Closed Loop Control of Forming Stability During Aluminum Stamping

- by -

Robert S. Bakkestuen

B. S., Mechanical Engineering
University of Wisconsin - Platteville
(1992)

Submitted to the
Department of Mechanical Engineering
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 1994

©1994 Massachusetts Institute of Technology
All rights reserved

Signature of Author

Department of Mechanical Engineering

Certified by

David E. Hardt, Professor of Mechanical Engineering
Thesis Advisor

Accepted by

Ain A. Sonin, Department of Mechanical Engineering
Chairman, Departmental Committee on Graduate Students

ARCHIVES
MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

AUG 01 1994
Page 1
LIBRARIES
This page intentionally left blank.
Closed Loop Control of Forming Stability
During Aluminum Stamping

- by -
Robert S. Bakkestuen

Submitted to the Department of Mechanical Engineering
June 1994 in partial fulfillment of the requirements for the
Degree of Master of Science in Mechanical Engineering.

Abstract

Closed-loop control of sheet metal forming has been an ongoing area of research in the Laboratory for Manufacturing and Productivity here at MIT the mid 1980's. This research and has been documented in one master's thesis, one doctoral thesis, one engineer's thesis and several publications. My work builds upon this earlier work and addresses some of the fine details that were excluded from the initial experiments.

This thesis begins with improvement of the existing experimental apparatus. In the earlier work, the press dynamics were ignored because the experiments were conducted under quasi-static conditions (i.e. very low punch velocities) which allowed the researchers to demonstrate the viability of the concept of closed-loop stability control without risking a large amount of time or money. After the success of this early research, we now wanted to more closely simulate a production stamping environment (i.e. high punch velocities). In order to conduct these forming experiments, the press had to be modernized because it was not capable of high punch velocities. This consisted of updating the computer data acquisition and control system and connecting the press to a high capacity hydraulic pump. After completing these improvements, thorough testing was conducted to characterize the press dynamics. This thesis documents these improvements and present the results of the dynamic testing.

This thesis then moves into forming trials. The closed-loop control system is demonstrated in the forming of square aluminum pans with equal corner radii. Non-axisymmetric parts present new challenges when forming aluminum versus steel because of aluminum's anisotropic behavior. Improved tracking performance of the closed loop controller was observed in these trials over that obtained in the prior research.

Thesis Advisor: David E. Hardt
Professor of Mechanical Engineering
This page intentionally left blank.
Acknowledgments

I would like to acknowledge the financial support of the Aluminum Company of America (Alcoa) which made this work possible. In particular, I would like to thank Dr. James Story of Alcoa for his advise and for his assistance in obtaining material for conducting forming experiments.

I would also like to thank Prof. David E. Hardt for his guidance and encouragement during the course of this research. Additionally, I would like to thank all of the other students and staff at MIT who helped me complete this work.

Lastly, I would like to thank my family for their support and encouragement over the years. Without them, I would not be in the position that I am today.
# Table of Contents

**Chapter 1: Background** ............................................................................. 9  
  - Introduction ......................................................................................... 9  
  - Terminology ....................................................................................... 11  
  - Conical Cup Forming ......................................................................... 12  
  - What is Closed Loop Control of Sheet Metal Forming? ..................... 16  
  - Results of Prior Research .................................................................. 20

**Chapter 2: Improvements to Experimental Apparatus** .......................... 21  
  - Introduction ....................................................................................... 21  
  - Press Hardware ................................................................................ 22  
  - Computer Hardware ......................................................................... 25  
  - Software Development ..................................................................... 26  
  - Calibration ....................................................................................... 28  
  - Summary .......................................................................................... 33

**Chapter 3: Press Dynamics** ................................................................. 35  
  - Introduction ....................................................................................... 35  
  - Punch Dynamics ............................................................................... 35  
  - Binder Dynamics ............................................................................. 42  
  - Summary .......................................................................................... 48

**Chapter 4: Square Pan Experiments** .................................................. 49  
  - Introduction ....................................................................................... 49  
  - Constant Binder Force Experiments .................................................. 50  
  - Closed Loop Control Experiments ..................................................... 55  
  - Summary .......................................................................................... 60

**Chapter 5: Conclusions and Future Research** ...................................... 61

**Appendix A: Effects of Binder Force on Conical Cup Shape Profile** ....... 63  
  - Introduction ....................................................................................... 63  
  - Experiments ..................................................................................... 63  
  - Discussion of Results ...................................................................... 65

**Appendix B: Material Tensile Test Data** ............................................. 67

**Appendix C: Controller Programs** ....................................................... 69  
  - FB_CONST.C .................................................................................. 70  
  - CLPF.C ............................................................................................ 86

**Bibliography** .......................................................................................... 103
Chapter 1: Background

Introduction

When I was a child, I remember the television repair technician coming to our house on a frequent basis to make adjustments and to replace vacuum tubes. This level of reliability was considered acceptable at the time because it was true of almost all televisions of the day. Today, I get upset if my television ever needs repair. Clearly, my definition of what standards of performance that a television had to meet evolved since my childhood. This evolution was driven by the changing technology that was available to make televisions more reliable. Once one manufacturer had improved the reliability of their television sets, other manufactures had no choice but to do the same because consumers would no longer tolerate low reliability when there was an alternative.

This sort of evolution in the definition of what standards a quality product has to meet has occurred in all industries and product lines. The evolution was driven by manufacturers competing with each other for market share. In the past, great leaps were made in improving the quality of products by inventing new technologies that were more reliable. However more recently, increases in quality have come from improvements made to the processes be which products are made. By reducing the variability in manufacturing operations, quality is improved because a more consistent product is produced. It is this improvement in manufacturing that this thesis will address.

There are many ways to improve the various manufacturing processes. This thesis deals with the application of real-time, closed loop process control techniques to sheet metal forming. Sheet metal forming is a vital process in manufacturing throughout the world. Many industrial and consumer products contain three dimensional sheet metal parts. One
of the largest users of sheet metal stamping is the automotive industry. The quantity of sheet metal used in this industry alone justifies the need for improvements in the manufacture of sheet metal components.

This thesis concerns itself with a particular type of sheet metal forming that uses matched dies to impart a three-dimensional shape to a flat piece of sheet metal stock. See Figure 1.1. In this type of sheet metal stamping, the sheet is clamped at the edges to control material flow during forming in order to prevent the sheet from buckling. During the forming cycle, material from the edges flows into the die as the part is formed. Several methods exist for controlling material flow: draw beads, stepped dies, frictional binders and combinations of these. This thesis only considers frictional binding and is further restricted to investigation of the failure modes that occur in the unsupported material prior to full die closure.

![Figure 1.1 Schematic of typical matched die sheet metal forming.](image-url)
Terminology

Before proceeding with a discussion of the application of closed loop process control to sheet metal forming, it is useful to define the terminology that will be used throughout this thesis. Written definitions of the terminology are presented below and are followed by pictorial descriptions that clarify these definitions in Figures 1.2 and 1.3.

Blank - Flat piece of sheet metal that has been cut to a desired size and shape prior to stamping.

Die - Female half of a matched die set.

Punch - Male half of a matched die set.

Binder (Blank Holder) - Device that clamps the edge of the blank during forming to control the amount of draw-in.

Flange - The annular region at the outer edge of the blank which is clamped between the binder and the die.

Free Section - The part of the sheet metal which is not in contact with the punch or the die during forming.

Punch Force (F_p) - The force being exerted by the punch on the blank.

Binder Force (F_b) - The force being exerted by the binder on the flange.

Tangential Force (F_t) - The reaction force in the blank in a direction tangent to the sheet which opposes the punch force.

Restraining Force (F_r) - The frictional restraining force on the flange due to the binder force.

Draw-In (X_b) - The amount that the edge of the blank has been pulled into the free section.

Punch Displacement (Y_p) - Movement of the punch as measured from the point when initial contact with the blank has been made.

Binder Displacement (Y_b) - Movement of the binder as measured from the initial, closed position at the start of forming.
Conical Cup Forming

In investigating the behavior of the free section during matched die stamping, it is common to form conical cups without matched dies. This provides the same unsupported or free section as is found in matched die forming prior to full die closure and also allows
the forming cycle to continue until part failure since the punch will never bottom out in the die. See Figure 1.4. Other variations of this tooling that are non-axisymmetric, such as square pans, can also be used.

![Diagram of tooling used in conical cup experiments](image)

**Section A-A**

101.6 mm

63.5 mm

**Note:** All Fillets 6.35 mm.

Figure 1.4 Tooling used in all conical cup experiments discussed in this thesis.
In forming these parts failure occurs in two ways. The material may tear near the punch nose because of high tensile stresses caused by stretching of the material during forming, or the material may buckle near the die because of high compressive hoop stresses that develop as material is drawn into smaller radius sections. See Figure 1.5. Buckling often initiates in the binder area but the part is not considered to have failed until the buckles have propagated into the free section since it is assumed that the flange will be trimmed from the part after forming.

Figure 1.5 Photographs of a torn and a buckled conical cup.
In traditional forming practice the binder force is held constant throughout the forming cycle. Constant binder force (CBF) forming of conical cups results in a characteristic failure height diagram like the one shown in Figure 1.6. Note that there is a sharply defined 'optimal' constant binder force (CBF₀) which results in the maximum or 'optimal' part height (H₀) before failure. At CBF's lower than the CBF₀, the parts fail by buckling because of compressive hoop stresses which occur in circular sections as the material is drawn into smaller and smaller diameter circles. At CBF's higher than the CBF₀, the parts fail by tearing because of high tangential stresses which occur because the material is being tightly held by the binder and not being allowed to flow into the die area.

![Failure Height Diagram](image)

**Figure 1.6** Typical failure height diagram for formed sheet metal parts. (Fenn 1989)

The failure height curve shown above would be defined for a particular set of forming conditions. Any changes in these forming conditions would cause the curve to shift. For example if the lubrication changed such that the coefficient of friction between the binder and the sheet metal was higher, the curve would shift to the left because an equivalent restraining force would be provided at a lower binder force. Conversely if the coefficient of friction was lower, the curve would shift to the right. Because the CBF₀ is sharply defined, small shifts in the forming height curve would result in large changes in
failure height if you were operating at the 'optimal' point. Since variations occur in all manufacturing operations, sheet metal stampings are usually only formed to a height that is much lower than $H_o$.

**What is Closed Loop Control of Sheet Metal Forming?**

*Webster's Ninth New Collegiate Dictionary* defines closed loop as "an automatic control system for an operation or process in which feedback in a closed path or group of paths is used to maintain output at a desired level." (Woolf 1988) Thus in a closed loop control system, some output variable or variables of operation or process are fed back to the controller for comparison to their desired value. This allows the controller to adjust the inputs to the operation or process in order to maintain the desired output of the operation or process.

In applying the closed loop technique to sheet metal forming, some globally measurable variable of the forming process is fed back to a controller which adjusts the binder force to maintain the desired value of the feedback variable. In this way, we attempt to keep the forming process operating at the 'optimal' point. To accomplish this, forming experiments are conducted under controller conditions at the 'optimal' point and the global process variable trajectories are recorded. A closed loop controller is then used to force the process to follow these trajectories during subsequent experiments when the forming conditions are varied. See Figure 1.7.

![Figure 1.7 Example block diagram of a closed loop process controller.](image-url)
When this type of closed loop process control algorithm is used, the binder force starts at some initial value that can be arbitrarily chosen before the forming cycle begins. As the forming cycle progresses, the global process variable that the controller is trying to track (tangential force in the example shown) will not match the input trajectory if the forming process is occurring at any conditions other than the 'optimal' conditions. For example, the tangential force will be lower than the target if the binder force is set too low, and the controller will increase the binder force to make the tangential force match the target. The reverse is also true. In this way, the controller forces the tangential force to match the desired trajectory throughout the forming cycle. Similarly, this control scheme can be used to track an 'optimal' draw-in trajectory as well.

The result of using the closed loop trajectory tracking controller in forming conical cups and square pans is that the failure height is made insensitive to the initial binder force. Figures 1.8 - 1.10 show the failure height plots obtained by Fenn (1989) and Jalh (1993) in their research at MIT using tangential force feedback. In each figure, the failure height curves for constant binder force forming and for closed loop tangential force forming are shown for comparison. Figure 1.11 shows the action of the CLTF controller in adjusting the binder force during the forming cycle. Notice that the binder force histories converge to the 'optimal' CBF value.
Figure 1.8 Failure height diagram for AKDQ steel conical cups showing both CBF and CLTF forming results (Fenn 1989).

Figure 1.9 Failure height diagram for AKDQ steel square pans with unequal corner radii showing both CBF and CLTF forming results (Fenn 1989).
Figure 1.10 Failure height diagram for 2008-T4 aluminum conical cups showing both CBF and CLTF forming results (Jalkh et al. 1993).

Figure 1.11 Typical binder force histories during forming of AL 2008-T4 conical cups when using CLTF controller (Jalkh et al. 1993).
In addition to investigating the influence of the initial binder force on the failure height, Fenn (1989) also investigated other disturbances. Process disturbances were introduced to the forming system in the form of blank diameter, blank thickness and lubrication condition. It was discovered that the 'optimal' tangential force trajectory was independent of blank diameter and lubrication condition. However, the CBF at which the 'optimal' forming conditions occurred varied over a wide range depending upon the particular blank diameter and lubricant used. This means that the CLTF controller does not need any prior knowledge of what blank diameter or lubricant is being used during forming. The CLTF controller will automatically compensate for any variations in these two parameters by adjusting the binder force accordingly.

The controller cannot compensate for variations in blank thickness without prior knowledge of the variation. The input tangential trajectory can be scaled to account for thickness variations if the thickness of the blank is measured prior to forming. A detailed description of how this scaling can be done is presented by Fenn and Hardt (1993). Thus with a feed forward thickness measuring devise, the CLTF controller can adjust itself to accommodate variations in blank thickness.

**Results of Prior Research**

The research conducted has shown that using a closed loop, trajectory tracking control algorithm can greatly reduce the variability in the failure height of conical cups and square pans. However, additional work must be conducted to further understand the forming process physics and the controller dynamics before this technique can be applied in an industrial setting. Also, further research needs to be conducted to identify what trajectory to follow when forming more complex parts.
Chapter 2: Improvements to Experimental Apparatus

Introduction

Although promising, the early results of the closed loop binder force control research are at this point only a laboratory curiosity. All of the experimental work that has been done in this area has been conducted under quasi-static conditions. This was necessary to promote understanding of the forming process and to investigate the feasibility of applying real time closed loop process control techniques to sheet metal forming. In industrial practice however, sheet metal forming is not a static process. Typical cycle times for sheet metal stamping presses are less than five seconds. Thus the sheet metal forming process is obviously not static.

One of the initial goals of this research was to conduct closed loop binder force control experiments under high speed conditions to demonstrate the viability of previously developed control strategies under dynamic forming conditions. The goal was a cycle time of a few seconds for a typical cup height of 30 mm. Thus, the press would have to be capable of sustained punch velocities on the order of 10 mm/s.

This chapter presents the modifications made to the press to allow high speed operation and other various improvements that have been made to the experimental apparatus. These other improvements were aimed at making the press easier to use and at improving the output data generated by the press software during forming experiments.
**Press Hardware**

To improve the dynamic operating range of the press some sort of hardware improvement was required because the displacement response of the punch displacement servo was limited by the flow rates obtainable in the system. It was not difficult to identify the hydraulic pump as the weak link in the hydraulic loop. The hydraulic circuits in the experimental apparatus are very simple. There are two double acting hydraulic cylinders, each connected to a spool type hydraulic servo valve which intern share a common pressure supply and return line to a single hydraulic pump and reservoir. See Figure 2.1.

![Diagram of press hydraulic circuits](image)

**Figure 2.1** Schematic of press hydraulic circuits.

Under no load conditions, the flow rate through the system is determined by one of two things: 1) the maximum flow rate capacity of the pump or 2) the maximum output pressure of the pump. It is obvious how the maximum flow rate capacity of the pump puts an upper bound on the flow rate in the system at any point in time. The maximum output pressure of the pump limits the flow rate through the system in a less obvious manner. The pressure drop through the system due to fictional and other losses varies directly with the flow rate. For example, the pressure drop through the servo valves can be modeled by the equation:
\[ \Delta P_v = \frac{Q_v^2}{K \cdot i} \]

Where:
\( \Delta P_v \) = valve pressure drop (Moog Type) \hspace{1cm} (2.1)
\( Q_v \) = valve flow rate
\( K \) = servo valve sizing factor
\( i \) = input current

Thus, the flow rate in the system will be determined by the lesser of the following: 1) the maximum flow rate capacity of the supply or 2) the flow rate at which the pressure losses in the system equal the maximum output pressure of the supply. This will hold true of any fluid flow system.

In this particular system, it was readily apparent that the flow rate capacity of the pump was the limiting factor. The punch displacement servo valve (Moog® model 30 12. 3000 I 1000 4CA R BUN) is rated at 12 cubic inches per second (cis) no load flow and 6.9 cis at 1000 psi valve pressure drop. (Moog Type...) Where as, the flow rate capacity of the hydraulic pump (Planet Electro-Hydraulic Exciter, Mark II) is only 1 gpm (3.9 cis) at 3000 psi. (Lee 1986) In addition, the actual flow rates observed in the system were much lower than the rated output of the hydraulic pump. The target punch velocity of 10 mm/s translates to a flow rate of 4.9 cis using the flowing equation:

\[ Q_p = V_p \cdot A_c \]

Where:
\( Q = \) flow rate to the punch cylinder \hspace{1cm} (2.2)
\( V_p = \) punch velocity
\( A_c = \) area of the punch cylinder

Since the flow rate capacity of the pump is lower than the flow rate required to achieve the desired punch velocity, the pump had to be upgraded to a larger capacity pump.

In the lab there already existed large capacity hydraulic pump connected to a different press. This pump (Parker model number PAVC65) has a maximum flow rate of 30 gpm (116 cis) at 3000 psi. (Ousterhout 1991) This far exceeds the flow rate
requirements of the active binder force forming press. Since both forming presses are only used for research purposes and the forming experiments only require that the presses be utilized a small fraction of time, it was decided that this single large pump could be shared by the two presses. Therefore, the feasibility of using this pump to supply the active binder force forming press was investigated. Two issues had to be addressed.

First, a method of connecting both presses to the same pump had to be identified. The connections have to allow operation of either press individually while keeping the hydraulic systems separate. It was also required that it be easy to switch operation back and forth between the two presses. All of these requirements were met by the installation of a two position hydraulic selector valve. In one valve position, the pumps supply and return lines would be connected to one of the presses, and in the other position, the lines would be connected to the other press. This is show schematically in Figure 2.2.

![Schematic diagram of hydraulic selector valve.](image)

Second, the suitability of using the large pump with the active binder force forming press had to be assessed. The maximum flow rates that are anticipated in this press are on the order of 5 cfs which is more than twenty times less than the output of the larger pump. Since this pump is a positive displacement pump, excess capacity would result in the pump continually circulating fluid through the pressure relief valve into the reservoir causing
excessive heat buildup. However, this pump was a swash plate type piston pump with an actuator that could vary the angle of the swash plate and thus the flow rate output of the pump. Therefore, the output of the pump could be set to match the needs of either system.

Now that the pump had been changed, the limiting factor in the punch velocities obtainable became the pressure drop through the system with the servo valves being the largest component of this pressure drop. However, I had achieved our goal of being able of forming cups with a cycle time on the order of a few seconds. The maximum punch velocity of the system was now 18.3 mm/s under no load conditions. (The punch dynamics will be discussed in more detail in the next chapter.)

**Computer Hardware**

Now that the press was capable of short cycle times, attention was turned to the data acquisition and control systems. Since the cycle time required to a complete forming experiment was now only a few seconds and since it is desirable to record as many data points as possible, the goal for the data acquisition system was a sample time (time between data points) of approximately 10 milliseconds. In addition to acquiring data, the controller has to calculate new binder force set points to send to the binder servo loop between data points, thus putting computational demands on the controller’s microprocessor.

In the previous system, the limiting component was the computer. The data acquisition and control system was based on a personal computer with a 8086 series microprocessor. The analog input and output functions were handled by a Data Translation DT2811 board that plugged into the ISA bus of the host computer. This board is capable of throughputs of 15 kHz.[Data Translation 1990] All of the other components in the system were analog, and thus have no effect on the sample time. (However, these analog components do effect the dynamic behavior of the press.) Therefore, it was easy to conclude that the problems lie in the computer.
Having identified the computer as the obstacle to being able to acquire data and execute the control algorithm fast enough, the computer hardware was updated. Other options, such as programmable logic controllers or analog implementation, were not considered as suitable replacements for this system because this press must remain flexible to conduct research into various control strategies and because the experience base in the lab was all related to personal computer based controllers. All of the personal computers with the ISA bus architecture are based on the series of x86 microprocessors, and with the high level of competition in the computer industry, prices had been driven so low that it did not make sense to buy anything but the fastest class of personal computer on the market at the time of purchase. This decision was further reinforced by the need for the computer to perform other research related computing tasks in addition to operating the active binder force forming press. Therefore, a computer was purchased with a 486DX2 microprocessor operating at a clock frequency of 66 MHz.

This change of computer hardware required the software to be updated as well. The old software used subroutines that accessed a clock on the expanded memory board of the old computer to control the timing of the program execution. These subroutines would not function in the new computer because this clock was not present. In addition, these subroutines had been written in assembly language, and thus they would be very difficult to adapt to the new hardware. Also, the main block of the controller program add been written using an old version of the BASIC programming language. Some of the commands and syntax used in this program were not compatible with the BASIC compiler that had been supplied with the new computer.

Software Development

After studying the existing source code, it was decided that the most efficient course of action would be to scrape the old source code and to rewrite the entire program. Rewriting the entire program allowed complete freedom to develop a better program. The
old program was dated and was not designed to take advantage of the power of today's higher speed computers. It was also not user friendly and contained very little documentation.

The C programming language (a high level, structured language) was chosen for the controller software. The C language allows the program code to be modularized by subdividing the program tasks into separate functions. These functions can then each be written in their own block of code. These individual program blocks or modules are called upon by the main body of the program as needed. This makes the program both easier to follow by people who are trying to understand its logic and easier for the programmer to write.

A modular program also allows separation of the different modes of operation of the press into different programs. The basic functions of the press do not change from one type of experiment to the next (i.e., constant binder force experiments versus variable binder force experiments). Thus, several generic blocks of code were developed that were used to write a variety of programs to handle the different types of experiments that would need to be conducted. Usually, only one block of code, the controller portion, had to be changed from one program to the next. Since each different experiment had its own program, the performance of the software did not have to be compromised to allow single program to handle all of the experimental possibilities. (See Appendix C for a printout of the source code for the controller programs.)

The program was made as user friendly as possible. This requirement was two fold. First, the program had to give enough prompts to the user so that the press could be operated easily by anyone simply by following the on screen prompts. Second, the program had to contain enough internal documentation so that it could be easily understood and modified by anyone in the future who carries on this research. To achieve the first goal, a wealth of print statements were included in the program that output instructions to the user in plain English. These statements provide step by step instructions that do not
assume any prior knowledge of the program. To achieve the second goal, comment lines were added to the source code that explain what each section of commands in doing. Also, the logic of the program was kept as simple as possible. In addition, user defined constants were used throughout the program to allow changes to be made to the program's parameters simply by editing a definition statement in the header of the source code and recompiling the program. The person editing the program does not have to have any prior knowledge of where or how the parameter is used by the program.

One additional improvement in the software which does not affect the operation of the press is the quality of the output generated. As the experiment runs, a real time graph is displayed on the computer screen which can be printed at the end of the experiment. In the new program, the graphics are displayed in high resolution, and all of the experimental parameters are displayed in plain English. See Figure 2.3. Also, the new program puts this plain English record of the parameters at the top of the data file. Whereas the old program again used the numeric codes. This required anyone trying to find a particular set of data to understand the code system. A procedure that can become very difficult since the codes may change over time.

Calibration

Good experimental procedure requires that equipment be calibrated regularly. All electric equipment will experience some drift in its zero voltage and also in its gain. Other measurement systems are susceptible to their own loses of accuracy. In the case of the active binder force forming press, this natural drift could have been compounded by the change of computer hardware.

The DT2811 board was calibrated first since this board affects all of the measurements. Two features of the board had to be calibrated: the analog output channels and the analog input channels. To facilitate calibration, a program had been supplied with the board by the manufacturer -LPCALIB.EXE- that guided the user through the calibration
Figure 2.3  Sample of the graphical output generated by the new controller software.
process (Data Translation 1987). First the analog output channels were calibrated using a
digital multimeter as a reference. Then the analog outputs were used as reference voltage to
calibrate the analog input channels. All calibrations were accomplished by adjusting
various potentiometers on the DT2811 board as instructed by the program.

**Displacement Transducers**

After calibration of the data acquisition board, the displacement transducers were
calibrated. All of the displacement transducers used in the press were DC input, DC output
linear differential displacement transformers (DCDT's). These transducers consist of a
oscillator circuit to convert the DC input into AC, a transformer with a movable core which
is the connected to the moving system and a rectifier circuit to convert the AC output of the
transformer into a DC signal which is the output voltage. These transducers are
constructed such that the output voltage varies linearly with the displacement of the core.

Since all of the displacements taken in the system are relative to a starting position
and since a starting voltage is measured before each experiment, the only quantity that
needed to be determined for each transducer was the scale factor (slope of the voltage
versus displacement curve). To compensate for any errors in the DT2811 calibration and
for any differences between the different analog input channels, the scale factors for each
displacement measurement were determined at a system level. That is the transducers were
connected to their respective input channel and displaced by known amounts while the
digital output of the DT2811 were observed. Known displacements of the DCDT cores
were accomplished using gage blocks.

Linear behavior was observed for all of the displacement measurement systems.
The scale factors and the R squared values from the linear regression analysis of the
calibration data are shown in Table 2.1. The scale factors are shown in two different units,
digital value/mm and V/mm, for clarity. The first number in units of digital value/mm is the
one used by the press software, and since these units are meaningless outside of the press
software, the second number in V/mm is also reported.
Table 2.1 Scale factors for calibration of the displacement transducers.

<table>
<thead>
<tr>
<th>Scale Factor</th>
<th>R Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Value/mm (V/mm)</td>
<td></td>
</tr>
<tr>
<td>Punch Displacement</td>
<td>1580 (1.929)</td>
</tr>
<tr>
<td>Binder Displacement</td>
<td>22360 (54.59)</td>
</tr>
<tr>
<td>Draw-In</td>
<td>1384 (1.690)</td>
</tr>
</tbody>
</table>

**Force Transducers**

The force measurement devices in the press consisted of strain gage type load cells. These load cells were custom designed and built for this press by Lee. (Lee 1986) The binder force load cell is a hollow cylinder and the punch force load cell is a solid shaft. Both load cells have strain gages mounted at 90 degree increments around their circumference at the midpoint height that measure both the axial compression and the circumferential expansion. These strain gages are connected into a four-arm Wheatstone bridge configuration. The strain gage circuits are connected to high gain instrumentation amplifiers (Analog Devices model 2B31K) specially designed for low level inputs. These amplifiers also contain a regulated excitation voltage circuit to power bridge and bessel filters to condition the output signal. The output from these amplifiers was connected to the DT707 screw terminal.

Accurate calibration of these force measurement systems could not be accomplished since no calibrated portable load cell was available. However, when the press was first constructed the load cells were calibrated using a Baldwin testing machine as the reference. A rough check was made to verify the accuracy of the calibration data that had been obtained by Lee against a pressure gage. This consisted of the following procedure:

1. Place a bar of steel of other suitable into the press which is capable of withstanding the full force of the punch and binder.

2. Set the supply pressure to the press to a desired value using the variable pressure adjustment on the pump controller box. The pressure can be monitored by a pressure gage connected to the supply just after the selector valve.

3. Manually adjust the input current to positive full scale (+10 mA) to allow the cylinders to close and to become fully pressurized.
4. Measure the voltage output of the instrumentation amplifier.

5. Repeat steps 2 - 4 for a variety of pressures that cover the whole range of available output pressures from the pump.

The data that was obtained in the above manner agreed with the original calibration data obtained by Lee quite well for the punch force considering the low resolution of the pressure gage used to monitor the supply pressure. See Table 2.2.

Table 2.2 Verification of load cell calibration.

<table>
<thead>
<tr>
<th>Input Pressure (psi)</th>
<th>Output Voltage (V)</th>
<th>Voltage * Scale (N)</th>
<th>Pressure * Cyl. Area (N)</th>
<th>Percent Difference (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binder Force</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>550</td>
<td>1.89</td>
<td>54158</td>
<td>50423</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>3.57</td>
<td>101539</td>
<td>91678</td>
<td>10</td>
</tr>
<tr>
<td>1500</td>
<td>5.45</td>
<td>154439</td>
<td>137517</td>
<td>12</td>
</tr>
<tr>
<td>2000</td>
<td>7.31</td>
<td>206803</td>
<td>183356</td>
<td>12</td>
</tr>
<tr>
<td>2500</td>
<td>9.14</td>
<td>258323</td>
<td>229195</td>
<td>12</td>
</tr>
<tr>
<td>2610</td>
<td>9.61</td>
<td>271555</td>
<td>239279</td>
<td>13</td>
</tr>
<tr>
<td>Punch Force</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>550</td>
<td>1.09</td>
<td>30467</td>
<td>30744</td>
<td>-1</td>
</tr>
<tr>
<td>1000</td>
<td>2.04</td>
<td>56203</td>
<td>55898</td>
<td>1</td>
</tr>
<tr>
<td>1500</td>
<td>3.06</td>
<td>84142</td>
<td>83847</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>4.15</td>
<td>113687</td>
<td>111796</td>
<td>2</td>
</tr>
<tr>
<td>2500</td>
<td>5.20</td>
<td>142252</td>
<td>139745</td>
<td>2</td>
</tr>
<tr>
<td>2610</td>
<td>5.51</td>
<td>150686</td>
<td>145894</td>
<td>3</td>
</tr>
</tbody>
</table>

After completing the experiments on the conical cups and before beginning the experiments on the square pan, the gain of the binder force amplifier had to be reduced. When forming square pans the binder forces required are much higher than those used for the conical cups because the conical cups experience a great deal of self-restraining force due to the compressive hoop stresses developed. Since the DT2811 only allows for inputs in the range of +/- 5 volts, only half of the binder force capacity could be used with the old gain setting. See Table 2.2. The amplifier gain of the 2B31K is determined by the following equation:
\[ G = 1 + \frac{94 \, \text{k}\Omega}{R_g} \cdot \frac{20 \, \text{k}\Omega}{R_A + 16.2 \, \text{k}\Omega} \]

Where: [reference] \( R_g = \text{Gain Resistor} \)
\( R_A = \text{Fine Tune Adjustment} \) \( (2.3) \)

From this equation, it can easily be determined that doubling the gain resistor will result in the gain being reduced by half.

The gain resistor was changed and the above procedure for verifying calibration was repeated to double check the output voltage. The gain had indeed be reduced by a factor of two. However, at some point in the processes of verifying the new gain some damage was done to the instrumentation amplifier circuit. A new 2B31K instrumentation amplifier and AC1213 mounting card was purchased for the system. This remedied the problem, but because of the new components, the old calibration data was of no use. Therefore, the verification procedure was conducted one final time and the data obtained was used to set the calibration constant for the binder force. Linear regression analysis of the data yielded a scale factor of 49711 N/V with a R squared value of 0.9997. This calibration was conducted with a gain resistor of 94.8 Ω and an excitation voltage of 5.00 Volts.

Summary

The experimental apparatus has been much improved by the work presented in this chapter. The connection to a new hydraulic pump has increased its dynamic operating range (this is discussed in detail in the next chapter). The new control software has made the equipment easier to operate by prompting the user to perform all tasks required to complete an experiment in a sequential manner. This new software has also improved the data records of the experiment that are produced. First, the graphical output has been made higher resolution and plain English descriptions have been added. Second, the data files are clearly labeled with a header section at the top of each file that fully describes their
contents. Because of the high speed capability of the press, the new software was also made much faster to acquire data and update the controller with a cycle time as short as 10 milliseconds.
Chapter 3: Press Dynamics

Introduction

After completing the press modifications discussed in the previous chapter, experiments were conducted to understand the dynamics of the press. An understanding of the capabilities and limitations of both the punch displacement servo system and the binder force servo system was needed. This information would be used to determine what forming experiments the press was capable of conducting and also to characterize the closed loop controller performance. During a closed loop forming experiment, the controller would sometimes fail to accurately track the input trajectory that had been specified. Without data on the dynamic performance of the press, it was difficult to determine if this failure to accurately track the input was an artifact of poor controller performance or of hardware limitations of the press.

This chapter presents the experiments conducted to map the dynamic characteristics of the active binder force forming press. All of the tests involve writing a specialized software program to execute the experiment. However, since the program code was very modular, specialized testing programs were easy to create. One or two new blocks of C code combined with the standard program blocks were all that was required for each different test program.

Punch Dynamics

The punch displacement dynamics were studied first since one research goal was to operate the press at high punch velocities to simulate a production environment. A block diagram of the hardware punch displacement servo system is shown in Figure 3.1. This
The punch displacement servo loop uses an analog closed loop controller. The servo amplifier is a Moog model 121-103 configured for proportional control. The servo valve is a Moog model 30 12. 3000 I 1000 4CA R BUN. The punch cylinder is a Parker model JJ2H19 double acting cylinder with a 4 inch bore and a 8 inch stroke. The displacement transducer is a Hewlett Packard model 7DCDT-3000 linear variable displacement transformer.

![Block Diagram](image)

**Figure 3.1 Block diagram of the punch displacement servo loop.**

**Displacement Step Response**

Step tests were conducted first because they are rich in information and easy to conduct. Initially, a large step of 25 mm was given as an input to the punch displacement servo. This large displacement step resulted in saturation of the punch servo as can be seen by the constant velocity region in Figure 3.2. During the test, saturation of the servo valve was apparent because the current to the servo valve went to full scale. This saturation was caused by the flow rate limit of the servo valve. This test clearly shows that the servo valve limits the punch velocity to less than or equal to 18.3 mm/s under no load conditions.

This test was repeated with a blank in the press to determine what effect forming forces would have on the punch displacement response. See Figure 3.3. In this test, the maximum punch velocity was reduced to 17.0 mm/s due to the forming load on the punch. The higher pressure required in the cylinder reduces the available pressure drop across the valve, thereby limiting the maximum flow.
Figure 3.2 Punch displacement step response with an input step of 25 mm at $t = 0$.

Figure 3.3 Punch displacement step response with an input step of 25.4 mm at $t = 0$ and an AL 2008-T4 sheet metal blank in the press with conical cup tooling.
Because the preceding tests resulted in saturation of the punch displacement servo and thus did not provide much information on the dynamics of the analog servo controller loop. Therefore, a test with a smaller step size was conducted to better understand the closed loop servo performance. Figure 3.4 shows the results of a test with a 2.5 mm input step. This plot shows that the system exhibits a characteristic first order response with a time constant of 0.21 second. The response of a first order system to a step input can be modeled as:

\[ u(t) = K(1 - e^{-t/\tau}) \]

Where:

- \( K \) = the step size
- \( \tau \) = the time constant

(Ogata 1990) (3.1)

This equation is also plotted in Figure 3.4 to compare it to the actual time response of the system.

![Figure 3.4 Punch displacement step response with a input step of 2.5 mm at t = 0.](image)

Figure 3.4 Punch displacement step response with a input step of 2.5 mm at \( t = 0 \).
Displacement Ramp Response

After completion of the step response tests, several ramp displacement tests were conducted. During forming experiments, the input to the punch is a constant velocity ramp, and thus, an indication of the tracking accuracy of the punch was needed for various input ramps with different velocities (or slopes). The linear velocity operating range of the punch servo without saturation of the servo valve also needed to be determined.

Ramp tests show that the punch displacement lags behind the input. See Figures 3.5 - 3.6. This tracking error for a ramp input is characteristic of a first order system and can be explained by the operation of the servo system. The servo valve used to control the punch hydraulics is a flow control valve that produces an output flow rate proportional to the electrical current input. Since the punch position (Y) is an integration of the flow rate (Q) which is proportional to the current (I) which in turn is proportional to the error voltage (V_e), the punch velocity (dY/dt) is proportional to the error voltage. Thus during steady state, constant velocity operation, the punch position will always lag behind the input position signal in an amount proportional to the desired velocity. It can be easily shown that proportionality constant is equal to the first order time constant of the system. Figure 3.8 shows this proportionality, but the data also has a negative DC offset. The proportionality constant determined from the regression analysis of this data is 0.25 second which agrees fairly well with the time constant of 0.21 second determined from the step test above.
Figure 3.5 Punch displacement ramp test with a slope of 5 mm/s.

Figure 3.6 Punch displacement ramp test with a slope of 10 mm/s.
Figure 3.7 Punch displacement ramp test with a slope of 15 mm/s.

Figure 3.8 Punch position lag plot versus punch velocity.
In addition to the above ramp tests, another set of ramps tests was conducted to determine the linear operating range of the punch servo. In this set, the punch velocity was determined as a function of the punch servo command increment used in the controller program. The controller program used was the full featured version that would be used to conduct the constant binder force experiments. This program was used because it called all of the data input channels and executed all of the commands that would be used in a normal forming experiment. Therefore, the timing of this program was the same as it would be during forming trials. The timing of the software loop (assumed to be constant) and the servo command increment combine to determine the punch velocity. Figure 3.10 shows the results of these tests. This plot very clearly shows the saturation velocity of 18.3 mm/s as also seen earlier in the step tests. It also shows that the linear operating region of the punch servo is from 0 to ~15 mm/s.

Figure 3.10 Punch velocity as a function of the punch servo command increment.
Binder Dynamics

After having characterizing the punch servo dynamics, the binder servo dynamics were studied. Since closed loop control is used to vary the binder force during forming experiments, dynamic performance data is needed to make sure that the binder can change forces fast enough to keep up with the changing outputs of the digital control loop. Operating speed of the press can be limited either by the punch displacement response or by the binder force response.

The binder force servo system is very similar to the punch displacement servo system. A flow control servo valve is again used to control a double acting hydraulic cylinder. A block diagram of the binder force servo system is shown in Figure 3.11. The servo amplifier is a Moog model 121-103 configured for proportional control the same as in the punch displacement controller. The servo valve is a specially built Moog series 30 model 30-245A flow control servo valve designed with a lower flow rate thus reduced pressure gain over the stock valve. In a force control system such as this, a low pressure gain is required to avoid resonance in the valve spool when using a flow control servo valve. (Woodworth 1993) The binder cylinder is a Enerpac Holl-O Cylinder model RRH-1001 with an effective area of 20.61 square inches and a 1.5 inch stroke. The force element uses a load cell with a four-arm Wheatstone bridge and an Analog Devices 2B31K instrumentation amplifier.

![Block diagram of the binder force servo system.](image)

Figure 3.11 Block diagram of the binder force servo system.
In this servo system, the load cell acts as a stiff spring converting a binder displacement into a binder force allowing a flow control servo valve to be used to control force. More typically, a pressure control servo valve is used in systems when force control is desired, and flow control servo valves are used in positioning systems such as the punch displacement system. Since a flow control valve is used, the proportional gain of the servo amplifier must be kept to a small value to avoid resonance in servo valve as it tries to maintain a constant pressure drop across the hydraulic cylinder.

Figure 3.12 shows the characteristic resonance for this particular hardware combination. The frequency of the oscillations is 17.6 Hz and amplitude of these oscillations varies with the proportional gain of the servo amplifier. A trial and error method was used to set the proportional gain of the system to the maximum value without large oscillations. The gain was adjusted by turning a potentiometer on the Moog 121-103 servo amplifier.

![Characteristic binder force oscillations when using a high gain proportional controller.](image-url)
**Force Step Response**

Again, step tests were conducted first. To guarantee that the binder cylinder was in a initially closed position, the step started from a non-zero force value. This was necessary to make sure that the test was measuring the binder force response and not the displacement response. Figure 3.13 shows the binder force response to an input step from 1 KN to 40 KN. This figure shows that the binder force servo behaves as a higher order system with a rise time of 0.06 second.

![Graph showing force step response](image)

**Figure 3.12** Binder force step response for a step from 1 KN to 40 KN at t = 0.

**Velocity Limitations**

The above data shows that the binder force servo is very fast and well damped. However, this data was all obtained when the press was empty (i.e. no cup was being formed). This means that the binder cylinder did not have to undergo any large displacements to provide any of the forces that the tests required. However when forming conical cups at low binder forces, wrinkles form in the flange of the part that cause the
binder to open. In order for the binder servo to maintain the desired force, the cylinder must be able to open and close fast enough to allow expansion of the flange wrinkles or to compress the flange wrinkles. These flange wrinkles act as compression springs that grow in length as the forming cycle progresses. Thus, to maintain a constant binder force the binder displacement must increase during the forming cycle. See Figure 3.13. Note that the characteristic oscillations in the binder force disappeared after the flange wrinkles started to form (t = 1 sec). This change in dynamic behavior is caused by a change in the binder stiffness because of the addition of a soft spring (i.e. the flange wrinkles).

This figure clearly shows that during the formation of flange wrinkles the binder was not able to retract fast enough to maintain a constant binder force. The binder displacement plot shows that near the end of the forming cycle (t > 2.5 sec) the binder servo became saturated at its maximum velocity of 2.0 mm/s. However, the flange wrinkles were expanding faster than this, and thus, the binder force increased as the wrinkles tried to push the die and binder farther apart. At the very end of the forming cycle (t > 2.8 sec), the binder displacement began to catch up with the wrinkle formation, and thus, the binder force began to drop. From these plots, it is clear that the binder saturation velocity must be increased to make it possible to conduct a forming test under these conditions while maintaining a constant binder force.

To improve the displacement response of the binder servo, a larger flow rate valve in needed. However, as stated earlier a large capacity flow control servo large will result in resonance when used in a force control loop. A better valve to use in this application is a pressure control valve servo valve. Pressure control servo valves are designed to provide a differential output pressure (ΔP₀) that is proportional to input current (i) thus controlling force directly. Resonance is not a problem with these valves when used in a force control loop because a different feedback circuit is used within the valve that is suited to this application.
Figure 3.13 Conical cup forming data using AL 2008-T4 at a target CBF = 4.4 KN with a punch velocity = 13.7 mm/s. a) Binder displacement vs. time and b) Binder force vs. time.
load cylinder at a constant input current) in the range of 10 to 40 psi/cis. This translates to a binder force error in the range of 1.5 KN to 5.9 KN at a binder velocity of 2.0 mm/s in an open loop system as opposed to an error of 10 KN with the current setup. However, the closed loop controller would reduce this error even further by changing the current to the valve in response to the error. The use of a pressure control valve also permits larger control gains to be used in the system because resonance in the servo valve would no longer be a problem.

It is worth noting that this binder force tracking error is only present at low binder force settings. At high binder forces, flange wrinkles do not develop thus the binder is not required to undergo large displacements to maintain the desired binder force.

**Summary**

The punch displacement servo system was shown to be a first order system with a time constant of between 0.21 and 0.25 second and a saturation velocity of 18.3 mm/s. The binder force servo system behaves as a higher order system with a rise time of 0.06 second and a saturation velocity of 2.0 mm/s which limits the binder force tracking ability of the system if appreciable binder displacements are required to maintain the desired force. To increase the saturation velocity of the binder servo, a new control valve would have to be installed, preferably a pressure control valve not a flow control such as in the system now.
Chapter 4: Square Pan Experiments

Introduction

As a further demonstration of the closed loop process control methodology that has been developed for sheet metal forming, a series of experiments were conducted with aluminum sheet and the square pan tooling with equal corner radii. See Figure 4.1. It is important to conduct tests with non-axisymmetric tooling because the behavior is significantly different from the axisymmetric tooling (i.e. the conical cup). With axisymmetric tooling the material exhibits a large amount of self binding because the material is continually being drawn into smaller radius regions and thus is under compression. With straight sections, no compression of the material is present since the section length is constant as the material draws into the die, and thus, no self binding of the material occurs. This results in purely fictional binding of the sheet. The square pans tolling used contains elements of both types of behavior as well as transitions zones between the two.

As a baseline and to develop a controller input trajectory, a set of constant binder force experiments was conducted. These experiments were followed by a set of closed loop control experiments. This chapter presents the results of these experiments and also discusses the particular closed loop control methodology used. In all of the forming experiments forming experiments presented in this chapter, the following conditions were held constant: blank diameter = 230 mm, lubrication = 8 (~0.35 ml) drops of SAE 10W-30 oil/STP mixture (3:1 ratio) per side of the blank, material AL 2008-T4 (See Appendix B for tensile test data). Also, during the experiments, binder displacement data was not recorded because no binder displacement occurs.
Figure 4.1 Geometry of the square pan tooling with equal corner radii.

Note: All Fillet Radii 6.35 mm
Constant Binder Force Experiments

It is useful to conduct a series of CBF forming experiments to use as a baseline to compare the closed loop controller against. Also, data from these CBF tests is used to develop the input trajectory for the closed loop controller. Therefore, I conducted CBF tests over the range of binder forces from 50 KN to 200 KN. The results of these forming experiments is displayed in the traditional failure height diagram in Figure 4.2. The 'optimal' CBF is defined as the binder force at which the maximum pan height can be achieved without failure. In this case, the 'optimal' CBF is 115 KN.

![Diagram showing failure height vs. constant binder force](image)

Figure 4.2 Constant binder force failure height diagram for AL 2008-T4 square pans.

Failure Modes Defined

Before proceeding with the closed loop control experiments. The failure modes for the square pans need to be discussed. Tearing failure does not require a detailed description. Tensile fracture occurs at the punch nose in one of the corners of the pan this follows the behavior of the conical cup vary closely. See Figure 4.3. Tearing is evident
during the experiment by a popping sound and a sharp decline in the punch force. Torn cups are plotted as diamonds in Figure 4.2.

Figure 4.3 Photograph of a torn pan.

Buckling in the square pan geometry is more subtle to detect than in the conical cup. With the conical cups, buckling occurred as high frequency variations in the radius of the free section, and thus was easily quantifiable by measuring the radial variations with a dial indicator while the part was rotated in a lathe. The only shape effect in the conical cup that had to be filtered out of the data was a slight ovalness of the cup which would show up as a low frequency variation with a period of 180 degrees of cup rotation. (A shape effect is a global variation in the part shape.) Buckling in the square pans must be distinguished from overall shape effects as well. In the corner regions, the pan experience the same die wrap effect that was evident in the conical cup. (See Appendix A for conical cup shape data.) This causes the pan to experience bending with reverse bending in the corners. However, this behavior is not present in the straight sidewall regions of the pan. In these regions, the
sheet metal follows a tangent line from the die profile radius to the punch profile radius. Thus the sidewalls sections are recessed inward with respect to the corners sections of the pan which is considered to be a shape effect and not buckling. See Figure 4.4. Thus these cups were considered to be good, and they are plotted as circles in Figure 4.2.

![Figure 4.4 Photograph of a good pan with a recessed sidewall.](image)

Buckling was considered to have occurred when the sidewall sections exhibited out of plane deformation. This deformation would take the form of a long wave length wrinkle with a peak in the middle of the sidewall. See Figure 4.5. The wrinkle amplitude was determined using a depth micrometer to measure the height difference between the peak and valley of the wrinkle. See Figure 4.6. A wrinkle amplitude of greater than or equal to 0.002 inch was defined as buckled. Buckled cups are plotted as triangles in Figure 4.2.
Figure 4.5 Photograph of a buckled pan.

Figure 4.6 Measurement of wrinkle amplitude with a depth micrometer.
Closed Loop Control Experiments

The digital controller used in these experiments is the same proportional plus integral controller first used by Lee (1986). This controller was also used by Fenn (1989) and Jalkh (1993) in their closed loop experiments. The equation for digital implementation of this controller, assuming a constant time step, is as follows:

\[ F_s = F_{n} + (K_i + K_p) \cdot e_n - K_p \cdot e_{(n-1)} \]

Where:

\( F = \) Binder Force
\( K_i = \) Integral Gain
\( K_p = \) Proportional Gain
\( e = \) Error (target – actual)

Subscript \( n \) denotes the current time step
and subscript \( n - 1 \) denotes the previous.

The error term in the controller equation depended upon the variable being tracked during the experiment. It was defined as the difference between the target value and the actual value of the variable being tracked. Three variables have been used as controller inputs in previous research: binder displacement, tangential force and normalized average thickness. Each of these input variables worked successfully in making the failure height of the part being formed insensitive to initial binder force. Before I could begin conducting close-loop control forming experiments, I had to choose a variable to track with the controller and define the optimal trajectory of that variable.

Determination of an Input Trajectory

I chose to use the punch force directly as the controller input variable. Previous closed loop control experiments used the tangential force in the cup as the control variable (Fenn and Hardt 1993). This was chosen because it directly related to the stress in the sheet metal. However, the tangential force is merely a cosine scaling of the punch force. Since this scaling does not contribute any additional information, it does not provide any performance benefit to the controller.
the early stages, the closed loop controller could be activated earlier in the forming cycle. This results in faster convergence to the proper binder force for the forming conditions. Fast convergence of the binder force is desired because it will keep the forming progressing along the proper strain history.

![Graph](image)

**Figure 4.7** Punch force input trajectory along with the actual optimal punch force trajectory as measured in at the optimal CBF.

**Tuning the Digital Controller**

Because of the modifications of the press and of the control software, it was necessary to tune the controller for optimal tracking performance. Tuning of the controller could not be accomplished by simulation because we did not have a dynamic model of the press and it was deemed too laborious to develop one. Therefore, a series of experimental forming trials were conducted to determine the optimal values of the three controller parameters: the proportional gain ($K_p$), the integral gain ($K_i$) and the controller activation height ($Y_a$).
The experimental optimization of these three parameters proceeded in a trial and error fashion. A more scientific approach would have been to use design of experiments to investigate the three dimensional space spanned by these variables. However, this level of detail was not required to meet the performance requirements for the controller. In order to converge to the input trajectory as early in the forming cycle as possible it is desirable to active the controller as early in the experiment as possible. Also, increasing the proportional gain will cause the controller to converge to the desired trajectory more rapidly up to a limit. Beyond this limit, any further increase in the proportional gain will result in an oscillatory behavior. Increasing the integral gain will also result in a more rapid response of the controller. However, too large of a integral gain will result a large overshoot due a wind-up phenomena. As a result of experimentation, the following parameters were chosen:

\[ K_p = 2 \text{ N/N} \]
\[ K_i = 2 \text{ N/N} \]
\[ Ya = 3 \text{ mm} \]

These values resulted in a very stable controller performance with no oscillatory behavior and a rapid convergence to the desired trajectory within 10 mm of punch displacement. See Figure 4.8.
Figure 4.8 Controller tracking performance with $K_p = 2 \text{ N/N}$, $K_i = 2 \text{ N/N}$, $Y_a = 3 \text{ mm}$.

**Experiments**

The closed loop punch force controller was tested over a wide range of initial binder force values from 50 KN to 200 KN. The controller performed very well in all of the forming experiments regardless of the initial binder force setting. In all cases, the actual punch force in the experiment converged to the input trajectory within 10 mm of punch displacement. The binder forces during these experiments converged to some final value near the previously defined optimal constant binder force in the same amount of punch displacement. See Figure 4.9. However, the final value of the binder force varied from one experiment to the next even with the same initial binder force. See Figure 4.10. All of these final binder forces were within approximately 10% of the 'optimal' constant binder force defined earlier.
Figure 4.9 Typical binder force histories during forming when using CLPF control.

Figure 4.10 Two different binder force histories during forming when using CLPF control with an IBF = 50 KN.
Figure 4.11 CLPF control failure height diagram for AL 2008-T4 square pans. The CBF failure height curve is plotted as a reference.

The failure height diagram for forming square pans when using the CLPF controller is plotted in Figure 4.11. The circles again represent good parts, and the triangles represent buckled parts. Note that none of the formed parts were torn. This was likely due to the input punch force curve used in the experiments being slightly to the buckling side of the 'optimal' CBF.

Summary

The closed loop trajectory tracking control strategy has again shown its ability to make the failure height of a part insensitive to the initial binder force. In this chapter, it was successfully demonstrated for a symmetric square pan formed out of AL 2008-T4 automotive body sheet.
Chapter 5: Conclusions and Future Research

In today’s competitive environment where improvements in quality are a must, the concept of using a closed loop process controller in sheet metal forming has been shown to have many possible benefits. Fenn and Hardt (1993) showed that using a trajectory tracking controller could make the forming process robust to variations in lubrication conditions, material thickness, blank diameter and blank alignment when forming conical cups and square pans out of AKDQ steel. Jalkh et al. (1993) extended this work to include the forming of conical cups out of several aluminum alloys used in automotive applications. In this thesis, the work was further extended to the forming of square pans out of AL 2008-T4 automotive body sheet.

Clearly the trajectory tracking control algorithm has shown itself to be repeatable and applicable to various drawing geometries. The experimental work conducted over the past 10 years has resulted in good procedures for developing input trajectories for square pans and conical cups. The time has come to expend the research into more general part shapes which will promote transfer of this technology into industry. Several areas of research would help to promote this.

I propose that the next researcher investigate an actual stamping operation to determine was variable or variables would be best to track in a production environment. The research would have to answer the question of what trajectories would give the desired results along all dimensions of quality not just part failure. In addition, the research would have to determine what trajectories are practical to follow given the availability of instrumentation on production presses.
It would also be valuable to investigate other possible uses for the general control strategy that has been developed. Using a closed loop controller to track a global phenomena (such as punch force or draw-in) during a process to control local phenomena (such as stress and strain) is an ideal that can have a positive impact on many processes. It is worth spending some time to identify processes that lend themselves to this idea so that a general methodology can be developed for implementing this type of control strategy in the various systems.
Appendix A: Effects of Binder Force on Conical Cup Shape Profile

Introduction

In all of the research conducted here at MIT on closed-loop control of sheet metal forming, part failure has been defined as tearing or wrinkling. While these two defects definitely result in a unacceptable or failed part, there are other defects that also result in a sheet metal part being unacceptable including: geometric variations (shape errors), surface defects (scratches, flanking of coatings, dimples, etc.) and material variations (amount of plastic strain, microstructure, etc.). This list is by no means exhaustive. It is given here to point out that there are requirements on sheet metal parts in addition to the absence of tearing and wrinkling. In this appendix, a brief look is taken into the effects of binder force on the final shape of conical cups.

Experiments

To answer the question of what effect binder force has on cup shape, a series of cups were formed to a specified punch displacement with all of the forming parameters fixed except for the binder force. Three different constant binder force values were used: 2 KN, 40 KN and 80 KN. The other forming parameters were set as follows:

- blank diameter = 152.4 mm (6.00 in)
- punch diameter = 63.5 mm (2.50 in)
- die diameter = 101.6 mm (4.00 in)
- lubrication = 5 drops (~0.25 ml) SAE 10W-30 oil, STP mixture (3:1 ratio) per side
Figure A.1 Conical cup shape profiles after 15 mm of punch travel for CBF's of 2, 40 and 80 KN.

Figure A.2 Conical cup shape profiles after 15 mm of punch travel for CBF's of 2, 40 and 80 KN showing only the free section.
After the cups had been formed, a coordinate measuring machine was used to measure one vertical profile of each cup from the center out to the edge along the rolling direction of the sheet. This data is plotted in Figure A.1 and A.2. This data clearly shows that the binder force does have a measurable effect on the unloaded shape of conical cups.

Discussion of Results

The data presented in this chapter does not provide enough information to draw conclusions about how binder force variations introduced by the closed-loop process controller would influence final part shape. It does however indicate that some shape effects may be present. If the closed-loop control strategy is ever to be used in the stamping of production parts, the issue of controlling the final part shape must be addressed. Most production parts must be made to a very specific shape to meet their functional requirements, and thus variations introduced by the controller may not be tolerable.
Appendix B: Material Tensile Test Data

This appendix lists the tensile test data for the particular lots of AL 2008-T4 automotive body sheet used in the forming tests presented in this thesis. Lot A was used for all of the conical cup tests and was also used by Jalkh in his experiments. Lot B was used for all of the square pan tests.

Table B.1 Material property data for the two lots of material used in the forming experiments presented in this thesis.

<table>
<thead>
<tr>
<th></th>
<th>Lot A</th>
<th>Lot B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yield Stress (MPa)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0°</td>
<td>154.1</td>
<td>159.3</td>
</tr>
<tr>
<td>30°</td>
<td>147.1</td>
<td>149.0</td>
</tr>
<tr>
<td>45°</td>
<td>146.2</td>
<td>152.1</td>
</tr>
<tr>
<td>60°</td>
<td>149.2</td>
<td>152.9</td>
</tr>
<tr>
<td>90°</td>
<td>144.2</td>
<td>146.4</td>
</tr>
<tr>
<td>r Ratio</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0°</td>
<td>0.70</td>
<td>0.60</td>
</tr>
<tr>
<td>30°</td>
<td>0.52</td>
<td>0.48</td>
</tr>
<tr>
<td>45°</td>
<td>0.44</td>
<td>0.46</td>
</tr>
<tr>
<td>60°</td>
<td>0.42</td>
<td>0.66</td>
</tr>
<tr>
<td>90°</td>
<td>0.52</td>
<td>1.31</td>
</tr>
<tr>
<td>Power Law Strain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardening</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K (MPa)</td>
<td>528</td>
<td>515</td>
</tr>
<tr>
<td>n</td>
<td>0.265</td>
<td>0.26</td>
</tr>
</tbody>
</table>
Figure B.1 True stress vs. true strain plot for Lot A in the $0^\circ$ direction.

Figure B.2 True stress vs. true strain plot for Lot B in the $0^\circ$ direction.
Appendix C: Controller Programs

FB_CONST.C  Program for conducting constant binder force forming experiments.
CLPF.C  Program for conducting closed loop punch force experiments.
Program written by: Robert S. Bakkestuen

Controller program for conducting constant binder force experiments using the conical cup hydraulic press. The program displays a real time graph showing the punch force, binder force, draw-in, and binder disp. plotted vs. punch disp. After the experiment has run to completion or the loop has been stopped by a keyboard input, the program saves the data in a file with a name in the form mmddyyyn.dat where: mm is the month (01-12), dd is day of the month (01-31), yy is the year (00-99), and nn is the experiment number for that day (1-99). The program code is divided into several small blocks and makes use of many constant definitions to make editing the program easier. To change a program parameter, all that is required is to change the definition statement and recompile the program to update the code throughout. For documentation purposes, please note all changes made to the program in the revision section below.

<table>
<thead>
<tr>
<th>Rev</th>
<th>Name</th>
<th>Date</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R. Bakkestuen</td>
<td>7/23/93</td>
<td>added digital loop fb cont.</td>
</tr>
<tr>
<td>2</td>
<td>R. Bakkestuen</td>
<td>7/28/93</td>
<td>added binder disp. data call</td>
</tr>
<tr>
<td>3</td>
<td>R. Bakkestuen</td>
<td>8/3/93</td>
<td>removed digital loop fb cont.</td>
</tr>
<tr>
<td>4</td>
<td>R. Bakkestuen</td>
<td>8/14/93</td>
<td>changed DCDT calib. constants, BINDER_FORCE_CHANNEL, and PUNCH_SERVO_ZERO</td>
</tr>
<tr>
<td>5</td>
<td>R. Bakkestuen</td>
<td>8/19/93</td>
<td>added check for cup failure and punch over travel, retract punch line, and if statement to limit the punch servo command to 4095 to the controller routine</td>
</tr>
<tr>
<td>6</td>
<td>R. Bakkestuen</td>
<td>8/24/93</td>
<td>added REP_F to be used in calc. to data loss to due int. overflow</td>
</tr>
<tr>
<td>7</td>
<td>R. Bakkestuen</td>
<td>8/25/93</td>
<td>corrected sign error in binder disp. and draw-in measurement, added T4 to 2008 material spec., changed binder_disp. round off to 4 digits after the decimal</td>
</tr>
<tr>
<td>8</td>
<td>R. Bakkestuen</td>
<td>8/26/93</td>
<td>changed BINDER_SERVO_ZERO to 1420, changed draw_in meas. to single point from circumferential</td>
</tr>
<tr>
<td>9</td>
<td>R. Bakkestuen</td>
<td>8/27/93</td>
<td>removed neg. sign from draw_in calc</td>
</tr>
<tr>
<td>10</td>
<td>R. Bakkestuen</td>
<td>11/29/93</td>
<td>changed punch step value to a const. changed value from 10 to 4, added stabilize binder routine</td>
</tr>
<tr>
<td>11</td>
<td>R. Bakkestuen</td>
<td>1/13/94</td>
<td>changed PUNCH_DISP_COMMAND_INC from 4 to 1</td>
</tr>
<tr>
<td>12</td>
<td>R. Bakkestuen</td>
<td>3/2/94</td>
<td>added i loop to zero routines, added timer to control loop to control punch velocity, changed average velocity calc. to eliminate end effects, changed punch disp inc to match fb_var style</td>
</tr>
<tr>
<td>13</td>
<td>R. Bakkestuen</td>
<td>3/3/94</td>
<td>changed ave_punch_vel to global var and added it in data file, changed data file format to import into excel better</td>
</tr>
<tr>
<td>14</td>
<td>R. Bakkestuen</td>
<td>4/11/94</td>
<td>changed FREQUENCY from 200000 to 150000</td>
</tr>
<tr>
<td>15</td>
<td>R. Bakkestuen</td>
<td>4/13/94</td>
<td>changed MAX_FORCE from 50k to 100k and GAIN from 2 to 1, disabled delta force check in controller</td>
</tr>
</tbody>
</table>
 changed BINDER_FORCE_SCALE from 28153 to 56362 N/V to reflect change of gain resistor on strain gage amp, changed max force plot scales, separated punch and binder force plot scales, added tic marks to the top of the graph and FREQUENCY from 15000 to 10000
removed FREQUENCY from 10000 to 6000

*/
#include <dos.h>     /* include dos functions file */
#include <string.h>   /* include string functions file */
#include <time.h>     /* include time functions file */
#include <graph.h>    /* include graphics definitions file */
#include <math.h>     /* include math functions file */
#include <stdio.h>    /* include standard I/O file */
#include <stdlib.h>   /* include standard library file */
#include "lpcclerrrs.h" /* include LPCLAB error codes */
#include "lpccldefs.h" /* include LPCLAB function declarations */

/* Define A/D input channels on the DT2811 */
#define BINDER_FORCE_CHANNEL 7
#define PUNCH_FORCE_CHANNEL 1
#define PUNCH_DISP_CHANNEL 2
#define BINDER_DISP_CHANNEL 3
#define DRAW_IN_CHANNEL 4

/* Define D/A output channels on the DT2811 */
#define BINDER_SERVO_CHANNEL 0
#define PUNCH_SERVO_CHANNEL 1

/* DT2811 constants */
#define GAIN 1    /* Gain used when reading values */
#define FREQUENCY 6000  /* Clock frequency for timing */
#define TIMING_SOURCE 0 /* Internal clock and trigger */

/* Data acquisition constants */
#define REPS 10  /* Number of times to read each channel when averaging */
#define REPS_F 10.0  /* Float value of REPS used for calc */

/* Calibration constants */
#define BINDER_FORCE_SCALE 50000.  /* N/volt */
#define PUNCH_FORCE_SCALE 27205.  /* N/volt */
#define BINDER_DISP_SCALE 880.28  /* Analog/mm */
#define PUNCH_DISP_SCALE 31.104  /* Analog/mm */
#define DRAW_IN_SCALE 27.252  /* Analog/mm */
#define PUNCH_SERVO_ZERO 1560  /* Analog value that moves punch up until blank is in contact with to P

/* Controler constants */
#define PUNCH_DISP_INC 0.2  /* Analog punch movement step size */
#define LOOP_TIME 0.2  /* Time for each loop during forming */
cycle (seconds) */

/* Plotting constants */
#define XZZERO 105 /* X zero coordinate */
#define YZZERO 445 /* Primary Y zero coordinate */
#define XPIXELS 500 /* Number of X pixels in graph */
#define YPIXELS 400 /* Number of Y pixels in graph */
#define MAX_TIME 10 /* Max time (sec) */
#define MAX_PUNCH_DISP 50 /* Max punch displacement (mm) */
#define MAX_PUNCH_FORCE 100 /* Max punch force (Kn) */
#define MAX_BINDER_FORCE 250. /* Max binder force (Kn) */
#define MAX_DRAW_IN 10 /* Max draw-in (mm) */
#define ARRAY_SIZE 1500 /* Size of data arrays */

/* Declare global program variables */
char dos_date[15];
char dos_time[15];
int test_number;
int total_punch_disp;
int geometry_code;
floa fpunch_dia;
int radii_code;
floa fbblank_dia;
int material_code;
floa fsheطفthickness;
int lubrication_code;
floa finput_press;
char file_name[30];
long const_binder_force;
float binder_force_zero;
int binder_disp_zero;
float punch_force_zero;
int punch_disp_zero;
int draw_in_zero;
float ave_punch_vel;

/* Arrays to store experimental values before they are written to disk */
float t[ ARRAY_SIZE ];
floa fpunch_disp[ ARRAY_SIZE ];
floa fpunch_force[ ARRAY_SIZE ];
floa fbinder_disp[ ARRAY_SIZE ];
floa fbinder_force[ ARRAY_SIZE ];
floa fdraw_in[ ARRAY_SIZE ];

/* Declare function types */
void setup( void );
void get_file_name( void );
void zero_forces( void );
void initialize_punch( void );
void initialize_binder( void );
void wait( void );
void stabilize_binder( void );
void zero_ddct( void );
void draw_graph( void );
int controller( void );
void save_data( int );
float force( int, float, float);
float position( int, float, int);

void main( void )
/* Program execution block */

Page 72
{   int num_steps;

setup(); /* Setup experiment */
zero_force(); /* Zero punch and binder force */
initialize_punch(); /* Move punch to initial position */
initialize_binder(); /* Set binder force to initial value */
wait(); /* Wait for keyboard input */
zero_dcdt(); /* Zero draw-in and punch dcdt's */
draw_graph(); /* Draw graph on screen */
num_steps = controller(); /* Run controller */
save_data(num_steps); /* Save experimental data to disk */
lp_terminate(); /* Terminate LPCLAB operations */
exit(0);
}

} /* End of main */

void setup( void ) /* Function for setting up experimental variables */
{
   /* Setup screen for output */
   _setbgcolor( _BLUE );
   _clearscreen( _GCLEARSCREEN );
   _settextcolor( 15 );

   /* Setup DT2811 board */
   lp_initialize(); /* Initialize the LPCLAB subroutine */
   lp_select_board( 1 ); /* Select board 1 */
   lp_set_clock_frequency( FREQUENCY ); /* Sets clock frequency for data acquisition */

   /* Get DOS date and time */
   _strdate( dos_date );
   _strtime( dos_time );

   printf("\n Enter the test number: ");
   scanf("%d", &test_number);

   get_file_name(); /* Gets file name and checks to see if it exists */

   printf("\n Enter the desired total punch displacement (mm): ");
   scanf("%d", &total_punch Disp);

   get_code: printf("\n Enter the geometry code (1=conical cup, 2=square pan): ");
   scanf("%d", &geometry_code);

   switch( geometry_code ) {
      case 1:
         printf("\n Enter the punch diameter (mm): ");
         scanf("%f", &punch_dia);
         break;
      case 2:
         printf("\n Enter the radii code (1=equal radii, 2=unequal radii): ");
         scanf("%d", &radii_code);
         break;
      default:
         printf("\n Unrecognized geometry code.");
         goto get_code;
         break;
   }
}

/* End of switch */
printf("\n Enter the blank diameter (mm): ");
scanf("%f", &blank_dia);

printf("\n Enter the desired constant binder force (N): ");
scanf("%ld", &const_binder_force);

printf("\n Enter the material code (1=2008-T4, 2=5754-HO, 3=6111-T4, 4=Duralcan): ");
scanf("%d", &material_code);

printf("\n Enter the sheet metal thickness (mm): ");
scanf("%f", &sheet_thickness);

printf("\n Enter the lubrication code (1=STP mix, 2=dry): ");
scanf("%d", &lubrication_code);

printf("\n Enter the input pressure (MPa): ");
scanf("%f", &input_press);

} /* End of setup */

void get_file_name( void )
/* Gets filename string */
{
    char num[3];    /* Test number converted to a string */
    char ans;       /* Answer to replace file */
    FILE *outfile;  /* Output file pointer */

    name:; /* Line referred to in goto statement, does not do anything */  .
strcpy( file_name, "c:\press.dat\" );

    /* Copy date string to file_name */
    file_name[13] = dos_date[0];
    file_name[14] = dos_date[1];
    file_name[15] = dos_date[3];
    file_name[16] = dos_date[4];
    file_name[17] = dos_date[6];
    file_name[18] = dos_date[7];

    /* Convert test_number to a string and append to file_name */
    _itoa( test_number, num, 10 );
    strcat( file_name, num );

    /* Append suffix to file_name */
    strcat( file_name, ".dat" );

    /* Check to see if file_name already exists */

    outfile = fopen( file_name, "r" ); /* Opens output file for reading */
    fclose( outfile );
    /* Check to see if there was an error opening file_name for reading */
    if( outfile != NULL ) {
        /* Clear keyboard buffer */
        check_buffer1: if( _kbhit() ) _getch();
        if( _kbhit() ) goto check_buffer1;

        /* If outfile does not equal null then file_name exists */
        printf("\n File %s already exists.\n", file_name );
        printf("\n Overwrite (y/n): ");
        scanf("%c", &ans );
        if( (ans == 'y') || (ans == 'Y') ) goto endif;
        else (endif}
printf("\n Enter a new test number: ");
scanf("%d", test_number);
goto name;
} /* End of nested if */
} /* End of if */

/* Clear keyboard buffer */
check_buffer2: if(_kbhit()) _getch();
if(_kbhit()) goto check_buffer2;

) /* End of get_file_name */

void zero_force( void )
/* Get zero voltages for binder force and punch force */
{

unsigned short n,i; /* Counter */
unsigned short data[ REPS ]; /* Analog data */
unsigned long sum; /* Analog sum */
unsigned short analog_value; /* Analog average */

sum = 0;
lp_setup_adc( TIMING_SOURCE, BINDER_FORCE_CHANNEL, BINDER_FORCE_CHANNEL, GAIN);
for (i=0; i<REPS; i++) {
    lp_adc_series( REPS, data);
    for (n=0; n<REPS; n++) sum += data[n];
} /* end of for loop */
analog_value = sum / ( REPS_F * REPS_F );
lp_analog_to_volts( analog_value, GAIN, &binder_force_zero );

sum = 0;
lp_setup_adc( TIMING_SOURCE, PUNCH_FORCE_CHANNEL, PUNCH_FORCE_CHANNEL, GAIN);
for (i=0; i<REPS; i++) {
    lp_adc_series( REPS, data);
    for (n=0; n<REPS; n++) sum += data[n];
} /* end of for loop */
analog_value = sum / ( REPS_F * REPS_F );
lp_analog_to_volts( analog_value, GAIN, &punch_force_zero );
}

} /* End of zero_force */

void initialize_punch( void )
/* Move punch to initial position */
{
    lp_dac_value( PUNCH_SERVO_CHANNEL, PUNCH_SERVO_ZERO );
}

) /* End of initialize_punch */

void initialize_binder( void )
/* Set binder force to target force */
{
    float volts; /* Binder servo output voltage */
    volts = const_binder_force / BINDER_FORCE_SCALE + binder_force_zero;
lp_generate_volts( BINDER_SERVO_CHANNEL, volts );
}

) /* End of initialize_binder */
void wait( void )
/* Wait for keyboard input */
{
    /* Clear keyboard buffer */
    check_buffer3: if( __kbhit() ) _getch();
    if( __kbhit() ) goto check_buffer3;

    printf("\n\nPrepare blank and center on the binder."
            "\nPress <Enter> to continue: ");
    getch();

    printf("\n\nMake sure that the punch position DCDT is touching"
            "the top of the blank.\nPress <Enter> to continue: ");
    getch();

    printf("\n\nTurn the pump on."
            "\nPress <Enter> to continue: ");
    getch();

    printf("\n\nSet the punch servo to computer and turn it on."n            "\nPress <Enter> to continue: ");
    getch();

    printf("\n\nSet the binder servo to computer and turn it on."n            "\nPress <Enter> to continue: ");
    getch();

    printf("\n\nWait for binder cylinder to close."n            "\nPress <Enter> to continue: ");
    getch();

    printf("\n\nSet the draw-in measurement DCDT."n            "\nPress <Enter> to continue: ");
    getch();

    printf("\n\nSet the binder displacement DCDT."n            "\nPress <Enter> to continue: ");
    getch();

    printf("\n\nTurn the pump controller on and set to desired input pressure."
            "\nPress <Enter> to continue: ");
    getch();

    /* Call routine to stabilize the binder force to the desired constant
    binder force */
    printf("\n\nStabilizing binder force. Please wait...\nStabilize_binder();

    printf("\n\nPress <Enter> to begin experiment: ");
    getch();

    /* Clear keyboard buffer */
    check_buffer4: if( __kbhit() ) _getch();
    if( __kbhit() ) goto check_buffer4;
}
/* End of wait */

void stabilize_binder( void )
/* Set binder force to target force */
{
    short n;    /* counter variable */
    float sum;  /* sum of binder force data */
float binder_force; /* current binder force */
float error; /* error in binder force */
float volts; /* binder servo output voltage */
float delta_volts; /* change in binder servo output voltage */

volts = const_binder_force / BINDER_FORCE_SCALE + binder_force_zero;
do {
    sum = 0.0;
    for(n=0; n<10; n++)
        sum += force( BINDER_FORCE_CHANNEL, BINDER_FORCE_SCALE, 
                        binder_force_zero );
    binder_force = sum / 10.0;
    error = const_binder_force - binder_force;
    delta_volts = error / BINDER_FORCE_SCALE;
    volts += delta_volts;
    lp_generate_volts( BINDER_SERVO_CHANNEL, volts );
} /* End of do loop */
while( fabs(error) > 50 );

}/* End of stabilize_binder */

void zero_dcdt( void )
/* Get analog zero for draw-in and punch dcdt's */
{
    unsigned short n,i;  /* Counter */
    unsigned short data[ REPS ];  /* Analog data */
    unsigned long sum;  /* Analog sum */

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, DRAW_IN_CHANNEL, DRAW_IN_CHANNEL, GAIN);
    for (i=0; i<REPS; i++) {
        lp_adc_series( REPS, data);
        for (n=0; n<REPS; n++) sum += data[n];
    } /* end of for loop */
    draw_in_zero = sum / ( REPS_F * REPS_F );

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, PUNCH_DISP_CHANNEL, PUNCH_DISP_CHANNEL, GAIN);
    for (i=0; i<REPS; i++) {
        lp_adc_series( REPS, data);
        for (n=0; n<REPS; n++) sum += data[n];
    } /* end of for loop */
    punch_disp_zero = sum / ( REPS_F * REPS_F );

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, BINDER_DISP_CHANNEL, BINDER_DISP_CHANNEL, GAIN);
    for (i=0; i<REPS; i++) {
        lp_adc_series( REPS, data);
        for (n=0; n<REPS; n++) sum += data[n];
    } /* end of for loop */
    binder_disp_zero = sum / ( REPS_F * REPS_F );
}

} /* End of zero_dcdt */

void draw_graph( void )
/* Draw graph on screen */
{
    unsigned short x, y;  /* Screen pixel variables */
    char string[30], s[10];  /* Temporary string storage */
/* Initialize screen */
_clearscreen( _GCLEARSCREEN );
_setvideomode(_MAXRESMODE);
_setbgcolor(_BLUE);
_registerfonts( "C:\C700\LIB\*.*.FON" );

/* Print graph title */
_setfont( "t' Tms Rmn'h25w15b" );
moveto( 131, 15 );
_outgtext( "Constant Binder Force Experiment" );

/* Print axis titles */
_setfont( "t' Tms Rmn'h15w10b" );

/* X axis */
moveto( 254, 455 );
_outgtext( "Punch Displacement (mm) " );
_setgtextvector( 0, 1 ); /* Set text to vertical output */

/* Y axis */
moveto( 80, 350 );
_outgtext( "Punch and Binder Force (KN)" );

/* Secondary Y axis */
moveto( 615, 363 );
_outgtext( "Draw-in and Binder Disp. (mm)" );
_setgtextvector( 1, 0 ); /* Set text to horizontal output */

/* Print axis scales */
_setfont( "t' Tms Rmn'h10w8b" );

/* X axis */
moveto( 102, 450 );
_outgtext( "0" );
moveto( 600, 450 );
_itoa( MAX_PUNCH_DISP, string, 10 ); /* Converts int to a string */
_outgtext( string );

/* Primary Y axis */
moveto( 94, 440 );
_outgtext( "0" );
moveto( 68, 34 );
_outgtext( "PF" );
moveto( 84, 34 );
_gcvt( MAX_PUNCH_FORCE, 4, string ); /* Converts float to a string */
_outgtext( string );
moveto( 68, 46 );
_outgtext( "BF" );
moveto( 84, 46 );
_gcvt( MAX_BINDER_FORCE, 4, string ); /* Converts float to a string */
_outgtext( string );

/* Secondary Y axis */
moveto( 611, 40 );
_outgtext( "0" );
moveto( 611, 440 );
_itoa( MAX_DRAW_IN, string, 10 ); /* Converts int to a string */
_outgtext( string );
/* Print axis and tick marks */

_rectangle( _GBORDER, XZERO, YZERO, (XZERO+XPIXELS), (YZERO-YPIXELS) );

/* X axis */
for( x=XZERO; x<(XZERO+XPIXELS); x+=(XPIXELS/10) ) {
    _moveto( x, YZERO );
    _lineto( x, (YZERO-YPIXELS) );
} /* End of for loop */

/* top */
for( x=XZERO; x<(XZERO+XPIXELS); x+=(XPIXELS/10) ) {
    _moveto( x, (YZERO-YPIXELS) );
    _lineto( x, (YZERO-YPIXELS+5) );
} /* End of for loop */

/* Primary Y axis */
for( y=(YZERO-YPIXELS); y<YZERO; y+=(YPIXELS/10) ) {
    _moveto( XZERO, y );
    _lineto( XZERO+5, y );
} /* End of for loop */

/* Secondary Y axis */
for( y=(YZERO-YPIXELS); y<YZERO; y+=(YPIXELS/10) ) {
    _moveto( XZERO+XPIXELS, y );
    _lineto( XZERO+XPIXELS-5, y );
} /* End of for loop */

/* Print MIT reference */
_setfont( "t'Tms Rmn'h10w8b" );
_moveto( 588, 465 );
_outgtext( "MIT/LMP" );

/* Print summary information */
_setfont( "t'Tms Rmn'h10w8b" );
x = 5;
y = 185;
_moveto( x, y );
_outgtext( "Summary" );

y += 15;
_moveto( x, y );
_outgtext( "Date:" );
y += 10;
_moveto( x, y );
_outgtext( dos_date );

y += 15;
_moveto( x, y );
_outgtext( "Time:" );
y += 10;
_moveto( x, y );
_outgtext( dos_time );

y += 15;
_moveto( x, y );
_outgtext( "Geometry:" );
y += 10;
_moveto( x, y );
switch( geometry_code ) {
case 1:
    _outgtext( "Conical Cup" );
    y += 15;
    _moveto( x, y );
    _outgtext( "Punch Dia:" );
    y += 10;
    _moveto( x, y );
    _gcvt( punch_dia, 6, string ); /* Converts float to a string */
    strcat( string, " mm" );
    _outgtext( string );
    break;

case 2:
    _outgtext( "Square Pan" );
    y += 15;
    _moveto( x, y );
    _outgtext( "Radii:" );
    y += 10;
    _moveto( x, y );
    switch( radii_code ) {
    case 1:
        _outgtext( "Equal" );
        break;
    case 2:
        _outgtext( "Unequal" );
        break;
    } /* End of switch( radii_code ) */
    break;
} /* End of switch( geometry_code ) */

y += 15;
_moveto( x, y );
_outgtext( "Blank Dia:" );
y += 10;
_moveto( x, y );
_gcvt( blank_dia, 6, string ); /* Converts float to a string */
strcat( string, " mm" );
_outgtext( string );

y += 15;
_moveto( x, y );
_outgtext( "Material:" );
y += 10;
_moveto( x, y );
switch( material_code ) {
    case 1:
        _outgtext( "2008-T4" );
        break;
    case 2:
        _outgtext( "5754-H0" );
        break;
    case 3:
        _outgtext( '6111-T4' );
        break;
    case 4:
        _outgtext( "Duralcan" );
        break;
    default:
        _outgtext( "Unknown" );
        break;
} /* End of switch */

y += 15;
_moveto( x, y );
_outgtext( "Thickness:" );
y += 10;
__moveto( x, y );
__gcvt( sheet_thickness, 6, string ); /* Converts float to a string */
strcat( string, " mm" );
__outgtext( string );

y += 15;
__moveto( x, y );
__outgtext( "Lubrication:" );
y += 10;
__moveto( x, y );
switch( lubrication_code ) {
  case 1:
    __outgtext( "STP mix" );
    break;
  case 2:
    __outgtext( "Dry" );
    break;
  default:
    __outgtext( "Unknown" );
    break;
} /* End of switch */

y += 15;
__moveto( x, y );
__outgtext( "Input Press:" );
y += 10;
__moveto( x, y );
__gcvt( input_press, 6, string ); /* Converts float to a string */
strcat( string, " MPa" );
__outgtext( string );

y += 15;
__moveto( x, y );
__outgtext( "Ave punch vel:" );
y += 10;

y += 15;
__moveto( x, y );
__outgtext( "Data File:" );
y += 10;
__moveto( x, y );
__outgtext( file_name );
} /* End of draw_graph */

int controller( void )
/* Run controller */
{
  int n;
  /* Counter variable */
  float output_punch_disp; /* Punch disp output value */
  int punch_servo_value; /* Punch servo output value */
  clock_t start; /* Experiment timing variables */
  clock_t lstart; /* Loop timing variable */
  unsigned short x; /* Punch displacement pixel */
  unsigned short y1; /* Punch force pixel */
  unsigned short y2; /* Binder force pixel */
  unsigned short y3; /* Draw-in pixel */
  unsigned short y4; /* Binder displacement pixel */
  char ch; /* Keystroke monitor */
  char string[20]; /* Temporary text string */

  /* Initialize variables */
n = -1;
output_punch_disp = - PUNCH_DISP_INC;
start = clock(); /* Get starting time */

do {
    lstart = clock(); /* Get loop starting time */

    /* Check for keyboard input and if s then stop experiment */
    if( _kbhit() ) { /* Check for a keystroke */
        ch = _getch(); /* Get the keystroke */
        ch = toupper( ch ); /* Convert the keystroke to uppercase */
        if( ch == 'S' ) break; /* If keystroke is 's' then exit do loop */
    }

    /* End of if block */

    /* Increment the counter */
    n++;

    /* Increment the punch displacement output value */
    output_punch_disp += PUNCH_DISP_INC;
    punch_servo_value = output_punch_disp * PUNCH_DISP_SCALE + PUNCH_SERVO_ZERO;
    if( punch_servo_value > 4095 ) punch_servo_value = 4095;

    /* Output values to punch servo and binder servo */
    lp_dac_value( PUNCH_SERVO_CHANNEL, punch_servo_value );

    /* Get data */
    t[n] = (double)( clock() - start ) /CLOCKS_PER_SEC;
    punch_force[n] = force( PUNCH_FORCE_CHANNEL, PUNCH_FORCE_SCALE, punch_force_zero );
    binder_force[n] = - force( BINDER_FORCE_CHANNEL, BINDER_FORCE_SCALE, binder_force_zero );
    punch_disp[n] = position( PUNCH_DISP_CHANNEL, PUNCH_DISP_SCALE, punch_disp_zero );
    binder_disp[n] = - position( BINDER_DISP_CHANNEL, BINDER_DISP_SCALE, binder_disp_zero );
    draw_in[n] = position( DRAW_IN_CHANNEL, DRAW_IN_SCALE, draw_in_zero );

    /* Calculate screen pixel locations */
    X = XZERO + (punch_disp[n]/MAX_PUNCH_DISP) * XPIXELS;
    y1 = YZERO - (punch_force[n]/(MAX_PUNCH_FORCE*1000)) * YPIXELS;
    y2 = YZERO - (binder_force[n]/(MAX_BINDER_FORCE*1000)) * YPIXELS;
    y3 = YZERO - YPIXELS + (draw_in[n]/MAX_DRAW_IN) * YPIXELS;
    y4 = YZERO - YPIXELS + (binder_disp[n]/MAX_DRAW_IN) * YPIXELS;

    /* Plot data */
    _setpixel( x, y1 );
    _setpixel( x, y2 );
    _setpixel( x, y3 );
    _setpixel( x, y4 );

    /* Filter out large punch disp. readings */
    if( punch_disp[n] > 75 ) punch_disp[n] = 0.0;

    /* Check for cup failure or punch over travel */
    /* if( punch_force[n] > punch_force[n-1] + 500 ) break; */
    if( punch_disp[n] > MAX_PUNCH_DISP ) break;

    while( ((double)( clock() - lstart ) /CLOCKS_PER_SEC) < LOOP_TIME )
    {
    }
} while( punch_disp[n] < total_punch_disp );
/* Send retract command to punch servo */
lp_dac_value( PUNCH_SERVO_CHANNEL, 0 );

/* Print average punch velocity */
_setcolor( 15 );
_setfnt( "t'Tms Rmn'h10w8b" );
_moveto( 5, 435 );
gcvt( ave_punch_vel, 4, string ); /* Converts float to a string */
_strcat( string, " mm/sec" );
_outtext( string );

getch(); /* Wait for keyboard input */
_setvideomode( _DEFAULTMODE ); /* Set video mode back to default mode */
_clearscreen( _GCEARSSCREEN ); /* Clears the screen */
_displaycursor( _GCURSORON ); /* Turns cursor display to on */

return n;
) /* End of controller */

void save_data( int steps /* Number of data points */ )
/* Save experimental data to disk */
{
    int n; /* Counter */
    FILE *outfile; /* File pointer */

    printf( "\n Saving data to disk ..." );

    outfile = fopen( file_name, ‘w’ ); /* Opens output file for writing */

    /* Print summary information */
    fprintf( outfile, "\nConstant Binder Force Test" );
    fprintf( outfile, "\n\nData File:\t\t%", file_name );
    fprintf( outfile, "\n\nSummary" );
    fprintf( outfile, "\n---------------------------------------" );

    fprintf( outfile, "\n\nDate:\t\t%", dos_date );
    fprintf( outfile, "\nTime:\t\t%", dos_time );
    fprintf( outfile, "\nTotal travel (mm):\t%\-5.0d”, total_punch_disp );
    fprintf( outfile, "\nGeometry:" );
    switch( geometry_code ) {
        case 1:
            fprintf( outfile, "\nConical Cup" );
            fprintf( outfile, "\nPunch Dia (mm):\t%\-5.2f", punch_dia );
            break;
        case 2:
            fprintf( outfile, "\nSquare Pan" );
            fprintf( outfile, "\nPunch Radii:" );
            switch( radii_code ) {
            case 1:
                fprintf( outfile, "\nEqual" );
                
Page 83
break;
case 2:
    fprintf( outfile, "\t\tUnequal\" );
break;
} /* End of switch */
break;
default:
    fprintf( outfile, "\t\tUnknown\" );
break;
} /* End of switch */

fprintf( outfile, "\nBlank Dia (mm): \t\%5.2f\", blank_dia );

fprintf( outfile, "\nMaterial: \" );
switch( material_code ) {
    case 1:
        fprintf( outfile, "\t\t2008-T4\" );
break;
    case 2:
        fprintf( outfile, "\t\t5754-H0\" );
break;
    case 3:
        fprintf( outfile, "\t\t6111-T4\" );
break;
    case 4:
        fprintf( outfile, "\t\tDuralcan\" );
break;
default:
    fprintf( outfile, "\t\tUnknown\" );
break;
} /* End of switch */

fprintf( outfile, "\nThickness (mm): \t\%5.2f\", sheet_thickness );

fprintf( outfile, "\nLubrication: \" );
switch( lubrication_code ) {
    case 1:
        fprintf( outfile, "\t\tSTP mix\" );
break;
    case 2:
        fprintf( outfile, "\t\tDry\" );
break;
default:
    fprintf( outfile, "\t\tUnknown\" );
break;
} /* End of switch */

fprintf( outfile, "\nInput Press (MPa): \t\%5.1f\", input_press );

fprintf( outfile, "\nAve Punch Vel (mm/s): \t\%5.2f\", ave_punch_vel );

/* Print data */

fprintf( outfile, "\n\nData\" );
fprintf( outfile, "\n-----------------------------------------\" );

fprintf( outfile, "\nTime\tPunch\tPunch\tBinder\tBinder\tDraw-in\" );
fprintf( outfile, "\n\nTime\tPunch\tPunch\tBinder\tBinder\tDraw-in\" );
fprintf( outfile, "\n(n/sec)\t(mm)\t(N)\t(mm)\t(N)\t(mm)\" );
fprintf( outfile, "\n\n\" );
for( n=0; n<steps; n++ )
    fprintf( outfile, \"%5.3f\t%5.3f\t%5.0f\t%5.4f\t%5.0f\t%5.3f\n\",
        t[n], punch_disp[n], punch_force[n], binder_disp[n], binder
_force[n], draw_in[n] );

fclose( outfile ); /* Closes output file */

printf( "\n Done.\n" );

) /* End of save_data */

float force( int channel, float scale, float zero )
/* function to read A/D channel and calculate the corresponding force */
{
    unsigned short n; /* Counter */
    unsigned short data[REPS]; /* Analog data */
    unsigned long sum; /* Analog sum */
    unsigned short analog_value; /* Analog average */
    float volts; /* Channel voltage */
    float force; /* Calculated force */

    sum = 0;
lp_setup_adc( TIMING_SOURCE, channel, channel, GAIN);
lp_adc_series(REPS, data);
for (n=0; n<REPS; n++) sum += data[n];
analog_value = sum / REPS_F;
lp_analog_to_volts(analog_value, GAIN, &volts);
force = (volts - zero) * scale;
return force;
}

) /* End of force */

float position( int channel, float scale, int zero )
/* function to read A/D channel and calculate the corresponding position */
{
    unsigned short n; /* Counter */
    unsigned short data[REPS]; /* Analog data */
    unsigned long sum; /* Analog sum */
    float position; /* Calculated position */

    sum = 0;
lp_setup_adc( TIMING_SOURCE, channel, channel, GAIN);
lp_adc_series(REPS, data);
for (n=0; n<REPS; n++) sum += data[n];
position = (zero - sum/REPS_F) / (scale * GAIN);
return position;
}

) /* End of position */

/* End of program */
Program written by: Robert S. Bakkestuen

Controller program for conducting closed-loop punch force experiments using the conical cup hydraulic press. The program displays a real time graph showing the punch force, binder force, draw-in, and binder disp. plotted vs. punch disp. After the experiment has run to completion or the loop has been stopped by a keyboard input, the program saves the data in a file with a name in the form mnddyyzn.dat where: mm is the month (01-12), dd is day of the month (01-31), yy is the year (00-99), and nn is the experiment number for that day (1-99). The program code is divided into several small blocks and makes use of many constant definitions to make editing the program easier. To change a program parameter, all that is required is to change the definition statement and recompile the program to update the code throughout. For documentation purposes, please note all changes made to the program in the revision section below.

<table>
<thead>
<tr>
<th>Rev</th>
<th>Name</th>
<th>Date</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R. Bakkestuen</td>
<td>5/10/94</td>
<td>created from the program fb_cnst.c changed punch force trajectory COEFF's and reordered control statements</td>
</tr>
<tr>
<td>2</td>
<td>R. Bakkestuen</td>
<td>5/11/94</td>
<td></td>
</tr>
</tbody>
</table>

/include <dos.h>  /* include dos functions file */
/include <string.h>  /* include string functions file */
/include <time.h>  /* include time functions file */
/include <graph.h>  /* include graphics definitions file */
/include <math.h>  /* include math functions file */
/include <stdio.h>  /* include standard I/O file */
/include <stdlib.h>  /* include standard library file */
/include "lpclerrs.h"  /* include LPCLAB error codes */
/include "lpcldefs.h"  /* include LPCLAB function declarations */

/* Define A/D input channels on the DT2811 */
#define BINDER_FORCE_CHANNEL 7
#define PUNCH_FORCE_CHANNEL 1
#define PUNCH_DISP_CHANNEL 2
#define BINDER_DISP_CHANNEL 3
#define DRAW_IN_CHANNEL 4

/* Define D/A output channels on the DT2811 */
#define BINDER_SERVO_CHANNEL 0
#define PUNCH_SERVO_CHANNEL 1

/* DT2811 constants */
#define GAIN 1  /* Gain used when reading values */
#define FREQUENCY 6000  /* Clock frequency for timing */
#define TIMING_SOURCE 0  /* Internal clock and trigger */

/* Data acquisition constants */
#define REPS 10  /* Number of times to read each channel when averaging */
#define REPS_F 10.0  /* Float value of REPS used for calc */
/* Calibration constants */
define BINDER_FORCE_SCALE 50000. /* N/volt */
define PUNCH_FORCE_SCALE 27205. /* N/volt */
define BINDER_DISP_SCALE 880.28 /* Analog/mm */
define PUNCH_DISP_SCALE 31.104 /* Analog/mm */
define DRAW_IN_SCALE 27.252 /* Analog/mm */
define PUNCH_SERVO_ZERO 1560 /* Analog value that moves punch up until blank is in contact with to
p of binder */

/* Controller constants */
define PUNCH_DISP_INC 0.2 /* Analog punch movement step size */
define LOOP_TIME 0.2 /* Time for each loop during forming cycle (seconds) */

/* Punch force trajectory polynomial coefficients */
define COEFF3 -2.0376 /* Zero order term */
define COEFF2 94.7977 /* First order term */
define COEFF1 1684.76 /* Second order term */
define COEFF0 -520.42 /* Third order term */

/* Plotting constants */
define XZERO 105 /* X zero coordinate */
define YZERO 445 /* Primary Y zero coordinate */
define XPIXELS 500 /* Number of X pixels in graph */
define YPIXELS 400 /* Number of Y pixels in graph */
define MAX_TIME 10 /* Max time (sec) */
define MAX_PUNCH_DISP 50 /* Max punch displacement (mm) */
define MAX_PUNCH_FORCE 100. /* Max punch force (KN) */
define MAX_BINDER_FORCE 250. /* Max binder force (KN) */
define MAX_DRAW_IN 10 /* Max draw-in (mm) */
define ARRAY_SIZE 1500 /* Size of data arrays */

/* Declare global program variables */
char dos_date[15];
char dos_time[15];
int test_number;
float kp;
float ki;
float controller_on_height;
int total_punch_disp;
int geometry_code;
float punch_dia;
int radii_code;
float blank_dia;
int material_code;
float sheet_thickness;
int lubrication_code;
float input_press;
char file_name[30];
long initial_binder_force;
float binder_force_zero;
float binder_servo_zero;
int binder_disp_zero;
float punch_force_zero;
int punch_disp_zero;
int draw_in_zero;
float ave_punch_vel;

/* Arrays to store experimental values before they are written to disk */
float t[ ARRAY_SIZE ];
float punch_disp[ ARRAY_SIZE ];
float punch_force[ ARRAY_SIZE ];
float binder_disp[ ARRAY_SIZE ];
float binder_force[ ARRAY_SIZE ];
float draw_in[ ARRAY_SIZE ];
float error[ ARRAY_SIZE ];
float output_binder_force[ ARRAY_SIZE ];

/* Declare function types */

void setup( void );
void get_file_name( void );
void zero_force( void );
void initialize_punch( void );
void initialize_binder( void );
void wait( void );
void stabilize_binder( void );
void zero_dcdt( void );
void draw_graph( void );
int controller( void );
void save_data( int );
float force( int, float, float);
float position( int, float, int);

void main( void )
/* Program execution block */
{
    int num_steps;

    setup();          /* Setup experiment */
    zero_force();     /* Zero punch and binder force */
    initialize_punch(); /* Move punch to initial position */
    initialize_binder(); /* Set binder force to initial value */
    wait();          /* Wait for keyboard input */
    zero_dcdt();     /* Zero draw-in and punch dcdt's */
    draw_graph();    /* Draw graph on screen */
    num_steps = controller(); /* Run controller */
save_data( num_steps ); /* Save experimental data to disk */
lp_terminate(); /* Terminate LPCLAB operations */
exit( 0 );
}

/* End of main */

void setup( void )
/* Function for setting up experimental variables */
{

    /* Setup screen for output */
    _setbcolor( _BLUE );
    _clearscreen( _GCLEARSCREEN );
    _settextcolor( 15 );

    /* Setup DT2811 board */
    lp_initialize(); /* Initialize the LPCLAB subroutine */
    
    lp_select_board( 1 ); /* Select board 1 */
    lp_set_clock_frequency( FREQUENCY ); /* Sets clock frequency for data acquisition */

    /* Get DOS date and time */
    _strdate( dos_date );
    _strtime( dos_time );

Page 88
printf("\n Enter the test number: " );
scanf("%d", &test_number);

get_file_name(); /* Gets filename and checks to see if it exists */

printf("\n Enter the controller proportional gain (N/N): ");
scanf("%f", &kp);

printf("\n Enter the controller integral gain (N/N): ");
scanf("%f", &ki);

printf("\n Enter the controller start height (mm): ");
scanf("%f", &controller_on_height);

printf("\n Enter the desired total punch displacement (mm): ");
scanf("%d", &total_punch_disp);

get_code: printf("\n Enter the geometry code (1=conical cup, 2=square pan): ");
scanf("%d", &geometry_code);

switch( geometry_code ) {
    case 1:
        printf("\n Enter the punch diameter (mm): ");
        scanf("%f", &punch_dia);
        break;
    case 2:
        printf("\n Enter the radii code (1=equal radii, 2=unequal radii): ");
        scanf("%d", &radii_code);
        break;
    default:
        printf("\n Unrecognized geometry code.");
        goto get_code;
        break;
} /* End of switch */

printf("\n Enter the blank diameter (mm): ");
scanf("%f", &blank_dia);

printf("\n Enter the desired initial binder force (N): ");
scanf("%ld", &initial_binder_force);

printf("\n Enter the material code (1=2008-T4, 2=5754-H0, 3=6111-T4, 4=uralcan): ");
scanf("%d", &material_code);

printf("\n Enter the sheet metal thickness (mm): ");
scanf("%f", &sheet_thickness);

printf("\n Enter the lubrication code (1=STP mix, 2=dry): ");
scanf("%d", &lubrication_code);

printf("\n Enter the input pressure (MPa): ");
scanf("%f", &input_press);

} /* End of setup */

void get_file_name( void )
/* Gets filename string */
{
    char num[3]; /* Test number converted to a string */
    char ans; /* Answer to replace file */
    
    printf("Enter the test number: ");
    scanf("%d", &num);
    switch(num) {
        case 4:
            printf("Enter the controller proportional gain (N/N): ");
            scanf("%f", &kp);
            break;
        default:
            printf("Unrecognized geometry code.");
            goto get_file_name;
            break;
    } /* End of switch */
    
    printf("Enter the controller integral gain (N/N): ");
    scanf("%f", &ki);
    
    printf("Enter the controller start height (mm): ");
    scanf("%f", &controller_on_height);
    
    printf("Enter the desired total punch displacement (mm): ");
    scanf("%d", &total_punch_disp);
    
    printf("Enter the geometry code (1=conical cup, 2=square pan): ");
    scanf("%d", &geometry_code);
    
    switch( geometry_code ) {
        case 1:
            printf("Enter the punch diameter (mm): ");
            scanf("%f", &punch_dia);
            break;
        case 2:
            printf("Enter the radii code (1=equal radii, 2=unequal radii): ");
            scanf("%d", &radii_code);
            break;
        default:
            printf("Unrecognized geometry code.");
            goto get_file_name;
            break;
    } /* End of switch */
    
    printf("Enter the blank diameter (mm): ");
    scanf("%f", &blank_dia);
    
    printf("Enter the desired initial binder force (N): ");
    scanf("%ld", &initial_binder_force);
    
    printf("Enter the material code (1=2008-T4, 2=5754-H0, 3=6111-T4, 4=uralcan): ");
    scanf("%d", &material_code);
    
    printf("Enter the sheet metal thickness (mm): ");
    scanf("%f", &sheet_thickness);
    
    printf("Enter the lubrication code (1=STP mix, 2=dry): ");
    scanf("%d", &lubrication_code);
    
    printf("Enter the input pressure (MPa): ");
    scanf("%f", &input_press);
    
    } /* End of setup */

    void get_file_name( void )
/* Gets filename string */
{
    char num[3]; /* Test number converted to a string */
    char ans; /* Answer to replace file */
    
    printf("Enter the test number: ");
    scanf("%d", &num);
    switch(num) {
        case 4:
            printf("Enter the controller proportional gain (N/N): ");
            scanf("%f", &kp);
            break;
        default:
            printf("Unrecognized geometry code.");
            goto get_file_name;
            break;
    } /* End of switch */
    
    printf("Enter the controller integral gain (N/N): ");
    scanf("%f", &ki);
    
    printf("Enter the controller start height (mm): ");
    scanf("%f", &controller_on_height);
    
    printf("Enter the desired total punch displacement (mm): ");
    scanf("%d", &total_punch_disp);
    
    printf("Enter the geometry code (1=conical cup, 2=square pan): ");
    scanf("%d", &geometry_code);
    
    switch( geometry_code ) {
        case 1:
            printf("Enter the punch diameter (mm): ");
            scanf("%f", &punch_dia);
            break;
        case 2:
            printf("Enter the radii code (1=equal radii, 2=unequal radii): ");
            scanf("%d", &radii_code);
            break;
        default:
            printf("Unrecognized geometry code.");
            goto get_file_name;
            break;
    } /* End of switch */
    
    printf("Enter the blank diameter (mm): ");
    scanf("%f", &blank_dia);
    
    printf("Enter the desired initial binder force (N): ");
    scanf("%ld", &initial_binder_force);
    
    printf("Enter the material code (1=2008-T4, 2=5754-H0, 3=6111-T4, 4=uralcan): ");
    scanf("%d", &material_code);
    
    printf("Enter the sheet metal thickness (mm): ");
    scanf("%f", &sheet_thickness);
    
    printf("Enter the lubrication code (1=STP mix, 2=dry): ");
    scanf("%d", &lubrication_code);
    
    printf("Enter the input pressure (MPa): ");
    scanf("%f", &input_press);
    
    } /* End of setup */
FILE *outfile; /* Output file pointer */
name; /* Line referred to in goto statement, does not do anything */
strcpy( file_name, "c:\\press.dat\"");

/* Copy date string to file_name */
file_name[13] = dos_date[0];
file_name[14] = dos_date[1];
file_name[15] = dos_date[3];
file_name[16] = dos_date[4];
file_name[17] = dos_date[6];
file_name[18] = dos_date[7];

/* Convert test_number to a string and append to file_name */
_itoa( test_number, num, 10 );
strcat( file_name, num );

/* Append suffix to file_name */
strcat( file_name, ".dat");

/* Check to see if file_name already exists */
outfile = fopen( file_name, "r" ); /* Opens output file for reading */
fclose( outfile ); /* Check to see if there was an error opening file_name for reading */
if( outfile != NULL ) {
    /* Clear keyboard buffer */
    check_buffer1: if( _kbhit() ) _getch();
    if( _kbhit() ) goto check_buffer1;
    /* If outfile does not equal null then file_name exists */
    printf( "\n File %s already exists.\", file_name );
    printf( "\n Overwrite ( y/n ) : ");
    scanf( "%c", &ans );
    if( (ans == 'y') || (ans == 'Y') ) goto endif;
    else {
        printf( "\n Enter a new test number: ");
        scanf( "%d", test_number );
        goto name;
    }
    /* End of nested if */
endif;
} /* End of if */

/* Clear keyboard buffer */
check_buffer2: if( _kbhit() ) _getch();
if( _kbhit() ) goto check_buffer2;
}

/* End of get_file_name */

void zero_forc( void )
/* Get zero voltages for binder force and punch force */
{
    unsigned short n,i; /* Counter */
    unsigned short data[ REPS ]; /* Analog data */
    unsigned long sum; /* Analog sum */
    unsigned short analog_value; /* Analog average */

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, BINDER_FORCE_CHANNEL, BINDER_FORCE_CHANNEL, GAIN);
    for (i=0; i<REPS; i++) {
        lp_adc_series( REPS, data);
        for (n=0; n<REPS; n++) sum += data[n];
    }
}
analog_value = sum / (REPS_F * REPS_F);
lp_analog_to_volts( analog_value, GAIN, &binder_force_zero);

sum = 0;
lp_setup_adc( TIMING_SOURCE, PUNCH_FORCE_CHANNEL, PUNCH_FORCE_CHANNEL, GAIN);
for (i=0; i<REPS; i++) {
  lp_adc_series( REPS, data);
  for (n=0; n<REPS; n++) sum += data[n];
} /* end of for loop */
analog_value = sum / (REPS_F * REPS_F);
lp_analog_to_volts( analog_value, GAIN, &punch_force_zero);

} /* End of zero_force */

void initialize_punch( void )
/* Move punch to initial position */{
  lp_dac_value( PUNCH_SERVO_CHANNEL, PUNCH_SERVO_ZERO );
} /* End of initialize_punch */

void initialize_binder( void )
/* Set binder force to target force */{
  float volts; /* Binder servo output voltage */
  volts = initial_binder_force / BINDER_FORCE_SCALE;
  lp_generate_volts( BINDER_SERVO_CHANNEL, volts );
} /* End of initialize_binder */

void wait( void )
/* Wait for keyboard input */{
  /* Clear keyboard buffer */
  check_buffer3: if( _kbhit() ) _getch();
  if( _kbhit() ) goto check_buffer3;

  printf("\n\n Prepare blank and center on the binder."
    "\n Press <Enter> to continue: ");
  getch();

  printf("\n\n Make sure that the punch position DCDT is touching 
" the top of the blank.\n Press <Enter> to continue: ");
  getch();

  printf("\n\n Turn the pump on." 
    "\n Press <Enter> to continue: ");
  getch();

  printf("\n\n Set the punch servo to computer and turn it on." 
    "\n Press <Enter> to continue: ");
  getch();

  printf("\n\n Set the binder servo to computer and turn it on." 
    "\n Press <Enter> to continue: ");
  getch();

  printf("\n\n Wait for binder cylinder to close."
}

Page 91
"\n Press <Enter> to continue: ");
getch();

printf("\n\n Set the draw-in measurement DCDT. 
"\n Press <Enter> to continue: ");
getch();

printf("\n\n Set the binder displacement DCDT. 
"\n Press <Enter> to continue: ");
getch();

printf("\n\n Turn the pump controller on and set to desired input pressure. 
"\n Press <Enter> to continue: ");
getch();

/* Call routine to stabilize the binder force to the desired constant 
 binder force */
printf("\n\n Stabilizing binder force. Please wait... ");
stabilize_binder();

printf("\n\n Press <Enter> to begin experiment: ");
getch();

/* Clear keyboard buffer */
check_buffer4: if( _kbhit() ) _getch();
if( _kbhit() ) goto check_buffer4;
} /* End of wait */

void stabilize_binder( void )
/* Set binder force to target force */
{
    short n; /* counter variable */
    float sum; /* sum of binder force data */
    float binder_force; /* current binder force */
    float error; /* error in binder force */
    float volts; /* binder servo output voltage */
    float initial_volts; /* initial binder servo output voltage */
    float delta_volts; /* change in binder servo output voltage */

    initial_volts = initial_binder_force / BINDER_FORCE_SCALE;
    volts = initial_volts;

    do {
        sum = 0.0;
        for(n=0; n<10; n++)
            sum += - force( BINDER_FORCE_CHANNEL, BINDER_FORCE_SCALE, 
                            binder_force_zero );
        binder_force = sum / 10.0;
        error = initial_binder_force - binder_force;
        delta_volts = error / BINDER_FORCE_SCALE;
        volts += delta_volts;
        lp_generate_volts( BINDER_SERVO_CHANNEL, volts );
    } /* End of do loop */
    while( fabs(error) > 50 );

    binder_servo_zero = volts - initial_volts;

} /* End of stabilize_binder */
/* Get analog zero for draw-in and punch dcdt's */
{
    unsigned short n,i; /* Counter */
    unsigned short data[REPS]; /* Analog data */
    unsigned long sum; /* Analog sum */

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, DRAW_IN_CHANNEL, DRAW_IN_CHANNEL, GAIN);
    for (i=0; i<REPS; i++) {
        lp_adc_series( REPS, data);
        for (n=0; n<REPS; n++) sum += data[n];
    } /* end of for loop */
    draw_in_zero = sum / (REPS_F * REPS_F);

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, PUNCH_DISP_CHANNEL, PUNCH_DISP_CHANNEL, GAIN);
    for (i=0; i<REPS; i++) {
        lp_adc_series( REPS, data);
        for (n=0; n<REPS; n++) sum += data[n];
    } /* end of for loop */
    punch_disp_zero = sum / (REPS_F * REPS_F);

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, BINDER_DISP_CHANNEL, BINDER_DISP_CHANNEL, GAIN);
    for (i=0; i<REPS; i++) {
        lp_adc_series( REPS, data);
        for (n=0; n<REPS; n++) sum += data[n];
    } /* end of for loop */
    binder_disp_zero = sum / (REPS_F * REPS_F);
}
/* End of zero_dcdt */

void draw_graph( void )
/* Draw graph on screen */
{
    unsigned short x, y; /* Screen pixel variables */
    char string[30], s[10]; /* Temporary string storage */

    /* Initialize screen */

    _clearscreen( _GCLEARSCREEN );
    _setvideomode( _MAXRESMODE );
    _setbkcolor( _BLUE );
    _registerfonts( "C:\C700\LIB\*.FON" );

    /* Print graph title */
    _setfont( "t\Tms Rmn\h25w15b" );
    _moveto( 131, 15 );
    _outgtext( "Closed Loop Punch Force Test" );

    /* Print axis titles */
    _setfont( "t\Tms Rmn\h15w10b" );

    /* X axis */
    _moveto( 254, 455 );
    _outgtext( "Punch Displacement (mm) " );
    _setgtextvector( 0, 1 ); /* Set text to vertical output */
/* Primary Y axis */
_moveto( 80, 350 );
_outgtext( "Punch and Binder Force (KN)" );

/* Secondary Y axis */
_moveto( 615, 363 );
_outgtext( "Draw-in and Binder Disp. (mm)" );

_setgextvector( 1, 0 ); /* Set text to horizontal output */

/* Print axis scales */

_setfont( "t'Tms Rmn'h10w8b" );

/* X axis */
_moveto( 102, 450 );
_outgtext( "0" );
_moveto( 600, 450 );
_itoa( MAX_PUNCH_DISP, string, 10 ); /* Converts int to a string */
_outgtext( string );

/* Primary Y axis */
_moveto( 94, 440 );
_outgtext( "0" );
_moveto( 68, 34 );
_outgtext( "FP" );
_moveto( 84, 34 );
gcvt( MAX_PUNCH_FORCE, 4, string ); /* Converts float to a string */
_outgtext( string );
_moveto( 68, 46 );
_outgtext( "BF" );
_moveto( 84, 46 );
gcvt( MAX_BINDER_FORCE, 4, string ); /* Converts float to a string */
_outgtext( string );

/* Secondary Y axis */
_moveto( 611, 40 );
_outgtext( "0" );
_moveto( 611, 440 );
_itoa( MAX_DRAW_IN, string, 10 ); /* Converts int to a string */
_outgtext( string );

/* Print axis and tick marks */

_rectangle( _GBORDER, XZERO, YZERO, (XZERO+XPIXELS), (YZERO-YPIXELS) );

/* X axis */
for( x=XZERO; x<(XZERO+XPIXELS); x+=(XPIXELS/10) ) {
    _moveto( x, YZERO );
    _lineto( x, (YZERO-5) );
} /* End of for loop */

/* top */
for( x=XZERO; x<(XZERO+XPIXELS); x+=(XPIXELS/10) ) {
    _moveto( x, (YZERO-YPIXELS) );
    _lineto( x, (YZERO-YPIXELS+5) );
} /* End of for loop */

/* Primary Y axis */
for( y=(YZERO-YPIXELS); y<YZERO; y+=(YPIXELS/10) ) {
    _moveto( XZERO, y );
    _lineto( XZERO+5, y );
} /* End of for loop */
/* Secondary Y axis */
for (y = (YZERO - YPIXELS); y < YZERO; y += (YPIXELS/10)) {
    _moveto(XZERO + XPIXELS, y);
    _lineto(XZERO + XPIXELS - 5, y);
} /* End of for loop */

/* Print MIT reference */
_setfont("t'Tms Rmn'h10w8b");
_moveto(588, 465);
_outgtext("MIT/LMP");

/* Print summary information */
_setfont("t'Tms Rmn'h10w8b");
x = 5;
y = 110;
_moveto(x, y);
_outgtext("Summary");

y += 15;
_moveto(x, y);
_outgtext("Date:");
y += 10;
_moveto(x, y);
_outgtext(dos_date);

y += 15;
_moveto(x, y);
_outgtext("Time:");
y += 10;
_moveto(x, y);
_outgtext(dos_time);

y += 15;
_moveto(x, y);
_outgtext("Prop. Gain:");
y += 10;
_moveto(x, y);
_gcvt(kp, 3, string); /* Converts float to a string */
strcat(string, "N/N");
_outgtext(string);

y += 15;
_moveto(x, y);
_outgtext("Int. Gain:");
y += 10;
_moveto(x, y);
_gcvt(ki, 3, string); /* Converts float to a string */
strcat(string, "N/N");
_outgtext(string);

y += 15;
_moveto(x, y);
_outgtext("Start Cont.:");

/* Converts float to a string */
strcat(string, "mm");
_outgtext(string);

y += 15;
moveto( x, y );
_outgtext( "Geometry:" );
y += 10;
moveto( x, y );
switch( geometry_code ) {
case 1:
_outgtext( "Conical Cup" );
y += 15;
moveto( x, y );
_outgtext( "Punch Dia:" );
y += 10;
moveto( x, y );
gcvt( punch_dia, 6, string ); /* Converts float to a string */
strcat( string, " mm" );
_outgtext( string );
break;
case 2:
_outgtext( "Square Pan" );
y += 15;
moveto( x, y );
_outgtext( "Radii:" );
y += 10;
moveto( x, y );
switch( radii_code ) {
case 1:
_outgtext( "Equal" );
break;
case 2:
_outgtext( "Unequal" );
break;
} /* End of switch( radii_code ) */
break;
} /* End of switch( geometry_code ) */

y += 15;
moveto( x, y );
_outgtext( "Blank Dia:" );
y += 10;
moveto( x, y );
gcvt( blank_dia, 6, string ); /* Converts float to a string */
strcat( string, " mm" );
_outgtext( string );

y += 15;
moveto( x, y );
_outgtext( "Material:" );
y += 10;
moveto( x, y );
switch( material_code ) {
case 1:
_outgtext( "2008-T4" );
break;
case 2:
_outgtext( "5754-H0" );
break;
case 3:
_outgtext( "6111-T4" );
break;
case 4:
_outgtext( "Duralcan" );
break;
default:
_outgtext( "Unknown" );
break;
} /* End of switch */

y += 15;
_moveto( x, y );
_outgtext( "Thickness:" );
y += 10;
_moveto( x, y );
gcvt( sheet_thickness, 6, string ); /* Converts float to a string */
strcat( string, " mm" );
_outgtext( string );

y += 15;
_moveto( x, y );
_outgtext( "Lubrication:" );
y += 10;
_moveto( x, y );
switch( lubrication_code ) {
    case 1:
        _outgtext( "STP mix" );
        break;
    case 2:
        _outgtext( "Dry" );
        break;
    default:
        _outgtext( "Unknown" );
        break;
} /* End of switch */

y += 15;
_moveto( x, y );
_outgtext( "Input Press:" );
y += 10;
_moveto( x, y );
gcvt( input_press, 6, string ); /* Converts float to a string */
strcat( string, " MPa" );
_outgtext( string );

y += 15;
_moveto( x, y );
_outgtext( "Ave punch vel:" );
y += 10;

y += 15;
_moveto( x, y );
_outgtext( "Data File:" );
y += 10;
_moveto( x, y );
_outgtext( file_name );
}
} /* End of draw_graph */

int controller( void )
/* Run controller */
{
    int n;
    /* Counter variable */
    float output_punch_disp;
    /* Punch disp output value */
    int punch_servo_value;
    /* Punch servo output value */
    float target_punch_force;
    /* Calculated trajectory */
    float binder_volts;
    /* Output voltage to binder servo */
    clock_t start;
    /* Experiment timing variables */
    clock_t lstart;
    /* Loop timing variable */
    unsigned short x;
    /* Punch displacement pixel */
    unsigned short y1;
    /* Punch force pixel */
unsigned short y2;    /* Binder force pixel */
unsigned short y3;    /* Draw-in pixel */
unsigned short y4;    /* Binder displacement pixel */
char ch;              /* Keystroke monitor */
char string[20];      /* Temporary text string */

/* Initialize variables */
n = -1;
output_punch_disp = -PUNCH_DISP_INC;
start = clock();   /* Get starting time */

do {
  lstart = clock();   /* Get loop starting time */

  /* Check for keyboard input and if s then stop experiment */
  if( _kbhit() ) { /* Check for a keystroke */
    ch = _getch();  /* Get the keystroke */
    ch = toupper( ch );  /* Convert the keystroke to uppercase */
    if( ch == 'S' ) break;  /* If keystroke is 's' then exit do loop */
  }  /* End of if block */

  /* Increment the counter */
n ++;

  /* Increment the punch displacement output value */
  output_punch_disp += PUNCH_DISP_INC;
  punch_servo_value = output_punch_disp * PUNCH_DISP_SCALE + PUNCH_SERVO_ZERO;

  if( punch_servo_value > 4095 ) punch_servo_value = 4095;

  /* Output values to punch servo and binder servo */
  lp_dac_value( PUNCH_SERVO_CHANNEL, punch_servo_value );

  /* Get data */
  t[n] = (double)( clock() - start ) / CLOCKS_PER_SEC;
  punch_force[n] = force( PUNCH_FORCE_CHANNEL, PUNCH_FORCE_SCALE, punch_force_zero );
  binder_force[n] = -force( BINDER_FORCE_CHANNEL, BINDER_FORCE_SCALE, binder_force_zero );
  punch_disp[n] = position( PUNCH_DISP_CHANNEL, PUNCH_DISP_SCALE, punch_disp_zero );
  binder_disp[n] = -position( BINDER_DISP_CHANNEL, BINDER_DISP_SCALE, binder_disp_zero );
  draw_in[n] = position( DRAW_IN_CHANNEL, DRAW_IN_SCALE, draw_in_zero );

  /* Execute closed-loop punch force control algorithm */

  /* Calculate punch force error */
  target_punch_force = COEFF3*pow(punch_disp[n],3) + COEFF2*pow(punch_ 
  disp[n],2) + COEFF1*punchDisp[n] + COEFF0;

  /* Determine new binder force setting */
  if (punchDisp[n] < controller_on_height)
    outputBinder_force[n] = initial_binder_force;
  else
    outputBinder_force[n] = outputBinder_force[n-1] + kp*error[n-1] 
    - (kp+k)*error[n];

  /* Output new voltage command to the binder servo */
binder_volts = output_binder_force[n] / BINDER_FORCE_SCALE + binder_servo_zero;
lp_generate_volts( BINDER_SERVO_CHANNEL, binder_volts );

/* Calculate screen pixel locations */
x = XZERO + (punch_disp[n]/MAX_PUNCH_DISP) * XPIXELS;
y1 = YZERO - (punch_force[n]/(MAX_PUNCH_FORCE*1000)) * YPIXELS;
y2 = YZERO - (binder_force[n]/(MAX_BINDER_FORCE*1000)) * YPIXELS;
y3 = YZERO - YPIXELS + (draw_in[n]/MAX_DRAW_IN) * YPIXELS;
y4 = YZERO - YPIXELS + (binder_disp[n]/MAX_DRAW_IN) * YPIXELS;

/* Plot data */
_setpixel( x, y1 );
_setpixel( x, y2 );
_setpixel( x, y3 );
_setpixel( x, y4 );

/* Filter out large punch disp. readings */
if( punch_disp[n] > 75 ) punch_disp[n] = 0.0;

/* Check for cup failure or punch over travel */
/* if( punch_force[n] > punch_force[n-1] + 500 ) break; */
if( punch_disp[n] > MAX_PUNCH_DISP ) break;

while( ((double)( clock() - lstart ) / CLOCKS_PER_SEC) < LOOP_TIME )
{
    while( punch_disp[n] < total_punch_disp )
    {
        /* Send retract command to punch servo */
lp_dac_value( PUNCH_SERVO_CHANNEL, 0 );

        /* Print average punch velocity */

        _setcolor( 15 );
        _setfont( "t’Tms Rmn’h10w8b" );
        _moveto( 5, 435 );
        _gcvt( ave_punch_vel, 4, string ); /* Converts float to a string */
        strcat( string, " mm/sec" );
        _outgtext( string );

        getch(); /* Wait for keyboard input */

        _setvideomode( _DEFAULTMODE ); /* Set video mode back to default mode */
        _clearscreen( _GCLEARSCREEN ); /* Clears the screen */
        _displaycursor( _GCURSORON ); /* Turns cursor display on */

        return n;
    }
}

void save_data( int steps /* Number of data points */ )
/* Save experimental data to disk */
{
    int n; /* Counter */
    FILE *outfile; /* File pointer */

    printf( "\n Saving data to disk ..." );

    outfile = fopen( file_name, "w" ); /* Opens output file for writing */

    /* Print summary information */
fprintf( outfile, "\n\nClosed Loop Punch Force Test" );
fprintf( outfile, "\n\nData File:\t\t%s", file_name );
fprintf( outfile, "\n\nSummary" );
fprintf( outfile, "\n----------------------------------------" );
fprintf( outfile, "\nDate:\t\t%s", dos_date );
fprintf( outfile, "\nTime:\t\t%s", dos_time );
fprintf( outfile, "\nProp. Gain (N/N):\t%5.3f", kp );
fprintf( outfile, "\nInt. Gain (N/N):\t%5.3f", ki );
fprintf( outfile, "\nStart Cont. (mm):\t%5.3f", controller_or_height );
fprintf( outfile, "\nTotal travel (mm):\t%5.0d", total_punch_disp );
fprintf( outfile, "\nGeometry:" );
switch( geometry_code ) {
    case 1:
        fprintf( outfile, "\t\tConical Cup" );
        fprintf( outfile, "\t\nPunch Dia (mm):\t\t%5.2f", punch_dia );
        break;
    case 2:
        fprintf( outfile, "\t\tSquare Pan" );
        fprintf( outfile, "\t\nPunch Radii:" );
        switch( radii_code ) {
            case 1:
                fprintf( outfile, "\t\tEqual" );
                break;
            case 2:
                fprintf( outfile, "\t\tUnequal" );
                break;
        } /* End of switch */
        break;
    default:
        fprintf( outfile, "\t\tUnknown" );
        break;
} /* End of switch */
fprintf( outfile, "\nBlank Dia (mm):\t\t%5.2f", blank_dia );
fprintf( outfile, "\nMaterial:" );
switch( material_code ) {
    case 1:
        fprintf( outfile, "\t\t2008-T4" );
        break;
    case 2:
        fprintf( outfile, "\t\t5754-H0" );
        break;
    case 3:
        fprintf( outfile, "\t\t6111-T4" );
        break;
    case 4:
        fprintf( outfile, "\t\tDuralcan" );
        break;
    default:
        fprintf( outfile, "\t\tUnknown" );
        break;
} /* End of switch */

fprintf( outfile, "\nThickness (mm):\t%5.2f", sheet_thickness );

fprintf( outfile, "\nLubrication:* ");
switch( lubrication_code ) {
  case 1:
    fprintf( outfile, "\t\tSTP mix" );
    break;
  case 2:
    fprintf( outfile, "\t\tDry" );
    break;
  default:
    fprintf( outfile, "\t\tUnknown" );
    break;
} /* End of switch */

fprintf( outfile, "\nInput Press (MPa):\t%5.1f", input_press );

fprintf( outfile, "\nAve Punch Vel (mm/s):\t%5.2f", ave_punch_vel );

;/* Print data */

fprintf( outfile, "\n\nData" );
fprintf( outfile, "\n-------------------------------------------" );

fprintf( outfile, "\nTime\tPunch\tPunch\tBinder\tBinder\tOutput\tDraw-in
\nError" );
fprintf( outfile, "\nDisp\tForce\tDisp\tForce\tBins\nFor" );
fprintf( outfile, "\n(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(\t(" );
for( n=0; n<=steps; n++ )
  fprintf( outfile, "%5.3f\t%5.3f\t%6.0f\t%5.4f\t%6.0f\t%6.0f\t%5.3f\t
%6.0f\n",
    t[n], punch_disp[n], punch_force[n], binder_disp[n], binder
    _force[n], output_binder_force[n], draw_in[n], error[n] );
fclose( outfile ); /* Closes output file */

printf( "\n Done." );
} /* End of save_data */

float force( int channel, float scale, float zero )
/* function to read A/D channel and calculate the corresponding force */{
  unsigned short n; /* Counter */
  unsigned short data[ REPS ]; /* Analog data */
  unsigned long sum; /* Analog sum */
  unsigned short analog_value; /* Analog average */
  float volts; /* Channel voltage */
  float force; /* Calculated force */

  sum = 0;
  lp_setup_adc( TIMING_SOURCE, channel, channel, GAIN);
  lp_adc_series( REPS, data);
  for (n=0; n<REPS; n++) sum += data[n];
  analog_value = sum / REPS_F;
  lp_analog_to_volts( analog_value, GAIN, &volts );
  force = ( volts - zero ) * scale;
  return force;
float position( int channel, float scale, int zero )
/* function to read A/D channel and calculate the corresponding position */
{
    unsigned short n;       /* Counter */
    unsigned short data[REPS]; /* Analog data */
    unsigned long sum;      /* Analog sum */
    float position;         /* Calculated position */

    sum = 0;
    lp_setup_adc( TIMING_SOURCE, channel, channel, GAIN);
    lp_adc_series( REPS, data);
    for (n=0; n<REPS; n++) sum += data[n];
    position = ( zero - sum/REPS_F ) / ( scale * GAIN );
    return position;
}

) /* End of position */

/* End of program */
Bibliography


Moog Series 15 Pressure Control Servovalves, East Aurora, NY, Moog Inc. Catalog 150 1178.

Moog Type 30 Flow Control Servovalves, East Aurora, NY, Moog Inc., Catalog 301 385.


