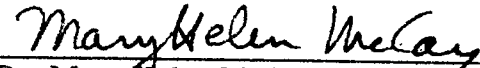


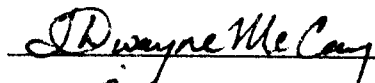
To the Graduate Council:

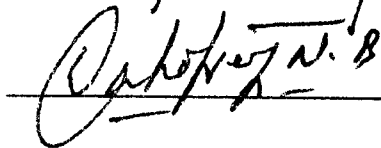
I am submitting herewith a thesis written by Cindy R. McKowen entitled "The Design, Fabrication, and Testing of a Centrifuge for Laser Materials Processing under Enhanced Gravity Conditions." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Engineering Science.



Dr. Mary Helen McCay, Major Professor

We have read this thesis
and recommend its acceptance:





Accepted for the Council:



Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in her absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Cindy R. McKown

Date 18 August 1993

**THE DESIGN, FABRICATION, AND TESTING OF A
CENTRIFUGE FOR LASER MATERIALS PROCESSING
UNDER ENHANCED GRAVITY CONDITIONS**

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Cindy R. McKowen

December 1993

DEDICATION

This thesis is dedicated to my husband

Mr. Bruce Wayne McKowen

who encouraged me to keep going

and provided a light at the end of the graduate school tunnel

and to my parents

Mr. Robert Irving Rowland

and

Mrs. Patricia Gale Rowland

who gave me my start.

ACKNOWLEDGMENTS

I would like to thank the faculty, staff and students of the UTSI Center for Laser Applications for their help in the course of the research and writing of this thesis. In particular, I would like to thank Drs. Mary Helen and T. Dwayne McCay, Dr. Narendra Dahotre, Robert Mueller, Carl Schou, and Franz-Josef Kahlen for their continued (and sometimes continuous) help. Special thanks goes to Mike Sharp who was both my mentor and friend during the course of my stay at UTSI.

ABSTRACT

A simple, cost-effective centrifuge for use in an enhanced gravity laser materials processing system was developed. This system is capable of producing an acceleration normal to the sample surface ranging from $2g_e$ to $10g_e$. Design constraints include a maximum resultant acceleration of $10g_e$ and a beam delivery system which is capable of using either a CO_2 or Nd:YAG laser system as the energy source. The available process diagnostics include photodiodes, thermocouples, a CCD camera, and accelerometers in three axes.

An experiment was run to test the system and determine the effect of gravity on the weld pool shape by performing spot welds in Inconel 718 at effective g levels ranging from $2g_e$ to $10g_e$ at $2g_e$ intervals. Post-process metallurgical analysis was performed to determine the shape of the weldment.

The results of the experimental data indicate that gravity seems to play a slight role in the size and shape of partial penetration welds; however, differences between results for cw data and ccw data also indicate the possibility of a coriolis effect.

TABLE OF CONTENTS

Chapter	Page
1. INTRODUCTION	1
2. BACKGROUND	4
3. ENHANCED GRAVITY LASER MATERIALS PROCESSING SYSTEM .	12
System Description	12
Design Constraints	14
A: Centrifuge Motion Package Module	16
B: Process Module	21
C: Diagnostics Module	30
D: Data Acquisition and Storage Module	34
E: Power Supply Module	35
F: System Control Module	36
G: Safety Enclosure	37
4. EXPERIMENT	39
Experimental Parameters	39
Overview	40
Experiment Procedure	42
5. RESULTS	49
Centrifuge Performance	49
Experimental Results	56

TABLE OF CONTENTS

Chapter	Page
Summary	66
6. CONCLUSIONS AND RECOMMENDATIONS	67
Conclusions	67
Recommendations	68
BIBLIOGRAPHY	69
APPENDICES	72
Appendix 1: Programs	73
Appendix 2: Technical Specifications	109
VITA	131

LIST OF FIGURES

Figure	Page
2-1 Parabolic Flight Path	5
2-2 Weld Pool Support Geometry	9
2-3 Welding Orientations	10
3-1 Enhanced Gravity Laser Materials Processing System	12
3-2 Block Diagram of Modules	15
3-3 Centrifuge Motion Package	16
3-4 Flywheel Support Flange	17
3-5 Centrifuge Gas Delivery Inlet and Support System	18
3-6 Cross-section of Shaft	19
3-7 Centrifuge Support Block	20
3-8 Resultant Gravity Vector	21
3-9 Process Module	23
3-10 Central Turning Mirror Assembly	24
3-11 Mirror Holder (Rear View)	25
3-12 Centrifuge Arm	26
3-13 Processing Box	27
3-14 Lens Holder	28
3-15 Sample Holder	29
3-16 Photodiode (PhD) Locations and General Field of View	31

LIST OF FIGURES

Figure	Page
3-17 CCD Camera	32
3-18 Process Module Acceleration Axes	33
3-19 Block Diagram of Power Supply Circuit	35
3-20 Safety Enclosure	38
5-1 Typical Accelerometer Data	51
5-2 Typical Photodiode Data	52
5-3 Example of Weld Surface Overview	57
5-4 Example of Weld Cross Section	58
5-5 Schematic of Weld Measurements	58
5-6 Max/Min Weld Diameter vs. G-level	59
5-7 Weld Depth vs. G-level	60
5-8 Pore Location vs. G-level	61
5-9 Overbead Height vs. G-level with Standard Deviation (overview)	62
5-10 Nailhead Depth vs. G-level (Leading Edge)	62
5-11 Nailhead Depth vs. G-level (Trailing Edge)	63
5-12 Width at Half Depth vs. G-level	64
5-13 Depth/Width Ratio vs. G-level	64
5-14 Area of Melt Cross Section vs. G-level	65

NOMENCLATURE

EGLMP	Enhanced Gravity Laser Materials Processing System
g_e	earth normal gravity (9.81 m/s ² or 32.2 ft/s ²)
g_r	resultant gravity
a_r	radial acceleration
θ	angle from horizontal

Chapter 1: Introduction

With the advent of the space age, great interest has arisen concerning the effect of gravity on materials processing. In particular, the welding or joining of materials in a microgravity environment has received interest in recent years because of its importance to long-term habitation of space (i.e., space station construction and repair). In view of this importance, several microgravity welding experiments have been conducted to determine if welds produced in microgravity differ from those produced under earth normal (one g_e or 9.81 m/s^2) conditions.

The effect of gravity on welding is not only of interest for space development, however. It is also significant in earth normal conditions. This becomes especially apparent in applications such as pipe welding where the orientation of the laser weld changes with respect to gravity during processing.

In earth normal conditions, separating the effects of gravity from other forces such as surface tension and vapor pressure is difficult. The traditional approach is to minimize the gravitational force by performing experiments in microgravity conditions such as those found in drop towers, NASA's KC 135, and experiments aboard spacecraft. All of these methods allow the experimenter to create scenarios in which the dominant influences (e.g., surface tension and viscosity) are independent of gravity. Unfortunately, experimental testing in microgravity conditions is expensive and the necessary equipment is only available from a few sources.

Enhanced gravity* studies provide a viable alternative for researching gravitational effects and improving commercial opportunities in materials processing. These studies provide the researcher with a virtually limitless variability of the gravitational acceleration in an environment which is relatively easy to reproduce. As a result, numerous tests can be performed in which the g-level is altered ($\geq 1g_e$) while all other conditions are held constant. This provides data concerning the effects of gravity on various processes which can be useful for validating computational models. These models, if accurate, enable researchers working in microgravity conditions to better anticipate their results when planning their experiments and use the available flight time and space more efficiently.

Centrifuges are an effective method for conducting enhanced gravity experiments since they can provide a number of gravitational ranges and orientations. In addition, centrifuges can be built to hold different amounts of material in various configurations from tiny centrifuges carrying fixed samples to larger models with the ability for sample motion. More importantly, perhaps, most researchers and manufacturers can afford to build and maintain at least a small one as they can often be modified from existing systems for work in materials processing.

Centrifuge studies of enhanced gravity materials processing are complimented by using lasers as the heat source. Lasers provide a high energy density, non-contact

* Since work in environments with greater than $1g_e$ has generally been carried out in one of two regimes ($\leq 50g_e$ or $\geq 10,000g_e$) in which the phenomenon involved appear to be entirely different [1], $50g_e$ or less will be referred to as enhanced gravity while the term high gravity will be reserved for environments with greater than $50g_e$.

heat source, and, since the laser beam is simply concentrated coherent light, the laser itself can remain stationary while its energy is brought to the workpiece through the use of moving optics. This makes even compact centrifuges useful for experiments since the centrifuge does not need to be large enough to accommodate the heat source.

Thus, a cost effective centrifuge for enhanced gravity research will make it possible to simplify and expand research opportunities and spread technological or commercial advances to a larger portion of the population. Industry, too, can gain access to centrifuges and take advantage of any benefits of enhanced gravity manufacturing techniques. Enhanced gravity laser materials processing will have a great impact on the knowledge and resources of both the scientific and business communities.

Chapter 2: Background

Some of the first attempts to understand the effects of gravity on welding involved microgravity welding studies performed by the United States in NASA's Skylab M551 metals melting experiment. In this study, electron beam welding was performed on variable thickness disks of 2219-T87 aluminum alloy, 304L stainless steel, and commercially pure Tantalum [2 - 5]. Each disk was rotated under the beam to produce a linear speed of approximately 1.61 cm/s. The beam was then defocused and allowed to dwell on a thick section of each disk to create a large molten pool. This study showed that microgravity welding was possible. In addition, a comparison of the microstructures of the earth based ($1g_e$) and Skylab (μg_e) samples revealed several interesting differences. The microgravity sample contained a fine grained equiaxed zone near the base material which transitioned to a columnar zone of elongated grains and, finally, to large equiaxed grains in the middle. The earth based sample, in contrast, lacks the central equiaxed zone, being comprised of a fine grained equiaxed zone near the base material which transitions to a columnar zone. The columnar zone curves to follow the heat flow as the centerline is approached. Differences were also noted in the weld size and shape, segregation, and the dwell penetration between the flight and ground samples. However, interpretation of the data was inconclusive as neither electron beam power nor disk traverse speed were well known or controlled for the Skylab samples.

The Soviets also performed studies of low gravity welding in space. One of their major investigations involved the "Vulkan" electron beam welding setup [6, 7]. The experiments included materials such as stainless steel, aluminum, and titanium between 1.5mm and 3mm thick. An analysis of the results indicated that weightlessness did not strongly influence the process weld seam formation or the depth of melting. However, much greater porosity (up to 40 times) was noted in the microgravity welds. Similar results were obtained in electron beam experiments flown onboard the Soviet TY-104 flying laboratory.

Although several experiments regarding the effect of microgravity on welding have been performed in space (ie. Skylab), the majority of the research has been conducted onboard aircraft. Some of these aircraft include NASA's KC-135, the Japanese MU-300*, and the Soviet TY-104 flying laboratory. The aircraft are flown in a parabolic trajectory to produce between 20 and 30 seconds of microgravity (gravity levels from less than $0.01g_e$ to around $0.1g_e$ are typical) during each parabola. This trajectory also permits the study of enhanced gravity on welding as periods of higher gravity are produced on either side of the parabola (Figure 2-1).

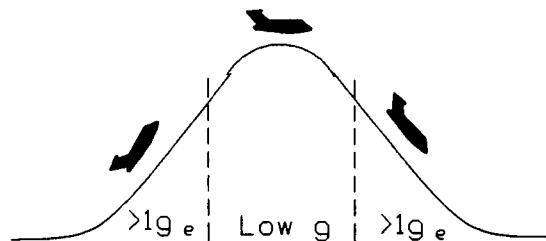


Figure 2-1. Parabolic Flight Path

* Owned and operated by Mitsubishi Heavy Industries.

One of the experiments performed onboard the KC-135 was Kaukler and Workman's study of laser welding of thin foil (0.127mm) 304 stainless steel [8, 9]. This study used an 18W Nd:YAG laser. A simulated space environment was provided by a small vacuum chamber and the microgravity (0.01g_e) portion of the flight path. For comparative purposes, welding was also done during the enhanced gravity (1.8g_e) segment of the flight. To insure that the gravity level was the only variable between the microgravity and enhanced gravity sections of a given sample, the welds were conducted such that the transition between enhanced gravity and microgravity occurred within the weld. Comparisons between the two sections of the welds exposed differences in both the structure and microstructure. First, the enhanced gravity section showed a suppression of the weld surface ripples when compared to the microgravity welds. The enhanced gravity sections also showed a randomly oriented equiaxed dendritic structure while the microgravity sections had a radially emphasized columnar structure. This contradicts the earlier Skylab studies in which the microgravity welds produced a central equiaxed dendritic structure. Differences in overall weld shape were also noted. Although not every sample showed changes in weld bead structure in sympathy with g-level changes, the partial penetration welds, in general, had an increase in weld penetration and a decrease in weld bead width in microgravity. Analysis was also made of the weld and heat affected zone (HAZ) hardness and (for those samples which showed no width variations) the change in tensile strength. The conclusions reached by this study were

that a reduction in convective flow in microgravity produced harder, stronger, deeper penetration welds with more surface rippling than $1g_e$ welds.

Researchers Tandon and Wang also participated in researching the effects of gravity on laser welding onboard the KC-135 [10]. They were involved in welding 1.0mm thick 316 stainless steel using a 120W CW CO_2 laser at a power of 70W and a speed of 0.01 cm/sec. Like Workman and Kaukler, Tandon and Wang found that the depth of penetration of the weld increased in microgravity. They also noted an increase in 'burning' or smoking during the microgravity experiment. This was attributed to an increase in the coupling coefficient at the sample during microgravity. The distribution of elements within the weld was also found to change as the gravity level increased from low to high causing an increase in the iron concentration and a decrease in the concentration of nickel.

Laser keyhole welding of plastics under various g levels has also been researched onboard the KC-135. Mueller [11] investigated penetration welding of optically clear polypropylene and polyethylene to prove the viability of keyhole welding in a low g_e environment and to observe changes in weld keyhole morphology with gravity. The investigation included both bead on plate and lap welding. The purpose of the bead on plate welding was to study the effects of changing the gravity level on keyhole shape and weld penetration while the lap welding allowed the effects on weld strength to be tested. This study found that the keyhole became narrower under low g_e while weld depletion (loss of material from the weld pool) was reduced. However, weld penetration in low g_e was similar to that obtained in one and higher

g_e . This lead to the conclusion that while gravity can substantially change keyhole shape, it plays a relatively small role in the determination of weld penetration. Additional findings of this study were that keyhole welding in low g_e is viable and stable and weld strength was not effected by the changes in gravity level.

In addition to the research performed on the KC-135 flights, microgravity studies have been conducted on Japan's MU-300. Masubuchi and Nayama [12] investigated gas tungsten arc welding of 2mm thick aluminum and 2219 aluminum alloy in an argon atmosphere under low gravity conditions. Although the arc plasma was wider than the plasma normally produced, Masubuchi and Nayama did not believe that the difference was significant enough to have affected the weld. The results of their research indicated that the weld bead made in low gravity contained none of the ripple lines which normally exist in $1g_e$ welds, a finding which contradicts Workman and Kaukler who stated that microgravity welds have more ripple lines than $1g_e$ welds. Masubuchi and Nayama also found that low gravity welds tend to contain a larger number of smaller pores spread uniformly over the weld. This is in comparison to $1g_e$ welds in which a small number of large pores tend to segregate near the bottom of the weld pool. However, they believed that some of the differences in pore size, number, and location could have been due to the change in the gas delivery systems between the ground and microgravity apparati from a flow system ($1g_e$) to a static inert atmosphere (μg_e) which may have created impurities near the arc. The impurities could have caused additional pores to form.

In addition to the studies which indicated that changes in gravity level affect the weld formation, welding experiments conducted by Foley and Banas [13] have shown that

the gravitational orientation, or attitude of the workpiece (horizontal, vertical, or upside down), also affects the weld by influencing such phenomena as fluid flow, molten pool stability, and weld bead support. These factors, in turn, influence the size and shape of the weld itself. For example, the study has shown that the maximum supportable weld bead width is a direct function of the magnitude of the gravitational force applied to the weld. An equation was given by Foley and Banas relating the maximum supportable weld pool area to the surface tension of the supporting meniscus. (It should be noted that in the normal, flat welding position, only the column directly above the root bead needs to be supported [Figure 2-2].)

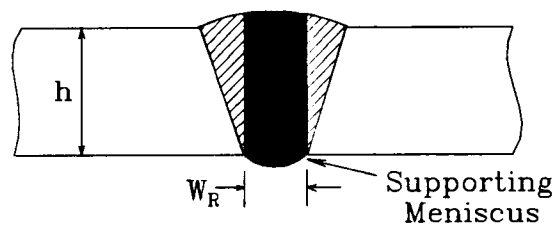


Figure 2-2. Weld Pool Support Geometry

The equation was then solved for the theoretical maximum width of the root bead.

The resultant equation,

$$W_R \leq \frac{2\sigma}{gh\rho}$$

where σ is the surface tension of the material, ρ is the material density, g is the resultant gravity, W_R is the width of the root bead, and h is the material thickness, shows that the maximum supportable beadwidth decreases as the effective magnitude of gravity increases.

Another aspect of out-of-position welding which may be applicable to enhanced gravity research is the effect on weld width and depth of fluid flow out of the keyhole cavity. In cases where such flow was inhibited, e.g. with vertical-down welds (Figure 2-3), the keyhole width increased and penetration depth decreased. However, if fluid flow out of the weld zone is promoted as in vertical-up welding, the weld tends to have a deeper penetration and narrower bead width. Overhead and flat welds did not show much variation since gravity did not interfere with the normal flow of material out of the keyhole for those positions.

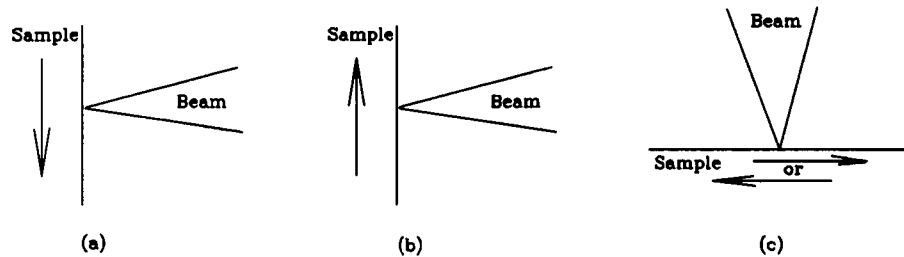


Figure 2-3. Welding Orientations: (a) Vertical Up, (b) Vertical Down, (c) Flat

The effect of gravitational orientation on weld penetration was also investigated by Mueller [11]. This investigation showed that flat welds had the least amount of penetration, sample up (same as Foley's vertical down) welding showed better penetration and sample down (vertical up) welding showed the most penetration. These results agreed with those of Foley and Banas.

Mueller also investigated the effect of different gravity levels on the various orientations. He found that for welds performed on samples moving across or up (weld pool beside or above the keyhole), the keyhole depth decreased with an increase

in g-level. For welds performed on samples moving down (weld pool below the keyhole), the keyhole depth increased slightly with an increase in g-level. The change in depth in these cases was not linear. It showed a greater change from low g_c to one g_c than from one g_c to higher g_c (1.6 to 1.8 g_c).

What effect does enhanced gravity have on solidification? According to Dr. L. Regel [14], enhanced gravity should "intensify convection and sedimentation and their influence on solidification." Increasing convection can affect the cooling rate and concentration profiles in the sample which, in turn, affects the dendritic structure as well as the overall grain structure. Specifically, solidification experiments have shown that the dendrite structure coarsens and the overall composition becomes more homogeneous until a critical gravity level is reached. After the critical level is reached, the dendritic structure becomes finer and the composition becomes less homogeneous. The grain size is also affected, becoming more coarse until the critical g-level at which point it usually becomes finer than the structure found at 1 g_c [14, 15]. One possible explanation for these results is that unsteady convection is stabilized by increasing the gravity level above a critical value. However, it has been found by Weber et al. [16] that at least some of the effects observed in enhanced gravity solidification of crystals were most likely caused by the effect of the coriolis force** on convection and this force is produced by centrifugation rather than by the increase in gravity.

** The coriolis force is an inertial force present only in a rotating system which appears to deflect a moving particle at right angles to its direction of motion.

Chapter 3: Enhanced Gravity Laser Materials Processing System

System Description

The goal of the enhanced gravity laser materials processing (EGLMP) system (Figure 3-1) is to provide an environment in which laser welding can be performed under enhanced gravity conditions while being monitored with standard process diagnostics. The system centrifuge needs to be useable with high power lasers (up to 3kW), produce both bead-on-plate and spot welds, and provide gravity levels between 2 and 10g_e. The diagnostics should monitor both the sample and the system.

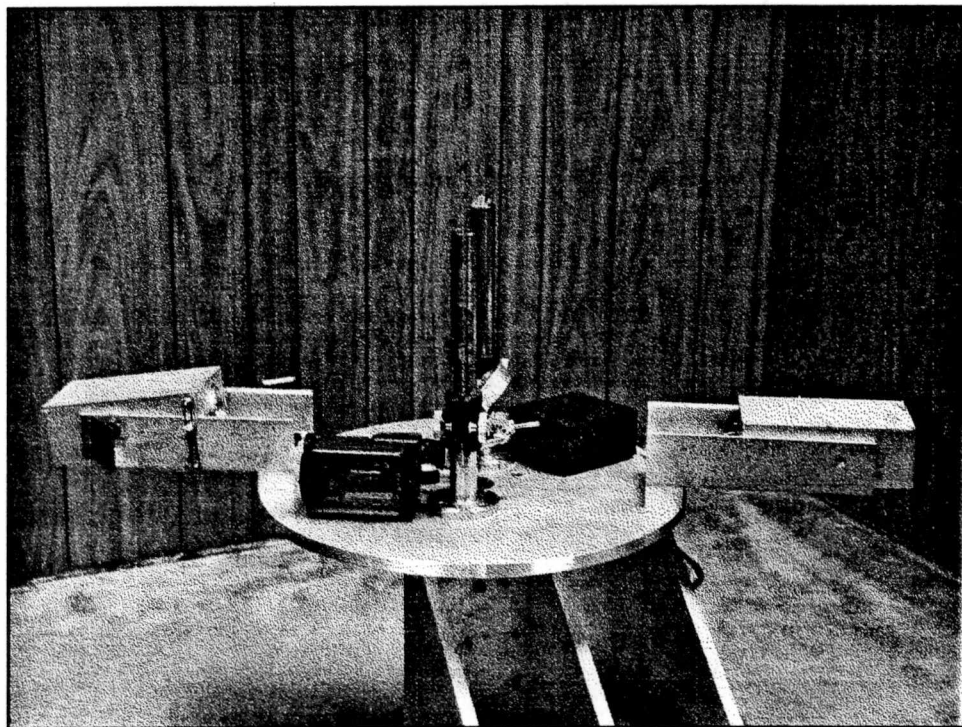


Figure 3-1. Enhanced Gravity Laser Materials Processing System

Design Specifications and Constraints

The following specifications are necessary for meeting the design goal:

- 1) process power source: CO₂ (10.6 micron) or Nd:YAG (1.06 micron) laser
- 2) a system of rotating optics to deliver the laser beam to the process module
- 3) rotation rates between 25 and 150 rpm
- 4) g level: 2g_e to 10g_e capable of being fixed normal to the sample surface
- 5) radius to the sample: 0.508m (20") from the centrifuge axis of rotation
- 6) process gas (e.g. Ar, He): delivered to the sample, rate adjustable
- 7) sample size: 10mm X 10mm (0.39" X 0.39") minimum to
25mm X 50mm (0.98" X 1.96") maximum
- 8) sample linear motion: 25mm/s (0.98 inch/s)
- 9) sample travel: up to 25mm (0.98")
- 10) sample diagnostics:
 - a. thermocouples for measuring the surface temperature of the sample,
 - b. photodiodes for measuring the plasma intensity both from the side (plasma signal isolated) and from the top (surface and plasma signals combined), and
 - c. a CCD camera for observing the size and location of the plasma.
- 11) system diagnostics:
 - a. accelerometers for measuring the resultant gravity vector at the sample.
- 12) data acquisition and storage: compact, battery operated system able to
withstand up to 10 g_e of gravitational loading and download the acquired data
to a host computer. Minimum sampling rates: photodiodes - 1 kHz,

thermocouples and accelerometers - 200 Hz. Storage capacity: sufficient to contain the data acquisition programs and five seconds of data.

In addition to the design specifications, several design constraints were identified.

The most important design constraint was safety. Additional constraints include

- 1) a radius less than 0.711m (28") to fit within the existing laser work area,
- 2) a modular design for maximum versatility,
- 3) a minimal number of optical elements to reduce the loss of laser power due to absorption,
- 4) low system vibrations/wobble,
- 5) easy adjustment for different gravity levels,
- 6) transportable design for removal from workstation when not in use, and
- 7) currently available or 'off the shelf' parts (where possible) to reduce cost and facilitate future maintenance and repair.

System Description

The EGLMP system is divided into seven major sections, or modules*. These are shown schematically in Figure 3-2. The modular approach simplifies design and construction and increases the versatility since a single module can be replaced or altered without changing the entire system. The modules consist of the centrifuge motion package module; the process module; the diagnostics module; the data acquisition and storage module; the power supply module; the system control module; and the safety enclosure. They are discussed individually in the following sections.

*More detailed design specifications are available in Appendix 2.

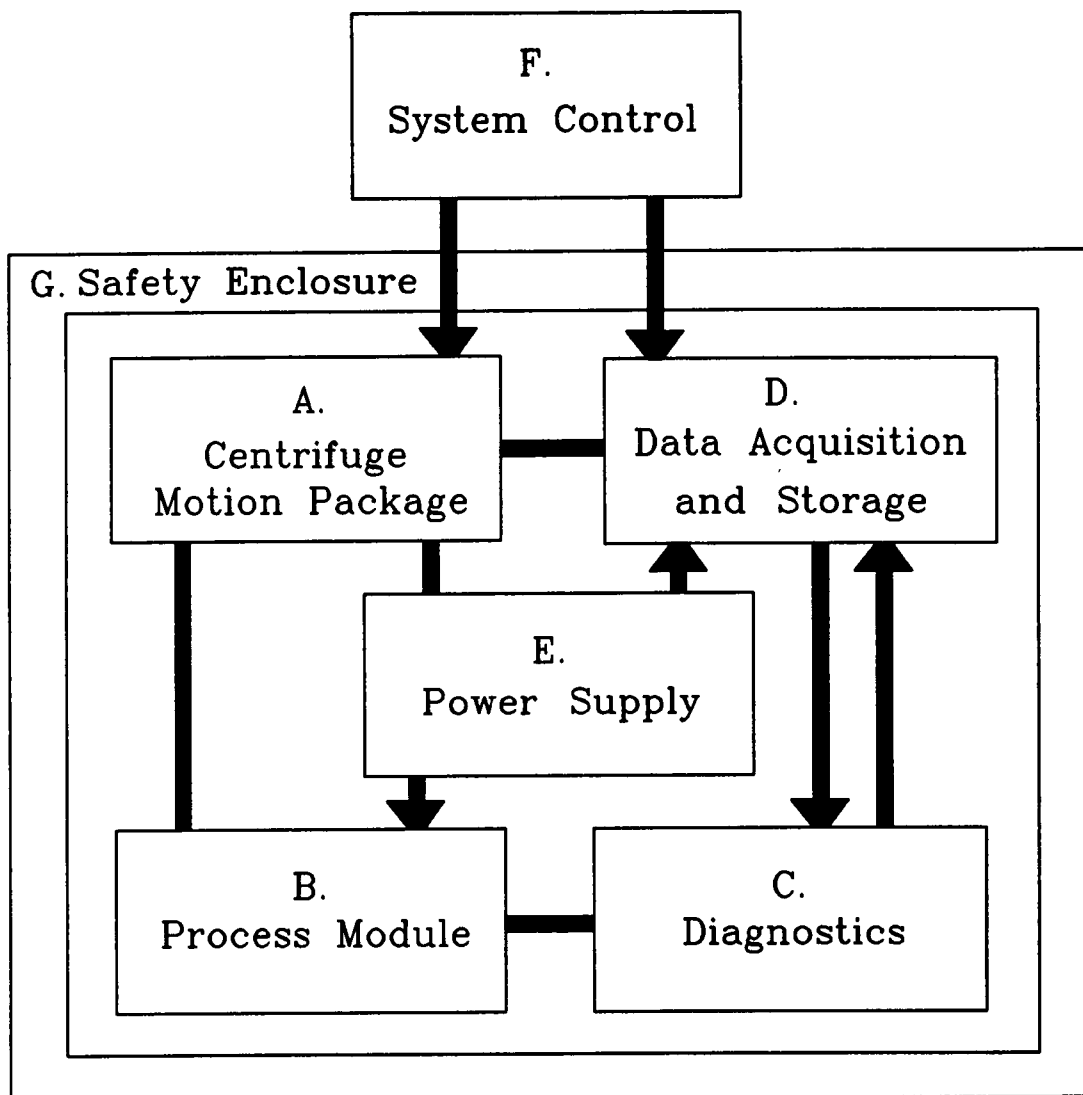


Figure 3-2. Block Diagram of Modules (The dark arrowed lines indicate data or power flow while the dark non-arrowed lines indicate physical connection.)

A: Centrifuge Motion Package Module

The centrifuge motion package provides resultant gravity levels between two and ten g_e at a radial distance of 50.8cm (20") from the axis of rotation of the centrifuge. This module consists of the flywheel support flange, the rotation motor, the gas delivery inlet and support system and the support block. Figure 3-3 shows the location of these components.

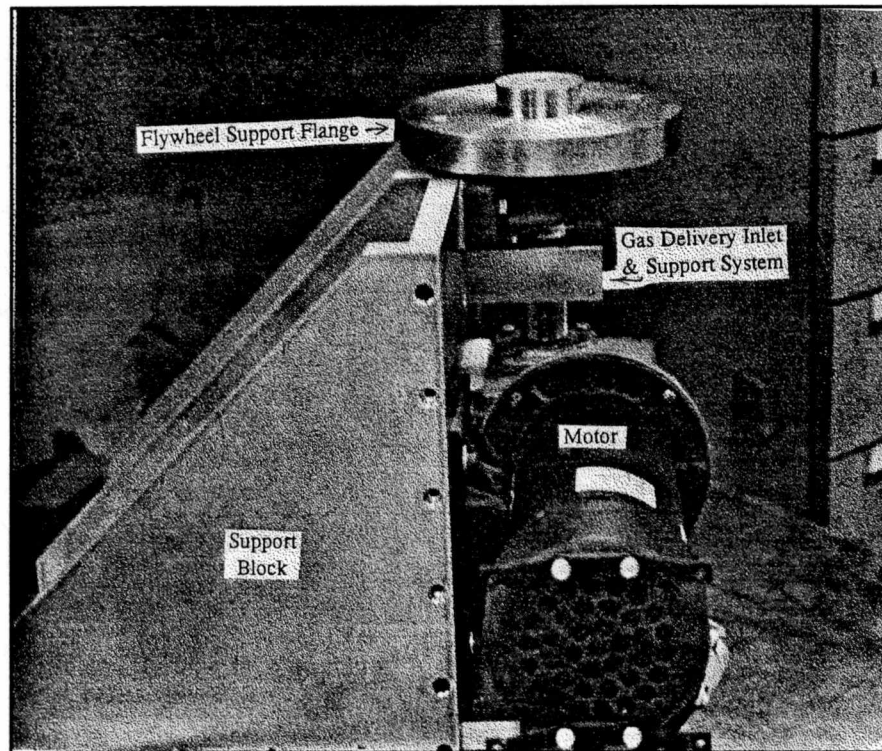


Figure 3-3. Centrifuge Motion Package

A.1 Flywheel Support Flange

The flywheel support flange is a 20.3cm (8") diameter disk with a raised central disk (Figure 3-4) which fits into the centrifuge flywheel. The flange can be bolted securely to the centrifuge flywheel with four 3/8" bolts. A 1/8" tapped hole through the top of the flywheel support flange provides access to the process gas. The central bore cut in the bottom of the support flange allows it to be press-fit to the centrifuge shaft.

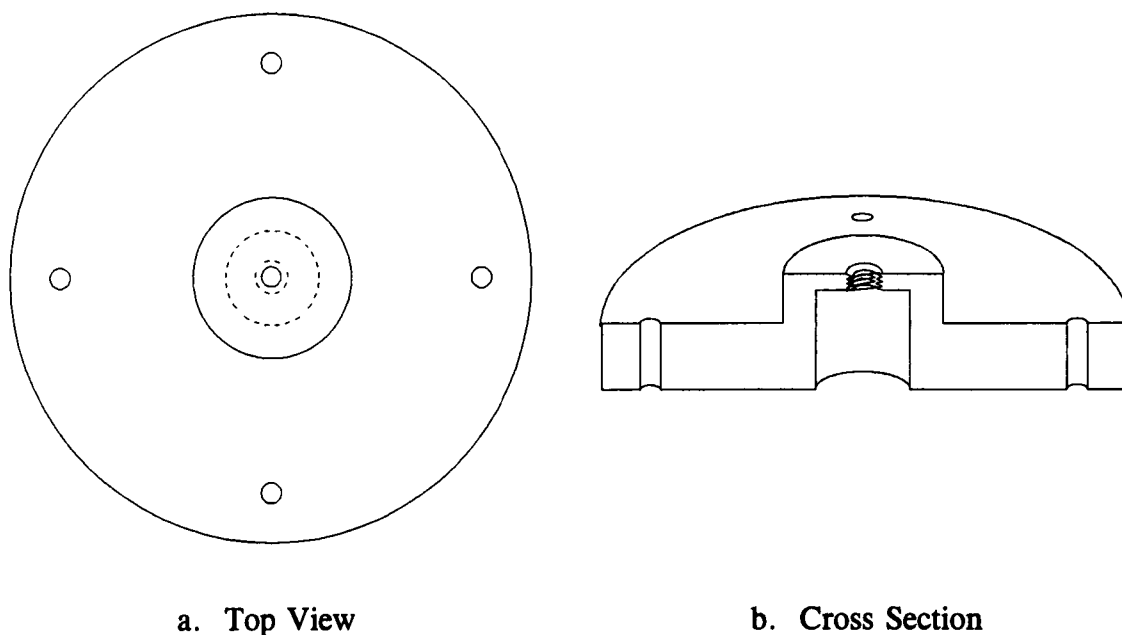


Figure 3-4. Flywheel Support Flange

A.2 Rotation Motor

A 3/4 HP variable speed motor with an electronic controller was combined with a right angle 10:1 speed reducer to provide centrifuge rotation rates ranging from 50 to 150 rpm. This rpm range provides a resultant acceleration from two to

ten g_c at a radial distance of 50.8cm (20") from the axis of rotation of the centrifuge. The configuration allows both clockwise and counterclockwise rotation of the centrifuge within the specified rpm range via the controller.

A.3 Centrifuge Gas Delivery Inlet and Support System

This system includes a shaft, shaft support bearing and gas delivery inlet (Figure 3-5). The shaft connects the motor to the centrifuge flywheel, provides gas delivery to the centrifuge through the gas delivery inlet and stabilizes the flywheel support flange with the shaft support bearing.

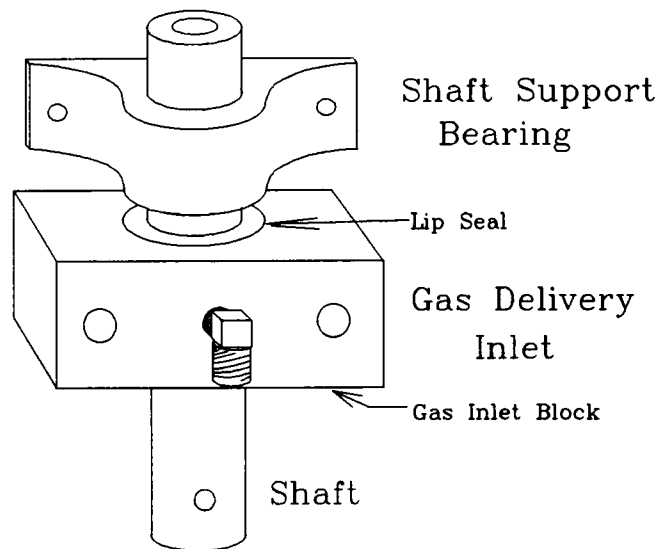


Figure 3-5. Centrifuge Gas Delivery Inlet and Support System.

The shaft is composed of a 3.8cm (1.5") diameter steel bar which has been bored out on each end (Figure 3-6). The bottom bore is sized to allow the two shafts to be coupled together. A second hole is drilled through both shafts for securing

them with a shear pin. This pin will shear if the centrifuge arm impacts a solid object with more than approximately 12 pounds of force.

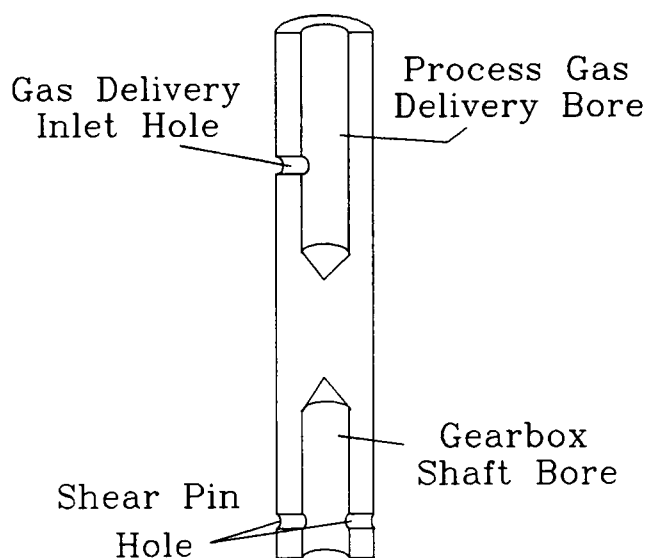


Figure 3-6. Cross-section of Shaft.

The top bore of the shaft intersects with the gas delivery inlet hole 10.8cm (4.25") from the top of the shaft. This provides a method for bringing the process gas from a stationary outside location onto the rotating centrifuge. The gas delivery hole is enclosed by the gas inlet block which is bolted securely to the centrifuge support block. A right angle, male to 1/4" hose connector is threaded into a 1/4" hole and connects the process gas delivery hose to the gas inlet. A lip seal was provided on each end of the gas inlet block to prevent gas leakage while allowing free rotation of the shaft.

The shaft support bearing provides free rotation of the shaft while holding the shaft in the proper orientation and preventing side to side motion or wobble. A Browning prelubricated precision ball bearing pillow block with a 3.81cm (1.5") inner diameter is used for this purpose. It is connected to the support block by two bolts and set to the correct distance from the support block with a 0.4cm (0.16") spacer.

A.4 Support Block

The centrifuge support block (Figure 3-7) provides a secure mount for the motor, gear box, shaft support bearing and the gas inlet block. It also acts as a counterbalance for the motor and gear box that will inhibit any inherent system vibrations.

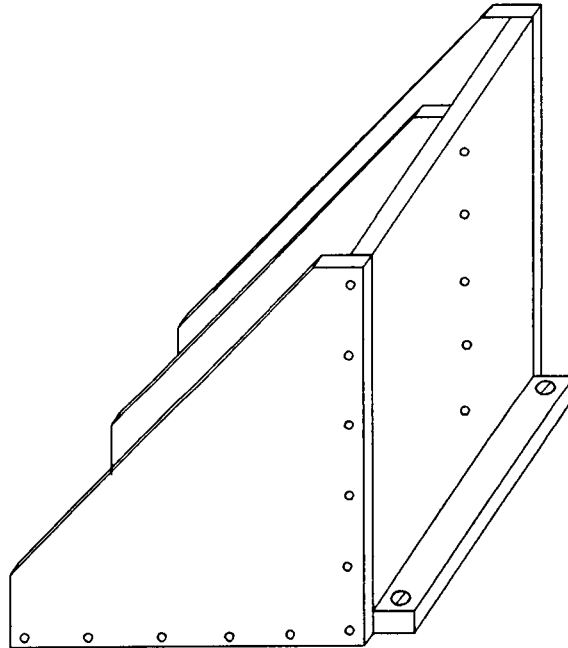


Figure 3-7. Centrifuge Support Block

The support block has overall dimensions of 35.6cm x 39.4cm x 33.7cm (14" x 15.5" x 13.25") and is composed of five plates of 19mm (3/4") thick aluminum joined by 1/4" countersunk bolts. The bottom plate has six mounting holes which match the hole pattern on the available laser workstation. The back plate has additional bolt holes for mounting the gas inlet block, shaft support bearing, motor, and right angle gearbox.

B: Process Module

The process module is responsible for delivering the focused laser beam normal to the sample surface while keeping the sample surface normal to the resultant gravity vector. To keep the sample surface normal to the resultant gravity vector, it is important to know both the magnitude and direction of the vector.

The resultant gravity level at the sample (g_r) is produced by a combination of the earth's gravity (g_e) and the radial acceleration of the centrifuge (a_r) (Figure 3-8).

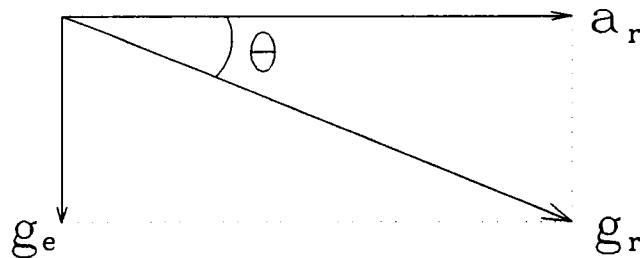


Figure 3-8. Resultant Gravity Vector

The resultant gravity vector has magnitude g_r and direction θ given by

$$g_r = \sqrt{g_e^2 + a_r^2} ,$$

and

$$\theta = \tan^{-1} \frac{g_e}{a_r} ,$$

where g_e is the magnitude of the earth's gravity vector (9.81 m/s² or 32.2 ft/s²), a_r is the magnitude of the centrifuge's radial acceleration vector, g_r is the magnitude of the resultant gravity vector, and θ is the angle of the resultant gravity vector from the horizontal. Table 3-1 contains the centrifuge height and angle settings for resultant gravity levels between $2g_e$ and $10g_e$ at $1g_e$ intervals.

Table 3-1. Centrifuge Parameters for Selected Resultant Gravity Levels

Resultant Gravity Level	Angle of Process Module (in degrees from horizontal)	Central Mirror Height (cm)	Centrifuge RPM
$2g_e$	30.0	36.95	55.2
$3g_e$	19.5	25.61	70.6
$4g_e$	14.5	20.76	82.6
$5g_e$	11.5	17.96	92.9
$6g_e$	9.6	16.21	102.1
$7g_e$	8.2	14.94	110.5
$8g_e$	7.2	14.04	118.3
$9g_e$	6.4	13.32	125.5
$10g_e$	5.7	12.69	132.4

The process module consists of the centrifuge flywheel, the central turning mirror assembly, the process module adjustment block, the centrifuge arm, the processing box and the counterweight (Figure 3-9).

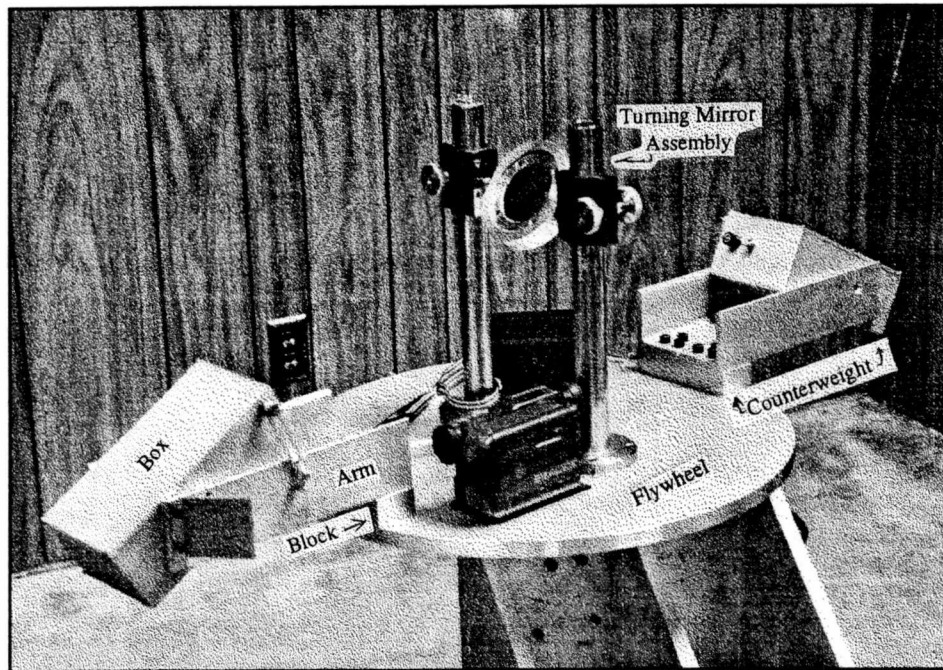


Figure 3-9. Process Module

B.1 Centrifuge Flywheel

The centrifuge flywheel provides a surface for connecting other modules as well as the remaining components of the process module. It also provides a large, balanced mass to counteract the tendency of incorrectly placed modules to unbalance the system and cause wobble or vibration. The centrifuge flywheel is a 61cm (24") diameter, 1.9cm (3/4") thick disk of aluminum with a central hole 6.35cm (2 1/2") in

diameter and has four 3/8" holes for connecting the flywheel to the flywheel support flange.

B.2 Central Turning Mirror Assembly

The central turning mirror assembly (Figure 3-10) directs the centrally located, stationary beam from the processing laser to the moving process module and keeps the beam stationary with respect to it. The primary components of the central turning mirror assembly are the mirror, the mirror holder and the mirror support rods.

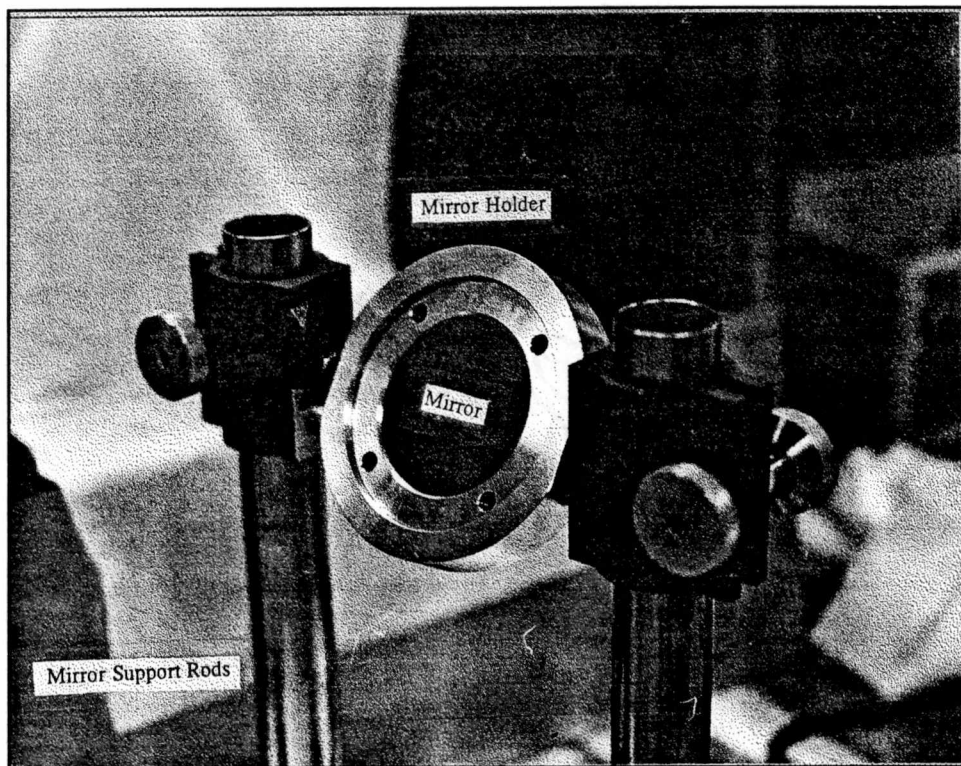


Figure 3-10. Central Turning Mirror Assembly

B.2.1 Mirror. The mirror is a 76.2mm (3") diameter gold plated copper mirror. This type of mirror was chosen because it is reflective for both CO₂ and Nd:YAG wavelengths.

B.2.2 Mirror Holder. The mirror holder (Figure 3-11) holds the mirror at orientations from vertical to nearly horizontal with both side to side and up/down fine tilt adjustments.

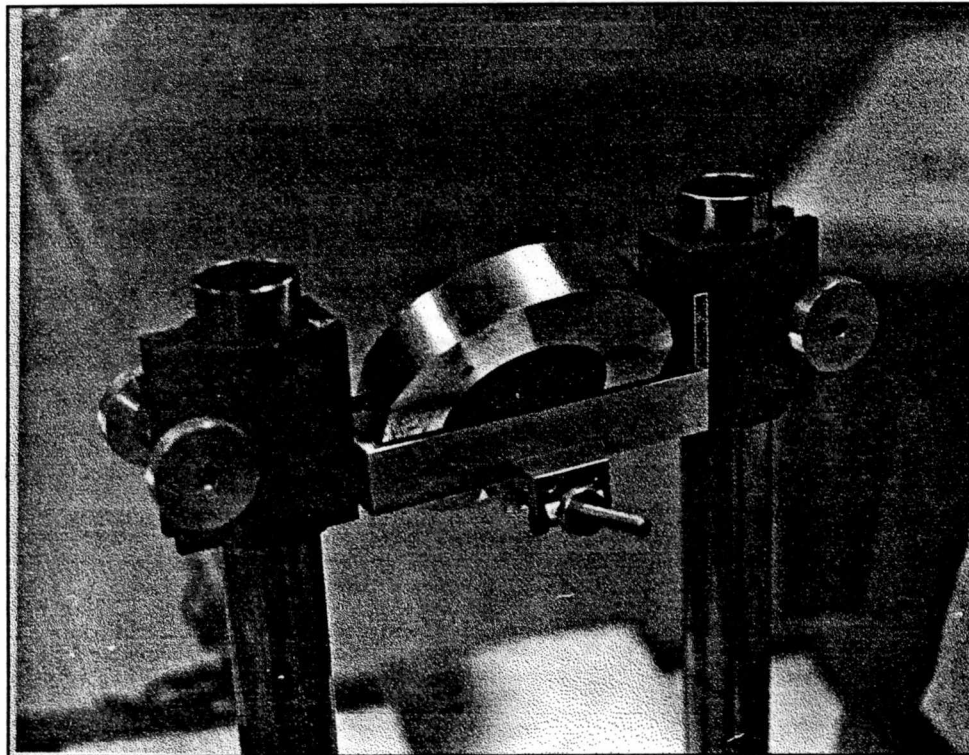


Figure 3-11. Mirror Holder (Rear View)

B.2.3 Mirror Support Rods. The mirror support rods provide a steady mount for the central mirror and mirror holder at heights up to 38.1cm (15") above the centrifuge flywheel.

B.3 Centrifuge Arm

The centrifuge arm (Figure 3-12) is designed to place the processing box at the correct horizontal distance from the flywheel center. The arm is 30.5cm (12") long by 7.6cm (3") high. Holes in the base of one end enable it to bolt to the flywheel while holes in the sides of the other end hold the processing box. In addition, a 45° mirror is attached near the processing box for use with the process diagnostics (camcorder).

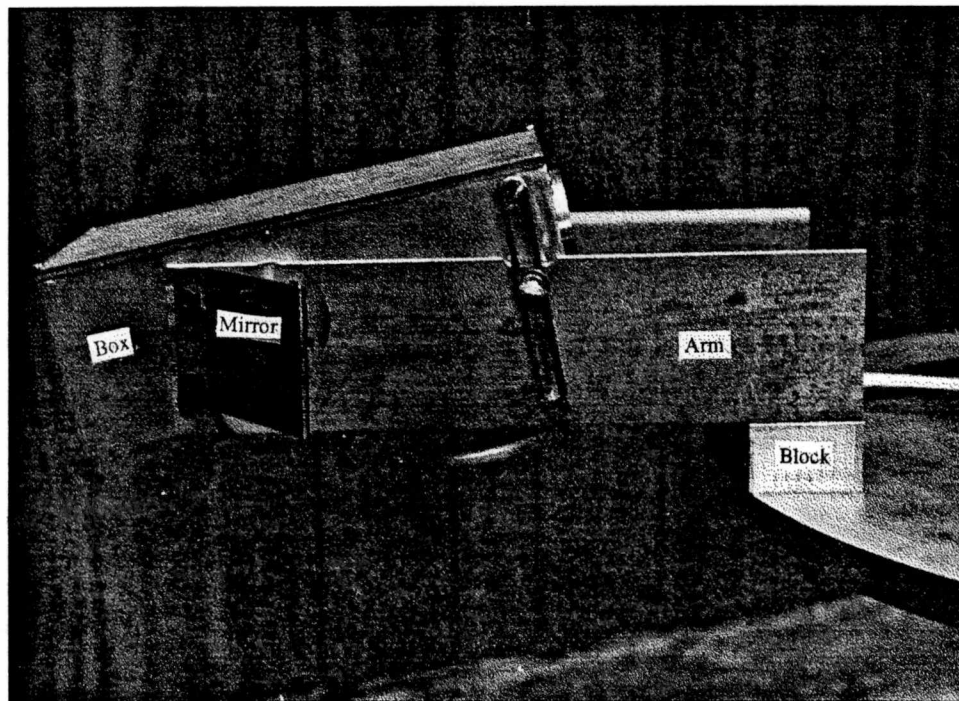


Figure 3-12. Centrifuge Arm

B.4 Process Module Adjustment Block

The block (Figure 3-12) is a 15.3cm (6") x 5.1cm (2") x 3.2cm (1.25") thick piece of aluminum which is used to align the optical centerline of the processing box with that of the central mirror.

B.5 Processing Box

The processing box (Figure 3-13) consists of the housing in which the welding is accomplished, the beam focusing lens, the sample holder and positioning system, and the sample motor.

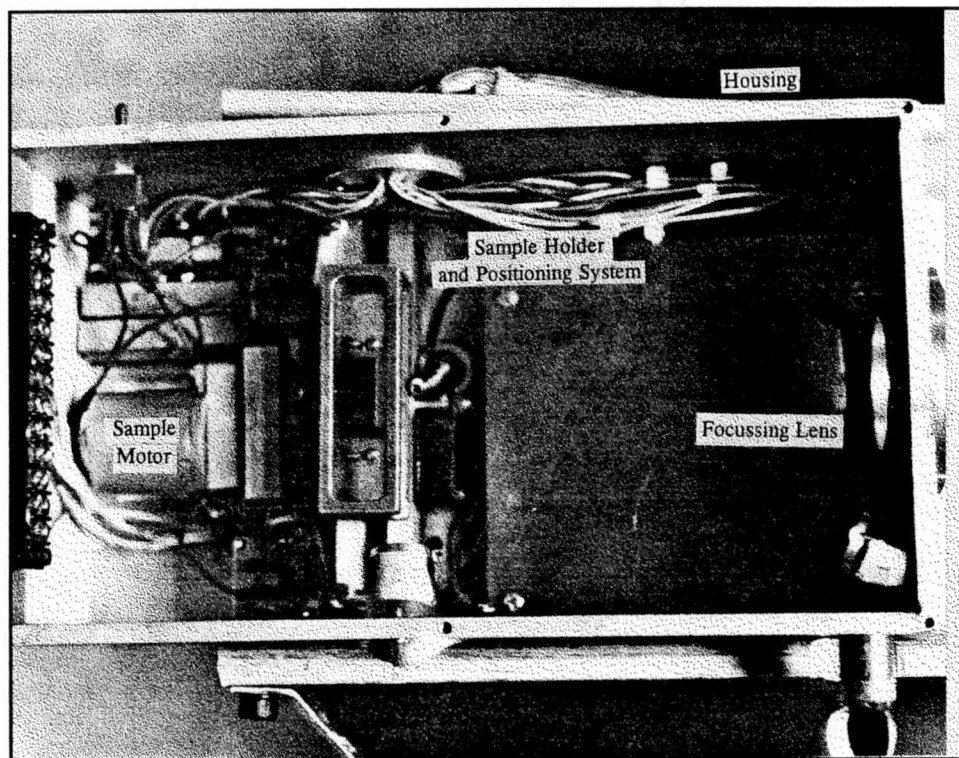


Figure 3-13. Processing Box

B.5.1 Housing. The housing provides an enclosed environment in which the laser beam is focussed onto the sample. It can be tilted to keep the sample surface normal to the resultant gravity vector. Since the focussing lens is contained within the housing, the laser beam stays normal to the sample surface regardless of the angle of the housing and prevents the focussed beam from hitting anything outside of the housing even if the beam becomes misaligned. The housing also holds the diagnostics (except for the CCD camera), the amplification circuits and the gas delivery nozzle.

B.5.2 Beam Focussing Lens. The focusing lens is a 38mm (1.5") diameter, 127mm (5") focal length ZnSe lens. It was chosen to match the focal length of the existing CO₂ laser welding system. The lens can be replaced with a suitable one without any further modification of the process box if other wavelengths are used for processing. Since the lens holder (Figure 3-14) screws into the housing, the focal position of the beam on the sample can be easily altered.

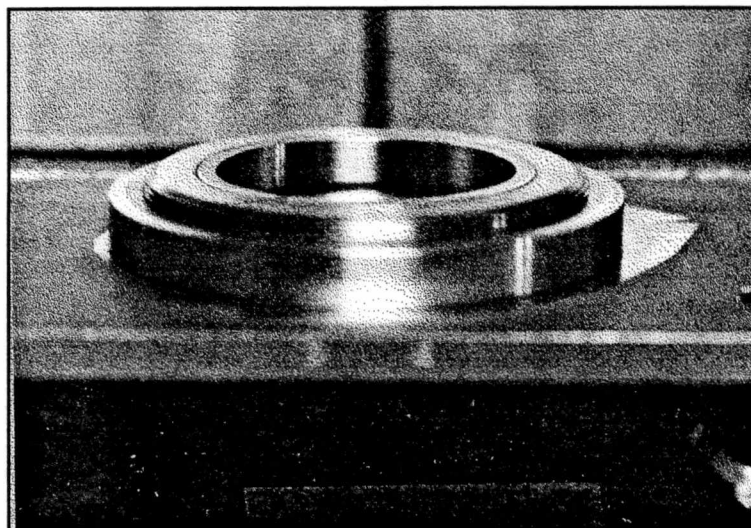


Figure 3-14. Lens Holder

B.5.3 Sample Holder and Positioning System. The sample holder (Figure 3-15) holds samples ranging in size from 20 mm by 20 mm (0.79" x 0.79") to 25mm X 50mm (0.98" x 1.96"). It is positioned such that the pivot point of the housing falls along the front face of the sample to maintain a uniform radial distance to the sample regardless of the angle of the housing. The sample holder is mounted on a precision miniature stage positioning system of the type used for positioning microscope optics. This provides smooth sample translation in a small package. A spacer was added under the stage to align the sample holder with the optical centerline of the focusing lens.

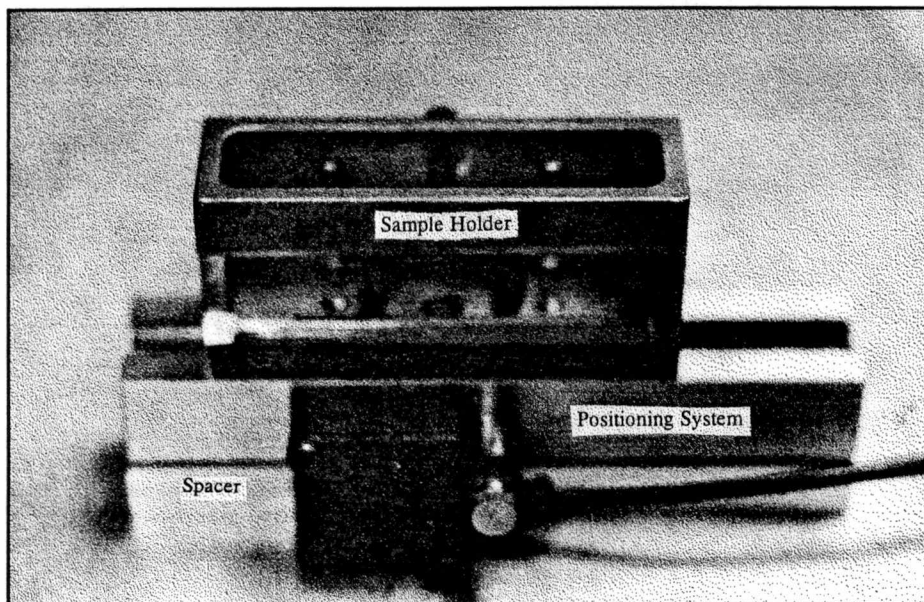


Figure 3-15. Sample Holder

B.5.4 Sample Motor. The sample motor is battery operated and provides linear movement of the sample by moving the miniature stage positioning system. The sample movement rate is set to 25 mm/s (0.9 inch/s) and can be varied slightly by adjusting the current flowing through the sample motor control board (see Appendix 2).

B.6 Counterweight

The counterweight to the process module is mounted opposite the process module on the centrifuge. It consists of an arm, a block, and a box which are nearly identical to the centrifuge arm, the process module adjustment block and the processing box housing, differing only in the hole pattern of the housing. Since the counterweight box contains the power supply, it has bolt holes for securing the power supply and accessing the power switch and fuse but no hole for the focusing lens. There is sufficient space remaining in the counterweight box to store extra weights inside if necessary.

C: Diagnostics Module

The diagnostics are similar to the type currently used by the UTSI Center for Laser Applications (CLA) laser materials processing group. Monitoring of the system's resultant gravity vector near the sample is also provided. The diagnostic tools include thermocouples, photodiodes, a CCD camera, and accelerometers.

C.1 Thermocouples

The system uses type K or chromel(Ni-Cr)/alumel(Ni-Al) thermocouples. These thermocouples have a wide useful range (-200°C to 1250°C). They also have a low standard error ($\pm 2.2^\circ\text{C}$) and good corrosion resistance [Stanford Research Systems 1992-1993 Catalog]. Since there was not enough room for ice point compensators, a 0°C to 100°C temperature sensor was installed at the thermocouple to copper wire junction to allow manual temperature compensation to be made.

C.2 Photodiodes

There are two general purpose photodiodes used for this system. One monitors the sample surface to note the onset of surface heating while the second monitors the weld plasma above the sample. Neutral density filters in the form of exposed photographic film can be used to prevent photodiode saturation. Ring type filter holders are provided for this purpose. Figure 3-16 shows the location of the photodiodes within the process module.

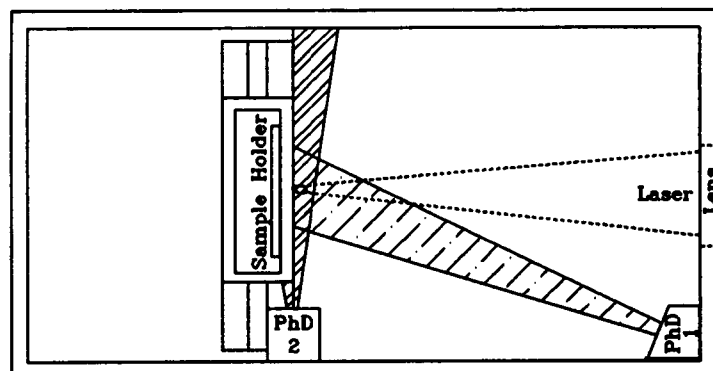


Figure 3-16. Photodiode (PhD) Locations and General Field of View

C.3 CCD Camera

The CCD camera (Figure 3-17) is an 8mm, 1.6 lb G.E. camcorder with 8x telephoto zoom lens and lock on recording. It has an internal monitor which can be rotated up by 90 degrees for easier viewing while mounted on the centrifuge and video in/out so that the picture can be viewed on a full size color television or transferred to another tape. The power supply for the camcorder is supplied by the manufacturer in the form of a rechargeable battery that can power the unit for approximately one hour. A two hour battery is also available. The camcorder provides an all-in-one unit for filming the weld plasma in a compact, easy-to-operate form.



Figure 3-17. CCD Camera

C.4 Accelerometers

Three accelerometers are used to determine the resultant gravity vector in the process module by measuring the component of the vector that falls along each accelerometer's axis. The accelerometers are placed along orthogonal axes so that the vector components can easily be added using vector algebra to obtain the magnitude and direction of the resultant gravity vector with respect to the process module. Unfortunately, it was not possible to place the accelerometers on the sample surface so the value recorded, while close, is not exactly the same as that felt at the sample surface.

IC Sensors model #3145 accelerometers are used for this purpose. This model has internal voltage regulation, amplification, signal conditioning, temperature compensation from -20°C to 85°C and overrange protection. In addition, these accelerometers are damped at around 1kHz. A $\pm 10g_x$ accelerometer measures radial acceleration at the sample surface while $\pm 5g_x$ models measure jitter along the tangential and z axes. The axes are shown in Figure 3-18.

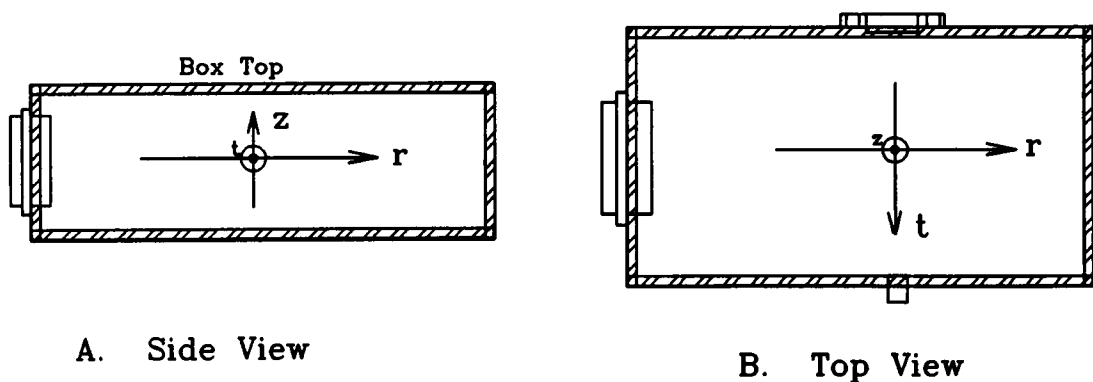


Figure 3-18. Process Module Acceleration Axes

When the accelerometers are combined with the data acquisition and storage module, radial accelerations as small as $0.0024g_c$ can be measured while the tangential and radial axes are capable of measuring g jitter as small as $0.0012g_c$. All three accelerometers are sampled at a frequency of 3.5kHz.

D: Data Acquisition and Storage Module

The data acquisition and storage module provides a self-contained system which can be mounted on the centrifuge during processing. It obtains and stores data from ten different channels for a five second period at a 56 kHz total sampling frequency. To meet these requirements an Ampro Coremodule 286, a DM 406 data acquisition board, a ten channel multiplexer and signal conditioning/amplification circuits were combined. The Coremodule 286, DM406, and multiplexer are contained within the PC/104 enclosure mounted on the centrifuge flywheel while the signal conditioning/amplification circuits are contained on a board within the processing box and protected from splatter or laser backscatter by a brass shield [made of 0.5mm (0.002") brass shimstock].

D.1 Signal Conditioning/Amplification (Circuits)

The signal conditioning/amplification circuits are contained within the process module. They are designed to filter out signals with frequencies higher than those detectable at the given sampling rate. They also provided sufficient amplification to

bring the signals from the diagnostics into the -5V to +5V range. Circuit diagrams are given for each type of diagnostic in Appendix 2.

E: Power Supply Module

Except for the self-power camcorder, the power supply (Figure 3-19) provides all power needed onboard the centrifuge. It consists of a 12V rechargeable battery and voltage regulators. The voltage regulators include $\pm 12V$ and +5V regulators for supplying power to the data acquisition and storage circuitry and a +12V regulator for supplying power to the sample motor.

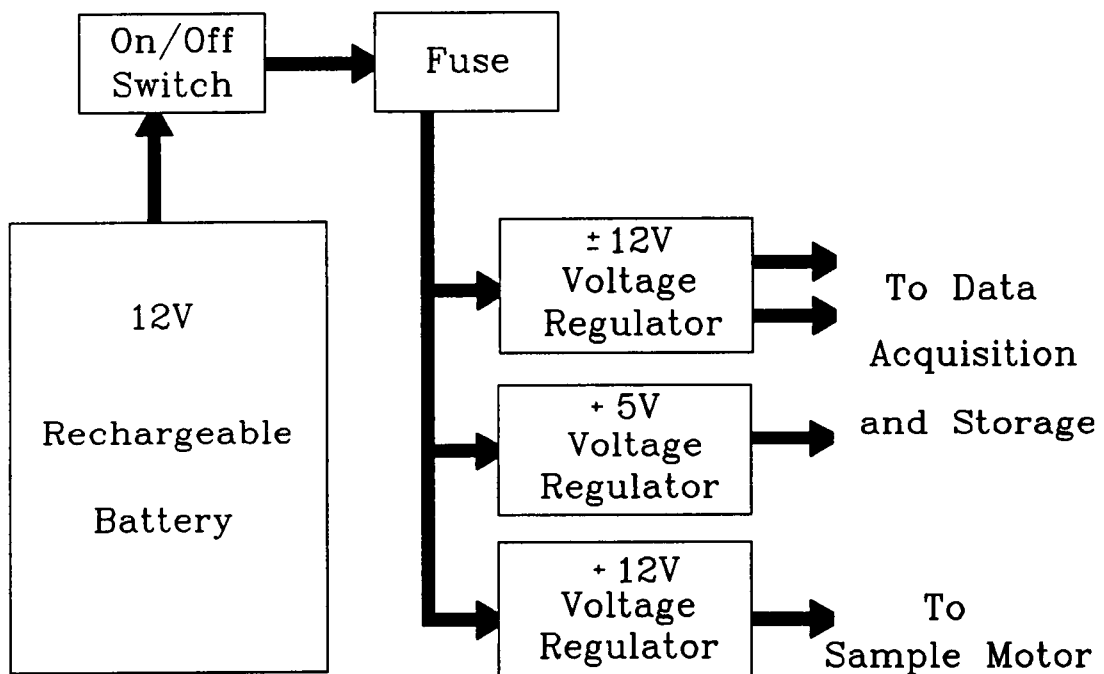


Figure 3-19. Block Diagram of Power Supply Circuit.

F: System Control Module

There are five controls that must be operated in the EGLMP system: the laser/CNC controller, the motor speed controller, the gas flow regulator, the Ampro motherboard on the centrifuge, and the switches on the camcorder.

The laser/CNC controller triggers the start of the sampling phase of the onboard data acquisition program, turns on the cover gas and turns the laser on and off. It triggers the onboard program by activating an IR photodiode with a digital signal accessed from the rear panel of the laser cabinet. The cover gas and laser parameters are regulated via a CNC program. An example program is included in appendix 1.

The motor speed controller (which should be set up near the laser controller for safety reasons) is used to set the proper centrifuge rpm for the desired effective gravity. The theoretical rpm values are listed in Table 3-1. An optical encoder is used to measure the centrifuge rpm.

The gas flow regulator is used in conjunction with the CNC program to regulate the gas flow out to the centrifuge arm. The regulator controls the pressure and flow rate of the gas before it reaches the centrifuge.

The Ampro motherboard on the centrifuge controls all data acquisition and storage during the experimental run as well as starting the sample motor. It also uploads the data acquisition programs and downloads the data through the board's serial port. It is reset at the beginning of each experimental run. It activates each control phase as needed during the run automatically.

The switches on the camcorder are used for starting and stopping the record mode on the camcorder. There are additional buttons for focusing the camcorder and viewing the previous runs as well as jacks for sending the video signal out to another site.

G: Safety Enclosure

In addition to the enclosed process module, shear pin and emergency stop mentioned earlier, a safety enclosure has been built to further safeguard the operator and other observers during operation.

The safety enclosure (Figure 3-20) is designed to contain the laser beam and any flying debris long enough for the operator to shut down the system in the event of a mechanical failure. It has the additional benefit of providing eye protection from laser beam scatter for the operator and any visitors during normal operation. The safety enclosure is made out of 6.4mm (1/4") Lexan sheet joined with pre-slotted angle and mounted on four lockable wheels. Lexan was chosen because it has a high impact resistance and is opaque to 10.6 μ m radiation. At the same time, it is transparent for visible wavelengths which allows observation of the centrifuge. The wheels were added to make the enclosure easier to move.

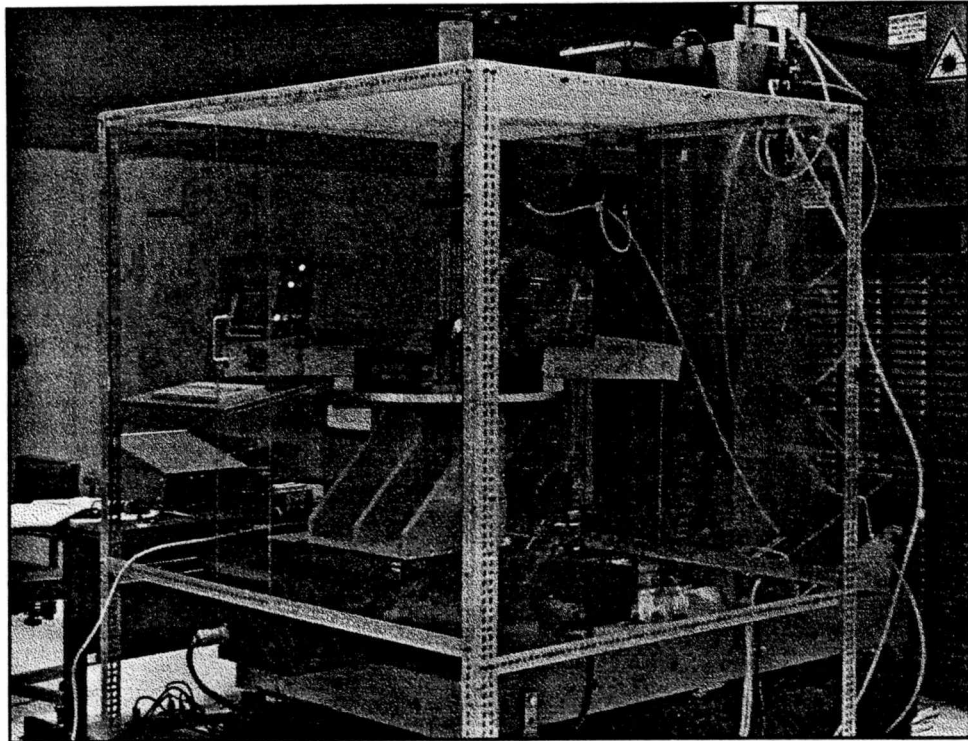


Figure 3-20. Safety Enclosure

Chapter 4: Experiment

Spot welding was selected for the first set of experiments. The objectives of the experiment were: to show the viability of the enhanced gravity LMP system and test the associated hardware and software and to determine the effect of gravity on the final shape of the weld pool.

Experimental Parameters

The set of experimental parameters is given in Table 4-1. The variables were the magnitude of the resultant gravity vector (g-level) and the direction of rotation of the centrifuge. The rotation direction was varied to provide clockwise and counterclockwise samples for each of the acceleration levels to help isolate the influence of the coriolis effect on the final shape of the weld pool.

Table 4-1. Experimental Parameters

Sample	Material	Inconel 718
	Size	20mm x 20mm x 3mm (0.79" x 0.79" x 0.12")
Laser	Wavelength	10.6 μ m
	Power Delivered	1200W average 2400W peak
	Duration	0.1 s
	Focus	At sample surface
Cover Gas	Type	Helium
	Flow rate	side jet, 30 units (8 l/min)
Centrifuge	Rotation	cw, ccw
	g-levels produced	2g _e , 4g _e , 6g _e , 8g _e , 10g _e

Overview

Experiment. Before setting up the experimental apparatus, the joule stick calorimeter used for in-situ power calibrations was calibrated to a Scientec water cooled calorimeter. This helped to ensure the accuracy of later power calibrations.

The enhanced gravity laser materials processing (EGLMP) system was then installed on the existing laser workstation. Care was taken to ensure that the optical axis of the Rofin-Sinar RS3000 laser, which provides the laser beam, was properly aligned with the rotational axis of the centrifuge. Once the optics were properly installed and aligned, the joule stick was used to calibrate the indicated laser power to the power delivered at the sample surface.

After initial setup and calibration were complete, the EGLMP system was configured for an experimental run in the following manner. The mirror and process module were set to the proper location for the selected process parameters and the laser power was calibrated with the use of the joule stick calorimeter. Next, the sample was installed and all onboard cables and connections were secured. After the computer initialization protocol was completed, the camcorder was aligned, focused, and placed in record mode. The safety enclosure was then secured. At this point, the laser was activated to begin its warmup and the centrifuge was spun up and stabilized to the appropriate rpm. Once the laser had stabilized at the correct power level, the CNC laser control program was activated and the weld was performed.

After completion of the weld, the centrifuge was gradually brought to a halt, the safety enclosure was opened, the data was downloaded to the host computer, and

the computer was reset. The sample was then removed for labelling and further analysis.

Two experimental runs for each combination of rotation direction and acceleration level were planned. This allowed the samples to be compared for repeatability.

Diagnostics. In order to determine the true magnitude and direction of the acceleration affecting the sample, readings from accelerometers aligned along the r, t, and z axes (Figure 3-18) were taken during the welding. These readings provided a record of the acceleration along each axis which could be examined later for evidence of disturbances. A resolution of $0.0024g_e$ (r axis) and $0.0012g_e$ (t and z axes) was obtained with a $286 \mu s$ delay between readings.

In order to obtain information for correlating plasma characteristics with acceleration level, the weld plasma was monitored with two different types of diagnostics: photodiodes and a camcorder. The photodiodes were positioned in the process module such that one monitored the surface during heating and, later, the combined signal from the surface and the plasma while the other monitored signals originating above the sample surface. The surface monitoring photodiode also provided a method for determining the approximate time the laser impacted the sample surface within the diagnostic record. A $71 \mu s$ resolution was obtained for these readings.

While the photodiodes monitored the intensity of the plasma, the camcorder provided a method for monitoring the location. This was important since the size or

location of the plasma might change but not the overall intensity monitored by the photodiodes.

Experiment Procedure

Power Calibration (Calorimetry)

- 1) Calibrate joule stick calorimeter to Scientec 390 water cooled calorimeter.
- 2) Calibrate CO₂ indicated power to delivered power between the turning mirror and the focusing lens (to protect the lens) with the joule stick calorimeter before starting runs for each new set of conditions.

Centrifuge Setup

- 1) Align HeNe alignment beam with CO₂ beam.
- 2) Place centrifuge on workstation table with washer centered under centrifuge block for axis alignment. Bolt loosely in place.
- 3) Remove central mirror and gas delivery hose from centrifuge flywheel (to remove obstructions along rotational axis of centrifuge).
- 4) Center HeNe on centrifuge using worktable adjustments (gas outlet hole is center of flywheel). Place lens tissue over gas outlet hole to improve visibility of HeNe and facilitate the centering of the beam.

- 5) Align central axis of centrifuge with HeNe laser.
 - a. Set a flat piece of glass on center of centrifuge flywheel.
 - b. Tighten corner bolts until centrifuge is secured and HeNe reflection is aligned with its primary beam.
- 6) Check HeNe alignment with centrifuge. Repeat steps 4 and 5 until beam stays centered on centrifuge and axes are aligned. Test with CO₂.
- 7) Replace central mirror in holder.
- 8) Check that camcorder is positioned correctly - sample is centered in viewfinder.
- 9) Mount triggering IR LED above the path of central mirror support tubes.
- 10) Put safety enclosure in place, lock wheels.
- 11) Connect trigger lines from control panel to box in rear of laser cabinet.
- 12) Connect all appropriate apparatus to outlet power.
- 13) Connect optical encoder to oscilloscope and circuit board.
- 14) Turn on power for oscilloscope.
- 15) Turn on power for the optical encoder's amplification circuit.
- 16) Adjust oscilloscope tilt, ground, and trigger to proper levels.

Note: See oscilloscope manual for further information.
- 17) Connect sample motor if desired.
- 18) Install camcorder and photodiode filters if desired.
- 19) Check gas flow level. Adjust if necessary.

Welding Protocol

Preweld Activities:

Initial setup for each new set of conditions:

- 1) Set central mirror height and process module angle to theoretical values for run (see Table 3-1).
- 2) Align central mirror with process module.
 - a. Place crossed wire optical target on process module (around lock nut on the focusing lens holder).
 - b. Adjust central mirror up/down (adjustment bolt) and left/right (set screws) until HeNe is properly located (centered) on target wires.
- 3) Check optical alignment.
 - a. Note location of HeNe.
 - b. Rotate centrifuge 360 degrees. Check that HeNe does not move significantly with respect to the sample.
- 4) Test alignment with CO₂ beam.
- 5) Calibrate power reading with joule stick calorimeter.
- 5) Bring up operations directory from host computer. (type *cd/CENTRIFU.GE*)
- 6) Install fresh battery in centrifuge power supply.

Note: Battery is good for approximately six (6) runs.

- 10) Turn on onboard power supply. Make sure the power LED on the onboard computer lights.

Before each run:

1) Begin computer startup protocol:

- a. Connect null modem cable from serial port (COM1) of host computer to serial port of centrifuge data acquisition and control computer.
- b. Reset onboard computer.

Note: The computer will give 1 short beep, pause, then one short beep to indicate successful bootup and an 'ok' beep to indicate readiness to receive files.

- c. Transfer all necessary programs to the onboard computer by starting the "filesend.exe" program on the host computer. (type *filesend*)

Note: The computer will 'ok' beep 3 times to indicate successful file transfer (host computer will display successful transfer as well).

Note: The onboard computer will 'ok' beep to indicate that it is waiting for the external trigger.

- d. Disconnect null modem cable from data acquisition and storage computer.

- 2) Clean focusing lens.
- 3) Clamp sample in sample carrier.
- 4) Hook up thermocouples if desired.
- 5) Move sample with respect to HeNe to proper location for experiment.
- 6) Make sure camcorder power is on. (set power switch to camera position)
- 7) Check that camcorder is displaying the correct time and date and that the shutter speed is correct.

- 8) Put camcorder in manual focus mode and focus camcorder to weld location.
- 9) Close processing box and secure lid.
- 10) Start camcorder record by pressing the 'record' button on the camcorder viewfinder.
- 11) Close safety enclosure and secure door.
- 12) Activate laser for warm-up period.
- 13) Make safety check of perimeter.
- 14) Begin centrifuge spinup.
 - a. Turn the power on.
 - b. Set the direction switch to the proper rotation direction (cw or ccw).
 - c. Slowly increase power to required setting by rotating knob clockwise.
- 15) Check "CENTRIFUGE.PP" program on CNC for readiness.
- 16) Check the centrifuge rpm with the oscilloscope. Make sure it is correct.
- 17) When laser indicates correct peak power, reduce laser power to desired level and switch laser and shutter control to the CNC.

Processing:

- 1) Activate CNC laser control program "CENTRIFUGE.PP".

Note: The computer should trigger (short single beep). Watch trigger LED.

Note: The onboard computer will give an 'ok' beep to indicate that the data acquisition is complete

Postweld activities:

- 1) Switch laser and shutter control to manual.
- 2) Gradually bring centrifuge to a stop.
 - a. Decrease power to zero.
 - b. Turn the power switch to 'off'.
- 3) Open the safety enclosure.
- 4) Stop the camcorder.
 - a. Press the 'record' switch a second time to end the record sequence.
 - b. Turn camcorder off. (set the power switch to the off position)
- 5) Attach serial cable from host computer to onboard computer.
- 6) Transfer data to host computer by starting the "filerec.exe" program on the host computer. (type *filerec*)
- 7) Toggle momentary switch to download data.

Note: The onboard computer will give an 'ok' beep twice to indicate the beginning of the download sequence.

Note: The host computer will give an error to end the download after 577 packets (tries to read past end of file).

- 8) Remove sample from sample carrier.
- 9) Label sample with run information (#,g-level,cw/ccw,etc).
- 10) Change the name of data file for storage. (type *rename test.str [new name].str*)

Note: For further runs, return to "preweld activities: Before each run:".

Final Shut Down:

- 1) Shut down laser.
- 2) Disconnect null modem cable from onboard computer.
- 3) Turn off onboard power supply.
- 4) Turn off power to oscilloscope and the optical encoder's amplification circuit.
- 5) Disconnect optical encoder from oscilloscope.
- 6) Transfer camcorder data from 8mm tape to VHS tape and label.

Note: See camcorder manual for instructions.

Chapter 5: Results

Centrifuge Performance

Each module will be critiqued individually for design and performance.

Centrifuge Motion Package

The centrifuge motion package bolted securely to the existing workstation while still enabling the user with the ability to adjust the rotational axis of the centrifuge so that it would coincide with the laser beam axis. It also provided sufficient angular velocity to produce g-levels between 2 and 10g_e at a radial distance of 20 inches. The produced acceleration remained stable within $\pm 0.1g_e$. Additionally, the motion package successfully delivered the process gas to the rotating process module and provided stability for the rest of the system.

Process Module

The process module was able to safely redirect the laser beam into the processing box while keeping the angle of the beam normal to the sample surface by adjusting the height and angle of the central mirror. At the same time, the angle of the processing box could be set such that the resultant gravity vector was also normal to the sample surface. Process gas delivery to the sample also functioned well.

The process gas was successfully transferred from the motion package to the processing box where it was directed onto the sample by the gas delivery nozzle. This

system was able to provide sufficient cover gas to protect the sample from oxidation during welding.

Unfortunately, the design of the lens holder was slightly flawed. While it allowed the user to adjust the focal point of the lens with respect to the sample, it did not provide for protection from weld spatter. Since the cross flow from the process gas was not sufficient to blow the spatter away from the lens, the spatter impacted the lens during the runs. This spatter could not be removed and has damaged the lens. It is recommended that a nozzle be installed to help protect the lens from further spatter.

The sample holder and positioning system was able to hold the sample firmly in place during the runs. Unfortunately, the sample motion package did not function properly. While the sample motor and motor control unit performed satisfactorily on the test rack, they failed to work when placed into the process box. It was determined that this was due to the motor being designed as a case ground system in which the motor chassis was electrically connected to the motor shaft. While it was possible to electrically isolate the case of the motor, it was not possible to isolate the shaft. This led to a system in which the sample motor was shorted to the centrifuge and, thus, would not work. It is suggested that the motor be exchanged for a dual wire motor with an electrically isolated case such as the 12V DC reversible motor from the H&R company (#TM92MTR2131).

Diagnostics

The diagnostics chosen performed well under the stresses experienced onboard the centrifuge. However, there were a few problems with the camcorder and the photodiodes.

The accelerometers were able to record the g-level present in all three axes (Figure 5-1). However, the slight sensitivity of the accelerometers along their secondary axes did show up in the records, especially for the tangential axis.

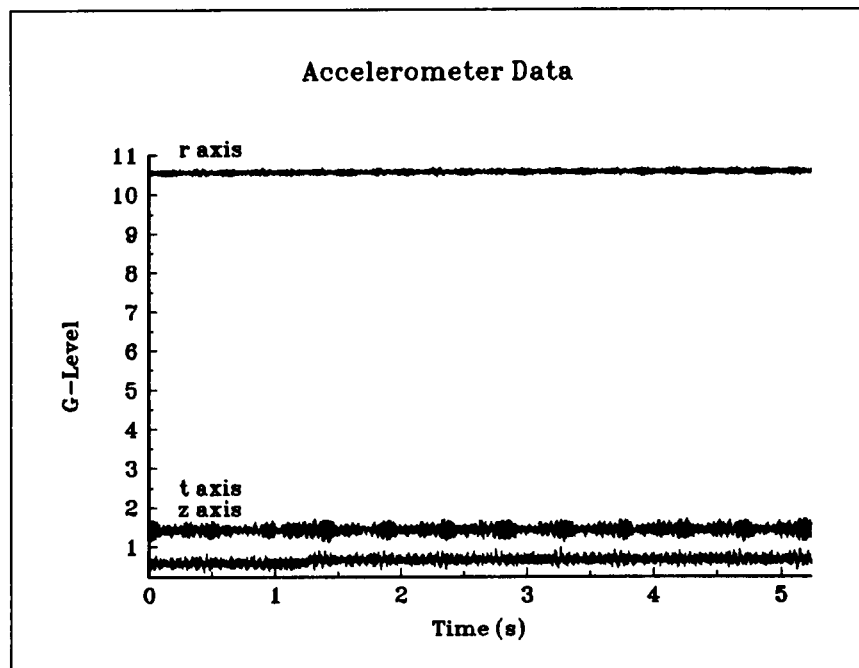


Figure 5-1. Typical Accelerometer Data

The camcorder was able to record properly under the stresses; however, the focus and shutter speed had to be set manually and would only remain set as long as the camcorder was either paused or actively recording. Since the camcorder would

only remain paused for five minutes, it had to run during the laser warmup. This resulted in approximately ten minutes of tape being wasted at the beginning of each run. It was also determined that the weld was too bright for the camcorder to record useful data under even the fastest shutter speed (1/1000s). The use of an appropriately selected neutral density filter will bring the light intensity within the useful limit of the camcorder.

The photodiodes saturated during the runs. This was determined to be caused by two factors: the intensity of the plasma and the electrical noise created by the plasma. The second problem was solved through the addition of shielding on the photodiode wiring and the first was helped with the addition of neutral density filters (exposed film). Once these problems were solved, the photodiode data fell within the useful voltage range (Figure 5-2).

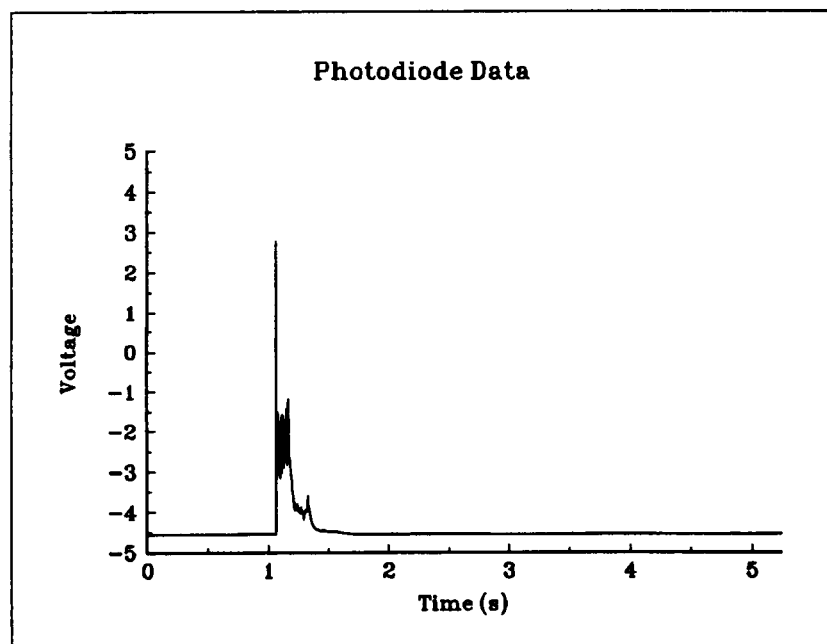


Figure 5-2. Typical Photodiode Data

Data Acquisition and Storage

The data acquisition and storage module was able to record data from each of the ten channels at the correct sampling frequencies for a total sampling frequency of 56kHz. The amplification circuits were able to bring these signals into the recording range of the data acquisition board. For the accelerometers, this resulted in a ratio of 0.4V/g_e in the r-axis and 0.56V/g_e in the t- and z-axes. The brass shield covering the amplification circuits adequately protected them from damage caused by both the laser beam and the weld spatter.

However, a few problems were encountered. First, the original data acquisition program called for the computer to turn on the sample motor two seconds into the data acquisition. This caused the system to periodically lose a sample as it checked clock time. The problem was solved temporarily by removing the motor trigger from the program. A more permanent solution calls for the motor to be triggered from the second clock available from the DM406 board.

A second problem encountered during the testing of the data acquisition system was that the voltage levels recorded by the computer did not always agree with the values seen at the checkpoint prior to the multiplexer. This was determined to be caused by the use of the 1458N chip in the buffer circuit. The chip was too slow to enable the voltages at the output to change and stabilize before being read. The problem was solved by exchanging the 1458N chip for a 353N chip which is a pin for pin replacement but much faster.

Another problem dealt with the usefulness of the data after downloading. It was difficult to identify the order in which the signals were recorded. This was solved by tying one channel to a known value (ground). Since the temperature sensor was not currently being used, it was removed and that channel was grounded. A thermocouple channel can be used in the future since there are four thermocouple channels available.

As previously mentioned, noise appeared on the channels during the weld. The noise was determined to be caused by the electromagnetic field created by the plasma. The photodiodes were affected because of their proximity and the fact that the leads were not shielded. The noise added enough voltage to the signal to saturate the photodiodes which in turn caused noise on the other channels. This was solved by shielding the leads with grounded aluminum tape and adding filters to prevent any saturation of the photodiodes caused by excessive light levels.

Power Supply

The power supply was able to power the computer and diagnostics for approximately six runs, then the battery voltage dropped below the minimum supply voltage for the voltage regulators. This cuts power to both the computer and the diagnostics. To prevent this, the battery should be swapped out before the sixth run. The design of the power supply (and the counterweight box in which it is stored) will allow the battery to be exchanged quickly and easily.

System Control

The system control module controlled the EGLMP system with few problems. The laser/CNC controller regulated the process gas and laser as well as triggered the data acquisition system. The camcorder switches were readily accessible and easy to use. The motherboard also performed its tasks. There were, however, occasional glitches which occurred during the downloading of the data file. This is not unusual for data transfers of this kind. If an error occurs during the download, the user should simply plug a keyboard into the onboard computer's chassis and type *sendfile.exe*. This will repeat the downloading of the data file.

Some problems were encountered with the control of the centrifuge rpm which was read with the help of the optical encoder. There was some variation in the spacing of the stripes utilized by the optical encoder which made it difficult to conduct correct measurement of the rpm value. This resulted in up to a 10% error in the g-level. To reduce these problems, the stripes should be remarked more uniformly. This is best done by someone with precision angle measurement equipment such as a machine shop.

Safety

The safety enclosure worked as expected. The experimenters were able to observe the centrifuge during the run while being protected from the laser beam. Access to the centrifuge was possible through the front sliding door between runs.

Experimental Results

Sample Analysis

The first step in the sample analysis was to plot and review the accelerometer data. This was important for determining the actual g-level obtained during each run. Once the actual g-level for the samples was determined, overview photographs (Figure 5-3) were taken for each weld surface. These photographs were used to measure the maximum and minimum diameter for each weld by measuring these distances with a set of calipers. The measurements were converted to true values with the use of a photograph of a scale taken under the same magnification. The next step in the sample analysis was to measure the overbead height from the surface. Measurements of the overbead height were taken under an optical microscope at four locations around the weld and averaged to determine the weld overbead height. These measurements were taken by focusing on the uppermost point of the overbead (disregarding spatter) within a quadrant and recording the value on the vertical scale then focusing on the original surface within the same quadrant and, again, recording the value on the vertical scale. The overbead height was determined from the difference between the two values.

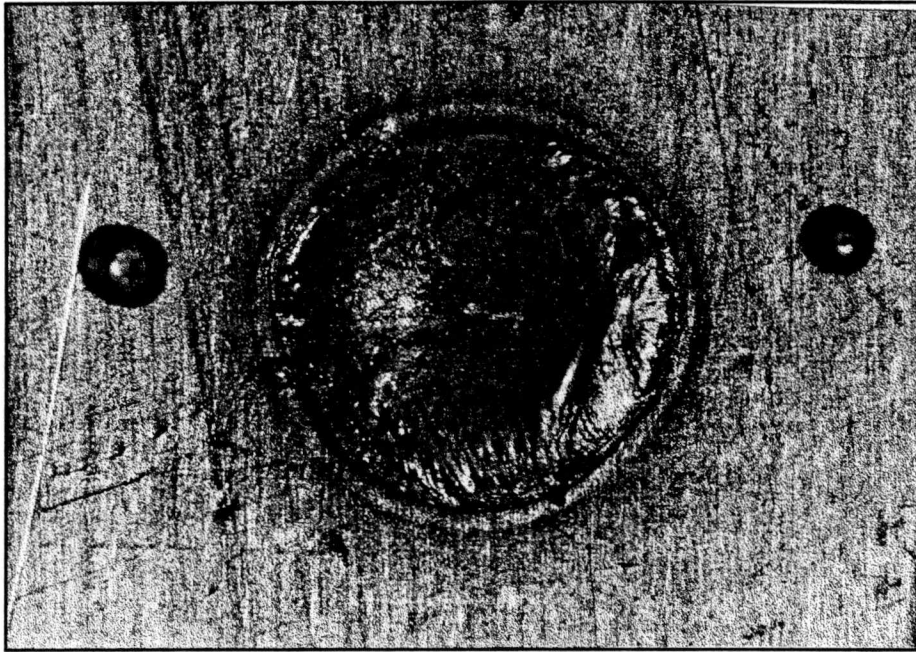


Figure 5-3. Example of the Weld Surface Overview

Once these measurements were completed, a set of runs for both the clockwise and counterclockwise directions was randomly chosen. These welds were carefully polished to the weld centerline and photographs were taken of each cross section (Figure 5-4). Each cross section was traced onto graph paper from the photograph negative for use in determining the weld cross section area. Next, measurements (Figure 5-5) were made for the weld depth, pore depth, leading and trailing edge nailhead depth, weld width at half depth, and the overbead height (averaged from values for both sides of the weld). The values for the weld measurements were plotted as a function of gravity level. Finally, the weld depth to width ratio was calculated and plotted.

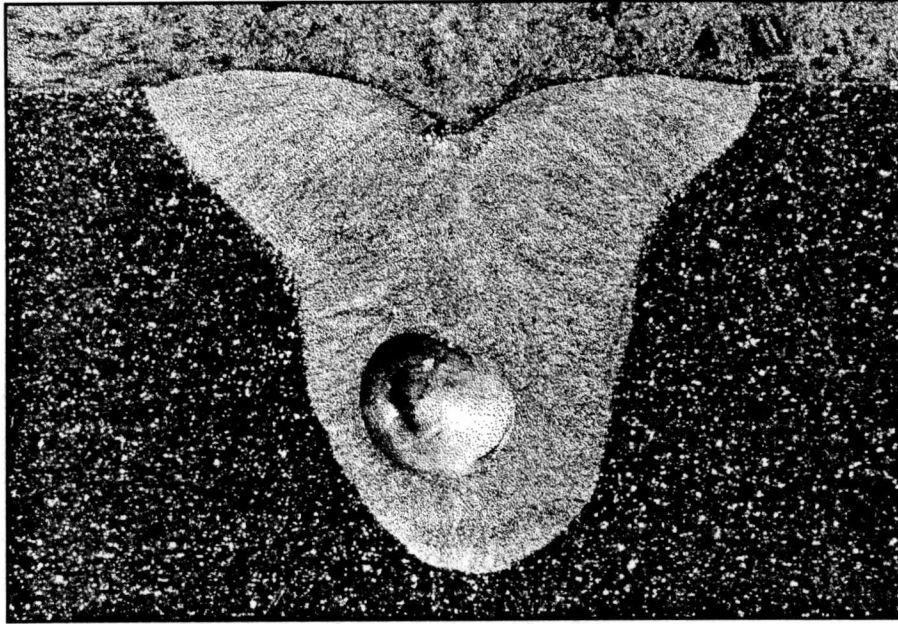


Figure 5-4. Example of the Weld Cross Section

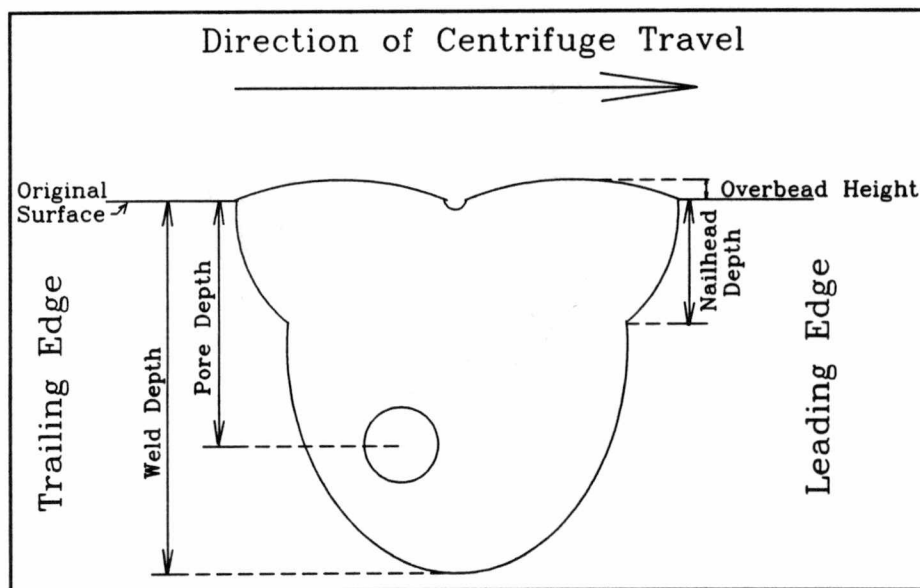


Figure 5-5. Schematic of Weld Measurements

Results

Weld Diameter (Figure 5-6). Although there was a fair amount of scatter in both the weld diameters and roundness (seen from the size of the bar), no trend was seen in the data for either the clockwise (cw) or counterclockwise (ccw) welds which could be attributed to a gravity effect.

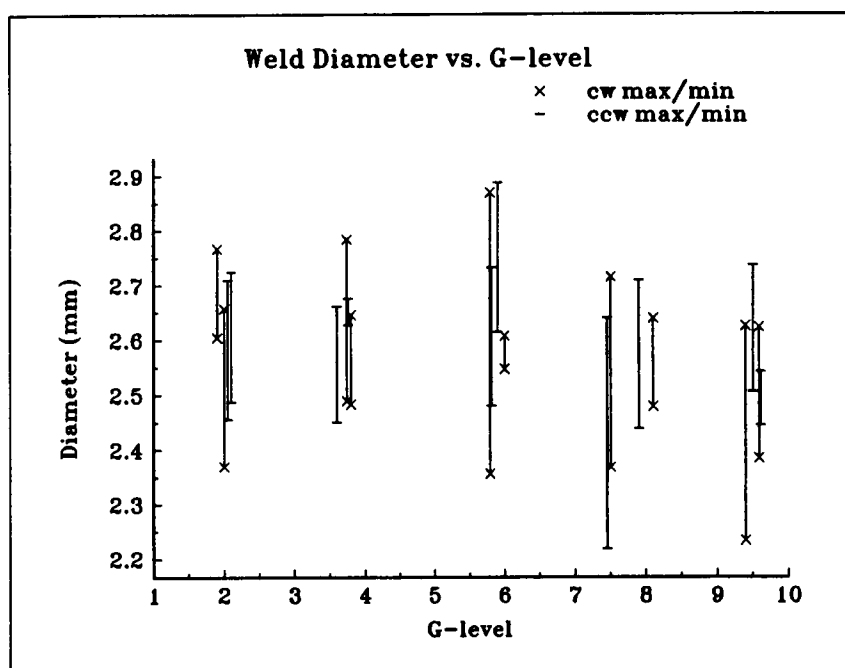


Figure 5-6. Max/Min Weld Diameter vs. G-level

Weld Depth (Figure 5-7). The cw samples showed a slight drop off in weld depth that increased after 6g_c. The ccw samples also showed a drop off in weld depth with the exception of the 2g_c sample. Overall, both the cw and the ccw welds show a total decrease in weld depth of approximately 10% as the g-level increases to 10g_c.

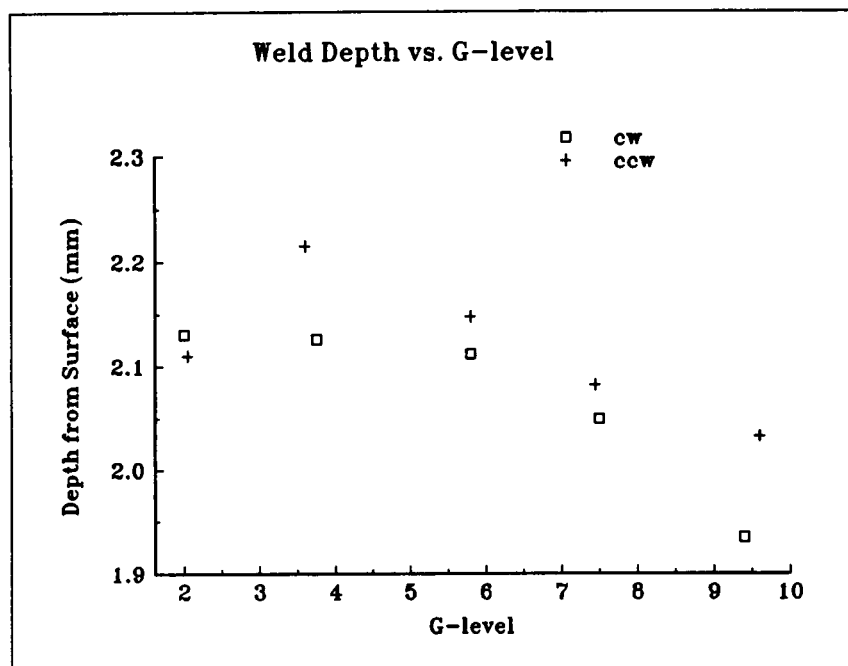


Figure 5-7. Weld Depth vs. G-level

Pore Depth (Figure 5-8). The location of porosity was determined when pores were visible in the cross-section. All of the welds sectioned did not have pores visible in this location probably due either to the pores not being located in the plane of the cross section or to the absence of pores of significant size. Due to the small number of pores available for measurement, it is difficult to determine the correlation between gravity level and pore location. However, the cw samples showed an inclination for the pore to get closer to the sample surface as the gravity level increases while there was insufficient porosity visible in the ccw samples to determine a pattern in pore location.

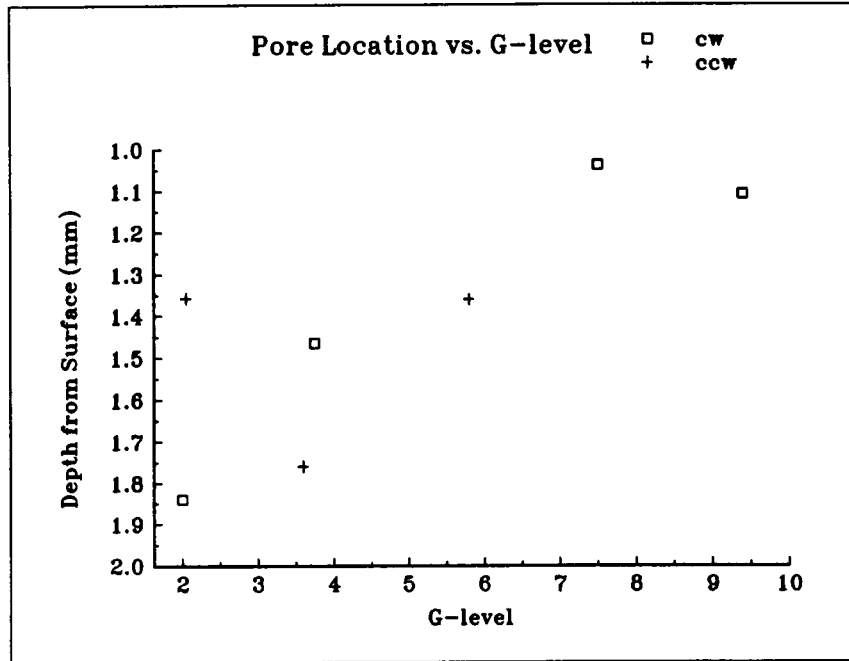


Figure 5-8. Pore Depth vs. G-level

Overhead Height (Figure 5-9). Although there was some variation present in the weld overhead height, none of the measured data showed any consistent relationship between the overhead height and the g-level.

Nailhead Depth (Figures 5-10 and 5-11). Neither the cw nor ccw samples showed any clear relationship between the leading edge nailhead depth and the g-level. The trailing edge nailhead depth, however, showed some interesting effects. In the cw samples, the nailhead depth shows the possibility of a decrease above 6g.

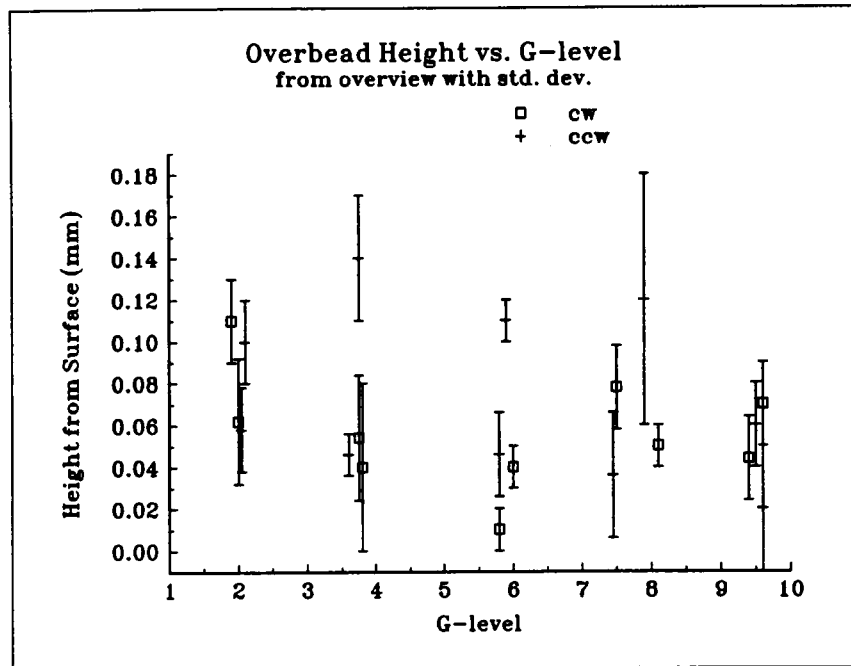


Figure 5-9. Overhead Height vs. G-level with Standard Deviation (overview)

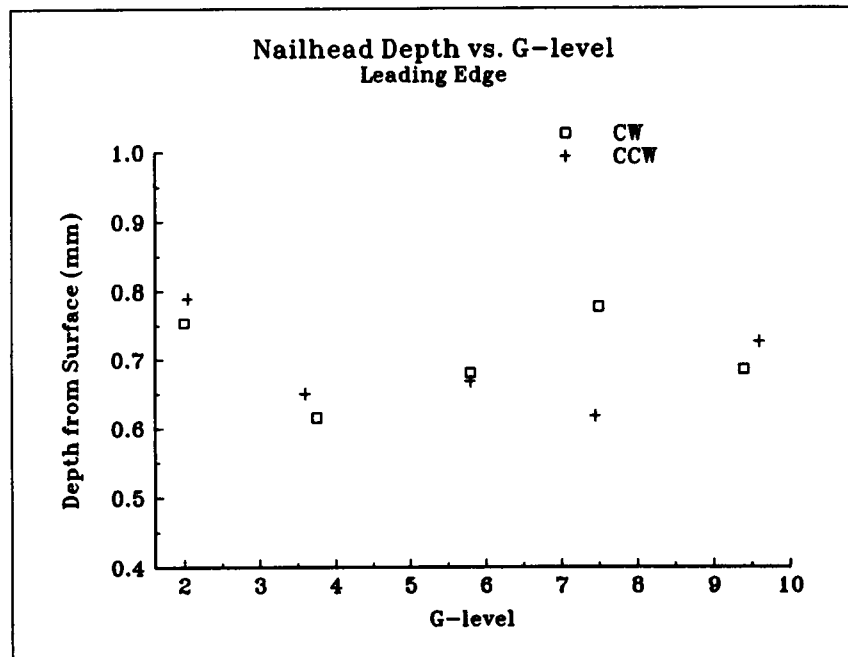


Figure 5-10. Nailhead Depth vs. G-level (Leading Edge)

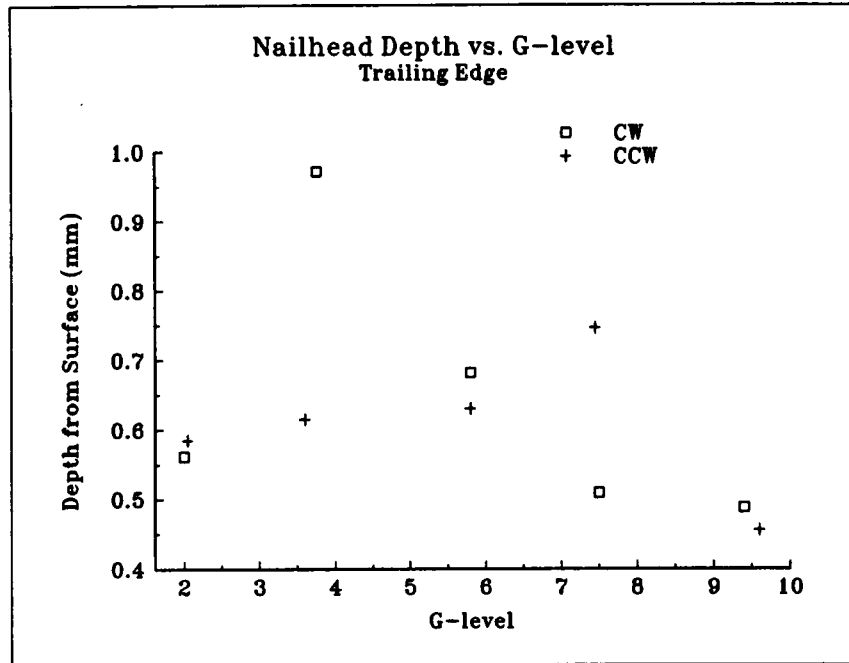


Figure 5-11. Nailhead Depth vs. G-level (Trailing Edge)

Width at Half Depth (Figure 5-12). The weld width measurements were taken at half the full depth of the weld to avoid the influences of both the nailhead and the weld root on the width measured. Again, there is a possible effect seen above $6g_c$ with the exception of the ccw $8g_c$ sample. This sample had a weld width greater than any other ccw sample.

Depth/Width Ratio (Figure 5-13). The weld depth to width ratio showed a large variation (about 28%) with a slight trend with respect to g-level evident for above $6g_c$.

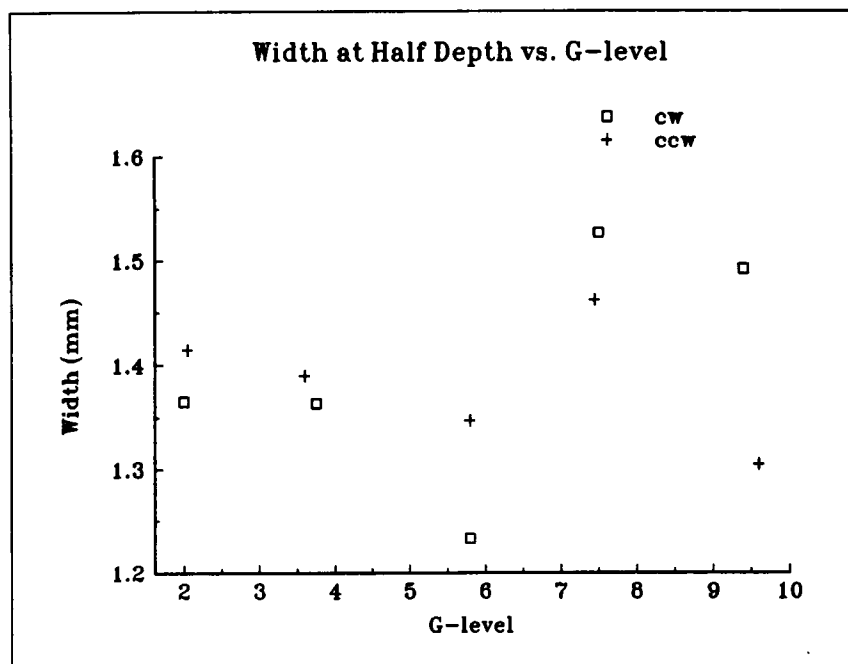


Figure 5-12. Width at Half Depth vs. G-level

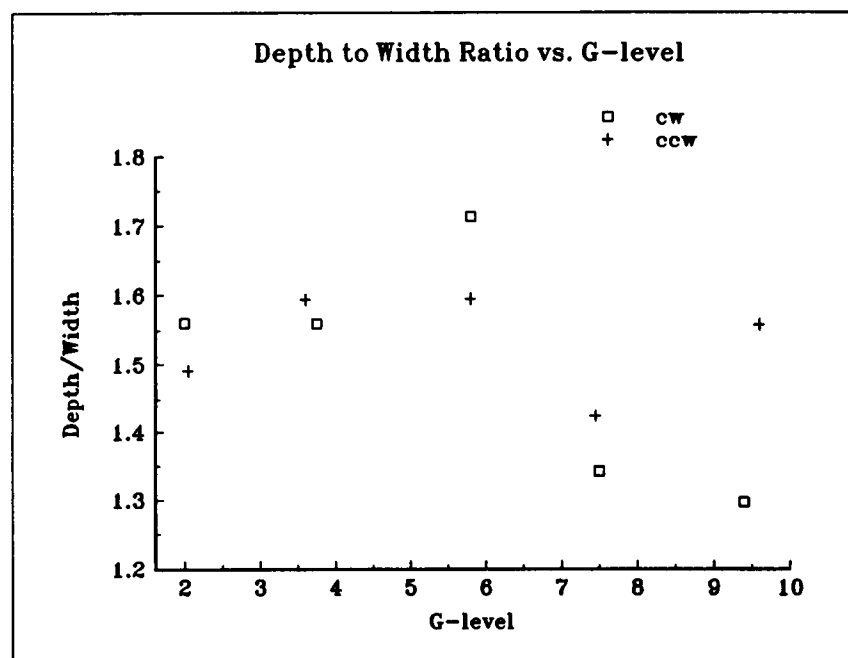


Figure 5-13. Depth/Width Ratio vs. G-level

Area of Melt Cross Section (Figure 5-14). The melt areas were measured from the weld outlines drawn on graph paper. They included only the part of the weld below the original sample surface. Although there was a relatively high variation in the area of the melt cross sections (about 20%), there was no strong relationship between this variation and the g-level for either the cw or ccw cases. However, the ccw samples showed a tendency to decrease in area as the g-level was increased.

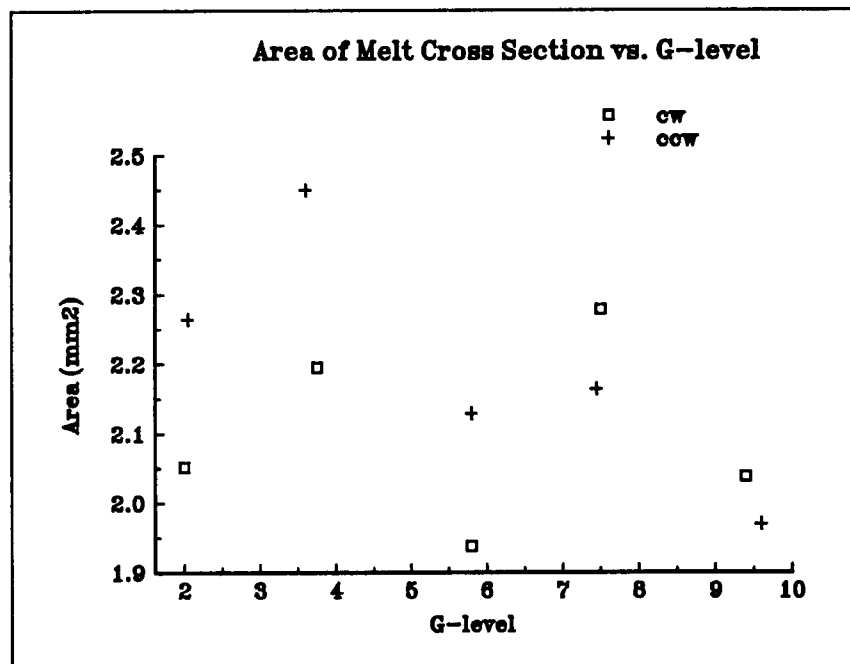


Figure 5-14. Area of Melt Cross Section vs. G-level

Summary

The EGLMP system satisfactorily accomplished the design goal. It can currently be used to produce spot welds with a 3kW CO₂ laser at g-levels between 2 and 10g_e with a $\pm 0.1g_e$ stability. The system provides the capability to monitor the experiment with photodiodes, thermocouples, accelerometers, and a CCD camera. With the use of a different focusing lens, the system is also useable for work with Nd:YAG lasers.

Although a statistically significant number of data points was not obtained in the limited preliminary experiments, a few interesting trends were noted. For the ccw runs, the weld depth and the width at half depth showed a general tendency to decrease with higher g-levels, in particular above 6g_e. This resulted in a decrease in the area of the melt for these runs. For the cw runs, the weld depth decreased significantly after 6g_e and the pores tended to be closer to the weld surface at higher g-levels.

Chapter 6: Conclusions and Recommendations

Conclusions

The EGLMP system was able to meet the design goal of providing an enhanced gravity welding environment with standard diagnostics. Although the system is not presently capable of producing bead-on-plate welds, it can produce and monitor spot welds. This capability is sufficient for answering many of the questions concerning the effect of gravity on the final size and shape of a weld. As a result, the EGLMP system will provide a tool for researchers to use in conducting further investigations of weld behavior in enhanced gravity conditions.

While the preliminary experiment conducted to test the EGLMP system did not provide enough samples to conduct a definitive statistical analysis of the results, it could be seen that for g-levels between 2 and 10g_e, increasing the gravitational force caused relatively minor changes in the shape and size of a partial penetration weld. The experiment indicated a possible overall reduction in weld depth with increasing gravity and a tendency for large pores to solidify closer to the surface. These changes could be caused by a change in convection patterns due to the increased role of buoyancy driven convection at higher g-levels. There were also differences between the cw and ccw data which indicates that the coriolis acceleration may be effecting the results. The most interesting experimental result is the possible occurrence of a "magic" gravity level similar to those observed by other centrifuge investigators. While there were minimal (if any) effects seen on spot weld

dimensions when the g-level changed from $2g_e$ to $6g_e$, there did seem to be a noticeable dependance on gravity above $6g_e$.

Recommendations

It is recommended that the corrective procedures specified in the evaluation of the centrifuge performance in chapter 5 be completed to bring the centrifuge performance to its full potential. This will provide a system which can produce both bead-on-plate and spot welds while monitoring the welds with thermocouples, photodiodes and a CCD camera. The addition of a 10 to 1 knob on the motor speed controller will also enhance the user's ability to fine tune the centrifuge rpm and, hence, the resultant acceleration at the sample.

It is also recommended that further experiments be conducted concerning the roles of the gravitational force and the coriolis acceleration on spot welds. In addition, research should be conducted with bead-on-plate welds so that transient effects can be avoided. Preliminary studies of full penetration welds under enhanced gravity have shown a some relationship between weld size and shape and g-level. It is recommended that a full experiment be conducted in this area to further understand these effects.

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] Wilcox, W.R. Materials Processing in Centrifuges, Clarkson University, Potsdam, N.Y., 1990.
- [2] Busch, G. "M551 Metals Melting Experiment," Grumman Research Department Report RE-536, Research Department, Grumman Aerospace Corporation, Bethpage, N.Y., 1977.
- [3] Bourgeois, S. V., and M. R. Brashears. "Fluid Dynamics and Kinematics of Molten Metals in the Low-Gravity Environment of Skylab," Materials Sciences in Space with Application to Space Processing Vol. 52, Ed. L. Steg, AIAA, New York, N.Y., 1977.
- [4] McCay, M. H. and T. D. McCay. "Processing of Metallic and Electronic Materials in Space," Principles of Solidification and Materials Processing Vol. 2, Eds. R. Trivedi, et al., Oxford and IBH Publishing Co. Pvt. Ltd., Bombay, India, 1989.
- [5] McKannan, E. "Results From Metallurgical Flight Experiments," Materials Sciences in Space with Application to Space Processing Vol. 52, Ed. L. Steg, AIAA, New York, N.Y., 1977.
- [6] Okhotin, A. S., V. F. Laptchinsky, and G. S. Shonin. "Some Results of Studies in Space Technology in the USSR," Materials Sciences in Space with Application to Space Processing Vol. 52, Ed. L. Steg, AIAA, New York, N.Y., 1977.
- [7] Shulym, V. F., et al. "Peculiarities and Future Development of Space Welding," Welding in Space and the Construction of Space Vehicles by Welding, Proceedings of the Conference, pp 12-24, American Welding Society, Miami, FL, 1991.
- [8] Kaukler, W. F. and G. L. Workman. "Laser Welding in Space," Welding in Space and the Construction of Space Vehicles by Welding, Proceedings of the Conference, pp 318-334, American Welding Society, Miami, FL, 1991.
- [9] Workman, G.L. and W.F. Kaukler. "Laser Welding in a Reduced Gravity Environment," AIAA Paper 846, 30th Aerospace Sciences Meeting and Exhibit, 1992.

- [10] Tandon, K. N. and G. Wang. "Laser Welding of Stainless Steel - Experiments Aboard KC-135," AIAA Paper 579, 31st Aerospace Sciences Meeting and Exhibit, 1993.
- [11] Mueller, R. E. Laser Processing of Materials under Low Gravity Conditions, PhD, York University, North York, Ontario, Canada, March 1993.
- [12] Masubuchi, K. and M. Nayama. "Fusion Welding Experiments under Low-Gravity Conditions using Aircraft," Welding in Space and the Construction of Space Vehicles by Welding, Proceedings of the Conference, pp 186-190, American Welding Society, Miami, FL, 1991.
- [13] Foley, J. S. and C. M. Banas. "Laser Welding Stability Limits," Focus on Laser Materials Processing, Proceedings of the 6th International Congress on Applications of Lasers and Electro-Optics, pp 47-54, ICALEO '87, Ed. S. L. Ream, IFS Publications, 1987.
- [14] Regel, L. Materials Processing in High Gravity, ed. W.R. Wilcox, Clarkson University, Potsdam, N.Y., 1990.
- [15] Regel, L. Materials Science in Space. Theory, Experiments, Technology v.1, ed. R.Z. Sagdeev, Consultants Bureau Division of Plenum Press, New York, NY, 1990.
- [16] Weber, W., G. Neumann and G. Muller. "Stabilizing Influence of the Coriolis Force During Melt Growth on a Centrifuge," Journal of Crystal Growth, Vol. 100, 1990.

APPENDICES

Appendix 1: Program Source Code

CENTRIFUGE.PP	.74
BINTOTXT.BAS	.76
DMASTRMR.C	.79
FILESEND.C	.86
SENDFILE.C	.93
FILEREC.C	100
RECFILE.C	108

CENTRIFUGE.PP

Parts program for the CNC laser controller. Controls the laser and gas flow (on/off). This program also provides a trigger source for the data acquisition program *dmastrmr.exe* on the onboard computer with the 'shutter open' command.

CENTRIFUGE.PP

```
*****
; PROGRAM TO TURN ON GAS AND RUN LASER
; AS REQUIRED FOR CENTRIFUGE EXPERIMENT
; *****
;
; SET LASING CONDITIONS
; M85 ; M85 + M86 = Pulse Width Modulated
; M86 ; M85 + M86 = GATED PULSING
; M85 + M86 = MIXED PULSING
; M92 ; GAS ON - RELAY 5
; (CLS,SHOP)
; G4 F2.0 ; 2s DELAY
; M94 ; DISCHARGE ON
; G4 F0.1 ; .1s DELAY TO ALLOW PULSE
; M84 ; DISCHARGE OFF
; (CLS,SHUT)
; M82 ; GAS OFF
; M2
; (DFS,SHOP)
; M98
; G4 F.2
; M88
; )
; (DFS,SHUT)
; M99
; G4 F.5
; M89
; )
```

BINTOTXT.BAS

Stored on the host computer in executable form as *bintotxt.exe*. Converts the binary data from the onboard computer to ASCII files which can be read by other programs for analysis. Each file contains data from an accelerometer or photodiode channel. The data from the thermocouple channels are not stored as ASCII files since the data is not presently being used. The user must indicate the order in which the data is taken (e.g. the first photodiode and accelerometer data is specified while running the program).

Compiled with Microsoft Quick Basic.

BINTOTXT.BAS

```

OPEN Infile$ FOR BINARY AS #1
OPEN Outfile1$ FOR OUTPUT AS #2
OPEN Outfile2$ FOR OUTPUT AS #3
OPEN Outfile3$ FOR OUTPUT AS #4
OPEN Outfile4$ FOR OUTPUT AS #5
OPEN Outfile5$ FOR OUTPUT AS #6

Blocks$ = LOF(1) / 16384
DO: LOOP UNTIL INKEY$ = ""
i0% = 1
T0! = TIMER

FOR k% = 1 TO Blocks$
  FOR i% = i0% TO i0% + 8191
    IF (INKEY$ <> "" OR EOF(1)) THEN EXIT FOR
    Num% = ReadInt$(1)
    SELECT CASE (i% MOD 16)
      CASE PD1%, PD1% + 4, PD1% + 8, PD1% + 12
        PRINT #2, USING "+####"; Num%
      CASE PD2%, PD2% + 4, PD2% + 8, PD2% + 12
        PRINT #3, USING "+####"; Num%
      CASE AC1%
        PRINT #4, USING "+####"; Num%
      CASE AC2%
        PRINT #5, USING "+####"; Num%
      CASE AC3%
        PRINT #6, USING "+####"; Num%
      CASE ELSE 'thermocouple channels, do nothing
        END SELECT
    NEXT i%
  Num% = 0
  FOR j% = i% TO i% + 1
    SELECT CASE (j% MOD 16)
      CASE PD1%, PD1% + 4, PD1% + 8, PD1% + 12
        PRINT #2, USING "+####"; Num%
      CASE PD2%, PD2% + 4, PD2% + 8, PD2% + 12
        PRINT #3, USING "+####"; Num%
      CASE AC1%
        PRINT #4, USING "+####"; Num%
      CASE AC2%
        PRINT #5, USING "+####"; Num%
      CASE AC3%
        PRINT #6, USING "+####"; Num%
      CASE ELSE 'thermocouple channels, do nothing
        END SELECT
    NEXT j%
  
```

```

Written by R. Mueller, 3 May 1993
Modified by C. McKowen, 23 July 1993

DECLARE FUNCTION ReadInt$(FileNum%)
CLS
Defdir$ = "C:\Centrifuge"
PRINT "In what drive and directory is the source file? (Default "; Defdir$; ")"
INPUT Indir$
IF Indir$ = "" THEN Indir$ = Defdir$
IF RIGHT$(Indir$, 1) <> "\" THEN Indir$ = Indir$ + "\"
100:
PRINT "What is the file name?"
INPUT Infile$

CLS
PRINT
Infile$ = Indir$ + Infile$
OPEN Infile$ FOR BINARY AS #1
FOR i% = 1 TO 16
  Num% = ReadInt$(1)
  PRINT USING "+#### "; Num%
NEXT i%
CLOSE #1

PRINT : PRINT
INPUT "Give the position of the first photodiode data (1 or 2) ", PD1%
IF (PD1% < 1 OR PD1% > 2) THEN PRINT "Invalid input, program halted."; STOP
PD2% = PD1% + 2
IF PD2% = 4 THEN PD2% = 0
INPUT "Give the position of the first accelerometer data (1 to 16) ", AC1%
IF (AC1% < 1 OR AC1% > 16) THEN PRINT "Invalid input, program halted."; STOP
AC2% = AC1% + 4
AC3% = AC1% + 8
IF AC1% = 16 THEN AC1% = 0
IF AC2% >= 16 THEN AC2% = AC2% - 16
IF AC3% >= 16 THEN AC3% = AC3% - 16
PRINT

Dot% = INSTR(Infile$, ".")
Dot1% = INSTR(Dot% + 1, Infile$, ".")
IF Dot1% > 0 THEN Dot% = Dot1%
Outfile1$ = LEFT$(Infile$, Dot%) + "PD1"
Outfile2$ = LEFT$(Infile$, Dot%) + "PD2"
Outfile3$ = LEFT$(Infile$, Dot%) + "AC1"
Outfile4$ = LEFT$(Infile$, Dot%) + "AC2"
Outfile5$ = LEFT$(Infile$, Dot%) + "AC3"
  
```

BINTOTXT.BAS

```

i0% = i0% + 2
PRINT k%
NEXT k%

CLOSE #1
CLOSE #2
CLOSE #3
CLOSE #4
CLOSE #5
CLOSE #6

PRINT "This took "; TIMER - T0!; " seconds."
END

FUNCTION ReadInt% (FileNum%)
  S$ = INPUT$(1, FileNum%)
  IF S$ = "" THEN
    N1& = 0
  ELSE
    N1& = ASC(S$)
  END IF
  S$ = INPUT$(1, FileNum%)
  IF S$ = "" THEN
    N2% = 0
  ELSE
    N2% = ASC(S$)
  END IF
  N& = N1& * 256 + N2%
  IF N& > 32767 THEN
    N& = N& - 65536
  END IF
  ReadInt% = N&
END FUNCTION

***** END OF PROGRAM LISTING *****

```

DMASTRMR.C

Send to the onboard computer in executable form as *dmastrmr.exe*. Transfers acquired data to memory by direct memory access (DMA). Once one page of memory is full, the data is stored on a second page while the first page of data is written to the disk (RAM drive). This continues until data acquisition is complete.

Compiled with Borland Turbo C.

DMASTRMR.C

```

/*
Real Time Devices, Inc.
Box 906
State College, PA 16804

Voice: 814-234-8087
FAX: 814-234-5218

Board: DM406

Compiler: Turbo C (r) version 1.0, 1.5, 2.0
          Turbo C++ (r) version 1.0
          Borland C++ (r) version 2.0

Last update: August 8, 1991
G.P.C.

Modified by R. Mueller, 12 March 1993

DESCRIPTION
~~~~~
This program streams acquired data to disk at a steady requested rate.
The acquisition occurs in the background and is transferred to memory
by Direct Memory Access (DMA). Writing to disk occurs in the foreground.
A pair of buffers are used. While one is being filled by the background
DMA routine, the other is being written to disk. When the active buffer
fills, the buffers are swapped. If a swap occurs before a filled buffer
is completely written to disk an overflow error occurs.

This program is written so that you can easily understand what is going
on. There are many areas which could use elegant but obscure routines.
We have opted for clarity.

CHOOSING AN IRQ AND INTERRUPT SOURCE
~~~~~
To use interrupts you must decide what will be the source of interrupts
and which interrupt request line will be activated by the interrupt
source. There are 8 different conditions which can generate interrupts
from the DM406 and these are documented in the DM406 manual. This
program uses the DMA done condition to generate interrupts.

There are also 6 different IRQ lines which can be used (IRQ2 - IRQ7)
however you must use an IRQ that is not already in use by other devices.
Try IRQ3 and IRQ7 first as they are generally not in use. If these don't
work you will have to check your computer's documentation to see which
IRQs are in use and which are free.

```

NOTES

~~~~~  
The PC jumper on connector P8 must be installed before running this program.

You must change the first two lines in the main function to reflect the base address you have set on your DM406 and the IRQ you wish to use.

Many functions in this program must be called in proper order. Calling them out of sequence will almost certainly cause a system hang.

```

*/

#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

#include "DM406.h" /* DM406 header file */
#include "DM406.inc" /* DM406 functions */

#define BUFFER_SIZE 16384
#define DEBUG -1

typedef struct
{
    char Status; /* 1 = Full, 0 = Not Full */
    unsigned char PageOffsetMSB, /* MSB of offset into page */
    PageOffsetLSB, /* LSB of offset into page */
    DMAPage, /* Page where buffer resides */
    Buffer[BUFFER_SIZE]; /* Actual buffer */
} DMABuffer;

DMABuffer DB[2];

unsigned NumOfBytesToTransfer, /* Number of bytes in transfer buffer */
TransferCount, /* Count of buffers written to disk */
DM406StatusByte, /* Address of STATUS port */
DM406StartConversion, /* Address of START CONVERSION port */
DM406ClearDMADone, /* Address of Clear DMA done port */
DMAPageReg, /* Address of DMA controller's page reg */
DMAOffsetReg, /* Address of controller's offset reg */
DMANumberOfBytesRegisters, /* Address of Number of bytes reg */

```

## DMASTRMR.C

```

float CLOCKS_PER_SEC; /* clock ticks per second */

unsigned char DMAEnable, /* DMA enable command */
DMADisable, /* DMA disable command */

NumBytesMSB, /* MSB of number of bytes */
NumBytesLSB, /* LSB of number of bytes */

ActiveBuffer, /* Current buffer */

OldIMR, /* Startup IMR */
IRQ, /* IRQ being used */
DMAChannel, /* DMA done byte */
DMADone, /* Overflow byte */
BufferOverflow; /* Overflow byte */
    unsigned char Twos[8] = { 1, 2, 4, 8, 16, 32, 64, 128};

float PacerClockRate; /* Rate for pacer clock */

void interrupt (*OldISR); /* Startup interrupt vector */

FILE *StreamingFile; /* File in which to place data */

void DMAStart(char i); /* Prototype of DMAStart function */

/*****

ClearBitInIMR

This function clears a bit in the 8259 Interrupt controller that will
enable interrupts on the appropriate IRQ line. For example to enable
interrupts from IRQ 3 bit 3 must be cleared. Remember that bits are
numbered 0..7.

*****/

void ClearBitInIMR(unsigned char Bit)
{
    unsigned char OldIMR, IMR;

    OldIMR = inportb(0x21); /* Read in current IMR */
    IMR = OldIMR & (~Twos[Bit]); /* Clear desired bit */
    outportb(0x21, IMR); /* cprintf out new IMR */
}

/*****

SetBitInIMR

This function sets a bit in the 8259 Interrupt controller that will
mask out interrupts on the appropriate IRQ line. For example to disable
interrupts from IRQ 3 bit 3 must be set. Remember that bits are
numbered 0..7.

*****/

void SetBitInIMR(unsigned char Bit)
{
    unsigned char OldIMR, IMR;

    OldIMR = inportb(0x21); /* Read in current IMR */
    IMR = OldIMR | Twos[Bit]; /* Set desired bit */
    outportb(0x21, IMR); /* cprintf out new IMR */
}

/*****

VectorInterrupt

The VectorInterrupt function is used to assign a new vector for the
designated interrupt.

*****/

void VectorInterrupt(int InterruptNumber, void interrupt (*ISR)())
{
    disable(); /* Don't want int called during revector */
    servect(InterruptNumber, ISR); /* Set the vector */
    enable(); /* Reenable interrupts */
}

/*****

NewISR

The NewISR is the interrupt service routine called when the DMA done
interrupt occurs. The ISR sets the DMADone flag so that the foreground
routines know that a buffer must be dumped to disk. Sampling is resumed
on the DMA06 (it is temporarily halted while the DMA done flag in the
status byte is set). The status byte of the current buffer is set to
indicate it must be dumped to disk. A new active buffer is swapped into
use with a check being made to see that the new buffer has been completely
saved to disk. If the new buffer still has unsaved data in it, the overflow
byte is set. Finally, the DMA controller is informed of the new buffer
and the End of Interrupt command is sent to the interrupt controller.

*****/

```

## DMASTRMR.C

```

ready to go before calling this function.

*****/
void DMAStart(char i)
{
    outportb(0x0A, DMADisable); /* Disable DMA on channel */

    outportb(DMAPageReg, DB[i].DMAPage);

    outportb(DMAOffsetReg, DB[i].PageOffsetLSB); /* LSB of offset into page */
    outportb(DMAOffsetReg, DB[i].PageOffsetMSB); /* MSB of offset into page */

    outportb(DMANumberOfBytesRegister, NumBytesLSB); /* LSB of # of bytes to transfer */
    outportb(DMANumberOfBytesRegister, NumBytesMSB); /* MSB of # of bytes to transfer */

    outportb(0x0A, DMAEnable); /* Enable DMA */
}

*****/

DMAShutdown

The DMAShutdown function disables further transfers by the DMA controller.
You should call this function when you terminate your program.

*****/

void DMAShutdown(void)
{
    outportb(0x0A, DMAChannel | 0x04); /* Disable DMA on channel */
}

*****/

InitializeBuffer

The InitializeBuffer function calculates the page number and offset into
that page of the specified buffer. This is required by the DMA controller
which uses a Page/Offset scheme instead of the more familiar Segment/Offset
scheme. A check is made to see whether the buffer straddles a page boundary.
If so, the program terminates.

*****/

void InitializeBuffer(DMABuffer *B)
{
    long LinearAddr;

```

---

```

*****/

void interrupt NewISR(void)
{
    DMADone = 1; /* Set DMA Done byte */

    outportb(DMA06ClearDMADone, 0);

    DB[ActiveBuffer].Status = 1; /* Buffer is full, must be dumped */

    if (ActiveBuffer == 0) /* Swap buffers */
        ActiveBuffer = 1;
    else
        ActiveBuffer = 0;

    if (DB[ActiveBuffer].Status == 1)
        BufferOverflow = 1;

    DMAStart(ActiveBuffer); /* Inform DMA controller of new buffer */

    outportb(0x20, 0x20); /* End of interrupt command to controller */
}

*****/

DMASetup

The DMASetup function set various parameters on the DMA controller.

*****/

void DMASetup(void)
{
    outportb(0x0A, DMAChannel | 0x04); /* Disable DMA on channel */
    outportb(0x0B, DMAChannel | 0x44); /* single, increment, noinit, cprintf */
    outportb(0x0C, 0); /* Clear byte pointer flip flop */
}

*****/

DMAStart

The DMAStart function sets up the DMA controller to place data in the
buffer whose index is passed as parameter i. After setting the address
and number of bytes to transfer, the DMA controller is enabled. After
this function is completed the DMA controller will automatically place
any data from the A-D converter into memory. Make certain that all is

```

## DMASTRMR.C

```

DMAOffsetReg = DMAChannel * 2;
DMANumberOfBytesRegister = DMAChannel * 2 + 1;

NumBytesMSB = (NumOfBytesToTransfer - 1) / 256;
NumBytesLSB = (NumOfBytesToTransfer - 1) % 256;

DMAEnable = DMAChannel | 0;
DMADisable = DMAChannel | 4;
}

/*****
InterruptShutDown

The InterruptShutDown function disables interrupts on the IRQ, restores
the original interrupt vector, and then restores the startup Interrupt
mask register. Be sure to call this function before exiting the program,
failing to do so will almost certainly cause a system hang. (Error handlers
should call this function if they cause early program termination).

*****/

void InterruptShutDown(void)
{
    SetBitInIMR(IRQ);          /* Disable interrupts on IRQ */
    servect(IRQ + 8, OldISR);  /* Restore original vector */
    outportb(0x21, OldIMR);    /* Restore original IMR */
}

/*****
TransferToDisk

The TransferToDisk function is used to transfer a data buffer that has
been filled by the interrupt service routine to disk. The number of
bytes actually written to disk is compared with the requested number and
if there is any discrepancy the program halts. (This is a very crude check
for disk or directory full error). Further, a check is made of the
BufferOverflow byte, and if it is set the program halts.

*****/

void TransferToDisk(void)
{
    char ErrorMessage[20];
    unsigned BytesWritten;

    if (DB[0].Status == 1)

```

```

    unsigned char PageNumber;
    unsigned Offset;

    LinearAddr = FP_SEG(B -> Buffer);
    LinearAddr = LinearAddr * 16 + FP_OFF(B -> Buffer);

    PageNumber = (LinearAddr / 65536);
    Offset = (LinearAddr % 65536);

    if ((long) (Offset + BUFFER_SIZE) > 65535)
    {
        #if DEBUG
        fprintf("DMA buffer straddles a PAGE boundary.");
        #endif
        beep(200,500);
        beep(150,400);
        exit(0);
    }

    B -> DMAPage = PageNumber;
    B -> PageOffsetMSB = Offset / 256;
    B -> PageOffsetLSB = Offset % 256;
    B -> Status = 0;
}

/*****
SetPortAddresses

The SetPortAddresses function calculates the addresses of different I/O
ports used in the program. Further, it pre-calculates the values of
several commands that must be sent to the DMA controller. This helps
speed up reprogramming the DMA controller during the DMA done interrupt.

*****/

void SetPortAddresses(void)
{
    DM406StatusByte = BaseAddress + STATUS_BYTE;
    DM406StartConversion = BaseAddress + START_CONVERSION;
    DM406ClearDMADone = BaseAddress + CLEAR_BOARD;

    switch(DMAChannel)
    {
        case 1 : DMAPageReg = 0x83; break;
        case 3 : DMAPageReg = 0x82; break;
    }
}

```

# DMASTRMR.C

```

{
    BytesWritten = fwrite(DB[0].Buffer, 1, NumOfBytesToTransfer, StreamingFile);
    DB[0].Status = 0;
    TransferCount++; /* Increment number of buffers dumped */
}

if (DB[1].Status == 1)
{
    BytesWritten = fwrite(DB[1].Buffer, 1, NumOfBytesToTransfer, StreamingFile);
    DB[1].Status = 0;
    TransferCount++; /* Increment number of buffers dumped */
}

DMADone = 0; /* Reset DMA Done byte */

/* Check for error conditions. If any, shut down and exit. */

if ((BufferOverflow == 1) || (BytesWritten != NumOfBytesToTransfer))
{
    fclose(StreamingFile);
    DMAShutdown();
    InterruptShutDown();
    if (BufferOverflow == 1)
        strcpy(ErrorMessage, "Overflow");
    else
        strcpy(ErrorMessage, "Disk printf");
    #if DEBUG
        clrscr();
        printf("%s error, program halted.", ErrorMessage);
    #endif
    beep(200,500);
    beep(150,400);
    exit(0);
}

/******
OpenStreamingFile

The OpenStreamingFile simply opens a disk file where the acquired data will
be streamed. If there is an error the program halts.

*****
void OpenStreamingFile(char S[80])
{
    if ((StreamingFile = fopen(S, "wb")) == NULL)

```

```

{
    #if DEBUG
        printf("Cannot open %s. Program halted", S);
    #endif
    beep(200,500);
    beep(150,400);
    exit(0);
}

/* *****
beep(frequency, duration)
/* a function to produce a sound from a PC speaker */
{
    /* unsigned int frequency, duration; (Hertz, msec) */
    sound(frequency);
    delay(duration);
    nosound();
    /* usage: beep(frequency, duration); */
}

/* *****
void main(void)
{
    float T0, DeltaT;
    unsigned char motor = 0;
    unsigned char Din = 0;

    #if DEBUG
        clrscr();
    #endif

    CLOCKS_PER_SEC = CLK_TCK; /* 18.20; */

    IRQ = 5; /* Set the IRQ */
    InitializeBoardSettings(768, 10.0); /* Set base address and volt range */
    ResetBoard(); /* Reset the board */
    ConfigureIOPorts(INPUT, OUTPUT);

    NumOfBytesToTransfer = 16384; /* Initialize Data Buffers */
    InitializeBuffer(&DB[0]);
    InitializeBuffer(&DB[1]);
    DMAChannel = 1; /* DMA channel to be used. Set this
    BEFORE calling SetPortAddresses */
    SetPortAddresses();

```

# DMASTRMR.C

```

while ( TransferCount <= 35 ) /* Run for 5 seconds */
{
    while (DMADone == 0) /* Wait until DMA Done */
    {
        TransferToDisk(); /* Transfer buffer to disk */
        #if DEBUG
        printf(" %i", TransferCount);
        #endif
    }
    WriteDigitalIO(2, 0); /* clear port 2 (C) */

    #if DEBUG
    printf("Done. Elapsed time: %7.3f seconds.\n\n", DeltaT);
    #endif
    beep(500,350);
    beep(800,350);

    #if DEBUG
    printf("Press any key to continue.\n\n");
    while(kbhit() == 0)
    {
    }
    getch();
    clrscr();
    #endif

    SetExternalTrigger(DISABLED); /* Disable external trigger */
    SetIRQStatus(DISABLED); /* Reset Interrupt source bit */
    DMAShutdown(); /* Disable further DMA */
    InterruptShutdown(); /* Disable interrupts on IRQ */
    ConfigureIOPorts(INPUT, INPUT);

    fclose(StreamingFile); /* Close DMA file */

    Din = 0;
    while(!Din)
    {
        Din = ReadDigitalIO(0) & Twos(6);
        #if DEBUG
        gotoxy(10,10);
        printf(" %u %u", Din, Twos(6));
        #endif
    }
    beep(500,350);
    beep(800,350);
}

**** END OF PROGRAM LISTING ****

```

---

```

OpenStreamingFile("TEST.STR");

OldIMR = inportb(0x21); /* Save startup IMR */
SetBitInIMR(IRQ); /* Temporarily mask interrupts on IRQ
    if not already masked */
PacerClockRate = 56000.00; /* Stream at 56 KHz */
SqSetPacerClock(PacerClockRate); /* Set the pacer clock to sampling rate */

SetIRQStatus(ENABLED); /* Interrupt on DMA done */

OldISR = getvect(IRQ + 8); /* Save startup vector */
VectorInterrupt(IRQ + 8, NewISR); /* Vector new interrupt */
ClearBitInIMR(IRQ); /* Enable interrupts on IRQ */

SetChannel(0); /* Sample from channel 0 */
ActiveBuffer = 0; /* Start using Buffer[0] */
TransferCount = 0; /* Initialize Transfer count */
DMADone = 0; /* Initialize DMA done byte */
DMASetup(); /* Setup DMA controller */

WriteDigitalIO(2, Twos(1)); /* enable external trigger receiver */

#if DEBUG
printf("Ready.\n");
#endif
beep(500,350);
beep(800,350);

while(!Din)
{
    Din = ReadDigitalIO(0) & Twos(7);
    #if DEBUG
    gotoxy(10,10);
    printf(" %u %u", Din, Twos(7));
    #endif
}

#if DEBUG
printf("\n Sampling... \n");
#endif
beep(500,200);

T0 = clock();
WriteDigitalIO(2, Twos(0)); /* port 2 (C) bit 0 to clear mux */

DMAStart(ActiveBuffer); /* Enable DMA controller */
SetExternalTrigger(ENABLED); /* Trigger from the pacer clock */

```

## FILESEND.C

Stored on host computer in executable form as *filesend.exe*. Used in conjunction with *recfile.exe* (on the onboard computer) to send the programs *sendfile.exe* and *dmastrmr.exe* to the onboard computer via the serial communications ports.

Compiled with Borland Turbo C.

Link the following during compilation:

d:\tc\lib\comm\filesend.c

d:\tc\lib\comm\pcl4c\_s.lib

# FILESEND.C

```

char *BaudRate[10] = {"300", "600", "1200", "2400", "4800", "9600",
    "19200", "38400", "57600", "115200"};
int BaudCode; /* baud rate code ( index into BaudRate[] ) */
char NCGchar = NAK;
int OneKflag = FALSE;
int BatchFlag = FALSE;
char Protocol = 'X';
static unsigned short CRCable[256];

/** main program */
main()
{
    int Delta; /* delta port status */
    int Status; /* port status */

    /* get port number */
    Port = 0; /* COM1 */

    /* set baud rate, refer to *Baudrate[] */
    BaudCode = 7;

    /* setup receive buffer */
    ErrorCheck( SioRxBuf(Port, RxBuf.Size2K));
    /* set parms & reset (initialize) COM port */
    ErrorCheck( SioParms(Port, NoParity, OneStopBit, WordLength8));
    MyStart(Port, BaudCode);
    /* set DTR and RTS */
    SioDTR(Port, 'S');
    SioRTS(Port, 'S');
    /* init CRC table */
    InitCRC0;
    #if RTS_CTS_CONTROL
    SioFlow(Port, 7*ONE_SECOND);
    /* printf("Flow Control enabled. CTS = "); */
    if(SioCTS(Port)) puts("ON");
    else puts("OFF");
    #endif
    /* Set FIFO level */
    if( SioFIFO(Port, LEVEL_14) ) puts("INS16550 detected");
    /* clear PCLAC receive buffer */
    ErrorCheck( SioRxFlush(Port));
    SioRxFlush(Port);
    Protocol = 'Y';
    OneKflag = TRUE;
    BatchFlag = TRUE;
    NCGchar = 'C';
}

```

```

/*
**
** --- fileSEND.c ---
**
** EXAMPLE CODE: File sender. Can transfer files using
** XMODEM, YMODEM, and YMODEM-G protocols.
**
** Same as SENDFILE.C; sends DMASTRMR.EXE and SENDFILE.EXE down to
** data logger.
**
** See TERM.H for configuration parameters.
**
** Do NOT select YMODEM-G when using a null modem cable unless you are
** certain that RTS & CTS are reversed ( this is usually not true ).
**
** Modified by R. Mueller, 12 March 1993
*/

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dos.h>

#include "pcl4c.h"
#include "ascii.h"
#include "term.h"

#define FALSE 0
#define TRUE !FALSE

#define DEBUG -1
#define DDEBUG 0
#define MAXTRY 5
#define LIMIT 20
#define POLY 0x1021

void MyStart();
void MyExit();
void ErrorCheck();
void InitCRC();

/** Global variables */
int Port; /* current COM port [0..3] */
char Filename[15]; /* file name buffer */
char Buffer[1024]; /* block buffer */
char RxBuf[2048]; /* receive buffer */

```

## FILESEND.C

```

/* YMODEM send */
strcpy(&Filename[0], "DMASTRMR.EXE\0");
TxyModem(Port, Filename, Buffer, OneKflag, BatchFlag);
SioDelay(5);

strcpy(&Filename[0], "SENDFILE.EXE\0");
TxyModem(Port, Filename, Buffer, OneKflag, BatchFlag);
SioDelay(5);

/* send empty filename */
Filename[0] = '\0';
TxyModem(Port, Filename, Buffer, OneKflag, BatchFlag);

/* was break detected ? */
if (SioBrtSig(Port, 'D'))
{
    #if DEBUG
        printf("BREAK detected ");
    #endif
    beep(200, 500);
    beep(150, 400);
}

/* any incoming over serial port ? */
GetChar(Port, 0);

/* any change in DCD or DSR ? */
Delta = SioModem(Port, DeltaDCD | DeltaDSR);
if (Delta)
{
    /* display new status */
    Status = SioModem(Port, (char)(DCD | DSR));
    if (!Status & DeltaDCD) MyExit(0, "Dropped DSD");
    if (!Status & DeltaDSR) MyExit(0, "Dropped DSR");
}
MyExit(0, "\0");

/* make multiple attempts to reset port */
void MyStart(Port, BaudCode)
int Port;
int BaudCode;
{int i, rc;
/* try up to 3 times to reset COM port */
for (i=0; i<3; i++)
{
    #if DEBUG
        printf("Resetting COM %d at %s baud\n", Port + 1, BaudRate[BaudCode]);
    #endif
    if (rc = SioReset(Port, BaudCode)) == 0) return;
    if (rc < 0) MyExit(rc, "Error resetting port");
    SioDone(Port);
    /* display errors */
    if (rc & OverrunError) puts("Overrun Error");
    if (rc & ParityError) puts("Parity Error");
    if (rc & FramingError) puts("Framing Error");
    if (rc & BreakDetect) puts("Break Detect");
}
    exit(1);
}

/* exit program */

void MyExit(code, msg)
int code;
char *msg;
{int rc;
if (code < 0) SioError(code);
/* Assert UART break & sign off */
SioBrtSig(Port, 'A');
#if DEBUG
    printf("\nTERMINATING: %s\n", msg);
#endif
}

SioDelay(ONE_SECOND/2);
SioDone(Port);
exit(0);

/* check for error */

void ErrorCheck(Code)
int Code;
/* trap PCL error codes */
if (Code < 0)
{SioError(Code);
 SioDone(Port);
 exit(1);
}

/* end ErrorCheck */

/* output character to serial port */
int PutChar(Port, ch)
int Port;

```

# FILESEND.C

```

char ch;
{int rc;
/* transmit character */
rc = SioPutc(Port,ch);
if(rc<0)
{
# if DEBUG
printf("SioPuc Error: COM%d: ",1+Port);
# endif
SioError(rc);
SioDone(Port);
exit(1);
}
return(rc);
}

/* receive character from serial port */

int GetChar(Port,Timeout)
int Port;
int Timeout;
{int rc;
rc = SioGetc(Port,Timeout);
if(rc<-1)
{
# if DEBUG
printf("SioGetc Error: COM%d: ",1+Port);
# endif
SioError(rc);
exit(1);
}
return(rc);
}

/* display the error text */

void SayError(Port,ptr)
int Port;
char *ptr;
{char temp[81];
sprintf(temp,"ERROR! COM%d : %s",1+Port,ptr);
# if DEBUG
printf("\n*** %s\n",temp);
# endif
beep(200,500);
beep(150,400);
/* cancel remote */
PutChar(Port,CAN);
}

PutChar(Port,CAN);
PutChar(Port,CAN);
} /* end SayError */

/* initialize CRC table */
void InitCRC()
{int i;
unsigned short CalcTable();
for(i=0;i<256;i++)CRCtable[i] = CalcTable(i,POLY,0);
}

/* calculate CRC table entry */
unsigned short CalcTable(data,genpoly,accum)
unsigned short data;
unsigned short genpoly;
unsigned short accum;
{static int i;
data <= 8;
for(i=8;i>0;i--)
{
if((data^accum) & 0x8000) accum = (accum <= 1) ^ genpoly;
else accum <= 1;
data <= 1;
}
return(accum);
}

/* compute updated CRC */
unsigned short UpdateCRC(crc,byte)
unsigned short crc;
unsigned char byte;
{
return( (crc <= 8) ^ CRCtable[ (crc > 8) ^ byte ] );
}

int TxyModem(Port,Filename,Buffer,OneKflag,BatchFlag)
int Port; /* COM port [0..3] */
char Filename[]; /* filename buffer */
char Buffer[]; /* 1024 byte data buffer */
int OneKflag; /* if TRUE, use 1K blocks when possible */
int BatchFlag; /* if TRUE, send filename in packet 0 */
{int i, k;
int Code; /* file Handle */
int Handle;
char c;
int p;
char PacketType;
char PacketNbr;
}

```

# FILESEND.C

```

int PacketSize;
int FirstPacket;
unsigned short CheckSum;
long filelength0;
int Number1K = 0; /* total # 1K packets */
int Number128 = 0; /* total # 128 byte packets */
char NCGchar = NAK;
long FileSize;
char temp[81];
int EmptyFlag = FALSE;
/* begin */
if(BatchFlag) if(FileName[0] == '\0') EmptyFlag = TRUE;
if(!EmptyFlag)
{ /* Filename is not empty */
    EmptyFlag = FALSE;
    Handle = open(FileName, O_RDONLY | O_BINARY, S_IREAD);
    if(Handle < 0)
    { strcpy(temp, "Cannot open ");
      strcat(temp, FileName);
      return(FALSE);
    }
}
}
/*if DEBUG
printf("XYMODEM send: waiting for Receiver\n");
#endif
beep(500,350);
beep(800,350);
while(SioKeyPress()) SioKeyRead0;
/* compute # blocks */
if(!EmptyFlag)
{ FileSize = filelength(Handle);
  if(OneKflag) Number1K = (int) (FileSize / 1024L);
  Number128 = 1 + (int) ((FileSize - 1024L * (long) Number1K - 1L) / 128L);
  printf(temp, "%d 1024 & %d 128 byte packets\n", Number1K, Number128);
}
/*if DEBUG
printf(temp);
#endif
else
{ /* empty file */
    Number128 = 0;
    Number1K = 0;
}
/* clear comm port ( there may be several NAKs queued up ) */
SioRxFush(Port);
/* get receivers start up NAK, 'C', or 'G' */
if(!TStartup(Port, &NCGchar)) return(FALSE);
/* loop over all packets */
if(BatchFlag) FirstPacket = 0;
else FirstPacket = 1;
for(p = FirstPacket; p <= Number1K + Number128; p++)
{ /* user aborts ? */
    if(SioKeyPress()) if((char) SioKeyRead0 == CAN)
    { TxCAN(Port);
      /*if DEBUG
      printf("**** Canceled by USER ****\n");
      #endif
      return(FALSE);
    }
    /* issue message */
    sprintf(temp, "Packet %d ", p);
    /*if DEBUG
    printf(temp);
    #endif
    /* load up Buffer */
    if(p == 0)
    {
        /* Filename packet ! */
        PacketSize = 128;
        k = 0;
        for(i = 0; i < strlen(FileName); i++) Buffer[k++] = FileName[i];
        Buffer[k++] = '\0';
        sprintf(temp, "%ld", FileSize);
        for(i = 0; i < strlen(temp); i++) Buffer[k++] = temp[i];
        while(k < 128) Buffer[k++] = '\0';
    }
    else /* p > 0 */
    { /* DATA Packet: use 1K or 128 byte block ? */
        if(p <= Number1K) PacketSize = 1024;
        else PacketSize = 128;
        /* read next block from disk */
        Code = read(Handle, Buffer, PacketSize);
        if(Code <= 0)
        { SayError(Port, "Error on disk read");
          return(FALSE);
        }
        for(i = Code; i < PacketSize; i++) Buffer[i] = 0x1a;
    }
    /* send this packet */
    if(!TxCAN(Port, p, PacketSize, Buffer, NCGchar)) return(FALSE);
    SioDelay(5);
    /* must 'restart' after non null packet 0 */
    if(!EmptyFlag && (p == 0)) TStartup(Port, &NCGchar);
    } /* end -- for(p) */
    /* done if empty packet 0 */
    if(EmptyFlag)

```

```

else PacketType = SOH;
PacketNbr &= 0x00ff;
/* make up to MAXTRY attempts to send this packet */
for(Attempt = 1; Attempt <= MAXTRY; Attempt++)
{
/* send SOH/STX */
Code = PutChar(Port, PacketType);
/* send packet # */
Code = PutChar(Port, PacketNbr);
/* send 1's complement of packet */
Code = PutChar(Port, 255 - PacketNbr);
/* send data */
CheckSum = 0;
for(i=0; i < PacketSize; i++)
{
Code = PutChar(Port, Buffer[i]);
if(NCGchar == NAK) CheckSum += Buffer[i];
else CheckSum = UpdateCRC(CheckSum, Buffer[i]);
}
/* send checksum */
if(NCGchar == NAK)
{
Code = PutChar(Port, CheckSum & 0x00ff);
}
else
{
Code = PutChar(Port, (CheckSum > 8) & 0x00ff);
Code = PutChar(Port, CheckSum & 0x00ff);
}
/* no ACK to wait for if 'G' */
if(NCGchar == 'G')
{
if(PacketNbr == 0) SioDelay(SHORT_WAIT*ONE_SECOND/2);
return(TRUE);
}
/* wait for receivers ACK */
Code = GetChar(Port, LONG_WAIT*ONE_SECOND);
if((char)Code == CAN)
{
/*if DEBUG
{
printf("*** Canceled by REMOTE ***\n");
}
endif
return(FALSE);
}
if((char)Code == ACK) return(TRUE);
if((char)Code == NAK)
{
PacketError(Port, PacketNbr, Attempt, "Out of sync");
return(FALSE);
}
/* Attempt again */
}
/* end -- for(Attempt) */
/* can't send packet ! */
SayError(Port, "packet timeout (3 NAKs)");
}
}

/*if DEBUG
printf("\nBatch transfer complete\n");
#endif

beep(500, 350);
beep(800, 350);
return(TRUE);
}

/* all done. send EOT up to 10 times */
close(Handle);
if(!TxEOT(Port))
{
SayError(Port, "EOT not acknowledged");
return(FALSE);
}

/*if DEBUG
printf("\nTransfer Complete\n");
#endif

beep(500, 350);
beep(800, 350);
return(TRUE);
} /* end -- TxyModem */

int TxCAN(Port)
int Port;
{
int i;
for(i=0; i < 6; i++) PutChar(Port, CAN);
return(0);
}

int TxPacket(Port, PacketNbr, PacketSize, Buffer, NCGchar)
int Port; /* COM port [0..3] */
int PacketNbr; /* Packet # to send */
int PacketSize; /* Packet size ( must be 128 or 1024 ) */
char Buffer[]; /* Data buffer */
char NCGchar; /* NAK, 'C', or 'G' */
{
int i;
int Code;
unsigned short CheckSum;
int Attempt;
int PacketType;
char temp[81];
/* begin */
/*if DDEBUG
printf("TxP: COM %d PacketNbr = %d PacketSize = %d NCGchar = %c",
1 + Port, PacketNbr, PacketSize, NCGchar);
#endif
/* better be 128 or 1024 packet length */
if(PacketSize == 1024) PacketType = STX;

```

# FILESEND.C

```

return(FALSE);
} /* end -- TxPacket */

PacketError(Port, Packet, Attempt, MsgPtr)
int Port;
int Packet;
int Attempt;
char *MsgPtr;
{char temp[81];
 printf(temp, "Packet %d : Attempt %d : %s", Packet, Attempt, MsgPtr);
 SayError(Port, temp);
}

int TxStartup(Port, NCGcharPtr)
int Port;
char *NCGcharPtr;
{int i;
 int Code;
 #if DDEBUG
 printf("### TxStartup");
 #endif
 /* clear Rx buffer */
 SioRxFlush(Port);
 /* wait for receivers start up NAK, 'C', or 'G' */
 for(i=1; i<LIMIT; i++)
 {if(SioKeyPress())
 {SayError(Port, "Aborted by user");
 return(FALSE);
 }
 Code = GetChar(Port, SHORT_WAIT*ONE_SECOND);
 if(Code == -1) continue;
 /* received a byte */
 if((char)Code == NAK)
 {*NCGcharPtr = NAK;
 #if DDEBUG
 printf("(CS) OK\n");
 #endif
 return(TRUE);
 }
 if((char)Code == 'C')
 {*NCGcharPtr = 'C';
 #if DDEBUG
 printf("(CRC) OK\n");
 #endif
 return(TRUE);
 }
 if((char)Code == 'G')
 {*NCGcharPtr = 'G';

```

---

```

 #if DDEBUG
 printf("(G) OK\n");
 #endif
 return(TRUE);
 } /* end -- for(i) */
 /* no response */
 SayError(Port, "No response from receiver");
 return(FALSE);
 } /* end -- TxStartup */

int TxEOT(Port)
int Port;
{int i;
 int Code;
 for(i=0; i<10; i++)
 {Code = PutChar(Port, EOT);
 /* await response */
 Code = GetChar(Port, SHORT_WAIT*ONE_SECOND);
 if((char)Code == ACK) return(TRUE);
 } /* end -- for(i) */
 return(FALSE);
 } /* end -- TxEOT */

beep(frequency, duration)
/* a function to produce a sound from a PC speaker */
{
 /* unsigned int frequency, duration; */
 sound(frequency);
 delay(duration);
 nosound();
 /* usage: beep(frequency, duration); */
}

**** END OF PROGRAM LISTING ****

```

## SENDFILE.C

Sent to onboard computer in executable form as *sendfile.exe*. Used in conjunction with *filerec.exe* (on the host computer) to send the data file *test.str* (data file created during the experiment run) to the host computer via the serial communications ports.

Compiled with Borland Turbo C.

Link the following during compilation:

d:\tc\lib\comm\sendfile.c

d:\tc\lib\comm\pcl4c\_s.lib

# SENDFILE.C

```

/*
**
**      --- sendfile.c ---
**
**  EXAMPLE CODE: File sender. Can transfer files using
**  XMODEM, YMODEM, and YMODEM-G protocols.
**
**  See TERM.H for configuration parameters.
**
**  Do NOT select YMODEM-G when using a null modem cable unless you are
**  certain that RTS & CTS are reversed ( this is usually not true ).
**
**  Modified by R. Mueller, 12 March 1993
**
*/

#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dos.h>

#include "pcide.h"
#include "ascii.h"
#include "term.h"

#define FALSE 0
#define TRUE 1

#define DEBUG 0
#define DDEBUG 0
#define MAXTRY 5
#define LIMIT 20
#define POLY 0x1021

void MyStart();
void MyExit();
void ErrorCheck();
void InitCRC();

/*** Global variables ***/

int Port; /* current COM port [0..3] */
char Filename[15]; /* file name buffer */
char Buffer[1024]; /* block buffer */
char RxBuf[2048]; /* receive buffer */
char *BaudRate[10] = {"300", "600", "1200", "2400", "4800", "9600",
"19200", "38400", "57600", "115200"};

int BaudCode; /* baud rate code ( index into BaudRate[] ) */

char NCGchar = NAK;
int OneKflag = FALSE;
int BatchFlag = FALSE;
char Protocol = 'X';
static unsigned short CRCtable[256];

/*** main program ***/

main()
{
    int Delta; /* delta port status */
    int Status; /* port status */

    /* get port number */
    Port = 0; /* COM1 */

    /* set baud rate, refer to *BaudRate[] */
    BaudCode = 7;

    /* setup receive buffer */
    ErrorCheck( SioRxBuf(Port, RxBuf, Size2K) );
    /* set parms & reset (initialize) COM port */
    ErrorCheck( SioParms(Port, NoParity, OneStopBit, WordLength8) );
    MyStart(Port, BaudCode);
    /* set DTR and RTS */
    SioDTR(Port, 'S');
    SioRTS(Port, 'S');
    /* init CRC table */
    InitCRC();
    #if RTS_CTS_CONTROL
    SioFlow(Port, 7*ONE_SECOND);
    /* printf("Flow Control enabled. CTS = "); */
    if(SioCTS(Port)) puts("ON");
    else puts("OFF");
    #endif
    /* Set FIFO level */
    if( SioFIFO(Port, LEVEL_14) ) puts("INS16550 detected");
    /* clear PCLAC receive buffer */
    ErrorCheck( SioRxFlush(Port) );

    SioRxFlush(Port);
    Protocol = 'Y';
    OneKflag = TRUE;
    BatchFlag = TRUE;
    NCGchar = 'C';

    strcpy(&Filename[0], "TEST.STR0");
    /* YMODEM send */

```

# SENDFILE.C

```

TxyModem(Port,Filename,Buffer,OneKflag,BatchFlag);
/* send empty filename */
Filename[0] = '\0';
SioDelay(5);
TxyModem(Port,Filename,Buffer,OneKflag,BatchFlag);

/* was break detected ? */
if( SioBrtSig(Port,'D'))
{
    #if DEBUG
        printf("BREAK detected ");
    #endif
    beep(200,500);
    beep(150,400);
}
/* any incoming over serial port ? */
GetChar(Port,0);

/* any change in DCD or DSR ? */
Delta = SioModem(Port,DeltaDCD|DeltaDSR);
if(Delta)
{
    /* display new status */
    Status = SioModem(Port,(char)(DCD|DSR));
    if(!Status&DeltaDCD) MyExit(0,"Dropped DSD");
    if(!Status&DeltaDSR) MyExit(0,"Dropped DSR");
}
MyExit(0,"0");
}

/* make multiple attempts to reset port */
void MyStart(Port,BaudCode)
int Port;
int BaudCode;
{
    int i, rc;
    /* try up to 3 times to reset COM port */
    for(i=0;i<3;i++)
    {
        #if DEBUG
            printf("Resetting COM%d at %s baud\n",Port+1,BaudRate[BaudCode]);
        #endif
        if( (rc = SioReset(Port,BaudCode)) == 0) return;
        if(rc < 0) MyExit(rc,"Error resetting port");
        SioDone(Port);
        /* display errors */
        if(rc&OverrunError) puts("Overrun Error");
    }
}

if(rc&ParityError) puts("Parity Error");
if(rc&FramingError) puts("Framing Error");
if(rc&BreakDetect) puts("Break Detect");
}
exit(1);
}

/* exit program */
void MyExit(code,msg)
int code;
char *msg;
{
    int rc;
    if(code < 0) SioError(code);
    /* Assert UART break & sign off */
    SioBrtSig(Port,'A');
    #if DEBUG
        printf("TERMINATING: %s\n",msg);
    #endif
    SioDelay(ONE_SECOND/2);
    SioDone(Port);
    exit(0);
}

/* check for error */
void ErrorCheck(Code)
int Code;
{
    /* trap PCL error codes */
    if(Code < 0)
    {
        SioError(Code);
        SioDone(Port);
        exit(1);
    }
} /* end ErrorCheck */

/* output character to serial port */
int PutChar(Port,ch)
int Port;
char ch;
{
    int rc;
    /* transmit character */
    rc = SioPutc(Port,ch);
    if(rc < 0)
    {
        #if DEBUG
            printf("SioPutc Error: COM%d: ",1+Port);

```

## 96

96

# SENDFILE.C

```

char temp[81];
int EmptyFlag = FALSE;
/* begin */
if(!BatchFlag) if(Filename[0] == '\0') EmptyFlag = TRUE;
if(!EmptyFlag)
{
    /* Filename is not empty */
    EmptyFlag = FALSE;
    Handle = open(Filename, O_RDONLY | O_BINARY, S_IREAD);
    if(Handle < 0)
    {
        strcpy(temp, "Cannot open ");
        strcat(temp, Filename);
        return(FALSE);
    }
}

/*if DEBUG
printf("XYMODEM send: waiting for Receiver \n");
#endif
beep(500,350);
beep(800,350);
while(SioKeyPress()) SioKeyRead();
/* compute # blocks */
if(!EmptyFlag)
{
    FileSize = filelength(Handle);
    if(OneKFlag) Number1K = (int) (FileSize / 1024L);
    Number128 = 1 + (int) ((FileSize - 1024L * (long) Number1K - 1L) / 128L);
    sprintf(temp, "%d 1024 & %d 128 byte packets \n", Number1K, Number128);
}
/*if DEBUG
printf(temp);
#endif
else
{
    /* empty file */
    Number128 = 0;
    Number1K = 0;
}

/* clear comm port ( there may be several NAKs queued up ) */
SioRxFlush(Port);
/* get receivers start up NAK, 'C', or 'G' */
if(!TxStartup(Port, &NCGchar)) return(FALSE);
/* loop over all packets */
if(!BatchFlag) FirstPacket = 0;
else FirstPacket = 1;
for(p = FirstPacket; p <= Number1K + Number128; p++)
{
    /* user aborts ? */
    if(SioKeyPress()) if(((char) SioKeyRead) == CAN)
    {
        TxCAN(Port);
        /*if DEBUG
        printf("*** Canceled by USER ***\n");
        */
    }
}

#endif
return(FALSE);
}
/* issue message */
sprintf(temp, "Packet %d ", p);
/*if DEBUG
printf(temp);
#endif
/* load up Buffer */
if(p == 0)
{
    /* Filename packet ! */
    PacketSize = 128;
    k = 0;
    for(i=0; i < strlen(Filename); i++) Buffer[k++] = Filename[i];
    Buffer[k++] = '\0';
    sprintf(temp, "%ld", FileSize);
    for(i=0; i < strlen(temp); i++) Buffer[k++] = temp[i];
    while(k < 128) Buffer[k++] = '\0';
}
else /* p > 0 */
{
    /* DATA Packet: use 1K or 128 byte block ? */
    if(k <= Number1K) PacketSize = 1024;
    else PacketSize = 128;
    /* read next block from disk */
    Code = read(Handle, Buffer, PacketSize);
    if(Code <= 0)
    {
        SayError(Port, "Error on disk read");
        return(FALSE);
    }
    for(i=Code; i < PacketSize; i++) Buffer[i] = 0x1a;
}
/* send this packet */
if(!TxPacket(Port, p, PacketSize, Buffer, NCGchar)) return(FALSE);
SioDelay(5);
/* must 'restart' after non null packet 0 */
if(!EmptyFlag && (p == 0)) TxStartup(Port, &NCGchar);
} /* end -- for(p) */
/* done if empty packet 0 */
if(EmptyFlag)
{
    /*if DEBUG
    printf("\nBatch transfer complete\n");
    */
    beep(500,350);
    beep(800,350);
    return(TRUE);
}
}

```

# SENDFILE.C

```

/* all done. send EOT up to 10 times */
close(Handle);
if(!TxEOT(Port))
{
    SayError(Port, "EOT not acknowledged");
    return(FALSE);
}
}
#endif
/*if DEBUG
printf("\nTransfer Complete\n");
#endif
beep(500,350);
beep(800,350);
return(TRUE);
} /* end -- TxyModem */

int TxCAN(Port)
int Port;
{
    int i;
    for(i=0;i<6;i++) PutChar(Port,CAN);
    return(0);
}

int TxPacket(Port,PacketNbr,PacketSize,Buffer,NCGchar)
int Port; /* COM port [0..3] */
int PacketNbr; /* Packet # to send */
int PacketSize; /* Packet size ( must be 128 or 1024 ) */
char Buffer[]; /* Data buffer */
char NCGchar; /* NAK, 'C', or 'G' */
{
    int i;
    int Code;
    unsigned short CheckSum;
    int Attempt;
    int PacketType;
    char temp[81];
    /* begin */
    #if DEBUG
    printf("TXP: COM %d PacketNbr = %d PacketSize = %d NCGchar = %c",
        1 + Port, PacketNbr, PacketSize, NCGchar);
    #endif
    /* better be 128 or 1024 packet length */
    if(PacketSize == 1024) PacketType = STX;
    else PacketType = SOH;
    PacketNbr &= 0x00ff;
    /* make up to MAXTRY attempts to send this packet */
    for(Attempt=1; Attempt<=MAXTRY; Attempt++)
    {
        /* send SOH/STX */
        Code = PutChar(Port, PacketType);
        /* send packet # */
        Code = PutChar(Port, PacketNbr);
    }
}

/* send 1's complement of packet */
Code = PutChar(Port, 255 - PacketNbr);
/* send data */
CheckSum = 0;
for(i=0; i<PacketSize; i++)
{
    Code = PutChar(Port, Buffer[i]);
    if(NCGchar == NAK) CheckSum += Buffer[i];
    else CheckSum = UpdateCRC(CheckSum, Buffer[i]);
}
/* send checksum */
if(NCGchar == NAK)
{
    Code = PutChar(Port, CheckSum & 0x00ff);
}
else
{
    Code = PutChar(Port, (CheckSum > 8) & 0x00ff);
    Code = PutChar(Port, CheckSum & 0x00ff);
}
/* no ACK to wait for if 'G' */
if(NCGchar == 'G')
{
    if(PacketNbr == 0) SioDelay(SHORT_WAIT*ONE_SECOND/2);
    return(TRUE);
}
/* wait for receivers ACK */
Code = GetChar(Port, LONG_WAIT*ONE_SECOND);
if((char)Code == CAN)
{
    #if DEBUG
    printf("**** Canceled by REMOTE ****\n");
    #endif
    return(FALSE);
}
if((char)Code == ACK) return(TRUE);
if((char)Code != NAK)
{
    PacketError(Port, PacketNbr, Attempt, "Ousof sync");
    return(FALSE);
}
/* Attempt again */
} /* end -- for(Attempt) */
/* can't send packet ! */
SayError(Port, "packet timeout (3 NAKs)");
return(FALSE);
} /* end -- TxPacket */

PacketError(Port, Packet, Attempt, MsgPtr)
int Port;
int Packet;
int Attempt;
char *MsgPtr;

```

# SENDFILE.C

```

{char temp[81];
sprintf(temp, "Packet %d : Attempt %d : %s", Packet, Attempt, MsgPtr);
SayError(Port, temp);
}

int TxStartup(Port, NCGcharPtr)
int Port;
char *NCGcharPtr;
{int i;
int Code;
#if DDEBUG
printf("## TxStartup");
#endif
/* clear Rx buffer */
SioRxFlush(Port);
/* wait for receivers start up NAK, 'C', or 'G' */
for(i=1; i<LIMIT; i++)
{if(SioKeyPress())
{SayError(Port, "Aborted by user");
return(FALSE);
}
Code = GetChar(Port, SHORT_WAIT*ONE_SECOND);
if(Code == -1) continue;
/* received a byte */
if((char)Code == NAK)
{NCGcharPtr = NAK;
printf("(CS) OK\n");
return(TRUE);
}
if((char)Code == 'C')
{NCGcharPtr = 'C';
#if DDEBUG
printf("(CRC) OK\n");
#endif
return(TRUE);
}
if((char)Code == 'G')
{NCGcharPtr = 'G';
#if DDEBUG
printf("(G) OK\n");
#endif
return(TRUE);
}
} /* end -- for(i) */
/* no response */
SayError(Port, "No response from receiver");
}

return(FALSE);
} /* end -- TxStartup */

int TxEOT(Port)
int Port;
{int i;
int Code;
for(i=0; i<10; i++)
{Code = PutChar(Port, EOT);
/* wait response */
Code = GetChar(Port, SHORT_WAIT*ONE_SECOND);
if((char)Code == ACK) return(TRUE);
} /* end -- for(i) */
return(FALSE);
} /* end -- TxEOT */

beep(frequency, duration)
/* a function to produce a sound from a PC speaker */
{
/* unsigned int frequency, duration; */
sound(frequency);
delay(duration);
nosound();
/* usage: beep(frequency, duration); */
}

**** END OF PROGRAM LISTING ****

```

## FILEREC.C

Stored on the host computer in executable form as *filerec.exe*. Used in conjunction with *sendfile.exe* (on the onboard computer) to receive the downloaded binary data file *test.str* via the serial communications ports. The program also reminds the user to change the name of the data file after downloading.

# FILEREC.C

```

/*
**      --- receivefile.c ---
**
**      EXAMPLE CODE: File receiver. Can receive files using
**      XMODEM, YMODEM, and YMODEM-G protocols.
**
**      See TERM.H for configuration parameters.
**
**      Do NOT select YMODEM-G when using a null modem cable unless you are
**      certain that RTS & CTS are reversed ( this is usually not true ).
**
**
** */
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dos.h>

#include "pcl4c.h"
#include "ascii.h"
#include "term.h"

#define FALSE 0
#define TRUE 1

#define DEBUG -1
#define DDEBUG 0
#define MAXTRY 5
#define LIMIT 20
#define POLY 0x1021

void MyStart0;
void MyExit0;
void ErrorCheck0;
void InitCRC0;

/**** Global variables ****/

int Port; /* current COM port [0..3] */
char Filename[15]; /* file name buffer */
char Buffer[1024]; /* block buffer */
char RxBuf[2048]; /* receive buffer */
char *BaudRate[10] = {"300", "600", "1200", "2400", "4800", "9600",
"19200", "38400", "57600", "115200"};
int BaudCode; /* baud rate code ( index into BaudRate[] ) */

char NCGchar = NAK;
int OneKflag = FALSE;
int BatchFlag = FALSE;
char Protocol = 'X';
static unsigned short CRCTable[256];

/**** main program ****/

main()
{
    int Delta; /* delta port status */
    int Status; /* port status */

    /* get port number */
    Port = 0; /* COM1 */
    /* set baud rate */
    BaudCode = 7;

    /* setup receive buffer */
    ErrorCheck( SioRxBuf(Port, RxBuf, Size2K));
    /* set parms & reset (initialize) COM port */
    ErrorCheck( SioParms(Port, NoParity, OneStopBit, WordLength8));
    MyStart(Port, BaudCode);
    /* set DTR and RTS */
    SioDTR(Port, 'S');
    SioRTS(Port, 'S');
    /* init CRC table */
    InitCRC0;
    /*if RTS_CTS_CONTROL
    SioFlow(Port, 7*ONE_SECOND);
    /* printf("Flow Control enabled. CTS = "); */
    if(SioCTS(Port)) puts("ON");
    else puts("OFF");
    #endif
    /* Set FIFO level */
    if( SioFIFO(Port, LEVEL_14)) puts("INS16550 detected");
    /* clear PCLAC receive buffer */
    ErrorCheck( SioRxFlush(Port));

    SioRxFlush(Port);
    Protocol = 'Y';
    OneKflag = TRUE;
    BatchFlag = TRUE;
    NCGchar = 'C';

    strcpy(&Filename[0], "TEST.STR0");
    /* XMODEM/YMODEM receive */
    while( strcmp(Filename, "0") )

```

## 102

```

{
    RxY.Modem(Port,Filename,Buffer,OneKFlag,BatchFlag);
}
/*
if (kbhit())
{
    getch();
    break;
}
*/
}

/* was break detected ? */
if (SioBrtSig(Port,'D'))
{
    #if DEBUG
    printf("BREAK detected ");
    #endif
    beep(200,500);
    beep(150,400);
}
/* any incoming over serial port ? */
GetChar(Port,0);

/* any change in DCD or DSR ? */
Delta = SioModem(Port,DeltaDCD|DeltaDSR);
if(Delta)
{
    /* display new status */
    Status = SioModem(Port,(char)(DCD|DSR));
    if(!Status&DeltaDCD) MyExit(0,"Dropped DSD");
    if(!Status&DeltaDSR) MyExit(0,"Dropped DSR");
}
MyExit(0,"0");
}

/* make multiple attempts to reset port */
void MyStart(Port,BaudCode)
int Port;
int BaudCode;
{
    int i, rc;
    /* try up to 3 times to reset COM port */
    for(i=0;i<3;i++)
    {
        #if DEBUG
        printf("Resetting COM%d at %s baud\n",Port+1,BaudRate[BaudCode]);
        #endif
    }
}

```

# FILEREC.C

```

char ch;
{int rc;
/* transmit character */
rc = SioPutc(Port,ch);
if(rc<0)
{
#if DEBUG
printf("SioPutc Error: COM%d: ",1+Port);
#endif
SioError(rc);
SioDone(Port);
exit(1);
}
return(rc);
}

/** receive character from serial port **/

int GetChar(Port,Timeout)
int Port;
int Timeout;
{int rc;
rc = SioGetc(Port,Timeout);
if(rc<-1)
{
#if DEBUG
printf("SioGetc Error: COM%d: ",1+Port);
#endif
SioError(rc);
exit(1);
}
return(rc);
}

/** display the error text **/

void SayError(Port,ptr)
int Port;
char *ptr;
{char temp[81];
sprintf(temp,"ERROR! COM%d : %s",1+Port,ptr);
#if DEBUG
printf("\n*** %s\n",temp);
#endif
beep(200,500);
beep(150,400);
/* cancel remote */
PutChar(Port,CAN);
}

PutChar(Port,CAN);
PutChar(Port,CAN);
} /* end SayError */

/* initialize CRC table */
void InitCRC()
{int i;
unsigned short CalcTable0;
for(i=0;i<256;i++) CRCTable[i] = CalcTable(i,POLY,0);
}

/* calculate CRC table entry */
unsigned short CalcTable(data,genpoly,accum)
unsigned short data;
unsigned short genpoly;
unsigned short accum;
{static int i;
data <= 8;
for(i=8;i>0;i--)
{
if((data^accum)& 0x8000) accum = (accum < 1) ^ genpoly;
else accum <= 1;
data <= 1;
}
return(accum);
}

/* compute updated CRC */
unsigned short UpdateCRC crc,byte)
unsigned short crc;
unsigned char byte;
{
return( (crc < 8) ^ CRCTable[ (crc > 8) ^ byte ] );
}

int RxyModem(Port,Filename,Buffer,NCGchar,BatchFlag)
int Port; /* COM port [0..3] */
char Filename[]; /* filename buffer */
char Buffer[]; /* 1024 byte data buffer */
char NCGchar; /* NAK, 'C', or 'G' */
int BatchFlag; /* if TRUE, get filename from packet 0 */
{int i;
int Handle; /* file Handle */
int p; /* packet index */
int Code; /* return code */
int FirstPacket;
char PacketNbr;
int PacketSize; /* 128 or 1024 */
}

```

# FILEREC.C

```

strcpy(Filename, Buffer);
/* done if null packet 0 */
if(Filename[0] == '\0')
{
    #if DEBUG
        printf("Batch Transfer Complete\n");
    #endif
    beep(500,350);
    beep(800,350);
    return(TRUE);
}
/* all done if EOT was received */
if(EOTFlag)
{
    close(Handle);
    #if DEBUG
        printf("Transfer Complete\n");
    #endif
    return(TRUE);
}
/* process packet */
if(p == 0)
{
    /* open file using filename in packet 0 */
    open(Filename, O_CREAT|O_TRUNC|O_WRONLY|O_BINARY, S_IWRITE);
    if(Handle < 0)
    {
        srcat(Buffer, " -- open failed");
        #if DEBUG
            printf("%s\n", Buffer);
        #endif
        return(FALSE);
    }
    /* get file length */
    FileSize = atoi(&Buffer[1 + strlen(Buffer)]);
    /* must 'restart' after packet 0 */
    RxStartup(Port, &NCGchar);
}
else /* DATA packet */
{
    /* write Buffer */
    if(BatchFlag)
    {
        if(FileSize < (long)PacketSize) i = (int) FileSize;
        else i = PacketSize;
        i = write(Handle, Buffer, i);
        FileSize -= (long)i;
    }
    else write(Handle, Buffer, PacketSize);
} /* end -- else */
} /* end -- for(p) */

```

```

long atoi();
long FileSize;
char temp[81];
int EOTFlag = FALSE;
/* begin */
EOTFlag = FALSE;
#if DEBUG
    printf("XYMODEM Receive: Waiting for Sender ");
#endif
beep(500,350);
beep(800,350);
while(SioKeyPress() SioKeyRead0;
/* clear connn port */
SioRxFlush(Port);
/* Send NAKs, 'C's, or 'G's */
if(!RxStartup(Port, &NCGchar))return(FALSE);
/* open file unless BatchFlag is on */
if(BatchFlag) FirstPacket = 0;
else
{
    /* start with packet 1 */
    FirstPacket = 1;
}
/* open file passed in Filename[] for write */
Handle=open(Filename, O_CREAT|O_TRUNC|O_WRONLY|O_BINARY, S_IWRITE);
if(Handle < 0)
{
    strcpy(temp, "Cannot open ");
    srcat(temp, Filename);
    #if DEBUG
        printf("%s\n", temp);
    #endif
    return(FALSE);
}
}
/* get each packet in turn */
for(p = FirstPacket; p < +)
{
    /* user aborts ? */
    if(SioKeyPress() if((char)SioKeyRead0 == CAN)
    {
        TxCAN(Port);
        return(FALSE);
    }
    /* issue message */
    sprintf(temp, "Packet %d", p);
    #if DEBUG
        printf("%s", temp);
    #endif
    /* get next packet */
    if(!RxPacket(Port, p, &PacketSize, Buffer, NCGchar, &EOTFlag))return(FALSE);
    if(p == 0)
    {
        /* copy Filename */

```

# FILERE.C

```

} /* end - RxyModem */

int TxCAN(Port)
int Port;
{int i;
 for(i=0; i<6; i++) PutChar(Port,CAN);
 return(0);
}

int RxPacket(Port,PacketNbr,PacketSizePtr,Buffer,NCGchar,EOTptr)
int Port; /* COM port [0..3] */
int PacketNbr; /* Packet # expected */
int *PacketSizePtr; /* Pointer to PacketSize received ( 128 or 1024) */
char Buffer[]; /* 1024 byte data buffer */
char NCGchar; /* NAK, C, or G */
int *EOTptr; /* Pointer to EOT flag */
{int i;
 int Code;
 int Attempt;
 int RxPacketNbr;
 int RxPacketNbrComp;
 unsigned short CheckSum;
 unsigned short RxCheckSum;
 unsigned short RxCheckSum1, RxCheckSum2;
 /* char PacketType; */
 char temp[81];
 /* begin */
 #if DDEBUG
 printf("RXP: COM %d PacketNbr = %d NCGchar = %c EOTflag = %d",
 1+Port,PacketNbr,NCGchar,*EOTptr);
 #endif
 PacketNbr &= 0x00ff;
 for(Attempt=1; Attempt<=MAXTRY; Attempt++)
 {/* wait for SOH / STX */
 Code = GetChar(Port,SHORT_WAIT*ONE_SECOND);
 if(Code== -1)
 {PacketError(Port,PacketNbr,Attempt,"timedout waiting for SOH/STX");
 return(FALSE);
 }
 switch((char)Code)
 {
 case SOH:
 /* 128 byte buffer incoming */
 /*PacketType = SOH; */
 *PacketSizePtr = 128;
 break;
 case STX:
 /* 1024 byte buffer incoming */

```

# FILEREC.C

```

/* compute CRC or checksum */
if(NCGchar!=NAK) CheckSum = UpdateCRC(CheckSum,Code);
else CheckSum = (CheckSum + Code) & 0x00ff;
}
/* receive CRC/checksum */
if(NCGchar!=NAK)
{
/* receive 2 byte CRC */
Code = GetChar(Port,SHORT_WAIT*ONE_SECOND);
if(Code==--1)
{PacketError(Port,PacketNbr,Attempt,"timedout waiting for 1st CRC byte");
return(FALSE);
}
RxCheckSum1 = Code & 0x00ff;
Code = GetChar(Port,SHORT_WAIT*ONE_SECOND);
if(Code==--1)
{PacketError(Port,PacketNbr,Attempt,"timedout waiting for 2nd CRC byte");
return(FALSE);
}
RxCheckSum2 = Code & 0x00ff;
RxCheckSum = (RxCheckSum1 < 8) | RxCheckSum2;
}
else
{
/* receive one byte checksum */
Code = GetChar(Port,SHORT_WAIT*ONE_SECOND);
if(Code==--1)
{PacketError(Port,PacketNbr,Attempt,"timedout waiting for checksum");
return(FALSE);
}
RxCheckSum = Code & 0x00ff;
}
/* don't send ACK if 'G' */
if(NCGchar=='G') return(TRUE);
/* checksum OK ? */
if(RxCheckSum!=CheckSum)
{
/*if DEBUG
printf("Bad packet checksum\n");
#endif
Code = PutChar(Port,NAK);
}
/* packet number OK ? */
else if(RxPacketNbr!=PacketNbr)
{
/*if DEBUG
printf("Bad packet number\n");
#endif
Code = PutChar(Port,NAK);
}
}

else
{
/* ACK the packet */
PutChar(Port,ACK);
return(TRUE);
}
/* end if */
}
/* end -- for(Attempt) */
/* can't receive packet */
SayError(Port,"RX packet timeout");
return(FALSE);
}
/* end -- RxPacket */

PacketError(Port,Packet,Attempt,MsgPtr)
int Port;
int Packet;
int Attempt;
char *MsgPtr;
{char temp[81];
sprintf(temp,"Packet %d : Attempt %d : %s",Packet,Attempt,MsgPtr);
SayError(Port,temp);
}

int RxStartup(Port,NCGcharPtr)
int Port;
char *NCGcharPtr;
{int i;
int Code;
/*if DDEBUG
printf("### RxStartup");
printf(" %c[%xH]",*NCGcharPtr,*NCGcharPtr);
#endif
/* clear Rx buffer */
SioRxFlush(Port);
/* Send NAKs, 'C's, or 'G's */
for(i=1;i<LIMIT;i++)
{if(SioKeyPress())
{printf("**** Canceled by USER ****");
return(FALSE);
}
}
/* stop attempting CRC/'G' after 1st 4 tries */
if((*NCGcharPtr!=NAK)&&(i==5))*NCGcharPtr = NAK;
/* tell sender that I am ready to receive */
Code = PutChar(Port,*NCGcharPtr);
Code = GetChar(Port,SHORT_WAIT*ONE_SECOND);
if(Code==--1) continue;
/* no error -- must be incoming byte -- push byte back onto queue */
SioUnGetc(Port,(char)Code);
/*if DDEBUG

```

```

        printf("OK\n");
    #endif
    return(TRUE);
} /* end -- for(i) */
/* no response */
SayError(Port, "No response from sender");
return(FALSE);
} /* end -- RxStartup */

int TxEOT(Port)
int Port;
{int i;
 int Code;
 for(i=0; i<10; i++)
 {Code = PutChar(Port, EOT);
  /* await response */
  Code = GetChar(Port, SHORT_WAIT*ONE_SECOND);
  if((char)Code == ACK) return(TRUE);
  } /* end -- for(i) */
 return(FALSE);
} /* end -- TxEOT */

beep(frequency, duration)
/* a function to produce a sound from a PC speaker */
{
    /* unsigned int frequency, duration; */
    sound(frequency);
    delay(duration);
    nosound();
    /* usage: beep(frequency, duration); */
}

```

## RECFILE.C

Same program as *filerec.c*. Stored on the onboard computer's EEPROM in executable form as *recfile.exe*. Used in conjunction with *filesend.exe* (on the host computer) to receive the downloaded programs *sendfile.exe* and *dmastrmr.exe* via the serial communications ports. Initiated during bootup sequence of onboard computer.

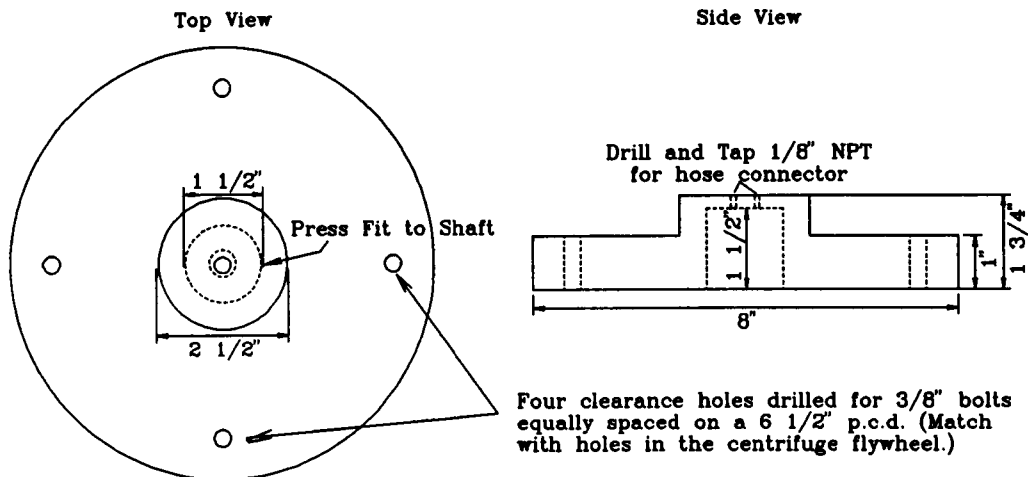
## Appendix 2: Centrifuge Part Description

|                                                                |     |
|----------------------------------------------------------------|-----|
| A: Centrifuge Motion Package Module . . . . .                  | 110 |
| A.1 Flywheel Support Flange . . . . .                          | 110 |
| A.2 Rotation Motor . . . . .                                   | 110 |
| A.3 Centrifuge Gas Delivery Inlet and Support System . . . . . | 111 |
| A.4 Support Block . . . . .                                    | 112 |
| B: Process Module . . . . .                                    | 113 |
| B.1 Centrifuge Flywheel . . . . .                              | 113 |
| B.2 Central Turning Mirror Assembly . . . . .                  | 113 |
| B.3 Centrifuge Arm . . . . .                                   | 114 |
| B.4 Process Module Adjustment Block . . . . .                  | 115 |
| B.5 Processing Box . . . . .                                   | 115 |
| B.6 Counterweight . . . . .                                    | 118 |
| C: Diagnostics Module . . . . .                                | 119 |
| C.1 Thermocouples . . . . .                                    | 119 |
| C.2 Photodiodes . . . . .                                      | 119 |
| C.3 CCD Camera . . . . .                                       | 120 |
| C.4 Accelerometers . . . . .                                   | 120 |
| D: Data Acquisition and Storage Module . . . . .               | 121 |
| D.1 Signal Conditioning/Amplification (Circuits) . . . . .     | 124 |
| E: Power Supply Module . . . . .                               | 128 |
| F: System Control Module . . . . .                             | 129 |
| G: Safety Enclosure . . . . .                                  | 130 |

## **A: Centrifuge Motion Package Module**

### **A.1 Flywheel Support Flange**

Manufactured by the UTSI Machine Shop (aluminum)



- Hose Connector:  
Brass, right angle,  $\frac{1}{8}"$  male to  $\frac{1}{4}"$  hose  
(Swagelok model # B-400-2-2)

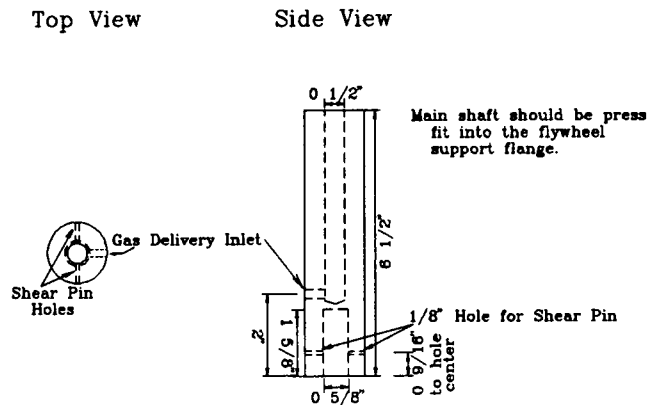
### **A.2 Rotation Motor**

Dayton  $\frac{3}{4}$  HP permanent magnet T.E.F.C. motor with N.E.M.A. 56C mounting face and electronic speed controller  
(Grainger catalog # 2Z846)

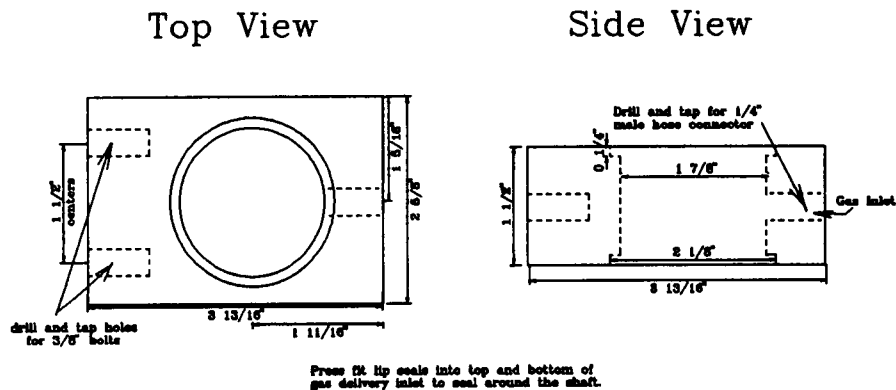
Dayton right angle c-face speed reducer with a 10:1 speed ratio  
(Grainger catalog # 4Z006)

### A.3 Centrifuge Gas Delivery Inlet and Support System

- Shaft:  
Manufactured by the UTSI Machine Shop (steel)



- Shaft Support Bearing:  
Browning prelubricated precision machined ball bearing pillow block (Grainger catalog # 5X695)
- Gas Delivery Inlet:
- Block:  
Manufactured by the UTSI Machine Shop (aluminum)



- Hose Connector:  
Brass, right angle, 1/4" male to 1/4" hose (Swagelok model # B-400-2-4)
- Lip seal (2X):  
Shelbyville Bearing and Supply Co, Inc (catalog # CRS14876)

## A.4 Support Block

Manufactured by the UTSI Machine Shop (3/4" aluminum)

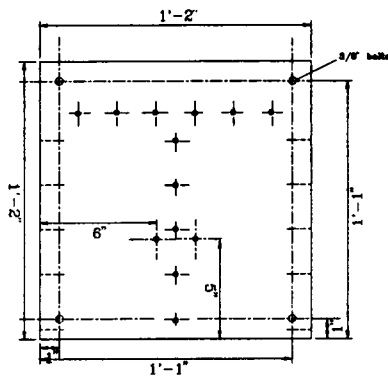
five (5) plates connected with 1/4-20 bolts

overall dimensions: 35cm X 39.4cm X 33.7cm (14" X 15.5" X 13.25")

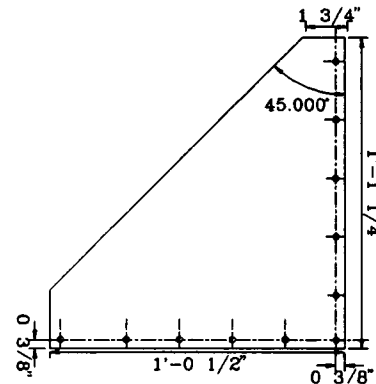
note: Circles indicate clearance holes for 1/4-20 bolts unless otherwise indicated.

Dashed lines indicate holes drilled and tapped for 1/4-20 bolts.

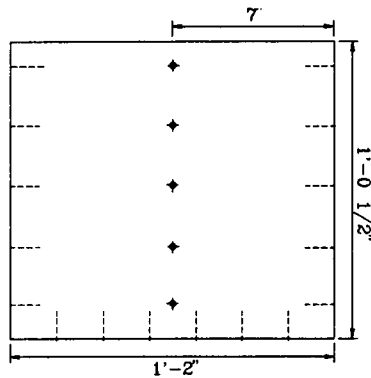
- Base (Bottom) Plate



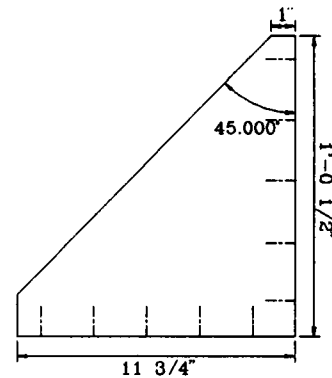
- Side Plate (2X)



- Back Plate



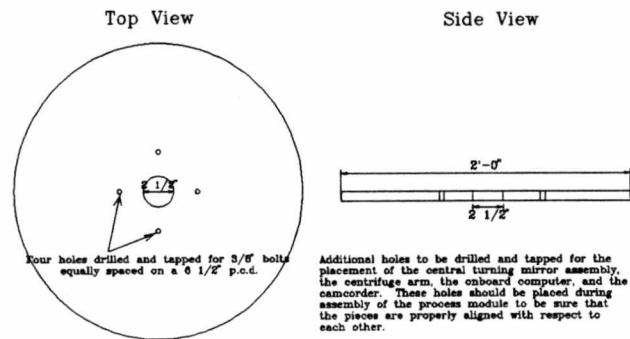
- Center Plate



## B: Process Module

### B.1 Centrifuge Flywheel

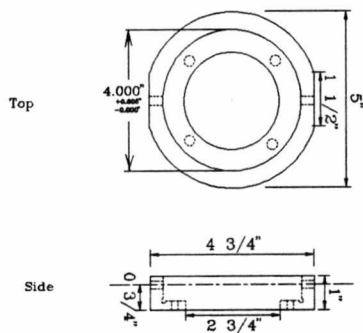
Manufactured by the UTSI Machine Shop (aluminum)



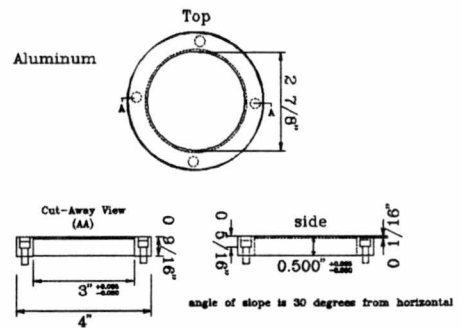
### B.2 Central Turning Mirror Assembly

- Mirror:  
12.7mm (1/2") thick, 76.2mm (3") diameter gold plated copper mirror
- Mirror Holder:  
Manufactured by the UTSI Machine Shop (aluminum)

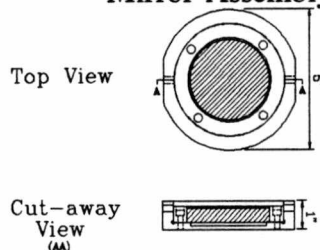
#### Mirror Base



#### Mirror Clamp

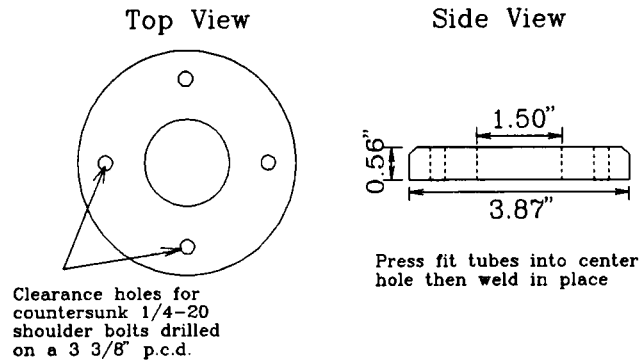


#### Mirror Assembly



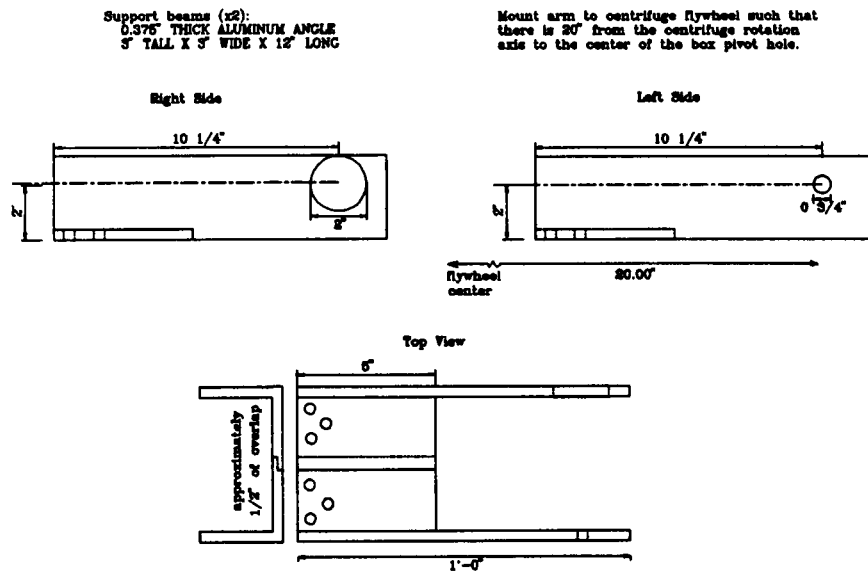
- Mirror Support Rods:

- Pipe  
1.5" diameter steel pipe cut to 15" long
- Pipe Clamps  
(NRC Model 345)
- Base  
Manufactured by UTSI Machine Shop (steel)



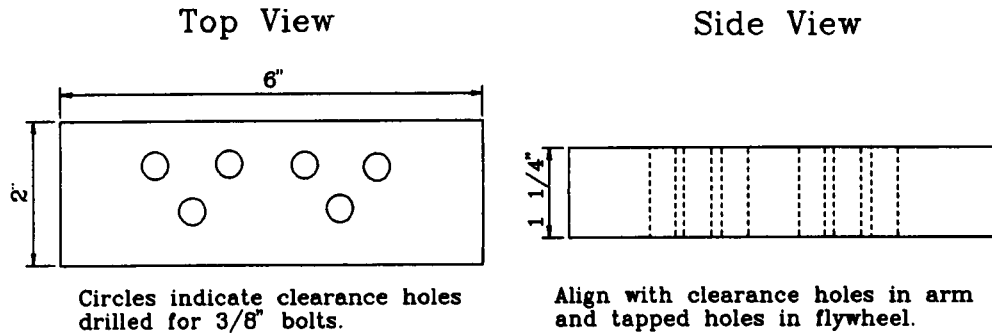
### B.3 Centrifuge Arm

Manufactured by the UTSI Machine Shop (steel)



## B.4 Process Module Adjustment Block

Manufactured by the UTSI Machine Shop (aluminum)

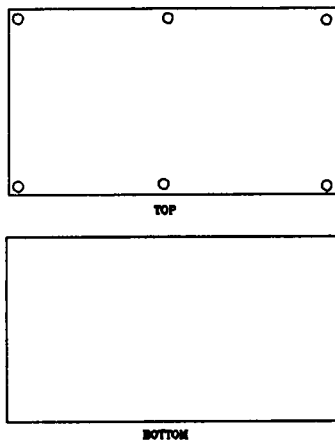


## B.5 Processing Box

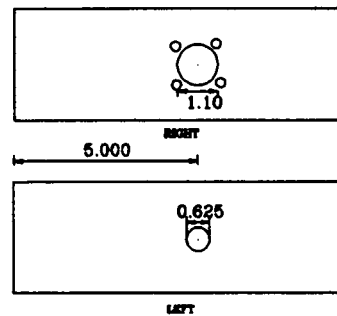
- Box: Manufactured by the UTSI Machine Shop (aluminum)  
overall dimensions: 9" X 5" X 3 3/8" with lid  
five (5) plates welded together plus a lid (held with screws)

Use 3/16" aluminum plate welded together to form bottom and sides.  
Top secured to body with six (6) 4/40 screws.

x2 - 9" X 5" plates (top and bottom panels)

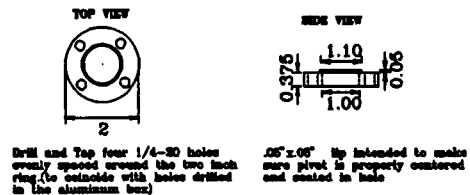
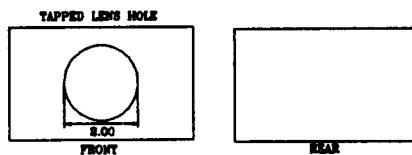


x2 - 8 5/8" X 3" plates (side panels)



### Box Pivot

x2 - 5" X 3" plates (front and rear panels)



2nd pivot is section of 5/8 o.d. pipe.

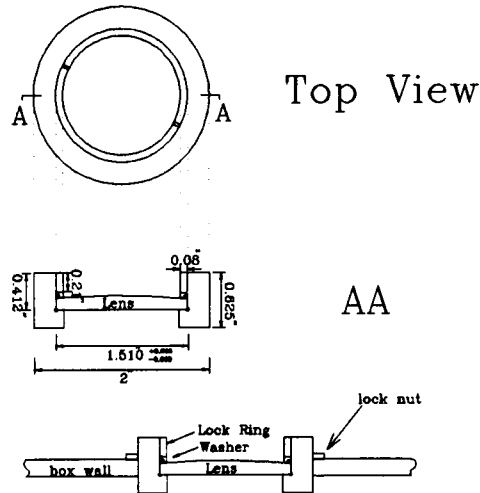
- Beam Focussing Lens:

- Lens:

38mm (1.5") diameter, 127mm (5") focal length ZnSe lens  
(Laser Power Optics catalog # LPZ 1.5-5.0-1331)

- Lens Holder:

Manufactured in the UTSI Machine Shop (aluminum)  
Thread Pitch: 32 threads/inch



LENS HOLDER has a fine-pitched thread on the outside to allow it to be screwed into place in the process box.

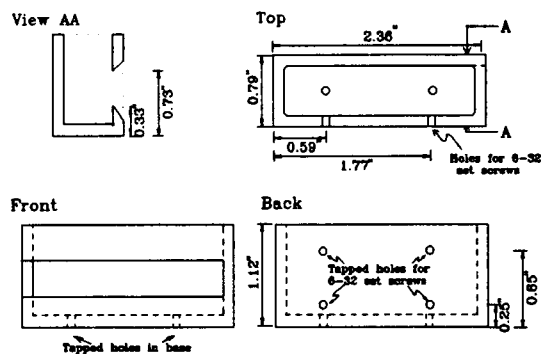
WASHER - 0.08" wide x 0.06" high with a 0.04" wide by 0.03" high notch cut from the outer bottom edge for the indium wire.

- Sample Holder and Positioning System:

- Sample Holder:

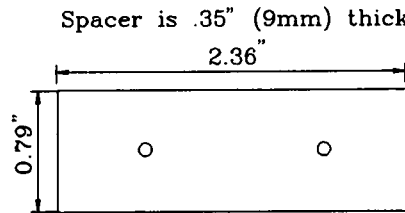
Manufactured in the UTSI Machine Shop (copper)

Sample Holder copper: walls 3mm (0.12") thick  
base 3.4mm (0.134") thick



- Stage:  
Precision miniature stage positioning system  
(Edmund Scientific catalog # A36,347)

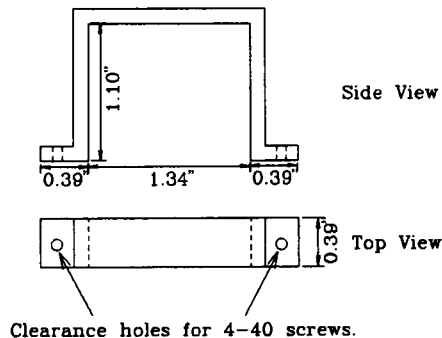
- Spacer:  
Manufactured in the UTSI Machine Shop (aluminum)



Holes align with mounting holes in stage.

- Sample Motor:
- Motor:  
Gearhead motor (one gear removed to increase output rpm)  
(H & R Co. catalog # TM92MTR2023)
- Tie Down Strap:  
Manufactured by the UTSI Machine Shop (aluminum)

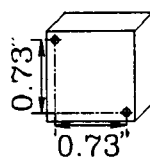
Aluminum strap is 0.12" (3mm) thick.



- Limit Switch Mounting Blocks (for motor control)

#### Mounting Blocks

- (1) 0.9" X 0.9" X 0.5" (high)
- (1) 0.9" X 0.9" X 0.945" (high)

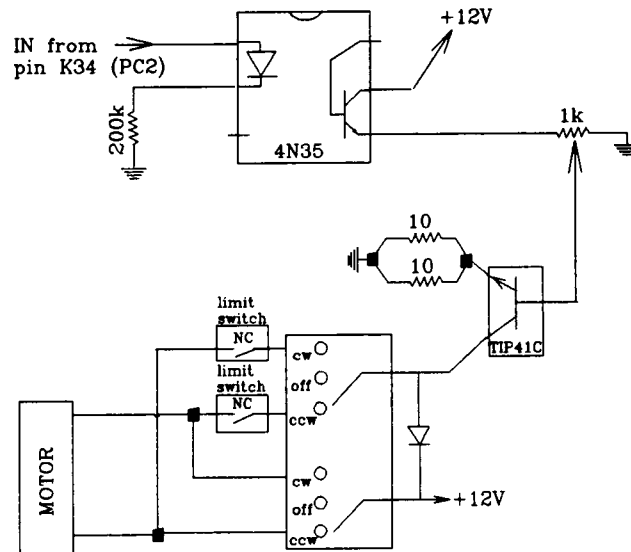


Circles indicate holes drilled and tapped for 4-40 screws. Additional tapped holes are to be added for mounting the limit switches.

- Motor Control Circuit:

- Parts list:

- (1) Opto isolator type 4N35
- (2)  $10\Omega$  power resistors
- (1) double pole, double throw switch
- (2) mini limit switches with rollers
- (1) diode
- (1) TIP41C power transistor
- (1)  $1k\Omega$  Potentiometer



## B.6 Counterweight

Manufactured by the UTSI Machine Shop (aluminum)

(Identical to the processing box, arm, and adjustment block except the front of the box has no lens hole. The front has holes for the fuse holder and power switch instead.)

## C: Diagnostics Module

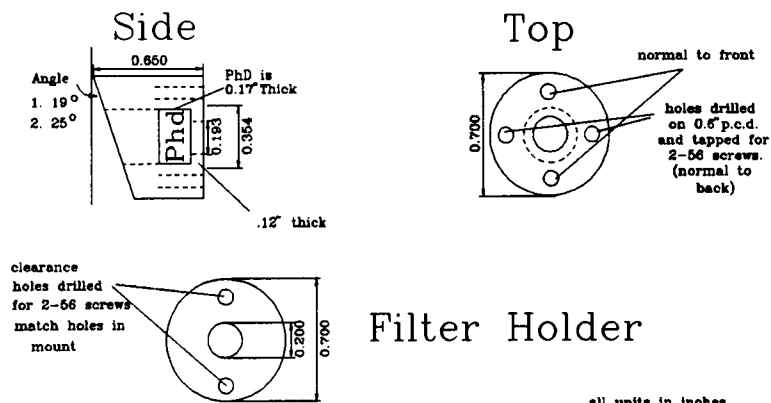
### C.1 Thermocouples

- Thermocouples:  
Type K (Chromel/Alumel), fine wire (0.003" diameter)  
(Omega catalog # CHAL-003)
- Temperature Sensor:  
0° - 100°C Temperature sensor  
(Digi-Key part # LM35DZ)

### C.2 Photodiodes

- Photodiode:  
PIN silicon photocell  
(type S1223)
- Holder:  
Manufactured by the UTSI Machine Shop

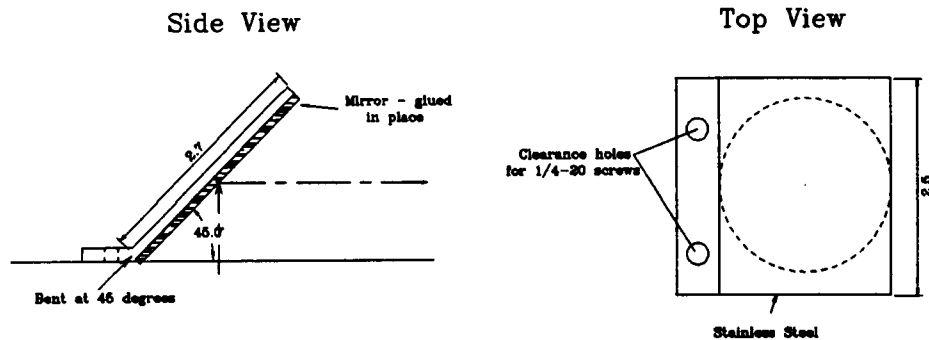
made of non-conductive material (plastic)



### C.3 CCD Camera

8mm, 1.6 lb G.E. 4 lux camcorder with 8X telephoto zoom lens  
(model # CG805)

- Turning Mirror
- Mirror  
WalMart Compact Mirror
- Mount  
Manufactured by the UTSI Machine Shop



### C.4 Accelerometers

- R axis:  
IC Sensors  $\pm 10g_c$  accelerometer  
(model # 3145-010)
- Z, T axes:  
IC Sensors  $\pm 5g_c$  accelerometers  
(model # 3145-005)

## **D: Data Acquisition and Storage Module**

### **Enclosure:**

Halfcube PC/104 Enclosure

(Enclosure Technologies, Inc.)

### **Computer:**

Ampro Coremodule 286

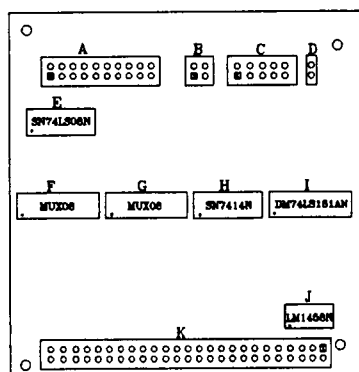
- CM286 Development Kit (16 MHz, 2M RAM)  
(Ampro catalog # CMX-286-K-54)
- Development Chassis  
(Ampro catalog # EXP-CHS-K-03)
- SSD Software  
(Ampro catalog # SWR-SSD-K-51)
- Dallas 32 pin Flash EEPROM 1024k (128k X 8)  
(Newark catalog # DS1245Y100)

### **Data Acquisition Board:**

Real Time Devices DM406

### **Multiplexer Board:**

Board Layout



- A. 20 Pin Header Strip: Analog Data Transmission
- B. 4 Pin Header Strip: Data Acquisition Trigger
- C. 10 Pin Header Strip: Process Box Power
- D. 2 Pin Header Strip: Download Trigger
- E. Quad 2-Input AND Gate (#SN74LS08N): IR Detector Circuit
- F. 8 Channel Multiplexer (#MUX08): Multiplexer Circuit
- G. 8 Channel Multiplexer (#MUX08): Multiplexer Circuit
- H. Hex Inverter (#SN7414N): Multiplexer Circuit
- I. Ripple Counter (#DM74LS161AN): Multiplexer Circuit
- J. Dual Op Amp (#LM1458N): Buffer Circuit
- K. Right Angle 50 Pin Header Strip: I/O Connector to DM406

## Header Strip Pinout:

### - A. 20 Pin (Analog Data Transmission)

- |                         |            |
|-------------------------|------------|
| 1. Photodiode 1 (front) | 2. Ground  |
| 3. Accelerometer 1 (r)  | 4. Ground  |
| 5. Thermocouple 1       | 6. Ground  |
| 7. Accelerometer 2 (z)  | 8. Ground  |
| 9. Thermocouple 2       | 10. Ground |
| 11. Accelerometer 3 (t) | 12. Ground |
| 13. Thermocouple 3      | 14. Ground |
| 15. Temperature Sensor  | 16. Ground |
| 17. Thermocouple 4      | 18. Ground |
| 19. Photodiode 2 (side) | 20. Ground |

### - B. 4 Pin (Data Acquisition Trigger)

- |           |           |
|-----------|-----------|
| 1. +5V    | 2. Signal |
| 3. Ground | 4. NONE   |

### - C. 10 Pin (Process Box Power)

- |                              |                  |
|------------------------------|------------------|
| 1. +12V                      | 2. -12V          |
| 3. Motor Trigger (from K.34) | 4. NONE          |
| 5. Motor +12V                | 6. Motor Ground  |
| 7. Motor +12V                | 8. Motor Ground  |
| 9. Motor +12V                | 10. Motor Ground |

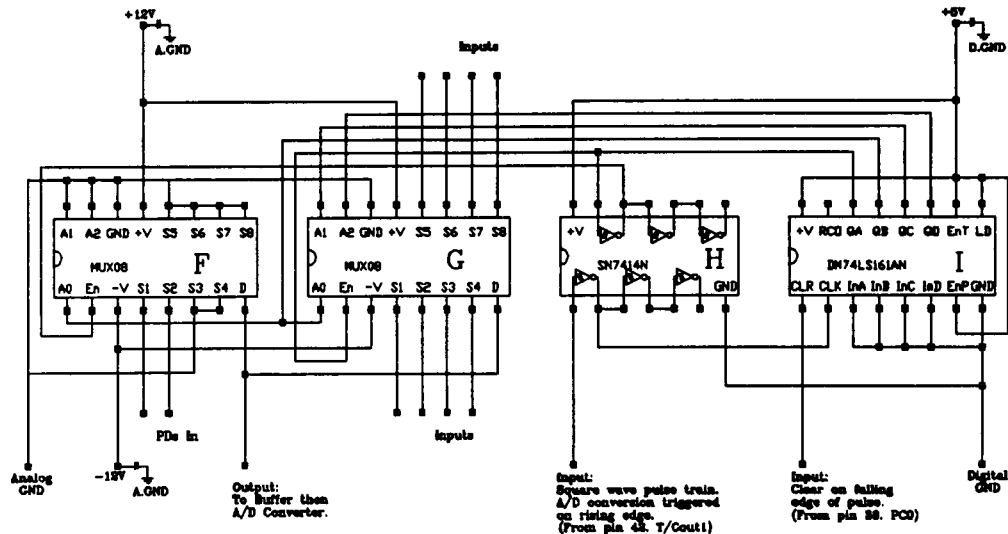
### - D. 2 Pin (Download Trigger)

- |        |           |
|--------|-----------|
| 1. +5V | 2. Ground |
|--------|-----------|

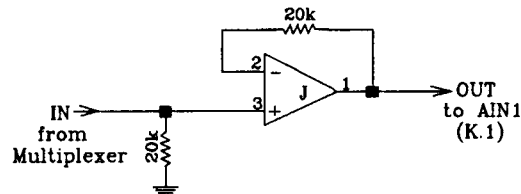
### - K. 50 Pin (I/O Connector to DM406)

- |                                  |                                                    |
|----------------------------------|----------------------------------------------------|
| 1. Analog IN 1 (AIN1)            | 2. AIN9                                            |
| 3. AIN2                          | 4. AIN10                                           |
| 5. AIN3                          | 6. AIN11                                           |
| 7. AIN4                          | 8. AIN12                                           |
| 9. AIN5                          | 10. AIN13                                          |
| 11. AIN6                         | 12. AIN14                                          |
| 13. AIN7                         | 14. AIN15                                          |
| 15. AIN8                         | 16. AIN16                                          |
| 17. Analog Out 1 (AOUT 1)        | 18. Analog Ground                                  |
| 19. AOUT 2                       | 20. Analog Ground                                  |
| 21. Analog Ground                | 22. Analog Ground                                  |
| 23. PA7 (Start Data Acquisition) | 24. PC7                                            |
| 25. PA6 (Start Download)         | 26. PC6                                            |
| 27. PA5                          | 28. PC5                                            |
| 29. PA4                          | 30. PC4                                            |
| 31. PA3                          | 32. PC3                                            |
| 33. PA2                          | 34. PC2 (Turn On Sample Motor)                     |
| 35. PA1                          | 36. PC1 (Set Data Acquisition Trigger - PhD array) |
| 37. PA0                          | 38. PC0 (Reset Counter)                            |
| 39. Trigger In                   | 40. Digital Ground                                 |
| 41. External Gate 1              | 42. T/C Out 1                                      |
| 43. Trigger Out                  | 44. T/C Out 2                                      |
| 45. External Clock               | 46. External Gate 2                                |
| 47. +12V                         | 48. +5V                                            |
| 49. -12V                         | 50. Digital Ground                                 |

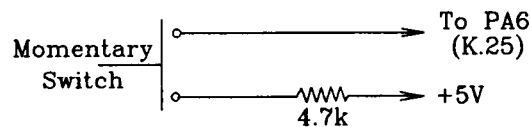
- Multiplexer Circuit:  
Manufactured at UTSI  
Parts list:  
(2) 8 Channel Analog Multiplexer (Part # LF13508)  
(1) Hex Inverter (Part # SN7414N)  
(1) Synchronous 4-bit Binary Counter (Part # DM74LS161AN)  
(4) 16 pin DIP Socket



- Buffer Circuit:  
Manufactured at UTSI  
Parts list:  
(1) Harris Dual Op Amp (# LM1458N)  
(2) 10k $\Omega$  resistors



- Download Trigger:  
Manufactured at UTSI  
Parts list:  
(1) Momentary switch  
(1) 4.7k $\Omega$  resistor

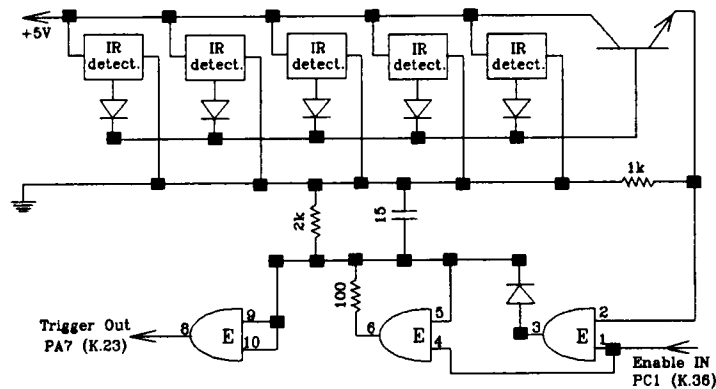


- Data Acquisition Trigger:

Manufactured at UTSI

Parts list:

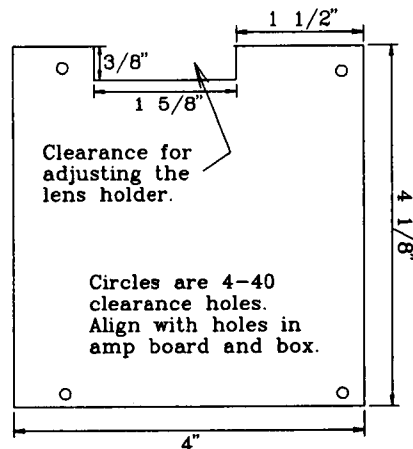
- (1) Quad 2-Input AND Gate (# ECG 7408)
- (5) IR Schmitt Trigger Detectors (# TM910PT1813)
- (5) Signal diodes
- (1)  $2k\Omega$  resistor
- (1)  $1k\Omega$  resistor
- (1)  $100\Omega$  resistor
- (1)  $15\mu F$  capacitor
- (1) Transistor (# NPC 3904)
- (1) Diode



## D.1 Signal Conditioning/Amplification (Circuits)

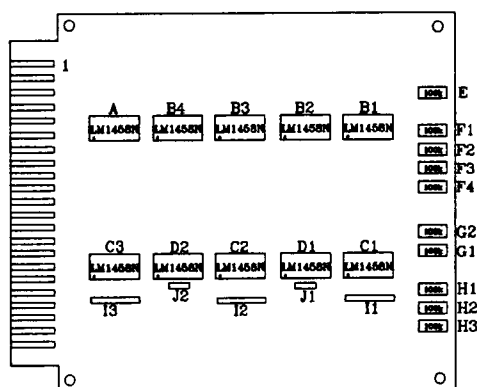
- Shield:

Manufactured at UTSI (brass shim stock)



- Amplification Board:

Board Layout



A. Temperature Sensor Circuit (Dual Op Amp # LM1458N)

B1-B4. Thermocouple Circuits (Dual Op Amp # LM1458N)

C1-C3. Accelerometer Circuits (Dual Op Amp # LM1458N)

D1-D2. Photodiode Circuits (Dual Op Amp # LM1458N)

E. 100k $\Omega$  trimpot for Temperature Sensor Circuit

F1-F4. 100k $\Omega$  trimpots for Thermocouple Circuits

G1-G3. 100k $\Omega$  trimpots for Accelerometer Circuits

H1-H2. 100k $\Omega$  trimpots for Photodiode Circuits

I1-I3. Connectors for Accelerometer Cables

J1-J2. Connectors for Photodiode Cables

Card Edge Pinout

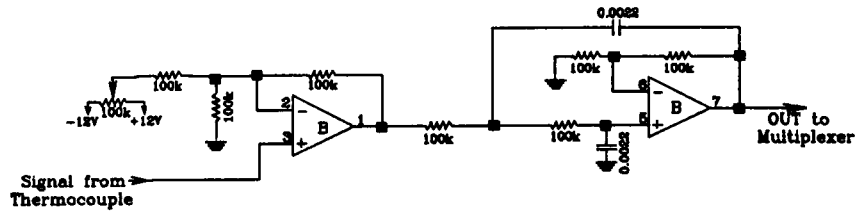
|                     |                             |
|---------------------|-----------------------------|
| 1. UNUSED           | 2. Thermocouple 1           |
| 3. Thermocouple 2   | 4. Thermocouple 3           |
| 5. Thermocouple 4   | 6. Temperature Compensation |
| 7. Ground           | 8. Ground                   |
| 9. Ground           | 10. -12V                    |
| 11. +12V            | 12. Ground                  |
| 13. Ground          | 14. Ground                  |
| 15. Ground          | 16. Photodiode 2            |
| 17. Accelerometer 3 | 18. Accelerometer 2         |
| 19. Accelerometer 1 | 20. Photodiode 1            |
| 21. Ground          | 22. UNUSED                  |

- Thermocouples

Manufactured at UTSI

Parts list:

- (1) Harris Dual Op Amp (# LM1458N)
- (1) 25 turn, 100k $\Omega$  trimpot
- (7) 100k $\Omega$  resistors
- (2) 0.0022 $\mu$ F capacitors

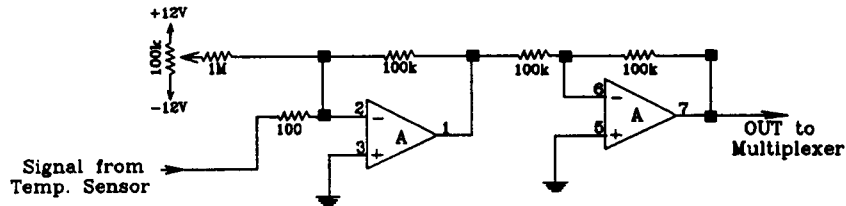


- Temperature Sensor

Manufactured at UTSI

Parts list:

- (1) Harris Dual Op Amp (# LM1458N)
- (1) 25 turn, 100k $\Omega$  trimpot
- (1) 1M $\Omega$  resistor
- (1) 100 $\Omega$  resistor
- (3) 100k $\Omega$  resistors

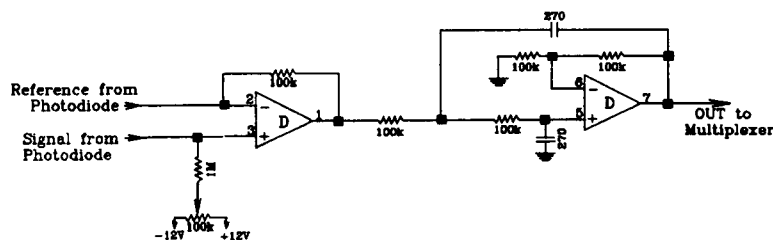


- Photodiodes

Manufactured at UTSI

Parts list:

- (1) Harris Dual Op Amp (# LM1458N)
- (1) 25 turn, 100k $\Omega$  trimpot
- (5) 100k $\Omega$  resistors
- (2) 270pF capacitors

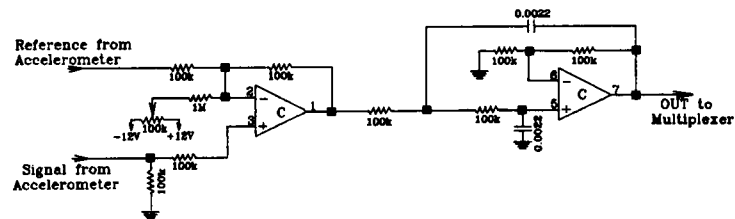


- Accelerometers

Manufactured at UTSI

Parts list:

- (1) Harris Dual Op Amp (# LM1458N)
- (1) 25 turn, 100k $\Omega$  trimpot
- (8) 100k $\Omega$  resistors
- (1) 1M $\Omega$  resistor
- (2) 0.0022 $\mu$ F capacitors



## **E: Power Supply Module**

### **Battery:**

- Dewalt 12V rechargeable battery  
(model # DW9050)
- Dewalt Universal Battery Charger  
(model # DW9104)

### **Voltage Regulators:**

- Computer Power Supply
  - $\pm 12V$ , 10W High Density DC/DC Converter  
(Computer Products model # NFC10-12D12)
  - +5V, 1.5A Voltage Regulator  
(Trendsetter model # 78SR05H)
- Sample Motor Power Supply
  - +12V, 10W High Density DC/DC Converter  
(Computer Products model # NFC10-12S12)

### **Fuse Holder:**

Plastic Fuse Holder, holds one fuse: 20A 250V max

### **Fuse:**

2A 250V (BUSSAGC2)

### **Switch:**

Double Pole, double throw switch

## **F: System Control Module**

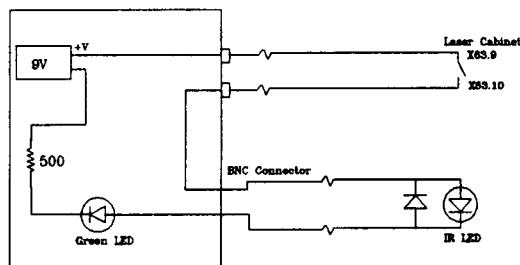
### **Laser/CNC Controller:**

#### **- IR Trigger:**

**Manufactured at UTSI**

#### **Parts list:**

- (1) Two line cable with a set of banana plugs on each end
- (1) Coaxial cable with a BNC connector on one end
- (1) 9V battery
- (1) Green LED
- (1) IR LED (Part # ????)
- (1) Diode
- (1) BNC jacks
- (2) Jacks for banana plugs



**Motor Speed Controller: see motor manual**

### **Gas Flow Regulator:**

**Cole Parmer (1 unit = 3.75 l/min)**

### **Ampro Motherboard:**

- Signal from the IR Detector: see Module 4 (Multiplexer Board)
- Signal from the Download Trigger: see Module 4 (Multiplexer Board)

**Camcorder Switches: see camcorder manual**

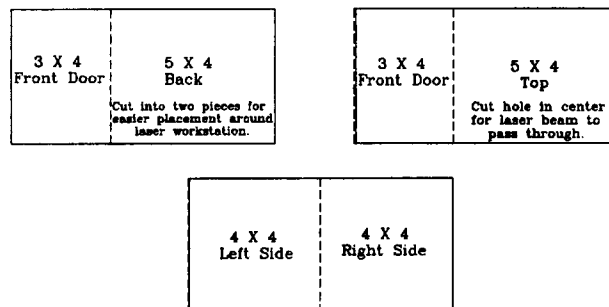
## G: Safety Enclosure

### Safety Enclosure:

Manufactured at UTSI

overall dimensions: 51 1/2" deep, 60 3/8" wide, 73" tall  
(bolted together with 1/4-28 bolts and nuts)

- frame: slotted angle (ez angle)
  - (4) legs: wide 1 5/8" X 2 3/8" X 70.5" long
  - (4) side rails: narrow 1 1/2" X 1 1/2" X 51" long
  - (4) side rails: narrow 1 1/2" X 1 1/2" X 60" long
- windows/door:
  - (3) 8' X 4' sheets of lexan



Drill and tap 1/4-28 holes in lexan for bolts.  
Align holes with pre-existing holes in frame.

- door track:
  - 5' long rails for 1/4" thick display case windows
- handle:
  - 4 5/16" Wire Pull Handle  
(Lowe's part # BPW355-18)
- wheels:
  - (4) Lowe's 2" Swivel Caster with Brake

## VITA

Cindy Renay (Rowland) McKowen was born in Norwalk, California on September 7, 1969. She attended high school at Georgetown High School in Georgetown, Texas where she graduated as Salutatorian in May 1987.

Her undergraduate studies were carried out at Angelo State University in San Angelo, Texas which she entered in August 1987. She financed her education with a Carr Academic Scholarship and received a Bachelor of Science degree in Physics in May 1991.

She entered the University of Tennessee Space Institute (UTSI) in Tullahoma, Tennessee in August 1991 with a Graduate Research Assistantship at the Center for Laser Applications (CLA). In May 1992, she married Bruce Wayne McKowen. She received a Master of Science degree in Engineering Science and Mechanics in December 1993.