Self Guided Catheter

Final Report

Derek Carlson & Caleb Anderson

Dr. Dempsey & Dr Stewart
5/16/07

Abstract: In surgical operations that require the use of a catheter, multiple guidewires must be used in order to direct the catheter through the patient's arteries. The goal of this project is to eliminate the need for different guidewires and to add precision control to the guidewire itself. Also, the surgeon will be able to control the guidewire from a remote location simply by viewing the patient through a camera. Control will be implemented using a joystick interfaced with Simulink using RS-232 as an output. This will drive the controller board that will run two stepper motors. These stepper motors will be used for lateral motion and guidewire advancement. A voltage reactive polymer will be used for precision tip control.

Introduction: A high-level functionality block diagram of the catheter control system is shown below.

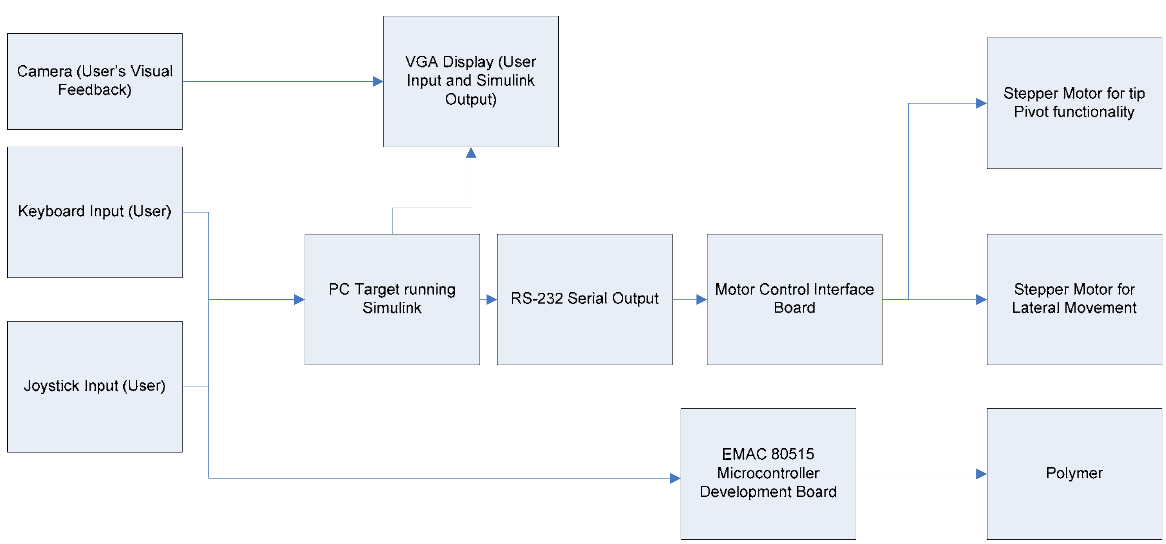


Figure 2-1: Catheter Guidewire Control System Block Diagram

Figure 2-2 shows that user input is available through both keyboard and joystick input into the control system. Keyboard input is realized via USB or PS/2 into Matlab. The joystick inputs into both the 80515 Microcontroller Development Kit along with Matlab allow for polymer and stepper motor control. Matlab is used to supply the signal to the motor controller board running the two stepper motors. Visual feedback is implemented through the use of a webcam and Matlab's Image Acquisition Toolbox. This will allow the surgeon to operate from a remote location.

Polymer Control: As explained before, the polymer control of the Catheter Guidewire Control system is to add precision tip control in the left and right directions for simple turning of the guidewire tip. Voltage vs. curvature testing was completed for two different polymers (platinum plated; gold plated.) Voltages ranging from 0 to 5 volts DC were the polymer power source. These tests were done a number of times, with the polymer soaked every few points to get a "best case" scenario. A run time measurement was also taken to determine the total time that the polymer could be deemed useful before drying out. Figures 3-1 , 3-2, 4-1, and 4-2 below and on the next page show voltage vs. curvature plots for both platinum and gold plated samples.

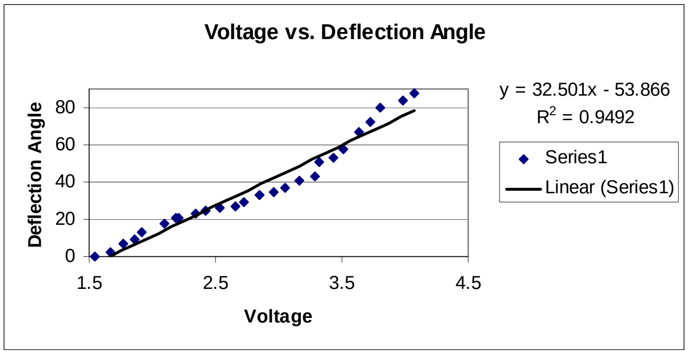


Figure 3-1: Platinum Plated Positive Voltage

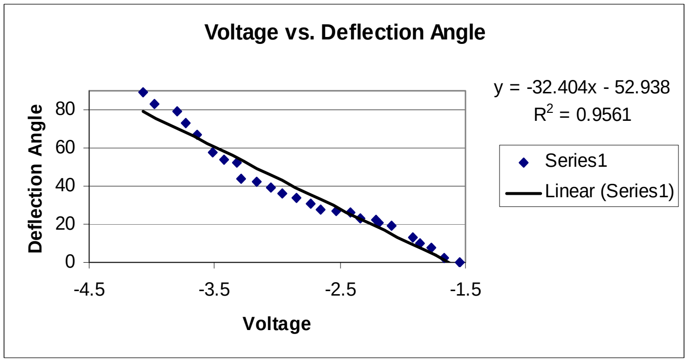


Figure 3-2: Platinum Plated Negative Voltage

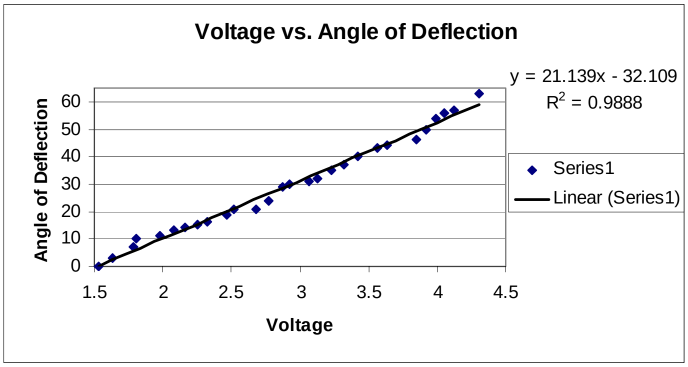


Figure 4-1: Gold Plated Positive Voltage

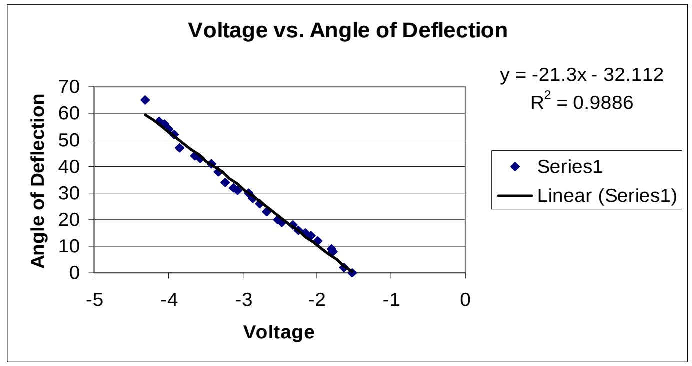


Figure 4-2: Gold Plated Negative Voltage

From the voltage vs. curvature charts shown on the previous page and above, it can be seen that the gold plated polymer has a slightly more linear response than the platinum plated polymer. The angle of deflection though is slightly higher for the platinum plated polymer. The voltage source input for both polymers is virtually the same to receive similar curvatures. For the guidewire control system, linearity of the polymer response and angle of deflection are both important. From the tests, the gold plated polymer has a maximum of approximately 65 degrees. This is beyond what is needed for the guidewire control system. Because of this, the gold plated polymer was chosen for the project solely based on its linearity.

The gold plated polymer was also tested for voltage vs. current measurement using the current draw reading off of the power supply, and also checked with a multimeter in series with the load. The voltage applied to the polymer was varied from 0 to 5 volts DC, and measurements were taken for three different samples of gold plated polymer. The results in these samples varied greatly but all seemed to have a maximum draw of 250 milliamps. This occurred anywhere from 4 to 4.5 volts and varied from sample to sample. The data was not always consistent, but at the very least a maximum current draw was able to be recorded for each of the three samples. (This inconsistency will be discussed later in the Polymer Problems and Concerns section.)

For control of the polymer, it was decided upon to try both PWM signals and regular DC voltages to see which had a better response with the polymer. Voltage vs. curvature data had already been taken for DC voltages by two students from the Mechanical Engineering department. For testing of the PWM signal, a 5 volt unipolar PWM signal was used as the power supply for the polymer. Frequencies of 20 Hz to 2 MHz were tested along with duty cycle percentages ranging from 0 to 100 %. No data was able to be recorded from these tests, since the polymer was virtually unresponsive for all duty cycles and frequencies tested. A maximum of 10 degrees of curvature was seen at 100% duty cycle at 200 Hz. This data ruled out a PWM signal as the method of control for the polymer tip of the guidewire.

Due to the need for DC voltage control to supply power to the polymer, an 80515 EMAC Microcontroller Development Board was used. The D/A converter portion of the board is utilized to provide a voltage from 0 to 5 volts. The original program allowed the user to input a percentage from 0-100% of five volts to control the polymer. The program was later updated to incorporate the use of two buttons of a Wingman Attack II joystick. (This left forward and backward functionality of the joystick for motor advancement.) Each button provided an “up” or “down” functionality. With a single press of a button, the D/A output can increase or decrease by 1% of the full five volts. This corresponds to 0.05 volt precision for control of the polymer tip. Finally, a readout on the LCD interface allows the user to see the current percentage of voltage being applied to the polymer. Figure 6-1 below shows a software flowchart for the polymer control via joystick.

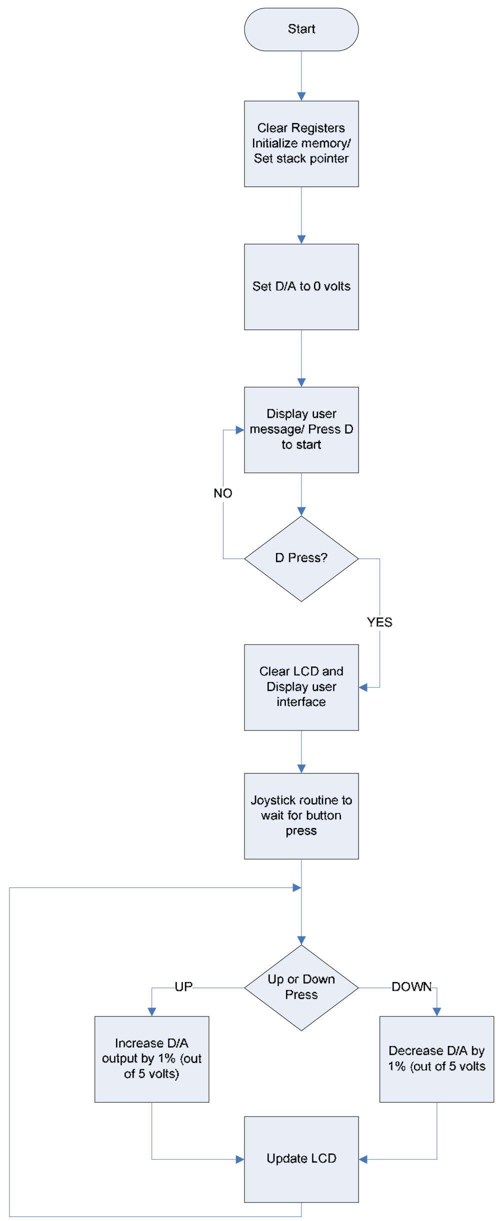


Figure 6-1: Flowchart for D/A software

Due to the large current needed to move the polymer an LM 3886 audio amplifier was used to amplify the D/A output current of the EMAC 80515 Microcontroller Development Board. This audio amplifier needed a gain of at least 10 along with a Current > 0.0005 Amps across the "muting" pin to turn off the "muting" function of the amplifier. Because the input signal is DC, there is no need to worry about signal gain. With a feedback capacitor of 10μF there will be unity gain at DC. A schematic of the configuration used is shown in Figure 7-1 below.

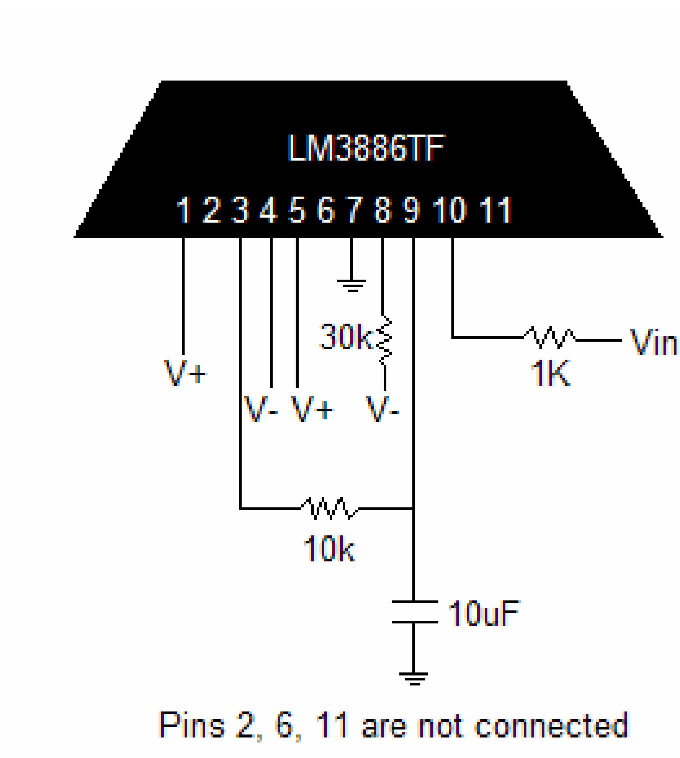


Figure 7-1: LM3886 schematic for D/A amplification

Polymer Problems and Concerns: After testing multiple polymers, it is apparent that this material is not ready for a device that needs the precision of guidewire control. All of the electroactive polymer utilized for this project was ordered from www.environmental-robots.com. While this company is one of the leaders in electroactive polymer development, the manufacturing process has not yet been perfected. Polymer strips of the same size, weight, and material react very differently than others. Some are very unresponsive after only a short time, while others last much longer. Some samples of the polymer bend very linearly when a DC voltage is applied, others curl or wiggle, and some do not have any functionality at all. Due to these imperfections, it is hard to characterize the polymer enough to develop a control system.

Along with the manufacturing process, it was easily seen during testing that the polymer could not provide much force when bending. At certain times, it could not even surpass its own tensile strength, and would jump a great deal from one applied voltage to the next without bending linearly. This could possibly be fixed by braiding small thin strands of the polymer to form a cable like structure to maximize power, but was not tested. The braiding of polymers greatly increases the total size, which causes another problem when dealing with the small area of a human artery.

The polymer also draws a significant amount of current throughout use. While it would be nice to have a nice set of data regarding polymer current draw, we tested each polymer multiple times and received a huge variance in current draw. Some would draw no current and would not move until it dried out a bit, at which point the current draw would spike up to about 0.5 amps. Others would show little to no current draw and would move, yet as they were used for a minute or so, the current draw would spike up to about 0.5 amps and the connection between the polymer and power supply would begin to smoke the polymer. Basically, the current draw of the polymer is roughly a max of 0.5 amps, which is a significant current when within the human body. This is another concern regarding the realistic use of this material at this point.

Stepper Motors: The first thing that needed to be purchased was a stepper motor control board. This board needs to be able to control two stepper motors from a PC running Matlab. It needs to be able to accept RS-232 commands, or use its own USB driver. Matlab has a fully featured serial toolkit, so having serial ability would allow the board to be controlled by Matlab or an 8051 Microprocessor. After searching around online, it was decided that the Bistep Motor Controller board from www.hobbyengineering.com fulfilled all of our needs. Directly from the website, the board has the following specifications:

BiStep Motor Controller (USB)

The BS0710-USB board is a complete Unipolar/Bipolar dual stepper motor controller system which can be controlled from a computer using a USB connection. It includes the capability of driving one or two stepper motors, each of which being either unipolar (4-pole) or bipolar (2 pole). This unit is a good choice for those designing products using linear actuators, especially since the micro stepping features will reduce noise levels and can increase positional accuracy by a large amount.

The board is specified as allowing up to 1 amp per winding, with 6.5 to 15 volt motors supported when in single power supply mode. It also supports a dual power supply feature, wherein the motors use a separate supply from the logic circuits; in this mode, up to a 34 volt power supply may be used for the motors.

Key Benefits:

- USB communications with a host computer may be used to control the board.
- TTL-Serial communications may be used to control the board using an external microprocessor, such as a Basic Stamp or SX.

- Up to two stepper motors may be independently controlled at one time.
- Each motor may be either 4-pole Unipolar or 2-pole Bipolar.
- Four modes of stepping the motor are supported:
    - Half steps (alternates 1 winding and two windings enabled at a time)
    - Full power full steps (2 windings enabled at a time)
    - Half power full steps (1 winding enabled at a time)Near-constant-current microsteps (each from 1/64 to 1/2 of a full step)
- A TTL "busy" signal is available, which can be used to see if the motors are still moving. This information is also available from the serial communications subsystem.
- Simple control of the motors may be done by switch closure: i.e., each motor can be told to slew left or right, or to stop by simply grounding some input lines. Similarly, the rate of motion can be controlled via stepping through a standard set of rates via grounding another input.
- Complete control of the motors, including total monitoring of current conditions, is available through the 2400 to 19,200 baud serial connection (by default, this is shipped configured for 9600 baud).
- Runs off of a single user-provided 6.5 to 15 volt DC power supply.

Figure 10-1 Bistep Motor Controller Data

This board was ordered and the documentation was read. This board uses an emulated serial interface that allows it to be connected and installed as a USB device, while still allowing the user to simply use serial commands to direct it. When it is first installed, it is recognized as a com port by the computer. As per the instructions on the data sheet, the com port was set up with the following settings:

Bits per second: 9600
Data Bits: 8
Parity: none
Stop Bits: 1
Flow Control: none
Receive Bytes: 64
Transmit Bytes: 64
Latency Timer: 1 ms
Serial Enumerator: On

Figure 10-2 Com port settings for motor board

Once the serial port was set up correctly, basic communication via windows Hyperterminal worked correctly. Next, motors needed to be chosen.

Two stepper motors are needed to control the guidewire. Many were looked at, and it was finally decided upon two 8.4V stepper motors from Jameco.com. These two stepper motors have a 0.9 degree step size, are bipolor, and run low voltage. Both lateral advancement and rotation need the same strength motor, so this motor will fulfill requirements for both. Also, based on the data sheet, it will interface with the control board we purchased. The basic data from the website follows:

MOTOR,STEP,8.4VDC,30 ohm

Stepper Motor

- Excellent for precision control
- Can be operated in forward/reverse mode
- Excellent torque/size ratio
- Wide variety of supply voltages
- Data sheet included
- Dual shaft
- Check for compatible power supplies
- Step angle: 0.9 degrees
- No. of phases: 2
- Drive System: Bipolar
- Voltage(VDC): 8.4
- Phase resistance (Ω): 30
- Current (mA): 280
- Phase Inductance (mH): 25
- Detent torque(g-cm): 36
- Holding Torque(g-cm): 791
- Mounting hole space digonal (in.): 1.73
- Mounting hole (in.) 0.15
- Shaft diameter (in.): 0.155
- Shaft length(in.): .29
- Motor diameter (in.) 1.64
- Motor height (in.) 1.20
- Weight: 0.53 lbs.

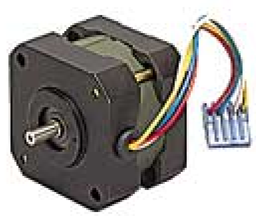
Jameco P/N	163395
Mfg	VARIOUS
Mfg #	APPLIED MOTION 5017-935
RoHS?	No
In Stock	Yes

MOTOR,STEP,8.4VDC,30 ohm
.9deg,SHFT:.16"X.39"

Sale

# of Units	$US EA
1+	5.59
10+	4.79
50+	3.59

Add to my products



Order Quantity

1

Add to cart
View cart

Figure 11-1 Jameco Motors data

These motors were ordered and ready to link up with the motor controller board. The board’s documentation actually recommended this model as one that works well with the board. It was easy to wire these motors up because the exact schematic is available in the motor controller board data sheets.

The motors were hooked up and functionality was tested. Each function of the motor controller board was tested and verified. All functions were verified working. Although there are many functions for this motor controller board, the main ones used are the following:

B – Select Both Motors
G – Go to position on the current motor(s)
I – Wait for motor ‘idle’
M – Mark location or go to marked location
X – Select motor X
Y – Select motor Y
Z – Stop current motor(s)
! – Reset – all values cleared and motors set to free. Duplicates power on conditions.
? – Report Status
= -- Define current position for the current motor to be ‘x’; stop the motor

Figure 12-1 Motor Controller Board Commands

These commands were simply written to the device serially through HyperTerminal, at first to test, and then later through Matlab using Matlab’s serial communication commands. The basic commands used to set up the port and begin communication are:

```
s = serial(‘COM3’);
fopen(s);
set(s, ‘BaudRate’, 9600, ‘Parity’, ‘none’, ‘DataBits’, 8, ‘StopBits’, 1);
fwrite(s, ‘command’);
```

Figure 12-2 Matlab Serial Initialization

Setting up the serial port as in Figure 12-2 was very basic and simple – getting the board to communicate correctly all the time was a bit more of a challenge. The board echoes information after every command you give it, and it tends to have problems waiting in between commands. The 'I' command of the motor controller board does in fact make the board wait until the motors are idle, but for some reason, it sometimes will time out when it shouldn't. Also, the board has a habit of not wanting to clear even when the '!' command is given to put it into the reset state. All of these reasons made it a bit difficult to work with. The basic command functionality through Matlab simply used an infinite loop while waiting for user input. Matlab would then move the motor to the position needed and then report back to the user. A basic GUI was designed in Matlab's GUIde environment to control the motors and report status back to the user.

Schedule:

- Weeks 1-3
  - Stepper motor MATLAB interface (Derek & Caleb)
  - Webcam MATLAB interface (Caleb)
  - RS-232 MATLAB interface (Derek)
- Weeks 4-6
  - GUI for user input (Both)
  - Develop polymer electrical characteristics (Both)
    - Detailed current draw calculations
- Weeks 7-8
  - Develop power electronics to actuate polymer (Derek)

    - Possible polymer shielding concerns (Caleb)
- Weeks 9-10
    - Stepper motor physical interface (Caleb)
    - Guidewire drive system (propulsion, rotation, etc.) (Derek)
- Weeks 11-14
    - Build test system (Both)
    - Finalize report & Presentation(Both)

Equipment List:

- (2) 1 degree stepper motors
- (1) Bistep Motor Controller Board
- (1) Electroactive polymer tip
- (1) Surgical Guidewire
- (3) DC voltage sources – (2) for motor control (1) for op-amp.
- (1) PC USB Webcam
- (1) Logitech Wingman Pro Attack 2 Joystick
- TBA – mechanical system to test guidewire

Conclusions: In theory, this idea for an Electronically Controlled Catheter Guidewire could be groundbreaking for the medical field. The benefits of reducing surgery time, and reducing the number of guidewire used are innumerable. At this time though, the gold plated polymer used for guidewire tip control is not developed enough for use. The problems explained above, accompanied with the difficulty to control the polymer with any precision make the Electronically Controlled Catheter Guidewire very improbable with current polymer technology. If a more responsive, longer lasting, and more easily controllable polymer were developed though, this product could be very feasible. The scope of our project involved starting from zero and creating a system to control an electroactive polymer within a human body. As our schedule suggested, it was hoped that by the end, a working large scale model could be made to move the polymer and advance the guidewire. Time constraints, as well as numerous problems with the polymer itself, stunted development and helped us learn that things do not always work as one would imagine. At this point, the polymer samples that were worked with proved to be too unpredictable to build a control system around them.

The use of electroactive polymers for robotic control, artificial muscles, and other medical applications, seems to be causing a great stir in the engineering world. It is highly probable that these electroactive polymers will become much more developed over the next few years, and used in multiple facets of science and engineering. At this time, a project like this could be revisited with greater success.

References:

Polymer material source

http://www.environmental-robots.com

Patent information

http://www.patentstorm.us/patents/6997870.html

http://www.micromuscle.com

Motor Control Board Source

www.hobbyengineering.com

Appendix:

```
s = serial('COM3');
fopen(s);
set(s, 'BaudRate', 9600, 'Parity', 'none', 'DataBits', 8, 'StopBits',
1);
fwrite(s, '?')
```

Figure 17-1 Initialization Code for Serial Communication

```
% Will accept data and echo to screen ascii data
%init = fread(s,64);
%disp(char(init'))
comm = 'N';
while (comm ~= 'Q')
disp('Available Commands:');
disp('A B E G H I K L M O P R S T V W X Y Z ! = ? "Q" TO QUIT');
comm = input('Enter command:> ', 's');
fwrite(s,comm);
resp = fgetl(s);
output = char(resp);
disp(output)
end
disp('You Quit.');
```

Figure 17-2 Command Line Code for Basic Command Entry

```
% Create a video input object.
cam = videoinput('winvideo');

% Create a figure window. This example turns off the default
% toolbar, menubar, and figure numbering.

figure('Toolbar','none','Menubar',
'none','NumberTitle','Off','Name','Guidewire Cam');

vidRes = get(cam, 'VideoResolution');
nBands = get(cam, 'NumberOfBands');
match = image( zeros(vidRes(2), vidRes(1), nBands) );

% Display the video data in your GUI.

preview(cam, match);
```

Figure 17-3 Webcam Interface for Matlab

80515 Assembly Code

Main Module

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;    When main begins I will start displaying ;;
;;    a message that prompts the user to press D to start.     ;;
;;    After the user presses C they will be asked to enter the desired percentage   ;;
;;    via keypad                                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

$NOMOD51
$INCLUDE(reg515.inc)

NAME    MAIN
PUBLIC  MAIN

EXTRN CODE (D2A)
EXTRN CODE (KEY)
EXTRN CODE (LCDOUT)
EXTRN CODE (MESSAGE)
EXTRN CODE (BOTTOM)
EXTRN CODE (BOTTOM2)
EXTRN CODE (ERROR)


main_seg     SEGMENT  CODE
        RSEG  main_seg


main:       ;PUSH ACC
        LCALL MESSAGE
start:      LCALL KEY
        CJNE  A,#'D',start
        MOV  A,#01H                 ;CLEAR SCREEN
        LCALL LCDOUT

loop:       LCALL BOTTOM                  ;Write to Bottom Line

        LCALL KEY
        MOV 50h,A

check:      CLR C
        PUSH B
        PUSH ACC
        ANL A,#00001111b
        MOV B,A
        MOV A,#01h
        SUBB A,B
        jc loop
        POP ACC
        POP B


        LCALL LCDOUT
        ANL 50h,#00001111b                  ;100's place
        LCALL KEY
        MOV 51h,A
        LCALL LCDOUT
        ANL 51h,#00001111b                  ;10's place
        LCALL KEY
```

```
        MOV 52h,A
        LCALL LCDOUT
        ANL 52h,#00001111b                  ;1's place
enter:    LCALL KEY
        CJNE A,#'E',enter                  ;wait for enter press

        LCALL BOTTOM2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; NOW CONVERT FROM 3 BCD DIGITS TO A SINGLE HEX NUMBER AND  ;;
;; STORE IT IN LOCATION 54H FOR PWM                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



combine:    PUSH  B
        MOV   B, #10d            ; put 10 decimal into B for multipication
        MOV   A, 51h             ; put tens place into A
        MUL   AB                 ; now high byte is in B, low Byte is in A
        ADD   A, 52h             ; add ones place to this value...cannot be more than 99
        MOV   54h, A             ; don't need to check anything...put into 54h
        MOV   B, #100d           ; put 100 decimal into B for multiplication
        MOV   A, 50h             ; put 100's place into accumulator
        MUL   AB                 ; high byte is in B, low is in A
        ADD   A, 54h             ; add the tens and ones places to A
        MOV   54h, A             ; put the full value into 54h
        MOV   B, #101d

err:      clr c
        SUBB  A,B
        JC   ok
        LCALL ERROR
        POP   B
hold:      LCALL KEY
        CJNE  A,#'C',hold
        MOV   A,#01h
        LCALL LCDOUT
        LJMP  Loop

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  NOW CALL D2A PORTION OF THE PROJECT                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ok:       POP B
        LCALL D2A

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  AFTER RETURN FROM PWM PORTION WAIT FOR 'C' PRESS ON KEY  ;;
;;  THEN CLEAR THE SCREEN AND JUMP BACK TO loop              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
clear:    LCALL KEY
        CJNE A, #'C', clear
        MOV A,#01h          ;clear screen
        LCALL LCDOUT
        MOV A,#02h          ;carriage return
        LCALL LCDOUT
        LJMP LOOP

ret
END
```

Figure 18,19-1: Main Module For D/A Assembly Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   This is the D2A code which will put a value from the lookup table (0-255) which ;;
;;  represents an analog signal of 0-5 volts into external memory location 10h.     ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

$NOMOD51
$INCLUDE(reg515.inc)

NAME    D2A
PUBLIC  D2A


EXTRN CODE (KEY)
EXTRN CODE (LCDOUT)
EXTRN CODE (GETVAL)


d2a_seg     SEGMENT  CODE
            RSEG  d2a_seg


D2A:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  This portion will use the user input stored in 54h.  It will then use the lookup ;;
;;  table to return the correct value.  THis value will then be put into external    ;;
;;  memory location 10h (channel A)                                      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



go:       mov  A, 54h
        mov  DPTR,#getval
        movc A,@A+DPTR                  ;put data table value in A
        mov  55h, A                     ;put this into storage in 55h


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  Now right the information in memory location 55h to memory location 10h.
  This is channel 1 of the D2A converter.                                    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

convert:   CLR  P5.1
        MOV  DPH,#10h
        MOVX @DPTR, A


ret
END
```

Figure 20-1: D/A conversion Assembly Code

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

NAME lookup
PUBLIC      getval
PUBLIC      lookup

lookup_seg   EQU  9000h
          CSEG AT lookup_seg

lookup:


getval:   DB
00d,03d,05d,08d,10d,13d,15d,18d,20d,23d,26d,28d,31d,33d,36d,38d,41d,43d,46d,48d,51d,54d,56d,59d,61d,64d,66d,6
9d
        DB
71d,74d,77d,79d,82d,84d,87d,89d,92d,94d,97d,99d,102d,105d,107d,110d,112d,115d,117d,120d,122d,125d,128d,130d,
133d
        DB
135d,138d,140d,143d,145d,148d,150d,153d,156d,158d,161d,163d,166d,168d,171d,173d,176d,179d,181d,184d,186d,1
89d,191d
        DB
194d,196d,199d,201d,204d,207d,209d,212d,214d,217d,219d,222d,224d,227d,230d,232d,235d,237d,240d,242d,245d,2
47d,250d
        DB  252d,255d


END
```

Figure 21-1: Lookup Table for D/A conversion