

2013

DC MOTOR SPEED CONTROL USING MICROCONTROLLER PIC-16F877A

- A Project Report Submitted In Partial Fulfillment Of The Requirements For The Degree Of Bachelor Of Technology In The Electrical Engineering Department.

Submitted By

Krishna Narayan Chowdhury
(Roll No: 09187016032)

Under The Supervision Of

Mrs. Ayindrila Roy



Department of Electrical Engineering
Techno India College Of Technology,
Newtown, Rajarhat, Kolkata-700156.



A PROJECT REPORT
ON

DC MOTOR SPEED CONTROL USING MICROCONTROLLER PIC-16F877A

**A Project Report Submitted
In Partial Fulfillment Of The Requirements
For The Degree Of Bachelor Of Technology
In The Electrical Engineering Department**

Submitted By

**Abhijit Maity
Krishna Narayan Chowdhury
Nilesh Kumar Mal
Pinaki Dutta
Shyamal Kumar Bera
Sohini Chowdhury
Soumik Chakraborty
Subhadip Karmahapatra
Suhel Mohammed**

Under The Supervision Of

Mrs. Ayindrila Roy



**Department of Electrical Engineering
Techno India College Of Technology
Newtown, Rajarhat, Kolkata-700156.**

ACKNOWLEDGEMENT

We feel proud to express our profound and deep gratitude to **Mrs. Ayindrila Roy** for suggesting this interesting topic. Her valuable assistance and constant guidance throughout the work was the source of inspiration, without her supervision it would have been difficult for us to complete the project report.

In this connection, we would also like to thank those people, directly or indirectly attached to the department without their assistance we could not have made it.

1. Abhijit Maity
2. Krishna Narayan Chowdhury
3. Nilesh Kumar Mal
4. Pinaki Dutta
5. Shyamal Kumar Bera
6. Sohini Chowdhury
7. Soumik Chakraborty
8. Subhadip Karmahapatra
9. Suhel Mohammed

Students' Name

CERTIFICATE OF APPROVAL

The project report prepared by,
entitled, “**DC MOTOR SPEED CONTROL USING MICROCONTROLLER PIC-16F877A**”, is hereby approved and certified as a creditable study of technological subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite to the Bachelor degree for which it has been submitted.

It is understood that by the approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approved the project report only for the purpose for which it has been submitted.

Mrs. Ayindrila Roy
(Lecturer)
(Department of EE)
(Techno India College of Technology)

Prof. (Dr.) Milan Basu
(Head of the Department)
(Department of EE)
(Techno India College of Technology)

Table of Contents

▪ 1. Abstract		6
▪ 2. Theory & Literature Reviews		7
▪ 3. Methodology		18
▪ 4. Result & Discussion		41
▪ 5. Conclusion & Recommendation		44
▪ 6. References		47
▪ 7. Appendix		48

ABSTRACT

Direct current (DC) motor has already become an important drive configuration for many applications across a wide range of powers and speeds. The ease of control and excellent performance of the DC motors will ensure that the number of applications using them will continue grow for the foreseeable future. This project is mainly concerned on DC motor speed control system by using microcontroller PIC 16F877A. It is a closed-loop real time control system, where optical encoder (built in this project) is coupled to the motor shaft to provide the feedback speed signal to controller. Pulse Width Modulation (PWM) technique is used where its signal is generated in microcontroller. Microcontroller acts as proportional (P) controller with $K_p = 1$ in this study.

The PWM signal will be send to motor driver to vary the voltage supply to motor to maintain at constant speed. A program in Visual Basic 6.0 is developed to provide a graphic user interface (GUI) for the user to enter desired speed at computer. Besides, it also shows a graph of motor speed versus time to let the user monitor the performance of the system easily. Based on the result, the reading of optical encoder built is quite reliable. Through the project, it can be concluded that microcontroller PIC 16F877A can control motor speed at desired speed although there is a variation of load.

THEORY AND LITERATURE REVIEW

Introduction :

This chapter includes the study of different types of DC motors, speed measurement of DC motor, model of separately excited DC motor, several types of DC motor speed controller. It also discusses briefly about microcontroller, RS232 serial port, MPLAB IDE and Visual Basic 6.0.

DC Motor :

There are several types of DC motors that are available. Their advantages, disadvantages, and other basic information are listed below in the table.

Advantages and disadvantages of various types of DC motor.

Type	Advantages	Disadvantages
<i>Stepper Motor</i>	Very precise speed and position control. High Torque at low speed.	Expensive and hard to find. Require a switching control circuit.
<i>DC Motor w/field coil</i>	Wide range of speeds and torques. More powerful than permanent magnet motors	Require more current than permanent magnet motors, since field coil must be

		energized. Generally heavier than permanent magnet motors. More difficult to obtain.
<i>DC permanent magnet motor</i>	Small, compact, and easy to find. Very inexpensive	Generally small. Cannot vary magnetic field strength.
<i>Gasoline (small two stroke)</i>	Very high power/weight ratio. Provide Extremely high torque. No batteries required.	Expensive, loud, difficult to mount, very high vibration.

Speed Measurement of DC Motor

To start with this project, we need a device that will measure the speed of the motor shaft. There are several methods which can use to measure the speed of motor. Here, we shall discuss about speed measurement of dc motor.

Speed Measurement by Using Tachometer

Tachometer is an instrument that measure the speed of the motor based on the concept of back EMF induced in motor while it is running. The EMF is voltage, appeared on the commutator segments, caused by the rotation in the magnetic field by some external force.

The magnitude of the EMF is given by [1],

$$EMF = K_E \phi N \dots\dots\dots[1]$$

Where K_E = a constant based on motor construction

ϕ = magnetic flux

N = speed of motor (in rpm)

The actual relationship between motor speed and EMF follows and is derived from equation

$$N = (EMF) / K_E \phi \dots\dots\dots[2]$$

Thus, the motor speed is directly proportional to the EMF voltage and inversely proportional to the field flux. For permanent magnet DC motor, when the EMF measured is increases, the speed of the motor is also increases with the gain. So, the speed of motor can be measured by measuring the back EMF using tachometer.

Speed Measurement by Using Optical Encoder

The best way to measure speed is to fit an optical encoder. This shines a beam of light from a transmitter across a small space and detects it with a receiver on the other end. If a disc is placed in the space, which has slots cut into it, then the signal will only be picked up when a slot is between the transmitter and receiver. An example of a disc is shown as Figure.

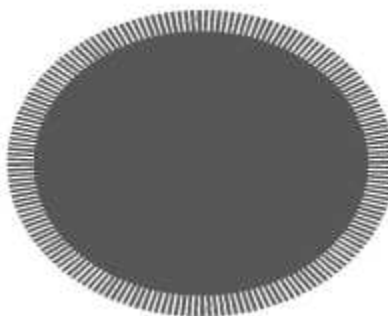


Figure I. Sample disc of encoder

The encoder transmitter must be supplied with a suitable current, and the receiver biased as figure II

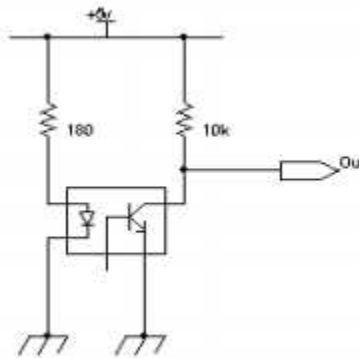


Figure II: Basic schematic circuit of optical encoder

This will have an output which swings to +5v when the light is blocked, and about 0.5 volts when light is allowed to pass through the slots in the disc. The frequency of the output waveform is given by,

$$f_{out} = (N * rpm) / 60$$

where, f_{out} = frequency of output waveform

rpm = speed in revolutions per minutes

N = number of slots at disc

So, from above equation, the speed of DC motor in rpm is given by,

$$Rpm = (f_{out} * 60) / N$$

DC motor Speed Controller

For precise speed control of servo system, closed loop- control is normally used. Basically, the block diagram and the flow chart of the speed control are shown in Figure III. The speed, which is sensed by analog sensing devices (e.g. tachometer) is compared with the reference speed to generate the error signal and to vary the armature voltage of the motor.

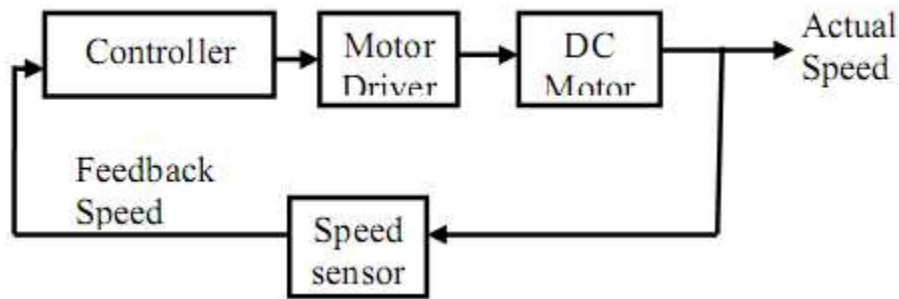


Figure III: Basic block diagram for DC Motor speed control

There are several controllers that can be used to control the speed of the motor such as by using thyristor, phase-locked-loop control, chopper circuit, Fuzzy Logic Controller and etc. Here, we will discuss only at the speed control system by using PWM technique.

Speed Control by Using PWM and Full H Bridge Motor Drive

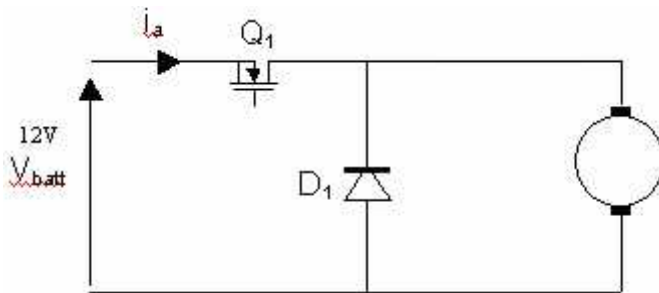


Figure IV: Simple motor circuit

Let us consider a simple circuit that connects a battery as power supply through a switch MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) as shown in Figure IV. When the switch is closed, the motor sees 12 Volts, and when it is open it sees 0 Volts. If the switch is open for the same amount of time as it is closed, the motor will see an average of 6 Volts, and will run more slowly accordingly.

This on-off switching is performed by power MOSFETs. A MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor) is a device that can turn very large currents on and off under the control of a low signal level voltage.

The average of voltage that supply to DC motor is given by,

$$V_{ave} = (t_{on}/T) * V_{in}$$

Where, V_{ave} = average voltage supply to DC motor

t_{on} = time ON of switches

T = period of PWM

t_{on}/T = duty cycle

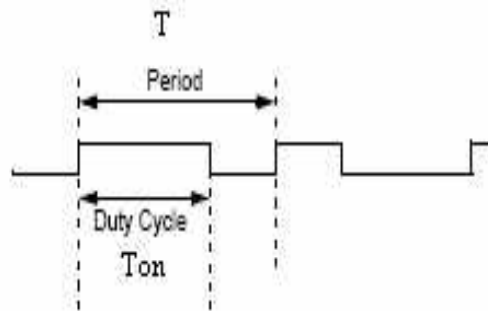


Figure V: PWM signal

As the amount of time that the voltage is *on* increases compared with the amount of time that it is *off*, the average speed of the motor increases and vice versa.

The time that it takes a motor to speed up and slow down under switching conditions depends on the inertia of the rotor (basically how heavy it is), and how much friction and load torque there is.

A full bridge circuit is shown in the diagram below. Each side of the motor can be connected either to battery positive, or to battery negative. Only one MOSFET on each side of the motor must be turned

on at any one time otherwise they will short out the battery and burn out.

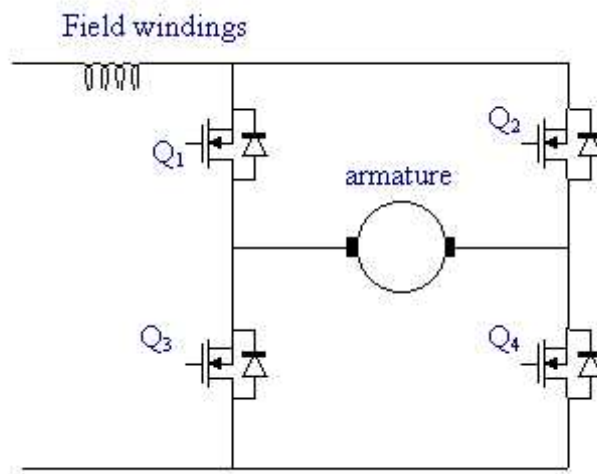


Figure VI:Full H bridge motor drive

To make the motor go forwards, Q4 is turned on, and Q1 has the PWM signal applied to it. Meanwhile, to make the motor go backwards, Q3 is turned on, and Q2 has the PWM signal applied to it:

Microcontroller

Microcontrollers must contain at least two primary components – random access memory (RAM), and an instruction set. RAM is a type of internal logic unit that stores information temporarily. RAM contents disappear when the power is turned off. While RAM is used to hold any kind of data, some RAM is specialized, referred to as registers. The instruction set is a list of all commands and their corresponding functions. During operation, the microcontroller will step through a program (the firmware). Each valid instruction set and the matching internal hardware that differentiate one microcontroller from another.

Most microcontrollers also contain read-only memory (ROM), programmable read-only memory (PROM), or erasable programmable read-only memory (EPROM). All of these memories are permanent: they retain what is programmed into them even during loss of power. They are used to store the firmware that tells the microcontroller how to operate. They are also used to store permanent lookup tables. Often these memories do not reside in the microcontroller; instead, they are contained in external ICs, and the instructions are fetched as the microcontroller runs. This enables quick and low-cost updates to the firmware by replacing the ROM.

Where would a microcontroller be without some way of communicating with the outside world? This job is left to input/output (I/O) port pins. The number of I/O pins per controllers varies greatly, plus each I/O pin can be programmed as an input or output (or even switch during the running of a program). The load (current draw) that each pin can drive is usually low. If the output is expected to be a heavy load, then it is essential to use a driver chip or transistor buffer.

Most microcontrollers contain circuitry to generate the system clock. This square wave is the heartbeat of the microcontroller and all operations are synchronized to it. Obviously, it controls the speed at which the microcontroller functions. All that needed to complete the clock circuit would be the crystal or RC components. We can, therefore precisely select the operating speed critical to many applications.

To summarize, a microcontroller contains (in one chip) two or more of the following elements in order of importance:

- i. Instruction set
- ii. RAM
- iii. ROM, PROM or EPROM

- iv. I/O ports
- v. Clock generator
- vi. Reset function
- vii. Watchdog timer
- viii. Serial port
- ix. Interrupts
- x. Timers
- xi. Analog-to-digital converters
- xii. Digital-to-analog converters

RS232 Serial Port

RS232 is a popular communications protocol for connecting modems and data acquisition devices to computers. RS232 devices can be plugged straight into the computer's serial port (also known as the COM or Comms port). Examples of data acquisition devices include GPS receivers, electronic balances, data loggers, temperature interfaces and other measurement instruments.

A nine pin D plug has become the standard fitting for the serial ports of PCs. The pin connections used are as shown in table below. The connector on the PC has male pins, therefore the mating cable needs to terminate in a DB9/F (Female pin) connector.

Table: RS232 pin assignments (DB9 PC signal set)

Pin 1	Input	DCD	Data Carrier Detect
Pin 2	Input	RXD	Received Data
Pin 3	Output	TXD	Transmitted Data
Pin 4	Output	DTR	Data Terminal Ready
Pin 5			Signal Ground
Pin 6	Input	DSR	Data Set Ready
Pin 7	Output	RTS	Request To Send
Pin 8	Input	CTS	Clear To Send

Pin 9	Input	RI	Ring Indicator
-------	-------	----	----------------

Normal PC hardware might well run with just T_x , R_x and Ground connected, most driver software will wait forever for one of the handshaking lines to go to the correct level. Depending on the signal state it might sometimes work, other times it might not. The reliable solution is to loop back the handshake lines if they are not used.

When the lines are handshake looped, the RTS output from the PC immediately activates the CTS input - so the PC effectively controls its own handshaking.

MPLAB IDE

MPLAB IDE is a Windows-based Integrated Development Environment for the Microchip Technology Incorporated PIC microcontroller (MCU) and dsPIC digital signal controller (DSC) families. In the MPLAB IDE, we can:

- i. Create source code using the built-in editor.
- ii. Assemble, compile and link source code using various language tools. An assembler, linker and librarian come with MPLAB IDE. C compilers are available from Microchip and other third party vendors.
- iii. Debug the executable logic by watching program flow with a simulator, such as MPLAB SIM, or in real time with an emulator, such as MPLAB IDE. Third party emulators that work with MPLAB IDE are also available.
- iv. Make timing measurements.
- v. View variables in Watch windows.
- vi. Program firmware into devices with programmers such as PICSTART Plus or PRO MATE II.

Visual Basic 6.0

There are literally hundreds of programming languages. Each was developed to solve particular type of problem. Most traditional languages, such as BASIC, C, COBOL, FORTRAN, and Pascal are considered procedural languages. That is, the program specifies the exact sequence of all operations. Program logic determines the next instruction to execute in response to conditions and user request.

The newer language, such as C++ and VISUAL BASIC, use a different approach: object-oriented programming (OOP) and event driven programming. Microsoft refers to Visual Basic as an event driven programming language, which has many elements of an object oriented language such as Java. In the event driven model, programs are no longer procedural; they don't follow a sequential logic. So, there is no need to take control and determine the sequence of execution.

As the world turn to graphic user interface (GUI), visual basic is one of the languages that changes to accommodate the shift. Visual Basic is designed to allow the program run under the windows without the complexity generally associated with windows programming. The designed screen can holds standard windows button such as command buttons, check boxes, option buttons, text boxes, and so on. Each of these windows object, operates as expected, producing a “standard” windows user interface.

Visual Basic that recently appears as one of the most popular programming language is chose. It provided standard windows object and graphic user interface that will make the program become user friendly.

METHODOLOGY

Introduction

In this project, microcontroller will be used as the controller to control DC motor speed at desired speed.

The actual speed of DC motor will be measured by encoder and feedback to microcontroller. In microcontroller, it will calculate the error between the desired speed with the actual speed. The error will determine duty cycle of pulse-width- modulation (PWM) in microcontroller. Then, the duty cycle will send to DC motor driver either accelerate or decelerate DC motor to maintain it at desired speed.

Hardware Implementation:

This section will discuss about components that had been used included DC motor, optical encoder, power supply 5V, microcontroller PIC 16F877A, pulse-width-modulation (PWM) and RS232 serial communication of microcontroller, DC Motor drive and loading unit LU150L.

DC Motor

Figure below shows the DC motor that will be used in this project. It is a permanent magnet 12V DC motor. The specification of this motor is shown in Table there.



Figure VII: 12V DC motor

Table: Specification of the motor

Voltage rating	12 V
R.P.M.	750
type	D.C.

Optical Encoder:

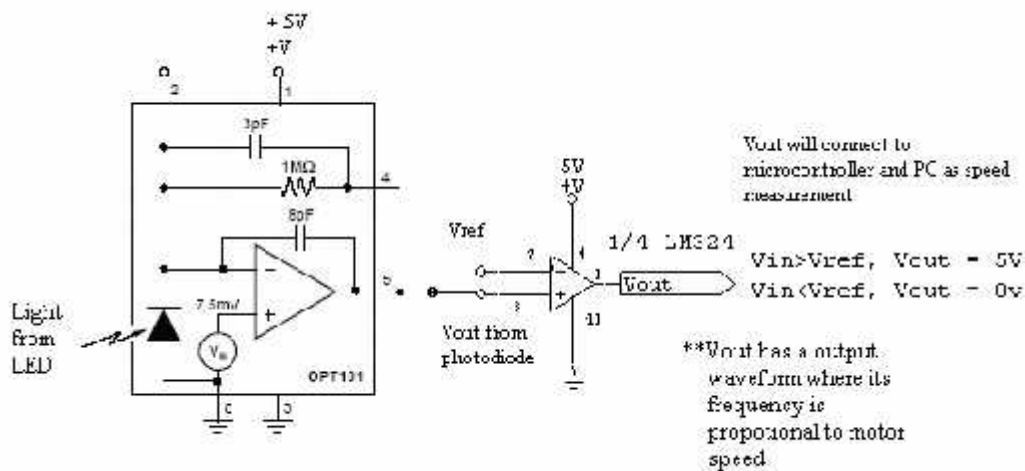
In this project, an optical encoder will be used to measure the DC motor speed. The fundamental reason for the superiority of this system is that the optical encoder used as the velocity sensor, is capable of much better performance than the generator type of tachometer (by using back EMF). When the optical disc is properly mounted on the motor shaft, it generates a frequency directly proportional to motor speed. Changes in gap, temperature, and magnet strength simply have no effect on the output of the optical tachometer. By contrast, an analog tachometer is directly affected by all the problems listed above.

The encoder in the market is very expensive. In order to reduce the cost of the project, an optical encoder is built. Figures show the

basic configuration and the schematic of the optical encoder in action respectively.

The speed accuracy over fractions of a revolution depends on the quality of the optical encoder. The DC motor has an optical disc (made by cardboard) mounted on its shaft. The disc has N radial lines on its surface. In this project, we will make four slots on the disc (N=4). This will give a resolution of 1/4 in one rotation. An LED (light emitting diode) as transmitter is put at one side of the disc and a photodiode, as receiver is fixed on the other side of the disc. Chip OPT 101 is selected as photodiode in this project.

To make sure the output waveform in digital signal (High/Low) which is readable by microcontroller, Chip LM324 is used as a comparator. When the V_{out} of photodiode is less than V_{ref} , the output of LM324 will be 0V (Low) and when the V_{out} is greater than V_{ref} , the output of LM 324 will be 5V (High). The output signal form LM324 has a frequency which is given below. Then, the output signal will be sent to microcontroller as representation of actual speed.



Power Supply +5 volts:

Most digital logic circuits and processors need a +5 volt power supply. To use these parts we need to build a regulated +5 volt source. Usually we start with an unregulated power supply ranging from 9 volts to 24 volts DC.

To make a +5 volt power supply, we use a LM7805 voltage regulator IC

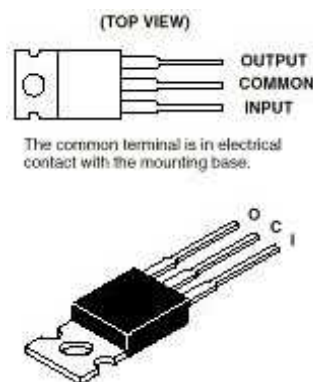


Figure VIII: IC LM7805

Sometimes the input supply line may be noisy. To help smooth out this noise and get a better 5 volt output, capacitors is usually added to the circuit.

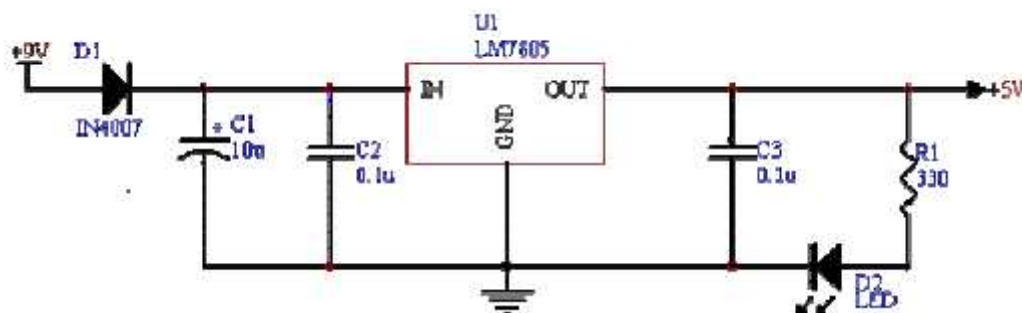
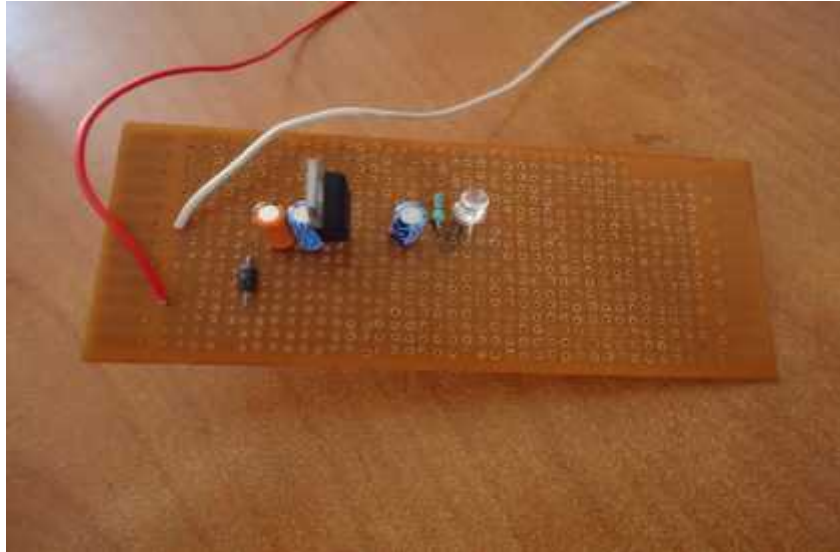


Figure IX: Schematic circuit of +5V power supply



Picture of +5 volt supply in our project.

Microcontroller PIC 16F877A

The microcontroller acts like the brain of the DC motor speed control system. The microcontroller chip that has been selected for the purpose of controlling the speed of DC motor is PIC16F877A manufactured by Microchip. This chip is selected based on several reasons:

- i. Its size is small and equipped with sufficient output ports without having to use a decoder or multiplexer.
- ii. Its portability and low current consumption.
- iii. It has PWM inside the chip itself which allow us to vary the duty cycle of DC motor drive.
- iv. It is a very simple but powerful microcontroller. Users would only need to learn 35 single word instructions in order to program the chip.

- v. It can be programmed and reprogrammed easily (up to 10,000,000 cycles) using the universal programmer in robotics lab.

Refer to Table below for the pin connection of PIC16F877A in DC Motor speed control system. Pins not stated in the table are not used and left hanging. Figure X shows the schematic circuit of microcontroller PIC16F877A. At the beginning, microcontroller will receive desired speed from PC through serial port. The detected motor speed from optical encoder will feedback to microcontroller through RA0 of PIC16F877A. The microcontroller will operate as it programmed (detail program at appendix) to produce a new duty cycle (from CCP2) that proportional to the error speed. Thus, average of voltage supply from DC motor drive can be varied in order to maintain the speed at desired value.

Table: Pin connection of PIC16F877A for DC motor speed control system

Pin name	Pin no	Description	Application
VDD	11,32	Positive supply (+5v)	Power supply to chip
VSS	12,31	Ground Reference	Ground Reference
OSC1	13	For oscillator or resonator	Connected to resonator 20MHz with 22pF
OSC2	14	For oscillator or resonator	Connected to resonator 20MHz with 22pF
MCLR\	1	Reset input	Always connected to +5V
RA0	2	Input/Output pin	Input of Vout from LM324 as

			speed counter
RB1	34	Input/Output pin	Output to control CW/CCW of left motor
RB2	35	Input/Output pin	Output to control CW/CCW of left motor
CCP2	16	Capture/Compare/PWM	Output of duty cycle(PWM) to control motor speed

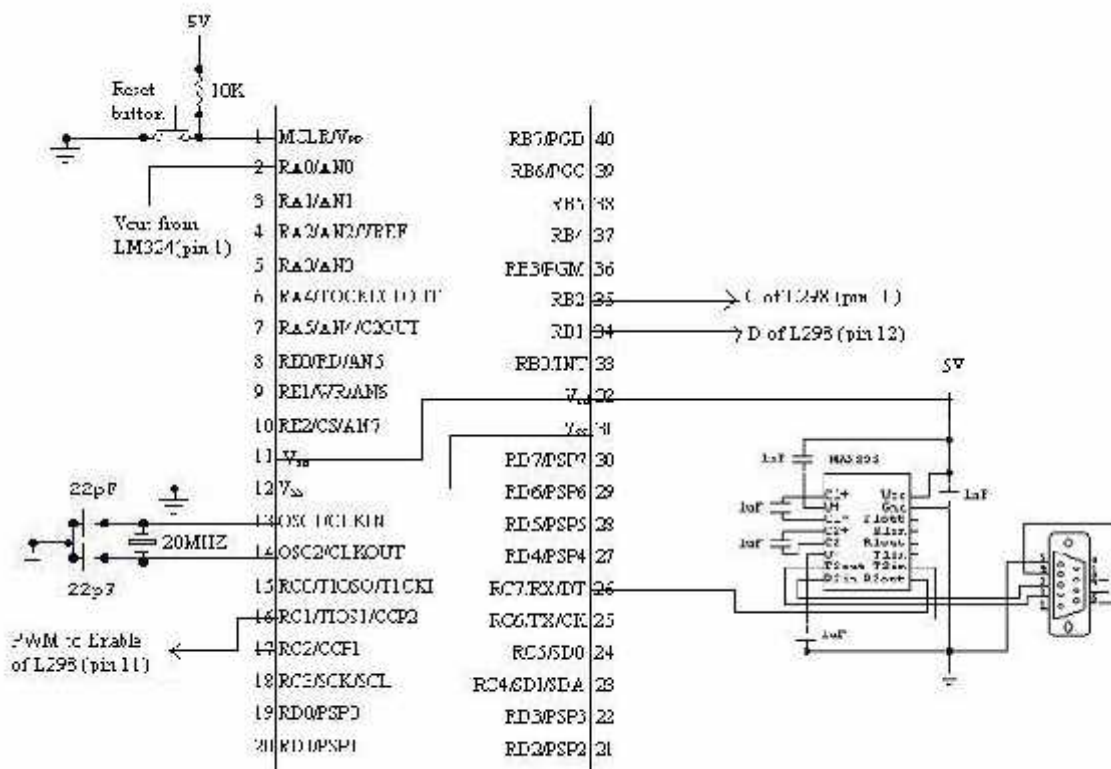
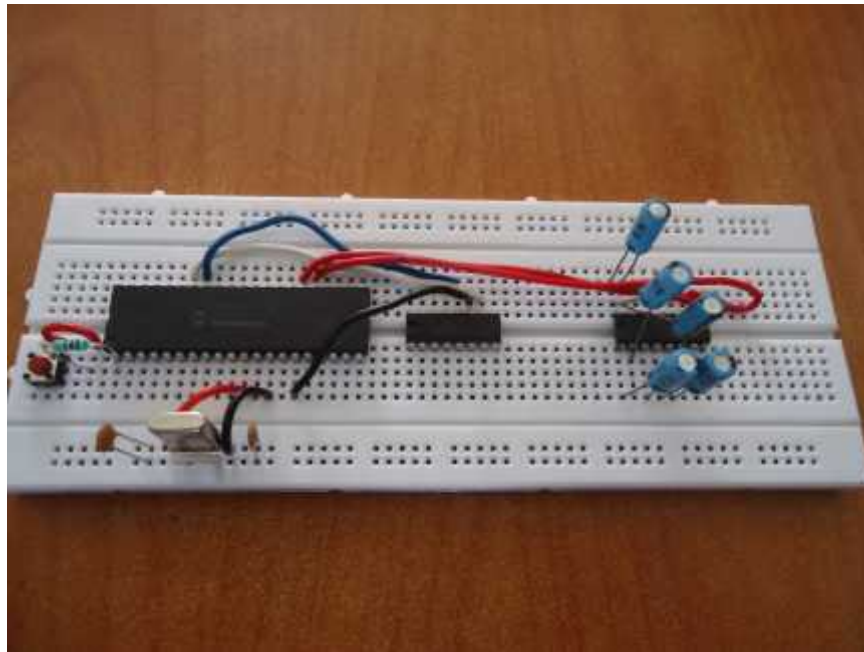


Figure X: Schematic circuit of PIC16F877A



Picture of circuit of PIC16F877A

Pulse-Width-Modulation (PWM) in Microcontroller:

The Pulse-Width-Modulation (PWM) in microcontroller is used to control duty cycle of DC motor drive.

PWM is an entirely different approach to controlling the speed of a DC motor. Power is supplied to the motor in square wave of constant voltage but varying pulse-width or duty cycle. Duty cycle refers to the percentage of one cycle during which duty cycle of a continuous train of pulses. Since the frequency is held constant while the on-off time is varied, the duty cycle of PWM is determined by the pulse width. Thus the power increases duty cycle in PWM.

The expression of duty cycle is determined by,

$$\%Duty\ cycle = (t_{on}/T)*100\% \dots\dots\dots[3]$$

Basically, the speed of a DC motor is a function of the input power and drive characteristics. While the area under an input pulse

width train is measure of the average power available from such an input.

RS232 Serial Communication:

SCI is an abbreviation for Serial Communication Interface and, as a special subsystem of microcontroller PIC16F877A. It provides RS232 serial communication with PC easily.

As with hardware communication, we use standard NRZ (Non Return to Zero) format also known as 8 (9)-N-1, or 8 or 9 data bits, without parity bit and with one stop bit. Free line is defined as the status of logic one. Start of transmission - Start Bit, has the status of logic zero. The data bits follow the start bit (the first bit is the low significant bit), and after the bits we place the Stop Bit of logic one. The duration of the stop bit 'T' depends

on the transmission rate and is adjusted according to the needs of the transmission. For the transmission speed of 9600 baud, T is 104 μ s.

In order to connect a microcontroller to a serial port on a computer, we need to adjust the level of the signals so communicating can take place. The signal level on a PC is -10V for logic zero, and +10V for logic one. Since the signal level on the microcontroller is +5V for logic one and 0V for logic zero, we need an intermediary stage that will convert the levels. One chip specially designed for this task is MAX232. This chip receives signals from -10 to +10V and converts them into 0 and 5V.

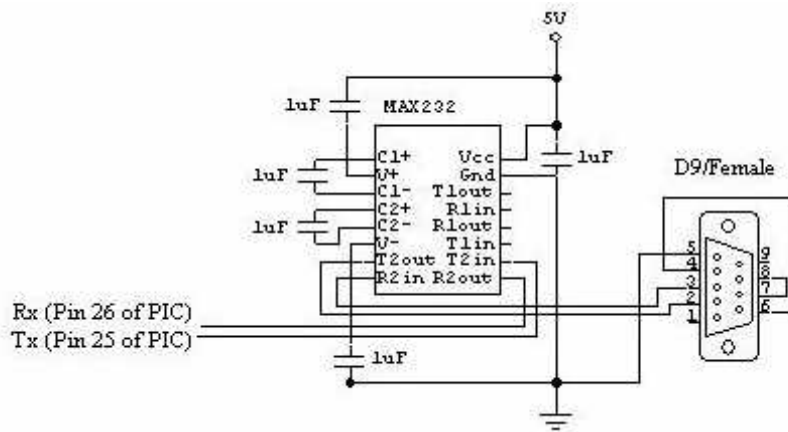


Figure XI: Connection between D9 Female serial port, MAX232 and PIC16F877A

Software Implementation

For software implementation, MPLAB IDE is used to program microcontroller in assembly language. Besides, Visual Basic 6.0 is used for user interface purpose and for monitoring the speed response of the system.

Algorithm and Programming in MPLAB IDE

Microcontroller acts as brain of the whole DC motor speed control system. It will receive the desired speed from user through PC that interface with RS232 serial port. The actual speed will be compared with the desired speed and the correction will be done by microcontroller to always maintain the DC motor speed at the desired speed.

There are two parts of the program which are main program and interrupt program. The microcontroller will always run the main program until there is an interrupt occurred. When microcontroller receives an interrupt flag, then it will jump to interrupt process.

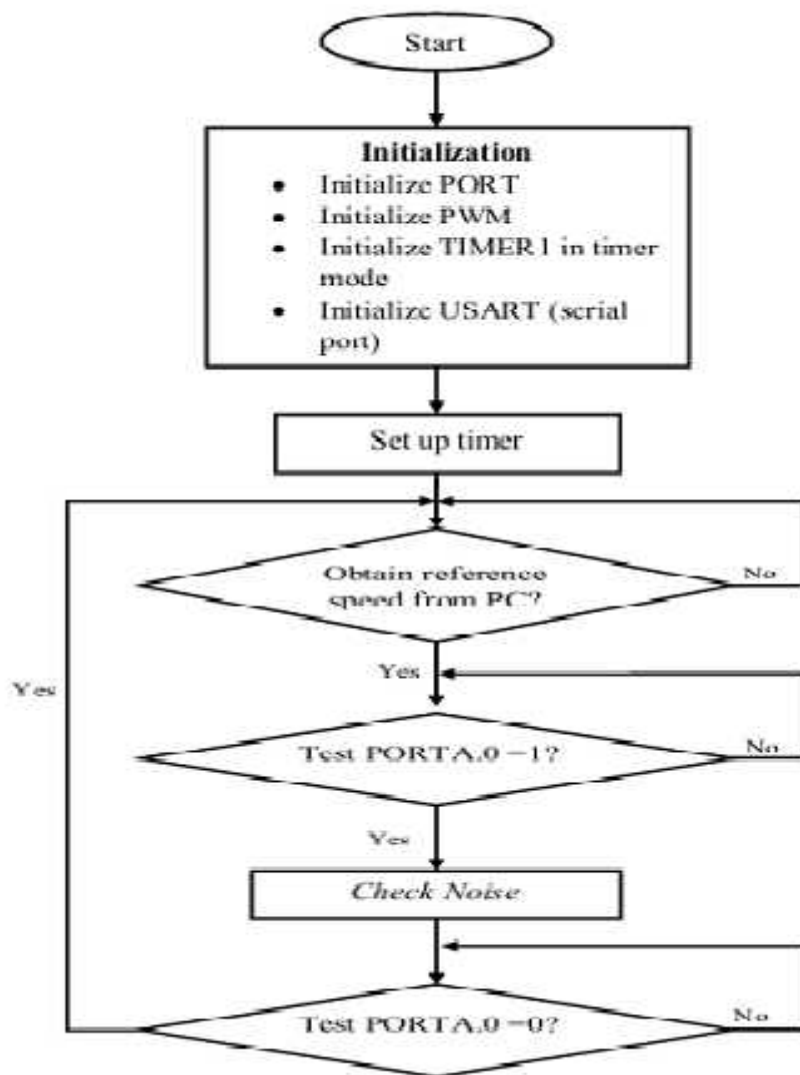


Figure XII: Flow chart of microcontroller's main program

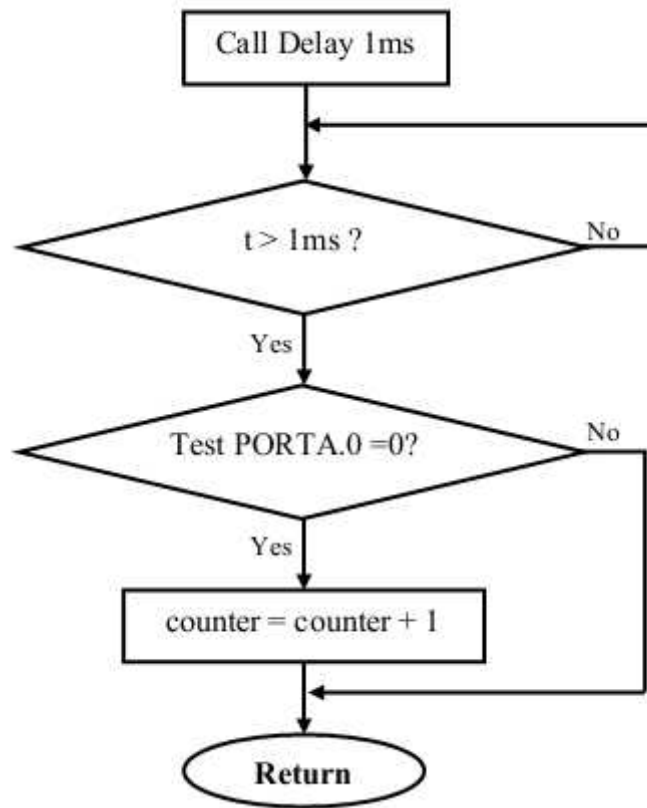


Figure XIII: Flow chart of check noise function

Processing Explanation of Main Program

There are six main parts of main program in microcontroller. There are initialization of ports, PWM, Timer1, setup for serial port, get reference speed and check noise function.

a) Initialization of the mode of ports A ,B

In this project, we use pin 0th of the Port A (RA0) as digital input where it receives input (H/L) from LM324 (pin1). A user register, X1 has been defined as speed counter for pulses received from RA0. When the input of RA0 is High (H), it will increase speed counter X1 where $X1 = X1 + 1$. Otherwise, the counter will remain its value.

All pins of Port B are set to output. RB1 is always set to High (H) and RB2 set to Low (L) to make the motor run in forward direction as described in appendix.

b) Initialization of PWM

A PWM output (Figure 3.14) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

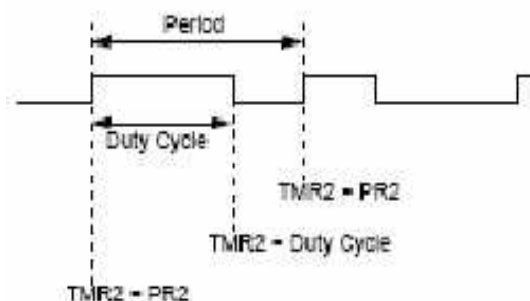


Figure XIV: PWM output

To setup for PWM operation, the following steps should be taken when configuring the CCP module for PWM operation:

- i. Set the PWM period by writing to the PR2 register.
- ii. Set the PWM duty cycle by writing to the CCPR2L register and CCP2CON<5:4> bits.
- iii. Make the CCP1 pin an output by clearing the TRISC<2> bit.
- iv. Set the TMR2 pre-scale value and enable Timer2 by writing to T2CON.
- v. Configure the CCP2 module for PWM operation.

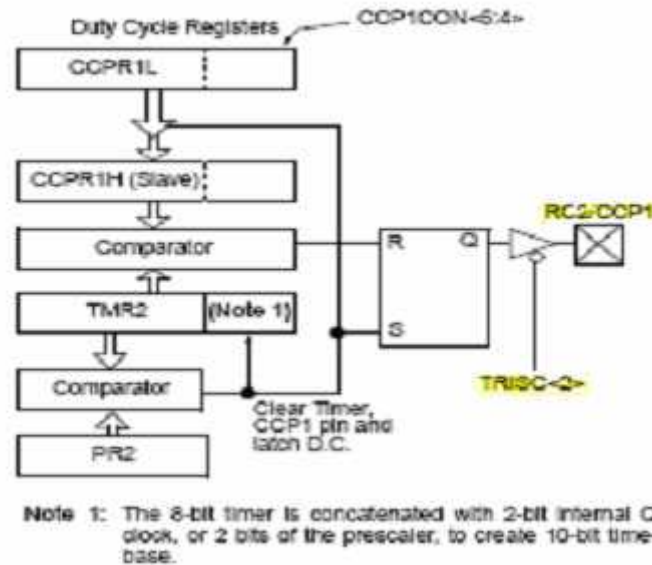


Figure XV: Simplified PWM block diagram

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] * 4 * TOSC * (TMR2 \text{ prescale value})$$

In this project, PR2 is set to 255 (maximum value of 8 bit for maximum range), so the PWM period is then become

$$\text{PWM period} = (255+1) * 4 * (1/20M) * (1)$$

$$= 51.2\mu\text{s}$$

$$= 19.53\text{kHz}$$

where $F_{\text{osc}} = 20\text{MHz}$ and TMR2 prescale =1

The PWM duty cycle is specified by writing to the CCPR2L register and to the CCP2CON<5:4> bits. Up to 10-bit resolution is available. The CCPR2L contains eight MSbs and the CCP2CON<5:4> contains two LSbs. This 10-bit value is represented by CCPR2L:CCP2CON<5:4>. The following equation is used to calculate the PWM duty cycle in time: **PWM**

$$\text{duty cycle} = (\text{CCPR2L:CCP2CON<5:4>}) * T_{\text{OSC}} * (\text{TMR2 prescale value})$$

The PWM duty cycle is specified by writing to the CCPR2L register and to the CCP2CON<5:4> bits. Up to 10-bit resolution is available. The CCPR2L contains eight MSbs and the CCP2CON<5:4> contains two LSbs. This 10-bit value is represented by CCPR2L:CCP2CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR2L:CCP2CON<5:4>}) * T_{\text{OSC}} * (\text{TMR2 prescale value})$$

c) Initialization of TIMER1 in Timer Mode

The Timer1 module in PIC16F877A is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L), which are readable and writable. Figure shows the Timer1 block diagram which operate in timer mode. The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh (decimal from 0 to 65535) and rolls over to 0000h. The TMR1 Interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt

can be enabled/disabled by setting/clearing TMR1 interrupt enable bit, TMR1IE (PIE1<0>). Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is FOSC/4.

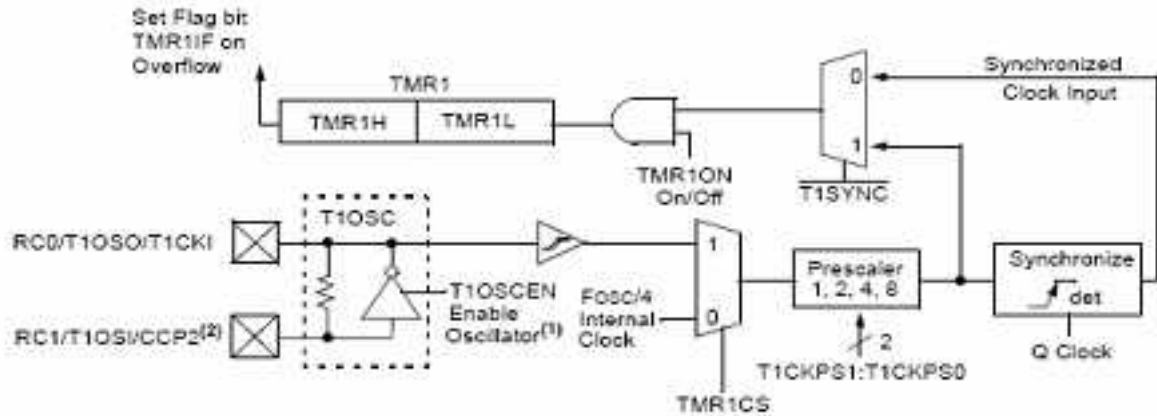


Figure 3.16 Timer1 block diagram

When Timer1 increases from 0000h to FFFFh, it takes 65535 cycle and the time consumed is given by,

$$t = (65535 - 0) * (1/(20\text{MHz}/4)) ; F_{osc} = 20\text{MHz}$$

$$= 13.107 \text{ ms.}$$

In this project, the interrupt is required to occur each 0.39321s where the pulses counted in X1 will be converted to speed and correction taken. A user register, X2 is defined as gain for this purpose.

$$\text{Gain, X2} = 0.39321\text{s} / 13.107\text{ms}$$

$$= 30$$

Besides, the interruption enable bit of CCP2 is set. Also GIE and PIE are set.

d) Setup for serial port

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.)

The USART is configuring in asynchronous mode. In this mode, the USART uses standard non-return-to zero (NRZ) format (one START bit, eight or nine data bits, and one STOP bit). The most common data format is 8-bits and it is used in this project. An on-chip, dedicated, 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The transmitter and receiver are functionally independent, but use the same data format and baud rate.

$$\text{Baud Rate} = F_{\text{soc}} / (16(\text{SPBRG}+1))$$

To set baud rate at 9600bps, we have to set SPBRG as

$$\begin{aligned}\text{SPBRG} &= (F_{\text{soc}} / \text{Baud Rate} / 16) - 1 \\ &= 20\text{M} / 9600 / 16 - 1 \\ &= 129\end{aligned}$$

Bit SPEN (RCSTA<7>) is set as 1 and bits TRISC<7:6> are set as 0 in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Asynchronous Receiver Transmitter.

e) **Get reference speed**

At beginning of the program, microcontroller will get the reference speed from user at PC through RS232 serial communication. It will wait there until it gets the reference speed from user.

f) **Check Noise function**

From experimental result, it is found that there are some noise besides real pulses from LM324, especially when motor running at high speed. This noise can cause over reading pulses at speed counter, X1 if it is not filter out. From experiment, it is found that the noise have a bandwidth less than 1ms (bandwidth of real pulses always greater than 1ms). So, when a High (H) pulses is receive at RA0, it will hold for 1ms. If the pulse detected still High (H), it means that the received is real pulse and the X1 will increase it value with one. But if the pulse detected is Low (L), then it is noise and ignored where X1 remains its value.

Processing Explanation of Interrupt Process

Interrupt process is occurred in each 0.39321s. The microcontroller will execute the program in interrupt process instead of main program that is running when it receives the interrupt flag. Each part of interrupt process is discussed in the following section.

a) Clearing of interruption flag

The interruption is occurs every 0.39321s with CCP2. The interruption flag of CCP2 should be cleared first. When not clearing this, the following interruption occurs without waiting desired time.

b) Speed counter

At each interruption (each 0.39321s), the value of speed counter, X1 will be taken as detected pulses. From optical encoder with 4 slots that built in this project, we know that it will produces 4 pulses when motor turn for 1 round. Data from speed counter, X1 in microcontroller will be loaded to PC each 0.39321s. So, the motor speed is given by,

detected speed=(detected pulses/N)*(1/t) Rounds/s

where, N = number of slots

t = time of getting data in second

detected speed= (detected pulses/4)*(1/.39321) Rounds/s

= (detected pulses/4)*(1/.39321) *60 RPM

c) Error speed

The detected speed will be compared to reference speed to calculate the error between them.

The error speed will then be converted into error voltage by using Equation from experiment with optical encoder that has been built.

Finally, the error voltage will be converted to duty cycle based on Equation to determine to speed up or slow down the motor. If there is no error, the duty cycle of PWM is remained.

Error voltage =(Error speed+80.89)/207.16

Finally, the error voltage will be converted to duty cycle based on Equation to determine to speed up or slow down the motor. If there is no error, the duty cycle of PWM is remained.

Error DC=Error voltage/ V_{in}

Where, $V_{in} = 12V$

d) Speed down process

When the detected speed of the motor is higher than the reference speed, a duty ratio is decreased and a motor drive electric current is suppressed. The rate that the duty ratio becomes smaller is decided by error speed.

e) Speed up process

When the detected speed of the motor is lower than the reference speed, a duty ratio is increased and a motor drive electric current is increased. The rate that the duty ratio becomes bigger is decided by error speed.

f) Send detected speed to PC

To monitor the performance of the system, detected speed will sent to PC each 0.39321s as detected pulses. This value will then convert to represent the detected speed (rpm) in PC by using Visual Basic 6.0 program.

g) Interruption ending process

The RETFIE instruction is executed at end of the interruption processing. Before that, the counter and timer have to clear and reset.

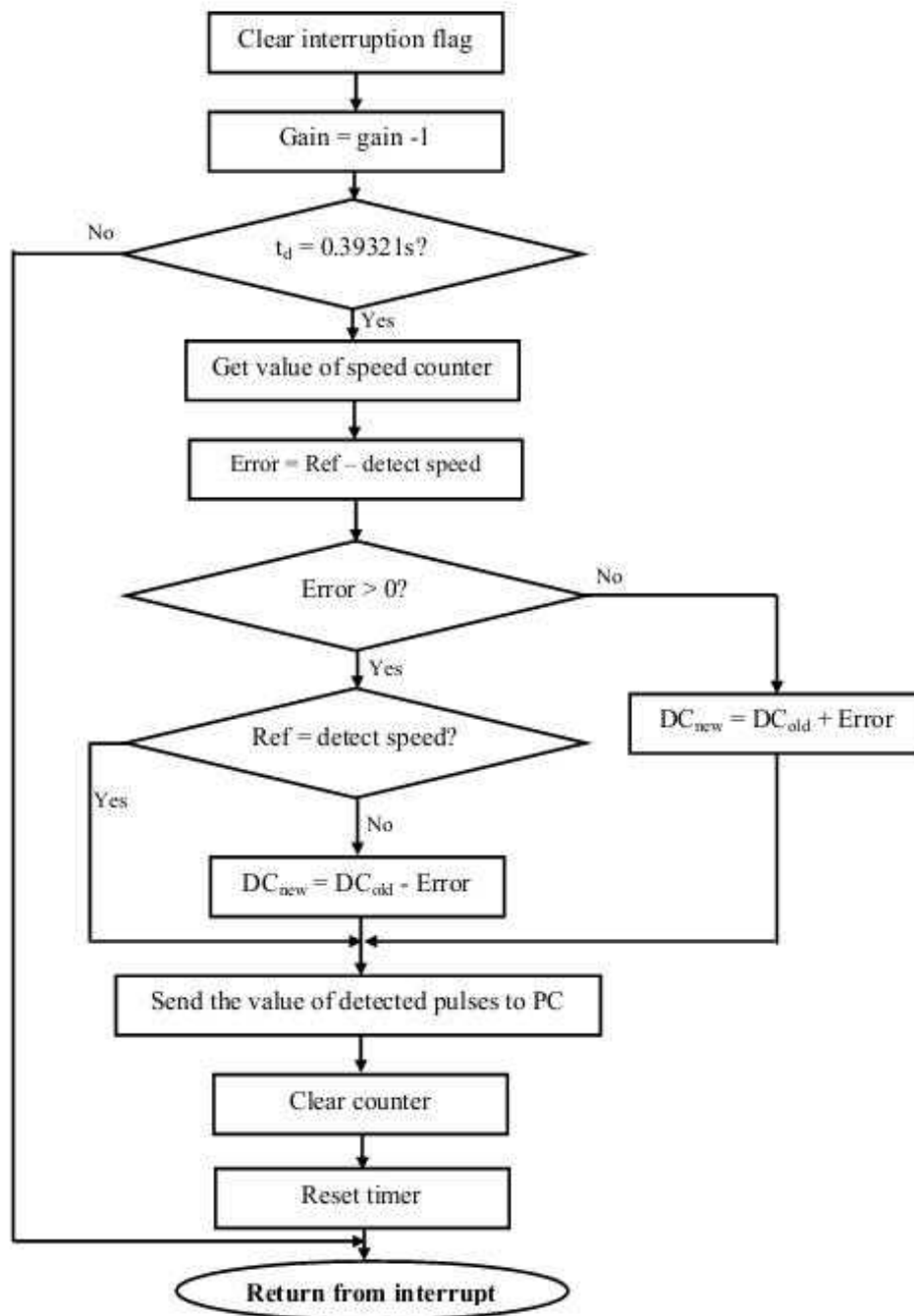


Figure 3.19 Flow chart of microcontroller's interrupt process

Programming in Visual Basic 6.0

In this project, a program will be developing using Visual Basic 6.0. This program is able to send data (desired speed from user) to microcontroller and plot a graph of detected speed versus time to monitor the performance of the system.

Reading Input from User and Microcontroller

To obtain desired speed from user, the user is required to select a desired speed at computer. The desired speed (rpm) will then convert into desired pulses before send to microcontroller through serial port. Equation of desired pulses is given by,

$$\text{desired pulses} = (\text{desired speed} * 4 * 0.39321) / 60$$

Plot Graph Speed versus Time

To monitor performance of the system, a graph of detected speed versus time will be obtained by using Visual Basic. The user will sent the desired speed to microcontroller by click on Run command button. It is also allowed to get a new graph by click on the New Graph command button.

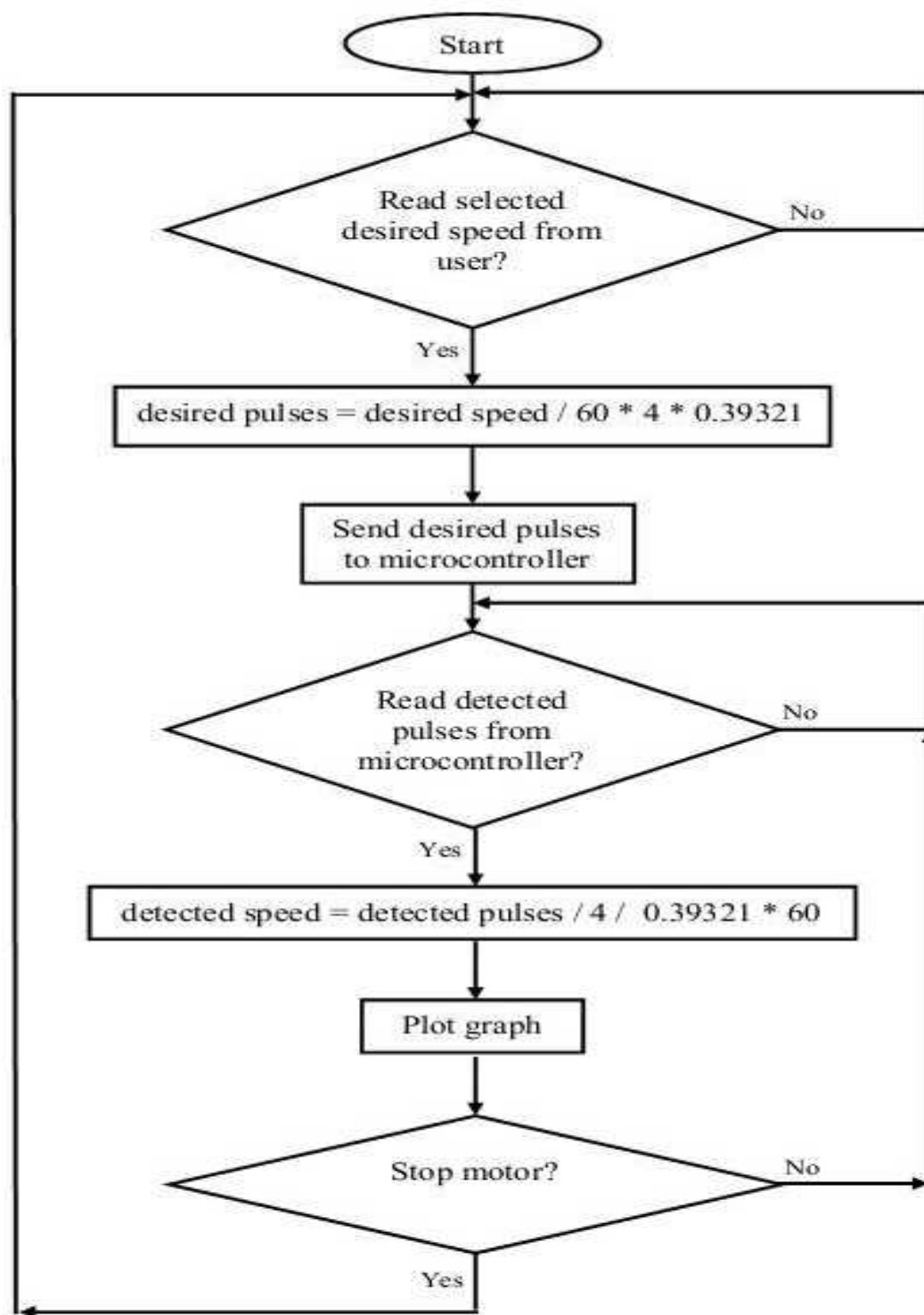


Figure 3.22 Flow chart of program in Visual Basic 6.0

RESULT AND DISCUSSION

Introduction

Some experiments had been conducted for the project. First and foremost, an experiment is conducted to find out the relationship between voltage supply and motor speed. Then, data collection is done at each speed for DC motor speed control system to observe performance of the system. Last but not least, an analysis on the dead time of the system is made.

Experiment:

Determine Relationship of Voltage Supply and Motor Speed.

An experiment is conducted to determine the relationship between voltage supply and speed. The procedures and the result will be discussed in following sections.

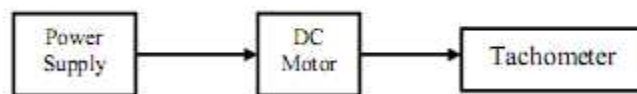


Figure 4.1 Experiment with tachometer

Procedures

- 1) The circuit was connected as Figure 4.1.
- 2) Voltage of 0.5V was supplied to motor.
- 3) Value of rpm at tachometer was recorded in Table 4.1.

4) The voltage increased in steps of 0.5V until 16V and step 3 was repeated.

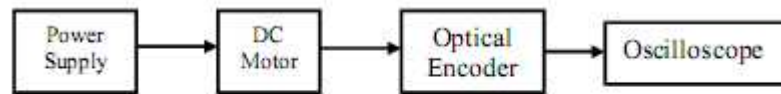


Figure 4.2 Experiment with optical encoder

Experimental Result Analysis:

The accuracy of the optical encoder can be checked by calculate its standard deviation's for percentage of error by using following Equation

$$s = \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{n-1}}$$

The standard deviation percentage of error is quite small, so it can be concluded that the reading of the optical encoder for speed measurement is quite reliable.

DC Motor Speed Control Result:

Microcontroller acts as proportional (P) controller in the DC motor speed control system. At each speed, the result was collected by applying normal load, overload and then suddenly the load was as no load condition.

DC Motor Speed Control Result Analysis:

For P controller, as the controller proportional gain, K_p is increased; the response to set point changes becomes more oscillatory, commonly called underdamped. In this project, the speed response will give an underdamped response when $K_p=1$.

At some greater gain, the response of the control loop becomes a steady-state oscillation. The system is called “marginally stable”. If the gain is increased past the point where steady oscillation is observed, the control loop will become unstable and the oscillations will increase in amplitude.

Speed versus time for a range of rpm, the controller is able to control the speed at their desired speed when applying normal load and overload. Without controller, the motor will slow down or maybe die out.

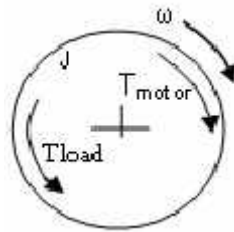


Figure: Free body diagram of the disc

$$T_{\text{motor}} - T_{\text{load}} = J \cdot \omega$$

Where, J = moment of inertia of the wheel about the axis of rotation

T_{load} = torque induced by load

T_{motor} = torque induced by voltage supply &

ω = speed motor

CONCLUSION AND RECOMMENDATION

Conclusion

Recent developments in science and technology provide a wide range scope of applications of high performance DC motor drives in area such as rolling mills, chemical process, electric trains, robotic manipulators and the home electric appliances require speed controllers to perform tasks. DC motors have speed control capabilities, which means that speed, torque and even direction of rotation can be changed at anytime to meet new condition.

The goal of this project is to design a DC motor speed control system by using microcontroller PIC16F877A. It is a closed-loop real time control system. The controller will maintain the speed at desired speed when there is a variation of load. By varying the PWM signal from microcontroller (P controller) to the motor driver, motor speed can be controlled back to desired value easily.

For this project, by applying $K_p = 1$ to P controller in microcontroller, the speed response become underdamped response. If $K_p < 1$, the speed response is not satisfied. At some greater gain, the speed response of the control loop becomes a steady-state oscillation. If the gain is increased past the point where steady oscillation is observed, the control loop will become unstable and the oscillations

will increase in amplitude. The motor will suddenly speed up and it will damage the motor.

In conclusion, with the $K_p = 1$ for P controller at microcontroller PIC 16F877A, the motor speed response can be maintained at desired value although there is a variation of load. The objective of the project is successfully fulfilled.

Although the controller can function as we expected, but the performance is slightly sluggish where it takes about 2 or 3 second to react properly when there is a disturbance especially at low speed. This is what we need to overcome in order to achieve quick control of motor speed smoothly.

Besides, there is a constraint with the optical encoder that had been built. Based on Equation 3.4, if we can increase the resolution of optical encoder by increasing its number of slots, then the sensitivity of optical encoder can be increased. So, the time for getting data and for microcontroller take correct action will be reducing where $t < 0.39321s$. As a result, the controller can react faster when there is a disturbance.

However, due to the size of photodiode which is quite big, the number of slots at cardboard that can be used is only four. Otherwise, the light of the LED cannot be blocked by the cardboard and photodiode will always give High (H) output. Then, the speed sensor is fail.

Recommendation

The performance of the system is slightly sluggish. For future works, some recommendations have been listed based on the problems in order to improve the performance.

i. Mathematical modeling of motor response

Mathematical model can be obtained from the graph of motor speed response. Then, from the mathematical model, it can be simulated using software such as Matlab to get the improved motor speed response by using controller packages such as PID controller, Fuzzy Logic Controller and others. Besides, it will reduce the total hardware complexity and cost at the same time.

ii. Hardware Improvement

Use infra red (IR) as transmitter and receiver for optical encoder for more narrow light. So, these will allow us to increase the resolution of optical encoder by increasing the number of slots. Thus, it will reduce the steady state error. Besides, time for getting data and for controller to take action also can be reduced. So, the motor speed response will become better.

iii. Software Improvement

Use fuzzy logic microcontroller which combine the idea of fuzzy logic in microcontroller to obtain a DC motor speed control system with excellent regulation and high robustness.

REFERENCES

1. Muhammad H. Rashid. Power Electronics Circuits, Devices and Applications. 3rd edition. United States of America: Prentice Hall. 2004.
2. P.C. Sen and M. L. MacDonald. Thyristorized DC Drives with Regenerative Braking and Speed Reversal. IEEE Transactions on Energy Conversion, 1978, Vol. IECI-25, No. 4: 347-354.
3.
<http://homepages.which.net/paul.hills/SpeedControl/SpeedControllersBody.html>
4. Iovine John. PIC Microcontroller Project Book. 2nd Edition. Mc. Graw-Hill. 121-123; 2000.
5. MPLAB IDE, Simulator, Editor User's Guide
6. <http://www.microchip.com>
7. <http://www.wikipedia.com>
8. Electrical Machinery by P.S. Bimbhra

APPENDIX A

Program

list p=16f877A ; list directive to define processor

ERRORLEVEL -302

ERRORLEVEL -305

#include <p16f877A.inc> ; processor specific variable definitions

__CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
_HS_OSC & _WRT_OFF & _LVP_ON & _CPD_OFF

===== HARDWARE CONNECTION=====

; Crystal - 20MHZ

; Capacitor - 22pF

; RB1(PIN34) & RB2 (PIN35) - CONNECT TO L298(H,L)

; CCP2 (PIN16) - CONNECT TO ENABLE L298

; RA0 as speed counter

=====

=====MACRO CHANGE BANK =====

BANK0 MACRO ;Change to BANK0

BCF STATUS,RP0

BCF STATUS,RP1

ENDM

BANK1 MACRO ;Change to BANK1

BSF STATUS,RP0

BCF STATUS,RP1

ENDM

=====

;Set up initialize value and cblock for the variables needed.

BAUD_RATE EQU 0X81 ;Set Baud Rate 9600 for 20MHZ for USART
Communication

CBLOCK 0X20

D1 ;For Delay

D2 ;For Delay

D3 ;For Delay

X1 ;For Counter

X2 ;For Gain

REFERENCE ;For Reference Speed

CHANGE ;For error correction

ENDC

----- ORG 0x00

GOTO MAIN

ORG 0x04

GOTO INT ;Interrupt Process

;-----

;===== MAIN PROGRAM =====

MAIN BANK0

CALL INIT ;Intialization

STANDBY CALL SERIALSCAN ; Get data for reference speed

MOVF REFERENCE,W

SUBLW 0X00

BTFSC STATUS,Z ; Reference speed = 0?

GOTO STANDBY ; Yes. Get data again

MOVLW D'30' ; Set Timer1 as $30 \times 13.107\text{ms} = 0.39321$

where X2 as gain

MOVWF X2

;----- RA0 - AS SPEED COUNTER -----

LOOP BTFSS PORTA,0

GOTO \$-1

CALL CHECKNOISE

BTFSC PORTA,0

GOTO \$-1

BTFSS PIR1,TMR1IF

GOTO LOOP

;Check Noise function

CHECKNOISE CALL DELAY1 ;If bandwith is less than 1ms, then it is noise

BTFSC PORTA,0

INCF X1,1 ;Not noise, increase counter

RETURN

;=====

#####INITIALIZATION#####

INIT BANK0

CLRF PORTA

CLRF PORTB

CLRF PORTC

CLRF PORTD

CLRF X1

CLRF X2

CLRF REFERENCE

;---Initialize PORT-----

```

BANK1
MOVLW 0x06
MOVWF ADCON1 ;PortA as digital input
MOVLW 0xFF ;PortA as input
MOVWF TRISA

MOVLW 0x00 ;Set PortB as output
MOVWF TRISB
BANK0
BSF PORTB,1 ;RB1,RB2(pin 34 and 35) sambung ke
pin 10 dan 12 of L298
BCF PORTB,2

;----Initialize PWM-----
BANK1
MOVLW 0xFF ;PWM Setup: Period KHZ(19.152KHZ)
MOVWF PR2
BCF TRISC,1
BCF TRISC,2

BANK0
CLRF TMR2
MOVLW 0X00 ;Duty Cycle = 0%
MOVWF CCPR2L
MOVLW 0x04 ;ON TMR2, PRESCALE = 1
MOVWF T2CON
MOVLW 0x0C
MOVWF CCP2CON ;PWM Mode

;-----Set TIMER1 AS TIMER MODE-----
BANK0
MOVLW 0X00
MOVWF TMR1H ;Timer= 0000-FFFFH =
; 65535*1/(20M/4)=13.107ms
MOVLW 0X00
MOVWF TMR1L
MOVLW b'00000001' ;Pre=1:1 TMR1=Int TMR1=ON
MOVWF T1CON

BANK1
MOVLW b'00000001' ;TMR1IE=1
MOVWF PIE1
BANK0
MOVLW b'11000000' ;GIE=1, PEIE=1
MOVWF INTCON

;-----Set up Serial Port-----
SERIAL_SETUP: BANK1
MOVLW 0XC0 ;RC6&7->Input,Others output
IORWF TRISC,F ;Keep in file register

```

```
MOVW BAUD_RATE  
MOVWF SPBRG  
MOVLW 0X24  
MOVWF TXSTA      ;Enable transmission & high baud rate  
  
BANK0  
MOVLW 0X90  
MOVWF RCSTA      ;Enable serial port & continuous reception  
  
RETURN  
  
##### END OF INITIALIZATION#####  
  
^^^^^^^^^^^^^^^^ SUBROUTINE ^^^^^^^^^^^^^^^^^  
  
;-----Correction of Overrun Error-----  
  
OVERRUN_ERROR BCF RCSTA,CREN ;Disable continuous reception  
BSF RCSTA,CREN ;Enable continuous reception  
RETURN  
  
;-----Correction of Framming Error-----  
  
FERR_ERROR MOVF RCREG,W ;Discard Framming Error  
RETURN  
  
;-----Send the detect speed to PC-----  
  
SERIAL_TRANSMIT: BTFSS PIR1,TXIF      ;Check if data TXREG is transfer  
to TSR ->TXREG is empty  
GOTO $-1  
MOVWF TXREG  
RETURN  
  
;-----Get data for reference speed -----  
  
SERIALSCAN CALL OVERRUN_ERROR ;Correction of Overrun Error  
CALL FERR_ERROR      ;Correction of Framming Error  
BTFSS PIR1,RCIF ;Check if data receive  
GOTO $-1 ;Wait until new data  
  
MOVF RCREG,W ;Get received data to W  
MOVWF REFERENCE  
RETURN  
  
;-----Short Delay-----  
  
DELAY1 MOVLW D'5' ;PAUSE FOR ABOUT 1mS  
MOVWF D3  
MOVLW D'9'
```

```
MOVWF D2
MOVLW D'36'
MOVWF D1
DECFSZ D1
GOTO $-1
DECFSZ D2
GOTO $-5
DECFSZ D3
GOTO $-9
RETURN
```

```
DELAY2 MOVLW D'255' ;PAUSE FOR ABOUT 1mS
```

```
MOVWF D3
MOVLW D'255'
MOVWF D2
MOVLW D'36'
MOVWF D1
DECFSZ D1
GOTO $-1
DECFSZ D2
GOTO $-5
DECFSZ D3
GOTO $-9
RETURN
```

;

```
***** INTERRUPT PROCESS*****
```

```
INT CLRIF PIR1 ; Clear interruption flag
DECFSZ X2,F ; Haven't reach 0.799527s
GOTO CONTINUE ; Continue counter
```

```
GETDATA  MOVF  X1,W  ; Get value of counter,detect speed
SUBWF   REFERENCE,W  ; Ref speed - Detect speed
MOVWF   CHANGE      ; CHANGE = error correction
```

```
BTFSCL STATUS,C      ; Reference < Detect ?
      GOTO GREATER    ; No. Jump to > or = check
```

```
LESS  MOVF CHANGE,W
      ADDWF CCPR2L,F    ; Add duty cycle with error correction
      GOTO  SHOWSPEED
```

```
GREATER BTFSC STATUS,Z    ; Reference = Detect?
GOTO SHOWSPEED ; Yes, no need correction.
```

```

MOVF  CHANGE,W ; Detect speed > ref speed, decrease
                duty cycle with error correction
SUBLW 0XFF; Convert the negative value to positive value
SUBWF CCPR2L,F

```

```
SHOWSPEED MOVF X1,W
CALL SERIAL_TRANSMIT
CALL OVERRUN_ERROR ;Correction of Overrun Error
CALL FERR_ERROR ;Correction of Framming Error
```

```
CLRF X1 ;Reset Counter
MOVLW D'30' ;Reset Timer
MOVWF X2
```

```
CALL SERIALSCAN ; Get data for reference speed
MOVF REFERENCE,W
SUBLW 0X00
BTFSS STATUS,Z ; Reference speed = 0?
GOTO CONTINUE ; No, continue the process
```

```
STOPMOTOR MOVLW 0X00 ; Yes. Stop motor
MOVWF CCPR2L
CALL DELAY2
```

```
CONTINUE RETFIE
```

```
;***** END OF INTERRUPT PROCESS*****
```

```
END
```

```
;-----END OF PROGRAM-----
```

APPENDIX B

Source Code of Visual Basic 6.0 for DC Motor Speed Control Using Microcontroller PIC 16F877A

```
Option Explicit
    Dim pulse As Integer           'Variable for desired_pulse
    Dim str As Variant             'Count pulse received from microcontroller
    Dim speed As Double            'Real speed
    Dim desired_speed As Double
    Dim Error_speed As Double
    Dim objExcel As Excel.Application ' Setup for OLE graph
    Dim wExcel As Excel.Workbook
    Dim xlchart As Excel.Chart
    Dim j As Integer

Private Sub cmdExit_Click()

    objExcel.Visible = True      'Exit the form and show the graph in Microsoft Excel
End

End Sub

Private Sub cmdNew_Click()
```

```

'Clear the graph
Workbooks("Speed2").Sheets("Sheet1").Range("A1:E1000") = ""
j = 2
OLE_Speed.Update
Counter1.Value = 0
Counter2.Value = 0
Counter3.Value = 0

End Sub

Private Sub cmdRun_Stop_Click()

If cmdRun_Stop.Caption = "&Run" Then

    desired_speed = Val(cboSpeed.Text) ' Get desired speed from user
    pulse = Round(desired_speed / 60 * 4 * 0.39321) 'Convert the desired speed
    into desired pulse

If Val(cboSpeed.Text) > 0 Then ' If receive desired speed from user
    MSComm1.Output = Chr$(pulse) ' Send desired count pulse to
microcontroller
    lblPulse.Caption = pulse
    Counter1.TimerEnabled = True ' Start the counter as timer
    cboSpeed.Locked = True ' Lock the desired speed selection
    cmdRun_Stop.Caption = "&Stop"
    cmdExit.Enabled = False ' Disable the command of Exit and New Graph
    cmdNew.Enabled = False

Else
    MsgBox "You must select a desired speed first!", vbOKOnly, "Invalid Data"
    ' If no desired speed receive from user
    MSComm1.Output = Chr$(0)
    cmdExit.Enabled = True
    cmdNew.Enabled = True
End If

ElseIf cmdRun_Stop.Caption = "&Stop" Then
    Counter1.TimerEnabled = False ' Stop the timer
    cmdRun_Stop.Caption = "&Run"
    pulse = 0
    MSComm1.Output = Chr$(0) ' Send speed = 0 rpm to stop the motor
    cboSpeed.Locked = False
    cmdExit.Enabled = True
    cmdNew.Enabled = True
End If

End Sub

```

Private Sub Form_Load()

'MSComm setup

With MSComm1

```
.CommPort = 2          ' Use CommPort2 to communicate with
    microcontroller
.Settings = "9600,N,8,1" ' Baud rate 9600, none parity, 8 data bits, 1 stop bit
.InBufferSize = 1024    ' Receiver buffer = 1024 bytes
.OutBufferSize = 1024   ' Transmitter buffer = 1024 bytes
.DTREnable = True       ' Enable the Data Terminal Ready signal
.EOFEnable = False      ' Disable the End of File type
.Handshaking = comNone  ' Disable all network handshaking
.InputLen = 1           ' Read all the character in buffer
.InputMode = comInputModeText ' Set the incoming messages to be ASCII text
characters
.NullDiscard = False    ' Discard bytes that are all zero's
.RThreshold = 1         ' Set the Receive Oncomm event to occur after 1
byte of data have been received
.RTSEnable = True       ' Enable the request to send data
.SThreshold = 1         ' Set the Transmitter Oncomm event to occur after 1
byte of data have been received
.PortOpen = True        ' Open CommPort
```

End With

'Setup for excel file

```
Set objExcel = GetObject("", "Excel.Application")
Set wExcel = objExcel.Workbooks.Open("C:\Speed2.xls") ' File Speed2 as storage
objExcel.Visible = False
```

j = 2

With Workbooks("Speed2").Sheets("Sheet1")

.Cells(1, 1) = 0

.Cells(2, 1) = 0

End With

OLE_Speed.Update

End Sub

Private Sub MSComm1_OnComm()

If MSComm1.CommEvent = comEvReceive Then

' This is used when data is received

str = Asc(MSComm1.Input) 'Get the counter pulse from microcontroller

lblCount_pulse.Caption = str

speed = Round(str / 4 / 0.39321 * 60, 2) ' Convert counter pulse into speed(rpm)

Error_speed = Round((speed - desired_speed), 2) ' Calculate the error between

```

detected speed with the
desired speed
lblSpeed.Caption = speed
lblError.Caption = Error_speed
MSComm1.Output = Chr$(pulse) 'Always send desired pulse to microcontroller
End If

```

```

If Counter1.Value > 0 Then
    If (Counter3.Value * 60 * 60) + (Counter2.Value * 60) + Counter1.Value >
Workbooks("Speed2").Sheets("Sheet1").Cells(j - 1, 1).Value Then

```

```

'Plotting graph
With Workbooks("Speed2").Sheets("Sheet1")
    .Cells(j, 1) = (Counter3.Value * 60 * 60) + (Counter2.Value * 60) +
Counter1.Value
    .Cells(j, 2) = Format(speed, "##.##")
    .Cells(j, 3) = desired_speed

```

```

End With

```

```

j = j + 1
OLE_Speed.Update ' Update the graph
End If

```

```

End If

```

```

'Timer : Counter1 as second, Counter2 as minute, and Counter3 as hour

```

```

If Counter1.Value = 60 Then

```

```

    Counter2.Value = Counter2.Value + 1
    Counter1.Value = 0
    Counter1.TimerEnabled = True
    If Counter2.Value = 60 Then
        Counter3.Value = Counter3.Value + 1
        Counter2.Value = 0

```

```

    End If

```

```

End Sub

```

```

;-----END OF PROGRAM-----

```