NASA Technical Memorandum 108535

Inverting the Pendulum Using Fuzzy Control

(Center Director's Discretionary Fund Final Report—Project 93–02)

R.R. Kissel and W.T. Sutherland
Marshall Space Flight Center • MSFC, Alabama

National Aeronautics and Space Administration
Marshall Space Flight Center • MSFC, Alabama 35812

May 1997

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS AND ACRONYMS

A	ampere
A/D	analog to digital
CDDF	Center Director's Discretionary Fund
dtheta	variable for pendulum rate
D/A	digital to analog
EEPROM	electrically erasable programmable read-only memory
k	thousand
max	maximum
min	minimum
Mpos	motor position
Mrate	motor (revolution) rate
RAM	random access memory
R/D	resolver digital
Theta	pendulum angle
thetadot	pendulum rate
V	volt

TECHNICAL MEMORANDUM

INVERTING THE PENDULUM USING FUZZY CONTROL

BACKGROUND

This work was done as the result of a desire to demonstrate fuzzy motor control with an emphasis on doing something that is difficult to do by conventional control methods. The demonstration was conducted in hardware rather than as a software simulation. The Center Director's Discretionary Fund (CDDF) was the avenue chosen for funding the task. A CDDF typically runs for 2 years; the funding commitment is $20,000 to $40,000. This work eventually stretched to 4 years. The project was not as extensive as originally intended, primarily because CDDF work is given a low priority. A highly nonlinear system was chosen as the best candidate for the demonstration because fuzzy control is inherently nonlinear and conventional controllers generally only work well with linear systems.

The most common nonlinear system for demonstrating a control strategy is perhaps the inverted pendulum. The version chosen for this demonstration is on a circular base rather than the usual track. This way, it cannot run off the track. Also, while the act of maintaining the pendulum in an inverted position is somewhat nonlinear, the act of bringing the pendulum to a vertical position from the hanging position is extremely nonlinear. The fuzzy controller does this, as well as maintaining the pendulum inverted, and no one could define another controller that could invert the pendulum at all.

After getting the controller to work with the single pendulum, the next task was to try to make the controller work with a double inverted pendulum. This has worked in simulation to a limited extent, but not well enough to try in hardware. After that, a high-frame-rate camera was purchased in order to attempt real-time control of the pendulum using only visual inputs. That is, image processing is done in real time to determine the pendulum angle, pendulum rate, and base rate (and base position, if desired). It may be possible to use conventional cameras with the high-frame-rate camera for stereo vision.

Although time for the CDDF work has been exhausted, some work may continue after the report is published. Two other projects were ready to be used for fuzzy control if time had been available. One involved a control-structures interaction suitcase demonstration that uses a piezoelectric damper to reduce structural vibration. The other was an electromechanical actuator with a conventional controller.

FUZZY CONTROL

Fuzzy controllers typically have more sensors than conventional controllers, while precision is generally not as high as with other controllers. However, due to the ease with which the control can be changed and its total nonlinear capability, fuzzy control often works better and can be built more quickly than anything else. Since there are so many parameters, there may be a need to use something like genetic algorithms to optimize everything, but, even without optimization, reasonable results can be obtained. Figure 1 is a block diagram of the inverted pendulum system. The system includes three inputs into the pendulum and one output into the motor.

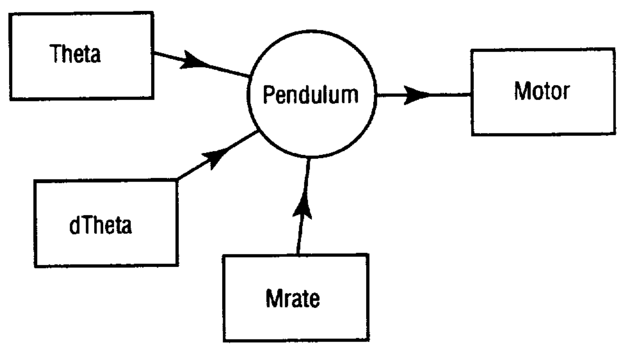


FIGURE 1.—Inverted pendulum control system block diagram.

The sequence inside the controller begins with sending the inputs through an analog-to-digital (A/D) converter. These digitized inputs are each placed (fuzzified) into one of typically three, five, or seven ranges; rules are written with these ranges as arguments rather than with the individual value as arguments. Then the rules are evaluated and fuzzy values are produced for each one using the max-min method, max-dot (used here, also called max-product), or other inference method. In rule evaluation, "or" implies selecting the maximum of the two values, while "and" selects the minimum of the two values. In either case, the value itself is at the intersection of the membership function and the actual input number. With max-dot, the output membership function is scaled to the result of the minimum value or maximum value, whereas with the max-min it is clipped to that value. Both methods give similar results. These rule outputs are then combined (defuzzified) by the centroid (used here), height, or other method to produce one numerical (crisp) value. The combination of all the rule outputs generally produces an irregularly shaped result. The centroid of this irregular shape (easily computed for trapezoidal membership functions) is the crisp output value. In u4e height method, just the height of each rule output is used and the weighted output of all contributions is used. However, in the height method, unsymmetrical output membership functions cause errors. Finally, a digital-to-analog (D/A) converter is used to produce the voltage needed to drive the actuator.

PENDULUM HARDWARE

The pendulum hardware was built on a large inverted trash can. A line drawing of the hardware is shown in figure 2, and a photograph is shown in figure 3. The motor in the center rotates the bar with the actual pendulum attached to one end and the counterweight at the other. A tachometer is geared to the motor for one of the three inputs. A resolver is geared to the pendulum to measure the position of the pendulum as the second input, and successive resolver values are compared to produce pendulum rate as the third input. The command goes to the motor for the systems' only output. A motor position measurement would be desirable, but the only method for obtaining that, as built, is to integrate the tachometer output. This is only an 8-bit value and is not accurate enough to be useful. A future redesign could provide improvement on this part.

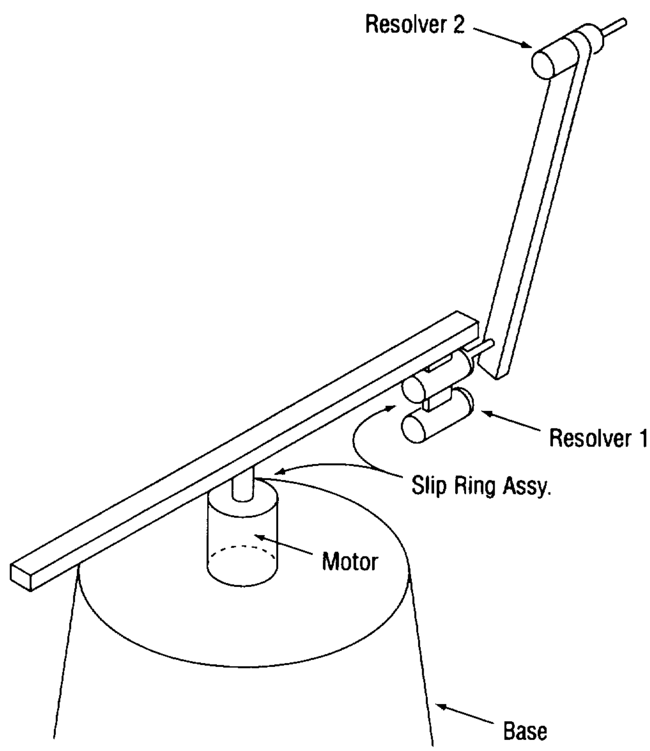


FIGURE 2.—Line drawing of the pendulum hardware.

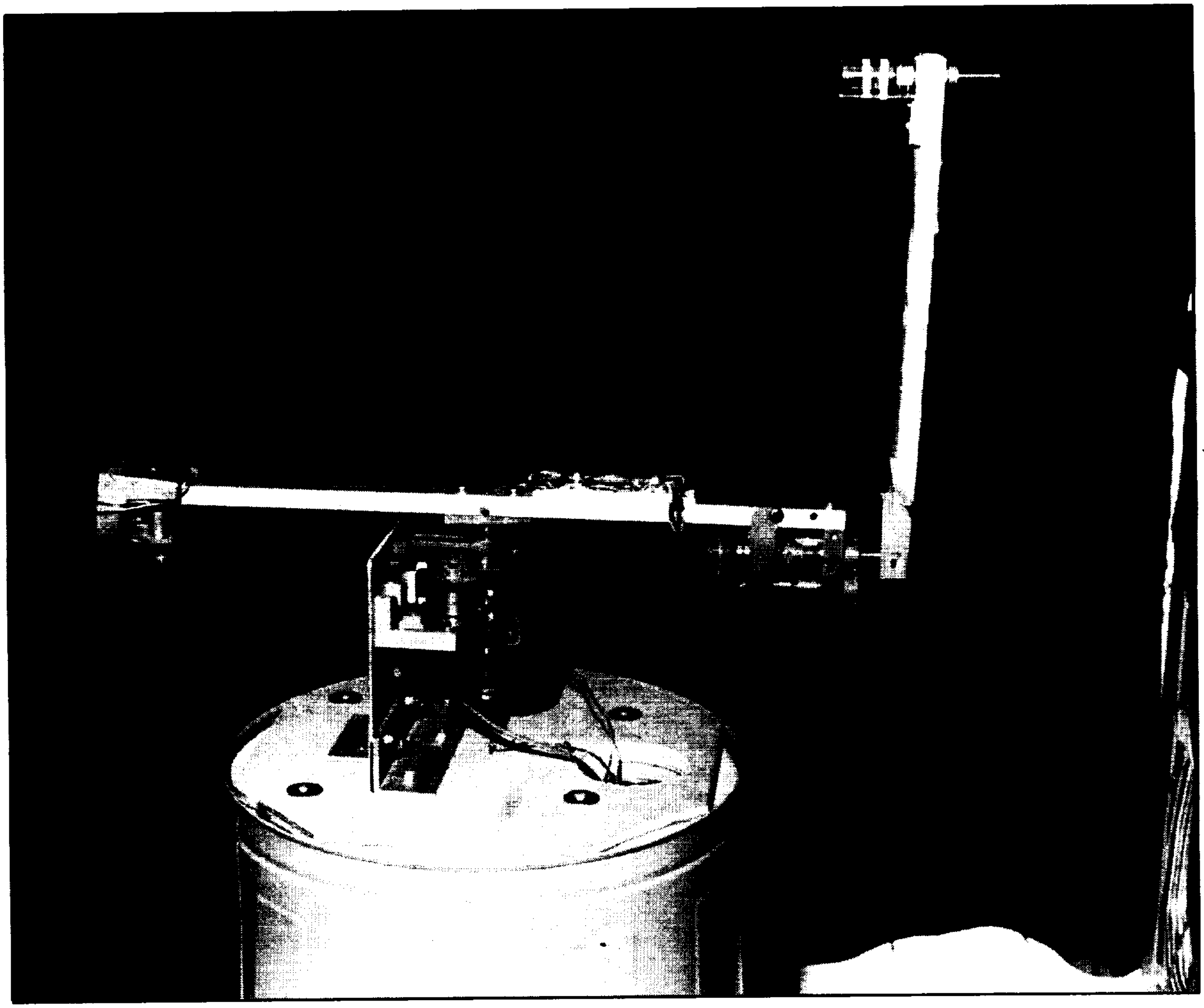


FIGURE 3.—Photograph of the single pendulum hardware.

The electronic hardware uses a 12-bit resolver/digital (R/D) converter for the pendulum input, an 8-bit converter (in the Motorola 6811 microprocessor) for the motor tachometer, and a 12-bit (uses 8 of the 12) D/A converter for the motor drive. The Motorola processor is a model 68HC11E8, which has 2 kilobytes of electrically erasable programmable read-only memory (EEPROM), 256 bytes RAM, and 4 A/D converters (8-bit). Figure 4 shows a photograph of this hardware, where theR/D and A/D converters are on the left side and the 6811 microprocessor is on the right side. There are three power supplies: 20 V at 3A for the motor, –20 V at 3A for the motor, and 5 V for the signals and central processing unit. Figure 5 shows the two large power supplies above the bench and the 5-V supply on the bench to the right. The motor is rated for 0.5 foot-pound at 5 A. The motor and tachometer are most visible in figure 3.

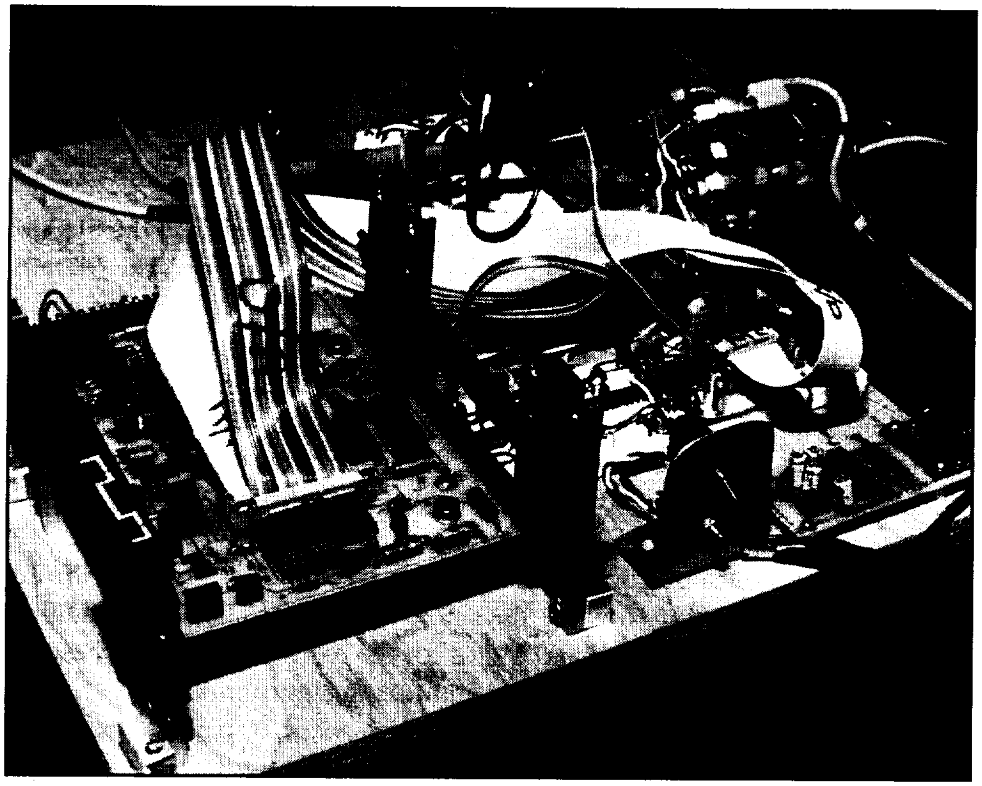


FIGURE 4.—Controller electronics with 6811 microprocessor on right.

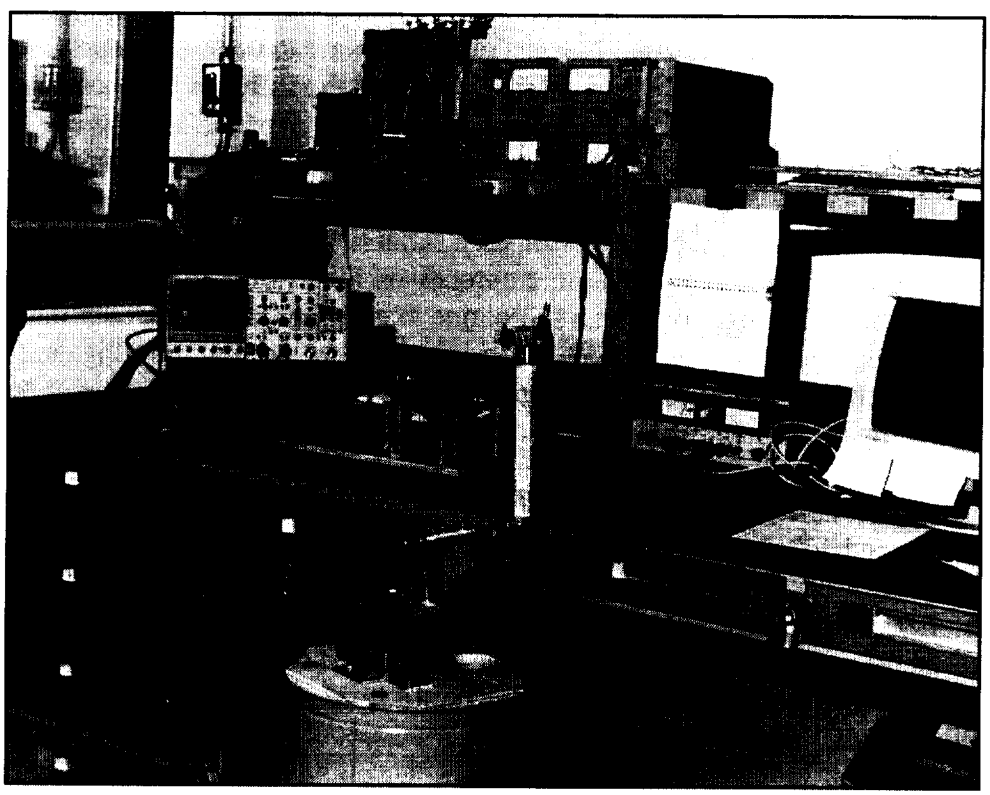


FIGURE 5.—Total system hardware.

SOFTWARE

Simulation was done first. Matlab software was programmed to derive the dynamic equations using the Lagrange method. This was done using the symbolic math toolbox. The Matlab code is shown in appendix A. The kinetic energy equation is k, the potential equation is p, the dissipation equation is d, and the forcing function is q1. Xd2 and xd4 are the two resulting differential equations describing the system dynamics. These are shown as xdot(2) and xdot(4) in the function xdot. Then Matlab was used to simulate these equations with numbers approximating actual hardware values. A small amount of tuning was done in order to set gains, check scale factors, etc. The controller was then built and simulated with the Togai fuzzy control package with the dynamic equations taken from Matlab.

The membership functions developed for the system were worked out with the Togai software and are shown in appendix B. There are four for the single pendulum: Theta for pendulum angle, dTheta for pendulum rate, Mrate for the motor rate, and Motor for the motor command. These are usually triangular, although trapezoidal is also common. An optimizing algorithm could determine that some other shape would work better, that different widths would be better, or perhaps that a different number of functions would be better. "Negative small," for example, refers to any data point within that particular membership function.

The real-time fuzzy controller software that actually ran in the 6811 microprocessor was generated by Togai in 6811 assembler language. The rest of the software to do sensor inputs, use the controller inputs and output, and do scaling, etc., was written by hand; all the software was combined and assembled by the Avocet assembler on a standard personal computer. Appendix C has the Togai-generated code for the membership functions, rules, and variables. It also has its simulation initialization code and then the simulation code itself. This is followed by the Avocet linker code and the resulting mapping. This was downloaded to the 6811 microprocessor via a serial link and then run. The object code for this is shown in ASCII format as sent to the 6811 microprocessor. Finally, the handwritten assembly code is shown which simply calls the Togai code as a subroutine to give the fuzzy controller output that results from the present inputs. This handwritten code also handles timers, scaling, limiting, and input/output. Procom was used to send the ASCII to the 6811 microprocessor. The ASCII routines at the end of the handwritten code were used to send data back to the computer for troubleshooting.

The control (rule) matrix in the fuzzy controller is full, i.e., there are no empty combinations. This usually means (and does here) that the rules have not been optimized (number of rules minimized). The complete control matrix is shown in figure 6. The theta-thetadot portion of the matrix is the standard pendulum controller used to keep the pendulum upright and stable once it is upright. The motor rate-theta portion keeps the motor rate to a minimum. The two shaded columns are all that is needed to invert the pendulum. The control matrix can be read, for example, by saying that if theta is negative small and thetadot is positive small, then the control is zero.

CONTROL MATRIX

					Th			
		NB	NU	NS	Z	PS	PU	PB
	PB	NB	Z	NB	NB	NB	PB	Z
	PS	NS	Z	Z	NS	NB	PS	Z
Thdot	Z	PS	Z	PS	Z	NS	Z	NS
	NS	Z	NS	PB	PS	Z	Z	PS
	NB	Z	NB	PB	PB	PB	Z	PB
					Th			
					Z			
				PB	PS			
			Mdot	Z	Z			
				NB	NS			

Notes: Shaded area controls the inversion process.
Th–Theta
Thdot–Thetadot
Mdot–Motor Rate

FIGURE 6.—Single pendulum complete control matrix.

An output results from evaluating all the rules and combining their contributions. If, for example, the theta value is –20 and the thetadot value is +6, both determined from the actual sensor readings scaled from –128 to +127, moving to the membership function intersections will give a value of membership for each and will show which rules apply. If a rule has positive large and no sensor value is positive large, then that rule does not apply. However, for –20 and +6, some rule(s) will apply. If none do, as could occur after optimization was done, no output change occurs. One way of using these rule results to determine the actual value for the output is to combine them using a center-of-gravity method (explained earlier). This output is what gets sent to the D/A converter for the motor. Fuzzy control is a good method for combining sometimes conflicting rules to produce a useful output and is an inherent method of doing smooth, nonlinear gain switching.

RESULTS

A VHS tape of the controller in action is available. Starting the pendulum in the already inverted position results in the pendulum's remaining inverted, but with some wavering and some slow rotation of the pendulum around the can. The pendulum must move (start to fall) before the controller has an error to correct. This movement was left somewhat exaggerated to illustrate visually how the control operates. The motor must also move the support bar under the pendulum to maintain control, and this causes the bar to wander around the can since the can is not very sturdy and the bar falls different amounts before the controller gets under it. If motor position (angle) was available, the controller would be able to limit the wander to a relatively narrow range about a fixed angle.

Starting the pendulum in the down position results in somewhat violent swinging as the controller begins to upright the pendulum. This control is extremely nonlinear and actually unstable until a nearly inverted position is reached. The pendulum can be jarred such that it falls, but it always rights itself. It can be prodded below the point of losing control, and it is quite robust. When control is lost, the pendulum rights itself again.

Figure 7 shows the single pendulum starting in its rest position. The darker line is Mrate while Theta is the one that settles in at about 3.3 sec. The third line is Mpos. The violent swinging is evident first, followed by stabilizing upright to the limit cycles of position and rate. Figure 8 shows a transient response of the double pendulum stabilizing to the inverted position. The lighter line is the lower bar position while the darker line is the upper bar position. The stable angle range is quite small, indicating either that the system is hard to control or that the controller is not optimized for this particular system.

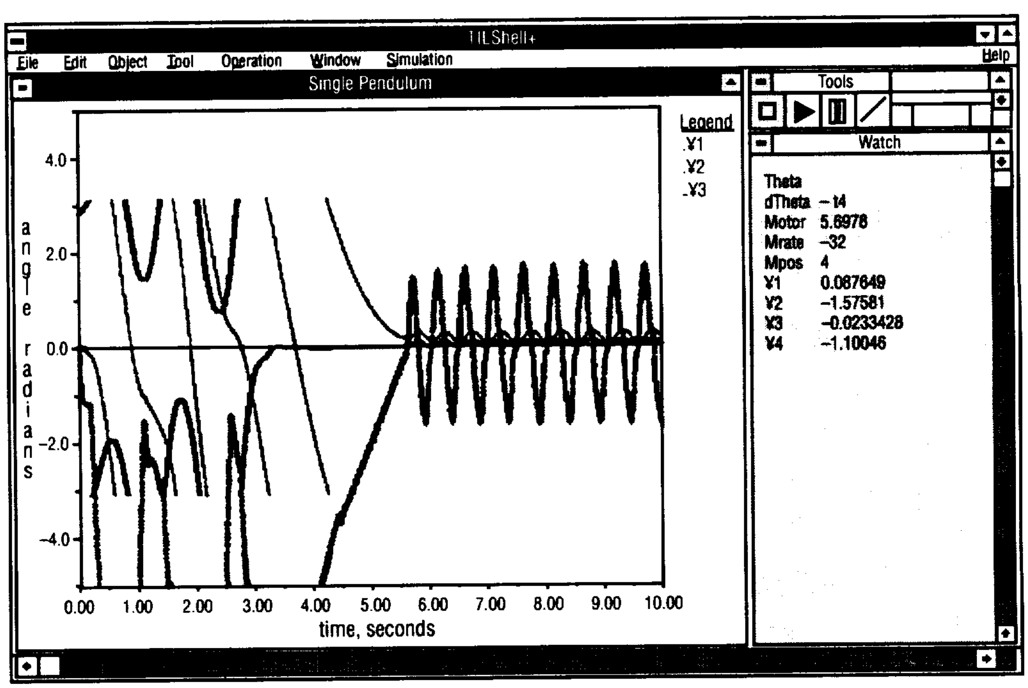


FIGURE 7.—Single pendulum transient response.

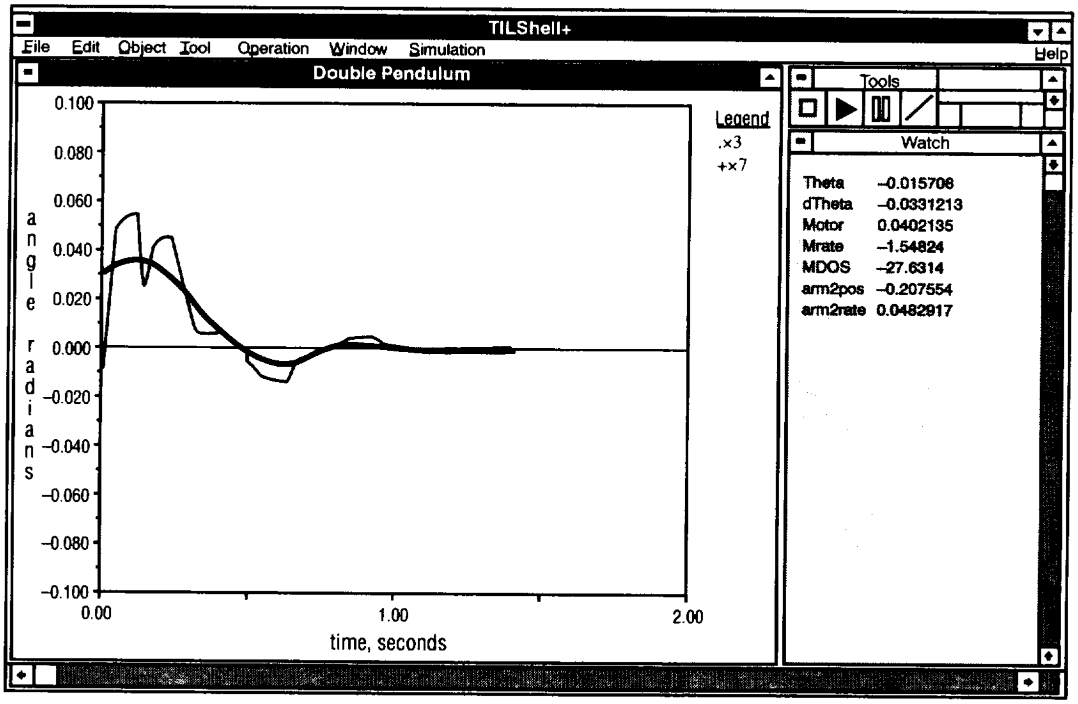


FIGURE 8.—Double pendulum transient response.

CONCLUSIONS

Although one objective of this work was to compare this controller with more conventional controllers, no other controller that would invert the pendulum could be defined. There was speculation that a piecewise linear controller with much switching could be made to work, but it was never produced. In fact, the fuzzy controller is the ultimate switching controller since it transitions smoothly between the various rules (pieces) to accomplish the control. Although keeping the pendulum inverted is not difficult for even a set-point controller, inverting it is where fuzzy control excels.

FUTURE WORK

Balancing the double pendulum has been done here in simulation but not in hardware. The simulation was not optimized for this, and the angle ranges for stability were only a few degrees. It is not certain if the present hardware would even allow the double pendulum to be inverted. The hardware may not be sturdy enough to make it work and, in fact, it may not even be theoretically possible to do. More work could be done here.

There is a suitcase demonstration of a control-structure interaction device that uses piezoelectric material to provide damping to a thin, flexible beam (about 10 inches long). The fuzzy control could be tested to determine how it works just by controlling the motor. Perhaps the piezo material could also be tied to an output for additional control. An earlier simulation of fuzzy control as applied to this device was made by a summer faculty employee and appeared to work well, but it was never tested on the hardware. A 6811 microprocessor had been added to the hardware and it was ready to be used.

An electromechanical actuator was to be controlled with a fuzzy controller and compared with the conventional controllers presently being used. There did not seem to be much improvement possible here since the controller was full on most of the time. Only when near the commanded point would some improvement be possible. Perhaps some worthwhile improvement could be made here.

A high-frame-rate camera (and frame-grabber board) was purchased to derive all the pendulum control inputs from image processing, thereby using no sensors (or wiring) on the pendulum itself. This particular implementation of the pendulum, however, with its rotation, presents various perspectives to any fixed camera position and this has to be resolved. Two standard video cameras were also purchased, and one or both of these could possibly substitute or be used with the high-frame-rate camera. It may be possible to use two cameras to solve the rotational perspective problem. But time did not permit any of this to be done.

This project received much attention and demonstrated that fuzzy control is a useful tool for the engineer when appropriate.

APPENDIX A
MATLAB CODE 1

PENDULUM.M.

```
clear;clc

k='1/2*mpl*((rbar*Dg-len/2*cos(th)*Dth) ^2+(len/2*sin(th)*Dth) ^2)+1/2*ibar*Dg^2+1/2*ip1*Dth^2';
pretty(k)
p='mp1*gr*len/2*cos(th)';    %gr=gravity
pretty(p)
d='1/2*dp1*Dth^2+1/2*dbar*Dg^2';
pretty (d)
q1='tm/rbar';
q2='0';
pretty(q1)
pretty(q2)

g='g(t)';
Dg=diff(g,'t');
th='th(t)';
Dth=diff(th,'t');

k1=diff(k,'Dth');
k1=subs(k1,th,'th');
k1=subs(k1,Dth,'Dth');
k1=subs(k1,Dg,'Dg');

k2=diff(k1,'t');
k2=subs(k2,'D2g','diff(diff(g(t),t),t)');
k2=subs(k2,'D2th','diff(diff(th(t),t),t)');
k2=subs(k2,'Dg','diff(g(t),t)');
k2=subs(k2,'Dth','diff(th(t),t)');
k2=subs(k2,'th','th(t');

k3=diff(k,'th');

k4=diff(p,'th');

k5=diff(d,'Dth');
```

```
k6=diff(k,'Dg');
k6=subs(k6,th,'th');
k6=subs(k6,Dth,'Dth');
k6=subs(k6,Dg,'Dg');

k7=diff(k6,'t');
k7=subs(k7,'D2g','diff(diff(g(t),t),t)');
k7=subs(k7,'D2th','diff(diff(th(t),t),t)');
k7=subs(k7,'Dg','diff(g(t),t)');
k7=subs(k7,'Dth','diff(th(t),t)');
k7=subs(k7,'th','th(t)');

k8=diff(k,'g');

k9=diff(p,'g');

k10=diff(d,'Dg');

R=symop(k2,'-',q2,'-',k3,'+',k4,'+',k5);
simplify(R);
%pretty(R)

S=symop(k7,'-',q1,'-',k8,'+',k9,'+',k10);
simplify(S);
%pretty(S)
[g1,g2]=solve(R,S,'D2th,D2g');
%pretty(g1)
%pretty(g2)

R=subs(R,.4,'mp1');
S=subs(S,.4,'mp1');
R=subs(R,.005,'ip1');
S=subs(S,.005,'ip1');
R=subs(R,9.8,'gr');
S=subs(S,9.8,'gr');
R=subs(R,.08,'ibar');
S=subs(S,.08,'ibar');
R=subs(R,.01,'dbar');
S=subs(S,.01,'dbar');
R=subs(R,.01,'dp1');
S=subs(S,.01,'dp1');
R=subs(R,.2,'len');
S=subs(S,.2,'len');
R=subs(R,.2,'rbar');
S=subs(S,.2,'rbar');
```

```
R=subs(R,0,'tm');
S=subs(S,0,'tm');

[g1,g2]=solve(R,S,'D2g,D2th');

g1=subs(g1,'b','g');
g1=subs(g1,'c','Dg');
g1=subs(g1,'d','th');
g=subs(g1,'e','Dth');
g2=subs(g2,'b','g');
g2=subs(g2,'c','Dg');
g2=subs(g2,'d','th');
h=subs(g2,'e','Dth');

g=simple(g);
h=simple(h);
g=collect(g)
h=collect(h)
pretty(g)
pretty(h)

g3='Dc=g';
h1='De=h';
g3=subs(g3,g,'g')
h1=subs(h1,h,'h')

%[b,c,d,e]=dsolve('Db=c',g3,'Dd=e',h1,'b(0)=.01','c(0)=0','d(0)=0','e(0)=0','t')

xd2=subs(g,'x(1)','b');
xd2=subs(xd2,'x(2)','c');
xd2=subs(xd2,'x(3)','d');

xd4=subs(h,'x(1)','b');
xd4=subs(xd4,'x(2)','c');
xd4=subs(xd4,'x(3)','d');

xd2=subs(xd2,'x(4)','e')
xd4=subs(xd4,'x(4)','e')

t0=0;tf=3;
x0=[0 0 .01 0]'; %initial conditions
[t,x]=ode23('penddiff',t0,tf,x0);
%plot(t,x(:,3))
plot(t,x)
```

PENDDIFF.M

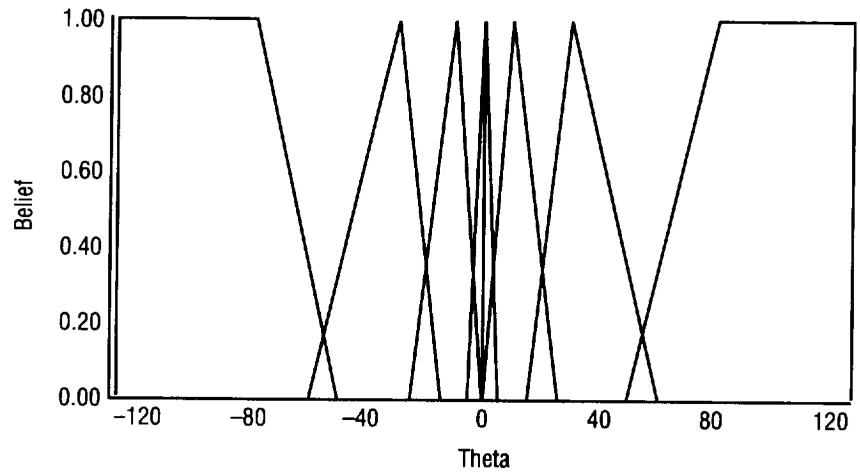
```
function xdot=penddiff(t,x)
xdot=zeros(4,1);

xdot(1)=x(2);
xdot(3)=x(4);

xdot(2)=9/4*sin(x(3))/(2*cos(x(3))^2-27)*x(4)^2+5/2*cos(x(3))/(2*cos(x(3))^2-27)*x(4)+1/16*
(-1568*cos(x(3))*sin(x(3))+45*x(2))/(2*cos(x(3))^2-27);

xdot(4)=2*cos(x(3))*sin(x(3))/(2*cos(x(3))^2-27)*x(4)^2+30/(2*cos(x(3))^2-27)*x(4)+1/
2*(5*cos(x(3))*x(2)-2352*sin(x(3)))/(2*cos(x(3))^2-27);
```

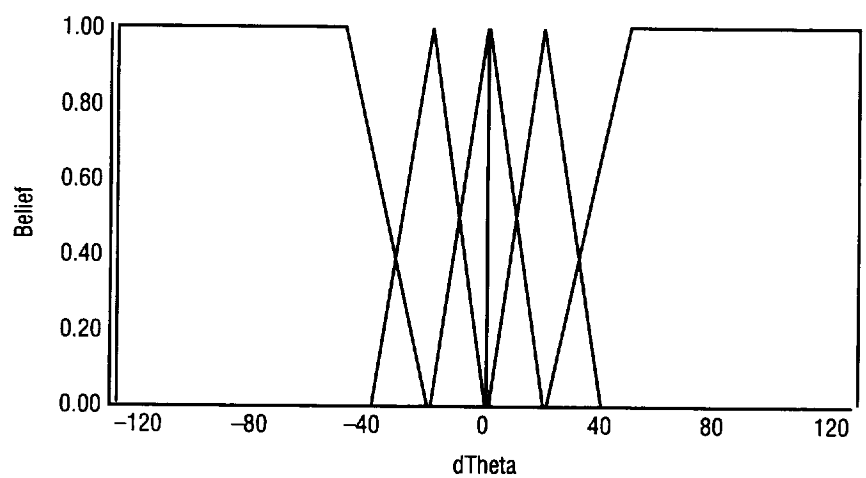
APPENDIX B
MEMBERSHIP FUNCTIONS FOR SINGLE PENDULUM



Variable Theta membership functions

The membership functions for this variable are:

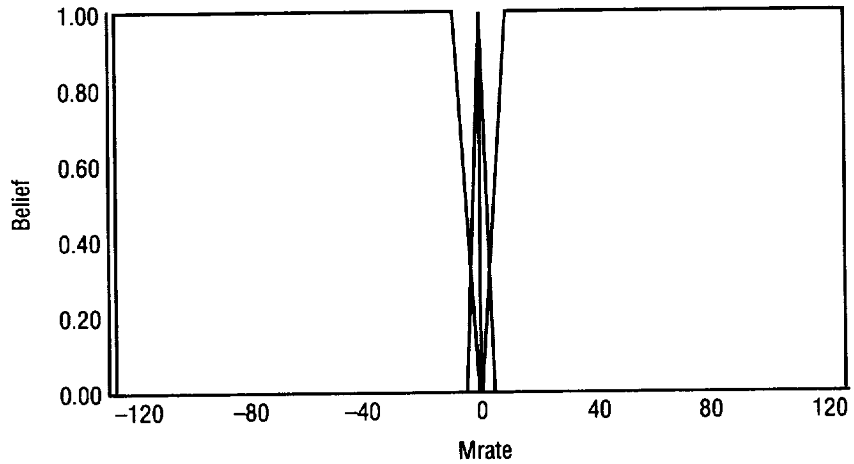
Membership function NB: Points list: –128, 1 –80,1 –50,0
Membership function NS: Points list: –25.18867925,0 –10, 1 0 ,0
Membership function Z: Points list: –5,0 0,1 5, 0
Membership function PS: Points list: 0,0 10, 1 25. 18867925,0
Membership function PB: Points list: 50,0 80, 1 80, 1 127, 1
Membership function NU: Points list: –60,0 –30, 1–14.38679245,0
Membership function PU: Points list: 15.58962264,0 30,1 60,0



Variable dTheta membership functions

The membership functions for this variable are:

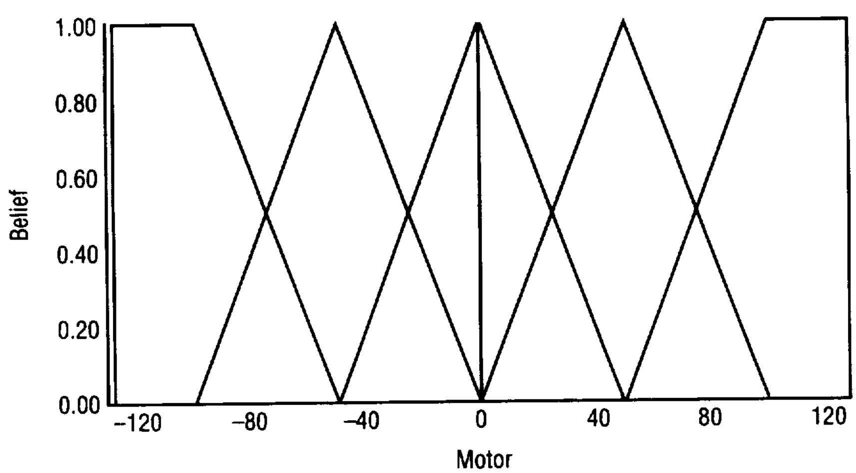
Membership function NB: Points list: –128, 1 –50,1 –20,0
Membership function NS: Points list: –40,0 –20,1 0,0
Membership function Z: Points list: –20,0 –0.5,1 20,0
Membership function PS: Points list: 0,0 20,1 40,0
Membership function PB: Points list: 20,0 50,1 127,1



Variable Mrate membership functions

The membership functions for this variable are:

Membership function NB: Points list: –128, 1 –10,1 0,0
Membership function Z: Points List: –4.716850829,0 0, 1
5.069060773,0.002673796791
Membership function PB: Points list: 0.0 10,1 127,1



Variable Motor membership functions

The membership functions for this variable are:

Membership function NB: Points list: –128, 1 –100,1 –50.0
Membership function NS: Points list: –100,0 –50,1 –0.5,0
Membership function Z: Points list: -50,0 –0.5, 1 50.0
Membership function PS: Points list: –0.5,0 50,1 100,0
Membership function PB: Points list: 50,0 100,1 127,1

APPENDIX C
TOGAI AND HANDWRITTEN CODE FOR SINGLE PENDULUM

```
*   .EXPORT _Mpos
*   .EXPORT _Mrate
*   .EXPORT _Theta
*   .EXPORT _dTheta
*   .EXPORT _Motor
*   .EXPORT _PENDULUM1
*   .EXPORT _PENDULUM1_init
; Input VARs
    DEFSEG  ramvars,START=$a
    SEG     ramvars
    ORG     $a
PENDULUM1_var:
_Mpos:
    RMB     1
_Mrate:
    RMB     1
_Theta:
    RMB     1
_dTheta:
    RMB     1
; Output VARs
_Motor:
    RMB     1
; Hidden VARs
PENDULUM1_temp:
; Input MEMBER temps
; MEMBER Z(Theta)
    RMB     1
; MEMBER Z(dTheta)
    RMB     1
; MEMBER NS(Theta)
    RMB     1
; MEMBER PS(Theta)
    RMB     1
; MEMBER PB(dTheta)
    RMB     1
; MEMBER NU(Theta)
    RMB     1
```

```
; MEMBER PS(dTheta)
    RMB       1
; MEMBER NS(dTheta)
    RMB       1
; MEMBER NB(dTheta)
    RMB       1
; MEMBER PU(Theta)
    RMB       1
; MEMBER NB(Theta)
    RMB       1
; MEMBER PB(Theta)
    RMB       1
; Output VAR temps
    RMB       7
    DEFSEG    romvars,START=$f3e5
    SEG       romvars
    ORG       $f3e5
_PENDULUM1:
    FDB       PENDULUM1_var
    FDB       PENDULUM1_temp
    FDB       PENDULUM1_inmbf
    FDB       PENDULUM1_outmbf
    FDB       PENDULUM1_code

*   .PAGE
; Input MEMBER table

PENDULUM1_inmbf:
    FDB       plZTheta
    FDB       plZdTheta
    FDB       plPMpos
    FDB       plNSTheta
    FDB       plZMpos
    FDB       plPBMrate
    FDB       plZMrate
    FDB       plNMpos
    FDB       plPSTheta
    FDB       plNBMrate
    FDB       plPBdTheta
    FDB       plNUTheta
    FDB       plPSdTheta
    FDB       plNSdTheta
    FDB       plNBdTheta
    FDB       plPUTheta
    FDB       plNBTheta
    FDB       plPBTheta
```

```
; MEMBER Z(Theta)
plZTheta:
    FCB     -5,0
    FCB     0,255
    FCB     5,0
    FCB     127,0
; MEMBER Z(dTheta)
plZdTheta:
    FCB     -20,0
    FCB     -1,255
    FCB     20,0
    FCB     127,0
; MEMBER P(Mpos)
plPMpos:
    FCB     -1,0
    FCB     20,255
    FCB     127,255
; MEMBER NS(Theta)
plNSTheta:
    FCB     -26,0
    FCB     -10,255
    FCB     0,0
    FCB     127,0
; MEMBER Z(Mpos)
plZMpos:
    FCB     -10,0
    FCB     -1,255
    FCB     10,0
    FCB     127,0
; MEMBER PB(Mrate)
plPBMrate:
    FCB     0,0
    FCB     10,255
    FCB     127,255
; MEMBER Z(Mrate)
plZMrate:
    FCB     -5,0
    FCB     0,255
    FCB     5,0
    FCB     127,0
; MEMBER N(Mpos)
plNMpos:
    FCB     -20,255
    FCB     -1,0
    FCB     127,0
```

```
; MEMBER PS(Theta)
plPSTheta:
    FCB     0,0
    FCB     10,255
    FCB     25,0
    FCB     127,0
; MEMBER NB(Mrate)
plNBMrate:
    FCB     -10,255
    FCB     0,0
    FCB     127,0
; MEMBER PB(dTheta)
plPBdTheta:
    FCB     20,0
    FCB     50,255
    FCB     127,255
; MEMBER NU(Theta)
plNUTheta:
    FCB     -60,0
    FCB     -30,255
    FCB     -15,0
    FCB     127,0
; MEMBER PS(dTheta)
plPSdTheta:
    FCB     0,0
    FCB     20,255
    FCB     40,0
    FCB     127,0
; MEMBER NS(dTheta)
plNSdTheta:
    FCB     -40,0
    FCB     -20,255
    FCB     0,0
    FCB     127,0
; MEMBER NB(dTheta)
plNBdTheta:
    FCB     -50,255
    FCB     -20,0
    FCB     127,0
; MEMBER PU(Theta)
plPUTheta:
    FCB     15,0
    FCB     30,255
    FCB     60,0
    FCB     127,0
```

```
; MEMBER NB(Theta)
plNBTheta:
     FCB        -80,255
     FCB        -50,0
     FCB        127,0
; MEMBER PB(Theta)
plPBTheta:
     FCB        50,0
     FCB        80,255
     FCB        80,255
     FCB        127,255
*    .PAGE
; Output MEMBER table

PENDULUM1_outmbf:
; MEMBER Z(Motor)
; M 6391.67 A 50 H 1
     FCB        $23,$06
     FCB        $C4,$10,$03
; MEMBER PB(Motor)
; M 11803.8 A 52 H 1
     FCB        $62,$06
     FCB        $46,$A9,$05
; MEMBER PS(Motor)
; M 8936.13 A 50.25 H 1
     FCB        $2B,$06
     FCB        $2D,$49,$04
; MEMBER NB(Motor)
; M 1508.67 A 53 H 1
     FCB        $81,$06
     FCB        $3B,$B9,$00
; MEMBER NS(Motor)
; M 3872.21 A 49.75 H 1
     FCB        $1B,$06
     FCB        $6D,$DB,$01
*    .PAGE
; Knowledge base code
; PROJECT PENDULUM1
PENDULUM1_code:
; MEMBER Z(Theta)
     FCB        $01,$00,$02
     FCB        $10,$00
; MEMBER Z(dTheta)
     FCB        $01,$02,$03
     FCB        $10,$01
```

```
; MEMBER NS(Theta)
    FCB     $01,$06,$02
    FCB     $10,$02
; MEMBER PS(Theta)
    FCB     $01,$10,$02
    FCB     $10,$03
; MEMBER PB(dTheta)
    FCB     $01,$14,$03
    FCB     $10,$04
; MEMBER NU(Theta)
    FCB     $01,$16,$02
    FCB     $10,$05
; MEMBER PS(dTheta)
    FCB     $01,$18,$03
    FCB     $10,$06
; MEMBER NS(dTheta)
    FCB     $01,$1A,$03
    FCB     $10,$07
; MEMBER NB(dTheta)
    FCB     $01,$1C,$03
    FCB     $10,$08
; MEMBER PU(Theta)
    FCB     $01,$1E,$02
    FCB     $10,$09
; MEMBER NB(Theta)
    FCB     $01,$20,$02
    FCB     $10,$0A
; MEMBER PB(Theta)
    FCB     $01,$22,$02
    FCB     $10,$0B
; FUZZY Pendulum_rules
; VAR Motor
    FCB     $12,$0C
; RULE Rule0001
    FCB     $0E,$00
    FCB     $0E,$01
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0053
    FCB     $01,$04,$00
    FCB     $0E,$02
    FCB     $05
    FCB     $0B,$05,$0C
; RULE Rule0052
    FCB     $01,$08,$00
```

```
    FCB     $0E,$00
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0046
    FCB     $01,$0A,$01
    FCB     $0E,$00
    FCB     $05
    FCB     $0B,$0A,$0C
; RULE Rule0055
    FCB     $01,$0C,$01
    FCB     $0E,$00
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0057
    FCB     $01,$0E,$00
    FCB     $0E,$03
    FCB     $05
    FCB     $0B,$0F,$0C
; RULE Rule0047
    FCB     $01,$12,$01
    FCB     $0E,$00
    FCB     $05
    FCB     $0B,$14,$0C
; RULE Rule0033
    FCB     $0E,$04
    FCB     $0E,$05
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0034
    FCB     $0E,$06
    FCB     $0E,$05
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0035
    FCB     $0E,$01
    FCB     $0E,$05
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0036
    FCB     $0E,$07
    FCB     $0E,$05
    FCB     $05
    FCB     $0B,$14,$0C
; RULE Rule0037
    FCB     $0E,$08
```

```
    FCB      $0E,$05
    FCB      $05
    FCB      $0B,$0F,$0C
; RULE Rule0038
    FCB      $0E,$04
    FCB      $0E,$09
    FCB      $05
    FCB      $0B,$05,$0C
; RULE Rule0039
    FCB      $0E,$06
    FCB      $0E,$09
    FCB      $05
    FCB      $0B,$0A,$0C
; RULE Rule0040
    FCB      $0E,$01
    FCB      $0E,$09
    FCB      $05
    FCB      $0B,$00,$0C
; RULE Rule0041
    FCB      $0E,$07
    FCB      $0E,$09
    FCB      $05
    FCB      $0B,$00,$0C
; RULE Rule0042
    FCB      $0E,$08
    FCB      $0E,$09
    FCB      $05
    FCB      $0B,$00,$0C
; RULE Rule0010
    FCB      $0E,$00
    FCB      $0E,$08
    FCB      $05
    FCB      $0B,$05,$0C
; RULE Rule0011
    FCB      $0E,$00
    FCB      $0E,$07
    FCB      $05
    FCB      $0B,$0A,$0C
; RULE Rule0012
    FCB      $0E,$03
    FCB      $0E,$08
    FCB      $05
    FCB      $0B,$05,$0C
; RULE Rule0013
    FCB      $0E,$03
```

```
    FCB     $0E,$07
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0002
    FCB     $0E,$02
    FCB     $0E,$07
    FCB     $05
    FCB     $0B,$05,$0C
; RULE Rule0008
    FCB     $0E,$02
    FCB     $0E,$01
    FCB     $05
    FCB     $0B,$0A,$0C
; RULE Rule0003
    FCB     $0E,$03
    FCB     $0E,$06
    FCB     $05
    FCB     $0B,$0F,$0C
; RULE Rule0009
    FCB     $0E,$03
    FCB     $0E,$01
    FCB     $05
    FCB     $0B,$14,$0C
; RULE Rule0006
    FCB     $0E,$02
    FCB     $0E,$08
    FCB     $05
    FCB     $0B,$05,$0C
; RULE Rule0007
    FCB     $0E,$03
    FCB     $0E,$04
    FCB     $05
    FCB     $0B,$0F,$0C
; RULE Rule0014
    FCB     $0E,$02
    FCB     $0E,$06
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0015
    FCB     $0E,$02
    FCB     $0E,$04
    FCB     $05
    FCB     $0B,$0F,$0C
; RULE Rule0016
    FCB     $0E,$00
```

```
    FCB     $0E,$06
    FCB     $05
    FCB     $0B,$14,$0C
; RULE Rule0017
    FCB     $0E,$00
    FCB     $0E,$04
    FCB     $05
    FCB     $0B,$0F,$0C
; RULE Rule0018
    FCB     $0E,$08
    FCB     $0E,$0A
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0022
    FCB     $0E,$08
    FCB     $0E,$0B
    FCB     $05
    FCB     $0B,$05,$0C
; RULE Rule0024
    FCB     $0E,$07
    FCB     $0E,$0A
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0025
    FCB     $0E,$01
    FCB     $0E,$0A
    FCB     $05
    FCB     $0B,$0A,$0C
; RULE Rule0026
    FCB     $0E,$06
    FCB     $0E,$0A
    FCB     $05
    FCB     $0B,$14,$0C
; RULE Rule0027
    FCB     $0E,$04
    FCB     $0E,$0A
    FCB     $05
    FCB     $0B,$0F,$0C
; RULE Rule0029
    FCB     $0E,$04
    FCB     $0E,$0B
    FCB     $05
    FCB     $0B,$00,$0C
; RULE Rule0030
    FCB     $0E,$06
```

```
    FCB       $0E,$0B
    FCB       $05
    FCB       $0B,$00,$0C
; RULE Rule0031
    FCB       $0E,$01
    FCB       $0E,$0B
    FCB       $05
    FCB       $0B,$14,$0C
; RULE Rule0032
    FCB       $0E,$07
    FCB       $0E,$0B
    FCB       $05
    FCB       $0B,$0A,$0C
    FCB       $0A,$0C,$04
    FCB $16, $4, $80
    FCB       $00
_PENDULUM1_init:
    LDX       #_PENDULUM1
    RTS

OPTIONS
END

PROJECT PENDULUM1

 OPTIONS
 DESCRIPTION="This is a single inverted pendulum"
 GRIDSPACE=0.5,0
 GRIDSHOW="ON"
 GRIDSNAP="ON"
 ICONSCALE=1
 NORMALSIZE="ON"
 REDUCETOFIT="OFF"
 SHOWTOOLS="ON"
 VIEWORIGIN=-0.1,-0.1
 VIEWSCALE=1
 END

 VAR Theta

 OPTIONS
 ICONCOLOR=0
 ICONPOS=1.5,2
 GRIDSPACE=0,0
 GRIDSHOW="ON"
```

```
GRIDSNAP="OFF"
END
TYPE signed byte
DEFAULT 0

MEMBER NB

OPTIONS
ICONCOLOR=1
END
POINTS -128,1 -80,1 -50,0
END

MEMBER NS

OPTIONS
ICONCOLOR=2
END
POINTS -25.18867925,0 -10,1 0,0
END

MEMBER Z

OPTIONS
ICONCOLOR=3
END
POINTS -5,0 0,1 5,0
END

MEMBER PS

OPTIONS
ICONCOLOR=4
END
POINTS 0,0 10,1 25.18867925,0
END

MEMBER PB

OPTIONS
ICONCOLOR=5
END
POINTS 50,0 80,1 80,1 127,1
END
```

```
MEMBER NU

OPTIONS
ICONCOLOR=5
OUTPUTSCOPE="PRIVATE"
END
POINTS -60,0 -30,1 -14.38679245,0
END

MEMBER PU

OPTIONS
ICONCOLOR=6
OUTPUTSCOPE="PRIVATE"
END
POINTS 15.58962264,0 30,1 60,0
END
END

VAR dTheta

OPTIONS
ICONCOLOR=0
ICONPOS=1.5,3.5
GRIDSPACE=0,0
GRIDSHOW="ON"
GRIDSNAP="ON"
END
TYPE signed byte

MEMBER NB

OPTIONS
ICONCOLOR=1
END
POINTS -128,1 -50,1 -20,0
END

MEMBER NS

OPTIONS
ICONCOLOR=2
END
POINTS -40,0 -20,1 0,0
END
```

```
MEMBER Z

OPTIONS
ICONCOLOR=3
END
POINTS -20,0 -0.5,1 20,0
END

MEMBER PS

OPTIONS
ICONCOLOR=4
END
POINTS 0,0 20,1 40,0
END

MEMBER PB

OPTIONS
ICONCOLOR=5
END
POINTS 20,0 50,1 127,1
END
END

VAR Motor

OPTIONS
ICONCOLOR=0
ICONPOS=5.5,2.5
GRIDSPACE=0,0
GRIDSHOW="ON"
GRIDSNAP="ON"
END
TYPE signed byte

MEMBER NB

OPTIONS
ICONCOLOR=1
END
POINTS -128,1 -100,1 -50,0
END

MEMBER NS
```

```
OPTIONS
ICONCOLOR=2
END
POINTS -100,0 -50,1 -0.5,0
END

MEMBER Z

OPTIONS
ICONCOLOR=3
END
POINTS -50,0 -0.5,1 50,0
END

MEMBER PS

OPTIONS
ICONCOLOR=4
END
POINTS -0.5,0 50,1 100,0
END

MEMBER PB

OPTIONS
ICONCOLOR=5
END
POINTS 50,0 100,1 127,1
END
END

VAR Mrate

OPTIONS
ICONCOLOR=0
ICONPOS=3,4
GRIDSPACE=0,0
GRIDSHOW="ON"
GRIDSNAP="ON"
END
TYPE signed byte

MEMBER NB

OPTIONS
ICONCOLOR=1
```

```
END
POINTS -128,1 -10,1 0,0
END

MEMBER Z

OPTIONS
ICONCOLOR=3
END
POINTS -4.716850829,0 0,1 5.069060773,0.002673796791
END

MEMBER PB

OPTIONS
ICONCOLOR=5
END
POINTS 0,0 10,1 127,1
END
END

FUZZY Pendulum_rules

OPTIONS
ICONCOLOR=0
ICONPOS=3.5,2.5
OUTPUTSCOPE="PRIVATE"
END

RULE Rule0001

OPTIONS
ENABLE="ON"
END

IF (Theta IS Z) AND (dTheta IS Z) THEN
Motor=Z
END

RULE Rule0050

OPTIONS
ENABLE="ON"
END
```

```
IF (Mrate IS Z) AND (Theta IS Z) THEN
Motor=Z
END

RULE Rule0046

OPTIONS
ENABLE="ON"
END

IF (Mrate IS PB) AND (Theta IS Z) THEN
Motor=PS
END

RULE Rule0047

OPTIONS
ENABLE="ON"
END

IF (Mrate IS NB) AND (Theta IS Z) THEN
Motor=NS
END

RULE Rule0033

OPTIONS
ENABLE="ON"
END

IF (dTheta IS PB) AND (Theta IS NU) THEN
Motor=Z
END

RULE Rule0034

OPTIONS
ENABLE="ON"
END

IF (dTheta IS PS) AND (Theta IS NU) THEN
Motor=Z
END
```

```
RULE Rule0035

OPTIONS
ENABLE="ON"
END

IF (dTheta IS Z) AND (Theta IS NU) THEN
Motor=Z
END

RULE Rule0036

OPTIONS
ENABLE="ON"
END

IF (dTheta IS NS) AND (Theta IS NU) THEN
Motor=NS
END

RULE Rule0037

OPTIONS
ENABLE="ON"
END

IF (dTheta IS NB) AND (Theta IS NU) THEN
Motor=NB
END

RULE Rule0038

OPTIONS
ENABLE="ON"
END

IF (dTheta IS PB) AND (Theta IS PU) THEN
Motor=PB
END

RULE Rule0039

OPTIONS
ENABLE="ON"
END
```

```
IF (dTheta IS PS) AND (Theta IS PU) THEN
Motor=PS
END

RULE Rule0040

OPTIONS
ENABLE="ON"
END

IF (dTheta IS Z) AND (Theta IS PU) THEN
Motor=Z
END

RULE Rule0041

OPTIONS
ENABLE="ON"
END

IF (dTheta IS NS) AND (Theta IS PU) THEN
Motor=Z
END

RULE Rule0042

OPTIONS
ENABLE="ON"
END

IF (dTheta IS NB) AND (Theta IS PU) THEN
Motor=Z
END

RULE Rule0010

OPTIONS
ENABLE="ON"
END

IF (Theta IS Z) AND (dTheta IS NB) THEN
Motor=PB
END
```

```
RULE Rule0011

OPTIONS
ENABLE="ON"
END

IF (Theta IS Z) AND (dTheta IS NS) THEN
Motor=PS
END

RULE Rule0012

OPTIONS
ENABLE="ON"
END

IF (Theta IS PS) AND (dTheta IS NB) THEN
Motor=PB
END

RULE Rule0013

OPTIONS
ENABLE="ON"
END

IF (Theta IS PS) AND (dTheta IS NS) THEN
Motor=Z
END

RULE Rule0002

OPTIONS
ENABLE="ON"
END

IF (Theta IS NS) AND (dTheta IS NS) THEN
Motor=PB
END

RULE Rule0008

OPTIONS
ENABLE="ON"
END
```

```
IF (Theta IS NS) AND (dTheta IS Z) THEN
Motor=PS
END

RULE Rule0003

OPTIONS
ENABLE="ON"
END

IF (Theta IS PS) AND (dTheta IS PS) THEN
Motor=NB
END

RULE Rule0009

OPTIONS
ENABLE="ON"
END

IF (Theta IS PS) AND (dTheta IS Z) THEN
Motor=NS
END

RULE Rule0006

OPTIONS
ENABLE="ON"
END

IF (Theta IS NS) AND (dTheta IS NB) THEN
Motor=PB
END

RULE Rule0007

OPTIONS
ENABLE="ON"
END

IF (Theta IS PS) AND (dTheta IS PB) THEN
Motor=NB
END
```

```
RULE Rule0014

OPTIONS
ENABLE="ON"
END

IF (Theta IS NS) AND (dTheta IS PS) THEN
Motor=Z
END

RULE Rule0015

OPTIONS
ENABLE="ON"
END

IF (Theta IS NS) AND (dTheta IS PB) THEN
Motor=NB
END

RULE Rule0016

OPTIONS
ENABLE="ON"
END

IF (Theta IS Z) AND (dTheta IS PS) THEN
Motor=NS
END

RULE Rule0017

OPTIONS
ENABLE="ON"
END

IF (Theta IS Z) AND (dTheta IS PB) THEN
Motor=NB
END

RULE Rule0018

OPTIONS
ENABLE="ON"
END
```

```
IF (dTheta IS NB) AND (Theta IS NB) THEN
Motor=Z
END

RULE Rule0022

OPTIONS
ENABLE="ON"
END

IF (dTheta IS NB) AND (Theta IS PB) THEN
Motor=PB
END

RULE Rule0024

OPTIONS
ENABLE="ON"
END

IF (dTheta IS NS) AND (Theta IS NB) THEN
Motor=Z
END

RULE Rule0025

OPTIONS
ENABLE="ON"
END

IF (dTheta IS Z) AND (Theta IS NB) THEN
Motor=PS
END

RULE Rule0026

OPTIONS
ENABLE="ON"
END

IF (dTheta IS PS) AND (Theta IS NB) THEN
Motor=NS
END
```

```
RULE Rule0027

OPTIONS
ENABLE="ON"
END

IF (dTheta IS PB) AND (Theta IS NB) THEN
Motor=NB
END

RULE Rule0029

OPTIONS
ENABLE="ON"
END

IF (dTheta IS PB) AND (Theta IS PB) THEN
Motor=Z
END

RULE Rule0030

OPTIONS
ENABLE="ON"
END

IF (dTheta IS PS) AND (Theta IS PB) THEN
Motor=Z
END

RULE Rule0031

OPTIONS
ENABLE="ON"
END

IF (dTheta IS Z) AND (Theta IS PB) THEN
Motor=NS
END

RULE Rule0032

OPTIONS
ENABLE="ON"
END
```

```
 IF (dTheta IS NS) AND (Theta IS PB) THEN
 Motor=PS
 END
 END

 CONNECT
 FROM Theta
 TO Pendulum_rules
 END

 CONNECT
 FROM dTheta
 TO Pendulum_rules
 END

 CONNECT
 FROM Pendulum_rules
 TO Motor
 END

 CONNECT
 FROM Mrate
 TO Pendulum_rules
 END
END

SIMULATE Simulation0000

 OPTIONS
 DURATION=6
 SAMPLETIME=0.001
 END

 CHART Chart0000
 HCHART="time"
 VCHART="Motor"
 VCHART="x3"
 TITLE="Single Pendulum"
 HTITLE="time, seconds"
 VTITLE="angle, radians"
 HMAX="6"
 HMIN="0"
 HTICK="0.5"
 VMAX="1"
 VMIN="-1"
```

```
VTICK="0.1"
END

CSPLOT CSPlot0000
XVARNAME="Theta"
YVARNAME="dTheta"
BIND="Theta=-128"
BIND="dTheta=-128"
END

MODEL Model0000

OPTIONS
DIFFEQ="OFF"
END

#CODE
pi=3.1415926;
dt=timestep;
gain=10;


x1=0;
x2=0;
x3=0;
x4=0;
pend_angle=x3;

cpr=256/6.28; /* counts/rad */
cprps=256/20; /* counts/rad/s */
torpc=.67/256; /* torque/count, (Nm) */
cpmrps=256/3/10; /* counts per motor rad/s */
#END_CODE

#CODE
Motorf=floor(Motor+.5);
tm=gain*torpc*Motorf;
if tm>.67 then tm=.67; end
if tm<-.67 then tm=-.67; end

s3=sin(x3);
c3=cos(x3);

den=2*c3*c3-27;
```

```
x1d=x2;
x3d=x4;

x2d=1/32*(72*s3*x4*x4+40*c3*x4+45*x2-3136*c3*s3-45000*tm)/den;
x4d=1/4*(8*c3*s3*x4*x4+60*x4-5000*c3*tm+5*c3*x2-4704*s3)/den;


x1=x1+x1d*dt;
x2=x2+x2d*dt;
x3=x3+x3d*dt;
x4=x4+x4d*dt;

motor_angle=x1;
pend_angle=pend_angle+x3d*dt;

if x1>pi then x1=x1-2*pi; end
if x1<-pi then x1=x1+2*pi; end
if x3>pi then x3=x3-2*pi; end
if x3<-pi then x3=x3+2*pi; end

Theta=x3*cpr; /* counts */
dTheta=x4*cprps;
Mrate=x2*cpmrps;

if Theta>127 then Theta=127; end
if Theta<-127 then Theta=-127; end
if dTheta>127 then dTheta=127; end
if dTheta<-127 then dTheta=-127; end
if Mrate>127 then Mrate=127; end
if Mrate<-127 then Mrate=-127; end

Theta=floor(Theta+.5);
dTheta=floor(dTheta+.5);
Mrate=floor(Mrate+.5);
#END_CODE
 END
END


;TEST PROGRAM FOR INVERTED ONE ARM PENDULUM USING THE 68HC811E2
MICROCONTROLLER
;BY TOM SUTHERLAND
;
; 1/7/94 with mods through 7/8/94
```

```
;************************************************************************
;
;DEFINITIONS FOR REGISTER LOCATIONS
;Purpose of initializing each of the registers is to be able to perform
;bit manipulations on the registers without having to use indexed
;addressing.

ORIG_INIT EQU $103D      ;Location of INIT register before
                         ;registers are remapped to page 0.

             ;Location of registers after being remapped to page 0

PORTA      EQU    $0000    ;I/O Port A
PIOC       EQU    $0002    ;Parallel I/O Control Register
PORTC      EQU    $0003    ;I/O Port C
PORTB      EQU    $0004    ;Output Port B
PORTCL     EQU    $0005    ;Alternate Latched Port C
DDRC       EQU    $0007    ;Data Direction for Port C
PORTD      EQU    $0008    ;I/O Port D
DDRD       EQU    $0009    ;Data Direction for Port D
PORTE      EQU    $000A    ;Input Port E
CFORC      EQU    $000B    ;Compare Force Register
OC1M       EQU    $000C    ;OC1 Action Mask Register
OC1D       EQU    $000D    ;OC1 Action Data Register
TCNT       EQU    $000E    ;Timer Counter Register (16 bits)
TIC1       EQU    $0010    ;Input Capture 1 Register (16 bits)
TIC2       EQU    $0012    ;Input Capture 2 Register (16 bits)
TIC3       EQU    $0014    ;Input Capture 3 Register (16 bits)
TOC1       EQU    $0016    ;Output Compare 1 Register (16 bits)
TOC2       EQU    $0018    ;Output Compare 2 Register (16 bits)
TOC3       EQU    $001A    ;Output Compare 3 Register (16 bits)
TOC4       EQU    $001C    ;Output Compare 4 Register (16 bits)
TI4O5      EQU    $001E    ;Output Compare 5 / Input Capture 4
TCTL1      EQU    $0020    ;Timer Control Register 1
TCTL2      EQU    $0021    ;Timer Control Register 2
TMSK1      EQU    $0022    ;Timer Interrupt Mask Register 1
TFLG1      EQU    $0023    ;Timer Interrupt Flag Register 1
TMSK2      EQU    $0024    ;Timer Interrupt Mask Register 2
TFLG2      EQU    $0025    ;Timer Interrupt Flag Register 2
PACTL      EQU    $0026    ;Pulse Accumulator Control Register
PACNT      EQU    $0027    ;Pulse Accumulator Count Register
SPCR       EQU    $0028    ;SPI Control Register
SPSR       EQU    $0029    ;SPI Status Register
SPDR       EQU    $002A    ;SPI Data Register
BAUD       EQU    $002B    ;SCI Baud Rate Control
```

```
SCCR1        EQU     $002C       ;SCI Control Register 1
SCCR2        EQU     $002D       ;SCI Control Register 2
SCSR2        EQU     $002E       ;SCI Status Register
SCDR         EQU     $002F       ;SCI Data Register
ADCTL        EQU     $0030       ;A/D Control Register
ADR1         EQU     $0031       ;A/D Result Register 1
ADR2         EQU     $0032       ;A/D Result Register 2
ADR3         EQU     $0033       ;A/D Result Register 3
ADR4         EQU     $0034       ;A/D Result Register 4
BPROT        EQU     $0035       ;EEPROM Block Protect Register
OPTION       EQU     $0039       ;System Configuration Options
COPRST       EQU     $003A       ;Arm/Reset COP Timer Circuit
PPROG        EQU     $003B       ;EEPROM Program Control Register
HPRIO        EQU     $003C       ;High Priority I-bit interrupt & misc
INIT         EQU     $003D       ;Ram and I/O Mapping Register
TEST1        EQU     $003E       ;Factory TEST Control Register
CONFIG       EQU     $003F       ;COP, ROM, and EEPROM Enables

;*********************************************************************
;DEFINITIONS FOR RAM LOCATIONS ( 192 bytes available )

        DEFSEG ramst, START=$40

        SEG ramst
        ORG $0040 ;Point to the first byte of RAM

POSITION: RMB 2 ;POSITION OF FREE ARM 1.
MSB_POS: RMB 1 ;8-BIT POSITION SAVED FROM VELOCITY.

OLD_POS: RMB 2 ;SAVE LAST POSITION TO GET RATE.

TIMER: RMB 3 ;INCREMENTED EVERY X ms

;*********************************************************************
;DEFINITIONS FOR EEPROM LOCATIONS
;
;These vectors point to the corresponding locations in memory.

EEPROM_START     EQU     $F800   ;Location of EEPROM in memory
VECTOR_TIC1      EQU     $FFEE   ;Location of vector for TIC1
VECTOR_TIC2      EQU     $FFEC   ;Location of vector for TIC2
VECTOR_TIC3      EQU     $FFEA   ;Location of vector for TIC3
VECTOR_TOC2      EQU     $FFE6   ;Location of vector for TOC2
VECTOR_RESET     EQU     $FFFE   ;Location of vector for reset
```

```
;*************************************************************************
;PROGRAM CODE


MFPL        EQU         $f800       ;Location of run-time module

            DEFSEG stkseg, START=$ff
            SEG stkseg
   ORG              $ff
   STAK DS          $1

      DEFSEG entseg, START=$ffe4
      SEG     entseg
   ORG        $ffe4
   FDB        ENTRY

      DEFSEG INTTOC2, START=VECTOR_TOC2
      SEG INTTOC2
      ORG VECTOR_TOC2
      FDB MAIN_ROUTINE

      DEFSEG      INTRESET, START=VECTOR_RESET
      SEG         INTRESET
      ORG         VECTOR_RESET
      FDB         START

      DEFSEG rom, START=$f800
      SEG rom
      ORG $f800 ;Set PC to start of EEPROM

;Initialization code.


START: CLR ORIG_INIT     ;Remap both the registers and
                         ;the ram to page 0. This reduces
                         ;the available ram to 192 bytes,
                         ;but allows use of the bit
                         ;manipulation and test instructions
                         ;with direct addressing. This
                         ;remapping can only be done during
                         ;the first 64 clock cycles after
                         ;reset.
```

```
ENTRY:  LDS     #$00FF          ;Initialize the stack pointer.
        LDAA    #$30
        STAA    BAUD            ;SET SERIAL TO 9600 BAUD
        LDAA    #$0C            ;SET BITS TO ENABLE RECV. AND TRANSMIT.
        STAA    SCCR2           ;ENABLE SERIAL I/O.

        LDAA    #$00            ;SET PORT-C AS INPUTS.
        STAA    DDRC            ; " "

        LDAA    #$80            ;SET INITIAL VALUE IN D/A PORT.
        JSR     MOTOR_DRIVE     ; " " "

                                ;SET UP ADC REGISTERS.
        LDAA    #$80            ;SET ADPU BIT, TO ENABLE CHARGE PUMP FOR ADC.
        STAA    OPTION          ; " "
        LDAA    #$20            ;SET BITS IN REG. ADCTL:SCAN=1,MULT=0
        STAA    ADCTL           ; " "


        CLRA                    ;INITIALIZE COUNT TO ZERO
        STAA    TIMER
        STAA    TIMER+1
        STAA    TIMER+2

        BCLR    TCTL1,$80       ;TOGGLE OUTPUT ON SUCCESSFUL COMPARE
        BSET    TCTL1,$40

        LDD     TCNT            ;ADD 60000 TO MAIN TIMER
        ADDD    #60000
        STD     TOC2            ;STORE VALUE IN COMPARATOR
        LDAA    #$40
        STAA    TFLG1           ;CLEAR TIMER FLAG
        BSET    TMSK1,$40       ;TIMER OUTPUT COMPARE 2
        CLI                     ;ENABLE INTERRUPTS
HANG:   BRA     HANG            ;WAIT HERE FOR INTERRUPT

MAIN_ROUTINE:                   ;THIS IS THE INTERRUPT SERVICE ROUTINE

        LDD     TOC2
        ADDD    #60000          ;ADD 30 ms TO TOC2
        STD     TOC2

        LDD     #1              ;INCREMENT 24-BIT COUNT BY 1
        ADDD    TIMER+1
        STD     TIMER+1
```

```
            LDAA #0
            ADCA TIMER
            STAA TIMER

            LDAA #$40
            STAA TFLG1      ;CLEAR TOC2 FLAG

            LDAA ADR1       ;8-BIT A/D OF MOTOR TACH.
            CLC             ;REMOVE BIAS.
            SUBA #$7F
            JSR SND_WORD
            STAA _Mrate     ;SAVE FOR CONTROLLER.
            BGE HEEP
            COMA
            INCA
            LSRA
            LSRA
;       LSRA
            LSRA
            COMA
            INCA
            BRA HEEP1
HEEP        LSRA
            LSRA
;       LSRA
            LSRA
HEEP1       ADDA _Mpos
            STAA _Mpos
            JSR SND_WORD
            JSR ARM_VELOCITY ;GET FREE ARM RATE (12-BIT) [SHML]
            STAA _dTheta    ;SAVE FOR CONTROLLER.
;       JSR SND_WORD
            LDD POSITION    ;GET CURRENT POSITION.
            STD OLD_POS     ;SAVE FOR NEXT RATE CALC.
            LDAA #$80       ;CONSTANT TO MATCH MFPL INPUT SCALE
            CLC
            SUBA MSB_POS    ;8-BIT POSITION OF FREE ARM
            STAA _Theta     ;SAVE FOR CONTROLLER
;       JSR SND_WORD
            LDX #_PENDULUM1 ;MicroFPL NAME
            JSR MFPL        ;DETERMINE ACTION THAT SHOULD BE TAKEN.
            LDAA #$7F       ;CONSTANT TO SHIFT 0-255.
            CLC
            SUBA _MOTOR     ;DO THE SHIFT
```

```
;           JSR     SND_WORD
                    CMPA #$40        ;LIMIT +SIDE.
                    BHS SKIP3        ;GO CHECK NEGATIVE VOLTS.
                    LDAA #$40        ;LIMIT TO +5 VOLTS.
                    BRA SKIP         ;CAN'T BE NEGATIVE.
SKIP3:              CMPA #$C0        ;LIMIT -SIDE.
                    BLS SKIP         ;NOT TOO NEGATIVE.
                    LDAA #$C0        ;LIMIT TO -5 VOLTS.
SKIP:               JSR MOTOR_DRIVE ;COMMAND MOTOR.
;                   JSR SND_WORD
                    LDAA #32
                    JSR SND_BYTE
                    RTI              ;RETURN FROM INTERRUPT

;SUBROUTINE SECTION

MOTOR_DRIVE:  COMA              ;SUBROUTINE TO DRIVE MOTOR.
              INCA              ;REVERSE SIGN OF DRIVE.
              STAA PORTB        ;SEND 8 BIT DATA TO DIGITAL-TO-ANALOG CON
                                VERTER.
              RTS               ;RETURN FROM SUBROUTINE MOTOR_DRIVE.

ARM_VELOCITY: ;SUBROUTINE, DETERMINING FREE ARM'S VELOCITY.
              LDAB PORTD        ;READ RESOLVER-TO-DIGITAL CONVERTER,4 BITS
                                (LOW OF 12 BITS).
              LDAA PORTC        ;READ RESOLVER-TO-DIGITAL CONVERTER,8 BITS
                                (HIGH OF 12 BITS).
              STAA MSB_POS      ;SAVE CURRENT 8-BIT POSITION.
              LSLB              ;SHIFT ONE BIT LEFT.
              LSLB              ;" "
              LSRD              ;SHIFT DOUBLE ACC'AB' RIGHT.
              LSRD              ;" "
              LSRD              ;" "
              LSRD              ;" "
              CLC               ;CLEAR CARRY BIT.
              STD POSITION      ;STORE THE FREE ARM'S CURRENT POSITION.
              SUBD OLD_POS      ;SUBTRACT POSITIONS.
              BMI NEG           ;CHECK SIGN OF VELOCITY.
POS:          CPD #$7F          ;SEE IF MAX POSITIVE EXCEEDED.
              BCS VEL1          ;NOT POSITIVE MAX YET.
              LDAB #$7F ;MAX VALUE OF 127.
              BRA VEL1 ;THROUGH WITH POSITIVE.
NEG:          CPD #$FF7F        ;SEE IF MAX NEGATIVE EXCEEDED.
              BCC VEL1          ;DON'T LIMIT.
              LDAB #$F0         ;MAX VALUE OF -128.
```

```
VEL1:       TBA               ;PUT IN A-REGISTER.
            COMA
            INCA
            RTS               ;RETURN.

SND_WORD:                     ;CONVERT ACC'A' TO ASCII AND SEND TO SERIAL
                              PORT.

            PSHA              ;SAVE ORIGINAL DATA.
            PSHA              ;SAVE ORIGINAL DATA.
            LSRA              ;SHIFT HIGH NIBBLE RIGHT.
            LSRA              ;" "
            LSRA              ;" "
            LSRA              ;" "
            JSR HEX_ASCII     ;CONVERT HIGH NIBBLE TO ASCII.
            JSR SND_BYTE      ;SEND HIGH NIBBLE TO SERIAL PORT.
            PULA              ;GET ORIGINAL DATA.
            JSR HEX_ASCII     ;CONVERT LOW NIBBLE TO ASCII.
            JSR SND_BYTE      ;SEND LOW NIBBLE TO SERIAL PORT.
            PULA              ;RESTORE ORIGINAL DATA.
            RTS               ;RETURN FROM SND_WORD SUBROUTINE


HEX_ASCII:                    ;SUBROUTINE TO CONVERT FROM HEX TO ASCII.
            ANDA #$0F         ;MASKOFF LOW NIBBLE.
            CMPA #$09
            BGT HEX_A_CHAR ;IF ITS GREATER THAN 9hex ITS A CHARACTER.
            ADDA #$30         ;ADD 30hex TO MAKE A NUMBER ASCII.
            RTS               ;RETURN FROM HEX_ASCCI SUBR
HEX_A_CHAR: ADDA #$37         ;ADD 37hex TO NIBBLE.
            RTS               ;RETURN FROM HEX_ASCII SUBROUTINE.

SND_BYTE:                     ;SUBROUTINE TO SEND A BYTE TO SERIAL PORT.
            BRCLR   SCSR2,$80,$
            STAA    SCDR
SND_WAIT:   BRCLR   SCSR2,$80,$
            RTS

RECV_BYTE:  BRCLR SCSR2,$20,$ ;WAIT FOR CHARACTER IN BUFFER
            LDAA SCDR         ;READ CHARACTER.
            RTS               ;RETURN FROM SUBROUTINE RECV_BYTE.


rem c:\til\mfpl6811 -v c:\til\penddown
```

```
copy onearm.asm+penddown.asm+tmpend temp.asm
avmac11 temp.asm >penddown.lst
    avlink penddown.mot = temp.obj, rtm6811.obj OF=MOT -PS(RTM_ROMSEG,f800h) -
PS(rom,fddfh) -PS(RAMSEG,48h) -PS(RAMVARS,51H)
```

AVLINK —— LOAD MAP
For: Mixed Languages

RELOCATED SEGMENTS - CLASS 'M'

SEGMENT NAME	START	STOP	LENGTH ovl/cat def/undef
RAMSEG	0000	0008	0009 Concat Defined
RAMVARS	000a	001f	0016 Concat Defined
RAMST	0040	0047	0008 Concat Defined
STKSEG	00ff	00ff	0001 Concat Defined
REGS	1000	103f	0040 Overld Defined
RTM_ROMSEG	f800	fbe6	03e7 Concat Defined
ROMVARS	fbe7	fdde	01f8 Concat Defined
ROM	fddf	fed8	00fa Concat Defined
ENTSEG	ffe4	ffe5	0002 Concat Defined
INTTOC2	ffe6	ffe7	0002 Concat Defined
INTRESET	fffe	ffff	0002 Concat Defined

ZERO LENGTH SEGMENTS

SEGMENT	START
PAGE0	0000
CODE	0000
DATA	0000

No Transfer Address.

```
pendulum.mot=temp.obj, (later, improved mapping)
  rtm6811.obj
  OF=MOT
  -PS(RTM_ROMSEG,F800H)
  -PS(ROM,FE47H)
  -PS(RAMSEG,48H)
  -PS(RAMVARS,51H)
    -PS(ROMVARS,FBE7H)
```

Object code in ASCII format as sent to the 6811 by Procom

```
S107FFE4FDE8FE280A
S105FFFEFDDF21
S123FDDF7F103DCC0600FD00438E00FF8630972B860C972D860097078680BDFE858600B720
S123FDFF004286809739862097304FB70045B70046B70047152080142040DC0EC3EA60DD18
S123FE1F18864097231422400EDC18C3EA60DD18CC0001F30046FD00468600B90045B70029
S123FE3F4586409723BDFE88B7000BFC0040FD004386809003B7000ACEFBE7BDF800F600A4
S123FE5F0C58588680B0000C434CBDFE8520BA963185802709847F448A80B7004239438412
S123FE7F7F44B7004239970439D6089603585858040404040CFD0040B3004381002B03C1
S123FE9F17200317434C39363644444444BDFEBBBDFEC732BDFEBBBDFEC73239840F810900
S11DFEBF2E038B30398B3739132E80FC972F132E80FC39132E20FC962F3931
S123FBE7000A000DFBF1FC65FC7EFC09FC11FC19FC1FFC27FC2FFC37FC3DFC45FC4DFC5743
S123FC07FC5DF600FFFF0A007F00EC00FFFF14007F00140032FF7FFFC400D8FFEC007F00BD
S123FC27000014FF28007F00D800ECFF00007F00CEFFEC007F00140028FF3C007F00FF0090
S123FC4714FF1E007F00E200ECFFECFFFF007F00B0FFC4007F003C0050FF50FF7FFF300732
S123FC673C97032707E42C029F07F9D8007A079AA1063A072D0505010000100001020110 8D
S123FC870101040110020106001003010801100401 0A011005010C011006010E001007019C
S123FCA71000100801120010090114001 00A011600100B120C0E000E01050B000C0E020E0F
S123FCC703050B000C0E040E03050B000C0E010E03050B000C0E050E03050B050C0E060E18
S123FCE703050B0A0C0E020E07050B0F0C0E040E07050B140C0E010E07050B000C0E050EC8
S123FD0707050B000C0E060E07050B000C0E000E06050B0F0C0E000E05050B140C0E080EAE
S123FD2706050B0F0C0E080E05050B000C0E090E05050B0F0C0E090E01050B140C0E080E73
S123FD4704050B0A0C0E080E01050B050C0E090E06050B0F0C0E080E02050B0A0C0E090E61
S123FD6704050B000C0E090E02050B0A0C0E000E04050B050C0E000E02050B0A0C0E060E64
S123FD870A050B000C0E060E0B050B0F0C0E050E0A050B000C0E010E0A050B000C0E040E30
S123FDA70A050B000C0E020E0A050B0A0C0E020E0B050B000C0E040E0B050B000C0E010E1B
S11BFDC70B050B000C0E050E0B050B000C0A0C0216028000CEFBE73918
S123F800FF0000EE08FF000220130140012801230129013101360137013 8FE0002A600087B
S123F820FF000281172D0220FECEF833161B1B163A6E007EF8787EF8897EF9167EF9597E13
S123F840F9867EF9BB7EF9C57EF9CD7EF9D47EF9E57EF9F07EFA727EFB0C7EFB2B7EFB4495
S123F8607EFB4B7EFB647EFB6B7EFB717EFB977EFBA87EFBAE7EFBC8390140012801240 1B3
S123F8802901D7015C015A015A18FE000218E6001808FE0000EE043AEE003CFE0000EE00DA
S123F8A018E60018083AA60018FF000238A1002E06E601377EF81AE600F70004E601F700B3
S123F8C0050808A10027EA2EEEE601F0000527E12521B000043DFD0006A600B00004B70012
S123F8E008FE00068600F600088F028FFB0005377EF81A50B000043DFD0006A600B00004EF
S123F900B70008FE00068600F600088F028F50FB0005377EF81A18FE000218E6001808FE31
S123F9200000EE043AEE003CFE0000EE0018E60018083AA60018FF000238A1002206E60182
S123F940377EF81AE600F70004E601F700050808A10027EA22EE7EF8C918FE000218E600F1
S123F9601808FE0000EE043AEE0018E600180818FF000218FE000018EE00183A18E6003A6E
S123F980A600367EF81A18FE000218E6001808FE0000EE043A18E600180818FF000218FE4A
S123F9A0000018EE00183A18E6008600C180250286FFE3008FA600367EF81A323311250100
S123F9C017367EF81A32331122F71720F432331B367EF81A32331B28082A04867F200286BB
S123F9E080367EF81A32331B240286FF367EF81AFE0000EE0218FE000218E600180818FF91
S123FA0000023AA60681002627A6028100262109A6068100261AA6028100261409A60281B1
```

S123FA2000262518FE0002180818FF00027EF81AA60681802412A6028180240C6801690206
S123FA4068046905690620E88600E602C10027FE3736A606E60538028F17FE0000EE001815
S123FA60FE000218E600180818FF00023AA7007EF81A8D6A323618E6003DEB008900B70015
S123FA8004E700323618E6013DEB018900FB00048900B70004E701A602BB0004A7023236BB
S123FAA018E6023DEB038900B70004E703323618E6033DEB048900FB00048900B70004E7A1
S123FAC0043218E6043DEB058900FB00048900B70004E705A606BB0004A7067EF81A18FE47
S123FAE0000018EE06FE0002E60008FF0002183A18FF0006FE0000EE0218FE000218E60094
S123FB00180818FF00023A18FE0006398DD086003337EB008900E700168600EB018900E779
S123FB2001168600EB02E7027EFA9EFE0000EE0018FE000218E600180818FF00023AA6001D
S123FB40367EF81AFE0000EE0220E5FE0000EE0018FE000218E600180818FF00023A32A79A
S123FB60007EF81AFE0000EE0220E53243367EF81AFE0000EE0218FE000218E60018081887
S123FB80FF00023A8600A700A701A702A703A704A705A7067EF81A18FE000218A600180874
S123FBA018FF0002367EF81A3236367EF81AFE0000EE0018FE000218E600180818FF0002F9
S123FBC03A3236A7007EF81AFE0000EE0018FE000218E60018083AA60018A000180818FF57
S10AFBE00002A7007EF81AE1
S9030000FC

APPROVAL

INVERTING THE PENDULUM USING FUZZY CONTROL

Compiled by R.R. Kissel and W.T. Sutherland

The information in this report has been reviewed for technical content. Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

E.C. Smith
Director, Astrionics Laboratory

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE A... ...TE... ...OVERED
	May 1997	Technical Memorandum

4. TITLE AND SUBTITLE
Inverting the Pendulum Using Fuzzy Control (Center Director's Discretionary Fund Final Report—Project 93–02)

5. FUNDING NUMBERS

6. AUTHORS
R.R. Kissel and W.T. Sutherland

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM–108535

11. SUPPLEMENTARY NOTES
Prepared by Astrionics Laboratory, Science and Engineering Directorate

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A single pendulum was simulated in software and then built on a rotary base. A fuzzy controller was used to show its advantages as a nonlinear controller since bringing the pendulum inverted is extremely nonlinear. The controller was implemented in a Motorola 6811 microcontroller. A double pendulum was simulated and fuzzy control was used to hold it in a vertical position. The double pendulum was not built into hardware for lack of time. This project was for training and to show advantages of fuzzy control.

14. SUBJECT TERMS
fuzzy control, inverted pendulum, nonlinear control, Lagrange

15. NUMBER OF PAGES 64

16. PRICE CODE NTIS

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited