Abstract

Two control workstations are currently available at Bradley University for controller algorithm development and testing. The systems are provided by Quanser Consulting at a cost of $5,000 a piece. In addition, each Quanser system requires a PC with an internal A/D and D/A converter with a cost of approximately $2,000. The goal of the project is to design a similar system with an 8051 8-bit microcontroller. A combination of hardware and software is used to implement the new design. The user inputs the appropriate controller algorithm through a keypad and LCD. Currently, functionality for a P or PI controller is provided by the microcontroller system. The final cost of the microcontroller system is $400 with a total cost savings of approximately $6,600.

Design of a Control Workstation for Controller Algorithm Testing

Final Report

Aaron Mahaffey
Dave Tastsides

Advisor: Dr. Dempsey

May 6, 2003

Table of Contents

Table of Figures

1 Document Purpose

The purpose of the report is to inform the reader of the final status of the senior design project *"Design of a Control Workstation for Controller Algorithm Testing."* After reading the report the reader is able to proceed with the necessary design steps to complete the project. The report contains only the final design and results. All intermediate design steps and results are provided in the laboratory notebooks. The reader should be familiar with control systems and the application of control systems to electronics and electromechanical systems.

2 Project Summary

The objective of the project is to design a low cost control workstation based on the 8051 EMAC development board, a DC motor/rotary encoder and an external load consisting of a DC generator and power dissipation resistance. Currently, similar control stations cost up to $5,000 apiece. The test station progresses through a number of stages with the overall objectives of minimizing cost and achieving acceptable performance characteristics.

2.1 System Block Diagram

Figure 1 is a block diagram of the control workstation.

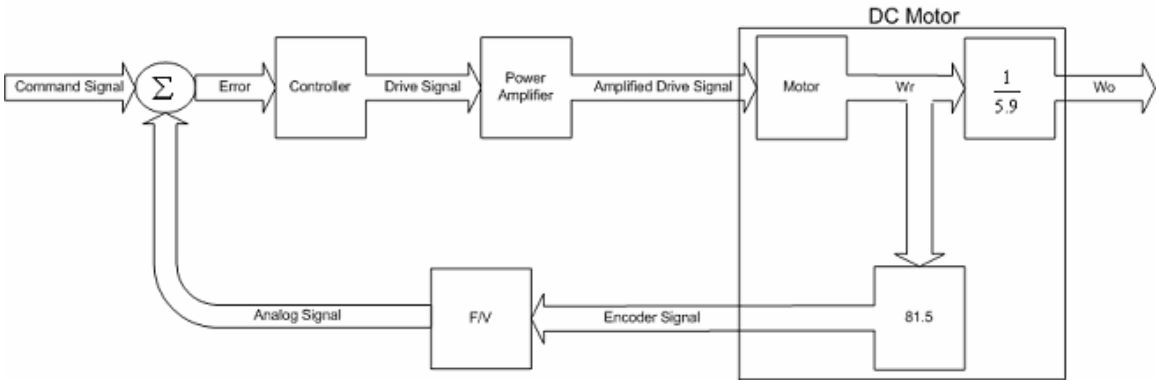

Figure 1 - Block Diagram of Control Workstation

The following sections are detailed descriptions of the components in Figure 1.

2.1.1 Summer

The command signal is specified by the user and controls motor speed. The analog signal is a feedback signal proportional to motor speed. The analog signal is subtracted from the command signal by the summer to produce the error signal. The error signal is used to drive the motor at the desired speed.

2.1.2 Controller

The controller is used to implement the controller algorithm to be tested. The user enters a control algorithm and the algorithm is implemented digitally on the 8051 EMAC development board. The system has provisions to implement proportional (P) or proportional-integral (PI) controllers.

2.1.3 Power Amplifier

The power amplifier amplifies the signal from the controller and drives the motor. The power amplifier is implemented with a combination of operational amplifiers and transistors to provide current and voltage boost.

2.1.4 DC Motor

The DC motor is the plant of the system. The motor is driven by the power amplifier output and the controller determines the operation. An internal gear reduction of 1/5.9 is present in the motor. Additionally, a rotary encoder is available and produces a frequency proportional to revolutions per minute (RPM). Thus, a gain block of 81.5 converts RPM to frequency.

2.1.5 Frequency-to-Voltage Converter

The frequency-to-voltage (F/V) converter is used to produce an analog voltage proportional to rotary encoder frequency. The proportional analog signal provides feedback to the system.

2.2 Control Block Diagram

Figure 2 is a control block diagram of the final system.

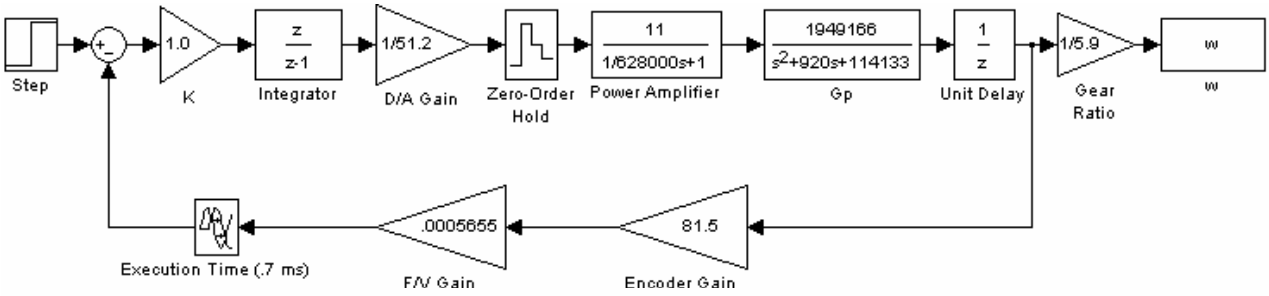

Figure 2 - Control Block Diagram of Final System

All blocks of the diagram are digital except for the power amplifier and plant. The following sections explain the subsystems found in Figure 2.

3 Hardware

The following sections are an overview of the hardware designed and tested for the final product.

3.1 Level Shifter

To implement the digital controller a signal is sent from the 8051 development board D/A converter. The output range of the D/A converter is 0-5 V. For bidirectional drive a ±2.5 V signal is required to drive the motor. Therefore, a level shifting circuit subtracts 2.5 V from the D/A converter signal. The software compensates for the -2.5 V shift to implement the control system correctly. The schematic is shown in Figure 3.

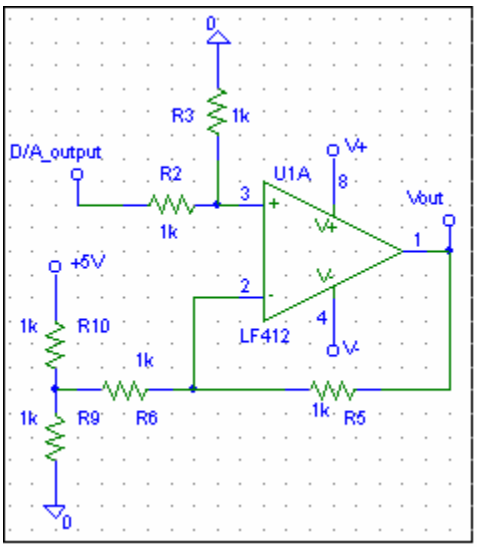

Figure 3 - Schematic of Level Shifter

3.2 Power Amplifier

The power amplifier is used to provide current and voltage boost for the DC motor. An overall gain of 11 is used. With ±2.5 V command inputs ±27.5 V drive is possible. Figure 4 is a schematic of the power amplifier.

The transistors are used to provide additional voltage boost because of the limitations on the output of the LF412 operational amplifier. The components R47, R48 and C19 form a lag network to stabilize the power amplifier with a phase margin of 81º and crossover frequency equal to 61.2 kHz. Resistors R3 and R7 sets the transistor voltage amplifier to a maximum gain of two. The resistors are necessary because of the high open loop gain of the operational amplifier (100,000 typical). The diodes at the output are used to create crossover distortion to ensure that the output transistors are not on at the same time.

A Matlab simulation of the power amplifier is in Figure 5. The m-file is found in Appendix B. The power amplifier is modeled by observing the open loop frequency response in PSPICE. By determining a simplified open loop model, cascading it with the

passive lag network used to stabilize the device, and closing the loop, a closed loop step response was produced.

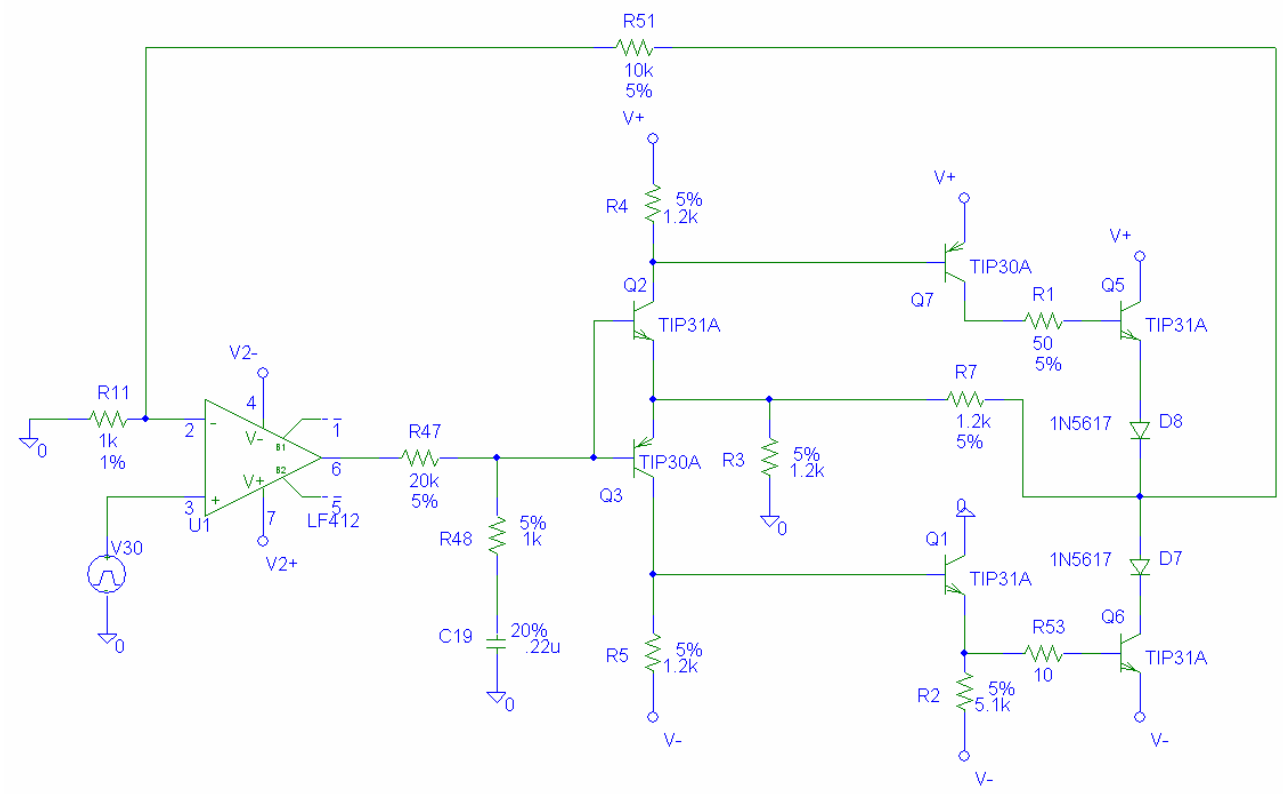

Figure 4 - Schematic of Power Amplifier

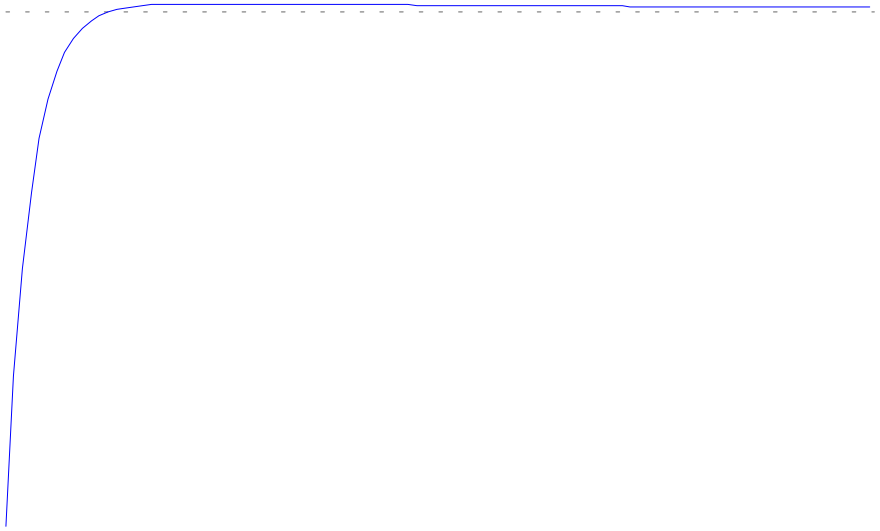

Figure 5 - Closed Loop Step Response of Power Amplifier Model

From the simulation the time constant is approximately 10 us. Thus, a pole is located at 628,000 rad/sec. The transfer function of the power amplifier is as follows:

$$G(s) = \frac{11}{s/628{,}000 + 1}$$

3.3 Plant

The plant is a Pittman 30.3 Volt GM9x36 DC motor. From the datasheet the following transfer function is derived:

$$G_p(s) = \frac{1{,}949{,}166}{s^2 + 920s + 114{,}133}\left(\frac{rad/\sec}{volt}\right)$$

The circuit used to derive the transfer function is in Figure 6.

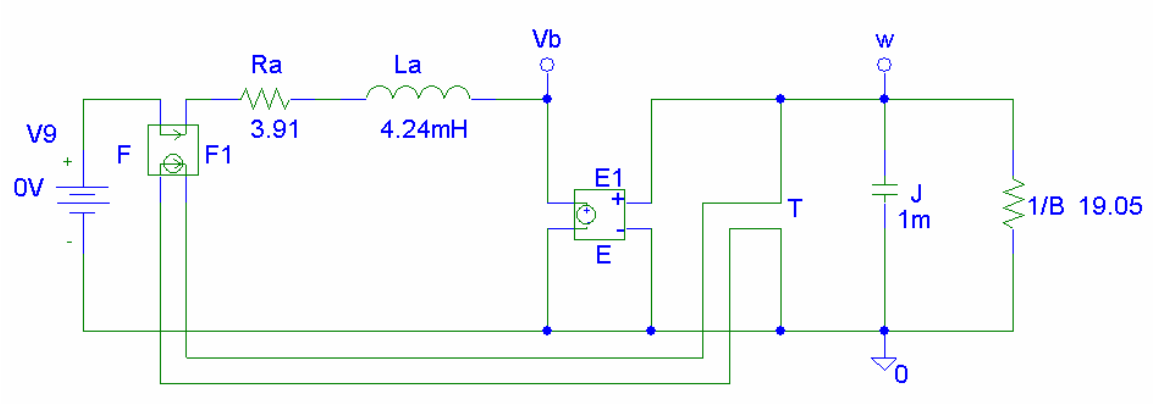

Figure 6 - Schematic of Motor Model

A unit delay is discovered in the system through experimental results. The unit delay is due to the time delay of the DC motor. The time delay of the motor is near the sampling period used experimentally (4 ms). Thus, a one sample delay occurs before the system feedback takes affect.

3.4 Digital-to-Analog Converter

The D/A converter is modeled with two blocks in Figure 2 shown as the "D/A Gain" and "Zero-Order Hold." The D/A gain is:

$$\frac{5Volts}{256Bits} = \frac{1Volt}{51.2Bits}$$

The zero-order hold is a model of the device dynamics and is equal to

$$G_{ho}(s) = \frac{1 - e^{-s*T}}{s}$$

where T is the sampling period.

3.5 Rotary Encoder

The rotary encoder is mounted with the DC motor package and is used to provide velocity feedback information. Each revolution of the internal motor shaft produces 512 pulses at the encoder output. The rotary encoder is modeled as a constant gain of 81.5 in Figure 2. The gain is derived via the following equation:

$$\frac{512Pulses}{\mathrm{Re}volution} * \frac{\mathrm{Re}volution}{2*\Pi Rad} = 81.5\frac{Pulses}{Rad}$$

4 Software

The following sections describe the software components of the system.

4.1 Controller

The controller implements a motor tracking system where the output shaft velocity follows the corresponding command input to the system. The motor is controlled at a constant velocity for various load changes from 762 RPM in the negative direction to 762 RPM in the positive direction (762 RPM corresponds to 27.5 V input to motor) with zero steady state error. Since the user may enter different controllers via the keypad, a specific controller design is not obtained. However, simulations are performed to compare experimental and expected results with a sampling period of 4 ms.

The following recursive equation is used to implement the digital controller shown in Figure 2:

$$(\mathrm{Pr}\,esentOutput) = K * (\mathrm{Pr}\,esentInput) + (PastOutput)$$

The equation represents a digital integrator with proportional gain.

4.1.1 Proportional Gain

The proportional gain determines the response by the motor to step inputs. Entering a low proportional gain implements a slower controller with no overshoot. As proportional gain is increased, the system contains more overshoot and is faster. However, entering too high of a proportional gain causes the system to be unstable.

4.1.2 Integrator

The integrator drives steady state error to zero to obtain a perfect tracking system. The plant in the system is type zero. Thus, an integrator is provided to eliminate steady state error for step inputs. Recall the user may choose not to implement the integrator. Test results are shown with the integrator since perfect tracking is desired. The provision to omit the integrator is present to show the system does not perform ideal tracking without the integrator.

4.1.3 Controller Results

Experimental and simulated results were obtained and compared for proportional gains of 1, 0.2, and 5. Experimentally for K=1 the system has an overshoot of 16.4% and t_p=60 ms. In simulations the system has an overshoot of 15.15% and t_p=55 ms (see Figures 10 and 11 in Appendix C). The experimental results match well with the simulated results. For K=0.2 the system has no overshoot in simulations and experimentally (see Figures 12 and 13). Finally, the system is unstable for K=5 in simulations and experimentally (see Figures 14 and 15). Note Figure 15 also shows the D/A converter output continuously saturates due to instability. The D/A converter is one of the practical limitations that does not allow the output to approach infinity for an unstable system.

The digital controller design is also shown in Appendix B as digital_controller.m. The file generates both the open loop frequency response and closed loop step response curves shown in Figures 16 and 17 respectively. Matlab results show the system has a

phase margin of 53.5º at a crossover frequency of 40.73 rad/sec. The closed loop step response has an overshoot of 14.3%.

4.2 Frequency-to-Voltage Converter

The F/V converter is used to convert the frequency of the rotary encoder signal to a proportional analog voltage. The voltage is used as feedback to produce the error signal. In addition, the F/V converter is used to calculate the motor RPM for display to the user.

The F/V converter is implemented in software by measuring the pulse width of the rotary encoder signal. Timer 2 of the 8051 is setup as a timer to measure the number of clock cycles between transitions of the signal. On a negative edge timer 2 is auto reloaded with 0x0000. When a positive transition occurs the value of timer 2 is captured and stored in the compare/capture registers. The captured value is then used to calculate the analog feedback voltage and motor rpm via the following equations:

$$Voltage = \frac{1531}{PulseWidth}$$

$$RPM = \frac{9152}{PulseWidth}$$

The proceeding equations are derived assuming ±27.5 Volt drive only.

The gain of the F/V converter is calculated as follows:

$$\frac{762 * 5.9}{2.5} = 1798.32\left(\frac{RPM}{Volt}\right)$$

$$\frac{9152RPM}{PulseWidth} * \frac{1Volt}{1798.32RPM} * \frac{51.2Bits}{1Volt} = \frac{260.5952}{PulseWidth}$$

$$Bits = Frequency * 2 * 1.085us * 260.5952$$

$$\frac{Bits}{Frequency} = .0005655 = F/VGain$$

4.3 Summer

The summer subtracts the analog feedback voltage from the command signal to produce the error signal. Direction of the motor is required since bidirectional drive is used. The direction is determined by the rotary encoder signal. The rotary encoder provides two signals that are out of phase. The motor direction is determined by the signal that occurs first in the sequence. Thus, a simple D flip-flop is used to create a direction bit. Figure 7 is the schematic of the D flip-flop.

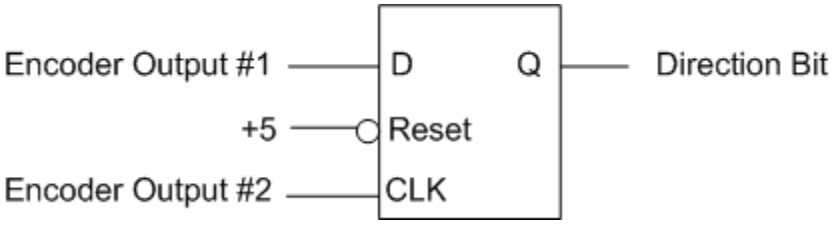

Figure 7 - Schematic of Circuit to Produce Direction Bit

One rotary encoder signal is used as a clock and the other rotary encoder signal is used as the input. If the output of the D flip-flop is high then the motor is moving in the forward direction. If the motor is moving in the reverse direction then the output of the D flip-flop is low.

4.4 User Interface

The user interface is used to communicate with the user. All parameters are entered by the user during the user interface routine. The user is asked for the following parameters (in order):

1. Sampling Period in us
2. Proportional Gain in 1/255 increments
3. Integration
4. Command Signal Magnitude and Sign

The sampling period is converted to a timer reload value and timer 0 is setup to create the sampling period. All sample driven functions are called from the timer 0 interrupt service routine.

5 Future Project Work

Section 5 outlines additional work to improve upon the project. First, provisions are added to implement more complex controllers with multiple poles and zeroes. Future students can test the system with ramp or impulse command inputs since the current system only accepts step inputs. Finally, other devices are tested with the control workstation in the form of position or velocity control of different plants.

Appendix A
Function Listing

user_input user_interface(void)
Gathers required information from user. Returns the error status and interrupt time requested by the user.

void bsp_timer0_init(timer0_setup timer0_type)
Initializes timer 0 according to the parameters set in the timer0_setup structure.

void bsp_timer0_start(void)
Runs timer 0.

void bsp_timer0_stop(void)
Stops the operation of timer 0.

void bsp_timer0_set_reload_value(unsigned_16 reload_value)
Sets the reload value for timer 0.

void bsp_timer2_init(timer2_setup timer2_type)
Initializes timer 2 according to the parameters set in the timer2_setup structure.

void bsp_timer2_start(timer2_setup timer2_type)
Runs timer 2.

void bsp_timer2_stop(void)
Stops the operation of timer 2.

void bsp_timer2_set_reload_value(unsigned_16 reload_value)
Sets the reload value for timer 2.

void summer(void)
Performs the summing operation to produce the error signal. The result is stored in the global variable sum.

void bsp_atod_init(atod_setup atod_type)
Initializes the A/D converter according to the parameters set in the atod_setup structure.

unsigned_8 bsp_atod_read(atod_setup atod_type)
Reads and returns the value of the specified A/D converter channel.

void bsp_keypad_init(void)
Initializes the keypad for use.

unsigned_8 bsp_keypad_read(void)
Reads the value entered by the user through the keypad. The result returned is the hex number corresponding to the key pressed.

void bsp_init_lcd(void)
Initializes the LCD display.

void bsp_lcd_mode(void)
Sets the mode of the LCD.

void bsp_lcd_display(void)
Activates the LCD display.

signed_16 bsp_lcd_getpos(void)
Returns the current position of the cursor.

unsigned_8 bsp_lcd_setpos(unsigned_8 row, unsigned_8 column)
Sets the position of the cursor. Returns successful or unsuccessful execution.

void bsp_lcd_home(void)
Clears the LCD screen and returns the cursor to the home position.

void bsp_lcd_putchar(unsigned_8 chardata)
Displays a character on the LCD display.

unsigned_8 bsp_lcd_getchar(void)
Returns the current character on the LCD display.

void bsp_lcd_cdshift(void)
Shifts the LCD display.

unsigned_8 bsp_lcd_busy(void)
Checks to see if the LCD is busy. A nonzero return value indicates the LCD is busy.

void bsp_lcd_putstr(signed_8 *strdata)
Displays a string of data on the LCD screen.

void bsp_lcd_backlight(unsigned_8 enable)
Enables or disables the LCD backlight.

void proportional_gain(void)
Performs the proportional gain function of the controller. The result is stored in the global variable controller_output. The gain desired is stored in the global variable gain.

void integrator(void)
Performs the integrator function of the controller. The result is stored in the global variable controller_output. Past controller outputs are stored in the global array past_controller_outputs[].

void main(void)
Performs the main routine of the program.

void timer0_isr(void) interrupt 1
Interrupt service routine for timer 0. All sample driven functions are called from the routine.

unsigned_8 bsp_dtoa_write(unsigned_8 channel, unsigned_8 value)
Writes the given value to the D/A channel specified. Returns successful or unsuccessful write.

void hex2ascii(unsigned_16 number)
Converts a 16 bit hex number to ASCII characters for display on the LCD. The results are stored in the global array ascii_conversion[].

unsigned_8 level_shift(unsigned_8 sign,unsigned_8 number1,unsigned_8 number2)
Used to level shift the command signal by 2.5 Volts (0x80). Required for bidirectional drive.

unsigned_16 timer_reload_value(unsigned_32 interrupt_time)
Calculates the 16 bit timer reload value for the given interrupt time in microseconds.

unsigned_16 rpm_calc(void)
Calculates and returns motor RPM based on the pulse width measurement.

unsigned_8 fv_voltage_calc(void)
Calculates and returns the analog feedback voltage for the summer operation.

Appendix B

Matlab M-files

```
%Power amplifier controller --> passive lag network
gcnumamp = [1/4545 1];
gcdenamp = [1/216 1];

%Estimated open loop transfer function of power amp from PSPICE simulations
gpnumamp = [400000];
gpdenamp = [1/126 1];

%Combine power amp and internal controller
[numamp,denamp] = series(gcnumamp,gcdenamp,gpnumamp,gpdenamp);

%Feedback attenuation
hnumamp = 1;
hdenamp = 11;

%Close loop and generate step response
[numcamp,dencamp] = feedback(numamp,denamp,hnumamp,hdenamp,-1);
step(numcamp,dencamp)
```

Figure 8 - M-file for Power Amplifier

```
T = .004;   %set sampling period

%model of complete power amp loop with controller and feedback
gcnumamp = [1/4545 1];  %power amp controller
gcdenamp = [1/216 1];
gpnumamp = [400000];    %estimated power amp transfer function from PSpice
gpdenamp = [1/126 1];
[numamp,denamp] = series(gcnumamp,gcdenamp,gpnumamp,gpdenamp);
hnumamp = 1;        %feedback to give gain of 11
hdenamp = 11;
[numcamp,dencamp] = feedback(numamp,denamp,hnumamp,hdenamp,-1);

%model of motor with time delay
[tnum1,tden1] = pade(.003,4);   %motor time delay is 3 ms
gpnummot = 1949166;     %motor transfer function from spec sheet data
gpdenmot = [1 920 114133];
[gpnummot,gpdenmot] = series(gpnummot,gpdenmot,tnum1,tden1);    %add time delay
to tf
[gpnmot,gpdmot] = series(numcamp,dencamp,gpnummot,gpdenmot);    %cascade with
power amp
[gpnmotd,gpdmotd] = c2dm(gpnmot,gpdmot,T,'zoh');            %bring to digital plane

%model of F/V converter
encodergain = 81.5;     %internal rotary encoder conversion factor
```

```
FVgain = .0005655;   %F/V converter gain to get 2.5 V at max speed
hnum = encodergain*FVgain;  %conbine encoder gain and F/V gain
hden = 1;
[tnum2,tden2] = pade(.0007,4);
[hnum,hden] = series(hnum,hden,tnum2,tden2);
[hnumd,hdend] = c2dm(hnum,hden,T,'tustin');  %convert to digital plane

[gponum,gpoden] = series(gpnmot,gpdmot,hnum,hden);  %combine motor, power amp,
and F/V for O.L. response
[gponumd,gpodend] = c2dm(gponum,gpoden,T,'zoh'); %convert to digital plane

kdesign = 1;        %controller gain
dagain = 1/51.2;    %conversion factor of D/A converter
gcnumd = kdesign*dagain*[1 0];    %controller transfer function
gcdend = [1 -1];

[goln,gold] = series(gponumd,gpodend,gcnumd,gcdend);    %combine entire open loop
system
unit_delay_n = 1;   %add in unit delay
unit_delay_d = [1 0];
[goln,gold] = series(goln,gold,unit_delay_n,unit_delay_d);  %combine open loop system
with unit delay
w = 0:.1:200;   %set range for w
dbode(goln,gold,T,w);    %find frequency response of open loop system
[mag,phase,w] = dbode(goln,gold,T,w);
margin(mag,phase,w) %find phase margin and crossover frequency

[gcln,gcld] = series(gpnmotd,gpdmotd,gcnumd,gcdend);    %combine controller with
motor and power amp
unit_delay_n = 1;   %add in unit delay
unit_delay_d = [1 0];
[gcln,gcld] = series(gcln,gcld,unit_delay_n,unit_delay_d);  %combine closed loop system
with unit delay
[clnumd,cldend] = feedback(gcln,gcld,hnumd,hdend,-1);   %close loop for entire system
figure
dstep(clnumd,cldend)    %find closed loop step response
```

Figure 9 - M-file for Digital Control System

Appendix C
Controller Results

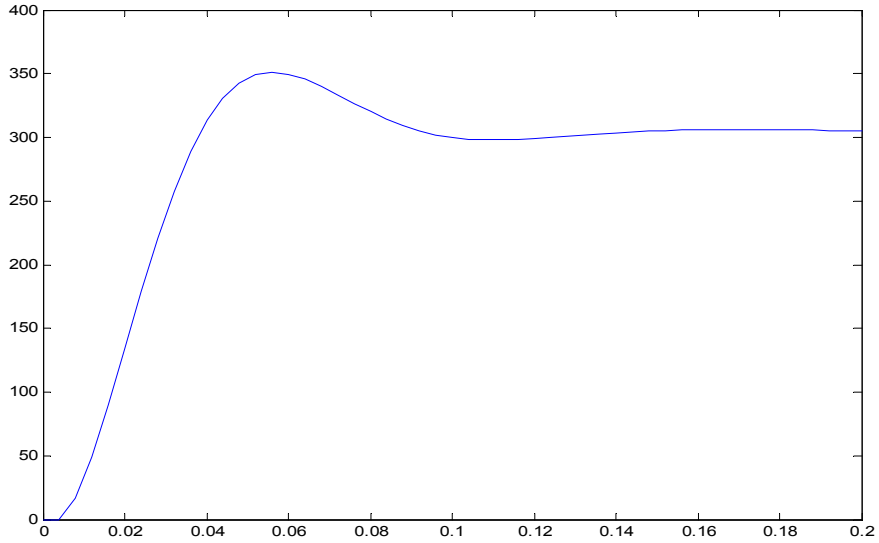

Figure 10 - Simulated Unit Step Response of System for K = 1

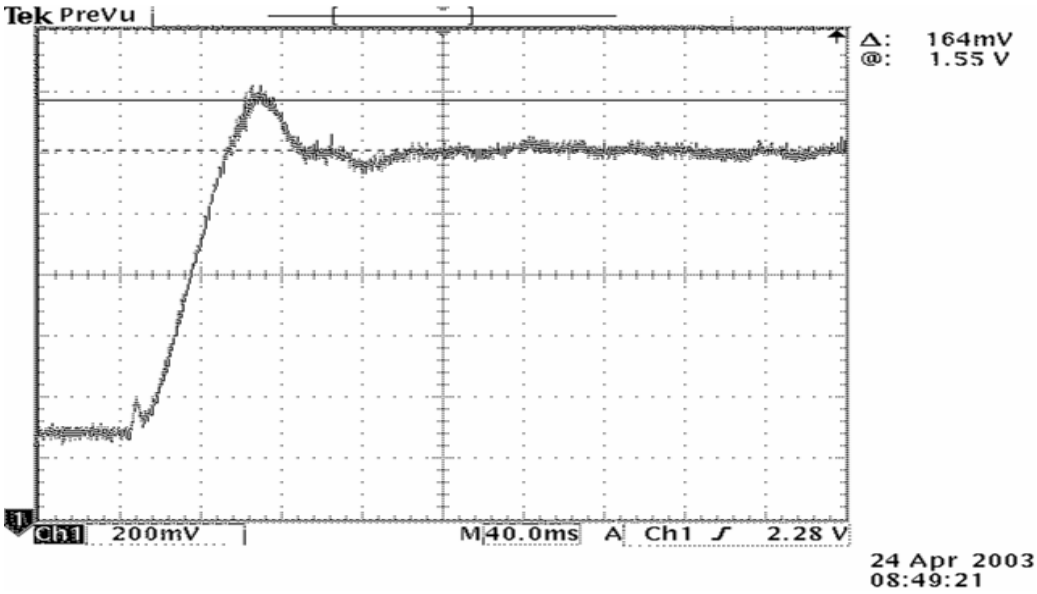

Figure 11 - Experimental Unit Step Response of System for K = 1

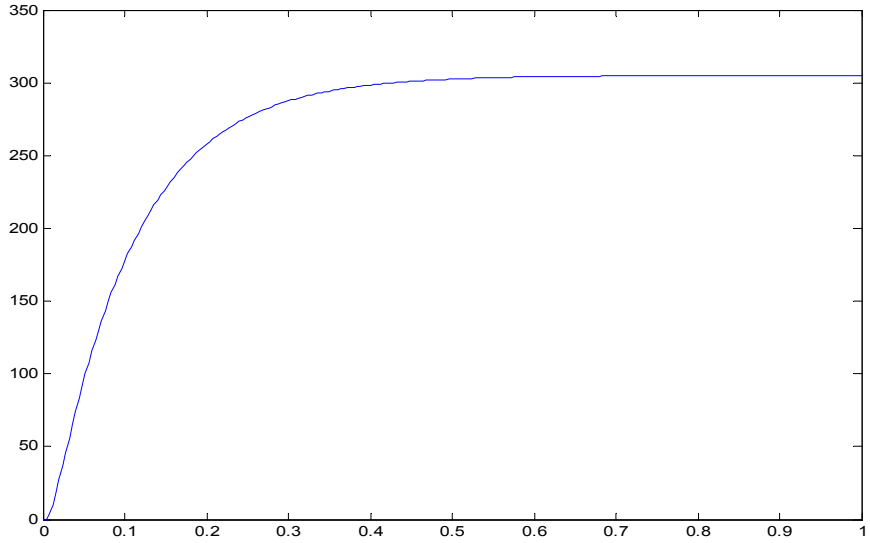

Figure 12 - Simulated Unit Step Response of System for K = 0.2

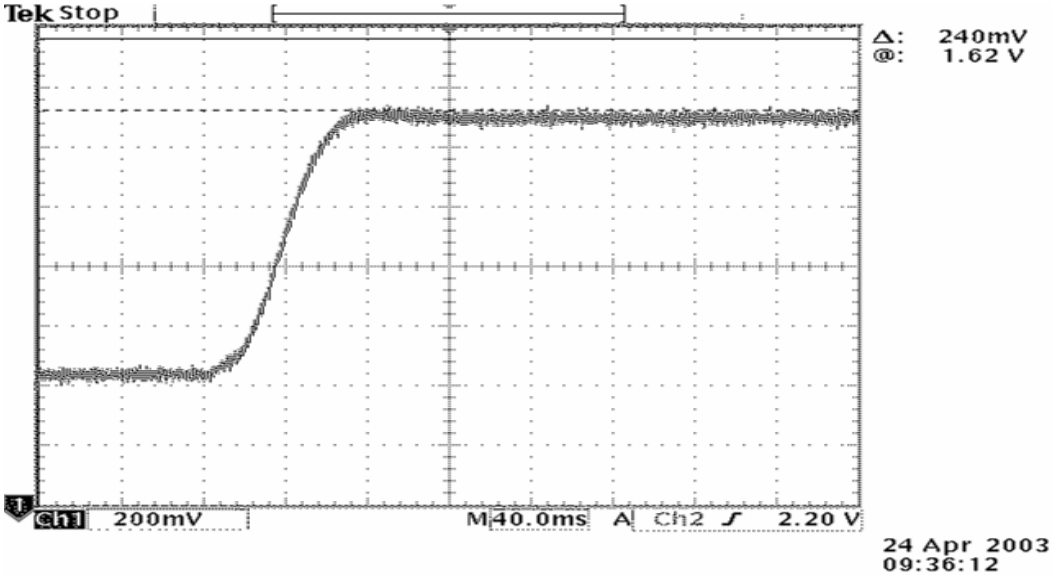

Figure 13 - Experimental Unit Step Response of System for K = 0.2

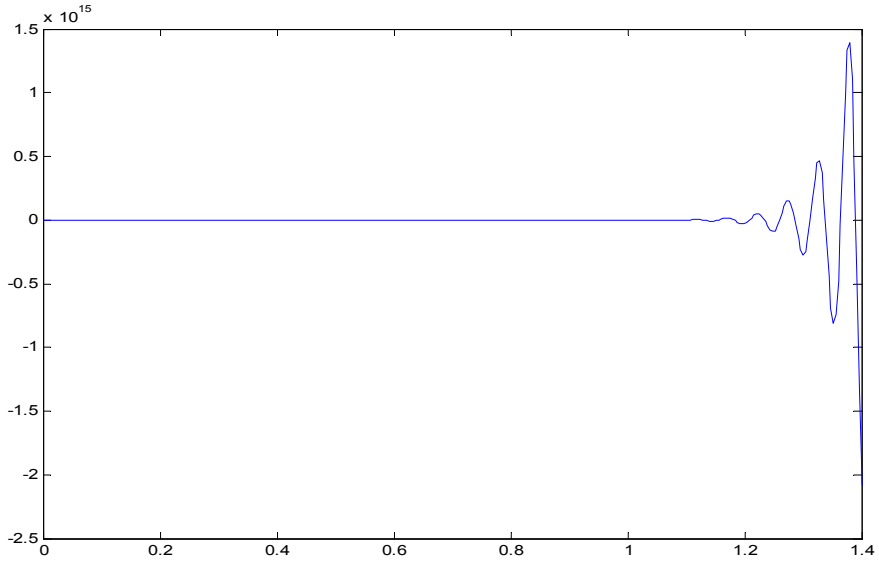

Figure 14 - Simulated Unit Step Response of System for K = 5

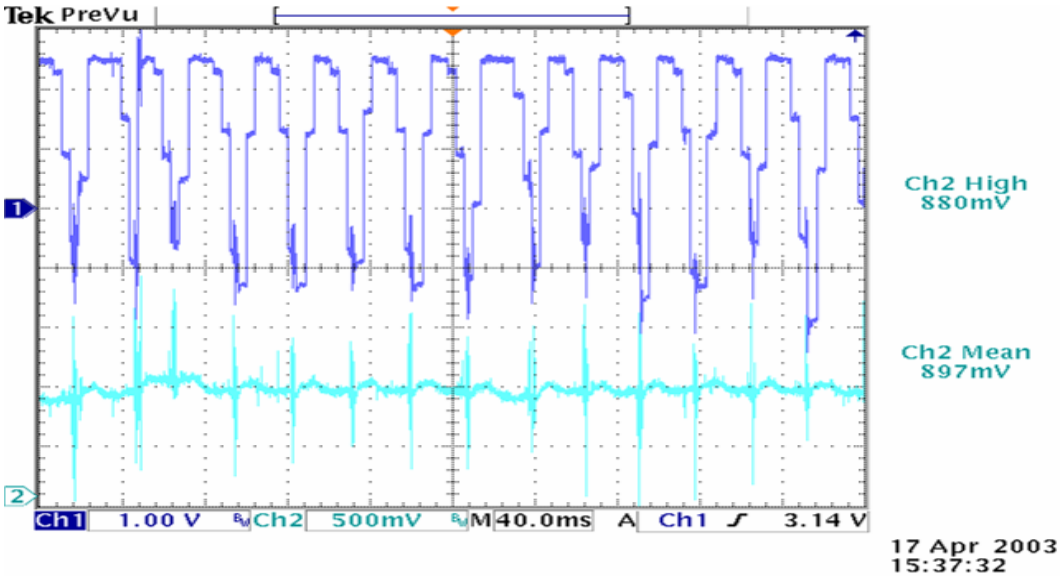

Figure 15 - Experimental Unit Step Response of System for K = 5

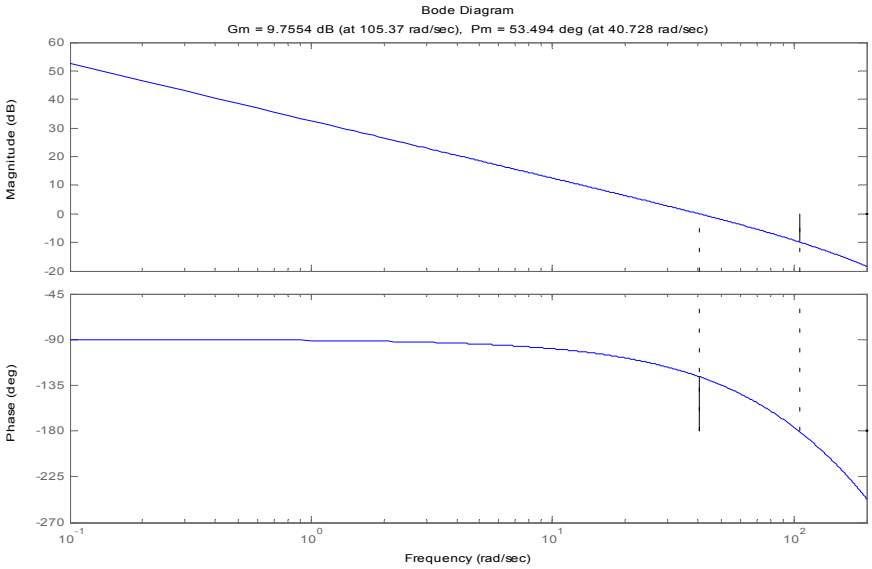

Figure 16 - Open Loop Frequency Response of Digital Control System

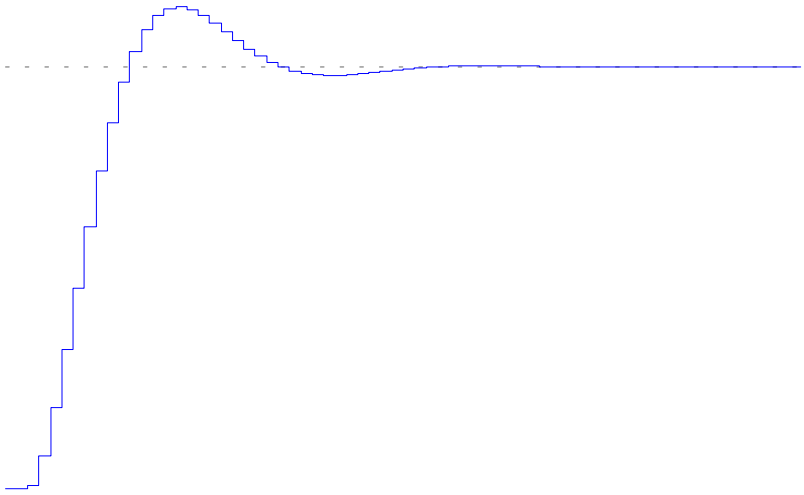

Figure 17 - Closed Loop Step Response of Digital Control System