EE300 PROJECT

PC BASED CONTROL OF ASEA MHU JUNIOR ROBOT

RON JACQUIER

S11042181

ABSTRACT

In this project, a PC-based controller is used to control the ASEA MHU junior robot. The controller is designed in Matlab and it is a Graphical user interface (GUI) kind. The GUI includes both manual and automatic control of the robot. This paper describes the interfacing of the PC to the DAQ card and the code used to develop the GUI.

CONTENTS

LITERATURE REVIEW

A Control system refers to a system whereby an operator manually controls a machine or process and can include logic or feedback for better functionality. For example, operator closing a pneumatic gripper (manual) but the gripper will close only when object is in reach (logic/feedback implemented).

An automatic sequential system refers to a system whereby series of commands are executed in a specific order (defined by operator) to perform a task. For example, the commands of an automated process of moving an object from one place to another with a robot can be like the following; the gripper should first open then close, and then the arm should move to destination and gripper open again to drop object.

A simple ON-OFF system is a system where the actuators of a machine behaves in only two ways; either on or off. This control system is simple, cheap and effective; however considerations should be given to the wear and tear of the actuator vales and power consumption at start up of actuators.

A Linear control system is a system which uses negative feedback to produce a control signal that is calculated based on other variables. The output from the system can be a directly variable signal, such as opening a valve from 0 to 100% (includes any value in between e.g. 50%), or it can be a repeatedly switching on and off of an actuator to regulate the duty cycle using pulse-width modulation. Examples of linear control system types are; Proportional control and PID control. PID control is better as it takes care of the rate of change of error and also the long term error whereas Proportional control only takes into account the current error.

INTRODUCTION

ASEA, which is short for Allmanna Svenska Eleckriska Aktiebolaget, is a Swedish company founded in 1889. It was originally an electrical light and generator manufacturer. They only started manufacturing robots in 1974.

The ASEA MHU Junior robot has a degree of freedom of five. This means that the robot has five joints or movable axis. The robot's primary function is to move objects from one place to another within its reach. The robot's arms are driven by pneumatic drives at six hundred kilo Pascal. The movements of the arms are controlled by the opening of the actuators' valves. With some sensors incorporated into the robot, a controller can be made to govern the robot. The robot is an easy ON-OFF control type due to it being powered by pneumatics and can be controlled by several control systems. These are microcontroller based control, PLC control and PC based control.

In the olden days, logic control systems were implemented at the mains voltage using interconnected relays. Today, most of these systems are implemented using PLC (Programmable logic controllers) or microcontrollers. Computer based controller can also be used but due to its large power consumption, reboot time, reliability and size, it is less preferable. However, computer based controllers have more information processing power and can be easily manipulated to produce a more user friendly controller that can have a lot more features. For example, a GUI having a webcam feed from the robot implemented on the computer based controller or a remote PC controlling the robot through LAN or internet.

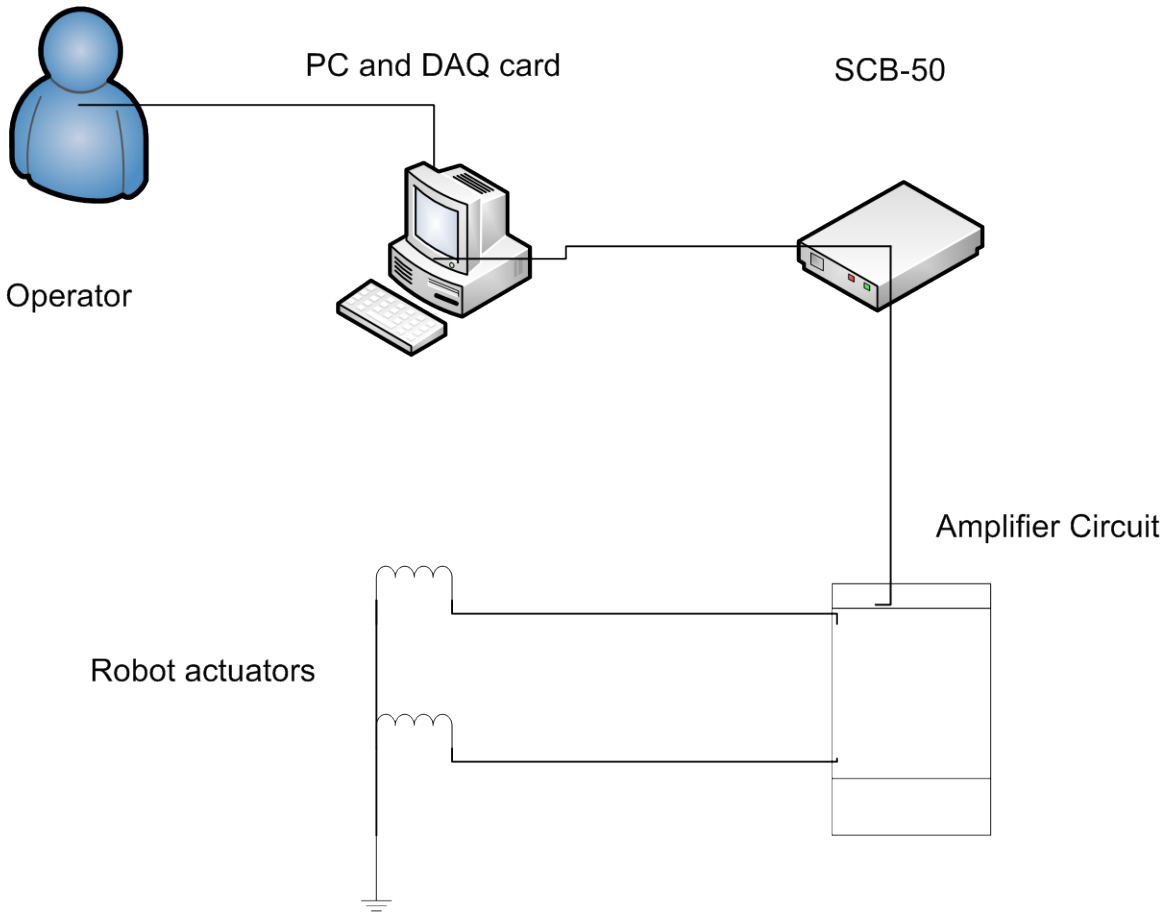
In this project, a PC based controller is designed and used to control the ASEA MHU Junior robot. The developer program used to create the GUI is Matlab. Matlab was used because of two main reasons. First of all, it has the Data Acquisition toolbox and necessary drivers which makes it possible and simple to use and configure the Data Acquisition (DAQ) card. Secondly, it is a very powerful program that allows users to solve problems, produce graphics, produce code efficiently and lots more, which makes it suitable for the current task.

The Data Acquisition card used was a PCI-DAS6025. It is a MCC card which has 16 channels of 12-bit A/D resolution and a max throughput of 200kS/s (max 50kS/s for any channel)

THE INTERFACE

The robot is interfaced to the computer by the PCI-DAS6025 DAQ card through an amplifier circuit. The amplifier circuit amplifiers the 5V signals from the DAQ card to 24Vdc which is the voltage required to power up the solenoids of the robot. The DAQ card has 100 I/O pins which can be connected to the robot by the shielded signal connection box (SCB-50). The shielded signal connection board provides screw terminal connections to the computer board.

<u>The Interface diagram:</u>



PCI-DAS6025

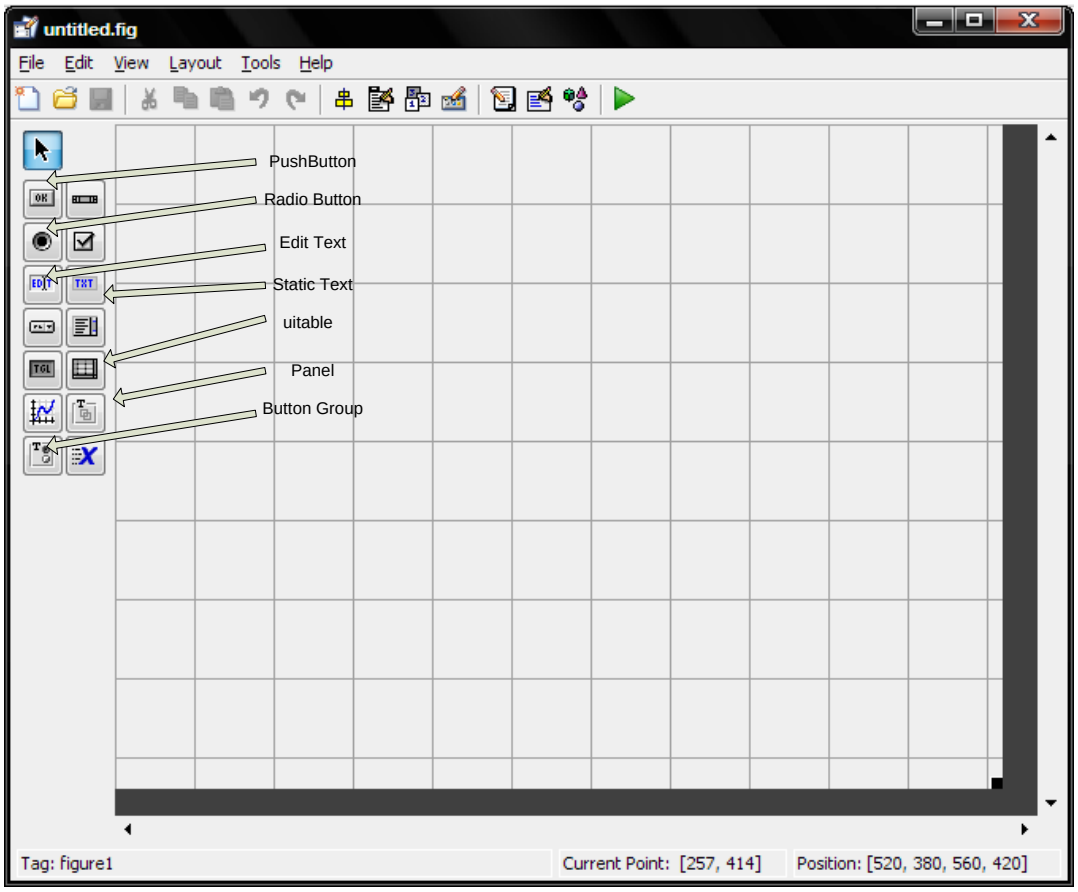
SCB-50

The following table shows how the robot is interfaced to the DAQ card:

Port	**Pin**	**Configuration**
1 (Port configured)	1	Gripper rotates clockwise
	2	Gripper rotates anti-clockwise
0 (Line configured)	1	Arm in
	2	Arm out
	3	Gripper closes
	4	Gripper opens
	5	Column down
	6	Column up
	7	Arm turns left
	8	Arm turns right

MATLAB GRAPHICAL USER INTERFACE (GUI)

The GUI was designed in matlab using the command 'GUIDE'. It gives you an empty template with different GUI objects available. The GUI objects used in this project were, pushbuttons, uitables, radio buttons, button groups, static texts, edit texts and button Panels. Pushbuttons are buttons that can be pressed and once pressed, the code embedded in the pushbutton will be executed. Uitables stores numerical data defined by the user. Radio buttons returns a true/false value depending if it is selected or not. Button groups allows for only one button in a group of buttons to be selected at any time. Panels allows for better organization of the GUI. Static texts allows for output text from the GUI and finally edit texts allows for User input text from the GUI.



GUI OPERATION

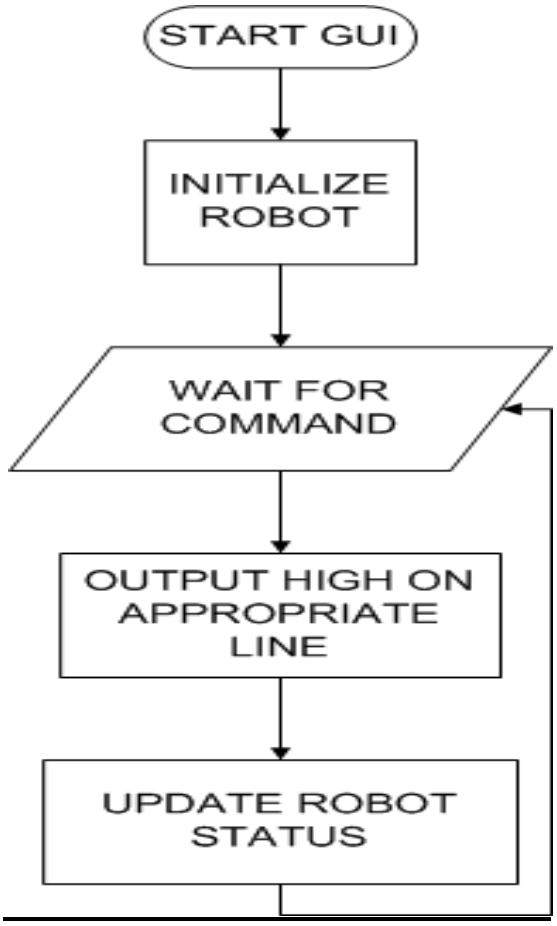
The basic operation of the GUI is to produce codes for the DAQ card to output 5V signals (digital signals) to the appropriate lines for the different robot motions. The GUI also handles input from the robot (feedbacks) to better its functionality.

The GUI has two modes of operation; the manual operation and automatic operation.

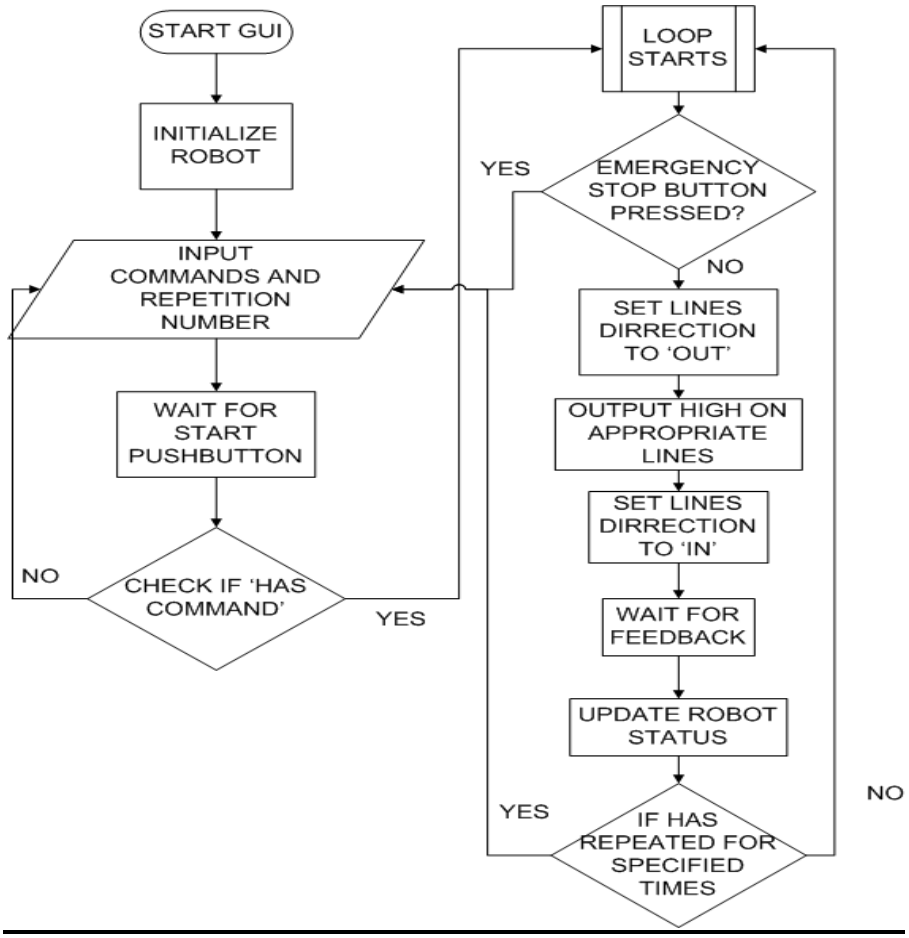
In manual operation, the operator/user clicks on the pushbutton corresponding to a robot movement and the DAQ card outputs 5V on the line corresponding to the specified robot movement. There is no feedback handling in manual operation.

In automatic operation, the operator/user specifies a series of movement by clicking on the pushbuttons and then the GUI will run each movement in the order specified. This mode of operation has feedback handling. Each line is used as both input and output. The automatic operation is designed in a way that in order for the next command to be executed, there must be a feedback from the previous movement implying that that movement has been executed.

FLOW CHART OF MANUAL OPERATION



FLOW CHART OF AUTOMATIC OPERATION



MAIN CODING DESCRIPTION

The following code creates a digital input/output object and adds lines for the MCC data acquisition card. That code allows for Matlab to communicate with the DAQ card.

```
dio = digitalio('mcc',0);
addline(dio,0:7,'out');
addline(dio,0:1,1,'Out');
```

This code sets the digital input/output object as a universal object therefore allowing for other functions to use the object.

```
handles.dio = dio;
```

The following code outputs a high (5V) then a low signal (0V) on the specified line. This is the code on every movement pushbuttons.

```
dio = handles.dio; //calls the digital input/output object
putvalue(dio.line(#),1); //sets the line high
pause(1)//waits for one second
putvalue(dio.line(#),0); // sets the line low
//where # is replaced by the line number
```

This code is for storing data in a uitable. This was the key function that allows for automatic operation.

```
set(handles.Table,'Data',[]);
//makes the table empty but providing it with dimension i.e. 0 x 0
data = get(handles.Table,'Data');//gets the data from the uitable
datasize = size(data);//determines the dimension of the data
row = datasize(1);//gets the number of row of the data
row = row+1;//increments the row number
data(row,1) = 1;//sets data in the new entry
set(handles.Table,'Data',data);//puts new data to uitable
```

The following code is for executing data in the uitable during automatic operation.

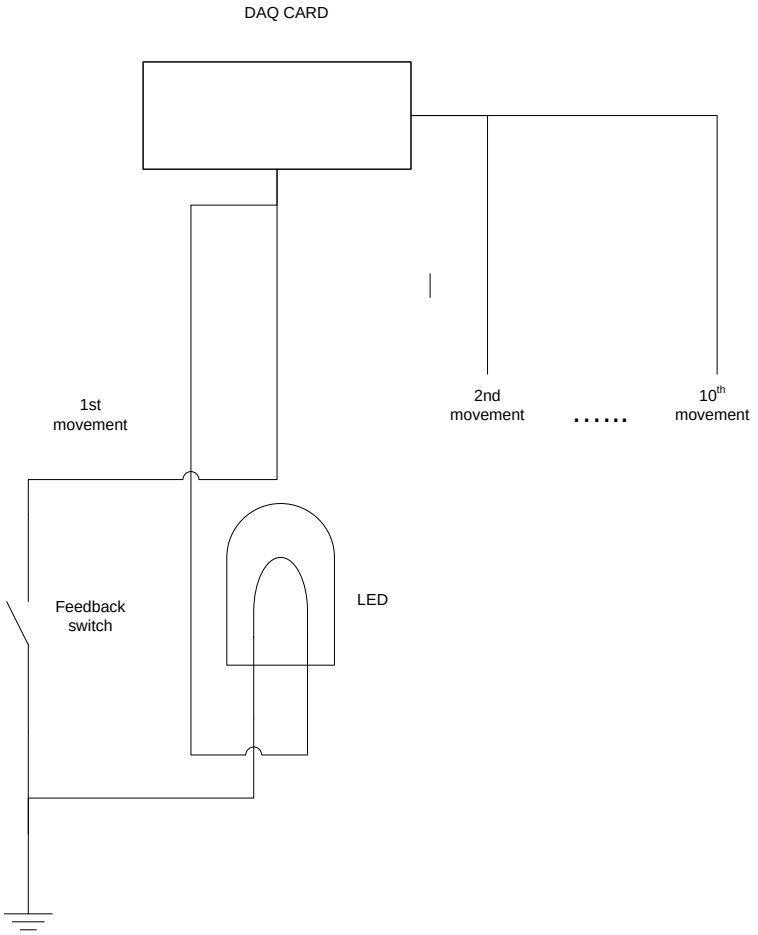
```
data = get(handles.Table,'Data');//gets data from uitable
datasize = size(data);//determine data dimension
row = datasize(1);//gets the number of rows of data

for x=1:row//runs the loop ‘number of rows’ times
      if data(x,1) == 1
      //out high on appropriate line
      //wait for feedback
      elseif data(x,1) == 2
      //output high on appropriate line
      //wait for feedback
      .
      .
      .
      .
end
```

TESTING OF GUI

The GUI was first tested with LEDs and switches. The LEDs acts as the robot solenoids while the switches acts as feedbacks from the robot. After passing the test with LEDs and switches, the test was done on the actual robot.

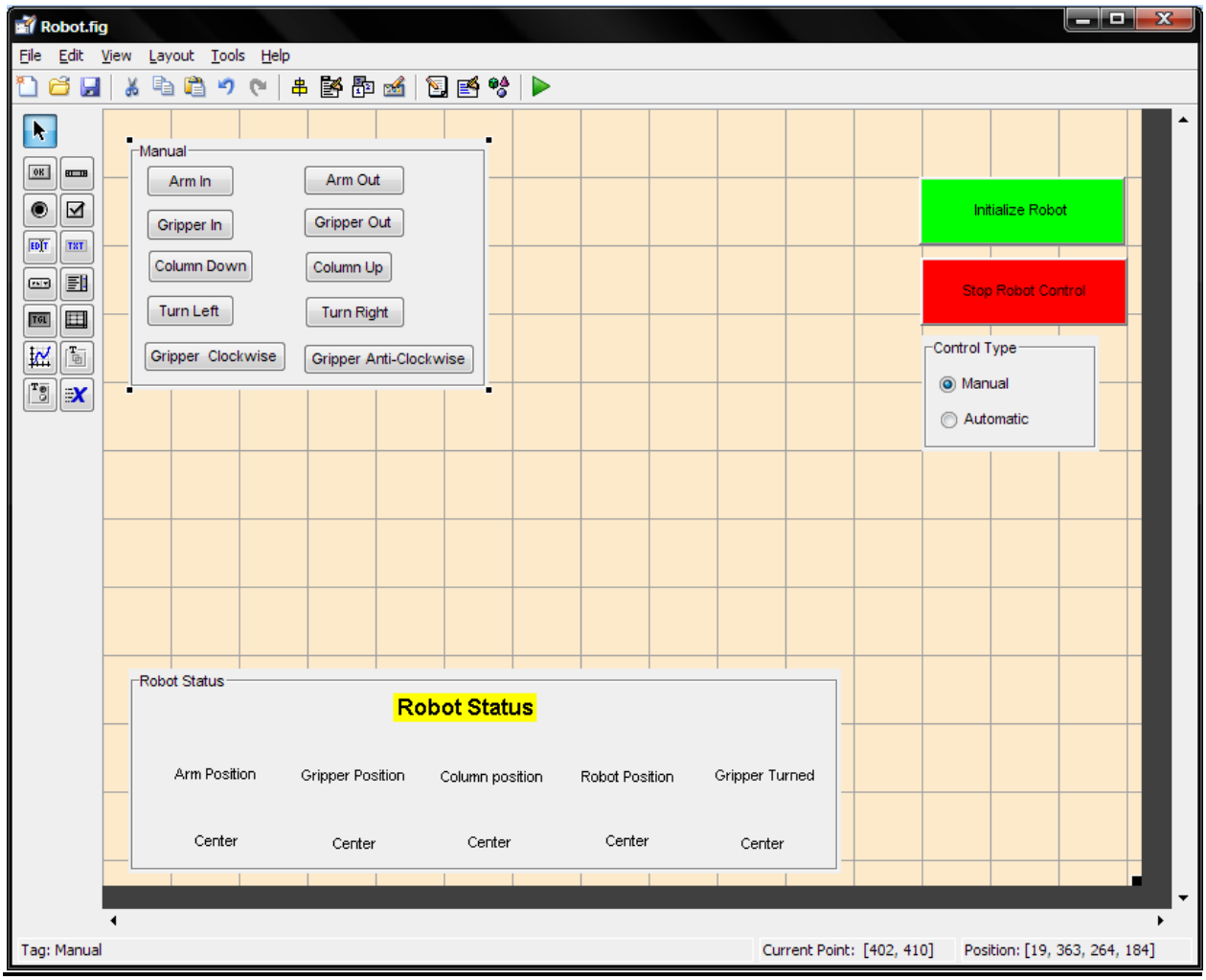
Below is a simplified drawing of the testing circuit:



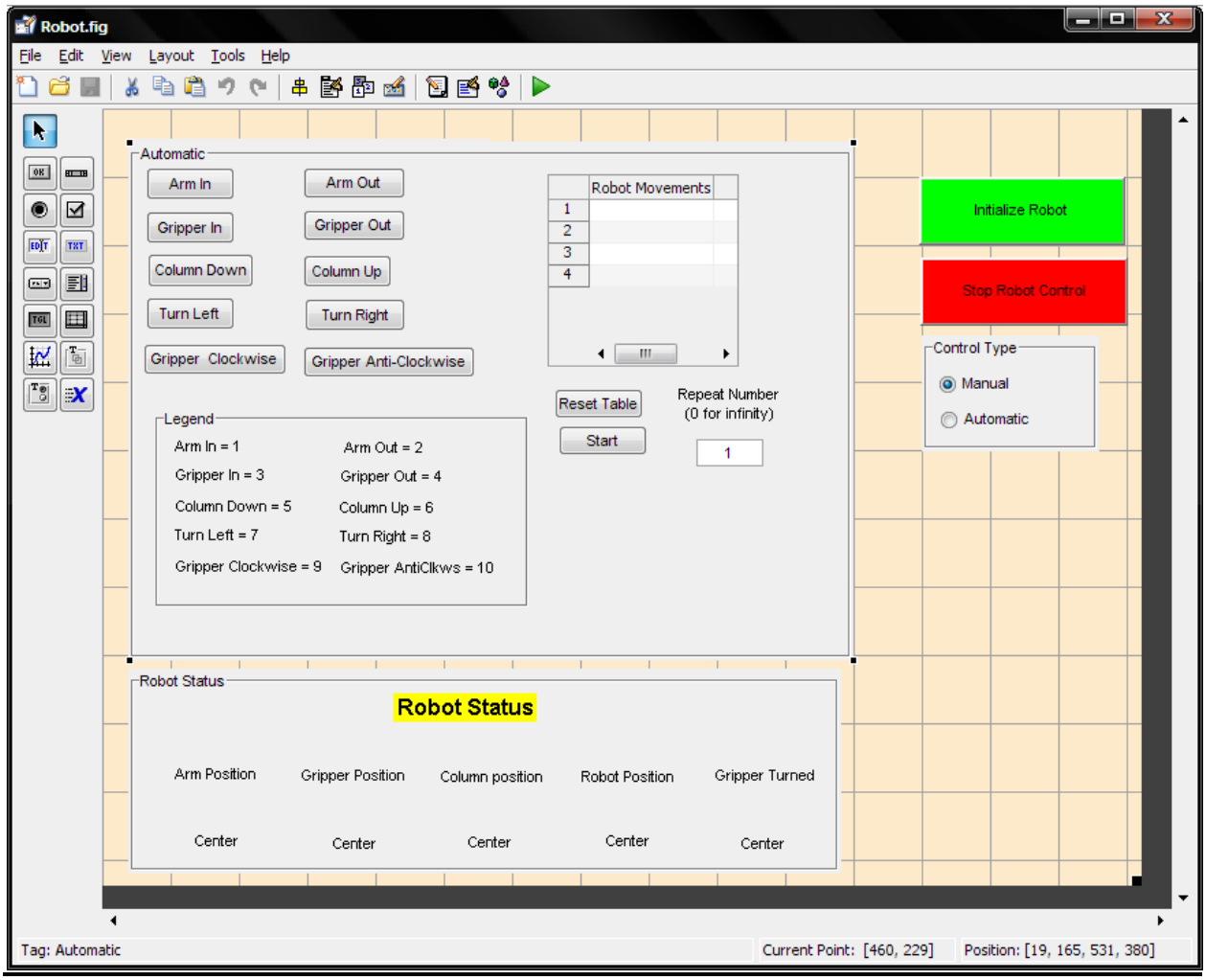
CONCLUSION

After all the different designs, the following pictures are the final GUI design.

MANUAL MODE:



AUTOMATIC MODE:



APPENDIX A: CODING

```
function varargout = Robot(varargin)
% ROBOT M-file for Robot.fig
%      ROBOT, by itself, creates a new ROBOT or raises the existing
%      singleton*.
%
%      H = ROBOT returns the handle to a new ROBOT or the handle to
%      the existing singleton*.
%
%      ROBOT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ROBOT.M with the given input arguments.
%
%      ROBOT('Property','Value',...) creates a new ROBOT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Robot_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Robot_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Robot

% Last Modified by GUIDE v2.5 20-Oct-2010 11:06:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Robot_OpeningFcn, ...
                   'gui_OutputFcn',  @Robot_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Robot is made visible.
function Robot_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Robot (see VARARGIN)
set(handles.Table,'Data',[]);
```

```
set(handles.stop_robot,'Visible','off');

% Choose default command line output for Robot
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Robot wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Robot_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in arm_in.
function arm_in_Callback(hObject, eventdata, handles)
% hObject    handle to arm_in (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(1),1);
pause(1)
putvalue(dio.line(1),0);
set(handles.arm_pos,'String','In');
guidata(hObject, handles);


% --- Executes on button press in arm_out.
function arm_out_Callback(hObject, eventdata, handles)
% hObject    handle to arm_out (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(2),1);
pause(1)
putvalue(dio.line(2),0);
set(handles.arm_pos,'String','Out');
guidata(hObject, handles);


% --- Executes on button press in grp_in.
function grp_in_Callback(hObject, eventdata, handles)
% hObject    handle to grp_in (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
dio = handles.dio;
putvalue(dio.line(3),1);
pause(1)
putvalue(dio.line(3),0);
set(handles.grp_pos,'String','In');
guidata(hObject, handles);


% --- Executes on button press in grp_out.
function grp_out_Callback(hObject, eventdata, handles)
% hObject    handle to grp_out (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(4),1);
pause(1)
putvalue(dio.line(4),0);
set(handles.grp_pos,'String','Out');
guidata(hObject, handles);


% --- Executes on button press in col_up.
function col_up_Callback(hObject, eventdata, handles)
% hObject    handle to col_up (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(6),1);
pause(1)
putvalue(dio.line(6),0);
set(handles.col_pos,'String','Up');
guidata(hObject, handles);


% --- Executes on button press in col_dwn.
function col_dwn_Callback(hObject, eventdata, handles)
% hObject    handle to col_dwn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(5),1);
pause(1)
putvalue(dio.line(5),0);
set(handles.col_pos,'String','Down');
guidata(hObject, handles);


% --- Executes on button press in turn_l.
function turn_l_Callback(hObject, eventdata, handles)
% hObject    handle to turn_l (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(7),1);
pause(1)
```

```
putvalue(dio.line(7),0);
set(handles.robot_pos,'String','Left');
guidata(hObject, handles);


% --- Executes on button press in turn_r.

function turn_r_Callback(hObject, eventdata, handles)
% hObject    handle to turn_r (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(8),1);
pause(1)
putvalue(dio.line(8),0);
set(handles.robot_pos,'String','Right');
guidata(hObject, handles);


% --- Executes on button press in grp_clk.
function grp_clk_Callback(hObject, eventdata, handles)
% hObject    handle to grp_clk (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(9),1);
pause(1)
putvalue(dio.line(9),0);
set(handles.angle_pos,'String','Clockwise');
guidata(hObject, handles);


% --- Executes on button press in grp_anti.
function grp_anti_Callback(hObject, eventdata, handles)
% hObject    handle to grp_anti (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
putvalue(dio.line(10),1);
pause(1)
putvalue(dio.line(10),0);
set(handles.angle_pos,'String','Anti-Clockwise');
guidata(hObject, handles);


% ---------------------------------------------------------------------
function Untitled_3_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in start_robot.
function start_robot_Callback(hObject, eventdata, handles)
% hObject    handle to start_robot (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = digitalio('mcc',0);
addline(dio,0:7,'out');
addline(dio,0:1,1,'Out');
handles.dio = dio;
putvalue(dio,0);
putvalue(dio.line([1 3 5 8 9]),[1 1 1 1 1]);
pause(1)
putvalue(dio,0);
set(handles.arm_pos,'String','In');
set(handles.grp_pos,'String','In');
set(handles.col_pos,'String','Down');
set(handles.robot_pos,'String','Right');
set(handles.angle_pos,'String','Clockwise');
set(handles.Manual,'Visible','on');
set(handles.Control_type,'Visible','on');
set(handles.Status,'Visible','on');
set(hObject,'Visible','off');
set(handles.stop_robot,'Visible','on');
guidata(hObject, handles);


% --- Executes on button press in stop_robot.
function stop_robot_Callback(hObject, eventdata, handles)
% hObject    handle to stop_robot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dio = handles.dio;
set(dio.line,'Direction','Out');
putvalue(dio,0);
delete(dio);
clear dio;
set(handles.arm_pos,'String','In');
set(handles.grp_pos,'String','In');
set(handles.col_pos,'String','Down');
set(handles.robot_pos,'String','Right');
set(handles.angle_pos,'String','Clockwise');
set(handles.Manual,'Visible','off');
set(handles.Automatic,'Visible','off');
set(handles.Control_type,'Visible','off');
set(handles.Status,'Visible','off');
set(handles.start_robot,'Visible','on');
set(handles.Man_control,'Value',1);
set(hObject,'Visible','off');
guidata(hObject, handles);


% --- Executes on button press in Man_control.
function Man_control_Callback(hObject, eventdata, handles)
% hObject    handle to Man_control (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Man_control
set(handles.Manual,'Visible','on');
```

```
set(handles.Automatic,'Visible','off');
guidata(hObject, handles);


% --- Executes on button press in Arm_in.
function Arm_in_Callback(hObject, eventdata, handles)
% hObject    handle to Arm_in (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 1;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Grp_in.
function Grp_in_Callback(hObject, eventdata, handles)
% hObject    handle to Grp_in (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 3;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Col_up.
function Col_up_Callback(hObject, eventdata, handles)
% hObject    handle to Col_up (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 6;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Turn_l.
function Turn_l_Callback(hObject, eventdata, handles)
% hObject    handle to Turn_l (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
```

```
data(row,1) = 7;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Grp_clk.
function Grp_clk_Callback(hObject, eventdata, handles)
% hObject    handle to Grp_clk (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 9;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Arm_out.
function Arm_out_Callback(hObject, eventdata, handles)
% hObject    handle to Arm_out (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 2;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Col_down.
function Col_down_Callback(hObject, eventdata, handles)
% hObject    handle to Col_down (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 5;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Turn_r.
function Turn_r_Callback(hObject, eventdata, handles)
% hObject    handle to Turn_r (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
```

```
row = row+1;
data(row,1) = 8;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Grp_anti.
function Grp_anti_Callback(hObject, eventdata, handles)
% hObject    handle to Grp_anti (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 10;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Grp_out.
function Grp_out_Callback(hObject, eventdata, handles)
% hObject    handle to Grp_out (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
row = row+1;
data(row,1) = 4;
set(handles.Table,'Data',data);
guidata(hObject, handles);


% --- Executes on button press in Auto_control.
function Auto_control_Callback(hObject, eventdata, handles)
% hObject    handle to Auto_control (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of Auto_control
set(handles.Manual,'Visible','off');
set(handles.Automatic,'Visible','on');
guidata(hObject, handles);



% --- Executes on button press in Reset.
function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Table,'Data',[]);
guidata(hObject, handles);
```

```
% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text1,'String','Robot Status');
dio = handles.dio;
set(dio.line,'Direction','Out');
set(handles.Control_type,'Visible','off')
cycle = str2num(get(handles.Cycle,'String'));
data = get(handles.Table,'Data');
datasize = size(data);
row = datasize(1);
if row>=1
        f = warndlg('Press Ok for Emergency Stop.', 'Stop');
    if cycle ~= 0
    for y=1:cycle
        if ~ishandle(f)
        %set(handles.text5,'String','Emergency Stop');
        %pause(1)
        break
        end
        for x=1:row
               if ~ishandle(f)
                  %set(handles.text5,'String','Emergency Stop');
                  %pause(1)
                  break
               elseif data(x,1) == 1
                  putvalue(dio.line(1),1);
                  pause(1)
                  putvalue(dio.line(1),0);
                  set(dio.line(1),'Direction','In');
                  while(ishandle(f) && getvalue(dio.line(1))==1)
                  pause(0.1)
                  end

                  set(dio.line(1),'Direction','Out');
                  set(handles.arm_pos,'String','In');
               elseif data(x,1) == 2
                  putvalue(dio.line(2),1);
                  pause(1)
                  putvalue(dio.line(2),0);
                  set(dio.line(2),'Direction','In');
                  while(getvalue(dio.line(2))==1)
                  pause(0.1)
                  end
                  set(dio.line(2),'Direction','Out');
                  set(handles.arm_pos,'String','Out');
               elseif data(x,1) == 3
                  putvalue(dio.line(3),1);
                  pause(1)
                  putvalue(dio.line(3),0);
                  set(dio.line(3),'Direction','In');
                  while(getvalue(dio.line(3))==1)
                  pause(0.1)
```

```
    end
    set(dio.line(3),'Direction','Out');
    set(handles.grp_pos,'String','In');
elseif data(x,1) == 4
    putvalue(dio.line(4),1);
    pause(1)
    putvalue(dio.line(4),0);
    set(dio.line(4),'Direction','In');
    while(getvalue(dio.line(4))==1)
    pause(0.1)
    end
    set(dio.line(4),'Direction','Out');
    set(handles.grp_pos,'String','Out');
elseif data(x,1) == 5
    putvalue(dio.line(5),1);
    pause(1)
    putvalue(dio.line(5),0);
    set(dio.line(5),'Direction','In');
    while(getvalue(dio.line(5))==1)
    pause(0.1)
    end
    set(dio.line(5),'Direction','Out');
    set(handles.col_pos,'String','Down');
elseif data(x,1) == 6
    putvalue(dio.line(6),1);
    pause(1)
    putvalue(dio.line(6),0);
    set(dio.line(6),'Direction','In');
    while(getvalue(dio.line(6))==1)
    pause(0.1)
    end
    set(dio.line(6),'Direction','Out');
    set(handles.col_pos,'String','Up');
elseif data(x,1) == 7
    putvalue(dio.line(7),1);
    pause(1)
    putvalue(dio.line(7),0);
    set(dio.line(7),'Direction','In');
    while(getvalue(dio.line(7))==1)
    pause(0.1)
    end
    set(dio.line(7),'Direction','Out');
    set(handles.robot_pos,'String','Left');
elseif data(x,1) == 8
    putvalue(dio.line(8),1);
    pause(1)
    putvalue(dio.line(8),0);
    set(dio.line(8),'Direction','In');
    while(getvalue(dio.line(8))==1)
    pause(0.1)
    end
    set(dio.line(8),'Direction','Out');
    set(handles.robot_pos,'String','Right');
elseif data(x,1) == 9
    putvalue(dio.line(9),1);
    pause(1)
    set(dio.line(9),'Direction','In');
```

```
                    while(getvalue(dio.line(9))==1)
                    pause(0.1)
                    end
                    set(dio.line(9),'Direction','Out');
                    set(handles.angle_pos,'String','Clockwise');
                elseif data(x,1) == 10
                    putvalue(dio.line(10),1);
                    pause(1)
                    set(dio.line(10),'Direction','In');
                    while(getvalue(dio.line(10))==1)
                    pause(0.1)
                    end
                    set(dio.line(10),'Direction','Out');
                    set(handles.angle_pos,'String','AntiClockwise');

            end



        end

        end
        %set(handles.text5,'String','Robot Stopped');
        if ishandle(f)
        delete(f)
        end

    else
        while true
            if ~ishandle(f)

              break
            end
            for x=1:row
                    if ~ishandle(f)
                    %set(handles.text5,'String','Emergency Stop');
                    %pause(1)
                    break
                elseif data(x,1) == 1
                    putvalue(dio.line(1),1);
                    pause(1)
                    putvalue(dio.line(1),0);
                    set(dio.line(1),'Direction','In');
                    while(getvalue(dio.line(1))==1)
                    pause(0.1)
                    end
                    set(dio.line(1),'Direction','Out');
                    set(handles.arm_pos,'String','In');
                elseif data(x,1) == 2
                    putvalue(dio.line(2),1);
                    pause(1)
                    putvalue(dio.line(2),0);
                    set(dio.line(2),'Direction','In');
                    while(getvalue(dio.line(2))==1)
                    pause(0.1)
```

```
    end
    set(dio.line(2),'Direction','Out');
    set(handles.arm_pos,'String','Out');
elseif data(x,1) == 3
    putvalue(dio.line(3),1);
    pause(1)
    putvalue(dio.line(3),0);
    set(dio.line(3),'Direction','In');
    while(getvalue(dio.line(3))==1)
    pause(0.1)
    end
    set(dio.line(3),'Direction','Out');
    set(handles.grp_pos,'String','In');
elseif data(x,1) == 4
    putvalue(dio.line(4),1);
    pause(1)
    putvalue(dio.line(4),0);
    set(dio.line(4),'Direction','In');
    while(getvalue(dio.line(4))==1)
    pause(0.1)
    end
    set(dio.line(4),'Direction','Out');
    set(handles.grp_pos,'String','Out');
elseif data(x,1) == 5
    putvalue(dio.line(5),1);
    pause(1)
    putvalue(dio.line(5),0);
    set(dio.line(5),'Direction','In');
    while(getvalue(dio.line(5))==1)
    pause(0.1)
    end
    set(dio.line(5),'Direction','Out');
    set(handles.col_pos,'String','Down');
elseif data(x,1) == 6
    putvalue(dio.line(6),1);
    pause(1)
    putvalue(dio.line(6),0);
    set(dio.line(6),'Direction','In');
    while(getvalue(dio.line(6))==1)
    pause(0.1)
    end
    set(dio.line(6),'Direction','Out');
    set(handles.col_pos,'String','Up');
elseif data(x,1) == 7
    putvalue(dio.line(7),1);
    pause(1)
    putvalue(dio.line(7),0);
    set(dio.line(7),'Direction','In');
    while(getvalue(dio.line(7))==1)
    pause(0.1)
    end
    set(dio.line(7),'Direction','Out');
    set(handles.robot_pos,'String','Left');
elseif data(x,1) == 8
    putvalue(dio.line(8),1);
    pause(1)
    putvalue(dio.line(8),0);
```

```
                    set(dio.line(8),'Direction','In');
                    while(getvalue(dio.line(8))==1)
                    pause(0.1)
                    end
                    set(dio.line(8),'Direction','Out');
                    set(handles.robot_pos,'String','Right');
                elseif data(x,1) == 9
                    putvalue(dio.line(9),1);
                    pause(1)
                    putvalue(dio.line(9),0);
                    set(dio.line(9),'Direction','In');
                    while(getvalue(dio.line(9))==1)
                    pause(0.1)
                    end
                    set(dio.line(9),'Direction','Out');
                    set(handles.angle_pos,'String','Clockwise');
                elseif data(x,1) == 10
                    putvalue(dio.line(10),1);
                    pause(1)
                    putvalue(dio.line(10),0);
                    set(dio.line(10),'Direction','In');
                    while(getvalue(dio.line(10))==1)
                    pause(0.1)
                    end
                    set(dio.line(10),'Direction','Out');
                    set(handles.angle_pos,'String','AntiClockwise');
                end


        end
    end
    %set(handles.text5,'String','Robot Stopped');
    end

else
    set(handles.text1,'String','Input Robot Command');
end
set(dio.line,'Direction','Out');
set(handles.Control_type,'Visible','on');
set(dio.line,'Direction','Out');
putvalue(dio,0);
guidata(hObject, handles);



function Cycle_Callback(hObject, eventdata, handles)
% hObject    handle to Cycle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Cycle as text
%        str2double(get(hObject,'String')) returns contents of Cycle as a
double
```

```
% --- Executes during object creation, after setting all properties.
function Cycle_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Cycle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

APPENDIX B: DATASHEETS