"I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)"

Signature	: ______________________________
Name	: AHMAD NOR KASRUDDIN BIN NASIR
Date	: 12 NOVEMBER 2008

PATH FOLLOWER MOBILE ROBOT USING PID CONTROLLER

MUHAMMAD BIN MAZLAN

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor of Electrical and Electronic Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER, 2008

"All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author."

Signature	: ______________________________
Author	: MUHAMMAD BIN MAZLAN
Date	: NOVEMBER 2008

To my beloved mother and father

ACKNOWLEDGEMENT

First and foremost, I am very grateful to the almighty ALLAH S.W.T for giving me the key and opportunity to accomplish my Final Year Project.

This work would not have been possible without the support and encouragement of my supervisor, En Ahmad Nor Kasruddin, under whose supervision I chose this topic and began the thesis, my advisor in the final stages of the work, has also been abundantly helpful, and has assisted me in numerous ways, including guiding to completing my final year project.

I cannot end without thanking my family, on whose constant encouragement and love I have relied throughout my time at the university. I am grateful also to the examples of my father and my mother. Their unflinching courage and conviction will always inspire me, and I hope to continue, in my own small way, the noble mission to which they gave their lives. It is to them that I dedicate this work.

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **PATH FOLLOWER MOBILE ROBOT USING PID CONTROLLER**

SESI PENGAJIAN: **2007/2008**

Saya **MUHAMMAD BIN MAZLAN(860327-23-6243)**
(HURUF BESAR)

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

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
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:

G1-13 BEGONIA KRISTAL
JLN WARNASARI 13/1
BANDAR PUNCAK ALAM
SELANGOR

AHMAD NOR KASRUDDIN NASIR
(Nama Penyelia)

Tarikh: **12 NOVEMBER 2008** Tarikh: : **12 NOVEMBER 2008**

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 penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

ABSTRACT

This project is about path follower mobile robot using PID Controller. As we know the PID controller is a generic control loop feedback mechanism widely used in industrial control system. The controller corrects the error that makes the mobile robot moving out of track. This project concentrates in the development path follower mobile robot which is moving in square path with straight line and turn 90 degree and integrating the PID Controller into steering path for the path follower mobile robot to make the mobile robot moving smooth straight line and turning 90degree. At the end of this project also discuss about the comparison between controllers that could integrate into the mobile robot system.

ABSTRAK

Projek ini menerangkan berkenaan robot mudah alih pengikut jalan menggunakan pengawal PID. Seperti sedia maklum, pengawal PID adalah mekanisme pengawal gelung suapbalik generik yang banyak digunakan dalam industri. Pengawal ini bertindak membetulkan kesalahan yang boleh menyebabkan robot mudah alih ini terkeluar dari landasan. Projek ini memfokuskan pada pembangunan robot mudah alih pengikut jalan di mana ia akan bergerak dalam jalan segi empat sama dengan jalan lurus dan 90 darjah belok, dan menyatukan pengawal PID ke dalam steering robot mudah alih pengikut jalan untuk menjadikan robot mudah alih dapat bergerak lancar lurus dan membelok 90 darjah. Di pengakhiran projek ini juga membincangkan tentang perbandingan antara pengawal-pengawal yang boleh di satukan ke dalam system robot mudah alih.

TABLE OF CONTENTS

LIST OF FIGURE

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

Nowadays, there are many mobile robots in different shape, size and application. Mobile robot with programmed path for movement is one of them. Many programmed path robot just hardware and program. A mobile robot is an automatic machine that is capable of movement in a given environment. Mobile robots have the capability to move around in their environment and are not fixed to one physical location. In contrast, industrial robots usually consist of a jointed arm (multi-linked manipulator) and gripper assembly (or end effectors) that are attached to a fixed surface. Mobile robots are the focus of a great deal of current research and almost every major university has one or more labs that focus on mobile robot research. Mobile robots are also found in industry, military and security environments. They also appear as consumer products, for entertainment or to perform certain tasks like vacuum cleaning or mowing.

The one way mobile robot which is when we turn on the mobile robot, the mobile robot will move and do the task we ask. In control term is known as open loop system. Therefore, I invented this mobile robot with programmed path with PID controller to upgrade the open loop system in robot into closed loop system mobile robot. The implementation of PID is to improve the controller circuit based on the data from the experiment.

There are many controllers in the industry such as PID, Fuzzy Logic, Artificial Neural Network and Linear Quadratic Regulator.

1.2 OBJECTIVE

At the end of this project:

i. Able to develop path follower mobile robot.
ii. Able to design PID in microcontroller to steering path follower mobile robot.
iii. Comparing the result the path follower mobile robot using PID and without PID.

1.3 SCOPE OF PROJECT

Scopes the need to be proposed are;

i. Making path follower mobile robot.
    - Making normal mobile robot with square path.

ii. Implement PID in path follower mobile robot.
    - After accomplish the mobile robot, run a experiment and correct its error by using PID controller

iii. Making comparison between PID system and others.
    - Finalized and making comparison between path follower mobile robot without PID and with PID.

1.4 PROBLEM STATEMENT

The main problem in making path follower mobile robot is easily move out of track when turning at the corner. Some robot cannot move straight even we static the speed of the robot. Most of the problem we can repair through program and hardware but the system is not complete and have condition to make the system is stabilized. Therefore, we need to design a system to stabilize the basic system without and condition to make is stable and here came PID controller.

1.5 THESIS ORGANIZATION

There are 5 chapters in this thesis including this chapter. The content of each chapter which are:

i. In chapter 2, contain detailed description each part of my project. It will be discussion about, overall project overview, Proportional-Integral-Derivative and path follower mobile robot.
ii. Next in chapter 3, discuss about project methodology. This methodology I used in the project.
iii. Then in chapter 4 is result and discussion. It will tell result about this project and costing & commercialization
iv. Lastly in chapter 5 is conclusion and future recommendation.

CHAPTER 2

LITERATURE REVIEW

2.1 PROJECT REVIEW

This project will be design the path follower mobile robot with PID as the controller. A system for path tracking of mobile robot is a program in the PIC is forward 20cm and turn 90degree for 5cm radius and the sensor at the front edge mobile robot. [3] This project is also making turn at maximum speed without move out from the track. Traditional method is making turn using remote control or program with fixed speed. The multi loop nature of the architecture ensures adequate stability at different levels yielding safe navigation and accomplishment of higher level tasks.[1] Navigation is a major issue when addressing mobile robotics because the concept is so wide that it includes all aspects of directing a robot's course as it traverses the environment.[1] The sensor for the development of the mobile robot path following algorithm is important.[2]

I have formulated the problem is to (1) create the system or controller that can follow desired or set point with variable speed, (2) keep the robot in the track even after one complete square, (3) the repeated program that I will be used to keep mobile robot on track which is move forward and turn 90degree and response to the error. The approach that I have taken is to develop normal mobile robot with square path movement and I add infra red sensor at the front edge of the robot. This sensor will detect the line of the track and send error signal to the controller. To maintain the mobile robot on the course is using rubber tire and light weight body material. The program will have the main program and the interrupt program for correct the movement of the mobile robot which is respond to the error.

2.2 Proportional Integral and Derivative Controller (PID)

A proportional-integral-derivative controller (PID controller) is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired set point by calculating and then outputting a corrective action that can adjust the process accordingly.

The PID controller calculation (algorithm) involves three separate parameters; the Proportional, the Integral and Derivative values. The Proportional value determines the reaction to the current error, the Integral determines the reaction based on the sum of recent errors and the Derivative determines the reaction to the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element.

By "tuning" the three constants in the PID controller algorithm the PID can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the set point and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system.

Some applications may require using only one or two modes to provide the appropriate system control. This is achieved by setting the gain of undesired control outputs to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are particularly common, since derivative action is very sensitive to measurement noise, and the absence of an integral value prevents the system from reaching its target value due to the control action.

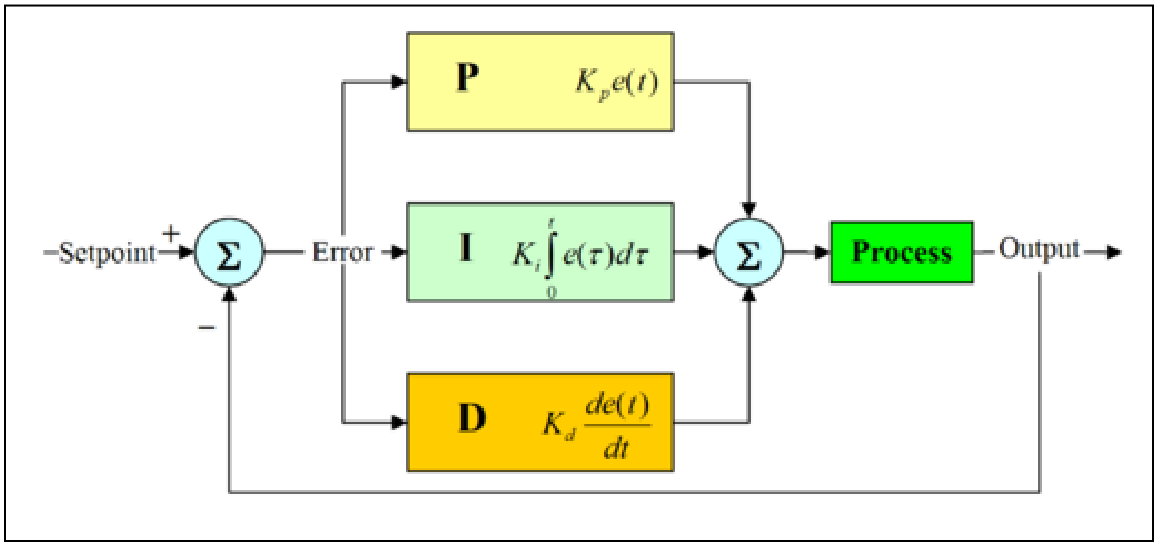


Figure 2.1: A block diagram of PID controller

2.3 Mobile Robot

A Mobile Robot is an automatic machine that is capable of movement in a given environment. Mobile robots have the capability to move around in their environment and are not fixed to one physical location. In contrast, industrial robots usually consist of a jointed arm (multi-linked manipulator) and gripper assembly (or end effectors) that is attached to a fixed surface. Mobile robots are the focus of a great deal of current research and almost every major university has one or more labs that focus on mobile robot research. Mobile robots are also found in industry, military and security environments. They also appear as consumer products, for entertainment or to perform certain tasks like vacuum.

2.4 PIC Microcontroller

The name of PIC initially referred to “Programmable Interface Controller” [4], but shortly thereafter was renamed “Programmable Intelligent Computer” [5].

PICs are popular with developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability. Microchip recently announced the shipment of its six billionth PIC processor.

Microcontroller that I have use is PIC 16F877 which have 3Timer; Timer0 and Timer 2 for 8 bit counter or timer, Timer1 for 16 bit counter or timer, 2 Capture, Compare, PWM modules, 10bit multi-channel Analog-to-Digital converter, Synchronous Serial Port (SSP), Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9 bit address detection, Parallel Slave Port (PSP) 8 bit, and 368bytes memory with 256 bytes EEPROM.

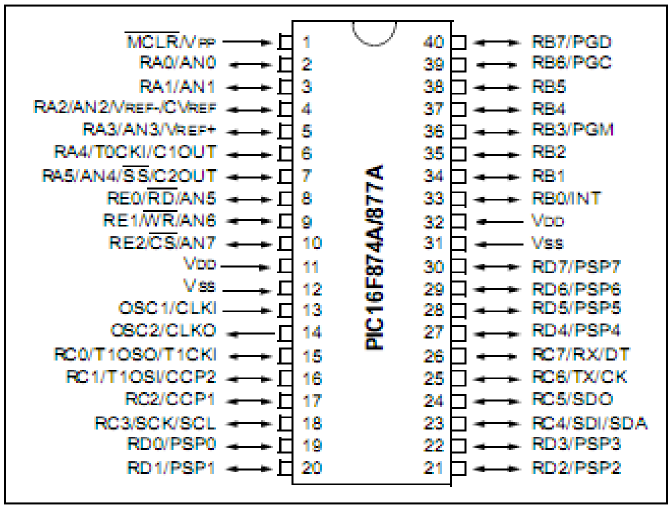


Fig 2.2: PIC 16F877A pin configuration

2.5 DC motor

A DC motor works by converting electric power into mechanical work. This is accomplished by forcing current through a coil and producing a magnetic field that spins the motor. The simplest DC motor is a single coil apparatus, used here to discuss the DC motor theory.

The voltage source forces voltage through the coil via sliding contacts or brushes that are connected to the DC source. These brushes are found on the end of the coil wires and make a temporary electrical connection with the voltage source. In this motor, the brushes will make a connection every 180 degrees and current will then flow through the coil wires. At 0 degrees, the brushes are in contact with the voltage source and current is flowing. The current that flows through wire segment C-D interacts with the magnetic field that is present and the result is an upward force on the segment. The current that flows through segment A-B has the same interaction, but the force is in the downward direction. Both forces are of equal magnitude, but in opposing directions since the direction of current flow in the segments is reversed with respect to the magnetic field. At 180 degrees, the same phenomenon occurs, but segment A-B is forced up and C-D is forced down. At 90 and 270-degrees, the brushes are not in contact with the voltage source and no force is produced. In these two positions, the rotational kinetic energy of the motor keeps it spinning until the brushes regain contact.

One drawback to the motor is the large amount of torque ripple that it has. The reason for this excessive ripple is because of the fact that the coil has a force pushing on it only at the 90 and 270 degree positions. The rest of the time the coil spins on its own and the torque drops to zero. The torque curve produced by this single coil, as more coils are added to the motor, the torque curve is smoothed out.

The resulting torque curve never reaches the zero point and the average torque for the motor is greatly increased. As more and more coils are added, the torque curve approaches a straight line and has very little torque ripple and the motor runs much more smoothly. Another method of increasing the torque and rotational speed of the motor is to increase the current supplied to the coils. This is accomplished by increasing the voltage that is sent to the motor, thus increasing the current at the same time.

Fig 2.3: DC motor 12V

2.6 Navigation System

Navigation for mobile robot is the main problem. Therefore, we need to calculate accurate and precisely to avoid the error regarding the path that we develop. We also use a sensor to detect the rotational of the tire. The path or course that we have made is square path. It have diameter 40cm^2. Therefore, the mobile robot will move straight for 30cm and turn 90degree for circle with radius 2.5cm.

The overall system, which we use the controller is PID and we continue the process according to the set point without error. Figure 2.4 is shown the hardware of the project and the flow of the hardware works.

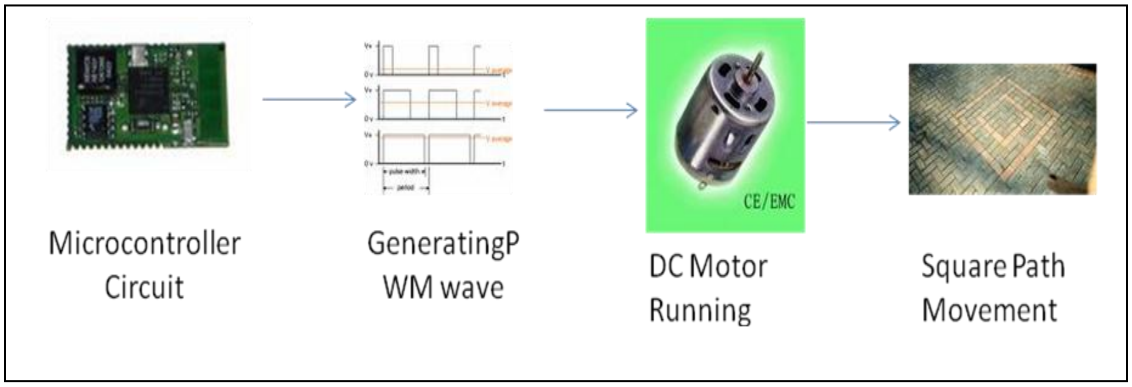


Fig 2.4: The Output System Connection.

2.7 Computer Programming

Programming or coding it is word familiar with the computer programming. Computer programming is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs. The computer program is written in programming language which has many types in the world right now. Examples of programming language are Machine Code, Assembly, BASIC, C, JAVA, FOTRON, PHYTON and PERL. There are many more programming languages which are in the internet. [7] The programmer need to choose based on their expertise and experience.

Machine Code is based on binary number which is either '1' or '0'. This programming language is category in low level language which is the most possible programming language that can be program. This programming faster execution and can control every component in a system but complex programming and need to understand system configuration. The programmer needs to memorize every group of number to do an instruction. Differ Machine Code for each processor and normally is upward compatible. Machine Code only language the computer can understand.

Assembly Language is an instruction based on the processor. This programming required an assembler to assemble into machine code. This programming we can understand but the computer cannot. Assembly language is much easier to write the program than the Machine Code.

High Level languages are using instruction by the compiler. It's required to follow the syntax and semantics. The problem with incompatible processor is solved by using this programming language. The high level languages have many types such as BASIC, C, FOTRON, PERL and JAVA. This programming language closed to our normal language. Therefore we can easily understand the program.

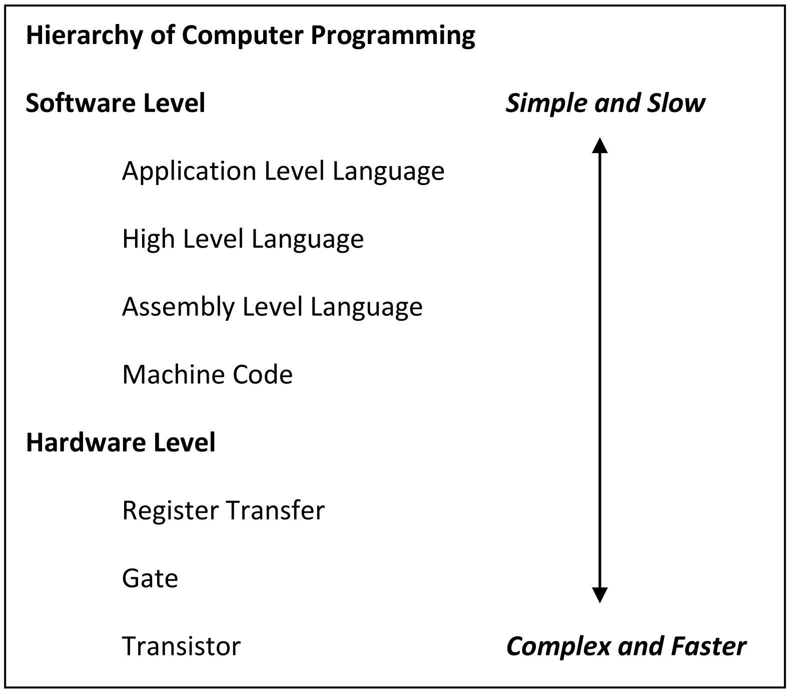


Figure 2.5: Hierarchy of Computer Programming

The computer programming is the process to make the program for computer to understand what the task and instruction that the computer need to do. The only computers understand is only Machine Code. Therefore the programming language other the Machine Code need an assembler (for Assembly Language) and the compiler (for the High Level Language) to convert the source code or program into Machine Code.

CHAPTER 3

METHODOLOGY

3.1 Hardware Configuration.

Hardware Installation

For hardware design, first is to design the power supply module which is to supply 5V fixed to PIC. Power supply module is importance to PIC to prevent damage if users give the higher input supply to device. The schematic diagram for power supply module is like in figure 3.1. Input to the power supply must greater than 7V to 7805 voltage regulator IC to achieve the 5V output supply to PIC and max232.

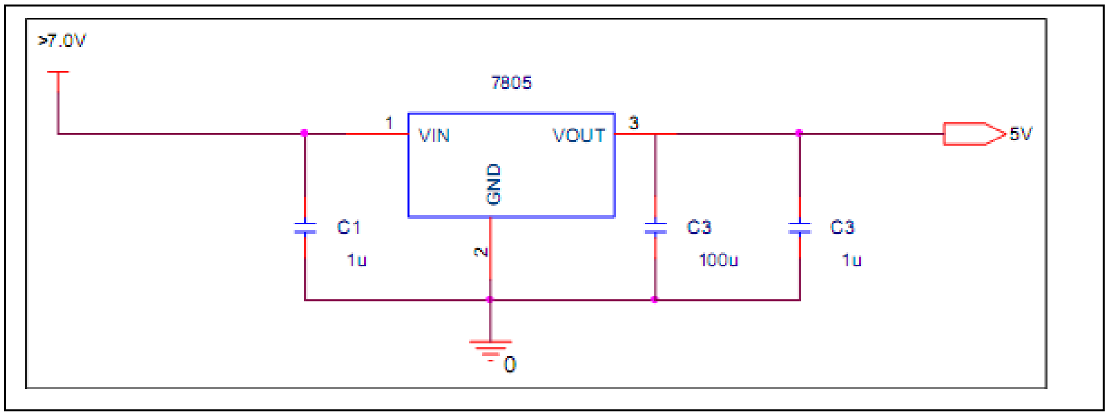


Figure 3.1: Power Supply Modules

The power module wasn't the only module to complete the basic circuit. There are another 2 circuit's module to complete the basic PIC circuit diagram, the clock circuit and the reset button circuit.

The clock circuit is to generate the clock pulse to PIC Microcontroller to operate. The range of the crystal is 4MHz until 20MHz. For low crystal clock value which is 4MHz we need 2 capacitors to smooth the pulse and stabilized the pulse. The range value of the capacitor is 10nF-33nF.

Figure 3.3: Example of crystal to generate clock.

The reset circuit is important even though it for system to revert to the initial state but it also for the PIC Microcontroller to start operate. The PIC Microcontroller will operate if only the Reset (MCLR pin 1) received or get the input high (4.5V - 5.0V) and the PIC will reset if only get the input low (below 4.5V). For high clock frequency, we need to put the capacitor parallel with the reset button. The function of this capacitor is to hold instance the pulse from the reset button.

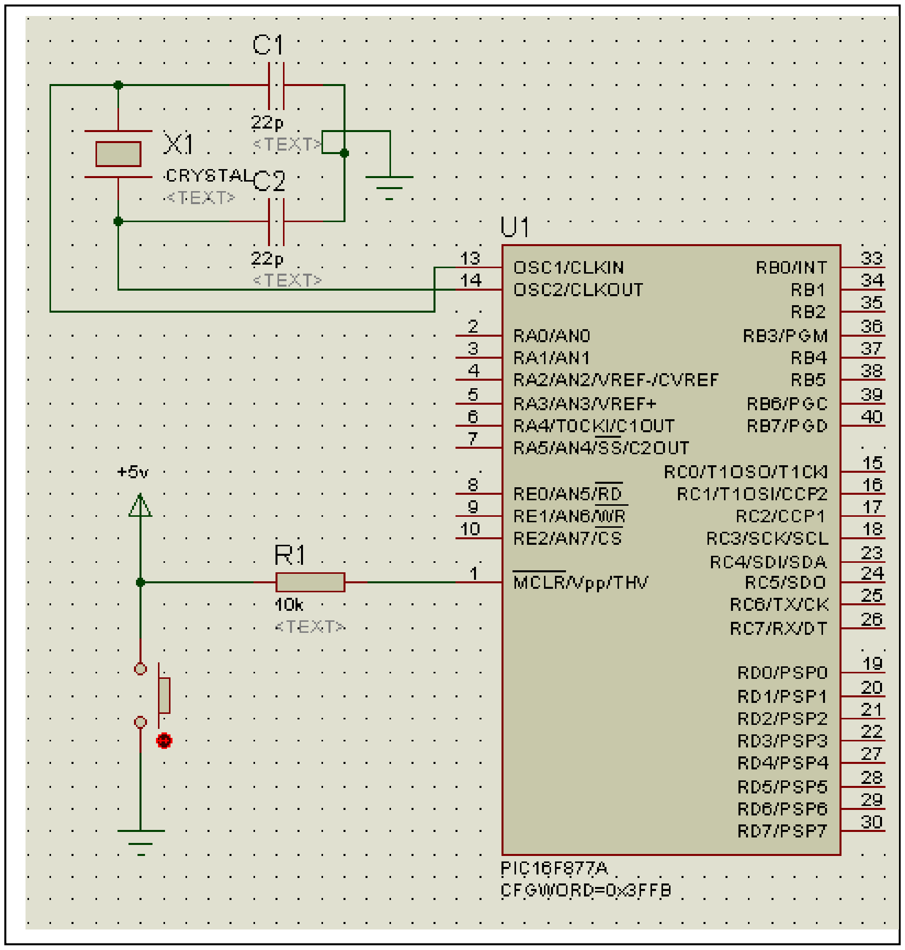


Figure 3.3: Basic Circuit PIC

The output on the PIC port is approximately 4.7 V low current which is cannot run the stepper motor or DC motor directly. So, to run the motor, switching approach is use by using additional source with high current supply. To overcome this problem, I use driver L293B to drive the both DC motor. With this driver, I can directly supply high power to both DC motor by supplying at pin supply voltage pin (Vs). By using this driver, user should aware which pin connects to high power and low power, failed to follow will result the driver broken down.

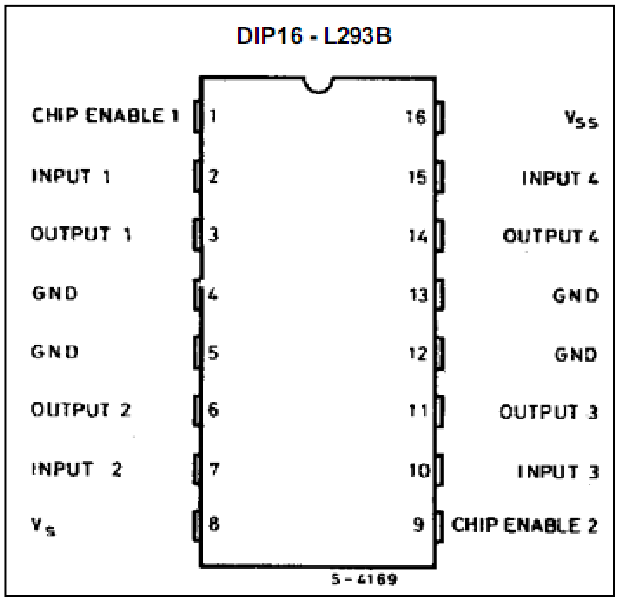


Figure 3.4: Pin configuration L293B

For sensor to detect the tape as the border of the square path, I use infra red sensor. The reason I used the infrared sensor is the simple circuit and easy to design. The main component of the circuit is LM324, IR receiver and IR transmitter. The 20k variable resistor is for configure the range of the IR sensor to detect. The supply voltage for the circuit is 5V. Therefore it uses the supply from the output Power Circuit above in Figure 3.1.

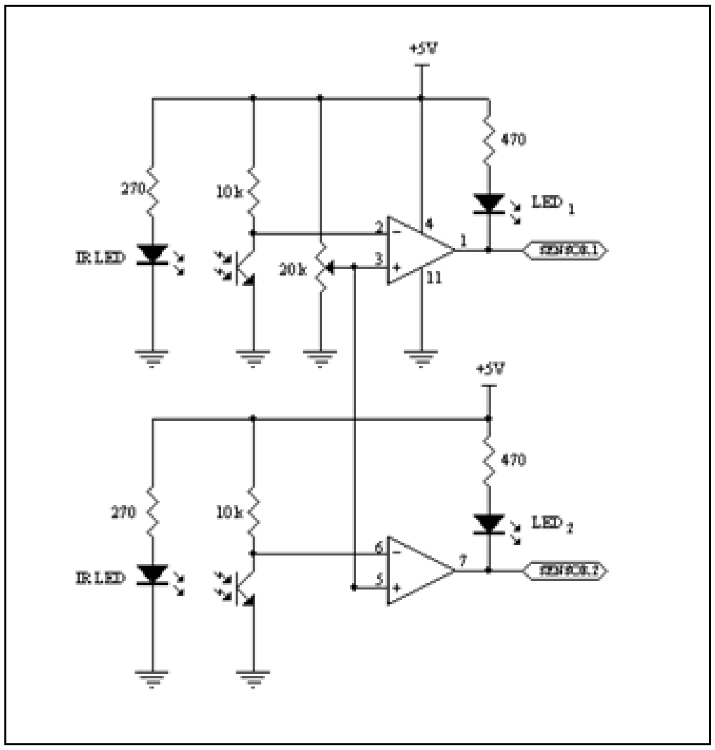


Figure 3.5: The Infrared Red Sensor Using LM324

The LM324 IC was a comparator which is have 2terminal and 1output. In electronics, a comparator is a device which compares two voltages or currents and switches its output to indicate which is larger. A dedicated voltage comparator will generally be faster than a general-purpose op-amp pressed into service as a comparator. A dedicated voltage comparator may also contain additional features such as an accurate, internal voltage reference, an adjustable hysteresis and a clock gated input.

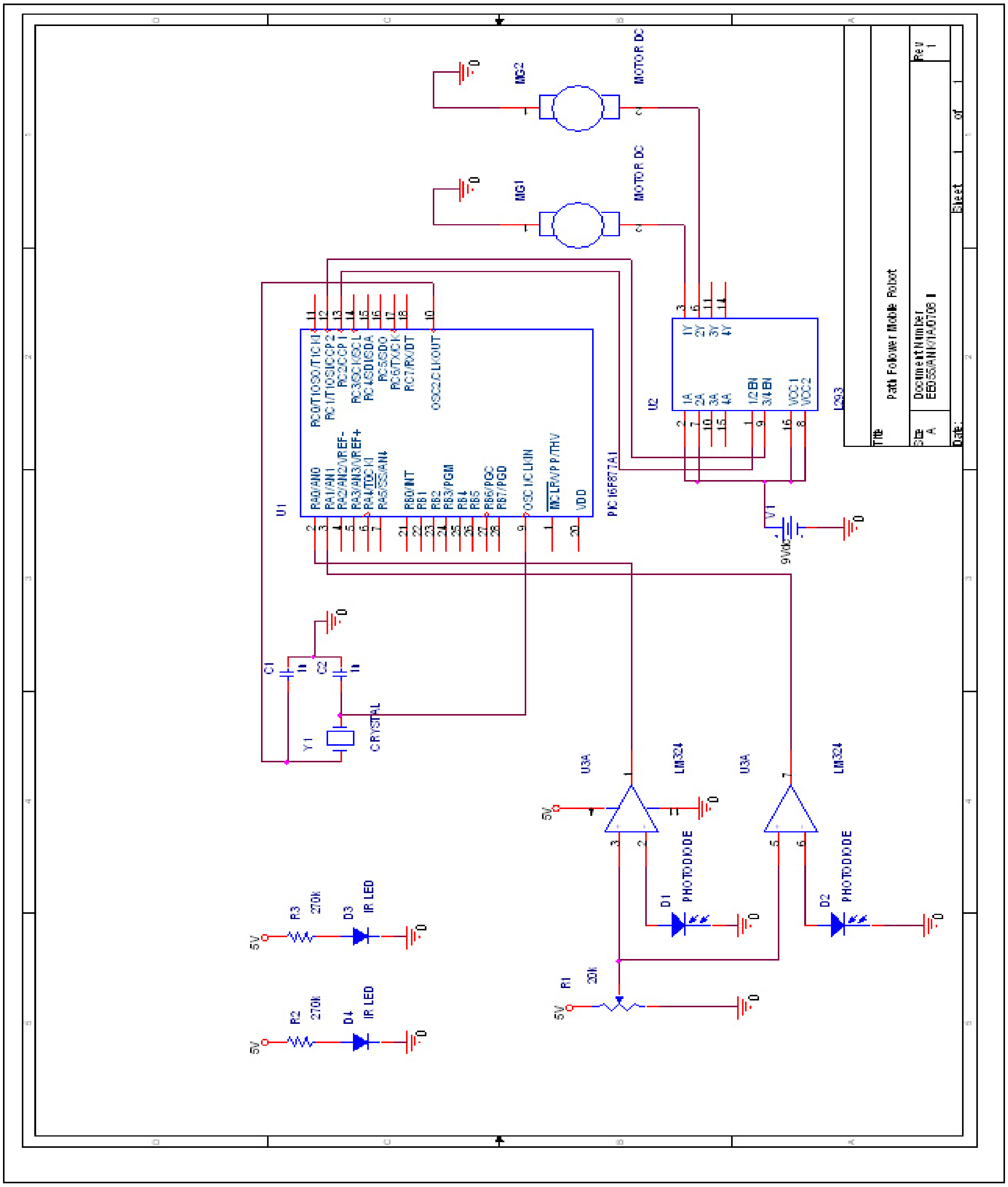


Figure 3.6: Whole Module Circuit

3.2 Software Configuration

After the successful hardware configuration and each module working, the important second part is programming. The programming can be written in any computer language such as C family, BASIC, Assembly or Machine Code language. Computer programming languages are divided into several categories which is High Level Language, Low Level Language, Assembly Level Language and Machine Language. The programming language is depending on the programmer (person) and the compiler. The comparison between High Level Language programming and Machine Level Language are:

i. The very lowest possible level at which you can program a computer is in its own native machine code, consisting of strings of 1's and 0's and stored as binary numbers. The main problems with using machine code directly are that it is very easy to make a mistake, and very hard to find it once you realize the mistake has been made.

ii. Assembly language is nothing more than a symbolic representation of machine code, which also allows symbolic designation of memory locations. Thus, an instruction to add the contents of a memory location to an internal CPU register called the accumulator might be add a number instead of a string of binary digits (bits). No matter how close assembly language is to machine code, the computer still cannot understand it. The assembly-language program must be translated into machine code by a separate program called an assembler. The assembler program recognizes the character strings that make up the symbolic names of the various machine operations, and substitutes the required machine code for each instruction. At the same time, it also calculates the required address in memory for each symbolic name of a memory location, and substitutes those addresses for the names. The final result is a machine-language program that can run on its own at any time; the assembler and the assembly-language program are no longer needed. To help distinguish between the "before" and "after" versions of the program, the original assembly-language program is also known as the source code, while the final machine-language program is designated the object code. If an assembly-language program needs to be changed or corrected, it is necessary to make the changes to the source code and then re-assemble it to create a new object program.

iii. Compiler languages are the high-level equivalent of assembly language. Each instruction in the compiler language can correspond to many machine instructions. Once the program has been written, it is translated to the equivalent machine code by a program called a compiler. Once the program has been compiled, the resulting machine code is saved separately, and can be run on its own at any time. As with assembly-language programs, updating or correcting a compiled program requires that the original (source) program be modified appropriately and then recompiled to form a new machine-language (object) program. Typically, the compiled machine code is less efficient than the code produced when using assembly language. This means that it runs a bit more slowly and uses a bit more memory than the equivalent assembled program. To offset this drawback, however, we also have the fact that it takes much less time to develop a compiler-language program, so it can be ready to go sooner than the assembly-language program.

iv. Major difference is the high level language consumes high memory and slows the process but the program is much simpler. Meanwhile, low level language consumes low memory and faster the process but the programming is long and complicated because it depend on the instruction string on the chip.

The free compiler is for an Assembly Language by Microchip™ which is MPLAB®. The latest MPLAB® is MPLAB® v8.02 which more stable than the previous MPLAB® as shown in Figure 3.6. This software also can compile another computer programming language but it required a plug-in of the compiler with MPLAB® software. Here are the other plug-in available to MPLAB® software:

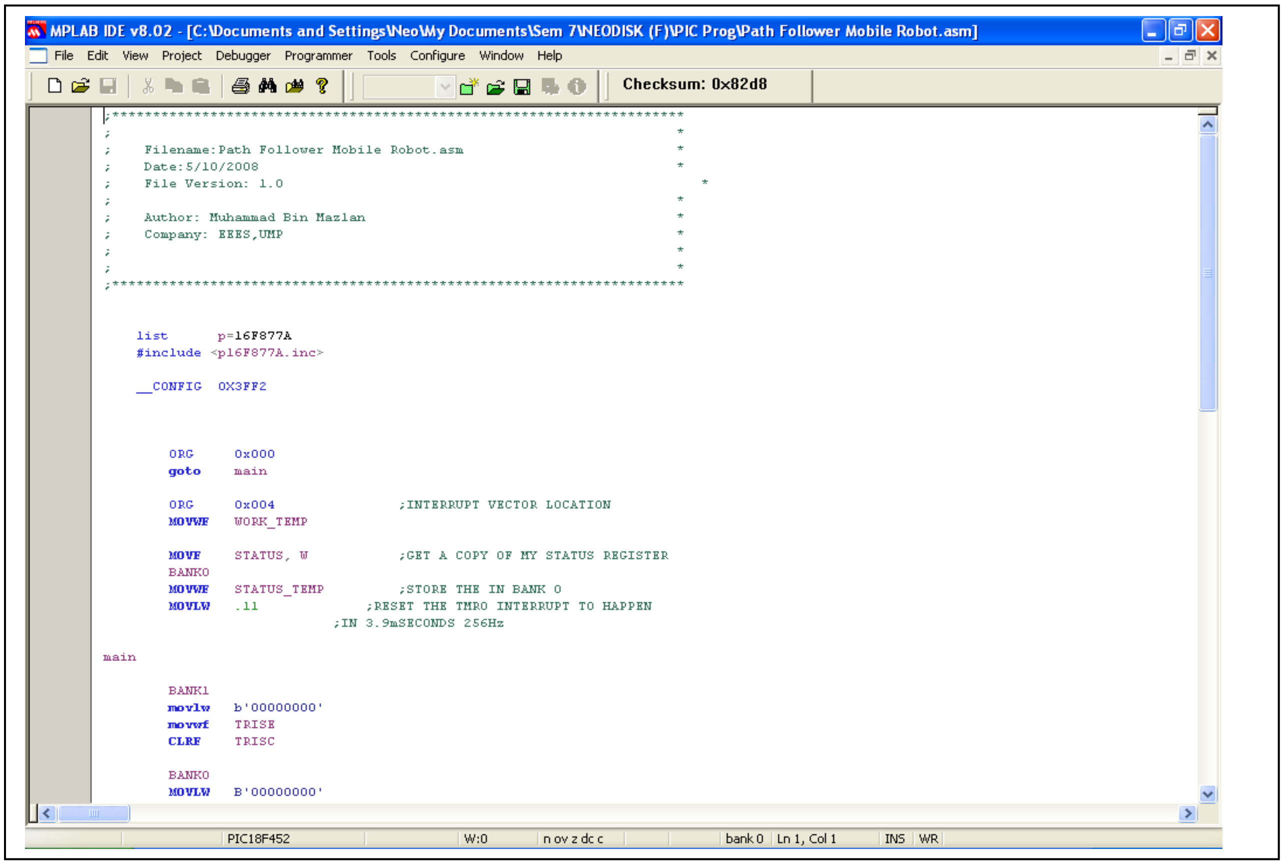
i. B Knudsen Data

ii. Byte Craft

iii. CCS C Compiler

iv. High Tech

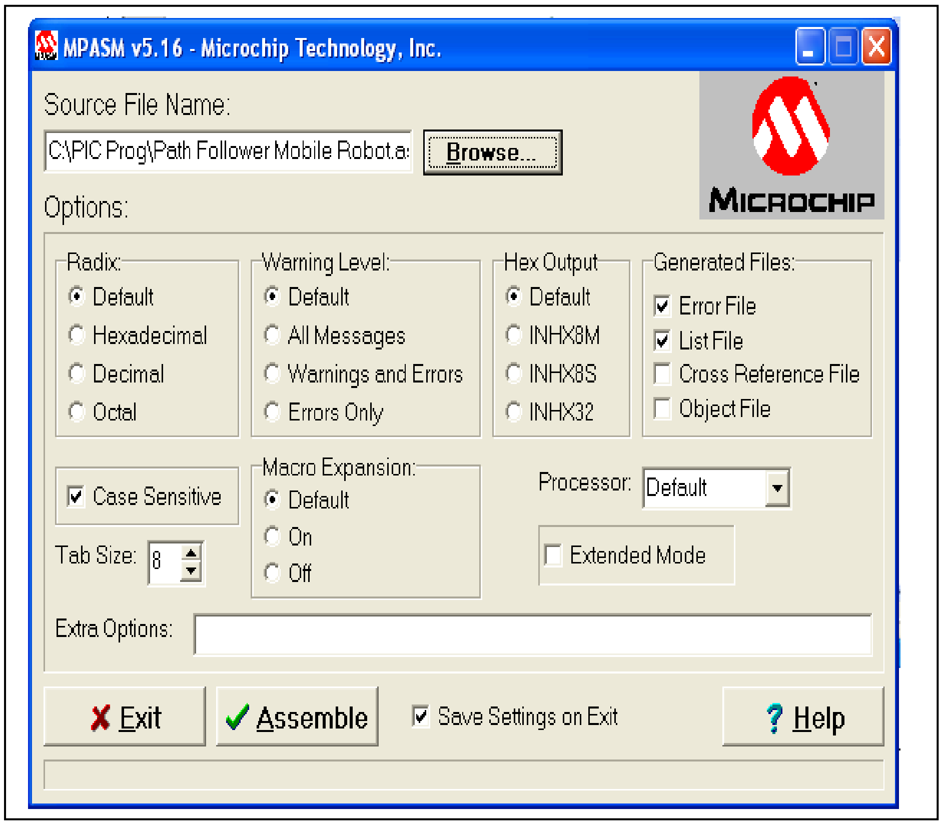
v. IAR


MPLAB IDE v8.02 - [C:\Documents and Settings\Neo\My Documents\Sem 7\NEODISK (F)\PIC Prog\Path Follower Mobile Robot.asm]
File Edit View Project Debugger Programmer Tools Configure Window Help
Checksum: 0x82d8
PIC18F452 W:0 n ov z dc c bank 0 Ln 1, Col 1 INS WR

```
;*************************************************************************
;                                                                        *
;    Filename:Path Follower Mobile Robot.asm                             *
;    Date:5/10/2008                                                      *
;    File Version: 1.0                                                      *
;                                                                        *
;    Author: Muhammad Bin Mazlan                                         *
;    Company: EEES,UMP                                                   *
;                                                                        *
;                                                                        *
;*************************************************************************


    list      p=16F877A
    #include <p16F877A.inc>

    __CONFIG  0X3FF2



        ORG     0x000
        goto    main

        ORG     0x004                   ;INTERRUPT VECTOR LOCATION
        MOVWF   WORK_TEMP

        MOVF    STATUS, W               ;GET A COPY OF MY STATUS REGISTER
        BANK0
        MOVWF   STATUS_TEMP             ;STORE THE IN BANK 0
        MOVLW   .11                 ;RESET THE TMR0 INTERRUPT TO HAPPEN
                                ;IN 3.9mSECONDS 256Hz

main

        BANK1
        movlw   b'00000000'
        movwf   TRISE
        CLRF    TRISC

        BANK0
        MOVLW   B'00000000'
```

Figure 3.7: MPLAB Software

The assembler that I used is MPASM which came with MPLAB software. This software only assembles the program written in Assembly Language (extension .asm).



MPASM v5.16 - Microchip Technology, Inc.

Source File Name: C:\PIC Prog\Path Follower Mobile Robot.a:

Options:

Radix: [✓ Default] Hexadecimal Decimal Octal

Warning Level: [✓ Default] All Messages Warnings and Errors Errors Only

Hex Output: [✓ Default] INHX8M INHX8S INHX32

Generated Files: [✓ Error File] [✓ List File] Cross Reference File Object File

[✓ Case Sensitive]

Tab Size: 8

Macro Expansion: [✓ Default] On Off

Processor: Default

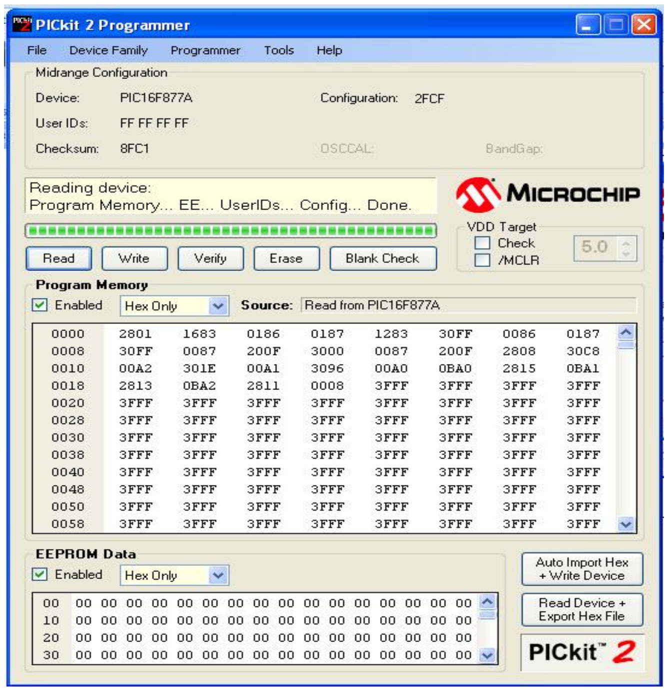
Extended Mode

Extra Options:

[✓ Save Settings on Exit]

Figure 3.8: MPASM Software

This software will generate the.hex, .lst, and .err (only if the source code has an error). The .hex file will generate when the source code didn't have any error. The .hex files contain only numbers. This file we need to insert into the PIC by using a PIC Programmer such as UIC00A with software Microchip® PICKit 2 v2.40 as shown in Figure 3.9. The .lst is for list file which contain the memory map, the instruction in .asm and the machine codes are used each instruction. The .err is an error message file which contain error message and type of error that have occurred. The example of .asm .lst and .err are at the appendix.



Midrange Configuration

Device: PIC16F877A Configuration: 2FCF
User IDs: FF FF FF FF
Checksum: 8FC1 OSCCAL: BandGap:

Reading device:
Program Memory... EE... UserIDs... Config... Done.

VDD Target
☐ Check
☐ /MCLR
5.0

Program Memory
✓ Enabled Hex Only Source: Read from PIC16F877A

```
0000   2801  1683  0186  0187  1283  30FF  0086  0187
0008   30FF  0087  200F  3000  0087  200F  2808  30C8
0010   00A2  301E  00A1  3096  00A0  0BA0  2815  0BA1
0018   2813  0BA2  2811  0008  3FFF  3FFF  3FFF  3FFF
0020   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0028   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0030   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0038   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0040   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0048   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0050   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
0058   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
```

EEPROM Data
✓ Enabled Hex Only

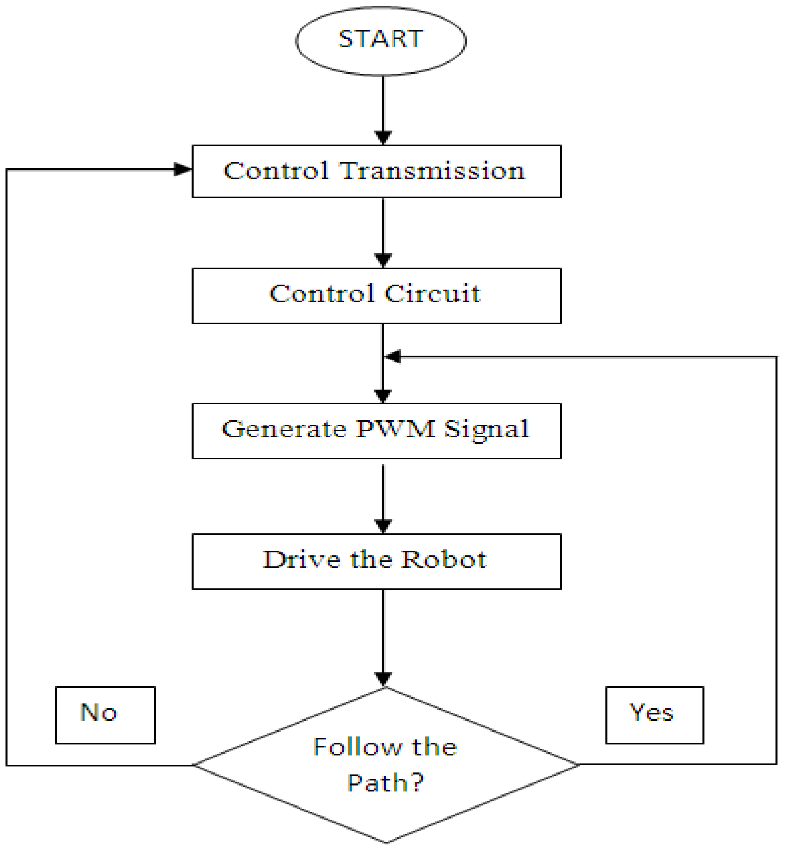
```
00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 3.9: PICKit 2 v2.4 software

All type of assembler and compiler for PIC will also produce a .hex file for burning process (insert the program into PIC Microcontroller memory and program it). This software usually came with the programmer. Since the PIC Microcontroller use Harvard Architecture, it separates the Program Memory and the Electronic Erase Programmable Read Only Memory (EEPROM) Data. Usually we program the Program Memory. For PIC 16F877A1, it only have the 8kbyte of memory. Therefore it can store up to 8kbyte which is from 0000 until FFFF in hex address. For program that over the limit of memory, it will be replace the top program that we have just program.

Another type of compiler is PICC-Lite and Micro Code. PICC-Lite compiler is for programming using C language. The Micro Code compiler is for programming using BASIC language.

The program will construct based on the Flow Chart below:



First, the control transmission is the PIC 16F877A pin CCP1 and CCP2 will be use to produce PWM signal with different duty cycle and trigger the control circuit which are driver circuit for the DC motor. The signal continues and drives the motor in the H-Bridge circuit to control the movement of the dc motor. The DC motor will run and the sensor at the edge is running. If the sensors detect the line of the track it will send an error signal to the Interrupt pin at the pin. The interrupt program will initialize to make more turn and stay on the track. If none of sensors at the front edge detect the line of the track, then the main program will continue as before.

The program is to navigate the mobile robot move square path. Robot will move 20cm forward and 90degree turn 5cm. The mobile robot will make square path. The sensor at the edge of front rear mobile robot will detect the error the mobile robot has made. There for the PID controller will determine the error and corrected it and thus stay on the track.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Result

In this paper, we have demonstrated the design and implementation of a PID controller. PID controller system provides the corrective action in path follower mobile robot. My first set of experiments demonstrated lighter the mobile robot and the right motor is better and wise decision. The second set of experiments showed that my algorithm is stable and can perform as I wanted. In addition to the performance advantages the model predictive approach also requires less tuning than pure pursuit.

4.2 Path Follower Mobile Robot

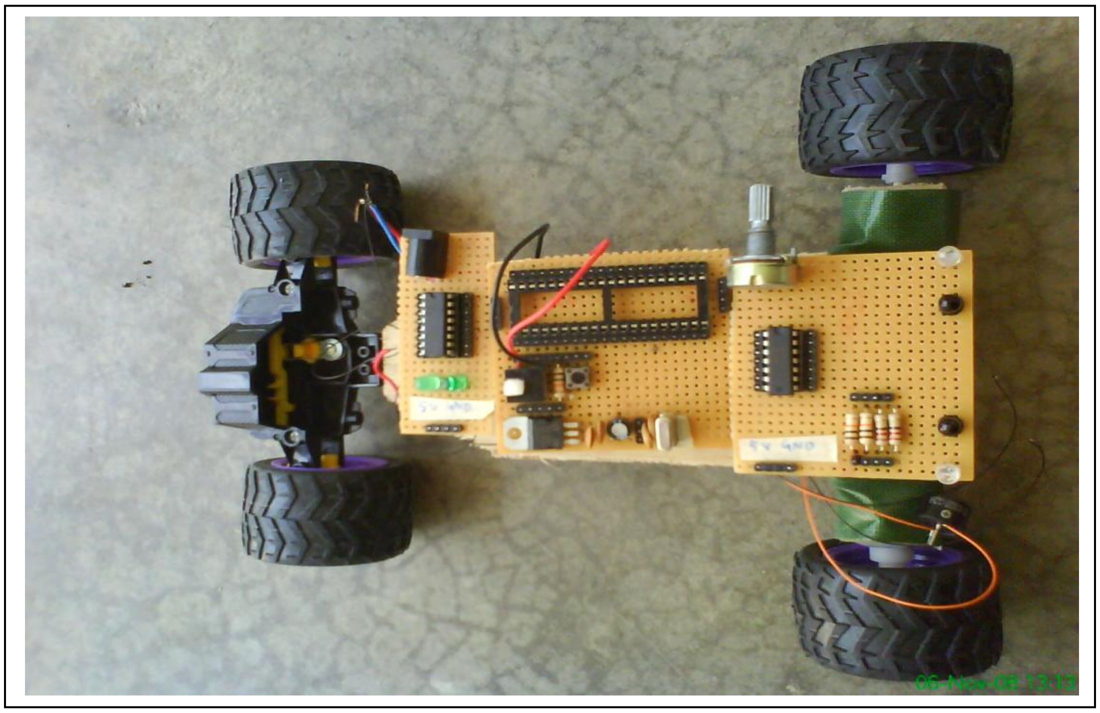


Figure 4.1: The overall circuit with prototype of the mobile robot.

The figure above is complete circuit with prototype mobile robot body. The front side of this robot is on the left of this figure. From right, the circuit is for DC motor. At the center is basic PIC Microcontroller with Power Supply circuit. At the front and left of this figure is IR sensor circuit.

4.3 Basic PIC Microcontroller Circuit.

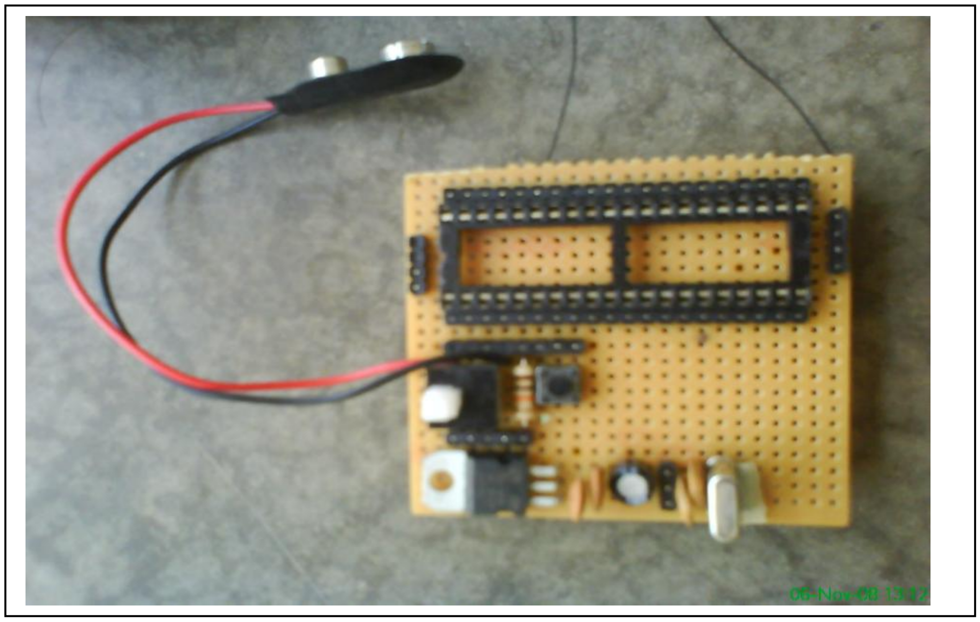


Figure 4.2: Basic PIC Microcontroller Circuit.

The Figure 4.2 above is for the basic PIC Microcontroller circuit module. In this module it contains Power Supply, Reset and Clock Circuit. The main component consist PIC 16F877A1, LM7805 IC Regulator, Crystal 8 MHz and Reset Button. Function of this module is as a brain for the path follower mobile robot.

4.4 Infra Red Sensor Circuit

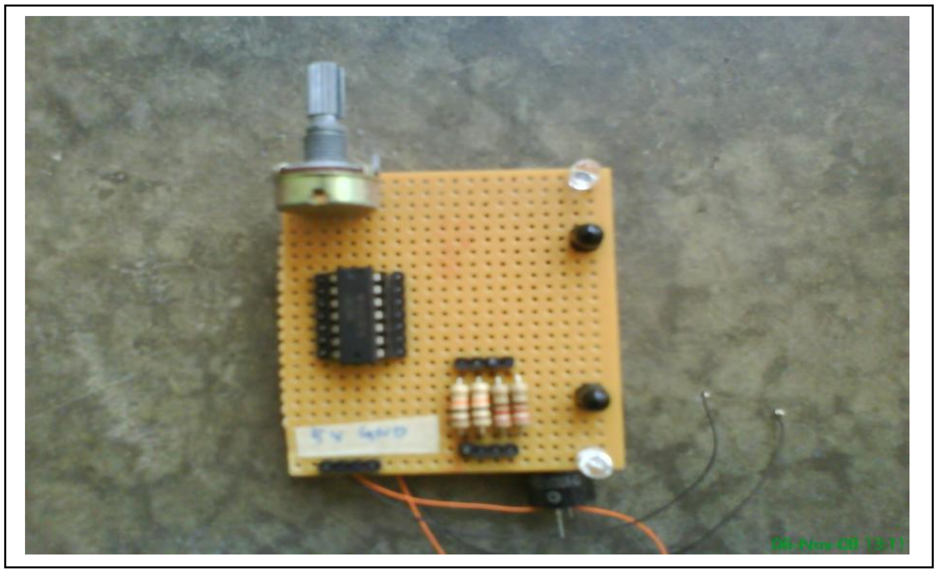


Figure 4.3: IR Sensor Circuit

The Infra Red (IR) sensor circuit is using 2 sensors to detect right and left edge. The circuit consists of LM324 comparator IC, 2 IR receiver, 2 IR transmitters and Variable Resistor. This sensor will detect the white and black colour only. If Black colour the sensor will transmit low signal and if White colour it will transmit high signal.

4.5 Direct Current Motor Circuit

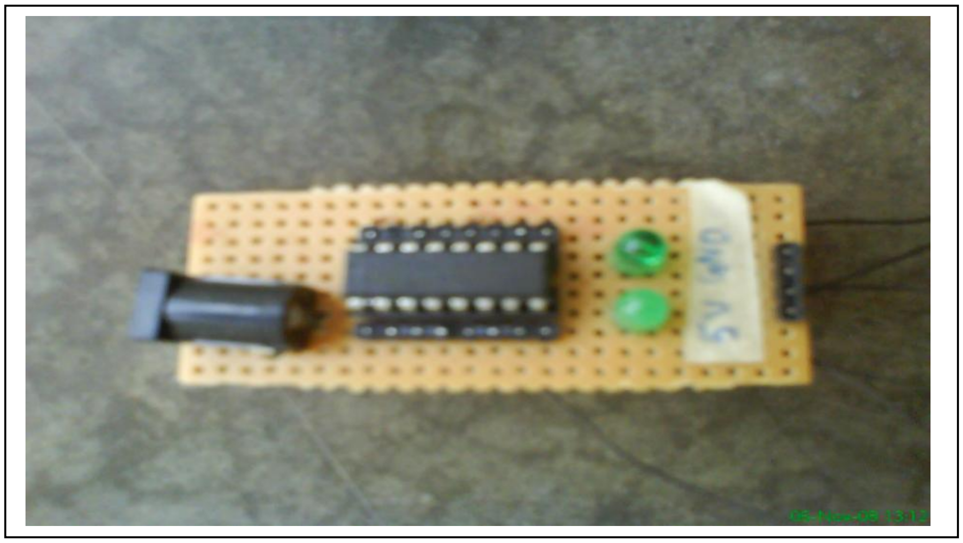


Figure 4.4: DC Motor Circuit.

The DC motor circuit is consist DC motor driver (L293B), adapter socket and LED as indicator. The supply voltages for the circuit are 12V and 5V. 12V connect through the adapter socket to Vs and Vss pin at L293B IC. The 5V supply is for enabling which motor should move. These configure by the programming in the PIC and generate Pulse Width Modulation (PWM) at pin CCP1 and CCP2.

There are many type of the motor and its driver, therefore the choosing for suitable is important for the successful of the robot. The wrong choice, the chance unsuccessful this project is high. The chosen should make based on type of work the mobile robot and the mobile robot body made of. Another criteria should consider is the weight overall mobile robot and the motor power.

4.6 Discussion

These projects consume much budget and energy due to failure of the complete circuits. I have configured the problem came from the soldering paste which I have been used to solder the all component on the dependent and independent board. The problem starts the clock circuit at the basic circuit of PIC Microcontroller. Then it shorts all other component on the board. The failure board is shown below at Figure 4.5.

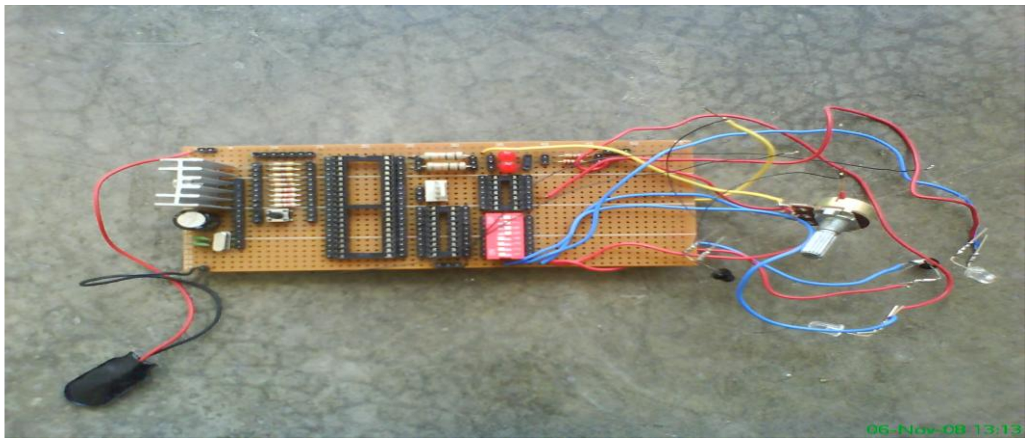


Figure 4.5: The complete one board circuit.

The solution to this problem is not use the soldering paste or use only on other device or component other than Integrated Circuit component.

4.7 Costing & Commercialization

This project cost is approximately RM200. This cost is divided into two categories which is robot body RM100 and the component RM100. Below are shown the price of each component and total. This is not included other material which is soldering iron, battery, solder, casing to keep the circuit.

No	Component/Material	Specification	Price/Unit(RM)	Quantities	Price(RM)
1	PIC1F8771	40-pin PDIP	23.00	1	23.00
2	LM7805		1.20	1	1.20
3	Crystal	20 MHz	8.00	1	8.00
4	Independent Board	10”x4”	5.00	1	5.00
5	PCB Header	40-ways	0.80	10	8.00
6	Wire Wrap		15.00	1	15.00
7	Heat Sink		0.90	1	0.90
8	Reset Button		0.50	1	0.50
9	L293B	16-pin PDIP	8.70	1	8.70
10	LM324		1.50	1	1.50
11	Resistor	10kΩ	0.10	3	0.30
12	Resistor	470Ω	0.10	2	0.20
13	Resistor	270Ω	0.10	2	0.20
14	Capacitor	100uF	0.40	1	0.40
15	Capacitor	1uF	0.20	2	0.40

16	Capacitor	22pF	0.20	2	0.40
17	LED		0.20	4	0.80
18	IR Transmitter		2.50	2	5.00
19	IR Receiver		2.50	2	5.00
20	Adapter Connector		3.00	1	3.00
21	9V Battery Cap		0.30	2	0.60
				Total	88.10

This project cans categories in middle range of expansion because it only consume about RM200. For commercialization, this mobile robot should do the work in path which is usually in the factory where to transport the goods from department to another department without making any error even though there is a disturbance in the path. The main part for commercialization in this project is the circuits of the robot, because this is where about the PID controller to control the robot.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

This project has implemented the Proportional-Integral-Derivative controller (PID controller) into the path follower mobile robot. The controller has improved the navigation for the mobile robot to traverse straight line and turn 90-degree right.

The objective of this project is to develop the path follower mobile robot.

5.2 FUTURE RECOMMENDATION

For the future recommendation, I recommend to add more controllers to this mobile robot in order to perfect the navigation system of the mobile robot. These can be done by adding angle detection to correct the error and determine how big the error has been made by the robot. The robot should manipulate the error and correct its without do any further error to the system.

Developing hardware would be perfect to acquire the result of the controller in the mobile robot such as oscilloscope with data recorded or hardware similar with it. The result of the path follower mobile robot is important to support the theories of this project. Judging by looking how it works not be enough because we still not know the mobile already ready to any job without need any tuning.

Designing the light weight of body should consider because of limitation of battery power. The mobile robot should only use battery to operate without using any additional power supply this will affecting the whole circuit, the body, and the path for the mobile robot working.

REFERENCE

[1] José Castro, Vitor Santos, M. Isabel Ribeiro (1998), *A Multi-Loop Robust Navigation Architecture for Mobile Robots*, IEEE International Conference on Robotics & Automation Leuven. Belgium, page 970-975, May 1998.

[2] Robert W. Hogg, Arturo L. Rankin, Stergios I. Roumeliotis, Michael C. McHenry, Daniel M. Helmick, Charles E Bergh, Larry Matthies, *Algorithms and Sensors for Small Robot Path Following*, IEEE International Conference on Robotics 8 Automation Washington, DC, page 3850-3857, May 2002

[3] Thomas M. Howard, Ross A. Knepper, and Alonzo Kelly, C*onstrained Optimization Path Following of Wheeled Robots in Rough Terrain*, NASA/JPL as the Mars Technology Program.

[4] *"MOS DATA 1976"*, General Instrument 1976 Data book

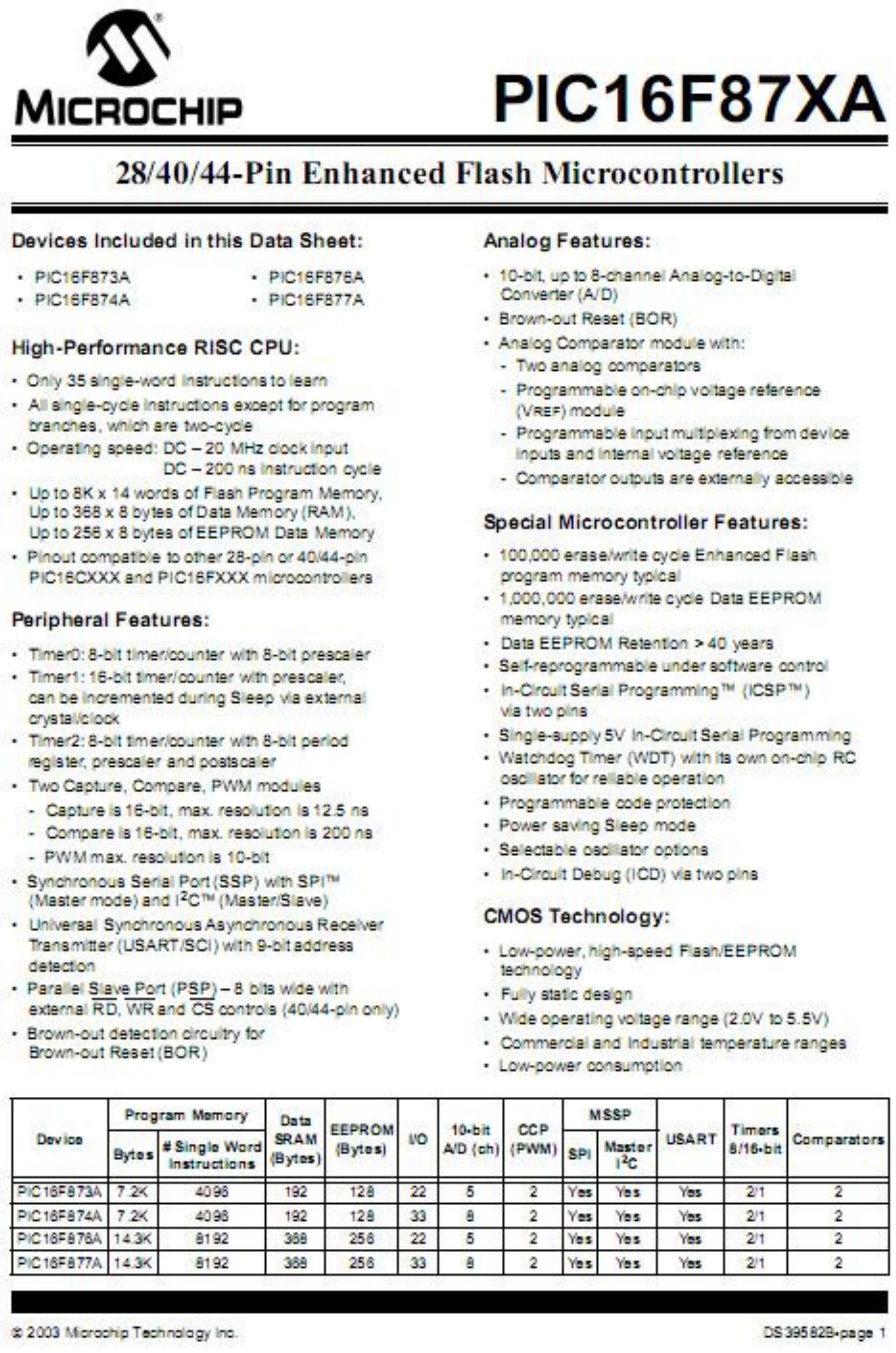
[5] *"1977 Data Catalog"*, Micro Electronics from General Instrument Corporation

[6] Microchip Technology (2008-02-27). "Microchip Technology Delivers Six Billionth PIC® Microcontroller", http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2018&mcparam=en534302

[7] Citing Internet Source, Wikipedia, List of Programming Language, http://en.wikipedia.org/wiki/List_of_programming_languages

Appendix A

Datasheet PIC 16F877A1



MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory, Up to 368 x 8 bytes of Data Memory (RAM), Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

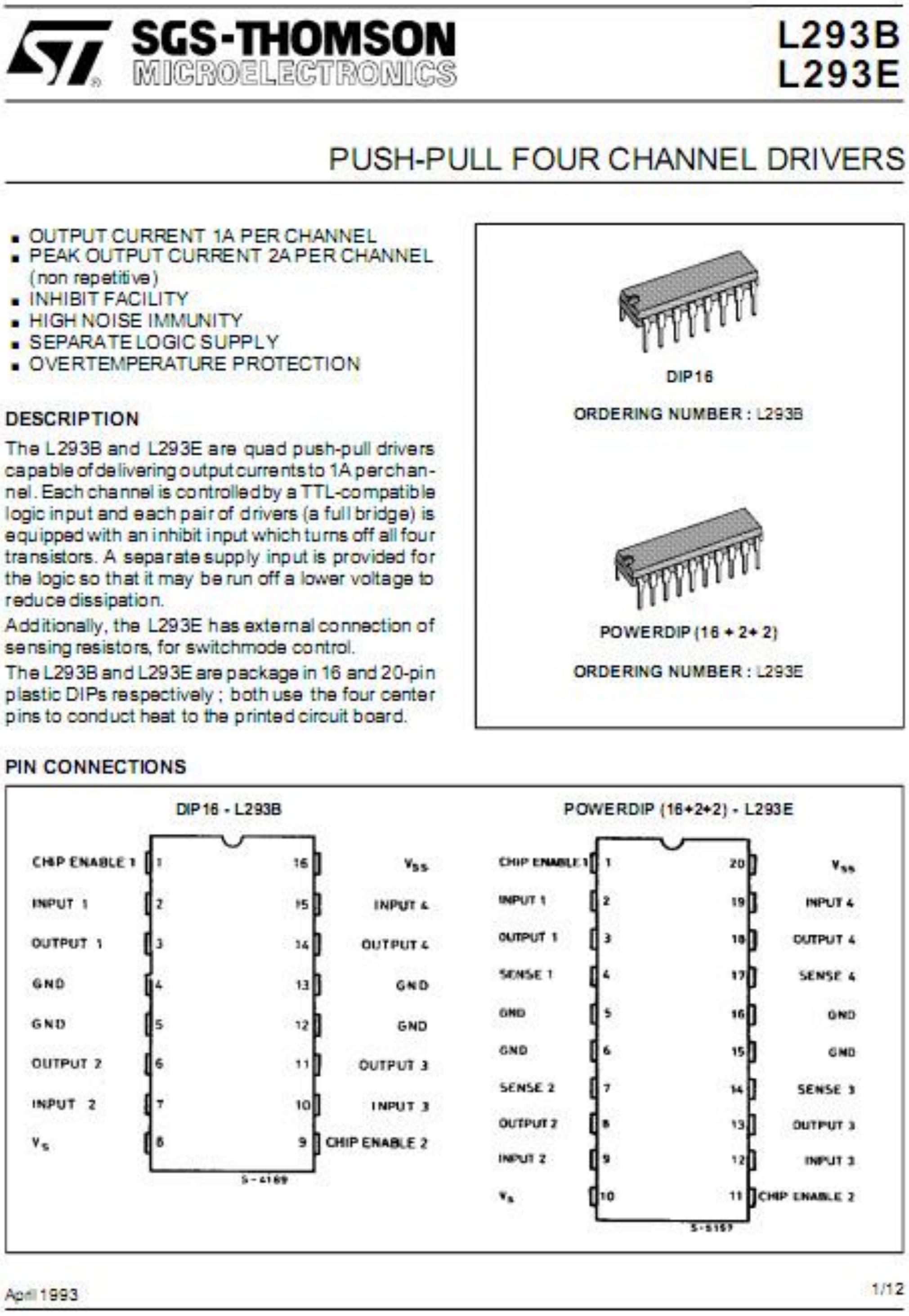
Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^{2}C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

© 2003 Microchip Technology Inc.

DS39582B-page 1

Appendix B

Datasheet L293B

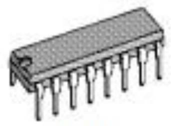


SGS-THOMSON MICROELECTRONICS

L293B
L293E

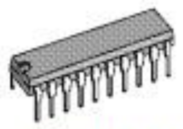
PUSH-PULL FOUR CHANNEL DRIVERS

- OUTPUT CURRENT 1A PER CHANNEL
- PEAK OUTPUT CURRENT 2A PER CHANNEL (non repetitive)
- INHIBIT FACILITY
- HIGH NOISE IMMUNITY
- SEPARATE LOGIC SUPPLY
- OVERTEMPERATURE PROTECTION



DIP16

ORDERING NUMBER : L293B



POWERDIP (16 + 2+ 2)

ORDERING NUMBER : L293E

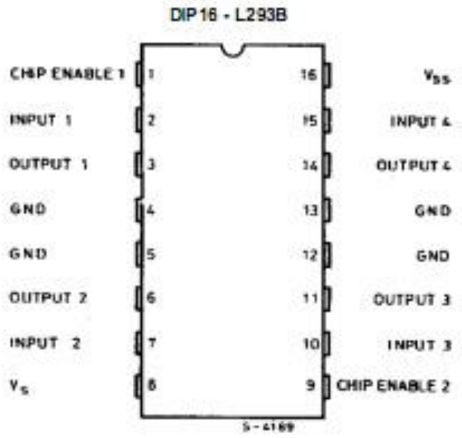
DESCRIPTION

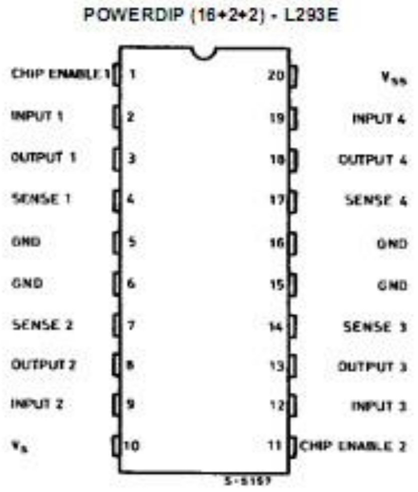
The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally, the L293E has external connection of sensing resistors, for switchmode control.

The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTIONS

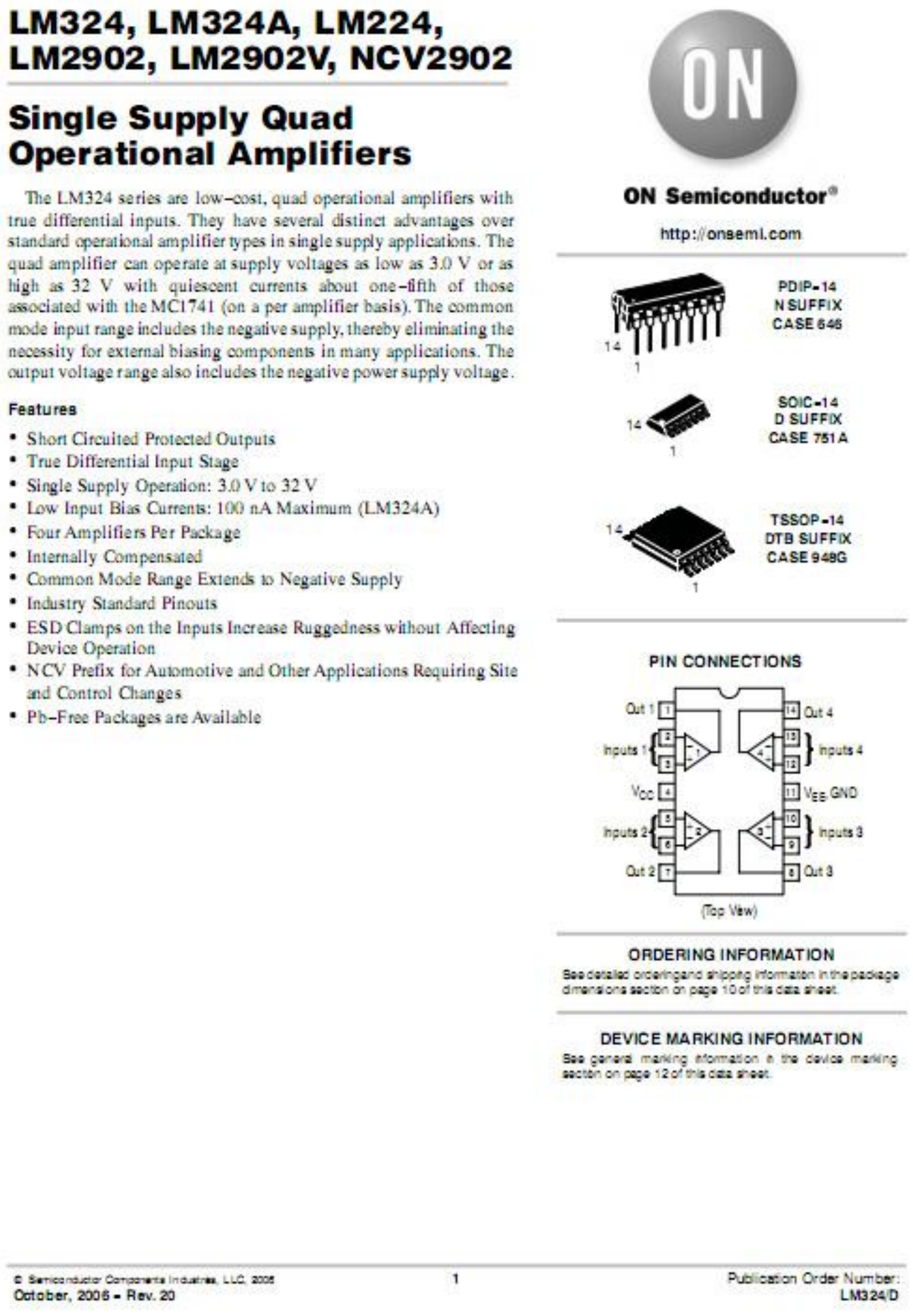




April 1993 1/12

Appendix C

Datasheet LM324



LM324, LM324A, LM224, LM2902, LM2902V, NCV2902

Single Supply Quad Operational Amplifiers

The LM324 series are low–cost, quad operational amplifiers with true differential inputs. They have several distinct advantages over standard operational amplifier types in single supply applications. The quad amplifier can operate at supply voltages as low as 3.0 V or as high as 32 V with quiescent currents about one–fifth of those associated with the MC1741 (on a per amplifier basis). The common mode input range includes the negative supply, thereby eliminating the necessity for external biasing components in many applications. The output voltage range also includes the negative power supply voltage.

Features

- Short Circuited Protected Outputs
- True Differential Input Stage
- Single Supply Operation: 3.0 V to 32 V
- Low Input Bias Currents: 100 nA Maximum (LM324A)
- Four Amplifiers Per Package
- Internally Compensated
- Common Mode Range Extends to Negative Supply
- Industry Standard Pinouts
- ESD Clamps on the Inputs Increase Ruggedness without Affecting Device Operation
- NCV Prefix for Automotive and Other Applications Requiring Site and Control Changes
- Pb–Free Packages are Available

ON Semiconductor®

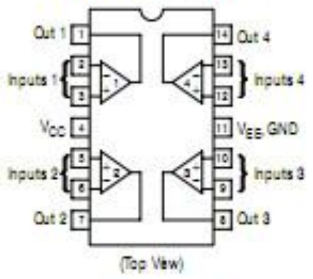
http://onsemi.com

PDIP–14
N SUFFIX
CASE 646

SOIC–14
D SUFFIX
CASE 751A

TSSOP–14
DTB SUFFIX
CASE 948G

PIN CONNECTIONS



ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 10 of this data sheet.

DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 12 of this data sheet.

Publication Order Number:
LM324/D

Appendix D1

Example Program

By Using Assembly Language

```
;*****************************************************************
;                                             *
;    Filename:1.asm                               *
;    Date:9/7/2008                                  *
;    File Version:                                *
;                                             *
;    Author: Muhammad Bin Mazlan                           *
;    Company: EEES,UMP                              *
;                                             *
;                                             *
;*****************************************************************


list      p=16F877A
#include <p16F877A.inc>

__CONFIG   0X3FF2

;**************** VARIABLE DEFINITIONS**********************

D1                    EQU          0X20   ;FOR DELAY
D2                    EQU          0X21   ;FOR DELAY
D3                    EQU          0X22   ;FOR DELAY

BANK1          MACRO
BSF            STATUS, RP0                      ;MACRO TO SELECT BANK 1
```

```
ENDM

BANK0           MACRO
BCF             STATUS, RP0                       ;MACRO TO SELECT BANK 0
ENDM



;*******************************************************************
ORG     0x000
goto    main

main

BANK1
MOVLW           B'11111111'
MOVWF           TRISA
MOVWF           TRISE
CLRF  TRISB
CLRF  TRISC

BANK0
MOVLW           B'00000000'
MOVWF           PORTA
MOVWF           PORTB
MOVWF           PORTC
MOVWF           PORTE

SCAN
CLRF  PORTB
BTFSC           PORTA,B'00000001'
GOTO PROG1
```

```
BTFSC       PORTA,B'00000010'
GOTO PROG2
BTFSC       PORTA,B'00000100'
GOTO PROG3
BTFSC       PORTE,B'00000001'
GOTO SENSOR1
BTFSC       PORTE,B'00000010'
GOTO SENSOR2
GOTO SCAN

PROG1               MOVLW       B'11111111'
MOVWF       PORTB
CALL  DELAY
MOVLW       B'00000000'
MOVWF       PORTB
CALL  DELAY
GOTO SCAN

PROG2               MOVLW       B'10101010'
MOVWF       PORTB
CALL  DELAY
MOVLW       B'01010101'
MOVWF       PORTB
CALL  DELAY
GOTO SCAN

PROG3               MOVLW       B'10000000'
MOVWF       PORTB
REPEAT              RRF    PORTB
CALL  DELAY
BTFSSPORTB,B'00000001'
```

```
GOTO REPEAT
GOTO SCAN

SENSOR1          MOVLW     B'10000000'
MOVWF     PORTB
GOTO SCAN

SENSOR2          MOVLW     B'00000001'
MOVWF     PORTB
GOTO SCAN

;******************************DELAY
SUBROUTINE****************************

DELAY                  MOVLW     D'200'
MOVWF     D3
MOVLW     D'30'
MOVWF     D2
MOVLW     D'150'
MOVWF     D1
DECFSZ    D1
GOTO $-1
DECFSZ    D2
GOTO $-5
DECFSZ    D3
GOTO $-9
RETURN


END            ; directive 'end of program'
```

Appendix D2

Example of List File

```
MPASM  5.16                          PFMR.ASM   10-11-2008  20:52:36
PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00001
;**********************************************************************
                      00002 ;
*
                      00003 ;    Filename:Path Follower Mobile
Robot.asm                      *
                      00004 ;    Date:5/10/2008
*
                      00005 ;    File Version: 1.0
*
                      00006 ;
*
                      00007 ;    Author: Muhammad Bin Mazlan
*
                      00008 ;    Company: EEES,UMP
*
                      00009 ;
*
                      00010 ;
*
                      00011
;**********************************************************************
                      00012
                      00013
                      00014         list      p=16F877A
                      00015         #include <p16F877A.inc>
                      00001         LIST
                      00002 ; P16F877A.INC  Standard Header File,
Version 1.00    Microchip Technology, Inc.
                      00400         LIST
                      00016
2007   3FF2           00017         __CONFIG  0X3FF2
                      00018
                      00019
                      00020
0000                  00021                 ORG     0x000
0000   2801           00022                 goto    main
                      00023
                      00024
0001                  00025 main
                      00026
Warning[207]: Found label after column 1. (BANK1)
0001                  00027                 BANK1
0001   3000           00028                 movlw   b'00000000'
```

```
Message[302]: Register in operand not in bank 0.  Ensure that bank bits
are correct.
0002   0089           00029                 movwf   TRISE
Message[302]: Register in operand not in bank 0.  Ensure that bank bits
are correct.
0003   0187           00030                 CLRF    TRISC
                      00031
Warning[207]: Found label after column 1. (BANK0)
0004                  00032                 BANK0
0004   3000           00033                 MOVLW   B'00000000'
0005   0087           00034                 MOVWF   PORTC
                      00035
0006                  00036 SCAN
0006   0187           00037                 CLRF    PORTC
0007   1989           00038                 BTFSC   PORTE,B'00000011'
0008   280A           00039                 GOTO    PROG1
                      00040
0009   2806           00041                 GOTO    SCAN
                      00042
000A   30FF           00043 PROG1           MOVLW   B'11111111'
000B   0087           00044                 MOVWF   PORTC
000C   280A           00045                 GOTO    PROG1
                      00046
MPASM  5.16                        PFMR.ASM   10-11-2008  20:52:36
PAGE  2


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00047                 END
MPASM  5.16                        PFMR.ASM   10-11-2008  20:52:36
PAGE  3


SYMBOL TABLE
  LABEL                              VALUE

ACKDT                             00000005
ACKEN                             00000004
ACKSTAT                           00000006
ADCON0                            0000001F
ADCON1                            0000009F
ADCS0                             00000006
ADCS1                             00000007
ADCS2                             00000006
ADDEN                             00000003
ADFM                              00000007
ADIE                              00000006
ADIF                              00000006
ADON                              00000000
ADRESH                            0000001E
ADRESL                            0000009E
BANK0                             00000004
BANK1                             00000001
BCLIE                             00000003
```

```
BCLIF                             00000003
BF                                00000000
BRGH                              00000002
C                                 00000000
C1INV                             00000004
C1OUT                             00000006
C2INV                             00000005
C2OUT                             00000007
CCP1CON                           00000017
CCP1IE                            00000002
CCP1IF                            00000002
CCP1M0                            00000000
CCP1M1                            00000001
CCP1M2                            00000002
CCP1M3                            00000003
CCP1X                             00000005
CCP1Y                             00000004
CCP2CON                           0000001D
CCP2IE                            00000000
CCP2IF                            00000000
CCP2M0                            00000000
CCP2M1                            00000001
CCP2M2                            00000002
CCP2M3                            00000003
CCP2X                             00000005
CCP2Y                             00000004
CCPR1H                            00000016
CCPR1L                            00000015
CCPR2H                            0000001C
CCPR2L                            0000001B
CHS0                              00000003
CHS1                              00000004
CHS2                              00000005
CIS                               00000003
CKE                               00000006
MPASM  5.16                          PFMR.ASM   10-11-2008  20:52:36
PAGE  4


SYMBOL TABLE
  LABEL                             VALUE

CKP                               00000004
CM0                               00000000
CM1                               00000001
CM2                               00000002
CMCON                             0000009C
CMIE                              00000006
CMIF                              00000006
CREN                              00000004
CSRC                              00000007
CVR0                              00000000
CVR1                              00000001
CVR2                              00000002
CVR3                              00000003
CVRCON                            0000009D
```

```
CVREN                             00000007
CVROE                             00000006
CVRR                              00000005
D                                 00000005
DATA_ADDRESS                      00000005
DC                                00000001
D_A                               00000005
EEADR                             0000010D
EEADRH                            0000010F
EECON1                            0000018C
EECON2                            0000018D
EEDATA                            0000010C
EEDATH                            0000010E
EEIE                              00000004
EEIF                              00000004
EEPGD                             00000007
F                                 00000001
FERR                              00000002
FSR                               00000004
GCEN                              00000007
GIE                               00000007
GO                                00000002
GO_DONE                           00000002
I2C_DATA                          00000005
I2C_READ                          00000002
I2C_START                         00000003
I2C_STOP                          00000004
IBF                               00000007
IBOV                              00000005
INDF                              00000000
INTCON                            0000000B
INTE                              00000004
INTEDG                            00000006
INTF                              00000001
IRP                               00000007
NOT_A                             00000005
NOT_ADDRESS                       00000005
NOT_BO                            00000000
NOT_BOR                           00000000
MPASM  5.16                          PFMR.ASM   10-11-2008  20:52:36
PAGE  5


SYMBOL TABLE
  LABEL                             VALUE

NOT_DONE                          00000002
NOT_PD                            00000003
NOT_POR                           00000001
NOT_RBPU                          00000007
NOT_RC8                           00000006
NOT_T1SYNC                        00000002
NOT_TO                            00000004
NOT_TX8                           00000006
NOT_W                             00000002
NOT_WRITE                         00000002
```

```
OBF                               00000006
OERR                              00000001
OPTION_REG                        00000081
P                                 00000004
PCFG0                             00000000
PCFG1                             00000001
PCFG2                             00000002
PCFG3                             00000003
PCL                               00000002
PCLATH                            0000000A
PCON                              0000008E
PEIE                              00000006
PEN                               00000002
PIE1                              0000008C
PIE2                              0000008D
PIR1                              0000000C
PIR2                              0000000D
PORTA                             00000005
PORTB                             00000006
PORTC                             00000007
PORTD                             00000008
PORTE                             00000009
PR2                               00000092
PROG1                             0000000A
PS0                               00000000
PS1                               00000001
PS2                               00000002
PSA                               00000003
PSPIE                             00000007
PSPIF                             00000007
PSPMODE                           00000004
R                                 00000002
RBIE                              00000003
RBIF                              00000000
RC8_9                             00000006
RC9                               00000006
RCD8                              00000000
RCEN                              00000003
RCIE                              00000005
RCIF                              00000005
RCREG                             0000001A
RCSTA                             00000018
RD                                00000000
MPASM  5.16                          PFMR.ASM   10-11-2008  20:52:36
PAGE  6


SYMBOL TABLE
  LABEL                             VALUE

READ_WRITE                        00000002
RP0                               00000005
RP1                               00000006
RSEN                              00000001
RX9                               00000006
RX9D                              00000000
```

```
R_W                               00000002
S                                 00000003
SCAN                              00000006
SEN                               00000000
SMP                               00000007
SPBRG                             00000099
SPEN                              00000007
SREN                              00000005
SSPADD                            00000093
SSPBUF                            00000013
SSPCON                            00000014
SSPCON2                           00000091
SSPEN                             00000005
SSPIE                             00000003
SSPIF                             00000003
SSPM0                             00000000
SSPM1                             00000001
SSPM2                             00000002
SSPM3                             00000003
SSPOV                             00000006
SSPSTAT                           00000094
STATUS                            00000003
SYNC                              00000004
T0CS                              00000005
T0IE                              00000005
T0IF                              00000002
T0SE                              00000004
T1CKPS0                           00000004
T1CKPS1                           00000005
T1CON                             00000010
T1INSYNC                          00000002
T1OSCEN                           00000003
T1SYNC                            00000002
T2CKPS0                           00000000
T2CKPS1                           00000001
T2CON                             00000012
TMR0                              00000001
TMR0IE                            00000005
TMR0IF                            00000002
TMR1CS                            00000001
TMR1H                             0000000F
TMR1IE                            00000000
TMR1IF                            00000000
TMR1L                             0000000E
TMR1ON                            00000000
TMR2                              00000011
TMR2IE                            00000001
MPASM  5.16                          PFMR.ASM   10-11-2008  20:52:36
PAGE  7


SYMBOL TABLE
  LABEL                             VALUE

TMR2IF                            00000001
TMR2ON                            00000002
```

```
TOUTPS0                           00000003
TOUTPS1                           00000004
TOUTPS2                           00000005
TOUTPS3                           00000006
TRISA                             00000085
TRISB                             00000086
TRISC                             00000087
TRISD                             00000088
TRISE                             00000089
TRISE0                            00000000
TRISE1                            00000001
TRISE2                            00000002
TRMT                              00000001
TX8_9                             00000006
TX9                               00000006
TX9D                              00000000
TXD8                              00000000
TXEN                              00000005
TXIE                              00000004
TXIF                              00000004
TXREG                             00000019
TXSTA                             00000098
UA                                00000001
W                                 00000000
WCOL                              00000007
WR                                00000001
WREN                              00000002
WRERR                             00000003
Z                                 00000002
_BODEN_OFF                        00003FBF
_BODEN_ON                         00003FFF
_CPD_OFF                          00003FFF
_CPD_ON                           00003EFF
_CP_ALL                           00001FFF
_CP_OFF                           00003FFF
_DEBUG_OFF                        00003FFF
_DEBUG_ON                         000037FF
_HS_OSC                           00003FFE
_LP_OSC                           00003FFC
_LVP_OFF                          00003F7F
_LVP_ON                           00003FFF
_PWRTE_OFF                        00003FFF
_PWRTE_ON                         00003FF7
_RC_OSC                           00003FFF
_WDT_OFF                          00003FFB
_WDT_ON                           00003FFF
_WRT_1FOURTH                      00003BFF
_WRT_256                          00003DFF
_WRT_HALF                         000039FF
_WRT_OFF                          00003FFF
_XT_OSC                           00003FFD

MPASM  5.16                          PFMR.ASM   10-11-2008  20:52:36
PAGE  8
```

```
SYMBOL TABLE
  LABEL                             VALUE

__16F877A                         00000001
main                              00000001


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXX--- ---------------- ---------------- -------------
---
2000 : -------X-------- ---------------- ---------------- -------------
---

All other memory blocks unused.

Program Memory Words Used:    13
Program Memory Words Free:  8179


Errors   :     0
Warnings :     2 reported,     0 suppressed
Messages :     2 reported,     0 suppressed
```

Appendix D3

Example of Error File

```
Message[302] C:\TESTING\CUBAAN.ASM 96 : Register in operand not in bank
0.  Ensure that bank bits are correct.
Message[305] C:\TESTING\CUBAAN.ASM 121 : Using default destination of 1
(file).
Message[305] C:\TESTING\CUBAAN.ASM 123 : Using default destination of 1
(file).
Message[305] C:\TESTING\CUBAAN.ASM 125 : Using default destination of 1
(file).
```