LINE FOLLOWING MOBILE ROBOT PROTOTYPE

LYDIANA BINTI SORUPI

This Report is submitted in Partial Fulfillment of Requirements for the Bachelor Degree of Electrical Engineering (Electronic Power and Drive)

Fakulti Kejuruteraan Elektrik
Universiti Teknikal Malaysia Melaka

APRIL 2007

ABSTRAK

Tujuan utama perlaksannan projek ini ialah untuk mengaplikasikan teknologi di dalam pengoperasian sesuatu teknologi yang boleh bergerak sendiri. Projek robot bergerak mengikut garisan ini melibatkan pengetahuan mengenai program PIC dan gabungan liatr kawalan seperti motor dan sensor. Robot ini mengandungi tiga sensor dan di program untuk mengikut garisan hitam di atas permukaan putih. Untuk memprogramkan robot ini, pengawal mikro PIC digunakan untuk mengawal pergerakan robot mobil ini. Dua motor pelangkah digunakan untuk mengerakkan robot mobil ini. Secara keseluruhan, pembinaan robot ini melibatkan interaksi antara sensor, pengawal mikro dan motor.

ABSTRACT

The purpose of doing this project is to apply the micro technologies to the operational of other technologies, so that it can work independently. Line follower mobile robot project contains an external knowledge of PIC program and mixture of control circuit such as motor and sensor this mobile robot uses three sensors and being program to follow the black line on the floor. To program the mobile robot, PIC microcontroller is used to detect the black line and make the mobile robot moving on track. Two stepper motors are used to move the mobile robot. This project involves the interaction of sensors, DC motor and microcontroller.

CHAPTER I

INTRODUCTION

This chapter will explain the objective and scope of the project. The purpose of this chapter is to explain the perspective and method that will be used in implementation and observation of how far the project is related with the research and theory.

1.1 Project Objective

The main objective of this project is to study the development of a mobile robot that will move following the black line on the floor. To achieve this main objective, there are some sub objectives that need to achieve first that are; to develop an efficient drive system that will able to travel following the line and make a 180° turn without offsetting, to develop a tracking system that is able to turn when sensors detected at end of the line, to develop an efficient power consumption and distribution so that the robot can start and finish the line with a single set of batteries and to study the development of PIC16F84A microcontroller.

1.2 Problem Statement

In factories, the workers usually moved objects or materials manually or using forklift. This method will waste a lot of man power and also will waste money to employ more man power to do the task. This robot can be used to substitute the man power in this system. It can be used to move objects to any desired location by its own. Line grids can be placed on the floor to indicate the path that the robot requires to go. The employ of this robot will save cost for man power.

1.3 Scope of Work

The scope of work for this project is divided into two parts, the hardware and the software. The hardware involves designing the mechanical parts and electrical parts of the line follower mobile robot. The mechanical part involves the chassis, the drive system, the sensors layout and the electrical part involves in microcontroller, drive circuit and senor interfaces.

The software involve designing and writing the program for the robot to move forward, backward, turn left or right. Software also used to develop an algorithm for tracking the line.

CHAPTER II

LITERATURE REVIEW

This chapter will discuss about the overall theories and concepts of the project. The purpose of this chapter is to explain the perspective and method used to implement the line follower mobile robot. The full understanding of the theory is very important as a guideline.

2.1 The Robot

A lot of robot's definition can be found. It is not easy to bear the exact and perfect meaning of robot. The difference of these definitions can be seen from many aspects and angles. Some defined robot from aspect that the robot can be re-programmed and some defined robot by its manipulation, behaviors, intelligence and etc.

According to history, robot has been invented in 1921 and introduced by a Czech dramatis, Karel Capek in "Rossum's Universal Robot" drama. While the pronunciation and the word robotic was introduced by Isaac Asimov in robot science fiction story in 1940. The Webstar's New World Dictionary has defined robotic as science and technology of robot development that include design, production, application and other function. In Europe, robotic defined as "robotilogi science" while robotilogi was defined as "method of how the robot can be manipulated to do work".

Most of us assume robot as a part of technology but robotic actually comprehend all field of technology such as mechanical, electrical, electronics, computer software and various type of sophisticated technologies.

2.1.1 Line Follower Mobile Robot

Basically, robot can classify into two categories; fixed robot and mobile robot. Fixed robot is a robot that placed in one static place and its work limited by area. Mobile robot can move from one place to another place. Mobile robots commonly used in dangerous task such as in bomb annihilate process. Industrial and agricultural field also use mobile robot to cultivate seeds and to harvest. At home, mobile robot probably manipulates as security and do the daily houses work [1].

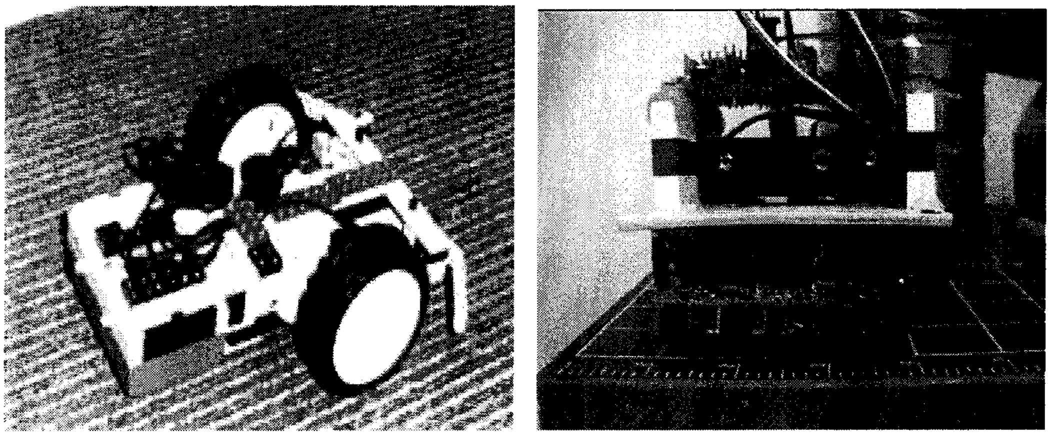


Figure 2.1: Types of mobile robot

The line follower mobile robot will basically made up of four main systems. The microcontroller which is the brains of the robot makes all the decision in moving the robot and makes the robot tracking the black line. The drive system which is the legs of the robot moves the robot anywhere it wants to go. The sensors systems represent the eyes of the robot help the robot to detect the line. Lastly, the power supply which is the heart of the mouse that will pumps power to all the other systems.

There are some examples of other line follower mobile robot that have been collected. Many of these are from United Kingdom and abroad.

Award winner from VingPeaw Competition 2543 was designed by Plermjai Inchuay. The robot built with 2051, L293D, and four IR sensors. Simple circuit and platform, quick tracking and easy-understand program using C language. It use two motors control rear wheels and the single front wheel is free. It has 4-infrared sensors on the bottom for detect black tracking tape. When the sensors detected black color, output of comparator, LM324 is low logic and the other the output is high. Microcontroller AT89C2051 and H-Bridge driver L293D were used to control direction and speed of motor.

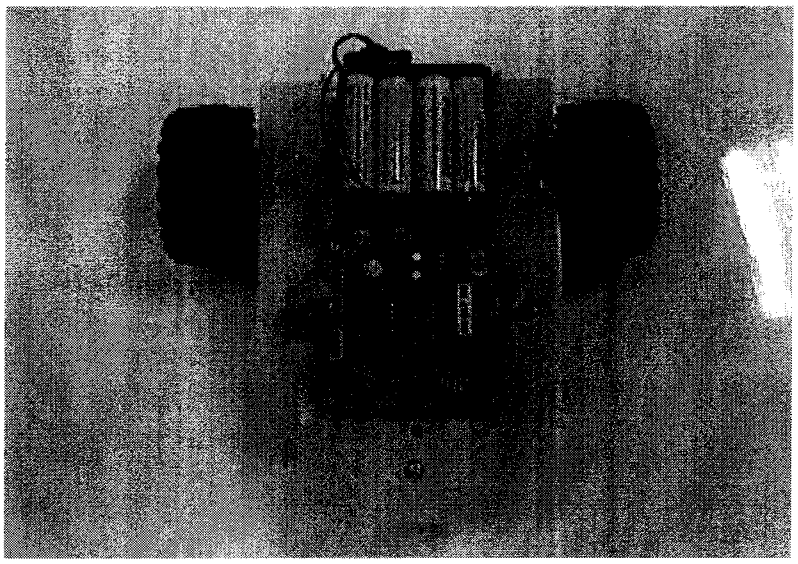


Figure 2.2: Line follower mobile robot by Plermjai Inchuay.

Another example is trekker line follower mobile robot and sumo wrestler. This robot can track line and also can be used as a sumo wrestler where it will find another robot and push them out of the ring while using the line following sensors to make sure it stays in the ring.

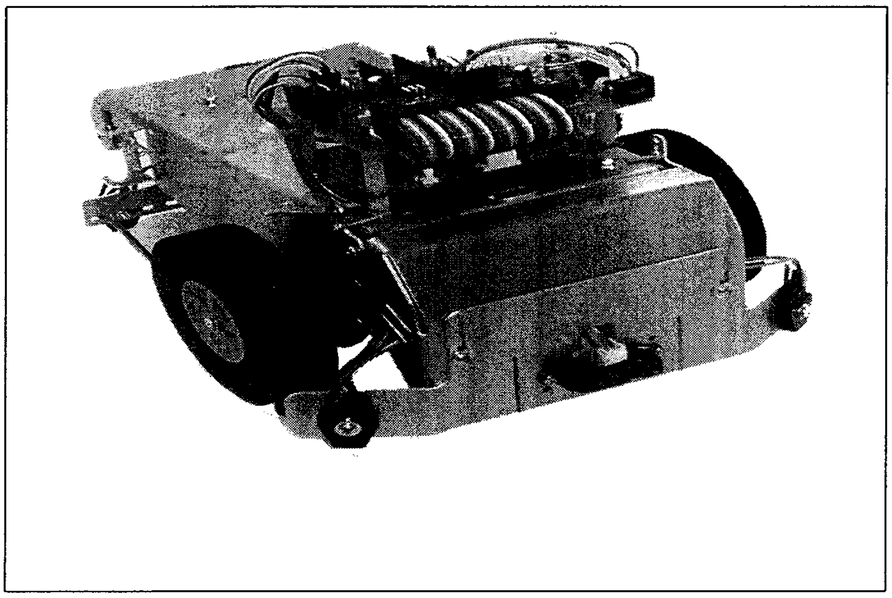


Figure 2.3: Trekker line follower mobile robot and sumo wrestler

2.2 Microcontroller

Computer revolution since the past fifteen years has produced computer that have high speed and power but still maintaining its small and compact size. This revolution has brought development in *Large Scale Integration (LSI)* and *Very Large Scale Integration (VLSI)* technology that contain thousands of transistors in a single chip. With the existing of this technology, so one technology called micro processor introduced. It includes input pin, output port, memory, timer and etc.

The main reason to use microcontroller is the cost and it easy to get. Even though it has various applications, the price is more cheaply compared with complex circuit such as IC MC14528B.

The second factor is its ability to re-program. User can program microcontroller to apply in their project. As an example, we can program microcontroller according to its output and input port provided.

Other than that, microcontroller also can operate logic and mathematics. This operation is very important in any project application. It can use for logic program to change analog input to digital *(ADC).*

Lastly is its ability to programmed in various type of software language such C++, C, *assembly language, basic pro* and any other language that make this microcontroller known as user friendly.

There are a lot of company that produced microcontroller such Atmel (Attiny 11), Hitachi (H8/3640), Microchip (PIC16CR54C), Motorola (68HC705KJI), Zilog (Z8E000) and etc.

The controller will works as the brain of robot that will do all the logic for the robot. It will interpret the output of sensor and control the drive system. All these tasks will be complete by programming the controller. In this project, PIC16F84A microcontroller will be use.

2.2.1 PIC16F84A

Microchip technology has series of microcontroller called PIC in purpose of better application and low cost. PIC usually assumed as an interface control between hardware and software. With this device, circuit efficiency problem can be overcome compared with application of common electronic components that easily influenced by temperature and noise factor.

There are some of PIC series such 12CXXX, 16C5X, 16CXXX, 17CXXX and the latest is 18CXXX. Each type of these PIC device have its own criteria such number of input and output port, memory, speed, and etc [2].

PIC16F84A is the microcontroller that used to control the robot. PIC16F84A unit is a 8-bit device composed of standard on-chip peripherals including 68 bytes of Data

Ram, 64 bytes of Data EEPROM, 14- bit wide instruction words and it also have four interrupt sources; external RBO/INT pin, TMRO timer overflow, PORTB<7:4> interrupt-on-change and Data EEPROM write complete.

The peripheral features is 13 I/O pins with individual direction control, high current sink/ source for direct LED drive; 25mA sink maximum per pin or 25mA source maximum per pin, TMRO: 8-bit timer/ counter with 8-bit programmable prescaler.

The special features for this microcontroller is its wide operating voltage range; for commercial and industrial, 2.0V to 5.5V, low power consumption :< 2mA typical @5V, 4MHz and also this device comes in PDIP and SSOP package.

This microcontroller can be re-program or erase the program anytime. User can modify their program in error occur or adding other device in the circuit.

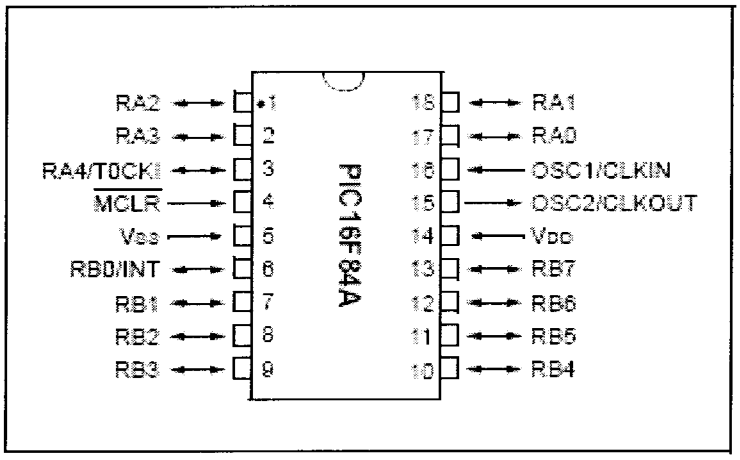


Figure 2.4: PIC16F84A Pin out Diagram

2.2.1.1 Memory

There are two memory blocks in PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can be further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The SFRs used to control the peripheral modules.

The data memory areas also contain the data EEPROM memory. EEPROM *Data Memory* is readable and writable during normal operation (full V_{DD} range). This memory is not directly mapped in the register file space. Instead it is indirectly addressed through *Special Register Function (*SRF). There are four SFRs used to read and write this memory. The EEPROM *data* memory allows byte read and writes. A byte write automatically erases the location and writes the new data (erase before write). The EEPROM *data memory* is rated for high erase/write cycles. The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature as well as from hip to chip.PIC16F84A microcontroller has 64 bytes of EEPROM *data memory.*

The PIC16F84A has a 13-bit program counter capable od addressing an 8K x 14 program memory space. The first 1K x 14 (0000h-03FFh) are physically implemented. Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h and 1C20h, the instruction will be the same.

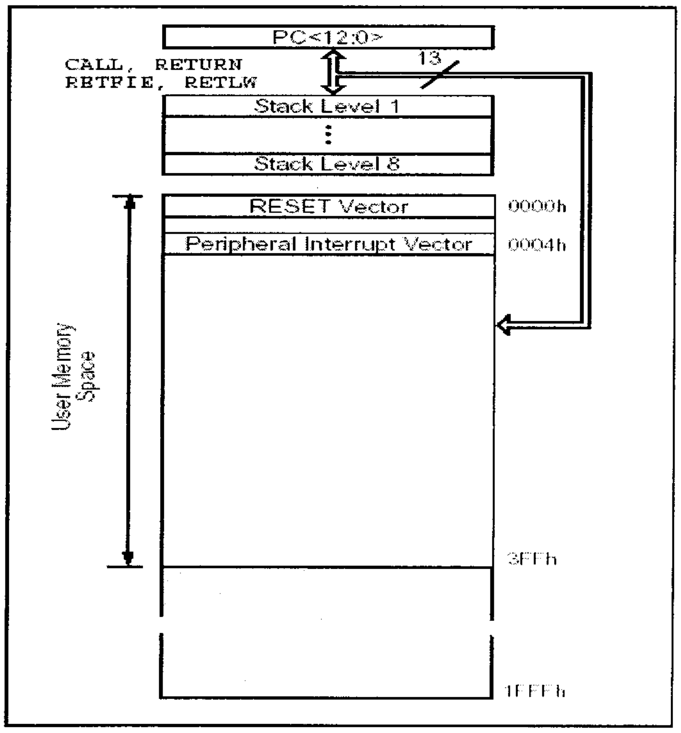


Figure 2.5: Program Memory Map and Stack

2.2.1.2 Peripherals

Some pins for these I/O ports are multiplexed with a n alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

PORTA is a 5-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (=1) will make the corresponding PORTA pin an input. Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output. Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-read operations. Therefore, a write to a port implies that the port pins are read. This value is modified and then written to the port data latch.

PORTB is an 8-bit wide, bi-directional port. This corresponding data direction register is TRISB. Setting a TRISB bit (=1) will make the corresponding PORTB pin an input. Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output. Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This can be performed by clearing bit RBPU (OPTION<7>). The weak pull-up is automatically turned off when the port pin is configured as an output.

The timer/counter in PIC16F84A is accurate and stable. The purpose of this timer/counter is to ensure the program running fast and smooth without time delay. The timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- Internal or external clock select
- Edge select for external clock
- 8-bit software programmable prescaler
- Interrupt-on-overflow from FFh to 00h.

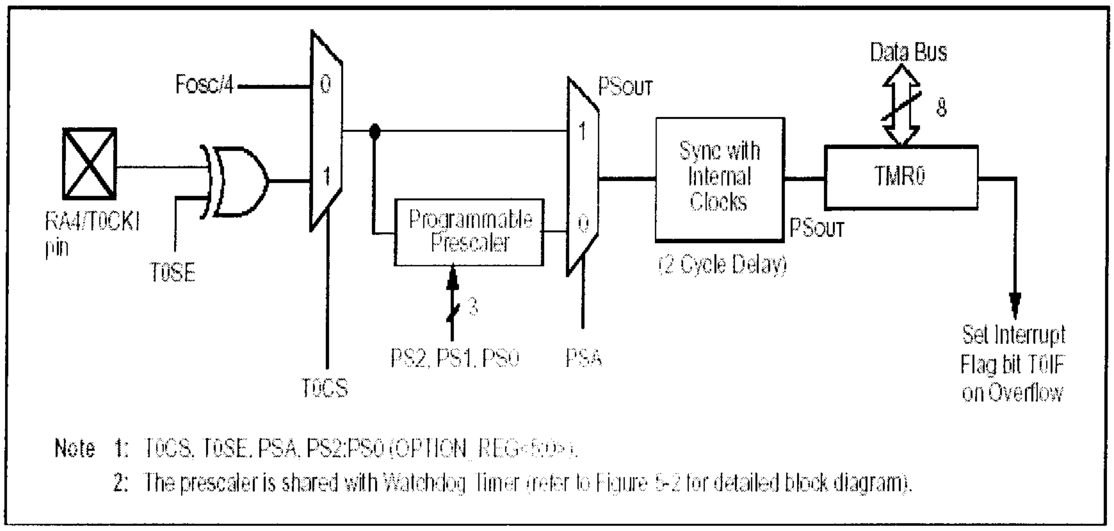

Data Bus
8
Fosc/4
0
1
PSOUT
1
0
Sync with Internal Clocks
TMR0
PSOUT
(2 Cycle Delay)
RA4/T0CKI pin
T0SE
Programmable Prescaler
3
PS2, PS1, PS0
PSA
T0CS
Set Interrupt Flag bit T0IF on Overflow
Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 5-2 for detailed block diagram).

Figure 2.6: Timer Block Diagram

2.2.1.3 Power Supply

PIC16F84A microcontroller needs +5V power supply to be function. It would not function if less than +5V and will cause the internal circuit damage or burn if over than +5V. Voltage regulator is the best choice to supply the accurate +5V power supply.

2.2.1.4 Oscillator

PIC16F84A microcontroller can be operating in range 4MHz - 20MHz frequencies. This oscillator can either use *Resonator* or *Crystal.*

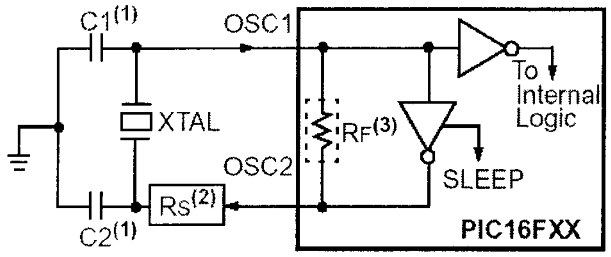


Figure 2.7: Crystal or Ceramic Resonator Operation

Table 2.1: Capacitor Selection for Ceramic Resonator

Ranges Tested:			
Mode	**Freq**	**OSC1/C1**	**OSC2/C2**
XT	455 kHz 2.0 MHz 4.0 MHz	47 - 100 pF 15 - 33 pF 15 - 33 pF	47 - 100 pF 15 - 33 pF 15 - 33 pF
HS	8.0 MHz 10.0 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

Table 2.2: Capacitor Selection for Crystal Oscillator

Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz 200 kHz	68 - 100 pF 15 - 33 pF	68 - 100 pF 15 - 33 pF
XT	100 kHz 2 MHz 4 MHz	100 - 150 pF 15 - 33 pF 15 - 33 pF	100 - 150 pF 15 - 33 pF 15 - 33 pF
HS	4 MHz 20 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

Recommended values of C1 and C2 are identical to the ranges tested in this table. Higher capacitance will increases the stability of the oscillator but also increases the start-up time.

2.2.2 MPLAB Integrated Development Environment (IDE)

The MPLAB IDE is software developed by Microchip's PICmicro for programming purposes with PIC microcontrollers. It is compatible with the PIC16F84A and with the use of this software, it is possible to write the assembly code, compile it and then download it to the PIC through the USB port of a computer which is connected to the PIC16F84A through the JDM multilink.

The MPLAB IDE provides debugging functionality when running the code on the PIC16F84A and also simulator mode to be used when running the program on the computer.

In the absence of the PIC16F84A, it is possible to debug the assembly code in the simulator mode of the MPLAB IDE. The software builds a simulation of the microcontroller with the compiled assembly code and the user may debug the code on the computer with windows showing the current state of the device memory locations and registers.

When debugging the assembly code that has been downloaded onto the PIC16F84A, the interface also provides the user with information of the PIC memory locations and registers as well similar to the simulator mode. The difference in this mode is that the code is executed on the PIC itself and is being debugged on the computer. The user interface of the real-time debugger is also the same as the simulator mode.

2.2.2.1 A Guide to Programming Using MPLAB IDE

Open the 'MPLAB' program. The program is located on three of the computers in Hicks 310. The window like the shown below will appear. (A dialog box will open - just hit "cancel" if do not wish to open the most recently used project file). If the menu looks different hit the "Swap Toolbar" button at the upper left of the MPLab until it looks right.

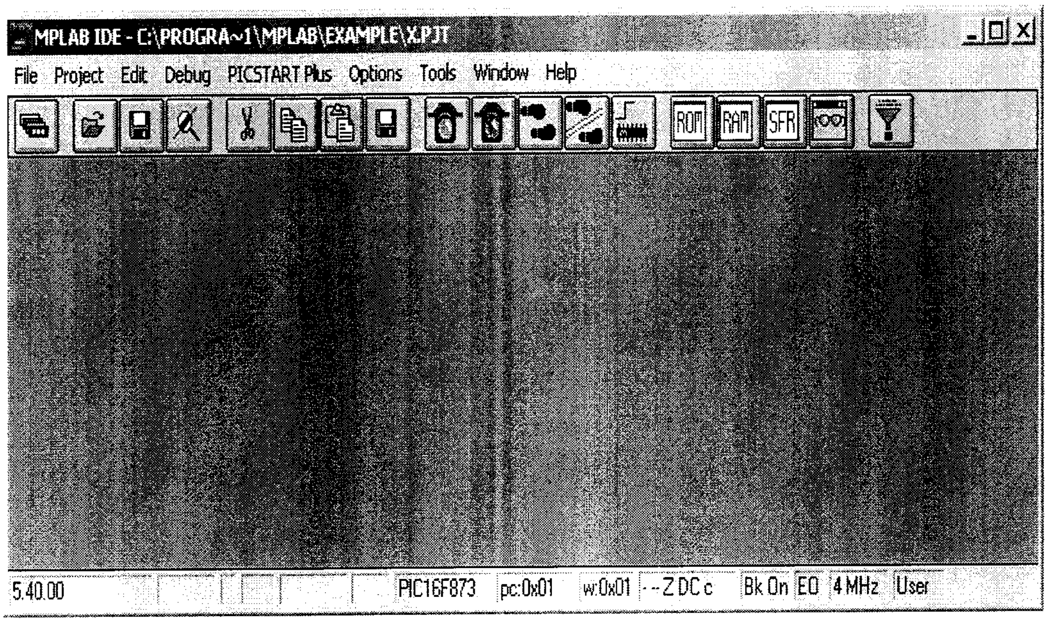
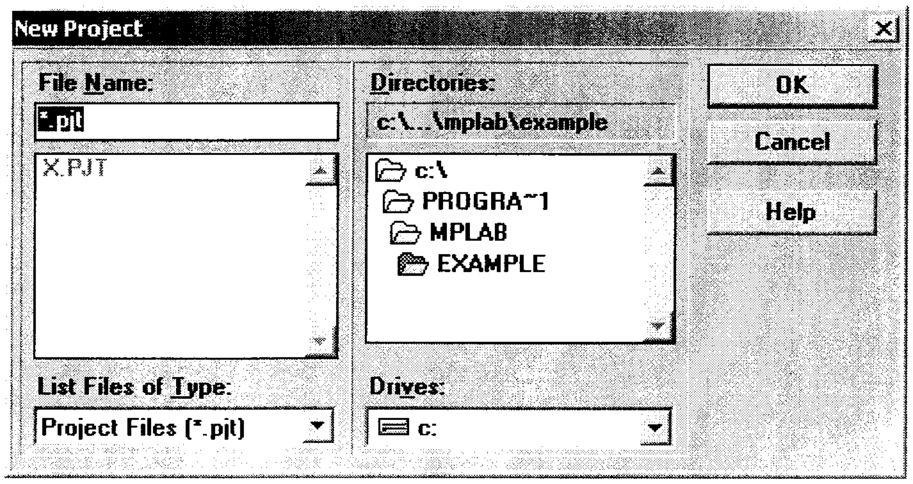

MPLAB IDE - C:\PROGRA~1\MPLAB\EXAMPLE\X.PJT
File Project Edit Debug PICSTART Plus Options Tools Window Help
ROM RAM SFR
5.40.00 PIC16F873 pc:0x01 w:0x01 --Z DC c Bk On EO 4MHz User

Figure 2.8: MPLab Open Menu

Go to the "Project" menu and choose "New Project". A dialog box will open up. Give the project a name ending with the extension ".pjt". Save the project in the folder in the MPLab/example folder.



File Name: *.pjt

X.PJT

Directories: c:\...\mplab\example

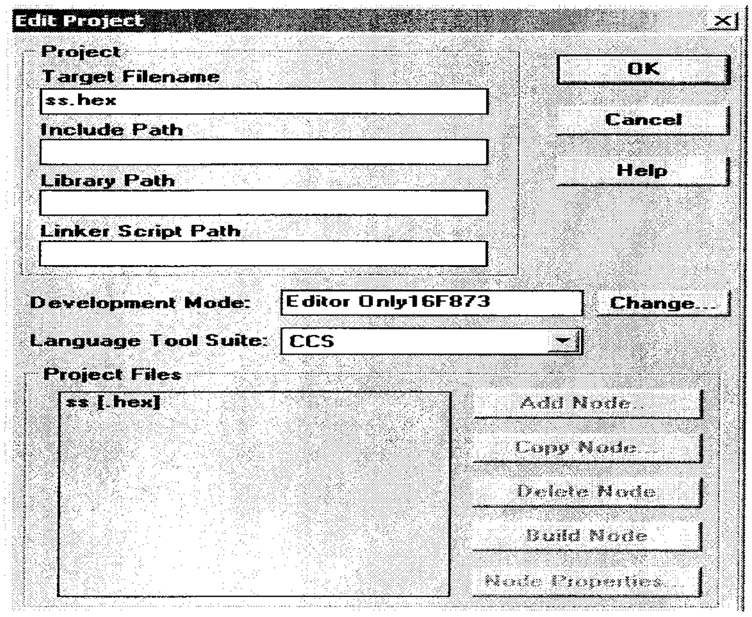
c:\
PROGRA~1
MPLAB
EXAMPLE

List Files of Type: Project Files (*.pjt)

Drives: c:

Figure 2.9: MPLab Project Menu

A dialog box "Edit Project" automatically pops open after click OK in the "New Project" box. Note that the "Language Tool Suite" must be set to "CCS".



Project

Target Filename: ss.hex

Include Path:

Library Path:

Linker Script Path:

Development Mode: Editor Only16F873

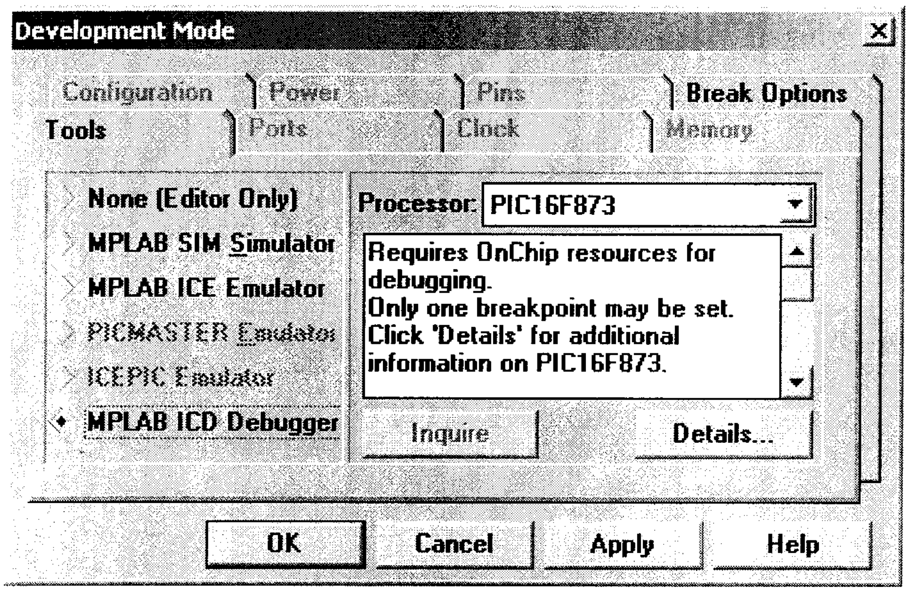
Language Tool Suite: CCS

Project Files

ss [.hex]

Figure 2.10: MPLab Edit Project

Select the microcontroller needed in the "Development Mode" dialog box by clicking on the "Change" button. The "Development Mode" dialog box opens up. Make selections as shown (MPLAB ICD Debugger and PIC16F873 processor).



Processor: PIC16F873

Requires OnChip resources for debugging.
Only one breakpoint may be set.
Click 'Details' for additional information on PIC16F873.

Figure 2.11: MPLab Development Mode

Make sure that "Language Tool Suite" box in the "Edit Project" dialog box has "CCS" selected. Move the cursor to the "Project Files" box at the bottom of the dialog box, click on ' .hex' file which is listed there. The "Node Properties" becomes active. Click on "Node Properties", or alternately double click on the displayed ' .hex' file in the "Project files" box. Under "Options", click on "PCM" which is the compiler that will be use for the PIC microcontroller. Click on the "OK" button at the bottom of the box. This process need to repeat as long as stay within the same project.

2.2.3 JDM Programmer

JDM is a simple PIC programmer which means that it will burn the program that we compile in MPLAB IDE to the PIC16F84A. It connects to the computers serial port through a DB9F female connecter.

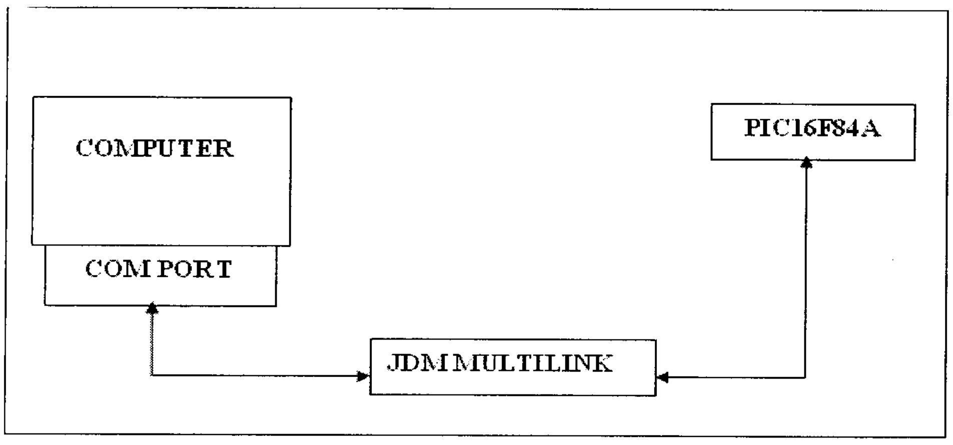


Figure 2.12: PIC16F84A and JDM Multilink setup

2.3 Software

Software is programming languages that can be understood by computer. Generally, there are three types of programming language; *machine language, assembly language* and *high level language.*

Machine language is a language that easily understood by computer. Programming in *machine language* no need to change or translate and ready to run. This language includes instruction from combination of binary number '0' and '1'.

Assembly language is a language that easier us to write program. It is the summary of instruction that we can understand but it need translator that called assembler to change the program to *machine language.* Each computer has they own *assembly language* for example, Intel 8086, MC68000 and MC68HC11.

High level language is easier to understand compared with *assembly language* because it is more simple and same like English language. Compiler needed to translate program in *high level language* to *machine language* that can understood by computer. **COBOL (C**ommon **B**usiness **O**riented **L**anguage)' **PASCAL, BASIC (B**eginners All

Purpose Symbolic Instruction Code), FORTRAN (Formula Translation), ADA, C and C++.

2.4 The Drive System

The drive system for the robot will employ the use of a differential drive system. Differential steering is also called skid steering. This type of steering is used on tanks and other vehicles with treads. Wheelchairs are also steered by differential steering.

When a vehicle uses differential steering, it commands two wheels on opposing sides to turn at different speeds. For example, if the right wheel is faster than the left wheel, the vehicle steers to the left. In this section, some of the motion control functions that can be accomplished with differential steering will be examine.

2.4.1 Displacements

A differentially steered vehicle has two displacements. One is linear and the other one is angular. The linear displacement controls how far the vehicle goes, while the angular displacement controls how much the vehicle turns. In this text, D_L for the linear displacement and D_A for the angular displacement will use. D_L is the total number of steps that both motors need to perform. This is not difficult to understand. D_A, on the other hand, is a little more complicated. D_A is the total difference of steps between the two motors. In other words, if forward is positive for both wheels, D_A is the difference between the displacement of the right wheel and the displacement of the left wheel. To keep track of D_L, every time either the right or the left wheel has a step forward, will add one to a counter. If a wheel has a step backward, will subtract one from the counter. To keep track of D_A, assign positive to forward for one motor (will use left-forward), and assign negative to forward for the other motor. This means for each step the left wheel goes forward, it adds one to the counter. For each step the right wheel goes forward, it subtracts one from the counter.

2.4.2 The Velocity

Just as there are two displacements, there are also two velocities. The linear velocity is v_L, whereas the angular velocity is v_A. For stepper motor designs, it is important to derive the velocity of each wheel. This is quite simple. The left motor velocity, using our convention (that left-forward is positive for angular), is $v_L + v_A$. This means the velocity of the right wheel is $v_L - v_A$.

2.4.3 The Acceleration

There are two accelerations, a_L and a_A. Note that for a robot, the maximum a_L is probably different from the maximum a_A. This is because the mass of a robot with respect to linear force available is often different from the angular momentum of a robot with respect to turning torque. As a quick example, increasing the length between the two wheels will increases turning torque without changing forward force. Once again, the acceleration for the left motor is a_L+a_A, whereas the acceleration for the right motor $a_L - a_A$.

2.4.4 The Motors

An electric motor converts electrical energy into mechanical motion. The reverse task, that of converting mechanical motion into electrical energy, is accomplished by a generator or dynamo. In many cases the two devices differ only in their application and minor construction details, and some applications use a single device to fill both roles.

For smaller robots such as line follower mobile robot, there are mostly two types of motors: stepper motors and DC motors. In this robot, stepper motors is the most suitable to use. This section explains how the stepper motors works.

2.4.4.1 Stepper Motor

Stepping motors come in a wide range of angular resolution. The coarsest motors typically turn 90 degrees per step, while high resolution permanent magnet motors are commonly able to handle 1.8 or even 0.72 degrees per step. With an appropriate controller, most permanent magnet and hybrid motors can be run in half-steps, and some controllers can handle smaller fractional steps or micro steps.

For both permanent magnet and variable reluctance stepping motors, if just one winding of the motor is energized, the rotor (under no load) will snap to a fixed angle and then hold that angle until the torque exceeds the holding torque of the motor, at which point, the rotor will turn, trying to hold at each successive equilibrium point.

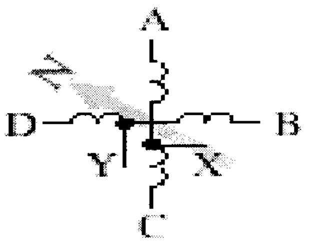


Figure 2.13: Stepper Motor Schematic

In this picture, the light blue "magnet" rotates at the center of the four coils. A, B, C, D, X and Y are electrical nodes that are connected to wires. There are different classes of stepper motors, depending on the configuration of point X and point Y.

- X and Y are both available and independent: most unipolar stepper motors are like this. This is the most general configuration as you can use bipolar or unipolar drive, with or without half stepping.
- X and Y are available but they are connected as one node: less common unipolar stepper motors are like this. This is much less general than the previous one. You can still use bipolar drive, but you cannot half step when the motor is bipolarly driven.
- X and Y are not available: this is the configuration for motors designed for bipolar drive

By switching current through the coils, the freely rotating magnet follows the changing magnetic field and rotates. The magnet is connected to the drive axle of a stepper motor to deliver torque.

In unipolar full-step drive, only one coil is active an any time. In order to turn the magnet clockwise, the following current sequence can be repeated:

1. A→X
2. B→ Y
3. C→X
4. D→Y

The beauty of unipolar drive is that the direction of current through the coils does not change. A controller merely needs to switch current on and off. Switching is easy when current direction does not change.

In bipolar full-step drive, points X and Y are not used. Instead, two coils in series are used at the same time. The current sequence to turn the magnet clockwise is as follows.

1. A→B
2. B→D
3. B→A
4. D→B

In this case, the controller needs to switch on and off as well as the direction of current. We will explain how to do this later. Why would anyone want to switch current direction when there is an easier solution (from the perspective of circuit design)?

It turns out bipolar drive ideally saves 50% energy while delivering the same amount of work. The strength of magnetic field is proportional to both the current and the number of winding of a coil. Let us use the following assumptions:

1. The resistance of each of the four coils is 30Ω
2. The number of winding of each of the four coils is 50
3. The supply voltage is 12V

In unipolar drive, the current is 12/30 = 0.4A, and there is a total of 50 winding for the active coil.

In bipolar drive, the current is 12/(30+30) = 0.2A, and there is a total of 50+50=100 winding. We can see that in bipolar drive, even though the current is dropped by 50% compared to unipolar drive, the number of winding is increased by 100%. The developed magnetic field is equally strong in both cases.

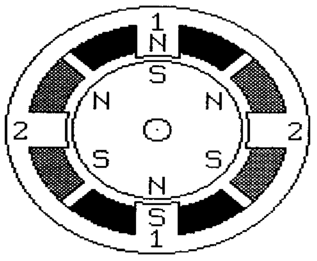


Figure 2.14: The Motor Cross Section

Bipolar permanent magnet and hybrid motors are constructed with exactly the same mechanism as is used on unipolar motors, but the two windings are wired more simply, with no center taps. Thus, the motor itself is simpler but the drive circuitry needed to reverse the polarity of each pair of motor poles is more complex. The motor cross section is shown in Figure 2.14.

To distinguish a bipolar permanent magnet motor from other 4 wire motors, measure the resistances between the different terminals. It is worth noting that some permanent magnet stepping motors have 4 independent windings, organized as two sets of two. Within each set, if the two windings are wired in series, the result can be used as a high voltage bipolar motor. If they are wired in parallel, the result can be used as a low voltage bipolar motor. If they are wired in series with a center tap, the result can be used as a low voltage unipolar motor.

From the perspective of power consumption, unipolar drive consumes $12\times (12/30) = 4.8$W, whereas bipolar drive consumes $12 \times (12/30) +30 = 2.4$W. Bipolar drive saves 50% energy

Stepper motors come in all sizes and resolutions. Even though our conceptual schematic of a stepper motor performs a full rotation in four steps, most steppers motors have higher resolution. Some stepper motors rotate 7.5°per step; others rotate as little as 0.9°per step. Some stepper motors are strong enough to control X-Y tables; others only need to move the head of a CD-player. In general, stepper motors of smaller sizes (1 inch diameter and about 0.75 inch thickness) have larger step sizes and less torque, while larger ones (1.5 inch cubed or so) have finer step sizes and more torque.

2.4.4.2 Stepper Motor Drives

The MC3479 is designed to drive a two-phase stepper motor in the bipolar mode. The circuit consists of four input sections, a logic decoding/sequencing section, two driver-stages for the motor coils, and an output to indicate the Phase A drive state.

The MC3479 integrated circuit is designed to drive a stepper positioning motor in applications such as disk drives and robotics. The outputs can provide up to 350 mA to each of two coils of a two-phase motor. The outputs change state with each low-to-high transition of the clock input, with the new output state depending on the previous state, as well as the input conditions at the logic controls.

The outputs (L1-L4) are high current outputs (see Figure 2.15), which when connected to a two-phase motor, provide two full-bridge configurations. The polarities applied to the motor coils depend on which transistor (QH or QL) of each output is on, which in turn depends on the inputs and the decoding circuitry.