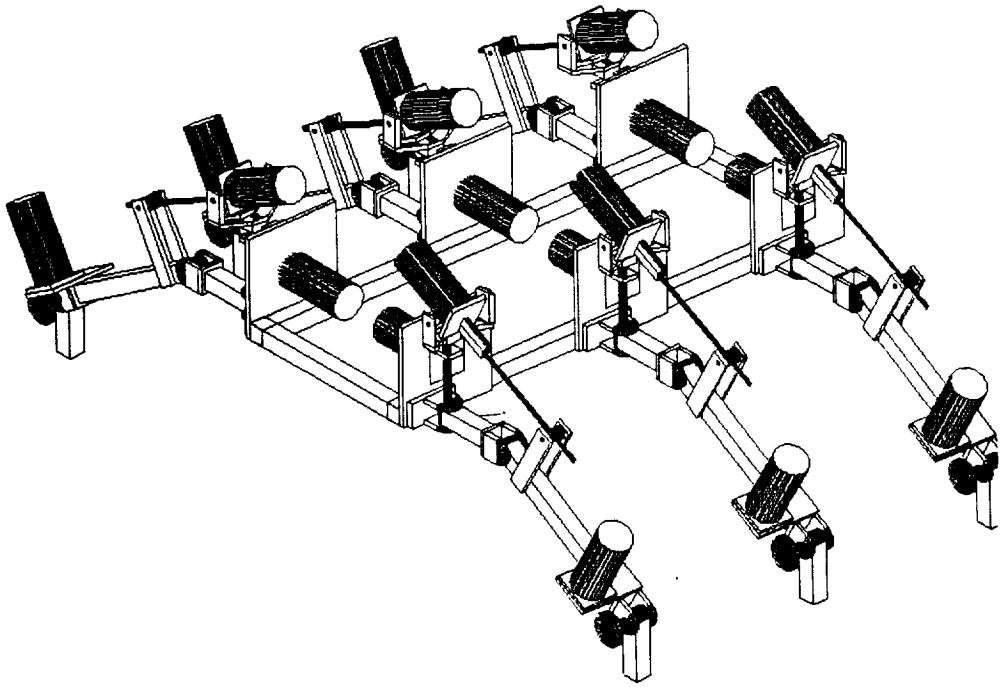
NASw-4435

Human-Like Robots for Space and Hazardous Environments

USRA / NASA Advanced Design Team
Kansas State University

Mechanical Engineering, Electrical and Computer Engineering,
Computer and Information Sciences, and Psychology



Faculty Advisors:
Dr. Allen Cogley, Dr. David Gustafson,
Dr. Jim Devault, and Dr. Warren White

Table of Contents

Figures and Tables

Project Goals

Three Year

The three year goal for the Kansas State USRA/NASA Senior Design team is to design and build a walking autonomous robotic rover. The rover should be capable of rough terrain crossing, traversing human made obstacles (such as stairs and doors), and moving through human and robot occupied spaces without collision. The rover is also to evidence considerable decision making ability, navigation and path planning skills.

When began, this project had the title "Human Like Robots in Space and Hazardous Environments". Within that scope the design team was free to choose what it felt the project's goals should be. The first group of criteria the design team picked was that the project should be some form of mobile robot, that it be as autonomous and intelligent as possible, and that some form of the robot actually be constructed. These goals were chosen as being the most interesting to the group as a whole. Several variants of these ideas were discussed:

Hazardous waste cleanup: The chernobyl accident demonstrated several instances where remotely operated or robotically controlled earth movers would have been useful (or life saving). This idea was discarded after noting that the large equipment needed was far beyond our budget.

Planetary Rover: The ability to range widely and collect samples, data, and pictures would be of benefit for any future space exploration missions. A mobile robot was envisioned that could traverse rough terrain, carrying a variety of sensors and instruments, with enough intelligence to travel without a human operator.

Hazardous waste site scout: A mobile robot that could move freely about a disaster site, allowing human operators to assess the damage without endangering themselves was another suggested idea.

The last two ideas were combined into the current three year goal when it was noted that the same abilities to cross rough terrain on other planets would allow a rover to explore hazardous waste sites. It was further noted that if the rover could also open doors, and climb stairs it would be able to access almost all indoor areas that humans could reach, and those abilities became part of the goal. The choice for a legged rover design seemed best for maximizing rough terrain crossing ability. The decision for supporting an autonomous robot as opposed to a tele-operated design was to allow a rover to operate at great distances from Earth. The high intelligence factor was

also supported as it would allow an operator to control several rovers.

First Year

The job of the design team the first year was to pick a project goal, and then attempt to define just what needed to be done to reach that goal. It became apparent rather quickly that the design group lacked practical experience building robots, and without that experience any design done would be flawed and fail. So the first year goal was set as building a working prototype of the walking robot rover for the purpose of learning as much as possible about mobile robot design.

The main criteria for the prototype were as follows:

Six legs: This was to begin the exploration into legged robot design. The choice of six legs allowed for a stable walking platform and a simpler overall control.

Three joints per leg: Three joints are the minimum needed for the robot to walk without its feet sliding or slipping.

On board intelligence: By restricting all processors to be on board the robot, the prototype would force realistic control schemes to be investigated.

Off board power: It was soon discovered that the cost of high efficiency electric motors needed to operate on battery power were out of our price range.

Obstacle climbing: The prototype needed some target obstacle to test its terrain handling abilities. Stairs are a readily available obstacle in the indoor laboratory environment, and exemplify a typical obstacle needed to meet the criteria for the three year goal.

Second Year

The year-one USRA/NASA team designed and built a six-legged rover that operated through manual controls. Automation designs were not implemented and the chassis was unacceptable for the long duration travel. The year-two team has redesigned and implemented extensive changes to the mechanical components and chassis. Automation hardware has also been redesigned and implemented in hardware for the first time. Microprocessors, motor driver chips, communication protocols and sensor circuits have been integrated with the mechanical systems. The final contribution to automatic control and rover independence is the software or microprocessors programing. The first automated prototype is complete and being analyzed for performance.

Project Organization

The USRA/NASA Design Team has about 20 members, 7 supporting faculty, using 4

machine shops, 2 electronics shops, and dozens of computers in several labs. The organization, managerial and communication aspects of this project are not trivial. To control some of the chaos, the design team created several lead positions and formed itself into subgroups. The subgroups were chosen to be:

Mechanical Group: Dealt with all aspects of leg, joint, and chassis design, material selection and machining.

Hardware Group: Dealt with all electrical systems, including microprocessor selection, motors, motor drivers, sensor design and construction, as well as power supply to the robot.

Software Group: Writing all microprocessor programs for logical control of all components of the robot.

The positions on the team were Project Lead Engineer, Software Group Leader, Hardware Group Leader, Mechanical Group Leader, and the Graduate Assistant. The responsibilities of the Lead Engineer were to coordinate the design process between the groups, aid in the design where difficulties arise, and promote the project's progress. The group leaders were responsible for helping their group members with the design and construction of their prototype sections, requesting needed supplies, equipment, and other support. The Graduate Assistant was given the responsibility of interfacing between the students and the faculty, handling the team's budget, and purchases, and aiding in acquiring needed equipment and supplies.

During the academic year of 1993-94 the positions were filled as follows:

- Lead Engineer - Jeremy Whitt (ME)
- Mechanical Leader - Jason Jacobson (ME)
- Hardware Leader - Marcus Fechter (EE)
- Software Leader - Doug Chapman (CIS)
- GTAs - Poonacha Machaiah (CIS)
  - Mac Crossett (ME)

I. Mechanics

1.1 Introduction

The purpose of this report is to summarize the progress of the Mechanics Group as part of the NASA/USRA Advanced Design Project for Fall 1994. The Mechanics Group objectives were to re-engineer the robot chassis and leg designs, and manufacture an operational one-leg prototype by the end of the Fall 93 semester. It was necessary to redesign some of the gear drives, leg lengths, chassis structure and drive mechanisms to meet the obstacle maneuverability goals set by the team in the previous year. The new designs had to overcome problems of structural rigidity, limited mechanical advantage and very high frictional losses. The Mechanics Group decided to form subgroups and redesign the robot chassis and leg simultaneously.

1.2 Year One Overview

The job of the Mechanics group in the first year was to develop a preliminary prototype. The main criteria for the prototype were as follows:

- Six legs: This was to begin the exploration into legged robot design. The choice of six legs allowed for a stable walking platform and a simpler overall control.
- Three joints per leg: Three joints are the minimum needed for the robot to walk without its feet sliding or slipping.
- Obstacle Climbing Considerations: The prototype needed some obstacle maneuverability that would be taken into consideration early in the design phase.

1.2.1 Leg Design

In the design of the legs of the robot, certain requirements had to be considered. First, the leg had to allow the robot to move at the original designed speed of one foot per second. This speed was set arbitrarily to insure that the robot would have a fast response time in real-life applications. Next, the legs had to allow the robot to maneuver around or over obstacles in its path. The design also had to provide the robot with the ability to climb a set of stairs.

Certain design parameters had then to be considered that would satisfy these requirements. These parameters were the types of motion the legs would use, and the geometry of the legs.

The types of motion that were considered for the legs were linear actuation and rotational actuation. Linear actuation would consist of leg sections that would slide inside one another to extend and contract the leg.

Rotational actuation would consist of leg sections that would be attached through joints which would swing through an arc to provide leg movement. Of the two types of motion, rotational actuation was chosen because it provided a less complicated leg design.

After the type of motion was chosen, a base coordinates system for the robot was defined and is given in Figure I.1. With the robot standing up-right, the origin of this coordinate system was chosen to pass through the center of the robot. The vertical axis of the coordinate system is defined as the Z-axis. Rotation about this axis is defined as Alpha rotation. Right to left, or from side to side of the robot is the Y-axis. Rotation about the Y-axis is defined as Gamma rotation. Lengthwise along the robot is the X-axis. Rotation about the X-axis is defined as Beta rotation.

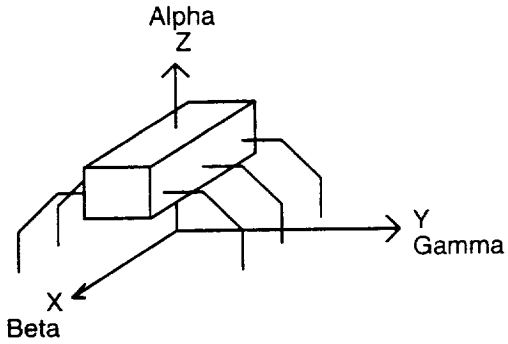


Figure I.1
Coordinate Axis

It was then necessary to consider more detailed aspects of the leg geometry including: the number of leg joints and the type of rotation at each joint. It was decided to use a

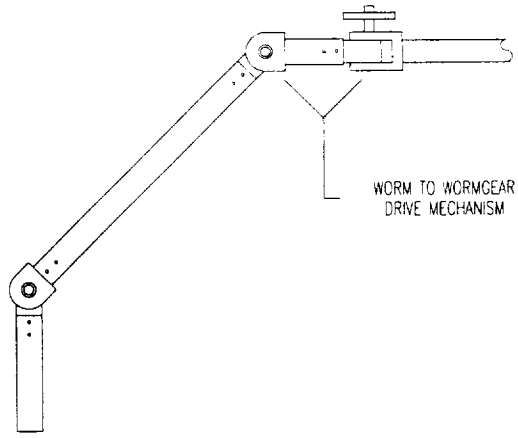


Figure I.2
Old Leg Design

three-jointed leg with one Alpha rotation joint and two Beta rotation joints (See Figure I.2).

These three drives provided each leg with three degrees of freedom. Each worm-gear drive uses a 30:1 gear reduction. Unfortunately, the year-one leg design allowed for relative motion between the motor mounts and the robot for each drive. This relative motion led to frictional losses as high as 75% in each leg.

1.2.2 Chassis Design

The year one robot used a skeletal chassis constructed of several hallow aluminum sections bolted together into a rectangular frame. The electronics hardware components were mounted to the topside of the chassis to provide easy access for repair or replacement.

1.3 Year Two

To improve upon the year-one mechanical design, the Mechanics Group decided to

perform some preliminary testing to highlight the robot's strengths and weaknesses.

After running each joint through its full range of motion several times, the group saw many areas where re-engineering was necessary.

First, the Beta 1 drive and mounting system had the following problems:

- Need for perfect alignment between the worm gears to avoid high friction losses.
- Efficiencies as low as 25% due relative motion in the worm gears and the absence of any type of lubrication.
- Slippage in the Beta 1 joint due to the loosening of the set screw caused by vibrations.
- Play in the motor shaft allowed movement in the Beta 1 joint causing the robot to rock back and forth.
- Lack of modularity in the drive and mounting system made it very difficult to assemble and repair.

Second, the chassis had the following problems:

- Low chassis rigidity.
- Bolt holes allow for too much relative motion.
- No place to rigidly attach the Alpha mount.
- Bolted intersections act like high friction joints.
- Chassis does not shield hardware components.

Third, the Alpha drive and mounting system had the following problems:

- Need for perfect alignment between the worm gears to avoid high friction losses.
- Efficiencies as low as 25% due relative motion in the worm gears and the absence of any type of lubrication.
- Slippage in the Alpha joint due to the loosening of the set screw caused by vibrations.
- Play in the motor shaft allowed movement in the Alpha joint causing the robot to rock back and forth.
- Lack of modularity in the drive and mounting system made it very difficult to assemble and repair.

Fourth, the dimensional problems in the year-one design were the following:

- Chassis width of 14" made it difficult for the robot to maneuver through doorways.
- Alpha, Beta 1, and Beta 2 leg lengths were not conducive to climbing a 9" stair.

- No uniformity in any of the part dimensions.

The Mechanics Group, therefore, decided to prioritize its responsibilities and develop a working time line under which design alterations could be addressed. The group decided upon the following major re-engineering tasks:

- Redesign of the entire chassis.
- Redesign of the Beta 1 drive and mounting system.
- Redesign of the Alpha drive and mounting system.
- Optimization of the leg lengths for stair climbing activities.
- Construct a one-leg prototype.
- Perform repeatability testing on the one-leg prototype to highlight possible design alterations.
- Construct a working chassis with modular leg mounts.
- Allow for hardware integration.
- Assemble the new robot.

The Mechanics Group decided to spend the first three weeks of year two developing and analyzing possible solutions to the problems discovered in the year one design. In order to maximize the available time, the group decided to break into smaller subgroups and work on several different tasks in parallel.

1.3.1 Chassis

The robot chassis had difficulties with structural rigidity and weight. The Chassis Group was formed to research possible material and design alternatives for the chassis reconstruction. After consulting with several team members from the previous year, it was decided to research the possibility of using Hexcell's Aluminum Honeycomb board. Although the honeycomb board had sufficient mechanical properties to accommodate the chassis design, it was relatively expensive and difficult to mount the motor to its thin, flat exterior. Therefore the honeycomb material was eliminated as an alternative.

After, consulting Dr. Hugh Walker and Dr. Byron Jones (Mechanical Engineering professors at Kansas State University),it was decided to construct the chassis from one-eighth inch thickness square aluminum tubing, which is 36 inches long. The alpha motors are mounted directly through the wall. It was decided to make

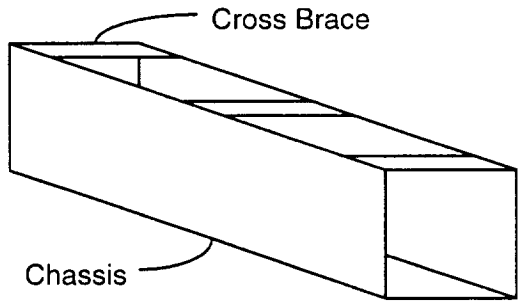


Figure I.3
Schematic of Chassis

the chassis six inches wide (from outside to outside) and thirty six inches long, to make the robot relatively easy to maneuver through doorways and corridors. Furthermore, the new chassis design made it possible to shield many of the hardware components inside the robot's body during operation. For this reason, it was decided to remove the top section of the aluminum tubing and replace it with hardened steel cross braces to simulate a closed channel (see Figure I.3).

These cross braces served to reduce the angle of twist per unit length during walking motion or stair climbing activities. This new design reduced the relative motion in the Alpha mounts by allowing the Alpha motors to be mounted directly to the inside wall of the chassis. Mounting the Alpha motors inside the chassis also provides the robot with a lower center of gravity to prevent tipping when climbing stairs.

The new design also allowed the hardware components to be mounted on the inside the chassis to provide limited shielding while maintaining the option of easy access in case of repairs.

1.3.2 Beta 1 Drive

The group explored several options for modifying the existing worm-gear drive mechanism to address some of the problems with the year-one design which have already been discussed . However, after researching several journals on robotics design and consulting with Dr. Warren White and Dr. Hugh Walker, the group decided to go with a new linear drive mechanism. The new design

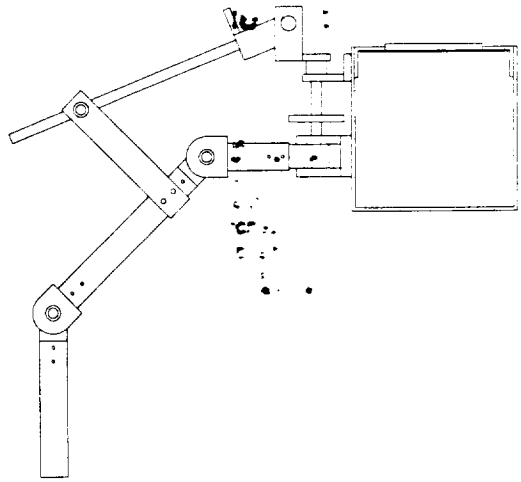


Figure I.4
New Leg Design

used an all-thread shaft to activate a linear drive collar (see Figure I.4).

This linear drive collar was attached to a lever arm on the Beta 1 leg. As the shaft spins, the moment arm moves the Beta 1 leg to the desired position. This new Beta 1 drive mechanism sacrificed speed for strength in the joint and also eliminated much of the relative motion in the drive.

Theoretically, the linear screw drive would require lower motor torque to function under normal operating conditions. Table I.1 represents the nominal axial force and motor torque requirements for the full angular range of the beta-1 joint (derivations of the equations used to generate the tabled data have been provided in the Appendix I.A). After reviewing the data in Table I.1, the Mechanics Group found a maximum stalling torque exceeding 14 (in-lb) was required to avoid failure in the linear screw drive.

Beta1 Torque Requirements

Parameters

	variables
Beta 1 Lever angle =	60 degrees
BetaMot=	125 degrees
LenBeta_1 =	8 inches
LenLever =	4.125 inches
LenMotor =	6.1 inches
Chassis & Legs ~	50 lbs
Motors =	31 lbs
Power Equipment =	0 lbs
Other Equipment ~	5 lbs

(Other includes wire interfacing, circuit boards, ets.)

friction loss calculation

μ =	0.74
Lead=	0.083333
Radius=	0.1875
Phi=	0.63707
Theta=	0.070618

Results:
Max Torque Required:
14.64 in-lb

Weight = 86 lbs
(sum of four weight groups listed above)

MaxForce = 43 lbs
(MaxForce is calculated as 1/2 of Weight)

Tabular Data

This is the Fs values calculated from the formula on the graph

Indicates intermediate calculation steps

Force Required by the Screw Drive as a Function of Beta1 Angle

Beta_1 degrees	LenX inches	Torque in-lbs	F_PerpLev lbs	LenScrew inches	AngPerF degrees	Fscrew in-lbs	Motor Torque
30	6.928	297.91	72.22	3.61	13.99	74.43	11.94
25	7.250	311.77	75.58	3.96	7.95	76.31	12.24
20	7.518	323.25	78.36	4.32	2.50	78.44	12.58
15	7.727	332.28	80.55	4.68	2.50	80.63	12.93
10	7.878	338.77	82.13	5.04	7.14	82.77	13.28
5	7.970	342.69	83.08	5.39	11.50	84.78	13.60
0	8.000	344.00	83.39	5.74	15.63	86.60	13.89
-5	7.970	342.69	83.08	6.08	19.58	88.17	14.14
-10	7.878	338.77	82.13	6.42	23.37	89.47	14.35
-15	7.727	332.28	80.55	6.74	27.04	90.44	14.51
-20	7.518	323.25	78.36	7.06	30.60	91.04	14.60
-25	7.250	311.77	75.58	7.36	34.07	91.24	14.64
-30	6.928	297.91	72.22	7.66	37.46	90.99	14.60
-35	6.553	281.79	68.31	7.94	40.79	90.23	14.47
-40	6.128	263.52	63.88	8.20	44.07	88.91	14.26
-45	5.657	243.24	58.97	8.45	47.30	86.95	13.95
-50	5.142	221.12	53.60	8.69	50.48	84.25	13.51
-55	4.589	197.31	47.83	8.91	53.64	80.68	12.94
-60	4.000	172.00	41.70	9.12	56.76	76.07	12.20
-65	3.381	145.38	35.24	9.30	59.85	70.18	11.26
-70	2.736	117.65	28.52	9.48	62.93	62.67	10.05
-75	2.071	89.03	21.58	9.63	65.98	53.02	8.51
-80	1.389	59.73	14.48	9.77	69.01	40.44	6.49
-85	0.697	29.98	7.27	9.89	72.04	23.57	3.78
-90	0.000	0.00	0.00	9.99	75.05	0.00	0.00

This is the angle that the longest leg segment makes with the horizontal.

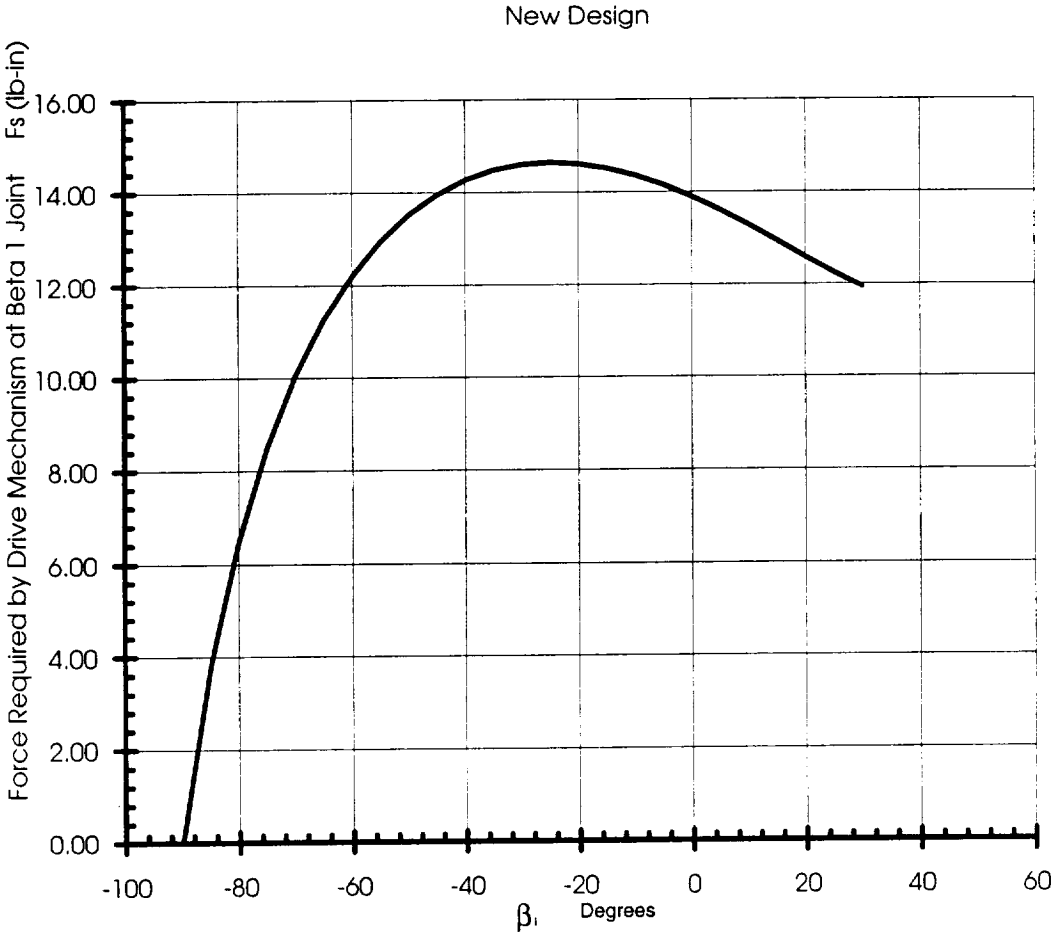
This calculation includes losses due to friction in the screw.

Please note:

These calculations reflect the values required taken at time zero plus.
Another words these are values needed to overcome STATIC forces only!

More research is underway to incorporate dynamic forces.

Force Required for Beta 1 Drive Mechanism verses Beta 1 Leg Angle



$$F_S = \frac{(L_1)(F_{MAX})COS(\beta_1)}{(L_{1L})COS\left[90 - ARCSIN\left(\frac{(L_{1M})SIN((\beta_{1M} - (\beta_{1Z} + \beta_1))}{\sqrt{L_{1M}^2 + L_{1L}^2 - 2(L_{1M})(L_{1L})COS((\beta_{1M} - (\beta_{1Z} + \beta_1)))}}\right)\right]}$$

Axial alignment of the beta-1 motor and the all-thread shaft was a critical design factor in the linear screw drive mechanism. If any significant eccentricity were experienced in the alignment of the shaft, the entire drive would be subject to binding and high friction losses. Consequently, it was necessary to design a new motor mount to ensure proper alignment of the beta-1 motor and the all-thread shaft (see drawing labeled Motor Yoke in Appendix I.A). the Motor Yoke provides the beta-1 motor with two degrees of rotational freedom to maintain the motor shaft alignment required for operation.

Therefore, the group decided to design the Beta 1 motor mount so that it would mount directly to the side of the new chassis wall. The group also designed a "u-shaped" rotational collar, which mounted axially to the Alpha mount on the external wall of the chassis. This new mounting system allowed the Beta 1 motor to rotate on the z-axis without causing the all-thread shaft to suffer torsional bending, resulting in failure of the Beta 1 drive.

Finally, thrust bushings were incorporated into the Beta 1 face plate to reduce the transference of axial thrust forces which could potentially damage the Beta 1 motor.

1.3.3 Alpha Drive

The Mechanics Group discussed the option of making the Alpha drive a linear mechanism to eliminate the relative motion between the worm gears. However, the new chassis design allowed the group to simply modify the existing Alpha mounts and attach them directly to the chassis wall. Since both worm gears were attached directly to the chassis wall, they no longer experienced the highly destructive frictional forces that

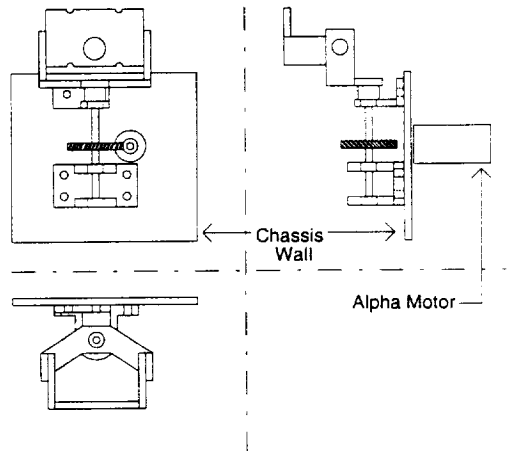


Figure I.5
Alpha Drive Mechanism

previously led to the low efficiencies in the drive mechanism (see Figure I.5).

1.3.4 Leg Lengths

A great deal of the Mechanics Group discussions dealt with the question of appropriate leg lengths for ROBOCAT. For ROBOCAT to climb stairs effectively, some design constraints were set forth. Since optimum leg lengths would vary with stair height, the group decided to optimize the leg lengths for activities on a 9" stair. For design purposes, the stair cross section was assumed to be a square.

Therefore, a working PVC model was constructed for the purpose of angle analysis. After simulating the robot's anticipated

motions during stair climbing activities, the group decided upon the appropriate Alpha, Beta 1 and Beta 2 leg lengths. They were four inches, eight inches, and six inches, respectively.

1.3.5 Assembling the Robot

Assembling a working robot in a timely fashion required the Mechanics Group to plan a schedule and meet the objectives. The group decided to first take the preliminary drawings to the machine shop and have enough parts made to construct an operational one-leg prototype for hardware and software testing. By doing so, the Mechanics Group would also be able to make any necessary final modifications before constructing the other five leg modules in a scale-up process.

After receiving the finished parts back from the machine shop, the group began to construct the working one-leg prototype. Several part modifications resulted from repeatability testing and robot weight considerations.

The Beta 1 motor mount and lever arm dimensions were changed to accommodate necessary angular rotational range of the Beta 1 joint for climbing stairs. It was also necessary to perform some minor machining operations on the Alpha and Beta 1 face plates so proper meshing of the worm gears could be achieved.

After several other small modifications the group was ready to begin the scale-up process. First, the chassis design was contracted out to Kasa. It should be noted that Kasa did a superb job of constructing the entire chassis body for the cost of materials only. The group then sent the modified drawings to the machine shop to begin parts fabrication.

After receiving the parts back from the shop, the Mechanics Group assembled the working leg units for the robot and attached them to the chassis. Through frequent communications, the Mechanics Group learned that the Software and Hardware Groups needed the one-leg prototype intact for a few extra weeks to perform valuable testing and integration operations.

Therefore, the Mechanics Group assembled the entire robot with the exception of the left center leg before beginning the hardware integration.

1.3.6 Hardware Integration

As stated before, the year one prototype progressed as far as a remote-controlled unit. Therefore, it had no rotational limit switches, potentiometers, or other sensory feed back hardware. The year two robot, however, was to be an autonomous robot with limited intelligence. Consequently, the Mechanics Group needed to design mounting systems for each piece of sensory feed back equipment needed to provide the absolute rotational position of each joint. The rotational limit switches were used to keep ROBOCAT from overextending its joints in case of a software malfunction, while the potentiometers allowed the software to keep track of the angular displacement in each section of each leg.

The Hardware Group also requested that the Mechanics Group design a mounting system for the eighteen mini boards which were to be mounted inside the chassis. In response to their request, the group decided to mount all the mini boards on a Plexiglas tray under the Alpha motors. The Plexiglas electrically insulated the mini boards, and allowed easy access for maintenance.

Finally, the Mechanics Group installed a Plexiglas covering which mounted to the chassis cross braces. The covering protects the six sensory printed circuit boards that are mounted on top of the chassis, although they are in clear view.

1.4 Future Work

A significant amount of work has been accomplished in years one and two. However, there are still tasks that require attention. For example, due to the time constraints on the project, the Mechanics Group did not redesign the Beta 2 drive. The Beta 2 drive has many of the same problems that the Beta one drive had, only on a smaller scale. Some of the options that the year three team should consider are changing the Beta 2 mounting system, using a direct Beta 2 drive, or totally redesigning the Beta 2 drive and mounting systems. The year three group should also consider running some extensive repeatability and robustness tests on each of the joints to highlight potential weaknesses.

However, the majority of the year three Mechanics Group work will involve the integration of more electronic hardware and sensor equipment into the existing mechanical design of ROBOCAT.

1.5 Acknowledgments

The present design of ROBOCAT is the product of the combined efforts of a cross-functional engineering team, countless faculty advisors, and several fine machinists. The Mechanics Group would like to take this opportunity to acknowledge all of the hard work and effort by the following people:

- Gary Thornton, ME Developmental Specialist
- Dr. Allen Cogley, ME Professor
- Dr. Warren White, ME Professor
- Dr. Hugh Walker, ME Professor
- Dr. Dave Gustafson, CIS Professor
- Dr. Jim Devault, EECE Professor
- Dr. Carl Wilson, IE Professor
- Troy Brockway, IE Machine Shop Director
- Kasa, Private Machine Shop
- Hardware Group Members
- Software Group Members

II. Hardware

2.1 Introduction

The hardware group has the responsibility of designing the electronic control system for the robot and design and construction of all the hardware components. The goals this semester were to build a one-legged prototype and a six-legged robot. The hardware group was successful in building the one-legged prototype and the six-legged robot.

The group has been responsible for designing and implementing the limit switches, optical encoder, motor driver, cabling, foot down sensor, potentiometer, motor and power system, de-multiplexer, and Printed Circuit Board (PCB). The optical encoder design was not used due to complexities of the design. The group has procured and installed all the necessary components for implementing total control of the six-legged robot. The six-legged robot was completed this semester. The sensors that will allow for obstacle avoidance will be added next semester.

2.2 Year One Summary

2.2.1 HC16 to HC11 Network

Last year's hardware network consisted of one MC68HC16EVB evaluation board and eighteen MC68HC11 mini board microprocessors. The HC16 is the central "brain" of the network. It contains the program that instructs the legs to move in a certain sequence. One HC11 is connected to each of the eighteen joints. These processors receive the instructions from the HC16 to move and then control motor driver, limit switch, potentiometer, and foot down circuits to make the joint move to the desired position. These processors also handle error conditions of each joint such as joint over-extension and obstacle detection using the sensors circuits. The eighteen HC11s are connected to the HC16 by a daisy chain network connected to the high speed serial peripheral interface(SPI) ports of the processors. This network enables the HC16 to communicate bi-directionally with the selected HC11 processor. A HC11 is selected by the HC16 by using a de-multiplexer circuit that has one output line connected to each HC11 as shown in Figure II.1.

2.2.2 HC11 Circuits

The digital and analog input/output ports of one HC11 processor are connected to a limit switch, potentiometer, and motor driver circuits for one joint. In addition, the processor for the Beta 2 joint is connected to

the foot down sensor circuit. The purpose of these circuits is to control the joint's motion, detect the joint's position, and detect any errors that arise.

2.2.2.1 Limit Switches

The purpose of the limit switch circuit is to prevent over-extension of the joint. It consists of two switches located at the limit of the joint's safe motion. The switch is connected to an analog port of the processor. When the switch is triggered, the processor immediately turns off the motor driver to stop the motor from destroying the joint.

2.2.2.2 Potentiometer

The purpose of the potentiometer circuit is to detect the angular position of the joint. The potentiometer is a variable resistance device whose resistance changes when the shaft is turned. The shaft, in this case, is attached to the axle of the joint. The sense line of the potentiometer connected to a analog input of the HC11. The HC11 reads the analog input and calculates the angle of the joint by matching the read voltage to the correct angle.

2.2.2.3 Force Sensing Resistor

The purpose of the foot down circuit is to determine if the foot is touching the ground. The circuit uses a voltage divider circuit with a variable resistance force sensing resistor(FSR), located in at the end of the leg, as one of the resistors. The FSR changes resistance like the potentiometer, but it senses compression instead of shaft rotation. The pressure exerted on the end of the leg is measured by connecting the output of the divider circuit to a analog input of the Beta 2 HC11 processor and reading the input. The HC11 determines if the foot is down by comparing the read value to a threshold value. If the read value is below the threshold, the foot is touching ground.

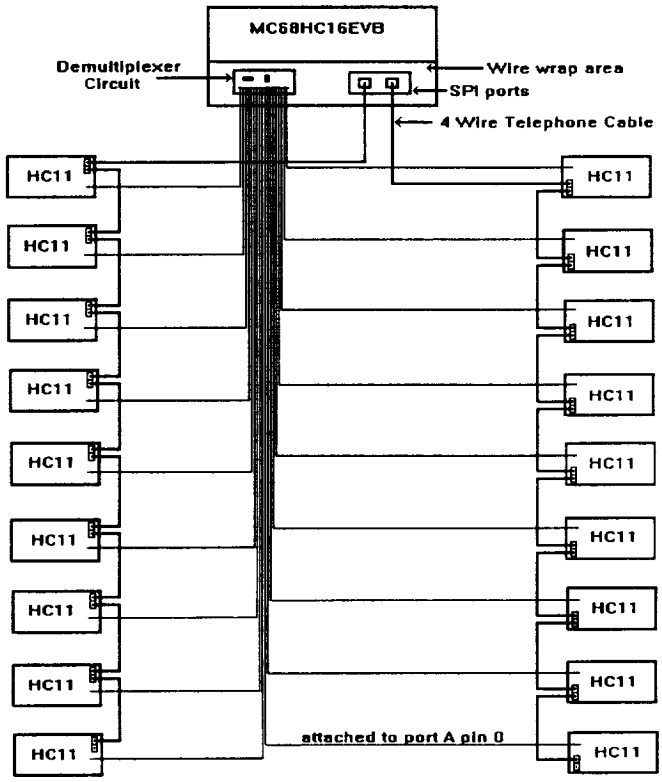


Figure II.1
HC16 to HC11
Daisy Chain and De-Multiplexer Network

2.2.3 Joint Position Control

The purpose of the motor driver circuit it to take inputs from the HC11 that dictate the speed and which way to turn the motor. In turn, signals are sent to supply the DC motor with the information to move in the specified direction and speed. The driver circuit

consists of a L6203 motor driver chip and several capacitors, diodes, and resistors. The circuit receives three signals from the HC11 processor. First, it receives a pulse width modulated (PWM) signal which specifies the speed at which the motor is to move by varying the duty cycle of the wave. Second, it receives one signal for each of the directions of motion that are set to logic one or logic zero corresponding, respectively, to whether or not the motor is to move in that direction.

2.3 Year Two Summary

The following design changes were made:

2.3.1 Limit Switch Changes

The design for the limit switches relied on the switch signaling the HC11 that the joint was over-extending and the processor turning off the motor. This did not account for the event of the processor or program failure which could result in the motor not receiving the instruction to stop even though the limit switch is triggered. The new design corrects this possible error by performing the logical AND operation between the limit switch signal and the HC11 signal and supplying this signal to move the motor in each direction. This combined signal is then used as the input into the motor driver. With this design, the limit switch circuit can manually turn off the motor in one direction without HC11 processor intervention. The HC11 can still move the motor in the opposite direction to correct the error, however.

2.3.2 Optical Encoder Circuit Addition

An optical encoder circuit was added to the sensor design to read the joint position in case of potentiometer failure or as a potentiometer position check. An optical encoder is attached to the shaft of the motor to measure the number of rotations of the motor shaft. The output from this encoder is used as the input into the optical encoder circuit. The optical encoder circuit changes the signals from the optical encoders into a twenty four bit binary number. This number changes when the shaft is turned. The HC11 reads the number and then calculates the angle of the joint by referencing the number to the movement from the previous location of the shaft.

2.3.3 Driver Circuit Component Change

The design for the motor driver circuit functioned as expected, however, it did not perform optimally. The new design changed the values of the capacitors, diodes, and resistors to achieve the best performance. The potentiometer and foot down sensor design functioned as expected. No changes were made with the exception of changing the value of the resistor in the divider circuit for the FSR to maximize the voltage change at the input into the HC11 analog input.

2.3.4 What Was Used This Year

All of the circuits that worked last year and the redesigned circuits were used for this year with the exception of the optical encoder circuit. This circuit was not included because

of space limitations on the printed circuit board containing all of the sensor circuits. Since the HC16 to HC11 communication was not established, an alternate network was designed. This network consisted of the same daisy chain SPI network, but it had a MC68HC11EVB evaluation board as the "brain". This network was not powerful enough to communicate will all of the processors because of the high impedance of the RJ11 phone cables. The design was then changed to using a MC68HC11 mini board as the "brain". This design extended the communication to eight HC11 processors and was implemented. The high impedance of the phone cables is due to the RJ11 male connectors on the ends of the cable. The connectors drive four sharp edges into the four wires in the cable to establish contact which results in a high chance of not making a good contact.

2.4 Limit Switches

2.4.1 Purpose

Limit switches are needed on each joint to protect the motor and joint from over extension. The limit switches will stop the motor from turning in the direction that has caused the limit switch violation. The limit switch circuit will also send a signal to the HC11 to indicate which limit switch is in violation. The HC11 will then be able to turn the motor in the opposite direction to back the motor off the violated limit switch. This system is capable of stopping the motor without intervention from the HC11, which protects against the possibility of processor failure causing joint or motor over extension.

2.4.2. Location

The limit switch circuits are located on the printed circuit board with the motor driver and foot down sensor. The optical interrupters used as the switch are located on the joints. The limit switches are electrically located between the HC11 and the motor driver as can be seen in Figure II.2(a).

2.4.3 Stopping the Motor

The forward and reverse direction will have the logic as shown for the forward direction in Figure II.2(b). If the limit switch signal and the HC11 signal are both high, then the signal to the motor driver will be high (IN1 input), causing the motor to turn. If the joint extends far enough to violate a limit switch, the limit switch signal to the AND gate shown in Figure II.2(b) will be low, which will force the signal to the motor driver to be low causing the motor to stop turning in that direction.

2.4.4 Signal to HC11

The limit switch signal is +5V (logic high) while the joint is not at the limit and 0V (logic low) while the joint is at the limit, this signal is inverted using a CMOS inverter and sent to the HC11. The inverted signal is also used to give on board indication in the form of a +5V blinking LED.

2.4.5 Action by HC11

Once the HC11 receives the signal from the limit switch circuit that the joint is at the limit, it has the opportunity to turn the motor in the opposite direction. This is possible because the forward and reverse direction controls are two separate inputs to the motor driver (IN1 and IN2).

If the HC11 would need the joint to move forward, the forward input to the motor driver (IN1) would be high and the reverse input (IN2) would be low. IN1 and IN2 control the motor with the following logic: with IN1 high and IN2 low, the motor will turn forward; with IN1 low and IN2 high, the motor will turn reverse; with both low the motor will stop. If the motor would continue to turn until it violated the forward limit switch the circuit would force IN1 to be low, while IN2 would not change. With both IN1 and IN2 low the motor driver would stop the motor. The LED would then start to blink and the signal to the HC11 would go high. The HC11 will then need to reverse the motor direction. It will do this by making IN1 low and IN2 high. IN1 is already being forced low by the limit switch circuit, which is acceptable because the limit switch violation in the forward direction does not affect the reverse direction (IN2). Therefore IN2 will be high and IN1 will be low causing the motor to turn in the opposite direction and backing off the limit switch. Once the joint has cleared the forward limit switch the circuit will automatically give control of IN1 back to the HC11.

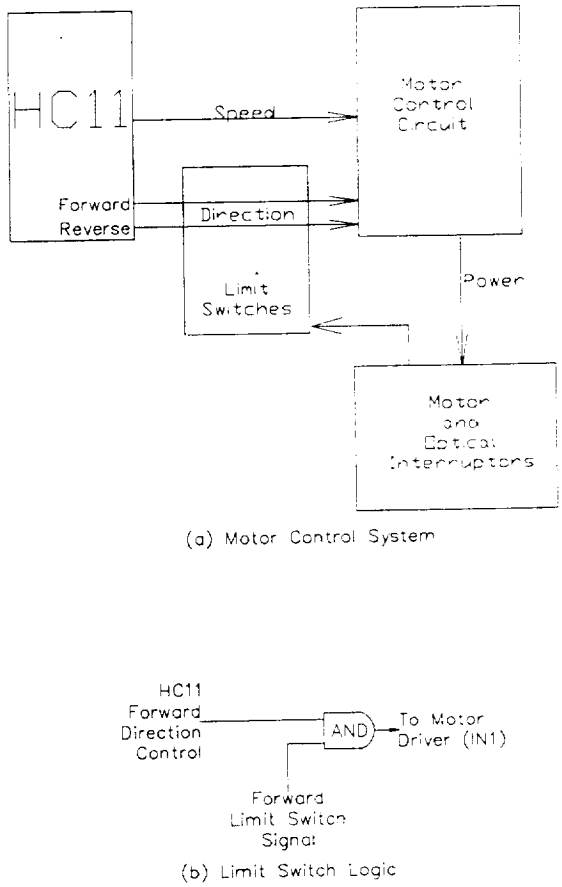


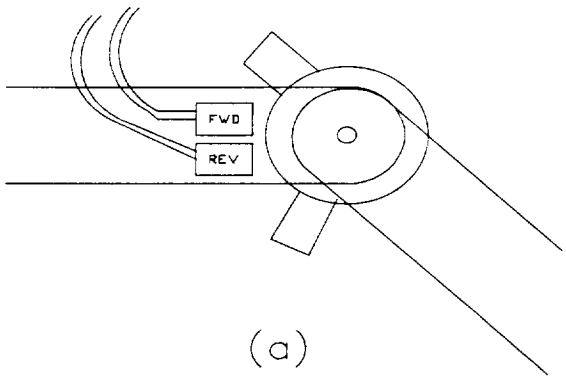
Figure II.2
Motor and Logic Circuits

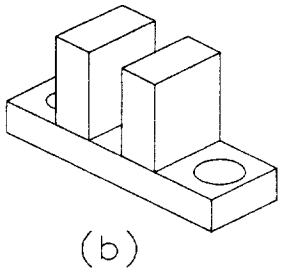
2.4.6 Components Used

2.4.6.1 Optical Interrupters

The switch is a optical interrupter device consisting of an infrared emitter and detector separated by a slot. The emitter is a gallium arsenide diode and the detector is a NPN silicon diode. The interrupters are available from Digi-Key. The manufacturer type number is "H21A1 Opto-electronic slotted switch". Two interrupters are used per joint as shown in Figure II.3(a). When the tab shown on the drawing is in the slot shown in Figure

II.3(b) the transistor is cut off and the signal to the AND gate goes to 0V (low) as can be seen in the simplified circuit diagram in Figure II.3(c). When the tab is not in the slot the signal to the AND gate is +5V (high). The pinout for the optical interrupter is shown in Appendix II.A.





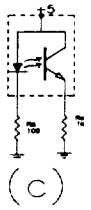


Figure II.3
Limit Switch Installation

2.4.6.2 Integrated Circuits

The AND gates are the standard 74LS08 TTL AND gate. The inverters are the standard CD4009C CMOS hex inverter. The supplier was Digi-Key, although these parts are commonly available through most electronics suppliers. The data sheets for the chips are in Appendix II.A.

2.4.6.3 Blinking LED

The LED used for on board indication of a limit switch violation is a +3V to +10V LED that includes a built in MOSFET. The MOSFET acts as the current limiting resistor and makes the LED blink, thus eliminating the need for external current limiting resistors and reducing the number of parts. When the LED has +5V between its two terminals, it blinks.

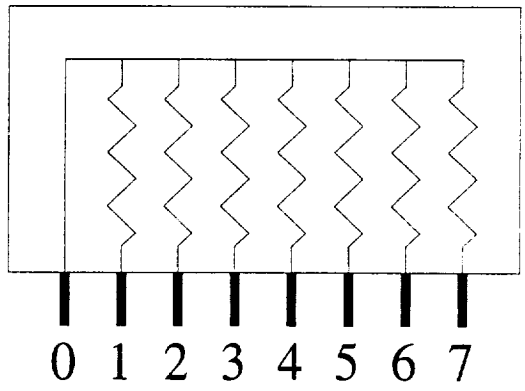


Figure II.4
Resistor Pack

2.4.6.4 Bussed Resistors

The optical interrupters require two resistors for each interrupter as can be seen on

Figure II.3(c). The printed circuit board uses bussed resistors to ease installation. Pin one of the bussed resistors is common to one side of all 7 resistors, which is grounded. The other pin for each resistor is connected to either the emitter or detector of the optical interrupter. The pin diagram is shown in Figure II.4.

2.5 Optical Encoder Circuit

2.5.1 Purpose

The purpose of the optical encoder is to enable the precise location (rotational) of a joint as a function of time. This is done by measuring the amount of movement from a reference position. The reference position can be set at a limiting point. The encoder is mounted on the end of the motors and came with the motors.

2.5.2. Operation of Optical Encoder

The encoder consist of four devices: two photo diodes, 1-led, and one multi-slotted disk. There is a gap between the led and the photo diodes which the disk passes. The slots in the disk allow light from the led to trigger the photo diodes. These two diodes are placed so that they are precisely 90 degrees out of phase electrically. These two photo diodes each produce a output signal (ch-A, ch-B). These signals are pulses of 50% duty cycle, one always leads the other depending upon the rotational direction. For more detailed information refer to the data sheets which are in Appendix II.C.

2.5.3 HP2016 Integrated Circuit

The HP2016 chip was designed specifically for the incremental encoder. The chip has three major portions: filtering, sequential logic, and 2 8-bit counting registers. The filtering is done with a Schmitt trigger. The sequential logic determines if ch-A is ahead of ch-B and if so to count up. If ch-B leads ch-A the count is decremented. The chip has four input lines: output enable (OE), reset (RST), a high speed clock (CLK), and a select (SEL). The chip also has 8 output lines (D0-D7). More detailed information is available from the data sheets in Appendix II.D.

2.5.4 HC11 Sending and Retrieving Data

The 2 8-bit registers are held as two words and retrieved one word at a time in a parallel fashion. The select pin determines which word is sent first when a call is requested. The remaining two lines are self descriptive CLK, the clock and RST the reset.

2.5.5 Optical Encoder Circumvention

The use of this device was cut from the final plans because of the need for shielded cables, more wiring, and a slightly more complicated printed circuit board. Time restrictions required a simpler circuit board and overall design simplification. The encoder is a very viable option and would allow very precise movement. This circuitry should be

reconsidered next year as the precision can not be matched by any other method.

2.5.6 Components Used

The optical encoder circuit was very simple to construct, the only chips used were the HP2016. The existing motor drivers worked fine on the prototype which was constructed. The major concern was the lack of documentation on the model of encoder. We did note an HP insignia upon part of it which lead us to documentation on the decoder chip and encoders that HP made. We did make an assumption on how the chip worked but this assumption was verified in lab. The HP2016 chips were available at Alliance Electronics in Kansas City. We found a few in the lab and a source for more, but dropped the idea of using the circuitry.

2.5.7 Recommendations for the Future

The use of the incremental encoder is a very viable option and would allow very precise movement monitoring. This circuitry should be reconsidered next year as the precision can not be matched by any other method. The printed circuit board would have to be much larger or multi-layered to enable all the traces necessary. The cost for a multi-layered board will be much higher, therefore a lot of effort should be spend verifying the completeness, and accuracy before one would be ordered.

2.6 Motor Driver Circuit

2.6.1 Purpose

The purpose of this circuit was to control the speed and direction of rotation of the motors which moved each joint. This is necessary to obtain a fluid motion action.

2.6.2 CMOS Integrated Circuits

Selection of this chip was made solely on the basis that many of them were on hand and they could perform the desired task. Virtually no comparison was done with other possibilities. The chip should work fine and should handle up to 4 amps at 48 volts.

This chip works on a fairly simple idea of having three digital control signals (In1, In2, and PWM), and a higher voltage (24) to drive the motors. The control of rotation was determined by setting one bit high and the other low, of the In1 & In2 inputs. The speed was controlled by the PWM which is a high speed clock input that outputs more power, to the motor, for higher frequencies that it receives. Appendix II.F contains more detailed information about the motor driver chip.

2.6.3 Special Considerations for Circuit

As with most CMOS chips, static is a major concern. We damaged a couple of chips accidentally by not taking proper precautions such as working in a static resistant environment. A filter (RC) was placed in the motor line to reduce the noise. Great care was

taken installing these chips as they showed that they very easily damaged.

2.6.4 SGS-Thomson L6203 Problems

The first thing we had trouble with in using this chip was the fact that it had to be soldered into place: No chip could be found sockets which matched or came close to matching. As mentioned earlier, these chips are very sensitive to static, thus care must be taken during installation.

2.6.5 Components Used

The selection of the components was made by the data sheet we had. The components required 2- 10 nF capacitors, 2-220 pF capacitors, 2- 1N0001 50 volt diodes, and 1- 10 ohm resistor. These were considered typical values from the data sheet. It was recommended to use monolithic capacitors but we used some ceramic (220pF) when we ran out of monolithic. We had no capacitor trouble.

2.6.6 Recommendations for Future

This driver chip could easily be required to run near the limits of its capabilities, thus a better motor driver should be considered. The motor driver circuitry should be placed on it's own printed circuit board along with the filtering circuitry. In this manner the motor driver circuitry becomes much more modular, thus debugging will be easier.

2.7 Cabling and Numbers

2.7.1 Purpose

To provide interconnection between the processors and the sensors. The cabling from the printed circuit board to the sensors on the leg is a 24 wire shielded cable. The cabling between the HC11s and the HC16 is 2 pair phone cable.. The cabling diagram showing how all the cables interconnect is shown in Figure II.5.

2.7.2 HC11 Port Assignments

The HC11 has 5 ports for external connections. Ports A, B, C, and E are used to connect to the PCB and the sensors. Pin 0 of Port A is used in the communication system as a way to select which HC11 the HC16 is communicating with. The remainder of the pins of Port A are used to connect to the PCB. The pin assignments for Port A are shown in Table II.1. Port B is used exclusively to communicate from the HC11 to the optical encoder, its pin assignments are shown in Table II.2. Port C is used exclusively to retrieve data from the optical encoders, its pin assignments are shown in Table II.3. Port D is used for the connection to the communication network. Port E is used to connect to the foot down sensor and the potentiometer sensor circuits that are on the PCB, its pin assignments are shown in Table II.4.

Port A		
Pin #	Input or Output	Use
0	I	signal from HC16 encoder
1	I	limit switch pos. angle
2	I	limit switch neg. angle
3	I/O	
4	O	motor driver neg. angle
5	O	motor driver pos. angle
6	O	motor PWM signal
7	I/O	

Table II.1
HC11 Port A Pin Assignments

Port B	
Pin #	Function
0	optical encoder reset bar
1	optical encoder OE bar
2	optical encoder select

Table II.2
HC11 Port B Pin Assignments

Port C		
Pin #	Input or Output	Function
0	I	optical encoder data
1	I	optical encoder data
2	I	optical encoder data
3	I	optical encoder data
4	I	optical encoder data
5	I	optical encoder data
6	I	optical encoder data
7	I	optical encoder data

Table II.3
HC11 Port C Pin Assignments

Port E	
Pin	Function
0	analog I potentiometer
1	analog foot down sensor (Beta)

Table II.4
HC11 Port E Pin Assignments

20 Pin Header			
Pin	Function	Wire Colors	Cb l
1	Alpha Pot GND	Blue W/White	1
2	Beta 1 Pot GND	White w/Brown	1
3	Beta 2 Pot GND	Blue w/White	2
4	Alpha Opt. Interrupt. +5	White w/Green	1
5	Alpha Neg. Limit Emitter.	Orange w/White	1
6	Alpha Neg. Limit Detector	Red w/Blue	1
7	Alpha Pos. Limit Emitter	White w/ Blue	1
8	Alpha Pos. Limit Detector	Gray w/White	1
9	Beta1 Opt. Interrupt. +5	White w/ Gray	1
10	Beta1 Neg. Limit Emitter.	Green w/White	1
11	Beta1 Neg. Limit Detector	Blue w/Red	1
12	Beta1 Pos. Limit Emitter	White w/ Orange	1
13	Beta1 Pos. Limit Detector	Orange w/White	2
14	Beta2 Opt. Interrupt. +5	White w/Brown	2
15	Beta2 Neg. Limit Emitter.	Brown w/White	2
16	Beta2 Neg. Limit Detector	White w/Green	2
17	Beta2 Pos. Limit Emitter	White w/Gray	2
18	Beta2 Pos. Limit Detector	Red w/Blue	2
19	Foot Down Sense	White w/Blue	2
20	Foot Down GND	Gray w/White	2
	Alpha Pot Sense	White w/ Orange	2
	Beta1 Pot Sense	Green w/White	2
	Beta2 Pot Sense	Blue w/Red	2

Table II.5
PCB 20 Pin Header

14 Pin Encoder Header			
Pin	Function	Cable Colors	Cabel#
1	α +5	Blue W/ White	3
2	α GND	White w/ Brown	3
3	α A	Brown w/ White	3
4	α B	White w/ Green	3
5	B1 +5	Orange w/ White	3
6	B1 GND	Red w/ Blue	3
7	B1 A	White w/ Blue	3
8	B1 B	Gray w/ White	3
9	B2 +5	White w/ Gray	3
10	B2 GND	Green w/ White	3
11	B2 A	Blue w/ Red	3
12	B2 B	White w/ Orange	3

Table II.6
PCB 14 Pin Header

2.7.3 Printed Circuit Board Headers

These are the two cables connecting the sensors on the leg to the PCB. The cables are 24 conductor shielded cable. The cables connect to the PCB via a 20 pin header and a 14 pin header. The pin assignments for the headers are in Table II.5 and Table II.6.

2.7.4 HC11 to the Printed Circuit Board

The connections between the HC11 and the PCB are shown in Table II.7.

2.7.5 Components used

- The 20 pin connectors are Dupont "Quickie III" system available from Newark. These connectors conform to the standard IDC 0.10" 2 by 10 connector. The Quickie III system adds levers that eject the plug from the socket, which reduces the stress on the circuit board.
- The 14 pin connectors are also Dupont

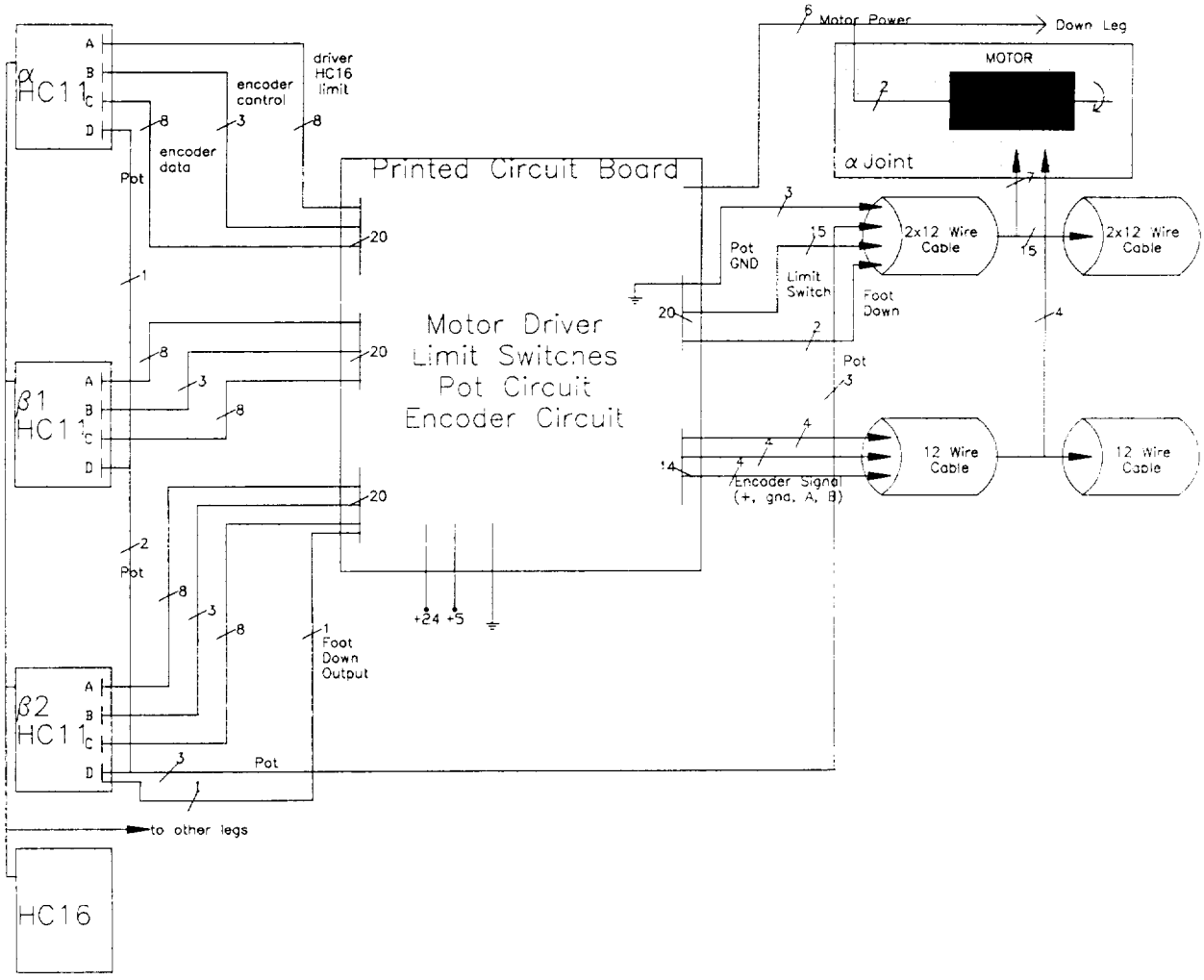


Figure II.5
Cabling Diagram

Quickie III connectors in a 2 by 7 configuration.

PCB Pin	Function	HC11 Port	HC11 Pin
1	Limit Switch Pos. Angle	A	1
2	Limit Switch Neg. Angle	A	2
3	NC	A	3
4	Motor Driver Pos. Angle	A	4
5	Motor Driver Neg. Angle	A	5
6	Motor PWM Signal	A	6
7	Encoder Clock	A	7
8	Opt. Encoder Reset Bar	B	1
9	Optical Encoder OE Bar	B	2
10	Optical Encoder Select	B	3
11	Optical Encoder Data	C	0
12	Optical Encoder Data	C	1
13	Optical Encoder Data	C	2
14	Optical Encoder Data	C	3
15	Optical Encoder Data	C	4
16	Optical Encoder Data	C	5
17	Optical Encoder Data	C	6
18	Optical Encoder Data	C	7
19	Pot Sense	E	1
20	Foot Down (Beta 2 only)	E	2

Table II.7
PCB Pin Assignments From HC11

2.8 Foot Down Sensor

2.8.1 Purpose

Force sensing resistors will be used to sense when a foot is on the ground. A signal will be sent to the HC16 indicating whether or not the foot is on the ground. The HC16 will then use that input in the walking algorithm.

2.8.2 Signal to HC11

The force sensing resistors will create a low noise voltage signal that will indicate whether or not a foot is on the ground. That signal will then be sent to the HC16 for processing. The signal will be dependent upon how much force is being applied to the foot.

2.8.3 Components Used

The force sensing resistors (FSR) change resistance value based on the external force. The FSRs were procured from Interlink Electronics in addition to application circuit suggestions. The circuit used is a voltage divider with the output going to the HC16.

2.8.4 Foot Design

The foot design has gone through several iterations. The final design utilizes a rubber boot placed on the end of the foot. When the foot contacts the ground, the rubber compresses and puts pressure on the FSR, changing the output voltage.

2.9 Potentiometer

2.9.1 Purpose

The purpose of the potentiometer circuit is to provide an indication of the absolute position of the joint to the HC11. The circuit does this using a potentiometer (pot) attached to each joint so that joint movement will cause a change in the output of the circuit.

2.9.2 Location

The potentiometers are mounted on the axle of each joint. As the joint moves the

resistance of the pot changed. This change was sensed by the HC11. The circuit uses one resistor and one potentiometer in a voltage divider configuration for each joint. The resistor was mounted on the PCB. The output of the circuit was then sent to the HC11.

2.10 Printed Circuit Board

2.10.1 Software Used

The software used to generate the PCB artwork is called PADS. The circuit design is done by first placing the components and the connections on the drawing area. The software will then automatically route the connections over the board in such a manner that no traces cross. The software automatically uses as many layers as needed. The pad size, pad spacing, and trace spacing are adjustable. The artwork created by the software is shown in Appendix II.E.

2.10.2 Circuit Diagram and Description

The circuit diagram used to create the circuit board in shown in Figure II.6 minus the optical encoder circuit. The circuit includes the limit switches, the motor drivers, the foot down sensor, the potentiometer, and the optical encoder.

2.10.3 First PCB Design

The first design used a pad size of 30 mil, pad spacing of 15 mil, trace size of 10 mil, and trace spacing of 10 mil. The first iteration of the first design was done by John Rogge at Kansas State. The PCB tracings were of very high quality. The problem with the PCB was that John Rogge had no way to drill the holes for the pins of the components. The drilling that we did was not accurate enough to allow the use of the headers and IC sockets. Therefore we tried a commercial PCB production company from Manhattan. The PCB they produced was not as high quality as that John Rogge had produced, but the holes were drilled correctly. The board was unusable because of the many errors in the traces. The company suggested that the traces were too small and that a larger trace size and spacing would result in fewer errors. The optical encoder circuit had many connections, thus many trace on the PCB. For that reason the optical encoder circuit was removed from the PCB. Once the trace size, pad size, pad spacing, and trace spacing were increased and the optical encoder was removed, the software easily routed the circuit and thus the second PCB design worked.

2.10.4 Second PCB Design

The second design used a pad spacing of 30 mil, trace size 20 mil, and trace spacing of 20 mil. The artwork was again sent to the Manhattan company. The board produced appeared to be acceptable. All six boards were immediately ordered.

2.10.5 Testing and Debugging

As mentioned above, all six boards were produced before the first board was sufficiently tested. There were both wiring

errors and trace errors on the board. The wiring errors were a result of incorrect connections in the PADS software. The two layers of the circuit board also caused some problems. There were many places where the circuit jumped from layer to layer, at those points there is a hole in the board. The normal procedure is to "plate through" the hole, thus connecting both layers at that point. Because of budget and time constraints, plated through holes were not available. The problem was magnified because the header and socket pins were very difficult to connect to a trace if that trace were on the top layer of the board. After extensive testing, all the errors were found and all 6 boards were installed.

2.10.6 Conclusion

The PADS software worked exceptionally well, allowing changes to any of the parameters of the circuit routing. The PCB that John Rogge produced was of excellent quality although the holes were not drilled. The problems with the first design were due to the small size of the pads and traces. The Manhattan company did acceptable and quick work. The holes on the circuit board should have been plated through. The trace errors were the result of improper etching of the board, these errors are unavoidable in this case. The wiring errors would have been found earlier if the circuit had been checked before the boards were produced. In general the group now recognizes that any PCB production process in the future should allow for extensive testing and debugging before the PCB is actually produced.

2.11 Power and Motors

2.11.1 Power Supply Analysis

To sufficiently power the robot there are two power supply choices:

- Modular Switching Power Supply: The output voltage is 24 V dc and the output current is 25 amps. The input voltage is 85-132 V ac or 170-250 V ac. The weight is 9 lb. Cost of the device: $275.00. This device is listed in the 1994-95 catalogue of Servo Systems.

- Deep Cycle Marine Battery: The output voltage of one battery is 12 V dc and output current is 235 amps. Two batteries can be used in series to obtain a voltage of 24 V dc. The weight of two batteries is 30 lb. Cost of both the batteries: $90 (approx.) These batteries can be bought at any battery shop.

The current power supply is a modular power supply that was used last year. The power supply is sufficient for the current testing.

2.11.2 Description of Motors

The motor chosen by last year's design team is a 24 V motor produced by Matsushita

Electric and distributed by Servo Systems. A major advantage of the motor is a built-in optical encoder, which supplies digital data relating exact relative positioning.

The motor is rated for 21.8 ft. lb. at 185 rpm, which will be the operable speed. The locked rotor torque for the motor is 33.3 ft. lb. and has a rated current of 5A. This means that the motor is capable of handling (24V) (5A) = 120 Watts of power.

2.11.3 Motor Analysis and Suggestions

If the Modular Switching Power Supply weight would be added to the weight of the robot the maximum torque would be 258.72 oz. in. as calculated by the mechanics group. The motor will be able to handle this torque with approximately 150 rpm and 3 A. The batteries would be required to produce a total current of (6 motors) (3 A) = 18 A.

If the Deep Cycle Marine Batteries weight would be added to the weight of the robot the maximum torque would be 315.84 oz. in. as calculated by the mechanics group. The motor will be able to handle this torque with approximately 50 rpm and 4 A. The batteries would be required to produce a total current of (6 motors) (4 A) = 24 A.

2.12 De-Multiplexer Circuit

The purpose of the de-multiplexer circuit is to receive inputs signals from the HC16 that specify which HC11 the HC16 wants to communicate with and output the appropriate signal to that HC11. The connections from the HC16 to the HC11s are shown in Figure II.1. The first design of this circuit used the data lines of the HC16 as inputs into the de-multiplexer. Data lines D0 to D3 are connected to input pins G,A,B,and C of one three-to-eight de-multiplexer chip. Six of the outputs of the 74LS153 chip go through a 74LS08 de-multiplexer inverter chip to the first six HC11 processors. The remaining data lines, D4 to D15, are connected directly to the remaining HC11 processors. This design is not feasible because the data bus is also connected to the random access memory of the HC16 which must be used simultaneously with the communication. The second design consisted of using four lines into a four to sixteen de-multiplexer with inverters on the outputs and two lines connected straight into the remaining two HC11 processors. This design has not been implemented because the software group discontinued the work on HC16 to HC11 communication and there was no need for the circuit until next year.

2.13 Conclusion and Objectives for Next Semester

The hardware group has designed and implemented the systems that are used to control the robot. Several of the circuits were designed into a PCB and implemented into the robot. The optical encoder circuit was removed in the interest of reducing the complexity of the circuits. The limit switch

circuit was bypassed until it can be further debugged.

Next year the limit switches will be implemented. The obstacle avoidance sensors will be added and HC11 cable networking will be improved.

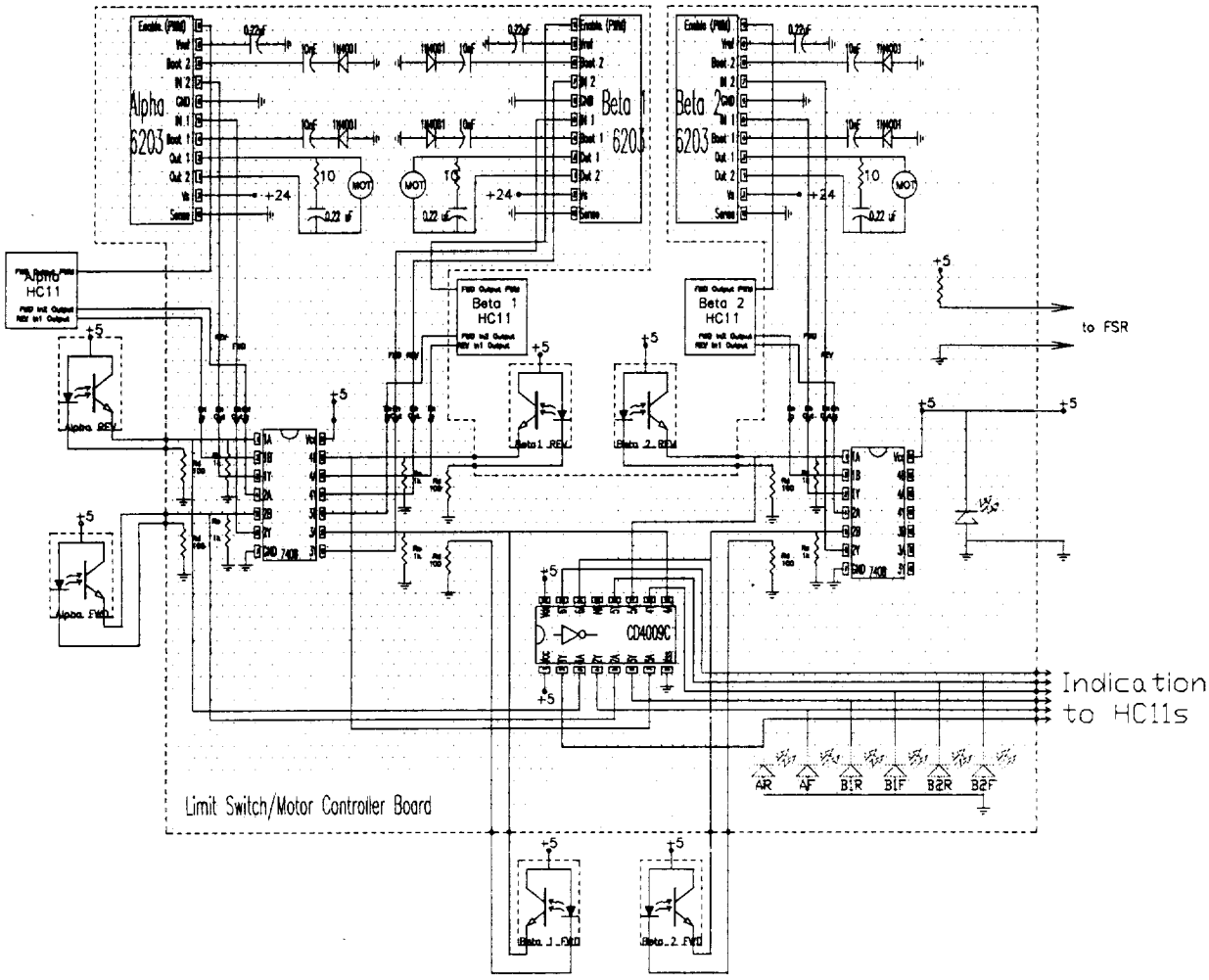


Figure II.6
PCB Circuit Diagram

III. Software

3.1 Software Introduction

The software group has the responsibility of creating intelligence for robocat. Due to the lack of actual hardware available during the first semester of year 2 much of that semester was spent dealing with the theoretical issues of artificial intelligence. Very little was implemented during that semester. The second semester was spent almost entirely on implementation of low level hardware control such as reading sensor values, controlling motors and setting up communication between the processors.

Certain problems impeded the progress of the software group.

The low level programming was much more difficult than had originally been planed, especially with the network communication programming. Many small problems were encountered that took time to debug.

Hardware problems slowed progress as well. Much of the software testing was delayed by hardware not being available thus halting progress by the software group.

For testing, the MC68HC11E2 processors were used. This processor is very convenient for testing since it uses EEPROM (Electrically Erasable Programable Read Only Memory). The advantage to this processor is that new programs can easily be loaded. This disadvantage is that it contains only 2Kb of program space limiting what can be done. Once debugging is finished the processors will be replaced with MC68HC11E9 processors. These contain 12Kb of EPROM (Erasable Programable Read Only Memory). This is 6 times the program space, but these processors need to be erased by ultraviolet light in order for a new program to be loaded.

Due to these problems not all that was implemented by the software team is being used on the robot. This report covers what has been implemented and will be used once the problems are resolved as well as what is actually being used on robocat.

The software tasks can be divided up into the following sections:

- Sensory input
- Communication
- Walking Algorithm
- Future Plans
- Temporary Modifications

3.2 Year 1 Summary

The goals of Year 1 design team was to build a working prototype of the walking robot rover. The purpose of the prototype is to learn about the mobile prototype designs. In this first year of USRA/NASA Robotics, the

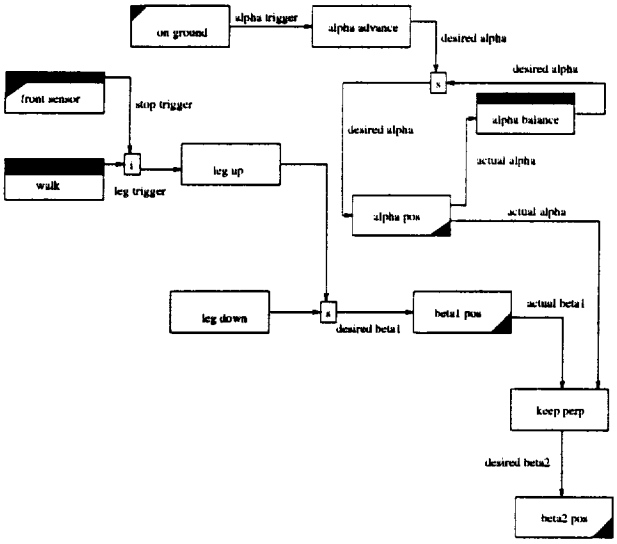


Figure III.1
Subsumption Network of Behaviors
for Level Ground

Software Group produced two major developments in robot control. These are subsumption architecture and the basis for a walking algorithm.

3.2.1 Subsumption

The design of the software is based on the subsumption architecture. Subsumption architecture was developed by Dr. Rodney Brooks at the M.I.T. mobile robot laboratory. This approach to software design is based on task achieving layers. Each layer reads sensor input and controls actuator output. The Year-1 Software Group suggested that the lowest level layers should control basic tasks such as walking on level ground. Higher level competence would then be aimed at more difficult tasks, such as walking on uneven ground, or climbing obstacles. Subsumption architecture controls the actions of the robot by higher levels subsuming (taking control away from) lower levels. If the robot software obtains sensory information that the robot is in a situation where a higher level specialty applies (such as uneven ground) the higher level subsumes the lowers levels. Although the lower levels have no control over the actuators at this time, the lower levels do not stop sending commands. This way, if the higher levels should fail, the lower levels can take control and at least the robot will not fail.

With this knowledge from the previous year, Year 2 Software Design Team implemented Finite State Machines (FSMs) as a basis for subsumption architecture. Each FSM is concerned with details concerning its function thus minimizing complexity. When many of these FSM's work together higher level behaviors emerge. The FSM's for implementing walking are leg up down, alpha advance, keep perp, keep foot in position, leg down and alpha balance.

3.3.2 Walking Algorithm

The Year-1 Software Design Team also began work on an walking algorithm. This design was for walking on level ground. In Figure 1, each box represents a behavior. Boxes without bands on top are repeated six times, once for each leg. Boxes with triangles in the lower right corner control the legs. Boxes with solid bands are the "central control" of the robot. Finally, boxes with triangles in the upper corner obtain sensory information. These behaviors work together to form a walking algorithm for level ground.

This walking algorithm include four basic states.

Standing

The simplest behavior is standing. This behavior is the lowest level of the subsumption architecture therefore it will only be active when no other FSM wants control. Standing uses the behavior leg down. This walking algorithm sets all the angles on the leg to a desired position causing the robot to stand with a level posture.

Leg Lifting

The next state is leg lifting. This behavior uses the leg up FSM. When there is no input to the leg up FSM it remains inactive so that the leg remains in the standing state. The leg up FSM is triggered by a signal from the walk FSM which is responsible for coordination of all six legs. When this trigger is received the leg up FSM becomes active and moves the beta-1 joint to a raised position.

Leg Swinging

Leg swinging is a state in which the alpha joint of the leg is used to move the leg forward or backward.

There is a machine, alpha balance, which receives the alpha positions from all six legs. The sum of these alpha positions tells alpha balance to move forward (positive sum) or backward (negative sum). Alpha balance will then send a signal to each of the alpha position actuators which will adjust the legs to keep the body centered. If one leg moves forward, alpha balance will move all other backward to compensate. This does two things; it keeps the body balanced, and it causes the legs that are on the ground to move backward as a reflex action when the other legs are moved forward. The behavior that emerges from these simple behaviors is walking.

There is also an alpha advance machine which will swing the leg forward. When up-leg-trigger raises a leg, alpha advance swings it forward and subsumes alpha balance.

Walking

A behavior must be added to trigger the up-leg-trigger machines to produce the appropriate gait for the environment. Year-1 Software Design Team decided upon a tripod gait for walking on level ground.

In conclusion, Year-2 learned a great deal from Year-1's trials and errors. Subsumption architecture proved useful when applied to Finite State Machines. The current walking algorithm is based on the algorithm described here. The new walking algorithm is broken down into greater detail in a following section.

3.3 Sensors

Sensor feedback is vital to robot movement. Without sensor feedback, the environment around the robot cannot be measured. The sensor input on each leg needed to walk includes:

- Is the bottom of the foot making contact with a surface.

- What is the position of the leg relative to the ground.
- What is the position of the leg relative to the robot.

With this input, the leg position is determined. The sensors used on robocat include, a pressure-sensitive foot-down sensor, limit switches, potentiometers, and optical encoders.

3.3.1 Foot Down Sensor

In order to know when the bottom of a foot is making contact with a surface, a pressure-sensitive sensor is placed on the bottom of each foot. The foot down sensor sends information to the HC11 of the Beta 2 joint. The signal is in analog form. The sensor creates variable resistance dependent on the surface pressure on the sensor. When the resistance is the least, the foot is down. When the resistance is high, the foot is not down. This helps determine when a leg is to be moved forward or backward as discussed in the section covering the walking algorithm.

3.3.2 Limit Switches

In order to prohibit the robot from walking on itself or moving a joint to an extreme angle, limit switches have been installed on each joint. The limit switches do what their name imply, limit the movement of a joint. The limit switches are positioned so that if one is triggered, the joint has been extended or contracted out of the normal range that the software uses. The instant the switch is triggered, the hardware prevents the joint from moving any further in that direction. When this happens a signal is sent to the HC11 processor that controls the joint via a digital signal line. The joint can the be moved in the opposite direction. The limit switch can also be used as a rude leg position indicator.It is used as a secondary sensory input for this use. The limit switch returns a digital signal when the joint is at its maximum position or minimum position. When this occurs the joint is re-calibrated to prevent this from happening again.

3.3.3 Joint position sensors

Potentiometer

In order to know the current position of a joint, each joint is equipped with a potentiometer. The potentiometer is turned by joint movement. The angle of the joint can be calculated based on the voltage returned from the potentiometer. This, unlike the limit switch, can give exact positions. The three potentiometers on each leg are used to determine leg geometry. Each potentiometer is pre-calibrated to allow calculation of the joint angle based on the voltage. When the calibrations are correct, precise measurements are given. When calibrations are incorrect, a limit switch may be triggered due to the poor precision in the measurements. When this happens, the potentiometer can be re calibrated by the software on the HC11. Due to the limited amount of code space on the

current HC11's the re-calibration software in not in use. This will be added once HC11's with a larger code space are installed.

Optical Encoder

Another sensor that can be used to determine joint position is the optical encoders that are built into each motor. The encoder contains some logic and sends 2 bytes of information to the processor. This information is the number of pulses counted by the encoder. Each pulse is a movement of the motor shaft a constant distance of rotation. The computation however to calculate the leg position is more complicated than the potentiometers. The shaft rotation of the motor is reduced through gear reduction when moving a joint. Due to the complexity of the hardware circuit the optical encoder is not being used. From current tests, the potentiometers seem to be adequate as the source for joint position sensing.

3.4 Communication

3.4.1 General Layout

The control of robocat is divided up among 19 separate processors. One HC11 on each joint (18 total) and one higher level HC16 processor for main control as shown in figure 2. The HC11s are chained together using the two built in RJ11 jacks on the HC11 boards which are attached to the SPI. Two chains of nine boards each leave the HC16.

The Synchronous Serial Peripheral Interface (SPI) is used to allow communication between the HC16 and HC11s. It runs at a maximum of 1 Mhz, half the bus speed of an HC11, providing a data transfer rate of 1 Mbit per second. This bandwidth is more than adequate for the demands of this project.

Each HC11 and HC16 also has an asynchronous serial communication port (SCI) which would be adequate for communication. This port is used to download programs to the

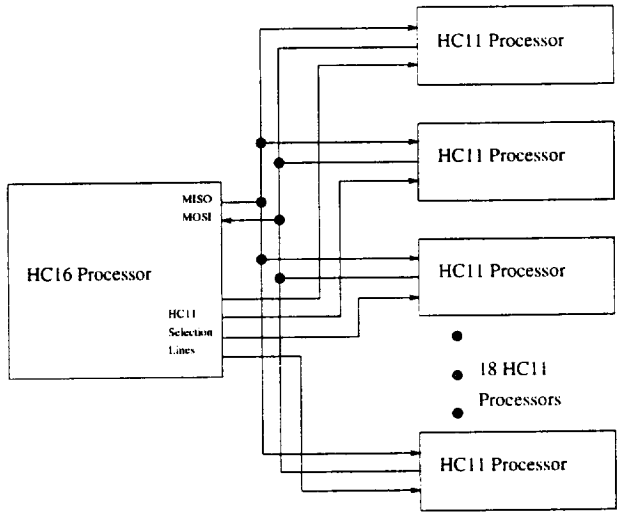


Figure III.2
Communication Diagram

HC11 processors and to monitor debugging output. Because of this, it is not suitable for inter-processor communication.

3.4.2 Implementation Details

The SPI on an HC11 must be constantly monitored by the processor in order to keep up with the incoming data. The HC16 has more advanced serial hardware called the Queued

Serial Peripheral Interface (QSPI) which allows it to continue executing its program while communication is taking place; it is then interrupted at the end of a transmission.

Since the HC11 boards lack hardware support for SPI collision detection, the communication software must insure that only one processor is transmitting at a time. This is accomplished by putting the HC16 in control of all communication. It has a signal line to a digital input of each HC11 which is asserted when the HC16 is ready to communicate with that particular HC11. The HC11 processors are unable to initiate communication; this isn't a problem because the HC16 communicates with each HC11 many times per second.

In SPI communication one processor must act as the master; its serial clock is used by all others on the network. Only one master may be active at a time. On the robot, an HC11 never communicates with another HC11, information which must be passed between two joint processors is passed through the HC16. This makes implementing the networking software fairly straightforward, which is necessary due to the limited amount of code space available on each HC11.

To communicate, the HC16 puts data into a transmit buffer and configures the QSPI to be the slave. Once the HC16 is ready, it asserts the line to the proper HC11 and is then free to do other things. When the HC11 notices that the HC16 wants to communicate it configures its SPI to be the master and begins simultaneously shifting bits in and out if it's SPI. When the last byte of the data has been received it shuts down the SPI so that another processor can become master. At the end of the transmission the HC16 is interrupted by the QSPI and it retrieves data from its receive RAM.

3.4.3 Communication Protocol

A packet sent between the two processors consists of 8 bytes. The first is a synchronization byte, chosen so as not to coincide with other bytes in a transmission. If the first character a processor receives in a transmission doesn't have the expected value, then it throws out all bytes until it does receive the synchronization byte.

Bytes 2-6 of a packet are data. Following this is the number of the joint processor; this allows both the HC11 and HC16 to notice if a connection is wrong. The last byte of a packet is a checksum of the data and processor number values; this allows bad packets to be detected and thrown away.

All of the communication is encapsulated into a send-receive routine for the HC16 and HC11s. The prototype is the same for both:

```
int sendrec (char* buf,char procnum)
```

Where buf points to a data buffer and procnum is the number of the processor for the HC16 to talk to, or the number to send for an HC11. The buffer pointed to by buff will contain the data to be sent. After a transmission the data in this buffer is replaced with the data that was received. Possible errors from sendrec are: ECHKSUM, for a checksum error and EWRONGCHIP if the

processors don't agree on which HC11 is being talked to.

3.4.4. Problems

Communication is currently unreliable; the signals must travel through 18 cables and 36 phone jacks in order to reach all of the processors, and a bad connection can disable up to nine processors, depending on where in the chain it occurs. Also, if an HC11 becomes wedged while its SPI is in master mode the entire network is unusable since all further communication will collide.

3.5 Walking algorithm

The walking algorithm is based on the subsumption programming style as developed by Rodney Brooks at the M.I.T. artificial intelligence laboratory. It is designed in a way that little communication is needed. Instead of sending detailed commands from the main processor to each of the joints a simple trigger is sent to invoke a behavior. Much of the behaviors act in a reflex manner.

The walking algorithm is based on the walking algorithm that was designed during the first year of the project. The modifications were nessasary due to changes in the distribution of control. The original design had a processor for each leg. The current design has a processor for each joint. Control is divided up into 4 distinct parts, 3 on each leg and 1 high level processor. The leg is divided up into alpha, beta1, and beta2 control representing each joint. Each of these only need to take care of the specifics for their own joint. The higher level processor communicates with all 18 joint processors and takes care of coordination of the legs and passing the pertinent information between the processors.

3.5.1 Central Control

Central control of the robot is achieved using a Motorola MC68HC16 processor that communicates with all 18 joint processors via a high speed network. The network is described in detail in another section. This processor is responsible for coordination of the six legs during walking. Coordination is achieved with two behaviors on the HC16: alpha balance and walk.

Walk

The walk behavior is responsible for causing the correct legs to step at the right times. The walking is done as a tripod gait modeled after six legged insects. With this gait three legs are on the ground at all times creating a stable stance. To create this walking gait the main processor sends a walk signal to the appropriate beta 1 joint processors telling the leg to raise. The rest of the details of walking are taken care of by lower level behaviors that reside in the individual joint processors. This will be described in more detail in the sub sections on the joints.

Alpha Balance

The alpha balance behavior is also nessasary for walking. This behavior has control over the legs that are not being controlled by the walk behavior. The angles of all six alpha joints are summed together. The angles are defined so that 0 degrees is when the alpha joint is perpendicular to the body of the robot. If the sum of the angles is greater than 0 alpha balance sends a signal for all joints to move backward, if the sum is less than 0 a signal is sent for the joints to move forward. This attempts to keep the body centered over the legs. Since this behavior has no control over the legs that are being controlled by the walk behavior it will move the three legs that are on the ground backward. This gives the result of the robot moving forward. The result is smooth tripod locomotion.

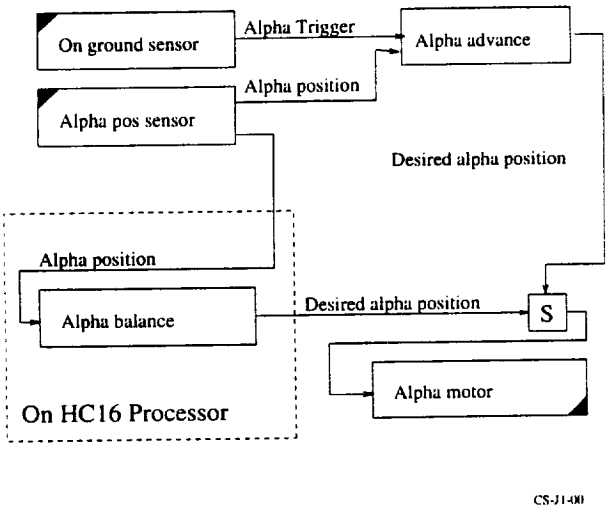


Figure III.3
Alpha Joint Subsumption

3.5.2 Alpha joint

This alpha joint moves the leg in the forward and backward directions. This joint is controlled by two separate behaviors: alpha balance and alpha advance.

3.5.2.1 Alpha balance

This is the receiving end of the alpha balance control from the main processor. This end simply moves the alpha joint to the specified position.

3.5.2.2 Alpha advance

This behavior is responsible for the stepping motion of the leg. The alpha advance is triggered by the foot sensor. When the foot is no longer on the ground the reflex action is to move the leg forward. Resulting in a step.

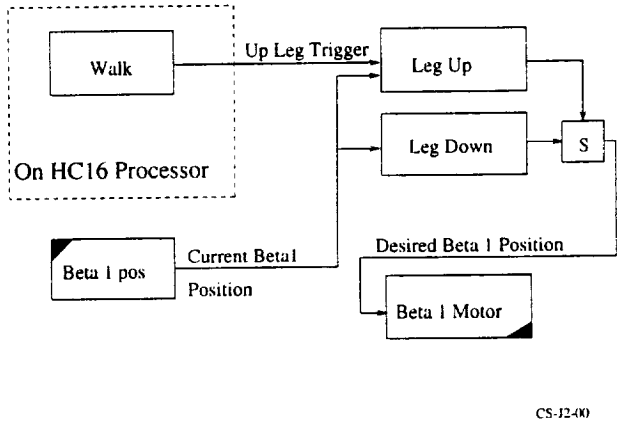


Figure III.4
Beta 1 Joint Subsumption

3.5.3 Beta1 joint

This joint is responsible for raising and lowering the leg. Control of this leg is achieved by two behaviors: leg up and leg down. Priority is given to the leg up behavior.

3.5.4 Leg up

This behavior is triggered by the walk trigger which is a signal sent from the main processor. When active the leg up behavior simply raises the leg to a desired walking hight. This raising of the leg causes the foot down sensor to come off the ground which in turn causes the the alpha advance behavior to become active. The resulting higher level behavior of these very simple behaviors is a walking step. When this is done in a well coordinated manner the tripod gait emerges.

3.5.5 Leg down

This behavior will be active whenever the leg up is not subsuming it. This behavior simply keeps the beta 1 joint at an angle so that the foot is on the ground.

3.5.6 Beta2 joint

The beta 2 joint is responsible for two tasks. When the foot is on the ground the keep perpendicular behavior is active. When off the ground the keep position behavior is active. These two behaviors keep the feet from slipping.

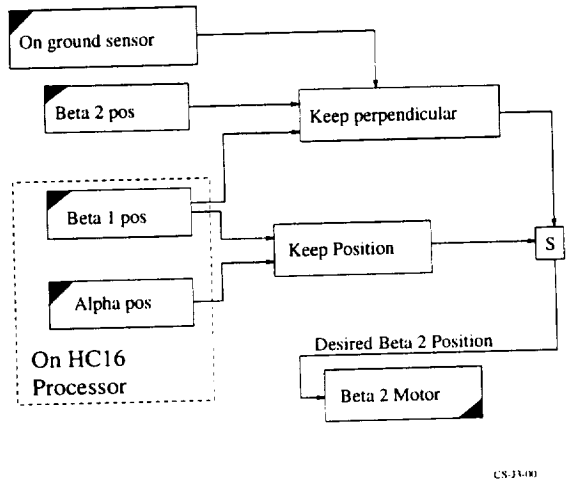


Figure III.5
Beta 2 Joint Subsumption

3.5.6.1 Keep Perpendicular

This behavior is triggered by the foot on ground sensor. When the foot is off of the ground the Keep Perpendicular behavior adjusts the angle of the beta 2 joint so that it is at all times perpendicular to the surface. The perpendicular angle for beta 2 is calculated as 90 - beta 1 angle .

3.5.6.2 Keep Position

This behavior is active whenever it is not being subsumed by the Keep Perpendicular behavior. The Keep Position behavior adjusts the angle of the beta 2 joint so that it keeps the same position on the ground. This is done using feedback from the positions of the alpha and beta 1 joints.

3.5.7 Temporary Modifications

As mentioned earlier there are problems with the communication network that have not yet been resolved. Because of this the walking algorithm as mentioned in the previous section has not been put into implementation at this time. Another walking algorithm has been developed that does not need the communication network. With this non-communication walking algorithm it is possible to test much of the lower level software (e.g. reading sensors and controlling actuators) as well as the hardware. The design

of the non-communication walking algorithm is fairly simple. The general layout of the processors is the same, one main control processor or 'brain' and another processor on each joint. Only 12 joints are used, the beta 2 joint is kept at a constant angle so no control is currently needed for it. Each joint processor is connected by two signal lines to the main processor. These lines allow for very simple communication as an on or off signal. One is used for the brain to invoke a behavior on the joint and the other is to notify the brain that it has finished its task. Since only one of the joints on each leg can be moving at a time both joints on the leg share the same outgoing signal line. Only the joint that is currently moving in that state will respond.

3.5.8 Brain processor

The brain processor checks all the signal wires coming from the joints and when all wires are set, it moves on to the next state. The sequence on the brain consists of the following:

Signal legs 0, 3, and 4 to raise up.

When 0, 3, and 4 are all the way up, signal them to move forward.

When 0, 3, and 4 are all forward, signal them to drop.

When 0, 3, and 4 are all down, signal all legs to move back half way.

When all legs are all back, signal 1, 2, 5 to raise up.

When 1, 2, and 5 are all the way up, signal them to move forward.

When 1, 2, and 5 are all forward, signal them to drop.

When 1, 2, and 5 are all down, signal all legs to move back half-way.

Go back to step 1

3.5.9 The Legs

The legs consist of a set of nine states. When the in signal wire is set the leg moves to the next state. When the joint has completed the desired task the signal wire to the brain processor is set. The state cycle consists of the following:

3.5.10 Alpha

State 0: This is a initial state for legs 0, 3, and 4. This is just to let them hang until the brain is ready for them to move.

State 1: This is a dummy state where the joint does nothing.

State 2: This moves the alpha joint forward.

State 3: Dummy

State4: Move alpha joint back half way.

Goto state 6

State 5: Initial state for legs 1, 2, and 5.

Here the other 3 legs are moving So we stall for 3 states.

State 6: Dummy

State 7: Dummy

State 8: Dummy

State 9: Move alpha joint back.

Goto state 1.

Beta1

State 0: Initial state for legs 0, 3, 4.

State 1: Move beta 1 joint up.

State 2: Dummy, Allow the alpha joint to move forward.

State 3: Move beta 1 joint down.

State 4: Dummy, allow the alpha joint to move backward.

Goto state 6.

State 5: Initial state for legs 1, 2, 5.

States 6,7,8 and 9 allow the other three joints to move.

State 6: Dummy.

State 7: Dummy.

State 8: Dummy.

State 9: Dummy.

Goto state 1.

3.6 Future plans

Most of the current semester has been dedicated to the implementation of the lower levels of control. With these details finished the software group can now focus more attention on higher level details.

Some of the suggested future plans are:

- Solve the problems with the communication network so that the walking algorithm that was originally designed can be used. While it is possible for the robot to walk without communication it will not be possible to add any higher levels of intelligence until this issue is resolved.
- Make the software more robust. Currently the software does not make any provisions for any problems that may occur during walking. This includes providing a way for the software to recover in the event that the sensors do not return proper values.
- Addition of more sensors to allow higher level behaviors.
    1. Addition of a current sensor on each joint that could be used to stop the motor if the joint becomes stuck due to hitting an obstacle or mechanical failure.

2. Addition of sonar or other range sensing device to detect large obstacles in the path.

- Create new behaviors to add the ability to walk over uneven terrain and eventually climb stairs.

References

Brooks, Rodney A. 1986. A Robust Layered Control System for a Mobile Robot. IEEE *Journal of Robotics and Automation*, **RA-2**(April), 14-23.

Brooks, Rodney A. 1989. A Robot that Walks: Emergent Behavior from a Carfully Evolved Network. *Neural Computation*, **1**(2), 253-262.

Brooks, Rodney A. 1990. *The Behavior Language; User's Guide.* Tech. rept. 1277. MIT AI Lab Memo.

Cross, N. 1989. Engineering Design Methods. John Wiley & Sons.

Faires, Virgil Moring. 1965. *Design of Machine Elements*. Fourth edn. Toronto: The MacMillian Company.

Ulman, D.C. 1992. The Mechanical Design Process. McGraw-Hill, Inc.

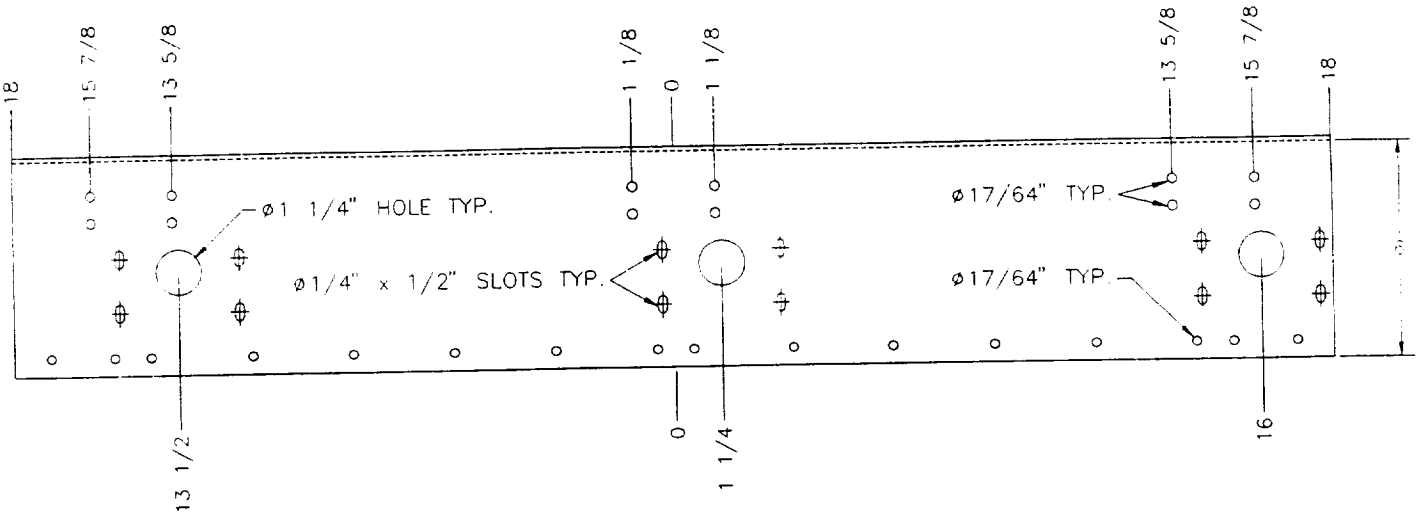
Yourdon, E. 1988. Managint the System Life Cycle. Prentice Hall.

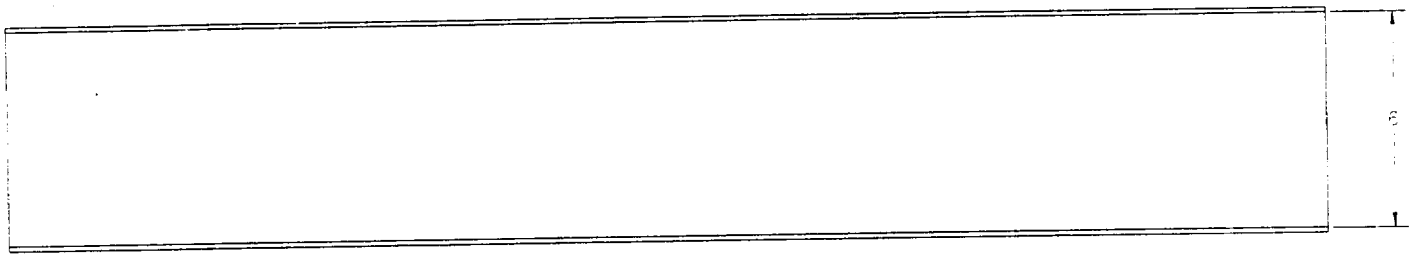
Appendices

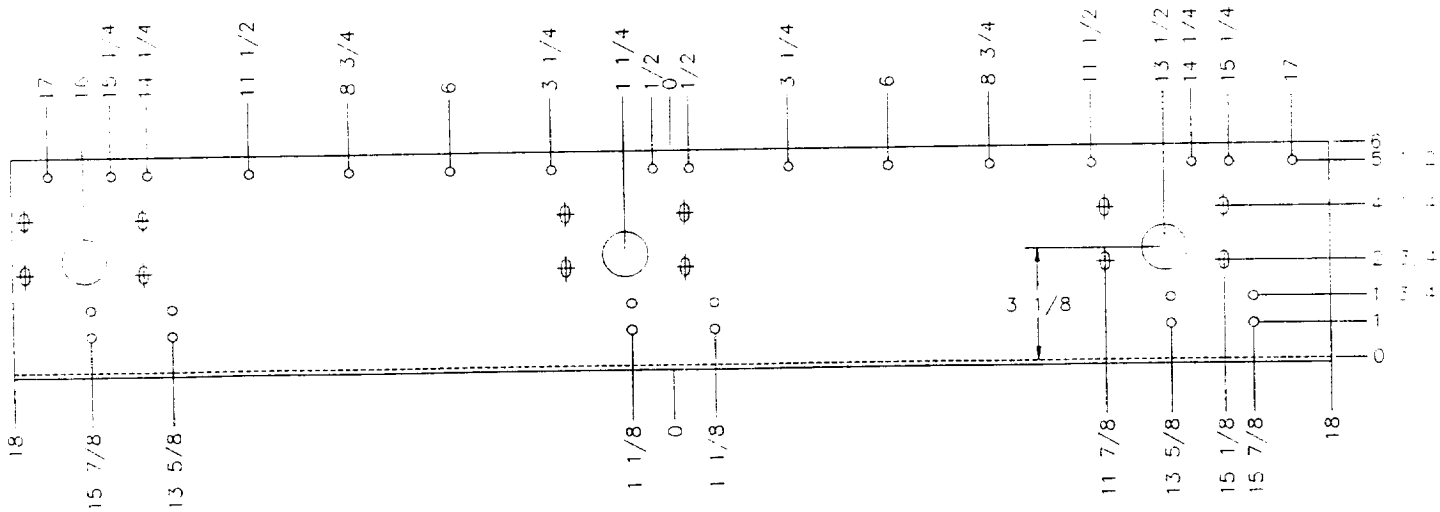
APPENDIX I.A

Mechanics Parts Drawings

Part Name	Description	Part #
Chassis	aluminum channel tube 6" x 6" x 36"	ME-CH-01.DWG
Chassis Angle	angle aluminum lip on open edges of chassis	ME-CH-02.DWG
Chassis Braces	tie rods used to secure open side of chassis (top)	ME-CH-03.DWG
Chassis Cover	plexiglass covering for top of chassis	ME-CH-04.DWG
Chassis Bends	preliminary bending layout	ME-CH-05.DWG
Alpha Face Plate	motor face plate attaching to inside chassis wall	ME-J1-01.DWG
Alpha Axle	axle that alpha joint rotates about	ME-J1-02.DWG
Alpha Lower Mount	lower alpha hinge on chassis side	ME-J1-03.DWG
Alpha Upper Mount	upper alpha hinge on chassis side	ME-J1-04.DWG
Alpha Sensor Mount	mount used to hold pot and limit switches	ME-J1-05.DWG
Beta1 Face Plate	motor face plate that fits into bat yoke	ME-J2-01.DWG
Beta1 Axle	joint axle	ME-J2-02.DWG
Bat Yoke	upper beta yoke attached to alpha axle	ME-J2-03.DWG
Beta1 Coupling	coupling for beta1 motor to screw drive	ME-J2-04.DWG
Threaded Shaft	screw drive threaded shaft	ME-J2-05.DWG
Screw Collar	screw drive pivot collar	ME-J2-06.DWG
Beta1 Lever	drive lever attached to beta1 leg	ME-J2-07.DWG
Threaded Shaft	screw drive threaded shaft	ME-J2-08.DWG
Beta2 Face Plate	motor face plate	ME-J3-01.DWG
Beta2 Axle	joint axle	ME-J3-02.DWG
Beta2 Mount	mount attaching face plate to leg	ME-J3-03.DWG
Leg Lengths	materials and leg length drawings	ME-LG-LL.DWG
Disk Interrupter	photo disk interruptor mounted on joint axles	ME-LG-DI.DWG
Beta Sensor Mount	mount for pot. & photo switch for Beta joints	ME-LG-SM.DWG
Beta Joint	joint of rotation on Beta joints	ME-LG-BJ.DWG



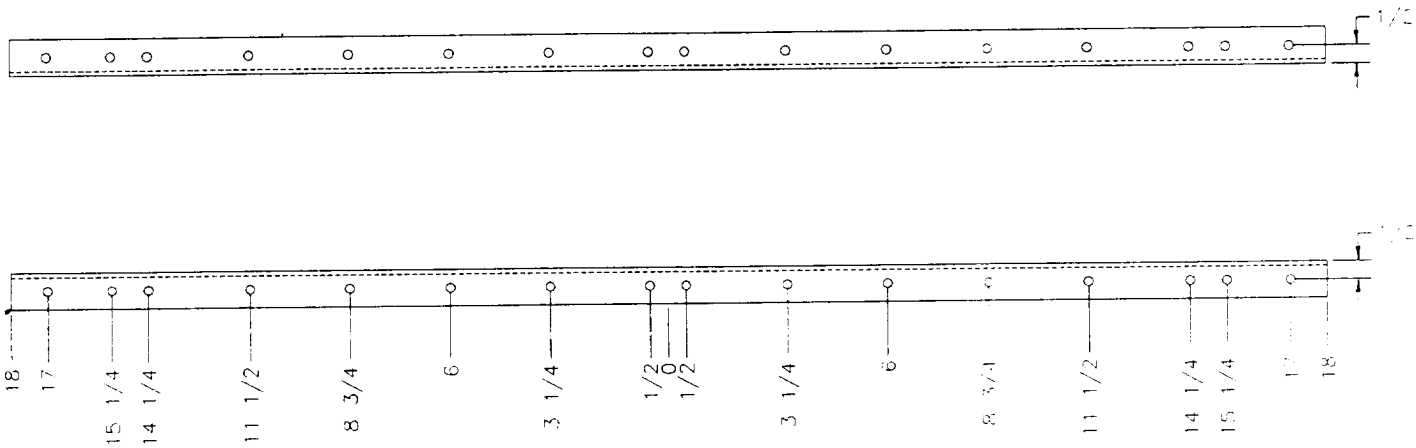




CHASSIS

(28) ø17/64" HOLES
(12) ø1/4" x 1/2" SLOTS
(3) 1 1/4" HOLES
TYPICAL EACH SIDE

K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-CH-01.DWG	SCALE: 3/16" = 1"
DWG. BY: E. AMES	DATE: 03-14-94



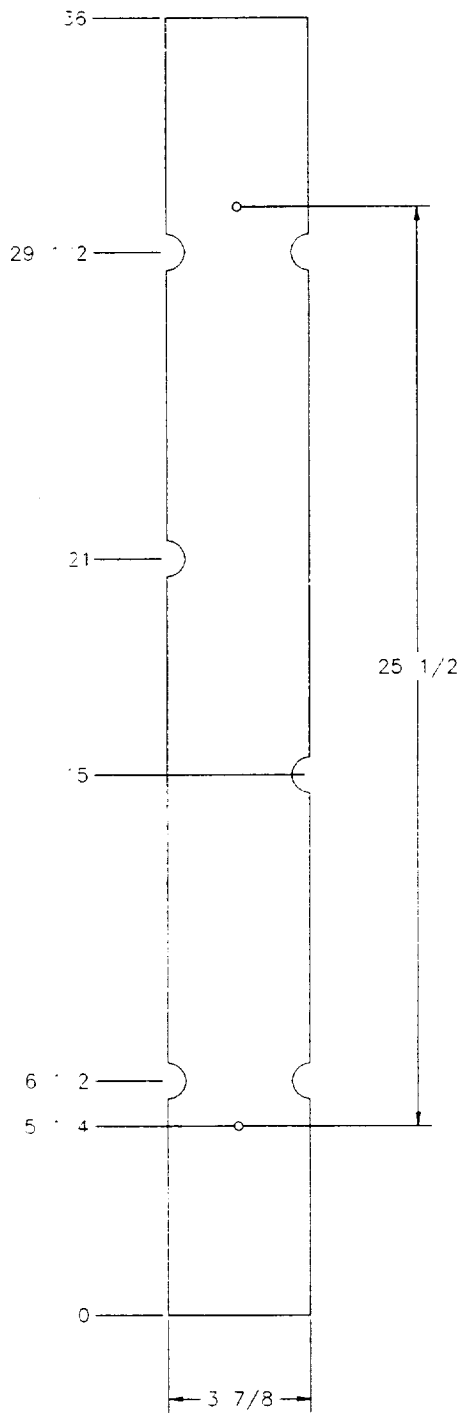
CHASSIS LIP

1" x 1" x 36"
(32) ø17/64" HOLES
(2) REQUIRED

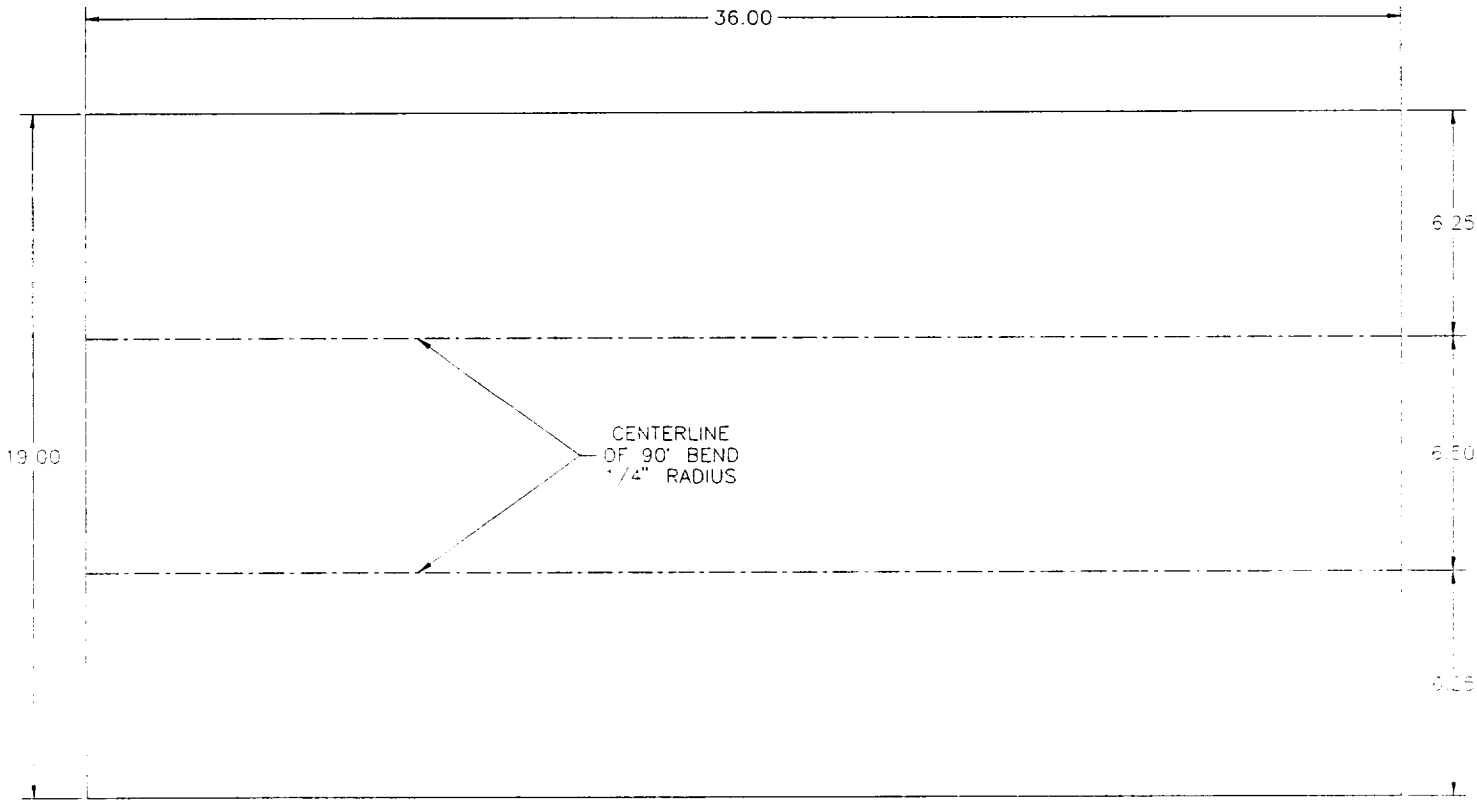
K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-CH-02.DWG	SCALE: 3/16" = 1"
DWG. BY: E. AMES	DATE: 03-14-94

CHASSIS COVER

1/4" PLEXIGLASS
(2) ø1/4" HOLES
(6) ø1" SLOTS
(1) REQUIRED



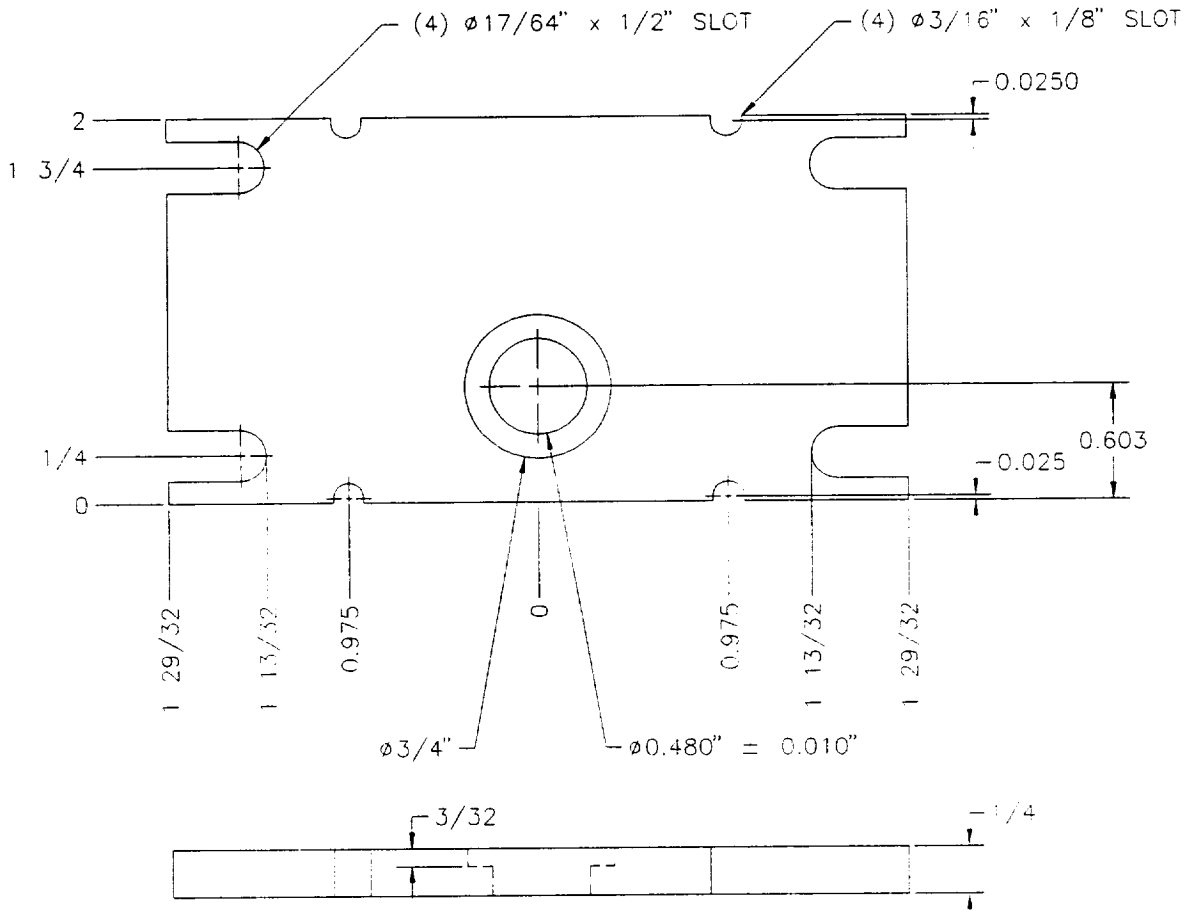
K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-CH-04.DWG	SCALE: 3/16" = 1"
DWG. BY: E. AMES	DATE: 03-14-94



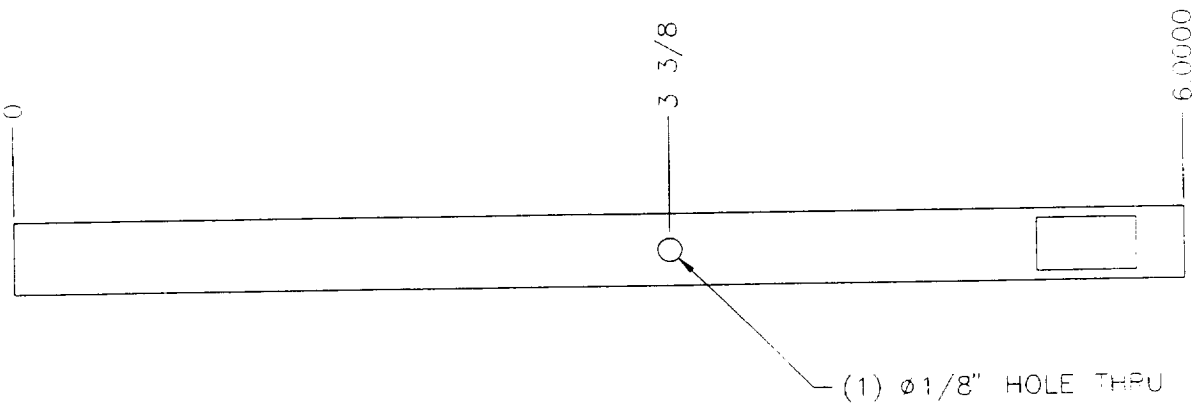
PRELIMINARY BENDING LAYOUT
FOR BODY CHASSIS
1/8" SHEET

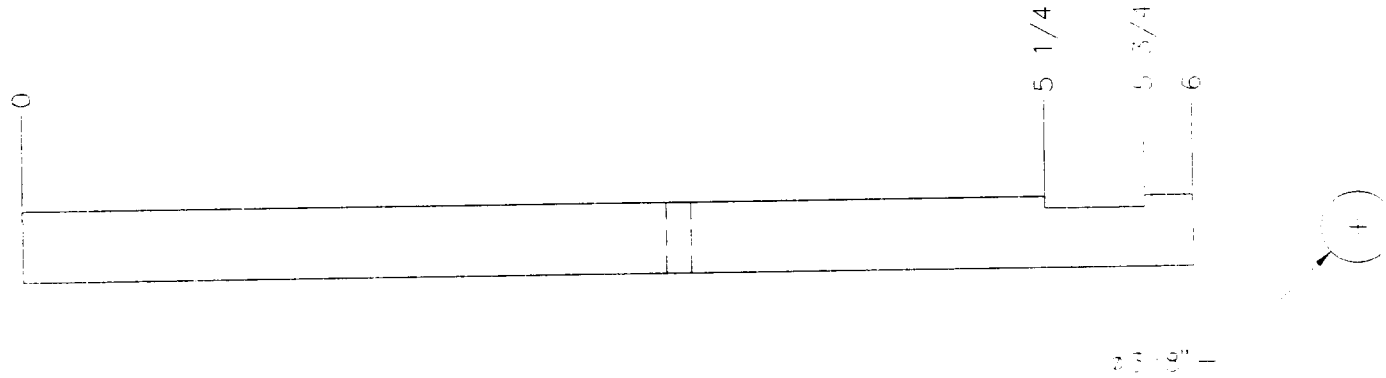
K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-CH-05.DWG	SCALE: 3/16" = 1"
DWG. BY: E. AMES	DATE: 03-16-94

ALPHA FACE PLATE
(6) REQUIRED



K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-J1-01.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 02-22-94

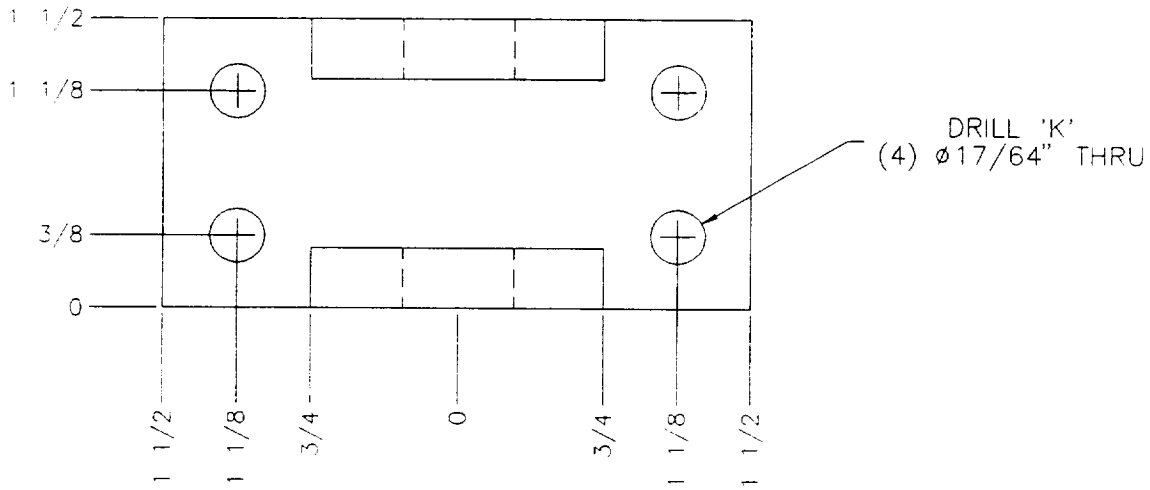


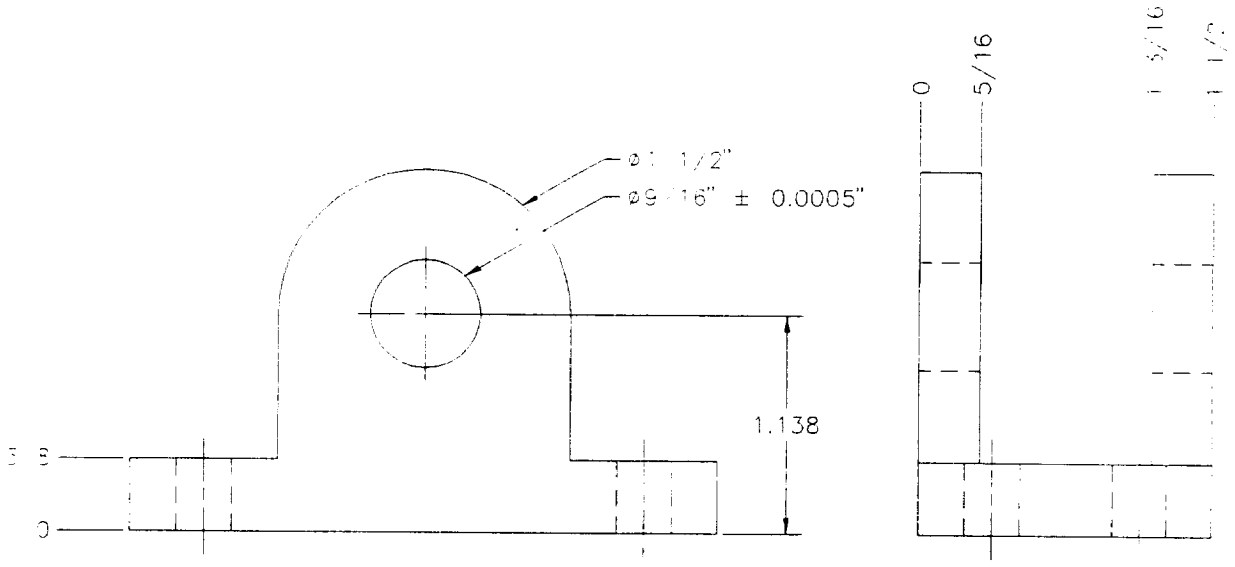


ALPHA AXLE
ø3/8" x 6" STEEL ROD
(6) REQUIRED

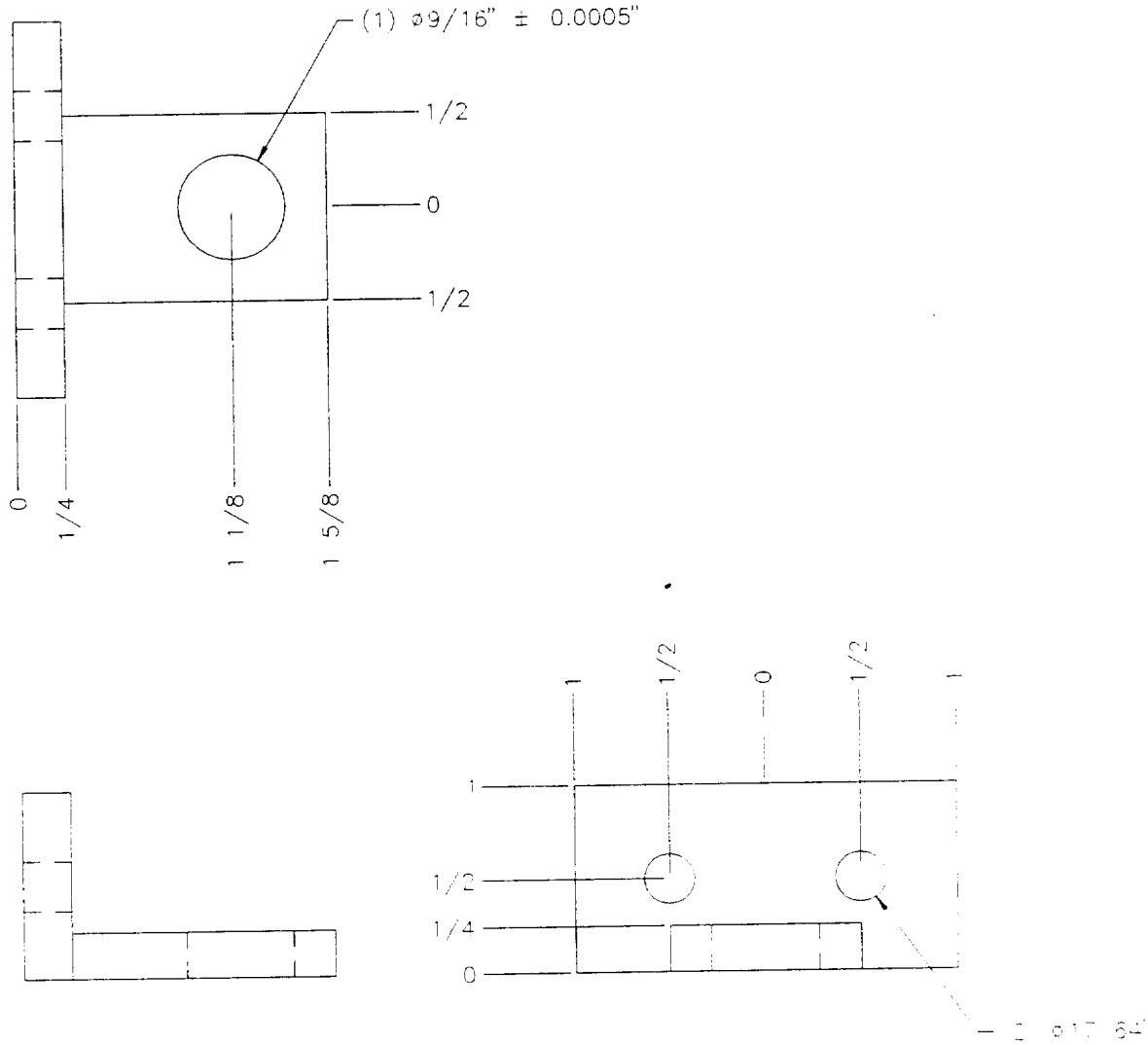
K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
DWG: ME-J1-02.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 03-14-94

ALPHA LOWER MOUNT
(6) REQUIRED





K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-J1-03.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 03-14-94

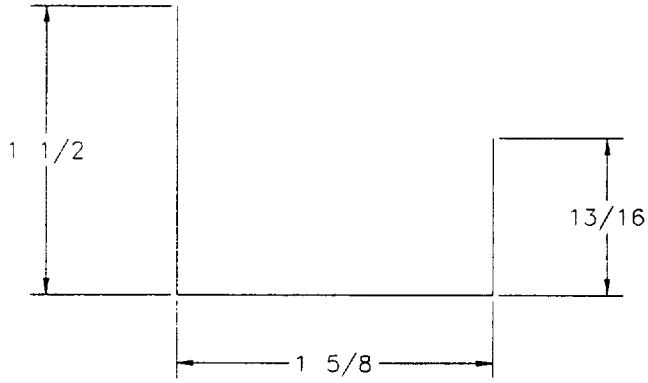


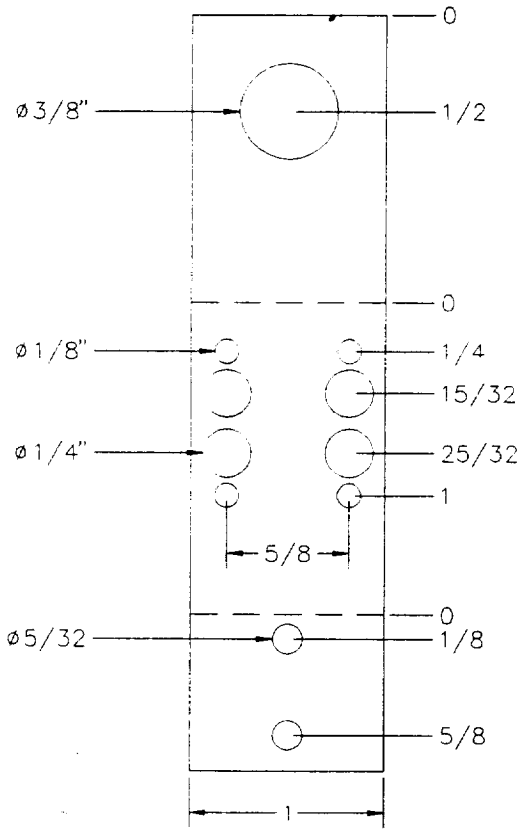
UPPER ALPHA MOUNT
(2) ø17/64" HOLES
(1) ø9/16" HOLE
(6) REQUIRED

K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-J1-04.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 03-14-94

ALPHA SENSOR MOUNT

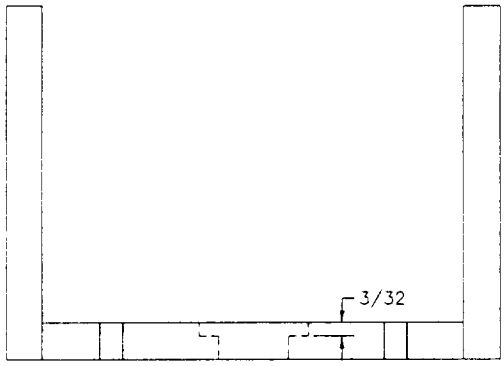
LIMIT SWITCH AND POTENTIOMETER
(12) REQUIRED

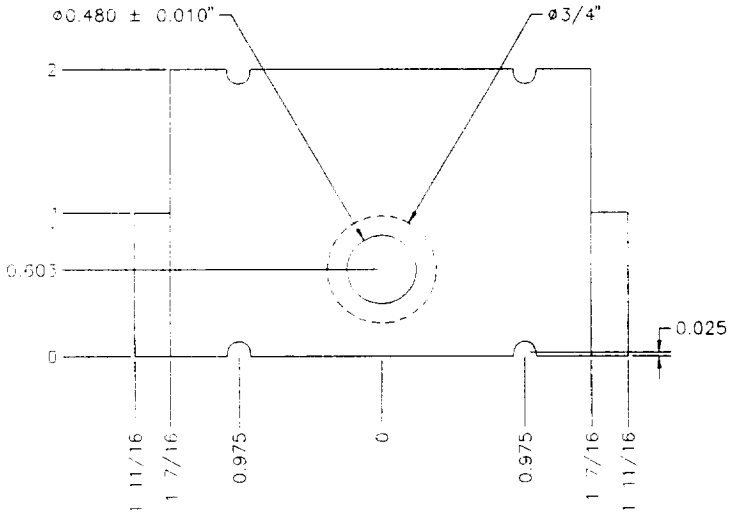


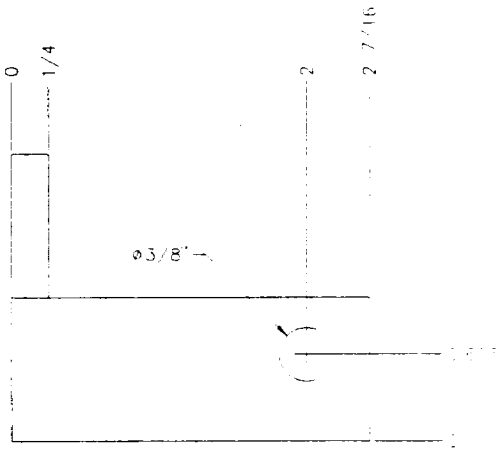


K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
DWG: ME-J1-05.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 03-14-94

BETA1 FACE PLATE
(6) REQUIRED

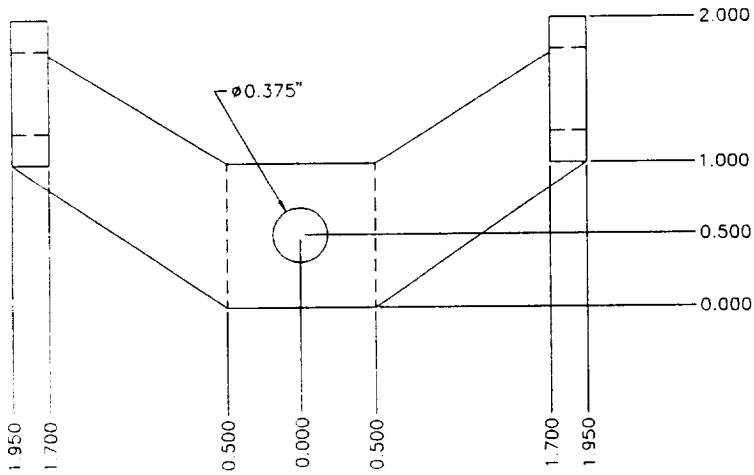


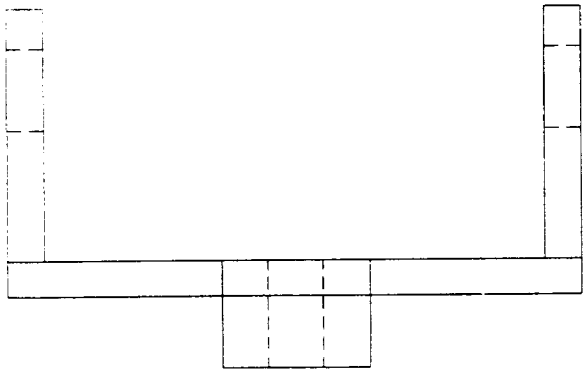


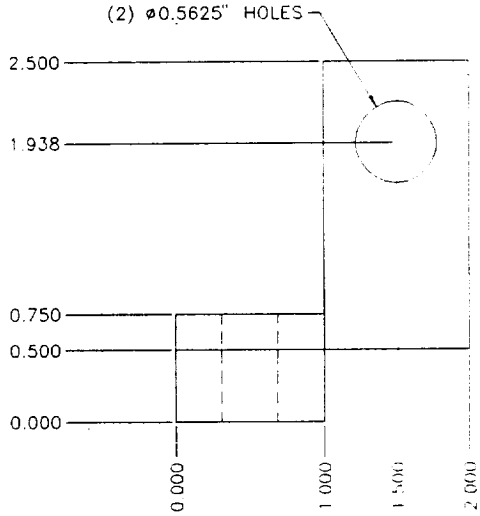


K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-J2-01.DWG	SCALE: 3/4" =
DWG. BY: E. AMES	DATE: 02-22-94

BAT YOKE
(6) REQUIRED



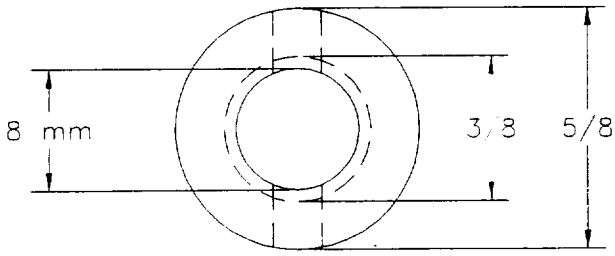


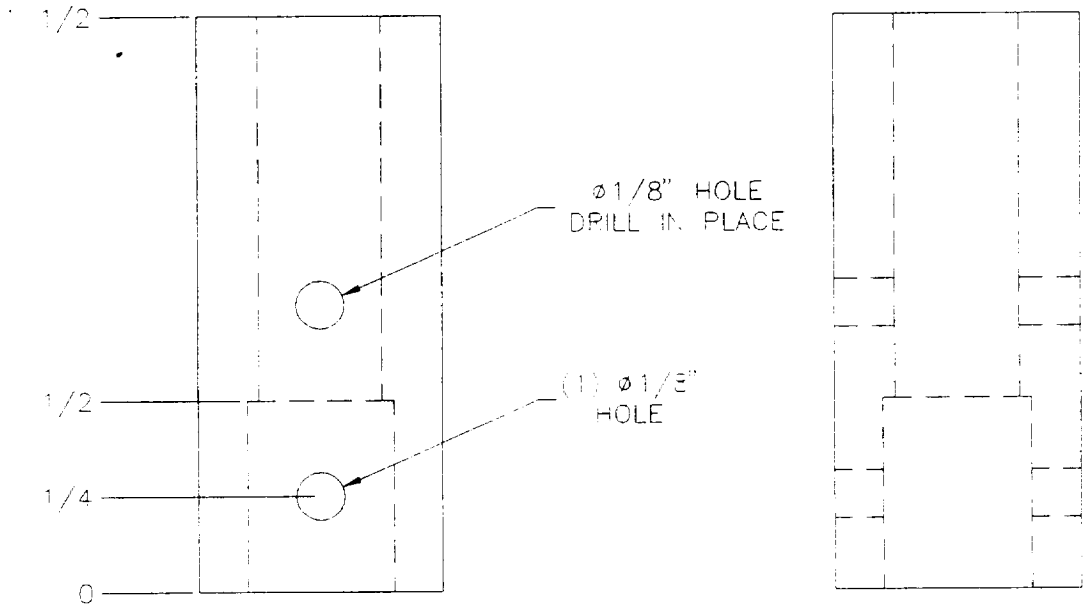


K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: MD-J2-03.DWG	SCALE: 3/4" = 1"
DWG. BY: E. AMES	DATE: 05-04-94

BETA1 COUPLING

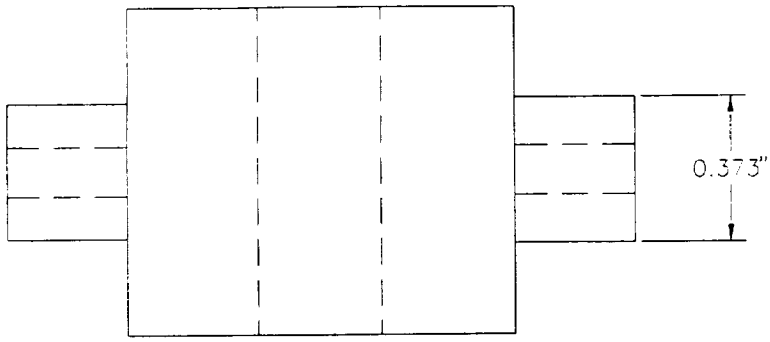
6) REQUIRED

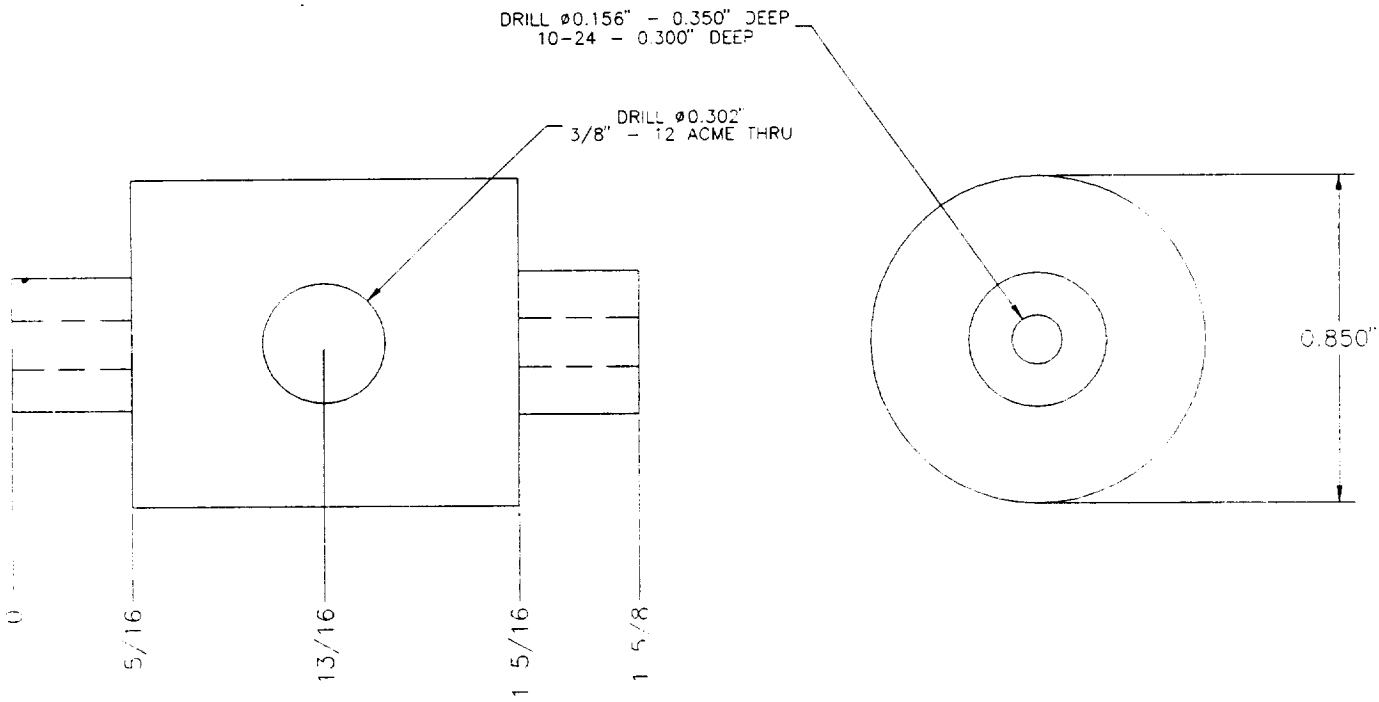




K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
DWG: ME-J2-04.DWG	SCALE: 2" = 1"
DWG. BY: E. AMES	DATE: 05-03-94

SCREW COLLAR
(6) REQUIRED



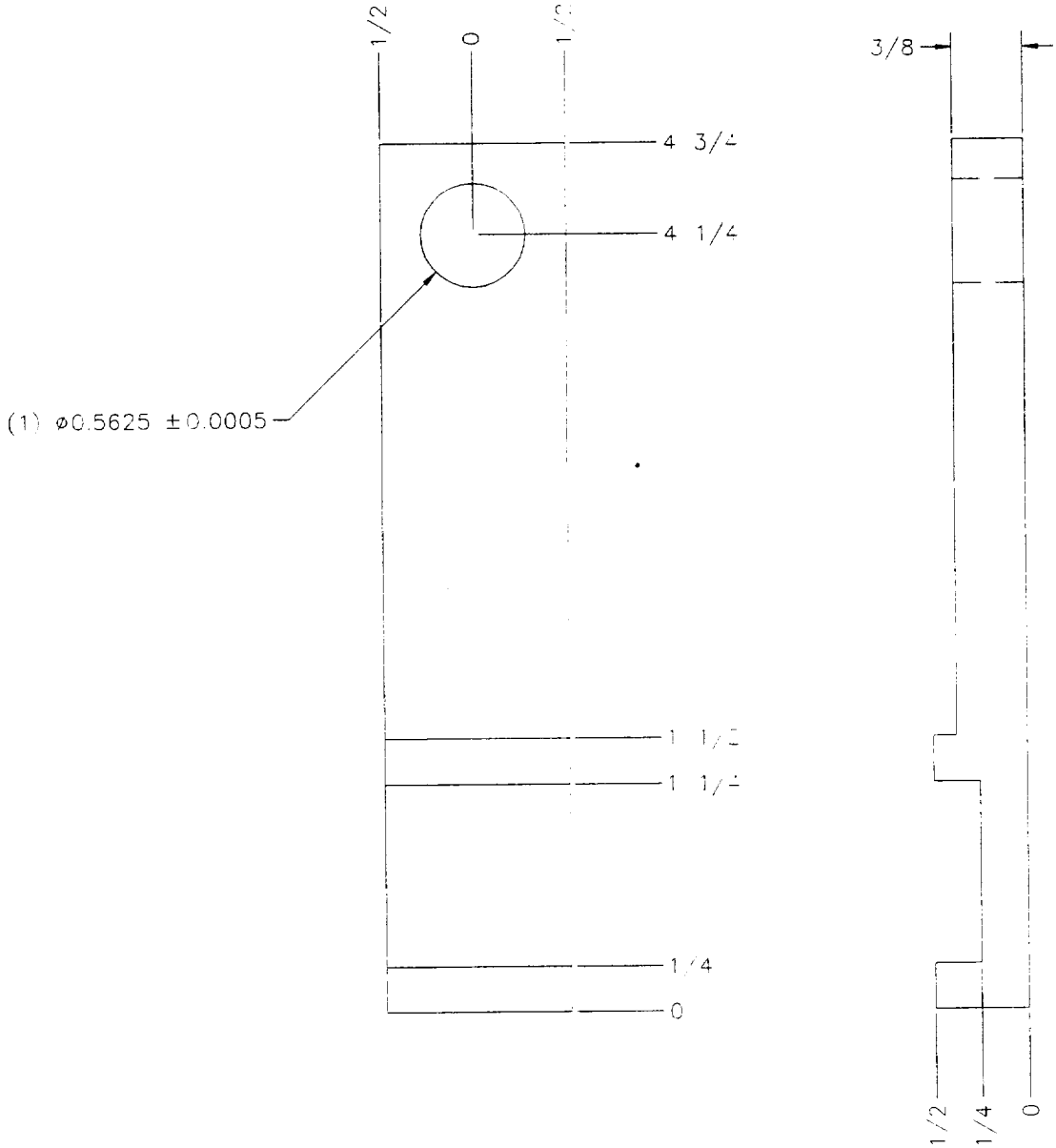


K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
PART: ME-J2-06.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 04-27-94

LEVER ARM

(12) REQUIRED

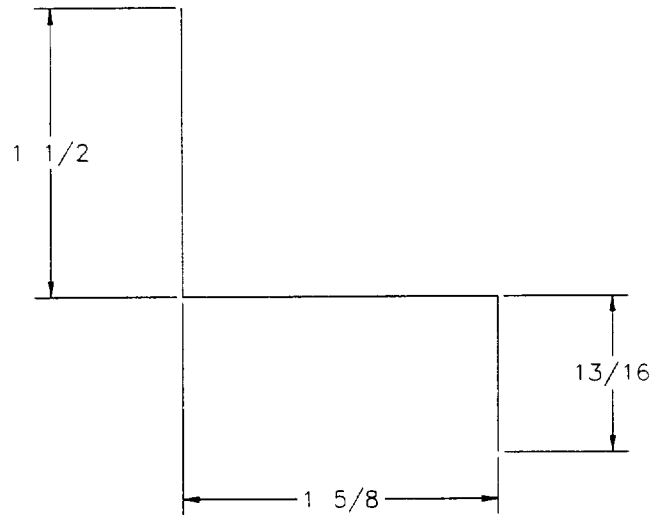
(1) ø0.5625 ±0.0005 HOLE

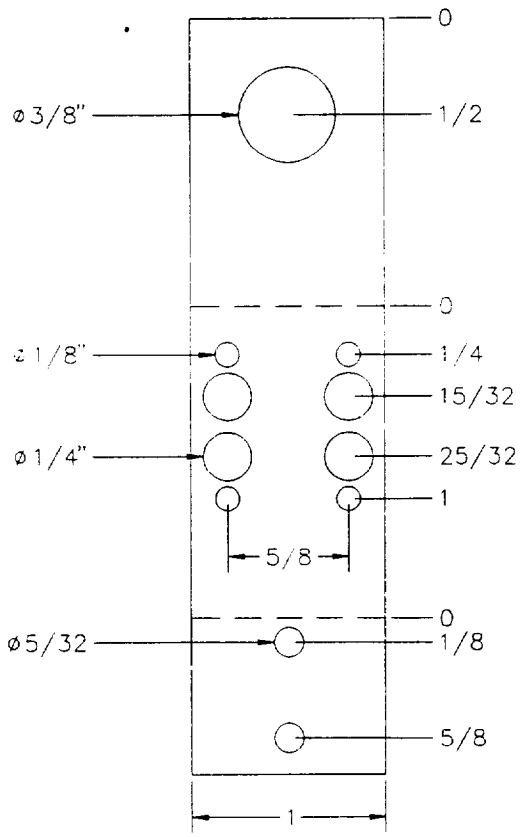


K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
DWG: ME-J2-07.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 05-03-94

BETA SENSOR MOUNT

LIMIT SWITCH AND POTENTIOMETER
(12) REQUIRED





K-STATE	
KSU ENGINEERING	NASA DESIGN TEAM
DWG: ME-LG-SM.DWG	SCALE: 1" = 1"
DWG. BY: E. AMES	DATE: 04-27-94

APPENDIX I.B

Mechanical Failure Analysis

Description of Part Analyzed	Part Number
Alpha Lower Joint Mount	ME-J1-03.DWG
Beta 1 Face Plate	ME-J2-01.DWG
Beta 1 Motor mount (Bat Yoke)	ME-J2-03.DWG
Beta 1 Lever Arm	ME-J2-07.DWG

Alpha Lower Mount
(Part drawing ME-J1-03)

Anticipated ways of failure:

1) Tension across shaft hole
2) Shear of Bolt

Definitions

A = cross sectional area
S = Shear modulas
N = Safety factor
F = Force
F_M = Maximum force

A safety factor "N" of 6 was used.

All forces are in Pound-force.

1) Tension across shaft hole.(Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (A \bullet S_U)/N$
   F_M = 4357.05

2) Shear of Bolt. .(Material - Steel C1020)
   S_{US} = 50 Ksi
   $F = (4 \bullet A \bullet S_{US})/N$
   F_M = 2025.85

BETA 1 FACE PLATE
(Part drawing ME-J2-01)

Anticipated ways of failure:

1) Tension across pin hole
2) Shear of Bolt

Definitions

A = cross sectional area
S = Shear modulas
N = Safety factor
F = Force
F_M = Maximum force

A safety factor "N" of 6 was used.

All forces are in Pound-force.

1) Tension across pin hole.(Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (A \bullet S_U)/N$
   F_M =2343.75

2) Shear of bolt. (Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (4 \bullet A \bullet S_U)/N$
   F_M = 4797.7

BAT YOKE
(Part drawing ME-J2-03)

Anticipated ways of failure:

1) Tension across pin hole
2) Shear across Mid-plate
3) Tension across shaft hole

Definitions

A = cross sectional area
S = Shear modulas
N = Safety factor
F = Force
F_M = Maximum force

A safety factor "N" of 6 was used.

All forces are in Pound-force.

1) Tension across pin hole.(Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (A \bullet S_U)/N$
   F_M =1642.5

2) Shear across mid-plate. (Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (2 \bullet A \bullet S_U)/N$
   F_M = 7500

3) Tension across shaft hole. (Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (A \bullet S_U)/N$
   F_M = 3313.4

LEVER ARM
(Part drawing ME-J2-07)

Anticipated ways of failure:

1) Tension across shaft hole
2) Shear of Bolt
3) Shear of Tabs

Definitions

A = cross sectional area
S = Shear modulas
N = Safety factor
F = Force
F_M = Maximum force

A safety factor "N" of 6 was used.

All forces are in Pound-force.

1) Tension across shaft hole.(Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (A \bullet S_U)/N$
   F_M = 1016.025

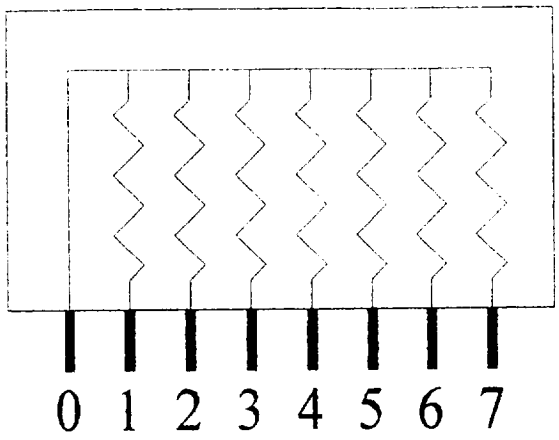
2) Shear of Bolt. .(Material - Steel C1020)
   S_{US} = 50 Ksi
   $F = (2 \bullet A \bullet S_{US})/N$
   F_M = 528

3) Shear of Tabs.(Material - Al 6061-T6)
   S_U = 40 Ksi
   $F = (2 \bullet A \bullet S_U)/N$
   F_M = 625

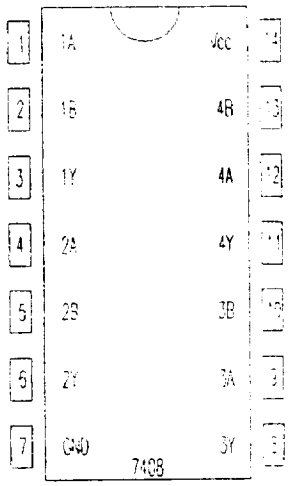
APPENDIX II.A

Pin Outs

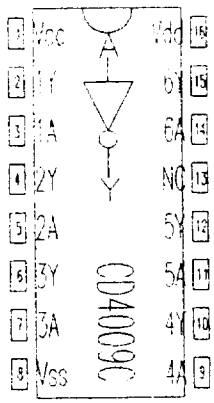
Resistor Pack



74LS08



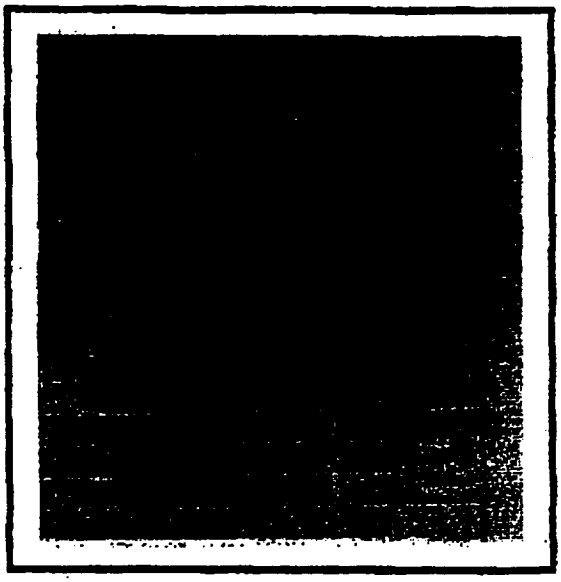
CD4009

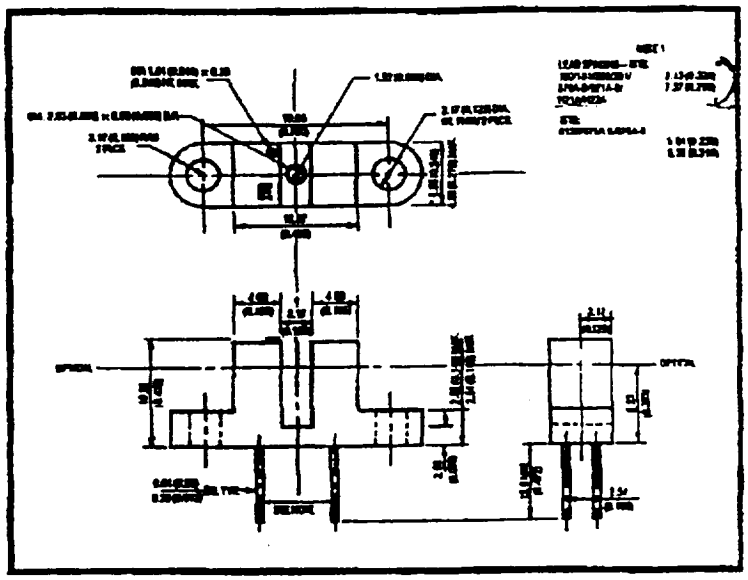


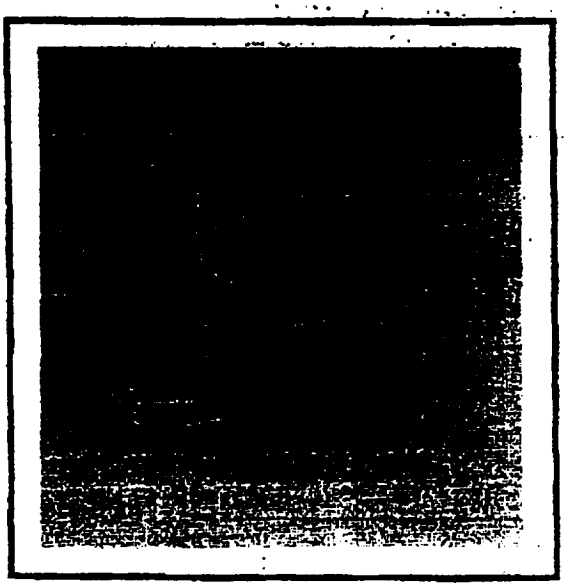
APPENDIX II.B

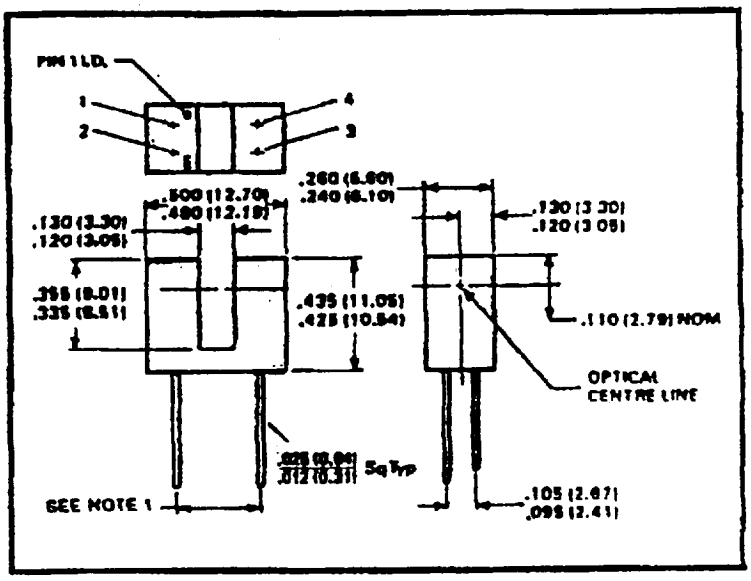
Optical Interrupter Data Sheets

H21A1, H21A2, H22A1, H22A2
Opto-electronic slotted switches









DESCRIPTION:

The Isocom H21XX and H22XX Series offers precision sensing in a low cost plastic housing switch. The housing filters out the visible light while allowing light passage in the infrared spectrum.

The switch consists of a gallium arsenide emitting diode and an NPN silicon photo transistor.

All electrical parameters are 100% tested by Isocom.

ABSOLUTE MAXIMUM RATINGS (25° unless otherwise noted)

Storage Temperature Range	-55°C TO +100°C
Operating Temperature Range	-55°C TO +100°C
Lead Soldering Temperature	260°C

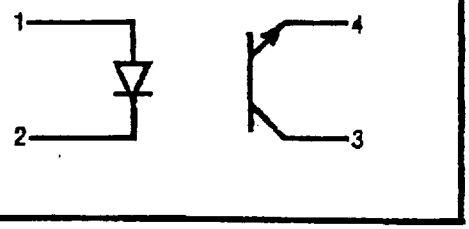
Input Diode

Forward Current	IF	60mA
Peak Forward Current	IFM	3A
Reverse Voltage	VR	6V
Power Dissipation	P	100mW

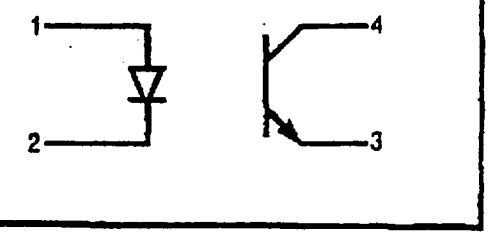
Output Transistor

Collector-Emitter Voltage	VCEO	30V
Emitter-Collector Voltage	VECO	6v
Collector Current	IC	100mA
Collector Power Dissipation	PC	150mW

1



2



ELECTRICAL CHARACTERISTICS (25°C UNLESS OTHERWISE NOTED)

SYMBOL	PARAMETER CONDITIONS	CHARACTERISTICS			
		MIN	TYP	MAX	UNIT
VF	IF = 60mA	—	—	1.7	V
IR	VR = 5V	—	—	100	
BVCEO	IC = 1mA	30V	—	—	V
BVCEO	IE = 100uA, IF = 0	6V	—	—	V
ICEO	VCE = 25V-IF = 0-IH = 0	—	—	100	
VCE (SAT)	IC = 18-IF = 20mA-If = 30mA	—	—	0.4	V
IC (ON)	IF = 5mA-VCE = 5V	0.15 mA	—	—	mA
IC (ON)	IF = 20mA-VCE = 5V	1mA	—	—	mA
IC (ON)	IF = 30mA-VCE = 5V	1.9 mA	—	—	mA
IC (ON)	IF = 5mA, VCE = 5V		—	—	mA
IC (ON)	IF = 20mA-VCE = 5V	2mA	—	—	mA
IC (ON)	IF = 30mA-VCE = 5V	3.0	—	—	
t (ON)	VCC = 5V-IF = 30mA RL = 2.5K	—	6uS	—	
t (OFF)	VCC = 5V-IF = 30mA RL = 2.5K	—	50uS	—	

**PARAMETER FOR H21A2—ALL OTHER CHARACTERISTICS ARE THE SAME*

APPENDIX II.C

Optical Encoder Data Sheets

HEWLETT PACKARD

Quick Assembly Two and Three Channel Optical Encoders

Technical Data

HEDS-5500/5540
HEDS-5600/5640

Features

- Two Channel Quadrature Output with Optional Index Pulse
- Quick and Easy Assembly
- No Signal Adjustment Required
- External Mounting Ears Available
- Low Cost
- Resolutions Up to 512 Counts Per Revolution
- Small Size
- -40°C to 100°C Operating Temperature
- TTL Compatible
- Single 5 V Supply

Description

The HEDS-5500/5540 and 5600/5640 are high performance, low cost, two and three channel optical incremental encoders. These encoders emphasize high reliability, high resolution, and easy assembly.

Each encoder contains a lensed LED source, an integrated circuit with detectors and output circuitry, and a codewheel which rotates between the emitter and detector IC. The outputs of the HEDS-5500 and 5600 are two square waves in quadrature. The HEDS-5540 and 5640 also have a third channel index output in addition to the two channel quadrature. This index output is a 90 electrical degree, high true index pulse which is generated once for each full rotation of the codewheel.

These encoders may be quickly and easily mounted to a motor. For larger diameter motors, the HEDS-5600/5640 feature external mounting ears.

The quadrature signals and the index pulse are accessed through five 0.025 inch square pins located on 0.1 inch centers.

Standard resolutions between 96 and 512 counts per revolution are presently available. Consult local Hewlett-Packard sales representatives for other resolutions.

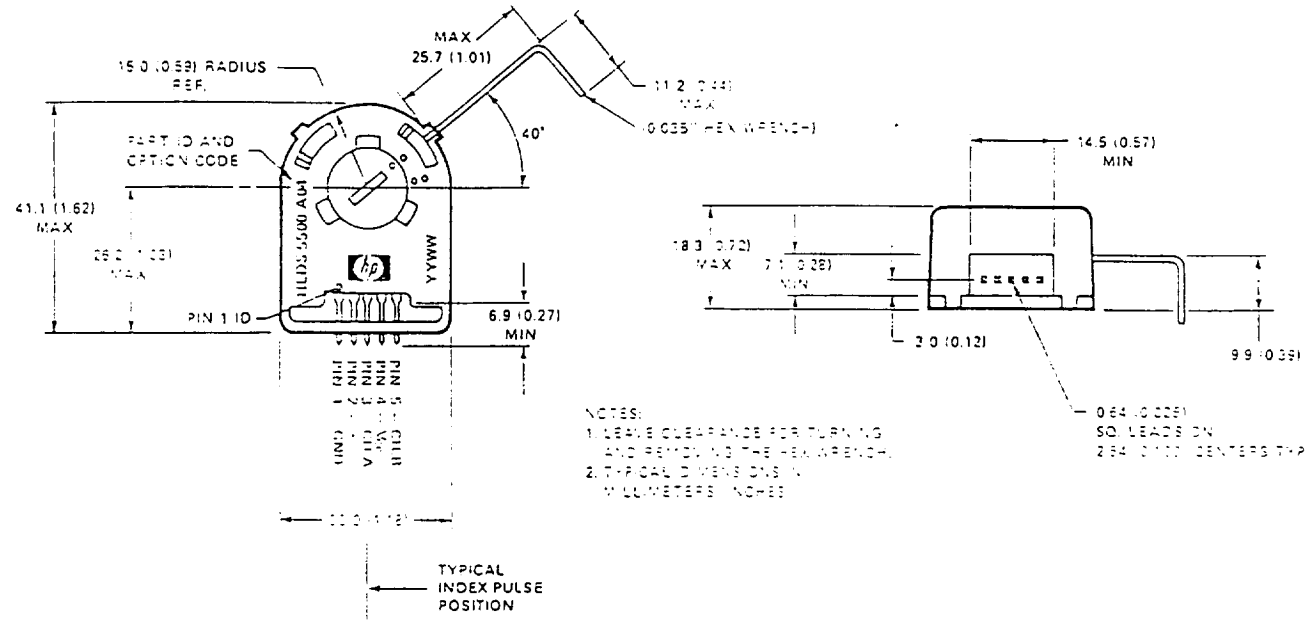
Applications

The HEDS-5500, 5540, 5600, and 5640 provide motion detection at a low cost, making them ideal for high volume applications. Typical applications include printers, plotters, tape drives, positioning tables, and automatic handlers.

ESD WARNING; NORMAL HANDLING PRECAUTIONS SHOULD BE TAKEN TO AVOID STATIC DISCHARGE.

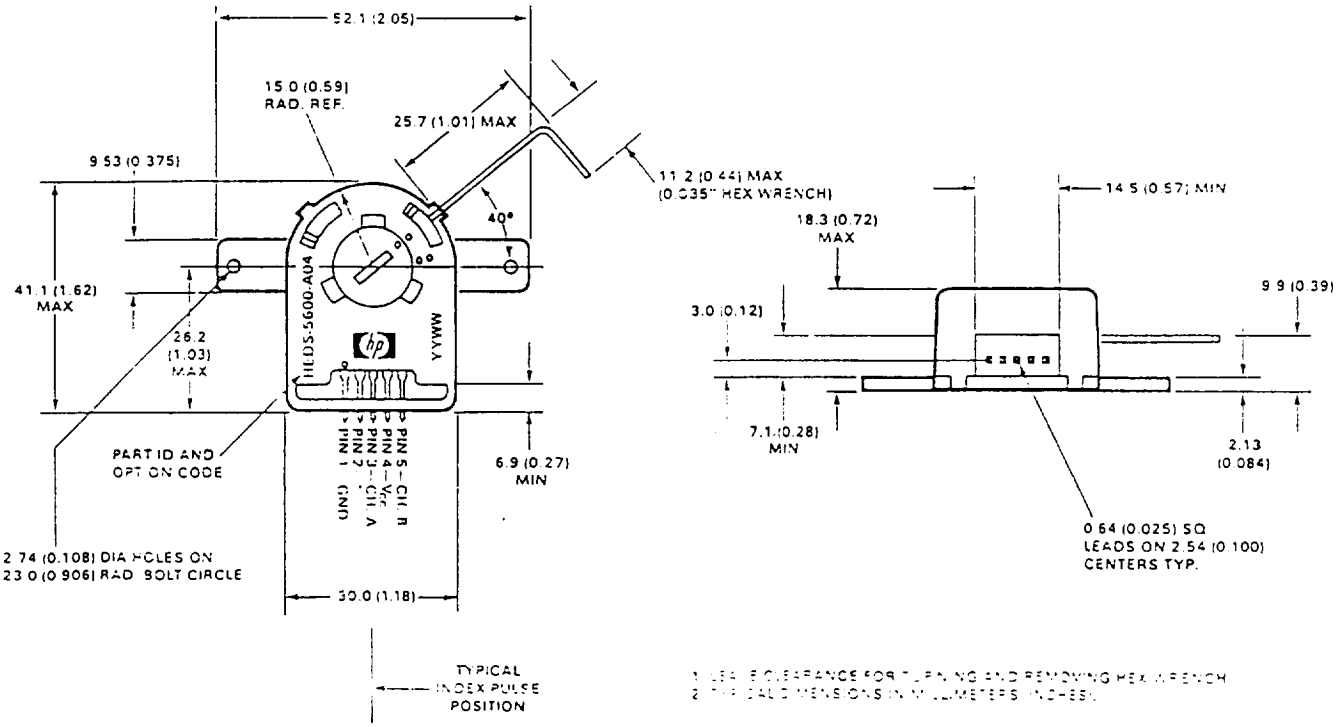
Package Dimensions

HEDS-5500/5540



*Note: For the HEDS-5500, Pin #2 is a No Connect. For the HEDS-5540, Pin #2 is CH. I, the index output.

HEDS-5600/5640



*Note: For the HEDS-5600, Pin #2 is a No Connect. For the HEDS-5640, Pin #2 is CH. I, the index output.

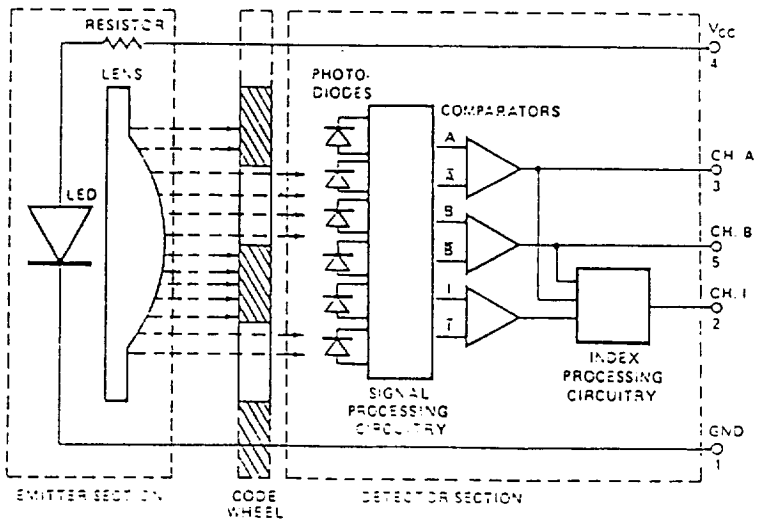
Theory of Operation

The HEDS-5500, 5540, 5600, and 5640 translate the rotary motion of a shaft into either a two- or a three-channel digital output.

As seen in the block diagram, these encoders contain a single Light Emitting Diode (LED) as its light source. The light is collimated into a parallel beam by means of a single polycarbonate lens located directly over the LED. Opposite the emitter is the integrated detector circuit. This IC consists of multiple sets of photodetectors and the signal processing circuitry necessary to produce the digital waveforms.

The codewheel rotates between the emitter and detector, causing the light beam to be interrupted by the pattern of spaces and bars on the codewheel. The photodiodes which detect these interruptions are arranged in a pattern that corresponds to the radius and design of the codewheel. These detectors are also spaced such that a light period on one pair of detectors corresponds to a dark period on the adjacent pair of detectors. The photodiode outputs are then fed through the signal processing circuitry resulting in A, $\overline{A}$, B and $\overline{B}$ (also I and $\overline{I}$ in the HEDS-5540 and 5640). Comparators receive these signals and produce the final outputs for channels A and B. Due to this integrated phasing technique, the digital output of channel A is in quadrature with that of channel B (90 degrees out of phase).

Block Diagram



NOTE: CIRCUITRY FOR CH. I IS ONLY IN HEDS-5540 AND 5640 THREE CHANNEL ENCODERS.

In the HEDS-5540 and 5640, the output of the comparator for I and $\overline{I}$ is sent to the index processing circuitry along with the outputs of channels A and B. The final output of channel I is an index pulse P_o which is generated once for each full rotation of the codewheel. This output P_o is a one state width (nominally 90 electrical degrees), high true index pulse which is coincident with the low states of channels A and B.

Definitions

Count (N): The number of bar and window pairs or counts per revolution (CPR) of the codewheel.

One Cycle (C): 360 electrical degrees (°e), 1 bar and window pair.

One Shaft Rotation: 360 mechanical degrees, N cycles.

Position Error ($\Delta\Theta$): The normalized angular difference between the actual shaft position and the position indicated by the encoder cycle count.

Cycle Error (ΔC): An indication of cycle uniformity. The difference between an observed shaft angle which gives rise to one electrical cycle, and the nominal angular increment of 1/N of a revolution.

Pulse Width (P): The number of electrical degrees that an output is high during 1 cycle. This value is nominally 180°e or 1/2 cycle.

Pulse Width Error (ΔP): The deviation, in electrical degrees, of the pulse width from its ideal value of 180°e.

State Width (S): The number of electrical degrees between a transition in the output of channel A and the neighboring transition in the output of channel B. There are 4 states per cycle, each nominally 90°e.

State Width Error (ΔS): The deviation, in electrical degrees, of each state width from its ideal value of 90°e.

Phase (ϕ): The number of electrical degrees between the center of the high state of channel A and the center of the high state of channel B. This value is nominally 90°e for quadrature output.

Phase Error ($\Delta\phi$): The deviation of the phase from its ideal value of 90°e.

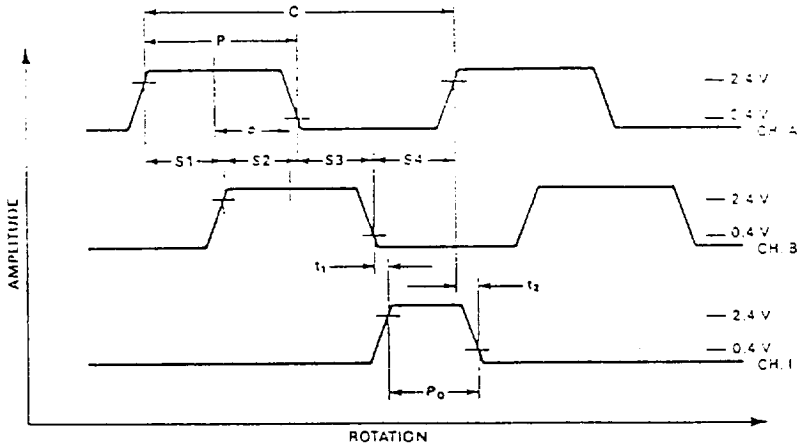
Direction of Rotation: When the codewheel rotates in the counterclockwise direction (as viewed from the encoder end of the motor), channel A will lead channel B. If the codewheel rotates in the clockwise direction, channel B will lead channel A.

Index Pulse Width (P_o): The number of electrical degrees that an index output is high during one full shaft rotation. This value is nominally 90°e or 1/4 cycle.

Absolute Maximum Ratings

Storage Temperature, T_S ..... -40°C to 100°C
Operating Temperature, T_A ..... -40°C to 100°C
Supply Voltage, V_{CC} ..... -0.5 V to 7 V
Output Voltage, V_O ..... -0.5 V to V_{CC}
Output Current per Channel, I_{OUT} ..... -1.0 mA to 5 mA
Vibration ..... 20 g, 5 to 1000 Hz
Shaft Axial Play ..... ±0.25 mm (±0.010 in.)
Shaft Eccentricity Plus Radial Play ..... 0.1 mm (0.004 in.) TIR
Velocity ..... 30,000 RPM
Acceleration ..... 250,000 rad/sec²

Output Waveforms



Recommended Operating Conditions

Parameter	Symbol	Min.	Typ.	Max.	Units	Notes
Temperature	T_A	-40		100	°C	
Supply Voltage	V_{CC}	4.5	5.0	5.5	Volts	Ripple < 100 mV_{P-P}
Load Capacitance	C_L			100	pF	2.7 kΩ pull-up
Count Frequency	f			100	kHz	Velocity (rpm) x N/60
Shaft Perpendicularity Plus Axial Play				±0.25 (±0.010)	mm (in.)	6.9 mm (0.27 in.) from mounting surface
Shaft Eccentricity Plus Radial Play				0.04 (0.0015)	mm (in.) TIR	6.9 mm (0.27 in.) from mounting surface

Note: The module performance is guaranteed to 100 kHz but can operate at higher frequencies. 2.7 kΩ pull-up resistors required for HEDS-5540 and 5640.

Encoding Characteristics

Encoding Characteristics over Recommended Operating Range and Recommended Mounting Tolerances unless otherwise specified. Values are for the worst error over the full rotation.

Part No.	Description		Sym.	Min.	Typ.*	Max.	Units
HEDS-5500 HEDS-5600 (Two Channel)	Pulse Width Error		ΔP		7	45	°e
	Logic State Width Error		ΔS		5	45	°e
	Phase Error		$\Delta\phi$		2	20	°e
	Position Error		$\Delta\Theta$		10	40	min. of arc
	Cycle Error		ΔC		3	5.5	°e
HEDS-5540 HEDS-5640 (Three Channel)	Pulse Width Error		ΔP		5	35	°e
	Logic State Width Error		ΔS		5	35	°e
	Phase Error		$\Delta\phi$		2	15	°e
	Position Error		$\Delta\Theta$		10	40	min. of arc
	Cycle Error		ΔC		3	5.5	°e
	Index Pulse Width		P_o	55	90	125	°e
	CH. I rise after CH. A or CH. B fall	-25°C to +100°C -40°C to +100°C	t_1 t_1	10 -300	100 100	250 250	ns ns
	CH. I fall after CH. B or CH. A rise	-25°C to +100°C -40°C to +100°C	t_2 t_2	70 70	150 150	300 1000	ns ns

Note: See Mechanical Characteristics for mounting tolerances.

Electrical Characteristics

Electrical Characteristics over Recommended Operating Range.

Part No.	Parameter	Sym.	Min.	Typ.*	Max.	Units	Notes
HEDS-5500 HEDS-5600	Supply Current	I_{CC}		17	40	mA	
	High Level Output Voltage	V_{OH}	2.4			V	I_{OH} = -40 µA max.
	Low Level Output Voltage	V_{OL}			0.4	V	I_{OL} = 3.2 mA
	Rise Time	t_r		200		ns	C_L = 25 pF
	Fall Time	t_f		50		ns	R_L = 11 kΩ pull-up
HEDS-5540 HEDS-5640	Supply Current	I_{CC}	30	57	85	mA	
	High Level Output Voltage	V_{OH}	2.4			V	I_{OH} = -200 µA max.
	Low Level Output Voltage	V_{OL}			0.4	V	I_{OL} = 3.86 mA
	Rise Time	t_r		180		ns	C_L = 25 pF
	Fall Time	t_f		40		ns	R_L = 2.7 kΩ pull-up

*Typical values specified at V_{CC} = 5.0 V and 25°C.

Mechanical Characteristics

Parameter	Symbol	Dimension	Tolerance[1] HEDS-5X00	Tolerance[1] HEDS-5X40	Units
Codewheel Fits These Standard Shaft Diameters		2 3 4 5 6 8	+0.000 -0.015	+0.000 -0.015	mm
		5/32 1/8 3/16 1/4	+0.0000 -0.0007	+0.0000 -0.0007	in
Moment of Inertia	J	0.6 (8.0 x 10^{-4})			g-cm^2 (oz-in-s^2)
Required Shaft Length[2]		14.0 (0.55)	±0.5 (±0.02)	±0.5 (±0.02)	mm (in.)
Bolt Circle[3]	2 screw mounting	19.05 (0.750)	±0.13 (±0.005)	±0.13 (±0.005)	mm (in.)
	3 screw mounting	20.90 (0.823)	±0.13 (±0.005)	±0.13 (±0.005)	mm (in.)
	ext. mtg. ears	46.0 (1.811)	±0.13 (±0.005)	±0.13 (±0.005)	mm (in.)
Mounting Screw Size[4]	2 screw mounting	M 2.5 or (2-56)			mm (in.)
	3 screw mounting	M 1.6 or (0-80)			mm (in.)
	ext. mtg. ears	M 2.5 or (2-56)			mm (in.)
Encoder Base Plate Thickness		0.33 (0.130)			mm (in.)
Hub Set Screw		(2-56)			(in.)

Notes:
1. These are tolerances required of the user.
2. The HEDS-55X5 and 56X5 provide an 8.9 mm (0.35 inch) diameter hole through the housing for longer motor shafts. See Ordering Information.
3. The HEDS-5540 and 5640 must be aligned using the aligning pins as specified in Figure 3, or using the alignment tool as shown in "Encoder Mounting and Assembly". See also "Mounting Considerations."
4. The recommended mounting screw torque for 2 screw and external ear mounting is 1.0 kg-cm (0.88 in-lbs). The recommended mounting screw torque for 3 screw mounting is 0.50 kg-cm (0.43 in-lbs).

Electrical Interface

To insure reliable encoding performance, the HEDS-5540 and 5640 three channel encoders require 2.7 kΩ (±10%) pull-up resistors on output pins 2, 3, and 5 (Channels I, A, and B) as shown in Figure 1. These pull-up resistors should be located as close to the encoder as possible (within 4 feet). Each of the three encoder outputs can drive a single TTL load in this configuration.

The HEDS-5500 and 5600 two channel encoders do not normally require pull-up resistors. However, pull-up resistors on output pins 3 and 5 (Channels A and B) are recommended to improve rise times.

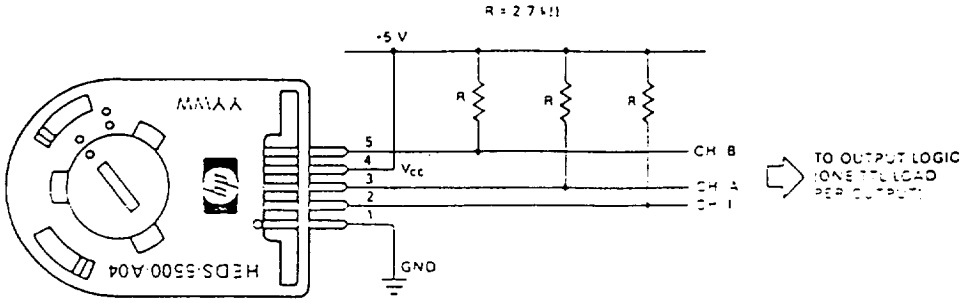


Figure 1. Pull-up Resistors on HEDS-5X40 Encoder Outputs.

Mounting Considerations

The HEDS-5540 and 5640 three channel encoders must be aligned using the aligning pins as specified in Figure 3, or using the HEDS-8910 Alignment Tool as shown in Encoder Mounting and Assembly.

The use of aligning pins or alignment tool is recommended but not required to mount the HEDS-5500 and 5600. If these two channel encoders are attached to a motor with the screw sizes and mounting tolerances specified in the mechanical characteristics section without any additional mounting bosses, the encoder output errors will be within the maximums specified in the encoding characteristics section.

The HEDS-5500 and 5540 can be mounted to a motor using either the two screw or three screw mounting option as shown in Figure 2. The optional aligning pins shown in Figure 3 can be used with either mounting option.

The HEDS-5600 and 5640 have external mounting ears which may be used for mounting to larger motor base plates. Figure 4 shows the necessary mounting holes with optional aligning pins and motor boss.

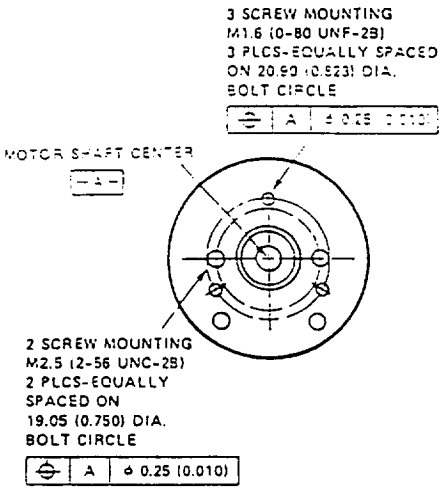


Figure 2. Mounting Holes.

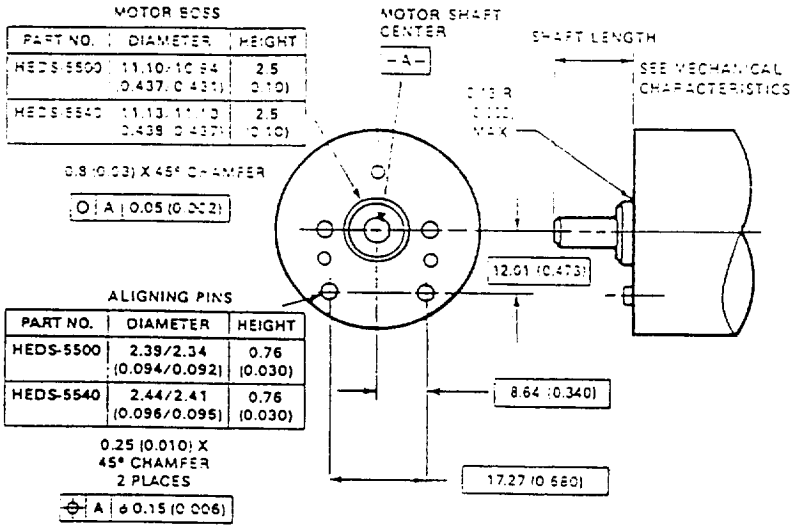


Figure 3. Optional Mounting Aids.

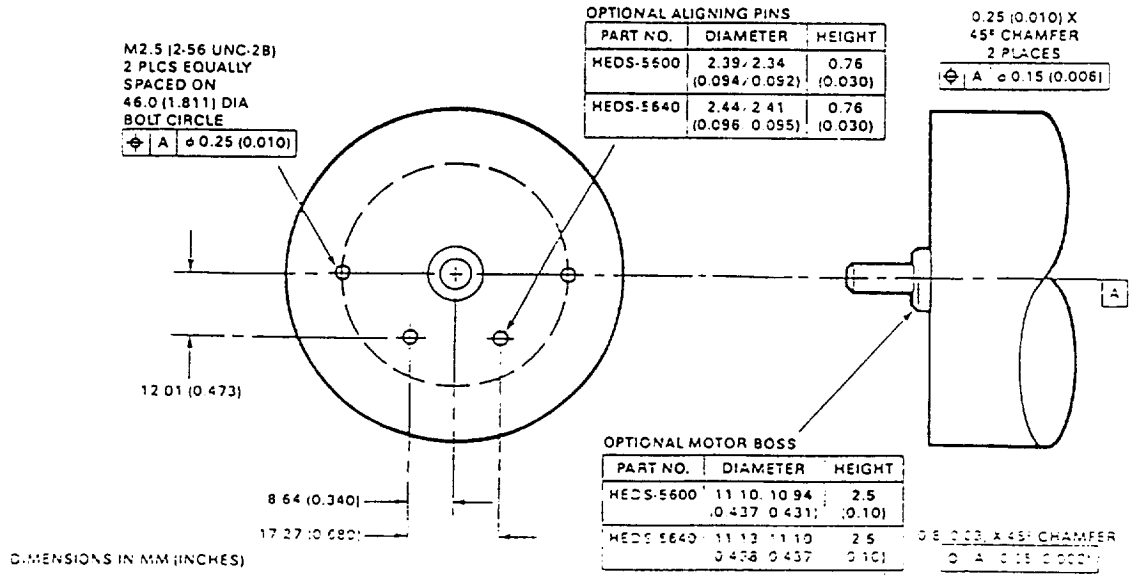
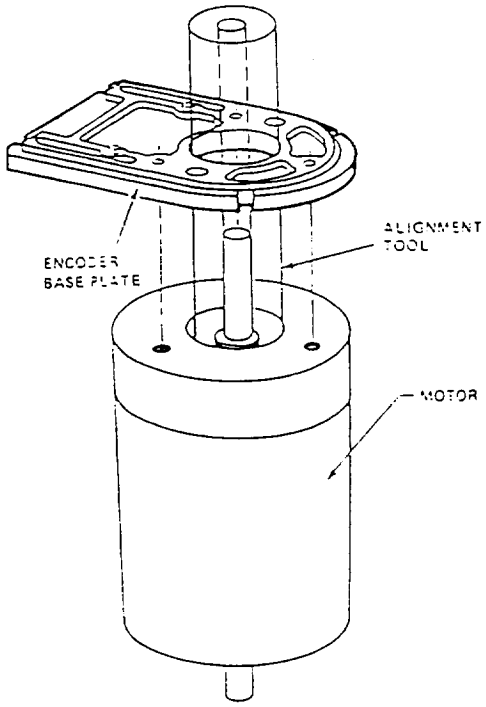


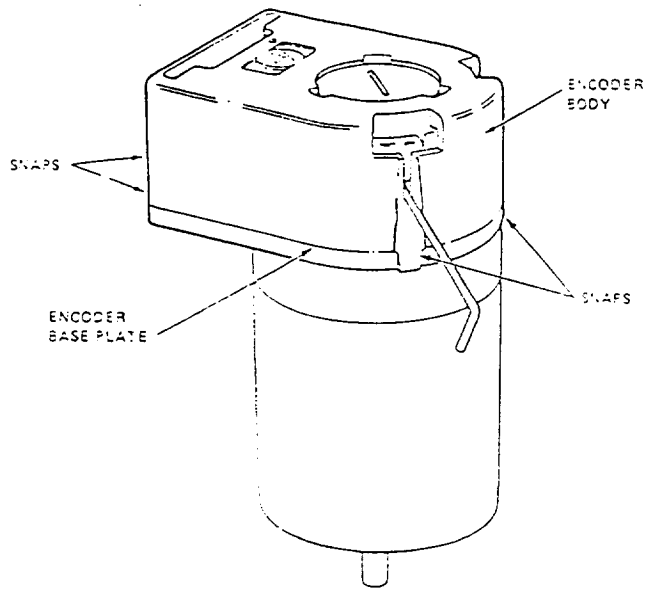
Figure 4. Mounting with External Ears.

Encoder Mounting and Assembly

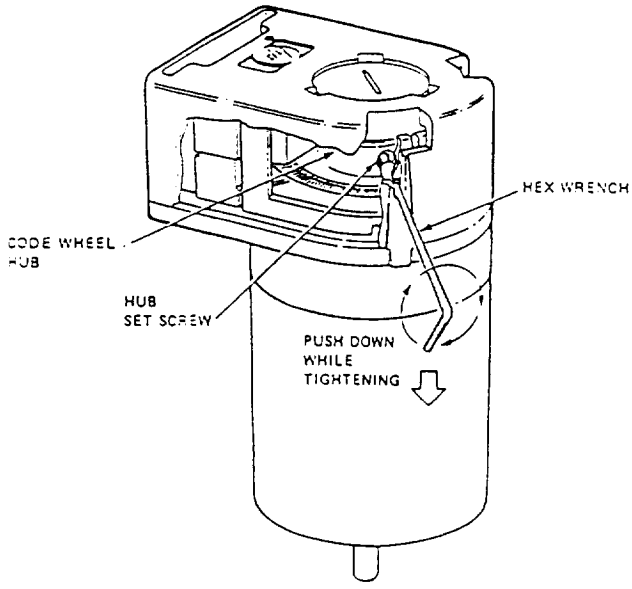


1. For HEDS-5500 and 5600: Mount encoder base plate onto motor. Tighten screws. Go on to step 2.

1a. For HEDS-5540 and 5640: Slip alignment tool onto motor shaft. With alignment tool in place, mount encoder baseplate onto motor as shown above. Tighten screws. Remove alignment tool.



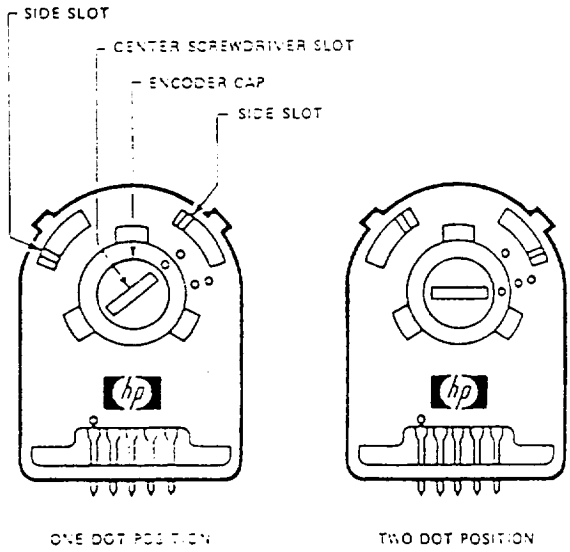
2. Snap encoder body onto base plate locking all 4 snaps.



3a. Push the hex wrench into the body of the encoder to ensure that it is properly seated into the code wheel hub set screws. Then apply a downward force on the end of the hex wrench. This sets the code wheel gap by levering the code wheel hub to its upper position.

3b. While continuing to apply a downward force, rotate the hex wrench in the clockwise direction until the hub set screw is tight against the motor shaft. The hub set screw attaches the code wheel to the motor's shaft.

3c. Remove the hex wrench by pulling it straight out of the encoder body.

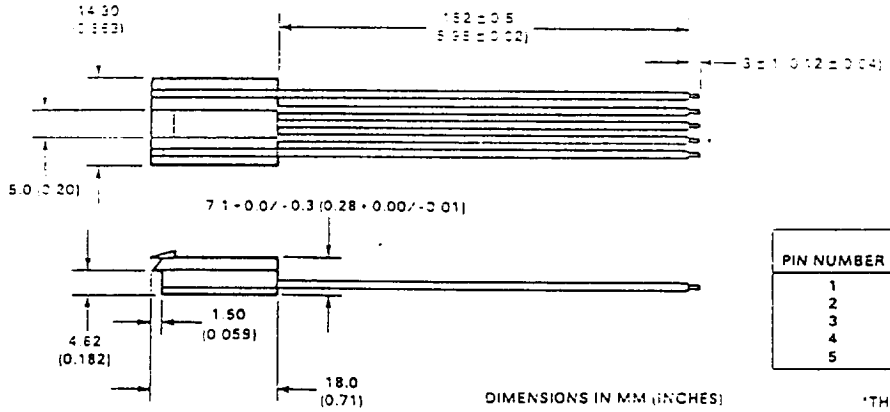


4. Use the center screwdriver slot, or either of the two side slots, to rotate the encoder cap dot clockwise from the one dot position to the two dot position. Do not rotate the encoder cap counterclockwise beyond the one dot position.

The encoder is ready for use!

Connectors

Manufacturer	Part Number
AMP	103686-4 640442-5
Berg	65039-032 with 4825X-000 term.
HP	HEDS-8902 (2 ch.) with 4-wire leads
	HEDS-8903 (3 ch.) with 5-wire leads
Molex	2695 series with 2759 series term.

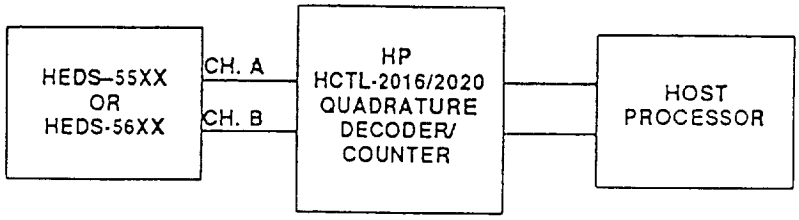


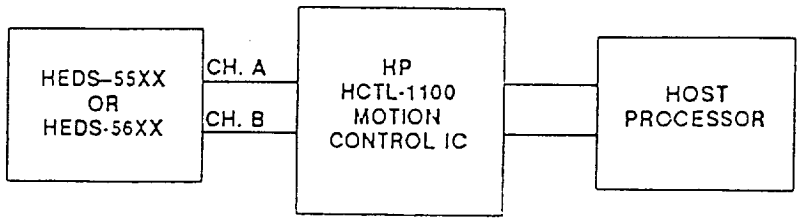
PIN NUMBER	PARAMETER	HEDS-8902 COLORS	HEDS-8903 COLORS
1	GROUND	BLACK	BLACK
2	CH. I	N/A*	BLUE
3	CH. A	WHITE	WHITE
4	V_{cc}	RED	RED
5	CH. B	BROWN	BROWN

*THIS WIRE IS NOT ON HEDS-8902.

Figure 5. HEDS-8902 and 8903 Connectors.

Typical Interfaces





HEWLETT PACKARD

Ordering Information

HEDS-5 □□□ Option □□

Mounting Type	Outputs	Through Hole	Resolution (Cycles/Rev)	Shaft Diameter
5 - Standard 6 - External Mounting Ears	0 - 2 Channel 4 - 3 Channel	0 - None 5 - 6.9 mm (0.35 in.)	(HEDS-550X, 560X 2 Channel) K - 96 CPR C - 100 CPR D - 192 CPR E - 200 CPR F - 256 CPR G - 360 CPR H - 400 CPR A - 500 CPR I - 512 CPR (HEDS-554X, 564X 3 Channel) F - 256 CPR G - 360 CPR A - 500 CPR I - 512 CPR	01 - 2 mm 02 - 3 mm 03 - 1/8 in. 04 - 5/32 in. 05 - 3/16 in. 06 - 1/4 in. 11 - 4 mm 14 - 5 mm 12 - 6 mm 13 - 8 mm

HEDS-8910 0□□ Alignment Tool

HEDS-55XX/56XX Option 000 Encoder Evaluation Kit
HEDS-8910 0XX Alignment Tool included with HEDS-554X/564X three channel encoders
(please specify resolution and shaft diameter when ordering)

For more information call:
United States: 1-800-752-0900*

Or write:
Hewlett-Packard Components
Customer Information Center
Building 49 AV
19310 Pruneridge Avenue
Cupertino, California 95014

Canada: (416) 678-9430*
Europe: (49) 7031/14-0*
Far East: (65) 271-9444*
Japan: (81) 03-331-6111*
Elsewhere in the world: (415) 857-5027*

*Or call your local HP sales office listed in your telephone directory. Ask for a Components representative.

Obsoletes 5954-9437

Printed in U.S.A. 5952-2472 (6/90)

APPENDIX II.D

HP2016 Integrated Circuit Data Sheets

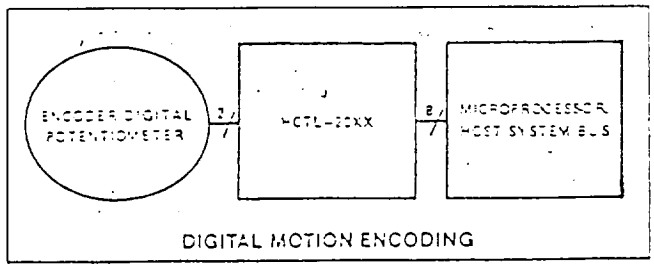
HEWLETT PACKARD

QUADRATURE DECODER/ COUNTER INTERFACE IC

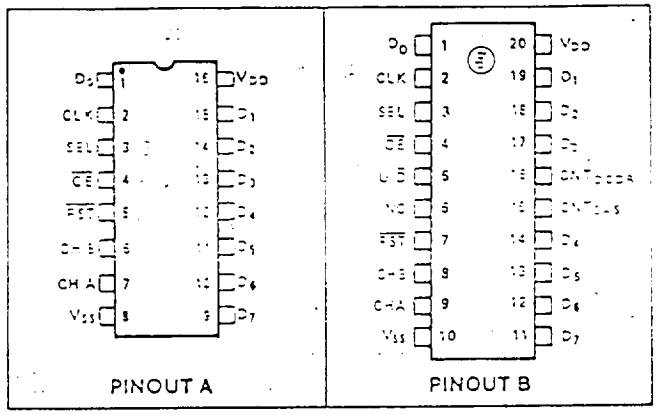
HCTL-2000
HCTL-2016
HCTL-2020

Features

- INTERFACES ENCODER TO MICROPROCESSOR
- 14 MHz CLOCK OPERATION
- FULL 4X DECODE
- HIGH NOISE IMMUNITY:
  SCHMITT TRIGGER INPUTS
  DIGITAL NOISE FILTER
- 12 OR 16-BIT BINARY UP/DOWN COUNTER
- LATCHED OUTPUTS
- 8-BIT TRISTATE INTERFACE
- 8, 12, OR 16-BIT OPERATING MODES
- QUADRATURE DECODER OUTPUT SIGNALS, UP/DOWN AND COUNT
- CASCADE OUTPUT SIGNALS, UP/DOWN AND COUNT
- SUBSTANTIALLY REDUCED SYSTEM SOFTWARE



DIGITAL MOTION ENCODING



PINOUT A

PINOUT B

Description

The HCTL-2000, 2016, 2020 are CMOS ICs that perform the quadrature decoder, counter, and bus interface function. The HCTL-20XX family is designed to improve system performance in digital closed loop motion control systems and digital data input systems. It does this by shifting time intensive quadrature decoder functions to a cost effective hardware solution. The entire HCTL-20XX family consists of a 4x quadrature decoder, a binary up/down state counter, and an 8-bit bus interface. The use of Schmitt-triggered CMOS inputs and input noise filters allows reliable operation in noisy environments. The HCTL-2000 contains a 12-bit counter. The HCTL-2016 and 2020 contain a 16-bit counter. The HCTL-2020 also contains quadrature decoder output signals and cascade signals for use with many standard counter ICs. The HCTL-20XX family provides LSTTL compatible tri-state output buffers. Operation is specified for a temperature range from -40 to +85°C at clock frequencies up to 14 MHz.

Applications

- INTERFACE QUADRATURE INCREMENTAL ENCODERS TO MICROPROCESSORS
- INTERFACE DIGITAL POTENTIOMETERS TO DIGITAL DATA INPUT BUSES

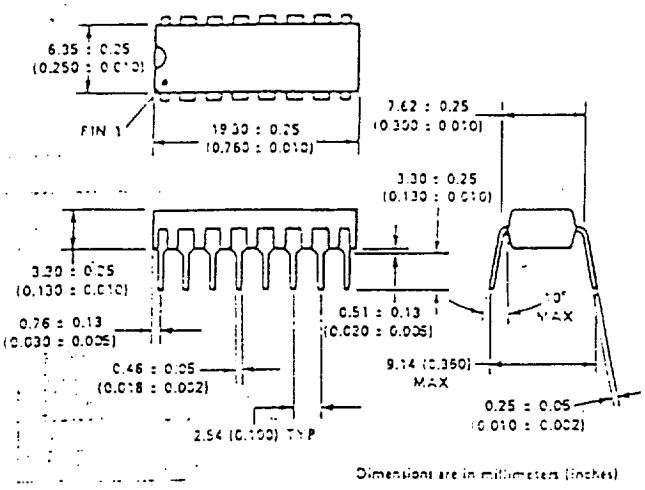
Table of Contents

ESD WARNING: Standard CMOS handling precautions should be observed with the HCTL-20XX family ICs.

Devices

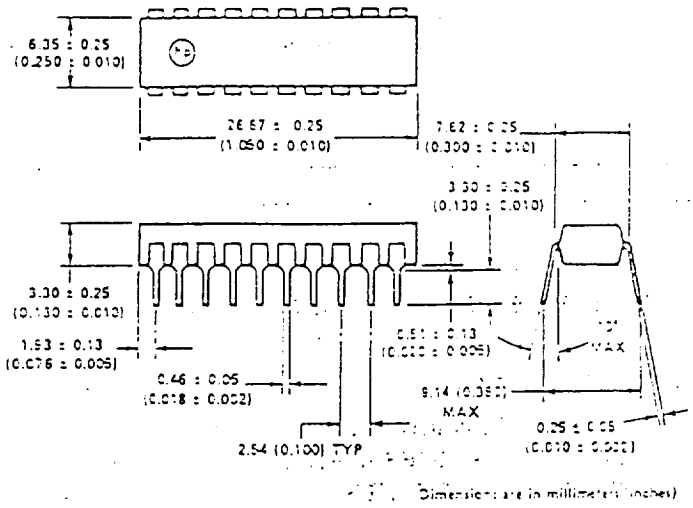
Part Number	Description	Package Drawing
HCTL-2000	12-bit counter. 14 MHz clock operation.	A
HCTL-2016	All features of the HCTL-2000. 16-bit counter.	A
HCTL-2020	All features of the HCTL-2016. Quadrature decoder output signals. Cascade output signals.	B

Package Dimensions



PACKAGE A LEAD FINISH: SOLDER DIPPED

PACKAGE A



PACKAGE B LEAD FINISH: SOLDER DIPPED

PACKAGE B

Operating Characteristics

Table 1. Absolute Maximum Ratings (All voltages below are referenced to V_{SS})

Parameter	Symbol	Limits	Units
DC Supply Voltage	V_{DD}	-0.3 to +5.5	V
Input Voltage	V_{IN}	-0.3 to V_{DD} +0.3	V
Storage Temperature	T_S	-40 to +125	°C
Operating Temperature	T_A[1]	-40 to +85	°C

Table 2. Recommended Operating Conditions

Parameter	Symbol	Limits	Units
DC Supply Voltage	V_{DD}	+4.5 to +5.5	V
Ambient Temperature	T_A[1]	-40 to -85	°C

Table 3. DC Characteristics V_{DD} = 5 V ± 5%; T_A = -40 to 85°C

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V_{IL}[2]	Low-Level Input Voltage				1.5	V
V_{IH}[2]	High-Level Input Voltage		3.5			V
V_{T+}	Schmitt-Trigger Positive-Going Threshold			3.5	4.0	V
V_{T-}	Schmitt-Trigger Negative-Going Threshold		1.0	1.5		V
V_H	Schmitt-Trigger Hysteresis		1.0	2.0		V
I_{IN}	Input Current	V_{IN} = V_{SS} or V_{DD}	-10	1	+10	µA
V_{OH}[2]	High-Level Output Voltage	I_{OH} -1.6 mA	2.4	4.5		V
V_{OL}[2]	Low-Level Output Voltage	I_{OL} = +4.8 mA		0.2	0.4	V
I_{OZ}	High-Z Output Leakage Current	V_O = V_{SS} or V_{DD}	-10	1	+10	µA
I_{DD}	Quiescent Supply Current	V_{IN} = V_{SS} or V_{DD}, V_O = HiZ		1	5	µA
C_{IN}	Input Capacitance	Any Input[3]		5		pF
C_{OUT}	Output Capacitance	Any Output[3]		6		pF

Notes:
1. Free Air
2. In general, for any V_{DD} between the allowable limits (+4.5 V to +5.5 V), V_{IL} = 0.3 V_{DD} and V_{IH} = 0.7 V_{DD}; typical values are V_{OH} = V_{DD} - 0.5 V @ I_{OH} = -40 µA and V_{OL} = V_{SS} + 0.2 V @ I_{OL} = 1.6 mA.
3. Including package capacitance.

Functional Pin Description

Table 4. Functional Pin Descriptions

Symbol	Pin 2000/2016	Pin 2020	Description
V_{DD}	16	20	Power Supply
V_{SS}	8	10	Ground
CLK	2	2	CLK is a Schmitt-trigger input for the external clock signal.
CHA CHB	7 6	9 8	CHA and CHB are Schmitt-trigger inputs which accept the outputs from a quadrature encoded source, such as incremental optical shaft encoder. Two channels, A and B, nominally 90 degrees out of phase, are required.
$\overline{RST}$	5	7	This active low Schmitt-trigger input clears the internal position counter and the position latch. It also resets the inhibit logic. $\overline{RST}$ is asynchronous with respect to any other input signals.
$\overline{OE}$	4	4	This CMOS active low input enables the tri-state output buffers. The $\overline{OE}$ and SEL inputs are sampled by the internal inhibit logic on the falling edge of the clock to control the loading of the internal position data latch.
SEL	3	3	This CMOS input directly controls which data byte from the position latch is enabled into the 8-bit tri-state output buffer. As in $\overline{OE}$ above, SEL also controls the internal inhibit logic. SEL = 0 → High byte SEL = 1 → Low byte
CNT_{DCDR}		16	A pulse is presented on this LSTTL-compatible output when the quadrature decoder has detected a state transition.
U/$\overline{D}$		5	This LSTTL-compatible output allows the user to determine whether the IC is counting up or down and is intended to be used with the CNT_{DCDR} and CNT_{CAS} outputs. The proper signal U (high level) or $\overline{D}$ (low level) will be present before the rising edge of the CNT_{DCDR} and CNT_{CAS} outputs.
CNT_{CAS}		15	A pulse is presented on this LSTTL-compatible output when the HCTL-2020 internal counter overflows or underflows. The rising edge on this waveform may be used to trigger an external counter.
D0	1	1	These LSTTL-compatible tri-state outputs form an 8-bit output port through which the contents of the 12/16-bit position latch may be read in 2 sequential bytes. The high byte, containing bits 8-15, is read first (on the HCTL-2000, the most significant 4 bits of this byte are set to 0 internally). The lower byte, bits 0-7, is read second.
D1	15	19	
D2	14	18	
D3	13	17	
D4	12	14	
D5	11	13	
D6	10	12	
D7	9	11	
NC		6	Not connected — this pin should be left floating.

Switching Characteristics

Table 5. Switching Characteristics Min Max specifications at V_{DD} = 5.0 ± 5%, T_A = -40 to + 85°C

	Symbol	Description	Min.	Max.	Units
1	t_{CLK}	Clock period	70		ns
2	t_{CHH}	Pulse width, clock high	28		ns
3	t_{CD}[1]	Delay time, rising edge of clock to valid, updated count information on D0-7		65	ns
4	t_{ODE}	Delay time, $\overline{OE}$ fall to valid data		65	ns
5	t_{ODZ}	Delay time, $\overline{OE}$ rise to Hi-Z state on D0-7		40	ns
6	t_{SDV}	Delay time, SEL valid to stable, selected data byte (delay to High Byte = delay to Low Byte)		65	ns
7	t_{CLH}	Pulse width, clock low	28		ns
8	t_{SS}[2]	Setup time, SEL before clock fall	20		ns
9	t_{OS}[2]	Setup time, $\overline{OE}$ before clock fall	20		ns
10	t_{SH}[2]	Hold time, SEL after clock fall	0		ns
11	t_{OH}[2]	Hold time, $\overline{OE}$ after clock fall	0		ns
12	t_{RST}	Pulse width, $\overline{RST}$ low	28		ns
13	t_{DCD}	Hold time, last position count stable on D0-7 after clock rise	10		ns
14	t_{DSD}	Hold time, last data byte stable after next SEL state change	5		ns
15	t_{DOD}	Hold time, data byte stable after $\overline{OE}$ rise	5		ns
16	t_{UDD}	Delay time, U/$\overline{D}$ valid after clock rise		45	ns
17	t_{CHD}	Delay time, CNT_{DOWN} or CNT_{CAS} high after clock rise		45	ns
18	t_{CLD}	Delay time, CNT_{DOWN} or CNT_{CAS} low after clock fall		45	ns
19	t_{UDH}	Hold time, U/$\overline{D}$ stable after clock rise	10		ns
20	t_{UDCS}	Setup time, U/$\overline{D}$ valid before CNT_{DOWN} or CNT_{CAS} rise	t_{CLK}-45		ns
21	t_{UDCH}	Hold time, U/$\overline{D}$ stable after CNT_{DOWN} or CNT_{CAS} rise	t_{CLK}-45		ns

Notes:

1. t_{CD} specification and waveform assume latch not inhibited.
2. t_{SS}, t_{OS}, t_{SH}, t_{OH} only pertain to proper operation of the inhibit logic. In other cases, such as 8 bit read operations, these setup and hold times do not need to be observed.

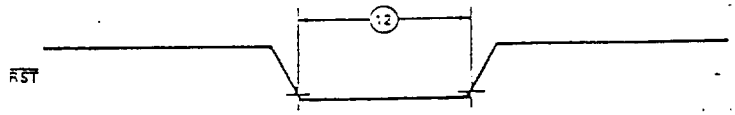


Figure 1. Reset Waveform

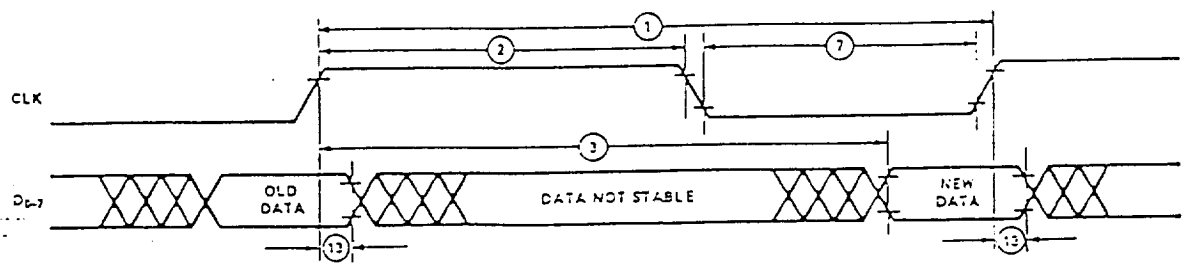


Figure 2. Waveform for Positive Clock Related Delays

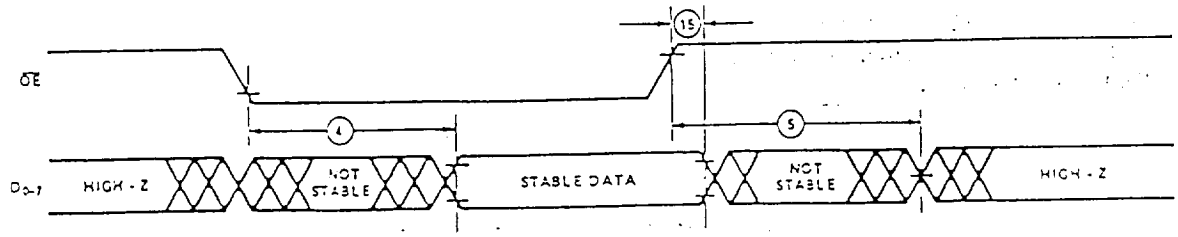


Figure 3. Tri-State Output Timing

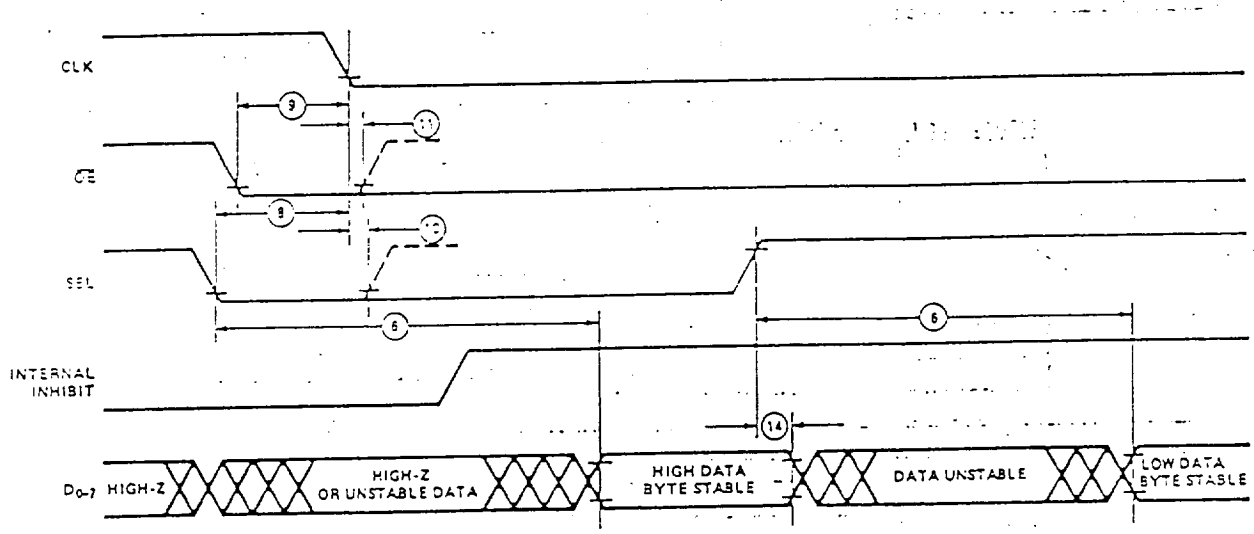


Figure 4. Bus Control Timing

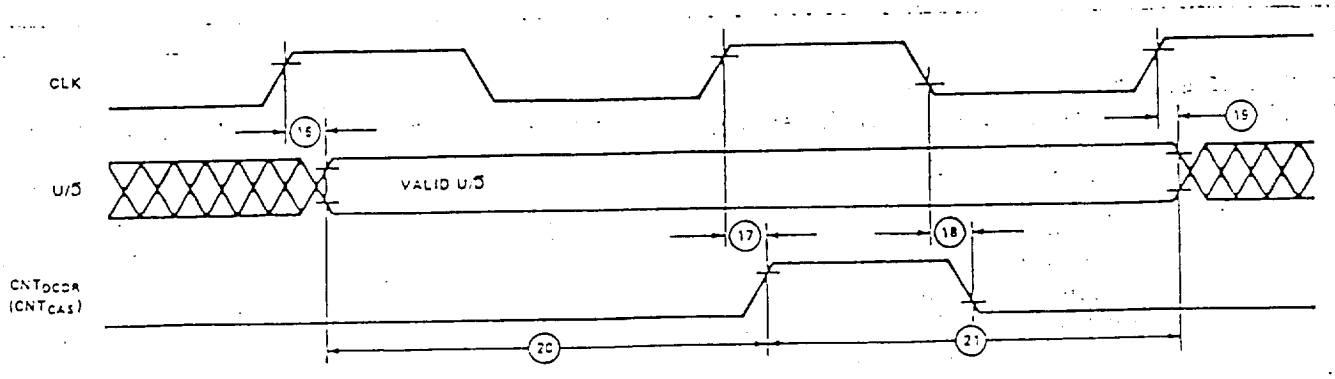


Figure 5. Decoder Cascade Output Timing (HCTL-2020 only)

Operation

A block diagram of the HCTL-20XX family is shown in Figure 6. The operation of each major function is described in the following sections.

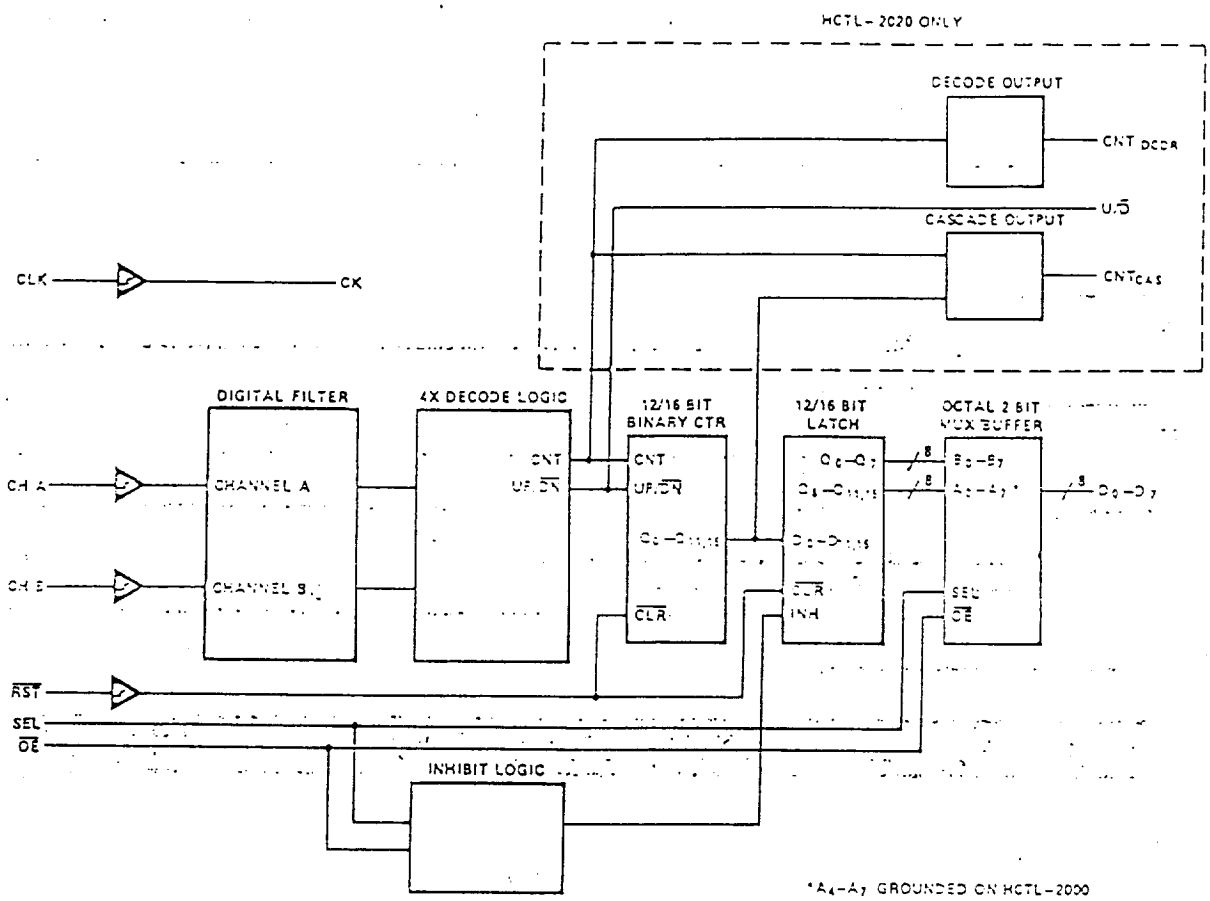


Figure 6. Simplified Logic Diagram

DIGITAL NOISE FILTER

The digital noise filter section is responsible for rejecting noise on the incoming quadrature signals. The input section uses two techniques to implement improved noise rejection. Schmitt-trigger inputs and a three-clock-cycle delay filter combine to reject low level noise and large, short duration noise spikes that typically occur in motor system applications. Both common mode and differential mode noise are rejected. The user benefits from these techniques by improved integrity of the data in the counter. False counts triggered by noise are avoided.

Figure 7 shows the simplified schematic of the input section. The signals are first passed through a Schmitt trigger buffer to address the problem of input signals with slow rise times and low level noise (approximately < 1 V). The cleaned up signals are then passed to a four-bit delay filter. The signals on each channel are sampled on rising clock edges. A time history of the signals is stored in the four-bit shift register. Any change on the input is tested for a stable level being present for three consecutive rising clock edges. Therefore, the filtered output waveforms can change only after an input level has the same value for three consecutive rising clock edges. Refer to Figure 8 which shows the timing diagram. The result of this circuitry is that short noise spikes between rising clock edges are ignored and pulses shorter than two clock periods are rejected.

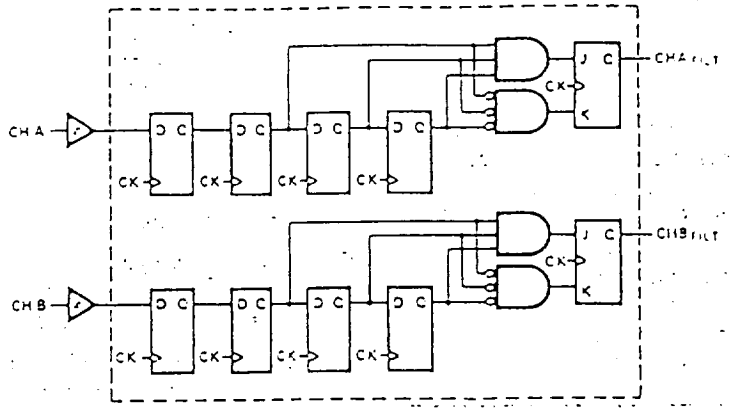


Figure 7. Simplified Digital Noise Filter Logic

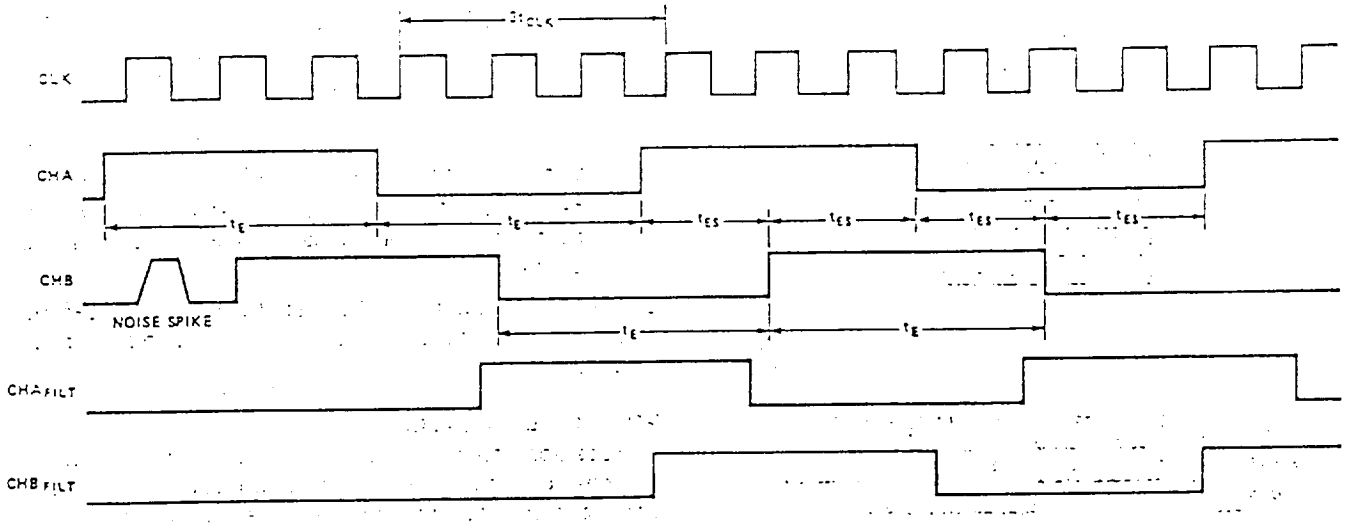


Figure 8. Signal Propagation Through Digital Noise Filter

QUADRATURE DECODER

The quadrature decoder decodes the incoming filtered signals into count information. This circuitry multiplies the resolution of the input signals by a factor of four (4X decoding). When using an encoder for motion sensing, the user benefits from the increased resolution by being able to provide better system control.

The quadrature decoder samples the outputs of the CHA and CHB filters. Based on the past binary state of the two signals and the present state, it outputs a count signal and a direction signal to the internal position counter. In the case of the HCTL-2020, the signals also go to external pins 5 and 16 respectively.

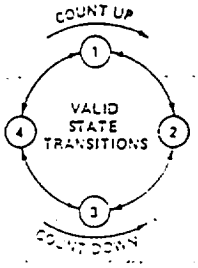
Figure 9 shows the quadrature states and the valid state transitions. Channel A leading channel B results in counting up. Channel B leading channel A results in counting down. Illegal state transitions, caused by faulty encoders or noise severe enough to pass through the filter, will produce an erroneous count.

DESIGN CONSIDERATIONS

The designer should be aware that the operation of the digital filter places a timing constraint on the relationship between incoming quadrature signals and the external clock. Figure 8 shows the timing waveform with an incremental encoder input. Since an input has to be stable for three rising clock edges, the encoder pulse width (t_E - low or high), has to be greater than three clock periods ($3t_{CLK}$). This guarantees that the asynchronous input will be stable during three consecutive rising clock edges. A realistic design also has to take into account finite rise times of the waveforms, asymmetry of the waveforms, and noise. In the presence of large amounts of noise, t_E should be much

greater than $3t_{CLK}$ to allow for the interruption of the consecutive level sampling by the three-bit delay filter. It should be noted that a change on the inputs that is qualified by the filter will internally propagate in a maximum of seven clock periods.

The quadrature decoder circuitry imposes a second timing constraint between the external clock and the input signals. There must be at least one clock period between consecutive quadrature states. As shown in Figure 9, a quadrature state is defined by consecutive edges on both channels. Therefore, t_{ES} (encoder state period) > t_{CLK}. The designer must account for deviations from the nominal 90 degree phasing of input signals to guarantee that $t_{ES} > t_{CLK}$.



CHA	CHB	STATE
1	0	1
1	1	2
0	1	3
0	0	4

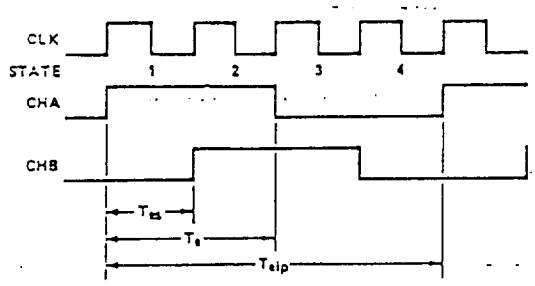


Figure 9. 4x Quadrature Decoding

POSITION COUNTER

This section consists of a 12-bit (HCTL-2000) or 16-bit (HCTL-2016/2020) binary up down counter which counts on rising clock edges as explained in the Quadrature Decoder Section. All 12 or 16 bits of data are passed to the position data latch. The system can use this count data in several ways:

A. System total range is ≤ 12 or 16 bits, so the count represents "absolute" position.

B. The system is cyclic with ≤ 12 or 16 bits of count per cycle. $\overline{RST}$ is used to reset the counter every cycle and the system uses the data to interpolate within the cycle.

C. System count is > 8, 12 or 16 bits, so the count data is used as a relative or incremental position input for a system software computation of absolute position. In this case counter rollover occurs. In order to prevent loss of position information, the processor must read the outputs of the IC before the count increments one-half of the maximum count capability (i.e. 127, 2047, or 32,767 quadrature counts). Two's-complement arithmetic is normally used to compute position from these periodic position updates. Three modes can be used:

1. The IC can be put in 8-bit mode by tying the SEL line high, thus simplifying IC interface. The outputs must then be read at least once every 127 quadrature counts.
2. The HCTL-2000 can be used in 12-bit mode and sampled at least once every 2047 quadrature counts.
3. The HCTL-2016 or 2020 can be used in 16-bit mode and sampled at least once every 32,767 quadrature counts.

D. The system count is > 16 bits so the HCTL-2020 can be cascaded with other standard counter IC's to give absolute position.

POSITION DATA LATCH

The position data latch is a 12/16-bit latch which captures the position counter output data on each rising clock edge, except when its inputs are disabled by the inhibit logic section during two-byte read operations. The output data is passed to the bus interface section. When active, a signal from the inhibit logic section prevents new data from being captured by the latch, keeping the data stable while successive reads are made through the bus section. The latch is automatically reenabled at the end of these reads. The latch is cleared to 0 asynchronously by the $\overline{RST}$ signal.

INHIBIT LOGIC

The Inhibit Logic Section samples the $\overline{CE}$ and SEL signals on the falling edge of the clock and, in response to certain conditions (see Figure 10 below), inhibits the position data latch. The $\overline{RST}$ signal asynchronously clears the inhibit logic, enabling the latch. A simplified logic diagram of the inhibit circuitry is illustrated in Figure 11.

STEP	SEL	$\overline{OE}$	CLK	INHIBIT SIGNAL	ACTION
1	L	L	⌊	1	SET INHIBIT, READ HIGH BYTE
2	H	L	⌊	1	READ LOW BYTE; STARTS RESET
3	X	H	⌊	0	COMPLETES INHIBIT LOGIC RESET

Figure 10. Two Byte Read Sequence

BUS INTERFACE

The bus interface section consists of a 16 to 8 line multiplexer and an 8-bit, three-state output buffer. The multiplexer allows independent access to the low and high bytes of the position data latch. The SEL and $\overline{OE}$ signals determine which byte is output and whether or not the output bus is in the high-Z state. In the case of the HCTL-2000 the data latch is only 12 bits wide and the upper four bits of the high byte are internally set to zero.

QUADRATURE DECODER OUTPUT (HCTL-2020 ONLY)

The quadrature decoder output section consists of count and up/down outputs derived from the 4X decode logic of the HCTL-2020. When the decoder has detected a count, a pulse, one-half clock cycle long, will be output on the CNT_{DCDR} pin. This output will occur during the clock cycle in which the internal counter is updated. The U/$\overline{D}$ pin will be set to the proper voltage level one clock cycle before the rising edge of the CNT_{DCDR} pulse, and held one clock cycle after the rising edge of the CNT_{DCDR} pulse. These outputs are not affected by the inhibit logic. See Figures 5 and 12 for detailed timing.

CASCADE OUTPUT (HCTL-2020 ONLY)

The cascade output also consists of count and up/down outputs. When the HCTL-2020 internal counter overflows or underflows, a pulse, one-half clock cycle long, will be output on the CNT_{CAS} pin. This output will occur during the clock cycle in which the internal counter is updated. The U/$\overline{D}$ pin will be set to the proper voltage level one clock cycle before the rising edge of the CNT_{CAS} pulse, and held one clock cycle after the rising edge of the CNT_{CAS} pulse. These outputs are not affected by the inhibit logic. See Figures 5 and 12 for detailed timing.

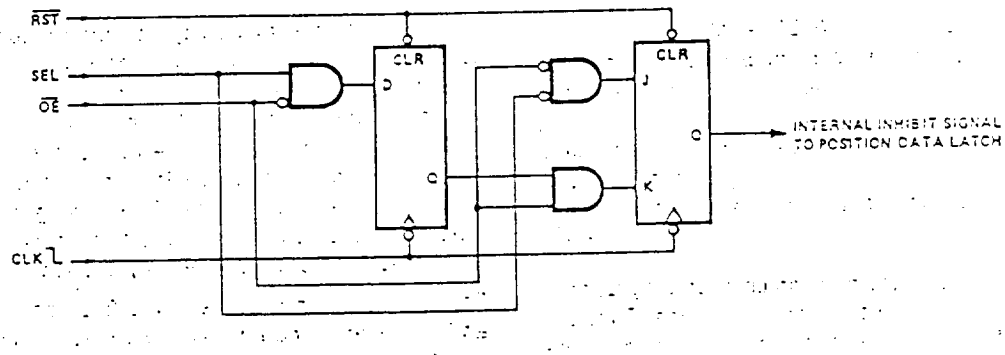
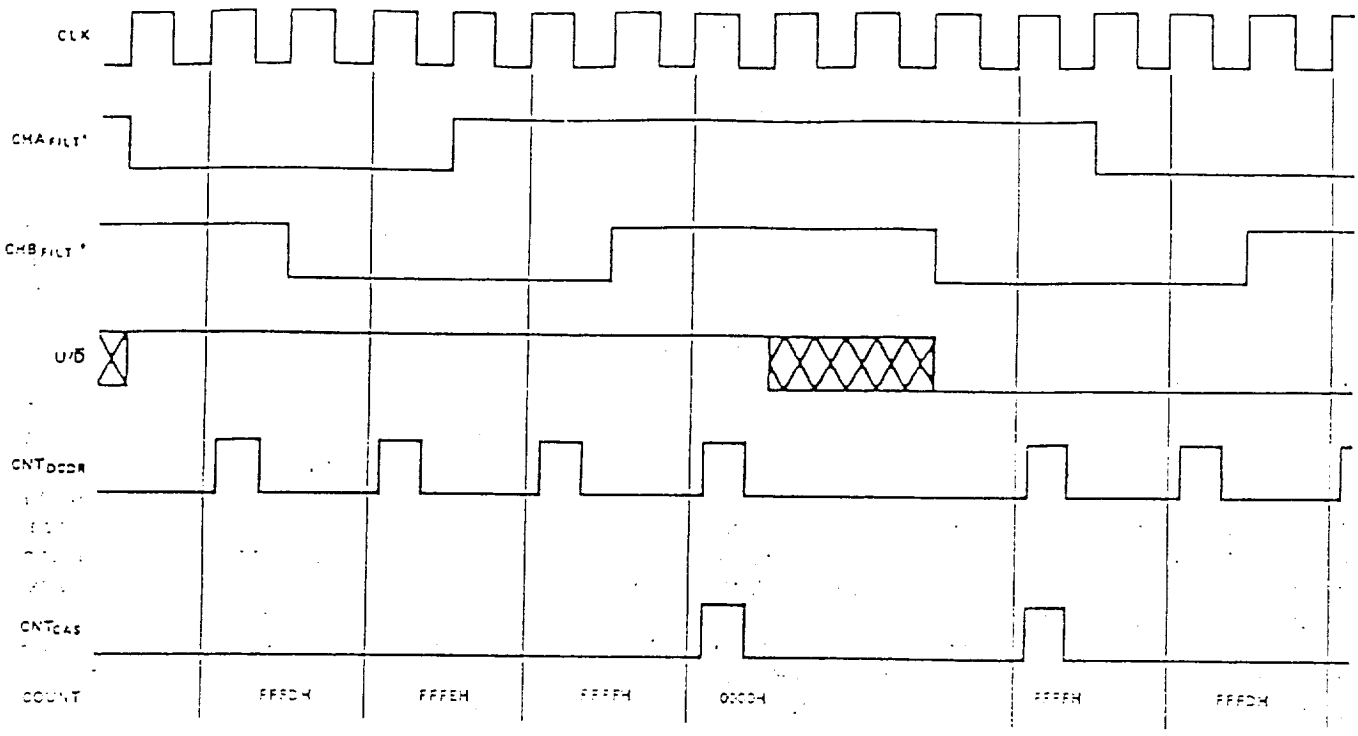


Figure 11. Simplified Inhibit Logic



*CHA_{FILT} and CHB_{FILT} are the outputs of the digital noise filter (see figures 7 and 8).

Figure 12. Decode and Cascade Output Diagram

CASCADE CONSIDERATIONS (HCTL-2020 ONLY)

The HCTL-2020's cascading system allows for position reads of more than two bytes. These reads can be accomplished by latching all of the bytes, and then reading the bytes sequentially over the 8-bit bus. Care must be taken to latch all of the bytes such that they represent the count as it actually is, despite propagation delays through the counters.

A good understanding of the mechanics of count propagation is important in designing a proper interface. Consider the sequence of events for a read cycle that starts as the HCTL-2020's internal counter rolls over. On the rising clock edge, count data is updated in the internal counter, rolling it over. A count-cascade pulse (CNT_{CAS}) will be generated with some delay after the rising clock edge (t_{CHO}). There will be additional propagation delays through the external counters and registers. Meanwhile, with $\overline{OE}$ and SEL low to start the read, this new count on the HCTL-2020 will be latched in on the falling clock edge of this cycle. If the external registers are latched too soon, before the CNT_{CAS} pulse has toggled the external counters and registers, a major count error will occur.

Valid data can be ensured by latching the external counter data on the first rising clock edge following the falling edge on which the internal count on the HCTL-2020 is latched (provided that all the delays are less than one clock cycle). This will ensure that a cascade pulse that occurs during the clock cycle when the read begins has adequate time to propagate. This also guarantees that a cascade pulse occurring on the clock cycle after the read is initiated will not be erroneously latched.

For example, suppose the HCTL-2020 count is at FFFFH and an external counter is at F0H, with the count going up. A count occurring in the HCTL-2020 will cause the counter to roll over and a cascade pulse will be generated. A read starting on this clock cycle will show 0000H from the HCTL-2020. The external counter should read F1H, but if the host latches the count before the cascade signal propagates through, the external counter will still read F0H.

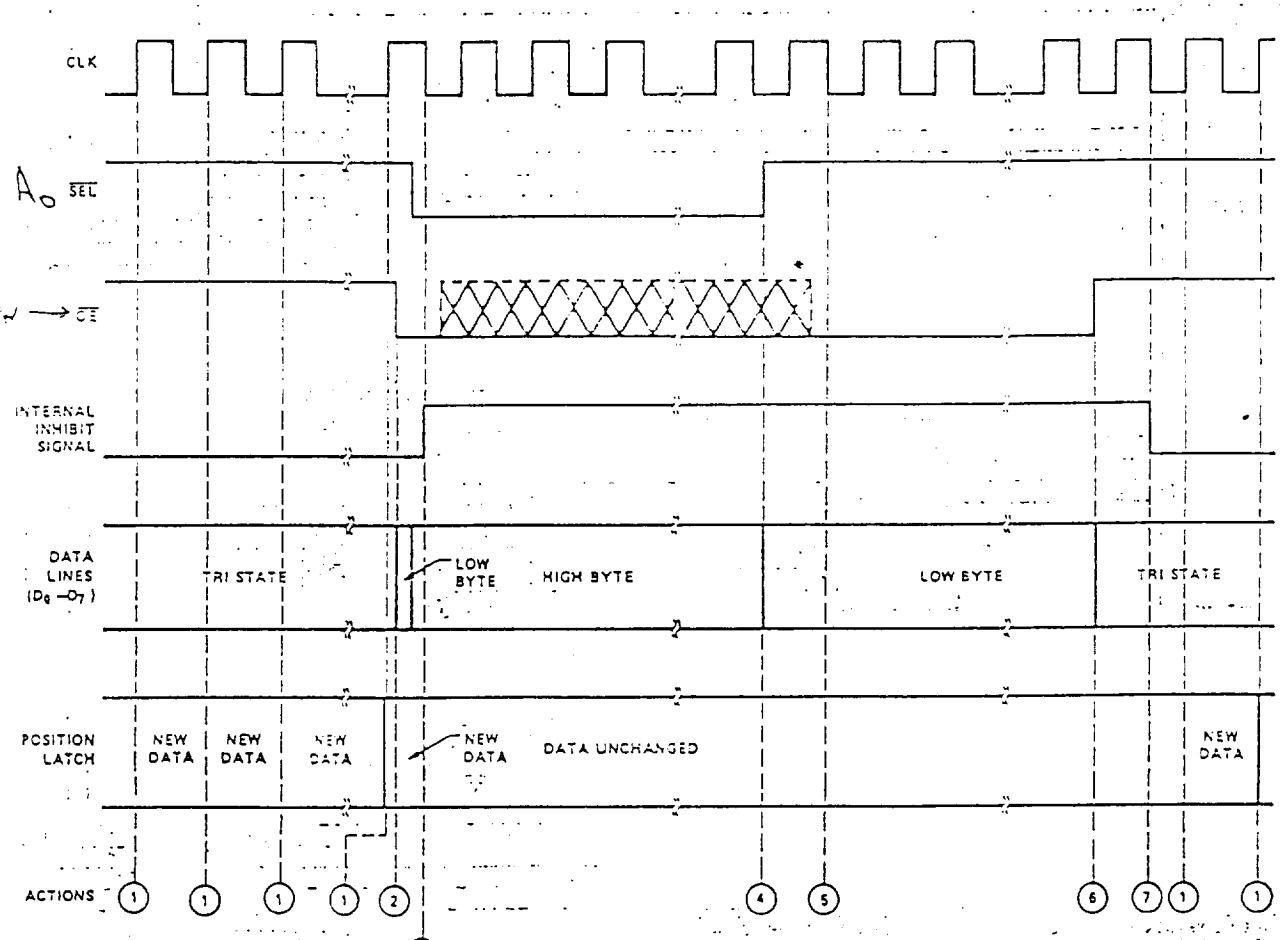
General Interfacing

The 12-bit (HCTL-2000) or 16-bit (HCTL-2016/2020) latch and inhibit logic allows access to 12 or 16 bits of count with an 8-bit bus. When only 8-bits of count are required, a simple 8-bit (1-byte) mode is available by holding SEL high continuously. This disables the inhibit logic. $\overline{OE}$ provides control of the tri-state bus, and read timing is shown in Figures 2 and 3.

For proper operation of the inhibit logic during a two-byte read, $\overline{OE}$ and SEL must be synchronous with CLK due to the falling edge sampling of $\overline{OE}$ and SEL.

The internal inhibit logic on the HCTL-20XX family inhibits the transfer of data from the counter to the position data latch during the time that the latch outputs are being read. The inhibit logic allows the microprocessor to first read the high order 4 or 8 bits from the latch and then read the low order 8 bits from the latch. Meanwhile, the counter can continue to keep track of the quadrature states from the CHA and CHB input signals.

Figure 11 shows the simplified inhibit logic circuit. The operation of the circuitry is illustrated in the read timing shown in Figure 13.



*$\overline{OE}$ can consist of two short low pulses, as well as one long pulse, and still satisfy the inhibit logic sequence. During the time that $\overline{OE}$ is high, the data lines are tri-stated.

Figure 13. Typical Interface Timing

ACTIONS

1. On the rising edge of the clock, counter data is transferred to the position data latch, provided the inhibit signal is low.

2. When $\overline{OE}$ goes low, the outputs of the multiplexer are enabled onto the data lines. If SEL is low, then the high order data bytes are enabled onto the data lines. If SEL is high, then the low order data bytes are enabled onto the data lines.

3. When the IC detects a low on $\overline{CE}$ and SEL during a falling clock edge, the internal inhibit signal is activated. This blocks new data from being transferred from the counter to the position data latch.

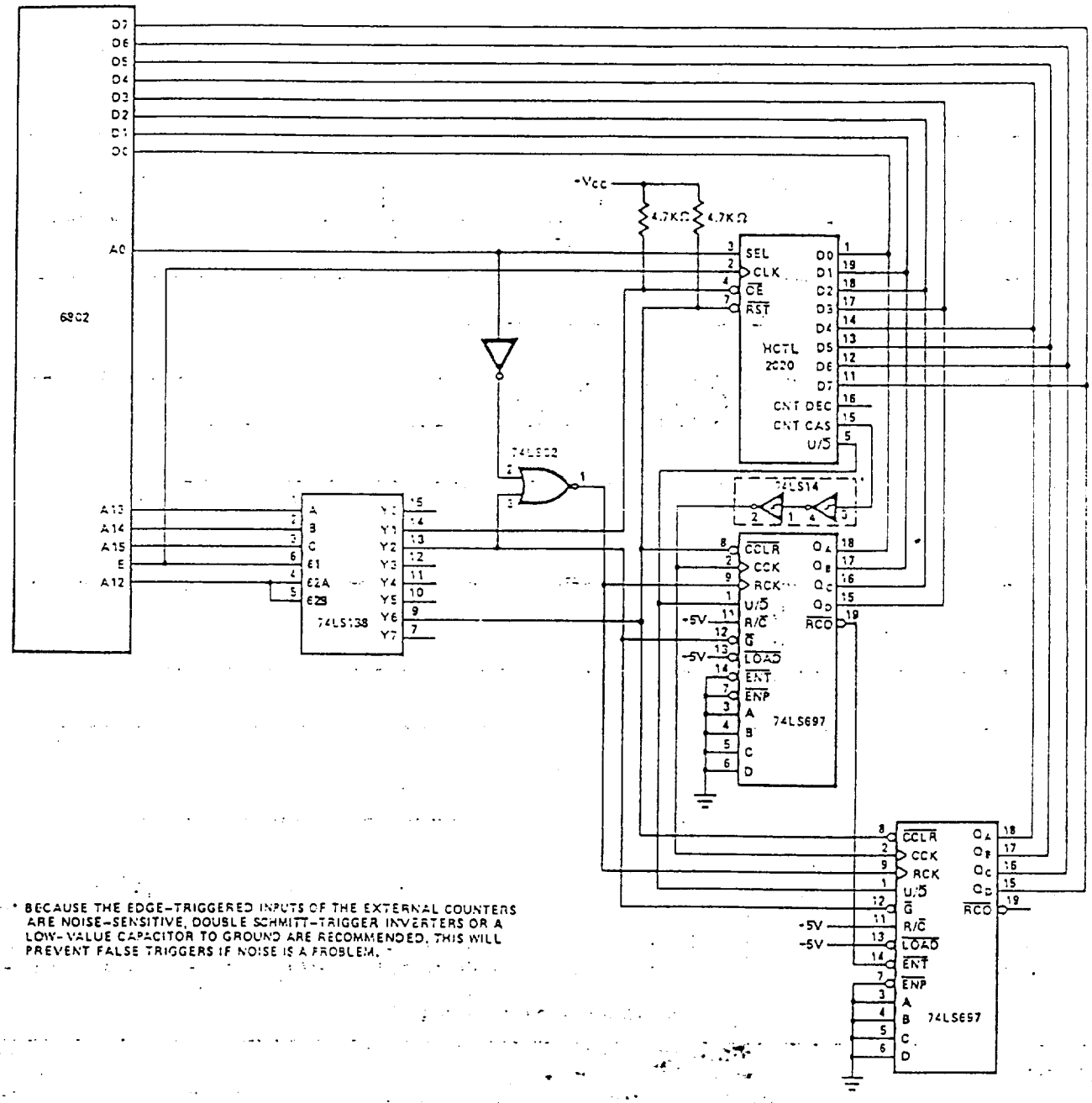
4. When SEL goes high, the data outputs change from the high byte to the low byte.

5. The first of two reset conditions for the inhibit logic is met when the IC detects a logic high on SEL and a logic low on $\overline{OE}$ during a falling clock edge.

6. When $\overline{OE}$ goes high, the data lines change to a high impedance state.

7. The IC detects a logic high on $\overline{OE}$ during a falling clock edge. This satisfies the second reset condition for the inhibit logic.

Interfacing the HCTL-2020 to a Motorola 6802/8 and Cascading the Counter for 24 Bits



* BECAUSE THE EDGE-TRIGGERED INPUTS OF THE EXTERNAL COUNTERS ARE NOISE-SENSITIVE, DOUBLE SCHMITT-TRIGGER INVERTERS OR A LOW-VALUE CAPACITOR TO GROUND ARE RECOMMENDED. THIS WILL PREVENT FALSE TRIGGERS IF NOISE IS A PROBLEM.

Figure 14. A Circuit to Interface to the 6802/8

In this circuit an interface to a Motorola 6802/8 and a cascading scheme for a 24-bit counter are shown. This circuit provides a minimum part count by: 1) using two 74LS697 Up/Down counters with output registers and tri-state outputs and 2) using a Motorola 6802/8 LDX instruction which stores 16 bits of data into the index registers in two consecutive clock cycles.

The HCTL-2020 $\overline{OE}$ and the 74LS697 $\overline{G}$ lines are decoded from Address lines A15-A13. This results in counter data being enabled onto the bus whenever an external memory access is made to locations 4XXX or 2XXX. Address line A12 and processor clock E enable the 74LS138. The processor clock E is also used to clock the HCTL-2020. Address A0 is connected directly to the SEL pin on the HCTL-2020. This line selects the low or high byte of data from the HCTL-2020.

Cascading is accomplished by connecting the CNT_{CAS} output on the HCTL-2020 with the counter clock (CCK) input on both 74LS697's. The U/$\overline{D}$ pin on the HCTL-2020 and the U/$\overline{D}$ pin on both 74LS697's are also directly connected for easy expansion. The $\overline{RCO}$ of the first 4-bit 74LS697 is connected to the $\overline{ENT}$ pin of the second 74LS697. This enables the second counter only when there is a $\overline{RCO}$ signal on the first counter.

This configuration allows the 6802 to read both data bytes with a single double-byte fetch instruction (LDX 2XX0). This instruction is a five cycle instruction which reads external memory location 2XX0 and stores the high order byte into the high byte of the index register. Memory location 2XX1 is next read and stored in the low order byte of the index register. The high byte of counter data is clocked into the 74LS697 registers when SEL is high and $\overline{OE}$ goes low. This upper byte can be read at any time by pulling the 74LS697 $\overline{G}$ low when reading address 4XXX. Figure 15 shows memory addresses and gives an example of reading the HCTL-2020. Figure 16 shows the interface timing for the circuit.

Address	Function
CXXX	Reset Counters
4XXX	Enable High Byte on Data Lines
2XX0	Enable Low Byte on Data Lines
2XX1	Enable Mid Byte on Data Lines

Read Example	
LDX 2000	Loads mid byte and then low byte into
STX 0100	memory locations 0100 and 0101
LDAA 4000	Loads the high byte into memory
STAA 0102	location 0102

Figure 15. Memory Addresses and Read Example

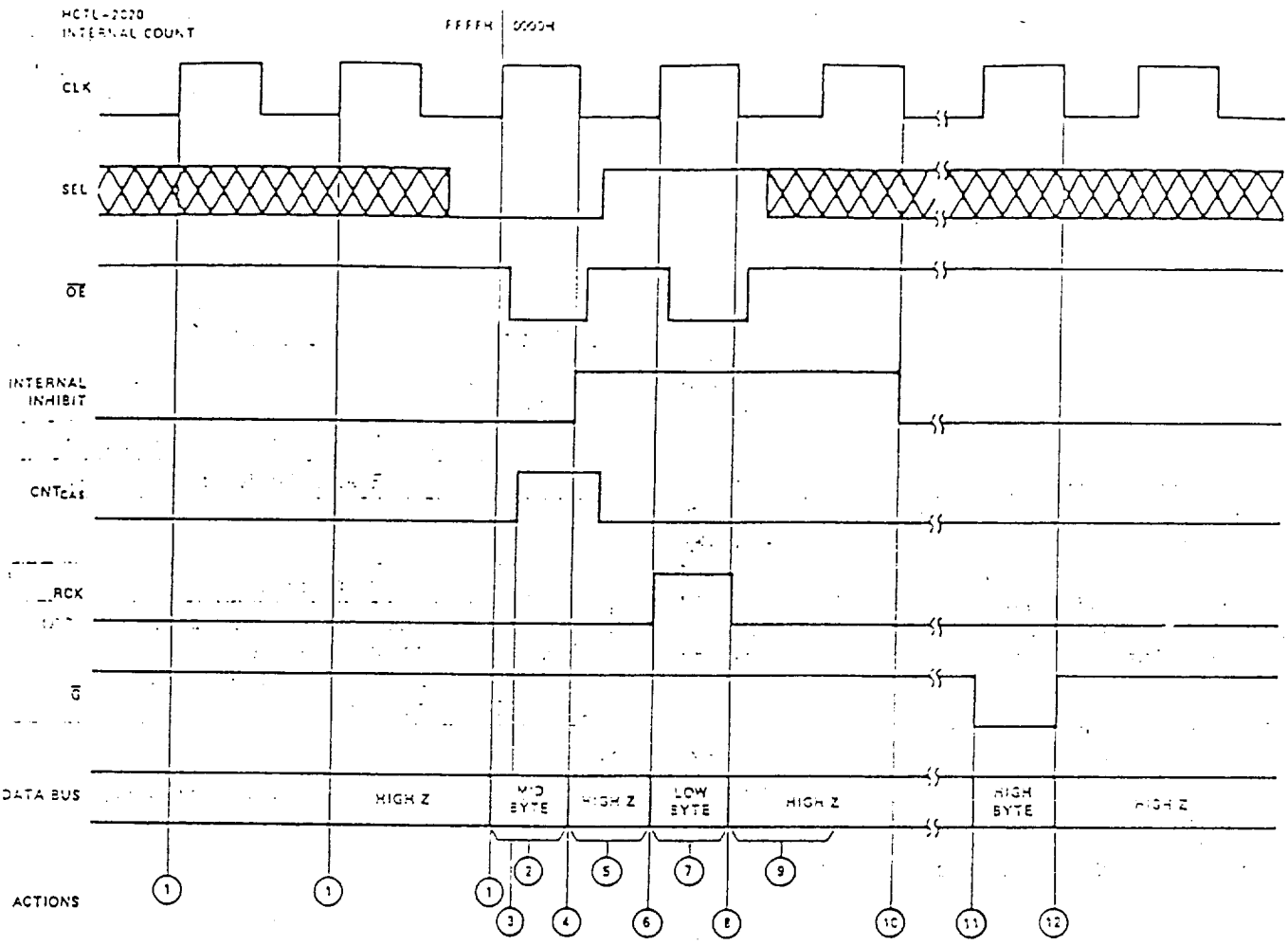


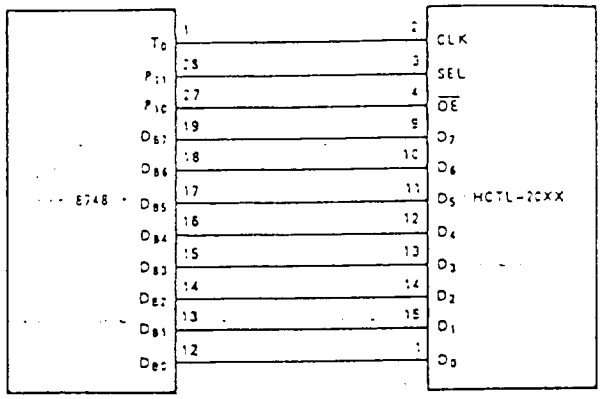
Figure 16. Interface Timing for the 6802/8

ACTIONS

1. The microprocessor clock output is E. If the internal HCTL-2020 inhibit is not active, new data is transferred from the internal counter to the position data latch.
2. An even address output from the 6802 causes SEL to go low. When E goes high, the address decoder output for the HCTL-2020 $\overline{OE}$ signal goes low. This causes the HCTL-2020 to output the middle byte of the system counter (high byte of the HCTL-2020 counter).
3. In this case, the HCTL-2020 counter has overflowed and there is an output on the CNT_{CAS} line. This pulse is counted by the 74LS697 but not loaded into the output register of the 74LS697 at this time.
4. The 6802 reads the data bus on the falling edge of E, storing the high order 2020 data byte (middle system byte) into the high byte of the index register. The chip detects that $\overline{OE}$ and SEL are low on the falling edge of E and activates the internal inhibit signal. The position data latch is inhibited and data cannot be transferred from the internal counter to the latch.
5. When E goes low, the address decoder output is disabled and $\overline{OE}$ goes high. The 6802 increments the address, causing SEL to go high. The position data latch is still inhibited.
6. When SEL is high and $\overline{OE}$ is low the 74LS697 register clock (RCK) goes high. The rising edge of RCK loads the 74LS697 count into the 74LS697 register. Delaying the RCK signal until the second $\overline{OE}$ allows for delays on the CNT_{CAS} signal.
7. The address decoder is enabled after E goes high. The $\overline{OE}$ line goes low and the low data byte is enabled onto the bus.
8. The 6802 reads the data bus on the falling edge of E, storing the low order data byte into the low byte of the index register. The HCTL-2020 detects that $\overline{OE}$ is low and SEL is high on the falling edge of E, thus meeting the first inhibit reset condition.
9. When E goes low, the address decoder is disabled, causing $\overline{OE}$ to go high and the data lines to go to the high impedance state. The 6802 continues its instruction execution, and the state of SEL is indeterminate.
10. The HCTL-2020 detects $\overline{OE}$ is high on the next falling edge of E. This satisfies the second inhibit reset condition and the inhibit signal is reset.
11. When E goes high, a new address causes the $\overline{G}$ line on the 74LS697 to go low and enables the high byte onto the data bus.
12. When E goes low, the high byte is read into the 6802. The data bus returns to tri-state.

Interfacing the HCTL-20XX to an Intel 8748

The circuit shown in Figure 17 shows the connections between an HCTL-20XX and an 8748. Data lines D0-D7 are connected to the 8748 bus port. Bits 0 and 1 of port 1 are used to control the SEL and $\overline{OE}$ inputs of the HCTL-20XX respectively. T0 is used to provide a clock signal to the HCTL-20XX. The frequency of T0 is the crystal frequency divided by 3. T0 must be enabled by executing the ENT0 CLK instruction after each system reset, but prior to the first encoder position change. An 8748 program which interfaces to the circuit in Figure 17 is given in Figure 18. The resulting interface timing is shown in Figure 19.



* NOTE: PIN NUMBERS ARE DIFFERENT FOR THE HCTL-2020

Figure 17. An HCTL-20XX-to-Intel 8748 Interface

LOC	Object Code	Source Statements	Comments
000	99 00	ANL P1, 00H	Enable output and higher order bits
002	08	INS A, BUS	Load higher order bits into ACC
003	A8	MOVE R0 A	Move data to register 0
004	89 03	ORL P1, 01H	Change data from high order to low order bits
006	08	INS A, BUS	Load order bits into AC
008	A9	MOV R1, A	Move data to register 1
009	89 03	ORL P1, 03H	Disable outputs
00B	93	RETR	Return

Figure 18. A Typical Program for Reading HCTL-20XX with an 8748

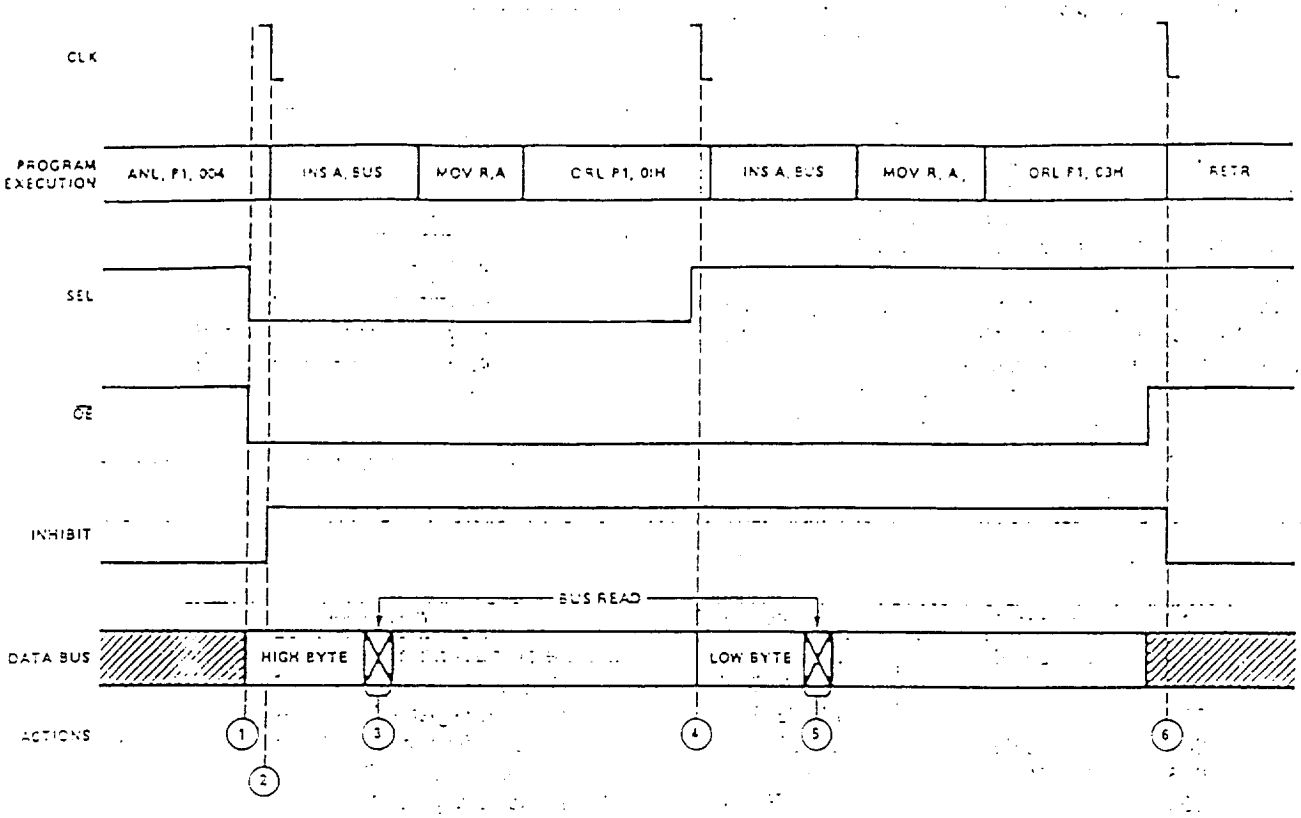


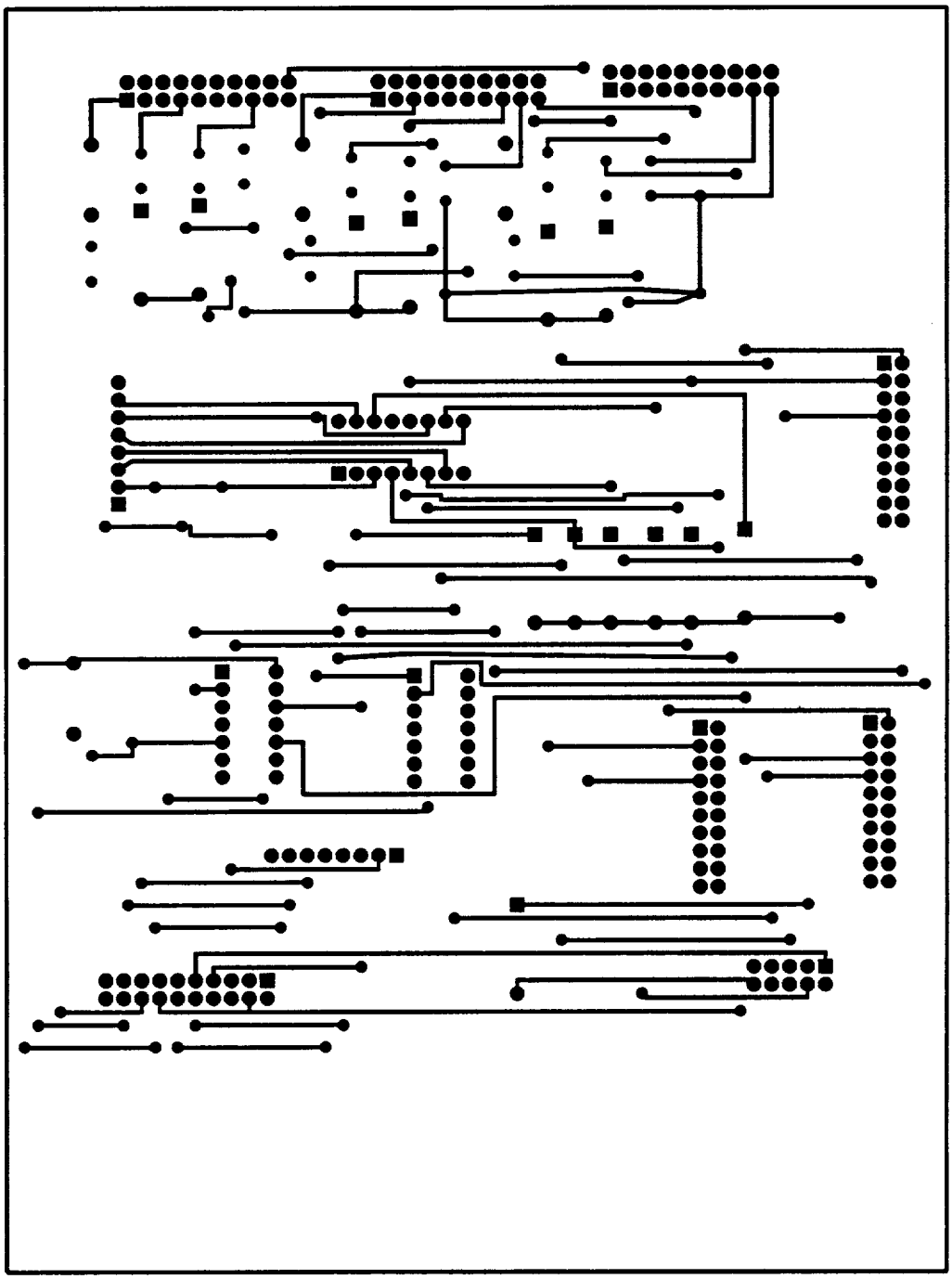
Figure 19. 8748 READ Cycle from Figure 18.

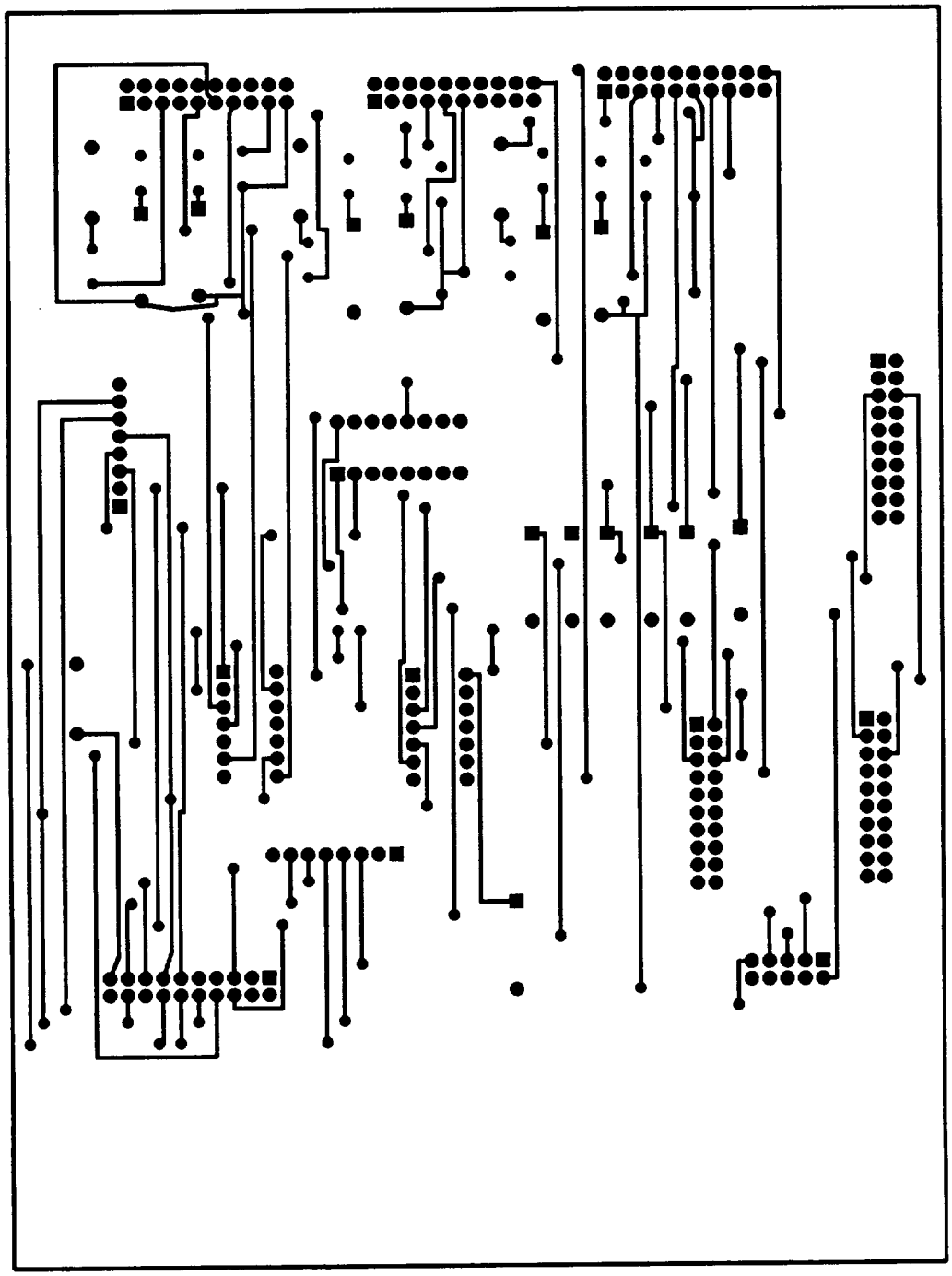
ACTIONS

1. ANL P1, 00H has just been executed. The output of bits 0 and 1 of Port 1 cause SEL and $\overline{OE}$ to be logic low. The data lines output the higher order byte.
2. The HCTL-20XX detects that $\overline{OE}$ and SEL are low on the next falling edge of the CLK and asserts the internal inhibit signal. Data can be read without regard for the phase of the CLK.
3. INS A, BUS has just been executed. Data is read into the 8748.
4. ORL PORT 1, 01H has just been executed. The program sets SEL high and leaves $\overline{OE}$ low by writing the correct values to port 1. The HCTL-20XX detects $\overline{OE}$ is low and SEL is high on the next falling edge of the CLK, and thus the first inhibit reset condition is met.
5. INS A, BUS has just been executed. Lower order data bits are read into the 8748.
6. ORL P1, 03H has just been executed. The HCTL-20XX detects $\overline{OE}$ high on the next falling edge of CLK. The program sets $\overline{OE}$ and SEL high by writing the correct values to port 1. This causes the data lines to be tri-stated. This satisfies the second inhibit and reset condition. On the next rising CLK edge new data is transferred from the counter to the position data latch.

APPENDIX II.E

Printed Circuit Board Art Work





APPENDIX II.F

Force Sensing Resistor Information

A TechNote:

Suggested Electrical Interfaces for Force Sensing Resistors™

INTERLINK ELECTRONICS
546 Flynn Road, Camarillo CA 93012 • 805-484-1331 • 805-484-8989

TABLE OF CONTENTS

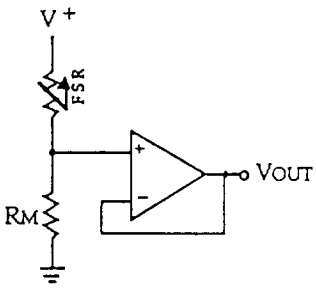
This TechNote is designed to help electrical designers build interfaces that result in the successful integration of FSR™ components into products. Interlink Electronics has sucessfully developed and marketed products incorporating the FSR. The most recognizable of these are VersaPoint® computer pointing devices. VersaPoint® products incorporate 4-zone FSRs and VersaPoint® electronics. VersaPoint® products provide one-touch cursor control with no moving parts (optical encoders, balls, etc.) to gum up, wear out or break.

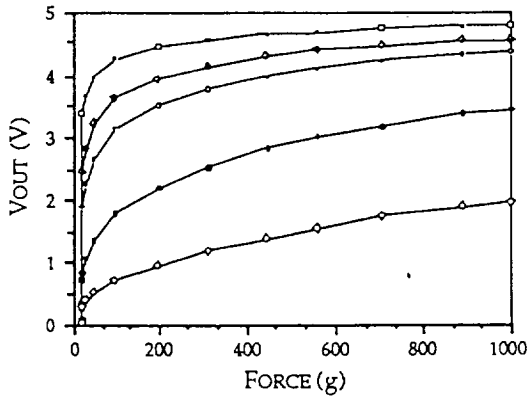
- PortaPoint® (for laptop and notebook computers)
- DuraPoint™ (for industrial computing)
- ProPoint™ (for hand-held interactive computing)

Available VersaPoint® OEM modules include:

- MicroModule™ (the world's thinnest in-board cursor controller)
- Micro Joystick (the joystick that fits anywhere)

Most successful FSR interfaces start with the same building blocks. In this TechNote is a collection of circuits that effectively empower the designer to modify and build customized circuits that complement their product.





RM Values
100k
47k
30k
10k
3k

Graph 1.1
F vs. V for part # 174
Interlink Force Tester
1 cm² circular flat
metal actuator.

FSR Voltage Divider:
For a simple force-to-voltage conversion, the FSR is tied to a measuring resistor in a voltage divider configuration. The output is described by the equation:

$$V_{OUT} = (V+) / [1 + R_{FSR}/R_M].$$

In the shown configuration, the output voltage increases with increasing force. If R_{FSR} and R_M are swapped, the output swing will decrease with increasing force. These two output forms are mirror images about the line $V_{OUT} = (V+)/2$.

The measuring resistor, R_M, is chosen to maximize the desired force sensitivity range and to limit current. The current through the FSR should be limited to less than 1 mA/square cm of applied force. Suggested op-amps for single sided supply designs are LM358 and LM324. FET input devices such as LF355 and TL082 are also good. The low bias currents of these op-amps reduce the error due to the source impedance of the voltage divider.

A family of FORCE vs. VOUT curves is shown on Graph 1.1 for a standard FSR in a voltage divider configuration with various RM resistors. A (V+) of +5V was used for these examples.

Since human touch is non-linear, with greater resolution at light forces and lesser resolution at harder forces, response curves like those provided by the 30kΩ or 47kΩ RM resistors can better approximate the human "feel" reaction, combining with human touch to give a more subjectively linear control system.

By adding a resistor in parallel with R_{FSR}, increased low force resolution can be attained. The presence of this resistor establishes a set voltage when the FSR is unloaded, and therefore, allows small changes from this set value to be more easily detected.

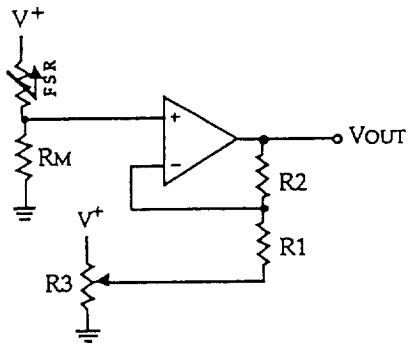


Figure 1.1

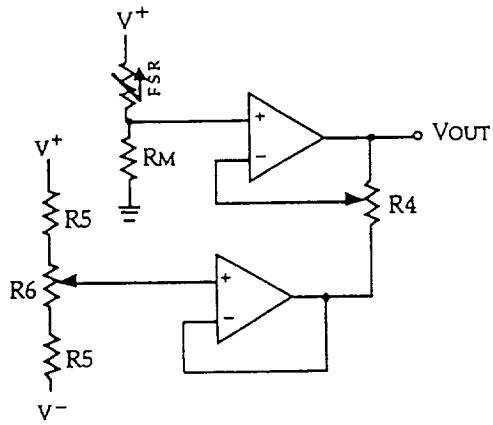
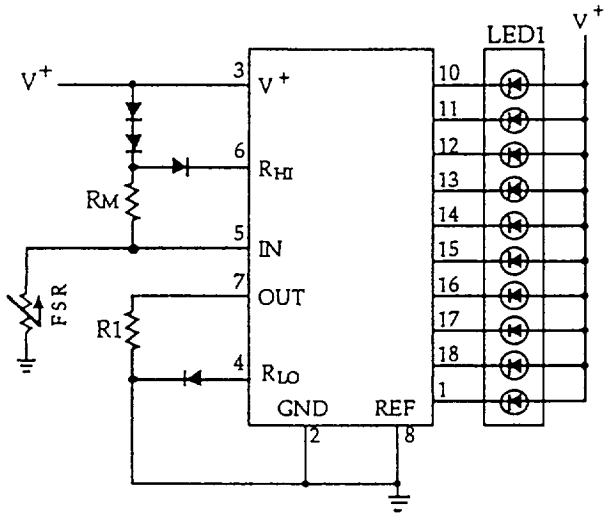


Figure 1.2

Adjustable Buffers:
Similar to the unity gain buffer, these interfaces isolate the output from the high source impedance of the FSR. These alternatives allow adjustment of the output offset and gain.

In Figure 1.1, the ratio of resistors R2 and R1 sets the gain of the output. Offsets resulting from the non-infinite FSR resistance at zero force (or bias currents) can be trimmed out with the potentiometer, R3. For best results, R3 should be about one-twentieth of R1 or R2. Adding an additional pot at R2 makes the gain easily adjustable. Broad range gain adjustment can be made by replacing R2 and R1 with a single pot.

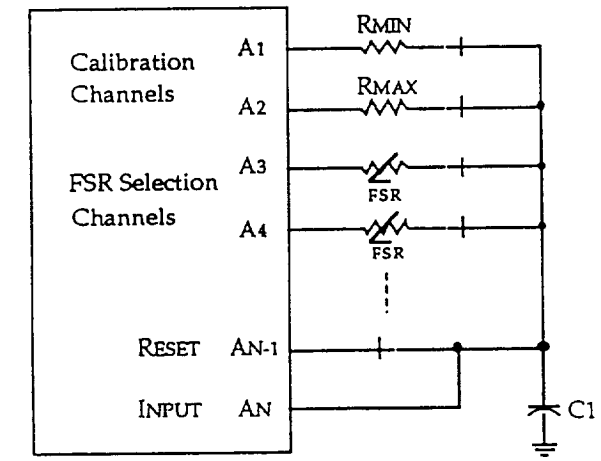
The circuit in Figure 1.2 yields similar results to the previous one, but the offset trim is isolated from the adjustable gain. With this separation, there is no constraint on values for R6. Typical values for R5 and R6 are around 10kΩ.



FSR LED Display Demo:
This simple FSR LED display demo uses a display driver to translate an analog input into ten separate voltage levels. As each level is attained, current is drawn through an LED, lighting it. Consecutive LEDs in a bank are lit as the input voltage goes from high to low. The LM3915 is a logarithmic display driver. Since the FSR response curve, and therefore the input, is roughly logarithmic, a logarithmic driver better follows the FSR response and thus gives a more linear feel to the displayed LED output.

The input consists of the FSR in a simple voltage divider with R_M. The resistor R_M (30kΩ) is chosen to maximize the use of the FSRs dynamics and to limit current through the FSR. Replacing R_M with a 10kΩ resistor in series with a 50kΩ pot will allow the user to adjust the sensitivity range of the FSR, while maintaining current limits. As the pot is adjusted to a low value, the high force dynamics of the FSR will be displayed. When the pot is adjusted to a high value, the low force dynamics will be displayed.

The display driver has an internal unity gain buffer at the input that isolates the FSR and minimizes bias current errors. In the configuration shown, a 9 volt battery is used as the voltage source. The top of the driver's ladder is about 7.2 volts (3 diode drops below V+), while the bottom is at about 0.6V (1 diode drop above ground). The input ranges from 7.8 volts (2 diode drops below V+) to ground. At the output, R1 is used to control the amount of current drawn through the LEDs, and therefore controls LED brightness. Pin 9 of the driver can be tied to the voltage source if bar output instead of dot output is desired.



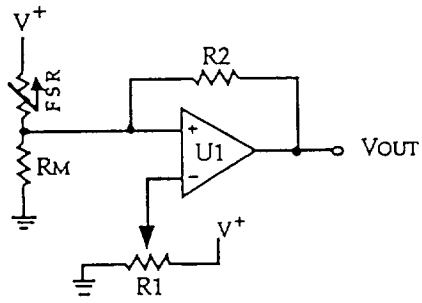
Multi-Channel FSR-to-Digital Interface:

Sampling Cycle (any FSR channel):

The microcontroller switches to a specific FSR channel, toggling it high, while all other FSR channels are toggled low. The RESET channel is toggled high, a counter starts and the capacitor C1 charges, with its charging rate controlled by the resistance of the FSR (t ~ RC). When the capacitor reaches the high digital threshold of the INPUT channel, the counter shuts off, the RESET is toggled low, and the capacitor discharges.

The number of "counts" it takes from the toggling of the RESET high to the toggling of the INPUT high is proportional to the resistance of the FSR. The resistors RMIN and RMAX are used to set a minimum and maximum "counts" and therefore the range of the "counts". They are also used periodically to re-calibrate the reference. A sampling cycle for RMIN is run, the number of "counts" is stored and used as a new zero. Similarly, a sampling cycle for RMAX is run and the value is stored as the maximum of the range (after subtracting the RMIN value). Successive FSR samplings are normalized to the new zero. The full range is "zoned" by dividing the normalized maximum "counts" by the number of desired zones. This will delineate the window size or width of each zone.

Continual sampling is done to record changes in FSR resistance due to changes in force. Each FSR is selected sequentially.

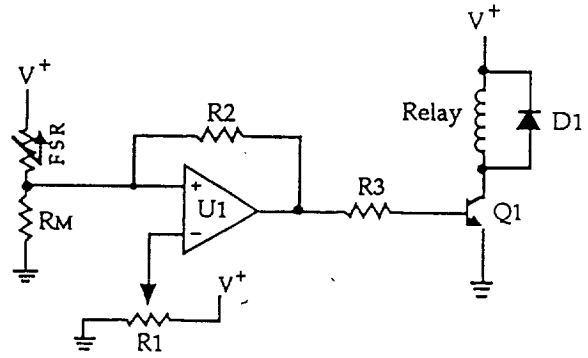


FSR Variable Force Threshold Switch:
This simple circuit is ideal for applications that require on-off switching at a specified force, such as touch sensitive membrane, cut-off, and limit switches. For a variation of this circuit that is designed to control relay switching, see TechNote 1-6.

The FSR is arranged in a voltage divider with RM. An op-amp, U1, is used as a comparator. The output of U1 is either high or low. The non-inverting input of the op-amp is driven by the output of the divider, which is a voltage that increases with force. At zero force, the output of the op-amp will be low. When the voltage at the non-inverting input of the op-amp exceeds the voltage of the inverting input, the output of the op-amp will toggle high. The triggering voltage, and therefore the force threshold, is set at the inverting input by the pot R1. The hysteresis resistor, R2, acts as a "debouncer", eliminating any multiple triggering of the output that might occur.

Suggested op-amps are LM358 and LM324. Comparators like LM393 and LM339 also work quite well. The parallel combination of R2 with RM is chosen to limit current and to maximize the desired force sensitivity range. A typical value for this combination is about 47kΩ.

The threshold adjustment pot, R1, can be replaced by two fixed value resistors in a voltage divider configuration.

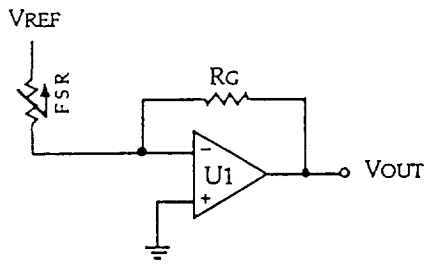


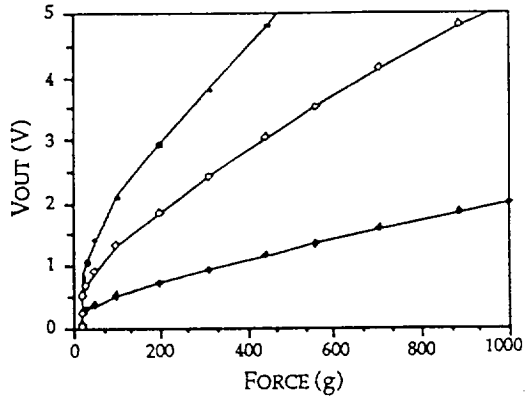
FSR Variable Force Threshold Relay Switch:
This circuit is a derivative of the simple FSR Variable Force Threshold Switch of TechNote 1-5. It has use where the element to be switched requires higher current, like automotive and industrial control relays.

The FSR is arranged in a voltage divider with R_M. An op-amp, U1, is used as a comparator. The output of U1 is either high or low. The non-inverting input of the op-amp sees the output of the divider, which is a voltage that increases with force. At zero force, the output of the op-amp will be low. When the voltage at the non-inverting input of the op-amp exceeds the voltage of the inverting input, the output of the op-amp will toggle high. The triggering voltage, and therefore the force threshold, is set at the inverting input by the pot R1. The transistor Q1 is chosen to match the required current specification for the relay. Any medium power NPN transistor should suffice. For example, an NTE272 can sink 2 amps, and an NTE291 can sink 4 amps. The resistor R3 limits the base current. The hysteresis resistor, R2, acts as a "debouncer", eliminating any multiple triggering of the output that might occur.

Suggested op-amps are LM358 and LM324. Comparators like LM393 and LM339 also work quite well, but must be used in conjunction with a pull-up resistor. The parallel combination of R2 with R_M is chosen to limit current and to maximize the desired force sensitivity range. A typical value for this combination is about 47kΩ.

The threshold adjustment pot, R1, can be replaced by two fixed value resistors in a voltage divider configuration. The diode D1 is included to prevent flyback, which could harm the relay and the circuitry.





Graph 1.2
F vs. V for part # 174 Interlink Force Tester 1 cm ^{2} circular flat metal actuator.

FSR Current-to-Voltage Converter:
In this circuit, the FSR is the input of a current-to-voltage converter. The output of this amplifier is described by the equation:

$$V_{OUT} = V_{REF} \bullet [-R_G / R_{FSR}].$$

With a positive reference voltage, the output of the op-amp must be able to swing below ground, from 0V to -VREF, therefore dual sided supplies are necessary. A negative reference voltage will yield a positive output swing, from 0V to +VREF..

Since this a simple inverse relation between VOUT and RFSR, the output equation can be re-arranged to:

$$V_{OUT} = (-R_G \bullet V_{REF}) / R_{FSR}$$

VOUT is inversely proportional to RFSR. Changing RG and/or VREF changes the response slope. The following is an example of the sequence used for choosing the component values and output swing:

For a human-to-machine variable control device, like a joystick, the maximum force applied to the FSR is about 1kg. Testing of a typical FSR shows that the corresponding RFSR at 1kg is about 4.6kΩ. If VREF is -5V, and an output swing of 0V to +5V is desired, then RG should be approximately equal to this minimum RFSR. RG is set at 4.7kΩ. A full swing of 0v to +5V is thus achieved. A set of FORCE vs. VOUT curves is shown on Graph 1.2 for a standard FSR using this interface with a variety of RG values.

The current through the FSR should be limited to less than 1 mA/square cm of applied force. As with the voltage divider circuit, adding a resistor in parallel with RFSR will give a definite rest voltage, which is essentially a zero-force intercept value. This can be useful when resolution at low forces is desired.

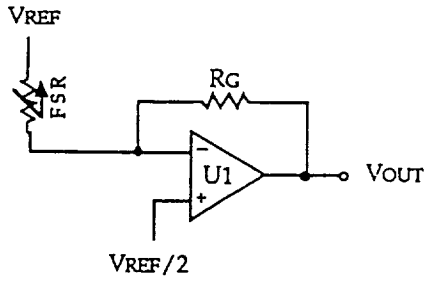


Figure 1.3

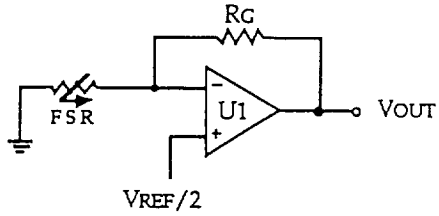


Figure 1.4

Additional FSR Current-to-Voltage Converters:
These circuits are a slightly modified versions of the current-to-voltage converter detailed on TechNote 1-7. Please see TechNote 1-7 for more detail.

The output of Figure 1.3 is described by the equation:

$$V_{OUT} = V_{REF}/2 \bullet [1 - R_G/R_{FSR}].$$

The output swing of this circuit is from ($V_{REF}/2$) to 0V. In the case where R_G is greater than R_{FSR}, the output will go into negative saturation.

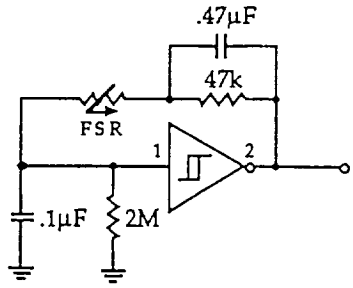
The output of Figure 1.4 is described by the equation:

$$V_{OUT} = V_{REF}/2 \bullet [1 + R_G/R_{FSR}].$$

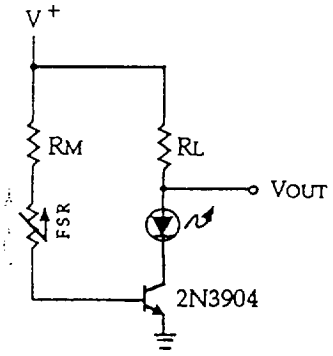
The output swing of this circuit is from ($V_{REF}/2$) to V_{REF}. In the case where R_G is greater than R_{FSR}, the output will go into positive saturation.

For either of these configurations, a zener diode placed in parallel with R_G will limit the voltage built up across R_G. These designs yield one-half the output swing of the previous circuit, but only require single sided supplies and positive reference voltages. Like the preceding circuit, the current through the FSR should be limited to less than 1 mA/square cm of applied force.

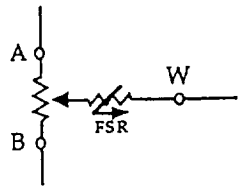
Suggested op-amps are LM358 and LM324.



FSR Schmitt Trigger Oscillator:
In this circuit, an oscillator is made using the FSR as the feedback element around a Schmitt Trigger. In this manner, a simple force-to-frequency converter is made. At zero force, the FSR is an open circuit. Depending on the last stage of the trigger, the output remains constant, either high or low. When the FSR is pressed, the oscillator starts, its frequency increasing with increasing force. The 2MΩ resistor at the input of the trigger insures that the oscillator is off when FSRs with non-infinite resistance at zero force are used. The 47kΩ resistor and the 0.47 µF capacitor control the force-to-frequency characteristic. Changes in the "feel" of this circuit can be made by adjusting these values. The 0.1 µF capacitor controls the frequency range of the oscillator. By implementing this circuit with CMOS or TTL, a digital process can be controlled by counting leading and/or trailing edges of the oscillator output. Suggested Schmitt Triggers are CD40106, CD4584 or 74C14.

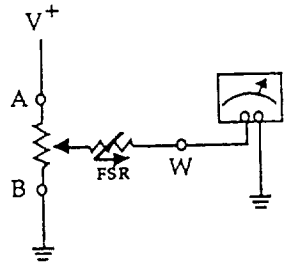


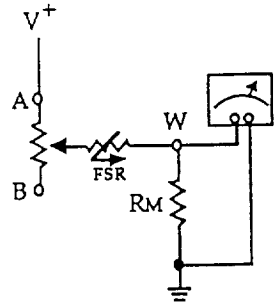
FSR LED Brightness:
For applications where some visual feedback is desired, this circuit is useful. Starting with the basics of the voltage divider, this circuit adds an LED that brightens with increasing force. The resistor RL limits the current through the LED. The transistor controls the current flowing through the LED. Since the circuit depends on the h_{fe} of the transistor, sensitivity may need to be tuned to accomodate the h_{fe} spread of common transistors.



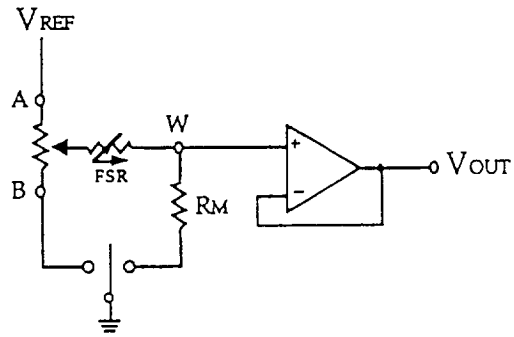
The Interlink Linear Potentiometer:
This diagram shows the equivalent circuit for an Interlink LP. The contact points A and B denote the ends of the fixed resistor that is used to measure position. W denotes the wiper contact. The equivalent circuits for force or position sensing are detailed below.

Positional Measurement with the LP:
In common LP usage, a voltage is applied across A and B to create a measurable gradient. Thus, the voltage measured at the wiper is proportional to the distance along the LP. If no current is drawn through the FSR element, positional measurement is made independent of force. A high impedance measuring device, like a digital multimeter, is excellent for positional measurement without force-resistance corruption. Notice that the voltage gradient is independent of the value of the positional resistor. A gradient of 0V to V+ is established regardless of the fixed resistor value. The important resistor parameter is linearity, which is typically better than 1%. Functionally, the purpose of the fixed resistor is to limit current while not substantially interfering with the force measurement.





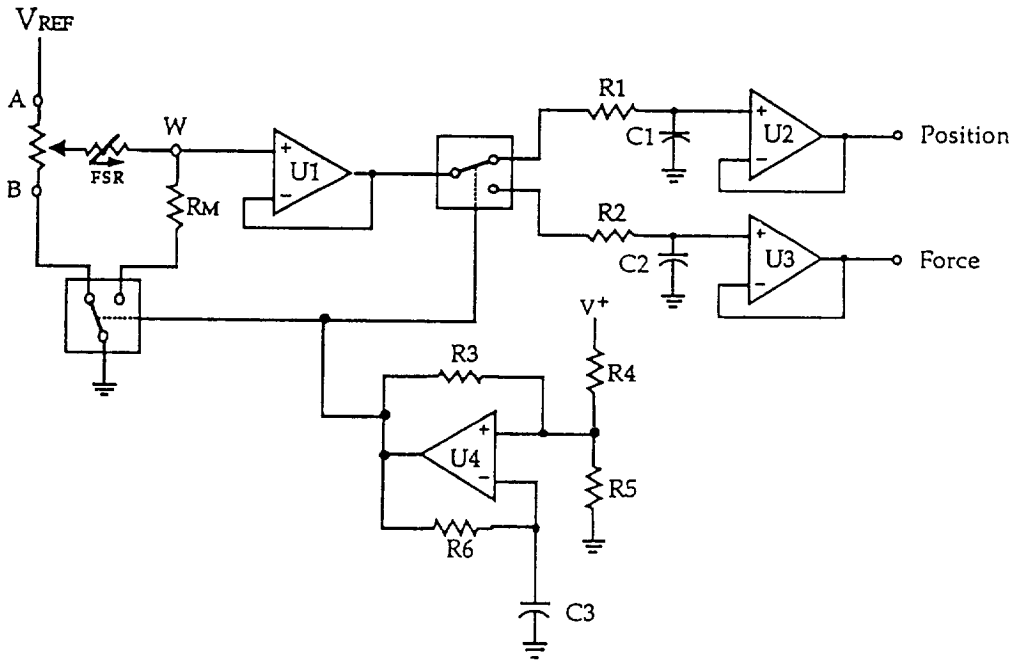
Force Measurement with the LP:
For rough force measurements, a resistor, R_M, is connected in series with the FSR. This forms a voltage divider with the FSR and R_M. The voltage measured across R_M is proportional to the force on the LP. Some error is inherent in this force measurement. By following the path of current from V+ to ground, it can be seen that, depending on position, more or less of the fixed resistor will be in series with R_{FSR}. This added series resistance causes error in the measured output of the voltage divider. Some minimization of the force error can be achieved by tying A and B together during force measurement. Alternatively, this error can be corrected for in the interpretive software.



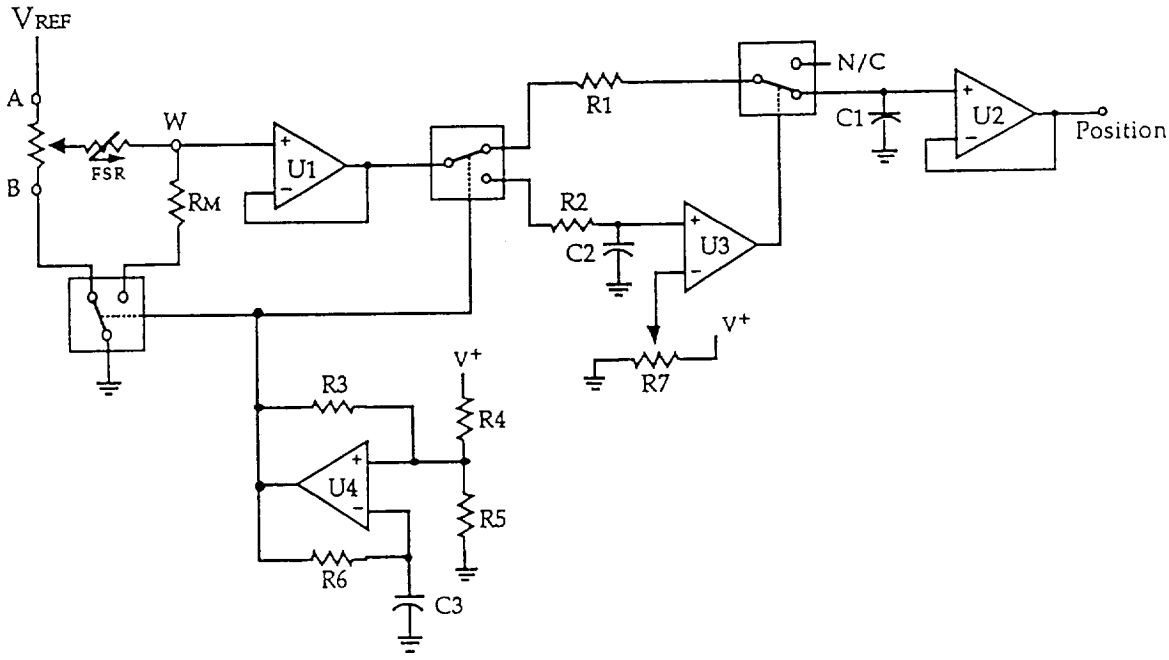
Simple LP Analog Interface:
This interface combines both of the previous circuits. A switch is used to toggle between force and position measurements. A unity gain buffer has been added to isolate the LP from the succeeding circuitry. Additionally, the low input impedance of this buffer keeps current from flowing through the FSR resistor (wiper) during position measurement, and drives current through the FSR resistor and RM during force measurement.

This interface is excellent for measurements against VREF through an A/D.

Suggested op-amps for this buffer are LM358 and LM324. FET input devices such as LF355 and TL072 are also good. Suitable reference voltages for this and similar LP circuits should limit current to below 1mA.



Separated Force and Position Analog LP Interface:
This circuit uses a multiplexer to yield nearly simultaneous force and position outputs. The multiplexer is driven by a Schmitt trigger oscillator. The resistor-capacitor combination at the non-inverting inputs of the op-amps act as simple sample-and-hold circuits between cycles. This interface utilizes two ICs, a CD4053 and a LM324.



Position Measuring Analog LP Interface with Force Threshold:
In cases where position is the desired measurement, this interface is useful. The force sensing aspect is being used to set a force threshold for positional data output. Once the force threshold is exceeded, the output of op-amp U3 goes high and the succeeding multiplexer segment toggles to positional output. This force threshold is adjusted with the pot, R7.

Resistor-capacitor combinations act as simple sample and hold circuits between cycles.

Suggested ICs for this interface are a CD4053 for the multiplexer and a LM324 for the op-amp.

An enhanced sample and hold can be achieved by replacing U2 with a JFET type input op-amp, such as a LF353 or a TL071.

Linear Potentiometer LED Display Demo:

This interface (Refer to Figure 2.1) can be broken down into five functional blocks:

1 **Regulated Reference:**
The regulated reference uses a npn transistor (Q1), two diodes and a pull down resistor (R1) to regulate the voltage at the top of the fixed resistor of the LP.

2 **Unity Gain Buffer:**
A unity gain buffer (U1A) is used to isolate the force sensing aspect of the LP from the multiplexer (U3) and the position sensing aspect of the LP. Adjustments at the measuring resistor RM will change the force sensitivity range of the LP. The resistor R2 filters noise out of the positional signal before it reaches the input of the multiplexer.

3 Multiplexer:
The multiplexer (U3) clocks back and forth between the force and position signals. The clocking rate is defined by the input from the oscillator block.

4 Oscillator:
The oscillator block uses a comparator for two functions. The segment labeled U2B is a simple A/D converter that gives a square wave output to drive the multiplexer. The input to U2B comes from the charging and discharging of capacitor C3. The segment labeled U2A is used to set a force threshold for the multiplexing to begin. The pot R8, allows adjustment of this force threshold.

5 Output:
The LP LED Display Demo (Figure 2.1) utilizes two display drivers coupled with LED banks to show the force and position sensing of the LP. The LM3915 logarithmic driver translates the analog force output of the multiplexer into ten LED indicated (D1) voltage levels. The logarithmic driver follows the logarithmic response of the FSR, giving a more linear feel to the displayed LED output. Similarly, a LM3914 linear driver translates the analog positional output of the multiplexer into ten LED indicated (D2) voltage levels. The linear driver is used because the position sensing of the LP is very linear. Each of the drivers uses a capacitor, C1 and C2, to hold the analog signal level between sampling cycles. The resistors R9 and R11 control the amount of current drawn through the LEDs, and therefore LED brightness.

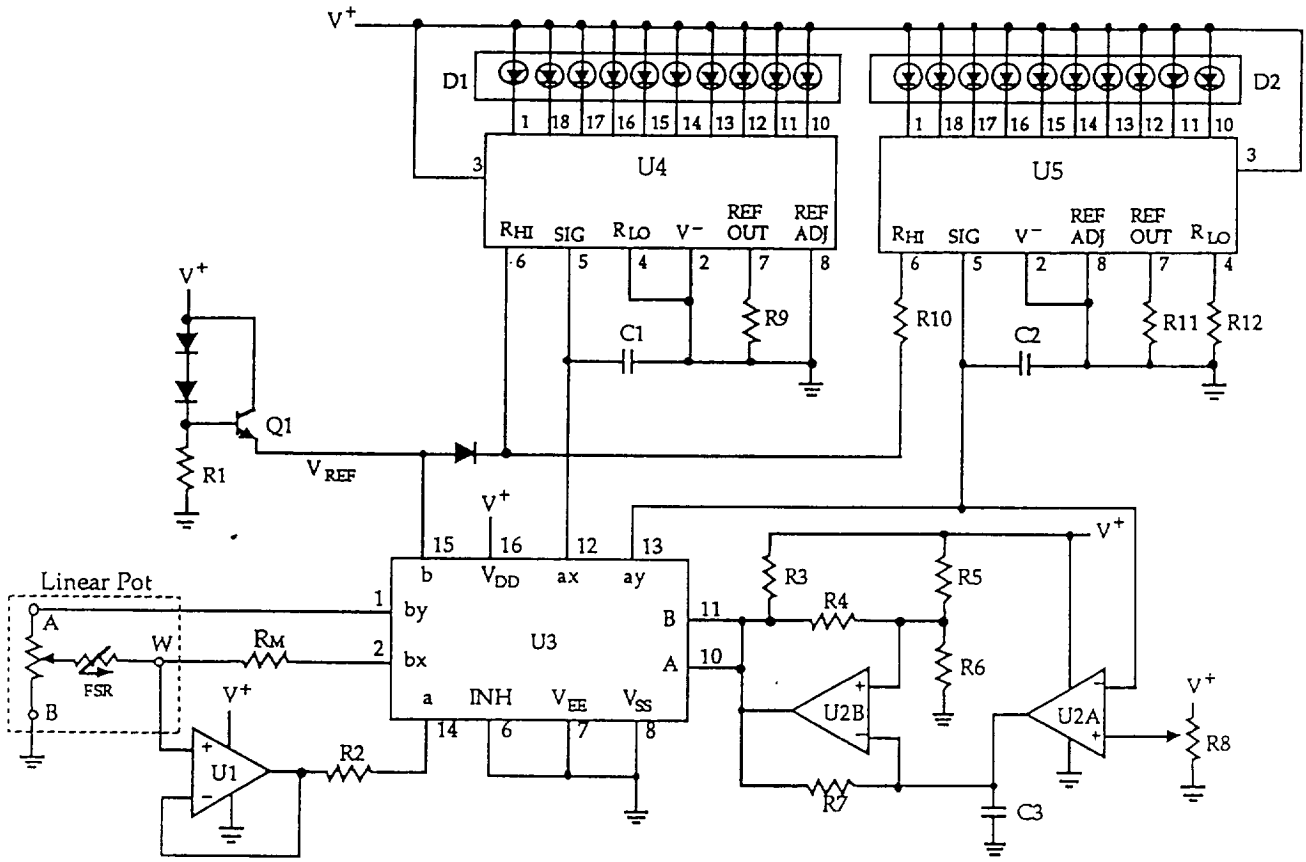


Figure 2.1, Linear Potentiometer LED Display Demo

Suggested Components for LP LED Display Demo:

Linear Pot	Interlink Part # 155	D1, D2	LED Bank, Ten Segment
R_M	47kΩ		
R1, R2, R12	10kΩ	Diodes	1N914
R3	4.7kΩ		
R4, R5, R6	100Ω	Q1	NPN 2N3904
R7	75kΩ		
R8	Pot, 10kΩ	U1	LM358
R9, R10 & R11	1kΩ	U2 A,B	LM393
		U3	CD4053
C1, C2	0.47 µF	U4	LM3914
C3	0.068 µF	U5	LM3915

Linear Potentiometer Pulse Width Modulation

These interfaces (Refer to Figures 2.2, 2.3 and 2.4) can be separated into seven functional blocks:

1 **Linear Potentiometer with Voltage Divider and Unity Gain Buffer:**
The LP is set up in a voltage ladder with resistors R1 and R2. R1 and R2 set the upper and lower ranges of the voltage output from the divider. The measuring resistor, RM, is used to set the sensitivity range of the force sensing aspect of the LP. The op-amp, U2A, acts as a unity gain buffer. The MUX segment U1C toggles back and forth between force and position measurement.

2 **Force Threshold Control:**
This aspect of the interface allows the user to set a force threshold for the switching of the multiplexer to positional output. The op-amp, U2B, is being used as a simple A/D converter. The pot, R5, adjusts the force threshold. When the force on the LP exceeds the chosen force threshold, the output of the op-amp goes high and the following MUX segment, U1A, switches to give the positional output. The capacitor C1 holds the force value between force and position sampling cycles.

3 **Oscillator:**
This circuit uses another op-amp as a Schmitt trigger oscillator. The input to the op-amp comes from the charging and discharging of the capacitor, C2. The output of the op-amp toggles high and low. This square wave is used to drive the multiplexer (U1) which clocks back and forth between force and position signals.

4 **Load and Pulse Width Modulation:**
In this interface, the load, or device to be controlled, is represented by the resistor RL. This load could be a DC motor, or a half wave universal device. The pulse width modulation begins at the inputs to the op-amp U2C. The voltage input to pin 10 is in the form of a constant triangle wave that is drawn from the oscillator capacitor C2. The input to pin 9 is the positional signal, which is a voltage that varies from the low to the high of the triangle wave. This range is set by the LP, R1 and R2 resistance ladder. The output of the op-amp is a square wave that varies in pulse width. As the voltage level at pin 9 of the op-amp goes up, the pulse width becomes broader. The transistor, Q1, acts as a current switch, allowing the flow of current from V+ to ground through RL. The transistor is open while the output of the op-amp is high. The resistor R7 limits the current at the base of the transistor.

5 **Simple Memory:**
The capacitor, C3, acts as a sample and hold circuit. This simple memory circuit can hold the positional value for short periods of time without excess error from leakage.

6 **Current Limiting:**
In Figure 2.3, a feedback circuit has been added. This enhancement allows active control of the voltage range across the resistance ladder by the formation of a closed loop system. The op-amp, U2D, measures the current through the load. If the load current attempts to surpass the threshold set by R15, the op-amp responds with a voltage change at R2, the lower leg of the resistance ladder. This feedback inhibits the overdriving of the load. Since the voltage at R6 is very low, high gain amplification is necessary. The capacitor C4 and the resistor R14 act as a filter for noise developed by the switching of the transistor, and any noise that may be inherent in the load. The feedback signal amplifier, U2D, utilizes the last segment of the quad op-amp. An additional op-amp, U3, is added to perform the oscillator function.

7 **Enhanced Memory:**
In Figure 2.4, an enhanced memory circuit has been added. This circuit replaces the simple memory circuit of Figure 2.2 and 2.3. For applications where the LP will be activated by touch, this addition is ideal for holding the value of the last touched position for long lengths of time. The capacitor, C5, holds the positional value. The op-amp, U3B, acts as a buffer, with its output clamped by the two diodes. The use of a JFET input op-amp greatly decreases any leakage due to the bias current of the succeeding op-amp, U2C.

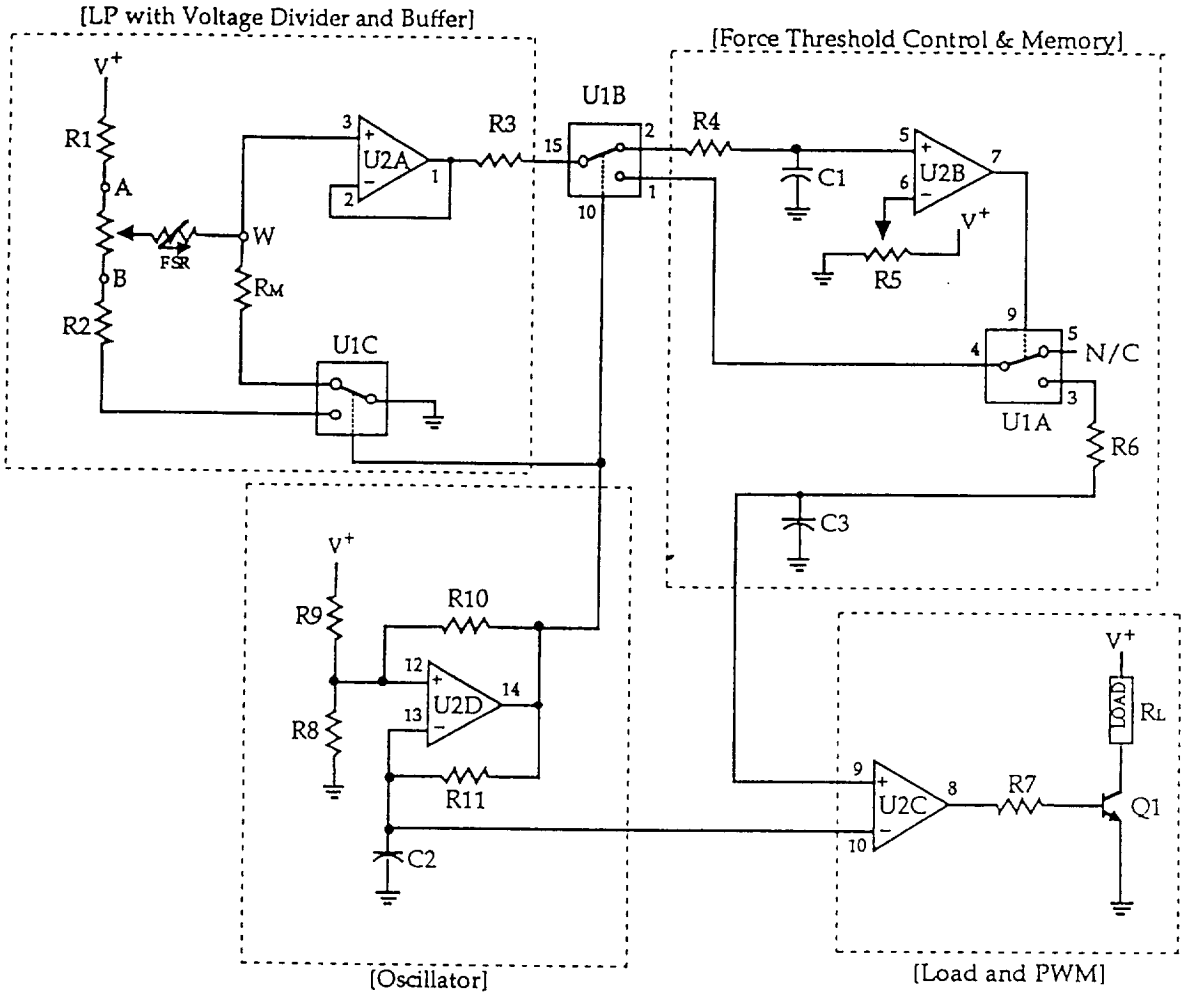


Figure 2.2, LP with PWM Output, Force Thresholding and Simple Memory.

Suggested Components:

R1 & R2	3.9kΩ	U1	Multiplexer, CD4053
RM	47kΩ	U2	Op-amp, LM324
R3	1kΩ		
R4	6.8kΩ	C1	0.1 µF
R5	5kΩ pot	C2	0.068 µF
R6	50Ω	C3	6.8 µF
R7	10kΩ		
R8, R9 & R10	100kΩ	Q1	NPN 2N3904
R11	75kΩ		

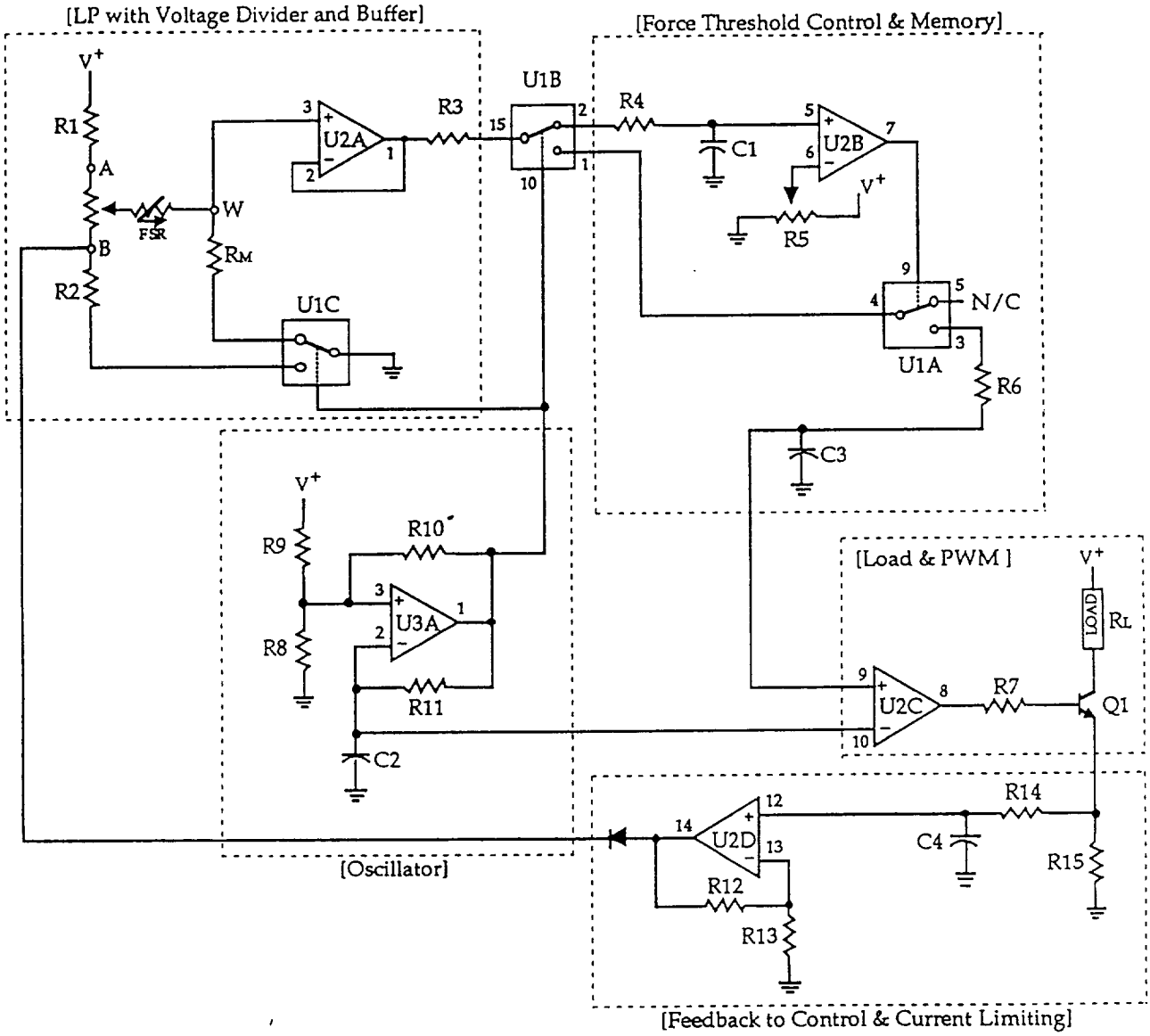


Figure 2.3, LP with PWM Output, Force Thresholding, Simple Memory and Current Limiting

Additional Components for Current Limiting:

R12	2MΩ	U3	LF353
R13	1kΩ	C4	3.3 µF
R14	47kΩ	Diodes	1N914
R15	1Ω		

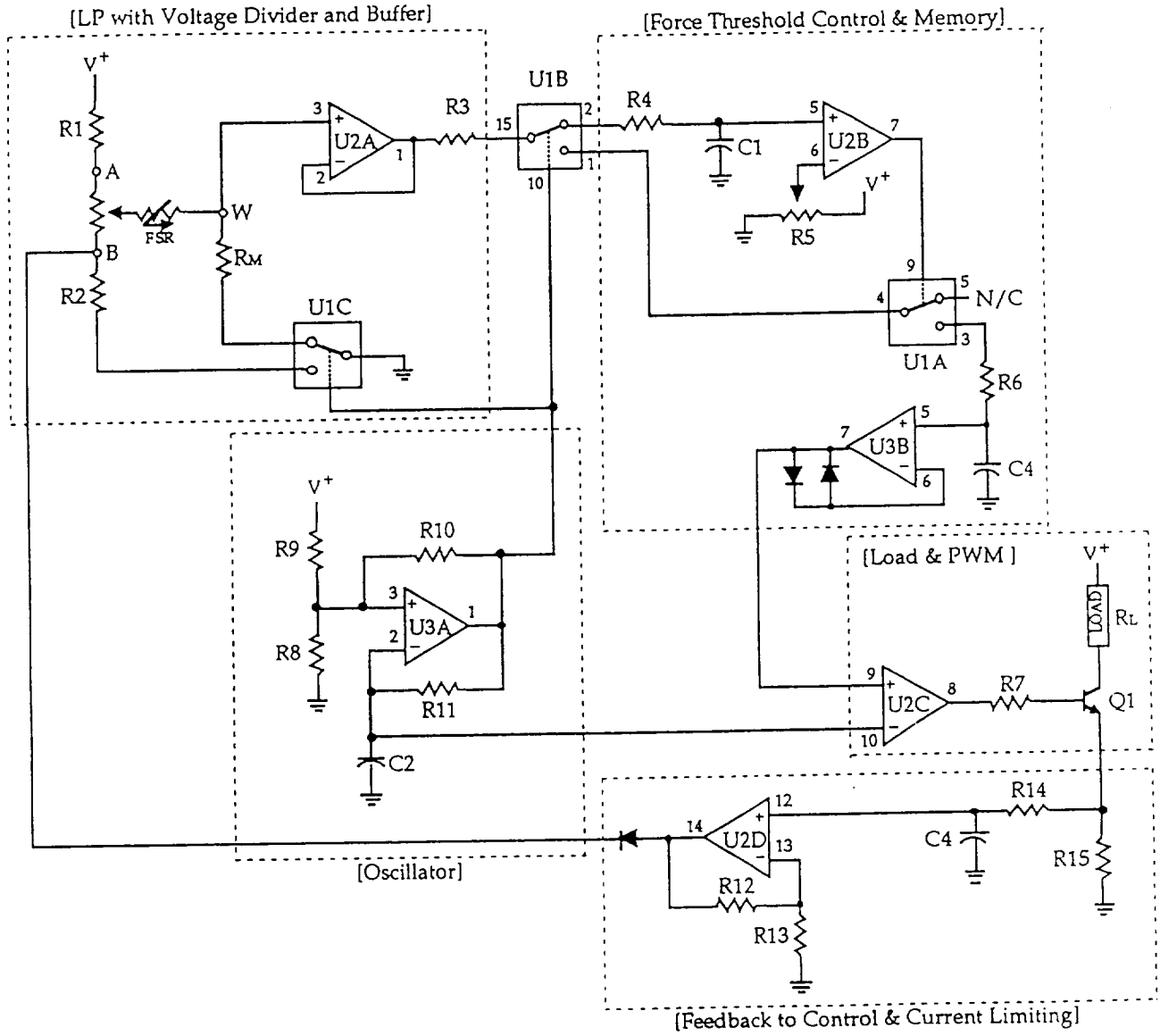
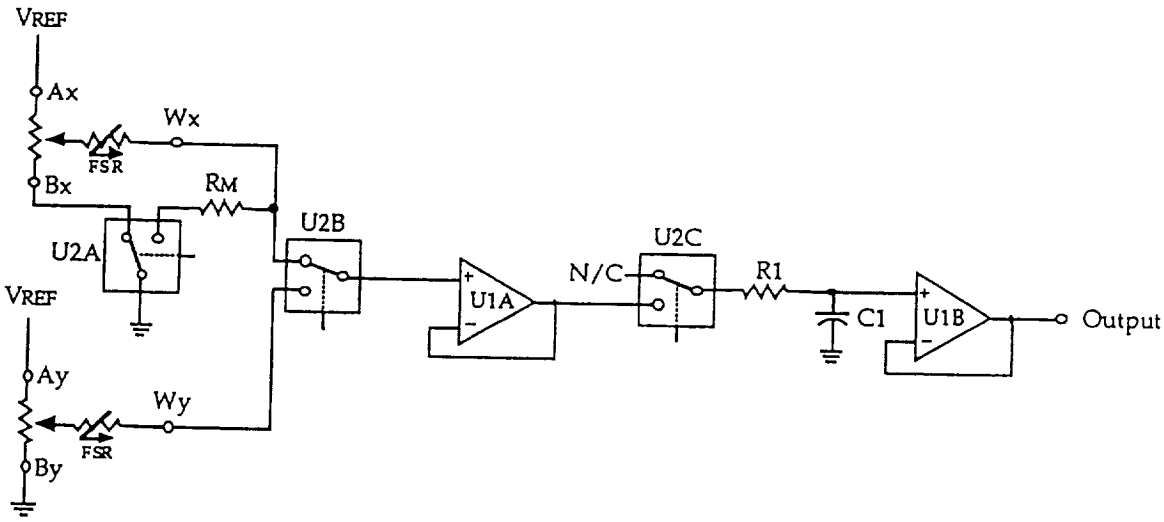


Figure 2.4, LP with PWM Output, Force Thresholding, Current Limiting and Enhanced Memory

Additional Components
for Enhanced Memory:

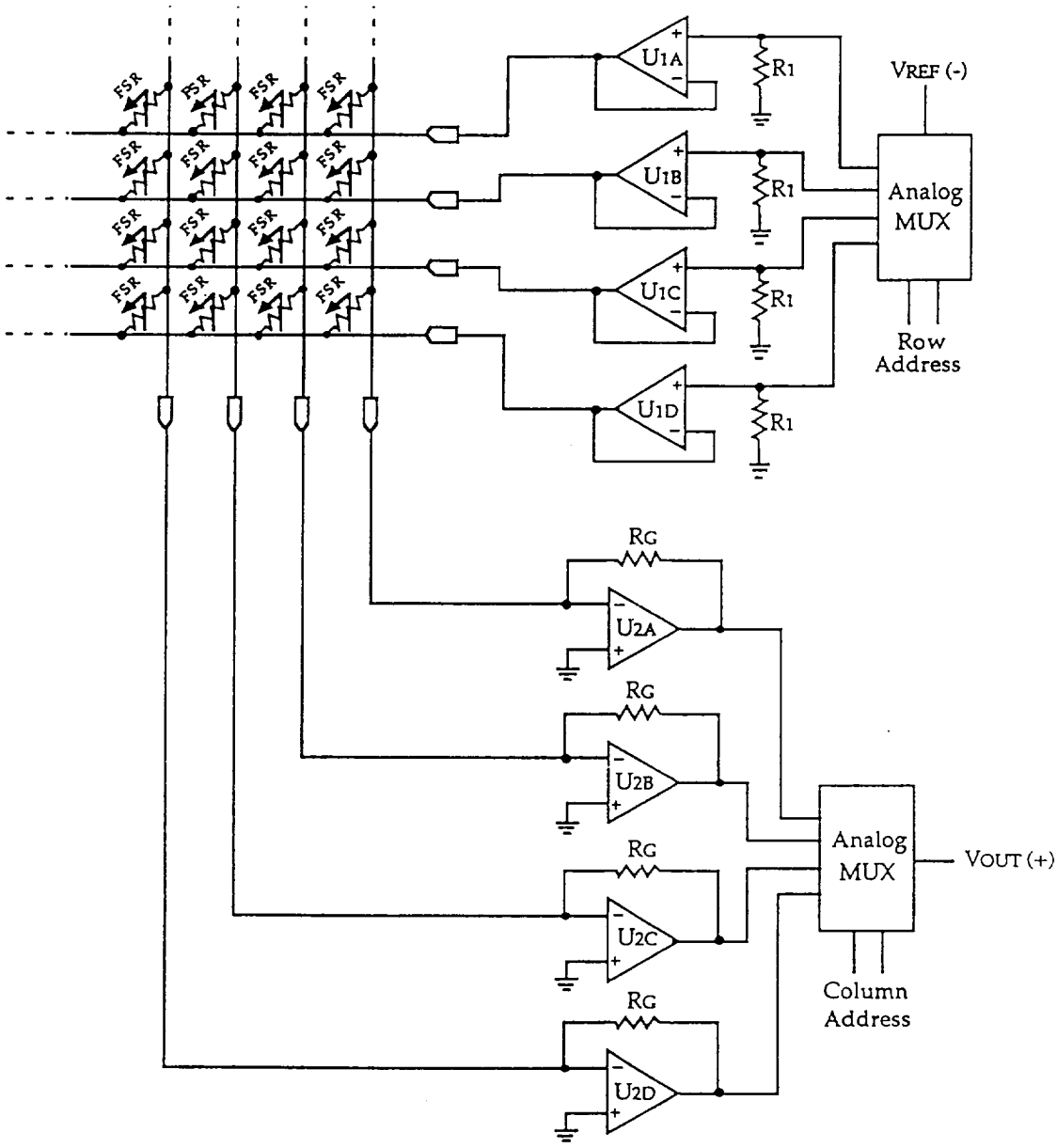
R15	50Ω
Diodes	1N914



Simple XYZ Interface:
This inexpensive and simple interface is excellent for XYZ applications. With properly controlled multiplexer switching, the output easily becomes time division multiplexed. This circuit, coupled with a microcontroller like the 68HC05, is ideal for touch pads that measure X and Y coordinates, as well as normal force (Z). RM is the measuring resistor that is chosen to optimize the output to the desired force sensitivity. U1A and U1B are unity gain buffers. The combination of R1, C1, U2C and U1B is a basic sample and hold buffer. The multiplexer, U2, can be driven by a microcontroller or by three separate oscillator lines. The MUX segment U2A switches between the X positional measurement and the force (Z) measurement. The segment U2B toggles between the X-Z and the Y measurements. Finally, U2C toggles between sampling and holding, to prevent any leakage into the output of U1A. This interface does not account for the positional corruption of the force measurement (see TechNote 2-1). This error can be handled by the controlling software if greater force accuracy is required.

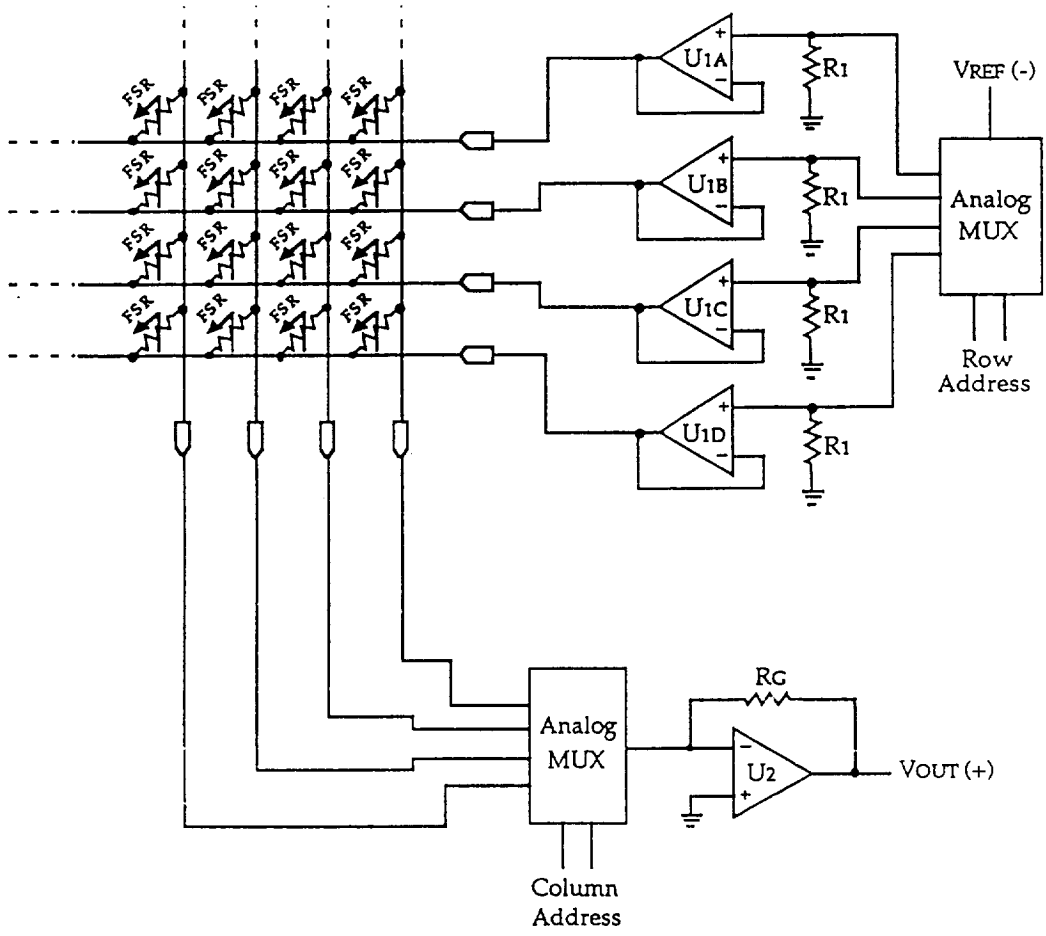
Suggested Components:

U1	Op-amp, LM358
U2	Multiplexer, CD4053
RM	100kΩ
R1	10kΩ
C1	0.001 µF



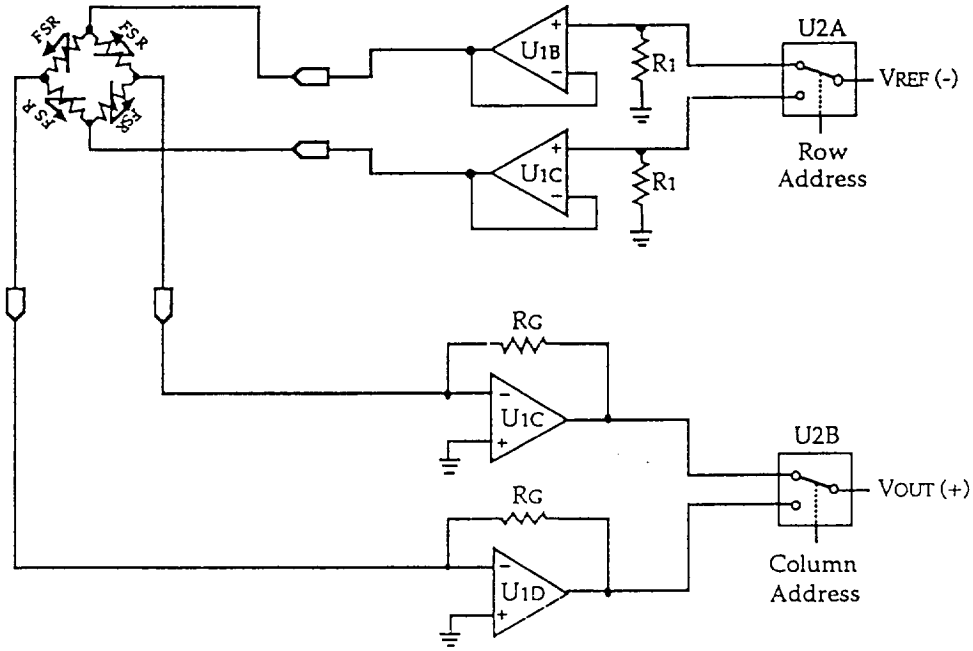
Cross Point Matrix Array Measuring Interface:
This circuit is the basis of the interface necessary to access a complex array that is set up in a row-column matrix. When implemented correctly, this interface eliminates all possible sneak paths (current paths that give incorrect readings) that may occur. For a detailed analysis of the current-to-voltage converters (U2A-D) that are used at the outputs, see TechNote 1-7.

Suggested Components:	
U1 (A-D)	LM324, TL074
U2 (A-D)	LF353, LF444
R1	100kΩ



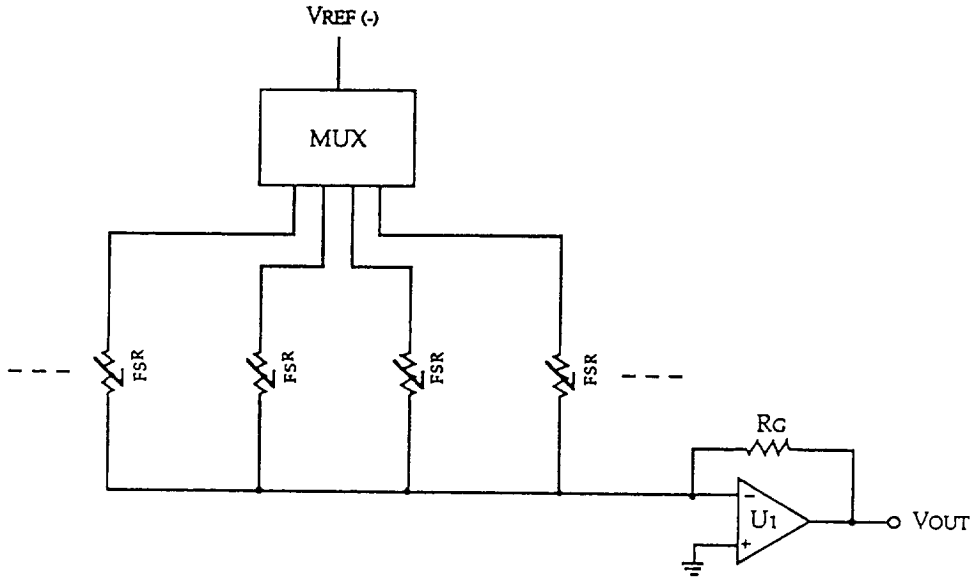
Simplified Cross Point Matrix Array Measuring Interface:
This circuit is similar to the interface illustrated in TechNote 4-1. The simplification of multiplexing to a single current-to-voltage converter decreases the component count, but increases the line settling time. For a detailed analysis of the current-to-voltage converter (U2) that is used at the output, see TechNote 1-7.

Suggested Components:	
U1 (A-D)	LM324, TL074
U2	LF353, LF444, LF411, LF441
R1	100kΩ



4 Zone Bridge Array Measuring Interface:
This interface is a derivative of the circuit described in TechNote 4-1. It can be useful in applications where a 4 zone array is desired, like joysticks and cursor controllers. For a detailed analysis of the current-to-voltage converters (U1C-D) that are used at the outputs, see TechNote 1-7.

Suggested Components:	
U1(A-D)	LM324
U2(A-B)	CD4053
R1	100kΩ



Common Bus Array Current-to-Voltage Converter:
For arrays of FSRs that have a single common bus for all of the active areas, this circuit is useful. Based on the current-to-voltage converter, this interface allows the time multiplexed sampling of any number of FSRs in a common bus array.

The performance of the current-to-voltage converter (U1) is described in TechNotes 1-7 and 1-8.

FSR™ Integration Guide & Evaluation Parts Catalog

Interlink has designed this document to help our customers integrate Force Sensing Resistor™ (FSR™) devices. It is intended to be used by engineers during the early phases of new product design. In it you will find:

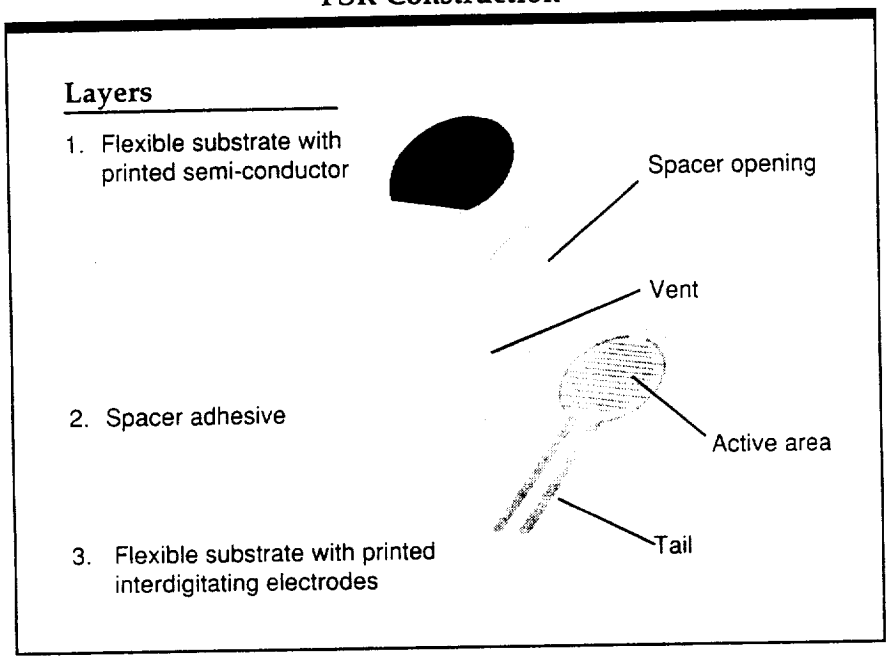
Interlink Electronics is manufactures custom FSR devices that meet the needs of specific customer applications. FSR devices can be produced in almost any shape, size, and geometry. Additionally, the FSR can be integrated with other flexible film and flat panel technologies, such as electroluminescent and fiber-optic backlighting, flexible LCDs, and flat or embossed graphic overlays. To discuss a custom design or to obtain a quote, contact your local Interlink Sales Representative or call Interlink Electronics at (805) 484-8855 and ask for Sales Support.

The FSR

An Overview of FSR Technology

The Force Sensing Resistor™ (FSR™) is a polymer thick film (PTF) device which exhibits a decrease in resistance with an increase in the force applied to the active surface. Its force sensitivity is optimized for use in human touch control of electronic devices. The FSR is not a load cell or strain gauge, though it has similar properties. The FSR is not suitable for precision measurements.

FSR Construction



Force vs. Resistance

The FSR force vs. resistance characteristic shown in Figure 1 provides an overview of the FSR's typical response behavior. For interpretational convenience, the force vs. resistance data are plotted on a log/log format. These data are representative of our typical devices, with this particular force-resistance characteristic being the response of evaluation part # 302 (0.5" [12.7 mm] diameter circular active area). A stainless steel actuator with a 0.4" [10.0 mm] diameter hemispherical tip of 60 durometer polyurethane rubber was used to actuate the FSR. In general, the FSR's response approximately follows an inverse power-law characteristic (roughly 1/R).

Referring to Figure 1, at the low force end of the force-resistance characteristic, a switch-like response is evident. This turn-on threshold, or "break force", that swings the resistance from greater than 1 MΩ to about 100 kΩ (the beginning of the dynamic range that follows a power-law) is determined by the substrate material, overlay thickness and flexibility, and spacer-adhesive thickness (the gap between the facing conductive elements). Break force increases with increasing substrate and overlay rigidity, and spacer-adhesive thickness. Eliminating the adhesive, or keeping it well away from the area where the force is being applied (such as the center of a large FSR device), will give the FSR a lower rest resistance (e.g. stand-off resistance). Any preloading of a FSR will also yield the same result.

Force vs. Resistance

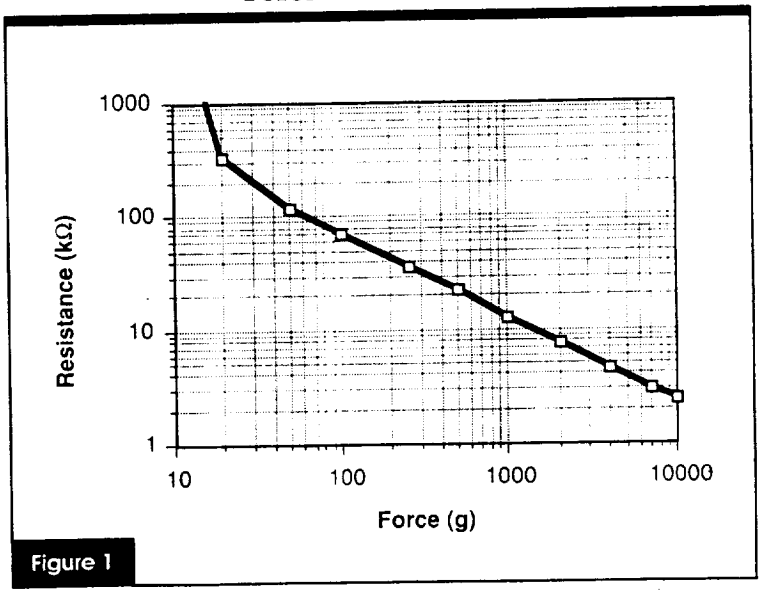


Figure 1

At the high force end of the dynamic range, the response deviates from the power-law behavior, and eventually saturates to a point where increases in force yield little or no decrease in resistance. For the FSR device under the test conditions of Figure 1, this saturation force is beyond 10 kg. The saturation point is more a function of pressure than force. The saturation pressure of the typical FSR is on the order of 100 to 200 psi. For the data shown in Figures 1, 2 and 3, the actual measured pressure range is 0 to 175 psi (0 to 22 lbs applied over 0.125 in²). Forces higher than the saturation force can be measured by spreading the force over a greater area; the overall pressure is then kept below the saturation pressure of the FSR, and dynamic response is maintained. However, the converse of this effect is also true, smaller actuators will saturate the FSR earlier in the dynamic range, since the saturation pressure is reached at a lower force.

Force vs. Conductance (0-10 Kg)

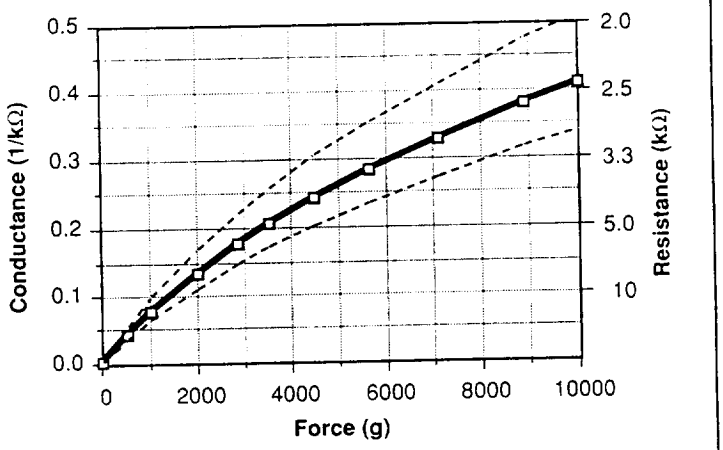


Figure 2

Force vs. Conductance (0-1 Kg) Low Force Range

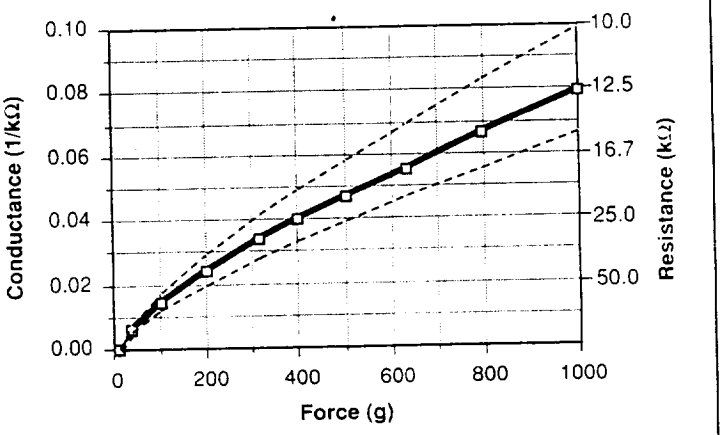


Figure 3

Force vs. Conductance

In Figure 2, the force is plotted vs. conductance (the inverse of resistance: 1/R). This format allows interpretation on a linear scale. For reference, the corresponding resistance values are also included on the right vertical axis. A simple circuit called a current-to-voltage converter (see *TechNotes - Suggested Interfaces*) gives a voltage output directly proportional to FSR conductance and can be useful where response linearity is desired. Figure 2 also includes a typical part-to-part repeatability envelope. This error band determines the maximum accuracy of any general force measurement. The spread or width of the band is strongly dependent on the repeatability of any actuating and measuring system, as well as the repeatability tolerance held by Interlink during FSR production. Typically, the part-to-part repeatability tolerance held during manufacturing ranges from ± 15% to ± 25% of an established nominal resistance.

Figure 3 highlights the 0-1 kg (0-2.2 lbs) range of the force-conductance characteristic. As in Figure 2, the corresponding resistance values are included for reference. This range is common to human interface applications. Since the conductance response in this range is fairly linear, the force resolution will be uniform and data interpretation simplified. The typical part-to-part error band is also shown for this touch range. In most human touch control applications this error is insignificant, since human touch is fairly inaccurate. Human factors studies have shown that repeatability errors of less than ± 50% are difficult to discern by touch alone.

FSR Integration Notes

A Step-by-Step Guide to Optimal Use

For best results, follow these seven steps when beginning any new product design, proof-of-concept, technology evaluation, or first prototype implementation:

1. **Start with Reasonable Expectations (Know Your Sensor).**
   The FSR is not a strain gauge, load cell or pressure transducer. While it can be used for dynamic measurement, only qualitative results are generally obtainable. Force accuracy ranges from approximately ± 5% to ± 25% depending on the consistency of the measurement and actuation system, the repeatability tolerance held in manufacturing, and the use of part calibration. For force and position sensing Linear Potentiometers (LPs) and XYZ Touch Pads, positional accuracy is excellent, generally better than ± 1% of full length.

   Accuracy should not be confused with resolution. The force resolution of FSR devices is better than ± 0.5% of full use force. The positional resolution of LPs and XYZs ranges from 0.003" to 0.020", depending on the nature of the actuator used.

2. **Choose the Sensor that Best Fits the Geometry of Your Application.**
   Usually sensor size and shape are the limiting parameters in FSR integration, so any evaluation part should be chosen to fit the desired mechanical actuation system. In general, standard FSR products have a common semiconductor make-up, and only by varying actuation methods (e.g. overlays and actuator areas) or electrical interfaces can different response characteristics be achieved.

3. **Set-up a Repeatable and Reproducible Mechanical Actuation System.**
   When designing the actuation mechanics, follow these guidelines to achieve the best force repeatability:

   - **Provide a consistent force distribution.** FSR response is very sensitive to the distribution of the applied force. In general, this precludes the use of dead weights, since exact duplication of the weight distribution is rarely repeatable cycle-to-cycle. A consistent weight (force) distribution is more difficult to achieve than merely obtaining a consistent total applied weight (force). As long as the distribution is the same cycle-to-cycle, then repeatability will be maintained. The use of a thin elastomer between the applied force and the FSR can help absorb error from inconsistent force distributions.

   - **Keep the actuator area, shape, and compliance constant.** Changes in these parameters significantly alter the response characteristic of a given sensor. Any test, mock-up or evaluation conditions should be closely matched to the final use conditions. The greater the cycle-to-cycle consistency of these parameters, the greater the device repeatability. In human interface applications where a finger is the mode of actuation, perfect control of these parameters is not generally possible. However, human force sensing is somewhat inaccurate; it is rarely sensitive enough to detect differences of less than ± 50%.

   - **Control actuator placement.** In cases where the actuator is to be smaller than the FSR active area, cycle-to-cycle consistency of actuator placement is necessary. A caution: the FSR's layers are held together by an adhesive that surrounds the electrically active areas. If force is applied over an area which includes the adhesive, the resulting response characteristic will be drastically altered. In an extreme case (e.g., a large, flat, hard actuator that bridges the bordering adhesive), the adhesive can prevent the FSR from being actuated.

   - **Keep actuation cycle time consistent.** Because of the time dependence of the FSR resistance to an applied force, it is important when characterizing the sensor system to assure that increasing loads (e.g. force ramps) are applied at consistent rates (cycle-to-cycle). Likewise,

static force measurements must take into account the settling time of the FSR. This time is dependent on the mechanics of actuation and the amount of force applied, and is usually on the order of seconds.

4. **Use the Optimal Electronic Interface.**
In most product designs, the critical characteristic is Force vs. Output Voltage, which is controlled by the choice of interface electronics. A variety of interface solutions are detailed in Interlink's *TechNotes - Suggested Interfaces*. Summarized here are some suggested circuits for common FSR applications.

- **For FSR Pressure or Force Switches,** use the simple interfaces detailed on pages 1-5 and 1-6 of the *TechNotes - Suggested Interfaces*.
- **For dynamic FSR measurements or Variable Controls,** a current-to-voltage converter (*TechNotes - Suggested Interfaces*, pages 1-7 and 1-8) is recommended. This circuit produces an output voltage that is inversely proportional to FSR resistance. Since the FSR resistance is roughly inversely proportional to applied force, the end result is a direct proportionality between force and voltage; in other words, this circuit gives roughly linear increases in output voltage for increases in applied force. This linearization of the response optimizes the resolution and simplifies data interpretation.
- **For position and force measurement with Linear Potentiometers,** use the circuits detailed on pages 2-1 and 2-2 of the *TechNotes - Suggested Interfaces*.
- **For X-Y position and force (Z) measurement with XYZ Touch Pads,** use the interface on page 3-1 of the *TechNotes - Suggested Interfaces*. Also refer to the notes on pages 2-1 and 2-2, since the XYZ Touch Pad is a combination of two orthogonal Linear Potentiometers.
- **For pressure profiling with a FSR Matrix Array,** the circuit on page 4-1 of the *TechNotes - Suggested Interfaces* illustrates the basics of the interface needed for an array in a row-column matrix.

5. **Develop a Nominal Voltage Curve and Error Spread.**
When a repeatable and reproducible system has been established, data from a group of FSR parts can be collected. Test several FSR parts in the system. Record the output voltage at various pre-selected force points throughout the range of interest. Once a family of curves is obtained, a nominal force vs. output voltage curve and the total force accuracy of the system can be determined.

6. **Use Part Calibration if Greater Accuracy is Required.**
For applications requiring the highest obtainable force accuracy, part calibration will be necessary. Two methods can be utilized: gain and offset trimming, and curve fitting.

- **Gain and offset trimming can be used as a simple method of calibration.** The reference voltage and feedback resistor of the current-to-voltage converter are adjusted for each FSR to pull their responses closer to the nominal curve.
- **Curve fitting is the most complete calibration method.** A parametric curve fit is done for the nominal curve of a set of FSR devices, and the resultant equation is stored for future use. Fit parameters are then established for each individual FSR (or sensing element in an array) in the set. These parameters, along with the measured sensor resistance (or voltage), are inserted into the equation to obtain the force reading. If needed, temperature compensation can also be included in the equation.

7. **Refine the System.**
Spurious results can normally be traced to sensor error or system error. If you have any questions, contact Interlink's Applications Engineers to discuss your system and final data.

FSR Usage Tips

The Do's and Don'ts

Do's

- **Do follow the seven steps of the FSR Integration Guide.**
- **Do, if possible, use a firm, flat and smooth mounting surface.**
- **Do be careful if applying FSR devices to curved surfaces.** Pre-loading of the device can occur as the two opposed layers are forced into contact by the bending tension. The device will still function, but the dynamic range may be reduced and resistance drift could occur. The degree of curvature over which an FSR can be bent is a function of the size of the active area. The smaller the active area, the less effect a given curvature will have on the FSR's response.
- **Do avoid air bubbles and contamination when laminating the FSR to any surface.** Use only thin, uniform adhesives, such as Scotch® brand double-sided laminating adhesives. Cover the entire surface of the sensor.
- **Do be careful of kinks or dents in active areas;** they can cause false triggering of the sensors.
- **Do protect the device from sharp objects** (e.g. pens on XYZ pads). Use an overlay, such as a polycarbonate film or an elastomer, to prevent gouging of the FSR.
- **Do use soft rubber or a spring as part of the actuator** in designs requiring some travel.

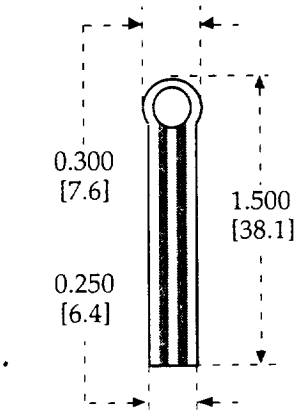
Don'ts

- **Do not kink or crease the tail of the FSR** device if you are bending it; this can cause breaks in the printed silver traces. The smallest suggested bend radius for the tails of evaluation parts is about 0.1" [2.5 mm]. In custom sensor designs, tails have been made that bend over radii of 0.03" [0.8 mm]. Also, be careful if bending the tail near the active area, this can cause stress on the active area and may result in pre-loading and false readings.
- **Do not block the vent.** FSR devices typically have an air vent that runs from the open active area down the length of the tail and out to the atmosphere. This vent assures pressure equilibrium with the environment, as well as allowing even loading and unloading of the device. Blocking this vent could cause the FSR to respond to any actuation in a non-repeatable manner. Also note that if the device is to be used in a pressure chamber, the vented end will need to be kept vented to the outside of the chamber; this allows for the measurement of the differential pressure.
- **Do not solder directly to the exposed silver traces.** With flexible substrates, the solder joint will not hold and the substrate can easily melt and distort during the soldering. Use Interlink's standard connection techniques, such as solderable tabs, housed female contacts, Z-axis conductive tapes, or ZIF (zero insertion force) style connectors.
- **Do not use cyanoacrylate adhesives** (e.g. Krazy Glue®) **and solder flux removing agents.** These degrade the substrate and can lead to cracking.
- **Do not apply excessive shear force;** this can cause delamination of the layers.
- **Do not exceed 1 mA of current per square centimeter of applied force** (actuator area); this can irreversibly damage the device.

Evaluation Parts

Descriptions and Dimensions

Part #300 (1/5" Circle)



Active Area 0.2" [5.0] diameter

Nominal Thickness 0.010" [0.25]

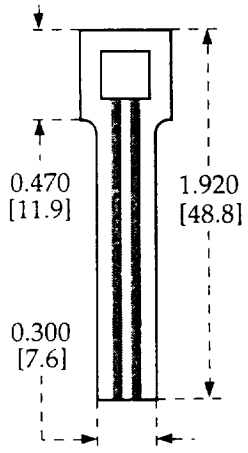
Material Build:

Semiconductive Layer
0.004" [0.10] Stabar (PES)

Spacer Adhesive
0.002" [0.05] Acrylic

Silver Layer
0.004" [0.10] Stabar (PES)

Part #301 (1/4" Square)



Active Area 0.25" [6.3] x 0.25" [6.3]

Nominal Thickness 0.014" [0.36]

Material Build:

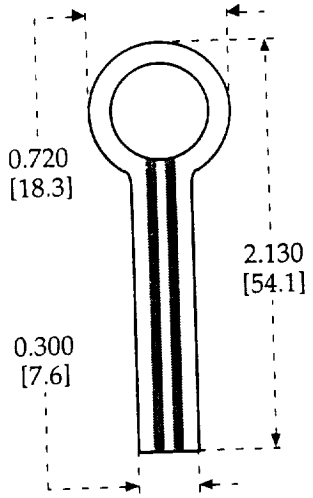
Semiconductive Layer
0.005" [0.13] Ultem (PEI)

Spacer Adhesive
0.004" [0.10] Acrylic/polyester

Silver Layer
0.005" [0.13] Ultem (PEI)

Dimensions in brackets: millimeters. Dimensional Tolerance: ± 0.015" [0.4]. Thickness Tolerance: ± 10%

Part #302 (1/2" Circle – Ultem)



Active Area	0.5" [12.7] Diameter
Nominal Thickness	0.014" [0.36]

Material Build:

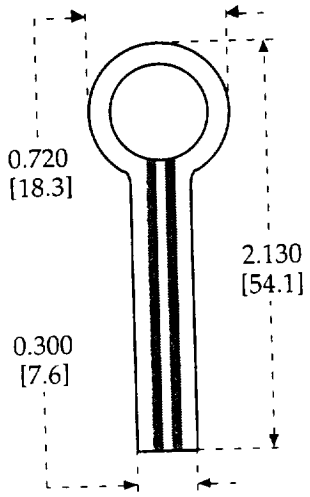
Semiconductive Layer
0.005" [0.13] Ultem (PEI)

Spacer Adhesive
0.004" [0.10] Acrylic/polyester

Silver Layer
0.005" [0.13] Ultem (PEI)

Note: Although #302 and #303 are the same design, #302 is made with Ultem, a base film that is amber in color, somewhat inflexible, and very environmentally durable; #303 is made with Stabar, a base film that is transparent, thinner, more flexible, less chemically resistant, and slightly more sensitive to low forces than the Ultem based version.

Part #303 (1/2" Circle – Stabar)



Active Area	0.5" [12.7] Diameter
Nominal Thickness	0.012" [0.30]

Material Build:

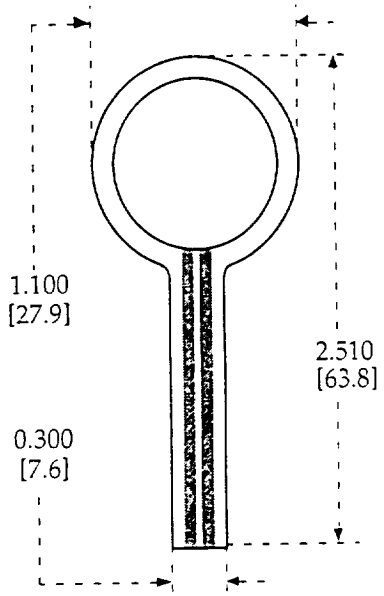
Semiconductive Layer
0.004" [0.10] Stabar (PES)

Spacer Adhesive
0.004" [0.10] Acrylic/polyester

Silver Layer
0.004" [0.10] Stabar (PES)

Dimensions in brackets: millimeters. Dimensional Tolerance: ± 0.015" [0.4]. Thickness Tolerance: ± 10%

Part #304 (7/8" Circle)



Active Area 0.875" [22.2] Diameter

Nominal Thickness 0.017" [0.43]

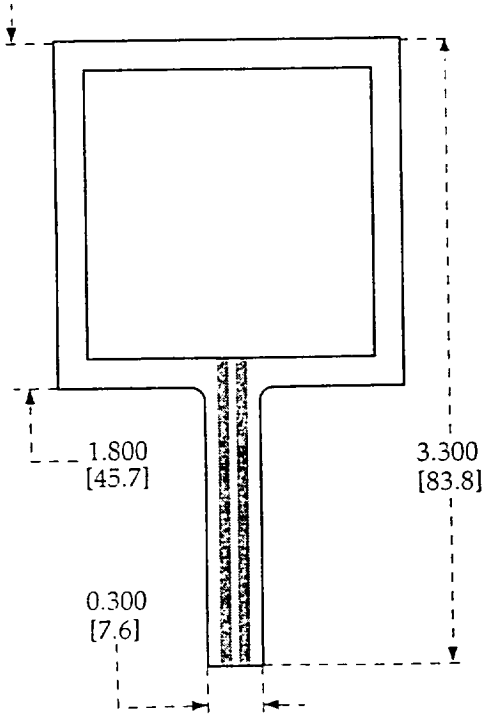
Material Build:

Semiconductive Layer
0.005" [0.13] Ultem (PEI)

Spacer Adhesive
0.007" [0.17] Acrylic/polyester

Silver Layer
0.005" [0.13] Ultem (PEI)

Part #306 (1-1/2" Square)



Active Area 1.5" [38.1] x 1.5" [38.1]

Nominal Thickness 0.017" [0.43]

Material Build:

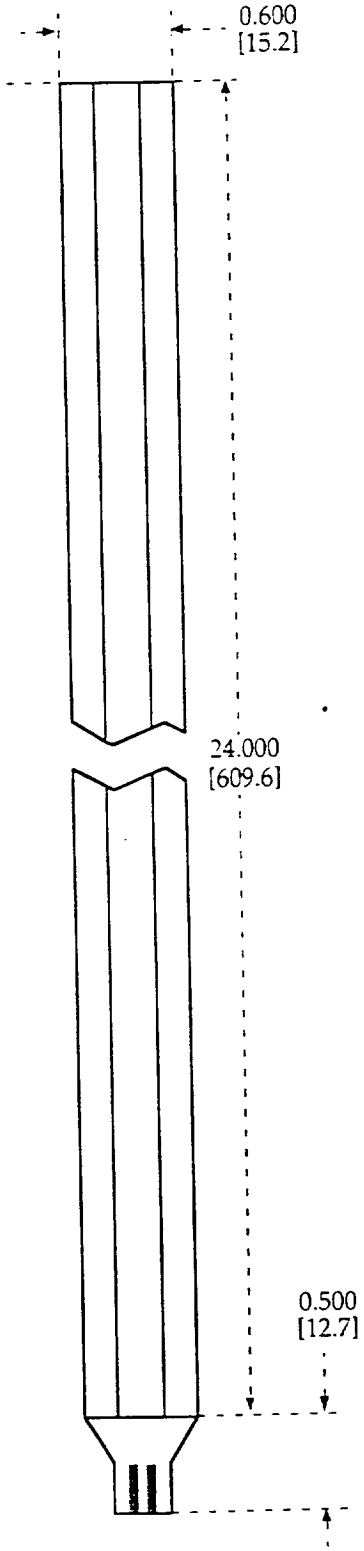
Semiconductive Layer
0.005" [0.13] Ultem (PEI)

Spacer Adhesive
0.007" [0.17] Acrylic/polyester

Silver Layer
0.005" [0.13] Ultem (PEI)

Dimensions in brackets: millimeters. Dimensional Tolerance: ± 0.015" [0.4]. Thickness Tolerance: ± 10%

Part #308 (24" Trimmable Strip)



Active Area	24" [609.6] x 0.25" [6.3]
Nominal Thickness	0.010" [0.25]

Material Build:

Semiconductive Layer
0.004" [0.10] Stabar (PES)

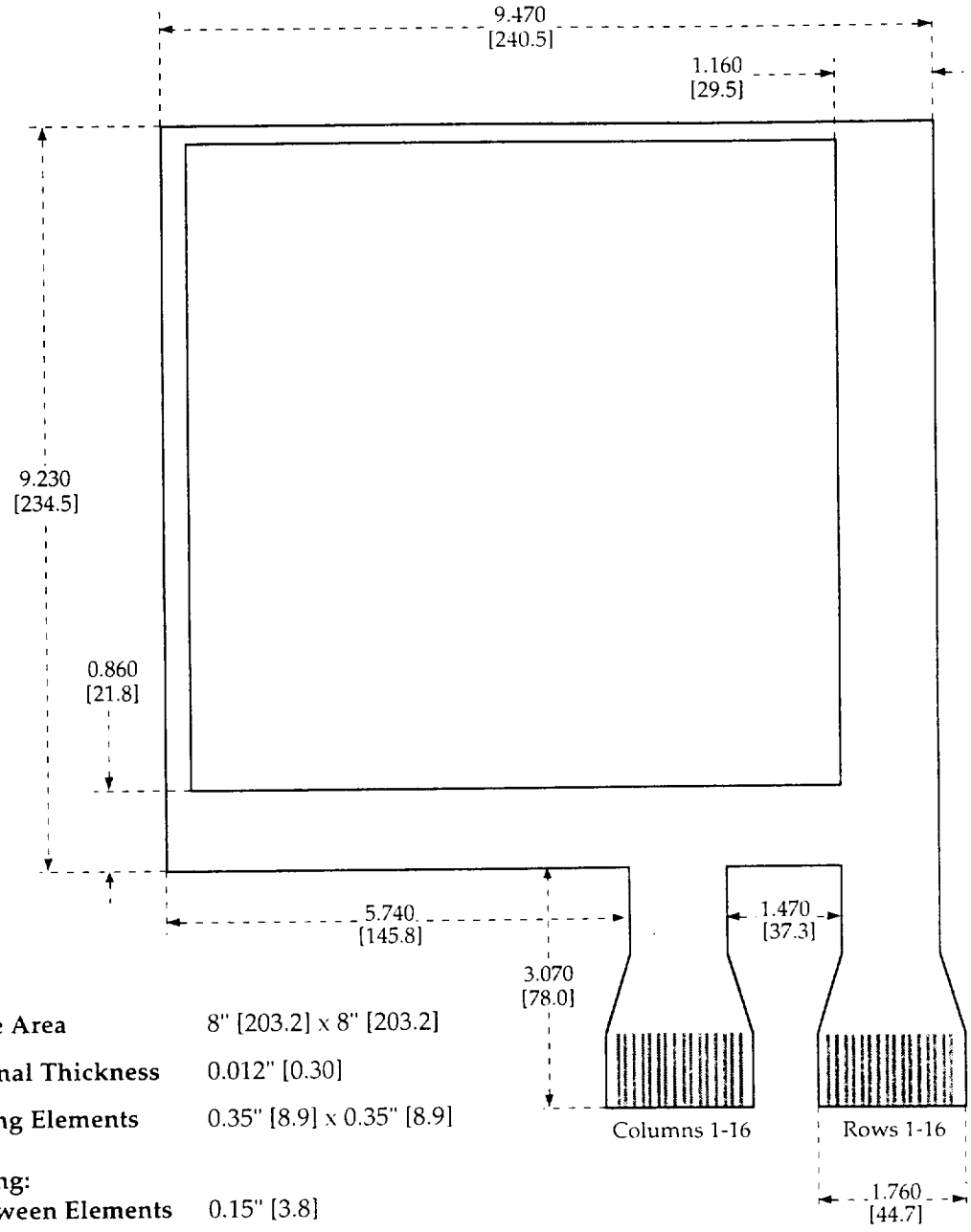
Spacer Adhesive
0.0035" [0.089] Acrylic

Silver Layer
0.004" [0.10] Stabar (PES)

Dimensions in brackets: millimeters. Dimensional Tolerance: ± 0.015" [0.4]. Thickness Tolerance: ± 10%

Part #350 (256 Zone Matrix Array)

16 sensor by 16 sensor addressable array



Active Area	8" [203.2] x 8" [203.2]
Nominal Thickness	0.012" [0.30]
Sensing Elements	0.35" [8.9] x 0.35" [8.9]
Spacing:	
Between Elements	0.15" [3.8]
Center-to-Center	0.5" [12.7]

Material Build:

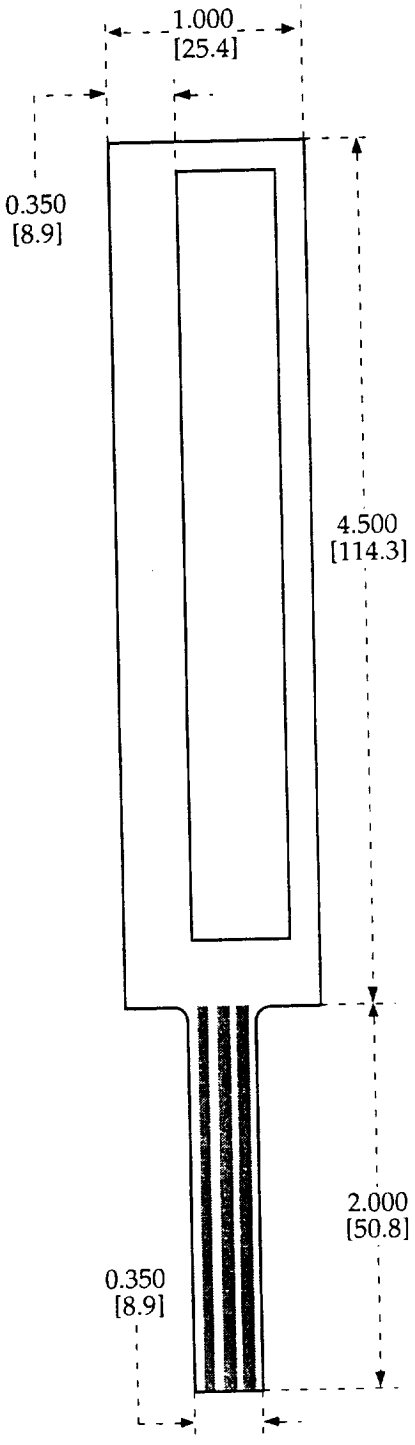
Semiconductive Layer
0.004" [0.10] Stabar (PES)

Spacer Adhesive
0.004" [0.10] Acrylic/Polyester

Silver Layer
0.004" [0.10] Stabar (PES)

Dimensions in brackets: millimeters. Dimensional Tolerance: ± 0.015" [0.4]. Thickness Tolerance: ± 10%

Part #360 (4" Linear Potentiometer)



Active Area 0.5" [12.7] x 4" [101.6]

Nominal Thickness 0.017" [0.43]

Material Build:

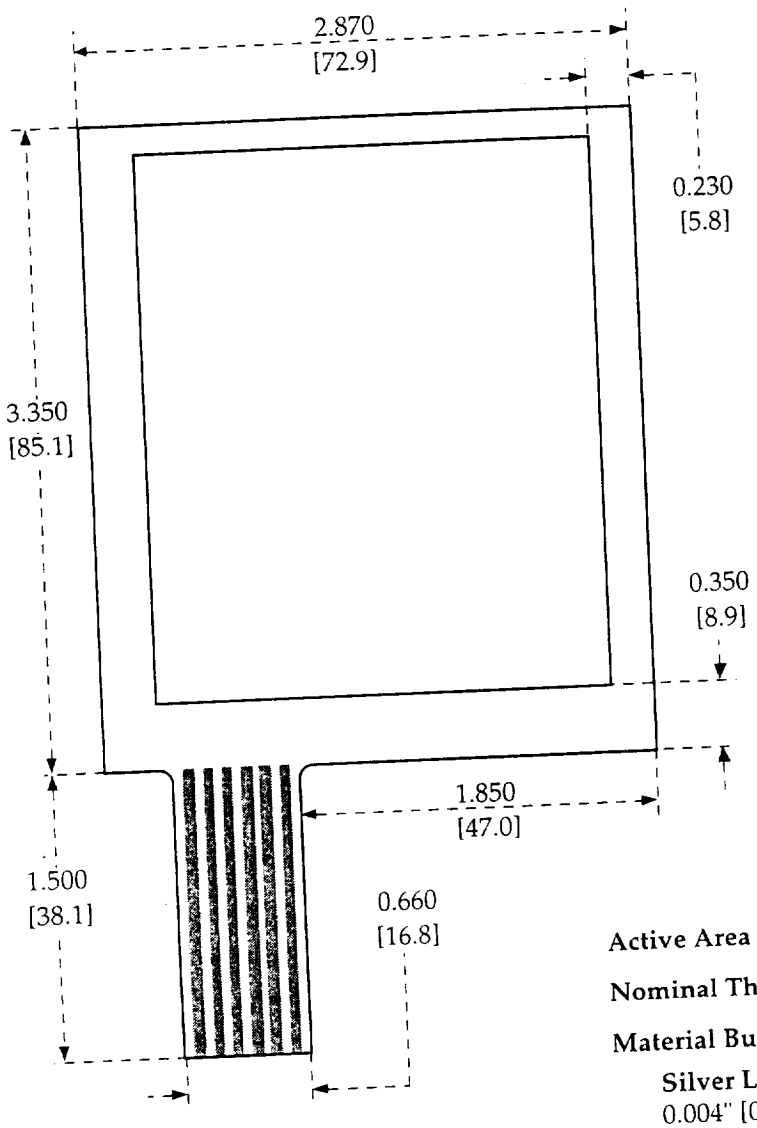
Semiconductive Layer
0.005" [0.13] Ultem (PEI)

Spacer Adhesive
0.007" [0.17] Acrylic/polyester

Silver Layer
0.005" [0.13] Ultem (PEI)

Dimensions in brackets: millimeters. Dimensional Tolerance: ± 0.015" [0.4]. Thickness Tolerance: ± 10%

Part #370 (XYZ Pad)



Active Area	2.3" [58.4] x 2.8" [71.1]
Nominal Thickness	0.016" [0.40]

Material Build:

Silver Layer - X
0.004" [0.10] Melinex (PET)

Spacer Adhesive
0.002" [0.05]] Acrylic

Semiconductive Layer
0.004" [0.10] Stabar (PES)

Spacer Adhesive
0.002" [0.05] Acrylic

Silver Layer-Y
0.004" [0.10] Melinex (PET)

Dimensions in brackets: millimeters. Dimensional Tolerance: ± 0.015" [0.4]. Thickness Tolerance: ± 10%

General FSR Characteristics

These are typical parameters. The FSR is a custom device and can be made for use outside these characteristics. Consult Sales Engineering with your specific requirements.

Simple FSR Devices and Arrays

Parameter	Value	Notes
Size Range	Max = 20" x 24" (51 x 61 cm) Min = 0.2" x 0.2" (0.5 x 0.5 cm)	Any shape
Device Thickness	0.008" to 0.050" (0.20 to 1.25 mm)	Dependent on materials
Force Sensitivity Range	< 100 g to > 10 kg	Dependent on mechanics
Pressure Sensitivity Range	< 1.5 psi to > 150 psi (< 0.1 kg/cm² to > 10 kg/cm²)	Dependent on mechanics
Part-to-Part Force Repeatability	± 15% to ± 25% of established nominal resistance	With a repeatable actuation system
Single Part Force Repeatability	± 2% to ± 5% of established nominal resistance	With a repeatable actuation system
Force Resolution	Better than 0.5% full scale	
Break Force (Turn-on Force)	20 g to 100 g (0.7 oz to 3.5 oz)	Dependent on mechanics and FSR build
Stand-Off Resistance	> 1 MΩ	Unloaded, unbent
Switch Characteristic	Essentially zero travel	
Device Rise Time	1–2 msec (mechanical)	
Lifetime	> 10 million actuations	
Temperature Range	-30°C to +170°C	Dependent on materials
Maximum Current	1 mA/cm² of applied force	
Sensitivity to Noise/Vibration	Not significantly affected	
EMI/ESD	Passive device—not damaged by EMI or ESD	
Lead Attachment	Standard flex circuit techniques	

For Linear Pots and XYZ Touch Pads

Parameter	Value	Notes
Positional Resolution	0.003" to 0.02" (0.075 to 0.5 mm)	Dependent on actuator size
Positional Accuracy	better than ± 1% of full length	

FSR terminology is defined on pages 15 and 16 of this guide.

Glossary of Terms

Active Area The area of an FSR™ device that responds to normal force with a decrease in resistance.

Actuator The object which contacts the sensor surface and applies force to the FSR.

Applied Force The force applied by the actuator on the active area of the sensor.

Array Any grouping or matrix of FSR sensors which can be individually actuated and measured.

Break Force The minimum force required, with a specific actuator size, to cause the onset of the FSR response.

Cross-talk Measurement noise or inaccuracies of a sensor as a result of the actuation of another sensor on the same substrate. See also False Triggering.

Drift The change in resistance with time under a constant (static) load. Also called Resistance Drift.

Durometer The measure of the hardness of rubber.

EMI Electromagnetic Interference.

ESD Electrostatic Discharge.

False Triggering The unwanted actuation of a FSR device from unexpected stimuli; e.g., bending or cross-talk.

Fixed Resistor The printed resistor on Linear Potentiometers and XYZ pads that is used to measure position.

Footprint Surface area and force distribution of the actuator in contact with the sensor surface.

Force Resolution The smallest measurable difference in force.

FPSR™ Force and Position Sensing Resistor™. Force sensing Linear Potentiometers and XYZ pads.

FSR™ Force Sensing Resistor. A polymer thick film device which exhibits a decrease in resistance with an increase in force applied normal to the device surface.

Graphic Overlay A printed substrate that covers the FSR. Usually used for esthetics and protection.

Housed Female Connector A stitched on AMP connector with a receptacle (female) ending. A black plastic housing protects the contacts. Suitable for removable ribbon cable and header pin attachment.

Hysteresis In a dynamic measurement, the difference between instantaneous force measurements at a given force for an increasing load versus a decreasing load.

Interdigitating Electrodes The conductor grid. An interweaving pattern of linearly offset conductor traces used to achieve electrical contact. This grid is shunted by the semiconductor layer to give the FSR response.

Lead Out or Busing System The method of electrically accessing each individual sensor.

Lexan® Polycarbonate. A substrate used for graphic overlays and labels. Available in a variety of surface textures.

Linear Potentiometer (LP) A FSR device which simultaneously measures force and position of an actuator along a line.

Melinex® Polyester (PET), also Mylar™. A hazy semi-transparent substrate with low temperature and chemical resistance, but with excellent flexibility and low cost.

Part or Device The FSR. Consists of the FSR semiconductive material, conductor, adhesives, graphics or overlays, and connectors.

Pin Out The description of a FSR's electrical access at the connector pads (tail).

Positional Accuracy For LPs and XYZs, the ability to measure the absolute position of a given force.

Positional Resolution Also for LPs and XYZs, the smallest measurable difference in position.

Repeatability The ability to repeat, within a tolerance, a previous response characteristic.

Response Characteristic The relationship of force or pressure vs. resistance.

Saturation Pressure The pressure level beyond which the FSR response characteristic deviates from its inverse power law characteristic. Past the saturation pressure, increases in force yield little or no decrease in resistance.

Sensor Each area of the FSR device that is independently force sensitive (as in an array).

Solder-tabs Stitched on AMP connectors with tab endings. Suitable for direct PC board connection or for soldering to wires.

Space and Trace The widths of the gaps and fingers of the conductive grid; also called pitch.

Spacer Adhesive The adhesive used to laminate FSR devices together. Dictates stand-off.

Stabar® Polyethersulfone (PES). A transparent substrate with moderate temperature and chemical resistance and good flexibility.

Stand-off The gap or distance between the opposed polymer film layers when the sensor is unloaded and unbent.

Stand-off Resistance The FSR resistance when the device is unloaded and unbent.

Substrate Any base material on which the FSR semi-conductive or metallic polymers are printed. (For example, polyetherimide, polyethersulfone and polyester films).

Tail The region where the lead out or busing system terminates. Generally, the tail ends in a connector.

Ultem® Polyetherimide (PEI). A yellow, semi-transparent substrate with excellent temperature and chemical resistance and limited flexibility.

XYZ Pad A FSR device which simultaneously measures force and position of an actuator in a plane.

APPENDIX II.G

Motor Driver Data Sheets

SGS-THOMSON MICROELECTRONICS

L6201
L6202-L6203

DMOS FULL BRIDGE DRIVER

PRELIMINARY DATA

- SUPPLY VOLTAGE UP TO 48 V
- 5A MAX PEAK CURRENT (2A max. for L6201)
- TOTAL RMS CURRENT UP TO L6201: 2A; L6202: 1.5A; L6203: 4A
- $R_{DS(ON)}$ 0.3 Ω (typical value at 25 °C)
- CROSS CONDUCTION PROTECTION
- TTL COMPATIBLE DRIVE
- OPERATING FREQUENCY UP TO 100 KHz
- THERMAL SHUTDOWN
- INTERNAL LOGIC SUPPLY
- HIGH EFFICIENCY

MULTIPOWER BCD TECHNOLOGY

Powerdip 12+3+3

SO20 (12+4+4)

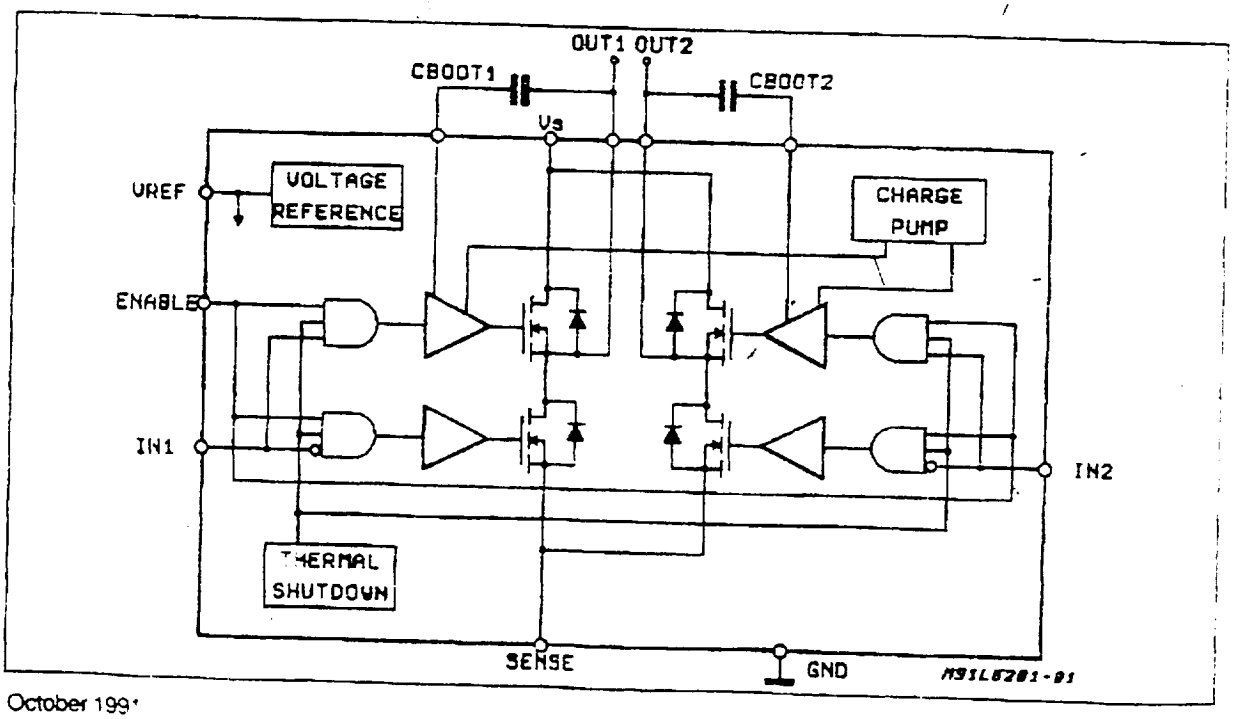
Multiwatt11

ORDERING NUMBERS:
L6201 (SO)
L6202 (Powerdip)
L6203 (Multiwatt)

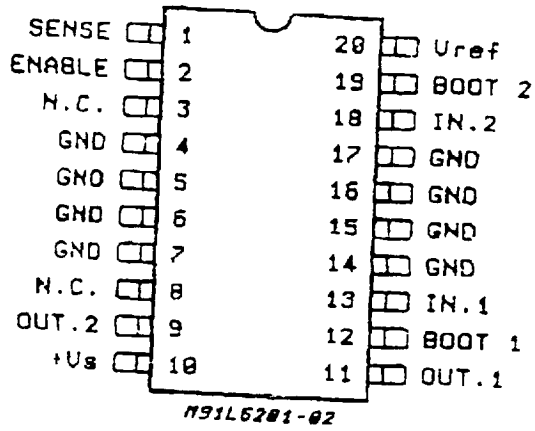
DESCRIPTION

The I.C. is a full bridge driver for motor control applications realized in Multipower-BCD technology which combines isolated DMOS power transistors with CMOS and Bipolar circuits on the same chip. By using mixed technology it has been possible to optimize the logic circuitry and the power stage to achieve the best possible performance. The DMOS output transistors can operate at supply voltages up to 48 V and efficiently at high switching speeds. All the logic inputs are TTL, CMOS and µC compatible. Each channel (half-bridge) of the device is controlled by a separate logic input, while a common enable controls both channels. The I.C. is mounted in three different packages.

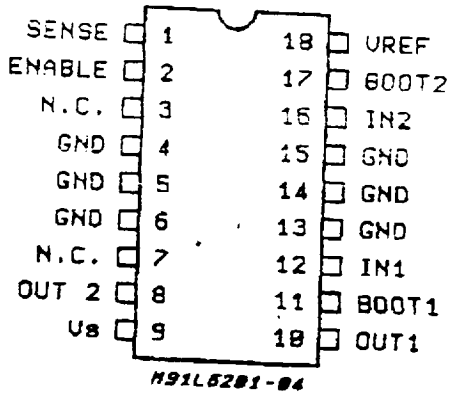
BLOCK DIAGRAM



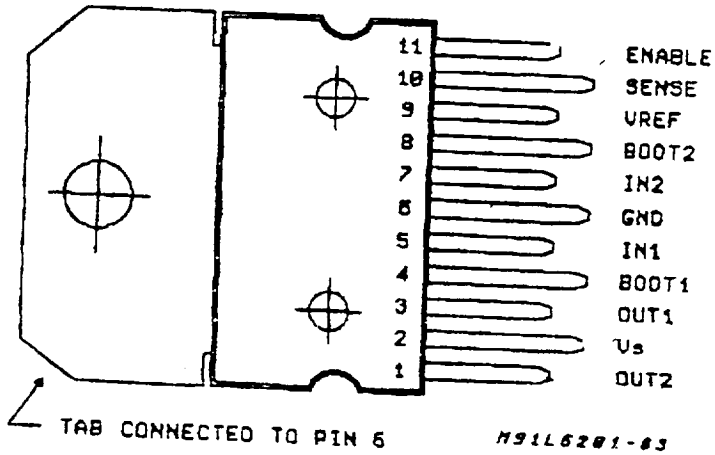
PIN CONNECTIONS (Top view)



SO20



POWERDIP



MULTIWATT11

PINS FUNCTIONS

Device			Name	Function
L6201	L6202	L6203		
1	1	10	SENSE	A resistor R_{sense} connected to this pin provides feedback for motor current control.
2	2	11	ENABLE	When a logic high is present on this pin the DMOS POWER transistors are enabled to be selectively driven by IN1 and IN2.
3	3		N.C.	Not Connected
4,5	4	6	GND	Common Ground Terminal
–	5		GND	Common Ground Terminal
6,7	6		GND	Common Ground Terminal
8	7		N.C.	Not Connected
9	8	1	OUT2	Ouput of 2nd Half Bridge
10	9	2	V_s	Supply Voltage
11	10	3	OUT1	Output of first Half Bridge
12	11	4	BOOT1	A boostrap capacitor connected to this pin ensures efficient driving of the upper POWER DMOS transistor.
13	12	5	IN1	Digital Input from the Motor Controller
14,15	13	6	GND	Common Ground Terminal
–	14		GND	Common Ground Terminal
16,17	15		GND	Common Ground Terminal
18	16	7	IN2	Digital Input from the Motor Controller
19	17	8	BOOT2	A boostrap capacitor connected to this pin ensures efficient driving of the upper POWER DMOS transistor.
20	18	9	V_{ref}	Internal voltage reference. A capacitor from this pin to GND is recommended. The internal Ref. Voltage can source out a current of 2mA max.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Power Supply	52	V
V_{OO}	Differential Output Voltage (between Out1 and Out2)	60	V
V_{IN}, V_{EN}	Input or Enable Voltage	– 0.3 to + 7	V
I_o	Pulsed Output Current – Non Repetitive (< 1 ms) for L6202/03 (Note 1) for L6201 for L6202/03 DC Output Current for L6201 (Note 1)	5 5 10 1	A A A A
V_{sense}	Sensing Voltage	– 1 to + 4	V
V_b	Boostrap Peak Voltage	60	V
P_{tot}	Total Power Dissipation: T_{pins} = 90°C for L6201 for L6202 T_{case} = 90°C for L6203 T_{amb} = 70°C for L6201 (Note 2) for L6202 (Note 2) for L6203 (Note 2)	 4 5 20 0.9 1.3 2.3	 W W W W W W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to + 150	°C

ote 1: Pulse width limited only by junction temperature and transient thermal impedance (see thermal characteristics)
ote 2: Mounted on board with minimized dissipating copper area.

THERMAL DATA

Symbol	Parameter		Value			Unit
			L6201	L6202	L6203	
$Rth_{j\text{-pins}}$	Thermal Resistance Junction-pins	max	15	12	–	°C/W
$Rth_{j\text{-case}}$	Thermal Resistance Junction Case	max.	–	–	3	
$Rth_{j\text{-amb}}$	Thermal Resistance Junction-ambient	max.	85	60	35	

ELECTRICAL CHARACTERISTICS (Refer to the Test Circuits; T_j = 25°C, Vs = 42V, V_{sens} = 0, unle otherwise specified).

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
V_s	Supply Voltage			12	36	48	V
V_{ref}	Reference Voltage	I_{REF} = 2mA			13.5		V
I_{REF}	Output Current					2	mA
I_s	Quiescent Supply Current	EN = H V_{IN} = L EN = H V_{IN} = H EN = L (Fig. 1,2,3)	I_L = 0		10 10 8	15 15 15	mA mA mA
f_c	Commutation Frequency (*)				30	100	KHz
T_j	Thermal Shutdown				150		°C
T_d	Dead Time Protection				100		ns

TRANSISTORS

OFF							
I_{DSS}	Leakage Current	Fig. 11 V_s = 52 V				1	mA
ON							
R_{DS}	On Resistance	Fig. 4,5			0.3	0.55	Ω
$V_{DS(ON)}$	Drain Source Voltage	Fig. 9 I_{DS} = 1A I_{DS} = 1.2A I_{DS} = 3A	**L6201** **L6202** **L6203**		0.3 0.36 0.9		V V V
V_{sens}	Sensing Voltage			–1		4	V

SOURCE DRAIN DIODE

V_{sd}	Forward ON Voltage	Fig. 6a and b I_{SD} = 1A **L6201** EN = L I_{SD} = 1.2A **L6202** EN = L I_{SD} = 3A **L6203** EN = L			0.9 (**) 0.9 (**) 1.35(**)		V V V
t_{rr}	Reverse Recovery Time	$\frac{dif}{dt}$ = 25 A/µs I_F = 1A I_F = 1.2A I_F = 3A	L6201 L6202 L6203		300		ns
t_{fr}	Forward Recovery Time				200		ns

LOGIC LEVELS

$V_{IN\,L}$, $V_{EN\,L}$	Input Low Voltage		–0.3		0.8	V
$V_{IN\,H}$, $V_{EN\,H}$	Input High Voltage		2		7	V
$I_{IN\,L}$, $I_{EN\,L}$	Input Low Current	V_{IN}, V_{EN} = L			–10	µA
$I_{IN\,H}$, $I_{EN\,H}$	Input High Current	V_{IN}, V_{EN} = H		30		µA

ELECTRICAL CHARACTERISTICS (Continued)

LOGIC CONTROL TO POWER DRIVE TIMING

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
t_1 (V_i)	Source Current Turn-off Delay	Fig. 12		300		ns
t_2 (V_i)	Source Current Fall Time	Fig. 12		200		ns
t_3 (V_i)	Source Current Turn-on Delay	Fig. 12		400		ns
t_4 (V_i)	Source Current Rise Time	Fig. 12		200		ns
t_5 (V_i)	Sink Current Turn-off Delay	Fig. 13		300		ns
t_6 (V_i)	Sink Current Fall Time	Fig. 13		200		ns
t_7 (V_i)	Sink Current Turn-on Delay	Fig. 13		400		ns
t_8 (V_i)	Sink Current Rise Time	Fig. 13		200		ns

(*) Limited by power dissipation

(**) In synchronous rectification the drain-source voltage drop VDS is shown in fig. 4 (L6202/03); typical value for the L6201 is of 0.3V.

Figure 1: Typical Normalized Is vs. T_j

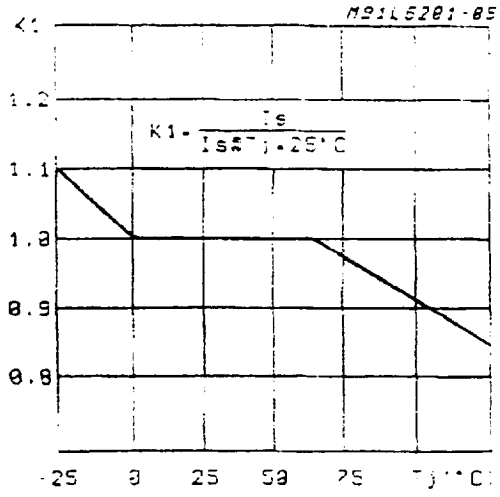


Figure 2: Typical Normalized Quiescent Current vs. Frequency

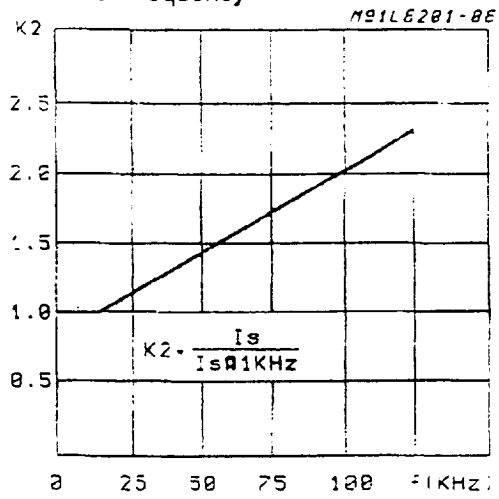


Figure 3: Typical Normalized Is vs. Vs

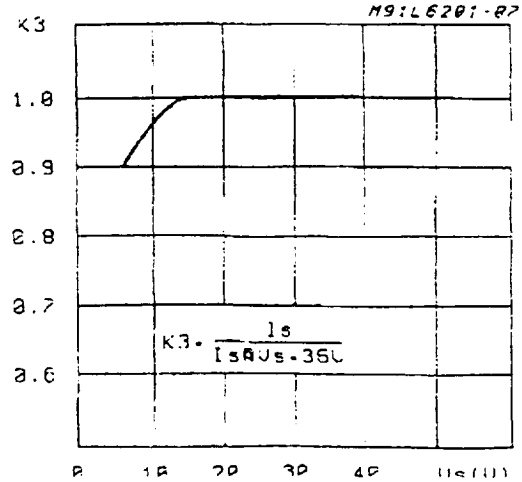


Figure 4: Typical $R_{DS\ (ON)}$ vs. Vs ~ V_{ref}

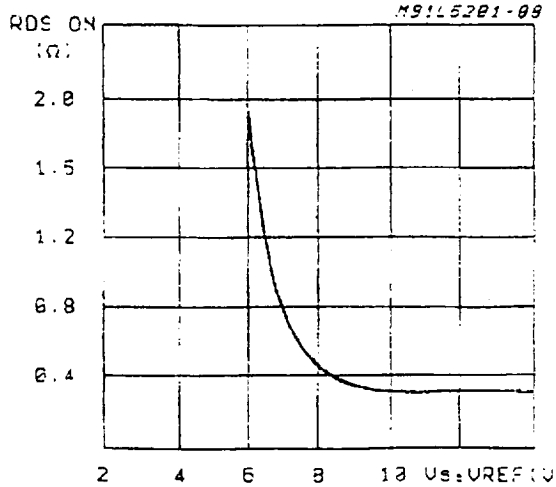


Figure 5: Normalized $R_{DS(ON)}$ at 25°C vs. Temperature Typical Values

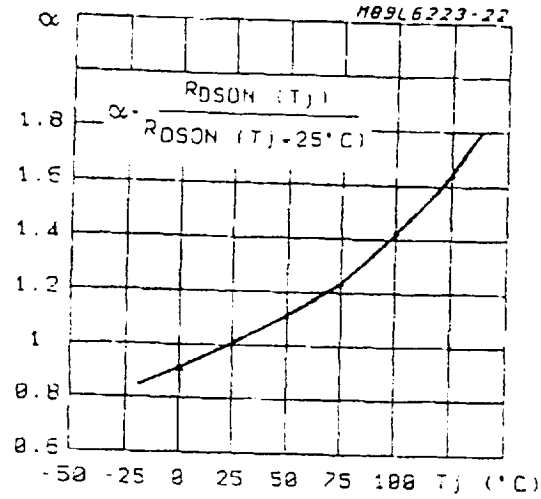


Figure 6a: Typical Diode Behaviour In Synchronous Rectification (L6201)

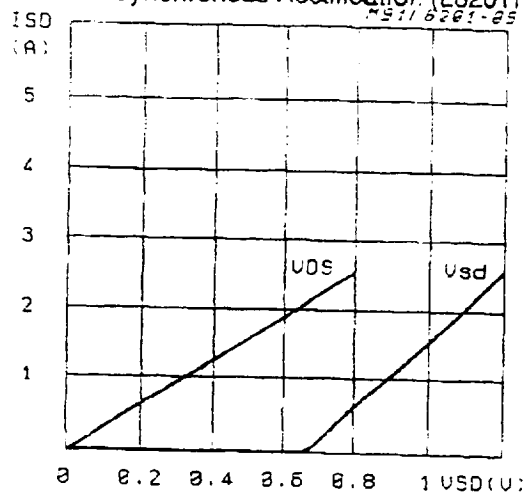


Figure 6b: Typical Diode Behaviour in Synchronous Rectification

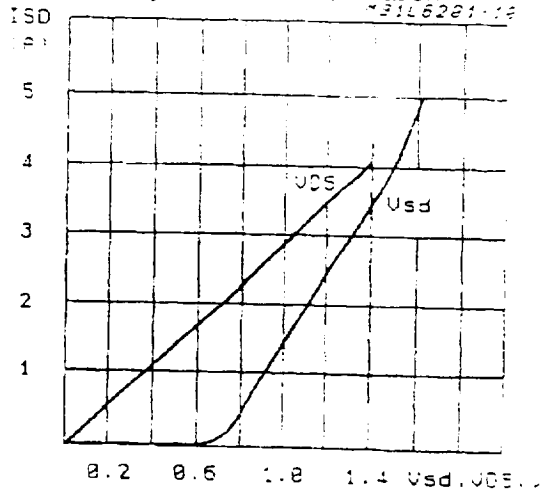


Figure 7a: Typical Power Dissipation vs I_L (L6201)

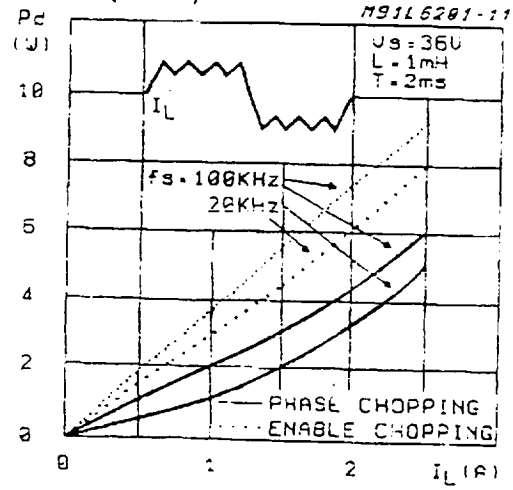


Figure 7b: Typical Power Dissipation vs I_L (L6202,L6203))

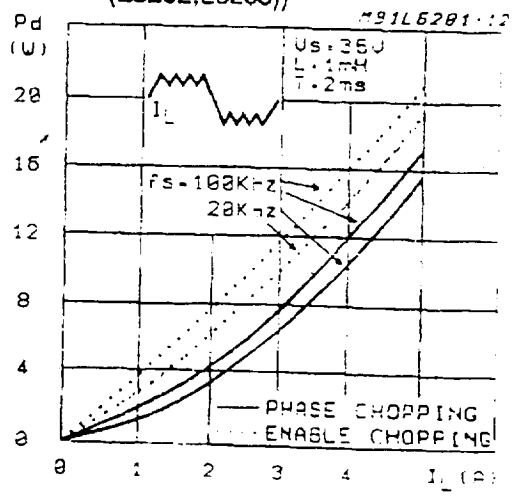


Figure 8a: Two Phase Chopping

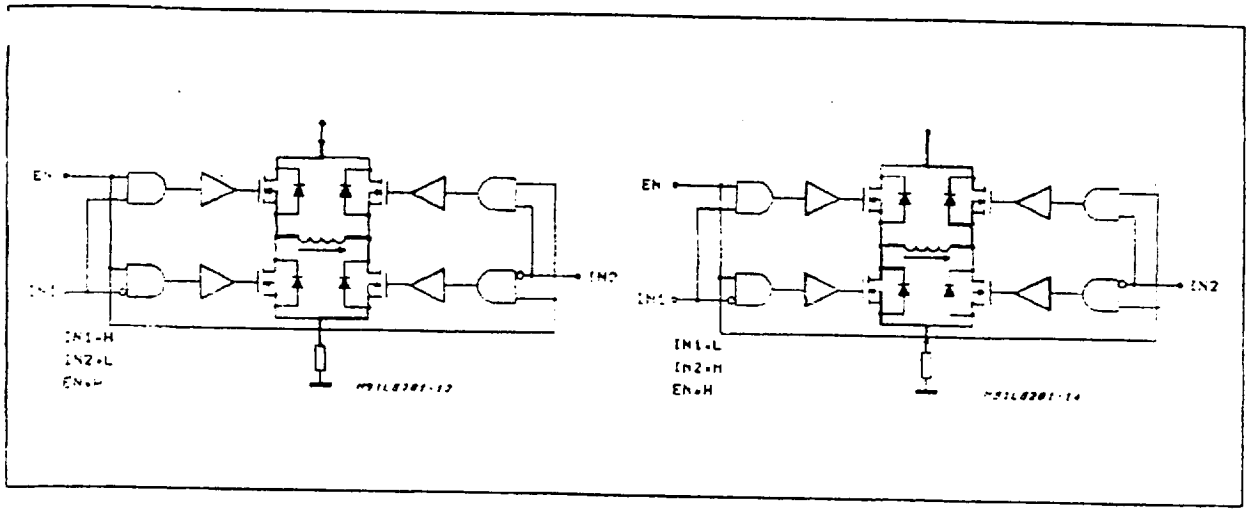


Figure 8b: One Phase Chopping

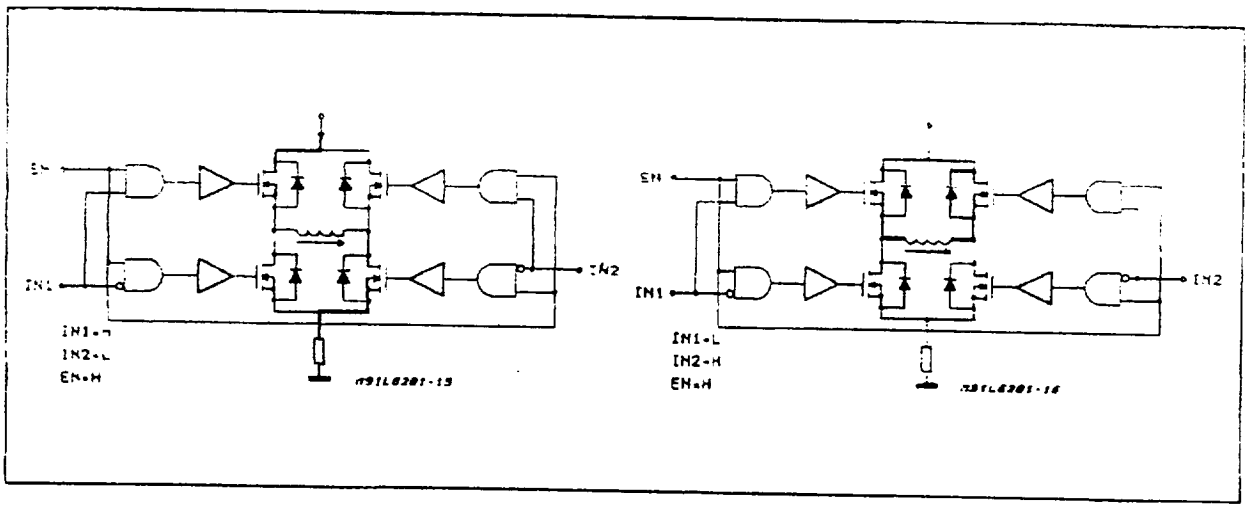
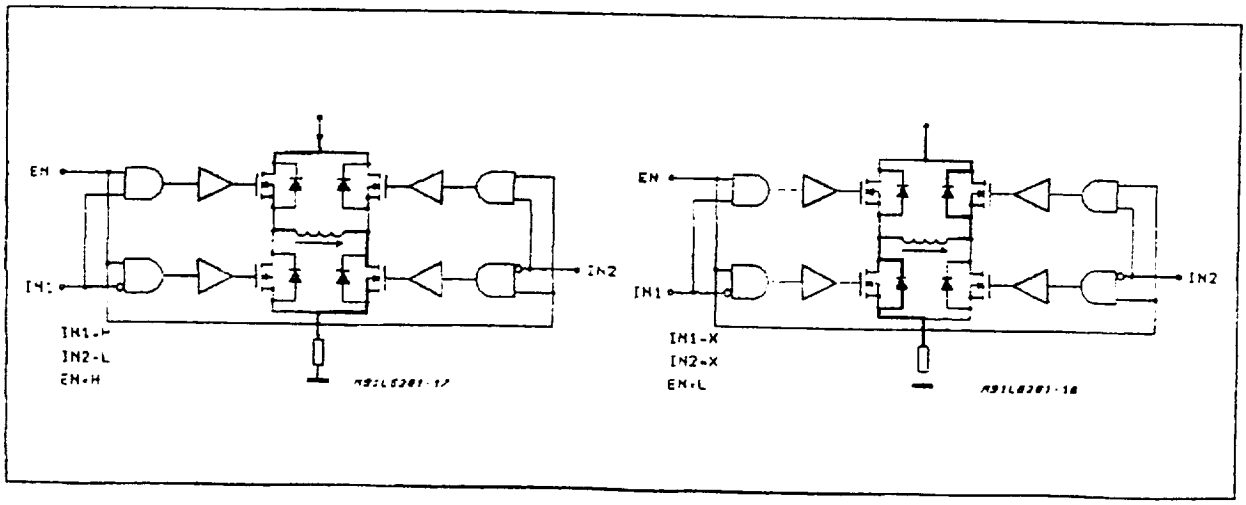


Figure 8c: Enable Chopping



TEST CIRCUITS

Figure 9: Saturation Voltage

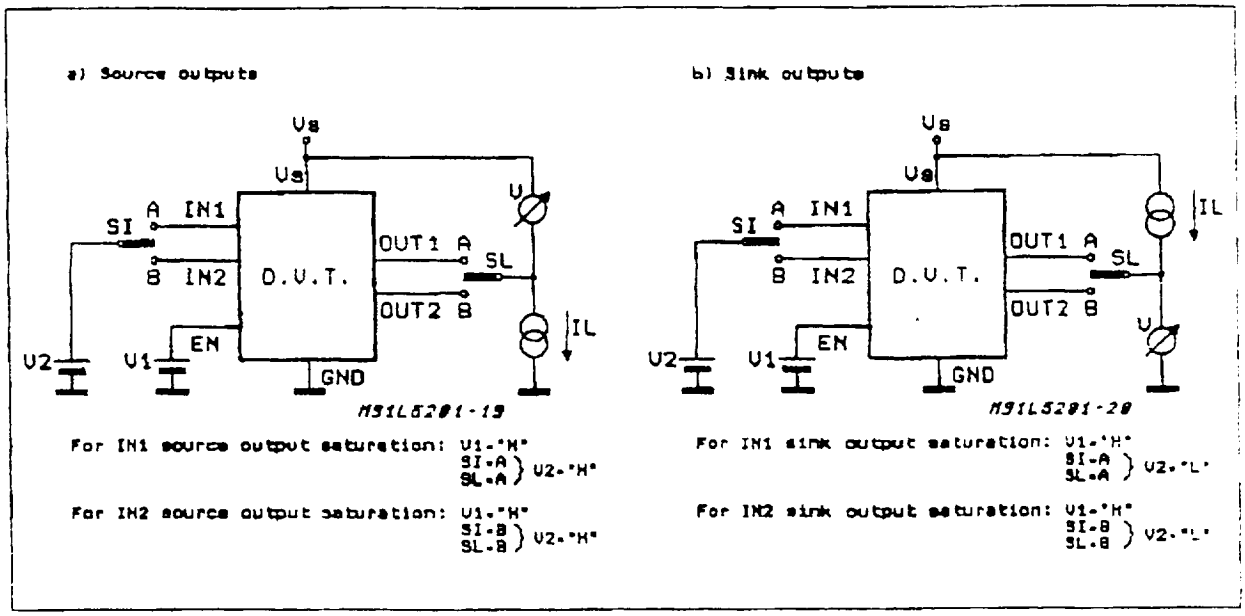


Figure 10: Quiescent Current

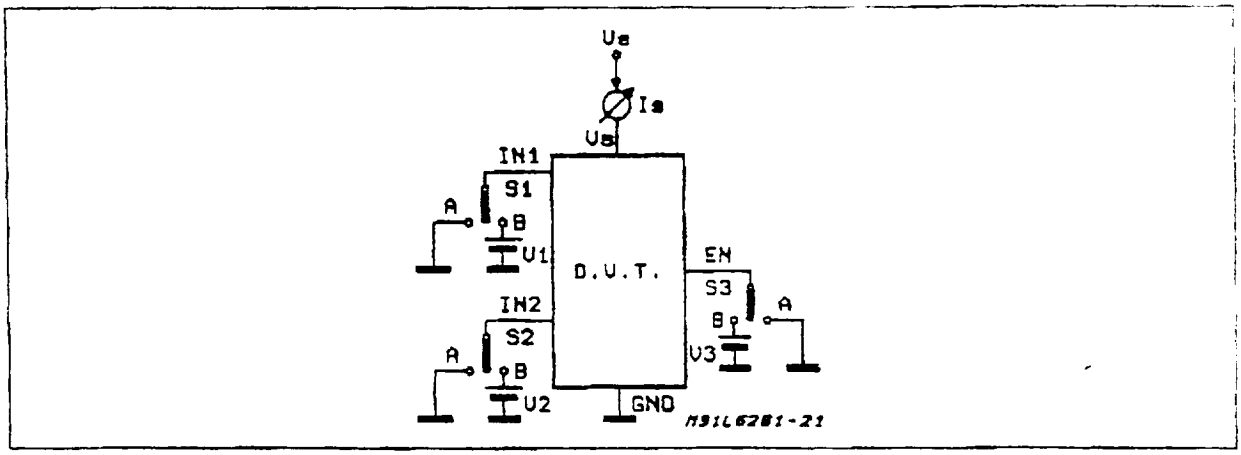


Figure 11: Leakage Current

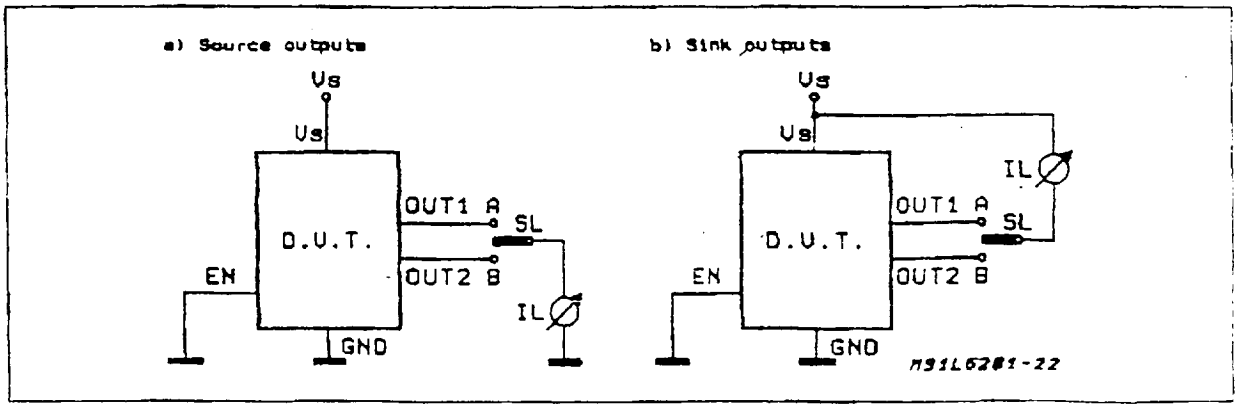


Figure 12: Source Current Delay Times vs. Input Chopper

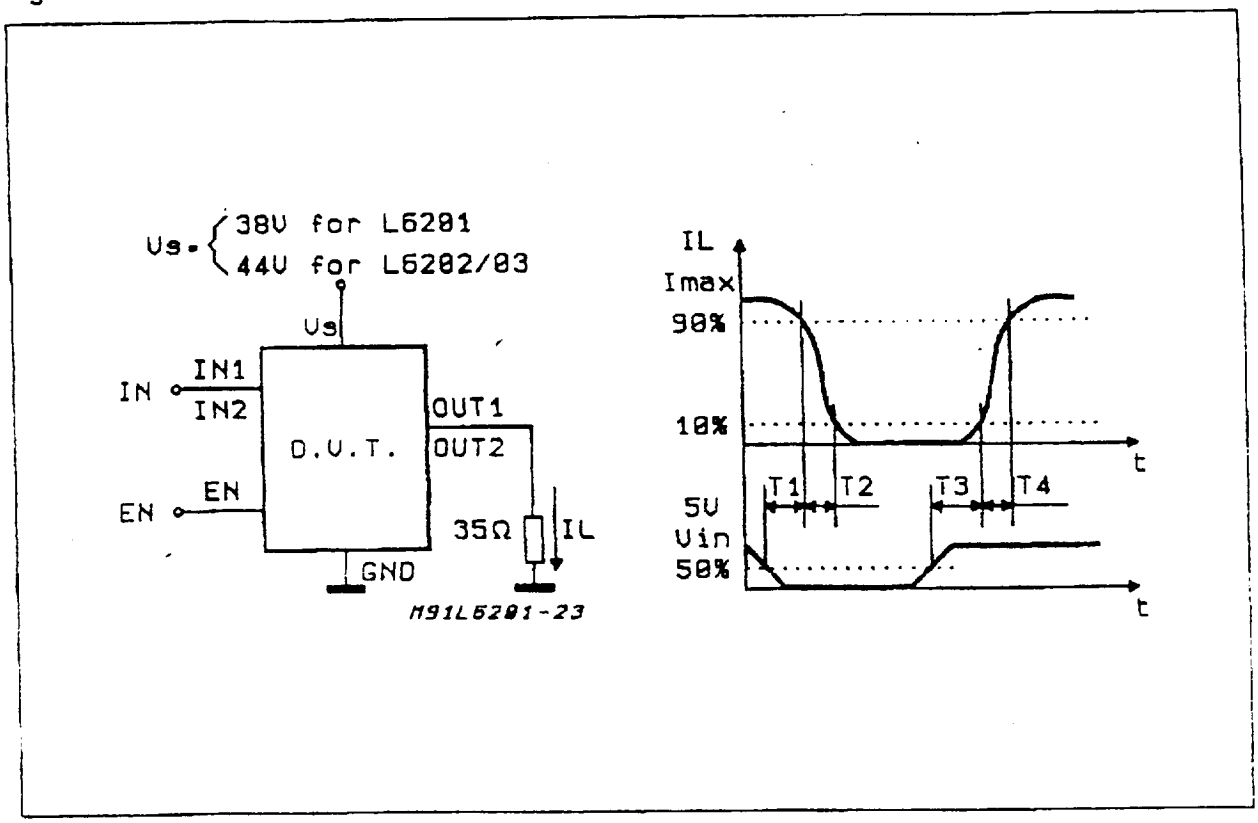
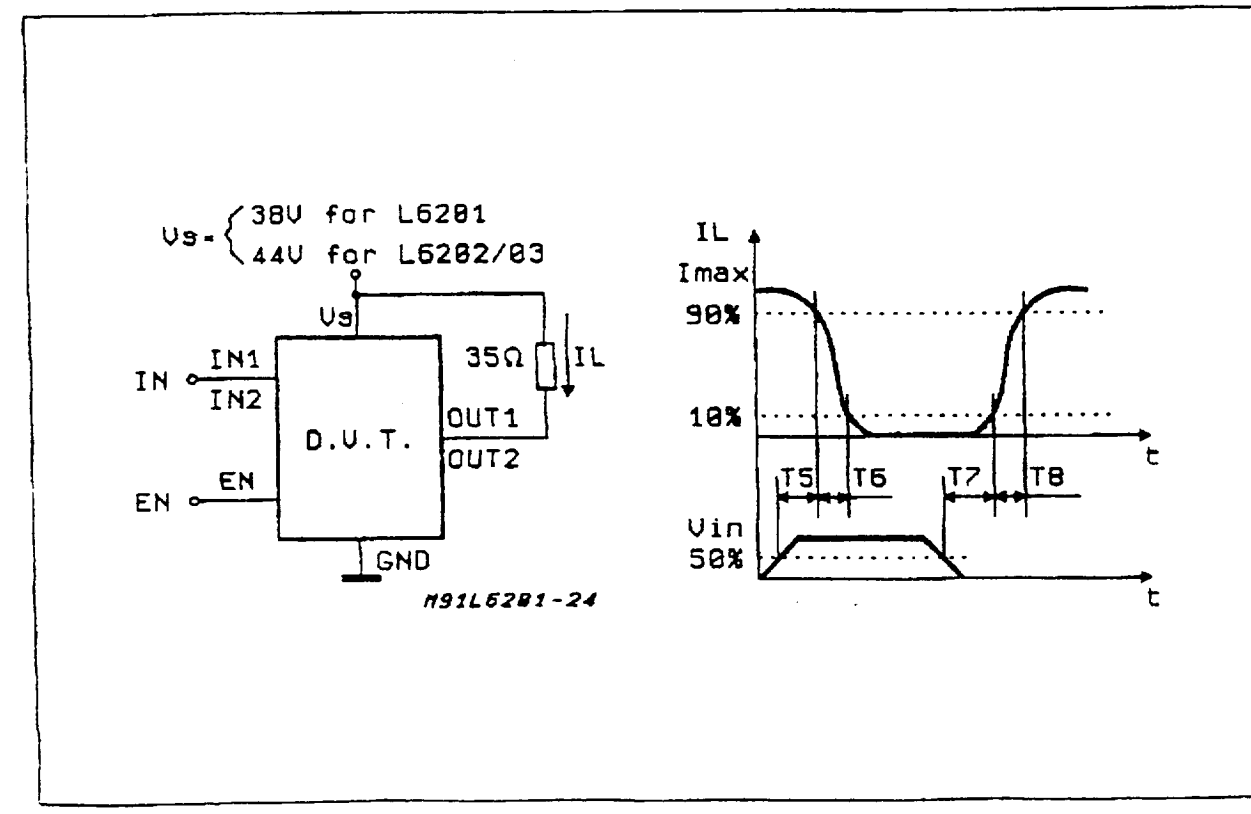


Figure 13: Sink Current Delay Times vs. Input Chopper



CIRCUIT DESCRIPTION

The L6201/2/3 is a monolithic full bridge switching motor driver realized in the new Multipower-BCD technology which allows the integration of multiple, isolated DMOS power transistors plus mixed CMOS/bipolar control circuits. In this way it has been possible to make all the control inputs TTL, CMOS and µC compatible and eliminate the necessity of external MOS drive components. The Logic Drive is shown in table 1.

Table 1

Inputs			Output Mosfets (*)
	IN1	IN2	
V_{EN} = H	L	L	Sink 1, Sink 2
	L	H	Sink 1, Source 2
	H	L	Source 1, Sink 2
	H	H	Source 1, Source 2
V_{EN} = L	X	X	All transistors turned oFF

L = Low H = High X = DON't care
(*) Numbers referred to INPUT1 or INPUT2 controlled output stages

Although the device guarantees the absence of cross-conduction, the presence of the intrinsic diodes in the POWER DMOS structure causes the generation of current spikes on the sensing terminals. This is due to charge-discharge phenomena in the capacitors C1 & C2 associated with the drain source junctions (fig. 14). When the output switches from high to low, a current spike is generated associated with the capacitor C1. On the low-to-high transition a spike of the same polarity is generated by C2, preceded by a spike of the opposite polarity due to the charging of the input capacity of the lower POWER DMOS transistor (fig. 15).

Figure 14: Intrinsic Structures in the POWER DMOS Transistors

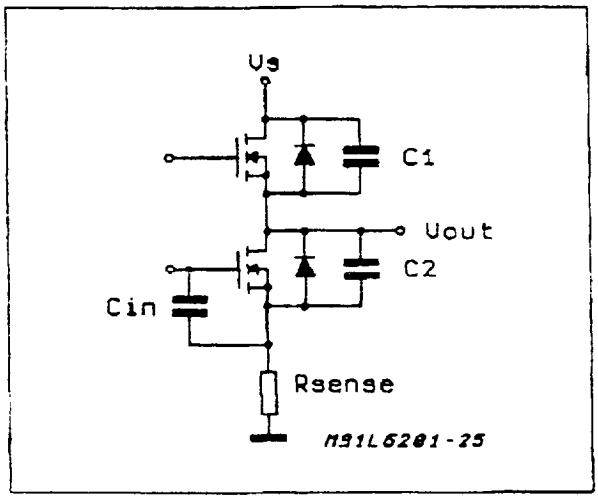
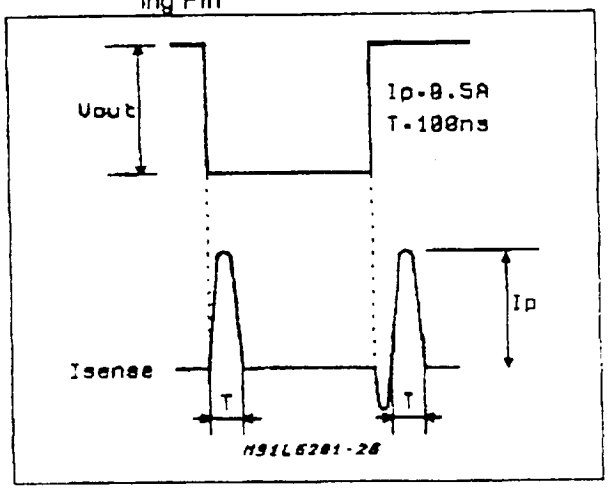


Figure 15: Current Typical Spikes on the Sensing Pin



TRANSISTOR OPERATION

ON State

When one of the POWER DMOS transistor is ON it can be considered as a resistor $R_{DS\ (ON)}$ throughout the recommended operating range. In this condition the dissipated power is given by :

$$P_{ON} = R_{DS\ (ON)} \cdot I_{DS}^2\ (RMS)$$

The low $R_{DS\ (ON)}$ of the Multipower-BCD process can provide high currents with low power dissipation.

OFF State

When one of the POWER DMOS transistor is OFF the V_{DS} voltage is equal to the supply voltage and only the leakage current I_{DSS} flows. The power dissipation during this period is given by :

$$P_{OFF} = V_S \cdot I_{DSS}$$

The power dissipation is very low and is negligible in comparison to that dissipated in the ON STATE.

Transitions

As already seen above the transistors have an intrinsic diode between their source and drain that can operate as a fast freewheeling diode in switched mode applications. During recirculation with the ENABLE input high, the voltage drop across the transistor is $R_{DS\ (ON)} \cdot I_D$ and when it reaches the diode forward voltage it is clamped. When the ENABLE input is low, the POWER MOS is OFF and the diode carries all of the recirculation current. The power dissipated in the transitional times in the cycle depends upon the voltage-current waveforms and in the driving mode. (see Fig. 7ab and Fig. 8abc).

$$P_{trans.} = I_{DS}\ (t) \cdot V_{DS}\ (t)$$

Boostrap Capacitors

To ensure that the POWER DMOS transistors are driven correctly gate to source voltage of typ. 10 V must be guaranteed for all of the N-channel DMOS transistors. This is easy to be provided for the lower POWER DMOS transistors as their sources are refered to ground but a gate voltage greater than the supply voltage is necessary to drive the upper transistors. This is achieved by an internal charge pump circuit that guarantees correct DC drive in combination with the boostrap circuit. For efficient charging the value of the boostrap capacitor should be greater than the input capacitance of the power transistor which is around 1 nF. It is recommended that a capacitance of at least 10 nF is used for the bootstrap. If a smaller capacitor is used there is a risk that the POWER transistors will not be fully turned on and they will show a higher RDS (ON). On the other hand if a elevated value is used it is possible that a current spike may be produced in the sense resistor.

Reference Voltage

To by-pass the internal Ref. Volt. circuit it is recommended that a capacitor be placed between its pin and ground. A value of 0.22 µF should be sufficent for most applications. This pin is also protected against a short circuit to ground: a max. current of 2mA max. can be sinked out.

Dead Time

To protect the device against simultaneous conduction in both arms of the bridge resulting in a rail to rail short circuit, the integrated logic control provides a dead time greater than 40 ns.

Thermal Protection

A thermal protection circuit has been included that will disable the device if the junction temperature reaches 150 °C. When the temperature has fallen to a safe level the device restarts the input and enable signals under control.

APPLICATION INFORMATION

Recirculation

During recirculation with the ENABLE input high, the voltage drop across the transistor is RDS (ON)· IL clamped at a voltage depending on the characteristics of the source-drain diode. Although the device is protected against cross conduction, current spikes can appear on the current sense pin due to charge/discharge phenomena in the intrinsic source drain capacitances. In the application this does not cause any problem because the voltage spike generated on the sense resistor is masked by the current controller circuit.

Rise Time T_r (See Fig. 16)

When a diagonal of the bridge is turned on current begins to flow in the inductive load until the maximum current I_L is reached after a time T_r. The dissipated energy $E_{OFF/ON}$ is in this case :

$$E_{OFF/ON} = [R_{DS\ (ON)} \cdot I_L^2 \cdot T_r] \cdot 2/3$$

Load Time T_{LD} (See Fig.16)

During this time the energy dissipated is due to the ON resistance of the transistors (E_{LD}) and due to commutation (E_{COM}). As two of the POWER DMOS transistors are ON, E_{ON} is given by :

$$E_{LD} = I_L^2 \cdot R_{DS\ (ON)} \cdot 2 \cdot T_{LD}$$

In the commutation the energy dissipated is :

$$E_{COM} = V_S \cdot I_L \cdot T_{COM} \cdot f_{SWITCH} \cdot T_{LD}$$

Where :

$T_{COM} = T_{TURN\text{-}ON} = T_{TURN\text{-}OFF}$

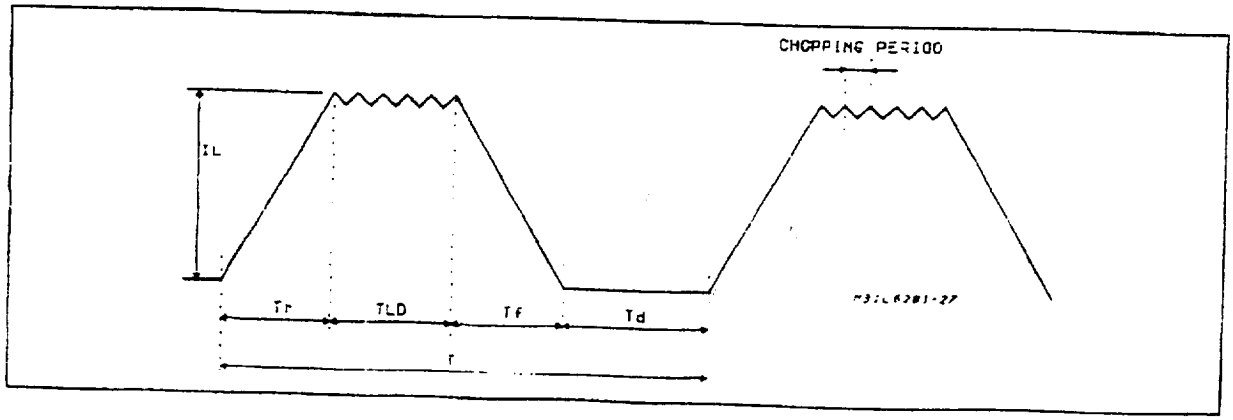
f_{SWITCH} = Chopping frequency.

Fall Time T_f (See Fig. 16)

It is assumed that the energy dissipated in this part of the cycle takes the same form as that shown for the rise time :

$$E_{ON/OFF} = [R_{DS\ (ON)} \cdot I_L^2 \cdot T_f] \cdot 2/3$$

Figure 16.



Quiescent Energy

The last contribution to the energy dissipation is due to the quiescent supply current and is given by:

$$E_{QUIESCENT} = I_{QUIESCENT} \cdot V_S \cdot T$$

Total Energy Per Cycle

$$E_{TOT} = E_{OFF/ON} + E_{LD} + E_{COM} + E_{ON/OFF} + E_{QUIESCENT}$$

The Total Power Dissipation P_{DIS} is simply :

$$P_{DIS} = E_{TOT}/T$$

T_r = Rise time

T_{LD} = Load drive time

T_f = Fall time

T_d = Dead time

T = Period

$T = T_r + T_{LD} + T_f + T_d$

DC Motor Speed Control

Since the I.C. integrates a full H-Bridge in a single package it is idealy suited for controlling DC motors. When used for DC motor control it performs the power stage required for both speed and direction control. The device can be combined with a current regulator like the L6506 to implement a transconductance amplifier for speed control, as shown in figure 17. In this particular configuration only half of the L6506 is used and the other half of the device may be used to control a second motor.

The L6506 senses the voltage across the sense resistor Rs to monitor the motor current: it compares the sensed voltage both to control the speed and during the brake of the motor.

Between the sense resistor and each sense input of the L6506 a resistor is recommended; if the connections between the outputs of the L6506 and the inputs of the L6203 need a long path, a resistor must be added between each input of the L6203 and ground.

A snubber network made by the series of R and C must be foreseen very near to the output pins of the I.C.; one diode (BYW98) is connected between each power output pin and ground as well.

The following formulas can be used to calculate the snubber values:

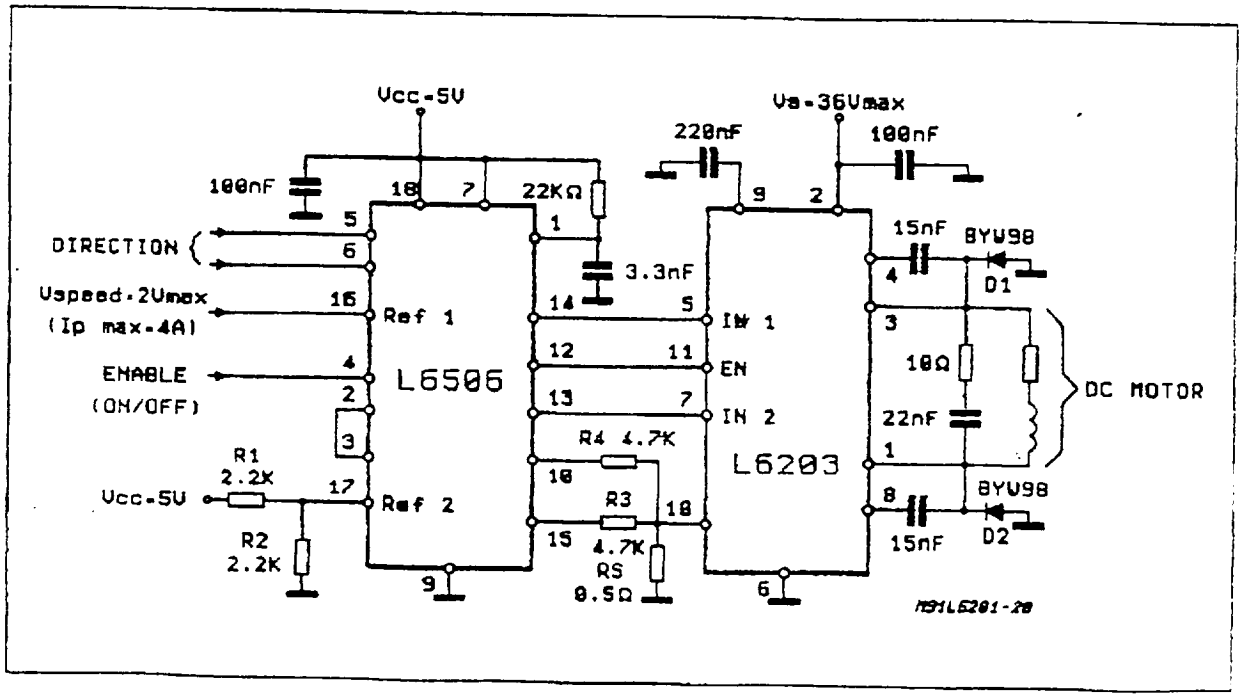
$R \cong Vs/I_p$

$C = I_p/(dV/dt)$ where:

Vs is the maximum Supply Voltage foreseen on the application;
I_p is the peak of the load current;
dv/dt is the limited rise time of the output voltage (200V/µs is generally used).

If the Power Supply Cannot Sink Current, a suitable large capacitor must be used and connected near the supply pin of the L6203. Sometimes a capacitor at pin 17 of the L6506 let the application better work. For motor current up to 2A max., the L6202 can be used in a similar circuit configuration for which a typical Supply Voltage of 24V is recommended.

Figure 17: Bidirectional DC Motor Control



BIPOLAR STEPPER MOTORS APPLICATIONS

Bipolar stepper motors can be driven with one L6506 or L297, two full bridge BCD drivers and very few external components. Together these three chips form a complete microprocessor-to-stepper motor interface is realized.

As shown in Fig. 18 and Fig. 19, the controller connect directly to the two bridge BCD drivers. External component are minimalized: an R.C. network to set the chopper frequency, a resistive divider (R1; R2) to establish the comparator reference voltage and a snubber network made by R and C in series (See DC Motor Speed Control).

Figure 18: Two Phase Bipolar Stepper Motor Control Circuit with Chopper Current Control

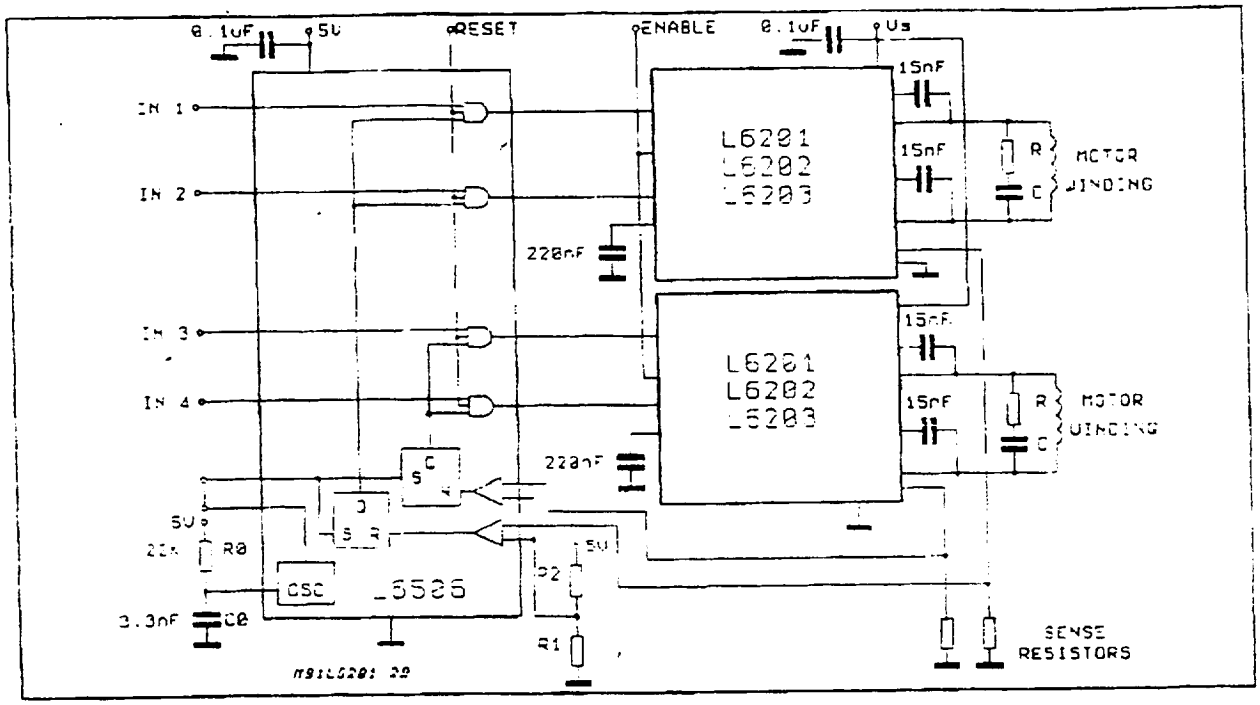
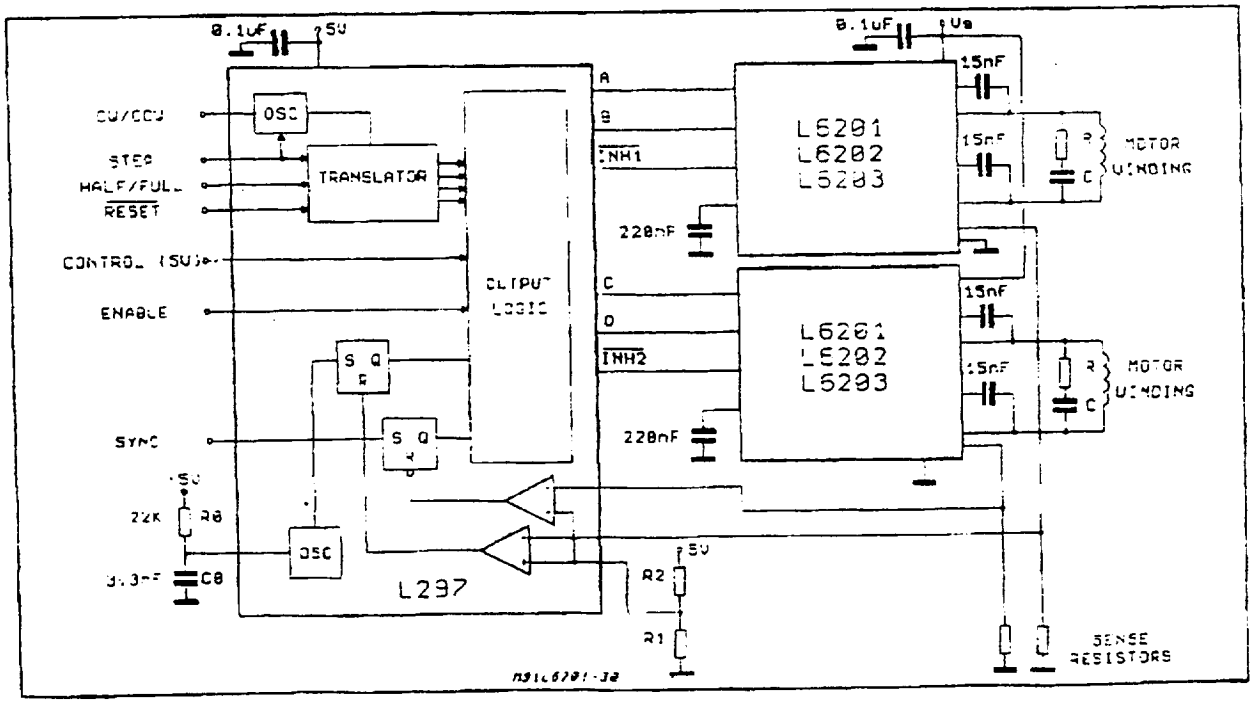
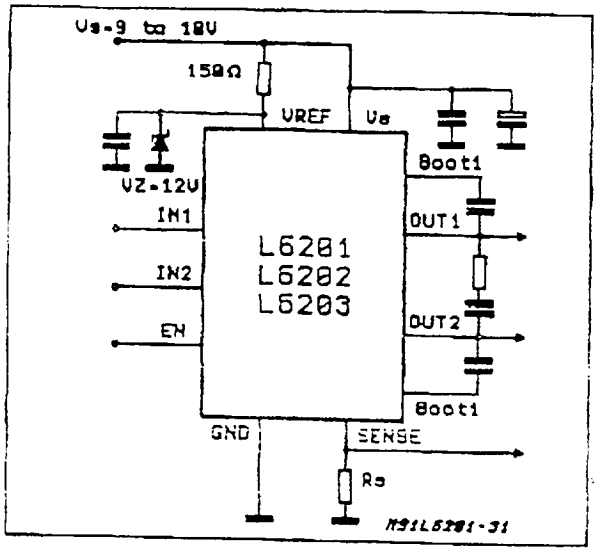


Figure 19: Two Phase Bipolar Stepper Motor Control Circuit with Chopper Current Control and Translator



It could be requested to drive a motor at Vs lower than the minimum recommended one of 12V (See Electrical Characteristics); in this case, by accepting a possible small increas in the $R_{DS\ (ON)}$ resistance of the power output transistors at the lowest Supply Voltage value, may be a good solution the one shown in Fig. 20.

Figure 20: L6201/2/3 Used at a Supply Voltage Range Between 9 and 18V



THERMAL CHARACTERISTICS

Thanks to the high efficiency of this device, often a true heatsink is not needed or it is simply obtained by means of a copper side on the P.C.B. (L6201/2).
Under heavy conditions, the L6203 needs a suitable cooling.
By using two square copper sides in a similar way as it shown in Fig. 23, Fig. 21 indicates how to choose the on board heatsink area when the L6201 total power dissipation is known since:

$$R_{Th\ j\text{-}amb} = (T_{j\ max.} - T_{amb\ max}) / P_{tot}$$

Figure 22 shows the Transient Thermal Resistance vs. a single pulse time width.
Figure 23 and 24 refer to the L6202.
For the Multiwatt L6203 addition information is given by Figure 25 (Thermal Resistance Junction-Ambient vs. Total Power Dissipation) and Figure 26 (Peak Transient Thermal Resistance vs. Repetitive Pulse Width) while Figure 27 refers to the single pulse Transient Thermal Resistance.

Figure 21: Typical $R_{Th\ J\text{-}amb}$ vs. "On Board" Heatsink Area (L6201)

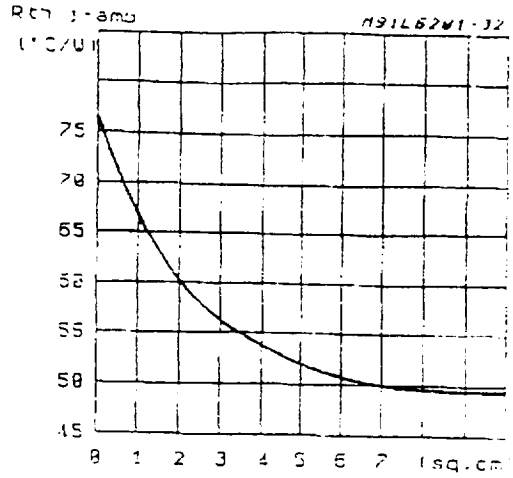


Figure 22: Typical Transient R_{TH} in Single Pulse Condition (L6201)

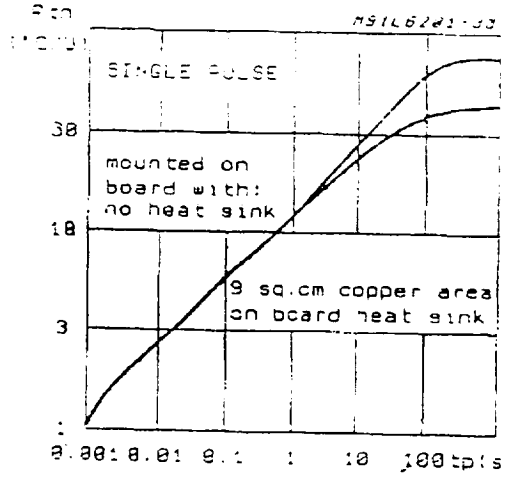


Figure 23: Typical $R_{Th\ J\text{-}amb}$ vs. Two "On Board" Square Heatsink (L6202)

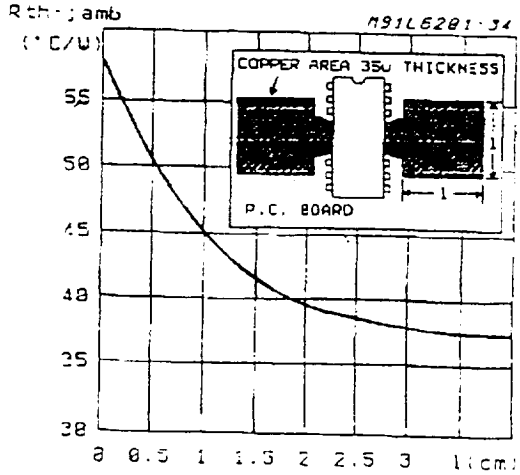


Figure 24: Typical Transient Thermal Resistance for Single Pulses (L6202)

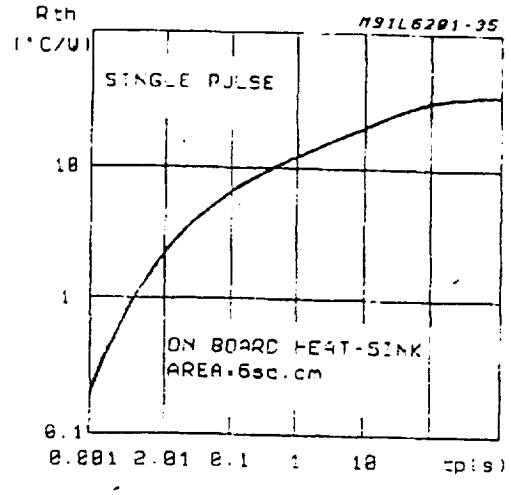


Figure 25: Typical $R_{th\ J\text{-}amb}$ of Multiwatt Package vs. Total Power Dissipation

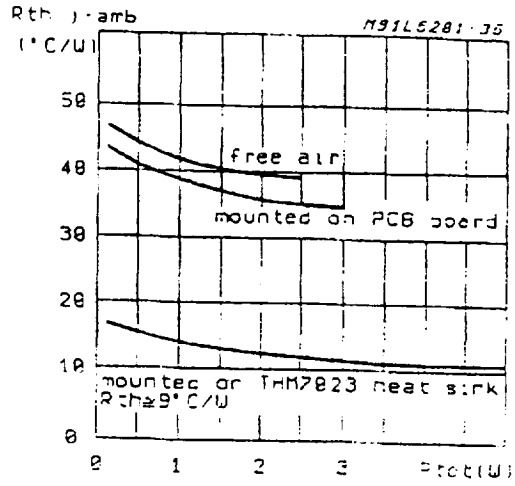


Figure 26: Typical Transient Thermal Resistance for Single Pulses with and without Heatsink (L6203)

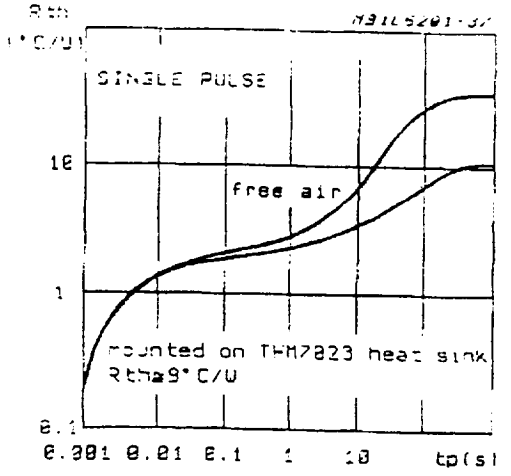
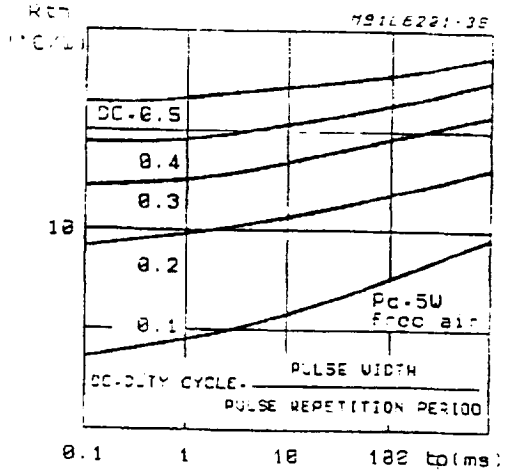


Figure 27: Typical Transient Thermal Resistance versus Pulse Width and Duty Cycle (L6203)



APPENDIX III.A.1

Code for master end of SPI communication

```
/* master.h
 * John Regehr 4/94
 */

#define DELAY          10

/* the port and bit that signals a communication; this is common
 * to all joint processors
 */
#define SIGNAL_PORT    PORTC
#define SIGNAL_MASK    0x01

/* set up the SPI - must be called before any communication can
 * take place
 */
int master_init (void)
{
   int i;

   /* wait for a little while so the main 11 can get organized */
   for (i=0; i<3000; i++);

/*   poke (DDRC, 0x00);*/
     asm "   LDAA  #0";
     asm "   LDX   #BASE";
     asm "   STAA  DDRC,X";

 }

 int sendrec (buf, procnum)
 unsigned char * buf;
 unsigned char procnum;
 {
     int i;

     /* set the green led for motor 1 during the communication */
 /*  poke (PORTB, 0x10);*/
     asm "   LDAB  #%00010000";
     asm "   LDX   #BASE";
     asm "   STAB  PORTB,X";

     /* turn on the SPI */
     initmaster ();
```

```
/*   for (j=0; j<50; j++);*/

     i = _sendrec_ (buf, procnum);

     /* light an LED */
     if (i == OK){
/*        poke (PORTB, 0x00);*/
          asm "   LDAA  #0";
          asm "   LDX   #BASE";
          asm "   STAA  PORTB,X";
     }
     else {
          /* poke (PORTB, 0x11);*/
          asm "   LDAA  #00010001";
          asm "   LDX   #BASE";
          asm "   STAA  PORTB,X";
     }

     /* turn off the SPI */
     deinit ();

/*   for (j=0; j<50; j++) { };*/

     return i;
}

/* all the real work gets done here */
int _sendrec_ (buf, procnum)
unsigned char * buf;
unsigned char procnum;
{
   unsigned char i, j, chksum, magic, otherprocnum;
   unsigned char rec [5];

   if ((procnum < 0) || (procnum >= NUMCHIPS))
     return EBADNUM;

   /* wait for the main processor to signal us. since the
     digital inputs on a miniboard seem to default
     to high we look for a low bit since we don't want a
     processor that becomes disconnected to start wildly
     transmitting. */
/*   do {
     c = peek (SIGNAL_PORT);
   } while ((c & SIGNAL_MASK) == 1);*/

   /* the checksum is defined to be the least significant byte
    * of the sum of the processor number and the data bits
    */
   chksum = procnum;
```

```
  /* send and receive the magic first byte, but don't check if
   * it's wrong since we want to go through with the transmission
   * anyway so the slave doesn't hang
   */

  if ((magic = m_sendrec(MAGIC)) != MAGIC) return ESYNC;

  for (j=0; j<DELAY; j++);

  /* push the data through */
  for (i=0; i<5; i++) {
    rec[i] = m_sendrec(buf[i]);
    for (j=0; j<DELAY; j++);
    chksum += buf[i];
  }

  /* send and receive the procnum */
  otherprocnum = m_sendrec (procnum);

  for (j=0; j<DELAY; j++);

  /* send and receive the checksum */
  chksum = m_sendrec (chksum);

  /* bail if the procnums don't agree */
  if (procnum != otherprocnum) {
    printdec (otherprocnum);
    return EWRONGCHIP;
  }

  /* compare the received data with the received
   * checksum
   */
  chksum -= otherprocnum;
  for (i=0; i<5; i++) chksum -= rec [i];
  if (chksum != 0) {
/*  for (i=0; i<5; i++) printdec (rec[i]);*/
    return ECHKSUM;
  }

  /* move the received bytes to the buffer where the caller
   * can see them
   */
  for (i=0; i<5; i++) buf[i] = rec[i];

  return OK;
}

/* return true if the main processor wants to talk */
/*int chkcom ()
```

```
{
    if ((peek (SIGNAL_PORT) & SIGNAL_MASK) == 0)
        return 1;
        else return 0;
}
*/
```

APPENDIX III.A.2

Code for slave end of SPI communication

```
/* slave.h
 * John Regehr 4/94
 */

/* ports and masks used to notify joint processors
 * that we want to talk
 */
int port [NUMCHIPS] = { PORTC, PORTC, PORTC, PORTC, PORTC, PORTC, PORTC,
PORTC, PORTB, PORTB, PORTB, PORTB, PORTB, PORTB, PORTB, PORTA,
PORTA };

unsigned char mask [NUMCHIPS] = { 0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0xfe, 0xfd,
0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f, 0xdf, 0xbf };

#define MAXCHAR     (int)256

/* set up the SPI - should be called first thing to make sure
 * ports are set high before any joint processor reaches the
 * point in the code where it sees if the main processor
 * wants to communicate
 */
slave_init ()
{
   initslave ();

   /* set port C data direction - all output */
   poke (DDRC, 0xff);

   /* set all the bits high */
   poke (PORTC, 0xff);

   /* set port B high */
   poke (PORTB, 0xff);

   /* zero port A control */
   poke (PACTL, 0x00);

   /* set up port A */
   poke (PORTA, 0xff);
 }

 int sendrec (buf, procnum)
 unsigned char * buf;
 unsigned char procnum;
```

```
{
    int i;

    i = _sendrec_ (buf, procnum);
    poke (port[procnum], 0xff);
    return i;
}

/* all the real work gets done here */
int _sendrec_ (buf, procnum)
unsigned char * buf;
unsigned char procnum;
{
  unsigned char rec [7];
  unsigned int ret;
  unsigned char i, chksum, magic, otherprocnum;

  if ((procnum < 0) || (procnum >= NUMCHIPS))
    return EBADNUM;

  /* set the bit on the port corresponding to the processor.
   * note that it is critical that only one of the 18 bits
   * going to joint 11s gets set low at any one time, or else
   * collisions occur. */
  poke (port[procnum], mask[procnum]);


  /* the checksum is defined to be the least significant byte
   * of the sum of the processor number and the data bits
   */
  chksum = procnum;

  /* send and receive the magic first byte, but don't check if
   * it's wrong since we want to go through with the transmission
   * anyway
   */

  /* each time s_sendrec is called, it should be called like this:
   *
   * unsigned char = integer = s_sendrec (unsigned char);
   *
   * The integer returned from s_sendrec contains the returned
   * character in the low byte, and 0 in the high byte if the
   * communication was successful.  So, if the integer value
   * is greater than MAXCHAR, a timeout occurred.
   */
  magic = ret = s_sendrec (MAGIC);
  if (ret > MAXCHAR) return ETIMEOUT;
```

```
    /* if we didn't get the magic character, then we're out of
     * sync with the joint processor.  since the bit to that
     * processor is already low, it will have to communicate with
     * us next time through it's loop, so we read characters until
     * we get the magic character, and then we should be in sync
     */
    while (magic != MAGIC) {
      magic = ret = s_sendrec (MAGIC);
      if (ret > MAXCHAR) return ETIMEOUT;
    }
    /* transmit the data */
    for (i=0; i<5; i++) {
      rec[i] = ret = s_sendrec (buf[i]);
       if (ret > MAXCHAR) return ETIMEOUT;
      chksum += buf[i];
    }
       /* send and receive the procnum */
    otherprocnum = ret = s_sendrec (procnum);
    if (ret > MAXCHAR) return ETIMEOUT;

    /* send and receive the checksum */
    chksum = ret = s_sendrec (chksum);
    if (ret > MAXCHAR) return ETIMEOUT;

    /* unset the bit */
    poke (port[procnum], 0xff);

    /* bail if the procnums don't agree */
    if (procnum != otherprocnum) {
      putstr ("procnum = ");
      printdec (procnum);
      putstr ("otherprocnum = ");
      printdec (otherprocnum);
      return EWRONGCHIP;
    }

    /* compare the received data with the received
     * checksum
     */
    chksum -= otherprocnum;
    for (i=0; i<5; i++) chksum -= rec [i];
    if (chksum != 0) return ECHKSUM;

    /* move the received bytes to the buffer where the caller
     * can see them
     */
    for (i=0; i<5; i++) buf[i] = rec[i];
     return OK;
}
```

APPENDIX III.B.1

Walking algorithm code for alpha joint.

```
/*
 alpha.c

 Doug R. Chapman
 NASA Robot Simulator V1.0

 Do a simple test of the leg process
 */

/* change these next three lines when re-compiling for each leg */
#define MY_LEG_NUMBER      1
#define pot_table          pot_A_1


#include "c:\one_leg\macro.h"
#include "c:\one_leg\constants.h"
#include "c:\one_leg\alpha.h"
#include "c:\one_leg\sendto.h"

byte new_alpha_pos; /* these contain the values that are to be sent to the
                       simulator previously each FSM called the joint command
                       to set the value this causes a problem when a FSM that
                       is to be subsumed sets a joint position */

byte alpha_bal;   /* value sent from the HC16 */
byte Beta1_pos;
byte Beta2_pos;
byte foot_down;

/* the table of potentiometer values will be held in a library file.  There
   will be 18 separate library files, each will define a block of eprom memory
   for pot_table. */

#define MY_PROCESSOR_NUMBER MY_LEG_NUMBER * JOINTS_PER_LEG + BETA2

extern byte * pot_table;

joint()
/* sets the joint to the angle specified by bye new_position */
{
    return;
}
```

```
initialize(void)
/* get the current position based on the potentiometers */
{
  new_alpha_pos = joint_pos();
/*   e_zero();*/ /* set the encoder value to 0 */
  return;
}

/* the following code was written by Steve Owens (spangles@cis)
  revisions made by Doug Chapman (chappy@cis). Most revisions are simply to match
  the simulator. Some revisions made for corectness */

/* This is the Finite State Machine for Alpha Advance */

int advance()
{
  static int advance_state;
  int advance_time;


  START_STATE(advance_state)
  {
   STATE(FULL_FORWARD)
   {
     STAMP(advance_time);
     if (!foot_down)
     {
       if (joint_pos() < desired_fw_position){
          new_alpha_pos = desired_fw_position;
          GOTO_STATE (advance_state,ADVANCE,ACTIVE);
       }
       GOTO_STATE(advance_state,FULL_FORWARD,ACTIVE);  /* remains active
because this
                                   leg needs to stay forward until
                                   it is back on the ground.  If
                                   it is inactive the alpha
                                   balance will move it */
     }
     else
     {
       /* foot is on ground */
       GOTO_STATE(advance_state,ON_GROUND,INACTIVE);
     }
     RELEASE;
   }
   STATE(ON_GROUND)
   {
     STAMP(advance_time);
     if (foot_down)
```

```
        {
          GOTO_STATE(advance_state,ON_GROUND,INACTIVE);
        }
        else
        {
          if (joint_pos () < desired_fw_position)
          {
            new_alpha_pos = desired_fw_position;
            GOTO_STATE(advance_state,ADVANCE,ACTIVE);
          }
        }
        RELEASE;
      }
      STATE(ADVANCE)
      {
        STAMP(advance_time);
        if(!foot_down)
        {
          if (joint_pos() < desired_fw_position){
            new_alpha_pos = desired_fw_position;
            GOTO_STATE(advance_state,ADVANCE,ACTIVE);
          }
        }
        else
        {
          /* foot is on ground */
          GOTO_STATE(advance_state,ON_GROUND,INACTIVE);
        }
        RELEASE;
      }
    }
}


int alpha_balance()
{

   new_alpha_pos = alpha_bal + joint_pos();
   return;
}



byte buffer[MAX_DATA_LEN];


int setup_communicate (void)
/* front end to the communication with the HC16 */
{
   int i;
```

```
  /* copy outgoing data into outgoing_buffer
    this order must agree with the HC16 */
  buffer[0] = joint_pos ();
  buffer[1] = f_limit ();
  buffer[2] = b_limit ();

  for (i = 3 ; i < MAX_DATA_LEN; i++)
    buffer[i] = 0;

  return 0;

}

int finish_communicate(void)
{

  /* copy data from incomming buffer */
  /* this needs to agree with what the HC16 sends */


  alpha_bal = buffer[0];
  Beta1_pos = buffer[1];
  Beta2_pos = buffer[2];
  foot_down = buffer[3];

  return 0;
}

main(){

  /* initialize the new position variables, this is needed so that
    if none of the FSM's set it the call to joint will not send garbage
    values to the simulator */

  initialize();

  for ( ; ; ){
    if (advance() != ACTIVE)
      alpha_balance();
    joint ();

    if (tr_rdy()){
      setup_communicate();
      if (sendto(buffer,MY_PROCESSOR_NUMBER) == VALID)
        finish_communicate();
    }
  }
}
```

APPENDIX III.B.2

Walking algorithm code for beta 1 joint.

```
/*
 beta1.c

 Doug R. Chapman
 NASA Robot One Leg Prototype V1.0

 Do a simple test of the leg process
 */

/* change these next three lines when re-compiling for each leg */
#define MY_LEG_NUMBER    1
#define pot_table        pot_B1_1


#include "c:\one_leg\macro.h"
#include "c:\one_leg\beta1.h"
#include "c:\one_leg\constants.h"
#include "c:\one_leg\sendto.h"

int new_beta1_pos;   /* previously each FSM called the joint command to
                        set the values.  This causes a problem when a
                        FSM that is to be subsumed sets a joint position */



/* signals from the HC16 */
byte up_leg_trigger; /* 0 = no trigger, 1 = trigger */
byte emergency_stop; /* 0 = OK, 1 = stop */

/* data for other joints on this leg (sent from HC16) */
byte Alpha_pos;
byte Beta2_pos;
byte foot_down; /* 0 = off ground, 1 = on ground */

#define MY_PROCESSOR_NUMBER MY_LEG_NUMBER * JOINTS_PER_LEG + BETA2

extern byte * pot_table;


joint()
/* sets the joint to the angle specified by bye new_position */
{

   return;
```

```
}

initialize(void)
/* get the current position based on the potentiometers */
{
   new_beta1_pos = joint_pos();
/*    en_zero();*/ /* set the encoder value to 0 */
   return;
}


/* the following code was written by Steve Owens (spangles@cis)
  revisions made by Doug Chapman (chappy@cis). Most revisions are simply to match
  the simulator. Some revisions made for corectnes */

/* This is the Finite State Machine for the Leg Up/Down process */


int leg()
{
   static int leg_state;
   int leg_time;

   START_STATE (leg_state)
   {
    STATE(LEG_IS_DOWN)
    {
      STAMP (leg_time);
      if (up_leg_trigger)
      {
       /* We are starting to lift the leg */
       new_beta1_pos = desired_up_pos;
       GOTO_STATE(leg_state,LEG_IS_RAISING,ACTIVE);
      }
      else
      {
       /* We are staying on the ground */
       GOTO_STATE(leg_state,LEG_IS_DOWN,INACTIVE);
      }
      RELEASE;
    }
    STATE(LEG_IS_RAISING)
    {
      STAMP(leg_time);
      if(up_leg_trigger)
        if(joint_pos() < desired_up_pos)
        {
          /* We are still raising the leg */
          new_beta1_pos = desired_up_pos;
          GOTO_STATE(leg_state,LEG_IS_RAISING,ACTIVE);
```

```
    }
    else
    {
      /* Max B1 position reached Can't go higher */
      GOTO_STATE(leg_state,LEG_IS_UP,ACTIVE);
    }
  else
  {
   /* Time to lower leg */
   new_beta1_pos = desired_dn_pos;
   GOTO_STATE(leg_state,LEG_IS_LOWERING,ACTIVE);
  }
  RELEASE;
}
STATE(LEG_IS_UP)
{
  STAMP(leg_time);
  if(up_leg_trigger)
  {
   /*leg is staying up so stay here */
   GOTO_STATE(leg_state,LEG_IS_UP,ACTIVE);
  }
  else
  {
   /* Time to lower the leg */
   new_beta1_pos = desired_dn_pos;
   GOTO_STATE(leg_state,LEG_IS_LOWERING,ACTIVE);
  }
  RELEASE;
}
STATE(LEG_IS_LOWERING)
{
  STAMP(leg_time);
  if (up_leg_trigger)
  {
   /* Legs go back up */
   new_beta1_pos = desired_up_pos;
   GOTO_STATE(leg_state,LEG_IS_RAISING,ACTIVE);
  }
  else if (foot_down)
  {
   GOTO_STATE(leg_state,LEG_IS_DOWN,ACTIVE);
  }
  else
  {
   /* Keep putting me down */
   new_beta1_pos = desired_dn_pos;
   GOTO_STATE(leg_state,LEG_IS_LOWERING,ACTIVE);
  }
  RELEASE;
```

```
    }
  }
  return;
}

byte buffer[MAX_DATA_LEN];

int setup_communicate (void)
/* front end to the communication with the HC16 */
{

  int i;

  /* copy outgoing data into outgoing_buffer
    this order must agree with the HC16 */

  buffer[0] = joint_pos ();
  buffer[1] = f_limit ();
  buffer[2] = b_limit ();

  for (i = 3; i < MAX_DATA_LEN; i++)
    buffer[i] = 0;

  return 0;

}

int finish_communicate(void)
{

  /* copy data from incomming buffer */
  /* this needs to agree with what the HC16 sends */

  up_leg_trigger = buffer[0];
  emergency_stop = buffer[1];
  Alpha_pos = buffer[2];
  Beta2_pos = buffer[3];
  foot_down = buffer[4];

  return 0;
}

main(){

  /* initialize the new position variables, this is needed so that
    if none of the FSM's set it the call to joint will not send garbage
    values to the simulator */

  initialize();
```

```
  for ( ; ; ){
    leg();
    joint ();

    if (tr_rdy()){
       setup_communicate();
       if (sendto() == VALID)
         finish_communicate();
     }

  }
}
```

APPENDIX III.B.3

Walking algorithm code for beta 2 joint.

```
/*
 beta2.c

 Doug R. Chapman
 NASA Robot One Leg Prototype

 Do a simple test of the leg process
 */


/* change these next three lines when re-compiling for each leg */
#define _LEG_1_
#define MY_LEG_NUMBER      1
#define pot_table          pot_B2_1


#include "c:\one_leg\beta2.h"
#include "c:\one_leg\macro.h"
#include "c:\one_leg\constants.h"
#include "c:\one_leg\sendto.h"

int new_beta2_pos;/* previously each FSM called the joint command to set the values*/
/* this causes a problem when a FSM that is to be subsumed sets
  a joint position */

byte emergency_stop;   /* value sent from the HC16 */
byte Alpha_pos;
byte Beta1_pos;

#define MY_PROCESSOR_NUMBER MY_LEG_NUMBER * JOINTS_PER_LEG + BETA2

extern byte * pot_table;

joint()
/* sets the joint to the angle specified by new_position */
{

  return;
}

initialize(void)
/* get hte current position based on the potentiomenters */
{
    new_beta2_pos = joint_pos();
```

```
    e_zero(); /* set the encoder value to 0 */
    return;
}

byte buffer[MAX_DATA_LEN];

int setup_communicate (void)
/* front end to the communication with the HC16 */
{

    int i;

    /* copy outgoing data into outgoing_buffer
       this order must agree with the HC16 */

    buffer[0] = joint_pos ();
    buffer[1] = f_limit ();
    buffer[2] = b_limit ();
    buffer[3] = (foot_sensor() < FOOT_TRESHOLD) ? FALSE : TRUE;

    for (i = 4; i < MAX_DATA_LEN; i++)
      buffer[i] = 0;

    return 0;

}

int finish_communicate(void)
{

    /* copy data from incomming buffer */
    /* this needs to agree with what the HC16 sends */


    emergency_stop = buffer[0];
    Alpha_pos = buffer[1];
    Beta1_pos = buffer[2];

    return 0;
}

main(){

    /* initialize the new position variables, this is needed so that
       if none of the FSM's set it the call to joint will not send garbage
       values to the simulator */

    initialize();
    for ( ; ; ){
      new_beta2_pos = joint_pos();
```

```
    joint ();

    if (tr_rdy()){
       setup_communicate();
       if (sendto() == VALID)
         finish_communicate();
    }

  }

  /* beta 2 has not yet been implemented */


}
```

APPENDIX III.C.1

Temporary walking algorithm code for alpha joint.

```
#include c:\newwalk\constant.h
int state;

int signal ()
{
      /* delete this later!! */
      putchr (state+'0');
      if ((peek (PORTC) & IN_MASK) == 0) {
            /* green led */
            poke (PORTB, 0x10);
            return TRUE;
      } else {
            /* red led */
            poke (PORTB, 0x11);
            return FALSE;
      }
}

int set_out ()
{
      poke (PORTB, 0x20);
      poke (PORTC, OUT_SET);
      return(0);
}

int clear_out ()
{
      poke (PORTB, 0x22);
      poke (PORTC, OUT_CLEAR);
      return(0);
}

int main ()
{
      /* setup bit 0 of C for output, rest are input */
      poke (DDRC, 0x01);

      state = INITIAL_STATE;

      while (1) {

            /* FSM! */
            switch (state) {
            case 0: {
```

```
        /* initial state for 0, 3, 4 */
        set_out ();
        while (!signal());
        /* no break; */
        }
    case 1: {
        /* dummy loop */
        while (signal());
        while (!signal());
        state = 6;
        break;
        }
    case 2: {
        /* move alpha forward */
        motor (FSPEED);
        while (analog(0) < FORWARD_VALUE);
        motor (0);
        clear_out ();
        while (signal());
        set_out ();
        while (!signal());
        /* no break; */
        }
    case 3: {
        /* dummy loop */
        while (signal());
        while (!signal());
        /* no break; */
        }
    case 4: {
        /* move alpha back */
        motor (BSPEED);
        while (analog(0) > BACK_VALUE);
        motor (0);
        clear_out ();
        while (signal());
        set_out ();
        while (!signal());
        /* no break; */
        }
    case 5: {
        set_out ();
        while (!signal());
        /* no break; */
        }
    case 6: {
        /* dummy loop */
        while (signal());
        while (!signal());
        /* no break; */
```

```
            }
        case 7: {
            /* dummy loop */
            while (signal());
            while (!signal());
            /* no break; */
            }
        case 8: {
            /* dummy loop */
            while (signal());
            while (!signal());
            /* no break; */
            }
        case 9: {
            /* move alpha back */
            motor (BSPEED);
            while (analog(0) > BACK_VALUE);
            motor (0);
            clear_out ();
            while (signal());
            set_out ();
            while (!signal());
            /* no break; */
            }
        }
    }
    return(0);
    /* never reached */
}
```

APPENDIX III.C.2

Walking algorithm code for beta 2 joint.

```
#include c:\newwalk\constant.h
int state;

int signal ()
{
     /* delete this later!! */
     putchr (state+'0');
     if ((peek (PORTC) & IN_MASK) == 0) {
          /* green led */
          poke (PORTB, 0x10);
          return TRUE;
     } else {
          /* red led */
          poke (PORTB, 0x11);
          return FALSE;
     }
}

int set_out ()
{
     poke (PORTB, 0x20);
     poke (PORTC, OUT_SET);
     return(0);
}

int clear_out ()
{
     poke (PORTB, 0x22);
     poke (PORTC, OUT_CLEAR);
     return(0);
}

int main ()
{
     /* setup bit 0 of C for output, rest are input */
     poke (DDRC, 0x01);

     state = INITIAL_STATE;

     while (1) {

          /* FSM! */
          switch (state) {
          case 0:
```

```
        /* initial state for 0, 3, 4 */
        set_out ();
        while (!signal());
        /* no break; */
case 1:
        /* move beta1 up */
        motor (USPEED);
        while (analog(0) < UP_VALUE);
        motor (0);
        clear_out ();
        while (signal());
        set_out ();
        while (!signal());
        /* no break; */
case 2:
        /* dummy loop */
        while (signal());
        while (!signal());
        /* no break; */
case 3:
        /* move beta1 down */
        motor (DSPEED);
        while (analog(0) > DOWN_VALUE);
        motor (0);
        clear_out ();
        while (signal());
        set_out ();
        while (!signal());
        /* no break; */
case 4:
        /* dummy loop */
        while (signal());
        while (!signal());
        state = 6;
        break;
case 5:
        set_out ();
        while (!signal());
        /* no break; */
case 6:


        /* dummy loop */
        while (signal());
        while (!signal());
        /* no break; */
case 7:
        /* dummy loop */
        while (signal());
        while (!signal());
```

```
                /* no break; */
            case 8:
                /* dummy loop */
                while (signal());
                while (!signal());
                /* no break; */
            case 9:
                /* dummy loop */
                while (signal());
                while (!signal());
                /* no break; */
            }
        }
        return(0);
        /* never reached */
}
```

APPENDIX III.C.3

Main control code for temporary walking algorithm.

```
#include c:\constant.h
#define step1 0xE6  /* 11100110 */
#define step2 0x19  /* 00011001 */

main()
{
      int state;
      poke(DDRC, 0x00);
      /* clear */
      poke(PORTB,0xff);

      for (state=0; state<3; state++)
      {
            putstr("We are in the first half of the step \n");
            /* signal */
            poke (PORTB,0x00);
            putstr("We have signaled \n");
            /* wait */
            while(peek(DDRC)&step2!=0);
            putstr("0, 3, and 4 have cleared \n");
            poke(PORTC,0xff);
      }

      /* alpha balance */
      putstr("We are doing the alpha balance \n");
      poke (PORTC,0x00);

      while (!peek(PORTC));
      putstr("all processors have cleared \n");
      /*clear */
      poke(PORTC,0xff);


      for (state=0; state<3; state++)
      {
            putchr("We are in the second part of the walk \n");

            /* signal */
            poke (PORTC,0x00);
            putstr("We have signaled \n");
            /* wait */
            while (peek(PORTC)!=step2);
```

```
        putstr("1, 2, and 5 have cleared \n");
        poke(PORTC,0xff);
    }

    /* alpha balance */
    putstr("We are doing the alpha balance \n");
    poke (PORTC,0x00);

    while (!peek(PORTC));
    putstr("all processors have cleared \n");
    /*clear */
    poke(PORTC,0xff);


}
```