DESIGN AND DEVELOPEMENT OF ENCODER SENSOR WITH GRAPHICAL USER INTERFACE (GUI)

IHSAN BIN AHMAD ZUBIR

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **DESIGN AND DEVELOPEMENT OF ENCODER SENSOR WITH GRAPHICAL USER INTERFACE (GUI)**

SESI PENGAJIAN: **2007/2008**

Saya **IHSAN BIN AHMAD ZUBIR (851103-08-5947)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Kolej Universiti Kejuruteraan & Teknologi Malaysia.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

[ ]	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
[ ]	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
[√]	**TIDAK TERHAD**	

Disahkan oleh:

(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

Alamat Tetap:

NO 4 KAMPUNG BANJIR,
JALAN TAAYAH,
33000 KUALA KANGSAR,
PERAK DARUL RIDZUAN.

MUHAMMAD SHARFI BIN NAJIB
(Nama Penyelia)

Tarikh: **30 NOVEMBER 2007**

Tarikh: : **30 NOVEMBER 2007**

CATATAN:
- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan

"I hereby declare that I have read this thesis and in
My opinion this thesis is sufficient in terms of scope and
Quality for the award of the Degree of
Bachelor of Electrical & Electronic Engineering"

Signature : .....

Supervisor : Mr. Muhammad Sharfi bin Najib

Date : 30 November 2007

DESIGN AND DEVELOPEMENT OF ENCODER SENSOR WITH GRAPHICAL USER INTERFACE (GUI)

IHSAN BIN AHMAD ZUBIR

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor Degree of Electrical Engineering (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER, 2007

Signature : _____

Author : IHSAN BIN AHMAD ZUBIR

Date : 30 November 2007

Dedicate to my beloved family and friends
who always give me a courage to finish this thesis.

Also, to those people who have been supportive through all this time.
Thank you for the kindness and advices that have been given.

God bless you all,-amin-

ACKNOWLEDGEMENT

Alhamdulillah, I'm grateful to the creator Allah S.W.T because of His regards I finally finish this final year project. Without His blessing it is difficult for me to overcome and face all problems while completing this project. I also would like to express thousand of thank to my supervisor, Mr. Muhammad Sharfi bin Najib who give highly encouragement, supporting and guideline in order to finish this task.

Not forgetting to my beloved parents that always prays for me and give me strength with unlimited effort. They always remind and give lots of motivation about patient and ask me to never give up. Thank you mum and my father, may Allah bless you always.

Beside that, thank you very much to my entire friend who always shares ideas and co-operation in order to finish this project. I wish you all best of luck.

Lastly, thank you for those who are involved directly or indirectly and your co-operation will never be forgotten.

Thank you

Ihsan bin Ahmad Zubir

ABSTRACT

MATLAB is one of the software that can create Graphical User Interface (GUI). The GUI will be designed using guide to create the layout editor. The objective of this project is to display the signal that is generated by encoder sensor in GUI. There are three phases to develop this project. Phase one is a development of the sensor circuit to produce signal in analogue form. The next phase is a construct of the controller circuit in order to convert the analogue signal to digital signal. In controller circuit it also used Intergrated Circuit (IC) MAX233. The function of the IC is to transmit or retrive the data from external device to the computer or vice versa and at the same time it remains the signal in stable condition by amplified the signal in order to reduce the losses and noise. At computer, DB9 used to connect with the port in computer. The changes of the signal form from analogue signal to digital signal are necessary because the computer only can accept the data in digital form. The final phase is a development of the GUI in MATLAB. In the end of this project, the signal generated by encoder sensor displayed in GUI.

ABSTRAK

MATLAB adalah satu perisian yang boleh mencipta GUI. GUI akan dicipta menggunakan 'guide' untuk membuat 'layout editor'. Objektif projek ini adalah untuk mempamerkan isyarat yang dijanakan oleh 'encoder sensor'. Terdapat tiga fasa untuk membangunkan projek ini. Fasa pertama adalah membangunkan litar pengesan untuk menghasilkan isyarat yang man isyarat tersebut ialah isyarat analog. Seterusnya ialah membuat litar kawalan untuk menukar isyarat analog kepada isyarat digital. Dalam litar kawalan tersebut ia juga menggunakan litar bersepadu MAX233. Fungsi litar bersepadu tersebut adalah untuk hantar atau terima data daripada perkakasan luar ke komputer atau sebaliknya dan pada masa yang sama ia akan kekalkan signal dalam keadaan stabil dengan menguatkan isyarat bertujuan untuk mengurangkan kelesapan atau gangguan. Di komputer DB9 telah digunakan untuk hubungkan perkakasan luar dengan port dalam komputer. Perubahan bentuk isyarat daripada isyarat analog kepada isyarat digital adalah perlu kerana komputer hanya boleh terima data dalam isyarat digital sahaja. Fasa yang terakhir ialah membuat GUI dalam MATLAB. Pada akhir projek ini, isyarat yang dihasilkan oleh 'encoder sensor' boleh dipamerkan dalam GUI.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GUI	Graphical Users Interafce
MCU	Microcontroller Unit

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.0 Background

This project scope is to design and develop encoder sensor using microcontroller Motorola 6811 through MATLAB GUI. The main contribution is the interfacing of the MATLAB with microcontroller, GUI and Encoder Sensor. This project focuses on the measurement of the signal that generate by encoder sensor. The signal is in digital because the encoder sensor generate digital signal, it will make easy to us to see the result and also can avoid from noise.

The GUI will make easy to us to see the graph in form of graphic. GUI developed using MATLAB software because it is suitable with mathematic equation.

This project will help the students to check or trouble shooting their circuit using this software in their rooms. The cost to make this project is low than the price of the oscilloscope, the most important things we can determine and check the signal, same as the function of the oscilloscope. Besides that, the maintenance cost of oscilloscope is very high.

This project is suitable for education purpose. The students can use this software to see the signal and measure the signal easily.

1.1 Problem Statement

This project was done because it is user friendly especially for student to make analysis in computer as compared to oscilloscope which price is more expensive. However, the future of this project until recently seemed unimpressive in find the correct coding to create GUI.

It will become more complicated when GUI involves with complicated analysis and mathematical formula. Here, the problem is to create the new coding need more research towards GUI.

The other problem is to interface the GUI with microcontroller. The selection of microcontroller is quite difficult in order to find suitable microcontroller. Besides that, measuring Matlab using command window is not as user friendly as GUI.

1.2 Objective

The aim of this project is to display the actual signal that generate by encoder sensor in GUI. The signal will display in digital signal.

Basically a graph will be displaying as an electrical signal. In most applications the graph shows how signals change over time: the vertical (Y) axis represents voltage and the horizontal (X) axis represents time.

The main objective of this project is to plot the actual signal that generate by encoder sensor.

1.3 Scope of Project

This project is design to measure and displays the signal in GUI. The screen must display the digital signal. The signal can be adjust by adjust the scale of voltage per division or second per division. The function of voltage per division or second per division is easily to get the accurate value.

Besides, it is necessary to interface of MATLAB with microcontroller, GUI and Encoder Sensor. This project will use microcontroller as a connecter between software and hardware. By using microcontroller it will make easy to program and the type of microcontroller that will be used in this project is microcontroller Motorola 6811. Most importantly, this project is to fulfill two scopes that are display the signal on GUI and also can interface between GUI and microcontroller.

CHAPTER 2

LITERATURE REVIEW

2.0 Introduction

In order to perform this project, literature review have been made from various sources likewise journal, books and other references such as article. In simple term, the reference sources emphasize on few aspects and the important aspect is the assembly mechanism analysis and how to design and develop encoder sensor, create GUI using MATLAB and construct microcontroller circuit. This chapter will describe about GUI, MATLAB GUI, creating GUIs with GUIDE, Sensor and Analog sensor, Encoder, microcontroller Motorola 6811 and Oscilloscope.

2.1 Graphical User Interface (GUI)

A GUI is a human-computer interface that uses windows, icons and menus and which can be manipulated by a mouse and a good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. [1][2][3] In medical Simulink (MATLAB), it will compare the process of implementing a Pharmaco-kinetic/ Pharmaco-dinamic model of a biological system in traditional package. During simulation, a scope block automatically produces a time-course of the concentration of drug in the model compartments over a specified period.[2]

A window is a (usually) rectangular portion of the monitor screen that can display its contents (e.g., a program, icons, a text file or an image) seemingly independently of the rest of the display screen. A major feature is the ability for multiple windows to be open simultaneously. Each window can display a different application, or each can display different files (e.g., text, image or spreadsheet files) that have been opened or created with a single application. The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders -- just to name a few. In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components. [1][2]

2.2 MATLAB GUI

The following Graphical User Interface MATLAB programs have been developed for the computational aids in the electrical engineering topics outlined in the menu at left. These GUI programs with point-and-click features are designed for ease of use. These programs together with the traditional hand-written problems can help students to develop a stronger intuition and deeper understanding of these topics.[4]

The following aims to present some code which is repeatedly used and establishes what is expected from objects. The common code as below:

1. Radio Buttons

Where a group of buttons is inter-related, the following code should do:

```
set(handles.option1, 'Value', 1);
set(handles.option2, 'Value', 0);
set(handles.option3, 'Value', 0);
```

where *option1* is the source of the callback. For the other options, the only part which needs changing is the latter one where the state of the button is defined by a zero or a one.[5]

2. Check Boxes

A naive yet useful implementation of those will involve an element *state* which hold some information about the state of the checkbox or its meaning.

```
if (get(handles.state,'Value') == 0),
set(handles.checkbox, 'Value', 0);
set(handles.state, 'String', '0');
else
set(handles.checkbox, 'Value', 1);
set(handles.state, 'String', '1');
end [5]
```

3. Drop-down Menus

In the following, the menu object needs to first be identified using:

```
contents = get(hObject,'String');
```

Subsequently, the menu entry needs to be checked for extraction and setting of information.

```
if (strcmp(contents{get(hObject,'Value')},'MenuEntry1')),
set(handles.data,'String','Data1');
elseif (strcmp(contents{get(hObject,'Value')},'MenuEntry2')),
set(handles.data,'String','Data2');
else
msgbox('Error with menu callback. Parameter passed is not recognised.');
end[5]
```

4. Sliders

The following code will fetch the quantised value of the slider and assign it to a new object called *slider_value*.set(handles.slider_value, 'String', num2str(ceil(get(handles.slider,)

For items that need to be ticked and unticked, the following can be useful.

```
set(handles.menuitem1,'Checked','off');
set(handles.menuitem2,'Checked','off');
set(handles.menuitem3,'Checked','off');
set(handles.menuitem4,'Checked','on');
set(handles.menu_selection,'String','item4') [5]
```

For GUI editing, more efficient work on the code can be done by opening all relevant files in advance. I personally write M-files to open all relevant windows at the start. Set up a file which includes the following lines:

```
edit <m-file1> <m-file2>...
guide <fig-file1> <fig-file2>...
```

where of course the files listed are these which are frequently worked upon. [5]

MATLAB GUI is a very powerful tool when used correctly. It takes a lot of experimenting with and a good background in programming. We also must have a good understanding of MATLAB and be able to use the MATLAB language and commands to create routines.

2.3 Creating GUIs with GUIDE

MATLAB implements GUIs as figure windows containing various uicontrol objects. You must program each object to perform the action you intend it to do when a user activates the component. In addition, you must be able to save and run your GUI. All of these tasks are simplified by GUIDE, the MATLAB graphical user interface development environment.[9]

GUIDE, the MATLAB Graphical User Interface development environment, provides a set of tools for creating GUIs. These tools greatly simplify the process of laying out and programming a GUI. GUIDE is displayed when GUI is opened in the Layout Editor, which is the control panel for all of the GUIDE tools. The Layout Editor will enables to lay out a GUI quickly and easily by dragging components, such as push buttons, pop-up menus, or axes, from the component palette into the layout area. The following picture shows the Layout Editor.[9]

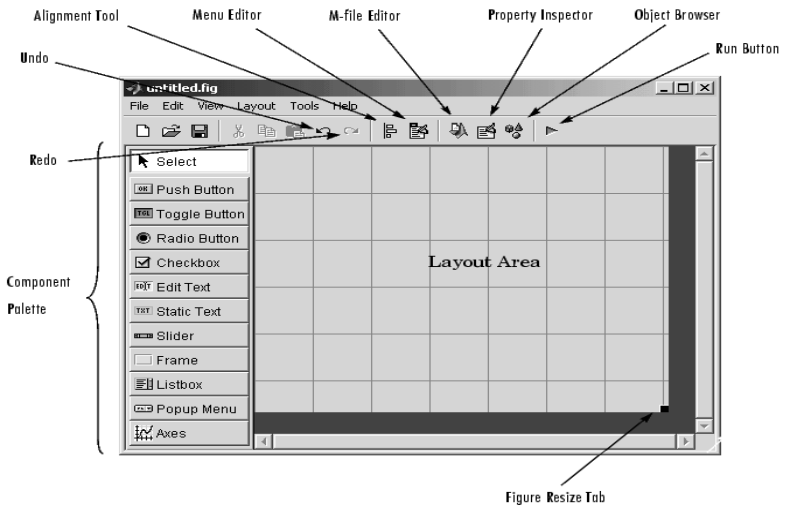


Figure 2.1: Graphical User Interface (GUI) layout

Once GUI lay outed of and set each component's properties, using the tools in the Layout Editor, GUI can be programmed with the M-file Editor. Finally, press the Run button on the toolbar, the functioning GUI appears outside the Layout to see the result.

GUI Development Environment

Creating a GUI involves two basic tasks. They are laying out the GUI components and programming the GUI components. GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the *callbacks* -- the functions that execute when users activate components in the GUI.[9]

GUIDE Generated Files

While it is possible to write an M-file that contains all the commands to lay out a GUI, it is much easier to use GUIDE to lay out the components interactively. When you save or run the GUI, GUIDE automatically generates two files:

- A FIG-file -- a file with a .fig file name extension, which contains a complete description of the GUI figure and all of its children (uicontrols and axes), as well as the values of all object properties. You make changes to the FIG-file by editing the GUI in the Layout Editor.
- An M-file -- a file with a .m file name extension, which contains the functions that run and control the GUI and the callbacks. This file is referred to as the *GUI M-file*.
-

Note that the M-file does not contain the code that lays out the uicontrols; this information is saved in the FIG-file.[9]

Features of the GUI M-file

GUIDE simplifies the process of creating GUIs by automatically generating the GUI M-file directly from your layout. GUIDE generates callbacks for each component in the GUI that requires a callback. Initially, GUIDE generates just a function definition line for each callback. You can add code to the callback to make it perform the operation you want. The M-file contains two other functions where you might also need to add code:

- Opening function -- performs tasks before the GUI becomes visible to the user, such as creating data for the GUI. GUIDE names this function my_gui_OpeningFcn, where my_gui is the name of the GUI.
- Output function -- outputs variables to the command line, if necessary. GUIDE names this function my_gui_OutputFcn, where my_gui is the name of the GUI.[9]

2.4 Sensor and Analog sensor

Sensors translate between the physical world and the abstract world of Microcontrollers. Sensors help translate physical world attributes into values that the computer on a robot can use. The translation produces some sort of output value that the Microcontroller can use. In general, most sensors fall into one of two categories.They are analog sensor and digital sensor.[7]

An analog sensor, such as a CdS cell (Cadmium Sulfide cells measure light intensity), might be wired into a circuit in a way that it will have an output that ranges from 0 volts to 5 volts. The value can assume any possible value between 0 and 5 volts. An 'Analog Signal' is one that can assume any value in a range. An interesting way to think about this is an Analog Signal works like a tuner on an older radio. You can turn it up or down in a continuous motion. You can fine tune it by turning the knob ever so slightly.[7]

Remember, to successfully use an Analog sensor, some way are needed to convert the data into a digital form. All of the circuits shown in this section are intended to be connected to a A/D converter port. Many Microcontrollers, such as the 68HC11, have A/D ports built in. Others require that you add an additional support chip, such as the ADC805 or other equivalent chip.[7]

Cadmium-Sulfide is an interesting compound. Its resistance changes readily when exposed to light energy. Typically, the more light, the lower the resistance. This is useful for measuring the intensity of light.[7]

2.5 Encoder

An encoder comes in two architectures. The first architecture is linear. The second architecture is rotary. Both types sense mechanical motion and translate the information (velocity, position, acceleration) into useful electrical data. Besides that, an encoder defines as a device used to change a signal (such as a bitstream) or data into a code. The code may serve any of a number of purposes such as compressing information for transmission or storage, encrypting or adding redundancies to the input code, or translating from one code to another. This is usually done by means of a programmed algorithm, especially if any part is digital, while most analog encoding is done with analog circuitry. [6] [7]

There are a few subtle differences between absolute and incremental rotary encoders. Incremental encoders have output signals that repeat over the full range of motion. It is important to understand that each mechanical position is not uniquely defined. When the incremental encoder is turned on, the position of an incremental encoder is not known since the output signals are not unique to any singular position. Absolute encoders have a unique value (voltage, binary count, etc) for each mechanical position. When an absolute encoder is turned on, the position of an absolute encoder is known (this function resembles a resolver, although the principles of operation have no similarity.) The similarities of both absolute and incremental encoders are form factor and the issues of count and directional

information. They can be obtained from both absolute and incremental encoders equally.[6]

2.6 Microcontroller Motorola 6811

Figure 2.2 : The MC16F877

The microcontroller MC68HC11A1 is a high performance 8-bit microcontroller units (MCUs) base on the MC68HC11 family. It uses the HCMOS technology to produce faster and small controller with less power consumption and high tolerance for noisy signal. These high speed, low power consumption chips have multiplexed buses and a fully static design. The chips can operate at frequencies from 3 MHz to dc.[9][10]

This microcontroller offers a lot of features than other microcontroller in MC68HC11 family. It has its own CPU , power saving stop and wait modes features, 8 Kbytes ROM, 512 Bytes on-chip EEPROM , 256 bytes of on-chip RAM, 16 bit timer system, 8 bit pulse accumulator, real time interrupt circuit, COP watch dog system, synchronous serial peripheral interface (SCI), asynchronous no return to zero SCI, 8-channel, 8 bit ADC and 38 general purpose I/O pins.[10]

In MC68HC11, the programming used in this microcontroller is assembly language. This language is upward compatible, means that the latest version of MC68HC11 family can run program from the old version of MC68HC11 family but the old version of MC68HC11 family cannot run program from the latest version MC68HC11 family.[9]

Figure 2.3 : 48-pin DIP pin assignment

MC68HC11 have 4 modes operation, bootstrap mode, special test mode, expanded-multiplexed mode and single-chip mode. In bootstrap mode, all the program are place into the RAM, whereas, special test is used by manufacturer to test the chip in factory, so this mode only can be used manufacturer. Single chip operation by using internal memories and expanded-multiplexed is mode where users can expand memory and I/O lines by uses the port B and port C as an address and data buses. All mode of operation is determine by status MODA and MODB pins during RESET operation.[9][10]

Ports in MC68HC11 are multiplexed; that is each port offers various function with each port perform one task at one time.

Table 2.0: Port and Function for 68HC11

PORT	FUNCTION
A	Parallel I/O or timer/counter
B	Output port or upper address (A8-A15) in expanded mode.
C	I/O port or lower address (A0-A7) and data bus (D0-D7) in expanded mode.
D	6 bits I/O port or serial communication interface (SCI) and serial peripheral interface (SPI)
E	Input port or 8-cahnnels input analog for ADC

Latest version of MC68HC11 family such as version F has port F and G and low cost version such as D version only has port A, B, C and D.[8]

2.7 Oscilloscope

An oscilloscope (sometimes abbreviated CRO, for cathode-ray oscilloscope, or commonly just scope or O-scope) is a piece of electronic test equipment that allows signal voltages to be viewed, usually as a two-dimensional graph of one or more electrical potential differences (vertical axis) plotted as a function of time or of some other voltage (horizontal axis).[11]

One of the most frequent uses of scopes is troubleshooting malfunctioning electronic equipment. One of the advantages of a scope is that it can graphically show signals: where a voltmeter may show a totally unexpected voltage, a scope may reveal that the circuit is oscillating. In other cases the precise shape of a pulse is important.

In a piece of electronic equipment, for example, the connections between stages (e.g. electronic mixers, electronic oscillators, amplifiers) may be 'probed' for the expected signal, using the scope as a simple signal tracer. If the expected signal is absent or incorrect, some preceding stage of the electronics is not operating correctly. Since most failures occur because of a single faulty component, each measurement

can prove that half of the stages of a complex piece of equipment either work, or probably did not cause the fault. [11]

Once the faulty stage is found, further probing can usually tell a skilled technician exactly which component has failed. Once the component is replaced, the unit can be restored to service, or at least the next fault can be isolated. [11]

Another use is to check newly designed circuitry. Very often a newly designed circuit will misbehave because of design errors, bad voltage levels, electrical noise etc. Digital electronics usually operate from a clock, so a dual-trace scope which shows both the clock signal and a test signal dependent upon the clock is useful. "Storage scopes" are helpful for "capturing" rare electronic events that cause defective operation. [11]

Another use is for software engineers who must program electronics. Often a scope is the only way to see if the software is running the electronics properly. [11]

CHAPTER 3

METHODOLOGY

3.0 Introduction

In developing this project, methodologies is one of the most important element to be consider to make sure that the development of the project is smooth and get the expected result. A good methodologies can described the structure or the flow of the project where by it can be the guideline in managing it. It is also to avoid the project to alter course from the objectives that have been stated or in other words the project follow the guideline based on the objectives. Figure 3.1 shows the flow chart of methodology of this project. Below is the step to develop this project:

i. Developing hardware
ii. Developing software

3.1 Methodology of work flow

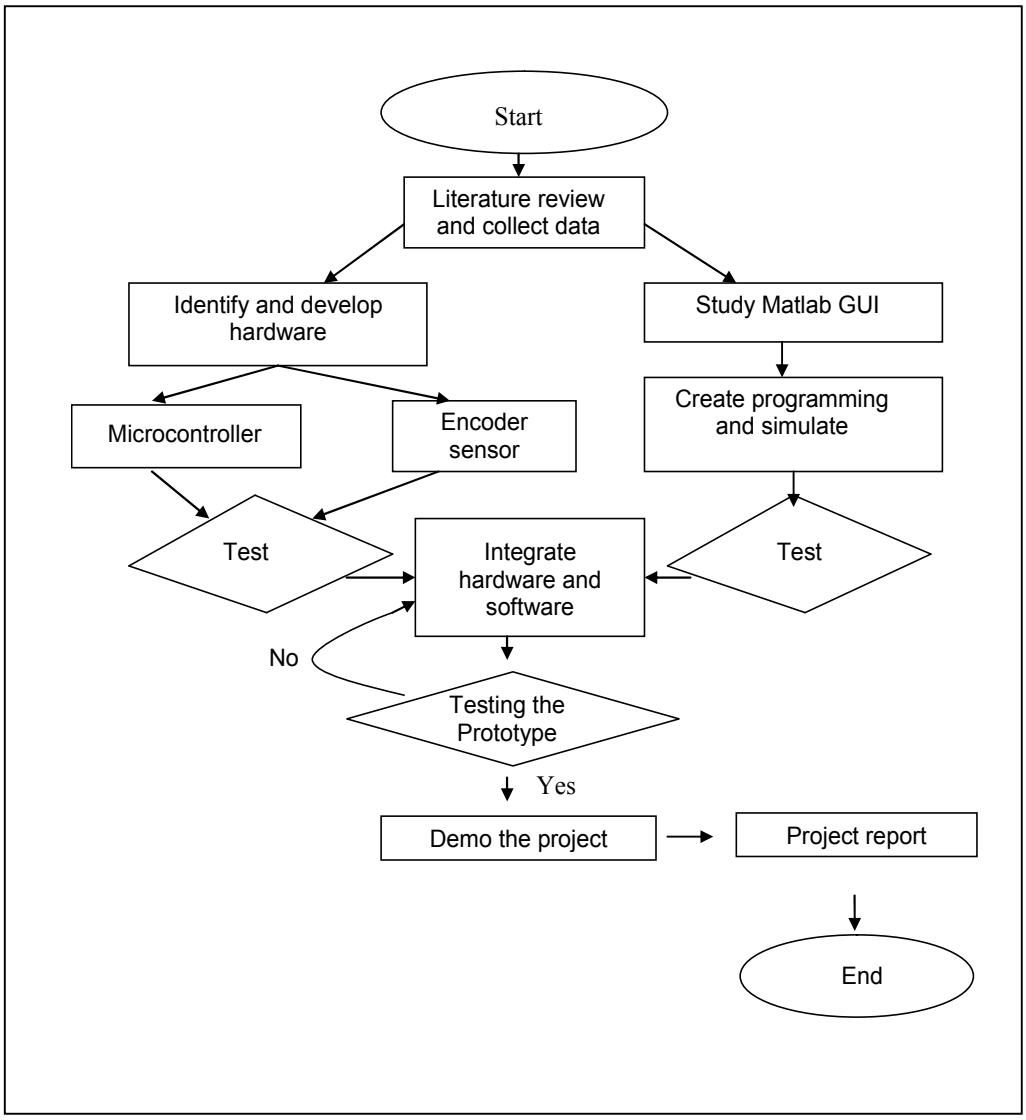


Figure 3.1: Flowchart of project operation

The block diagram above is the method to execute this project. First determine the title. Than, collect data by make some research and literature review. This method divided into two sections: develop of hardware and create programming. For hardware, there are three circuit will develop. They are encoder sensor circuit and microcontroller Motorola 6811 circuit, at the same time develop the programming and then both of them, hardware and the software will test. If the testing is successful, both of them will integrate and then test and demo, and also submit the report.

3.2 Research methodology

Based on the research of this project, it is suitable to design GUI using MATLAB. It is because this project uses more calculation programming, it is not suitable to use visual basic because it is suitable for simple mathematical calculation. This project will use microcontroller Motorola 6811 to interface between hard ware and software, GUI. In this project signal will generate by encoder sensor. Encoder sensor consists of transmitter and receiver circuit. The signal from transmitter circuit will transmit the signal to receiver circuit, the opaque object will cut the signal before it get into microcontroller and then go through the MAX 233 before it display on GUI.

3.3 Developing Hardware

3.3.0 Introduction

There are a lot of components used in developing this project. The main components are microcontroller Motorola 6811 as an interface medium between hardware and software GUI , infrared transmitter and receiver as a sensor to create signal and IC MAX233 will functioning as a connector between microcontroller and computer. The signal will generate by encoder sensor than, the signal will display on GUI through Microcontroller Motorola 6811.

3.3.1 Design and development of MC6811 circuit

Basically, microcontroller unit has internal CPU, memory and registers. Externally, it has pins for I/O and bus signals. The I/O pins are grouped in sets of eight called ports. For the references see figure 3.2 where it shows the pin assignments. The MC68HC11 can be operated in different modes. They are single chip, expanded, boot-strap, and special test.

MODB	MODA	Mode Selected
1	0	Single Chip
1	1	Expanded Multiplexed
0	0	Special Bootstrap
0	1	Special Test

Figure 3.2: Pin assignments

The mode selected is determined by how pins MODA and MODB are connected at the time of reset. To set it become bootstrap mode, both pins MODA and MODB must be grounded to get logic '0'. The bootstrap mode is considered a special operating mode as distinguished from the normal single-chip operating mode.

This is a very versatile operating mode since there are essentially no limitations on the special purpose program that can be loaded into the internal RAM.

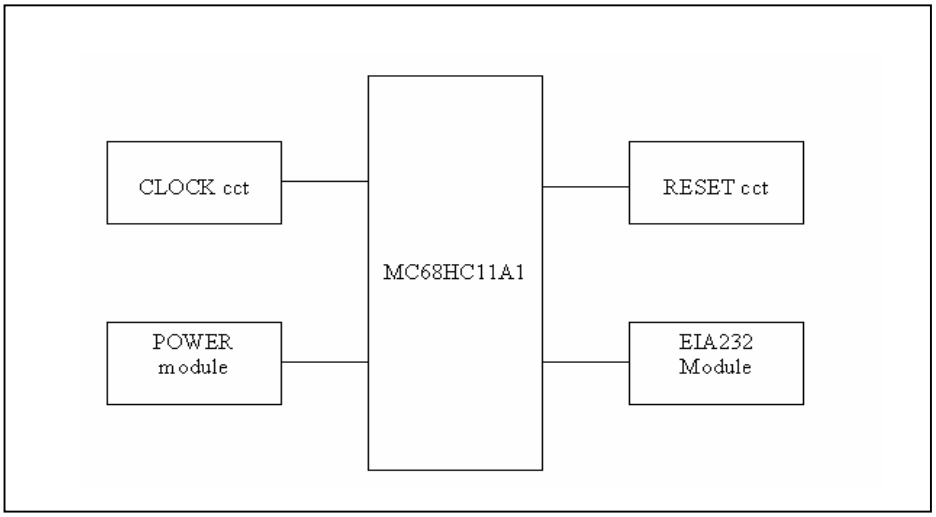


Figure 3.3: A simplified block diagram of MC68HC11-based system

3.3.2.0 Power Supply Module

MC68HC11 is must be design based on it specification, this is important to ensure the system could operate properly and more important is to avoid permanent damage to the microcontroller. Thus, the main purpose of power supply module is to be as power source to the microcontroller which fulfills the criteria of the MC68HC11. The figure 3.4 illustrates the circuitry for the power supply module. This circuit functions to supply dc source at fixed voltage level at 5V and avoid the over current from entering microcontroller. The condition could be achieved by using IC regulator 7805 which provide fixed 5V although the load is changed.

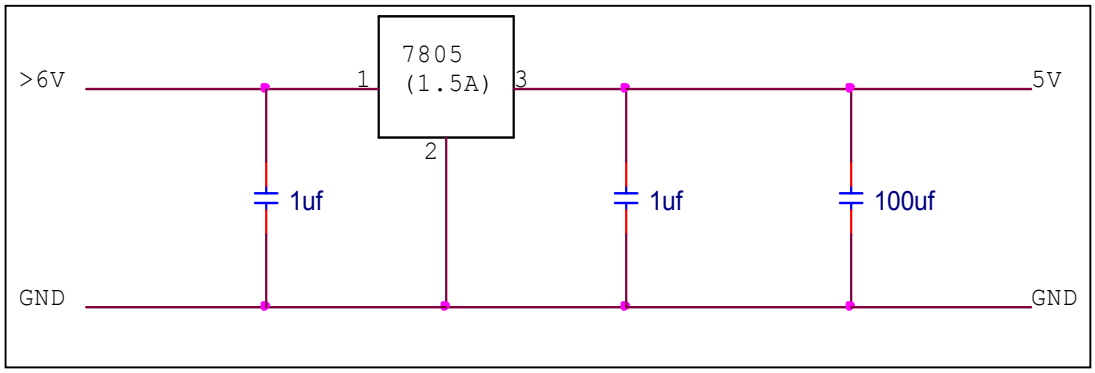


Figure 3.4: Power supply module

3.3.2.1 Crystal Driver and External Clock Input (XTAL, EXTAL)

These two pins provide the interface for either a crystal or a CMOS compatible clock to control the internal clock generator circuitry. The frequency applied to these pins shall be four times higher than the desired E clock rate. The XTAL pin is normally left exterminated when using an external CMOS compatible clock input to the EXTAL pin. However, a 10K to 100K load resistor to ground may be used to reduce RFI noise emission. The XTAL output is normally intended to drive only a crystal. The XTAL output may be buffered with a high-input-impedance buffer such as the 74HC04, or it may be used to drive the EXTAL input of another M68HC11. In all cases take extra care in the circuit board layout around the oscillator pins. Load capacitances shown in the oscillator circuits include all stray layout capacitances.

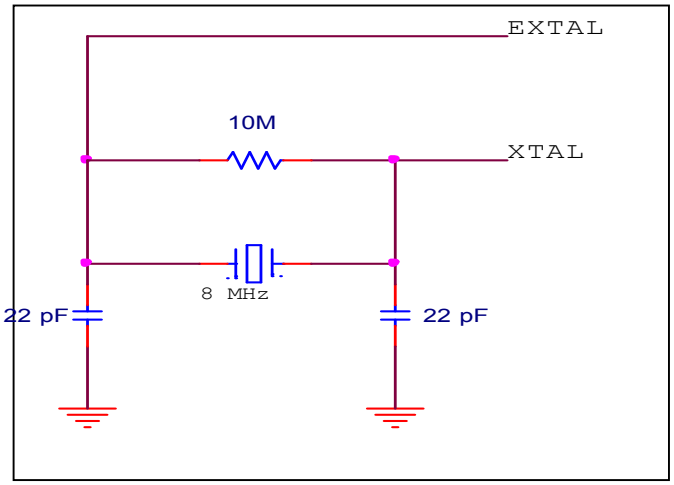


Figure 3.5: Clock circuit

3.3.2.2 Reset (RESET)

This active low bidirectional control signal is used as an input to initialize the MC68HC11A1/A8 to a known start-up state, and as an open-drain output to indicate that an internal failure has been detected in either the clock monitor or computer operating properly (COP) watchdog circuit. This reset signal is significantly different from the reset signal used on other Motorola MCUs.

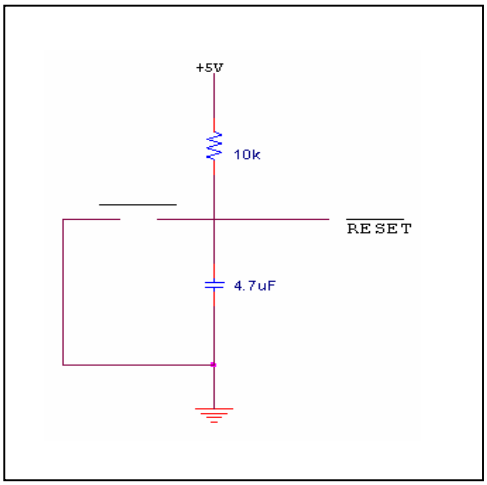


Figure 3.6: A reset circuit

3.3.2.3 MAX233

The function of MAX233 is to amplify signal in order to reduce the losses during transmit or receive data from Microcontroller 6811 to the PC.

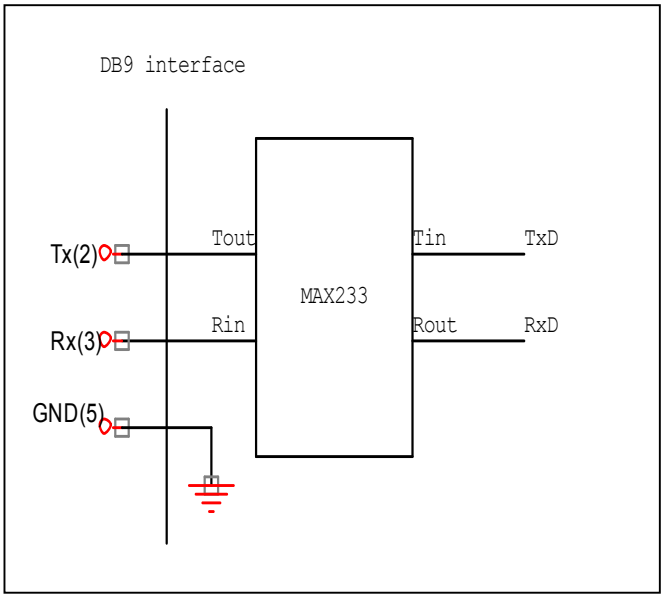


Figure 3.7: (EIA 232 MODULE) or MAX233 Interface

3.3.2.4 COMPLETE CIRCUIT MICROCONTROLLER 6811

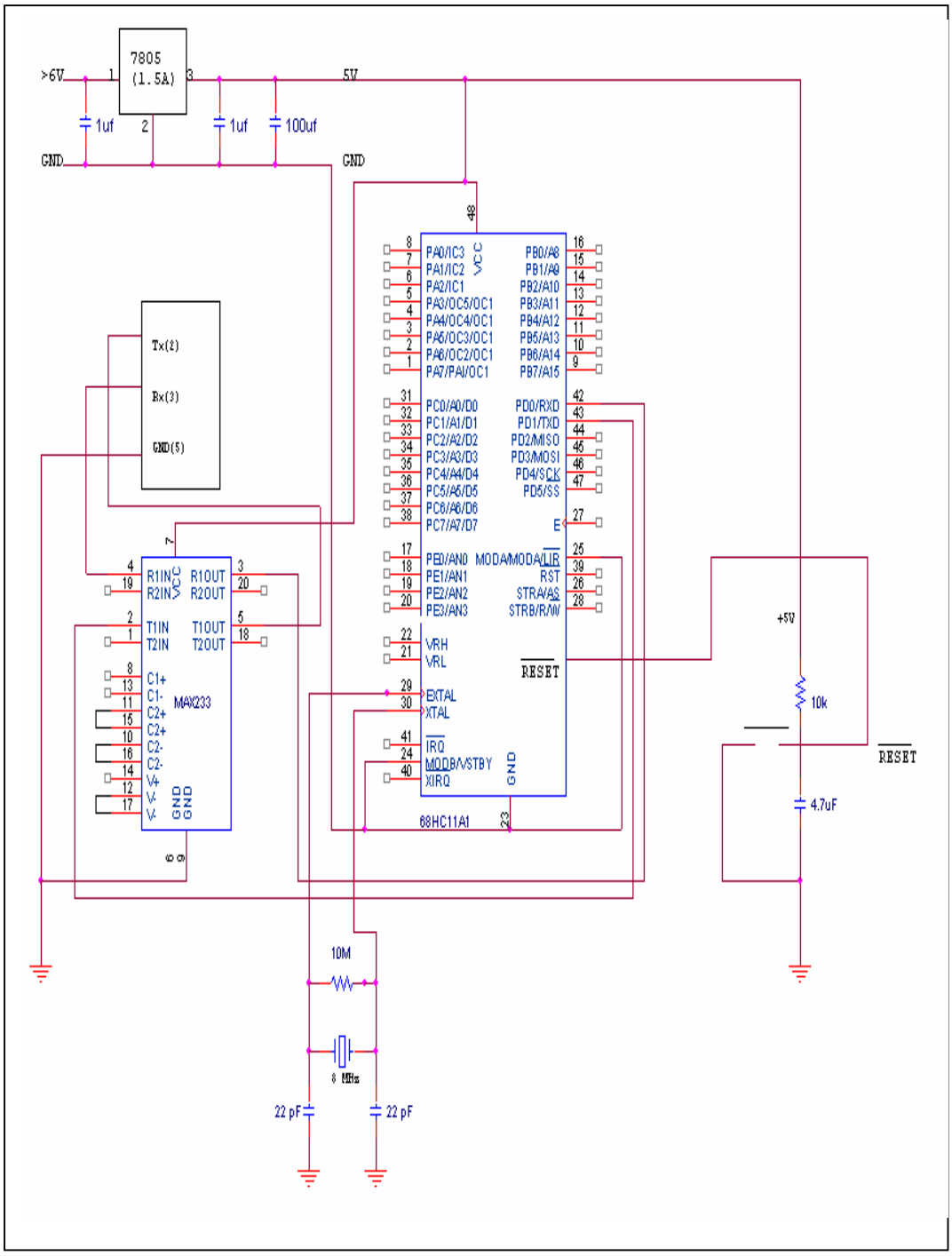


Figure 3.8: Schematic circuit bootstrap mode connection

3.3.2 Developing the Programming of MICROCONTROLLER 6811

The coding of microcontroller 6811 developed in assembly language. Then the coding will simulate in THRsim. The format of coding is in asm. Below is the complete coding to generate the encoder sensor signal in pulse:

```
REGS      EQU    $1000
OPTION    EQU    $39
ADCTL     EQU    $30
ADR1      EQU    $31
SCCR1     EQU    $2C
SCCR2     EQU    $2D
BAUD      EQU    $2B
SCSR      EQU    $2E
SCDR      EQU    $2F

          ORG    $B600
          LDS    #$FF
          LDX    #REGS

***************************INITIALIZE ADC***************************
INI_ADC   LDAA   #$80
STAA      OPTION,X
JSR       DELAY
LDAA      #$20
STAA      ADCTL,X

***************************INITIALIZE SCI***************************
INI_SCI   LDAA   #30
STAA      BAUD,X     ;SETTING BAUD RATE TO 9600 KB/S
CLRA      ;WAKE UP METHOD=IDDLE LINE,CHARACTER LENGHT=(1-ATART,8-DATA,1-STOP)
STAA      SCCR1,X
LDAA      #$08       ;ENABLE TRNSMIT
STAA      SCCR2,X

*******************************************************************
LOOP      CLRA
SCN_CCF   BRCLR   ADCTL,X $80  SCN_CCF
LDAA      ADR1,X
JSR       DELAY
STAA      SCDR,X
TRNSMIT   BRCLR   SCSR,X $80  TRNSMIT   ;BRANCH TO TRANSMIT IF TDRE=1
BRA       LOOP

DELAY     PSHA
          PSHX
          LDAA   #$5
REPEAT    LDX    #$FF
AGAIN     DEX
          BNE    AGAIN
          DECA
          BNE    REPEAT
          PULX
          PULA
          RTS

          ORG    $FFFE
          FDB    $B600
          END
```

Figure 3.9: Coding to generate encoder sensor signal

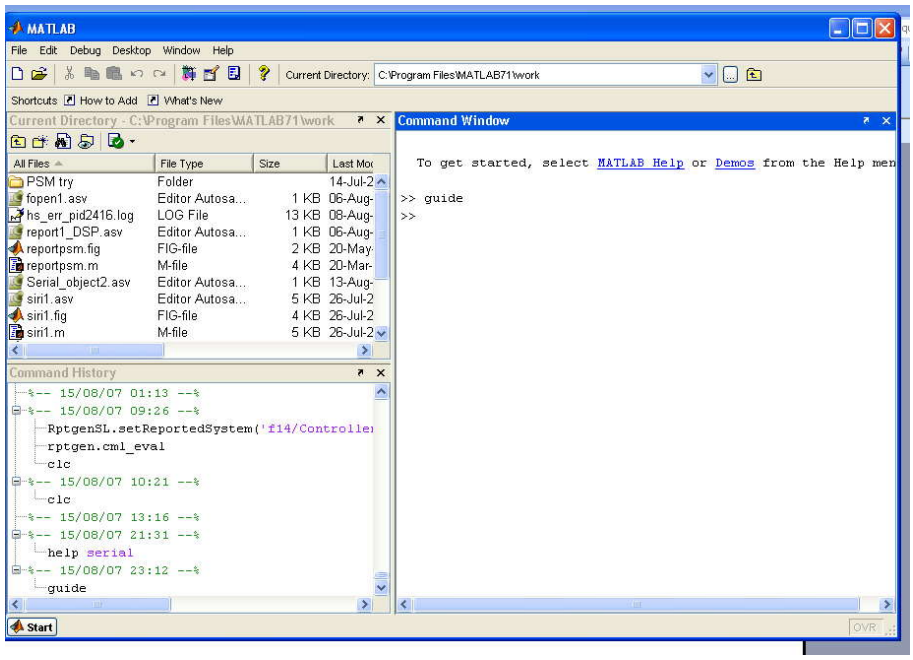
3.4 Developing software

3.4.0 Introduction

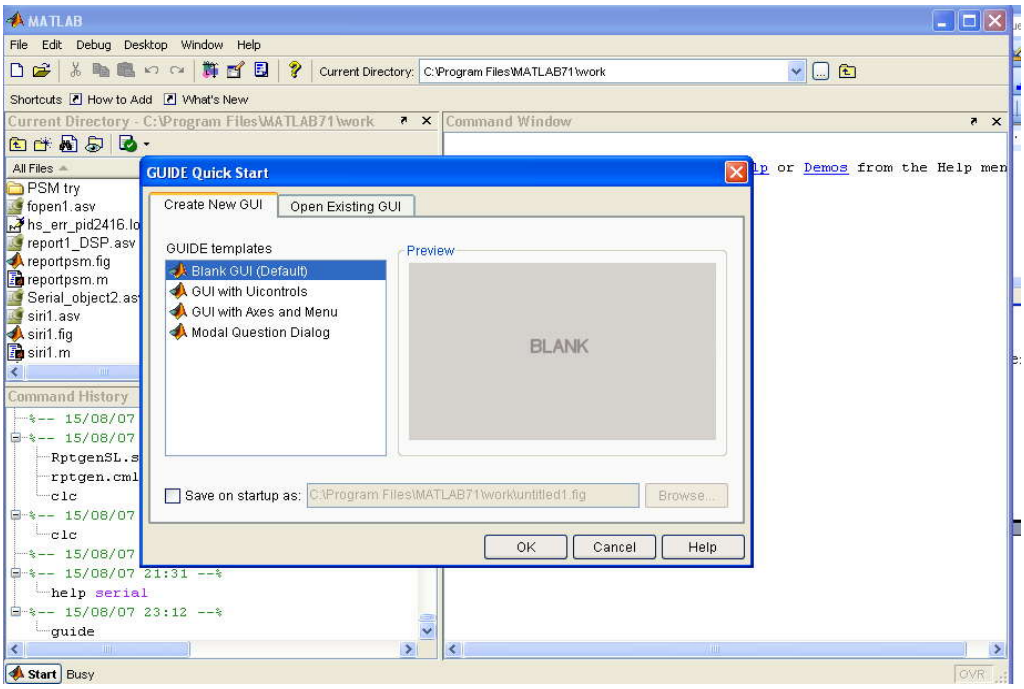
The software was used in to create GUI is MATLAB. It is because this project uses more calculation programming, it is not suitable to use visual basic because it is suitable for simple mathematical calculation. The signal from encoder sensor will display on GUI using PIC as the interface medium. The type of MATLAB that used to create the GUI is MATLAB 7.1 version. MATLAB GUI is Script files that have interaction with user by using Windows.

3.4.1 Step to create MATLAB GUI Development Environment

1. Type GUIDE at the command window.

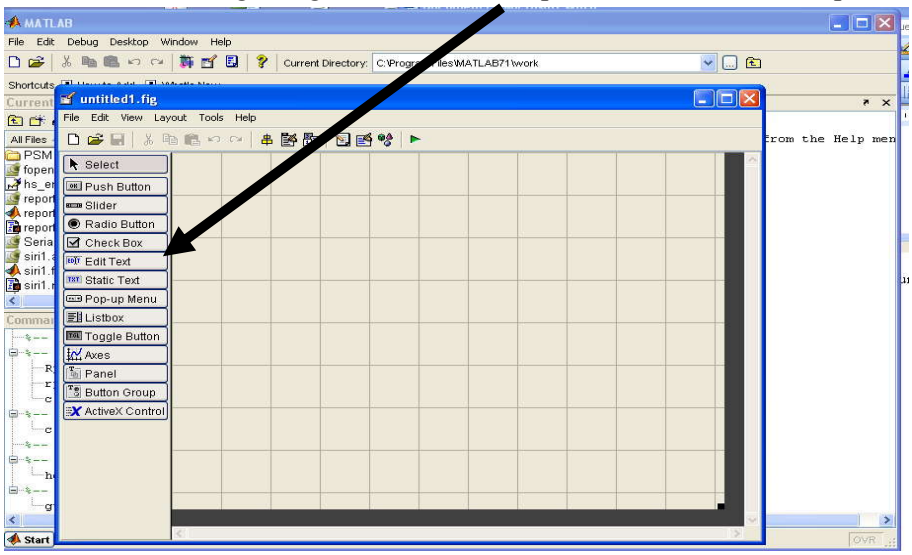


2. The GUIDE start dialog will pop up and select the pre-built templates. Use the default blank GUI template.

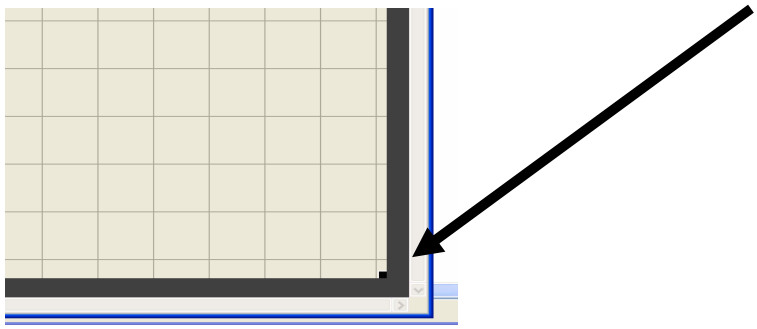


3. The GUIDE layout editor that appears for design the layout of GUI.

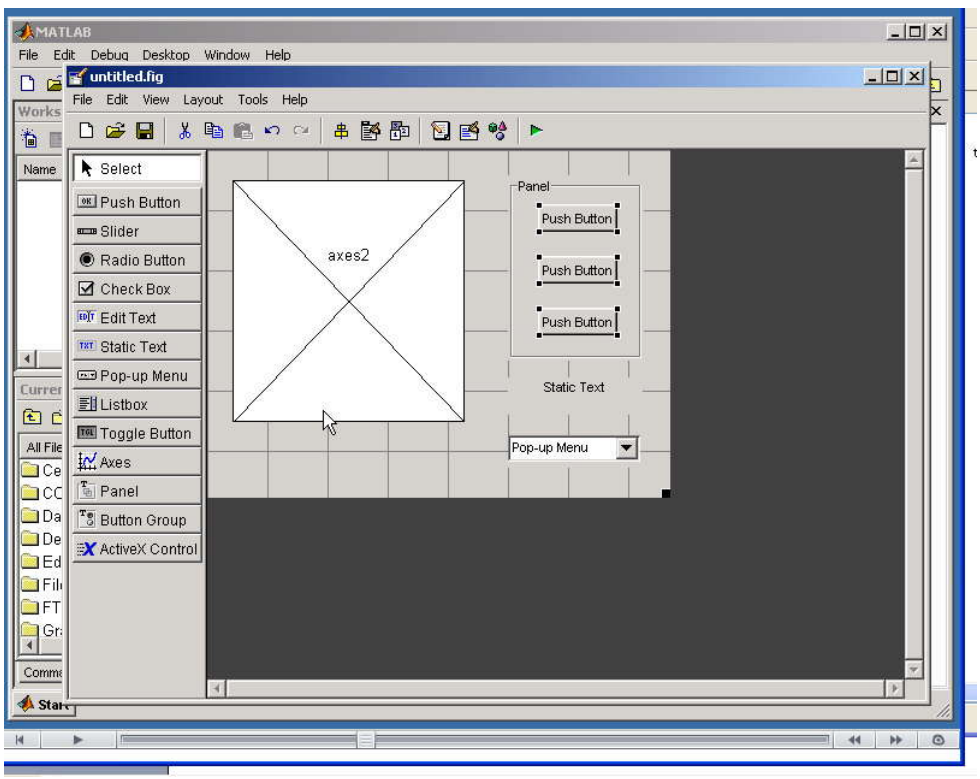
GUI can be creating using collection of components available in the pallet.



4. Resize the GUI

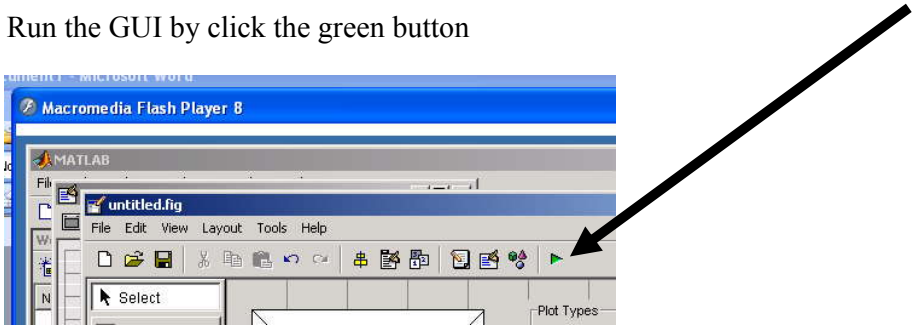


5. Add push button, pop up menu and axis from the pallet to the layout by left click and drag on the layout.

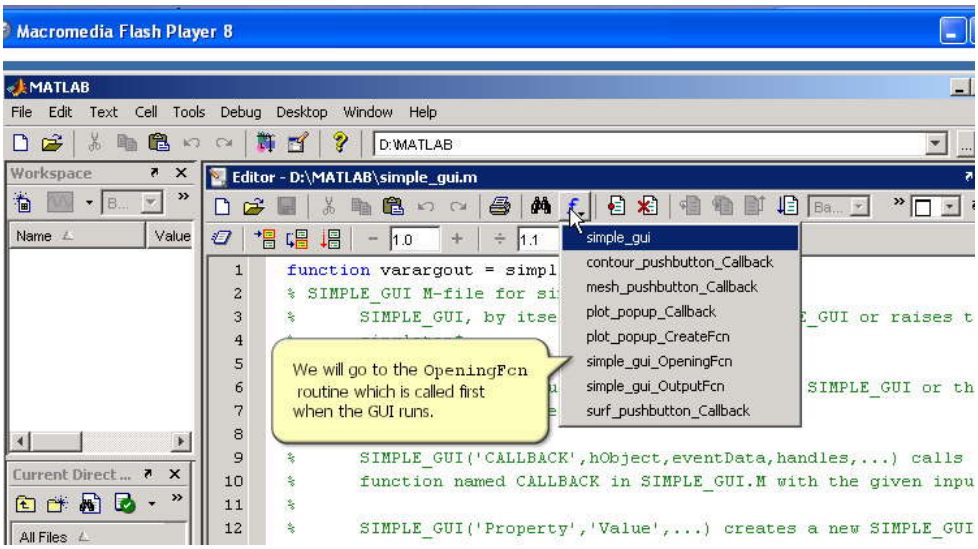


6. Open the Property Inspector at View > Property Inspector to change the name of pushbutton, Panel and other function name.

7. Run the GUI by click the green button



8. After click the green button, save the GUI M-file layout, the automatic generate code will display at M-file. Click the F symbol too



9. In the function, data can be load to do the function. Then syntax for MATLAB GUI Development Environment can be developed.

3.4.2 MATLAB Coding to create GUI

3.4.2.0 Initialize Coding for Graphical User Interface (GUI)

In MATLAB, GUI can be develop by create the coding in M-File. The initialize coding must be creating first, before create the main coding. The purpose to create the initialize coding is to assign port which is available and enable the port in computer. The function of the port is to transfer and receive data. It will be linker between hardware and software in computer. Below is the initialize coding in M-File:

```
s=serial('COM');
fopen(s)
handles.op=s  % store data
guidata(hObject, handles); %save data
```

Description:

1. s=serial('COM') or obj = serial('port') creates a serial port object associated with the serial port specified by port. If port does not exist, or if it is in use, the serial port object will not be able to connect to the device. User can justify which port by looking at start >control panel>system at system roperties select Hardware>device manager>Port COM&LPT)> there is he port number eg: COM7.
2. fopen(obj) connects obj to the device.
   'obj' must be connected to the device with the fopen function before execute read and write operation. When obj is connected to the device:

   - Data remaining in the input buffer or the output buffer is flushed.

   - The Status property is set to open.

   - The BytesAvailable, ValuesReceived, ValuesSent, and BytesToOutput properties are set to 0.

An error will occur when the obj is not connected to the device which is the cannot be read or write. Some properties are read-only while the serial port object is open (connected), and must be configured before using fopen. Examples include InputBufferSize and OutputBufferSize. Refer to the property reference pages to determine which properties have this constraint. The values for some properties are verified only after obj is connected to the device. If any of these properties are incorrectly configured, then an error is returned when fopen is issued and obj is not connected to the device. Properties of this type include BaudRate, and are associated with device settings.

3. handles.op=s: When GUI is running, the M-file creates a handles structure That contains all the data for GUI objects, such as controls, menus, and axes. The handles structure is passed as an input to each callback. The handles structure can be use to share data between callbacks and access GUI data.
4. guidata(hObject, handles)or guidata(object_handle,data) will stores the variable data as GUI data. If object_handle is not a figure handle, then the object's parent figure is used. data can be any MATLAB variable, but is typically a structure, which is enables to add new fields as required.

3.4.2.1 Content Coding for Graphical User Interface (GUI)

In this project, there are several important coding that must be understood. Below is the coding in M-File:

i. Coding to plot the graph.

```
s=handles.op %retrieve data
out=fread(s)
plot(out)
```

The function of s=handles.op coding is to retrieve data. Data will callback by using this coding, it will define again the data. While, the function of out=fread(s)or A=fread(obj)is reads binary data from the device connected to obj, and returns the

data to A. Data will be read in 10 second. The GUI Plot button callback creates a plot of the run data and adds a legend. The data to plot is passed to the callback in the handles structure, which also contains the gain settings used when the simulation ran. When a user clicks on the Plot button, the callback executes the following steps:

- Collects the data for each run selected in the Results list, including two variables (time vector and output vector) and a color for each result run to plot.
- Generates a string for the legend from the stored data.
- Creates the figure and axes for plotting and saves the handles for use by the Close button callback.
- Plots the data, adds a legend, and makes the figure visible.

ii. Coding to insert the picture:

```
guidata(hObject, handles);
[x,map]=imread('project','jpg');
image(x)
set(gca,'visible','off')
```

The function of coding above will put the picture that we select in GUIThe picture must be in jpeg format.

iii. Coding to connect between the windows in GUI

```
function varargout=xxx_Callback(h,eventdata,handles,varagin)
figure(yyy)
```

3.4.2.2 Close Coding for Graphical User Interface (GUI)

Below is the coding to close the port of the GUI:

```
s=handles.op %retrieve data
fclose(s)
```

CHAPTER 4

RESULT AND ANALYSIS

4.0 Result

The figures below show the environment of GUI (Graphical User Interface) created using MATLAB.

Figure 4.1: Main window of GUI

The figure 4.1 shows the main window of GUI. At the front of the GUI there are three main button. There are introduction button,analysis button and close button.the introduction button will display the introduction window when it is clicking. The content of the introduction window is the general view of the overall project. For the analysis scope button it will display the analysis window. The graph will displayed the signal that generate by encoder sensor. The function of close button is it will close the analysis scope window.

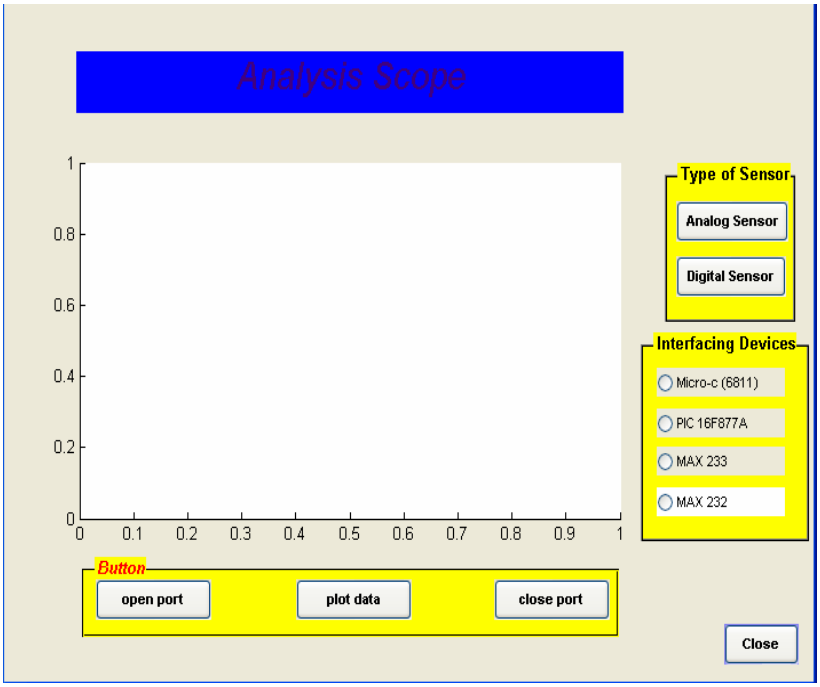


Figure 4.2: Analysis Scope window of GUI before plot the data

The figure 4.2 shows analysis scope window. Now, the window is in initial condition before plot the graph. The signal will disply in the window. There are three main button in this window. They are open port window, plot data button and close port button. The function of the open port button is to open the serial port to allow the data to transmit or receive between hardware and software in computer. The function of the plot data button is it will read the data from external device and plot the data in GUI. For the close port button it will close the serial port, in other words it will disconnect the communication between hardware and software in computer.

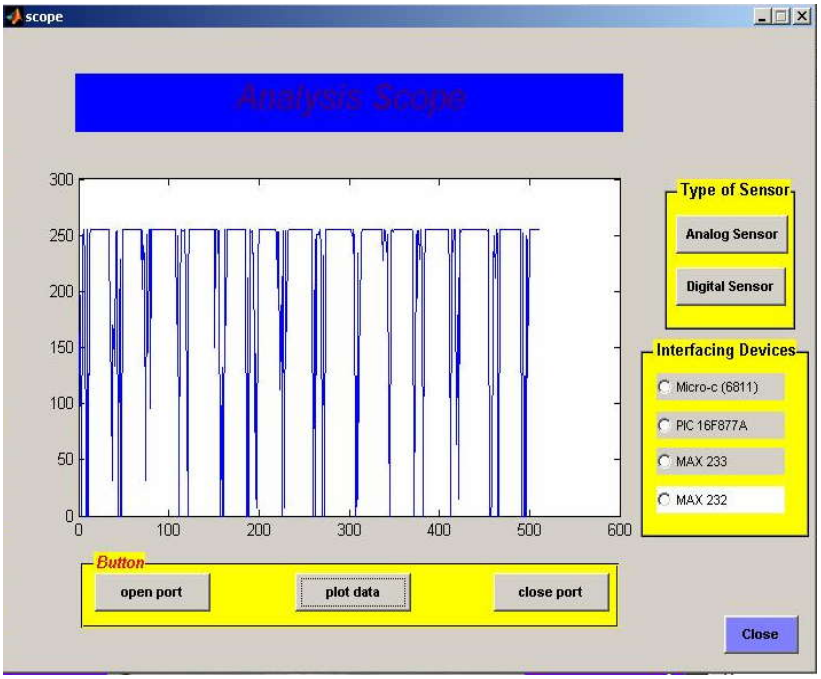


Figure 4.3: Analysis scope window of GUI after plot the data.

The figure 4.3 shows the analysis scope window after plot the data. The data displayed in digital form. The form of signal that displayed in GUI is not accurate because the noise occurred. Beside that, this window can measured various type of signal from various sensor at the same time.

4.1 Analysis

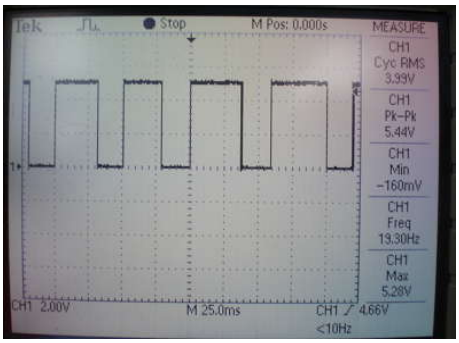


Figure 4.4 The signal of encoder sensor in oscilloscope

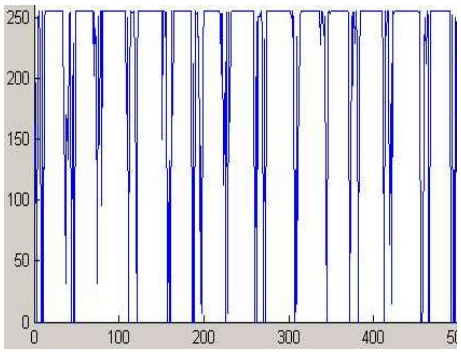


Figure 4.5 The signal of encoder sensor in GUI

The figure above showed the comparison between the signal display in GUI the signal display in oscilloscope. The signal display in GUI is not accurate because it has noise and distortion occurred that give effect to the original signal.

This project used microcontroller as a connecter to link between hardware and software (MATLAB).The microcontroller will convert the signal and transfer the signal through MAX233. The function of MAX233 in this project is to communicate to the computer and at the same time it remains the signal in stable condition by amplified the signal in order to reduce the losses and to avoid noise.

In this project, to link between hardware and software or to transfer the data it used DB9.DB9 also known as the D-subminiature or D-sub is a common type of electrical connector used particularly in computers. It is one of the largest common connectors used in computer after USB. DB9 usually used for RS-232 serial communication interface (SCI) to transmit data from microcontroller.

At the end of this project, the voltage signal can be displayed and measured through GUI The signal is in digital form, display in GUI , MATLAB.

CHAPTER 5

CONCLUSION AND RECOMMANDATION

5.0 Conclusion

One of the primary goals of this project is to display signal on the visual form which is using GUI. (Besides that, the more important here is the project is user friendly. Thus, the project is design to give as much facilities to the people especially to student for the educational purpose. In the end of this project, the signal that generate by encoder sensor can displayed in GUI. Microcontroller 6811 has been used in this project to interface between hardware and software. Beside that, GUI also must be developed as a tool to display the signal that generated by encoder sensor. As a conclusion, this project achieved the objective and scopes. This project also is useful because it offers the user can analyze the signal quite accurate.

5.1 Recommendation

For the future plan of this project, it is recommended to other candidate to do more studies on the related information with this topic in order to continue with the real method that has been discuss in methodology chapter. Besides that, GUI can be created to make analysis from several of sensor.

5.1.1 Costing and commercialization

There are some components are needed to build and develop of encoder sensor with GUI. Figure 5.1 shows the entire component related to build the system, all the components are easy to get accept microcontroller. This system using Motorola MC68HC11A1 microcontroller, A1 is an early version of MC68HC family so in the future A1 version of microcontroller maybe not produces anymore. But as in chapter 2 already explain, MC68HC family is an upward compatible so program from early version can run at the latest version of MC68HC family but not in vice versa.

I believe this system can be commercialized because it's a user friendly system that can be used in any personal computer (PC) or laptop to analyze signal.

Table 5.0: List of component and it's cost

No.	Components	Specification	Estimation price/unit	Quantity	Estimation price
1	Micro.C	MC68HC11	RM40.00	1	RM 40.00
2	Regulator	7805	RM1.00	1	RM 1.00
3	PCB Header		RM0.80	5	RM 4.00
4	I.C Base	48 pin	RM0.50	1	RM 0.50
		10 pin	RM0.10	1	RM 0.17
5	Crystal	8MHz	RM1.20	1	RM 1.20
6	Strip board	10'x 4'	RM4.00	2	RM 8.00
7	Reset Switch		RM0.30	2	RM 0.60
8	MAX233		RM9.50	1	RM 9.50
9	Capacitor	4.7 uF (25V)	RM0.12	3	RM 0.36
		1 Uf (25V)	RM0.12	6	RM 0.72
		0.1 uF (25V)	RM0.15	3	RM 0.45
		22 pF(25V)	RM0.08	3	RM 0.24
		10uF(25V)	RM0.15	3	RM 0.45
10	Resistor	220R	RM0.02	10	RM 0.20
		1M	RM0.02	3	RM 0.06
		15k	RM0.02	2	RM 0.04
		22k	RM0.04	2	RM 0.08
11	DB9	Female	RM0.60	1	RM 0.60
12	DB9 Cover		RM0.60	1	RM 0.60
13	Heat Sink		RM0.70	1	RM 0.70
14	Wrapping Wire		RM15.00	1	RM 15.00
TOTAL ESTIMATION PRICE					**RM 84.47**

REFERENCES

[1] GUI Definition, (2004)
http://www.bellevuelinux.org/gui.html

[2] Chapman, Stephen J., (2001),MATLAB Programming for Engineers, Brooks Cole, 2001.

[3] Esazonov,(2003),"Building GUI interfaces in Matlab"
http://www.intelligent-systems.info/classes/ee509/gui.htm

[4] R. S. Schestowitz, (2004),"Collated advice on construction of user Interfaces"
http://www.mathworks.com/matlabcentral/files/5439/gui_tips.pdf

[5] Thomas, (2007),"Digital Encoder is used for motion control applications."
http://news.thomasnet.com/fullstory/801734

[6] Wikipedia, (2007), Encoder
http://en.wikipedia.org/wiki/Encoder

[7] Wikipedia, (2007), Sensor
http://en.wikipedia.org/wiki/Sensor

[8] Tom Huber, (1997),"Creating a GUI in Matlab"
http://physics.gac.edu/~huber/envision/matgui/matgui.htm

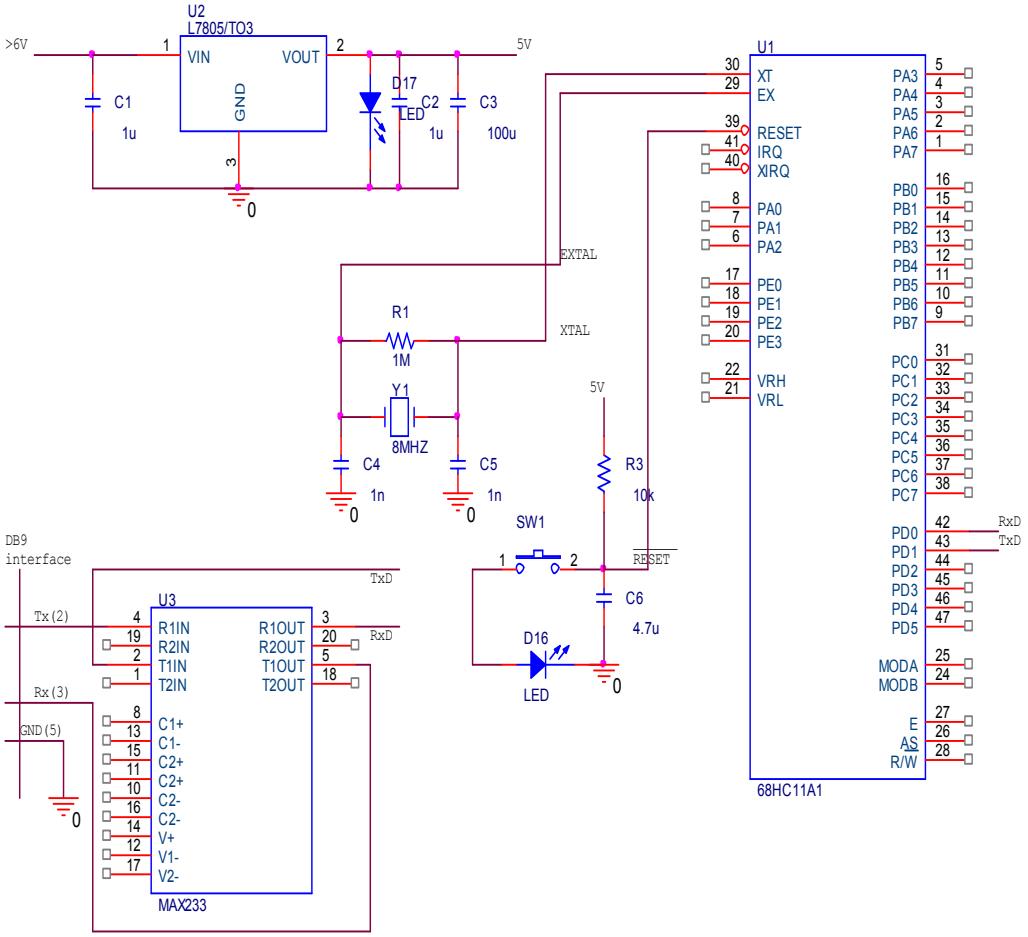
[9] Peter Spasov (1996)."Microcontroller Technology The 68HC11".
Englewood Cliffs, N.J.: Prentice Hall

[10]"68HC11 Reference Manual": Motorola

[11] Michael smith, (2003),"The oscilloscope"
http://www.doctronics.co.uk/scope.htm

APPENDICES

APPENDIX A



The complete circuit for Bootstrap mode MC68HC11A1

APPENDIX B

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ....-0.3V to +6V
Input Voltages
T_{IN}....-0.3V to (V_{CC} - 0.3V)
R_{IN} (Except MAX220) ....±30V
R_{IN} (MAX220)....±25V
T_{OUT} (Except MAX220) (Note 1) ....±15V
T_{OUT} (MAX220)....±13.2V
Output Voltages
T_{OUT}....±15V
R_{OUT}....-0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND.....Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C)....842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C)....889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) ..440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ...696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C).....762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C).....762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C)....800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C).....800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C).....842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ .....0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ .....-40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _.....-55°C to +125°C
Storage Temperature Range .....-65°C to +160°C
Lead Temperature (soldering, 10sec) .....+300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.

Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kb/s
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ....-0.3V to +6V
Input Voltages
T_{IN}....-0.3V to (V_{CC} - 0.3V)
R_{IN} (Except MAX220) ....±30V
R_{IN} (MAX220)....±25V
T_{OUT} (Except MAX220) (Note 1) ....±15V
T_{OUT} (MAX220)....±13.2V
Output Voltages
T_{OUT}....±15V
R_{OUT}....-0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND.....Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C)....842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C)....889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) ..440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ...696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C).....762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C).....762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C)....800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C).....800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C).....842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ....0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ....-40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _....-55°C to +125°C
Storage Temperature Range ....-65°C to +160°C
Lead Temperature (soldering, 10sec) ....+300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kb/s
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

+5V-Powered, Multichannel RS-232 Drivers/Receivers

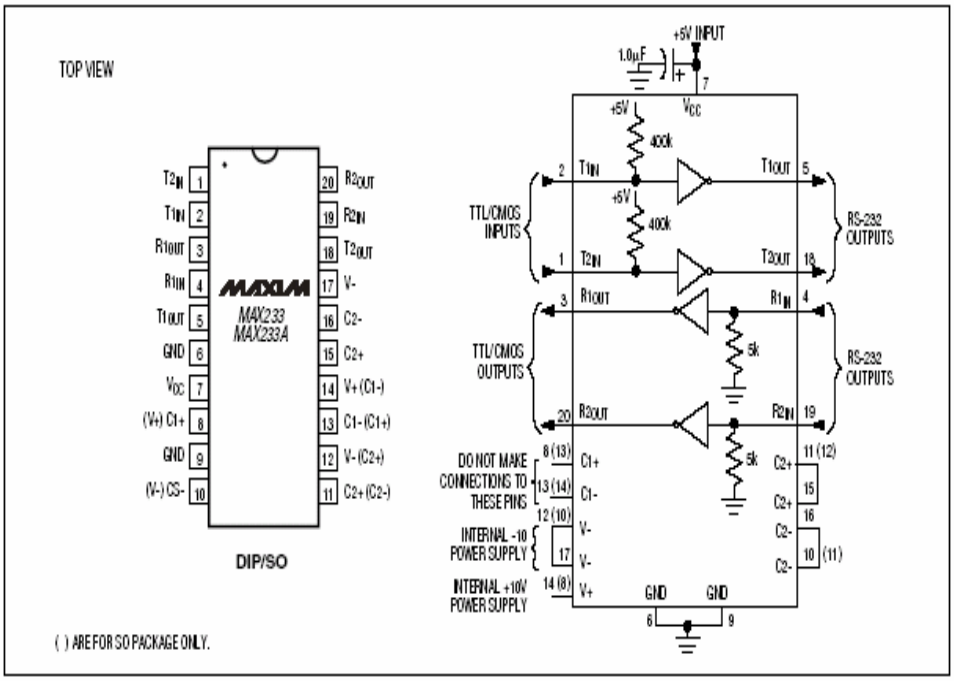


Figure 11. MAX233/MAX233A Pin Configuration and Typical Operating Circuit

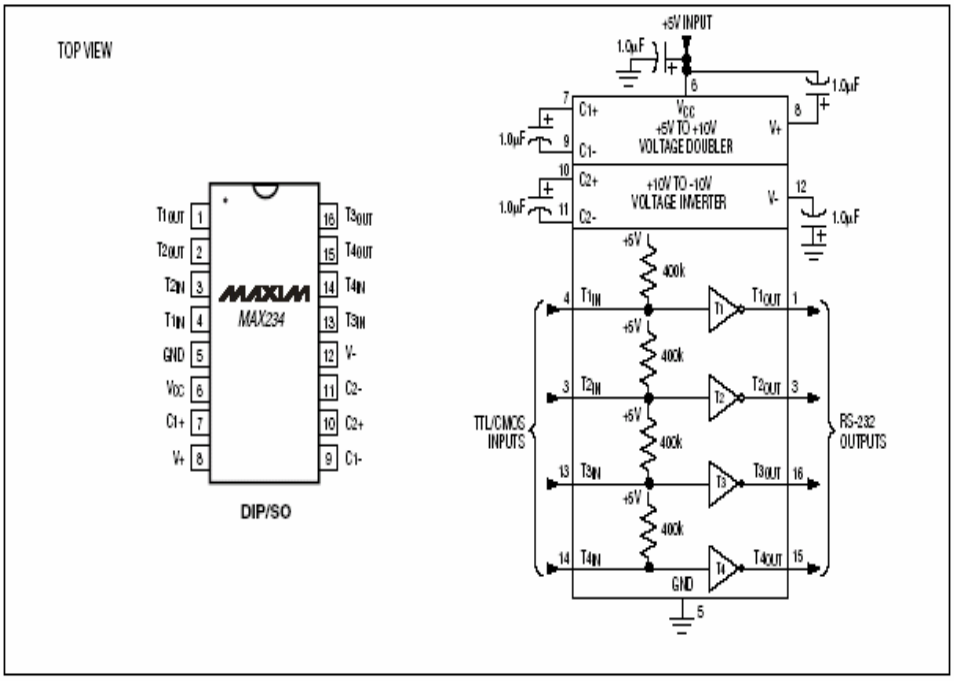


Figure 12. MAX234 Pin Configuration and Typical Operating Circuit

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX225/MAX244–MAX249

Supply Voltage (V_{CC}) .....-0.3V to +6V
Input Voltages
T_{IN}, $\overline{ENA}$, $\overline{ENB}$, $\overline{ENR}$, $\overline{ENT}$, $\overline{ENRA}$, $\overline{ENRB}$, $\overline{ENTA}$, $\overline{ENTB}$ .....-0.3V to (V_{CC} + 0.3V)
R_{IN} .....±25V
T_{OUT} (Note 3) .....±15V
R_{OUT} .....-0.3V to (V_{CC} + 0.3V)
Short Circuit (one output at a time)
T_{OUT} to GND .....Continuous
R_{OUT} to GND .....Continuous

Continuous Power Dissipation (T_A = +70°C)
28-Pin Wide SO (derate 12.50mW/°C above +70°C) 1W
40-Pin Plastic DIP (derate 11.11mW/°C above +70°C) ...611mW
44-Pin PLCC (derate 13.33mW/°C above +70°C) 1.07W
Operating Temperature Ranges
MAX225C_ _, MAX24_C_ _ .....0°C to +70°C
MAX225E_ _, MAX24_E_ _ .....-40°C to +85°C
Storage Temperature Range .....-65°C to +160°C
Lead Temperature (soldering,10sec) .....+300°C

Note 4: Input voltage measured with transmitter output in a high-impedance state, shutdown, or V_{CC} = 0V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX225/MAX244–MAX249

(MAX225, V_{CC} = 5.0V ±5%; MAX244–MAX249, V_{CC} = +5.0V ±10%, external capacitors C1–C4 = 1µF; T_A = T_{MIN} to T_{MAX}; unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	Tables 1a–1d	Normal operation		10	50	µA
		Shutdown		±0.01	±1	
Data Rate	Tables 1a–1d, normal operation			120	64	kbits/sec
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±7.5		V
Output Leakage Current (shutdown)	Tables 1a–1d	$\overline{ENA}$, $\overline{ENB}$, $\overline{ENT}$, $\overline{ENTA}$, $\overline{ENTB}$ = V_{CC}, V_{OUT} = ±15V		±0.01	±25	µA
		V_{CC} = 0V, V_{OUT} = ±15V		±0.01	±25	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V (Note 4)		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±30		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±25	V
RS-232 Input Threshold Low	V_{CC} = 5V		0.8	1.3		V
RS-232 Input Threshold High	V_{CC} = 5V			1.8	2.4	V
RS-232 Input Hysteresis	V_{CC} = 5V		0.2	0.5	1.0	V
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		
TTL/CMOS Output Leakage Current	Normal operation, outputs disabled, Tables 1a–1d, 0V ≤ V_{OUT} ≤ V_{CC}, $\overline{ENR\_}$ = V_{CC}			±0.05	±0.10	µA

APPENDIX C

Main window coding

```
function varargout = main(varargin)
% MAIN M-file for main.fig
%      MAIN, by itself, creates a new MAIN or raises the existing
%      singleton*.
%
%      H = MAIN returns the handle to a new MAIN or the handle to
%      the existing singleton*.
%
%      MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAIN.M with the given input arguments.
%
%      MAIN('Property','Value',...) creates a new MAIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before main_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to main_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 main

% Last Modified by GUIDE v2.5 27-Oct-2007 10:42:55

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @main_OpeningFcn, ...
                   'gui_OutputFcn',  @main_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 main is made visible.
function main_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 main (see VARARGIN)

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

% Update handles structure

guidata(hObject, handles);
[x,map]=imread('project','jpg');
image(x)
set(gca,'visible','off')


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


% --- Outputs from this function are returned to the command line.
function varargout = main_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 introduction.
function varargout=introduction_Callback(h,eventdata,handles,varagin)
figure(information)
% hObject    handle to introduction (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 pushbutton2.
function varargout=pushbutton2_Callback(h,eventdata,handles,varagin)
figure(scope)

% hObject    handle to pushbutton2 (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 History.
function History_Callback(hObject, eventdata, handles)
% hObject    handle to History (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 Close.
function Close_Callback(hObject, eventdata, handles)
% hObject    handle to Close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1
```

Information window coding

```
function varargout = Information(varargin)

% Last Modified by GUIDE v2.5 26-Oct-2007 23:24:25

if nargin == 0  % LAUNCH GUI

fig = openfig(mfilename,'reuse');

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

axes(handles.axesInfo)    % Select the proper axes
axis off
str1(1) = {['\bf Design And Developement Of '
            '  Encoder Sensor with GUI      ']};

str2(1)= {['MATLAB is one of the software that can create GUI (Graphical User
Interface).          '
           'The GUI will create using guide to create the GUI layout editor. The
objective          '
           'of this project is to display the signal that generated by encoder sensor
in          '
           'GUI (Graphical User Interface). There are three phase to develop this
project.          '
           'Phase one is developing the sensor circuit to produce signal which is the
signal        '
           'is in analogue form. The second phase is developing the controller
circuit in          '
           'order to convert the analogue signal to digital signal. The changes of
the signal       '
           'form from analogue signal to digital signal are necessary because the
computer only    '
           'can accept the data in digital form.The third phase is developing the GUI
(Graphical   '
           'User Interface)in MATLAB. In the end of this project, the signal
generated by encoder  '
           'sensor will display in GUI (Graphical User Interface).
']};

text(0.2,  1,  str1, 'FontSize', 14, 'color', [0 0 0.502])
text(0,  .5, str2, 'FontSize', 10, 'color', [0 0 0.502])

 %text(0.36, -3.55,'\bf\copyright', 'FontSize', 10, 'color',[1 0 0])

 if nargout > 0
        varargout{1} = fig;
    end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
```

```
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.



% ------------------------------------------------------------------


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

Analysis Scope window coding

```
function varargout = scope(varargin)
% SCOPE M-file for scope.fig
%      SCOPE, by itself, creates a new SCOPE or raises the existing
%      singleton*.
%
%      H = SCOPE returns the handle to a new SCOPE or the handle to
%      the existing singleton*.
%
%      SCOPE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SCOPE.M with the given input arguments.
%
%      SCOPE('Property','Value',...) creates a new SCOPE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before scope_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to scope_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 scope

% Last Modified by GUIDE v2.5 28-Nov-2007 07:18:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @scope_OpeningFcn, ...
                   'gui_OutputFcn',  @scope_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 scope is made visible.
function scope_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 scope (see VARARGIN)
s=serial('COM1')
handles.op=s  % store data
guidata(hObject, handles); %save data

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = scope_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 open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fopen(s)


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

s=handles.op %retrieve data
out=fread(s)
plot(out)

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fclose(s)



% --- Executes on button press in analog.
function varargout=analog_Callback(h,eventdata,handles,varagin)
figure(analog)
% hObject    handle to analog (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 digital.
function varargout=digital_Callback(h,eventdata,handles,varagin)
figure(digit)
% hObject    handle to digital (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 radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (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 radiobutton1


% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (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 radiobutton4


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (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 radiobutton3


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to radiobutton2 (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 radiobutton2


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

close


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (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 pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Analog window coding

```
function varargout = analog(varargin)
% ANALOG M-file for analog.fig
%      ANALOG, by itself, creates a new ANALOG or raises the existing
%      singleton*.
%
%      H = ANALOG returns the handle to a new ANALOG or the handle to
%      the existing singleton*.
%
%      ANALOG('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ANALOG.M with the given input arguments.
%
%      ANALOG('Property','Value',...) creates a new ANALOG or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before analog_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to analog_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 analog

% Last Modified by GUIDE v2.5 27-Oct-2007 11:54:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @analog_OpeningFcn, ...
                   'gui_OutputFcn',  @analog_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 analog is made visible.
function analog_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 analog (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = analog_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 speed.
function varargout=speed_Callback(h,eventdata,handles,varagin)
figure(speed)
% hObject    handle to encoder (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 Temperature.
function varargout=Temperature_Callback(h,eventdata,handles,varagin)
figure(temperature)
% hObject    handle to Temperature (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Digital window coding

```
function varargout = digit(varargin)
% DIGIT M-file for digit.fig
%      DIGIT, by itself, creates a new DIGIT or raises the existing
%      singleton*.
%
%      H = DIGIT returns the handle to a new DIGIT or the handle to
%      the existing singleton*.
%
%      DIGIT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DIGIT.M with the given input arguments.
%
%      DIGIT('Property','Value',...) creates a new DIGIT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before digit_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to digit_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 digit

% Last Modified by GUIDE v2.5 27-Oct-2007 14:04:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @digit_OpeningFcn, ...
                   'gui_OutputFcn',  @digit_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 digit is made visible.
function digit_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 digit (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = digit_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 pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (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 foto.
function varargout=foto_Callback(h,eventdata,handles,varagin)
figure(foto)
% hObject    handle to foto (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 infrared.
function varargout=infrared_Callback(h,eventdata,handles,varagin)
figure(infrared)
% hObject    handle to infrared (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 pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close

% --- Executes on button press in encoder.
function varargout=encoder_Callback(h,eventdata,handles,varagin)
figure(encoder)
% hObject    handle to encoder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Encoder sensor window coding

```
function varargout = encoder(varargin)
% ENCODER M-file for encoder.fig
%      ENCODER, by itself, creates a new ENCODER or raises the existing
%      singleton*.
%
%      H = ENCODER returns the handle to a new ENCODER or the handle to
%      the existing singleton*.
%
%      ENCODER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ENCODER.M with the given input arguments.
%
%      ENCODER('Property','Value',...) creates a new ENCODER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before encoder_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to encoder_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 encoder

% Last Modified by GUIDE v2.5 27-Oct-2007 13:55:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @encoder_OpeningFcn, ...
                   'gui_OutputFcn',  @encoder_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 encoder is made visible.
function encoder_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 encoder (see VARARGIN)
s=serial('COM4')
handles.op=s  % store data
guidata(hObject, handles); %save data

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = encoder_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 open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fopen(s)


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

s=handles.op %retrieve data
out=fread(s)
plot(out)

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fclose(s)



% --- Executes on button press in analog.
function analog_Callback(hObject, eventdata, handles)
% hObject    handle to analog (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 digital.
function digital_Callback(hObject, eventdata, handles)
% hObject    handle to digital (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 radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (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 radiobutton1


% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (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 radiobutton4


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (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 radiobutton3


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (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 radiobutton2


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

Close
```

Foto sensor window coding

```
function varargout = foto(varargin)
% FOTO M-file for foto.fig
%      FOTO, by itself, creates a new FOTO or raises the existing
%      singleton*.
%
%      H = FOTO returns the handle to a new FOTO or the handle to
%      the existing singleton*.
%
%      FOTO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FOTO.M with the given input arguments.
%
%      FOTO('Property','Value',...) creates a new FOTO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before foto_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to foto_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 foto

% Last Modified by GUIDE v2.5 25-Oct-2007 14:35:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @foto_OpeningFcn, ...
                   'gui_OutputFcn',  @foto_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 foto is made visible.
function foto_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 foto (see VARARGIN)
s=serial('COM4')
handles.op=s  % store data
guidata(hObject, handles); %save data

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = foto_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 open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fopen(s)


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

s=handles.op %retrieve data
out=fread(s)
plot(out)
```

Infrared sensor coding

```
function varargout = infrared(varargin)
% INFRARED M-file for infrared.fig
%      INFRARED, by itself, creates a new INFRARED or raises the existing
%      singleton*.
%
%      H = INFRARED returns the handle to a new INFRARED or the handle to
%      the existing singleton*.
%
%      INFRARED('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INFRARED.M with the given input arguments.
%
%      INFRARED('Property','Value',...) creates a new INFRARED or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before infrared_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to infrared_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 infrared

% Last Modified by GUIDE v2.5 27-Oct-2007 14:08:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @infrared_OpeningFcn, ...
                   'gui_OutputFcn',  @infrared_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 infrared is made visible.
function infrared_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 infrared (see VARARGIN)
s=serial('COM4')
handles.op=s  % store data
guidata(hObject, handles); %save data

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = infrared_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 open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fopen(s)


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

s=handles.op %retrieve data
out=fread(s)
plot(out)

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fclose(s)
```

Speed Sensor window coding

```
function varargout = speed(varargin)
% SPEED M-file for speed.fig
%      SPEED, by itself, creates a new SPEED or raises the existing
%      singleton*.
%
%      H = SPEED returns the handle to a new SPEED or the handle to
%      the existing singleton*.
%
%      SPEED('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SPEED.M with the given input arguments.
%
%      SPEED('Property','Value',...) creates a new SPEED or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before speed_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to speed_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 speed

% Last Modified by GUIDE v2.5 27-Oct-2007 08:49:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @speed_OpeningFcn, ...
                   'gui_OutputFcn',  @speed_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 speed is made visible.
function speed_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 speed (see VARARGIN)
s=serial('COM4')
handles.op=s  % store data
guidata(hObject, handles); %save data

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = speed_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 open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fopen(s)


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

s=handles.op %retrieve data
out=fread(s)
plot(out)

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fclose(s)
```

Temperature sensor window coding

```
function varargout = temperature(varargin)
% TEMPERATURE M-file for temperature.fig
%      TEMPERATURE, by itself, creates a new TEMPERATURE or raises the existing
%      singleton*.
%
%      H = TEMPERATURE returns the handle to a new TEMPERATURE or the handle to
%      the existing singleton*.
%
%      TEMPERATURE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TEMPERATURE.M with the given input arguments.
%
%      TEMPERATURE('Property','Value',...) creates a new TEMPERATURE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before temperature_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to temperature_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 temperature

% Last Modified by GUIDE v2.5 27-Oct-2007 08:50:08

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @temperature_OpeningFcn, ...
                   'gui_OutputFcn',  @temperature_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 temperature is made visible.
function temperature_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 temperature (see VARARGIN)
s=serial('COM4')
handles.op=s  % store data
guidata(hObject, handles); %save data

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = temperature_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 open.
function open_Callback(hObject, eventdata, handles)
% hObject    handle to open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fopen(s)


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

s=handles.op %retrieve data
out=fread(s)
plot(out)

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s=handles.op %retrieve data
fclose(s)
```