PID Controller Using the TMS320C31 DSK for Real-Time DC Motor Speed and Position Control

Jianxin Tang
Division of Electrical Engineering
Alfred University
Alfred, NY 14802
Email: ftang@bigvax.alfred.edu

Rulph Chassaing
Dept. of Electrical and Computer Engineering
University of Massachusetts Dartmouth
North Dartmouth, MA 02747
Email: rchassaing@umassd.edu

ABSTRACT

This paper addresses real-time DC motor speed and position control using the low-cost TMS320C31 digital signal processing starter kit (DSK). A PID controller is designed using MATLAB functions to generate a set of coefficients associated with a desired controller's characteristics. The controller coefficients are then included in an assembly language program that implements the PID controller. A MATLAB program is used to activate the PID controller, calculate and plot the time response of the control system. When the MATLAB program is run, it plots the time response of the system on the PC screen, assembles the PID assembly language program, and loads/runs the resulting executable file on the TMS320C31 to achieve real-time control. Both speed and position control are investigated on a DC motor system with speed feedback and position feedback. Results show the improvement of system outputs as expected with a PID controller, with actual system outputs matching theoretical calculations.

I. INTRODUCTION

The rapid and revolutionary progress in power electronics and microelectronics in recent years has made it possible to apply modern control technology to the area of motor and motion control. The use of digital signal processors (DSPs) has permitted the increasingly stringent performance requirements and fast, efficient, and accurate control of servo motor and motion control systems. DSPs, such as the TMS320C31 (C31) from Texas Instruments, are currently used for a wide range of applications from controls and communications to speech processing. They continue to be more and more successful because of available low-cost support tools. DSP-based systems can be readily reprogrammed for a different application.

The C31-based $99 DSK includes Texas Instruments' C31 floating-point digital signal processor, and an analog interface circuit (AIC) chip with A/D and D/A converters, input (anti-aliasing) and output (reconstruction) filters, all on a single chip. It also includes an assembler, a debugger, and many application examples.

This paper addresses real-time DC motor control using the C31 DSK. A PID controller is designed using MATLAB functions to generate a set of coefficients associated with a desired controller's characteristics. The controller coefficients are then included in an assembly language program that implements the PID controller. A MATLAB program is used to activate the PID controller, calculate and plot the time response of the control system. When the MATLAB program is run, it plots the time response of the system on the PC screen, assembles the PID assembly language program, and loads/runs the resulting executable file on the C31 to achieve real-time control.

In Section II, design of the PID controller is discussed. Implementation of the PID controller using the C31 is addressed in Section III. Test results of the motor system and comparison with theoretical calculations are presented in Section IV. Finally conclusion and future work are given in Section V.

II. DESIGN OF THE DIGITAL CONTROLLER

Following is a block diagram of the system under consideration.

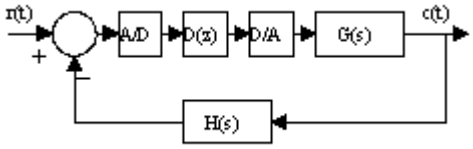


Figure 1. Block diagram of the digital control system

In figure 1, r(t) is the input (or set point), c(t) is the output, D(z) is the digital controller, G(s) is the plant transfer function, and H(s) is the sensor transfer function. A digital PID controller has the following form [2]:

$$D(z) = K_P + K_I \frac{T}{2}\frac{z+1}{z-1} + K_D \frac{z-1}{Tz}$$

$$= \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}} \qquad (1)$$

with

$$a_0 = K_P + \frac{K_I T}{2} + \frac{K_D}{T}$$

$$a_1 = -K_P + \frac{K_I T}{2} - \frac{2K_D}{T}$$

$$a_2 = \frac{K_D}{T},$$

where K_P, K_I, and K_D are the proportional, integral, and derivative parameters of the controller, and T is the sampling period. For programming convenience, let

$$D(z) = D(z)\frac{M(z)}{M(z)} = \frac{Y(z)}{X(z)}$$

where X(z) and Y(z) are the input and output of the controller in the z-domain, respectively, then

$$Y(z) = (a_0 + a_1 z^{-1} + a_2 z^{-2})M(z) \qquad (2)$$

and

$$X(z) = (1 - z^{-1})M(z)\,. \qquad (3)$$

Using the inverse z-transformation,

$$y(k) = a_0 m(k) + a_1 m(k-1) + a_2 m(k-2) \qquad (4)$$

and

$$m(k) = x(k) + m(k-1)\,. \qquad (5)$$

III. IMPLEMENTATION OF THE DIGITAL PID CONTROLLER USING THE C31 DSK

Figure 2 shows the MATLAB main program (CONTROL2.M) for a DC motor control system with a PID controller. PID parameters and amplifier/motor transfer functions are pre-calculated and inputted to the program. PID parameters can be calculated using equations in many standard digital control systems books (see, for example [2]). The MATLAB program in Figure 2 does the z-transform, calculates the system closed-loop transfer function and the step response. In the program "numg, deng" is the amplifier/motor transfer function, "numf, denf" is the sensor transfer function, "numd, dend" is the PID controller transfer function, "T" is the sampling period, "c2dm" does the z-transform using zero-order-hold, "series" multiplies the PID controller transfer function with the amplifier/motor transfer function, "feedback" calculates the closed-loop transfer function of the system, "nums, dens" adds an amplitude of 2.5 to the step input, and "dstep" calculates the step response of the system.

The actual implementation of the PID controller on the DSK is through the assembly language program

CONTROL2.ASM (figure 3). This program is a modification of an IIR filter program in [1]. After defining starting addresses for the text and data, the AIC communication routine AICCOM31.ASM [1] is included for input to and output from the C31 (via the I/O routine AICIO_P). The

```
%CONTROL2.M--Main MATLAB program
%for motor control using the
%TMS320C31 DSK
%
numg=22;
deng=[1 4];
numf=[0.18];
denf=[1];
T=0.0001;
[numz,denz]=c2dm(numg,deng,T,...
'zoh');
numd=[1.00004 -0.99996];
dend=[1 -1];
[numz,denz]=series(numd,dend,...
numz,denz);
[numzc,denzc]=feedback(numz,...
denz,numf,denf);
[nums]=[2.5];
[dens]=[1];
[numc,denc]=series(nums,dens,...
numzc,denzc);
c=dstep(numc,denc,10001);
t=0:0.0005:5;
dos('dsk3a control2');
dos('dsk3load control2 BOOT');
plot(t,c),grid
xlabel('Time (seconds)')
ylabel('System output (rps)')
title('Time reponse')
```

Figure 2. Main MATLAB Program CONTROL2.M

input sample is then processed according to equations (4) and (5). One of the features of the C31 is that it can perform multiplication and addition/subtraction in parallel. This is especially suitable for processing difference equations such as in (4) and (5). An output signal y(k) is generated and the delays m(k) are updated. Continuous processing takes place within a loop starting at the label PID in the program in figure 3. PID parameters are set in the statement with label COEFF. Initial delays are set to zero in the statement with label DLY. The sampling frequency is configured to be 10 KHz in the statement labeled AICSEC.

From the MATLAB program CONTROL2.M, the assembly language program CONTROL2.ASM is assembled, loaded into and run on the C31 on board the DSK. While the motor is running, the MATLAB program CONTROL2.M plots the time response of the motor system and displays the response on the PC monitor.

IV. TEST RESULTS

Tests were performed on a DC motor control system, the SFT154, manufactured by Feedback Inc. The system includes a power amplifier, a DC motor, a tachometer as the speed sensor, a potentiometer as the position sensor, and a data acquisition unit to display real-time system outputs on a PC monitor. Both speed control and position control were investigated.

A. Speed Control

The transfer function of the motor for speed control is

$$G_s(s) = \frac{22}{s+4}.$$

This transfer function was derived based on the time constant and the DC gain of the motor, which were obtained from a simple open-loop test. The speed sensor has a ratio of 0.18. The input signal was 2.5 volts, representing a desired output speed of 15 rps or 900 rpm.

1. The first test was to run the motor without a controller (D(z)=1). The output speed was only about half of the desired output as shown in figure 4a. This is due to the fact that the system is a type 0 system and large steady-state error exists. This result was also confirmed from theoretical calculations as shown in figure 4b.

2. The second test was then performed with a PID controller added, with PID parameters K_P=1, K_I=0.8, and K_D=0. This corresponds to a_0=1.00004, a_1=-0.99996, and a_2=0 in equation (1) since the sampling period T=0.0001. Notice that one cannot approximate a_0 with 1 and a_1 with -1 otherwise the integral part of the PID controller will not be in effect. This achieved a steady-state output speed of 900 rpm as desired and shown in figure 5a. In other words, the system type was increased by 1 and steady-state error was eliminated. However the rise-time is longer with the controller added, as confirmed from theoretical calculations as shown in figure 5b. The rise-time can be reduced by increasing K_I with a possibility of having an overshoot.

B. Position Control

The transfer function for position control is

$$G_P(s) = \frac{22}{s(s+4)}$$

which is $G_s(s) \times \frac{1}{s}$ (speed to position conversion). The position sensor has a ratio of 0.625. The system input was $\pm$5v, representing a desired output of $\pm$90 degrees (or a total of 180 degrees). Again, the system was run without (D(z)=1) and with a controller. Since the system is already of type 1, no steady-state errors were expected. Figure 6a shows that the system output had an overshoot of about 12% without a controller, which matched the corresponding theoretical results as shown in figure 6b. When a PID controller was added with parameters K_P=1, K_I=0, and K_D=1.9, the overshoot was reduced to zero (figure 7a), which matched the corresponding theoretical results (figure 7b).

V. CONCLUSION AND FUTURE WORK

A digital PID controller was successfully implemented using the C31 DSK and tested on a DC motor speed and position control system for real-time control. For speed control, the test results showed that with the PID controller added, the steady-state error was eliminated and the desired output speed was obtained. For position control, the results showed that, with the PID controller added, the desired output position was obtained without overshoot. Actual system outputs also agreed with theoretical results, indicating the accuracy of the system transfer functions. At the present time the controller parameters have to be set in the assembly language program CONTROL2.ASM and cannot be adjusted in real-time. Future work includes designing the PID controller within the MATLAB program CONTROL2.M and adjusting the PID parameters in real-time.

REFERENCES

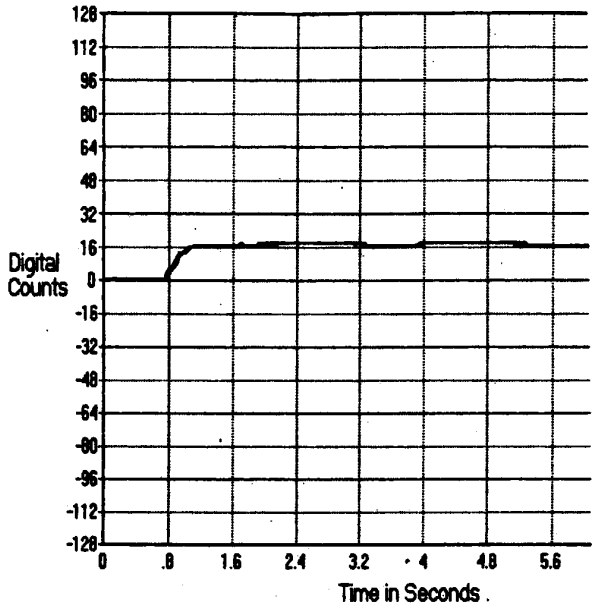
1. R. Chassaing, *Digital Signal Processing, Laboratory Experiments Using C and the TMS320C31 DSK*, Wiley, 1999.

2. C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design*, Prentice Hall, Inc., 1995.

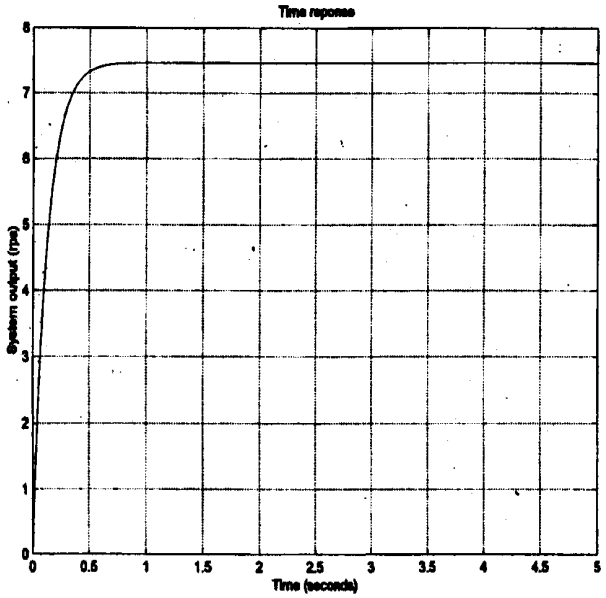
3. J. Tang, "Laboratory Development for a Digital Control System Course," Journal of Engineering Technology, Vol. 14, No. 2, fall 1997.

```
*CONTROL2.ASM - PID controller using the C31 DSK
        .start   ".text", 0x809900 ;starting address of text
        .start   ".data", 0x809C00 ;starting address of data
        .include "AICCOM31.ASM"    ;include AIC comm routines
        .entry  BEGIN              ;start of code
        .text                      ;assemble into text
BEGIN   LDP     @COEFF_ADDR        ;init to data page 128
        CALL    AICSET             ;initialize AIC
PID     LDI     @COEFF_ADDR,AR0    ;AR0 points to coefficients address
        LDI     @DLY_ADDR,AR1      ;AR1 points to addr of delay samples
        CALL    AICIO_P            ;call AIC for polling
        FLOAT   R6,R3              ;stage input
        MPYF3   *AR0++,*AR1++,R0   ;b[0]*dly[0]
        LDI     STAGES-1, RC       ;initialize stage counter
        MPYF3   *AR0++,*AR1--,R1   ;b[1]*dly[1]
||      SUBF3   R0,R3,R3           ;input-b[0]*dly[0]
        MPYF3   *AR0++,*AR1++,R0   ;a[1]*dly[0]
||      SUBF3   R1,R3,R2 ;dly=input-b[0]*dly[0]-b[1]*dly[1]
        MPYF3   *AR0++,*AR1--,R1   ;a[2]*dly[1]
        ADDF3   R0,R1,R3           ;a[2]*dly[1]+a[1]*dly[0]
        LDF     *AR1,R4            ;dly[2]
||      STF     R2,*AR1++          ;dly[0] = dly
        MPYF3   R2,*AR0++,R2       ;dly*a[0]
||      STF     R4,*AR1++          ;dly[1] = dly[0]
        MPYF3   *AR0++,*AR1++,R0   ;b[0]*dly[0]
||      ADDF3   R2,R3,R3           ;controller output
        FIX     R3,R7              ;convert output to integer
        BR      PID
        .data  ;b[0]       b[1]        a[1]        a[2]       a[0]
COEFF  .float  -1.0000E+0, 0.0000E+0, -0.99996E+0, 0.0000E+0, 1.00004E+0
DLY        .float  0, 0              ;init delay var for each stage
STAGES     .set    1                 ;number of stages
COEFF_ADDR .word   COEFF             ;address of COEFF
DLY_ADDR   .word   DLY               ;address of DELAY
AICSEC     .word   162Ch,1h,3872h,63h ;AIC config data, Fs = 10 kHz
           .end                      ;end
```

Figure 3. Assembly language program CONTROL2.ASM for a PID controller

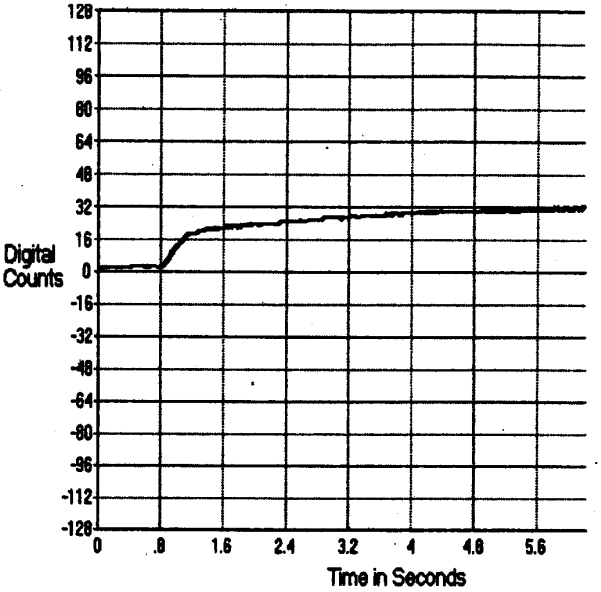


4(a). Actual motor speed (16 digital counts=1.25volts, which is equivalent to 450 rpm)

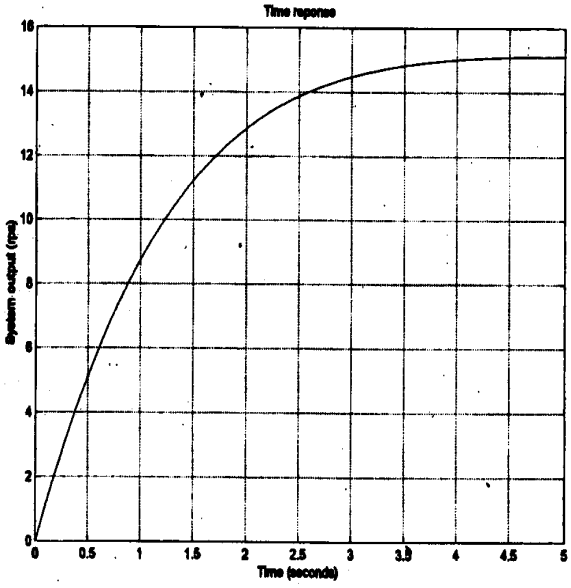


4(b). Simulation using MATLAB (output speed is 7.5 rps, or 450 rpm)

Figure 4. System output speed without a controller

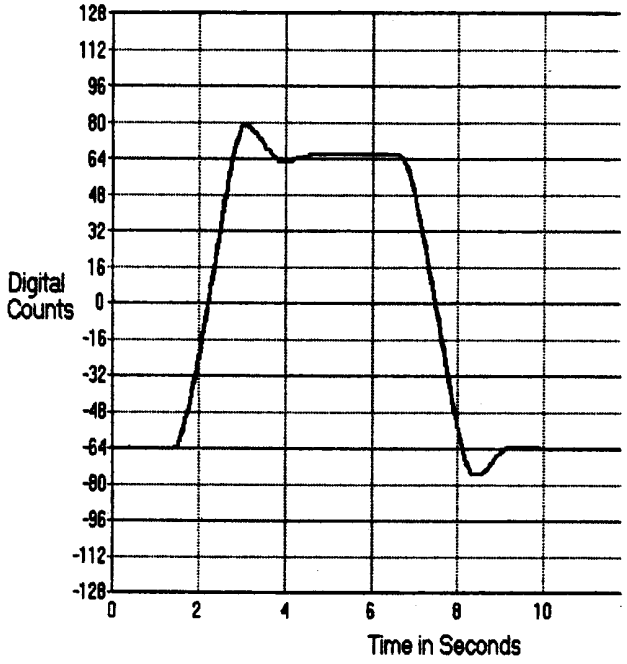


5(a). Actual motor speed (32 digital counts=2.5volts, which is equivalent to 900 rpm)

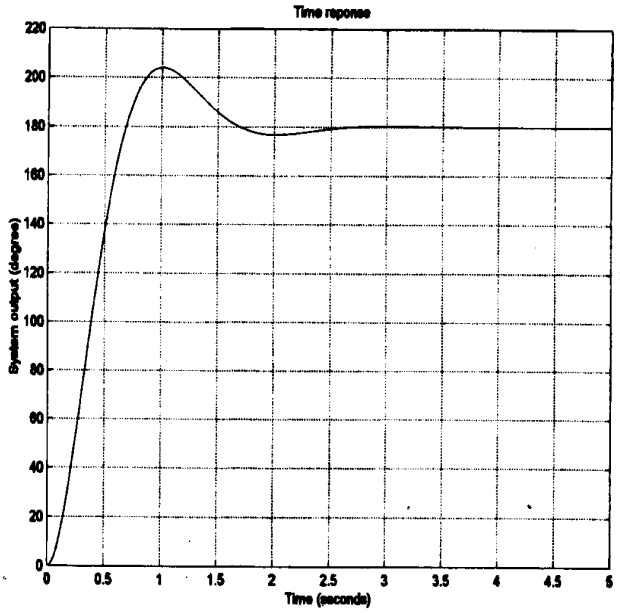


5(b). Simulation using MATLAB (output speed is 15 rps, or 900 rpm)

Figure 5. System output speed with a PID controller

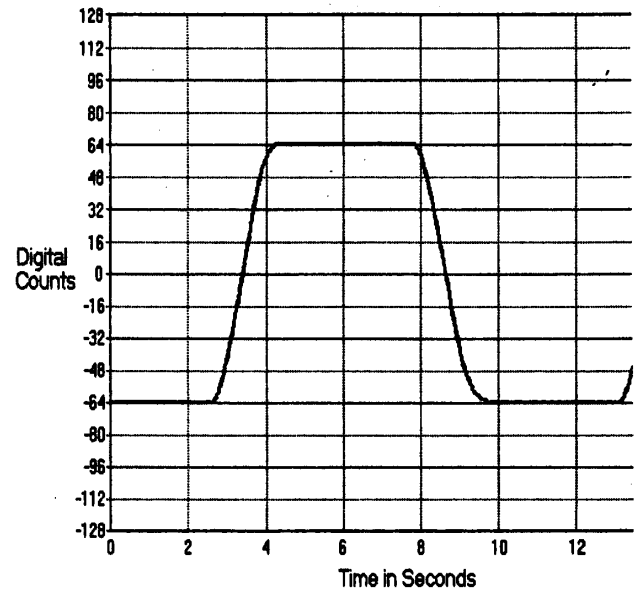


6(a). Actual motor position (±64 digital counts =±5 volts, which is equivalent to ±90 degrees, or a total of 180 degrees

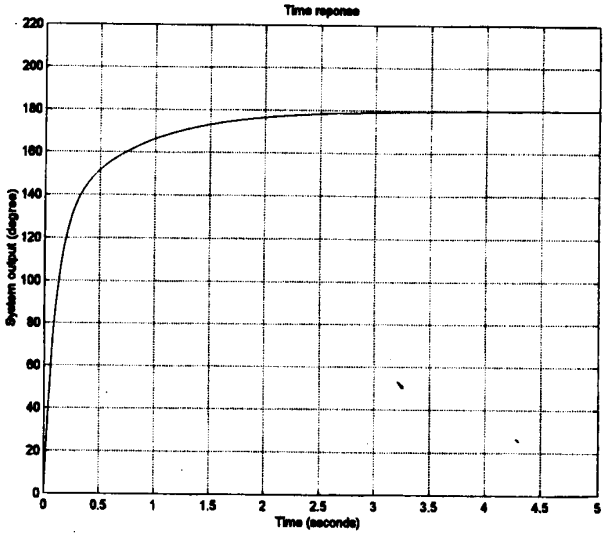


6(b). Simulation using MATLAB (output position is 180 degrees)

Figure 6. System output position without a controller



7(a). Actual motor position (±64 digital counts =±5 volts, which is equivalent to ±90 degrees, or a total of 180 degrees



7(b). Simulation using MATLAB (output position is 180 degrees)

Figure 7. System output position with a PID controller