Motor Control of an Oscillating Pendulum

By Nick Myers and (Chirag Patel)
Paper Written by Nick Myers

Bradley University Department of Electrical Engineering
EE452 Senior Laboratory
Advised by Dr. James Irwin and Mr. José Sánchez
May 12, 2004

Abstract

The goal of this project is to create a microcontroller based pendulum control that could be used in applications such as clock pendulums, self-rocking baby cradles, and robot arm movement. This paper discusses how the EMAC microcontroller development system is used to regulate the power supply of a Mabuchi DC (RS-385SH) motor in order to oscillate the weighted pendulum from a standstill to a predefined angle. Once the pendulum reaches the predefined angle, the oscillation is kept at that constant angle. The uses of H-bridge circuitry to enable bi-directional motor control, and optical sensors (H21A1) to provide feedback pertaining to the pendulum's location are also described in detail. Using the LCD and keypad of the EMAC microcontroller development system as a user interface, the system was able to start on user command, increase the angle of oscillation, maintain the desired angle of oscillation, and stop at any time on user command.

Table of Contents

Introduction

The objective of this project is to design and build a system that initiates, and controls, the oscillation of a weighted pendulum using an EMAC microcontroller development system to control motor power bursts. Once the pendulum's oscillation has been initiated, the EMAC microcontroller development system will cause the angle of oscillation to increase until it has achieved a predefined angle of oscillation. The EMAC microcontroller development system will adjust the motor power bursts so that pendulum will then oscillate at this constant angle until the system is shut down. This type of motor control has applications in clock pendulums, self-rocking baby cradles, and can be extended to any type of robot arm movement.

The inputs to the system are a user entered "Start" or "Stop", while the output of the system is the actual angle of pendulum oscillation. These inputs and outputs are shown in the system level block diagram (Figure 1).

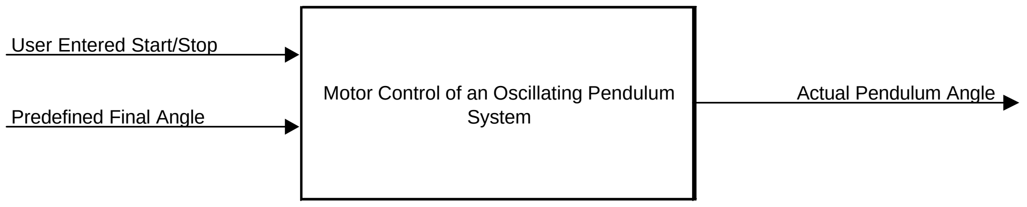


Figure 1. System Level Block Diagram

System Description

The pendulum is oscillated by a small DC motor that runs in the clockwise and counterclockwise directions (see attached data sheet in appendix A1). The selected motor is a non-geared motor so that when the motor is not being pulsed, the pendulum is able to swing freely. The bi-directional bursts are controlled by H-bridge hardware. This H-bridge hardware consists of two n-channel and two p-channel transistors set as inverters on each of the inputs of the motor. The H-bridge will switch the direction of the burst before each motor pulse. Additional hardware is required to interface the EMAC development system board to the H-bridge.

The EMAC development system controls the constant timing for the initial motor bursts. The bursts will then gradually increase in pulse length to increase the pendulum's angle. After a set number of pulses from the initial and increasing pulse length codes, the EMAC microcontroller development system sends bursts only after external interrupt three, IE3, is triggered. External interrupt three is associated with the equilibrium sensor. IE3 is triggered every time that the pendulum passes through the equilibrium sensor. The length of the motor pulses will continue to increase, and the pulses will continue to switch polarity before every pulse. This motor pulsing will continue until the angle sensor is crossed. The angle sensor trigger is set at a predefined angle to set the desired angle of oscillation. External interrupt four, IE4, is triggered when the equilibrium sensor is blocked. Once this interrupt has been triggered, the motor pulse length is kept constant. The system then checks before each pulse to see if the pendulum is still at a constant angle of oscillation. This is done by counting the number of times that the pendulum triggers IE4 on each oscillation. If the pendulum is overshooting the desired angle of oscillation, then the pulses will be shortened, and if the pendulum is not reaching its desired angle, the pulses will be incremented. Examples of the angle sensor outputs are shown in Figure 2.

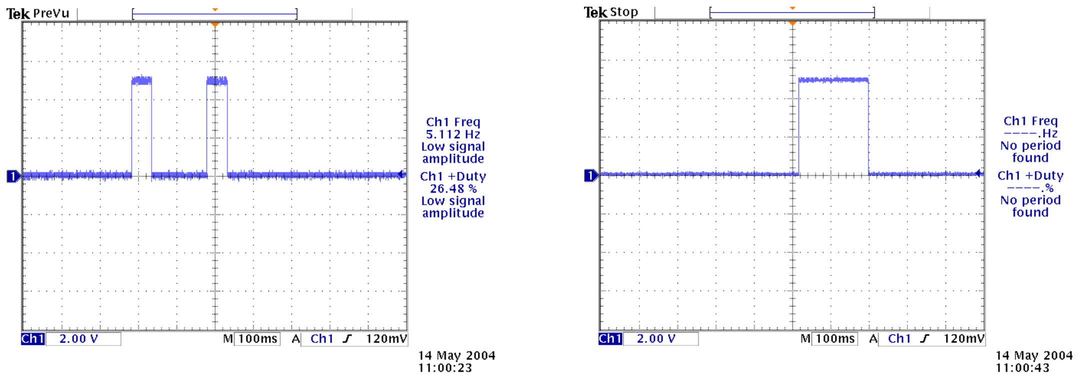


Figure 2. Oscilloscope screen captures of angle sensor outputs when triggered twice and once (left to right)

The inputs to the system are a user entered "Start" command and a user entered "Stop" command. The user is prompted to start or stop the system using the LCD of the EMAC Development System, where then the user can enter the command using the keypad. Both of these inputs are tied to external interrupt one, IE1, of the microcontroller. The block diagram of the entire described system is shown in the subsystem level block diagram (Figure 3).

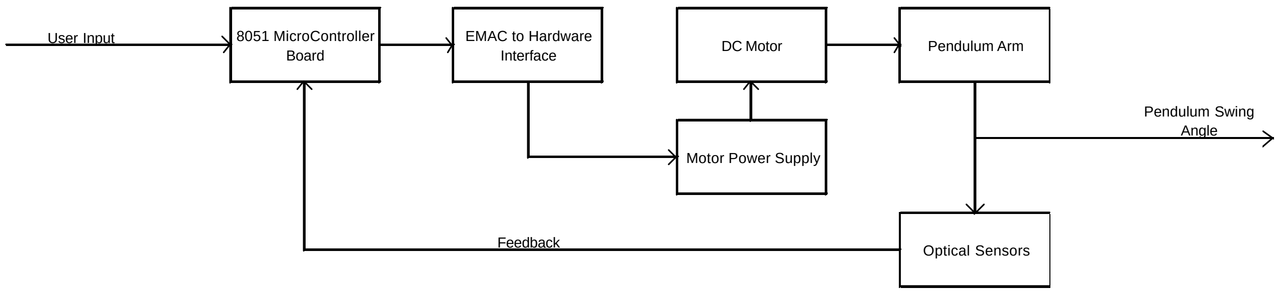


Figure 3. Subsystem Level Block Diagram

Block Diagrams

I. EMAC Microcontroller Development System

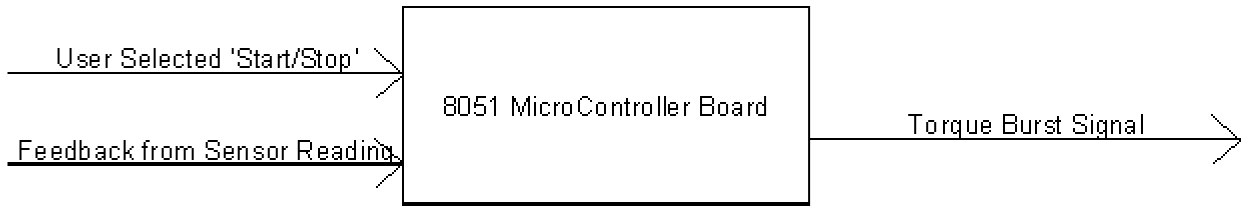

User Selected 'Start/Stop'
Feedback from Sensor Reading
8051 MicroController Board
Torque Burst Signal

Figure 4. EMAC Microcontroller Development System Block Diagram

The inputs to the EMAC microcontroller development subsystem are the user entered “Start” and “Stop”, and the feedback information from the optical sensors. The user interface is accomplished using the built in keypad and LCD on the EMAC development board. The user is given command prompts via the LCD, and the inputs are entered by pressing keys on the keypad.

The sensor feedback is provided through P1.0 and P1.1 of the EMAC development board. The information provided by the sensors is whether they are obstructed by the pendulum. When the sensors are obstructed, a high, ~5V, signal is provided by the sensor. This voltage level then provides an interrupt for the code. The interrupt service routine associated with the equilibrium sensor sends the signal for a motor power burst, while the interrupt associated with the angle sensor provides information pertaining to the pendulum’s angle of oscillation.

The EMAC microcontroller development system also provides the motor power burst, pulse length timing, and H-bridge switching codes. These code modules run continually to oscillate the pendulum while the sensor feedback makes adjustments to keep a constant oscillation.

II. H-bridge Circuitry

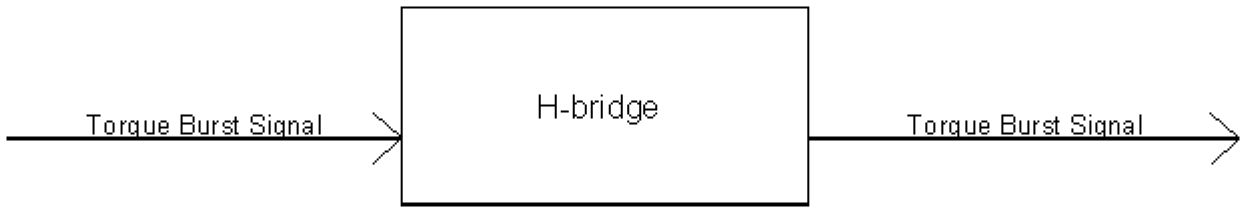


Figure 5. H-bridge System Block Diagram

The H-bridge circuitry allows for bi-directional rotation of the motor to increase the efficiency of the system. The bi-directional rotation is accomplished by reversing the polarity of the motor power before each torque burst.

The H-bridge circuitry is comprised of four MOSFET transistors. One p-channel and one n-channel transistor are arranged as an inverter with the output connected to one of the motor inputs. The other two transistors are arranged in the same manner and attached to the remaining motor input. Opposite input levels are then given to each of the inverters to provide the power for the motor in the clockwise or counterclockwise direction (Figure 5).

The direction of the H-bridge is controlled by the EMAC microcontroller development system. The pulse length signals are provided on pins P1.4 and P1.5. P1.4 is connected to one of the inverter pairs, while P1.5 is connected to the other (see Figure 12 on p.7). The EMAC microcontroller development system code switches the polarity of the motor by switching between the output pins. The EMAC microcontroller development system code ensures that both output pins are turned off before one of them is turned on to prevent any damage to the motor, H-bridge, or power supply by sending power to both inputs (see appendix B-?).

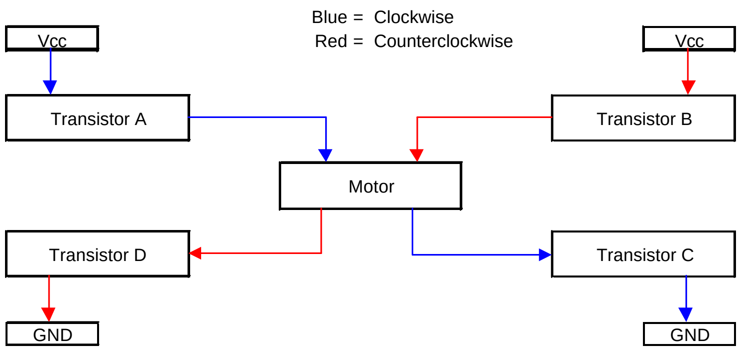


Figure 6. H-bridge Current Flow

III. DC Motor Subsystem

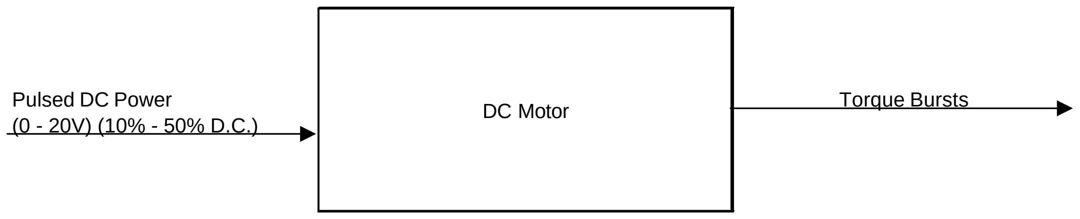


Figure 7. DC Motor Subsystem Block Diagram

The DC Motor subsystem is a small (< 1lb.) Mabuchi DC motor (Appendix A-1). The motor is rated at 20VDC, but operates at supplies as low as 6VDC. A non-geared motor is more suitable for this application, so that it will be able to turn freely when the motor is not powered. This is important so that the oscillation of the pendulum is not hindered.

IV. Pendulum Subsystem

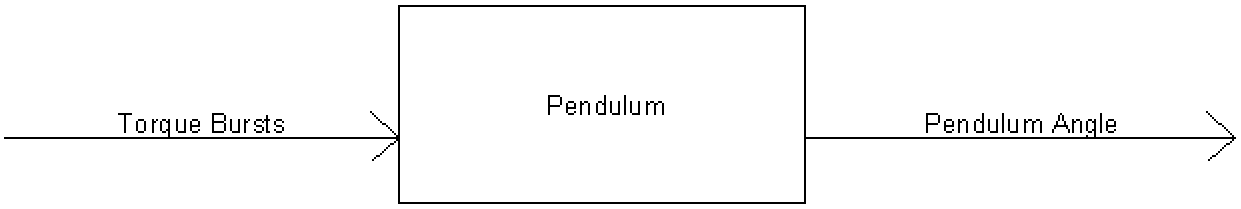


Figure 8. Pendulum Subsystem Block Diagram

The pendulum subsystem is comprised of a weighted object affixed to the motor shaft. The pendulum weight is too heavy for the motor to be able to oscillate freely. The design used for the pendulum is a circular disc with weights attached to the bottom and sensor triggers attached on it. One sensor trigger is attached at equilibrium, while the other is movable to select the desired final angle of oscillation. The pendulum subsystem is shown in Figure 9. The equilibrium sensor and sensor trigger, as well as the angle sensor and sensor trigger are highlighted in Figure 9. The lead weights are affixed to the bottom of the circular pendulum.

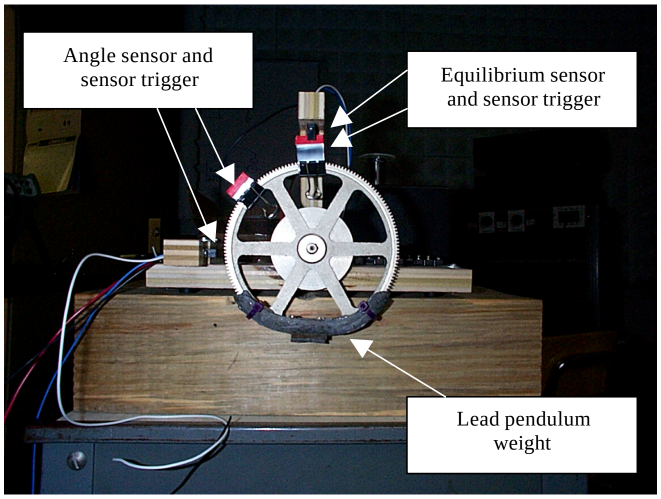

Angle sensor and sensor trigger
Equilibrium sensor and sensor trigger
Lead pendulum weight

Figure 9. Constructed Pendulum Subsystem

V. Optical Sensor Subsystem

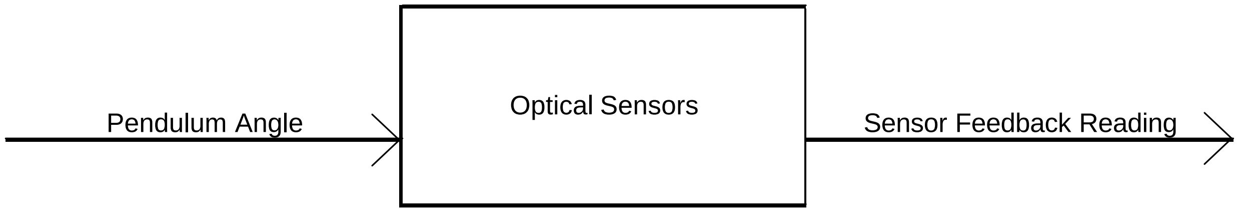


Figure 10. Optical Sensor Subsystem Block Diagram

Two optical sensors are used to provide feedback to the EMAC microcontroller development system. The sensors are two Fairchild H21A1 optical sensors that provide a 0-5V signal when provided with a supply voltage of 5V DC. One of the sensors is positioned to be the equilibrium sensor. This sensor triggers an interrupt in the code every time that the pendulum passes through equilibrium, resulting in a pulse signal being sent.

The second sensor trigger can be positioned anywhere on a given section of the pendulum. This sensor provides the desired angle of oscillation for the system. This sensor triggers another interrupt in the EMAC microcontroller development system code. This interrupt service routine checks the number of times that the pendulum triggers the interrupt between pulses. If the number is zero, then the pendulum is not high enough, so the pulse length is increased. If the number is two, then the pendulum is overshooting the desired angle, so the pulse length is decreased. If the number is one, the pendulum is at the correct angle of oscillation, so the pulse length will remain constant. Examples of angle sensor outputs are shown in Figure 2.

Circuit Diagrams

The hardware portion of this project can be separated into five main circuits.

- H-bridge circuitry
- Two (2) transistor switches to interface EMAC microcontroller development system to H-bridge
- Two (2) optical sensors

The H-bridge, as shown in Figure 12, allows the motor to spin in both the clockwise and counterclockwise directions by switching the polarity of the motor. The truth table for the H-bridge is shown in Figure 11. Only three states of the H-bridge are shown because the idle, clockwise, and counterclockwise states are the only states that are used by the system. This is regulated by the H-bridge switching code.

A	B	C	D	Motor Direction
0	0	0	0	Idle
1	0	0	1	Forward
0	1	1	0	Reverse

Figure 11. Truth Table for H-bridge Circuitry

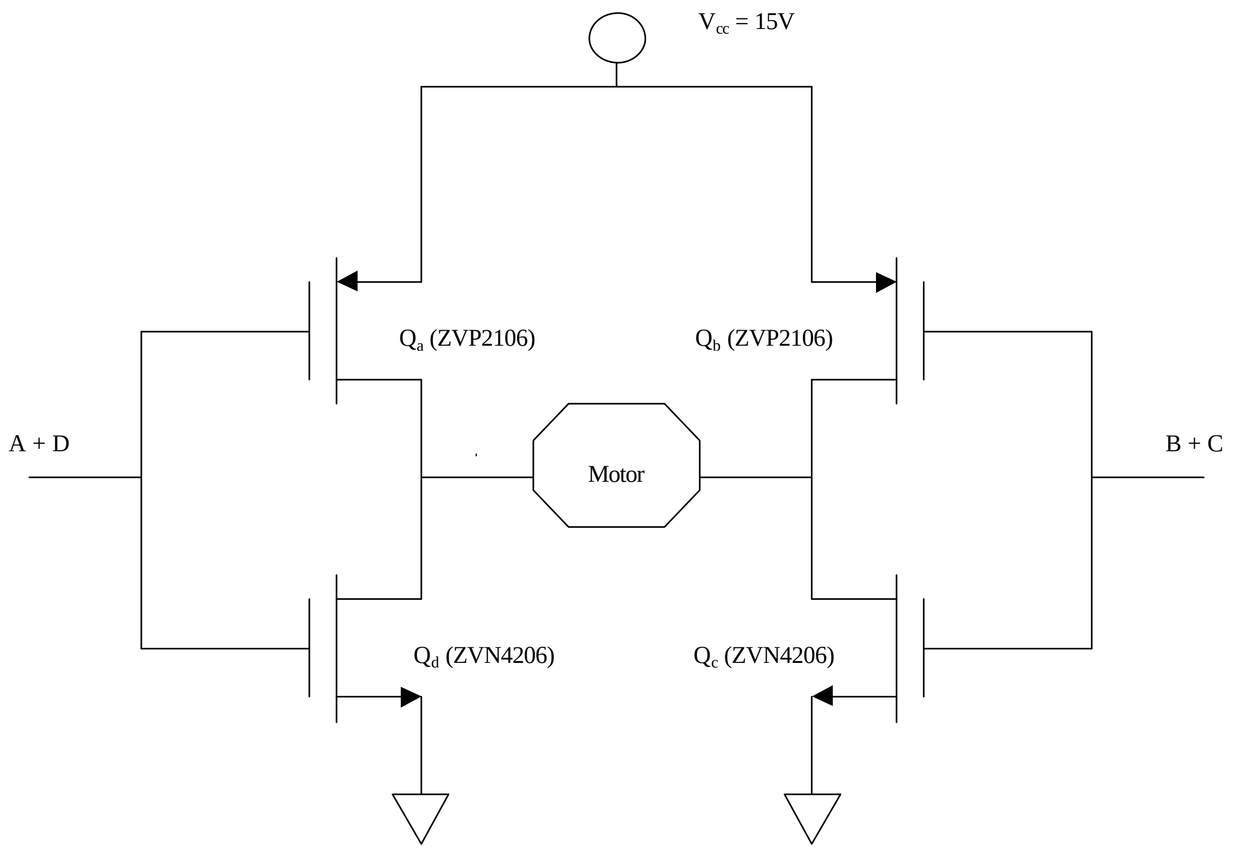


Figure 12. H-bridge Circuitry Schematic

The next two circuits used in this system are identical sets of transistor switches. These switches interface the EMAC development board to the H-bridge circuitry. The transistor switches were originally designed to be used with an H-bridge consisting of bipolar transistors. The port 1 pins from the EMAC development board provide only 5V at 80uA of current. This current level was not high enough to run the H-bridge, so transistor switches were designed to turn the EMAC microcontroller development system output into the desired H-bridge input. Each transistor switch is a set of two cascaded

2N2222a transistors (Figure 13). When the 5V signal from the EMAC microcontroller development system is sent, the base-emitter voltage is 0.7V, so the output voltage is 4.3V. By choosing the resistor values in the switches, the base current for the second transistor could be designed to be 15uA. This would be enough to turn on the second transistor and pull the output to ground. When the second transistor is turned off, the output voltage is constant at V_{cc}.

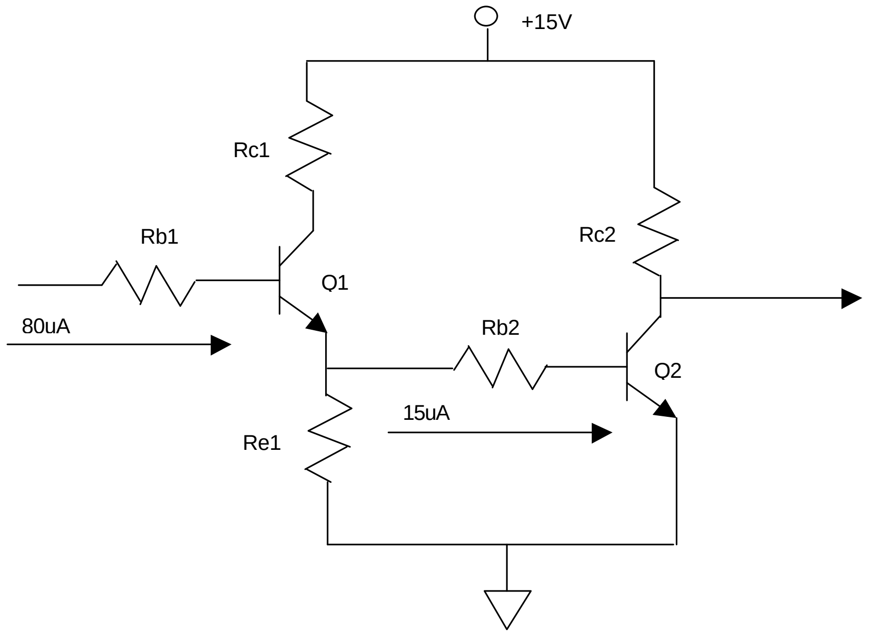


Figure 13. EMAC microcontroller development system to H-bridge Interfacing Transistor Switches

The final two circuits used in this system are the two optical sensors and the resistors associated with the sensor switching times (Figure 14). The two optical sensors, as previously discussed, provide feedback to the EMAC microcontroller development system involving the location of the pendulum. The first sensor triggers an interrupt when the pendulum is at equilibrium, while the second sensor triggers an interrupt when the pendulum is at the desired angle of oscillation. Because the pendulum could be considered relatively slow moving, the switching times of the sensors should not present any major problems.

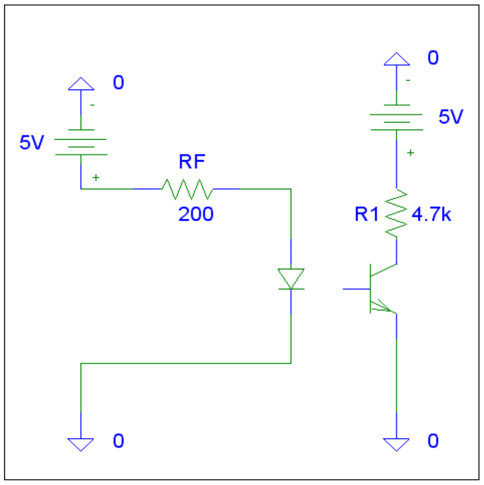


Figure 14. Optical Sensor Circuitry

Design Equations and Calculations

As previously stated, the resistors of the transistor switching circuitry used to interface the EMAC microcontroller development system to the H-bridge can be adjusted to produce a desired current flow through the transistors. To attain this desired current the equations in Equation Set 1 were used. All resistor and transistor names are in reference to Figure 13.

for Rc2 = 10k Ohms
Ic2(sat) = 15/10 = 1.5mA => Ib2(sat) = 0.015mA

Ie1 = Ib2 + Ire1

Choose Ib1 = 40uA
so Ic1 = 4mA max at saturation
(Rc1 + Re1) * 4 = 15
Rc1 + Re1 = 15/4 ~ 4k

Ve1 < or = 4.3V so choose 2V
=> assuming saturation (4mA) (2V)
Re1 = 2/4 = 0.5k or 510 Ohms
Rc1 = 3.5k or 3.6k Ohms

When Q is saturated Ve = 2V

$$Rb2 = \frac{Ve1 - Vbe2}{0.03} = \frac{2 - 0.8}{0.03} = 40k \text{ or } 39k \text{ Ohms}$$

$$Rb1 = \frac{Vp1 - Vbe1 - Ve1}{0.04} = \frac{5 - 0.8 - 2}{0.04} = 55k \text{ or } 56k \text{ Ohms}$$

so: Rb1 = 56k Ohms Rc1 = 3.6k Ohms Re1 = 510 Ohms
Rb2 = 39k Ohms Rc2 = 10k Ohms

Equations Set 1. Design Equations for the Transistor Switches

The resistors associated with the optical sensor circuitry shown in Figure 14 affect the on and off switching times of the optical sensors. The equations shown in Equation Set 2 were used to set switching times that would not hinder the performance of the system.

Rf = 200
Limits Current to 20 mA
If = 5V/200 = 20mA

R1 = 4700 to account for desired on/off switching times
ON Switching Time
8us * 1.7 = 14.4 us
OFF Switching Time
50us * 1.6 = 80us

Equations Set 2. Design Equations for the Optical Sensor Circuitry

Program Flowcharts

There are four major software sections that the system runs during operation. The first section of code is the LCD initialization and waiting for the user to start the pendulum. The second section is the initial motor pulse code that initializes the pendulum oscillation. The third software section is the code that gradually increases the angle of oscillation of the pendulum using motor pulses when the equilibrium sensor is crossed. The final software section is the oscillation stabilization code that maintains the pendulum's desired angle of oscillation.

The LCD initialization code clears the LCD and prompts the user to press the 'A' button of the keypad. The code sits in a dummy loop waiting for the external interrupt to be triggered by a user key-press. Once the correct button has been pressed, the LCD is changed to prompt the user to press the 'B' button, and the initial motor pulse code is called. The LCD initialization software flowchart is shown in Figure 15.

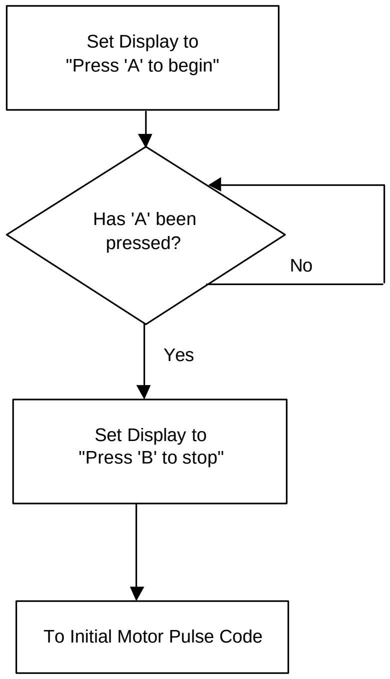

Set Display to "Press 'A' to begin"
Has 'A' been pressed?
No
Yes
Set Display to "Press 'B' to stop"
To Initial Motor Pulse Code

Figure 15. LCD User Prompt Software Flowchart

The initial motor pulse code provides burst to the motor at a constant pulse length and duty cycle. The purpose of this code is to initialize the movement of the pendulum. The constant pulse length works in initializing the movement of the pendulum because the first fifteen degrees of the pendulum's oscillation are linear. Once the pendulum has reached a constant oscillation, the code can switch to the increasing pulse length code. The software flowchart for the initial motor pulse code is shown in Figure 16.

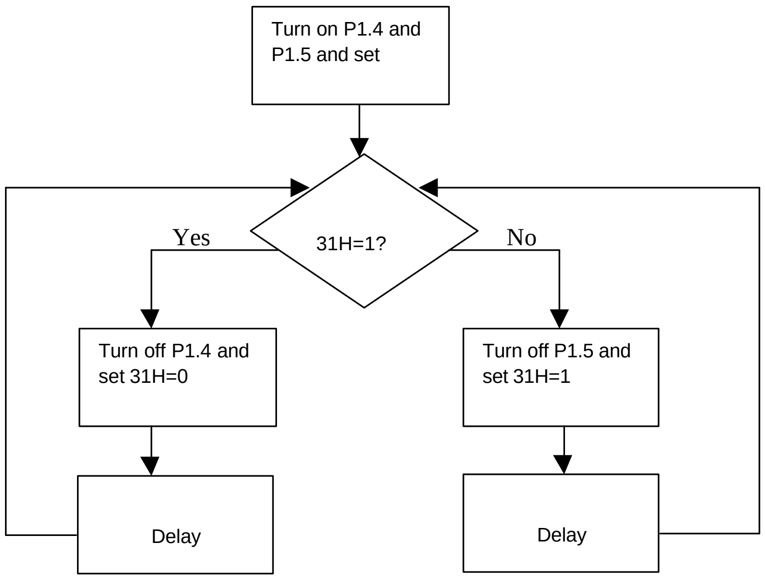

Turn on P1.4 and P1.5 and set
31H=1?
Yes
No
Turn off P1.4 and set 31H=0
Turn off P1.5 and set 31H=1
Delay
Delay

Figure 16. Initial Motor Pulse Software Flowchart

Once the initial motor pulse code loop has cycled for a set number of times, the pulse length will gradually increase. This allows for the angle of oscillation of the pendulum to increase to reach the desired final angle. The pulse length is increased by incrementing the delay before each motor pulse. This delay continues increasing until the angle sensor is reached. At this point, the oscillation stabilization code takes over. The increasing motor pulse software flowchart is shown in Figure 17.

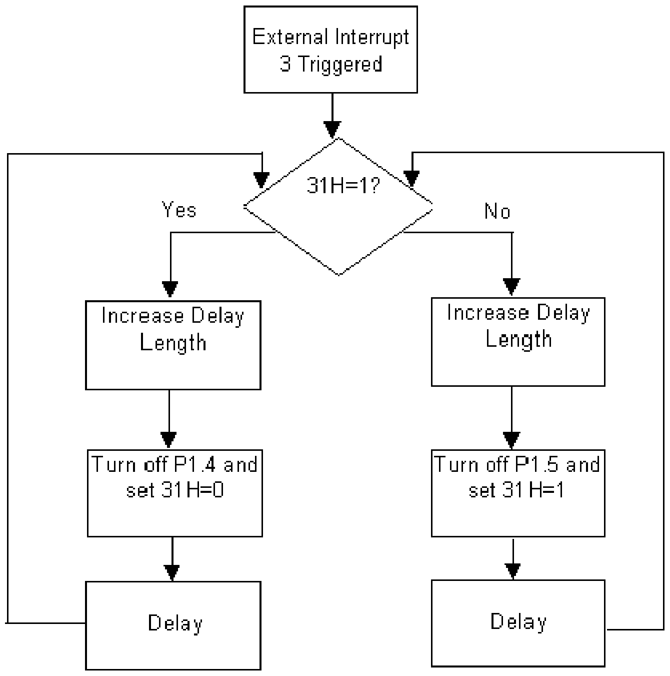

External Interrupt 3 Triggered
31H=1?
Yes
No
Increase Delay Length
Increase Delay Length
Turn off P1.4 and set 31H=0
Turn off P1.5 and set 31H=1
Delay
Delay

Figure 17. Increasing Pulse Length Software Flowchart

The oscillation stabilization code begins once the angle sensor is triggered. Once the angle sensor triggers the interrupt, the EMAC microcontroller development system begins counting the number of times that the sensor is triggered during each oscillation, so that it can compensate for any undershoot or overshoot by the pendulum. If the pendulum overshoots the desired angle, the sensor will be triggered twice. If this occurs, the pulse length will be shortened. If the sensor is not triggered, then the pendulum is not reaching the desired angle, so the pulse length must be increased. These checks are performed on each oscillation so that they EMAC microcontroller development system can keep the pendulum steady at the desired angle of oscillation. The oscillation stabilization software flowchart is shown in Figure 18.

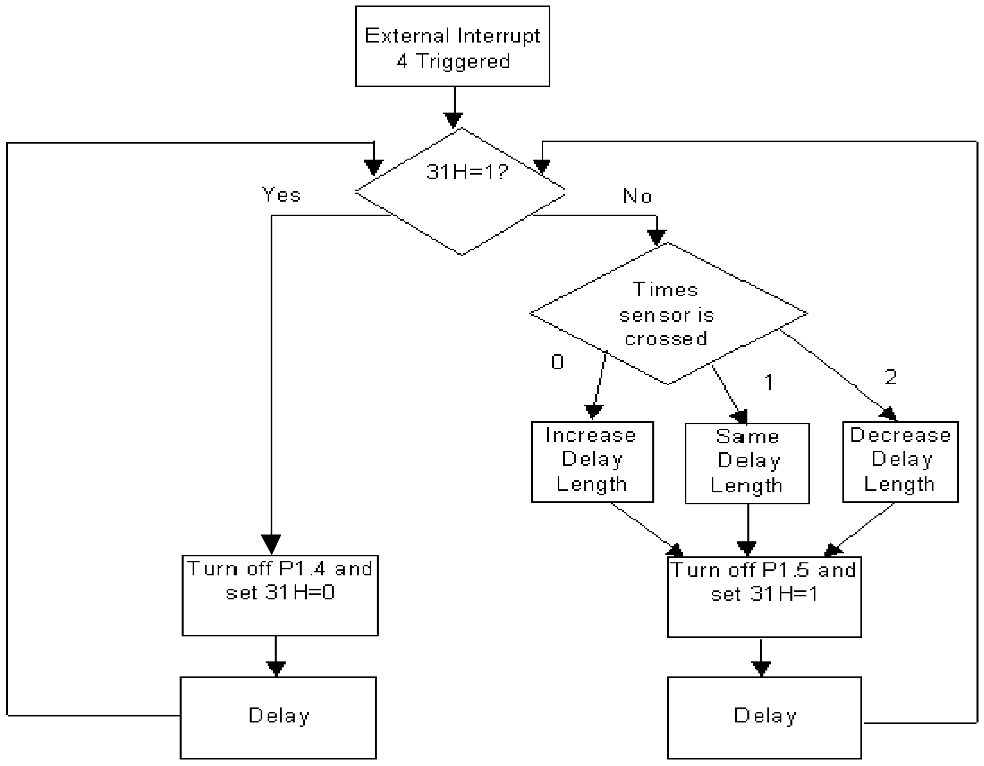


Figure 18. Oscillation Stabilization Software Flowchart

Software

All EMAC microcontroller development system code modules are provided on attached CD-ROM.

Accomplishments

- The H-bridge hardware was designed and built in lab and functioned correctly.
- The H-bridge switching code automatically switched the H-bridge direction at the correct time in the code.
- The EMAC development board to H-bridge interface switches allowed the EMAC microcontroller development system output to drive the H-bridge so that the motor could burst in both directions.
- The initial motor pulse software sent bursts of constant pulse length and duty cycle to initialize the oscillation of the pendulum.
- The increasing motor burst software gradually increased the angle of oscillation by gradually increasing the pulse length.
- The optical sensor hardware sent the correct 0-5V signal to the EMAC microcontroller development system to trigger the correct interrupts.

- The optical sensor interrupt software correctly sent a burst at equilibrium and correctly monitored the pendulum’s angle of oscillation.
- The oscillation stabilization software maintained the pendulum’s desired angle of oscillation and compensated for both undershoots and overshoots.
- The user interface software correctly prompted the user and handled the user input correctly.
- The pendulum unit was constructed and functioned as needed.

Demonstration of Accomplishments

Video of completed system running is provided on the attached CD-ROM.

Conclusions

The initial goal of this project was to use an EMAC microcontroller development system based system to control the oscillation of a pendulum that was overloading the motor. Over the course of the semester, both hardware and software systems were designed to achieve this goal. The final system was able to provide a clear user interface, initialize the oscillation of the weighted pendulum, oscillate the pendulum up to a desired angle of oscillation, and maintain that angle of oscillation until the user chooses to stop the system. The system was able to handle a varying desired angle of oscillation, and would work with slight load increases. Modifications to the initial motor burst code would be needed to allow the system to handle varying loads. Additional angle sensors could also be added to provide a choice to the user of which angle to use. Overall, the system operated as planned for in the initial development.

Data Sheets

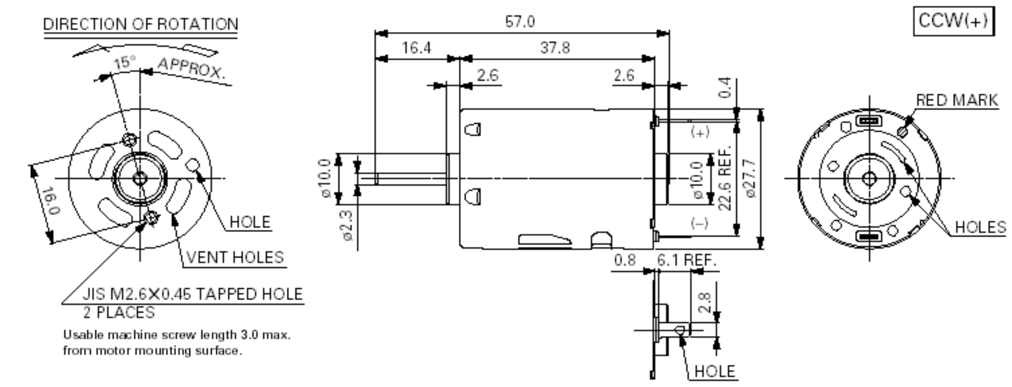
RS-385SH DC Motor

MABUCHI MOTOR **RS-385SH** OUTPUT: APPROX 0.9W~35W *Carbon-brush motors*

Typical Applications **Office Automation Equipment :** Printer / Copy Machine
Home Appliances : Hair Dryer

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
RS-385SH-2270	6 ~ 24	20V CONSTANT	16400	0.18	14010	1.06	9.56	97.5	14.0	65.7	670	6.20

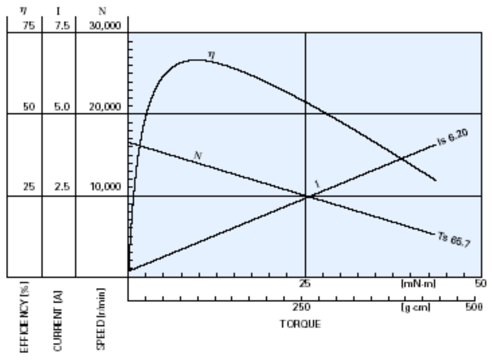
The terminal position against the tapped holes varies depending on CW+/NEUTRAL. UNIT: MILLIMETERS



Usable machine screw length 3.0 max. from motor mounting surface.

WEIGHT: 70g (APPROX)

RS-385SH-2270 20.0V

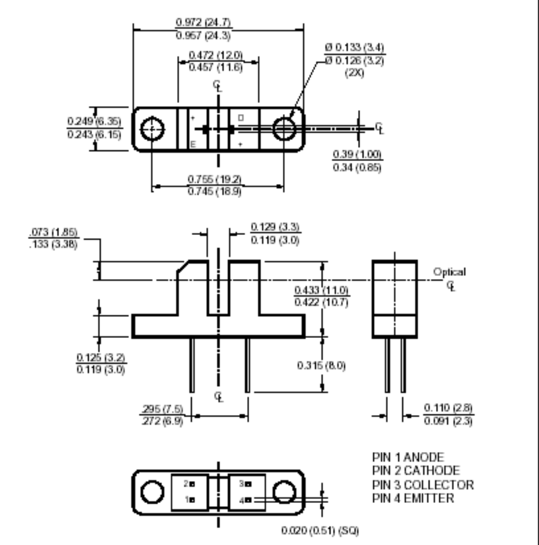


H21A1 Optical Sensor

FAIRCHILD SEMICONDUCTOR®

H21A1 / H21A2 / H21A3
PHOTOTRANSISTOR OPTICAL INTERRUPTER SWITCH

PACKAGE DIMENSIONS

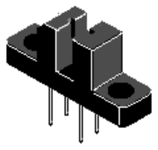


NOTES:

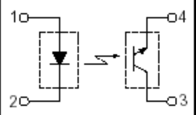
1. Dimensions for all drawings are in inches (mm).
2. Tolerance of ± .010 (.25) on all non-nominal dimensions unless otherwise specified.

DESCRIPTION

The H21A1, H21A2 and H21A3 consist of a gallium arsenide infrared emitting diode coupled with a silicon phototransistor in a plastic housing. The packaging system is designed to optimize the mechanical resolution, coupling efficiency, ambient light rejection, cost and reliability. The gap in the housing provides a means of interrupting the signal with an opaque material, switching the output from an "ON" to an "OFF" state.



SCHEMATIC



FEATURES

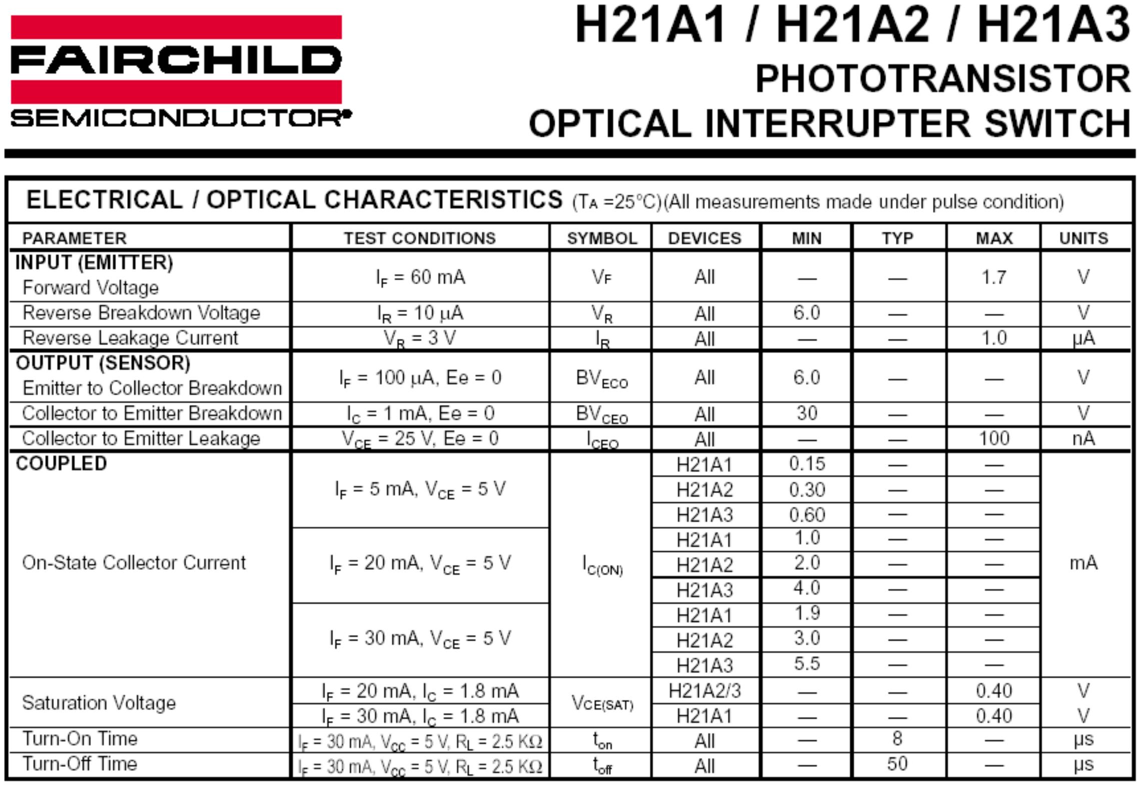
- Opaque housing
- Low cost
- .035" apertures
- High $I_{C(ON)}$

1. Derate power dissipation linearly 1.33 mW/°C above 25°C.
2. RMA flux is recommended.
3. Methanol or isopropyl alcohols are recommended as cleaning agents.
4. Soldering iron tip 1/16" (1.6mm) minimum from housing.

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise specified)

Parameter	Symbol	Rating	Unit
Operating Temperature	T_{OPR}	-55 to +100	°C
Storage Temperature	T_{STG}	-55 to +100	°C
Soldering Temperature (Iron)(2,3 and 4)	T_{SOL-I}	240 for 5 sec	°C
Soldering Temperature (Flow)(2 and 3)	T_{SOL-F}	260 for 10 sec	°C
INPUT (EMITTER) Continuous Forward Current	I_F	50	mA
Reverse Voltage	V_R	6	V
Power Dissipation (1)	P_D	100	mW
OUTPUT (SENSOR) Collector to Emitter Voltage	V_{CEO}	30	V
Emitter to Collector Voltage	V_{ECO}	4.5	V
Collector Current	I_C	20	mA
Power Dissipation (T_C = 25°C)(1)	P_D	150	mW

H21A1 Optical Sensor continued



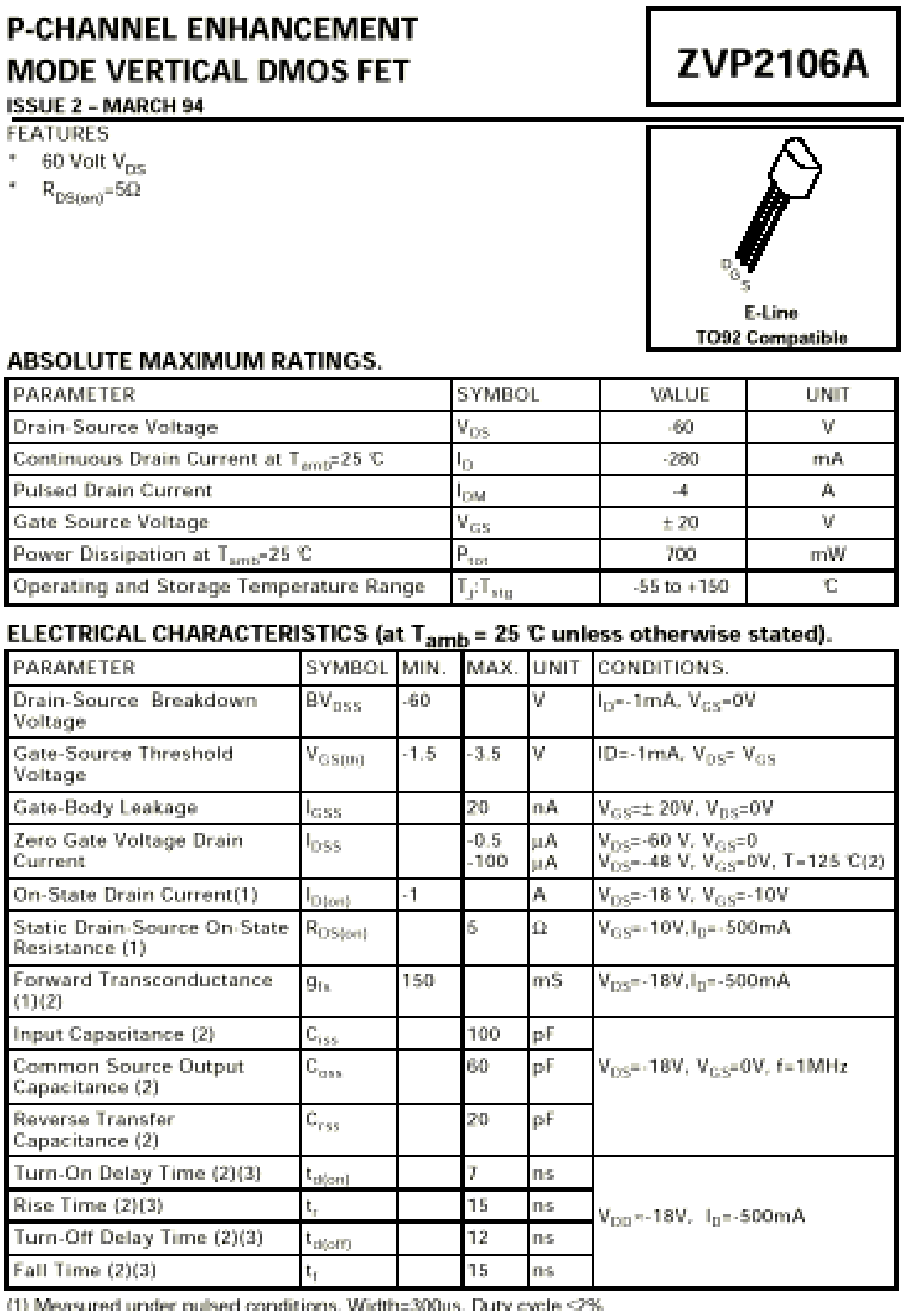
FAIRCHILD SEMICONDUCTOR®

H21A1 / H21A2 / H21A3
PHOTOTRANSISTOR
OPTICAL INTERRUPTER SWITCH

ELECTRICAL / OPTICAL CHARACTERISTICS (T_A =25°C)(All measurements made under pulse condition)

PARAMETER	TEST CONDITIONS	SYMBOL	DEVICES	MIN	TYP	MAX	UNITS
INPUT (EMITTER) Forward Voltage	I_F = 60 mA	V_F	All	—	—	1.7	V
Reverse Breakdown Voltage	I_R = 10 μA	V_R	All	6.0	—	—	V
Reverse Leakage Current	V_R = 3 V	I_R	All	—	—	1.0	μA
OUTPUT (SENSOR) Emitter to Collector Breakdown	I_F = 100 μA, Ee = 0	BV_{ECO}	All	6.0	—	—	V
Collector to Emitter Breakdown	I_C = 1 mA, Ee = 0	BV_{CEO}	All	30	—	—	V
Collector to Emitter Leakage	V_{CE} = 25 V, Ee = 0	I_{CEO}	All	—	—	100	nA
COUPLED On-State Collector Current	I_F = 5 mA, V_{CE} = 5 V	$I_{C(ON)}$	H21A1	0.15	—	—	mA
			H21A2	0.30	—	—	
			H21A3	0.60	—	—	
	I_F = 20 mA, V_{CE} = 5 V		H21A1	1.0	—	—	
			H21A2	2.0	—	—	
			H21A3	4.0	—	—	
	I_F = 30 mA, V_{CE} = 5 V		H21A1	1.9	—	—	
			H21A2	3.0	—	—	
			H21A3	5.5	—	—	
Saturation Voltage	I_F = 20 mA, I_C = 1.8 mA	$V_{CE(SAT)}$	H21A2/3	—	—	0.40	V
	I_F = 30 mA, I_C = 1.8 mA		H21A1	—	—	0.40	V
Turn-On Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{on}	All	—	8	—	μs
Turn-Off Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{off}	All	—	50	—	μs

ZVP2106a P-Channel Transistor



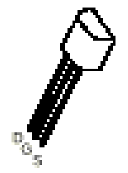
P-CHANNEL ENHANCEMENT MODE VERTICAL DMOS FET

ZVP2106A

ISSUE 2 – MARCH 94

FEATURES

- * 60 Volt V_{DS}
- * $R_{DS(on)}=5\Omega$



E-Line
TO92 Compatible

ABSOLUTE MAXIMUM RATINGS.

PARAMETER	SYMBOL	VALUE	UNIT
Drain-Source Voltage	V_{DS}	-60	V
Continuous Drain Current at T_{amb}=25 °C	I_D	-280	mA
Pulsed Drain Current	I_{DM}	-4	A
Gate Source Voltage	V_{GS}	± 20	V
Power Dissipation at T_{amb}=25 °C	P_{tot}	700	mW
Operating and Storage Temperature Range	T_j:T_{stg}	-55 to +150	°C

ELECTRICAL CHARACTERISTICS (at T_{amb} = 25 °C unless otherwise stated).

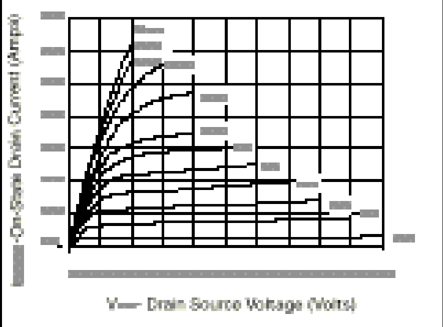
PARAMETER	SYMBOL	MIN.	MAX.	UNIT	CONDITIONS.
Drain-Source Breakdown Voltage	BV_{DSS}	-60		V	I_D=-1mA, V_{GS}=0V
Gate-Source Threshold Voltage	$V_{GS(th)}$	-1.5	-3.5	V	ID=-1mA, V_{DS}= V_{GS}
Gate-Body Leakage	I_{GSS}		20	nA	V_{GS}=± 20V, V_{DS}=0V
Zero Gate Voltage Drain Current	I_{DSS}		-0.5 -100	µA µA	V_{DS}=-60 V, V_{GS}=0 V_{DS}=-48 V, V_{GS}=0V, T=125 °C(2)
On-State Drain Current(1)	$I_{D(on)}$	-1		A	V_{DS}=-18 V, V_{GS}=-10V
Static Drain-Source On-State Resistance (1)	$R_{DS(on)}$		5	Ω	V_{GS}=-10V,I_D=-500mA
Forward Transconductance (1)(2)	g_{fs}	150		mS	V_{DS}=-18V,I_D=-500mA
Input Capacitance (2)	C_{iss}		100	pF	V_{DS}=-18V, V_{GS}=0V, f=1MHz
Common Source Output Capacitance (2)	C_{oss}		60	pF	
Reverse Transfer Capacitance (2)	C_{rss}		20	pF	
Turn-On Delay Time (2)(3)	$t_{d(on)}$		7	ns	V_{DD}=-18V, I_D=-500mA
Rise Time (2)(3)	t_r		15	ns	
Turn-Off Delay Time (2)(3)	$t_{d(off)}$		12	ns	
Fall Time (2)(3)	t_f		15	ns	

(1) Measured under pulsed conditions. Width=300µs. Duty cycle ≤2%

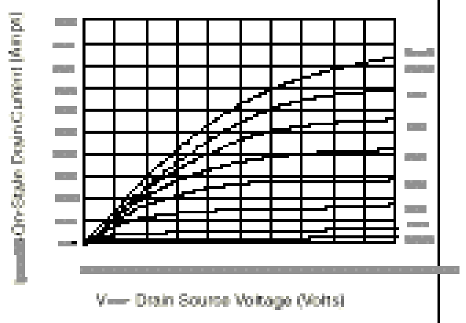
ZVP2106a continued

ZVP2106A

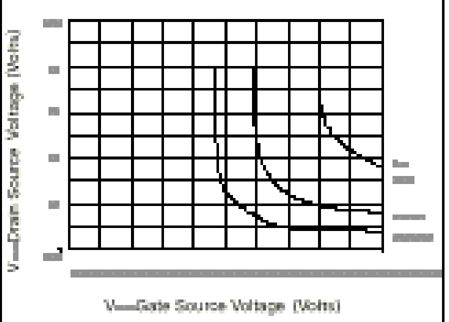
TYPICAL CHARACTERISTICS



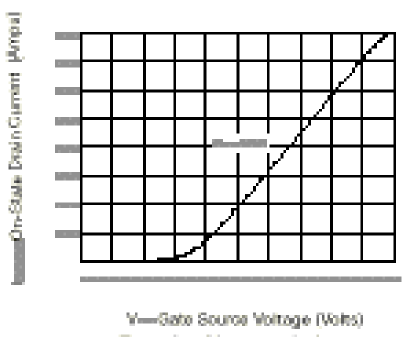
Output Characteristics



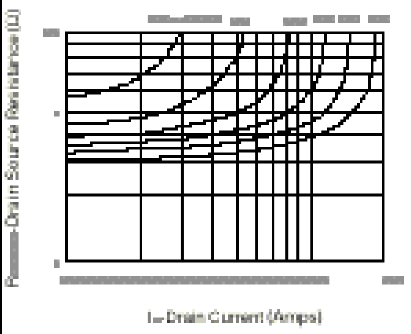
Saturation Characteristics



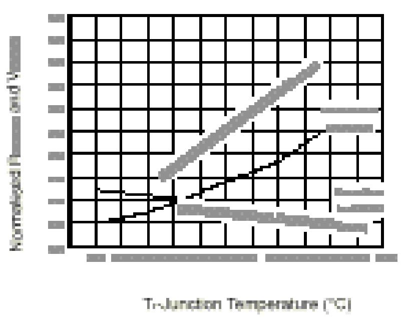
Voltage Saturation Characteristics



Transfer Characteristics



On-resistance v drain current



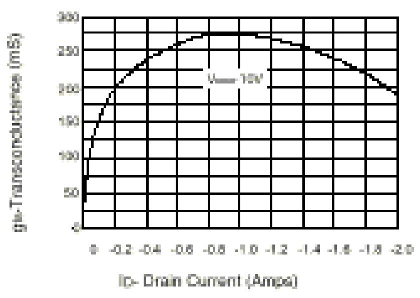
Normalised $R_{DS(on)}$ and $V_{GS(th)}$ vs Temperature

3-418

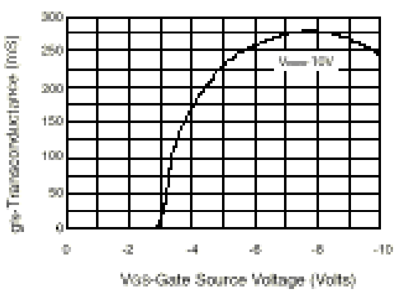
ZVP2106a continued

ZVP2106A

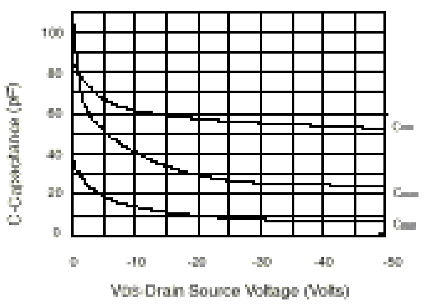
TYPICAL CHARACTERISTICS



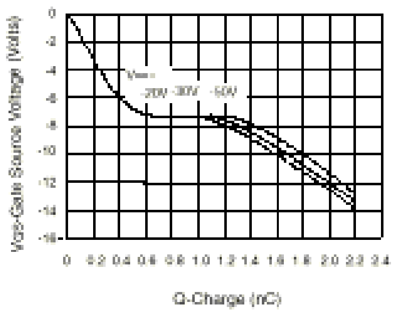
Transconductance v drain current



Transconductance v gate-source voltage

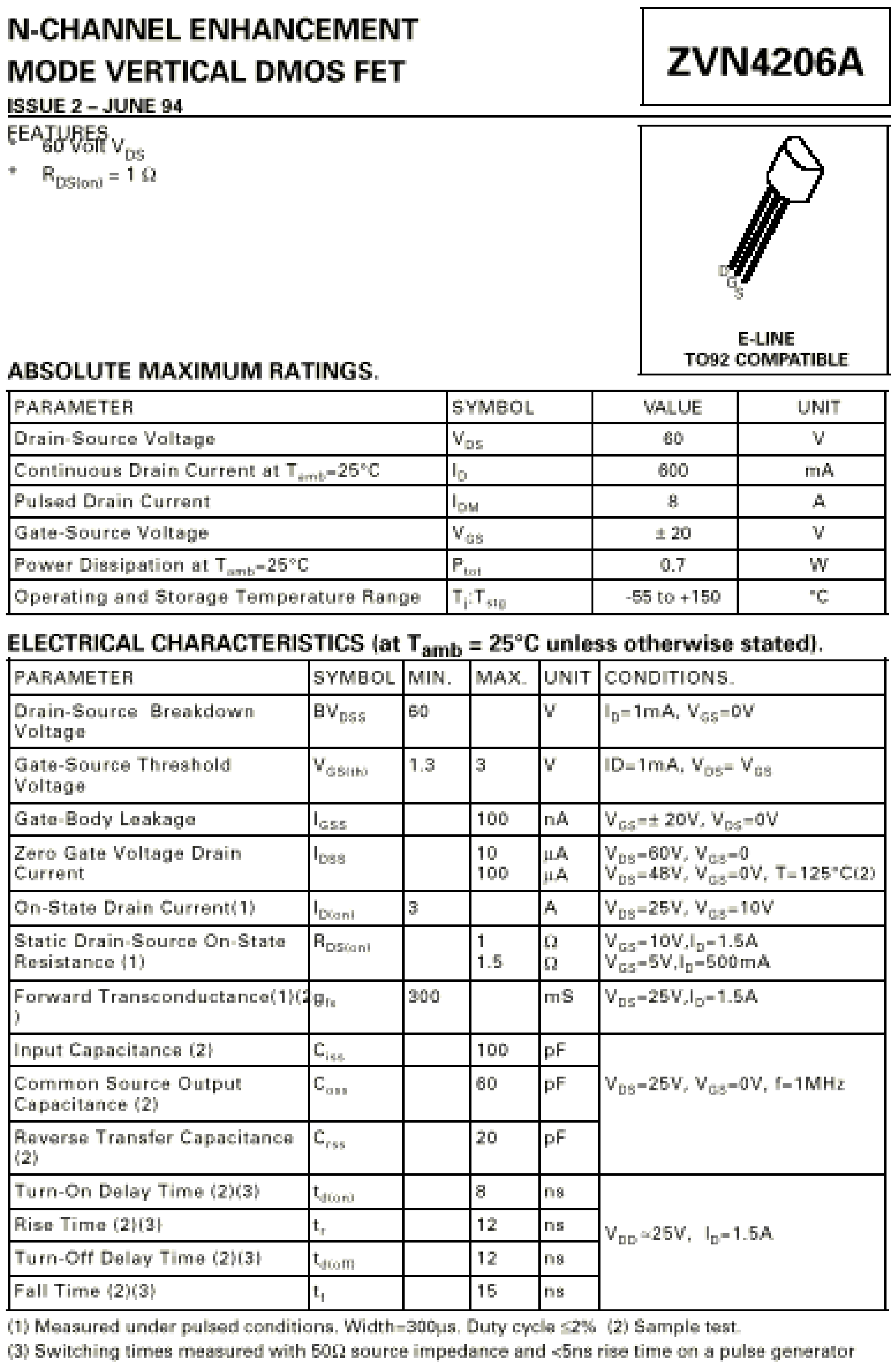


Capacitance v drain-source voltage



Gate charge v gate-source voltage

ZVN4206a N-Channel Transistor



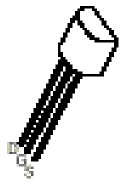
N-CHANNEL ENHANCEMENT MODE VERTICAL DMOS FET

ISSUE 2 – JUNE 94

ZVN4206A

FEATURES

* 60 Volt V_{DS}
* $R_{DS(on)} = 1\ \Omega$



E-LINE
TO92 COMPATIBLE

ABSOLUTE MAXIMUM RATINGS.

PARAMETER	SYMBOL	VALUE	UNIT
Drain-Source Voltage	V_{DS}	60	V
Continuous Drain Current at T_{amb}=25°C	I_D	600	mA
Pulsed Drain Current	I_{DM}	8	A
Gate-Source Voltage	V_{GS}	± 20	V
Power Dissipation at T_{amb}=25°C	P_{tot}	0.7	W
Operating and Storage Temperature Range	T_j:T_{stg}	-55 to +150	°C

ELECTRICAL CHARACTERISTICS (at T_{amb} = 25°C unless otherwise stated).

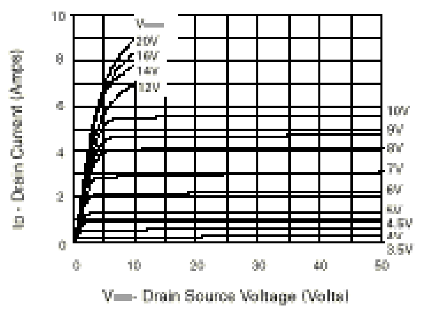
PARAMETER	SYMBOL	MIN.	MAX.	UNIT	CONDITIONS.
Drain-Source Breakdown Voltage	BV_{DSS}	60		V	I_D=1mA, V_{GS}=0V
Gate-Source Threshold Voltage	$V_{GS(th)}$	1.3	3	V	ID=1mA, V_{DS}= V_{GS}
Gate-Body Leakage	I_{GSS}		100	nA	V_{GS}=± 20V, V_{DS}=0V
Zero Gate Voltage Drain Current	I_{DSS}		10 100	µA µA	V_{DS}=60V, V_{GS}=0 V_{DS}=48V, V_{GS}=0V, T=125°C(2)
On-State Drain Current(1)	$I_{D(on)}$	3		A	V_{DS}=25V, V_{GS}=10V
Static Drain-Source On-State Resistance (1)	$R_{DS(on)}$		1 1.5	Ω Ω	V_{GS}=10V,I_D=1.5A V_{GS}=5V,I_D=500mA
Forward Transconductance(1)(2)	g_{fs}	300		mS	V_{DS}=25V,I_D=1.5A
Input Capacitance (2)	C_{iss}		100	pF	V_{DS}=25V, V_{GS}=0V, f=1MHz
Common Source Output Capacitance (2)	C_{oss}		60	pF	
Reverse Transfer Capacitance (2)	C_{rss}		20	pF	
Turn-On Delay Time (2)(3)	$t_{d(on)}$		8	ns	V_{DD} ≈25V, I_D=1.5A
Rise Time (2)(3)	t_r		12	ns	
Turn-Off Delay Time (2)(3)	$t_{d(off)}$		12	ns	
Fall Time (2)(3)	t_f		15	ns	

(1) Measured under pulsed conditions. Width=300µs. Duty cycle ≤2% (2) Sample test.
(3) Switching times measured with 50Ω source impedance and <5ns rise time on a pulse generator

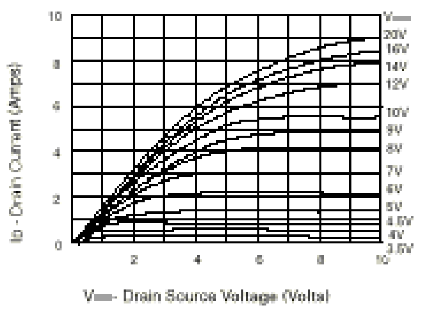
ZVN4206a continued

ZVN4206A

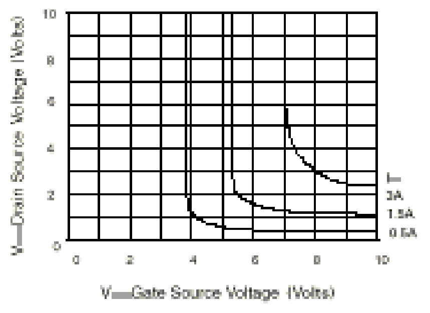
TYPICAL CHARACTERISTICS



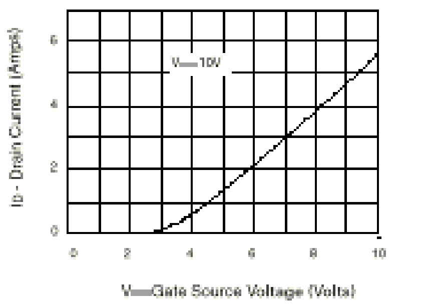
Output Characteristics



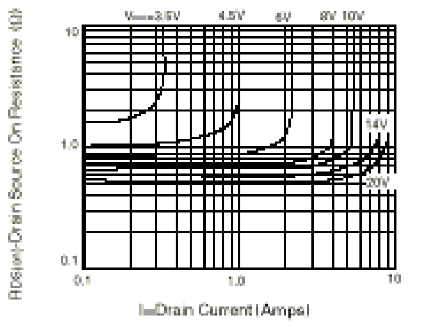
Saturation Characteristics



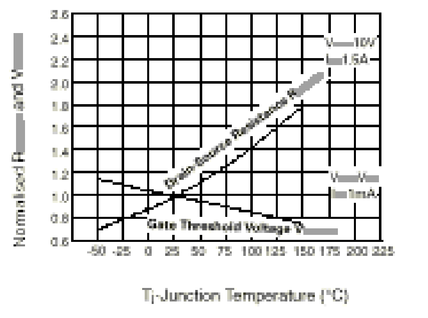
Voltage Saturation Characteristics



Transfer Characteristics



On-resistance v drain current



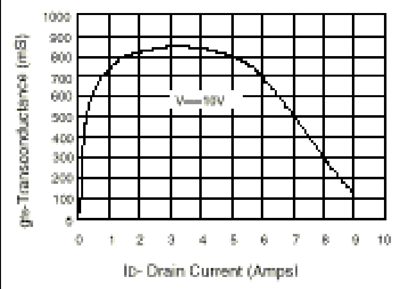
Normalised $R_{DS(on)}$ and $V_{GS(th)}$ v Temperature

3-382

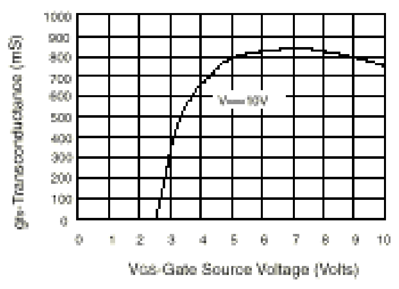
ZVN4206a continued

ZVN4206A

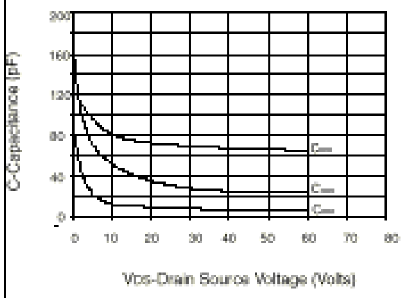
TYPICAL CHARACTERISTICS



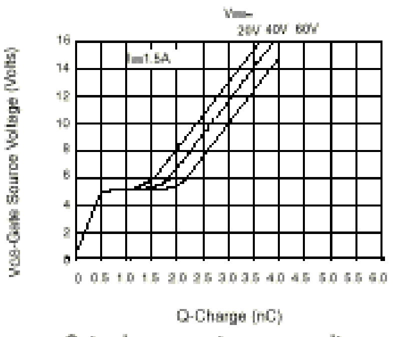
Transconductance v drain current



Transconductance v gate-source voltage



Capacitance v drain-source voltage



Gate charge v gate-source voltage

Appendix B

```
$NOMOD51                                ;disable predefined 8051 registers
$INCLUDE(reg515.inc)

NAME                    INIT_DISP
PUBLIC                  st_lcd, loop, kbdtbl, pro2

EXTRN CODE              (lcdinit,charout,cmdout,initb,in1srv,int3hnd,int4hnd)

        org     8000h
        ljmp    st_lcd

        org     8013h           ;External interrupt 1 (IE1).
        ljmp    in1srv

        org     8053H
        ljmp    int3hnd

        org     805BH
        ljmp    int4hnd


        INIT_DISP     SEGMENT     CODE ; reserve RAM space for st_lcd code
segment

        RSEG            INIT_DISP       ; places st_lcd code segements at
                                ; this point in assembled code

        USING                   0       ; indicates to the assembler that register
                                ; bank 0 will be used, but does not select
                                ; register bank 0

st_lcd:

        lcall   lcdinit

      mov    a,#lcd_clr     ;clear display
      lcall  cmdout
      mov     a,#80h
      lcall  cmdout
      mov       dptr,#pro1
      sjmp      loop
loop:
      clr    a
clr                 ie1
        movc    a,@a+dptr       ;get next character
        jz      loopexit              ;done if null character
        lcall   charout         ;send it out
        inc     dptr            ;point to next character
        sjmp    loop            ;repeat

loopexit:
            ;setb    it1
            ;clr     ex1
            mov     dptr,#kbdtbl
            lcall   set1

set1:
            mov     a,#0C0h
    lcall   cmdout
    clr     ie1
    clr     c
    sjmp    main3
            ;ljmp    initb

main3:
            setb    EAL             ;enable interrupts to be enabled
            setb    EX1             ;enable external interrupt 1
            clr             it1                     ;enable falling edge triggered
            ORL             IP1,#04H
            ORL             IP0,#02H
            ljmp    initb

lcd_clr         equu    01h             ;clear LCD command
lcdcmd          equ     28h             ;value for P2 to select lcd command port
lcddat  equ     29h             ;value for P2 to select lcd data port
kdpt            equ     38h             ;keypad port

pro1:
                db      "Press 'A' To Begin",0
                ret
pro2:
                db      "Press 'B' To Stop",0
                ret

kbdtbl:         db      '123C456D789EA0BF',0

                end
```

```
$NOMOD51                                ;disable predefined 8051 registers
$INCLUDE(reg515.inc)

NAME                    seconddisp
PUBLIC                  st_lcd2

EXTRN CODE              (lcdinit,charout,cmdout,pro2,kbdtbl,bridge)



     seconddisp    SEGMENT    CODE ; reserve RAM space for st_lcd code
segment

     RSEG          seconddisp      ; places st_lcd code segements at
                        ; this point in assembled code

     USING              0          ; indicates to the assembler that register
                        ; bank 0 will be used, but does not select
               ; register bank 0

st_lcd2:

     lcall   lcdinit

   mov    a,#lcd_clr     ;clear display
   lcall  cmdout
   mov    a,#80h
   lcall  cmdout
   mov       dptr,#pro2
   sjmp      loop2

loop2:
   clr    a
   clr               ie1
   movc   a,@a+dptr      ;get next character
   jz     loopexit2        ;done if null character
   lcall  charout       ;send it out
   inc    dptr          ;point to next character
   sjmp   loop2          ;repeat

loopexit2:
     ;setb   it1
```

```
;clr    ex1
        mov     dptr,#kbdtbl
        lcall   set2

set2:
        mov     a,#0C0h
   lcall   cmdout
   clr    ie1
   clr    c
   sjmp main4

main4:
        ;setb   EAL             ;enable interrupts to be enabled
        ;setb   EX4
        ljmp    bridge

lcd_clr         equu    01h             ;clear LCD command
lcdcmd          equ     28h             ;value for P2 to select lcd command port
lcddat  equ     29h             ;value for P2 to select lcd data port
kdpt            equ     38h             ;keypad port


                end
```

```
$NOMOD51
$INCLUDE(reg515.inc)


NAME            hbridge
PUBLIC                  bridge, initb
EXTRN CODE                      (delay, bridge1,st_lcd2,stop);,main)



hbridge         SEGMENT CODE
                RSEG hbridge

lcd_clr         equu    01H            ;clear LCD command
lcdcmd          equ     28h           ;value for P2 to select lcd command port
lcddat  equ     29h           ;value for P2 to select lcd data port
kdpt            equ             38H                             ;keypad         port

initb:
        mov     31h,#1               ;initialize memory location 31H to 1
   mov 32h,#4
        mov     29h,#0
        clr     p1.4
   cpl  p1.4
   clr  p1.5            ;set both outputs A and B to 5V
        cpl     p1.5
        mov     R6,#0Fh
        mov     41h,#14h

loopstart:
        mov     a,41h
        cjne    a,#12h,loopstart
        mov     41h,#14h
        ljmp    st_lcd2

;enter:
;       mov     a,#0
;       jnb     ie1,enter
;       mov     P2,#kdpt
;       movx    a,@r1
;       cjne    a,#0Ch,enter
;       clr     ie1
;       mov     P2,#0
;   sjmp        st_lcd2
```

```
bridge:
        dec     32h
        mov     P2,#kdpt
        movx    a,@r1
        anl     a,#0Fh
        cjne    a,#0Eh,bridger
        ljmp    stop


bridger:
        mov     a,31h
        cjne    a,#1,forward    ;jump to forward oscillation if 31H=0
        cjne    a,#0,backward   ;jump to backward oscillation if 31H=1

forward:
        clr     p1.4            ;Turn off P1.4 for forward oscillation
        mov     31h,#1          ;set memory location 31H to 1
        lcall delay
        cpl p1.4        ;set P1.4 to high
        lcall delay
        mov     a,32h
        cjne    a,#0,bridge    ;repeat bridge code
        mov     32h,#4
        ljmp    bridge1

backward:
        clr     p1.5            ;Turn off P1.5 for backward oscillation
        mov     31h,#0          ;set memory location 31H to 0
        lcall delay
        cpl p1.5                ;set p1.5 to high
        lcall delay
        mov     a,32h
        cjne    a,#0,bridge    ;repeat bridge code
        mov     32h,#4
        ljmp    bridge1
end
```

```
$NOMOD51            ; omit assembler micro definitions
$Include(reg515.inc)  ; define 515 micro


                Name   delay
                PUBLIC delay


dly     SEGMENT CODE
                RSEG dly


                secs SET R4                  ;use R4 as delay loop count
                cnt SET R2
delay:
                        ;mov TMOD,#01                ;initialize timer 0
                mov     a,TMOD
                        anl     a,#0F0h
                        orl     a,#1
                        mov     TMOD,a
                        clr ET0         ;disable timer 0 interrupt

sec_cnt:        mov R5,#0Fh
redo:           mov cnt,#50

loop:           clr TF0                      ;clear timer 0 overflow flag


                        mov th0,#0ffh    ;load timer registers to
        mov tl0,#047h      ;overflow in 0.2ms=5000Hz
                    ;this loop must be run 20 times to equal 1 second
                        setb TR0                        ;timer zero run

wait:           jnb TF0, wait
                        djnz cnt,loop            ;run loop 250 times
                        djnz R5,redo       ;run the 250 times loop 20 times for 1 sec. delay
                        clr TR0                                 ;stop timer 0
                        ret
                        end
```

```
$NOMOD51
$INCLUDE(reg515.inc)


NAME            pulse
PUBLIC                  bridge1
EXTRN CODE                      (delay1,delay,main2,stop)

pulse   SEGMENT CODE
                RSEG pulse

kdpt            equu            38H                             ;keypad         port

bridge1:
        inc R6
        mov     P2,#kdpt
        movx    a,@r1
        anl     a,#0Fh
        cjne    a,#0Eh,bridger1
        ljmp    stop


bridger1:
        mov     a,31h
        cjne    a,#1,forward1   ;jump to forward oscillation if 31H=0
        cjne    a,#0,backward1          ;jump to backward oscillation if 31H=1


forward1:
        clr     p1.4            ;Turn off P1.4 for forward oscillation
        mov     31h,#1      ;set memory location 31H to 1
        lcall delay1
        cpl p1.4        ;set P1.4 to high
        lcall delay
        mov     a,32h
        cjne    a,#0,jump1
        lcall delay
        ljmp    main2

jump1:
        dec     32h
        sjmp bridge1
```

```
backward1:
        clr     p1.5            ;Turn off P1.5 for backward oscillation
        mov     31h,#0      ;set memory location 31H to 0
        lcall delay1
        cpl p1.5                ;set p1.5 to high
   lcall delay
   mov a,32h
        cjne    a,#0,jump2
        lcall delay
        ljmp    main2

jump2:
        dec 32h
        sjmp bridge1

end
```

```
$NOMOD51            ; omit assembler micro definitions
$Include(reg515.inc)  ; define 515 micro


                Name   delay1
                PUBLIC delay1


dly1    SEGMENT CODE
                RSEG dly1


                secs1 SET R4                ;use R4 as delay loop count
                cnt1 SET R2
delay1:
                        ;mov TMOD,#01               ;initialize timer 0
                mov     a,TMOD
                        anl     a,#0F0h
                        orl     a,#1
                        mov     TMOD,a
                        clr ET0         ;disable timer 0 interrupt
                   mov a,R6

sec_cnt1:       mov R5,a
redo1:          mov cnt1,#50

loop1:          clr TF0                     ;clear timer 0 overflow flag


                        mov th0,#0ffh    ;load timer registers to
        mov tl0,#047h       ;overflow in 0.2ms=5000Hz
                   ;this loop must be run 20 times to equal 1 second
                        setb TR0                    ;timer zero run

wait1:          jnb TF0, wait1
                        djnz cnt1,loop1             ;run loop 250 times
                        djnz R5,redo1       ;run the 250 times loop 20 times for 1 sec. delay
                        clr TR0                             ;stop timer 0
                        ret
                        end
```

```
$NOMOD51
$INCLUDE(reg515.inc)

NAME            pulse2
PUBLIC                  bridge2
EXTRN CODE                      (delay2,stop)

pulse2  SEGMENT CODE
                RSEG pulse2

kdpt            equu            38H                             ;keypad         port

bridge2:
        mov     a,31h
        cjne    a,#1,forward2  ;jump to forward oscillation if 31H=0
        cjne    a,#0,backward2          ;jump to backward oscillation if 31H=1


forward2:
        clr     p1.4            ;Turn off P1.4 for forward oscillation
        mov     31h,#1      ;set memory location 31H to 1
        lcall delay2
        cpl p1.4        ;set P1.4 to high
        ret


backward2:
        mov     a,29h
        cjne    a,#1,decr
        mov     29h,#0
        jmp     bw2

decr:
        cjne    a,#2,incr
        dec     R6
        mov     29h,#0
        jmp     bw2

incr:
        inc     R6
        mov     29h,#0

bw2:
        clr     p1.5            ;Turn off P1.5 for backward oscillation
```

```
mov	31h,#0	;set memory location 31H to 0
	lcall delay2
	cpl p1.5			;set p1.5 to high
	ret

end
```

```
$NOMOD51            ; omit assembler micro definitions
$Include(reg515.inc)  ; define 515 micro


                Name   delay2
                PUBLIC delay2


dly2    SEGMENT CODE
                RSEG dly2


                secs2 SET R4                ;use R4 as delay loop count
                cnt2 SET R2
delay2:
                        ;mov TMOD,#01              ;initialize timer 0
                mov     a,TMOD
                        anl     a,#0F0h
                        orl     a,#1
                        mov     TMOD,a
                        clr ET0         ;disable timer 0 interrupt
                        mov     a,R6

sec_cnt2:       mov R5,a
redo2:          mov cnt2,#5

loop2:          clr TF0                     ;clear timer 0 overflow flag


                        mov th0,#0ffh     ;load timer registers to
        mov tl0,#047h      ;overflow in 0.2ms=5000Hz
                   ;this loop must be run 20 times to equal 1 second
                        setb TR0                   ;timer zero run

wait2:          jnb TF0, wait2
                        djnz cnt2,loop2            ;run loop 250 times
                        djnz R5,redo2     ;run the 250 times loop 20 times for 1 sec. delay
                        clr TR0                            ;stop timer 0
                        ret
                        end
```

```
$NOMOD51   ; disable predefined 8051 registers
$INCLUDE(reg515.inc)

NAME jump1
PUBLIC       main2,int3hnd

EXTRN CODE (bridge2,stop)

jump1 SEGMENT CODE
RSEG jump1

kdpt        equu        38H                         ;keypad      port

main2:
     ;intcnt    set R4  ;interrupt 0 count
        ;mov    intcnt,#1
        setb    I3FR          ;enable positive edge triggered
        ;setb   EAL           ;enable interrupts to be enabled
  setb  EX3           ;enable external interrupt 3
        ;setb   EAL           ;enable interrupts to be enabled
        setb    EX4
        mov     R6,#7Dh

measure:
        mov     P2,#kdpt
        movx    a,@r1
        anl     a,#0Fh
        cjne    a,#0Eh,bridger2
        ljmp    stop

bridger2:
        jmp     measure              ;loop until interrupt detected
     ;mov       A,intcnt
        ;jnz    measure
        ;mov    intcnt,#01h
        ;jmp    measure

int3hnd:
        lcall   bridge2       ;jump to code to output pulse to motor

check:
        reti

end
```

```
$NOMOD51   ; disable predefined 8051 registers
$INCLUDE(reg515.inc)

NAME	int4hand
PUBLIC		int4hnd



int4hand SEGMENT CODE
RSEG int4hand


int4hnd:
		mov	a,29h
		;add	a,#1
		inc	a
		mov	29h,a
		reti

end
```

```
$NOMOD51   ; disable predefined 8051 registers
$INCLUDE(reg515.inc)

NAME keypress
PUBLIC          in1srv

EXTRN CODE (st_lcd,st_lcd2)

keypress SEGMENT CODE
RSEG keypress

kdpt            equ             38H                             ;keypad         port




in1srv:
        mov     P2,#kdpt
        movx    a,@r1
        anl     a,#0Fh
        cjne    a,#0Ch,returner
        mov     41h,#12h
        mov     a,41h
        reti


returner:
        reti


end
```

```
$NOMOD51   ; disable predefined 8051 registers
$INCLUDE(reg515.inc)

NAME     stopper
PUBLIC          stop

EXTRN CODE (st_lcd)

stopper SEGMENT CODE
RSEG stopper


stop:
        clr     EX1
        setb    it1
        clr     EX3
        clr     EX4
        clr     ie1
        clr     p1.4
   cpl  p1.4
   clr  p1.5            ;set both outputs A and B to 5V
        cpl     p1.5
        ljmp    st_lcd

end
```

```
$NOMOD51                                  ;disable predefined 8051 registers
$INCLUDE(reg515.inc)

NAME                                lcd
PUBLIC                              lcdinit,charout,cmdout,dlaya
;EXTRN          CODE            ()

lcd_clr         equ     01H             ;clear LCD command
lcdcmd          equ     28h             ;value for P2 to select lcd command port
lcddat  equ     29h             ;value for P2 to select lcd data port
kdpt            equ             38H                             ;keypad         port

LCD_CODE    SEGMENT     CODE     ; reserve RAM space for lcdinit code segment

        RSEG  LCD_CODE              ; places lcdinit code segements at
                    ; this point in assembled code

                USING           0         ; indicates to the assembler that register
                        ; bank 0 will be used, but does not select
                        ; register bank 0


initdata:
        db     38h,06h,0Ch,01h,0
;38h = Function Set - 8 bit interface, 2 line display, 5X7 display
;06h = Entry Mode Set - increment address
;0Fh = Display ON/OFF Control - on, no cursor, no blink
;01h = Clear Display

lcdinit:

                mov             p2,#lcdcmd
                lcall     dlaya
                lcall     dlaya
        lcall         dlaya
        lcall         dlaya
        mov                     a,#30h
        movx        @r1,a
        lcall         dlaya
        movx        @r1,a
        lcall         dlaya
        mov                     dptr,#initdata

        mov         b,#80
```

```
lcdnit2:
        movx    a,@r1           ; read lcd command port
        jnb     acc.7,lcdnit1   ; exit if not busy
        djnz    b,lcdnit2       ; loop till timeout
        sjmp    lcdexit         ; exit if timeout

lcdnit1:
        movx    a,@r1           ; read lcd command port
        jb      acc.7,lcdnit1   ; loop if busy flag set
        clr     a
        movc    a,@a+dptr       ; get byte from init table
        jz      lcdexit         ; exit if 0
        inc     dptr            ; point to next byte
        movx    @r1,a           ; output byte
        sjmp    lcdnit1         ; loop

lcdexit:
        ret

;
; 5 ms delay
;
dlaya:  push    acc
        mov     a,#10

dlaya1:
        push    acc
        mov     a,#0ffh
        djnz    acc,$           ; inner loop
        pop     acc
        djnz    acc,dlaya1      ; outer loop

        pop     acc
        ret


;output a character

charout:
        mov     r2,a            ; save char in r2
        mov     p2,#lcdcmd      ; point to command port

reg0out:
        movx    a,@r1           ; read lcd command port
        ;clr        acc.7
        jb      acc.7,reg0out     ; loop if busy flag set
        mov     a,r2              ; restore char
        mov     p2,#lcddat
        movx    @r1,a             ; output it
        ret


;output a command

cmdout:
        mov     r2,a              ; save command char in r2
        mov     p2,#lcdcmd        ; point to command port


reg1out:
        movx    a,@r1             ; read lcd command port
        ;clr        acc.7
        jb      acc.7,reg1out     ; loop if busy flag set
        mov     a,r2              ; restore char
        movx    @r1,a             ; output it


            ret             ; jumps back to the next executable line after
            END
```

Functional Description: Motor Control of an Oscillating Pendulum

Objective: To design and build a motor controlled oscillating stiff arm pendulum that will be capable of initiation and termination of the oscillation based upon the user command, as well as achieve various heights and speeds to be selected by the user.

Description: We have decided to design and build a motor controlled oscillating pendulum. The initial design will include only one height, however, if time permits the user will be allowed to select the swinging height of the pendulum which will be dependent upon the number of sensors. The user will also be allowed to select the start and the termination of the oscillation system. The general block diagram for the system can be seen in Fig 1. The pendulum will be controlled by short bursts from a DC motor. We will design the burst of the motor to occur when the pendulum swings past the equilibrium point by placing a sensor at our equilibrium point to act as the burst sensor. Once the pendulum has reached the predefined height (through feedback from the sensor at that predefined height) the pendulum will continue to oscillate at that frequency until the user selects the option to terminate the oscillation, at which time, the pendulum will slowly return to equilibrium. Our initial design will have a predefined height with only one sensor. If time permits we will place several sensors along the oscillation of the pendulum. A 12 Volt DC motor will be used to generate the oscillation of the pendulum. However the pendulum will be selected in order to place strain upon the DC motor. For example, if the motor has a maximum torque of 20 oz/in, then a pendulum will be selected that will require 25oz/in. Also, the motor will be under-powered with respect to the power supply. Circuitry will have to be designed in order to store power between the motor bursts. A more detailed full system block diagram can be seen in Fig 2. The software will be used to send the signals to the motor in order to control the bursts according to the input of the user and the feedback of the sensors. A software flow chart can be seen in Fig 3. In addition to the single stiff arm pendulum, we will place a second stiff arm pendulum that will hang from the first in order to place a greater load on the DC motor as well as complicate the control of the system.

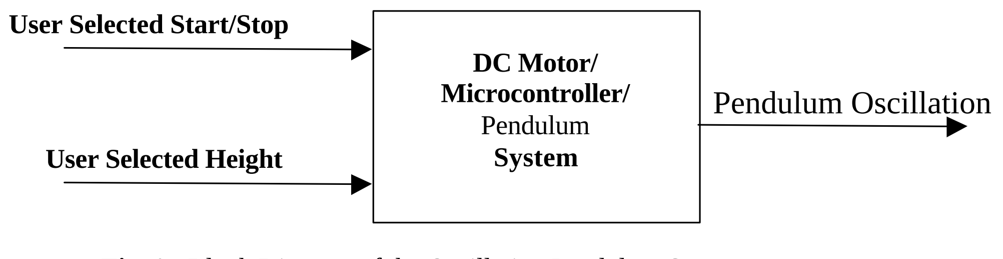


Fig. 1: Block Diagram of the Oscillating Pendulum System

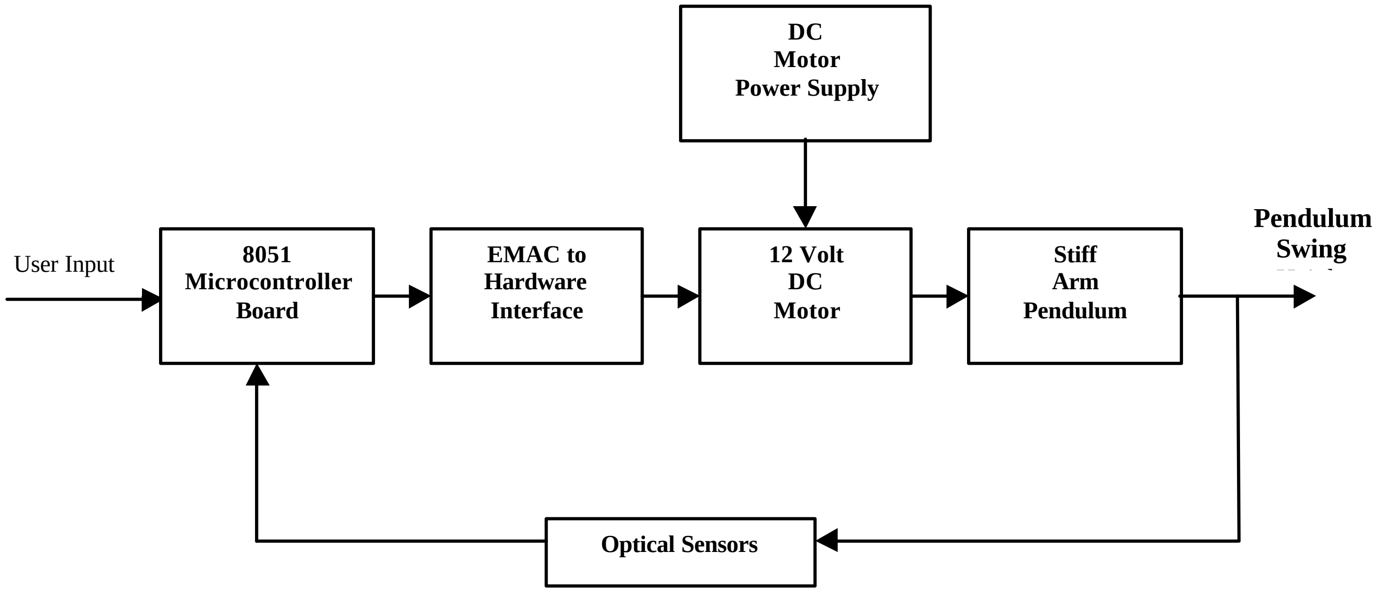


Fig. 2: Full System Block Diagram

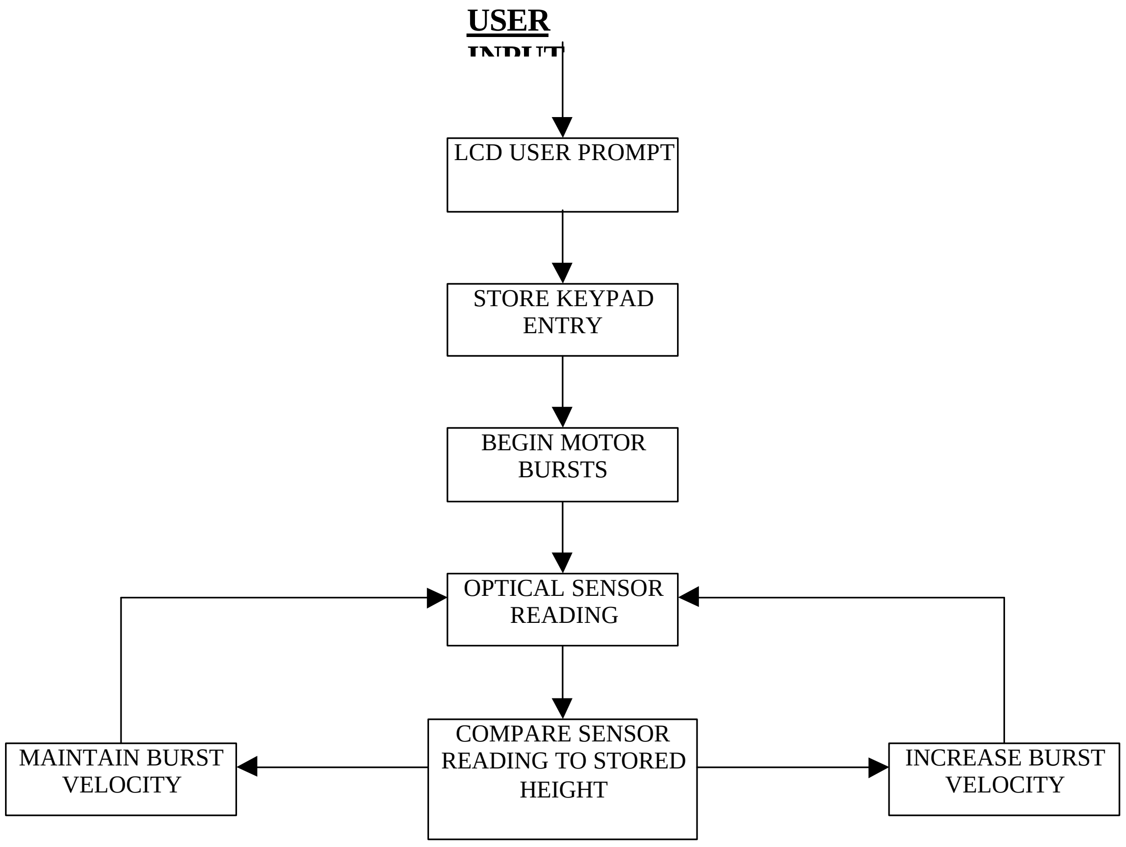


Fig. 3: Software System Flow Chart

Motor Controlled Pendulum Arm

Objective: To design and build a motor controlled oscillating pendulum that will be able to start and stop at a user command, as well as achieve various heights as selected by the user.

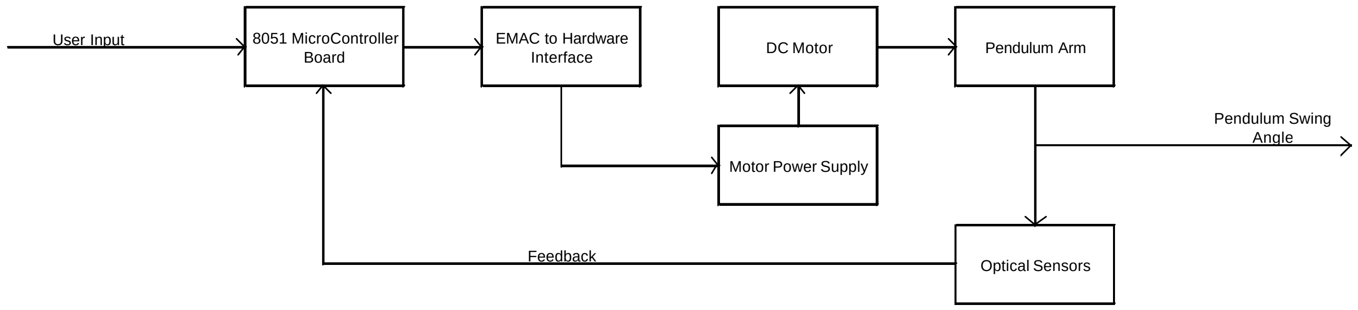


Figure 1. High Level Block Diagram

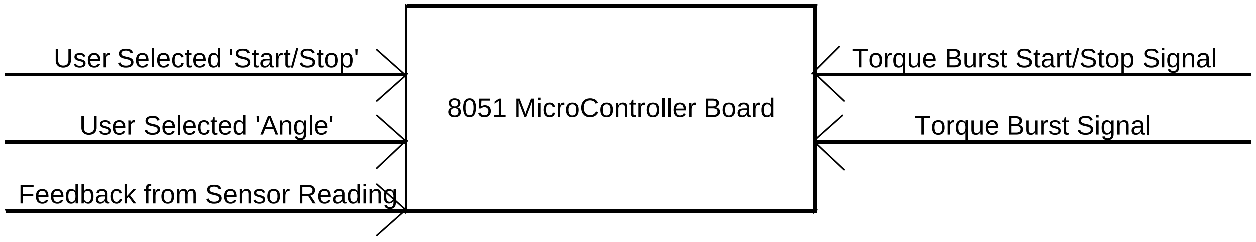
Figure 1 shows the high-level block diagram for the previously described system. The user will input a desired height and instruct the motor to begin. The user's inputs are given through the EMAC microcontroller keypad, which then sends the desired angle, and start command. The microcontroller outputs must then travel through a hardware interface that will convert the EMAC signals to an appropriate motor control signal.

The motor, draws power from a DC power supply that will produce a torque pulse for a limited period. The burst length will be determined by both the microcontroller according to desired height, and the power available to the motor from its supply.

The torque bursts will then propel the pendulum arm. Further research and testing will determine the point in the pendulum's oscillation at which the torque burst will take place. Optical sensors will be monitoring the angle at which the pendulum is swinging, and will be providing feedback to the microcontroller. The motor will continue to provide bursts until the sensors report that the pendulum has reached the appropriate height.

Once the pendulum has reached the desired height, the sensors will continue to provide feedback pertaining to the height of the pendulum. If the pendulum falls below the sensor level, another motor burst will be required to return the pendulum to the appropriate height. The user will have the ability to end the oscillation at any point by selecting the 'Stop' option on the microcontroller keypad.

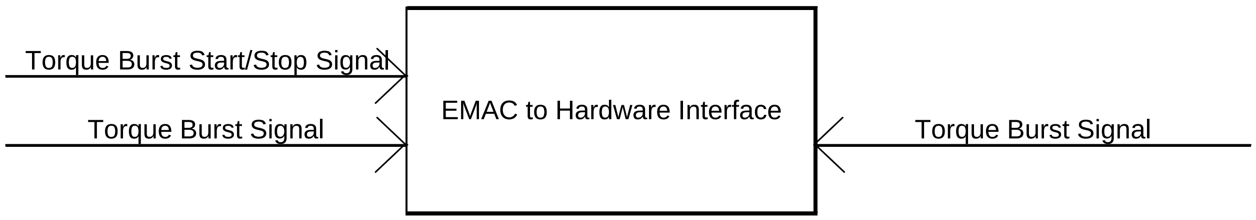
VI. Microcontroller Subsystem


User Selected 'Start/Stop'
User Selected 'Angle'
Feedback from Sensor Reading
8051 MicroController Board
Torque Burst Start/Stop Signal
Torque Burst Signal

The microcontroller subsystem is contained on the EMAC microcontroller kit. The microcontroller's keypad and LCD will be used for user interface ability. The inputs to the microcontroller will be the user selected 'Start' and 'Stop', which will be entered through the microcontroller keypad and the feedback signal from the optical sensors.

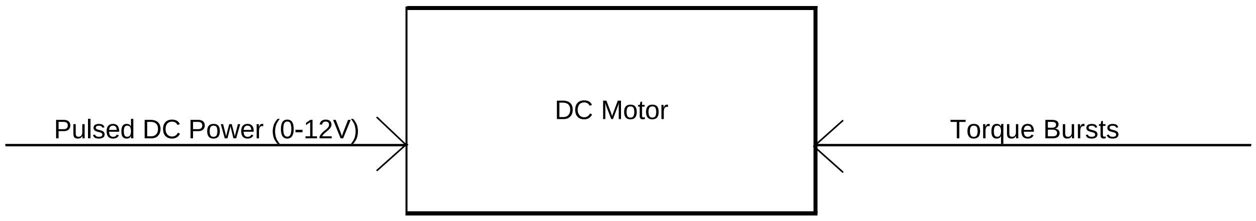
The microcontroller will send a high level signal to begin the torque bursts upon accepting a 'Start' command from the user. This motor burst signal will be sent to the hardware used to interface the microcontroller and the motor. The burst signals will continue until feedback from the sensors indicates that the pendulum has reached the desired angle. The sensors will continue to send feedback so that when the pendulum arm falls below the desired angle, the microcontroller will begin to send the burst signal again.

VII. MicroController to Motor Interface Hardware Subsystem



The purpose of the microcontroller to motor interface hardware subsystem will be to convert the microcontroller's output signal to a signal that is at the appropriate levels to be usable by the motor power supply. The interface hardware will also serve as protection for both the EMAC kit as well as the motor. The interface hardware will be designed to be minimal.

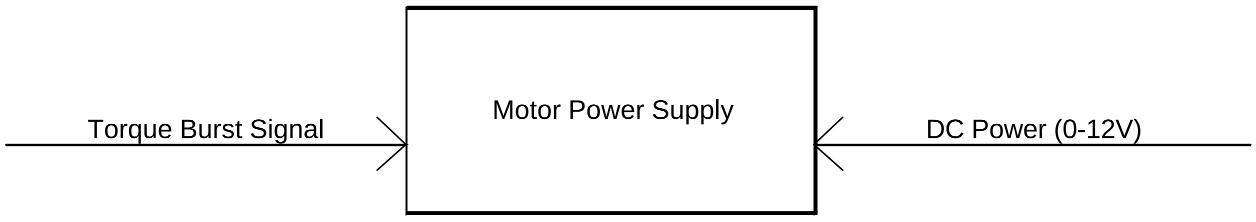
VIII. DC Motor Subsystem



The DC Motor subsystem will comprise of a small (< 1lb.) Pittman DC motor. The motor is rated at 12VDC, which will be able to run at supplies as low as 1VDC. A non-geared motor is more suitable for this application, so that it will be able to turn freely when the motor is not powered. This is important so that the oscillation of the pendulum is not hindered.

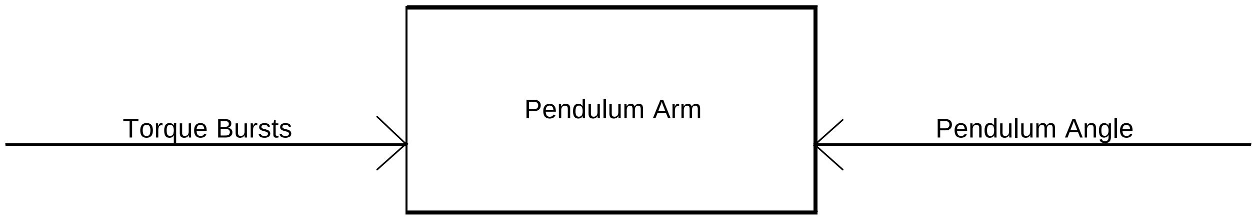
The motor will accept its input through a transistor switch. This switch will allow the power supply to send a pulse to the motor to create a torque burst. Additional H-bridge circuitry may later be added so that the motor will be able to be pulsed on both the rising and falling oscillations.

IX. Motor Power Supply Subsystem



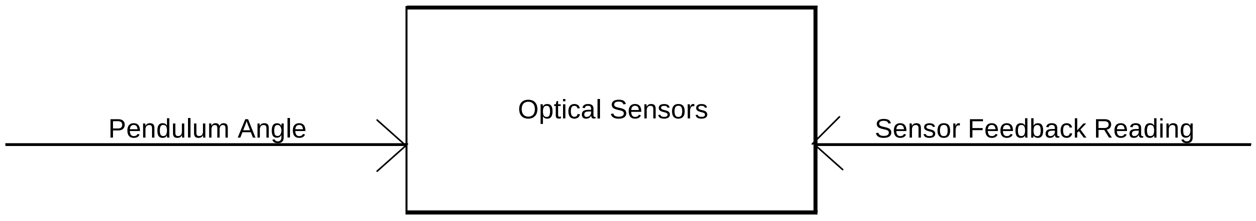
The motor power supply will be a DC power supply less than 12V. The power supply will be designed to have minimal reserve power and for economical purposes will not be able to drive the motor continuously. Testing will be done to determine the lowest supply voltage necessary for the motor to still function at a high enough level to still raise the pendulum arm. Supplies such as 9V batteries will also be explored as alternatives.

X. Pendulum Arm Subsystem



The pendulum arm will be the mass that is oscillated by the motor. The pendulum will be directly attached to the shaft of the motor, and it will extend to the sensors. Multiple designs for the pendulum will be explored. The most common design would be a clock-type pendulum arm, while an alternate design that will be researched is a complete circle mass with the lower section weighted. The pendulum must have an area that will be able to trigger the sensors, because the angle output provided by the pendulum is recorded by the optical sensors. The pendulum will also have to be of an appropriate weight for the DC motor.

XI. Optical Sensor Subsystem



The optical sensors will at all times be recording the height level that the pendulum is reaching. This height will be sent back to the microcontroller as feedback. The number of sensors to be used is still to be determined. Multiple sensors will allow for a larger variance in angle that would be able to be selected by the user.