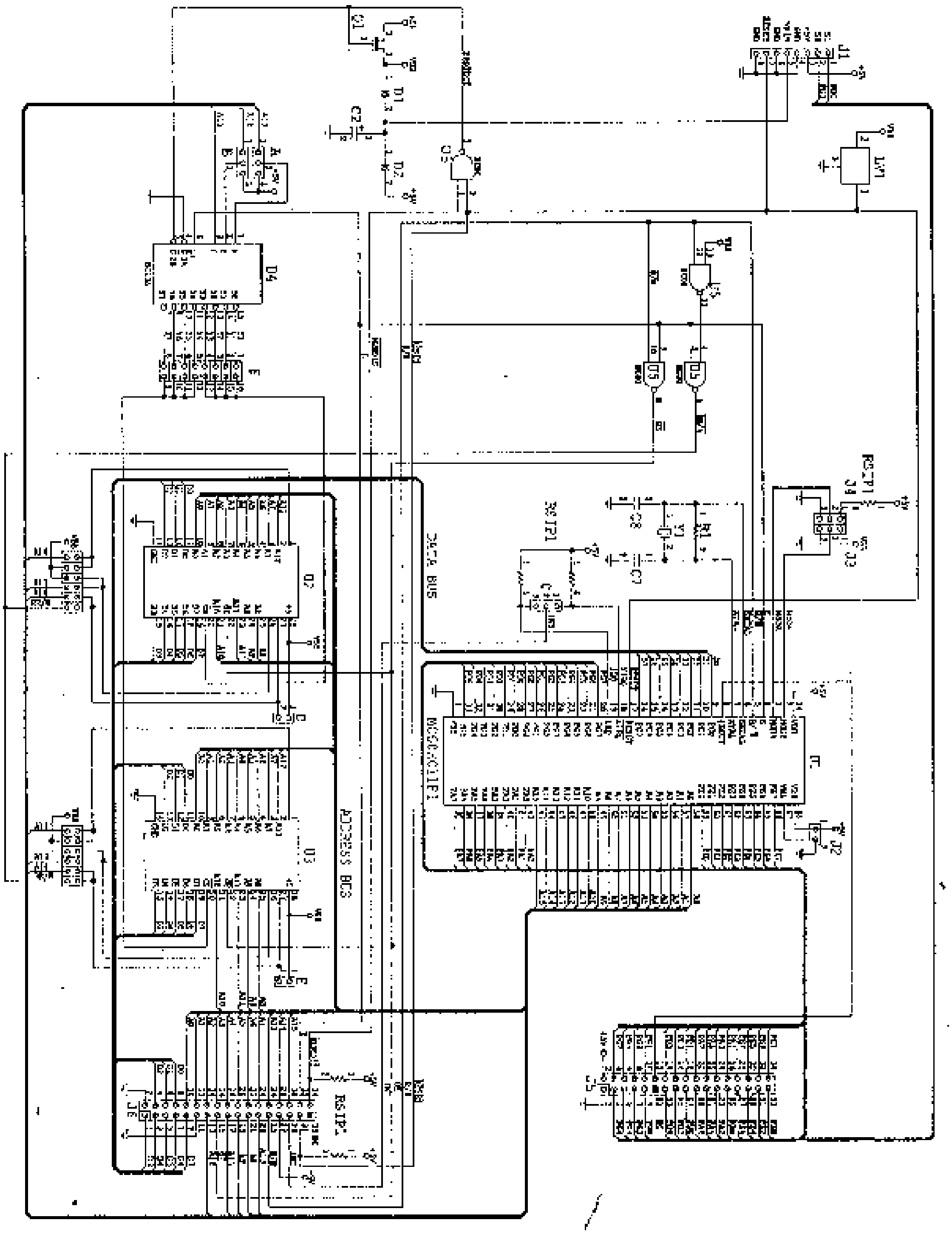
- System
- RS-232 Interface
- NMIS-0024
- CGN1001

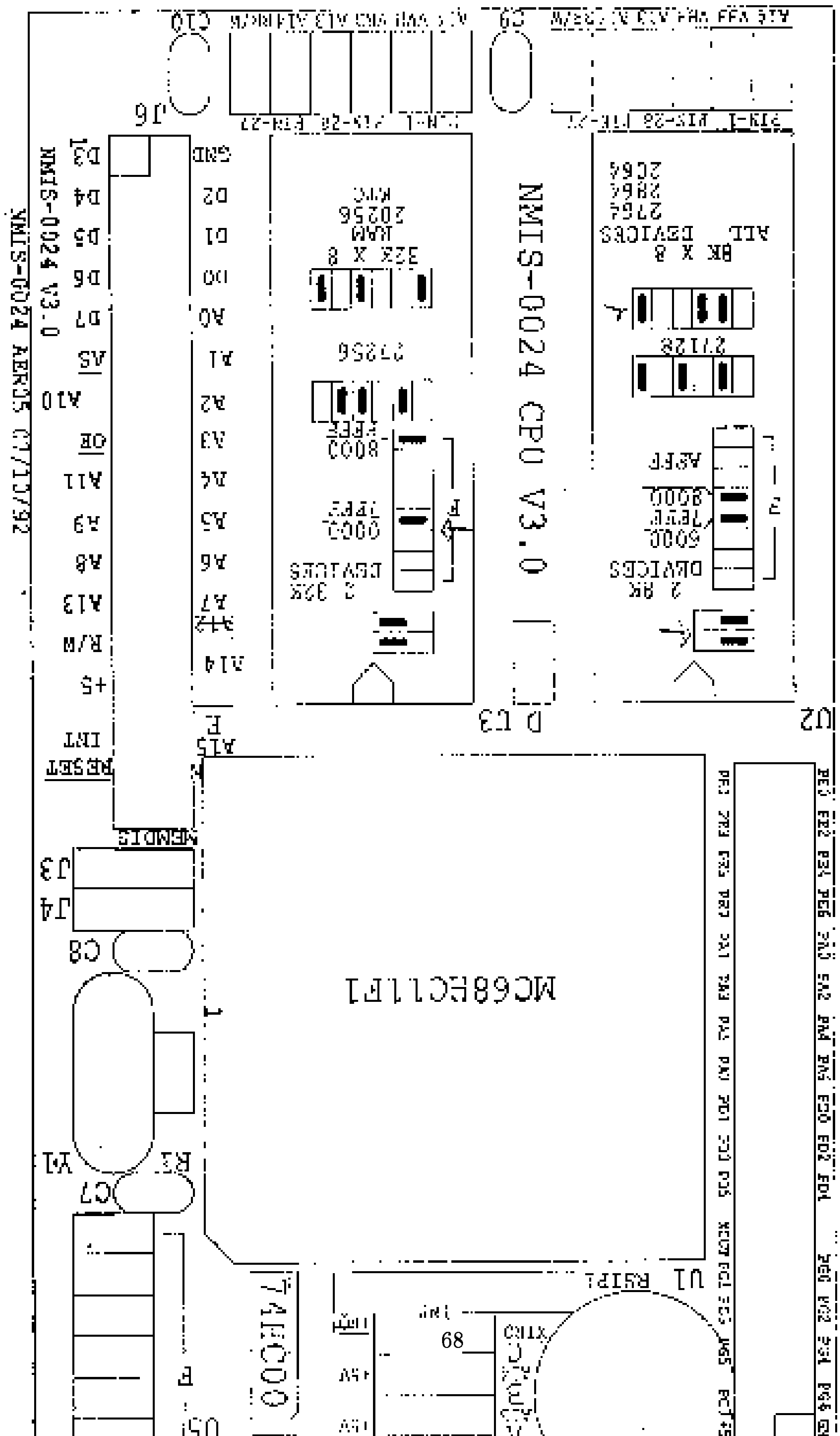


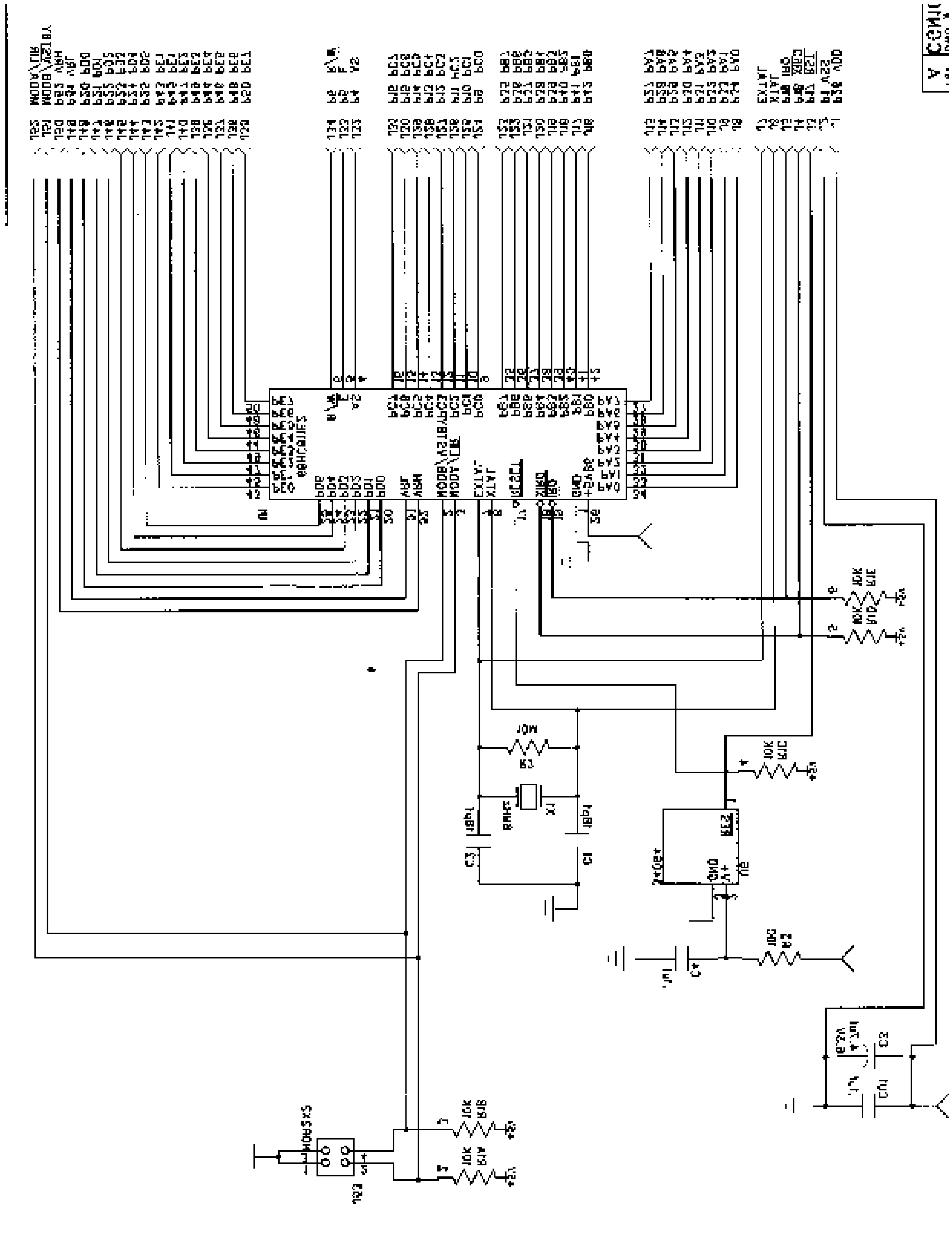
System Schematic



RS−232 Interface







Appendix B

Demo Code

- Leg code
- Central code

```
*****
*
*  This file contains the leg controller
*          made up of servo.asm and spislv.asm
*****
*
*
***   SERVO SPECS:
***
***     POSITION        PULSE WIDTH     CYCLES (2 MHz TIMER W/ PR1,PR2=0,0)
***     --------        -----------     --------------------
***     -90 deg         0.7ms           1400
***     0 deg           1.2ms           2400
***     90 deg          1.7ms           3400
***
***     PWM PERIOD = 20ms = 40,000 CYCLES (2 MHz TIMER) =  $9C40
***
***     NOTE: In general, the joint zeros are different from the servo zeros.
***  SWING       SERVO0  OC2
***  SHOULDER    SERVO1  OC3
***  ELBOW       SERVO2  OC4
TMRSPD  EQU     !20     ;PROGRAMMABLE TIMER BASE SPEED (IN MHz) TIMES 10
                        ;FOR 8MHz CRYSTAL, TIMER = 2MH
DEGPERT EQU     !105    ;CHANGE IN SERVO POSITION (IN DEGREES) PER 1ms CHANGE
                        ;IN PWM PULSE WIDTH
NEUTPW  EQU     !120    ;PULSE WIDTH (IN ms) FOR SERVO ZERO-ED TIMES 100
PERCNT  EQU     !40000  ;MACH. CYCLE COUNT FOR PWM PERIOD (PERCNT = 20ms/1.5ms)
*** leg 1, 2
*JZLOC0  EQU     !13     ;LOCATION OF JOINT ZERO RELATIVE TO SERVO ZERO (IN DEG)
*JZLOC1  EQU     !88     ;$D3 = -45 DEGREES     (was $c5)
*JZLOC2  EQU     $EE
*** leg 3,4
JZLOC0  EQU     !15     ;LOCATION OF JOINT ZERO RELATIVE TO SERVO ZERO (IN DEG)
JZLOC1  EQU     $C4     ;$D3 = -45 DEGREES
JZLOC2  EQU     $36
PWMISR  EQU     !78    ;LENGHT OF PWM INTERUPT SERVICE ROUTINES IN MACH. CYL
EEPROM  EQU     $F800   ;LOCATION OF EEPROM in mc68hc811e2fn
STACK   EQU     $0060   ;LOCATION OF STACK FOR CGN
REGBAS  EQU     $1000  ; Starting address for register block
DELAY1  EQU     $02     ;time delay=DELAY1*0.3s
DELAY2  EQU     $FFFF   ;0.3s COUNT
PORTB   EQU     $04    ; Output port B
*PORTG   EQU     $02    ;PORT G
*PORTA   EQU     $00    ;PORT A
PORTC   EQU     $03     ;PORTC
PORTD   EQU     $08
OC1M    EQU     $0C    ; OC1M7,OC1M6,OC1M5,OC1M4;OC1M3,-,-,-
OC1D    EQU     $0D    ; OC1D7,OC1D6,OC1D5,OC1D4;OC1D3,-,-,-
TCNT    EQU     $0E    ; Free running counter (16-bit)
TIC1    EQU     $10    ; IC1 register (16-bit)
TOC1    EQU     $16    ; OC1 register (16-bit)
TOC2    EQU     $18    ; OC2 register (16-bit)
TOC3    EQU     $1A    ; OC3 register (16-bit)
TOC4    EQU     $1C
TCTL1   EQU     $20    ; OM2,OL2,OM3,OL3;OM4,OL4,OM5,OL5
TCTL2   EQU     $21    ; -,-,EDG1B,EDG1A;EDG2B,EDG2A,EDG3B,EDG3A
TMSK1   EQU     $22    ; OC1I,OC2I,OC3I,OC4I;OC5I,IC1I,IC2I,IC3I
TFLG1   EQU     $23    ; OC1F,OC2F,OC3F,OC4F;OC5F,IC1F,IC2F,IC3F
TMSK2   EQU     $24    ; TOI,RTII,PAOVI,PAII;-,-,PR1,PR0
TFLG2   EQU     $25    ; TOF,RTIF,PAOVF,PAIF;-,-,-,-
PVIC1   EQU     $00E8  ; EVB Pseudo Vector for IC1
PVTOF   EQU     $00D0  ; EVB Pseudo Vector for TOF
PVOC2   EQU     $00DC  ; EVB Pseudo Vector for OC2
PVOC1   EQU     $00DF  ; EVB Pseudo Vector for OC1
PVOC3   EQU     $00D9
PVOC4   EQU     $00D6
PVSPI   EQU     $00C7   ;PSUEDO VECTOR FOR SPI TRANSFER DONE
COUNT   EQU     $0001
***SPI EQUATES
SPCR    EQU     $28    ;|SPIE|SPE|DWOM|MSTR|CPOL|CPHA|SPR1|SPR0
SPSR    EQU     $29      ;|SPIF|WCOL| |MODF| | | | | (FLAGS)
DDRD    EQU     $09      ;| | |SS|SCK|MOSI|MISO|TXD|RXD|
*DDRA    EQU     $01      ;OUTPUT CONTROL FOR PORT A
```

```
DDRC    EQU     $07     ;OUPUT CONTROL PORT C
*DDRG    EQU     $03   ;OUTPUT CONTROL FOR PORT G
SPDR    EQU     $2A   ;SPI DATA REGISTER
***END SPI EQUATES
*** RAM Variable Assignments
       ORG      $0000  ; Start variables AT TOP OF RAM FOR CGN
*        ORG      $0100   ;START VARIABLES HERE FOR NM BOARD
POS0    RMB     1        ;ANGLE FOR SERVO0 TO GO TO  (THESE MUST REMAIN
POS1    RMB     1        ;   CONSECUTIVE IN
POS2    RMB     1        ;   MEMORY.)
***
ANGLE   RMB     1        ;ANGLE TO BE CONVERTED FROM DEGREES TO TIMER COUNTS
MOVETO  RMB     1        ;DESIRED ANGLE TO MOVE TO
SERVO   RMB     1        ;ACTIVE SERVO SELECT:  -----|S2|S1|S0|
DEG2CYL RMB     1        ;MACHINE CYCLES PER JOINT DEGREES
COMP    RMB     1        ;TEMPORARY REGISTER FOR BIT COMPARE
SZERO   RMB     2        ;TIMER HIGH COUNT FOR SERVO ZERO (NOT JOINT ZERO!)
JZERO0  RMB     2        ;TIMER HIGH COUNT FOR JOINT ZERO
JZERO1  RMB     2
JZERO2  RMB     2
HICNT0  RMB     2        ;NUMBER OF TIMER CYCLES TO STAY HI
LOCNT0  RMB     2        ;NUMBER OF TIMER CYCLES TO STAY LO
HICNT1  RMB     2
LOCNT1  RMB     2
HICNT2  RMB     2
LOCNT2  RMB     2
***$1C RAM VARIABLE LOCATIONS NEEDED
*******************
***    MAIN    ****
*******************
        ORG     $F800   ;code startS here  (TOP OF EEPROM) FOR CGN
*        ORG     $0130   ;CODE FOR NM BOARD
***INITIALIZATION
        LDS     #STACK  ;stack FOR CGN
        LDX     #REGBAS
        BSET    DDRC,X,%11111111        ;ENABLE PORT C OUTPUTS (DATA REC'D)
        JSR     INITSERVOS    ;INITIALIZE SERVOS, START TIMING INTERRUPTS
        BSR     INITSPISLV          ;INITIALIZE SPI AS SLAVE
*        BSET    PORTC,X,%00001000
*STOP    BRA     STOP
***BODY
LOOP
*        LDAA    #%01010101
*        STAA    PORTC,X
*       TPA
*       STAA     PORTC,X
*        BRA     STOP
        LDAA    #%00000001
        STAA    SERVO
        LDAA    POS0                    ;GET POS0
        STAA    MOVETO
        JSR     MOVE
        LDAA    #%00000010
        STAA    SERVO
        LDAA    POS1                    ;GET POS1
        STAA    MOVETO
        JSR     MOVE
        LDAA    #%00000100
        STAA    SERVO
        LDAA    POS2                    ;GET POS2
        STAA    MOVETO
        JSR     MOVE
        BRA     LOOP
************************
***   SUBROUTINES  ***
***********************
        ORG     $F850   ;SUBROUTINES START HERE  FOR CGN
*        ORG     $0180   ;SUBROUTINES FOR NM BOARD
******************************************************
***SPI SUBROUTINES
INITSPISLV
***INITIALIZES THE SPI FOR SLAVE OPERATION
*        BSET    PORTC,X,%00000001
        LDAA    #$7E    ;SET UP INTERRUPT HANDLER (Jump Opcode)
        STAA    PVSPI   ;SPI Pseudo Vector
        LDX     #SPISERV ;Address of SPI service routine
        STX     PVSPI+1 ;Finish jump instruc to SPI routine
        LDX     #REGBAS  ;RESTORE X
*        BSET    PORTC,X,%00000010
        BSET    DDRD,X,%00000100      ;ENABLE MISO
```

```
*        BCLR    DDRD,X,%00000000      ;ENABLE SS, SCK, MOSI INPUTS (NOT NEC.)
        LDAA    #%11000000      ;ENABLE INTERUPT, ENABLE SPI, NORM OUTPUTS,
        STAA    SPCR,X          ;BE A SLAVE, SCK LOW IDLE, CPOL=0, CPHA=0 (SS
                                ;CAUSES START OF TRANSFER)
                                ;FASTEST SCK
*        BSET    PORTC,X,%00000100
        LDAA    SPSR,X    ;CLEARS SPIF (FLAG) IF SET
        LDAA    SPDR,X
        RTS
*******************************************************
*************************************************
***END SPI SUBROUTINES
***INITIALIZATION FOR PWM SERVO CONTROL SUBROUTINE
***   SETS UP TIMING VARIABLES
***   STARTS SERVOS AT THEIR ZERO POSITIONS
***   ENABLES THE TIMER INTERUPTS
INITSERVOS
        PSHA
        PSHB
        PSHX
        LDAA    #TMRSPD  ;Calculate JOINT DEGREES TO MACH. CYCLE CONVERSION
        LDAB    #!100 ;
        MUL
        LDX     #DEGPERT
        IDIV
        XGDX
        STAB    DEG2CYL ;DEG2CYL = (TMRSPD * 100)/DEGPERT (ONLY WILL USE LOWER
                        ;   BYTE)
        LDAA    #NEUTPW  ;CALCULATE SERVO ZERO HIGH COUNT
        LDAB    #TMRSPD
        MUL
        STD     SZERO   ;SZERO = TMRSPD * NEUTPW
        LDAA    #JZLOC0 ;CALCULATE JOINT ZERO HIGH COUNT *SERVO0*
        STAA    ANGLE   ;relative position of joint zero to servo zero (in deg)
        JSR     CONVERT         ;CONVERTS DEGREES TO TIMER COUNTS
        ADDD    SZERO   ;add relative joint zero to absolute servo zero
        STD     JZERO0  ;will add this to move angles to get new HICNT, LOCNT
        STD     HICNT0  ;INITIALIZE JOINT LOCATION TO ZERO (HICNT)
        LDD     #PERCNT  ;calculate LOCNT
        SUBD    HICNT0
        STD     LOCNT0
        LDAA    #JZLOC1 ;CALCULATE JOINT ZERO HIGH COUNT  *SERVO1*
        STAA    ANGLE   ;relative position of joint zero to servo zero (in deg)
        JSR     CONVERT               ;convert to timer cycles
        ADDD    SZERO   ;add relative joint zero to absolute servo zero
        STD     JZERO1  ;will add this to move angles to get new HICNT, LOCNT
        STD     HICNT1  ;INITIALIZE JOINT LOCATION TO ZERO (HICNT)
        LDD     #PERCNT  ;calculate LOCNT
        SUBD    HICNT1
        STD     LOCNT1
        LDAA    #JZLOC2 ;CALCULATE JOINT ZERO HIGH COUNT  *SERVO2*
        STAA    ANGLE   ;relative position of joint zero to servo zero (in deg)
        JSR     CONVERT               ;convert to timer cycles
        ADDD    SZERO   ;add relative joint zero to absolute servo zero
        STD     JZERO2  ;will add this to move angles to get new HICNT, LOCNT
        STD     HICNT2  ;INITIALIZE JOINT LOCATION TO ZERO (HICNT)
        LDD     #PERCNT  ;calculate LOCNT
        SUBD    HICNT2
        STD     LOCNT2
        LDAA    #$7E    ;SET UP INTERRUPT HANDLER (Jump Opcode) *FOR SERVO0*
        STAA    PVOC2   ;OC2 Pseudo Vector
        LDX     #SV7OC2 ;Address of OC2 service routine
        STX     PVOC2+1 ;Finish jump instruc to OC2 routine
        STAA    PVOC3   ;OC3 Pseudo Vector  *FOR SERVO1*
        LDX     #SV7OC3 ;Address of OC3 service routine
        STX     PVOC3+1 ;Finish jump instruc to OC3 routine
        STAA    PVOC4   ;OC4 Pseudo Vector *FOR SERVO2*
        LDX     #SV7OC4 ;Address of OC4 service routine
        STX     PVOC4+1 ;Finish jump instruc to OC4 routine
        LDX     #REGBAS
        LDAA    #%11111100 ;ENABLE OC OUTPUTS
        STAA    TCTL1,X    ;   First OC2,3,4 starts high
        LDD     #$0000  ;SPACE THE FIRST INTERUPT CALLS TO AVOID COLLISION
        STD     TOC2,X  ;Start first PWM period at TCNT=$0000
        LDD     #$2000
        STD     TOC3,X  ;START FIRST OC3 PERIOD AT TCNT=$2000
        LDD     #$4000
        STD     TOC4,X  ;START FIRST OC4 PERIOD AT TCNT=$4000
```

```
        LDAA    #%01110000      ;ENABLE TIMER INTERUPTS
        STAA    TFLG1,X ;OC2,3,4F=1 to clear any old OC2,3,4 flag
        STAA    TMSK1,X ;then OC2,3,4I=1 to enable OC2,3,4 interrupt
        CLI
        PULX
        PULB
        PULA
        RTS
***END INITIALIZATION
***JOINT ANGLE TO TIMER COUNT CONVERSION SUBROUTINE
***     ANGLE TO CONVERT IN ANGLE
***     RETURNES THE TIMER COUNT IN D
CONVERT
        LDAB    DEG2CYL ;GET CONVERSION FACTOR
        LDAA    ANGLE
        BGT     POSI     ;CHECK TO SEE IF POSITIVE ANGLE
        NEGA            ;MAKE POSITIVE
        MUL             ;MULTIPLY BY CONVERSION FACTOR
        COMA            ;MAKE NEGATIVE AGAIN (16 BITS)
        COMB
        ADDD    #$0001
        RTS
POSI    MUL             ;convert to timer cycles
        RTS
***END CONVERSION
***MOVE SUBROUTINE
***
***   (2'S COMP) ANGLE TO MOVE TO IS CONTAINED IN MOVETO
***   ACTIVE SERVO CONTAINED IN SERVO
MOVE    PSHB
        PSHX
        BSET    PORTB,X,%00000010
        LDAA    MOVETO  ;GET DESIRED POSITION
        STAA    ANGLE
        JSR     CONVERT ;CONVERT DEGREES TO TIMER COUNTS
        XGDX            ;SAVE RESULTS IN X
        LDAA    #%00000010   ; DETERMINE WHICH SERVO IS ACTIVE
        CMPA    SERVO
        BEQ     HILO1
        LDAA    #%00000100
        CMPA    SERVO
        BEQ     HILO2
HILO0   XGDX                    ;GET POSITION BACK
        ADDD    JZERO0  ;CALCULATE HICNT *SERVO0*
        STD     HICNT0
        LDD     #PERCNT  ;CALCULATE LOCNT
        SUBD    HICNT0
        STD     LOCNT0  ;LOCNT0 = PERCNT - HICNT0
        BRA     MVDONE
HILO1   XGDX                    ;GET POSITION BACK
        ADDD    JZERO1  ;CALCULATE HICNT *SERVO1*
        STD     HICNT1
        LDD     #PERCNT  ;CALCULATE LOCNT
        SUBD    HICNT1
        STD     LOCNT1  ;LOCNT1 = PERCNT - HICNT1
        BRA     MVDONE
HILO2   XGDX                    ;GET POSITION BACK
        ADDD    JZERO2  ;CALCULATE HICNT *SERVO2*
        STD     HICNT2
        LDD     #PERCNT  ;CALCULATE LOCNT
        SUBD    HICNT2
        STD     LOCNT2  ;LOCNT2 = PERCNT - HICNT2
MVDONE  PULX
        PULB
        BCLR    PORTB,X,%0000010
        RTS
***END MOVE SUBROUTINE
***DELAY SUBROUTINE
*DELAY   LDAA    #DELAY1
*        LDX     #REGBAS
*        BSET    PORTB,X,%00001000
*        STAA    COUNT
*LOOP2   LDD     #DELAY2
*LOOP1   SUBD    #$0001  ;DECREMENT D
*        BNE     LOOP1
*        LDAA    COUNT
*        SUBA    #$0001  ;DECREMENT COUNT1
*        BNE     UPDATE
*        RTS
*UPDATE  STAA    COUNT
*        BRA     LOOP2
***END DELAY SUBROUTINE
***INTERUPT SERVICE ROUTINES
SPISERV
***TRANSFER WAS COMPLTED
```

```
***SPIOUT = DATA TO SEND TO MOTHER
***THREE DATA PACKETS RECIEVED--PUT IN POS0, POS1, POS2
        LDX     #REGBAS ;Point to register block          (3 CYCLES)
        BSET    PORTB,X,%00000001
        BCLR    PORTB,X,%00000001
        LDAB    SPSR,X  ;RESET SPIF
        LDAB    SPDR,X    ;RECEIVE DATA FROM MOTHER (ALSO CLR SPIF)
        STAB    PORTC,X         ;DISPLAY DATA RECEIVED
        BSET    PORTB,X,%00000001
        BCLR    PORTB,X,%00000001
        CLRA
        ASLD                    ;GET SERVO CODE, AND MULTIPLY ANGLE BY 4
        ASLD
        CMPA    #%00000000      ;SERVO 0
        BEQ     SPI0
        DECA                    ;SERVO 1
        BEQ     SPI1
        DECA                    ;SERVO 2
        BEQ     SPI2
        DECA                    ;ERROR
        BEQ     SPIERR
SPI0    STAB    POS0
        RTI
SPI1    STAB    POS1
        RTI
SPI2    STAB    POS2
        RTI
SPIERR  RTI
***
* SV7OC2 - Output Compare 2 service routine
***
***ADDS OFFHI0 TO CURRENT COUNT AND SETS OUTPUT HIGH IF OUTPUT IS CURRENTLY LOW
***ADDS OFFLO0 TO CURRENT COUNT AND SETS OUTPUT LOW IF OUTPUT IS CURRENTLY HIGH
* ASSUME 12 + 3 = 15 CYCLES TO JUMP TO INTERUPT SERVER     (15 CYCLES)
SV7OC2  LDX     #REGBAS ;Point to register block          (3 CYCLES)
*        BSET    PORTB,X,%00001000
        BRCLR   TCTL1,X,%01000000,ADDLO2 ;See which half of cyc (7)
        LDD     HICNT0   ;High part so we will add HICNT0 to OC2 (5)
        BRA     UPOC2    ;                                       (3)
ADDLO2  LDD     LOCNT0   ;Low part so we will add LOCNT0 to OC2 (5)
UPOC2   ADDD    TOC2,X  ;Add to last compare value          (6)
        STD     TOC2,X  ;Update OC2 (schedule next edge)    (5)
        LDAA    TCTL1,X ;Change OL2 to setup for next edge  (4)
        EORA    #%01000000      ;Inverts OL2 bit            (2)
        STAA    TCTL1,X ;Update control reg                 (4)
        BCLR    TFLG1,X,%10111111     ;  Clear OC2F        (7)
*        BCLR    PORTB,X,%00001000
        RTI                     ;  ** Return from OC2 service **  (12)
***                                                       (78 CYCLES TOTAL)
***
* SV7OC3 - Output Compare 3 service routine
***
***ADDS OFFHI1 TO CURRENT COUNT AND SETS OUTPUT HIGH IF OUTPUT IS CURRENTLY LOW
***ADDS OFFLO1 TO CURRENT COUNT AND SETS OUTPUT LOW IF OUTPUT IS CURRENTLY HIGH
SV7OC3  LDX     #REGBAS ;Point to register block
*        BSET    PORTB,X,%00001000
        BRCLR   TCTL1,X,%00010000,ADDLO3 ;See which half of cyc
        LDD     HICNT1   ;High part so we will add HICNT1 to OC3
        BRA     UPOC3
ADDLO3  LDD     LOCNT1   ;Low part so we will add LOCNT1 to OC3
UPOC3   ADDD    TOC3,X  ;Add to last compare value
        STD     TOC3,X  ;Update OC3 (schedule next edge)
        LDAA    TCTL1,X ;Change OL3 to setup for next edge
        EORA    #%00010000      ;Inverts OL2 bit
        STAA    TCTL1,X ;Update control reg
        BCLR    TFLG1,X,$DF     ;$DF = %11011111, Clear OC3F
*        BCLR    PORTB,X,%00001000
        RTI                     ;  ** Return from OC3 service **
***
* SV7OC4 - Output Compare 4 service routine
***
***ADDS OFFHI2 TO CURRENT COUNT AND SETS OUTPUT HIGH IF OUTPUT IS CURRENTLY LOW
***ADDS OFFLO2 TO CURRENT COUNT AND SETS OUTPUT LOW IF OUTPUT IS CURRENTLY HIGH
SV7OC4  LDX     #REGBAS ;Point to register block
        BRCLR   TCTL1,X,%00000100,ADDLO4 ;See which half of cyc
```

```
        LDD     HICNT2   ;High part so we will add HICNT2 to OC4
        BRA     UPOC4
ADDLO4  LDD     LOCNT2   ;Low part so we will add LOCNT2 to OC4
UPOC4   ADDD    TOC4,X  ;Add to last compare value
        STD     TOC4,X  ;Update OC4 (schedule next edge)
        LDAA    TCTL1,X ;Change OL4 to setup for next edge
        EORA    #%00000100      ;Inverts OL4 bit
        STAA    TCTL1,X ;Update control reg
        BCLR    TFLG1,X,$EF     ;$EF = 11101111, Clear OC4F
        RTI                     ;  ** Return from OC4 service **
```

```
******
***MOTHER***
***This file contains the mother processor controller
***
***
***Memory alocation:
***VARIABLES    $0100
***STACK IN BUFFALO
***SUBROUTINES  $0200
***MAIN         $0500  (1K)
REGBAS  EQU     $1000
STACK   EQU     $014F   ;STACK LOCATION
SPCR    EQU     $28    ;|SPIE|SPE|DWOM|MSTR|CPOL|CPHA|SPR1|SPR0
SPSR    EQU     $29    ;|SPIF|WCOL| |MODF| | | | | (FLAGS)
DDRD    EQU     $09    ;| | |SS|SCK|MOSI|MISO|TXD|RXD|
DDRA    EQU     $01    ;OUTPUT CONTROL FOR PORT A
DDRG    EQU     $03    ;OUTPUT CONTROL FOR PORT G
SPDR    EQU     $2A    ;SPI DATA REGISTER
PORTG   EQU     $02    ;PORT G
PORTA   EQU     $00    ;PORT A
        ORG     $0100   ;FOR VARIABLES
SPEED   RMB     2       ;SET THIS FOR THE CYCLE SPEED
***STEP1 (LEG2)
POSTUR1   FCB  $07,$F0,$E8,$12,$00,$E8,$EB,$00,$F8,$F1,$F0,$F8  ;POSTURE 1
***            <--LEG 1--> <--LEG 2--> <--LEG 3--> <--LEG 4-->
***            S0, S1, S2||S0, S1, S2||S0, S1, S2||S0, S1, S2
***STEP2 (LEG 4)
POSTUR2   FCB  $07,$F0,$F8,$12,$00,$18,$EB,$00,$08,$F1,$F0,$E8
POSTUR3   FCB  $07,$F0,$F8,$12,$0F,$18,$EB,$00,$08,$F1,$00,$E8
POSTUR4   FCB  $07,$F0,$F8,$12,$1F,$18,$EB,$0F,$08,$F1,$0F,$E8
***STEP3 (LEG 3)
POSTUR5   FCB  $07,$F0,$08,$12,$1F,$08,$EB,$0F,$18,$F1,$0F,$18
POSTUR6   FCB  $07,$0F,$08,$12,$00,$08,$EB,$E0,$18,$F1,$0F,$18
POSTUR7   FCB  $07,$0F,$08,$12,$00,$08,$EB,$00,$18,$F1,$0F,$18
***STEP4 (LEG 1)
POSTUR8   FCB  $07,$0F,$18,$12,$00,$F8,$EB,$00,$E8,$F1,$0F,$08
POSTUR9   FCB  $07,$0F,$18,$12,$00,$F8,$EB,$E0,$E8,$F1,$0F,$08
POSTUR10  FCB  $07,$00,$18,$12,$00,$F8,$EB,$E0,$E8,$F1,$0F,$08
***(STEP1 (LEG2))
POSTUR11  FCB  $07,$00,$E8,$12,$00,$E8,$EB,$E0,$F8,$F1,$0F,$F8
POSTUR12  FCB  $07,$F0,$E8,$12,$1F,$E8,$EB,$00,$F8,$F1,$F0,$F8
POSTEND   FCB  $00
POSTCNT RMB     2
ANGCNT  RMB         2
LEGSEL  RMB     1
COUNT   RMB     1
        ORG     $0200   ;SUBROUTINES
*********************************************************
INITSPIMSTR
***INITIALIZES THE SPI FOR MASTER OPERATION
*        PSHA            ;PRESERVE REGISTERS
        BSET    DDRG,X,%00001111        ;ENABLE G0 OUTPUT (FOR SLAVE SELECT)
        BSET    PORTG,X,%00001111          ;RESET G0 TO DESELECT SLAVES
        BSET    DDRD,X,%00111000      ;ENABLE SCK, DISABLE SS, ENABL MOSI OUTPUTS
        BCLR    DDRD,X,%00000000      ;MAKE MISO AN INPUT
        LDAA    #%01010000      ;DISABLE INTERRUPT, ENABLE SPI, NORM OUTPUTS,
        STAA    SPCR,X          ;BE A MASTER, SCK IDLE LOW, CPOL=0, CPHA=0
                                ; FASTEST SCK
        LDAA    SPSR,X    ;CLEARS SPIF (FLAG) IF SET
        LDAA    SPDR,X
*       PULA
        RTS
**********************************************************
IOSPIMSTR
***SENDS DATA TO SLAVE
***RECEIVES: ACCB = DATA TO SEND, ACCA = LEG TO WRITE TO  (b0-b4)
***RETURNES: ACCA = DATA RECEIVED FROM SLAVE
        STAA    PORTG,X   ;SELECT SLAVE (ONLY ONE BIT SHOULD BE SET)
        STAB    SPDR,X    ;TRANSMIT DATA TO SLAVE
POLL    BRCLR   SPSR,X,%10000000,POLL       ;WAIT UNTIL DATA TRANSFER COMPLETE
        LDAA    SPDR,X    ;RECEIVE DATA FROM SLAVE
        BSET    PORTG,X,%00001111         ;DESELECT SLAVE(S)
        RTS
***DELAY***
***ACCA HAS FIRST LOOP COUNT
DELAY
        PSHB
```

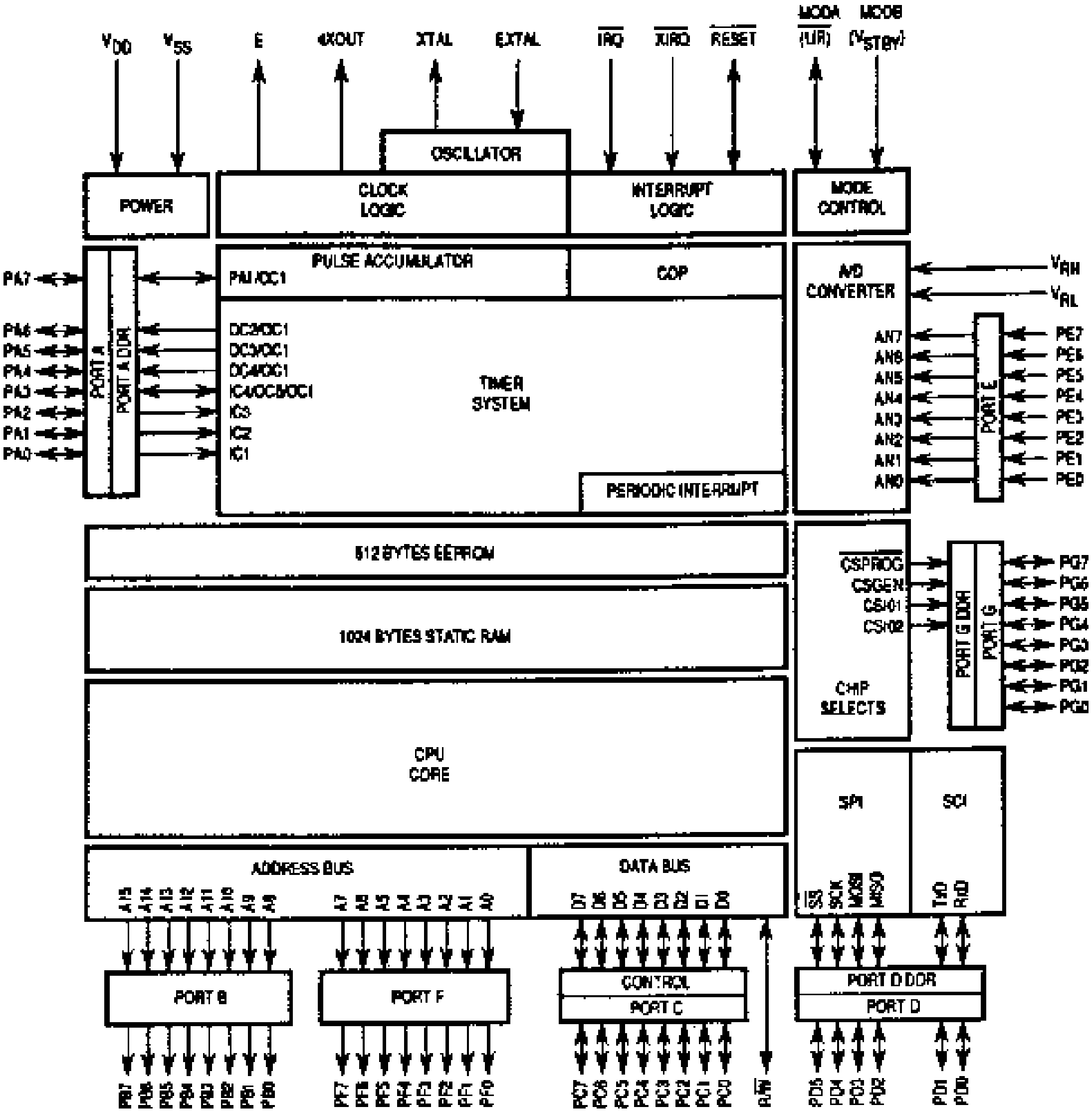
```
        STAA    COUNT    ;REMEMBER OUTER LOOP COUNT
LOOP2   LDD     SPEED
LOOP1   SUBD    #$0001  ;DECREMENT D
        BNE     LOOP1
        LDAA     COUNT              ;GET OUTER LOOP COUNT
        SUBA    #$01  ;DECREMENT COUNT1
        BNE     UPDATE
        PULB
        RTS
UPDATE  STAA    COUNT
        BRA     LOOP2
*************************************************
***   MAIN   *******
*******************
        ORG     $0500   ;MAIN PROGRAM
        LDX     #REGBAS
*        LDS     STACK            ;LOCATE STACK
*        BSET    DDRG,X,%00001111
*       BSET    PORTG,X,%11111111
       BSET    DDRA,X,%00001111  ;ENABLE A0 OUTPUT (FOR DATA RECV'D DISPLAY)
        JSR     INITSPIMSTR          ;INITIALIZE SPI AS MASTER
*STOP    BRA     STOP
***PACKET FORMAT:   |SS1|SS0|A5|A4|A3|A2|A1|A0|
***                 SS1,SS0 = 00 --->SERVO 0 SELECT
***                         = 01 --->SERVO 1 SELECT
***                         = 10 --->SERVO 2 SELRCT
***                 A0-A5  =  ANGLE / 4 (2'S COMP)
REPEAT
        LDX     #POSTUR1
        STX     POSTCNT
        LDX     #REGBAS
POSTLOOP                ;POSTURE LOOP
        LDAA    #$FF
        JSR     DELAY
        LDAA    #%11111110  ;RESET LEG COUNT
        STAA    LEGSEL
        LDX    POSTCNT          ;LOCATION OF NEXT POSTURE
        STX    ANGCNT
        LDX    #REGBAS
LEGLOOP
SERVO0
        LDAA    #%00000000       ;SELECT SERVO 0
        LDX     ANGCNT
        LDAB    $00,X       ;GET ANGLE  0
        LDX     #REGBAS
        LSRD                     ;DIVIDE BY 4
        LSRD
        LDAA     LEGSEL       ;SELECT LEG  (ACTIVE LOW)
        JSR      IOSPIMSTR            ;SEND/RECEIVE DATA
*        STAA     PORTA,X            ;DISPLAY REC'D DATA
        LDAA    #$01
        JSR     DELAY
SERVO1
*        BCLR     PORTA,X,%00001111       ;RESET REC'D DATA DISPLAY
        LDAA    #%00000001       ;SELECT SERVO 1
        LDX     ANGCNT
        INX
        STX     ANGCNT
        LDAB    $00,X      ;GET ANGLE 1
        LDX     #REGBAS
        LSRD                     ;DIVIDE BY 4
        LSRD
        LDAA     LEGSEL       ;SELECT LEG  (ACTIVE LOW)
        JSR      IOSPIMSTR            ;SEND/RECEIVE DATA
*        STAA     PORTA,X            ;DISPLAY REC'D DATA
        LDAA    #$01
        JSR     DELAY
SERVO2
*        BCLR     PORTA,X,%00001111       ;RESET REC'D DATA DISPLAY
        LDAA    #%00000010       ;SELECT SERVO 2
        LDX     ANGCNT
        INX
        STX     ANGCNT
        LDAB    $00,X       ;GET ANGLE 2
        LDX     #REGBAS
        LSRD                     ;DIVIDE BY 4
        LSRD
        LDAA     LEGSEL       ;SELECT LEG  (ACTIVE LOW)
        JSR      IOSPIMSTR            ;SEND/RECEIVE DATA
*        STAA     PORTA,X            ;DISPLAY REC'D DATA
        LDAA    #$01
        JSR     DELAY
```

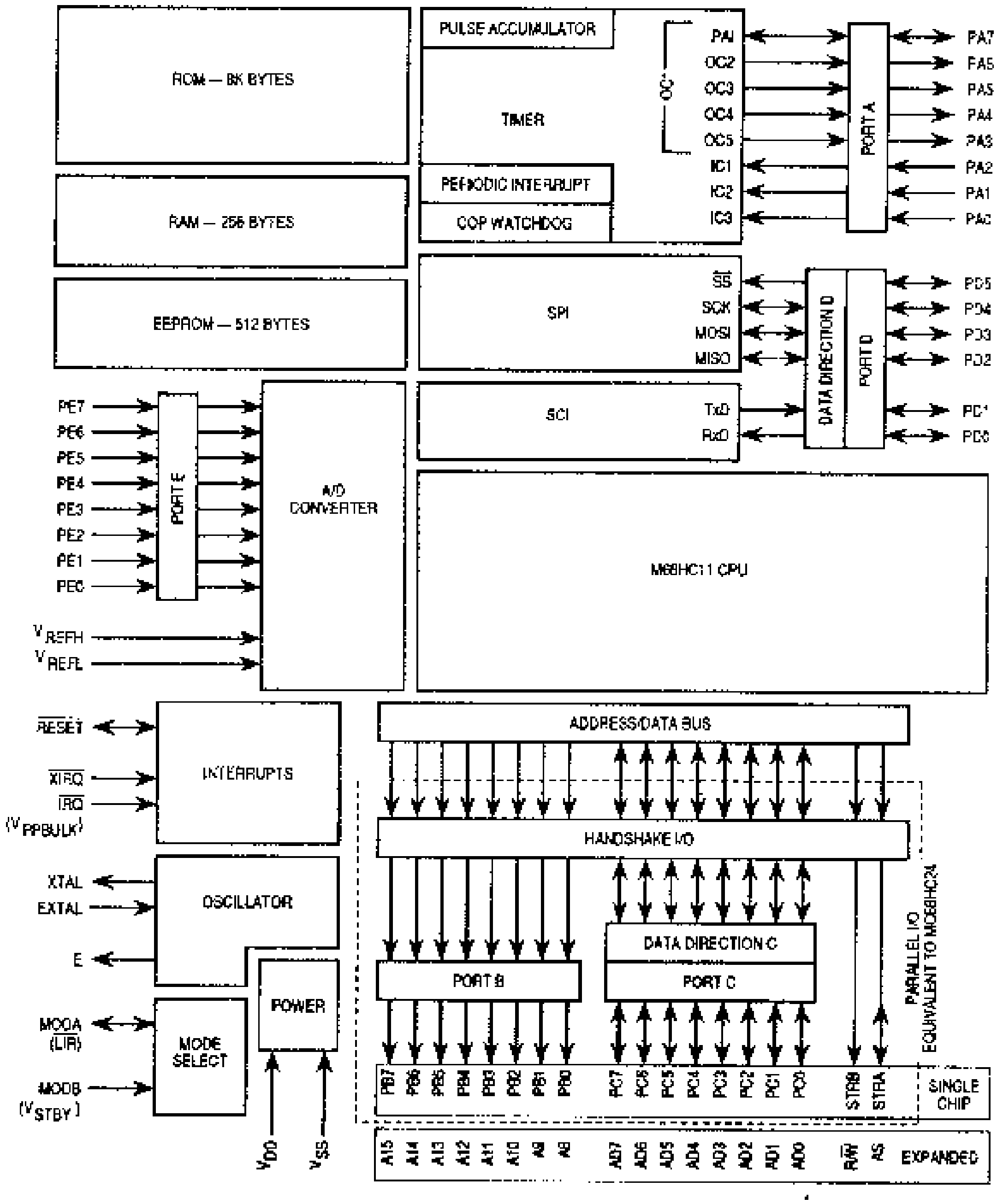
```
        LDX     ANGCNT          ;increment to next leg
        INX
        STX     ANGCNT
        LDX     #REGBAS
        LDAA    LEGSEL            ;GET LEG COUNT
        SEC
        ROLA
        STAA    LEGSEL
        CMPA    #%11101111
        BNE     LEGLOOP
        LDX     POSTCNT
        LDAB    #$0C
        ABX             ;INCREMENT TO NEXT POSTURE
        STX     POSTCNT
        CPX    #POSTEND
        BNE     JUMP
STOP   JMP     REPEAT
JUMP    JMP     POSTLOOP
```

Appendix C

HC11 Specifications

- MC68HC11F1
- MC68HC811E2FN


VDD
VSS
E
4XOUT
XTAL
EXTAL
IRQ
XIRQ
RESET
MODA (LIR)
MODB (VSTBY)
OSCILLATOR
POWER
CLOCK LOGIC
INTERRUPT LOGIC
MODE CONTROL
PULSE ACCUMULATOR
COP
A/D CONVERTER
VRH
VRL
PORT A
PORT A DDR
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PAI/OC1
OC2/OC1
OC3/OC1
OC4/OC1
IC4/OC5/OC1
IC3
IC2
IC1
TIMER SYSTEM
PERIODIC INTERRUPT
AN7
AN6
AN5
AN4
AN3
AN2
AN1
AN0
PORT E
PE7
PE6
PE5
PE4
PE3
PE2
PE1
PE0
512 BYTES EEPROM
1024 BYTES STATIC RAM
CSPROG
CSGEN
CS/01
CS/02
CHIP SELECTS
PORT G DDR
PORT G
PG7
PG6
PG5
PG4
PG3
PG2
PG1
PG0
CPU CORE
SPI
SCI
ADDRESS BUS
DATA BUS
A15 A14 A13 A12 A11 A10 A9 A8
A7 A6 A5 A4 A3 A2 A1 A0
D7 D6 D5 D4 D3 D2 D1 D0
SS SCK MOSI MISO
TxD RxD
PORT B
PORT F
CONTROL
PORT C
PORT D DDR
PORT D
PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0
PF7 PF6 PF5 PF4 PF3 PF2 PF1 PF0
PC7 PC6 PC5 PC4 PC3 PC2 PC1 PC0
R/W
PD5 PD4 PD3 PD2
PD1 PD0



ROM — 8K BYTES
RAM — 256 BYTES
EEPROM — 512 BYTES
PULSE ACCUMULATOR
TIMER
PERIODIC INTERRUPT
COP WATCHDOG
PAI
OC2
OC3
OC4
OC5
OC1
IC1
IC2
IC3
PORT A
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
SPI
SS
SCK
MOSI
MISO
SCI
TxD
RxD
DATA DIRECTION D
PORT D
PD5
PD4
PD3
PD2
PD1
PD0
PE7
PE6
PE5
PE4
PE3
PE2
PE1
PE0
PORT E
A/D CONVERTER
M68HC11 CPU
V_{REFH}
V_{REFL}
RESET
XIRQ
IRQ
(V_{PPBULK})
INTERRUPTS
XTAL
EXTAL
E
OSCILLATOR
MODA (LIR)
MODB (V_{STBY})
MODE SELECT
POWER
V_{DD}
V_{SS}
ADDRESS/DATA BUS
HANDSHAKE I/O
DATA DIRECTION C
PORT B
PORT C
PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0
PC7 PC6 PC5 PC4 PC3 PC2 PC1 PC0
STRB STRA
SINGLE CHIP
A15 A14 A13 A12 A11 A10 A9 A8
AD7 AD6 AD5 AD4 AD3 AD2 AD1 AD0
R/W AS
EXPANDED
PARALLEL I/O EQUIVALENT TO MC68HC24

Appendix D

Forward Kinematics Lookup table

```
Forward Kinematics of THING
chi = swing angle
theta1 = servo 1 = shoulder angle
theta2 = servo 2
gamma = elbow angle
(x,y,z) = cartesian coordinates of the foot wrt the origin of the leg
                          ******************
xi: -45  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (-1.1, -0.4, -8.0)
xi: -45  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (-0.4, 0.3, -8.4)
xi: -45  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (-0.7, 0.0, -7.6)
xi: -45  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.1, 0.6, -7.9)
xi: -45  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (0.6, 1.3, -8.0)
xi: -45  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (1.2, 1.9, -8.0)
xi: -45  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (1.8, 2.5, -7.9)
xi: -45  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (-0.5, 0.2, -7.1)
xi: -45  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (0.1, 0.8, -7.3)
xi: -45  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (0.7, 1.4, -7.5)
xi: -45  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (1.4, 2.1, -7.5)
xi: -45  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (2.0, 2.7, -7.3)
xi: -45  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (2.5, 3.2, -7.1)
xi: -45  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (-0.4, 0.3, -6.5)
xi: -45  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (0.1, 0.8, -6.7)
xi: -45  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (0.7, 1.4, -6.8)
xi: -45  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (1.4, 2.1, -6.8)
xi: -45  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (2.0, 2.7, -6.7)
xi: -45  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (2.6, 3.3, -6.4)
xi: -45  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (1.2, 2.0, -6.2)
xi: -45  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (1.9, 2.6, -6.1)
xi: -45  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (2.6, 3.3, -5.8)
xi: -45  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (2.4, 3.1, -5.2)
                          ******************
xi: -27  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (-0.9, -0.7, -8.0)
xi: -27  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (-0.5, 0.2, -8.4)
xi: -27  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (-0.7, -0.3, -7.6)
xi: -27  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.3, 0.5, -7.9)
xi: -27  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (0.1, 1.4, -8.0)
xi: -27  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (0.6, 2.2, -8.0)
xi: -27  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (0.9, 2.9, -7.9)
xi: -27  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (-0.5, 0.0, -7.1)
xi: -27  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.2, 0.7, -7.3)
xi: -27  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (0.2, 1.6, -7.5)
xi: -27  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (0.7, 2.4, -7.5)
xi: -27  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (1.1, 3.2, -7.3)
xi: -27  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (1.4, 3.8, -7.1)
xi: -27  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (-0.5, 0.2, -6.5)
xi: -27  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.2, 0.8, -6.7)
xi: -27  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (0.2, 1.6, -6.8)
xi: -27  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (0.7, 2.4, -6.8)
xi: -27  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (1.1, 3.2, -6.7)
xi: -27  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (1.4, 3.9, -6.4)
xi: -27  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (0.6, 2.2, -6.2)
xi: -27  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (1.0, 3.1, -6.1)
xi: -27  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (1.4, 3.9, -5.8)
xi: -27  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (1.3, 3.7, -5.2)
                          ******************
xi:  -9  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (-0.7, -1.0, -8.0)
xi:  -9  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (-0.5, 0.0, -8.4)
xi:  -9  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (-0.6, -0.5, -7.6)
xi:  -9  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.4, 0.4, -7.9)
xi:  -9  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-0.3, 1.4, -8.0)
xi:  -9  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-0.1, 2.3, -8.0)
xi:  -9  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (0.0, 3.1, -7.9)
xi:  -9  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (-0.5, -0.1, -7.1)
xi:  -9  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.4, 0.7, -7.3)
xi:  -9  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-0.3, 1.6, -7.5)
xi:  -9  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-0.1, 2.5, -7.5)
```

```
xi:  -9  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (0.0, 3.3, -7.3)
xi:  -9  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (0.1, 4.0, -7.1)
xi:  -9  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (-0.5, 0.0, -6.5)
xi:  -9  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.4, 0.7, -6.7)
xi:  -9  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-0.3, 1.6, -6.8)
xi:  -9  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-0.1, 2.5, -6.8)
xi:  -9  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (0.0, 3.4, -6.7)
xi:  -9  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (0.2, 4.2, -6.4)
xi:  -9  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-0.1, 2.3, -6.2)
xi:  -9  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (0.0, 3.3, -6.1)
xi:  -9  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (0.1, 4.1, -5.8)
xi:  -9  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (0.1, 3.9, -5.2)
                      ******************
xi:   9  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (-0.3, -1.1, -8.0)
xi:   9  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (-0.5, -0.2, -8.4)
xi:   9  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (-0.4, -0.6, -7.6)
xi:   9  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.5, 0.3, -7.9)
xi:   9  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-0.7, 1.2, -8.0)
xi:   9  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-0.8, 2.1, -8.0)
xi:   9  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-1.0, 2.9, -7.9)
xi:   9  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (-0.5, -0.3, -7.1)
xi:   9  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.6, 0.5, -7.3)
xi:   9  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-0.7, 1.4, -7.5)
xi:   9  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-0.9, 2.3, -7.5)
xi:   9  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-1.0, 3.2, -7.3)
xi:   9  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-1.1, 3.9, -7.1)
xi:   9  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (-0.5, -0.1, -6.5)
xi:   9  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.6, 0.6, -6.7)
xi:   9  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-0.7, 1.4, -6.8)
xi:   9  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-0.9, 2.3, -6.8)
xi:   9  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-1.0, 3.2, -6.7)
xi:   9  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-1.1, 4.0, -6.4)
xi:   9  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-0.8, 2.2, -6.2)
xi:   9  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-1.0, 3.1, -6.1)
xi:   9  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-1.1, 4.0, -5.8)
xi:   9  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-1.1, 3.8, -5.2)
                      ******************
xi:  27  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (0.0, -1.2, -8.0)
xi:  27  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (-0.4, -0.3, -8.4)
xi:  27  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (-0.2, -0.7, -7.6)
xi:  27  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.6, 0.1, -7.9)
xi:  27  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-1.0, 0.9, -8.0)
xi:  27  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-1.5, 1.8, -8.0)
xi:  27  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-1.8, 2.5, -7.9)
xi:  27  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (-0.3, -0.4, -7.1)
xi:  27  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.7, 0.3, -7.3)
xi:  27  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-1.1, 1.1, -7.5)
xi:  27  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-1.6, 1.9, -7.5)
xi:  27  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-1.9, 2.7, -7.3)
xi:  27  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-2.3, 3.3, -7.1)
xi:  27  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (-0.4, -0.3, -6.5)
xi:  27  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.7, 0.3, -6.7)
xi:  27  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-1.1, 1.1, -6.8)
xi:  27  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-1.6, 2.0, -6.8)
xi:  27  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-2.0, 2.8, -6.7)
xi:  27  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-2.3, 3.5, -6.4)
xi:  27  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-1.5, 1.8, -6.2)
xi:  27  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-1.9, 2.6, -6.1)
xi:  27  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-2.3, 3.4, -5.8)
xi:  27  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-2.2, 3.2, -5.2)
                      ******************
xi:  45  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (0.4, -1.1, -8.0)
xi:  45  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (-0.3, -0.4, -8.4)
xi:  45  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (0.0, -0.7, -7.6)
xi:  45  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.6, -0.1, -7.9)
```

```
xi:  45  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-1.3, 0.6, -8.0)
xi:  45  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-1.9, 1.2, -8.0)
xi:  45  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-2.5, 1.8, -7.9)
xi:  45  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (-0.2, -0.5, -7.1)
xi:  45  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.8, 0.1, -7.3)
xi:  45  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-1.4, 0.7, -7.5)
xi:  45  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-2.1, 1.4, -7.5)
xi:  45  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-2.7, 2.0, -7.3)
xi:  45  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-3.2, 2.5, -7.1)
xi:  45  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (-0.3, -0.4, -6.5)
xi:  45  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.8, 0.1, -6.7)
xi:  45  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-1.4, 0.7, -6.8)
xi:  45  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-2.1, 1.4, -6.8)
xi:  45  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-2.7, 2.0, -6.7)
xi:  45  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-3.3, 2.6, -6.4)
xi:  45  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-2.0, 1.2, -6.2)
xi:  45  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-2.6, 1.9, -6.1)
xi:  45  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-3.3, 2.6, -5.8)
xi:  45  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-3.1, 2.4, -5.2)
                            ******************
xi:  63  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (0.7, -0.9, -8.0)
xi:  63  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (-0.2, -0.5, -8.4)
xi:  63  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (0.3, -0.7, -7.6)
xi:  63  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.5, -0.3, -7.9)
xi:  63  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-1.4, 0.1, -8.0)
xi:  63  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-2.2, 0.6, -8.0)
xi:  63  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-2.9, 0.9, -7.9)
xi:  63  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (0.0, -0.5, -7.1)
xi:  63  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.7, -0.2, -7.3)
xi:  63  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-1.6, 0.2, -7.5)
xi:  63  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-2.4, 0.7, -7.5)
xi:  63  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-3.2, 1.1, -7.3)
xi:  63  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-3.8, 1.4, -7.1)
xi:  63  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (-0.2, -0.5, -6.5)
xi:  63  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.8, -0.2, -6.7)
xi:  63  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-1.6, 0.2, -6.8)
xi:  63  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-2.4, 0.7, -6.8)
xi:  63  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-3.2, 1.1, -6.7)
xi:  63  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-3.9, 1.4, -6.4)
xi:  63  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-2.2, 0.6, -6.2)
xi:  63  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-3.1, 1.0, -6.1)
xi:  63  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-3.9, 1.4, -5.8)
xi:  63  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-3.7, 1.3, -5.2)
                            ******************
xi:  81  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (1.0, -0.7, -8.0)
xi:  81  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (0.0, -0.5, -8.4)
xi:  81  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (0.5, -0.6, -7.6)
xi:  81  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.4, -0.4, -7.9)
xi:  81  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-1.4, -0.3, -8.0)
xi:  81  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-2.3, -0.1, -8.0)
xi:  81  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-3.1, 0.0, -7.9)
xi:  81  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (0.1, -0.5, -7.1)
xi:  81  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.7, -0.4, -7.3)
xi:  81  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-1.6, -0.3, -7.5)
xi:  81  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-2.5, -0.1, -7.5)
xi:  81  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-3.3, 0.0, -7.3)
xi:  81  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-4.0, 0.1, -7.1)
xi:  81  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (0.0, -0.5, -6.5)
xi:  81  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.7, -0.4, -6.7)
xi:  81  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-1.6, -0.3, -6.8)
xi:  81  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-2.5, -0.1, -6.8)
xi:  81  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-3.4, 0.0, -6.7)
xi:  81  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-4.2, 0.2, -6.4)
xi:  81  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-2.3, -0.1, -6.2)
xi:  81  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-3.3, 0.0, -6.1)
```

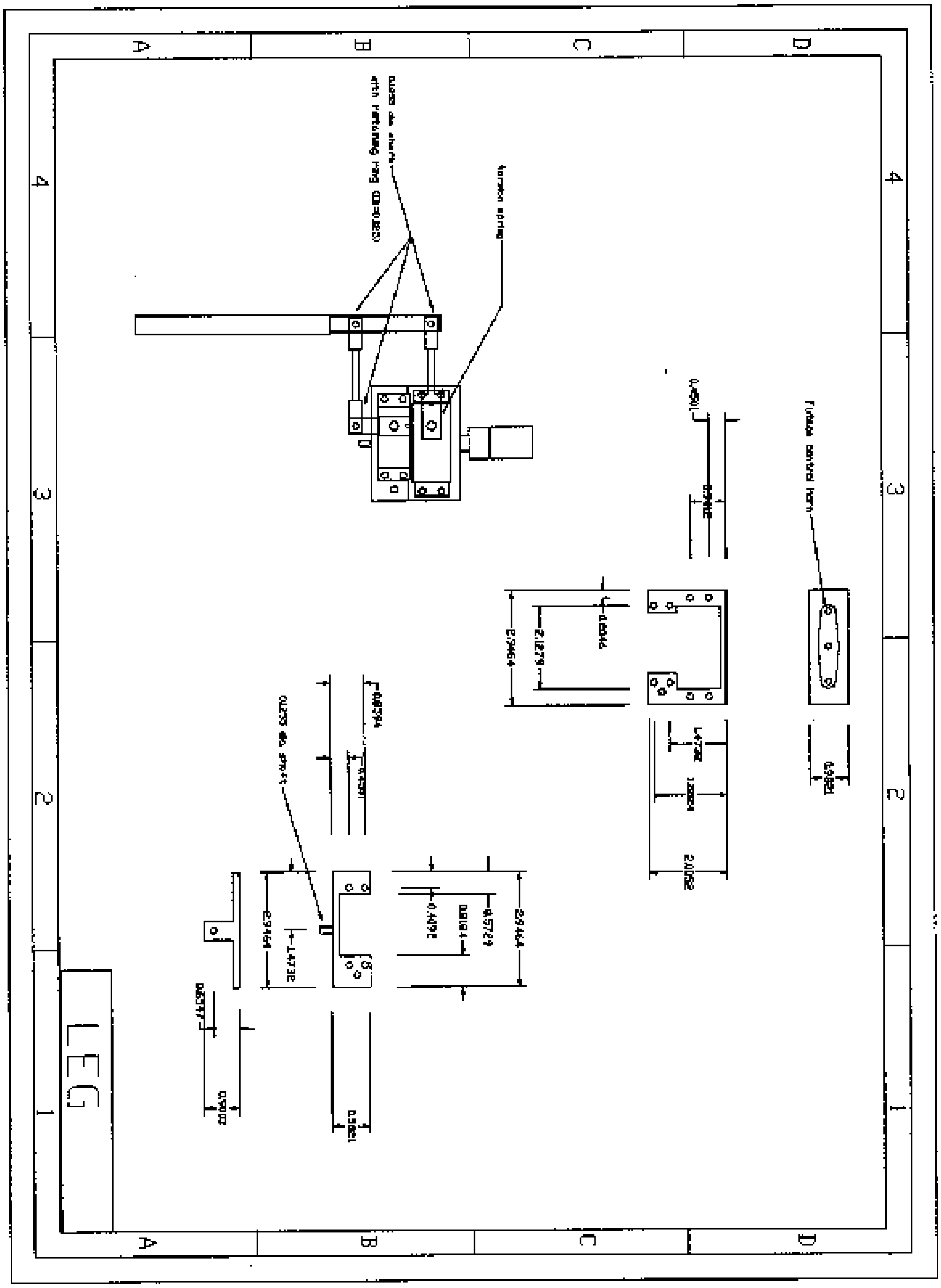
```
xi:  81  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-4.1, 0.1, -5.8)
xi:  81  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-3.9, 0.1, -5.2)
                 ******************
xi:  99  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (1.1, -0.3, -8.0)
xi:  99  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (0.2, -0.5, -8.4)
xi:  99  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (0.6, -0.4, -7.6)
xi:  99  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.3, -0.5, -7.9)
xi:  99  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-1.2, -0.7, -8.0)
xi:  99  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-2.1, -0.8, -8.0)
xi:  99  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-2.9, -1.0, -7.9)
xi:  99  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (0.3, -0.5, -7.1)
xi:  99  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.5, -0.6, -7.3)
xi:  99  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-1.4, -0.7, -7.5)
xi:  99  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-2.3, -0.9, -7.5)
xi:  99  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-3.2, -1.0, -7.3)
xi:  99  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-3.9, -1.1, -7.1)
xi:  99  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (0.1, -0.5, -6.5)
xi:  99  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.6, -0.6, -6.7)
xi:  99  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-1.4, -0.7, -6.8)
xi:  99  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-2.3, -0.9, -6.8)
xi:  99  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-3.2, -1.0, -6.7)
xi:  99  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-4.0, -1.1, -6.4)
xi:  99  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-2.2, -0.8, -6.2)
xi:  99  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-3.1, -1.0, -6.1)
xi:  99  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-4.0, -1.1, -5.8)
xi:  99  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-3.8, -1.1, -5.2)
                 ******************
xi: 117  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (1.2, 0.0, -8.0)
xi: 117  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (0.3, -0.4, -8.4)
xi: 117  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (0.7, -0.2, -7.6)
xi: 117  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (-0.1, -0.6, -7.9)
xi: 117  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-0.9, -1.0, -8.0)
xi: 117  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-1.8, -1.5, -8.0)
xi: 117  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-2.5, -1.8, -7.9)
xi: 117  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (0.4, -0.3, -7.1)
xi: 117  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.3, -0.7, -7.3)
xi: 117  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-1.1, -1.1, -7.5)
xi: 117  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-1.9, -1.6, -7.5)
xi: 117  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-2.7, -1.9, -7.3)
xi: 117  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-3.3, -2.3, -7.1)
xi: 117  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (0.3, -0.4, -6.5)
xi: 117  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.3, -0.7, -6.7)
xi: 117  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-1.1, -1.1, -6.8)
xi: 117  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-2.0, -1.6, -6.8)
xi: 117  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-2.8, -2.0, -6.7)
xi: 117  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-3.5, -2.3, -6.4)
xi: 117  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-1.8, -1.5, -6.2)
xi: 117  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-2.6, -1.9, -6.1)
xi: 117  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-3.4, -2.3, -5.8)
xi: 117  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-3.2, -2.2, -5.2)
                 ******************
xi: 135  theta1:  45  theta2: -45  gamma: 69.3  (x,y,z): (1.1, 0.4, -8.0)
xi: 135  theta1:  45  theta2: -27  gamma: 59.5  (x,y,z): (0.4, -0.3, -8.4)
xi: 135  theta1:  63  theta2: -45  gamma: 85.9  (x,y,z): (0.7, 0.0, -7.6)
xi: 135  theta1:  63  theta2: -27  gamma: 76.9  (x,y,z): (0.1, -0.6, -7.9)
xi: 135  theta1:  63  theta2:  -9  gamma: 67.6  (x,y,z): (-0.6, -1.3, -8.0)
xi: 135  theta1:  63  theta2:   9  gamma: 58.6  (x,y,z): (-1.2, -1.9, -8.0)
xi: 135  theta1:  63  theta2:  27  gamma: 50.9  (x,y,z): (-1.8, -2.5, -7.9)
xi: 135  theta1:  81  theta2: -45  gamma: 102.5  (x,y,z): (0.5, -0.2, -7.1)
xi: 135  theta1:  81  theta2: -27  gamma: 94.4  (x,y,z): (-0.1, -0.8, -7.3)
xi: 135  theta1:  81  theta2:  -9  gamma: 85.5  (x,y,z): (-0.7, -1.4, -7.5)
xi: 135  theta1:  81  theta2:   9  gamma: 76.5  (x,y,z): (-1.4, -2.1, -7.5)
xi: 135  theta1:  81  theta2:  27  gamma: 68.2  (x,y,z): (-2.0, -2.7, -7.3)
xi: 135  theta1:  81  theta2:  45  gamma: 61.2  (x,y,z): (-2.5, -3.2, -7.1)
xi: 135  theta1:  99  theta2: -45  gamma: 118.9  (x,y,z): (0.4, -0.3, -6.5)
```

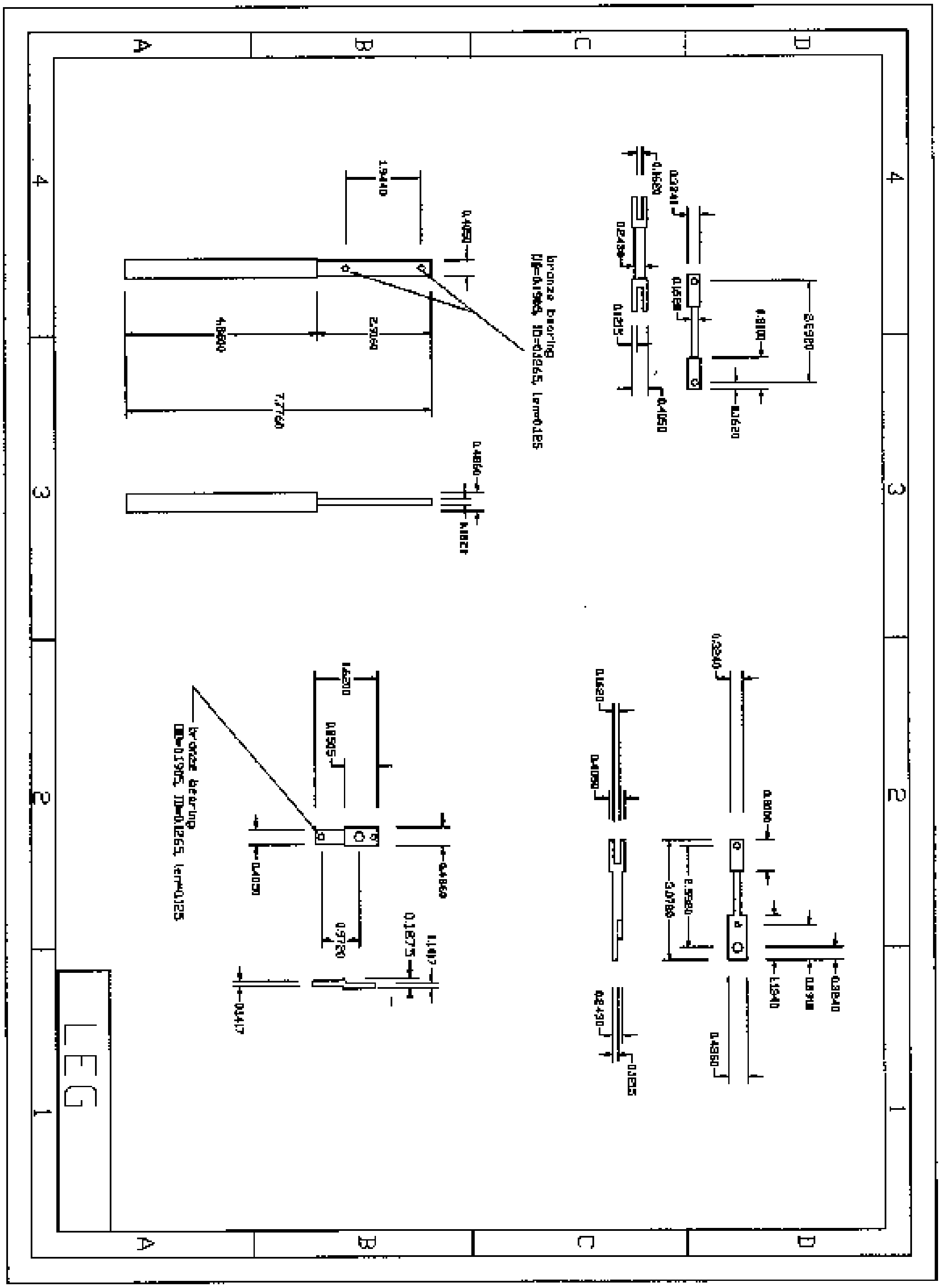
```
xi: 135  theta1:  99  theta2: -27  gamma: 111.9  (x,y,z): (-0.1, -0.8, -6.7)
xi: 135  theta1:  99  theta2:  -9  gamma: 103.5  (x,y,z): (-0.7, -1.4, -6.8)
xi: 135  theta1:  99  theta2:   9  gamma: 94.5  (x,y,z): (-1.4, -2.1, -6.8)
xi: 135  theta1:  99  theta2:  27  gamma: 85.6  (x,y,z): (-2.0, -2.7, -6.7)
xi: 135  theta1:  99  theta2:  45  gamma: 77.6  (x,y,z): (-2.6, -3.3, -6.4)
xi: 135  theta1: 117  theta2:   9  gamma: 112.4  (x,y,z): (-1.2, -2.0, -6.2)
xi: 135  theta1: 117  theta2:  27  gamma: 103.1  (x,y,z): (-1.9, -2.6, -6.1)
xi: 135  theta1: 117  theta2:  45  gamma: 94.2  (x,y,z): (-2.6, -3.3, -5.8)
xi: 135  theta1: 135  theta2:  45  gamma: 110.8  (x,y,z): (-2.4, -3.1, -5.2)
```

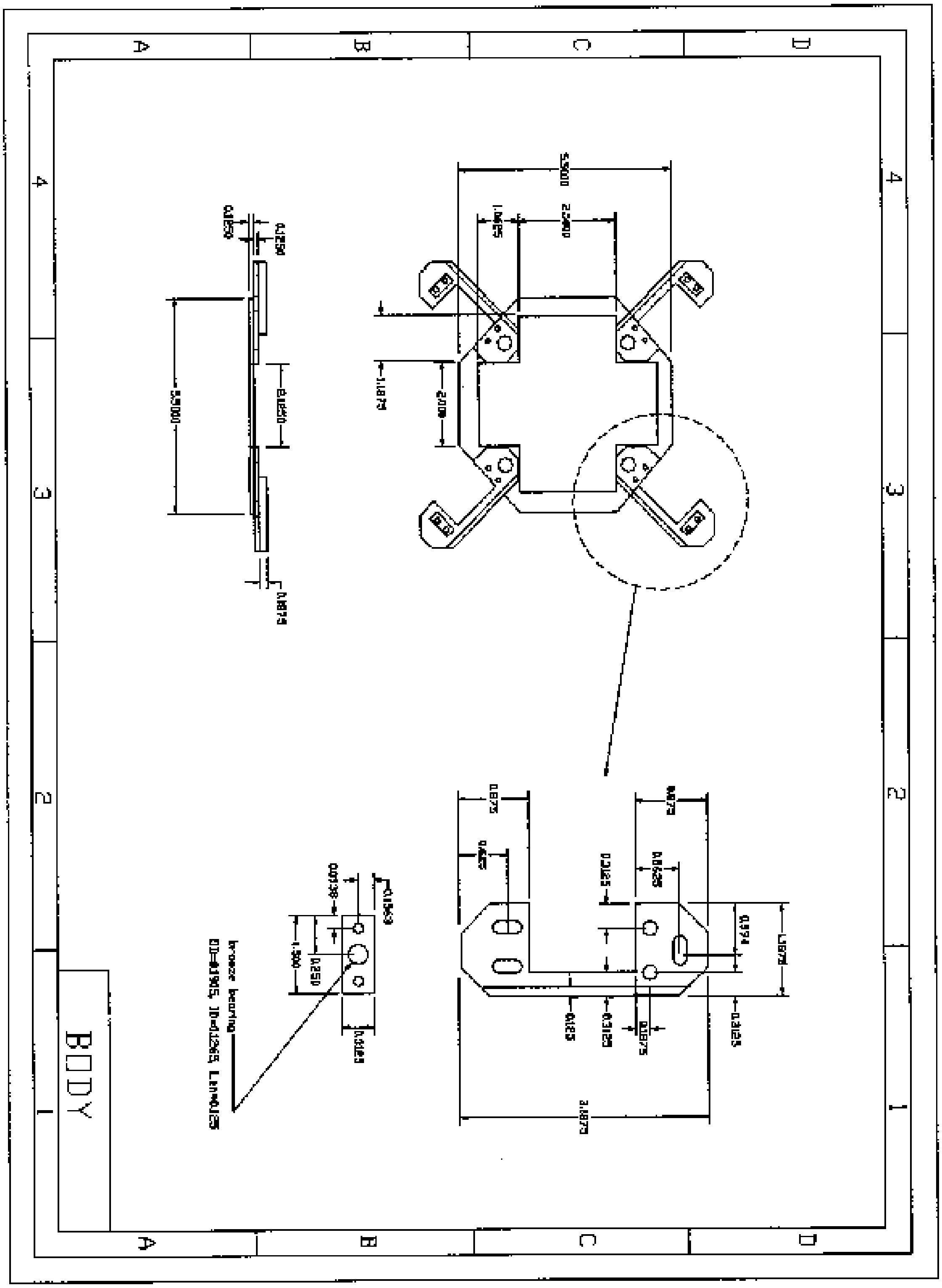
Appendix E

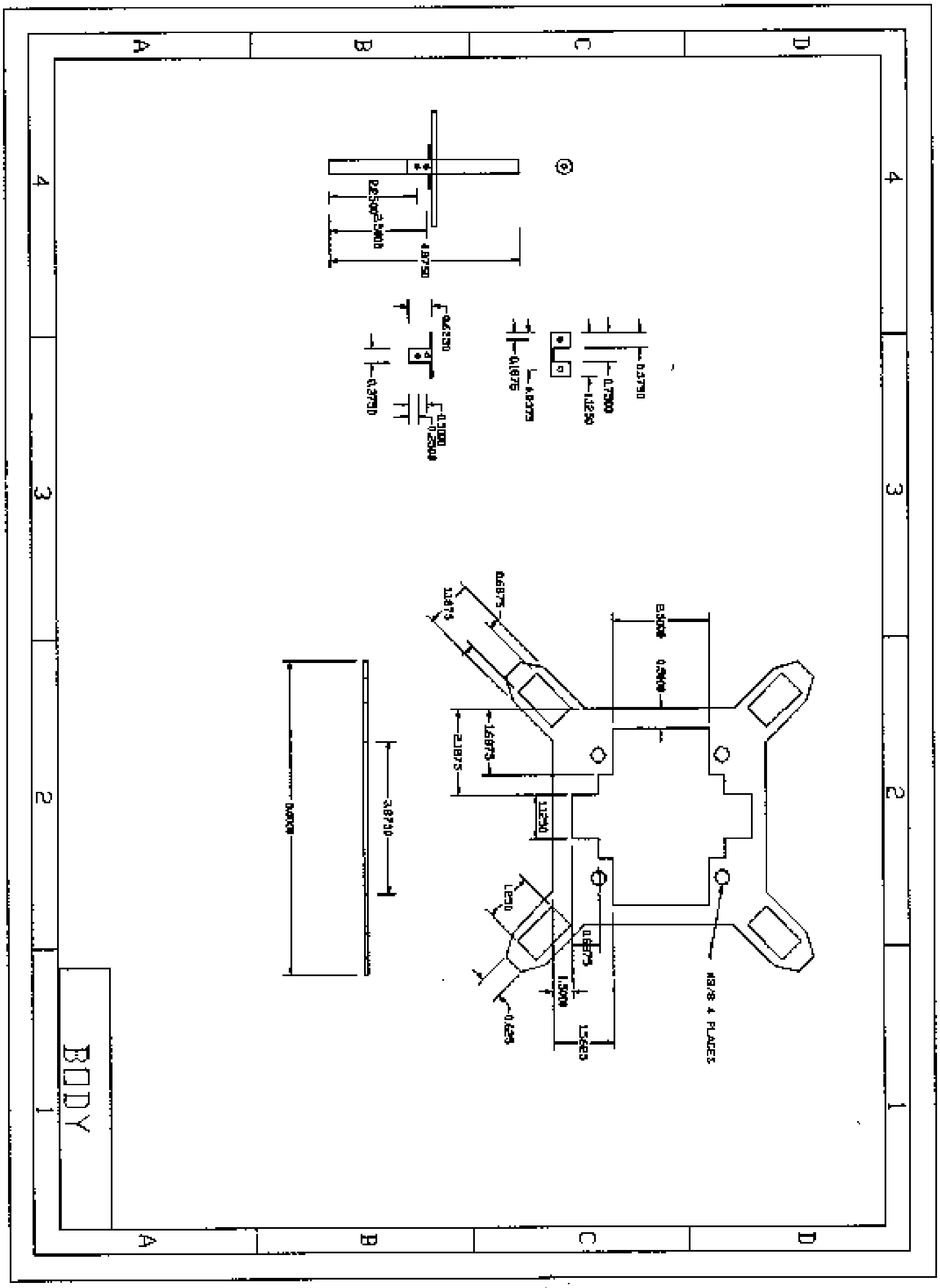
Mechanical Drawings

- Leg
- Body


LEG
torsion spring
Futaba control horn


LEG


BODY


BODY
4.8750
6.0000
3.8750
1.1250
2.1875
1.6875
1.5000
1.5625
0.6875
0.3750