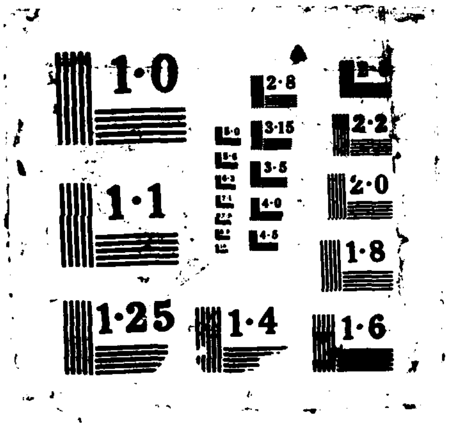
AD-A185 233 THE DYNAMIC CONTROLS OF A HYDRAULIC PRESS BY CONTROLLING THE PUMP MOTOR(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH F EFFRECE JUN 87 1/2

UNCLASSIFIED AFIT/CI/NR-87-56T F/G 13/9 NL


1·0
2·8
2·5
3·2
3·6
4·0
2·2
2·0
1·1
1·8
1·25
1·4
1·6

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

1

DTIC FILE COPY

AD-A185 233

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
AFIT/CI/NR 87-56T

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
The Dynamic Controls Of A Hydraulic Press By Controlling The Pump Motor

5. TYPE OF REPORT & PERIOD COVERED
THESIS/~~DISSERTATION~~

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Frank Effrece, Jr.

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
AFIT STUDENT AT:
Ohio University

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
AFIT/NR
WPAFB OH 45433-6583

12. REPORT DATE
June 1987

13. NUMBER OF PAGES
?

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

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

18. SUPPLEMENTARY NOTES
APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1

Lynn Wolaver
LYNN E. WOLAVER 11 Aug 87
Dean for Research and
Professional Development
AFIT/NR

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

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

87 10 14 209

56

THE DYNAMIC CONTROLS OF A HYDRAULIC PRESS
BY CONTROLLING THE PUMP MOTOR

A Thesis Presented to

The Faculty of the College of Engineering and Technology

Ohio University

In Partial Fulfillment

of the Requirements for the Degree

Master of

Mechanical Engineering

by

Frank Effrece Jr.

June, 1987

This thesis has been approved

for the Department of Mechanical Engineering

and the College of

Engineering and Technology

Kenneth R. Halliday

Associate Professor of Mechanical Engineering

Dean of the College of

Engineering and Technology

TABLE OF CONTENTS

THE DYNAMIC CONTROLS OF A HYDRAULIC PRESS BY CONTROLLING THE PUMP MOTOR

Chapter 1 : Introduction

With the increasing demand for the accurate control of power, recent studies in the area of modern control theory have concentrated on the design, and application of control systems. This desire to control power has produced interest in hydraulics, (Merrit, 1967, p. 1) especially in the field of manufacturing.

There are many unique features of hydraulic systems as compared to electromechanical systems. Hydraulic systems are dynamically stiff when compared to electromechanical drives. This stiffness leads to little drop in the speed of the ram as loads are applied. Using closed-loop systems, this stiffness leads to better position control (Merrit, 1967, p. 2). Hydraulic systems are also superior to mechanical and electrical systems in that the hydraulic fluid of the system carries away the heat generated by internal losses. This heat is a basic limitation of any machine since it causes lubricants to deteriorate, mechanical parts to seize, and insulation to break down as temperature increases. This feature also permits lighter components to be used since they do not have to withstand the heat as well as the stress applied by the system. The hydraulic fluid even acts as a lubricant and makes possible long component life (Merrit, 1967, p.1). Hydraulic systems introduce possibilities that, otherwise, may not exist using electromechanical systems.

However, hydraulics systems do have several significant disadvantages that limit their use. For adequate performance, hydraulic components must have small allowable tolerances resulting in high costs. Hydraulic systems are messy since it is difficult to maintain a system free from leaks. If a leak is large enough, total loss of fluid may occur resulting in a complete loss of power and a great deal of damage as well. This fact is the primary reason for constant inspections of critical hydraulic systems in such applications as aircraft systems. Contamination and dirt within the hydraulic fluid can cause serious problems by clogging valves and actuators resulting in severe loss in performance and/or failure. Hydraulic control systems are also limited in their sophistication because of the complexity of hydraulic control analysis. Hydraulic resistors, for instance, do not obey any simple law such as Ohm's Law. There are endless details to consider with a hydraulic resistor such as the Reynold's number, laminar or turbulent flow, passage geometry, friction factors, and discharge coefficients. Hydraulic systems are not very flexible, nor are they linear, and they are expensive at low power ratings. Hydraulic control systems, on the whole, are attractive control systems for significant power levels and are often the only way to generate large, controllable forces and pressures (Merrit, 1967, p.2-3.)

One manufacturing process particularly dependent on hydraulics is press-forging. The process of press-forging is an important operation for both shaping metals into useful objects and manipulating the desired microstructure of the object. While hammer-forging has been in use since prehistoric times, it was not until 1861 that the first successful hydraulic press, developed in Austria by the Englishman John Haswell, was applied to press-forging (Kobayashi, Herzog, Lapsley, & Thomsen, 1958). Since then press-forgings have become a very integral part of manufacturing such components as wings for aircraft.

Although press-forging has become a successful manufacturing technique, the mechanics of the process are still not completely known (Kobayashi,Herzog, et al. 1958). However, much work has been done to formulate mathematical relationships to analyze the press-forging process. Work done by Prandtl (1923), and later reexamined by Hill (1950), Green (1951), Alexander (1955), and Bishop (1958), analyzed the compression of an ideal plastic solid in plane strain between rough dies and obtained a solution for the stress distribution and average forging pressure.

Further analysis was conducted on press-forging consisting of a flat plate or solid disk which, from the results, was apparently applicable to the forging flash of the press-forging rather than the forging itself by Siebel (1953), Nadai (1939), Sachs (1931), Schroeder and Webster (1949), Hofman and Sachs (1953), and Stone (1953). Though, much work and understanding has come from these investigations, much more still needs to be learned (Kiuchi, 1986).

The process of press-forging requires constant monitoring to produce both accurate shaping and the desired work hardening and grain structure. This suggests computer control for the process. This need for careful process control has prompted this research project which is concerned with developing computer control for a twenty-five ton Walbash Hydraulic Press, model # 25-1212-2TMBACX.

The Walbash Hydraulic Press is a single speed, vertical hydraulic press with heated platens. In the original configuration, the oil is delivered to the cylinder by a constant speed piston pump driven by a 3-phase AC motor. This drive system produces a constant speed press which makes it unacceptable for use in testing procedures since variable speeds are needed to analyze the process of press-forging.

There are two general schemes used to produce variable speed hydraulic systems: servo valve control and pump-motor control. Servo valve control is the most popular scheme of the two. The schematic for a typical servo valve control system is presented in Figure 1. The control system consists of a hydraulic pump, which pumps with constant flow from a reservoir to two pressure compensators, one acting as a relief valve to prevent over loading of the pump and the other which compensates for variations in pressure due to the load. The relief valve, which is normally closed, exhausts into the reservoir. The compensator, which is normally open, is in line with the proportional flow control valve and receives its feedback signal from the outlet of the control valve. The proportional flow control valve meters the flow of fluid to the directional control valve which, in turn, feeds the cylinder. The directional control valve is responsible for directing the flow to either retract or extend the cylinder (Reed & Larman, 1985).

The pump-motor control scheme has many of the same components as the servo valve control scheme but the flow is governed by the output of the variable flow pump. A schematic for this typical pump-motor control system is shown in Figure 2. The flow is varied either through mechanical means within the pump, by controlling the length of its input stroke, or by varying the speed of the pump (Yeaple, 1966, p.131-139). The flow then leaves the pump and enters a relief valve, which exhausts to the reservoir on overload, and a directional control valve. The flow,

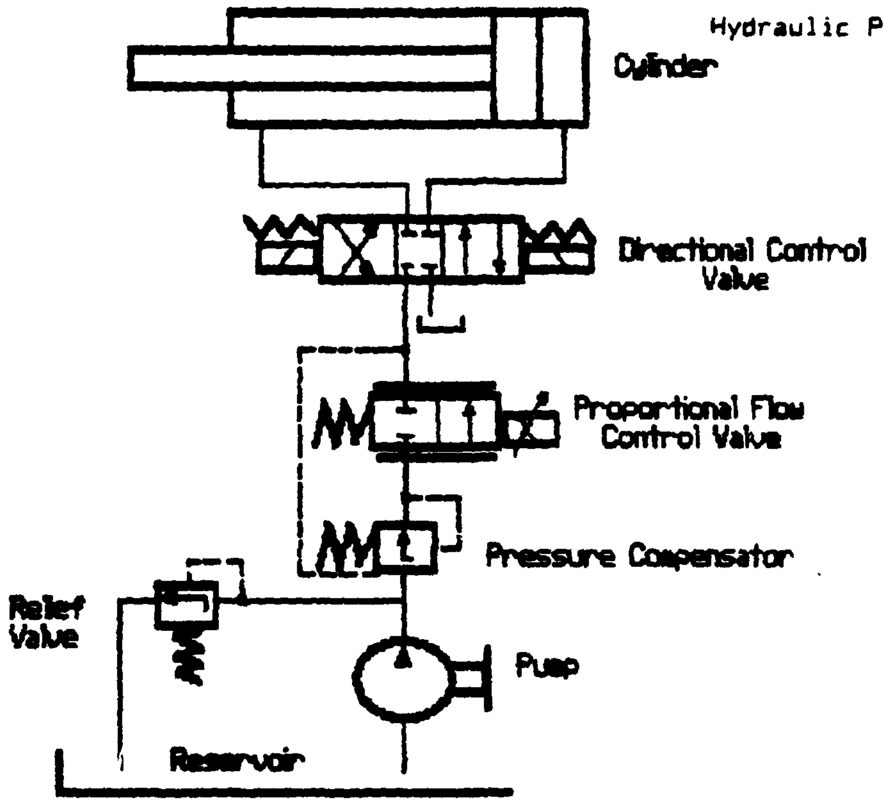


Figure 1: Hydraulic Circuit with servo (proportional flow) valve

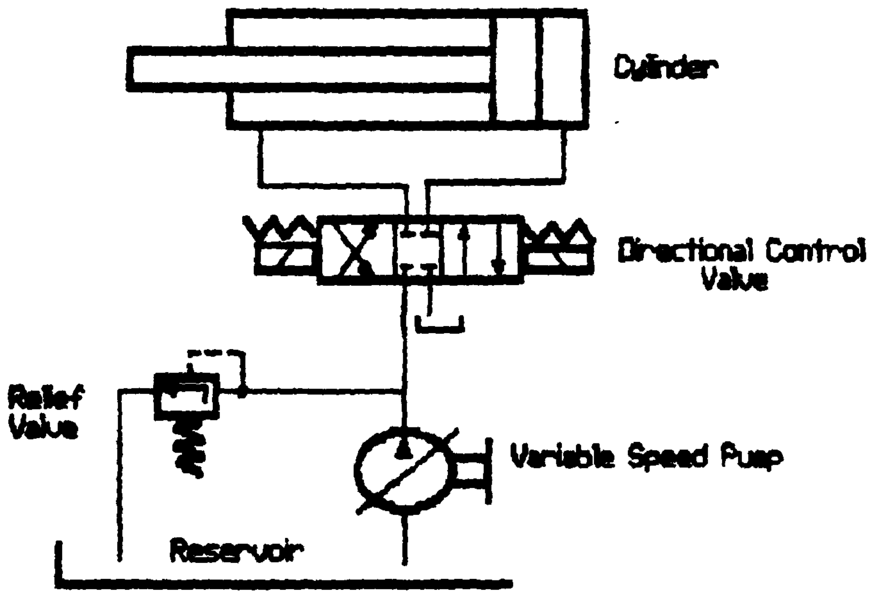


Figure 2: Hydraulic Circuit with variable speed pump

after being directed for either extension or retraction of the ram, enters the cylinder where the work is performed (Read et al. 1985).

The decision as to which system to implement for control of the Walbash press involved many parameters such as the present system's adaptability to the control system, cost, and response time. System adaptability is an extremely important aspect of this design. If the press will not easily accommodate the changes required for this modification then serious problems may occur with the systems performance. If, for example, a servo valve was to have been introduced into the system, the entire system would have to have been renovated; hydraulic hoses would have to have been cut, relief valves would have needed to be installed, and the press would have to be physically expanded since the servo valve would be much too large to be contained within the original unit. Implementing control of the pump-motor, on the other hand, merely meant replacing the existing AC motor with an equivalent DC motor and speed controlier. The original pump for the system could still be used since the pump has the characteristic that, at a given pressure, the pump's delivery rate (CIPM) was directly proportional to the shaft speed (RPM) of the pump over a range of 500 to 4000 rpm, according to the Barne Industrial Piston Pump Performance Data from the Walbash Hydraulic Press Instruction/Service Manual.

The cost of the control system was also a major contributing factor to the selection of the control scheme discussed in this thesis. Computer controllable servo valves are extremely expensive for the pressure valves required within a hydraulic press. This cost is on the order of thousands of dollars. The pump-motor combination is relatively inexpensive. A system that could be employed on the Walbash Press, for example, would cost about seven hundred dollars.

However, the response time of the proposed system is significantly less than that possible with servo valve control. This response time is the primary reason why the servo valve is more common in the control of hydraulic

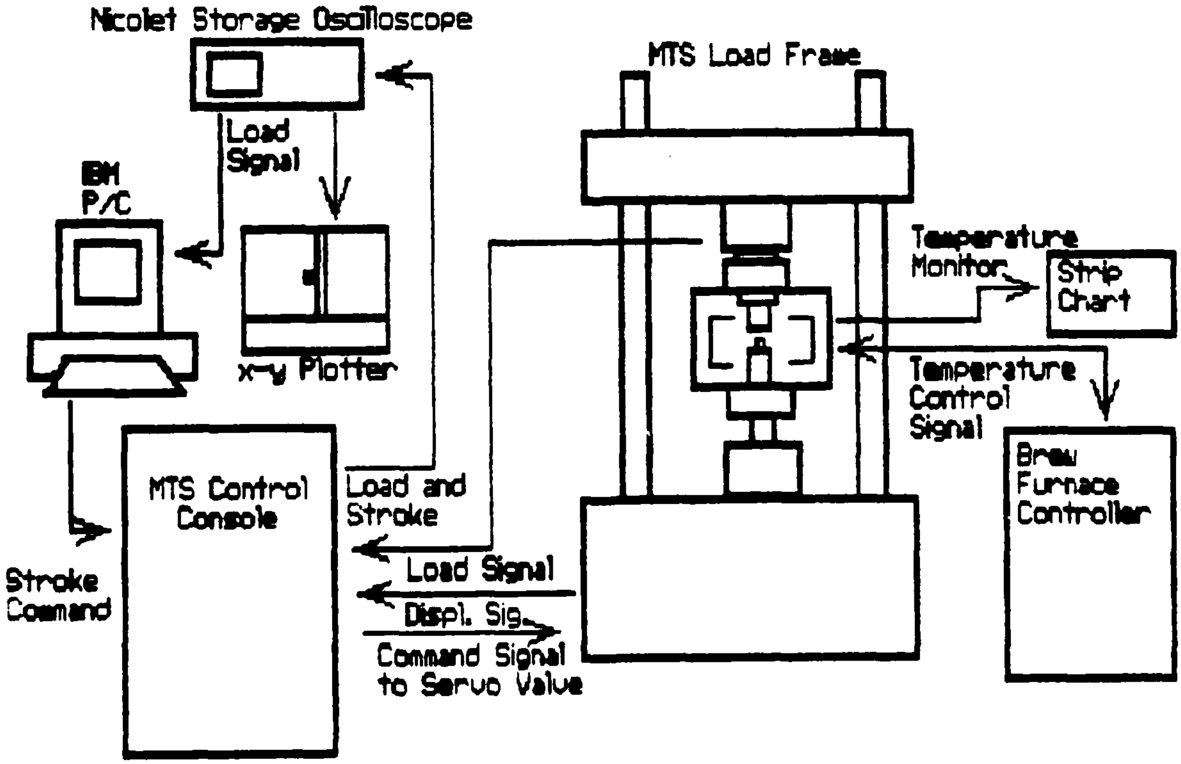


Figure 3: Wright State University Variable Speed Hydraulic Press Block Diagram

systems. A hydraulic press with variable platen speed using servo valve control is in operation at Wright State University. Tests performed with this press showed servo control to be an effective hydraulic control scheme. A block diagram of this system is presented in Figure 3. Also, in a paper on "Pneumatic Controllers" (Berends & Tal's, 1968), a control scheme was presented that used a servo valve and an integral action controller which provided an even more effective control strategy. Both of these systems used large servo valves to control the flow, which made the response inversely proportional to the time constant of the valve (Berends & Tal's, 1968).

However, the hydraulic press at Wright State University took up approximately ten times the space of the Walbash Press and was considerably more expensive. The Wright State Hydraulic Press did have a number of extra components and the heating system but the hydraulic system was still considerably larger than the Walbash Press. Dr. Raghu Srinivasan, a research associate at Wright State, said the hydraulic press with the servo control and without the heating system cost approximately thirty-five thousand dollars.

For all of these reasons it was decided to explore the design of a pump-motor control scheme. Controlling the hydraulic pump's output has not been a popular control scheme because of the system's large time constant. The unnecessary variables into the test. Cost, on the other hand, must also be examined to ultimately present a practical manufacturing process, after testing of the process is complete. Thus, a test apparatus was set up to study press-forgings. Two control schemes were considered, servo valve and pump motor control. After a brief cost-benefit analysis, the pump-motor control system was selected as a possible control system for this process.

An analytical study of the system using block diagrams and numerical techniques will be presented in chapter 2. A suitable system was designed and installed on the Walbash hydraulic press. The performance of this system was measured and the results are presented in chapter 3. A discussion of the results are presented in chapter 4. In addition some final conclusions about the pump-motor control system are also presented.

Chapter 2 : Analytical Analysis

The time-domain response of a pump-motor controlled hydraulic press was studied to predict its typical output and control characteristics. The pump-motor controlled press used the velocity of the ram compared with the proposed velocity of the ram to calculate the speed of the DC motor mounted to the hydraulic pump. A block diagram of the press is presented in figure 4.

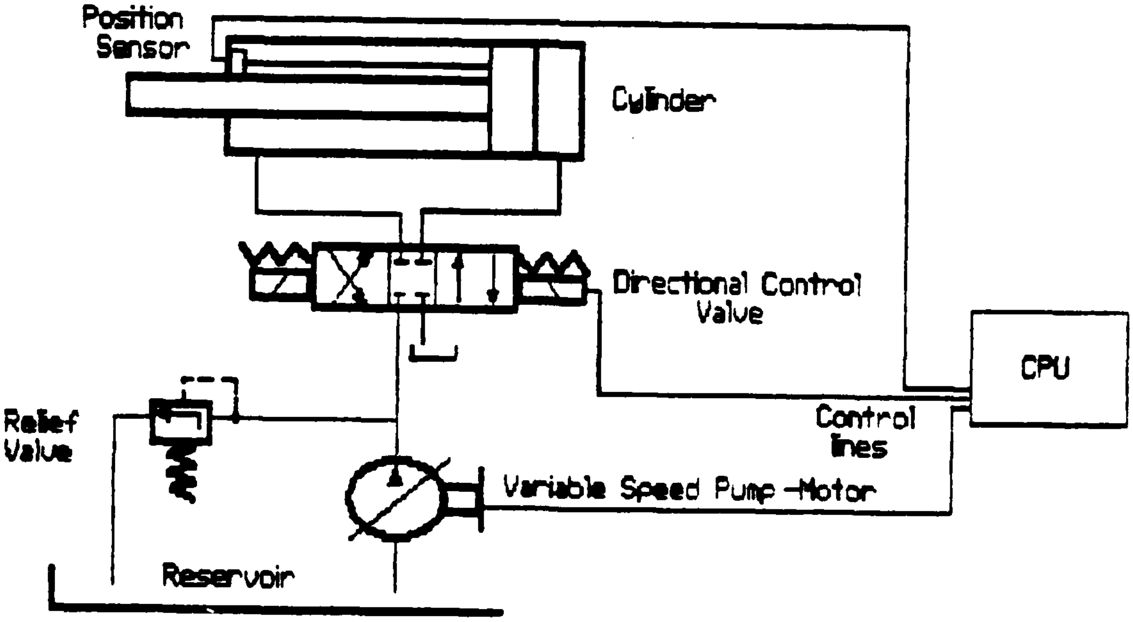


Figure 4: Block Diagram Hydraulic Control System

To analysis this system, the following parameters were identified:

DC Motor Torque	$T_m(t)$
Torque Constant of Motor	K_i =0.554 N-m/A
Inductance of Armature	L_a= Negligible
Resistance of Armature	R_a=1.09 Ohms
Armature Voltage	$e_a(t)$
Armature Current	$I_a(t)$
Back EMF Voltage	$e_b(t)$
Angular Velocity of Motor	$w_m(t)$
Inertia of System	J=1.0 N-m-sec^2
Friction of System	B= Negligible

These parameters were found and calculated in a number of ways. The torque constant of the motor was found using a prony brake setup. The motor was mounted to the prony brake, which measures the stall torque, and voltage was put across the armature of the motor. Measuring the current going into the armature, and using the following formula the stall torque constant can be determined (Kuo, 1987, p.328):

$$T_m(t) = K_i\ I_a(t) \qquad [1]$$

Using equation [1], table 1 was setup:

Stall Torque [N-m]	Amperes [A]	K_i [N-m/A]
0.602	1.0	0.602
0.842	1.6	0.526
1.203	2.2	0.547
1.234	2.3	0.536
1.534	2.8	0.548
1.805	3.4	0.531

Ave Ki = 0.544 [N-m/A]

Table 1 : Torque Constant Calculations

The inductance of the motor was assumed negligible because the effect of the inductance on the system is generally small. Therefore, for simplistic reasons, the inductance of the motor was assumed negligible (Thaler & Wilcox, 1966, p.95-150) (Kuo, 1987, p.328). The resistance of the armature was found using a multimeter across the windings. The back emf constant is exactly the same, dimensionally and numerically, as the torque constant of the motor if SI units are used (Thaler & Wilcox, 1966, p.108). The inertia of the system was found experimentally by comparing the press performance results and the Analytical data. The friction of the system was also a difficult quantity to find. The forces created by friction, however, are negligible when compared to the forces created by the torque of the motor and the force of the press. Thus, the frictional effects were neglected.

With the basic system parameters defined, the equations of the system had to be formulated. The permanent magnet DC motor has four primary equations to describe its performance (Kuo, 1987, p.328).

$$T_m(t) = K_i\ I_a(t) \qquad [1]$$

$$-L_a \frac{d\ I_a(t)}{d\ t} = R_a\ I_a(t) - e_a(t) + e_b(t) \qquad [2]$$

$$e_b(t) = k_b\ w_m(t) \qquad [3]$$

$$J \frac{d\ w_m(t)}{d\ t} = -B\ w_m(t) + T_m(t) \qquad [4]$$

Combining these equations and dropping the negligible terms, the differential equation that describes the permanent magnet DC motor is:

$$J \frac{d\ w_m(t)}{d\ t} + \frac{k_i^2}{R_a}\ w_m(t) = \frac{k_i}{R_a}\ e_a(t) \qquad [5]$$

The motor controller was simulated next. The primary characteristic of the motor controller is its ability to output a voltage to the motor proportional to the input reference voltage; see appendix 5 for the specifics on the motor controller used for this control system. This proportionality constant would also be affected by the amplification of the motor controller reference input within the computer controlling the entire system. Thus, the equation for the motor controller is:

$$e_a(t) = K_p\ e_i(t) \qquad [6]$$

where

K_p = Proportionality Constant of Motor Controller

$e_i(t)$ = Input reference voltage

The hydraulic press is an extremely complicated system compared to the electrical system. Viscous friction of the hydraulic fluid, sliding friction produced by the cylinder and pump, and the fluid compressibility cause serious nonlinearities and inconsistencies (Olson, 1980). However, for the purpose of simplicity, the velocity of the ram is assumed to be proportional to the angular velocity of the motor and the difference between the actual velocity and the velocity predicted using this assumption is the error of the assumption, which will be neglected for simplistic reasons. Thus, the equation of the hydraulic press performance was assumed to be:

$$V_r(t) = K_p\, w_m(t) - E(t) \qquad [7]$$

where

$V_r(t)$ = Velocity of the ram

K_h = Proportionality constant

$E(t)$ = Error caused by inconsistencies and nonlinearities (Assume zero)

The proportionality constant of the hydraulic system was found simply by measuring the velocity of the ram versus the angular velocity of the motor. It was found to be approximately 0.01 [in/rad].

To adequately control the system, feedback must be present. The velocity of the ram is, therefore, fed back to the input using negative feedback. This feedback must be scaled and fine tuned, for stability, by means of a proportionality constant. The equations for the feedback were:

$$e_f(t) = K_1 V_r(t) \qquad [8]$$

$$e_i(t) = e(t) - e_f(t) \qquad [9]$$

where

$e_f(t)$ = feedback signal

K_1 = Scaling-tuning proportionality constant

$e(t)$ = Input signal to the control system

After applying equations [6] through [9] to equation [5], the differential equation which describes the dynamics of the entire hydraulic press control system is:

$$\frac{J}{K_h}\frac{d\,V_r(t)}{d\,t} - \left(\frac{K_i\,K_p\,K_1}{R_a} - \frac{K_i^2}{R_a\,K_h}\right)V_r(t) = \frac{K_i\,K_p}{R_a}\,e(t) \qquad [10]$$

This equation describes the control system but does not take into account the slower time response of hydraulics as opposed to electrical systems since for simplistic reasons we assumed linear characteristics for the hydraulic components. The motor controller is also a time limiting component. Therefore, within the numerical analysis, a time delay was installed to compensate for this time lag. The amount of time delay was computed experimentally by comparing plots of the numerical analysis versus plots of the output from the press.

A control system design package, TUTSIM, was used to simulate the response of this control system and to examine its performance. The output of a simulation is presented in figure 5 which displays the velocity of the ram versus time.

The proportionality constants, K_l and K_p, were computed through comparison of the *TUTSIM* output to the output of the control system installed on the press. Figure 6 shows a typical output from the sensors attached to the actual hydraulic press control system. TUTSIM was run in conjunction with the control system in order to find the optimum values for K_l and K_p. The final values for K_l and K_p were approximately 100.0 and 12.0, respectively.

This control system is greatly simplified. It does not include the inconsistencies or nonlinearities of the hydraulic system, or the nonlinearities of the electrical system. However, these errors should not introduce so large an error into the system that the feedback control system cannot compensate for it.

Figure 5: TUTSIM Control System Simulation Output

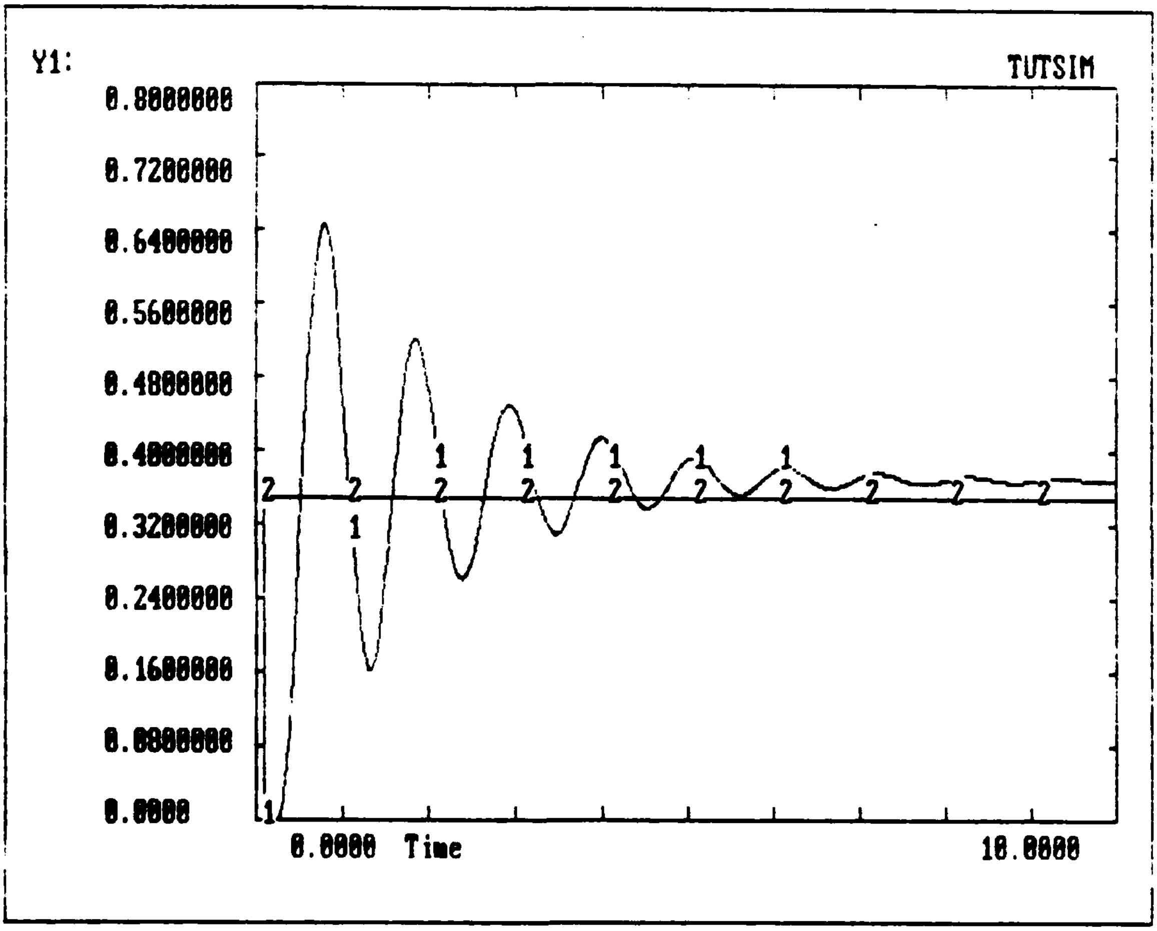
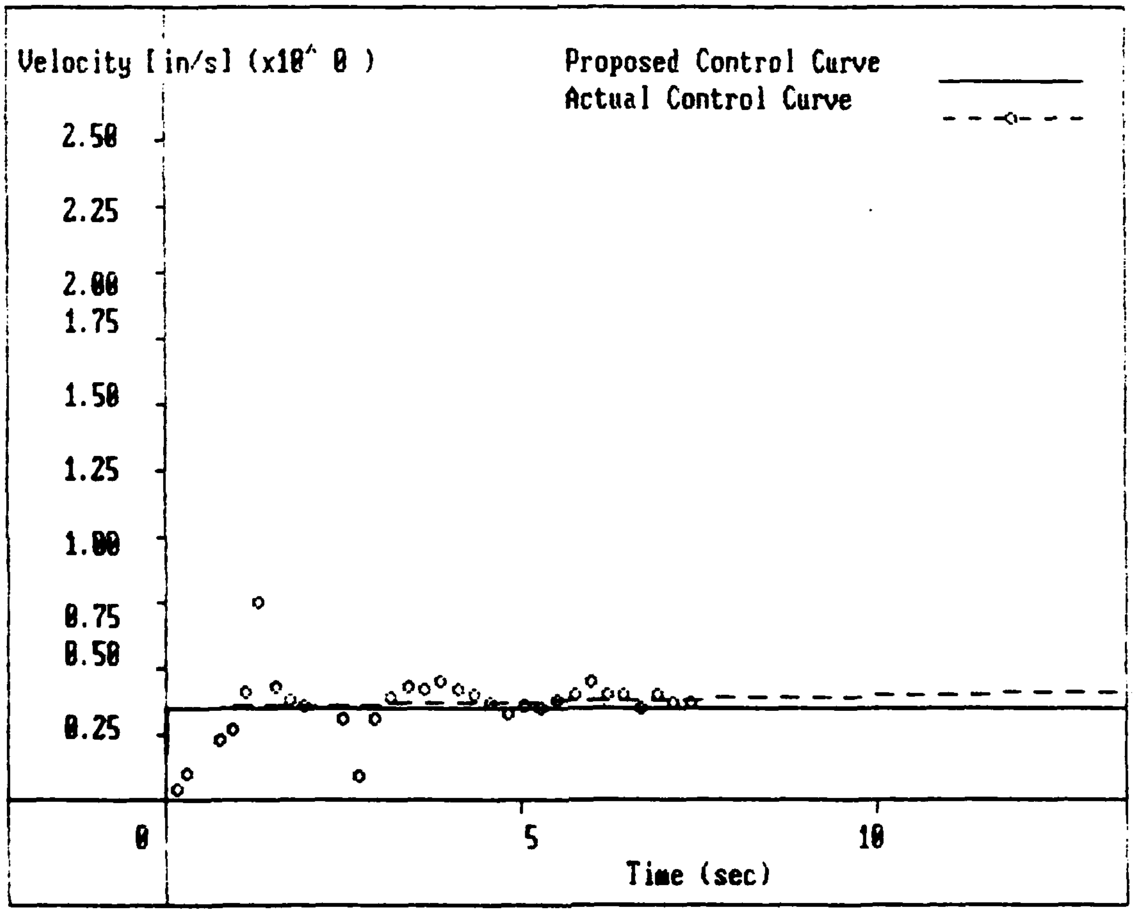


Figure 6: Walbash Hydraulic Press Control Output



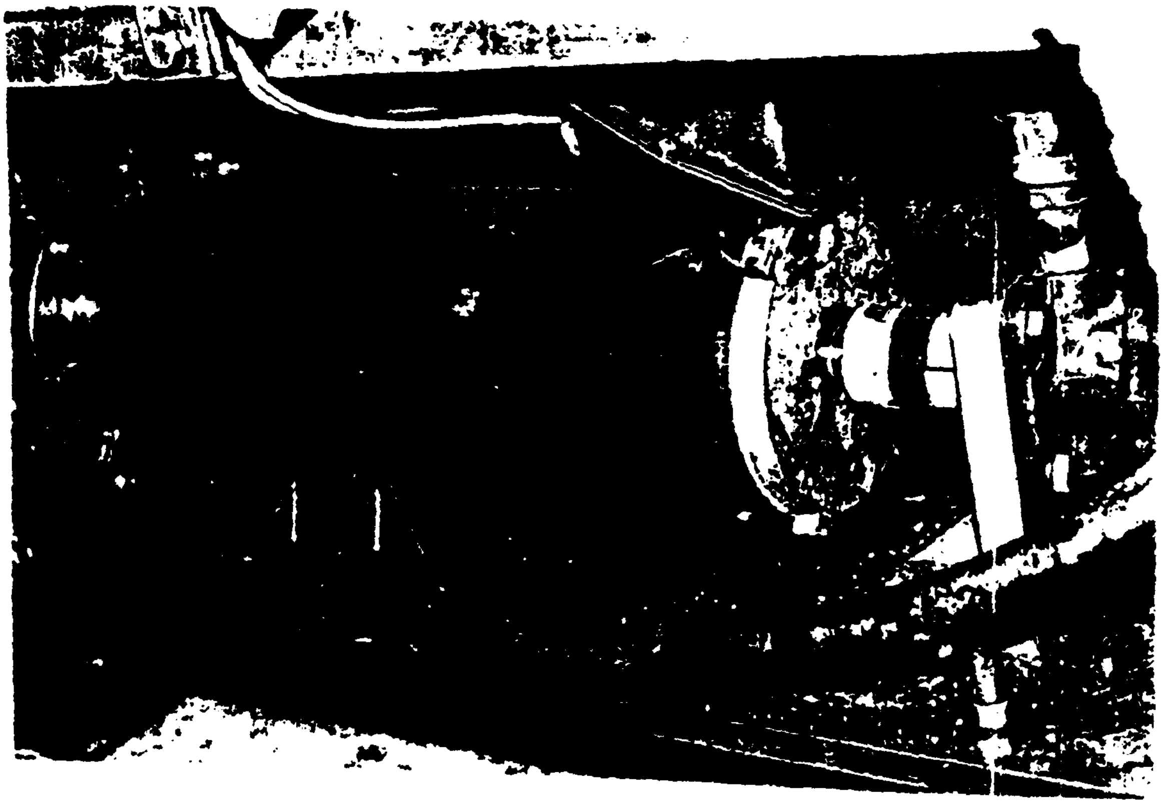
Chapter 3: The Control System

The control system for the Walbash hydraulic press was developed at the same time as the numerical analysis of the system was evaluated. The control system had certain physical constraints that had to be considered within the numerical analysis such as the motor controller proportionality constant, the hydraulic system time constant and any other factors which effect both analyses.

However, before the analysis of the system could be performed, the control system components had to be ordered and set up. The Walbash hydraulic press originally had a one horsepower, 1725 rpm, AC motor driving the hydraulic pump. The control strategy to be employed on this press called for a variable speed pump-motor combination. Since the pump was a piston pump and had the characteristic that the output flow rate was proportional to the input speed and for reasons of control and ease of adaptability to the present press set up, the AC motor was replaced with an equivalent DC motor, a one horsepower, 1725 rpm, Electrol series M-46 permanent magnet DC motor. A picture of the DC motor installed on the press is presented in figure 7. To control this motor, the manufacturer suggested its Electrol D-Trol series 754, single phase, adjustable speed DC motor control with the A-42 option to which allows interfacing to

Figure 7: Electrol Series M-46 Permanent Magnet DC Motor

Installed on the Walbash Press.



a computer. The specifications and various other features of this motor and motor controller are contained in Appendix 6.

To incorporate feedback into the control system to accurately control the speed of the press, the position and velocity of the ram had to be monitored. Thus, a position transducer, a Houston Scientific International series 1850 was selected to complete this task. The postion of the ram was measured while the velocity was calculated by performing a differentiation within the computer. The specifications for this transducer are located in Appendix 5. As purchased, the position sensor was not directly interfaceable with the computer, the signal output was too small (millivolts) for precise results, and there was a considerable amount of electrical noise within the sensor. So, a filter-amplification circuit was designed and built to filter out the noise in the sensor as well as amplify the output voltage for greater precision. A schematic of the circuit is located in Figure 8. The circuit makes use of two operational amplifiers: one to filter the input signal by decreasing the output voltage of the signals as the frequency of the signals increase making it ideal for DC applications, and another to amplify the signal to a voltage the computer can interpret with greater precision.

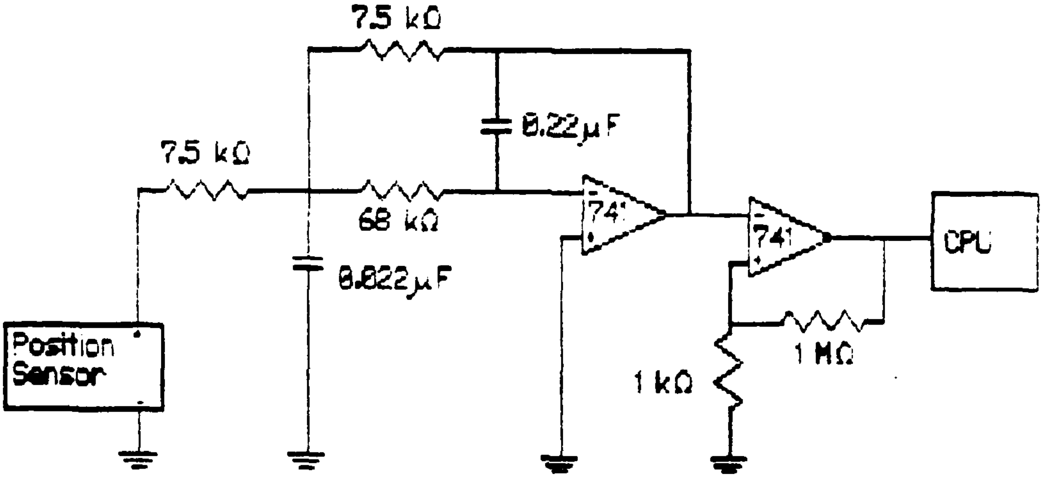


Figure 8: Position Sensor Filter-Amplification Circuit

A Houston Scientific International load cell was also installed for later testing. It also uses the circuit in Figure 8 for filtering and amplification of its output signal.

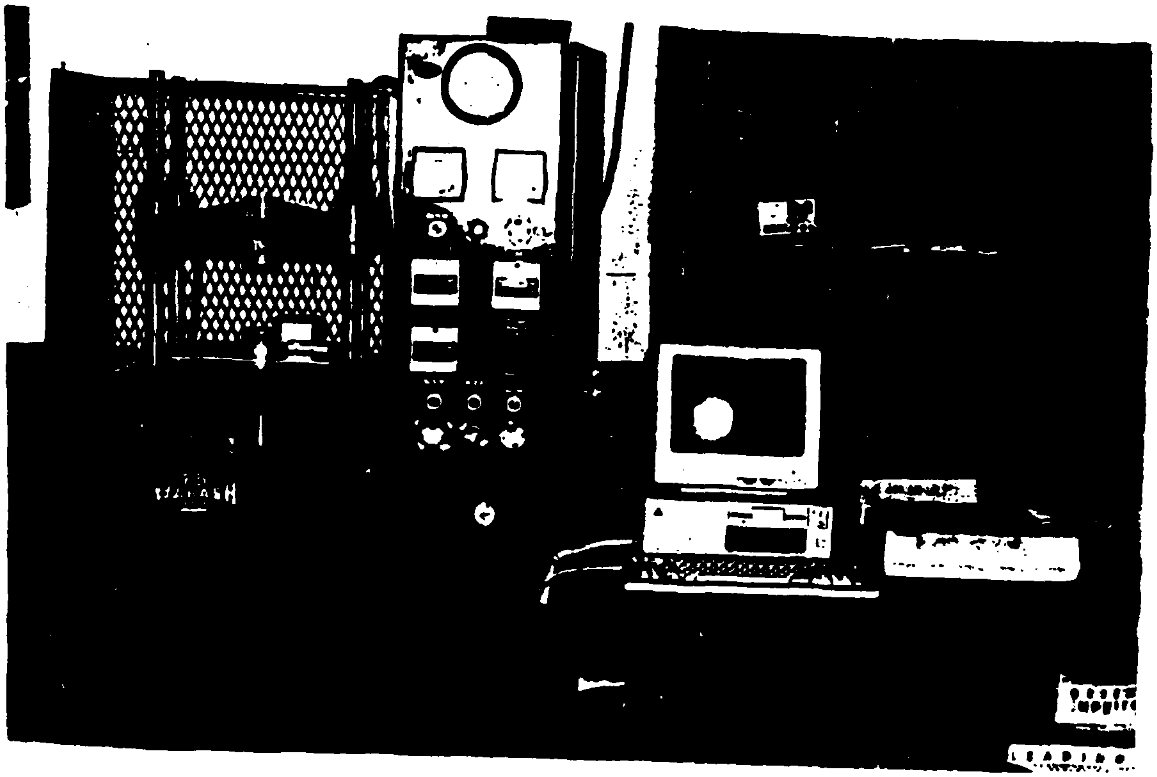
For the computer to communicate with these devices, analog to digital (A/D) and digital to analog (D/A) converters were installed in the computer controlling the process. MetraByte's Dash-8 A/D converter and DAC-02 D/A converter were selected for the conversions. Part of the manual for the Dash-8 is located in appendix 7 which gives instructions for software used to control the A/D converter as well as specifications about the A/D converter. The manual for the DAC-02 D/A converter is presented in

Appendix 8. This manual also gives specifications and programming instructions to control the D/A converter.

The motor controller, D/A converter, A/D converter, and position sensor and filter circuit were connected to an IBM compatible computer for central control of the system. Before the computer could control the process, a number of other programs had to be developed and deployed to find particular control variables required for the press control program. A program named "STEADY.BAS" found the proportionality constant of the no load velocity of the ram as a function of the input voltage to the motor controller. The program listing is located in Appendix 3. This program was used to analyze data from which the steady no load velocity / voltage constant was calculated. The value was found to be approximately 12 [volts-sec/in]. Another program, "CALIBRAT.BAS", was run to find the linear function that converted the signal from the position transducer into the signal required for the linear displacement of the ram using the method of least squares. The program listing and sample output are presented in Appendix 4. From the output, the actual function for the distance between platens (H) versus the input signal from the position transducer read by the A/D converter (AD) was:

$$H\ [in] = 0.01092075 * AD - 1.18604 \qquad [11]$$

Figure 9: Walbash Hydraulic Press Set Up



Both of these programs produce key variables for the central control program of the entire press. A picture of the Walbash hydraulic press with the control system is presented in Figure 9. The source code for the control program has been documented and is located in Appendix 1 along with a sample of the output. Within the program, the constants for calibration and fine tuning are initialized first. The program then asks the user to input the control algorithm, either a constant velocity function, exponentially decaying function, or an arbitrary function from a data file. If a constant velocity or exponentially decaying function is selected then the user inputs the function constants. If an arbitrary function from a data file is chosen, the user is asked to supply the name of the data file. This data file can be created using a data file creator program, "DATAFILE.BAS", documented in Appendix 2. The data is then read from the data file and the control function is set up.

The initialization of the A/D converter is next. The Dash-8 A/D converter comes with a basic subroutine which can be used to control the A/D conversion. This software, however, must be loaded as a binary file. Using this subroutine, the A/D converter is initialized, the channels being used are selected, and the A/D converter is set up for conversion. For more information on this software, consult the software manual, Appendix 7.

The user is then asked to turn the press on by pulling the red start/stop button on the press and is informed that the ram can be moved into position on the billet. By pressing any key the operation can toggle the position of the ram prior to running a test.

The press is now ready to complete its task of compressing the billet at the desired velocity. The test will begin at the press of a key. Anytime during the test, the press can be stopped by pressing any key. The control loop is now executed. The time is read off the computer's internal clock. The position and load are then read by the A/D converter with adjustments added from the calibration variables from the position calibration program. The velocity is then calculated using finite differencing. The ideal velocity is also calculated from the control function and the error in the response of the press is estimated. This error is amplified by a feedback constant and added to the ideal velocity so that if, for example, the error is negative, which means the velocity is too large, the output signal to the motor controller will reflect this error. This adjusted velocity is then output to the motor controller through the D/A converter. For more information on the DAC-02 D/A converter software, consult the manual located in Appendix B. The position of the press is then checked to see if it is less than the minimum platen distance. If it is not and no key has been pressed, the control loop continues and starts over again by reading the computer's internal clock and reading the position and load of the ram. It takes the computer approximately 100 milliseconds to perform one control cycle.

When the control loop eventually ends, the user is told to hit any key to return the ram, and then told to push the start/stop button to turn off the press.

Graphing the results is important and must be accurate for testing of materials. The plots constructed by the system software are the displacement versus time, the velocity versus time, and the load versus time graphs. In each case the boarders of the graph are set up, then the magnitude of each scale are calculated and linear functions are defined to plot the data. The data read by the A/D converter is plotted using small circles to show its experimental origin. A best fit curve is then constructed by means of the least squares technique and then plotted. For the position and the velocity curves, the proposed curve is then plotted to compare the results to that obtained in the experiment. Samples of each graph are located in Appendix 1 along with the documented source code for the central control program (the load cell is presently malfunctioning). After each graph is plotted there is a pause and if a hard copy is desired, the "Prt Sc" key is used. If not, hitting any key clears the screen and the next graph will be plotted. At the conclusion of the program the user is asked if he/she wants to continue. If yes, the entire program is rerun.

This control system makes use of much of the existing hardware of the hydraulic press. The close ram button has been replaced by a relay controlled by the computer and the open ram button has been bypassed by the computer. This allows the computer to control the direction the ram travels. When the relay is closed the ram is closing and will continue in that direction until the relay is opened which will cause a change in direction opening the ram. The start/stop button was not computer controlled in case of emergency and the press had to be shut down and/or stopped. All the other functions, such as the heater controls, have not been disturbed and can still be used on an uncontrolled basis.

The control system is versatile and can, with a few minor adjustments, be converted to any other pump-motor controlled hydraulic system.

Chapter 4: Error Analysis and Conclusion

The control system moved the ram at the specified velocity but not without a number of problems. The calculation of the velocity using numerical differentiation caused serious instability problems. The magnitude of the velocity also effected the performance of the press. And, the imperfections within the press produced errors within the feedback loop.

The calculation of the velocity was a very important part to the control of the system. Initially, the velocity was calculated using an ordinary finite difference technique with delta T equal to the time between retrieving data from the A/D converter.

$$V(t) = \{X(t) - X(t-1)\} / \{T(t) - T(t-1)\} \qquad [12]$$

The results of this control technique are graphically displayed in a plot of the velocity versus time located in Figure 10. As can be seen from the graph, the velocity appeared to be extremely inconsistent. But, if the displacement curve is examined, Figure 11, the plot shows a better correlation with the proposed data. This erratic behavior on the part of the velocity is not all from the press movement, as can be seen, but primarily from the

Figure 10: Ordinary Finite Difference Control Results

(Velocity vs Time)

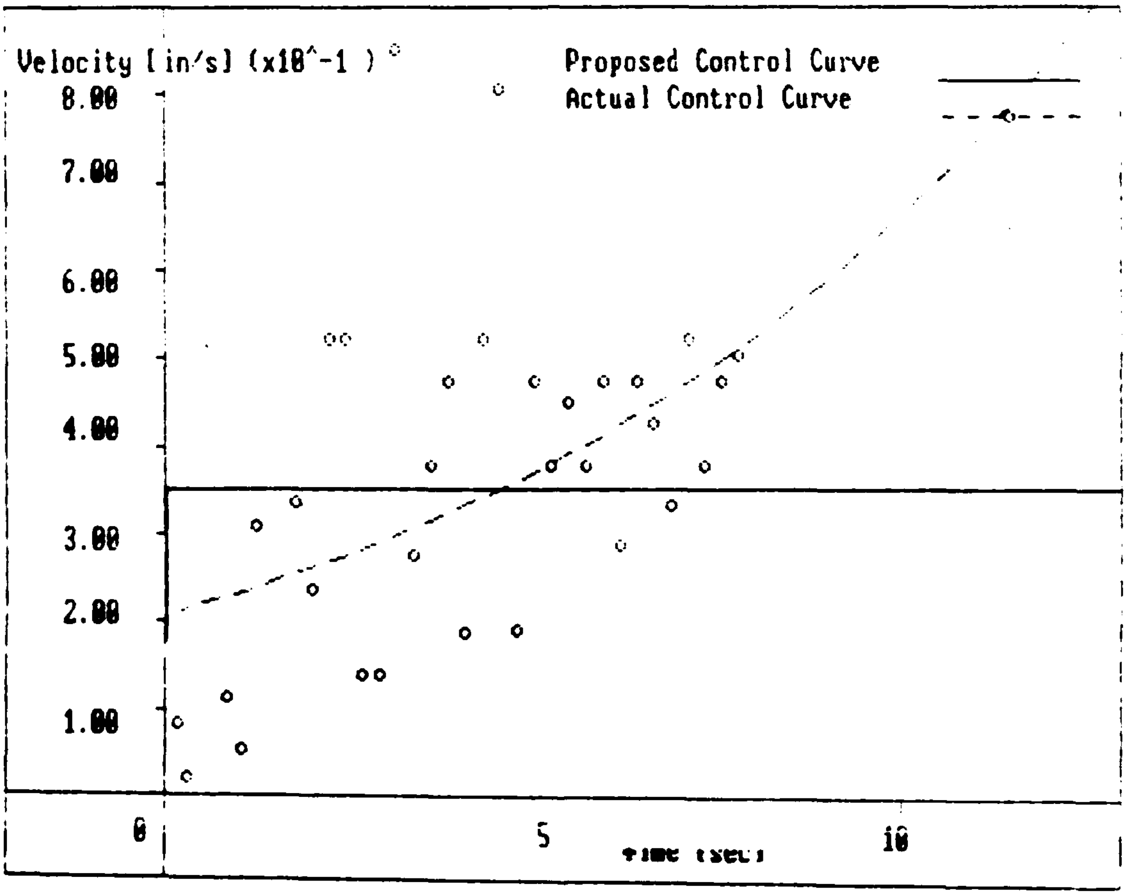


Figure 11: Ordinary Finite Difference Control Results

(Position vs Time)

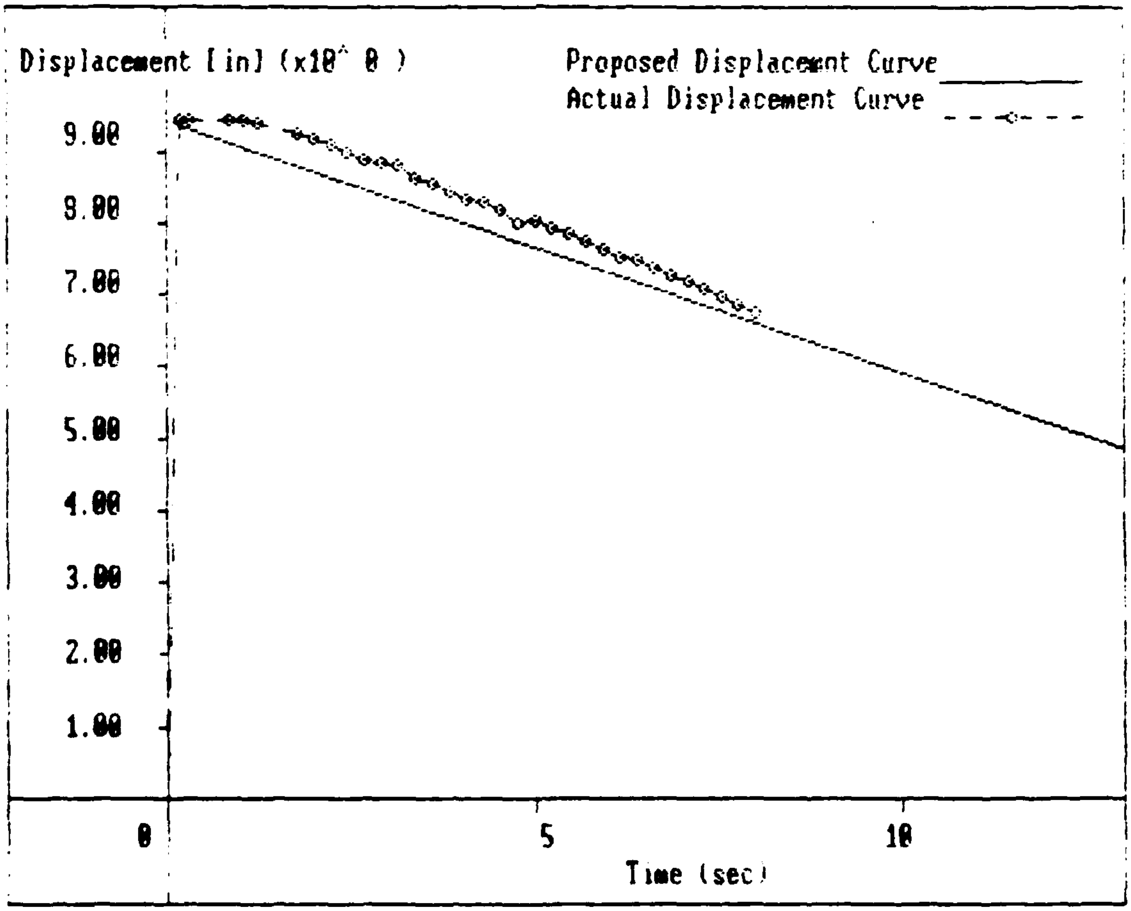
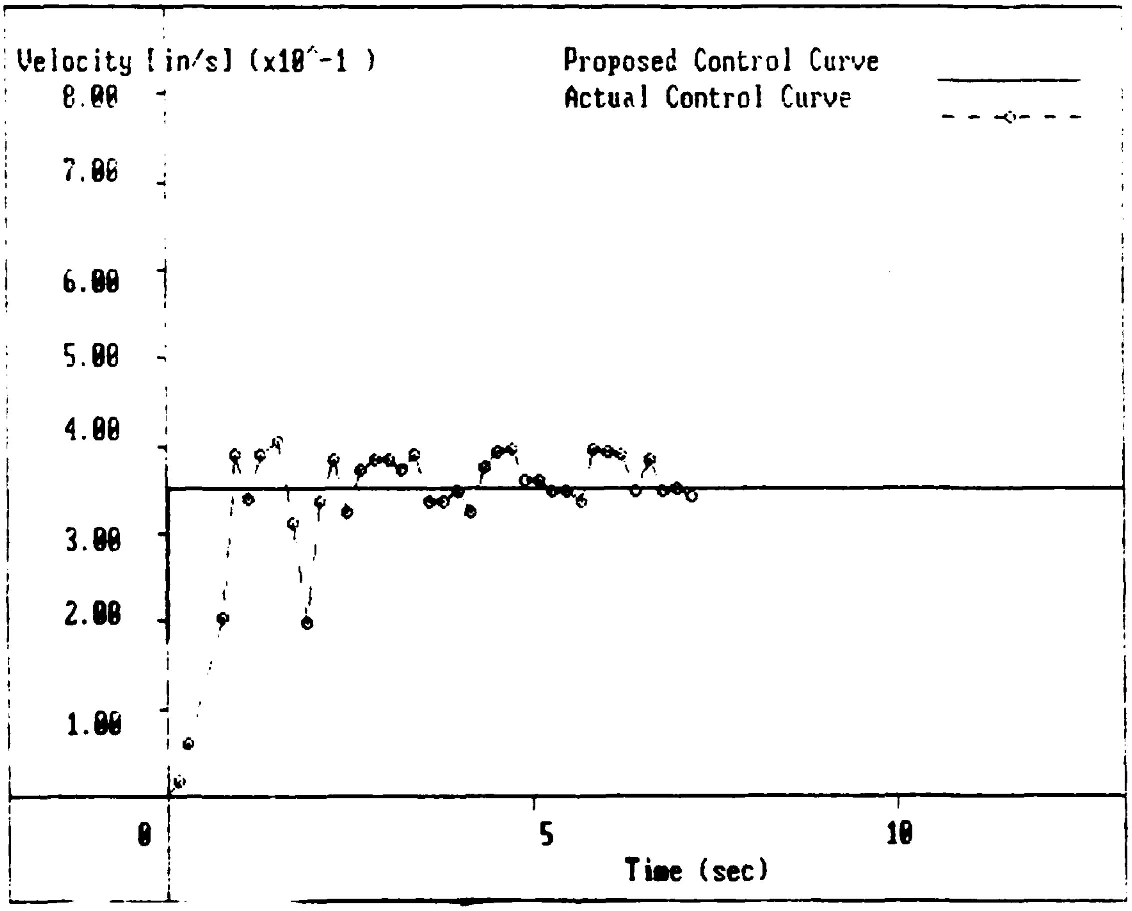


Figure 12: Averaging Finite Difference Control Results



numerical differentiation. The numerical derivative calculation used such a small step size that a small change in position produced a large change in velocity. Thus, the step size was increased by using previous positions along with the most recent position to calculate the velocity.

$$V(t) = \{X(t) - X(t-5)\} / \{T(t) - T(t-5)\} \qquad [13]$$

The previous position results used to finally control this press were the positions five readings before the most recent reading. Results of this control technique are located in Figure 12. The graph improved but could be even better if the sensor measuring the position was a velocity transducer because there still is considerable error within the numerical differentiation. Also, the position sensor makes use of a cable which, when the ram starts to move, "snaps" and causes a glitch which can be seen on the displacement versus time graph in figure 13. This glitch causes severe velocity calculation errors which could be corrected by a velocity transducer.

The magnitude of the velocity also causes problems in the performance of the press. The ram velocity is extremely limited because of the characteristics of the DC motor. At slow ram speeds, below 0.3 [in/s], the DC motor controller limits the current to the motor producing not enough torque to effectively move the press at the control

Figure 13: Example of Press Start Up "Snap"

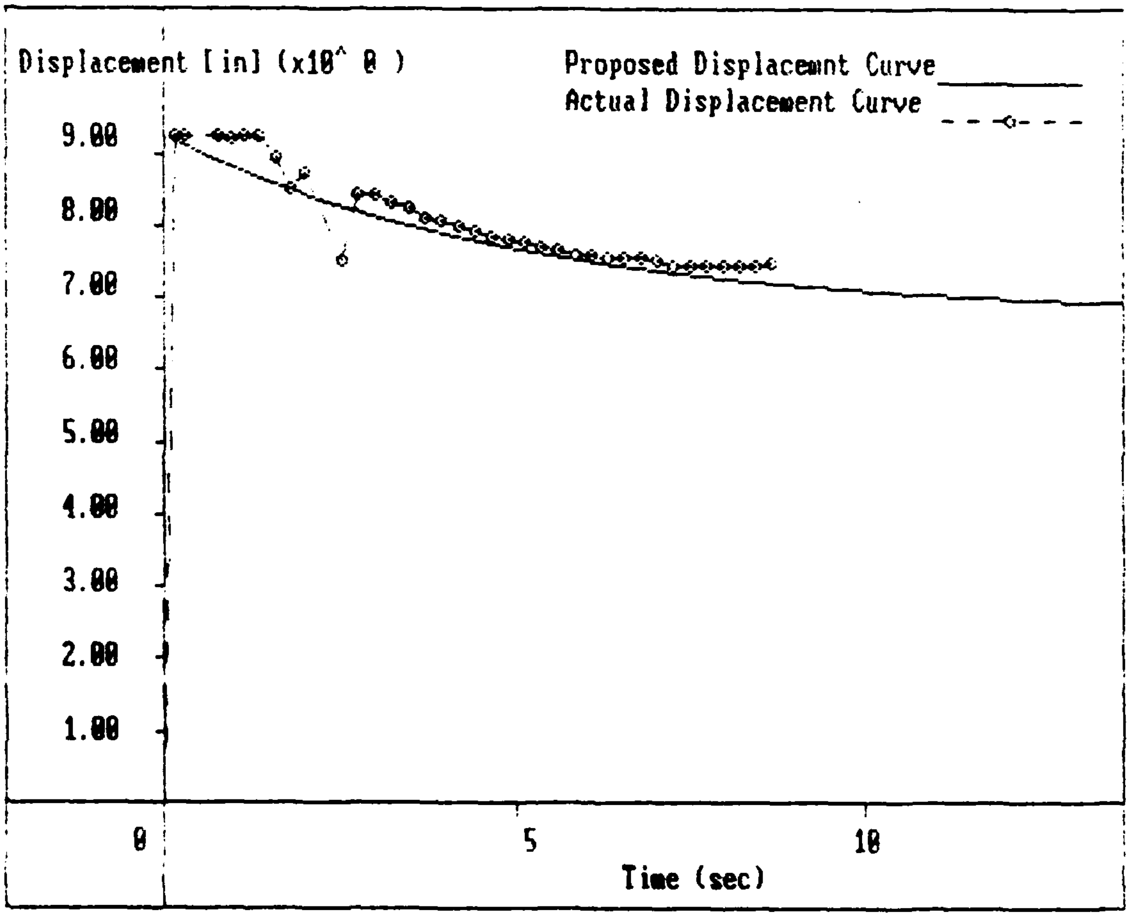


Figure 14: Example of Low Speed Ram Oscillations

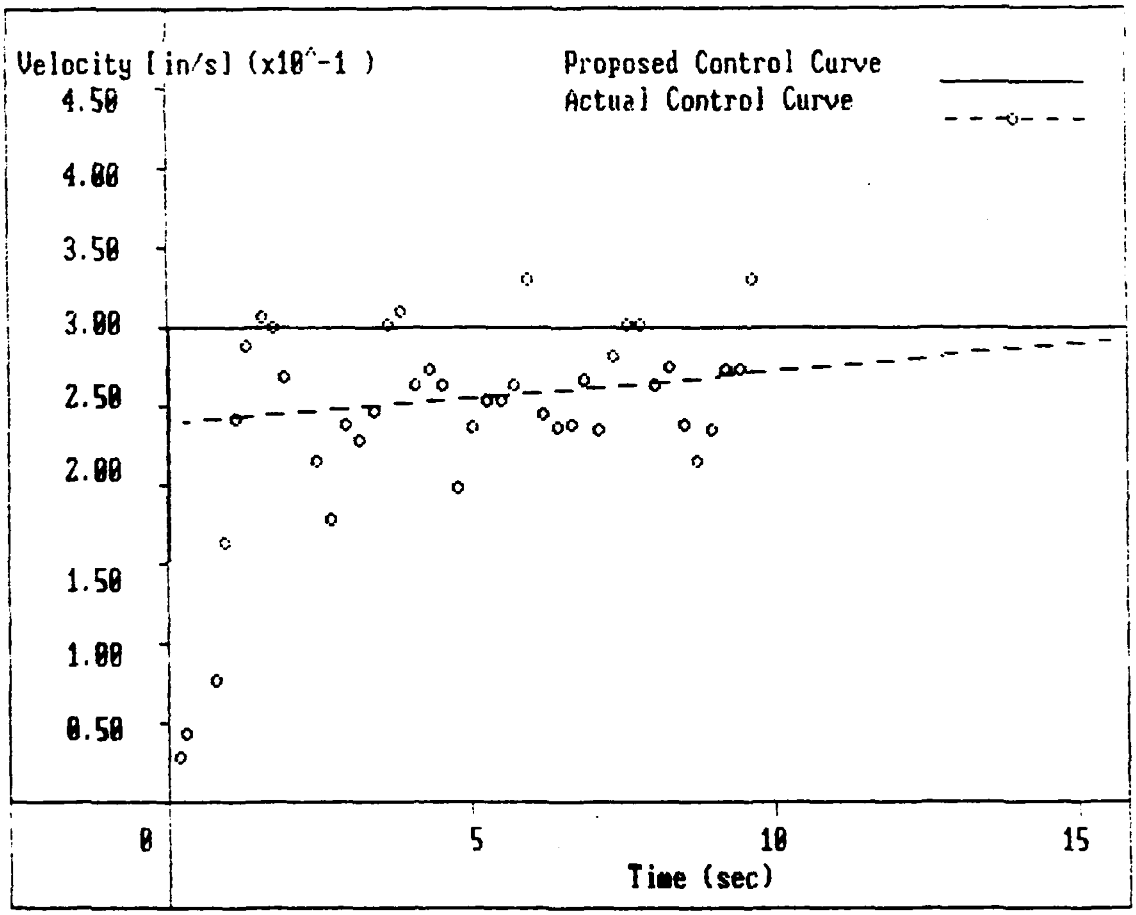
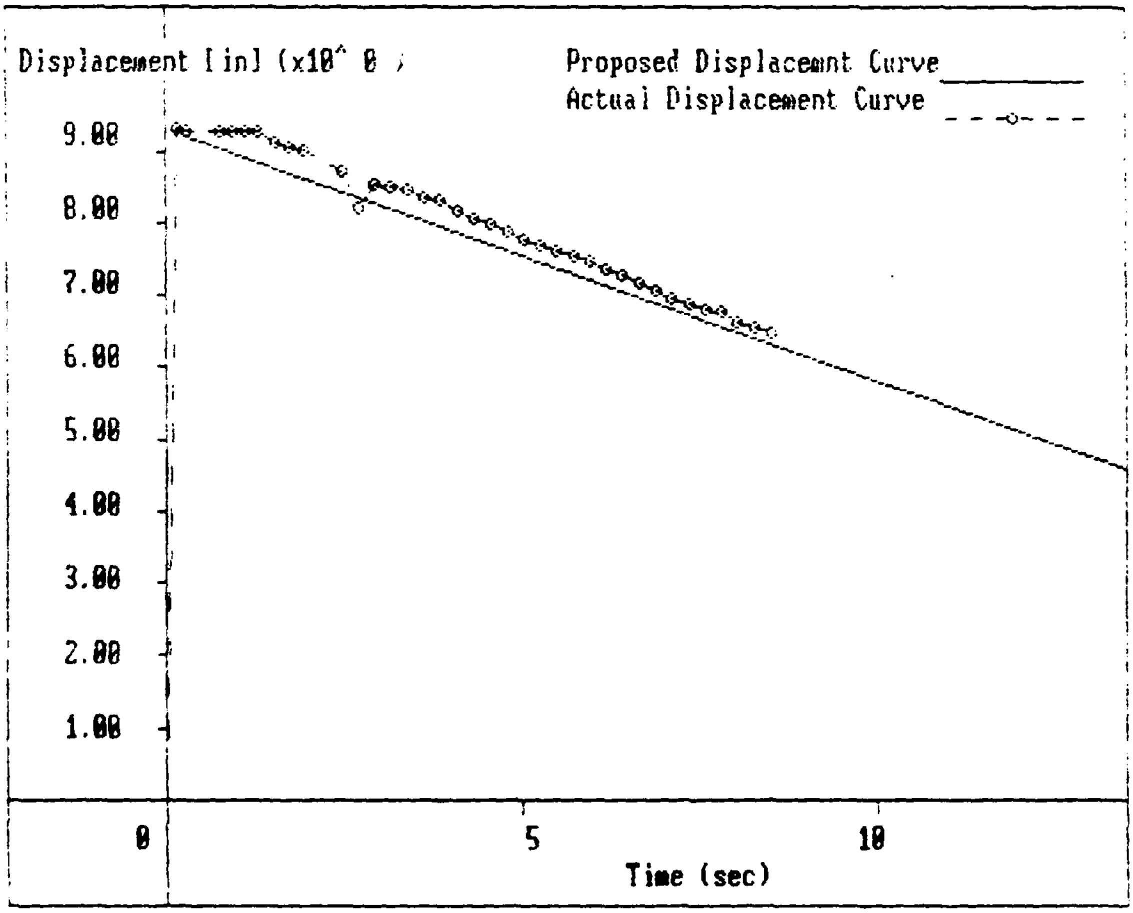


Figure 15: Example of an Intermediate Glitch



speed. Thus, the feedback causes the motor speed to oscillate as can be seen in figure 14. The motor should either be replaced with a motor with more low end torque or a gear box should be installed to increase the torque at lower output speeds. A similar result occurs during the application of a load. The billet being tested by the press produces a variable load which causes the oscillation of the platen of the press. The magnitude of this oscillation is dependent on the linearity of the load and the magnitude of force required by the press to accomplish the test. Once the maximum force of the press is reached, the speed decreases or stops completely with the attempted continuation of the test. This error is dependent on the hydraulic system's capacity and the motor selected. This present DC motor and controller were chosen using the specifications of the original AC motor which was not designed to operate at such slow speeds or at overload conditions.

The velocity of the ram is also affected by the imperfections of the press. Glitches caused by the main cylinder hanging up and the motor overcompensating for the increase in resistance causes serious velocity problems. Figure 15 shows an example of this intermediate glitch. As the press warms up these glitches tend to disappear but the oil temperature increases causing changes in the physical characteristics of the system. The motor velocity to ram velocity ratio changes slightly due to the decrease in the viscosity of the oil. If a multigrade hydraulic oil could be used, this change could be diminished.

Overall, the performance of the press is satisfactory. If a velocity transducer was used, the feedback signal would be of better quality thus increasing the accuracy of the press's velocity profile. The TUTSIM output shows no stability problems but it also showed that the response could be improved. If more transducers were available to measure the velocity, position, and acceleration a proportional-integral-derivative (PID) controller could be utilized. This control scheme could decrease response time and the amount of overshoot. The future of this press is consigned to the user and to the allowable error for the tests which the user will be conducting.

References

Berends, T.K. & Tal', A.A. (1968). Pneumatic Controllers with Automatic Readjustments According to the Load. In M.A. Aizerman (ed. and trans.). Pneumatic and Hydraulic Control Systems (Vol. 1, pp.20-26) Oxford, London: Pergamon Press Ltd. (Original work published 1959)

Instruction / Service Manual for the Walbash Hydraulic Press, Model # 25-1212-2TMBACX. Walbash, IN: Walbash Metal Products, Inc.

Kiuchi, M (1986). Complex Simulation System of Forging Based on UBET. CIPR Annals (Vol. 35, pp.147-150) Hallwag Publishers.

Kobayashi, S., & Thomsen, E.G. (1958). Approximate Solution to a Problem of Press Forging (Paper No. 58-A-140). New York, NY: American Society of Mechanical Engineers.

Kobayashi, S., Thomsen, E.G., Herzog, R., & Lapsley, J.T. Jr. (1958). Theory and Experiment of Press Forging Axisymetric Parts of Aluminum and Lead (Paper No. 58-A-154). New York, NY: American Society of Mechanical Engineers.

Kobayashi, S., Thomsen, E.G., & MacDonald, A.G. (1959). Some Prblems of Press Forging Lead and Aluminum. (Paper No. 59-A-164). New York, NY: American Society of Mechanical Engineers.

Kuo, B.C. (1987). Automatic Control Systems. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Merrit, H.E. (1967). Hydraulic Control Systems. New York: John Wiley & Sons, Inc.

Olson, R.M. (1980). Essentials of Engineering Fluid Mechanics. New York, NY: Harper & Row, Publishers, Inc.

Reed, E.W. & Larman, I.S. (1985). *Fluid Power with Microprocessor Control: An Introduction*. UK, Ltd: Prentice-Hall International.

Thaler, G.J. & Wilcox, M.L. (1966). *Electric Machines: Dynamics and Steady State*. New York: John Wiley & Sons, Inc.

Yeaple, F.D. (1966). *Hydraulic and Pneumatic Power and Control*. New York, NY: McGraw-Hill Book Company.

Appendix 1

Central Control Program "Control.bas" with Sample Output

```
10 CLEAR, 49152!
20 ' variable initialization
30 '
40 KEY OFF
50 START=0: VA=1.092075E-02 : VB=-1.18604 : FACT=.03 :
I%=0: J%=2: XMIN= 4.5 : VDMAX=0: STEADY=12! : XMAX=0 : K=.1
:PRE=0 : DELAY=10
60 LMAX=0 :VMAX=0
70 DIM T(2000), V(2000),  ARRAY%(2), LT%(2),
X(2000),L(2000)
80 CLS : LOCATE 5,25 : PRINT "HYDRAULIC PRESS CONTROL"
90 LOCATE 10,25 : PRINT "Select Control Algorithm:"
100 LOCATE 12,30 : PRINT "(1) Vout = A * exp(-B * t)"
110 LOCATE 13,30 : PRINT "(2) Vout from a Data File."
120 LOCATE 15,15 : PRINT "Note - For constant velocity,
enter (1) with B=0."
130 LOCATE 17,25 : INPUT A%
140 ON A% GOTO 160, 210
150 LOCATE 16,25 :PRINT "REENTER  1 or 2 "  : GOTO 90
160 LOCATE 20,25 : INPUT "Input function Constants A,B";A,B
170 DEF FNVOUT(X)=A*EXP(-B*X)
180 DEF FNXOUT(T)=ABS(A/B-X(1)-A/B*EXP(-B*T))
190 IF (B-.00001)<=0 THEN DEF FNXOUT(T)=X(1)-A*T
200 GOTO 270
210 LOCATE 20,25 : INPUT "Input Data File name";D$
220 OPEN "I",#1,D$
230 INPUT #1,L% : DIM TD(L%),VD(L%)
240 FOR Z%=1 TO L%
250 INPUT #1,TD(Z%),VD(Z%) : IF VDMAX<VD(Z%) THEN
VDMAX=VD(Z%)
260 NEXT
270 DEF SEG=0
280 SG=256*PEEK(&H511)+PEEK(&H510)
290 SG=49152!/16+SG
300 DEF SEG=SG
310 BLOAD "dash8.bin",0
320 DEF SEG=SG
330 DASH8=0
340 '
350 ' Initialize Routine and Scan Limits
360 '
370 MD%=0
380 BASADR%=&H330
390 CALL DASH8(MD%,BASADR%,FLAG%)
400 IF FLAG%<>0 THEN PRINT "Initialization Error ";FLAG% :
STOP
410 MD%=1
420 LT%(0)=0
430 LT%(1)=1
440 CALL DASH8(MD%,LT%(0),FLAG%)
450 IF FLAG%<>0 THEN PRINT "Multiplexer Scan Error ";FLAG%
```

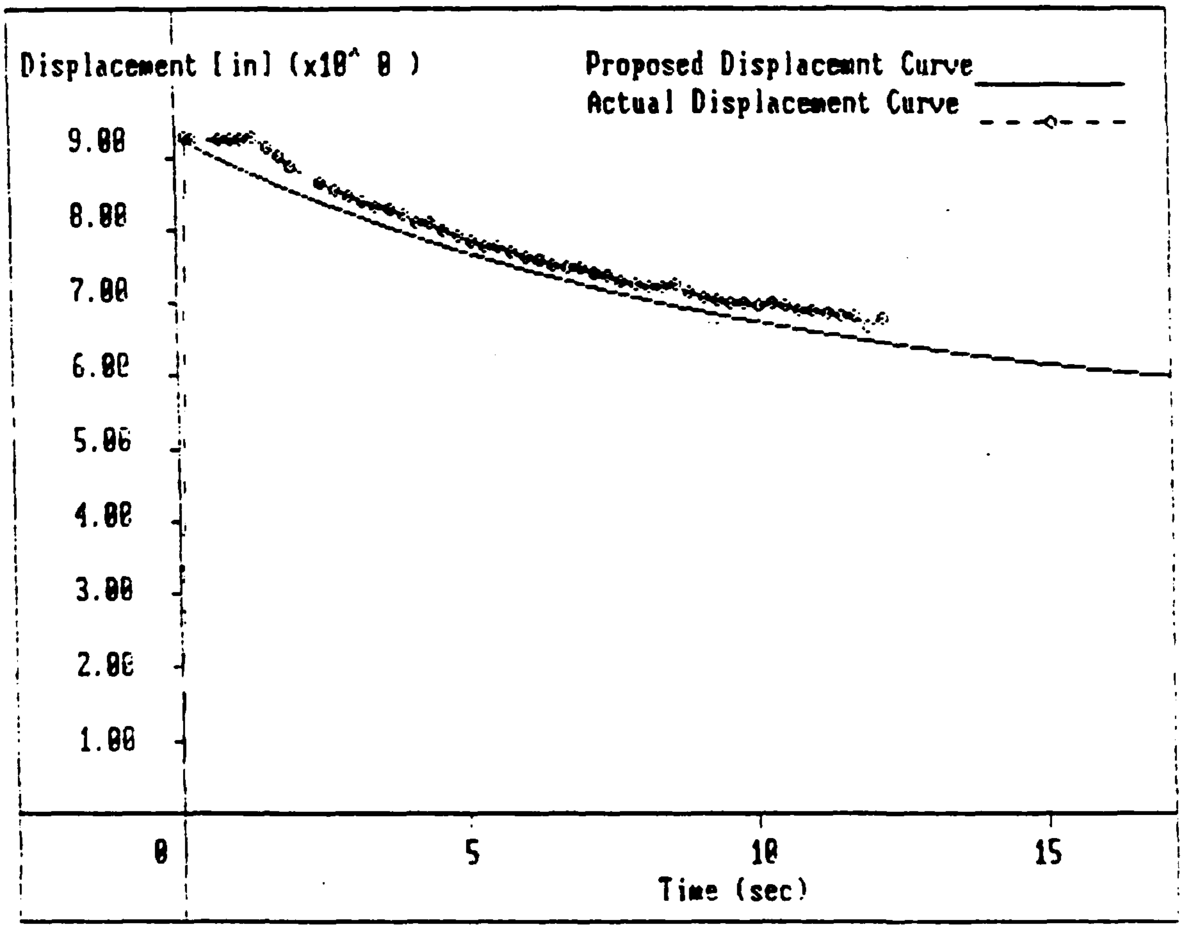
```
: STOP
460 CLS : LOCATE 10,10 : PRINT "Pull start/stop button on
the press then hit any key to continue."
470 IF INKEY$="" THEN 470
480 CLS : LOCATE 10,20 : PRINT "Move press into position by
pressing any key."
490 IF INKEY$="" THEN 490
500 OUT &H270,240 : OUT &H271,255
510 CLS : LOCATE 10,30 : PRINT "Hit any key to stop press."
520 IF INKEY$="" THEN 520
530 OUT &H270,0 : OUT &H271,0
540 CLS : LOCATE 10,30 : PRINT "Hit any key to begin test"
550 IF INKEY$="" THEN 550
560 CLS : LOCATE 10,25 : PRINT "Hit any key to stop press"
570 START=TIMER : MD%=13 : OP%=1
580 CALL DASH8(MD%,OP%,FLAG%)
590 IF FLAG%<>0 THEN PRINT "Press start Error ";FLAG% :
STOP
600 MD%=4
610 '
620 ' Control Loop
630 '
640 I%=I%+1
650 FOR II=1 TO DELAY
660 NEXT II
670 T(I%)=TIMER-START
680 CALL DASH8(MD%,ARRAY%(0),FLAG%) : CALL
DASH8(MD%,ARRAY%(1),FLAG%)
690 IF FLAG%<>0 THEN PRINT "Conversion Error ";FLAG% : STOP
700 IF I%=1 THEN L(1)=(-.00244*2.44143E-
03*ARRAY%(1)+.01)*1000*19000 : X1=VA*ABS(ARRAY%(0))+VB :
X(I%)=X1 : V(1)=0 : GOTO 640
710 X2=VA*ABS(ARRAY%(0))+VB : X(I%)=X2 : IF XMAX<X(I%) THEN
XMAX=X(I%)
720 V(I%)=(X2-X1)/(T(I%)-T(I%-1))
730 IF I%>5 THEN Z%=Z%+1 : V(Z%)=ABS((X(Z%)-X(Z%+5))/(T(Z%)-
T(Z%+5)))
735 IF Z%>3 THEN ZZ%=ZZ%+1 : XI=XI+T(Z%) : XI2=XI2+T(Z%)^2
: YI=YI+LOG(V(Z%)) :  XY=XY+T(Z%)*LOG(V(Z%))
740 IF VMAX<ABS(V(I%)) THEN VMAX=ABS(V(I%))
750 L(I%)=(-.00244*2.44143E-03*ARRAY%(1)+.01)*1000*19000
760 IF LMAX<L(I%) THEN LMAX=L(I%)
770 X1=X2
780 IF A%=1 THEN VOUT=FNVOUT(T(I%)/1.8) : GOTO 800
790 IF T(I%-1)<TD(J%) THEN VOUT=(VD(J%)-VD(J%-1))/(TD(J%)-
TD(J%-1))+T(I%)-VD(J%)-(VD(J%)-VD(J%-1))/(TD(J%)-TD(J%-
1))*TD(J%) ELSE J%=J%+1 : IF J%<=L% THEN 790 ELSE VOUT=0 :
D$="stop"
800 FACT=FACT+PRE : VOUT=VOUT+FACT*(VOUT-V(I%))
810 PRE=K*(VOUT-V(I%))
820 VOUT = VOUT *STEADY
```

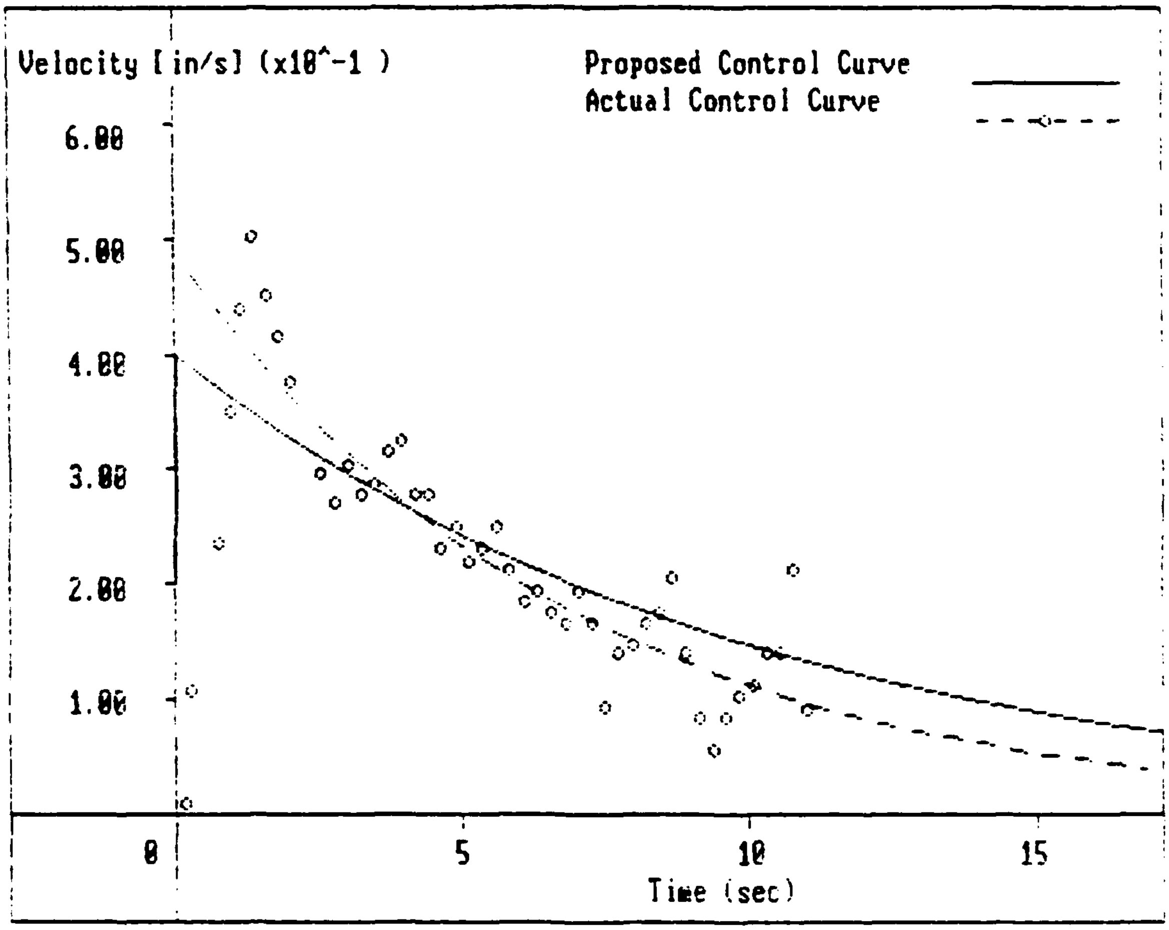
```
830 IF VOUT>10 THEN VOUT=10
840 IF VOUT<0 THEN VOUT=0
850 D%=INT(4095/10*VOUT)
860 DH%=INT(D%/16)
870 DL%=D%-16*DH%
880 DL%=16*DL%
890 OUT &H270,DL%
900 OUT &H271,DH%
910 IF ((X2>XMIN) AND (D$<>"stop")) AND (INKEY$="") THEN
640
920 OUT &H270,0 : OUT &H271,0
930 CLS : LOCATE 10,20 : PRINT "Hit any key to return
press."
940 IF INKEY$="" THEN 940
950 '
960 ' return and stop press
970 '
980 MD%=13 : OP%=0
990 CALL DASH8(MD%,OP%,FLAG%)
1000 IF FLAG%<>0 THEN PRINT "Press return error ";FLAG% :
STOP
1010 D%=4095
1020 DH%=INT(D%/16)
1030 DL%=D%-16*DH%
1040 DL%=16*DL%
1050 OUT &H270,DL%
1060 OUT &H271,DH%
1070 CLS : LOCATE 10,10 : PRINT "Push start/stop button on
press then hit any key to continue."
1080 IF INKEY$="" THEN 1080
1090 TMAX=T(I%)+5
1100 '
1110 ' Set up Graph - Displacement vs Time
1120 '
1130 SCREEN 3
1140 LINE (100,0)-(100,340)
1150 LINE (0,300)-(700,300)
1160 LINE (0,0)-(700,340),,B
1170 LOCATE 23,10 : PRINT "0"
1180 LOCATE 24,44 : PRINT "Time (sec)";
1190 N%=2 :NN%=0 : K%=5
1200 N%=N%-1
1210 IF XMAX<(10^N%) THEN 1200
1220 NN%=NN%+1
1230 IF NN%< INT(XMAX*10^(-N%)) THEN 1220 ELSE NN%=NN%+2
1240 IF NN%<=3 THEN S=.25 ELSE IF NN%<=6 THEN S=.5 ELSE
S=1
1250 YSCALE=NN%*10^N%
1260 YS=YS+S
1270 Y%=(NN%-YS)*300/NN%
1280 LINE (95,Y%)-(100,Y%)
```

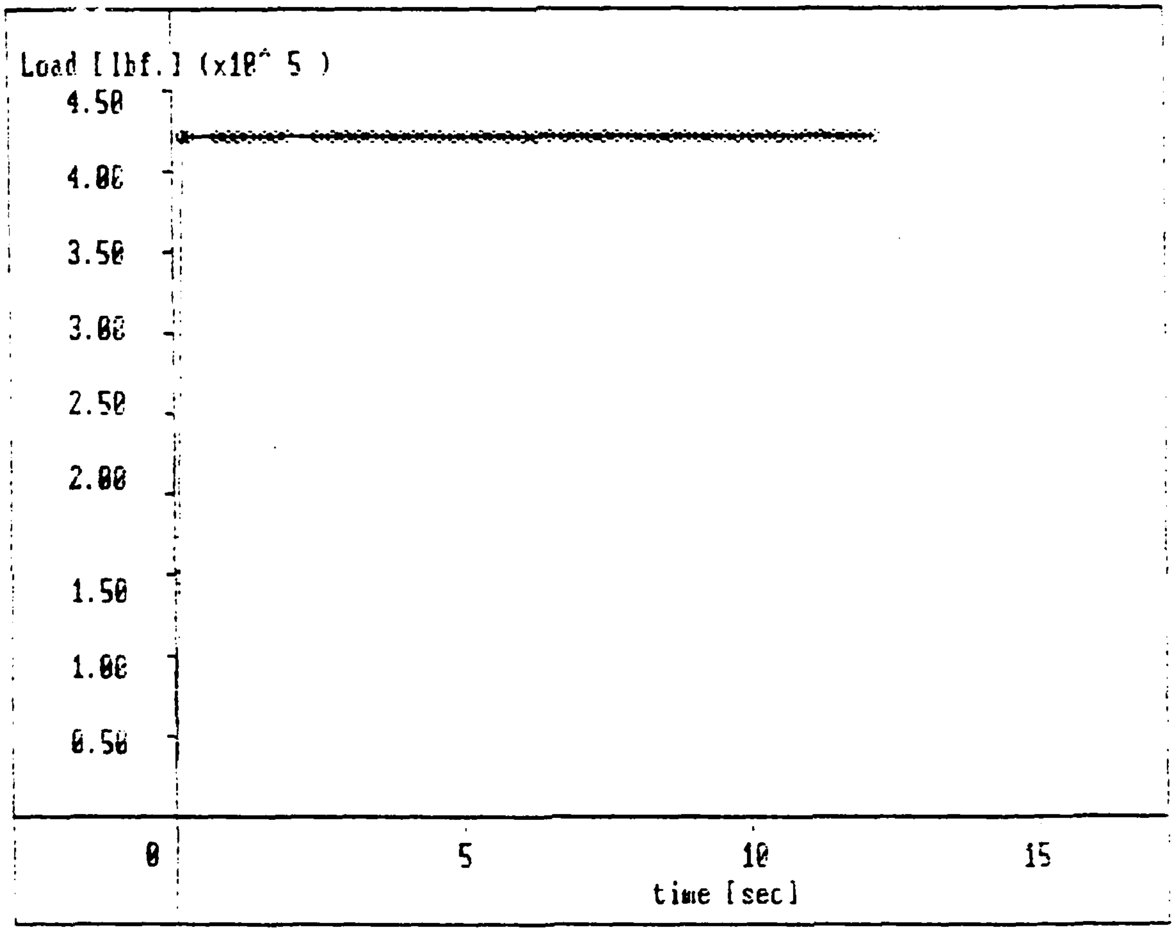
```
1290 LOCATE (Y%/348*25+.5),5 : PRINT USING "#.##";YS
1300 IF INT(YS+S+.0001)<NN% THEN 1260
1310 LOCATE 2,2 : PRINT "Displacement [in] (x10^";N%;")"
1320 XS%=K%*600/TMAX+100
1330 LINE (XS%,300)-(XS%,305)
1340 LOCATE 23,XS%/720*80 : PRINT USING "##";K%
1350 K%=K%+5
1360 IF K%<TMAX THEN 1320
1370 FOR J%=1 TO I%
1380 T(J%)=T(J%)*600/TMAX+100
1390 X(J%)=(YSCALE-X(J%))*300/YSCALE
1400 IF J%=1 THEN LINE (100,300)-(T(J%),X(J%)),,,&HFF00 :
CIRCLE (T(J%),X(J%)),3           : GOTO 1430
1410 LINE -(T(J%),X(J%)),,,&HFF00
1420 CIRCLE (T(J%),X(J%)),3
1430 X(J%)=YSCALE-X(J%)*YSCALE/300
1440 NEXT J%
1450 J%=110 : LINE (T(1),(YSCALE-X(1))*300/YSCALE)-
(J%,(YSCALE-FNXOUT((J%-100)*TMAX/600))*300/YSCALE)
1460 J%=J%+1 : LINE -(J%,(YSCALE-FNXOUT((J%-
100)*TMAX/600))*300/YSCALE)
1470 IF J%<700 THEN 1460
1480 LOCATE 2,40 : PRINT "Proposed Displacemnt Curve"
1490 LOCATE 3,40 : PRINT "Actual Displacement Curve"
1500 LINE (585,28)-(675,28)
1510 LINE (585,42)-(675,42),,,&HFF00
1520 CIRCLE (630,42),3
1530 IF INKEY$="" THEN 1530
1540 '
1550 ' Set up Graph - Velocity vs Time
1560 '
1570 SCREEN 3
1580 CLS : YS=0
1590 LINE (100,0)-(100,340)
1600 LINE (0,300)-(700,300)
1610 LINE (0,0 -(700,340),,B
1620 LOCATE 23,10 : PRINT "0"
1630 LOCATE 24,44 : PRINT "Time (sec)";
1640 N%=2 :NN%=0 : K%=5
1650 N%=N%-1
1660 IF VMAX<10^N%) THEN 1650
1670 NN%=NN%+1
1680 IF NN%<INT(VMAX*10^(-N%)) THEN 1670 ELSE NN%=NN%+1
1690 IF NN%<=3 THEN S=.25 ELSE IF NN%<=6 THEN S=.5 ELSE
S=1)
1700 YSCALE=NN%*10^N%
1710 YS=YS+S
1720 Y%=(NN%-YS)*300/NN%
1730 LINE (95,Y%)-(100,Y%)
1740 LOCATE (Y%/348*25+.5),5 : PRINT USING "#.##";YS
1750 IF INT(YS+S+.0001)<NN% THEN 1710
```

```
1760 LOCATE 2,2 : PRINT "Velocity [in/s] (x10^";N%;")"
1770 XS%=K%*600/TMAX+100
1780 LINE (XS%,300)-(XS%,305)
1790 LOCATE 23,XS%/720*80 : PRINT USING "##";K%
1800 K%=K%+5
1810 IF K%<TMAX THEN 1770
1820 FOR J%=1 TO Z%
1830 V(J%)=(YSCALE-V(J%))*300/YSCALE
1840 CIRCLE (T(J%),V(J%)),3
1850 NEXT J%
1860 C=(ZZ%*XY-XI*YI)/(ZZ%*XI2-XI^2)
1870 D=(YI*XI2-XY*XI)/(ZZ%*XI2-XI^2) : D=EXP(D)
1880 DEF FNEEXP(X)=(YSCALE-D*EXP(C*(X-
100)*TMAX/600))*300/YSCALE
1885 J%=110 : LINE (J%,FNEEXP(J%))-(J%,FNEEXP(J%))
1890 J%=J%+20 : LINE -(J%,FNEEXP(J%)),,,&HFF00
1895 IF J%<700 THEN 1890
1910 IF A%=2 THEN 1960
1920 J%=101 : LINE (100,300)-(J%,(YSCALE-FNVOUT((J%-
100)*TMAX/600))*300/YSCALE)
1930 J%=J%+1 : LINE -(J%,(YSCALE-FNVOUT((J%-
100)*TMAX/600))*300/YSCALE)
1940 IF J%<700 THEN 1930
1950 GOTO 2030
1960 J%=2
1970 FOR M=1 TO L%
1980 TD(M)=TD(M)*600/TMAX +100
1990 VD(M)=(YSCALE-VD(M))*300/YSCALE
2000 IF M=1 THEN LINE(100,300)-(TD(M),VD(M)) : GOTO 2020
2010 LINE -(TD(M),VD(M))
2020 NEXT M
2030 LOCATE 2,40 : PRINT "Proposed Control Curve"
2040 LOCATE 3,40 : PRINT "Actual Control Curve"
2050 LINE (585,28)-(675,28)
2060 LINE (585,42)-(675,42),,,&HFF00
2070 CIRCLE (630,42),3
2080 IF INKEY$="" THEN 2080
2090 '
2100 '  Graph - load vs time
2110 '
2120 SCREEN 3
2130 CLS
2140 LINE (100,0)-(100,340)
2150 LINE (0,300)-(700,300)
2160 LINE (0,0)-(700,340),,B
2170 LOCATE 23,10 : PRINT "0"
2180 LOCATE 24,44 : PRINT "time [sec]";
2190 N%=6 : NN%=0 : K%=5 : YS=0
2200 N%=N%-1
2210 IF LMAX<(10^N%) THEN 2200
2220 NN%=NN%+1
```

```
2230 IF NN%<INT(LMAX*10^(-N%)) THEN 2220 ELSE NN%=NN%+1
2240 IF NN%<=3 THEN S=.25 ELSE IF NN%<=6 THEN S=.5 ELSE
S=1!
2250 YSCALE=NN%*10^N%
2260 YS=YS+S
2270 Y%=(NN%-YS)*300/NN%
2280 LINE (95,Y%)-(100,Y%)
2290 LOCATE (Y%/348*25+.5),5 : PRINT USING "#.##";YS
2300 IF INT(YS+S+.0001) < NN% THEN 2260
2310 LOCATE 2,2 : PRINT "Load [lbf.] (x10^";N%;")"
2320 XS%=K%*600/TMAX+100
2330 LINE (XS%,300)-(XS%,305)
2340 LOCATE 23,XS%/720*80 : PRINT USING "##";K%
2350 K%=K%+5
2360 IF K%<TMAX THEN 2320
2370 FOR M=1 TO I%
2380 L(M)=(YSCALE-L(M))*300/YSCALE
2390 IF M=1 THEN LINE(100,300)-(T(M),L(M)):GOTO 2410
2400 LINE -(T(M),L(M))
2410 CIRCLE (T(M),L(M)),3
2420 NEXT M
2430 OUT &H270,0
2440 OUT &H271,0
2450 IF INKEY$="" THEN 2450
2460 SCREEN 0,0,0
2470 LOCATE 10,15 : INPUT "Would you like another run
(y/n)";A$
2480 IF A$="y" THEN 10
2490 END
```


Displacement [in] (x10^ 0)
Proposed Displacemnt Curve
Actual Displacement Curve
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0
5
10
15
Time (sec)


Velocity [in/s] (x10^-1)
Proposed Control Curve
Actual Control Curve
6.00
5.00
4.00
3.00
2.00
1.00
0
5
10
15
Time (sec)


Load [lbf.] (x10^ 5)
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0
5
10
15
time [sec]

Appendix 2

Data File Creator Program
"Datafile.bas"

```
10 ' Data File Creator
20 '
30 CLS :LOCATE 10,30 : PRINT "Data File Creator"
40 PRINT : INPUT "Name of Data File";A$
50 OPEN "O",#2,A$
60 INPUT "Enter number of points to be used";L
70 PRINT #2,L
80 FOR I=1 TO L
90 CLS : LOCATE 10,30 : PRINT "Group #";I;": ";: INPUT
"Enter T,V";T,V
100 PRINT #2,T,V
110 NEXT I
120 CLOSE #2
```

Appendix 3

Steady State Velocity Program
"Steady.bas"

```
10 ' Steady state velocity vs Input voltage
20 KEY OFF
30 CLEAR,49152!
40 N=0: XY=0: X=0: Y=0: X2=0: ADMAX=0: HMAX=0
50 DIM H(3000), AD(3000), LT%(2), ARRAY%(2)
60 DIM V(3000)
70 '
80 ' Load and Initialize A/D Converter
90 '
100 DEF SEG=0
110 SG=256*PEEK(&H511)+PEEK(&H510)
120 SG=49152!/16+SG
130 DEF SEG=SG
140 BLOAD"dash8.bin",0
150 DEF SEG=SG
160 DASH8=0
170 MD%=0
180 BASADR%=&H330
190 CALL DASH8(MD%,BASADR%,FLAG%)
200 IF FLAG%<>0 THEN PRINT "Initialization error ";FLAG% :
STOP
210 MD%=1
220 LT%(0)=0
230 LT%(1)=1
240 CALL DASH8(MD%,LT%(0),FLAG%)
250 IF FLAG%<>0 THEN PRINT "Multiplexer Scan Error ";FLAG%
: STOP
260 CLS : LOCATE 10,10 :PRINT "Pull start/stop button on
press then hit any key to continue"
270 IF INKEY$="" THEN 270
280 MD%=13 : OP%=1 : CALL DASH8(MD%,OP%,FLAG%)
290 IF FLAG%<>0 THEN PRINT "Press start error ";FLAG% :
STOP
300 MD%=4 : I%=1
310 INPUT "volts";V :V=V*4095/10
320 DH%=INT(V/16)
330 DL%=INT(V-16*DH%)
340 DL%=16*DL%
350 OUT &H270,DL%
360 OUT &H271,DH%
370 START=TIMER : TOLD=0 : I%=0
380 TNEW=TIMER-START
390 CALL DASH8(MD%,ARRAY%(0),FLAG%) : CALL
DASH8(MD%,ARRAY%(1),FLAG%)
400 XNEW=ARRAY%(0)
410 IF TOLD=0 THEN XOLD=XNEW : TOLD=TNEW : GOTO 380
420 I%=I%+1
430 V(I%)=ABS((XNEW-XOLD)/(TNEW-TOLD))
440 XOLD=XNEW
450 TOLD=TNEW
460 XNEW=1.183157E-02*ABS(XNEW)-1.1027
```

```
470 PRINT XNEW
480 IF XNEW>10.5 THEN 380
490 MD%=13 : OP%=0 : CALL DASH8(MD%,OP%,FLAG%)
500 VV=0
510 FOR J=1 TO I%
520 VV=VV+V(J)
530 NEXT
540 VV=VV/I%
550 PRINT VV/V
560 IF INKEY$="" THEN 560
570 OUT &H270,0
580 OUT &H271,0
```

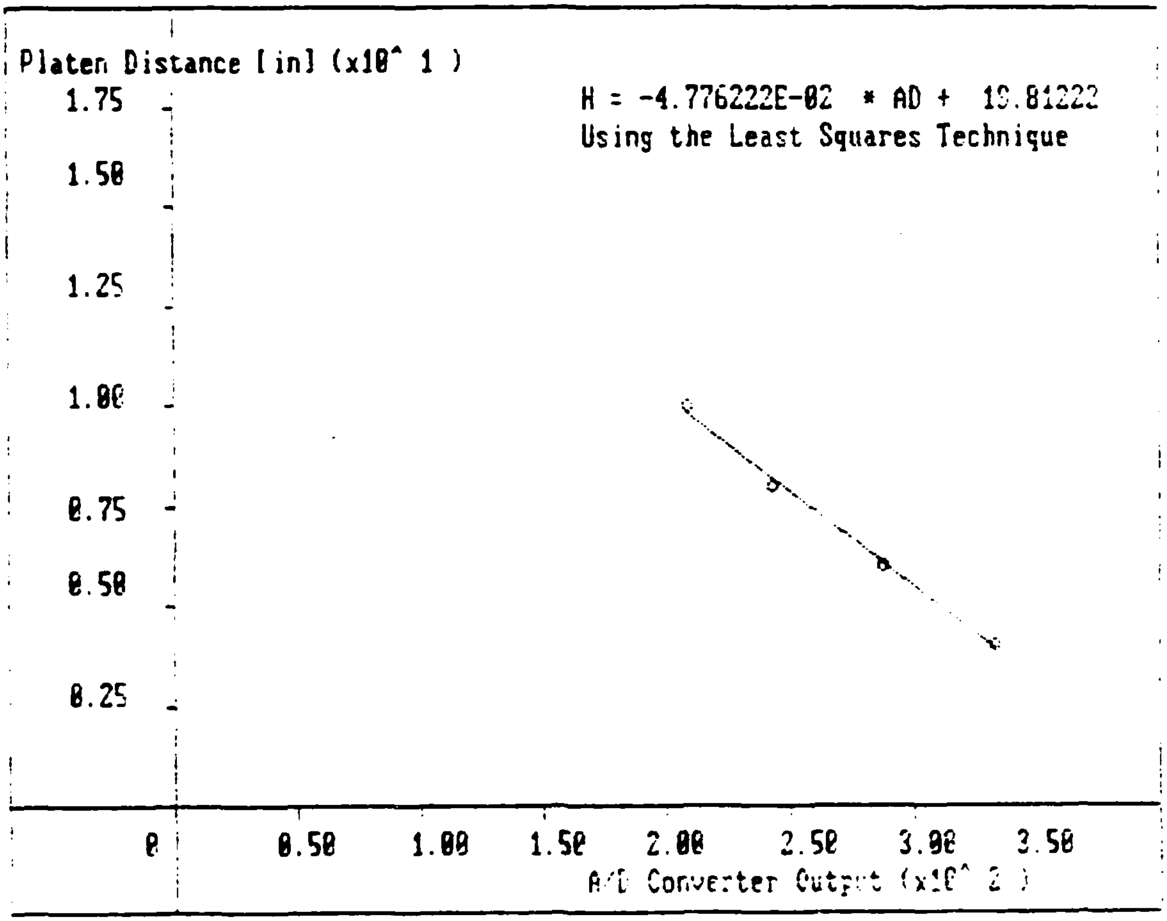
Appendix 4

Position Calibration Program "Calibrat.bas" with Sample Output

```
10 ' Position Calibration Using Least Squares
20 KEY OFF
30 CLEAR,49152!
40 N=0: XY=0: X=0: Y=0: X2=0: ADMAX=0: HMAX=0
50 DIM H(3000), AD(3000), LT%(2), ARRAY%(2)
60 '
70 ' Load and Initialize A/D Converter
80 '
90 DEF SEG=0
100 SG=256*PEEK(&H511)+PEEK(&H510)
110 SG=49152!/16+SG
120 DEF SEG=SG
130 BLOAD"dash8.bin",0
140 DEF SEG=SG
150 DASH8=0
160 MD%=0
170 BASADR%=&H330
180 CALL DASH8(MD%,BASADR%,FLAG%)
190 IF FLAG%<>0 THEN PRINT "Initialization error ";FLAG% :
STOP
200 MD%=1
210 LT%(0)=0
220 LT%(1)=1
230 CALL DASH8(MD%,LT%(0),FLAG%)
240 IF FLAG%<>0 THEN PRINT "Multiplexer Scan Error ";FLAG%
: STOP
250 CLS : LOCATE 10,10 :PRINT "Pull start/stop button on
press then hit any key to continue"
260 IF INKEY$="" THEN 260
270 MD%=13 : OP%=1 : CALL DASH8(MD%,OP%,FLAG%)
280 IF FLAG%<>0 THEN PRINT "Press start error ";FLAG% :
STOP
290 MD%=4 : I%=1
300 '
310 ' Calibration Loop
320 '
330 CALL DASH8(MD%,ARRAY%(0),FLAG%) : CALL
DASH8(MD%,ARRAY%(1),FLAG%)
340 IF FLAG%<>0 THEN PRINT "Conversion Error ";FLAG%;" on
step ";I% : STOP
350 CLS : LOCATE 10,10
360 INPUT "Enter Distance between Platens [in]";H(I%)
370 IF H(I%)>HMAX THEN HMAX=H(I%)
380 AD(I%)=ABS(ARRAY%(0))
390 IF AD(I%)> ADMAX THEN ADMAX=AD(I%)
400 XY=XY+H(I%)*AD(I%)
410 X=X+AD(I%)
420 Y=Y+H(I%)
430 X2=X2+AD(I%)^2
440 DH%=INT(4095/16)
450 DL%=4095-16*DH%
```

```
460 DL%=16*DL%
470 OUT &H270,DL%
480 OUT &H271,DH%
490 FOR J=1 TO 1000
500 NEXT J
510 DL%=0 : DH%=0
520 OUT &H270,DL%
530 OUT &H271,DH%
540 I%=I%+1
550 LOCATE 20,11
560 INPUT "Check the clamp light. Is it on (y/n)";A$
570 IF A$="n" THEN 330
580 '
590 ' return press
600 '
610 MD%=13 : OP%=0
620 CALL DASH8(MD%,OP%,FLAG%)
630 IF FLAG%<>0 THEN PRINT "Press return error ";FLAG% :
STOP
640 DH%=INT(4095/16)
650 DL%=4095-16*DH%
660 DL%=16*DL%
670 OUT &H270,DL%
680 OUT &H271.DH%
690 CLS : LOCATE 10,10
700 PRINT "Push stop/start button on press then hit any key
to continue."
710 IF INKEY$="" THEN 710
720 '
730 ' Plot Graph
740 '
750 SCREEN 3
760 LINE (100,0)-(100,340)
770 LINE (0,300)-(700,300)
780 LINE (0,0)-(700,340),,B
790 LOCATE 23,10 : PRINT "0"
800 N%=3 : NN%=0 : YS=0
810 N%=N%-1
820 IF HMAX <(10^N%) THEN 810
830 NN%=NN%+1
840 IF NN%<INT(HMAX*10^(-N%)) THEN 830 ELSE NN%=NN%+1
850 IF NN%<=3 THEN S=.25 ELSE IF NN%<=6 THEN S=.5 ELSE S=1
860 YSCALE=NN%*10^N%
870 YS=YS+S
880 Y%=(NN%-YS)*300/NN%
890 LINE (95.Y%)-(100.Y%)
900 LOCATE (Y%/348*25+.5).5 : PRINT USING "#.##";YS
910 IF INT(YS+S+.0001) <= NN% THEN 870
920 LOCATE 2.2 : PRINT "Platen Distance, H [in]
x10 ";N%;")"
930 N%=5 : NN%=0 : YS=0
```

```
940 N%=N%-1
950 IF ADMAX<(10^N%) THEN 940
960 NN%=NN%+1
970 IF NN%<INT(ADMAX*10^(-N%)) THEN 960 ELSE NN%=NN%+1
980 IF NN%<=3 THEN S=.25 ELSE IF NN%<=6 THEN S=.5 ELSE S=1!
990 XSCALE=NN%*10^N%
1000 YS=YS+S
1010 Y%=YS*600/NN%+100
1020 LINE (Y%,300)-(Y%,305)
1030 LOCATE 23,(Y%/720*80) : PRINT USING "#.##";YS
1040 IF INT(YS+S+.0001) <NN% THEN 1000
1050 LOCATE 24,40 : PRINT "A/D Converter Output
(x10^";N%;")";
1060 FOR J=1 TO I%-1
1070 H(J)=(YSCALE-H(J))*300/YSCALE
1080 AD(J)=AD(J)*600/XSCALE+100
1090 CIRCLE (AD(J),H(J)),3
1100 NEXT J
1110 '
1120 ' Least Squares Technique
1130 '
1140 VA=((I%-1)*XY-X*Y)/((I%-1)*X2-X^2)
1150 VB=(X2*Y-XY*X)/((I%-1)*X2-X^2)
1160 DEF FNF(X)=(YSCALE-(VA*XSCALE/600*(X-
100)+VB))*300/YSCALE
1170 Y1=FNF(AD(1)) : Y2=FNF(AD(I%-1))
1180 LINE (AD(1),Y1)-(AD(I%-1),Y2)
1190 LOCATE 3,40 : PRINT "H = ";VA;" * AD + ";VB
1200 LOCATE 4,40 : PRINT "Using the Least Squares
Technique"
1210 IF INKEY$="" THEN 1210
1220 OUT &H270,0
1230 OUT &H271,0
1240 SCREEN 0,0,0
```

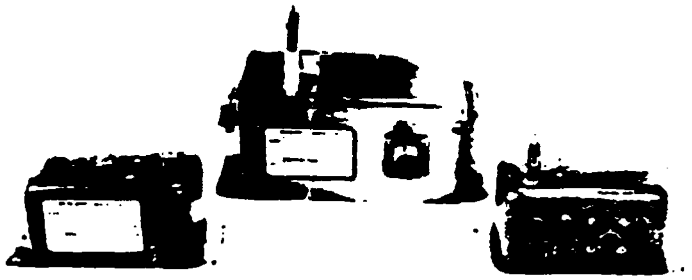

Platen Distance [in] (x10^ 1)
H = -4.776222E-02 * AD + 19.81222
Using the Least Squares Technique
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
0.50
1.00
1.50
2.00
2.50
3.00
3.50
A/D Converter Output (x10^ 2)

Appendix 5

Position Sensor Data

POSITION TRANSDUCER

1850 SERIES



The Solution is obvious!

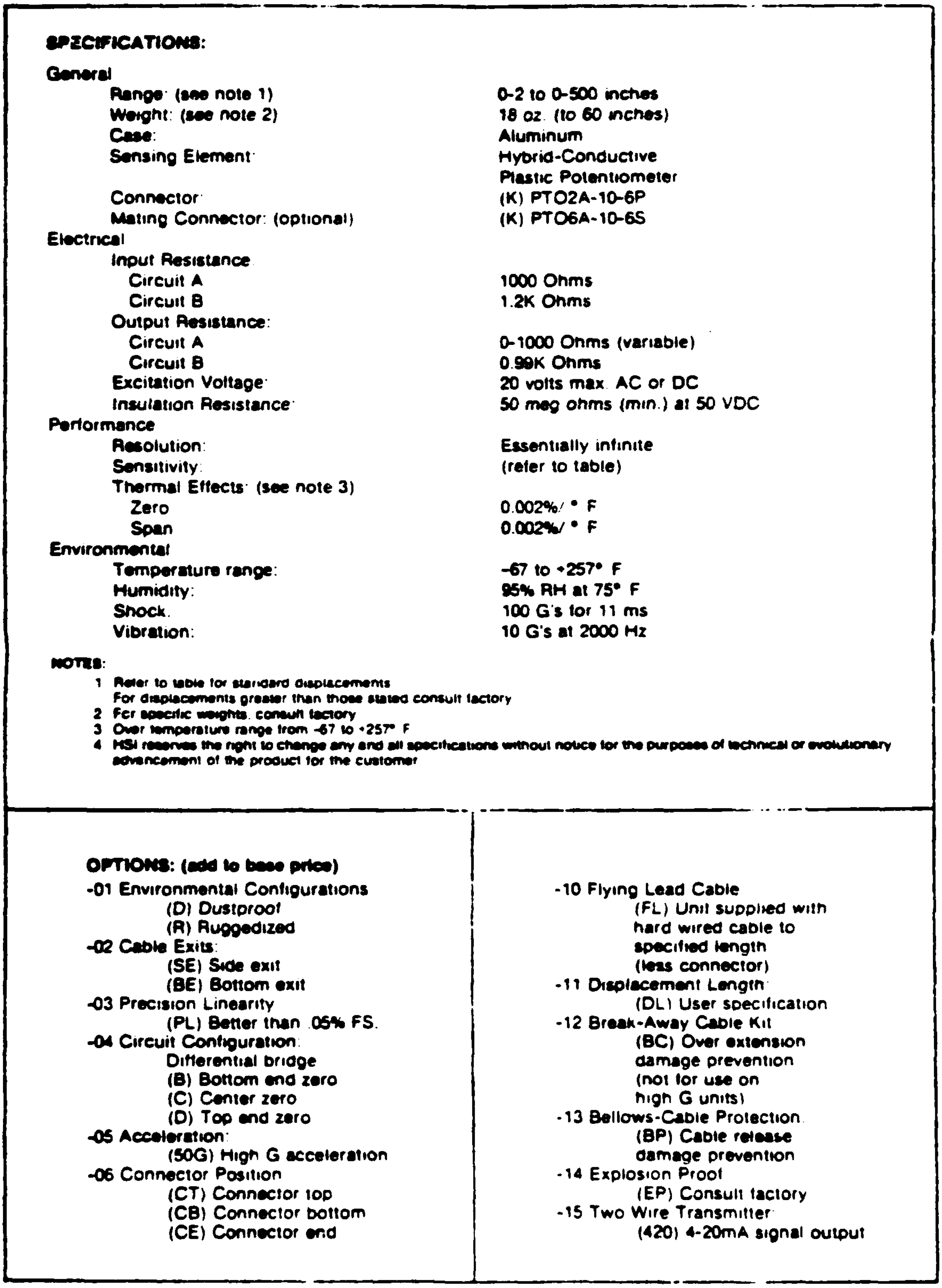
Houston Scientific has the solution for displacement measurement problems with a device whose history spans 25 years. From the very first design to the very latest, the Original Position Transducer has led the way to become the Industry Standard. Through years of engineering expertise in application and design, Houston Scientific can provide the finest cable actuated displacement transducers available, one that stands as the model for the competition.

The Model 1850 Spirator Motor Module (SMM) series is designed with performance in mind. All components are carefully designed for precision and long life service. The hybrid-conductive plastic potentiometer provides infinite resolution and linearity of better than 0.1% of full scale as standard and optional linearity of better than 0.05%. The A circuit configuration is a voltage divider which provides a high level output in volts. The optional B circuit, provides a low level differential output in millivolts for compatability when used with other differential transducers.

The Houston Scientific 1850 SMM series position transducers have seen wide use with peerless performance in such markets as Oil technology, Automotive, Robotics, Die casting, Food processing, Medical, Nuclear, Rail transportation, Aircraft, Hydraulics, and Process control. Demanding service has also been required in many areas of the Nations Defense Programs. Such service showing the reliability of the product has ranged from aircraft control surface measurement, ordinance handling and testing, weapons deployment, launch vehicle stage separation and attitude control measurement. The Manned Space Program has employed Houston Scientific Model 1850 SMM position transducers among instrumentation used in support of the Shuttle Program. It is used in such areas as research and development, solid fuel core insertion, engine testing, gimble monitoring, launch umbilical retraction and landing gear positioning.

In independent laboratory tests the 1850 SMM series was found to meet and exceed shock loading of 100 G's for 11 milliseconds and vibration testing of 10 peak G's at 2000 Hz without deviation. In humidity tests, it was found to meet 95% RH at 75° F, and the maximum safe operating temperature range was determined to be -67° F to +257° F without deviation. Terminal velocities are precisely calculated and repeatedly verified on each displacement length through the use of an optically sensitive chronograph.

When having to make a choice consider this — A step by step evolutionary change has brought about the most durable and the most dependable service available. The average service life is ten years or more. Enclosures are available to survive the most abusive environments, along with options to fit most applications. Houston Scientific's proven service and calibration departments provide the highest degree of accuracy, resolution, repeatability all with NBS traceability in a design that facilitates in field serviceability. The choice is obvious!



SPECIFICATIONS:

General	
Range: (see note 1)	0-2 to 0-500 inches
Weight: (see note 2)	18 oz. (to 60 inches)
Case:	Aluminum
Sensing Element:	Hybrid-Conductive Plastic Potentiometer
Connector:	(K) PTO2A-10-6P
Mating Connector: (optional)	(K) PTO6A-10-6S
Electrical	
Input Resistance	
Circuit A	1000 Ohms
Circuit B	1.2K Ohms
Output Resistance:	
Circuit A	0-1000 Ohms (variable)
Circuit B	0.99K Ohms
Excitation Voltage:	20 volts max. AC or DC
Insulation Resistance:	50 meg ohms (min.) at 50 VDC
Performance	
Resolution:	Essentially infinite
Sensitivity:	(refer to table)
Thermal Effects: (see note 3)	
Zero	0.002%/ ° F
Span	0.002%/ ° F
Environmental	
Temperature range:	-67 to +257° F
Humidity:	95% RH at 75° F
Shock.	100 G's for 11 ms
Vibration:	10 G's at 2000 Hz

NOTES:

1 Refer to table for standard displacements
For displacements greater than those stated consult factory
2 For specific weights, consult factory
3 Over temperature range from -67 to +257° F
4 HSI reserves the right to change any and all specifications without notice for the purposes of technical or evolutionary advancement of the product for the customer

OPTIONS: (add to base price)

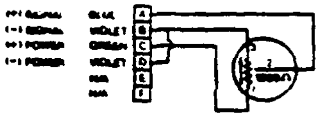
-01 Environmental Configurations
(D) Dustproof
(R) Ruggedized
-02 Cable Exits:
(SE) Side exit
(BE) Bottom exit
-03 Precision Linearity
(PL) Better than .05% FS.
-04 Circuit Configuration:
Differential bridge
(B) Bottom end zero
(C) Center zero
(D) Top end zero
-05 Acceleration:
(50G) High G acceleration
-06 Connector Position
(CT) Connector top
(CB) Connector bottom
(CE) Connector end

-10 Flying Lead Cable
(FL) Unit supplied with hard wired cable to specified length (less connector)
-11 Displacement Length:
(DL) User specification
-12 Break-Away Cable Kit
(BC) Over extension damage prevention (not for use on high G units)
-13 Bellows-Cable Protection.
(BP) Cable release damage prevention
-14 Explosion Proof
(EP) Consult factory
-15 Two Wire Transmitter:
(420) 4-20mA signal output

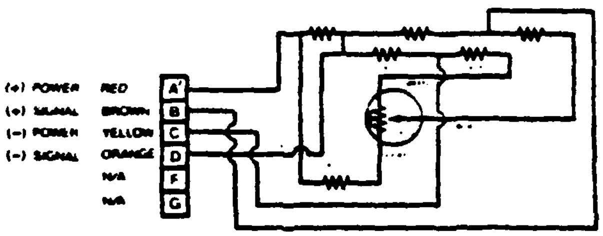
PERFORMANCE

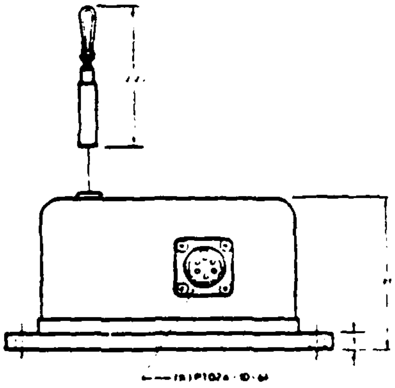
MODEL NO.	TRAVEL (inches)	OUTPUT (mv/v/inch)		CABLE TENSION (ozs.)	TERMINAL VELOCITY (in/sec.)										
		Ckt.A	Ckt.B			L1	L2	L3	W1	W2	A	B	D	H	T
1850-02	2	460	400	33	31	5.25	4 75	4 00	2 00	2 62	1 06	1 32	257	2 43	25
1850-05	5	184	471	24	76	5 25	4 75	4 00	2 00	2 62	1 06	1 32	257	2 43	25
1850-10	10	95	223	41	13	5 25	4 75	4 00	2 00	2 62	1 06	1 32	257	2 43	25
1850 15	15	64	160	24	21	5 25	4 75	4 00	2 00	2 62	1 06	1 32	257	2 43	25
1850-20	20	48	105	21	28	5 25	4 75	4 00	2 00	2 62	1 06	1 32	257	2 43	25
1850-30	30	32	072	16	46	5 25	4 75	4 00	2 00	2 62	55	1 32	257	2 43	25
1850-40	40	25	055	10	52	5.25	4 75	4 00	2 00	2 62	55	1 32	257	2 43	25
1850 50	50	19	042	8	60	5 25	4 75	4 00	2 00	2 62	55	1 32	257	2 43	25
1850-60	60	16	035	5	63	5 25	4 75	4 00	2 00	2 62	55	1 32	257	3 00	25
1850-80	80	12	026	39	116	6 50	5 75	5 00	3 37	4 75	68	1 79	257	3 80	25
1850-100	100	9	022	33	142	6.50	5 75	5 00	3 37	4 75	.68	1 79	257	3 80	25
1850-125	125	7 5	017	28	191	8 50	7 50	6 50	3 12	5 12	4 21	1 84	257	5 81	25
1850-150	150	6 4	014	42	224	8 50	7 50	6 50	3 12	5 12	4 21	1 84	257	5 81	25
1850-180	180	5 3	011	35	277	8 50	7 50	6 50	3 12	5 12	4 21	1 47	257	5 81	25
1850-200	200	4 8	010	24	305	9 50	8 75	8 00	4 25	5 25	1 06	4 18	257	4 00	25
1850 250	250	3 9	008	24	312	9 50	8 75	8 00	4 25	5 25	1 06	4 56	257	4 00	25
1850-300	300	3 1	007	32	321	9 50	8 75	8 00	4 25	5 25	1 08	4 89	257	4 00	25
1850 350	350	2 7	006	31	332	9 50	8 75	8 00	4 25	5 25	1 06	5 17	257	4 50	25
1850-400	400	2 4	005	31	349	9 50	8 75	8 00	4 25	5 25	1.06	5 45	257	5.25	25
1850-500	500	1 9	004	32	377	9 50	8 75	8 00	4 25	5 25	1 06	7 00	257	6 25	25

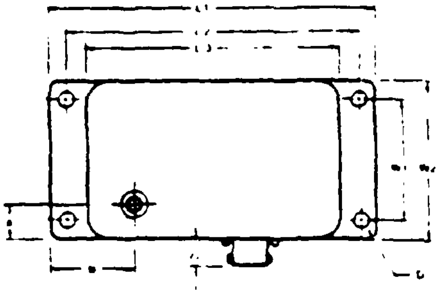
A CIRCUIT



B CIRCUIT

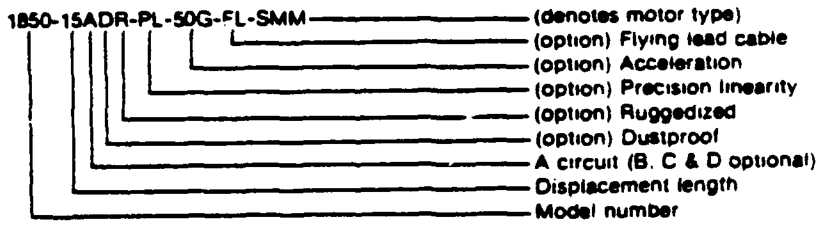






ORDERING INFORMATION:

Position transducers may be ordered from Houston Scientific by stating the Model number, the displacement length, and the parenthesised notations from the options list for any additional option requirements. See example below:



NOTE

A Please consult with factory when ordering multiple options to avoid conflicting requirements
B Mating connector (K)PTO6A-10-6S sold seperately
C Special designs and modifications available upon request
D Metric conversions available upon request
E All orders are subject to terms and conditions of HSI only

STATEMENT OF QUALITY ASSURANCE:

Houston Scientific is traceable to the National Bureau of Standards. Houston Scientific is also in compliance with the United States Government Military Standards MIL-I-45208, for Inspection systems, and MIL-C-45662, for Calibration systems, for the purposes of providing the highest possible quality to our customers.

Houston Scientific also provides the following precision products and services:

FORCE MEASUREMENT
Compression load cells
Sub-miniature force washers
Tension load cells
Precision load cells
Low capacity load washers

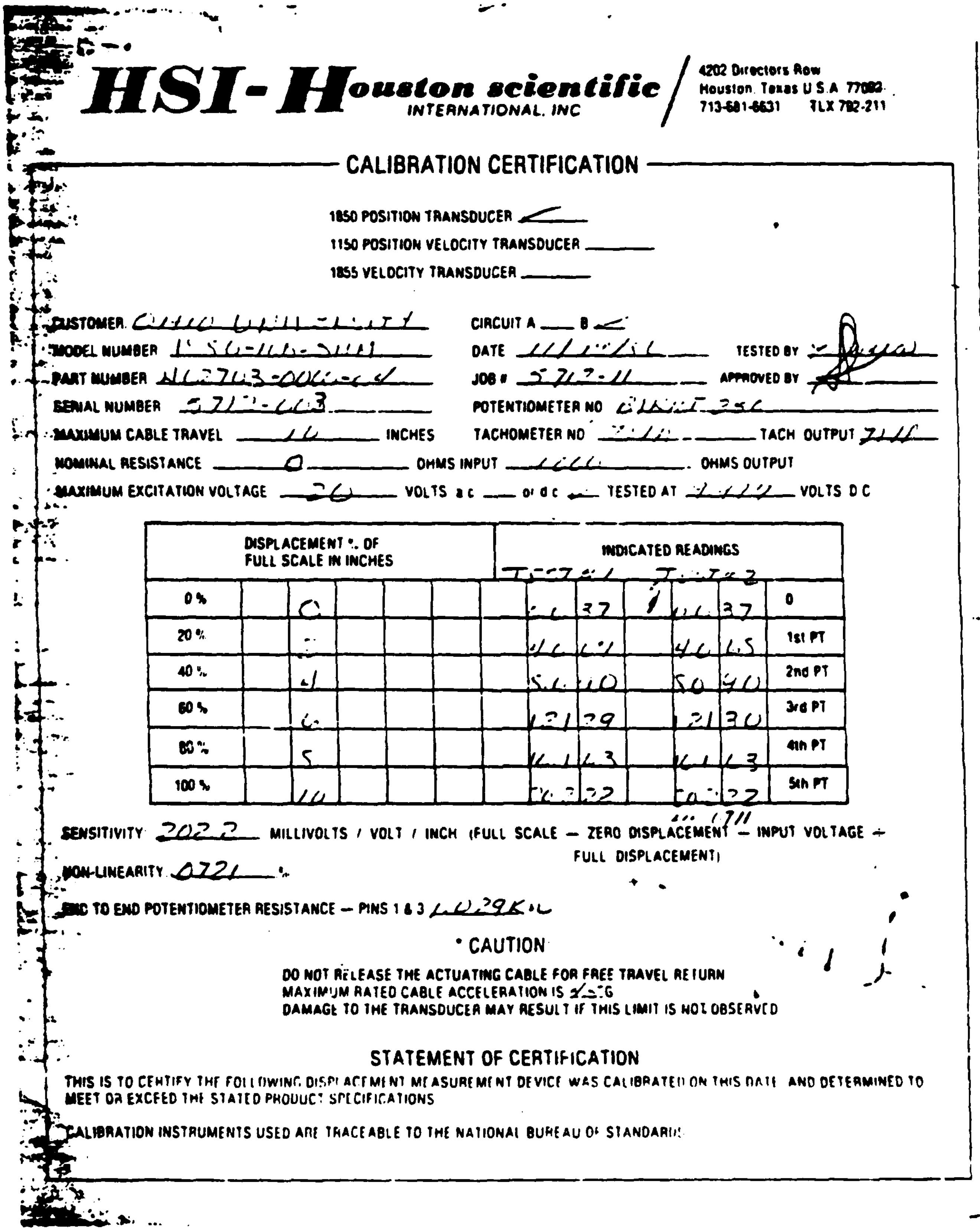
LINEAR MEASUREMENT
Velocity transducers
Position/velocity transducers
Digital transducers

SPECIAL SERVICES
Systems and applications engineering
Custom design services
Digital readouts and indicators
O.E.M. production

CALIBRATION SERVICES
Analysis and repair
Compression
Tension
Linear displacement
NBS traceability

HSI-Houston scientific
INTERNATIONAL INC
4202 Directors Row / Houston Texas 77092 USA
(713) 681-6631 / Telex 792 211

REPRESENTED BY



HSI - Houston scientific
INTERNATIONAL, INC

4202 Directors Row
Houston, Texas U.S.A. 77092
713-681-6631 TLX 792-211

CALIBRATION CERTIFICATION

1850 POSITION TRANSDUCER ✓
1150 POSITION VELOCITY TRANSDUCER ____
1855 VELOCITY TRANSDUCER ____

CUSTOMER Ohio University
CIRCUIT A ____ B ✓
MODEL NUMBER [illegible]
DATE [illegible]
TESTED BY [illegible]
PART NUMBER [illegible]
JOB # [illegible]
APPROVED BY
SERIAL NUMBER [illegible]
POTENTIOMETER NO [illegible]
MAXIMUM CABLE TRAVEL 10 INCHES
TACHOMETER NO [illegible]
TACH OUTPUT [illegible]
NOMINAL RESISTANCE 0 OHMS INPUT [illegible] OHMS OUTPUT
MAXIMUM EXCITATION VOLTAGE 26 VOLTS a c ____ or d c ✓ TESTED AT [illegible] VOLTS D C

DISPLACEMENT % OF FULL SCALE IN INCHES							INDICATED READINGS					
							[illegible]			[illegible]		
0 %		0					[illegible]	37	1	[illegible]	37	0
20 %		[illegible]					46	[illegible]		46	65	1st PT
40 %		4					56	10		50	90	2nd PT
60 %		6					121	29		121	30	3rd PT
80 %		5					61	63		61	63	4th PT
100 %		10					[illegible]	22		[illegible]	22	5th PT

SENSITIVITY 2022 MILLIVOLTS / VOLT / INCH (FULL SCALE — ZERO DISPLACEMENT .1711 — INPUT VOLTAGE ÷ FULL DISPLACEMENT)

NON-LINEARITY .0721 %

END TO END POTENTIOMETER RESISTANCE — PINS 1 & 3 10.29KΩ

*** CAUTION**

DO NOT RELEASE THE ACTUATING CABLE FOR FREE TRAVEL RETURN
MAXIMUM RATED CABLE ACCELERATION IS [illegible] G
DAMAGE TO THE TRANSDUCER MAY RESULT IF THIS LIMIT IS NOT OBSERVED

STATEMENT OF CERTIFICATION

THIS IS TO CERTIFY THE FOLLOWING DISPLACEMENT MEASUREMENT DEVICE WAS CALIBRATED ON THIS DATE AND DETERMINED TO MEET OR EXCEED THE STATED PRODUCT SPECIFICATIONS

CALIBRATION INSTRUMENTS USED ARE TRACEABLE TO THE NATIONAL BUREAU OF STANDARDS

HOUSTON SCIENTIFIC INTERNATIONAL INC.
4202 DIRECTORS ROW
HOUSTON, TEXAS
77092
(713) 681-6631
TELEX 792 211

SYSTEM CALIBRATION (POSITION TRANSDUCER)

INDICATOR MODEL #:	6270-4.5
INDICATOR SERIAL #:	5712-002
F.T. MODEL #:	1850-10B-SMM
F.T. SERIAL #:	5712-005
SHUNT CALIBRATION #:	26970

6 POINT CALIBRATION

APPLIED POSITION	INDICATED ON DISPLAY
0.000 INCHES.	00.000 INCHES.
2.000 INCHES.	2.000 INCHES.
4.000 INCHES.	3.990 INCHES.
6.000 INCHES.	5.990 INCHES.
8.000 INCHES.	7.990 INCHES.
10.000 INCHES.	10.000 INCHES.

CALIBRATED BY: Bob D. Frazier

BOB D.FRAZIER

11/18/86

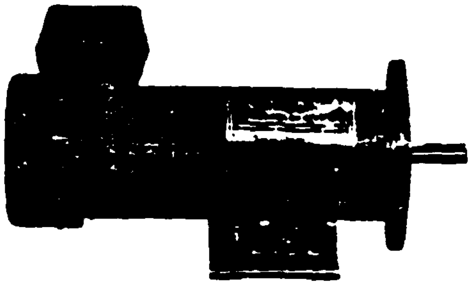
Appendix 6

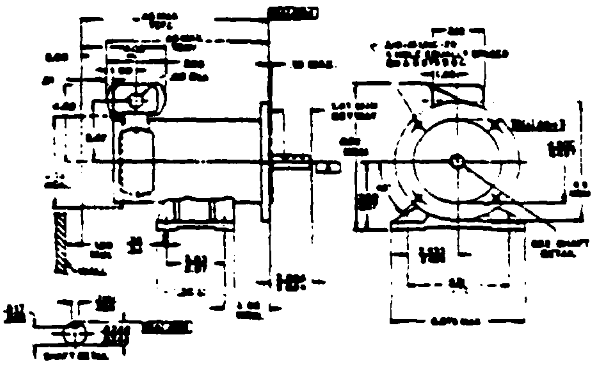
Motor and Motor Controller Specifications and Manual

SERIES M-46
PERMANENT MAGNET DC MOTORS
1/8 - 1 HP

ELECTROL

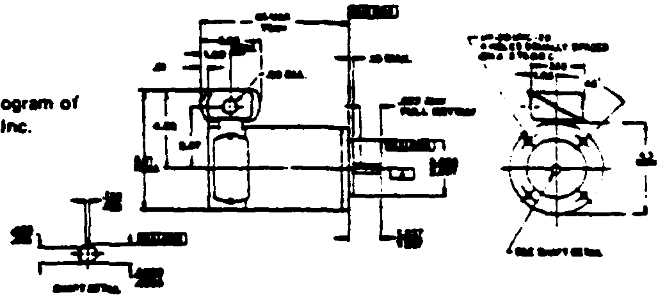
†Drawing A





Factory and Warehouse Stocks

††Drawing B



Recognized under Component Program of Underwriters Laboratories, Inc.

SERIES M46 - 1/8 to 1 HP - 90 AND 180V. ARM. - 56C FLANGE WITH REMOVABLE BASE

Part No.	HP	Armature Volts	Base RPM	Full Load Amps	Duty	Enclosure	Length Ins. AG Max	Wt. Lbs.
† *M-4610NV	1/8	90	1725	1.45	CONT	TENV	7.19	15
† *M-4611CZ	1/6	90	1725	1.71	CONT	TENV	7.19	15
† M-4612B	1/4	90	1725	2.85	CONT	TEFC	8.81	17
† M-4612BNV	1/4	90	1725	2.85	CONT	TENV	7.88	17
† M-4612ABNV	1/4	180	1725	1.41	CONT	TENV	7.88	17
† M-4613B	1/3	90	1725	3.5	CONT	TEFC	10.56	25
† M-4613BNV	1/3	90	1725	3.5	CONT	TENV	8.69	25
† M-4613AB	1/3	180	1725	1.71	CONT	TEFC	10.56	25
† M-4616B	1/2	90	1725	5.35	CONT	TEFC	10.56	25
† M-4616BNV	1/2	90	1725	5.35	CONT	TENV	11.69	25
† M-4616AB	1/2	180	1725	2.85	CONT	TEFC	10.56	25
† M-4619B	3/4	90	1725	8.1	CONT	TEFC	12.56	32
† M-4619AB	3/4	180	1725	3.5	CONT	TEFC	12.56	32
~~[illegible]~~	~~1~~	~~90~~	~~1725~~	~~10.4~~	~~CONT~~	~~TEFC~~	~~14.56~~	~~[illegible]~~
† M-4621AB	1	180	1725	5.35	CONT	TEFC	14.56	39

*Not supplied with base, base can be purchased seperately.

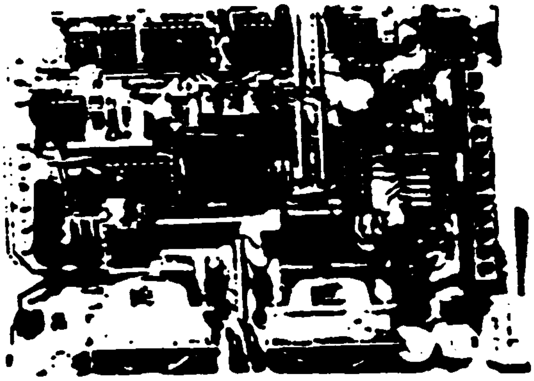
ELECTROL

OPERATING INSTRUCTIONS FOR OPTION A42

Process Control Interface Board

P/N 9084 #1081 Circuit

Effective
December 31, 1984



ELECTROL process control with speed profile capabilities has standard input command of 0 - 10VDC, or 4 - 20 MA DC. Other inputs from 0 - 200VDC or 0 - 45 MADC are selectable via dip switch and trim adjustment pots. Isolation is provided as standard. Output is adjustable 1 - 13.5VDC and AC power input is selectable for 115VAC or 230VAC for easy field application. Input impedence: Current 475 ohm, voltage 10K - 200K ohm.

Tools required.

1. Very small straight blade screwdriver
2. Digital voltmeter set on 2 to 20VDC scale

The process control P.C. Board is a very precision control. Readings are set in .01 of a volt. Precise measurements are required to set this P.C. Board up. By following the procedure below, you will find an exact set-up is relatively easy,

1. Pot Description

   Entry Pot
   Exit Pot
   Entry Speed
   Exit Speed
   Set Point Pot
   Set Point Hysteresis Pot
   Gain Pot

2. Test Posts have been installed for simple testing

1		1 +15VDC ± 5VDC
2	Are Supply #1	2 Common 1 and 3
3		3 - 15VDC ±5VDC

If an oscope is available, ripple on all supplies should be below 70 MV.

4		4 +15VDC ± .5V
5	Are Supply #2	5 Common
6		6 -15VDC ± .5V

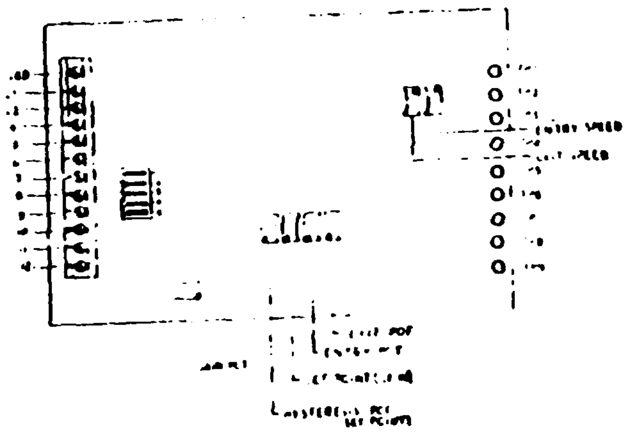
TP7 Voltage output scaled 0 to 9.00 VDC. This test point should not be set above 9.00 VDC because the next step in the circuit is A to D conversion of 0 to 10 KHZ with 0 to 9.00 input.

TP8 0 to 10 KHZ Square wave with 0 to 9.00 at TP7.

TP8 0 to 10 KHZ Square wave with 0 to 10 KHZ at TP8. This is the isolated command so 100% of information is transferred with no loss of accuracy.

TP10 This test pin is where you monitor the output. Adjusted to 13.5V DC MAX volts.

ADJUSTMENT PROCEDURES



1. First set up all trim pots as follows.

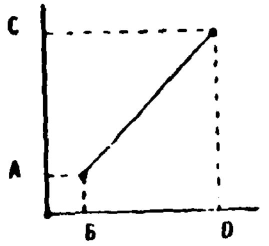
   Entry Pot - CCW
   Exit Pot - CW
   Exit Speed Pot - CCW
   Entry Speed Pot - CW
   Set Point - CCW
   Hysteresis - CCW
   Gain - CCW

2. Select input on chart and set switches as shown:

SW Position X = Closed

INPUT	1	2	3	4	5
4 - 20 ma					X
0 - 50 ma				X	X
0 - 10V					
0 - 100V		X	X		
0 - 200V	X		X		

3. Look at and complete speed profile chart as follows.



Control command - 0 - 10V DC.

Input command - 4 - 20, 0 - 10.

A. Lowest start input ______

B. Speed of control at A input command ______

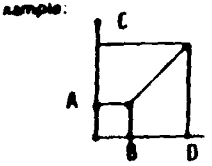
C. Highest command ______

D. Speed of control at C input command ______

Continued

3. Continued

Example:



Using 4 to 20 ma control
1. All pots set
2. 4 to 20 ma - close sw #5.
3. Fill sample chart

A. Lowest input 4 ma
B. Speed of control at A input command 0 speed
C. Highest command - 20 ma
D. Speed of control at B input command 1800 RPM or 90V DC to motor - 10 V DC output

4. Now, with chart filled out (following example 4 to 20 ma):

Step 1. Set input to minimum 4 ma into control. Slowly adjust entry pot (adjust CW) until tri-state light turns green

Step 2. Set voltage at TP #7 to B of adjustment chart (0 speed) with pot entry speed (adjust CCW) to set in minimum voltage to offset any required minimum speed of any drive. Example: Running A 200 drive requires a minimum pot setting of 1.00V DC so motor will turn at slightest request, so set entry speed pot (TP#7, TP#5 common) to 1.00 V DC

Step 3. Select highest input command - 20 ma. Slowly adjust exit pot (CCW) until tri state LED turns red.

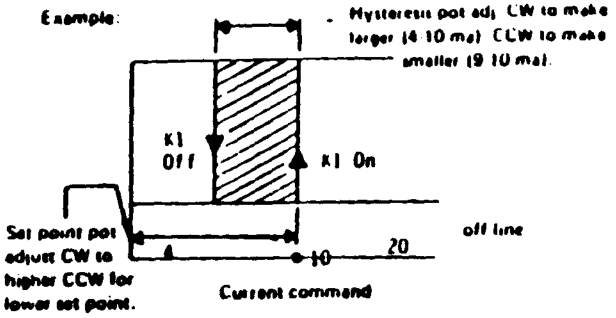
Step 4. With a (20 ma) command (tri state LED on red), set exit speed (adj. CW) set output voltage (from TP#5 common to TP#7) to 9.00V DC. This sequence had adjusted the front end of control.

Step 5. Now make command 4 to 20 go up and down in scale and watch tri state light. At 4 ma light will turn green above 4.1 light is out when you hit 20 ma light should turn red. If this does not happen, repeat Steps 1 through 4

Step 6. Adjust gain pot with 20 ma input on PC #1091 so that motor runs at full speed

Step 7. A set point relay output is available for customer usage. This set point moniters the input command, so and adjustment can be set the full range of that input

Example:



Hysteresis pot adj. CW to make larger (4-10 ma). CCW to make smaller (9-10 ma).

Set point pot adjust CW to higher CCW for lower set point.

Step 8. (Continued) Set point pot can be set full range (Adjust CW) of input (4 to 20 ma). Example: set point at 10 ma because customer does not want motor to ever be below 10 ma. Relay K1 will stay on as long as input command is above 10 ma. When command goes below 10 ma K1 will shut off (opening K1 contacts).

If the customer wants K1 on at 10 ma but does not want to shut off until 8 ma, the hysteresis pot may be set up for this. Adjust hysteresis CW. This opens up the on-latch of K1 shaded area.

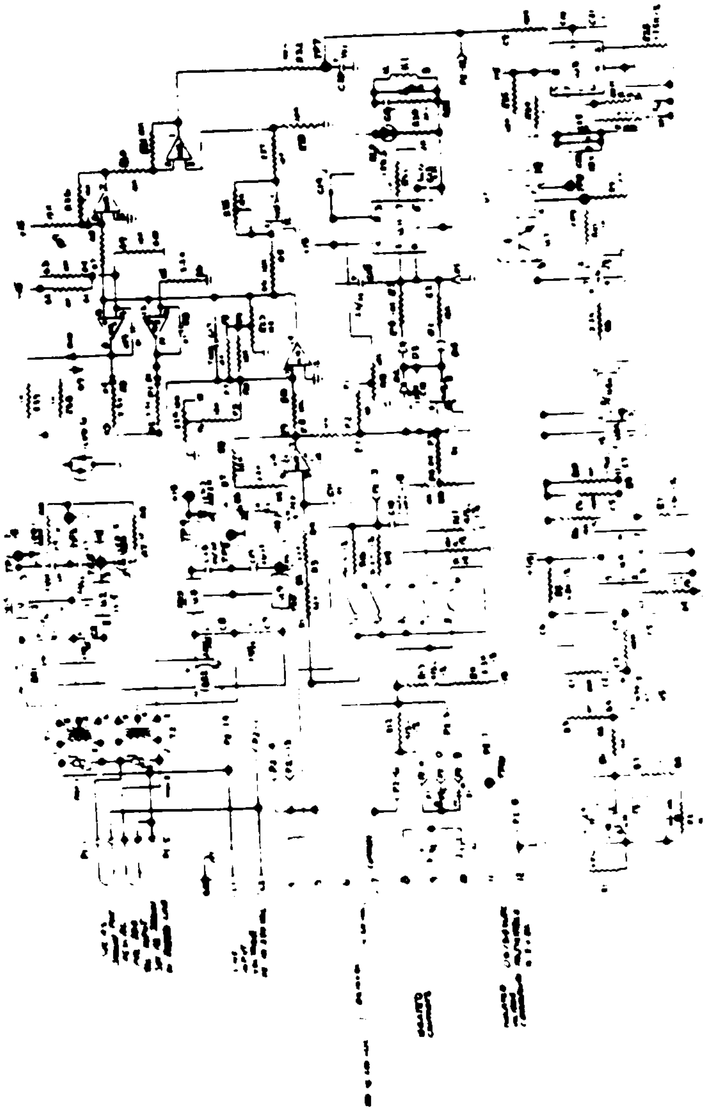
Step 9: PC #1091 is now ready to run controls.

Step 10: Special Ramps can be accommodated by simply setting up command on charts. Reverse ramp or any portion ofcontrol signal may be used. Example:

0 to full speed with 6 to 18 ma
50% to 100% speed with 4 to 20 ma
50% to 80% with 8 to 14 ma.
100 to 50% with 4 to 20 ma
80% to 30% with 10 to 18 ma.

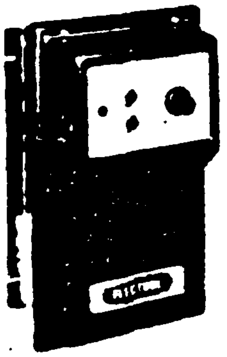
Any input for almost any output

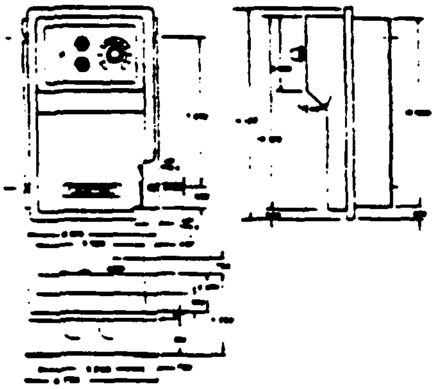
Step 11: For questions contact your area Representative or the Factory.



ELECTROL

INSTALLATION & MAINTAINENCE MANUAL
DC MOTOR CONTROLLER – D-TROL
MODEL C-MH-W/O-754-E
MODEL C-MH-13-754-CM





Standard Features

- Dual Input Voltage – 115/230V AC, 50/60 Hz., Single Phase/ adjustable with two PC mounted fast-on connectors
- Dual Output Voltage – 0-90/0-180V DC PM or Shunt Wound Motors (Field Voltage 100/200V DC)
- Horsepower Range – Multi-HP 1/8 - 1 HP 90V, 1/4 - 2 HP 180V
- Speed Regulation – 2% of Base Speed
- Speed Range – 30:1 Constant Torque
- Full Wave Rectification
- Minimum
  Speed Adjustment – Sets low end speed limit
- Maximum
  Speed Adjustment – Sets high end speed limit
- IR Compensation – Adjustable No Load to Full Load Motor RPM
- Torque (Current Limit) – Adjustable maximum current cut-off
- Built-in Transient and Surge Protection
- Built-in Line Voltage Compensation
- Fuse Protection – Line and motor
- On-Off Switch
- Master Speed Pot
- Power "On" Indicator Light
- NEMA 12 Enclosure – Attractive plastic lid with metal bottom

ELECTROL CO., INC. • 321 DEWEY ST. • YORK, PA 17404 • (717) 848-1722

WARRANTY

ELECTROL controls are warranted by ELECTROL CO., INC to the original user against defects in workmanship or materials under normal use (rental excluded) for one (1) year after purchase

Any part which is determined to be defective in material or workmanship must be returned to ELECTROL head quarters or an authorized service center, as ELECTROL designates, shipping costs prepaid. The control will be repaired or replaced at ELECTROL's option. Expenses incurred by buyer in repairing or replacing any defective product will not be allowed except where authorized in writing and signed by an officer of the company.

APPLICATION INFORMATION

1. If you replace an AC induction motor with a DC motor and adjustable speed drive, consideration must be given to the full load torque rating of the AC induction motor that is being replaced. The full load torque rating of the DC motor must be equal to or greater than the full load torque rating of the AC motor it is going to replace.
2. When replacing an AC induction motor with a DC motor and adjustable speed control the DC motors starting torque must be limited to 200% of full load torque (150% of full torque for gearmotors). The reason for these limits is to protect the motor or gear motor from damaging overloads. Cyclic type loads should be avoided
3. Soft Start – The DC motor accelerates from 0 to full load RPM smoothly and takes 1 to 3 seconds to reach full load RPM. Acceleration rate varies with respect to speed setting and amount of inertia in the system.

4. The motor controller has circuitry to protect it from normal line surges, and transients. If, however, the control will be used in an environment where these are present constantly, such as high frequency welding equipment, an isolation transformer or other line filtering device should be used
5. The Electrol adjustable speed DC motor control is designed for use on constant (or diminishing) torque applications such as conveyors, fans, blowers, etc.

   WARNING NOT INTENDED FOR USE WITH SAWS, DRILL PRESSES, OR OTHER CONSTANT HP APPLICATIONS. NOT TO BE USED IN AN EXPLOSIVE ATMOSPHERE!.

   If your control is equipped with DYNAMIC BRAKING, the following applies. *Use only on motors up to 1 HP.*

   Dynamic Braking functions in the control when the FWD/BRAKE/REV switch is moved to the BRAKE position while the motor is running. This allows the motor to come to a quick smooth stop.

   NOTE: Dynamic braking resistors are sized to function on the basis of *no appreciable external inertia*. The following is the maximum allowable motor starts and stops:

   1/6 - 1 HP DC motors – 5 per minute max.

CONNECTION

CAUTION: Disconnect power source before connecting controller or motor. Use No. 12 AWG (minimum size) wire for controller input lines, and for interconnection lines between controller and motor.

Make connections to the controller and the motor in accordance with the Connection Chart. The controller terminal strip is located inside the controller enclosure. To reach the terminal strip, loosen the captive screw in the top of the controller front panel, then swing the panel open. To feed wiring to the terminal strip, remove the two button plugs from the bottom of the controller enclosure.

CUSTOMER CONNECTION AND ADJUSTMENTS

CAUTION: Follow local electrical codes and proper electrical practices during hook up of controller. The customer is responsible for supplying and connecting an external power disconnect, such as a 20 Amp circuit breaker or DPDT toggle switch. Disconnect power source before connecting control and motor. Use a 12 gauge wire for input lines to the control and lines to motor armature. The control features P.C. mounted line and motor fuses, power on indicator LED and motor start stop push button switches. Turn power off at external disconnect when control is not in use.

TERMINAL BLOCK CONNECTIONS:

Customer Hook up	GND	Earth Ground
	L1 L2	Single Phase AC Input
	A1 A2	Motor Armature
	F1 F2	Motor Field (Shunt Wound)
Factory Hook up.	5 6 7	5K ohm Master Speed Pot
	1 4	*L.E.D. Indicator (Power On)*
	1 3	Start Push Button Switch (NO)
	2 3	Stop Push Button Switch (NO)

ADJUSTMENTS.

Controls are shipped set up and adjusted for 2 HP, 230VAC. If any other horsepower and/or voltage is desired, follow the instructions below.

A. Horsepower Selection

The DIP Switch on the PC Board is for horsepower selection.

ALL SWITCH POSITIONS OPEN EXCEPT
FOR HORSEPOWER DESIRED.

HP			FUSE	
115V	230V	CLOSE SWITCH	LINE	ARM.
1/6	1/4	1	3	3
1/4	1/2 3/4	2	5	5
1/3 1/2	1	3	10	10
3/4	1 1/2	4	15	10
1	2	5	20	15

B. Voltage Selection:

For 115V Connect both P.C. Jumpers to terminal A
Fo· 230V Connect both P.C. Jumpers to terminal B

C. Start up procedures.

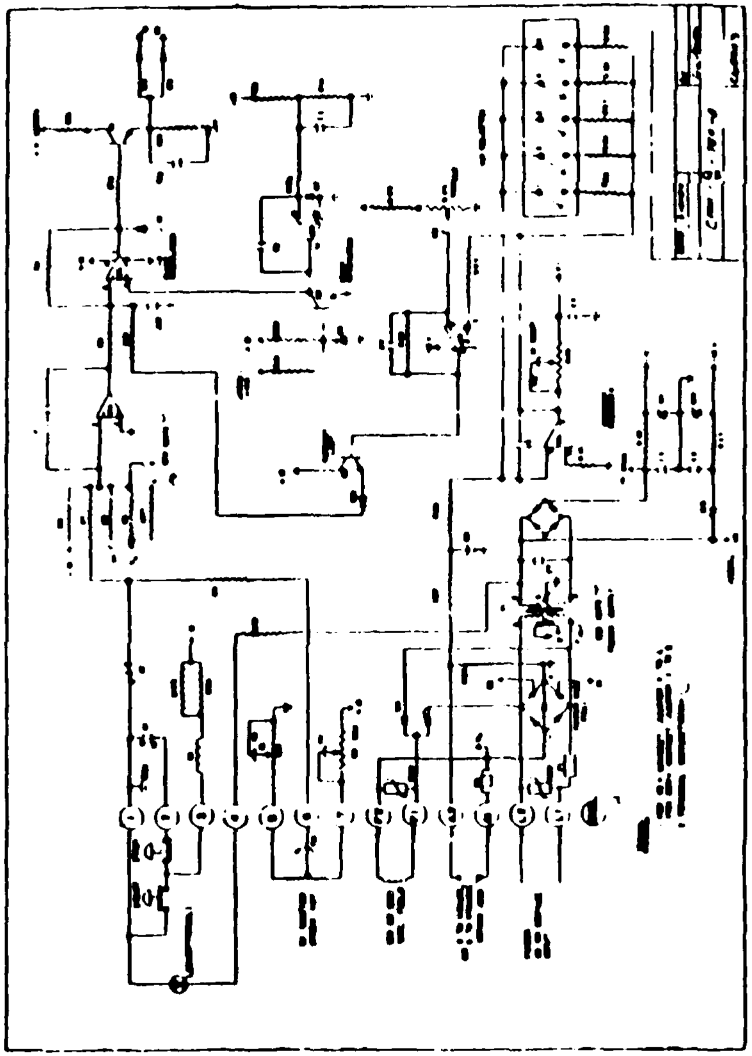
1. Set master speed pot to 0%.
2. Apply power to unit and select "Start" Switch position.
3. Turn speed pot up and check for proper rotation of motor shaft. Reverse motor leads to change rotation, if necessary.
4. Trim pot adjustments, if necessary.
   a) MIN RPM Trim: To adjust master pot low end speed range, turn CCW to decrease speed range. Turn CW to increase speed range
   b) MAX RPM Trim: To adjust master pot high end speed range, turn CCW to decrease speed range. Turn CW to increase speed range
   c) TORQUE Trim. To adjust *maximum current* available to motor armature, *do not exceed* full load current of motor.
   d) IR COMP: To maintain no load motor RPM with load applied, turn CW to increase compensation. Turn CCW to decrease compensation. CAUTION Overadjustment will cause motor RPM at low speed settings to rise excessively under full load conditions.

D. Dynamic Braking and Manual Reversing (MRDB) options:

When specifying the Manual Reversing Dynamic Braking (MRDB option), note the Forward/Brake/Reverse switch is rated for 3/4 HP. Max. If the master speed pot is set at zero and motor completely stops, the MRDB switch may be used for 1 HP.

The Dynamic Braking Resistors on the standard options are intended for intermittent operation. If continuous braking operations are necessary, a larger Dynamic Braking resistor will be required.

See options A 19 and A 20 in ELECTRO LINE Catalog for additional information.



Appendix 7

Dash-8 Manual

DASH-8 MANUAL

MBC MetraByte Corporation

5/84

WARRANTY

All products manufactured by MetraByte are warranted against defective materials and workmanship for a period of One Year from the date of delivery to the original purchaser. Any product that is found to be defective within the warranty period will, at the option of MetraByte, be repaired or replaced. This warranty does not apply to products damaged by improper use.

! WARNING !

MetraByte Corporation assumes no liability to damages consequent to the use of this product. This product is not designed with components of a level of reliability suitable for use in life support systems.

MetraByte Corporation, 440 Myles Standish Boulevard, Taunton
Mass. 02780 U.S.A. Phone: (617) - 880 - 3000 Telex: 503989

Chapter 1

INTRODUCTION

1.1 SUMMARY OF DASH-8 FUNCTIONS.

MetraByte's DASH-8 is an 8 channel 12 bit high speed A/D converter and timer/counter board for the IBM P.C.[1] The DASH-8 board is 5" long and can be fitted in a "half" slot. All connections are made through a standard 37 pin D male connector that projects through the rear of the computer. The following functions are implemented on the DASH-8:-

1. An 8 channel, 12 bit successive approximation A/D converter with sample/hold. The full scale input of each channel is +/-5 volts with a resolution of 0.00244 volts (2.44 millivolts). Inputs are single ended with a common ground and can withstand a continuous overload of +/-30 volts and brief transients of several hundred volts. All inputs are fail safe i.e. open circuit when the computer power is off. A/D conversion time is typically 25 microseconds (35 microseconds max.) and depending on the speed of the software driver, throughputs of up to 30,000 channels/sec are attainable.

2. An 8253 programmable counter timer provides periodic interrupts for the A/D converter and can additionally be used for event counting, pulse and waveform generation, frequency and period measurement etc. There are three separate 16 bit down counters in the 8253. One of these is connected to a submultiple of the system clock, and all I/O functions of the remaining two are accessible to the user. Input frequencies up to 2MHz can be handled by the 8253[2]

3. 7 bits of TTL digital I/O are provided composed of one output port of 4 bits and one input port of 3 bits.

----

[1] Registered trademark of International Business Machines Corporation.

[2] Contact MetraByte for higher input frequency requirements

4. 1 precision +10.00v (+/-0.1v) reference voltage output is derived from the A/D converter reference. This output can source/sink 2mA.

5. An external interrupt input is provided that can select any of the IBM P.C. interrupt levels 2-7 and allow user programmed interrupt routines to provide background data acquisition or interrupt driven control. The DASH-8 includes status and control registers that make interrupt handshaking a simple procedure. The interrupt input may be externally connected to the timer/counter or any other trigger source.

6. IBM P.C. buss power (+5, +12 & -12v) is provided along with all other I/O connections on the rear connector. This makes for simple addition of user designed interfaces, input signal conditioning circuits, expansion multiplexers etc.

The following utility software for DASH-8 is provided on a single sided PC-DOS 1.10 format 5-1/4" floppy disk (upward compatible with DOS 2.0):-

1) A machine language I/O driver (DASH8.BIN) for control of A/D, timer and digital I/O channel functions via BASIC CALL. The I/O driver can select multiplexer channels, set scan limits, perform software commanded A/D conversions, interrupt driven conversions and scans, set and read the timer counter and measure frequency and period. A source listing is available as an optional item (SLD-08) for the assembly language programmer.
2) Data linearization.
3) Initial setup and installation aids.
4) Graphics package for display of processed data[3].
5) Calibration and test programs.
6) Examples and demonstration programs.

Using state of the art data conversion components, the DASH-8 has been designed to provide a powerful and inexpensive analog/digital interface on a single half slot board. It is ideally suited to any application requiring high speed 12 bit data acquisition at the lowest possible cost. The freedom from complexity and the I/O mapped control make programming straightforward. Applications include data logging, process control, signal analysis, robotics, energy management, product testing, digitizers and touch screens, laboratory and medical instrumentation etc. A system block diagram appears in Fig. 1.1.

To extend the capabilities of DASH-8 the following expansion modules can be connected via flat insulation displacement cable to the main 37 pin D I/O connector:-

----

[3] Color graphics board required.

1) SCREW TERMINAL CONNECTOR BOARD - All I/O lines on the rear connector are connected to miniature screw terminal connectors. The digital I/O port lines are monitored by L.E.D.'s and a small breadboard area with +/-12v & +5v power is available for amplifiers, filters, and other user supplied circuits. The screw terminal connector board is MetraByte part number STA-08.

2) EXPANSION MULTIPLEXER AND INSTRUMENTATION AMPLIFIER The EXP-16 multiplexes 16 differential inputs to a single analog output suitable for connection to any of the analog input channels of DASH-8. EXP-16 boards are cascadable so that up to 8 EXP-16 boards can be attached to a single DASH-8 providing a total of 128 channels. The expansion multiplexer board includes a low drift instrumentation amplifier with preselected switchable gains of 0.5,1,2,10,50,100,200 or 1000 (other gains can be resistor programmed). A cold junction correction sensor is also included for software compensation of thermocouples which can be directly connected to EXP-16. Open thermocouple detection hardware is also included.

3) THERMOLAB - This is a software package intended for use with DASH-8 and up to 7 EXP-16('s) that provides up to 112 channels of thermocouple temperature measurement. The temperature measurement system can accomodate J,K,T,E,S, B, and R type thermocouples and different types may be connected to different EXP-16 expanders and in some cases to the same expander. The supplied software and hardware performs cold junction compensation, linearization and open thermocouple detection.

4) LABTECH NOTEBOOK - A powerful software package that allows you to perform many functions of the laboratory using a single set of tools. Data acquisition and process control, storage and screen display of experimental data, data manipulation and curve fitting are all placed at your fingertips. All commands are menu driven and there is no need to remember command names or sequences. Data files generated by LABTECH NOTEBOOK[4] are formatted for direct link to LOTUS 1-2-3[5] providing the full analytical power and graphing functions of LOTUS to your data reduction. LABTECH NOTEBOOK is strongly recommended for anyone involved in repetitive experimental data recording and analysis, or those who wish to minimise programming

----

[4] LABTECH NOTEBOOK is a trademark of Laboratory Technologies Corporation, 328 Broadway, Cambridge, Mass. 02137

[5] LOTUS 1-2-3 is a trademark of Lotus Development Corporation

involvement in using DASH-8.

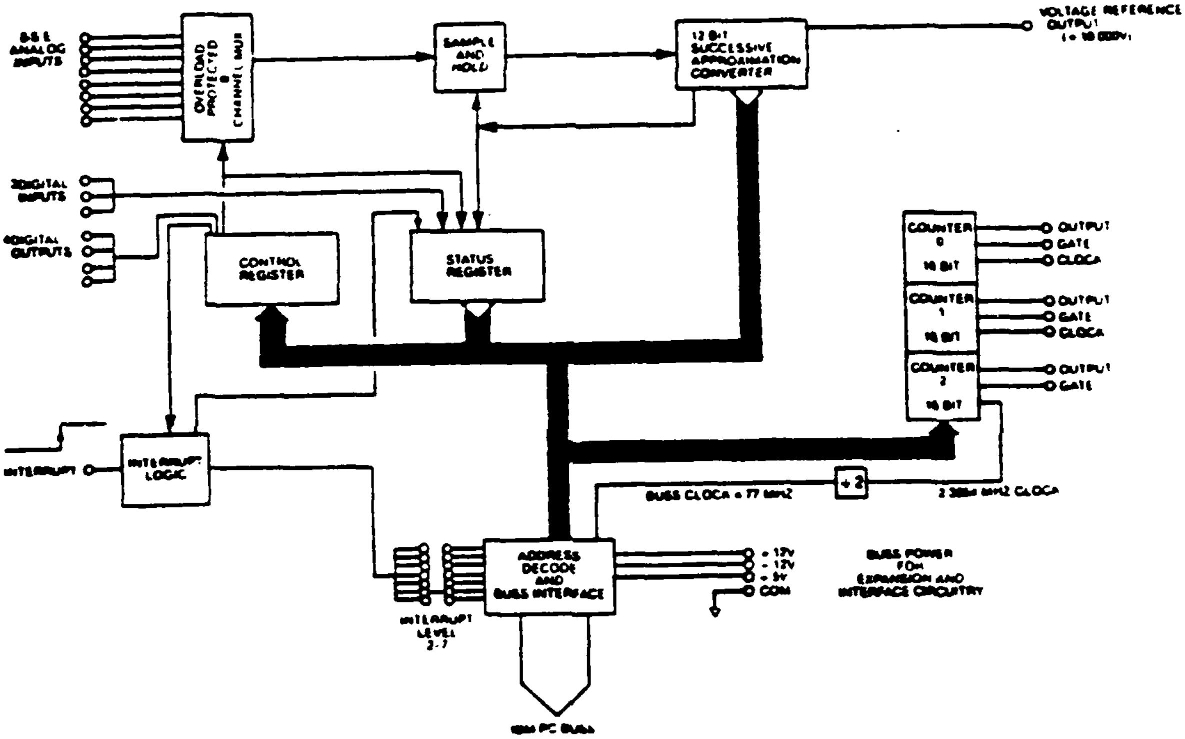


Fig. 1.1 BLOCK DIAGRAM OF DASH - 8

Chapter 2

INSTALLATION

2.1 BACKING UP THE DISK

The software supplied with DASH-8 is in DOS 1.10 single sided (160K) format which is compatible with double sided and DOS 2.0. It is good practice to make a back up copy before using the software. You can make a straight back up using DISKCOPY but since you require Basic & PC-DOS to run the programs, it will save time if you start with your DOS disk and format a new disk with the FORMAT /S option. Next copy BASICA.COM onto the backup from your I.B.M. DOS disk using COPY and finally use COPY *.* to copy the DASH-8 disk files onto the backup.

In step by step sequence:-

Insert DOS disk in drive A. Type:-

```
A> FORMAT /S
```

Insert new disk for backup copy as prompted by format. Next when the DOS prompt returns, (on single drive system replace back up with DOS disk) type:-

```
A> COPY BASICA.COM B:
```

Insert DASCON-1 master disk and type:-

```
A> COPY *.* B:
```

2.2 HARDWARE INSTALLATION

DASH-8 requires 8 consecutive address locations in I/O space. Some I/O address locations will be occupied by internal I/O and your other peripheral cards, so to provide flexibility in avoiding conflict with these devices DASH-8's I/O address can be set by the Base Address D.I.P. switch to be on an 8 bit boundary anywhere in the I.B.M. P.C. decoded I/O space. This I/O address space extends from decimal 256-1023 (Hex 100-3FF) which is many times larger than is ever likely to be fully occupied. Such a large space also allows use of more than one DASH-8 in a single computer. Summarising the I/O address map on page 2-23 of the "IBM Technical Reference Manual":-

ADRESS HEX RANGE	DEVICE
000-0FF	All used by internal I/O
200-20F	Game I/O adapter
278-27F	Reserved
2F8-2FF	Reserved
320-32F	Hard disk drive (P.C. XT only)
378-37F	Parallel printer port
3B0-3BF	IBM monochrome display & parallel printer adapter
3F0-3F7	5 1/4in. disk drive adapter
3F8-3FF	Asynchronous communications adapter.

This covers the standard I/O options, but if you have other I/O peripherals e.g. hard disk drives, special graphics boards, prototype cards etc. they will also be sharing I/O address space. Memory addressing is separate from I/O addressing so there is no possible conflict with any add-on memory.

Usually, a good choices is to put the DASH-8 at base address Hex &H300,&H308 or &H310 (Decimal 768,776,784). (Note if you are using an IBM prototype board, it uses the Hex 300-31F address space and would conflict, &H330 or &H340 would be a good choice in this case). As an aid to setting the base address D.I.P. switch located just to the left above the gold plated edge connector, type:-

```
A> BASICA INSTALL
```

When you get the "Desired base address?" prompt, type in your choice in decimal or IBM &H--- format and press return. The program will round your address to the nearest 8 bit boundary, check for possible conflicts with standard IBM I/O devices (and warn you if so) and draw a picture of the correct positions of the toggles on the base address DIP switch. If you like to understand the details, see Fig. 2.1 for an explanation of base address switch settings.

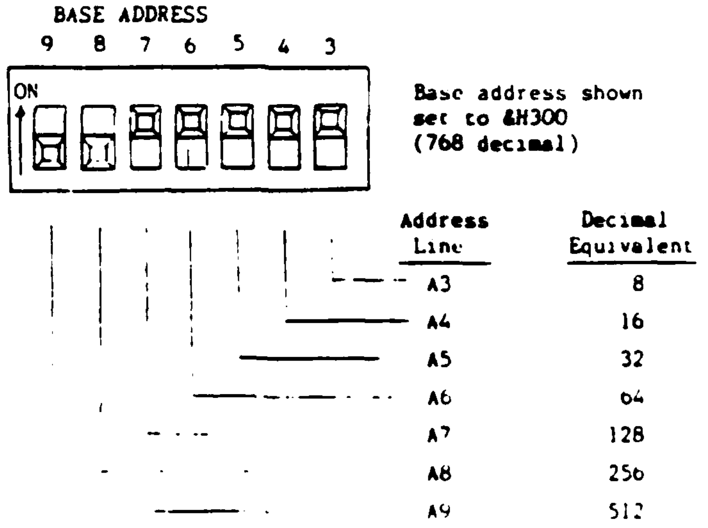
INSTALL.BAS performs one further optional function. You can generate a file named DASH8.ADR which contains the base I/O address that you have selected. If your application programs read this file instead of declaring the address in each program (list LOADCALL.BAS to see how it's done), then should you wish to change the board address in the future, all you have to do is alter the DASH8.ADR file instead of altering dozens of application programs. Your DASH8 disk comes with a DASH8.ADR file loaded with decimal 768 (Hex 300) which of course will be overwritten if you choose to generate another address file when you run INSTALL.

The next step is to remove the DASH-8 board from its protective electrostatic packaging and set the base address DIP switch. It is a good precaution to discharge any electrostatic charge you may have accumulated by touching the metal frame of your computer. Also, at this stage, check that the interrupt level selection jumper on header J2 is in the rightmost (X = inactive) position.This makes certain that the DASH-8 cannot initiate an interrupt until you are fully familiar with the requirements of interrupt operation (see Chapter 3).

TURN OFF THE POWER on your computer and remove the case (See IBM "Guide to Operations" pages 5.5 & 5.6 if you are not already expert at this maneuver). Remove a vacant back plate by undoing the screw at the top and plug the DASH-8 in and secure the backplate. DASH-8 *will fit in any of the regular full depth slots of the IBM P.C. or the "half" slots of the IBM X.T. or Portable computer.* Installation is now complete. You may plug any of the DASH-8 accessories or your own cable into the 37 pin D connector on the rear.

Remember, TURN OFF THE POWER whenever installing or removing any peripheral board including the DASH-8. Failing to observe this precaution can cause costly damage to the electronics of your computer and/or the DASH-8 board.

If for any reason you later remove the DASH-8 board, MetraByte recommends that you retain the special electrostatically shielded packaging and use it for storage.



Base address shown set to &H300 (768 decimal)

Address Line	Decimal Equivalent
A3	8
A4	16
A5	32
A6	64
A7	128
A8	256
A9	512

Switches have decimal values as above in the 'off' position. In the 'on' position, decimal value is zero.

FIG. 2.1 BASE I/O ADDRESS SWITCH SETTING

Chapter 3

PROGRAMMING

3.1 PROGRAMMING DASH-8

At the lowest level, DASH-8 is programmed using I/O input and output instructions. In BASIC these are the INP (X) and OUT X,Y functions. Assembly language and most other high level languages have equivalent instructions. Use of these functions usually involves formatting data and dealing with absolute I/O addresses. Although not demanding, this can require many lines of code and necessitates an understanding of the devices, data format and architecture of the DASH-8. To simplify program generation, a special I/O driver routine "DASH8.BIN" is included in the DASH-8 software package. This may be accessed from BASIC by a single line CALL statement. The various operating modes of the CALL routine select all the functions of the DASH-8, format and error check data, and perform frequently used sequences of instructions. An example is Mode 4 which performs a sequence of operations required to perform an A/D conversion, check A/D status, read data and increment the multiplexer, checking whether the upper scan limit has been exceeded and if so, restoring the lower limit. A routine to perform this operation in BASIC using INP's and OUT's would require many lines of code, be rather slow and very tedious to program. All these problems can be circumvented by:-

```
xxx10  MD% = 4  'select mode
xxx20  CALL DASH8 (MD%, DIO%, FLAG%) 'does A/D conversion
                                      and returns data in
                                      DIO%, errors in FLAG%
```

Obviously the DASH8.BIN driver greatly reduces programming time and effort. The driver also supports data collection on interrupt from an external source or the DASH-8 timer/counter. Note that BASIC has no interrupt processing functions and interrupt (or background) data collection is only available using the CALL routine.

Both methods of programming using INP and OUT functions and the CALL routine are described and you are free to choose either although usually the BASIC programmer will find the CALL routine much simpler to implement.

3.1.1 I/O ADDRESS MAP OF DASH-8

First of all let's take a look at the I/O address map of the DASH-8:-

ADDRESS	READ	WRITE
Base Address + 0	A/D Lo byte	Start 8 bit A/D conversion
+ 1	A/D Hi byte	Start 12 bit A/D conversion
+ 2	Read status	Write control register
+ 3	-	-
+ 4	Read Counter 0	Load Counter 0
+ 5	Read Counter 1	Load Counter 1
+ 6	Read Counter 2	Load Counter 2
+ 7	-	Counter control

Since the A/D provides 12 bits of data, it requires 2 bytes to handle each word of data. These are arranged in the order lo byte/hi byte which is mainly a convenience for assembly language programmers since the the 8088 processor in the IBM PC accesses data in this sequence. Now we know what's where, let's examine the data format of the registers and how data can be transferred using INP's and OUT's.

3.1.2 STARTING THE A/D CONVERTER

An A/D conversion is initiated by writing to location BASE ADDRESS + 0 or BASE ADDRESS + 1. If you write to BASE ADDRESS + 1, a full 12 bit A/D conversion is performed. Writing to BASE ADDRESS + 0 initiates a short cycle 8 bit conversion. A 12 bit conversion takes no more than 35 microseconds to complete, a short cycle 8 bit conversion takes less time and will not exceed 25 microseconds. (These times are dependent on the type and manufacturer of AD574 A/D converter used in your DASH-8 and may be less, but will not exceed the durations specified).

Starting an A/D conversion:-

```
 12 bits          xxx10 OUT BASADR% + 1, 0

  8 bits          xxx10 OUT BASADR%, 0
```

The data written to these locations is irrelevant and is lost, the decoded address write pulse is in fact what starts the A/D.

3.1.3 READING THE A/D DATA

After the end of conversion the data from the A/D may be read from locations BASE ADDRESS + 0 and BASE ADDRESS + 1. Data follows the usual Intel low byte/high byte sequence. BASE ADDRESS + 1 contains the most significant 8 bits from the conversion:-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE ADDRESS + 1)	B1 (MSB)	B2	B3	B4	B5	B6	B7	B8

The remaining 4 least significant bits followed by 4 zeroes are read from BASE ADDRESS + 0 :-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE ADDRESS + 0)	B9	B10	B11	B12 (LSB)	0	0	0	0

The A/D data bits B1-B12 correspond to an offset binary code:-

BINARY	HEX	ANALOG INPUT VOLTAGE
0000 0000 0000	000	-5.0000 v (-Full scale)
0000 0000 0001	001	-4.9976 v
. . .	.	.
. . .	.	.
0100 0000 0000	400	-2.5000 v (-1/2 scale)
. . .	.	.
. . .	.	.
1000 0000 0000	800	+/-0 v (zero)
1000 0000 0001	801	+0.0024 v
. . .	.	.
. . .	.	.
1100 0000 0000	C00	+2.5000 v (+1/2 scale)
. . .	.	.
. . .	.	.
1111 1111 1111	FFF	+4.9976 v (+Full scale)

A sequence of BASIC INP instructions to read the data would be:-

```
        xxx10  INP(BASADR%), XL%          'read low byte
        xxx20  INP(BASADR% + 1), XH%      'read high byte
        xxx30  X% = XH%*16 + XL%/16       'combine bytes, X% = data
```

From this point you can turn the data in bits into volts or other engineering units (start using real variables!):-

```
        xxx40  V = X%*10/4096             'output * span/resolution
        xxx50  V = V - 5                  'subtract zero offset, -5.0000 v
```

If we were using an input amplifier or attenuator with gain G, we could add another line to provide scaling:-

```
        xxx60  V = V * G
```

3.1.4 THE DASH-8 STATUS REGISTER

The status register provides information on the operation of DASH-8. It is a read only register at I/O location BASE ADDRESS + 2 and has the following format:-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE ADDRESS + 2)	EOC	IP3	IP2	IP1	IRQ	MA2	MA1	MA0

The bits have the following significance:-

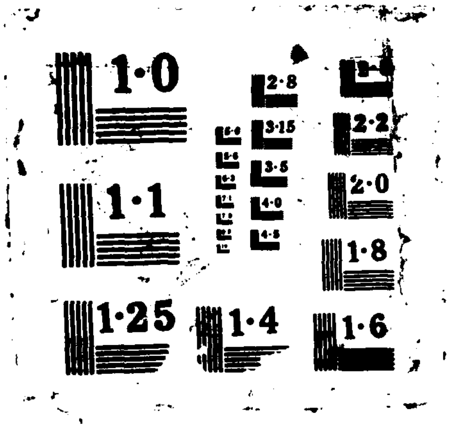
EOC: End of Conversion. If EOC is high (Logic 1) the A/D is busy performing a conversion. Data should not be read in this condition as it will be invalid. Wait for the EOC to return to logic 0 signifying valid data available.

IP3 - IP1: These bits correspond to the three digital input port lines IP3,IP2 and IP1. They may be used for any digital data input.

IRQ: After generation of an interrupt to the processor IRQ is set to logic high (1). It is reset to logic low (0) by a write to the control register. This provides a means of acknowledging or "handshaking" DASH-8 interrupts.

AD-A185 233 THE DYNAMIC CONTROLS OF A HYDRAULIC PRESS BY CONTROLLING THE PUMP MOTOR(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH F EFFRECE JUN 87 2/2

UNCLASSIFIED AFIT/CI/NR-87-56T F/G 13/9 NL


1·0
2·8
3·15
3·5
4·0
4·5
2·2
2·0
1·1
1·8
1·25
1·4
1·6

MA2-MA0: These bits provide the current analog multiplexer channel address as follows:-

MA2	MA1	MA0	CHANNEL
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

3.1.5 THE DASH-8 CONTROL REGISTER

The control register sets the multiplexer (channel) address, enables and disables interrupts and provides output data to the 4 general purpose digital outputs OP1-OP4. The control register is a write only register located at I/O address BASE ADDRESS + 2 (same location as status register). The data format of the control register is:-

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE ADDRESS + 2)	OP4	OP3	OP2	OP1	INTE	MA2	MA1	MA0

The bits have the following significance:-

OP4-OP1: These bits correspond to the four general purpose digital output lines OP1 thru OP4. These lines can be used for external control functions e.g. driving an input sub-multiplexer to increase the number of analog input channels. A 16 channel mux. on each of DASH-8's 8 analog channels can expand the system to 128 channels.

INTE: DASH-8 generated interrupts are enabled onto any of the selected IBM P.C. interrupt levels 2-7 if INTE = 1 (logic high). Interrupts are disabled if INTE = 0 (logic low). Interrupts from the INT.IN input (pin 24) are passed through to the selected level and are positive edge triggered. It is the programmer's responsibility to set up an interrupt handling routine, interrupt vectors and initialize the 8259 interrupt controller on the IBM P.C. processor board. Writing to the control register will clear the IRQ bit of the status register.

MA2-MA0: These bits select the current analog multiplexer channel address as follows:-

MA2	MA1	MA0	CHANNEL
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

The multiplexer channel address can be determined at any time by reading the status register.

One further note about the control register. During power up of the IBM P.C. when the RESET line of the IBM P.C. is asserted, the DASH-8 control register is cleared. This insures that DASH-8 interrupts are disabled, sets digital outputs OP1-4 to zero and sets the multiplexer channel address to zero.

3.1.6 THE COUNTER TIMER REGISTERS

An 8253-5 programmable interval timer is used on DASH-8. This is a very flexible device consisting of 3 separately programmable 16 bit down counters that may be operated in a variety of modes. A fuller description of the capabilities is in Chapter 4 (Counter Timer Operation).For additional technical information on this device, consult the "Intel Component Data Catalog"[6] or equivalent manufacturer's data sheet.

From a programming standpoint addressing counter timer functions is straightforward. The counter registers themselves are read write and located at I/O addresses:-

BASE ADDRESS + 4	:	Counter 0
BASE ADDRESS + 5	:	Counter 1
BASE ADDRESS + 6	:	Counter 2

Before reading or writing to the counter registers, you should write

----

[6] Available from Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA. 95051. Phone (408)-987-8080

to the counter timer control register to define the operating mode of each counter and the type of data transfer that you intend to make. The counter timer control register is write only and located at BASE ADDRESS + 7. It has the following format:-

BASE ADDRESS + 7 : 8253 Control (Write only)

BIT POSITION	D7	D6	D5	D4	D3	D2	D1	D0
(BASE ADDRESS + 7)	SC1	SC0	RL1	RL0	M2	M1	M0	BCD

SC1-0: These are the "select counter" bits that control which counter the following configuration bits will operate on. The format for the SC1-0 bits is:-

SC1	SC0	Addressed Counter
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Invalid

RL1-0: These are the "read/load" configuration bits that control the form of the data transfer to the selected counter. The format for the RL1-0 bits is:-

RL1	RL0	Data Xfer Operation
0	0	Counter latching operation
0	1	Read/load high byte
1	0	Read/load low byte
1	1	Read/load low then high byte (2 byte transfer)

See Chapter 4 on Counter Timer Operation for a fuller description of these data transfer modes.

M2-0: These are the selected counter operating mode control bits. Their format is:-

M2	M1	M0	Counter Mode
0	0	0	0 - Change on terminal count
0	0	1	1 - Programmable one-shot
0	1	0	2 - Rate generator
0	1	1	3 - Square wave generator
1	0	0	4 - Software triggered strobe
1	0	1	5 - Hardware triggered strobe

See Chapter 4 on Counter Timer Operation for a fuller description of these operating modes.

BCD: This bit controls whether the selected counter will count in binary or binary coded decimal

(8,4,2,1 BCD) code.

BCD	Counting Code
0	16 bit binary (65,535 max.count)
1	4 decade BCD (9,999 max. count)

3.1.7 SOME BASIC PROGRAMMING TIPS

Some BASIC commands which you may not have used frequently in the course of ordinary programming, but that are useful with DASH-8 are:-

1. **WAIT port, n[,m]**

Data read at the port is exclusive-or'ed (XORed) with integer expression "m" and ANDed with "n". If the result is zero, BASIC loops back and tests the port again until a non-zero result is obtained. This is an excellent way of making your program wait until some desired input condition is attained.

2. **ON TIMER (n) GOSUB line**

This command is only available in DOS 2.0. In effect it provides you with a pseudo-interrupt. After execution of each BASIC statement, BASIC checks the timer to see if the condition >n (1 < n < 86,400 seconds) is satisfied. If it is, control passes to the subroutine, otherwise the next line is executed. This polling of the timer is called trapping and is activated by:-

TIMER ON

Trapping is disabled by:-

TIMER OFF

Note that trapping only occurs while your BASIC program is executing (unlike a true interrupt) and can be slightly delayed by statements that require a lot of execution time. If you need to sample at long intervals the ON TIMER command is an alternative to using the internal DASH-8 counter timer.

3. If you wish to program an A/D conversion using BASIC INP and OUT rather than the more powerful CALL routine, the following sequence should be followed:-

```
xxx10 OUT BASADR%+2, MA%          'MA% = channel number, 0-7.
xxx20 OUT BASADR%+1, 0            'start 12 bit A/D conversion
xxx30 IF INP(BASADR%+2)>=128 GOTO xxx30   'status test loop
```

```
xxx40 XL%=INP(BASADR%):XH%=INP(BASADR%+1) 'read A/D data
xxx50 DIO% = 16*XH% + XL%/16             'combine bytes
```

Note that looping while the A/D converter is still busy (line xxx30) is not required in interpreted BASIC as the interpreter execution time greatly exceeds the A/D conversion time of 35 microseconds max. However, if the program is subsequently compiled using the BASIC COMPILER, the execution time is reduced to a point where the status test of line xxx30 is essential to avoid returning erroneous data in line xxx40.

3.2 LOADING THE MACHINE LANGUAGE CALL ROUTINE "DASH8.BIN"

As you probably now realise, direct I/O using BASIC INP and OUT can be somewhat tedious to implement although a lot of the required programming could be handled in subroutines. Use of the CALL routine described in the this section avoids these problems, circumvents some of the execution time delays of interpreted or compiled BASIC, and also permits interrupt driven operations which BASIC does not support.

In order to make use of the CALL routine "DASH8.BIN", it must first be loaded into memory. You must avoid loading it over any part of memory that is being used by another program e.g. BASIC, print spoolers or "Disk-RAM". If you do interfere with another program's use of memory, the CALL routine will not work and your PC will most likely hang up (Turn off power and wait a few seconds before turning on again). Note that the information given in this section is general and would apply to loading any CALL or USR routine and supplements the limited information in Appendix C of the "I.B.M. BASIC MANUAL".[7]

You have two options depending on the size of your available memory. The maximum memory segment that BASIC is able to use is 64K bytes. If BASIC is using its maximum 64K you will get the following message on power up or from DOS by entering A> BASIC(A) :-

```
CASSETTE        The IBM Personal Computer Basic
BASIC           Version C1.00 Copyright IBM Corp 1981
                62940 Bytes free
                Ok

BASIC           The IBM Personal Computer Basic
(DOS 1.1)       Version D1.10 Copyright IBM Corp. 1981, 1982
```

----

[7] For further enlightenment on this subject see "The 8086 Connection" by Dan Rollins, page 398 of "Byte" magazine, July 1983.

```
                61807 Bytes free
                Ok

 BASICA         The IBM Personal Computer Basic
 (DOS 1.1)      Version A1.10 Copyright IBM Corp. 1981, 1982
                61502 Bytes free
                Ok

 BASICA         The IBM Personal Computer Basic
 (DOS 2.0)      Version A2.0 Copyright IBM Corp. 1981, 1982, 1983
                60865 Bytes free
                Ok
```

When the number of memory bytes free is less than that shown above for the version of BASIC in use, then your PC's memory is already fully utilised and BASIC adjusts to this condition by using less than its possible 64K maximum. If this is the case, you must load the CALL routine by further forced contraction of the BASIC workspace and loading the routine at the end of the newly defined workspace. DASH-8.BIN will occupy about 1.8K bytes but to keep things simple, let's clear a 2K space for it.

Step 1 is to work out how much memory BASIC is actually using. Let's assume you see "XXXXX Bytes free" after loading BASIC as above. The amount of memory BASIC is using is then:-

```
XXXXX + (65535 - 62940) bytes for CASSETTE BASIC
XXXXX + (65535 - 61807) bytes for BASIC (DOS 1.1)
XXXXX + (65535 - 61502) bytes for BASICA (DOS 1.1)
XXXXX + (65535 - 60865) bytes for BASICA (DOS 2.0)
```

Now subtract 2048 (2K) from the above number. The result is the maximum amount of working space that can be allocated to BASIC with the CALL routine loaded. (You always have the option to allocate less working space if you wish.) Let's call the size of the workspace WS. This space can be allocated either when loading BASIC from DOS e.g.:-

```
A> BASIC(A) /M:WS
```

or usually more conveniently at the beginning of a program by CLEAR e.g.:-

```
xxx10 CLEAR, WS
```

Next, we need to know what segment BASIC is occupying in memory. This can be found from the contents of memory locations &H511 and &H510 which hold the current BASIC segment which we can call SG. SG can be determined as follows:-

```
xxx20 DEF SEG = 0  ;define current segment = 0000
                    before reading absolute
                    addresses 0000:0510 & 0000:0511
```

```
xxx30 SG = 256*PEEK(&H511) + PEEK(&H510)
```

The segment address at which we can can now load the CALL routine will simply be at the end of the working space i.e.:-

```
xxx40 SG = WS/16 + SG ;remember segment addresses
                       are on 16 bit boundaries
```

The routine can now be loaded as follows:-

```
xxx50 DEF SEG = SG
xxx60 BLOAD "DASH8.BIN",0   ;loads routine at SG:0000
```

A BLOAD must be used as we are loading a binary (machine language) program. Once loaded, the CALL can be entered as many times as needed in the program after initializing the call parameters MD%, DIO%, FLAG% prior to the CALL sequence as follows:-

```
xxx70 DEF SEG = SG
xxx80 DASH8 = 0
xxx90 CALL DASH8 (MD%, DIO%, FLAG%)
```

Note that DASH8 is a variable that specifies the memory offset of the starting address of the CALL routine from the current BASIC segment. We have chosen DASH8 as a name as it makes CALL DASH8 easy to remember and would distinguish it from any other CALL to some other routine that might be in the same program. This is purely a matter of choice and programming style, just easier to remember than writing CALL X (.....) or CALL AB(3) (.....) etc.. The variable DASH8 is the offset (actually zero) from the current segment as defined by the last DEF SEG statement that tells your BASIC interpreter where the CALL routine is located. Be careful that you do not inadvertently redefine the current segment somewhere in a program before entering the CALL. It is good practice to immediately precede the CALL statement by the appropriate DEF SEG statement (the same one you preceded your BLOAD with) even at the cost of duplication. This precaution can save a lot of wasted time and frustration from crashing your computer!

Another important detail to understand is that CLEAR sets working space from the bottom of the BASIC working area up whereas we must set aside space for our subroutine from the top of available memory down. If we attempt to CLEAR more space than is actually available, we will end up loading our routine over the end of the BASIC program, data space and stack and will hang up the computer. Be careful this does not happen inadvertently if you are memory limited and later load BASIC with DEBUG or some other coresident program without making a compensating reduction in the workspace (WS) declaration in the CLEAR statement. If possible, setting up a workspace that is a considerable amount less than the maximum available is a simple precaution. This all sounds a little on the complex side, but like all things it really isn't once you understand what you are doing. To further assist you, run and list the file

LOADCALL.BAS. This gives you an examples of loading and using the CALL routine and ready made loading and initializing code that you can merge into your own programs.

The second option is somewhat simpler to follow and applies when you have plenty of memory and you are able to load the CALL routine outside the BASIC workspace. In this case choose a segment that has 2K bytes clear at its beginning. For example we might choose &H1700 which is at 92K on a machine with a minimum of 96K memory. Then proceed as follows:-

```
xxx10 DEF SEG = &H1700        'Sets up load segment
xxx20 BLOAD "DASH8.BIN",0     'Loads at 1700:0000
xxx30 DASH8 = 0
   ..
   ..
   ..
xxxxx DEF SEG = &H1700
xxxyy CALL DASH8 (MD%, DIO%, FLAG%)
xxxzz etc.
```

An example of this approach is also contained in file LOADCALL.BAS. Before you try loading outside the workspace, be sure you really have an unused 2K of memory at 92K. You can change the DEF SEG statements in line 2010 and experiment with loading the CALL routine at other locations. Usually any clash with another program's use of the same memory results in obliteration of some of the routine code and a failure to exit and return from the routine. The computer hangs up, and the only cure is to switch off, wait a few seconds and turn on the power again.

3.3 FORMAT OF THE CALL STATEMENT

If you are new to using CALL statements, this explanation may assist you in understanding how the CALL transfers execution to the machine language (binary) driver routine. Prior to entering the CALL, the DEF SEG = SG statement sets the segment address at which the CALL subroutine is located. The CALL statement for the DASH8.BIN driver must be of the form:-

```
xxxxx  CALL DASH8 (MD%, DIO%, FLAG%)
```

As explained in the previous section, DASH8 is the address offset from the current segment of memory as defined in the last DEF SEG statement. In all of our examples, we have chosen to define the current segment to correspond with the starting address of the CALL routine, therefore this offset is zero and DASH8 = 0.

The three variables within brackets are known as the CALL parameters. On executing the CALL, the addresses of the variables (pointers) are passed in the sequence written to BASIC's stack. The CALL routine unloads these pointers from the stack and uses them to locate the variables in BASIC's data space so data can be exchanged with them. Three important format requirements must be met:-

1. - The CALL parameters are positional. The subroutine knows nothing of the names of the variables, just their locations from the order of their pointers on the stack. If you write:-

```
xxxxx CALL DASH8 (DIO%, MD%, FLAG%)
```

you will mix up the CALL routine, since it will interpret DIO% as the mode number, the mode MD% as the data etc.. The parameters must always be written in the correct order:-

```
    (mode, data, errors)
```

2. - The CALL routine expects its parameters to be integer type variables and will write and read to the variables on this assumption. If you slip up and use a non-integer (real single or double precision) variable in the CALL parameters, the routine will not function correctly.

3. - You cannot perform any arithmetic functions within the parameter list brackets of the CALL statement e.g:-

```
   CALL DASH8 (MD% + 2, DIO% * 8, FLAG%)
```

is illegal and will produce a syntax error.

4. - You cannot use constants for any of the parameters in the CALL statement. The following is illegal:-

```
   CALL DASH8 (7, 2, FLAG%)
```

This must be programmed as:-

```
   xxx10  MD% = 7
   xxx20  DIO% = 2
   xxx30  CALL DASH8 (MD%, DIO%, FLAG%)
```

Apart from these restrictions, you can name the integer variables what you want, the names in the examples are just convenient mnemonics. Strictly, you should declare the variables before executing the CALL. If you do not, the simple variables will be declared by default on execution, but array variable obviously cannot be dimensioned by default and must be dimensioned before the CALL to pass data correctly if used as a CALL parameter. Some modes of the CALL routine require that data is passed in an array. In this case DIO%(0) should be specified as the data variable so that the CALL routine can locate the position of the array.

You can use some elegant techniques with the CALL parameters. Let's say we wanted to record or output a whole series of data in a FOR . . . NEXT loop. You can dimension your DIO% array as a two or more dimension array, for example:-

```
xxx00 DIM DIO%(2,100)
xxx10 DEF SEG = SG
xxx20 FOR I = 0 to 100
xxx30 CALL DASH8 (MD%, DIO%(0,I), FLAG%)
xxx40 NEXT I
```

Likewise any of the other CALL parameters may be integer array variables if required, and you can name any number of different integer data arrays for output and input. It is O.K. to dimension arrays with more elements than will be used by the CALL, unused elements will be unchanged and as an example could be used for tagging data with time, date or other information. If you wish to transfer data into a particular element of an array then specify it in the CALL parameter list> For example:-

```
xxxxx CALL DASH8 (MD%, NUT%(9,4), FLAG%)
```

would transfer the data to element 9,4 of array NUT%.

3.4 EXAMPLES OF THE USE OF THE CALL ROUTINE

The following subsections give detail information and examples of the use of the CALL routine in all 17 modes. The modes are selected by the MD% parameter in the CALL as follows:-

MODE		FUNCTION
0	. . .	Initialize, input DASH-8 base address.
1	. . .	Set multiplexer low & high scan limits.
2	. . .	Set multiplexer (channel) address.
3	. . .	Read multiplexer (channel) address.
4	. . .	Perform a single A/D conversion Return data and increment multiplexer address.
5	. . .	Perform an N conversion scan after trigger. Scan rate set by Counter 2 or external strobe.
6	. . .	Enable interrupt operation.
7	. . .	Disable interrupt operation.

8 . . . Perform conversions on N interrupts and dump data in segment S of memory. (Background operation).

9 . . . Unload data from segment S and transfer to BASIC array variable.

10 . . . Set timer/counter configuration.

11 . . . Load timer/counter.

12 . . . Read timer/counter.

13 . . . Read digital inputs IP1-3.

14 . . . Output to digital outputs OP1-4.

15 . . . Measure frequency with timer/counter.

16 . . . Measure period with timer/counter.

17 . . . Tag lower nybble of data with channel number.

3.4.1 MODE 0 - INITIALIZE

Before using any other mode of the CALL routine, you must acquaint the routine with the I/O location, or BASE ADDRESS, of the DASH-8 board. Failure to provide this information will cause an error flag (FLAG% = 1, base address unknown) if any other mode of the CALL is selected. This need only be done once in the initialization section of your program.

On entry the following parameters should be initialized:-

```
        MD% = 0                          (mode)
        BASADR% = &H300 'for example     (I/O address)
        FLAG% = X                        (value does not matter)
    then:-
        CALL DASH8 (MD%, BASADR%, FLAG%)
```

On return the variables contain data as follows:-

```
        MD% = 0          (unchanged)
        BASADR% = &H300  (unchanged)
```

The following error codes apply to mode 0:-

```
        FLAG% = 0        (no error, o.k.)
              = 2        (mode number out of range, <0 or >17)
              = 3        (base address out of range <255 or >1016)
```

Note that error 3 will occur if you have specified an I/O address that is less than 255 (Hex FF) cr greater than 1016 (Hex 3F8). I/O addresses below Hex FF are all used internally by devices on the IBM P.C. system board and would always cause an address conflict with DASH-8 and addresses above Hex 3FF are not decoded on the IBM P.C.

3.4.2 MODE 1 - SET MULTIPLEXER SCAN LIMITS

Mode 1 is used to set the scan limits of the multiplexer prior to performing conversions. Two limits are passed in a 2 element array variable LT%(1). LT%(0) contains the lower limit and LT%(1) the higher limit. If mode 1 is not entered, the lower and higher limits default to 0 and 7 respectively.

To illustrate the action of mode 1 assume we set LT%(0) = 3 and LT%(1) = 6. The first conversion (commanded by modes 4 or 7) would be performed on channel 3, data returned and the channel incremented to 4. The next conversion would be performed on channel 4, data returned and the channel incremented to 5 etc.. This would continue until the conversion had been performed on channel 6 in which case the multiplexer address would be reset to 3 ready for the cycle to repeat. Scanning of channels would always be stepped between and including channels 3 and 6.

If we wish to perform continuous conversions on one channel, then set the low and high limits equal to each other and the desired channel number e.g. setting LT%(0) = 1 and LT%(1) = 1 would perform continuous conversions on channel 1.

Note that the starting channel number defaults to LT%(0), the lower limit. If for example we had set our lower limit LT%(0) = 2 and our higher limit LT%(1) = 5 and we wished to start scanning on channel 4, then we would use mode 2 to set the multiplexer address after setting the scan limits.

On entry the following parameters should be initialized:-

```
        MD% = 1                     (mode)
        LT%(0) = 0 thru 7           (lower scan limit)
        LT%(1) = 0 thru 7           (upper scan limit)
        FLAG% = X                   (value does not matter)
    then:-
        CALL DASH8 (MD%, LT%(0), FLAG%)
                             ^
```

Note that specifying the first element of the array will pass all other required array parameters.

On return the variables contain data as follows:-

```
          MD% = 1              (unchanged)
          LT%(0) = 0 thru 7    (unchanged)
          LT%(1) = 0 thru 7    (unchanged)
```

The following error codes apply to mode 1:-

```
          FLAG% = 0         (no error, o.k.)
                = 1         (base address unknown)
                = 2         (mode number out of range, <0 or >17)
                = 4         (scan limits out of range)
```

Error code 4 will be returned if either or both of the scan limits is out of range i.e. <0 or >7 or their order is reversed i.e. lower limit > upper limit.

3.4.3 MODE 2 - SET MULTIPLEXER ADDRESS

Mode 2 sets the starting multiplexer address ready for the next conversion. Default value is zero.

On entry the following parameters should be initialized:-

```
          MD% = 2                              (mode)
          CH% = 0 thru 7                       (channel number)
          FLAG% = X                            (value does not matter)
     then:-
          CALL DASH8 (MD%, CH%, FLAG%)
```

On return the variables contain data as follows:-

```
          MD% = 2              (unchanged)
          CH% = 0 thru 7       (unchanged)
```

The following error codes apply to mode 2:-

```
          FLAG% = 0         (no error, o.k.)
                = 1         (base address unknown)
                = 2         (mode number out of range, <0 or >17)
                = 5         (channel number out of range)
```

Error code 5 will be returned if you attempt to set the channel address outside the scan limits. If mode 1 has not been used to make an explicit declaration of these limits prior to entering mode 2, they will default to 0 and 7. If limits were set by mode 1, then these limits will apply.

3.4.4 MODE 3 - READ MULTIPLEXER ADDRESS

Mode 3 allows you to determine the current multiplexer (channel) address.

On entry the following parameters should be initialized:-

```
          MD% = 3                         (mode)
          CH% = X                         (value does not matter)
          FLAG% = X                       (value does not matter)
     then:-
          CALL DASH8 (MD%, CH%, FLAG%)
```

On return the variables contain data as follows:-

```
          MD% = 3          (unchanged)
          CH% = 0 thru 7   (current channel number)
```

The following error codes apply to mode 3:-

```
          FLAG% = 0        (no error, o.k.)
                = 1        (base address unknown)
                = 2        (mode number out of range, <0 or >17)
```

Note that if you are using any of the auto-incrementing modes 4,5 or 8, the multiplexer is incremented after the A/D conversion has been started (while the sample/hold is in hold) and the EOC (busy) signal of the A/D is high. Reading the multiplexer address while a conversion is in progress may yield an unexpected result. Before reading the multiplexer address, it is good practice to check the DASH-8 status register EOC bit if there is any doubt about the A/D activity.

3.4.5 MODE 4 - COMMAND SINGLE A/D CONVERSION AND INCREMENT MULTIPLEXER

Mode 4 initiates a sequence of actions:-

1 - Starts a 12 bit A/D conversion.
2 - After A/D has started and sample/hold is in hold, increments multiplexer and checks whether scan limit exceeded. If so sets to lower scan limit.
3 - Polls status to determine if A/D finished.
4 - Returns data in variable.

The A/D will perform conversions on channels in accordance with the conditions set in modes 1 & 2. If modes 1 & 2 have not been entered prior to mode 4, conversions will start on channel 0 and the upper scan limit will be channel 7.

On entry the following parameters should be initialized:-

```
          MD% = 4                         (mode)
          DIO% = X                        (value does not matter)
          FLAG% = X                       (value does not matter)
      then:-
          CALL DASH8 (MD%, DIO%, FLAG%)
```

On return the variables contain data as follows:-

```
          MD% = 4         (unchanged)
          DIO% = data     (converted data)
```

Note that if mode 17 has been enabled prior to entering mode 4, returned data will be "tagged" with the channel address (see mode 17).

The following error codes apply to mode 4:-

```
          FLAG% = 0       (no error, o.k.)
                = 1       (base address unknown)
                = 2       (mode number out of range, <0 or >17)
                = 6       (A/D timeout)
```

Error code 6 will only occur if the end of conversion signal (EOC, from the A/D remains nigh for more than 100 microseconds or stays low after a conversion has been initiated. Either of these conditions is indicative of a hardware fault and it is recommended that users replace the A/D converter I.C. (AD574A) as a first step in attempting correction. A normal A/D conversion should not take more than 35 microseconds.

3.4.6 MODE 5 - DO N CONVERSIONS ON TRIGGER AND TRANSFER TO ARRAY VARIABLE

Mode 5 is an expanded version of mode 4 except that a whole sequence or scan of conversions is initiated by a high input on IP1. Once the scan has been started, the rate at which conversions are performed is set by Counter 2. A number (N) conversions specified at entrance to the CALL are performed and transferred to any specified integer array ARRAY%(N) which should be dimensioned to have a number of elements not less than N. No error checking is performed to determine whether sufficient elements of the array exist to absorb the number of conversions programmed. If this is not so, bizarre results may ensue!

The interrupt handshake flip-flop on the DASH-8 board is used to acknowledge trigger inputs in this mode although the conversions are not interrupt driven. Mode 5 will perform a maximum of about 4000 conversions/sec. although some conversions may be delayed by the timer interrupt of the IBM P.C.. If you wish to trigger at a programmable time interval set by Counter 2 then:-

8.2 CALIBRATING THE A/D

The only user adjustments on DASH-8 are 2 trimmer potentiometers that control the A/D - and + Full Scale. The A/D output should be observed (CALDASH8.BAS does this) while applying a known calibration voltage to any or all analog input channels. Briefly the adjustment sequence is:-

1. Apply an analog input of -4.9988v and adjust the -F.S. pot so that the output flickers between -2048 & -2047. This pot is marked R2 on the DASH-8 board and the righthand one.

2. Apply an analog input of +0.0012v and adjust the +F.S. pot so that the output flickers between 0 and +1. This pot is is marked R1 and is the lefthand. You can check the +F.S. with a +4.9963v input which should produce a reading of +2046/2047. If there is a slight error e.g. 1 count you can alternate between +0.0012 & +4.9963v inputs and adjust for the best compromise.

Adjustments are done on "half" bit intervals to obtain a reading flickering about 50/50 between two adjacent values. This is more precise than applying center bit values such as -5.0000v, 0.0000v etc. and performs a better calibration.

Appendix A

CONNECTIONS

A.1 MAIN I/O CONNECTOR

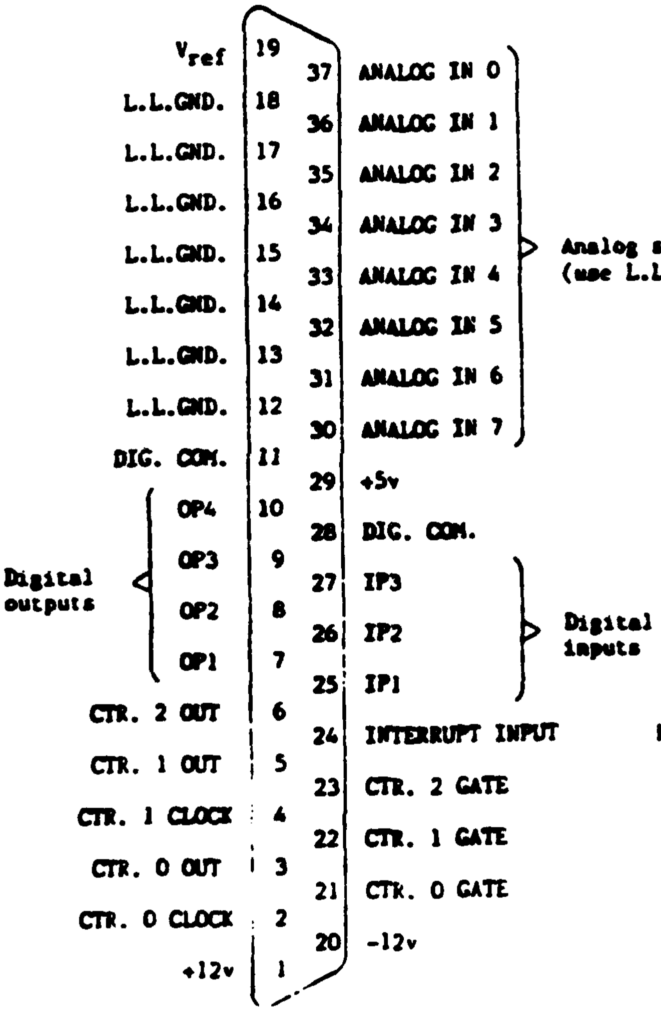
The main analog and digital I/O is via a 37 pin D type connector that projects through the computer case at the rear of the board. The pin functions are as follows (see Fig A.1 for locations):-

PIN	NAME	FUNCTION
1	+12v	+12v power supply from P.C. buss (observe loading limits)
2	CLK0	8253 Counter 0 clock input
3	OUT0	8253 Counter 0 output
4	CLK1	8253 Counter 1 clock input
5	OUT1	8253 Counter 1 output
6	OUT2	8253 Counter 2 output
7	OP1	Digital output #1
8	OP2	Digital output #2
9	OP3	Digital output #3
10	OP4	Digital output #4
11	DIG.COM.	Digital Common. Return for all logic signals and power supply currents. Connected to computer frame.
12	L.L.GND.	
13	L.L.GND.	

14	L.L.GND.	Low level grounds. These are common returns and shields for the analog input channels.
15	L.L.GND.	
16	L.L.GND.	
17	L.L.GND.	
18	L.L.GND.	
19	VREF	+10.00v(+/-0.1v) precision reference voltage from A/D.
20	-12v	-12v power from P.C. buss. (observe loading limits).
21	GATE0	8253 Counter 0 gate input
22	GATE1	8253 Counter 1 gate input
23	GATE2	8253 Counter 2 gate input
24	INT. IN.	Interrupt input. Positive edge triggerred input.
25	IP1	Digital input #1
26	IP2	Digital input #2
27	IP3	Digital input #3
28	DIG. COM.	Digital common. (same as pin 11).
29	+5v	+5v power from P.C. buss (observe loading limits).
30	IN 7	Channel #7 analog input
31	IN 6	Channel #6 analog input
32	IN 5	Channel #5 analog input
33	IN 4	Channel #4 analog input
34	IN 3	Channel #3 analog input
35	IN 2	Channel #2 analog input
36	IN 1	Channel #1 analog input
37	IN 0	Channel #0 analog input

The mating connector for DASCON-1 is a standard 37 pin D type female such as ITT/Cannon #DC-37S for soldered connections. Insulation displacement (flat cable) types are readily available e.g. Amp #745242-1.

A.2 REAR VIEW OF DASH-8 CONNECTOR



Note:
+5v,+12v & -12v are outputs from computer power and should <u>not</u> be overloaded.

Fig. A.1: Rear view of I/O connector (37 pin "D" type male)

Appendix B

SPECIFICATIONS

B.1 POWER CONSUMPTION

+5v supply	-	107mA typ. / 180mA max.
+12v supply	-	6mA typ. / 10mA max.
-12v supply	-	10mA typ. / 16mA max.

B.2 ANALOG INPUT SPECIFICATIONS

8 analog input channels each with the following specification:-

Resolution	-	12 bits. (2.4mV/bit)
Accuracy	-	0.01% of reading +/-1 bit.
Full scale	-	+/-5 volts
Coding	-	Offset binary
Overvoltage	-	Continuous single channel to +/-35v
Configuration	-	Single ended.
Input current	-	100nA max at 25 deg.C.
Temperature Coefficient	-	Gain or F.S., +/-25ppm/deg.C. max. Zero, +/-10 microvolt/deg.C. max.

B.3 A/D SPECIFICATION

Type	-	Successive approximation.
Resolution	-	12 bits
Conversion time.	-	25 microseconds typ. 35 microseconds max.
Monotonicity	-	Guaranteed over operating temperature range.
Linearity	-	+/-1 bit.
Zero drift	-	10ppm/deg. C. max.
Gain drift	-	50 ppm/deg. C max. (30 ppm/deg C. available to special order)

B.4 SAMPLE HOLD AMPLIFIER

Acquisition time	-	15 microsecs to 0.01% typ. for full scale step input
Dynamic sampling error	-	1 bit (2.44mV) @ 2000v/sec.

B.5 REFERENCE VOLTAGE OUTPUT

Reference voltage	-	+10.0v +/- 0.1v
Temperature coefficient	-	50 ppm/deg.C max. (30 ppm/deg.C available to special ordwer)
Load current	-	+/-2mA max.

B.6 DIGITAL I/O

OP1-4 output low voltage	-	0.5v max at Isink = 8.0mA
OP1-4 output high voltage	-	2.7v min at Isource = -0.4mA
IP1-3 input low voltage	-	0.8v max
IP1-3 input low current	-	-0.4mA max
IP1-3 input high voltage	-	2.0v min
IP1-3 input current	-	20uA max. @ 2.7v

B.7 INTERRUPT INPUTS

Type	-	Positive edge triggered
Level	-	2 - 7 jumper selectable
Enable	-	Via INTE of CONTROL register

Interrupts are latched in an internal flip-flop on the DASH-8 board. The state of this flip-flop corresponds to the INT bit in the STATUS register. Flip-flop is cleared by a write to the CONTROL register. Service routines should acknowledge and re-enable interrupt flop.

B.8 COUNTER/TIMER

Type	-	8253-5 programmable interval timer
Counters	-	3 down counters, 16 bit.
Output drive capability	-	2.2mA @ 0.45v (5 LSTTL loads)
Input, gate & clock load	-	TTL/DTL/CMOS compatible +/-10 uA current
Max. input clock freq.	-	Not less than 2.6MHz (may vary with manufacturer)
Active count edge	-	Negative
Minimum clock pulse widths	-	230nS high / 150nS low

For additional information on programming see Chapter 4.

B.9 POWER OUTPUTS

IBM P.C. buss supplies	-	+5v & +/-12v
Tolerance	-	+5v +/-5% +12v +/-5% -12v +/-10%
Loading	-	Dependent on other peripherals (see Section 7.7)

B.10 GENERAL ENVIRONMENTAL

Operating temperature range.	-	0 to 50 deg. C.
Storage temperature range	-	-20 to +70 deg.C.
Humidity	-	0 to 90% non-condensing.
Weight	-	4 oz. (120 gm.)

Appendix C

STORAGE OF INTEGER VARIABLES

Data is stored in integer variables (% type) in 2's complement form. Each integer variable uses 16 bits or 2 bytes of memory. 16 bits of data is equivalent to values from 0 to 65,535 decimal, but the 2's complement convention interprets the most significant bit as a sign bit so the actual range becomes -32,768 to +32,767 (a span of 65,535). Numbers are represented as follows:-

	High byte								Low byte							
	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
+32,767	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
+10,000	0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0
+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
-10,000	1	1	0	1	1	0	0	0	1	1	1	1	0	0	0	0
-32,768	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sign bit
1 if negative, 0 if positive

Integer variables are the most compact form of storage for the 12 bit data from the A/D converter and 16 bit data of the 8253 interval timer and so to conserve memory and disk space and optimise execution speed, all data exchange via the CALL is through integer type variables. This poses a programming problem when handling unsigned numbers in the range 32,768 to 65,535.

If you wish to input or output an unsigned integer greater than 32,767 then it is necessary to work out what its 2's compliment signed equivalent is. As an example, assume we want to load a 16 bit counter with 50,000 decimal. An easy way of turning this to binary is to enter BASIC and execute PRINT HEX$(50000). This returns C350 or binary:-

Appendix 8

DAC-02 Manual

DAC-02 MANUAL

MBC MetraByte Corporation

7/84

DAC-02 MANUAL

1.1 Description

The DAC-02 consists of 2 separate double buffered 12 bit multiplying D/A channels plus interface circuitry. The D/A converters may be used with a fixed D.C. reference as conventional D/A's. On board references of -5v and -10v are provide output ranges of 0-5v, 0-10v, +/-5v and +/-10v and 4-20mA for process control current loops. Alternatively, the D/A's may be operated with a variable or A.C. reference signal as multiplying D/A's, the output is the product of reference and digital inputs. With an A.C. reference, the unipolar outputs provide 2 quadrant multiplication and the bipolar outputs provide 4 quadrant operation. 12 bit accuracy is maintained up to 1KHz.

Since data is 12 bits, data is written to each D/A in 2 consecutive bytes. The first byte is the least significant and contains the 4 least significant bits of data (see Section 1.4 for format). The second byte is the most significant and contains the most significant 8 bits of data. The least significant byte is usually written first and is stored in an intermediate register in the D/A, having no effect on the output. When the most significant byte is written, its data is added to the stored least significant data and presented "broadside" to the D/A converter thus assuring a single step update. This process is known as double buffering.

The DAC-02 is packaged on a 5" long "half-slot" board suitable for use in all models of IBM P.C.'s (including the portable) and all compatibles including the T.I. Professional. The DAC-02 is addressed as an I/O device using 8 I/O locations and may have its I/O address set by means of an on-board DIP switch to any 8 bit boundary in the 255-1023 (decimal) I/O address space. The board uses the internal +5v, +12v and -12v computer supplies and consumes 850 milli-watts of power.

1.2 I/O Map

The DAC-02 requires 8 consecutive addresses in I/O address space. The locations of the D/A registers is as follows:-

Base	Address	+ 0	-	D/A #0	Low byte
"	"	+ 1	-	D/A #0	High byte
"	"	+ 2	-	D/A #1	Low byte
"	"	+ 3	-	D/A #1	High byte

The next 4 addresses are redundant and repeat the above pattern and can be ignored in programming:-

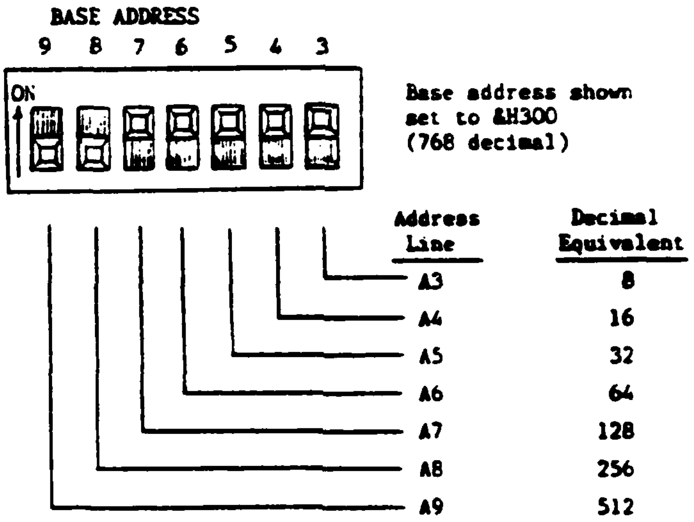
Base	Address	+ 4	-	D/A #0	Low byte
"	"	+ 5	-	D/A #0	High byte
"	"	+ 6	-	D/A #1	Low byte
"	"	+ 7	-	D/A #1	High byte

1.3 Installation

Before installing the DAC-02 in your computer's peripheral expansion slots, set the BASE ADDRESS switch to a suitable I/O address that will not conflict with any of your other peripheral boards. Certain I/O addresses have been reserved by IBM and may be in use in your computer. These are as follows:-

Address Hex Range	Device
000 - 0FF	Internal system board I/O
200 - 20F	Game I/O adapter
278 - 27F	Reserved
2F8 - 2FF	Reserved
320 - 32F	Hard disk controller (XT only)
378 - 37F	Parallel printer port
3B0 - 3BF	Monochrome display
3F0 - 3F7	Floppy disk controller
3F8 - 3FF	Asynchronous communications I/O

Unless you are using the IBM prototype board which uses addresses in the range 300 - 31F hex, a simple choice of base address for the DAC-02 would be 300 hex. The DAC-02 board will then occupy I/O addresses 300 - 307. The setting for the BASE ADDRESS switch is shown below:-



Base address shown set to &H300 (768 decimal)

Note that a switch in the ON position corresponds to a zero, and in the OFF position corresponds to the binary weight of the corresponding address bit (512, 256, 128, 64, 32, 16, 8). The base I/O address is the sum of all the OFF switch address bits.

After setting the BASE ADDRESS switch the board should be installed in the computer. Turn off the power on your computer before removing or installing any board. Remove a vacant back plate, engage and push home the DAC-02 in the 62 pin edge connector and secure the back plate with the screw that you just removed. Turn on the computer power and boot-up should be normal.

If there is any problem with boot-up or subsequent operation of the DAC-02 or other peripheral boards, most likely you chose an I/O address that conflicts with another peripheral device. In this case, change the BASE ADDRESS switch setting and try again.

The program INSTALL.BAS on the DAC-02 software disk will assist you in setting the BASE ADDRESS switch correctly. Load BASICA and then type RUN"INSTALL.

1.4 Data Format

Data format for the D/A registers is as follows:-

Low byte:	D7	D6	D5	D4	D3	D2	D1	D0
Base + 0 or 2	B9	B10	B11	B12 (LSB)	x	x (x = don't care)	x	x

High byte:	D7	D6	D5	D4	D3	D2	D1	D0
Base + 1 or 3	B1 (MSB)	B2	B3	B4	B5	B6	B7	B8

Writing the low byte stores it in an intermediate register. Writing the high byte loads the D/A with both the high byte and the stored low byte data. For 8 bit operation, first write zero to the low byte and all further operations may then be performed with the high byte only.

Assembly language programmers should note that if data is left justified, 16 bit output operations may be used (e.g. OUT DX, AX) as the data sequence is conventional Intel Low/High byte.

1.5 Programming

No special software is provided with the DAC-02 as programming is simple with I/O output instructions in whatever application language is used.

The following example shows how to output data in BASIC. It is readily "translatable" to other languages. Since the D/A's have 12 bit resolution, data D should be in the range 0- 4095 decimal. First split the data into 2 bytes DL% (low) and DH%(high):-

```
xxx00  DH% = INT(D/16)    'generate high byte
xxx10  DL% = D - 16*DH%   'derive remainder in low byte
xxx20 DL% = 16 * DL%      'shift low nybble4 places left
```

Next write the data to the D/A. The example assumes D/A #0 with a board base address of Hex 300:-

```
        xxx30  OUT &H300, DL%     'low byte
        xxx40  OUT &H301, DH%     'high byte and load
```

For a working example of programming in BASIC, list and run the program PROG.BAS on the DAC-02 software disk.

An assembly laqnguage routine is even simpler. Assume AX contains the data and DX has the board I/O address. To write to D/A #0:-

```
               MOV CL,4          ;set up for 4 left shifts
               SAL AX,CL         ;left justify data
               OUT DX,AX         ;write to D/A #0
```

1.6 Selecting Output Ranges

The operating output range is selected by jumpering pins on the rear D connector mating half. The various ranges are selected as follows:-

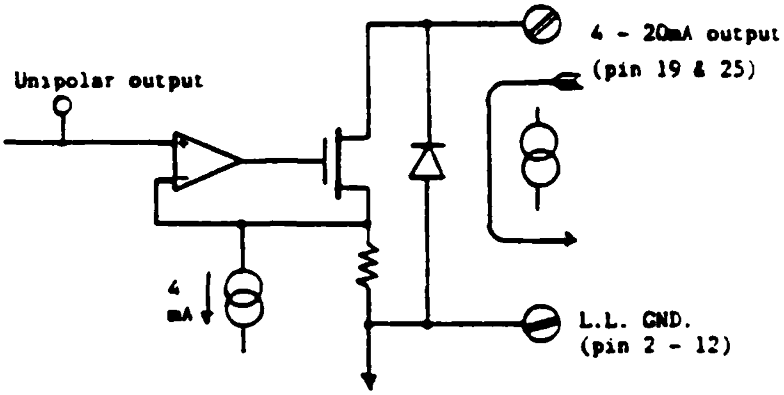
RANGE		JUMPER PINS	OUTPUT
0 - +5v	D/A #0:	21 to 22	Pin 24
	D/A #1:	15 to 16	Pin 18
0 - +10v	D/A #0:	20 to 22	Pin 24
	D/A #1:	14 to 16	Pin 18
-/+5v	D/A #0:	21 to 22	Pin 23
	D/A #1:	15 to 16	Pin 17
-/+10v	D/A #0:	20 to 22	Pin 23
	D/A #1:	14 to 16	Pin 17
4-20mA	D/A #0:	21 to 22	Pin 25
	D/A #1:	15 to 16	Pin 19
A.C. Ref.	D/A #0:	Input on pin 22	Pin 24 (2 quadrant)
		"	Pin 18 (4 quadrant)
"	D/A #1:	Input on pin 16	Pin 23 (2 quadrant)
		"	Pin 17 (4 quadrant)

Low level ground connections (common) can be made to any of the pins 2 thru 12.

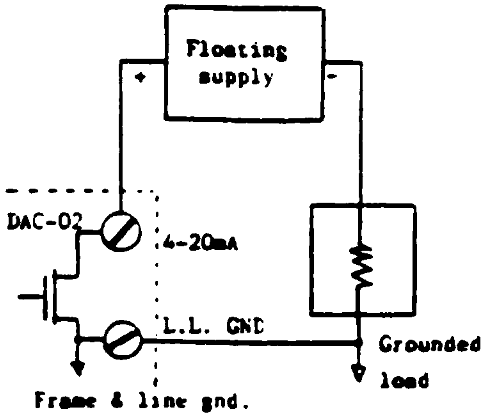
For unipolar outputs 0 - 5v, 0 - 10v and 4 - 20mA data coding is true binary. Due to the analog inversion in the bipolar output ranges, -/+5v and -/+10v, data coding is complementary offset binary i.e. zero digital corresponds to +Full Scale analog and 4095 digital corresponds to -Full scale analog.

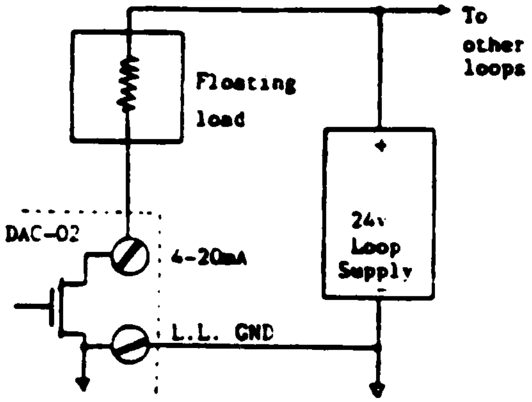
1.7 4-20mA Current Loop Output

The 4 - 20mA current loop output consists of a precision current sink formed by a VMOS power F.E.T. and reverse protection diode as shown below:-



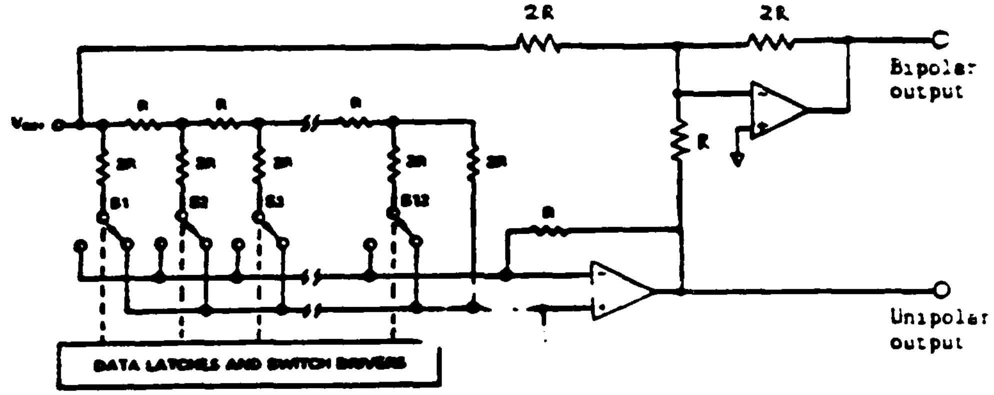
A minimum voltage of 8 volts must be maintained across this output circuit to insure correct operation. The maximum voltage should not exceed 36 volts for power dissipation reasons. A 24v or 36v loop supply is ideal. There are 2 ways of connecting the process loop, grounded load with floating supply or floating load with grounded supply. Obviously the second method allows many loops to be powered by the same supply but constrains the load to be 2 wire floating. The alternative connections are shown below:-





1.8 Use with A.C. Reference (Digital Attenuator)

Apart from its uses as a standard D.C. output D/A, the DAC-02 can be used with a bipolar or A.C. reference signal. The equivalent circuit when used in this mode is shown below:-



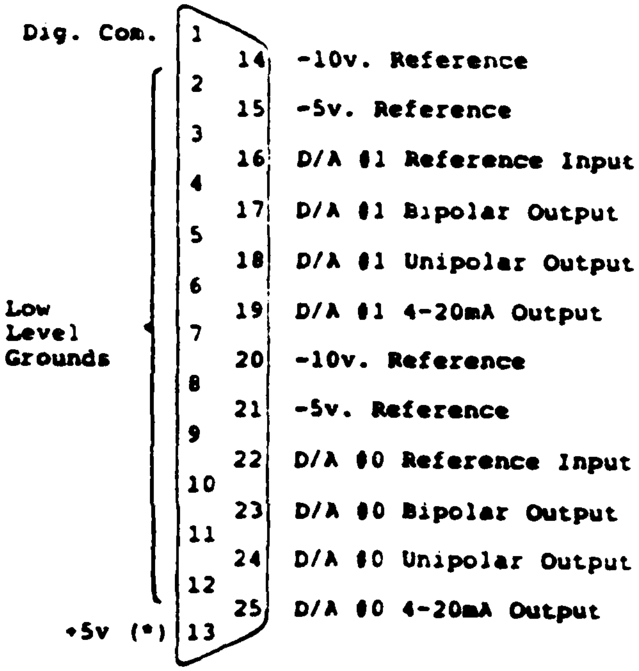
With an A.C. reference, the customary terminology of operation is somewhat different. If the output is taken from the unipolar outputs, 2 quadrant operation is obtained since the reference which may be positive or negative is multiplied with a positive only digital signal. If the output is taken from the bipolar output, the offset digital input can effectively be positive or negative which together with the possible positive or negative states of the reference results in 4 quadrant operation.

Two other parameters are of interest in A.C. operation. The first is feedthrough, the amount of residual signal at digital zero. The feedthrough which is mainly a function of stray capacitance rises with frequency. At 10KHz it is typically 5mV peak - peak with a +/-5v reference. The second parameter that is a limit at a lower frequency, is the accuracy/frequency characteristic. Due to distributed capacitance in the R-2R ladder network, the full 12 bit performance of the D/A falls off as the frequency rises. Above about 1KHz the dynamic performance of the D/A will be less than 12 bit accurate.

The DAC-02 will perform well in synchro-digital and resolver applications for sine/cosine generation with 400 Hz reference.

1.9 Connections

A rear view of the 25 pin D connector is shown below. The DAC-02 board has a female DB25 socket and a DB25P solder cup plug is required to make connections (MetraByte part # SMC-25). Usually only 3 or 4 wires (D/A outputs and ground) will be required for connections, so that a multi-wire flat cable is not required. (Note: 25 pin D connectors are identical to RS-232C connectors). Output range selection is controlled by jumpering pins on the plug body as described in Section 1.6.



* 5v power from the computer is supplied on Pin 13. If you use this power avoid shorting or overloading of the computer power supply.

2.0 Specifications

POWER SUPPLIES

+5v supply:	75mA typ., 100mA max.
-5v supply:	Not used
+12v supply:	15mA typ., 25mA max.
-12v supply:	25mA typ., 35mA max.
Total power dissipation:	0.85 watt typical.

OUTPUT RANGES

Channels:	2
I/O address:	DIP switch selected on any 8 bit boundary.
Resolution:	12 bits (1 part in 4095)
Relative accuracy:	1/2 LSB (0.01%) max.
Differential linearity:	1/2 LSB max.
Fixed reference ranges:	0 to +5v (unipolar) 0 to +10v (unipolar) +/-5v (bipolar) +/-10v (bipolar) 4-20mA current loop
Variable reference ranges:	+/-10v (2 or 4 quadrant)
Reference input resistance:	7Kohm min., 11Kohm typ., 20Kohm max.
Voltage output impedance:	< 0.1 ohm max.
Voltage output: drive current	+/-5mA min.
4-20mA compliance: (for current loop)	8 - 36v

ENVIRONMENTAL

Temperature coefficient: of gain	+/-25 ppM/deg.C.(with reference) +/-5 ppM/deg.C. (external ref.)
Zero drift:	+/-3 ppM/deg.C
Operating temperature:	0 - 70 deg.C.
Storage temperature:	-55 to +125 deg.C.
Humidity:	0 - 95% non-condensing
Weight:	4 oz. (120 g)

Appendix 9

Abstract

Effrece, Frank Jr., 2Lt, USAF, MS., June 1987, Mech. Engr.
Ohio University

The Dynamic Controls of a Hydraulic Press by Controlling the Pump Motor. (132pp.)

Director of Thesis: Dr. Kenneth R. Halliday

Because of the need to control power, a research project to control a 25 ton Walbash hydraulic press was undertaken. A pump-motor control scheme was selected to control this press, after considering adaptability, cost, and time response of the system, over a servo valve control system.

The analytical analysis of the pump-motor control scheme assumed linear characteristics for the press and motor which lead to a differential equation to describe the system:

$$\frac{J}{K_h}\frac{d\,V_r(t)}{d\,t} - \left(\frac{K_i\,K_p\,K_l}{R_a} - \frac{K_i^2}{R_a\,K_h}\right) V_r(t) = \frac{K_i\,K_p}{R_a}\,e(t) \quad [10]$$

where J is the inertia, R_a is the armature resistance, K_l is the stall torque of the motor, K_h is the proportionality constant of the press, K_p is the motor controller constant, K_l is the position transducer constant, V_r is the velocity of the ram, and e(t) is the input signal to the control system.

The actual control system was installed on the Walbash press using analog to digital and digital to analog

converters for computer linking. The control system produced a completely computer controlled press, except for the extra functions of the press, such as temperature control, which were not disturbed.

The control system worked satisfactory but not without a number of problems: numerical differentiation errors, magnitude of the velocity instabilities, and hydraulic inconsistencies. The press's futures is dependent on the user's tolerance to these errors.

Approved Kenneth R. Holliday