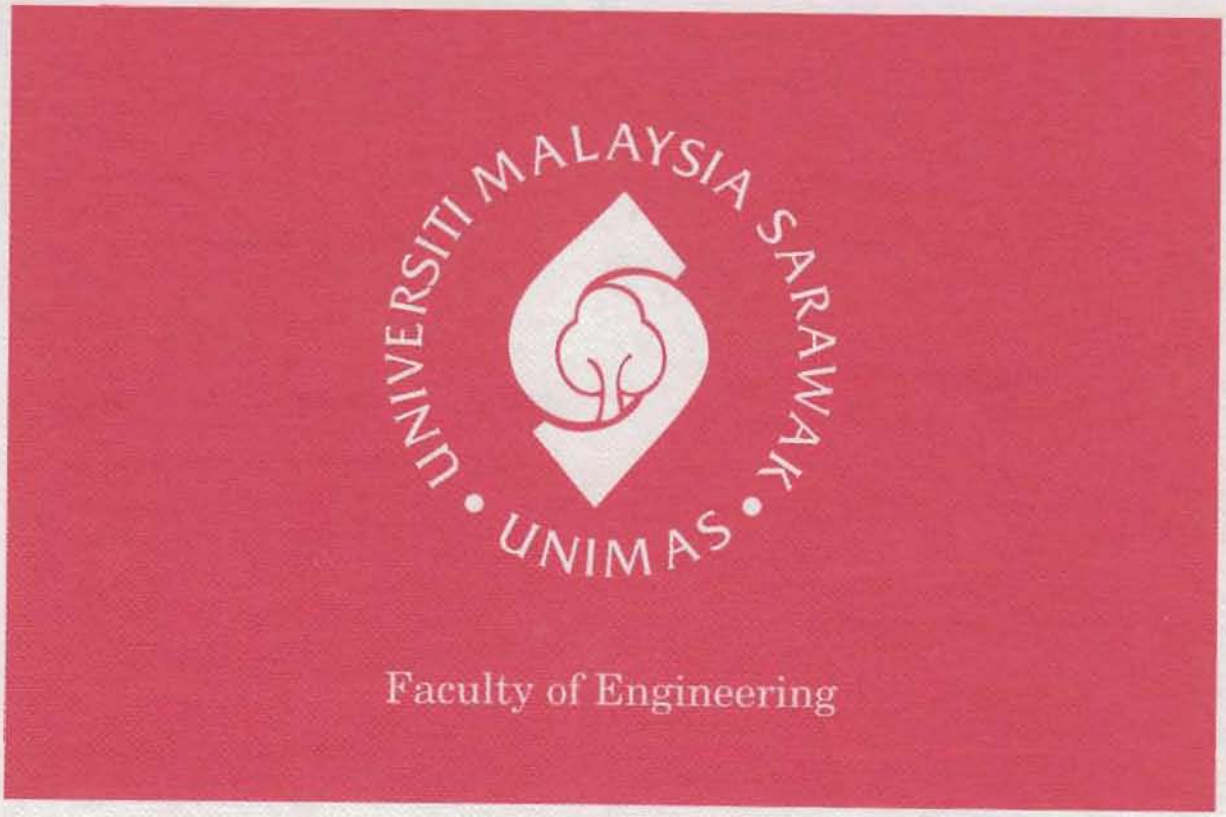


COMPUTERIZE REMOTE CONTROL CAR

Mohd Azudin Bin Mohd Arif

Bachelor of Engineering with Honours
(Electronics and Computer Engineering)
2004

Universiti Malaysia Sarawak

BORANG PENYERAHAN PROJEK TAHUN AKHIR

Judul: **Computerize Remote Control Car**

SESI PENGAJIAN: 2000-2004

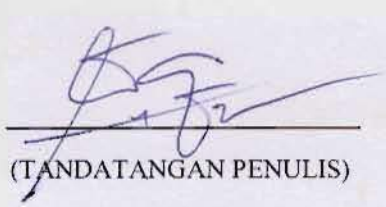
Saya **MOHD AZUDIN BIN MOHD ARIF**

mengaku membenarkan tesis ini disimpan di Pusat Perkhidmatan Akademik, Universiti Malaysia Sarawak dengan syarat-syarat kegunaan seperti berikut:

1. Hakmilik kertas projek ini adalah di bawah nama penulis melainkan penulisan sebagai projek bersama dan dibiayai oleh UNIMAS, hakmiliknya adalah kepunyaan UNIMAS.
2. Naskah salinan di dalam bentuk kertas atau mikro hanya boleh dibuat dengan kebenaran bertulis daripada penulis.
3. Pusat Khidmat Maklumat Akademik, UNIMAS dibenarkan membuat salinan untuk pengajian mereka.
4. Kertas projek hanya boleh diterbitkan dengan kebenaran penulis. Bayaran royalti adalah mengikut kadar yang dipersetujui kelak.
5. * Saya membenarkan/~~tidak membenarkan~~ perpustakaan membuat salinan kertas projek ini sebagai bahan pertukaran di antara institusi pengajian tinggi.
6. ** Sila tandakan (√)

☐	SULIT	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti ynag termaktub di dalam AKTA RAHSIA RASMI 1972).
☐	TERHAD	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan).
☑	TIDAK TERHAD	

Disahkan oleh:



(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

ALAMAT TETAP:
No. 5 Jalan 6 Taman Bukit Kuchai,
47100 Puchong,
Selangor Darul Ehsan.

EN. HUSHAIRI BIN ZEN

(Nama Penyelia)

Tarikh: 7/4/2004

Tarikh: 7/4/04

CATATAN * **Potong yang tidak berkenaan.**

** **Jika Kertas Projek ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyertakan sekali tempoh kertas projek. Ini perlu dikelaskan sebagai SULIT atau TERHAD.**

Approval Page

The project report attached here to, entitled "Computerize Remote Control Car" prepared and submitted by MOHD AZUDIN BIN MOHD ARIF in partial fulfilment of the requirement for Bachelor of Engineering with Honours in Electronics and Computer Engineering is hereby read and approved by:

Mr. Hushairi Bin Zen,
Supervisor

7/4/04
Date

April 2004

COMPUTERIZE REMOTE CONTROL CAR

MOHD AZUDIN BIN MOHD ARIF

This project is submitted in partial fulfilment of
the requirements for the degree of Bachelor of Engineering with Honours
(Electronics and Computer Engineering)

Faculty of Engineering
UNIVERSITI MALAYSIA SARAWAK
2004

To dad, mom, Abang Lan, Kak Lin and Ain

ACKNOWLEDGEMENT

The author would like to express appreciation to Mr. Hushairi Zen as a project first supervisor and Mr. Thaleha Masri as a project second supervisor for all the guidences and advices to complete this project.

Thank you very much to all electronics technicians especially to Mr. Wan Abu Bakar, Mr. Wan Mohd Hamizah, Mr. Zakaria, Mr. Nawawi, Mr. Awangku Amirol and Mr. Mahathir that helping me in the laboratory.

To beloved dad and mom, thank you very much for giving motivation and full moral support to me from the beginning and sacrifice a lot of money to make this project complete.

I also want to thank my friends, Sharuzi, Syafik, Hishamuddin, Zuhairi and Asma for moral support.

ABSTRACT

This project involved in designing a new remote control car system that is interfaced by a microcomputer. Improvements to the design were made in sensing object infront of the car and giving feedback the sensing result to the computer. A microcomputer and a remote control car were the main materials used to develop this system and the other devices were a PIC16F84 microcontroller, an ultrasonic proximity detector and a pair of remote control transmitter and receiver. This project integrated already available devices rather than design the devices from scratch. Process to develop this system involved disassembling, developing, assembling and testing. There were 14 tasks from the process have completed to developed the system. As a result, by using computer a user can controlled the car movements and directions and watched the car sensor result operation. The remote control car system that was controlled by a computer had been achieved. The car capable to sense object infront of it. The car can gave feedback to the computer.

ABSTRAK

Projek ini melibatkan merekabentuk sebuah sistem kereta kawalan jauh yang diantaramuka menggunakan mikrokomputer. Pembaharuan telah dibuat dengan menambah kebolehan mengesan objek di hadapan kereta dan memberi maklumbalas kepada computer hasil pengesanan tersebut. Sebuah mikrokomputer dan sebuah kereta kawalan jauh telah digunakan sebagai alatan utama untuk membina system ini dan peranti yang lain adalah sebuah *microcontroller* PIC16F84, pengesan ultrasonic dan sepasang penghantar dan penerima isyarat kawalan jauh. Projek ini hanya mencantumkan peranti yang telah sedia ada dan tidak merekabentuk daripada asal. Proses membina system ini ialah menceraikan, membina, mencantumkan dan menguji. Terdapat 14 tugas telah disiapkan daripada proses tersebut untuk menyiapkan system ini. Hasilnya, pengguna boleh mengawal gerakan dan arah kerata serta melihat hasil operasi pengesan. Kereta kawalan jauh yang dikawal menggunakan computer telah berjaya dibuat. Kereta tersebut mampu mengesan objek di hadapannya. Kereta itu juga mampu memberi maklumbalas kepada computer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST F ABBREVIATIONS

b	: Base of a bipolar junction transistor label
Bat	: Battery label
c	: Collector of a bipolar junction transistor label
C	: Capacitor label
D	: Decimal numbering system
e	: Emitter of a bipolar junction transistor label
EEPROM	: Electricle Erasable Programble Read Only memory
GUI	: Graphica User Interface
Gnd	: Ground terminal
Hz	: Hertz unit
IC	: Integrated circuit label
ICSP	: In-Circuit Serial Programming
k	: kilo unit prefix
LED	: Light Emitting Diode
m	: mili unit prefix
M	: Mega unit prefix
MFC	: Microsoft Foundation Class
n	: nano unit prefix
OST	: Oscillator Start-up Timer
p	: pico unit prefix
P	: Pins of microcontroller PIC 16F84 label

Pin	: Pins of Parallel Port / DB25 female socket labrl
Pnt	: Points of Data Transmitter label
POR	: Power-on Reset
Pt	: Points of Signal Transmitter label
PWRT	: Power-up Timer
Q	: Transistor label
R	: Resistor label
RAM	: Random Access Memory
S	: Sources of Signal Transmitter label
Sc	: Sources of Data Receiver label
Src	: Sources of Data Transmitter label
URX	: Ultrasonic Receiver
UTX	: Ultrasonic Transmitter
V	: Voltage Unit
+ve	: Positive terminal of Power Supply label
–ve	: Negative terminal of Power Supply label
+Vcc	: Positive terminal of Power Supply label
–Vcc	: Negative terminal of Power Supply label
μ	: micro unit prefix

Chapter 1

INTRODUCTION

Remote control technology has been widely used in many technical areas and the effectiveness has been proven. The simplest implementation of this technology has been applied to toys such as remote control car.

The computer controls external devices through its ports such as Parallel Port. Controlling external devices through any port is called port interfacing. To display output, most programs and applications use GUI (Graphical User Interface) to accommodate Microsoft Windows environment system and it is user-friendly too. Port interfacing and GUI can be created using most programming language.

This project is to combine computer and remote control car system. A program with GUI will be developed to maneuver the remote control car. The remote control transmitter is modified and connected to the parallel port. This car can sense object in front of it and give feedback to the computer.

1.1 OBJECTIVES

The objective of this project can be divided into three main tasks that are :

1) To interface remote control car with the computer.
2) To develop a remote control car that is capable to sense object in front it.
3) To build a remote control car that can give feedback to the remote controller (computer).

1.2 PROJECT OVERVIEW

Computerize remote control car is a combination of computer system and remote control system. This new system communicates in two ways communication where computer act as the car controller and the car provide information to the computer. In this case, the control signals are the car movement and directions whereas the information are the car object detection status.

There are 2 operation modes of the project. Each mode differs in terms of the source and the destination and also input and output.

a) Mode 1 : **Movement and Direction Control.**

In this mode, by using computer, user control the movement and the direction of the car as describe in Figure 1 below.

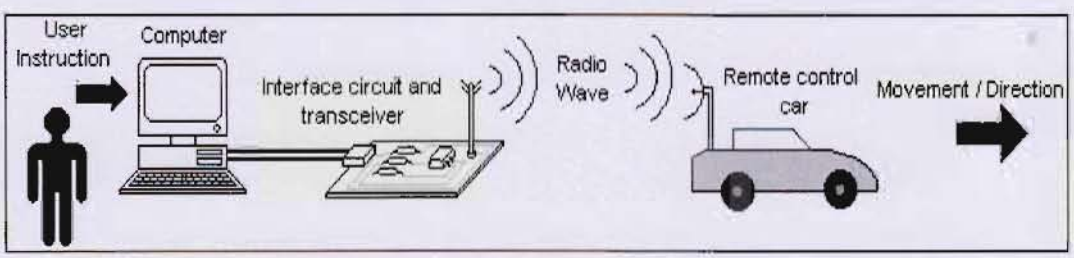


Figure 1.1 : Mode 1, Movement and direction control operation.

b) Mode 2 : **Object detection**

In this mode, the car detect object in front of it and inform the computer as describe in Figure 1.2 below.

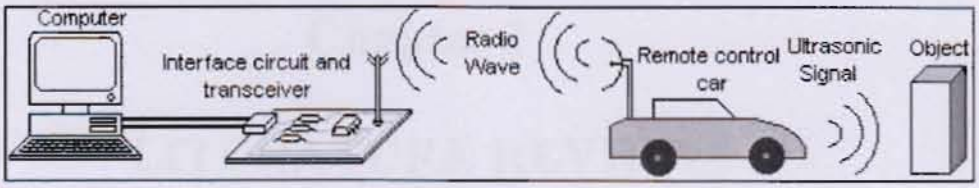


Figure 1.2 : Mode 2, Object detection operation.

1.3 PROJECT SCOPE

This project designs the system rather than the devices. The equipments that were used to developed this project are a computer, a Microsoft Visual C++ program, two remote control car, a PIC16F84 microcontroller, an ultrasonic sensor and typical electronic components such ICs(Integrated Circuit), capacitors and resistors. The project development methodologies are disassembling specific devices, developing each section, assembling all sections and testing the system.

Chapter 2

LITERATURE REVIEW

This topic was concentrate on two major points that are revised projects that has been developed by other people and the other one was to expose all devices that will be used later.

4.1 Previous Project Revision

For all these projects below, the remote control car was modified from any toy product that was available in market. The remote controller was modified and connected to parallel port. The car moving in six directions that are forward, back, forward right, forward left, back right and back left. Each project had their additional characteristics.

4.1.1 Surveillance Car

This RC car was equipped with wireless camera. It was a X10 technologies product. Its' receiver was connected to the USB port. The software to display the video was downloaded from the site http://www.x10.com/. The program was developed using Microsoft Visual C++ 6.0 Figure 2.1 shows the remote controller that was modified and interfaced Parallel Port. Figure 2.2 shows the remote control car that has been modified for this project. Figure 2.3 shows the program GUI that is the display part of the program[1].

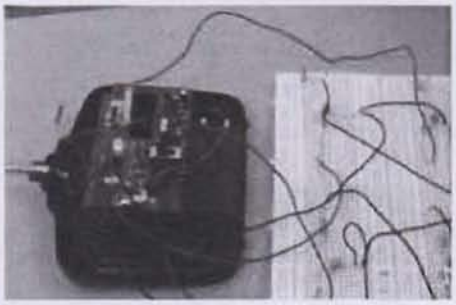


Figure 2.1 : The Surveillance Car modified remote controller[1]

Figure 2.2 : The Surveillance Car with wireless camera[1]

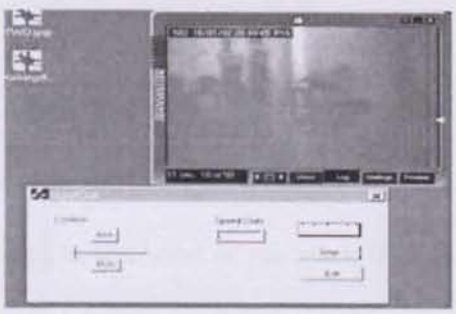


Figure 2.3 : The Surveillance Car program GUI[1]

4.1.2 RoboCar

The car acts as a robot. Two Nikko remote control cars at 40MHz and 27MHz were used. Equipped with magnetic boom arm and able to pick up metallic object. It had artificial mouth made by an array of LED to talk. The system can record macros (a group of actions) that can be played back. Example, move the car from point A to point B and pick up an object then place the car at point A. In the future, we can do the same actions as describe in this example. The program was developed using Visual Basic 6. Figure 2.4 below shows the car and the notebook computer that used to control the car. Figure 2.5 shows the GUI of the program[2].

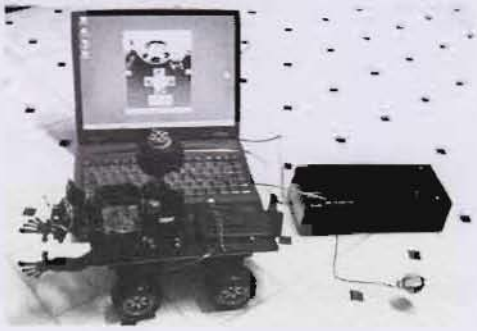


Figure 2.4 : The computer, the receiver and the RoboCar car[2]

Figure 2.5 : The RoboCar program GUI[2]

4.1.3 RC Car Construction and Interface

The interface circuit between remote controller and parallel port was composed of 5v relay. A camera was mounted to the hood of the truck. The car power supply was a 12 Volt Battery, 4.5 Amp Hour Gel-Cell. It will run the camera and transmitter for 21 hours and recharge fully in about an hour. Figure 2.6 shows the car receiver at the bottom of the car[3].

Figure 2.6 : The bottom of the RC car[3]

4.2 Related Information

4.2.1 Computer Program

Computer Software or Program is a component of computer system that makes the computer a multipurpose machine that can be applied in many areas. Computer hardware is quite complicated to manipulate and must strictly follows manufacture specification.. In contrast with hardware, software is quite easy, fast and flexible to design and manipulate it according to user need. A Software or a program is a list of instructions that was written by a programmer. A programmer can make the program that written by him to interact both with user and machine or either one. Examples, a program that interacts with user is GUI program whereas a program that interact with hardware is Parallel Port interface.

a) GUI

GUI (Graphics User Interface) program is a program that was developed in Windows OS (Operating System) environment. GUI program display is in graphic-based and a mouse has been frequently used rather than keyboard to control it. Microsoft has standardized Windows window and there are several components such as button, dialog box, slider, status bar and others that work together to operate the program. Most of the windows operating system functions have been created using C++ programming language and one of the best programming language to create a program in windows-based is Microsoft Visual C++.

b) Microsoft Visual C++ 6.0

Microsoft Visual C++ 6.0 is a type of C++ compiler and a part of Microsoft Visual Studio. Microsoft Visual Studio has provided most development tools to create any GUI program. Beside that, this compiler work closely with MFC classes that create all the parameters and functions of all windows components.

c) MFC

MFC (Microsoft Foundation Class) is a library collection of classes to create GUI program in standard Windows application. For windows programming beginner is advisable to use AppWizard facilities to develop GUI program.

d) AppWizard

AppWizard is a method to create a GUI program without doing it from scratch, where Visual C++ compiler creates basic structures of the program based on MFC classes. By AppWizard, the programmer still has to write the desired application source code but not to type the entire source codes and there are few steps to create the program and to modify it.

Steps to create a program :

i) Select *File* at menu bar and select *New* at *File* pull down menu.

ii) Select *MFCAppWizard (exe)*, insert project name under *Project name:* field and click *OK* button.

iii) Select *Single document*, click to uncheck *Document/View architecture support?* checkbox and click *Finish* button.

As an example, Figure 2.7 shows a GUI that was created using AppWizard. Appendix A shows the full source code of this program.

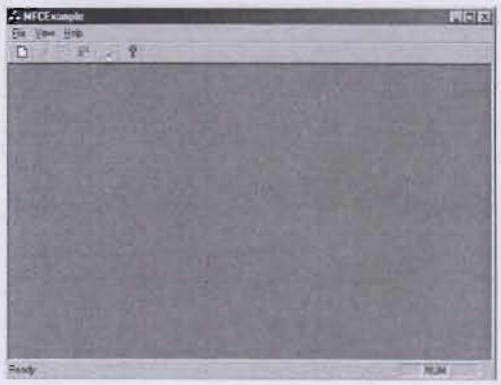


Figure 2.7 : Program MFCExample GUI

4.2.2 Parallel Port Interfacing

Computer has several ports to communicate with external peripherals. Methods to interface each port are different. Parallel Port is a type of computer port.

a) Parallel Port

The term parallel refers to the data that is transmitted through this port. This port transmits 1 byte at a time. It is also capable to receive signals in and sends signals out. Fig. 2.8 shows the Parallel Port socket that is called DB25 socket. Table 2.2 shows pins assignment of the DB25 socket.

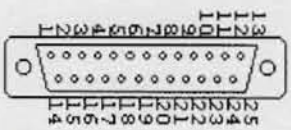


Figure 2.8 : DB25 Socket[4]

Table 2.1 : Pins Configuration of DB25 Socket

PIN	SIGNAL	PIN	SIGNAL
1	nStrobe	14	Auto Feed
2	Data 0	15	nError
3	Data 1	16	nInit
4	Data 2	17	nSelect In
5	Data 3	18	Ground
6	Data 4	19	Ground
7	Data 5	20	Ground
8	Data 6	21	Ground
9	Data 7	22	Ground
10	nAck	23	Ground
11	Busy	24	Ground
12	Paper End	25	Ground
13	Select		

Parallel port was build by combination of 3 buffers that are Data, Status and Control ports. Each buffer has their address, signal direction data that will be read by microprocessor.

1) Data Port :

   Address : 378H

   Direction : Out.

   Data :

Table 2.2 : Data Port Bits

BIT	7	6	5	4	3	2	1	0
SIGNAL	D7	D6	D5	D4	D3	D2	D1	D0

2) Status Port :

Address : 379H

Direction : In

Data :

Table 2.3 : Status Port Bits

BIT	7	6	5	4	3	2	1	0
SIGNAL	Busy	nAck	Paper End	Select	Error	-	-	-

3) Control Port :

Address : 37aH

Direction : Out

Data :

Table 2.4 : Control Port Bits

BIT	7	6	5	4	3	2	1	0
SIGNAL	-	-	-	IRQ	nSelect In	nInit	Auto Feed	nStrobe

Instructions to access any computer port :

1) Input data :

```
result = _inp( 0x0379 );
```

2) Output data :

```
_outp( 0x0378, data );
```

As an example, Figures 2.9 shows a GUI that controls a simple circuit that shown in Figures 2.10.

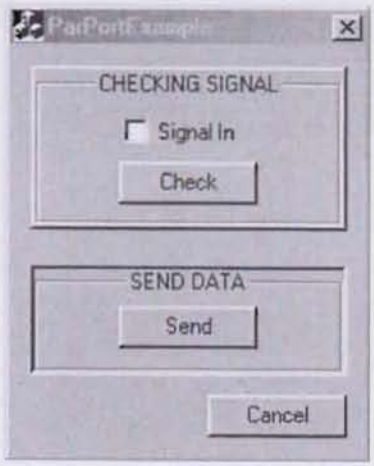


Figure 2.9 : Program ParPortExample GUI

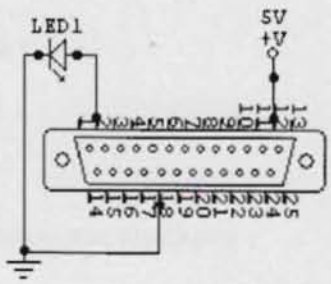


Figure 2.10 : Program ParPortExample schematic diagram

The Program file name is ParPortExample.exe and Appendix shows the full source code of this program. When the button Check was pressed the program any of read Status port, the Signal In checkbox was checked because the Status port received a High state external signal. +5V is a High signal for Parallel Port and Pin 12 is a Pin of Status Port. When the button Send was pressed, the LED was illuminated because the program was sending number 1 to Data Port, which means to make Pin 2 in High State. These are the important parts of the program according to the file name where it is belong to :

i) File : ParPortExampleDlg.h

These are variables used.

```
LINE
 18     Int Signal;
 19     Int StatusIn;


LINE
 24     CButton btnCheck;
 25     CButton btnSend;
```

ii) File : ParPortExampleDlg.cpp

This is additional header file :

```
LINE
  8     #include "conio.h"
```

These are controller for button and checkbox :

```
LINE
 79     DDX_Control(pDX, IDC_CHECK, btnCheck);
 80     DDX_Control(pDX, IDC_SEND, btnSend);
```

These are additional functions and handlers :

```
LINE
179     void CParPortExampleDlg::OnSignalIn()
180     {
181             // TODO: Add your control notification handler code here
182             Signal = _inp( 0x0379 );
183             StatusIn = Signal & 32;
184
185             if( StatusIn == 32)
186                     btnCheck.SetCheck(1);
187     }
188
189     void CParPortExampleDlg::OnCheck()
190     {
191             // TODO: Add your control notification handler code here
192
193     }
194
195     void CParPortExampleDlg::OnSend()
196     {
197             // TODO: Add your control notification handler code here
198             _outp( 0x0378, 1 );
199     }
200
201     void CParPortExampleDlg::OnCancel()
202     {
203             // TODO: Add extra cleanup here
204             _outp( 0x0378, 0 );
205             CDialog::OnCancel();
206     }
```

4.2.3 Remote Control Transmitter

This transmitter was the remote controller and its' task was to control car direction. This device was taken from a remote controller of a car toy.

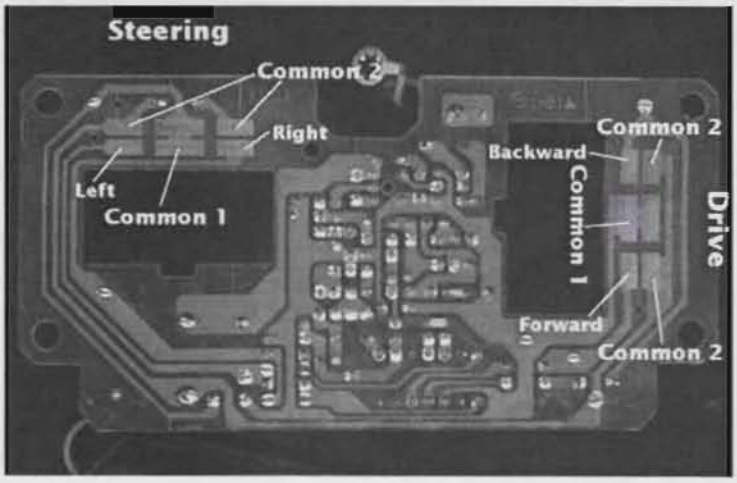


Figure 2.11 : Remote controller[3]

Figure 2.11 above, for the remote controller explanation. There for significant point that are Left, Right, Backward and Forward. To interface this device four lines must be connected to all these point. High voltage at that point will indicate the car direction. Other parts of this device remain the same.

4.2.4 Signals Transmitter and Receiver

These transmitter and receiver were to transmit and receive information signals. A pair of remote control transmitter and receiver of remote control car can be used for this communication system.

4.2.5 Microcontroller :

Microcontroller is a type of processor that integrates varies kind of digital circuits in a single IC. Microcontroller has its internal ROM and RAM.

PIC16F84A microcontroller is belonging to PIC16CXXX family[5]. It is the most popular of all PIC microcontroller. This Microcontroller is a RISC (Reduce Instruction Set Computer) type system. Another advantage of this Microcontroller is it can be program by PIC BASIC compiler which is quite similar to high-level language rather than assembly language. This microcontroller features[6] :

i) 1024 Flash EEPROM program memory.

ii) 14 bit wide instruction.

iii) 64 bytes data EEPROM.

iv) 68 bytes RAM.

v) 8 bit wide data bytes.

vi) 20 MHz clock input.

vii) 200 ns instruction cycle.

viii) 15 special function register.

ix) 8 level stack.

x) 4 interrupt sources.

xi) Sleep Mode and 1 Watchdog Timer.

xii) POR (Power-on Reset), PWRT (Power-up Timer) and OST (Oscillator Start-up Timer).

xiii) ICSP (In-Circuit Serial Programming).

xiv) 35 single word instruction.

Pin Configuration :

Figure 2.12 : PIC 16F84A pins configuration[6]

As an example, Figure 2.13 shows a simple organ circuit. The input was 8 push-button connected to Port B. The output was a speaker a speaker connected to Port A.

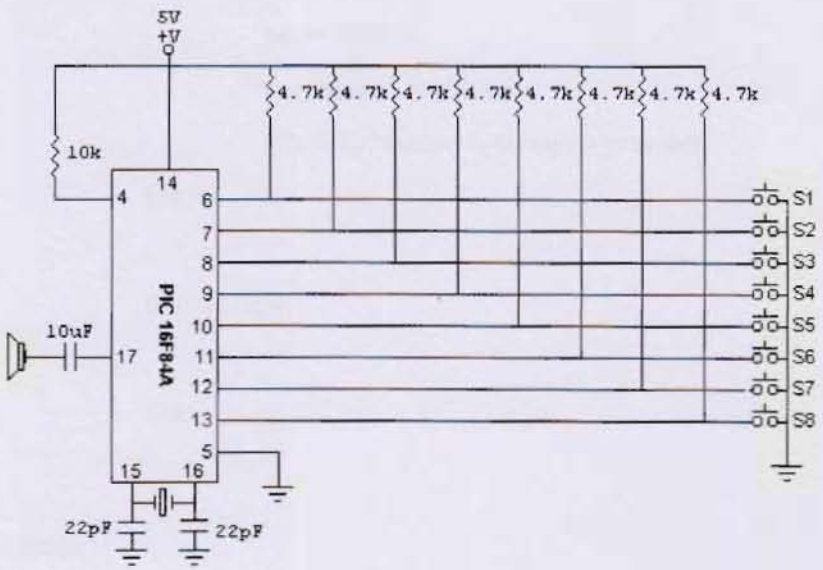


Figure 2.13 : Simple organ schematic diagram[5]

Source Code :

```
LINE                    SOURCE

1       buzzer var PORTA.0
2       notes var word[0]
3       key var byte
4       key_pressed var byte
5
6       notes[1] = 262 : notes[2] = 294 : notes[3] = 330 : notes[4] = 349
7       notes[5] = 392 : notes[6] = 440 : notes[7] = 494 : notes[8] = 524
8
9               TRISB = %11111111
10              TRISA = 0
11
12      loop :
13              IF PORTB <> 255 THEN
14                      key = ~PORTB
15                      key_pressed = NCB key
16                      FREQOUT buzzer, 5, notes[key_pressed]
17              ENDIF
18
19              GOTO LOOP
20
21              END
```

4.2.6 Sensor Circuit

The tasks of sensor circuit are to detect any object in front of the car and tell the result to the computer. To carry out these tasks, proximity detector is the best solution.

a) Proximity Detector

Proximity detector is an electronic transducer that is use to detect any object that is placed near to it. This device operation principle:

i) Device sends a signal in a waveform to a point near to it.

ii) Any object that being hit by the signal will reflect the signal back to the device.

iii) Device will detect the signal and indicate there is an object near to it.

To achieve this operation principle, this device must be build by a pair of transmitter and receiver of certain type of wave. Light, lasers, infrared and ultrasonic are common waves that are use by this device as a signal. Figure 2.14 shows the operation of Proximity Detector.

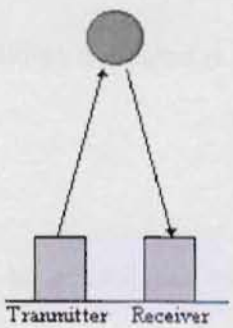


Figure 2.14 : Proximity Detector operation

b) Ultrasonic Proximity Detector

Ultrasonic Proximity Detector uses a high frequency of sound wave as a signal. The highest human voice frequency is 5 kHz and the highest stereo audio frequency is 15 kHz but ultrasonic signal has a higher frequency than both frequencies such as 40kHz.

This is an observation of an ultrasonic module of Application Module DT35 that is developed by L.J. ELECTRONICS[7].

c) Ultrasonic Module:

This module can be divided into 3 sections:

i) Transmitter :

The transmitter, UTX consists of an oscillator that generates a 40 kHz sound wave. To switch on the transmitter, Port PB6 must set to high (+5v).

ii) Receiver :

The receiver, URX is a circuit that had been designed to detect a 40 kHz sound wave. If it detects a 40 kHz ultrasonic signal, a TTL level square wave is produced at the output PB7. When no signal is detected, URX will be at Logic 1 (+5v).

iii) Gain controller:

Gain controller is use to control the sensitivity of the receiver. This controller makes the receiver to detect only an object that produces a stronger reflection within a specific range. To do that, the GAIN controller can be turn down and up. To use the ultrasonic transmitter and receiver as a proximity detector, switch on the transmitter to generate a pulse of ultrasound and then monitor URX (PB7) for a set period. If a 40 kHz waveform ids detected during this period, an alarm can be generated.

As an example of a program to run an Ultrasonic Module as a Proximity Detector[8]. When an object is placed directly above the Ultrasonic Unit, all of the Port 2 monitor LEDs were illuminated.

```
LINE    SOURCE      CODE                COMMENTS

1                   ORG   0200H         ; Defines the start address for
2                                       ; object code as 0080:0200H
3       UMODEREG    EQU   086H          ; Define address of Mode
4                                       ; Register
5       UPORT1CTL   EQU   088H          ; Defines address of Port
6                                       ; Control Register
7       UPORT1      EQU   090H          ; Defines address of Port 1
8       UPORT2      EQU   092H          ; Defines address of Port 2
9                   MOV   AL, 40H       ; Configures Port 1, bit 6 ( P16 )
10                  OUT   UPORT1CTL, AL ; as an output and bit 7 ( P17 )
11                                      ; as an input
12                  MOV   AL, 03H       ; Configures Port 2, as all
13                  OUT   UMODEREG, AL  ; outputs
14                  MOV   AL, 40H       ; Outputs a logic "1" on P16
15                  OUT   UPORT1, AL    ; ( UTX ) to switch on Ultrasound
16      DETECT:     IN    AL, UPORT1    ; Tests P17 ( URX ) for Ultrasound
17                  TEST  AL, 80H       ; Received
18                  JZ    ALARM
19                  MOV   AL, 00H       ; No Ultrasound detected so
20                  OUT   UPORT2, AL    ; output 00H at Port 2
21                  JMP   DETECT        ; Jump back to check for
22                                      ; Ultrasound again
23      ALARM:      MOV   AL, 0FFH      ; Ultrasound detected so
```

```
24                OUT   UPORT2, AL      ; output 00H at Port 2
25                JMP   DETECT          ; Jump back to check for
26                                      ; Ultrasound again
```

Chapter 3

METHODOLOGY

3.1 DIVISIONS AND SECTIONS

This project was divided into 2 divisions and 6 sections as describe in Figure 3.1 below. This arrangement was made according to physical aspect of all the devices where they were suppose to be attached logically.

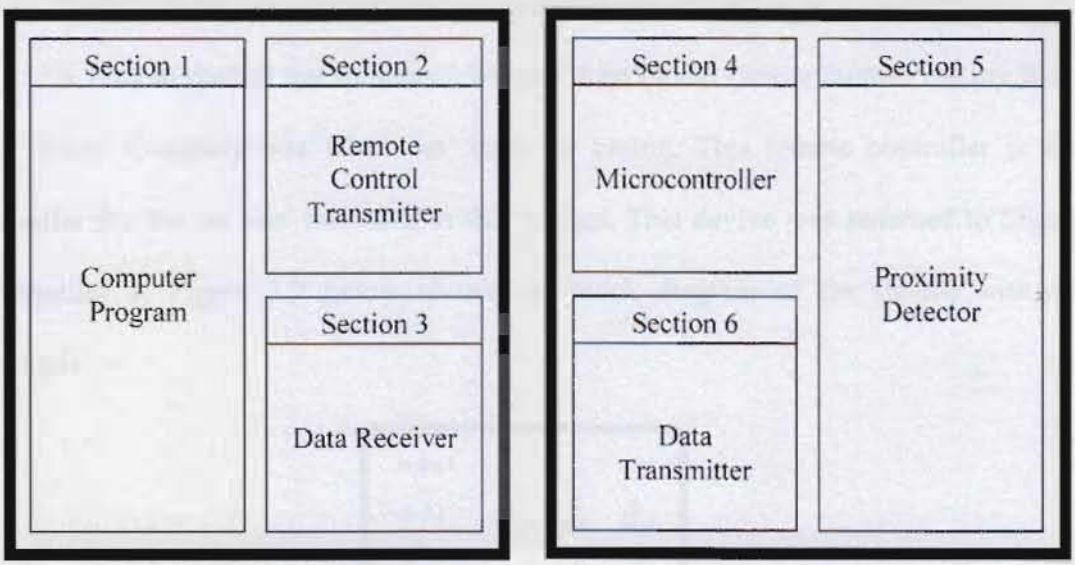


Figure 3.1 : Divisions and Sections Block Diagram

3.1.1 Division 1, Computer :

1) Section 1 : Computer Program

2) Section 2 : Remote Control Transmitter

3) Section 3 : Data Receiver

3.1.2 Division 2, Remote Control Car :

1) Section 4 : Microcontroller

2) Section 5 : Proximity Detector

3) Section 6 : Data Transmitter

3.2 DISASSEMBLING

This duty involves disassembled already made devices to form the system of the project.

3.2.1 Task 1 : A remote control transmitter was disassembled

A remote control transmitter of Wonder Arm model remote control car toy from Toy House Company was taken out from the casing. This remote controller is the controller for the car that was used in this project. This devive was renamed to Signal Transmitter 1. Figure 3.2 below shows the block diagram of the device with all terminals.

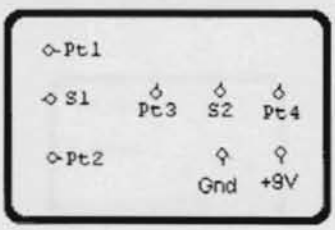


Figure 3.2 : Signal Transmitter 1 Block Diagram

3.2.2 Task 2 : A remote control transmitter was disassembled

A remote control transmitter of Cobra Car model of remote control car toy fron Jinhuan Company was taken out from the casing and was labelled as Signal Transmitter 2. Figure 3.3 shows the block diagram of Signal Transmitter 2.

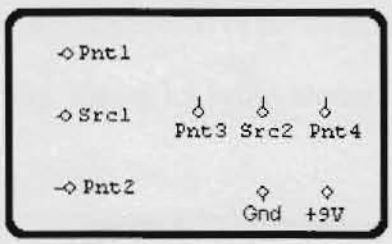


Figure 3.3 : Signal Transmitter 2 Block Diagram

3.2.3 Task 3 : A remote control receiver was disassambled

A remote control car reciver for the the same toy that has been described above in Task 2 was disassembled and the new name for this devive is Signal Receiver 2. Figure 3.4 below shows the block diagram of this device.

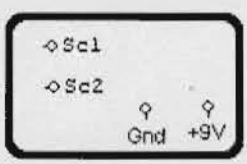


Figure 3.4 : Signal Receiver 2 Diagram

3.3 DEVELOPING

3.3.1 Task 4 : Computer Program was developed

The Computer program is a combinations of inputs and outputs terminals for the user to run both operations of this project. The program file name is Computerize Remote Control Car.exe. The two subsections of the Computer Program are the GUI and the Parallel Port Interfacing. Figure 3.5 below shows the Computer Program Block Diagram.

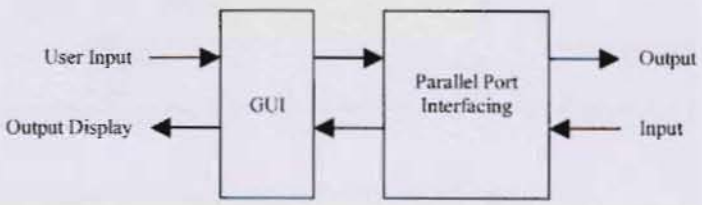


Figure 3.5: Computer Program Block Diagram

GUI is a Windows Dialogue type display. Parallel Port Interfacing is set of instructions that control the Parallel Port to transmit and receive specific data. List of GUI components:

1) Five push buttons
2) A Movements and Directions display
3) An Object Detection display

List of Parallel Port Interfacing instructions:

1) Five _outpw(0x0378, number); intructions
2) A inStatusP = _inpw(0x0379); intruction

Figure 3.6 below shows the GUI of the Computer Program. The program was created using Programming Language Microsoft Visual C++ 6.0 Enterprise Edition. The functions and classes to created this Computer Program were taken from MFC.

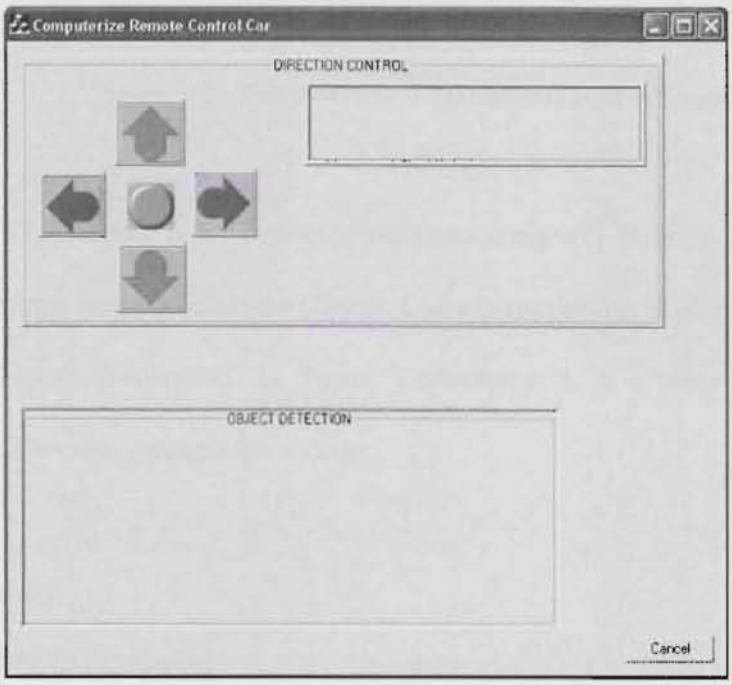


Figure 3.6 : The Computer Program GUI

Appendix C shows the Flow Charts of both operation of Computer Program and the source code of the Computer Program.

3.3.2 Task 5 : Remote Control Transmitter was configured

Remote Control Transmitter is a device that transmits movements and directions control signals from Division 1(computer) to Division 2(car). The three

subsections of Remote Control Transmitter are the Socket, the Interface Circuit 1 and the Signal Transmitter 1 as shown in Figure 3.7.

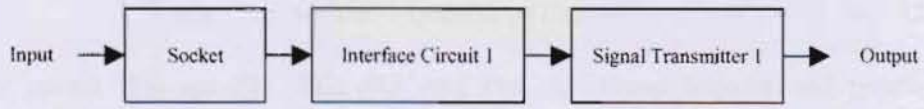


Figure 3.7 : Remote Control Transmitter Block Diagram

The Socket is a connector between computer Parallel Port to Remote Control Transmitter Device. Interface Circuit 1 is a combination of electrical-controlled switches for Signal Transmitter 1. Signal Transmitter 1 is a device that converts electrical signals to electromagnetic waves.

List of components:

1) A DB25 FEMALE Socket
2) Four 2N2222A NPN transistors
3) A Signal Transmitter

The connector is a DB25 Female Socket. It consist of two parts that are the Frontside with holes and Backside with pins. Each hole is connected to a respective pin. There are 25 holes and pins. Each one is numbered and is arrangged in sequence. The Frontside is connected to Parallel Port using a Parallel Port Male to Male Cable. The Backside is connected to the Interface Circuit 1 using copper wires that are soldered.

Interface Circuit 1 consist of four 2N2222A NPN transistors. For each transistor, the base b is connected to a pin of socket. The collector c and the emitter e are connected to the Signal Transmitter 1.

Signal Transmitter 1 consist of two sources that are S1 and S2 and also four points that are Pt1, Pt2, Pt3 and Pt4. All these sources and points are connected to the transistors of Interface Circuit 1. Figure 3.8 below shows the schematic diagram of Remote Control Transmitter connections.

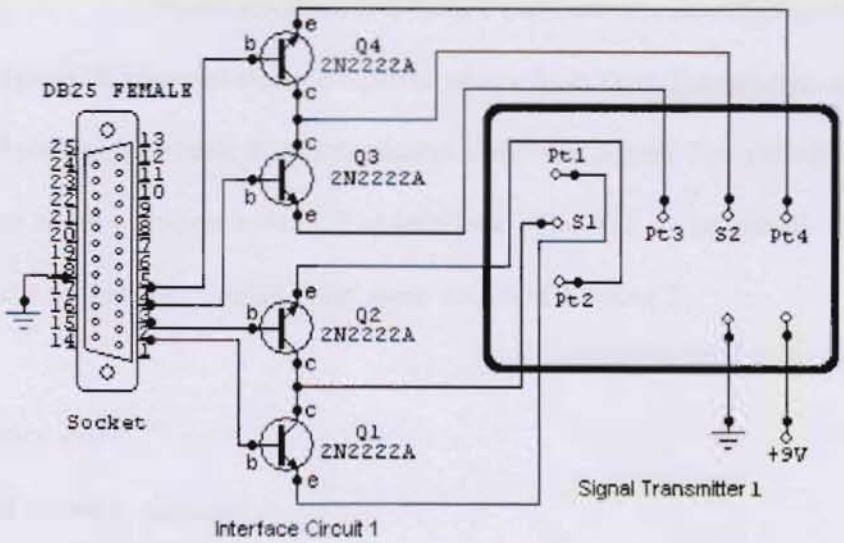


Figure 3.8 : Remote Control Transmitter Schematic Diagram

Appendix B shows the operation Flow Charts of Computer of Remote Control Transmitter.

3.3.3 Task 6 : Data Receiver was configured

Data Receiver is a device that receives Object Detection Signals for Division 1(computer) from Division 2(car). There are three subsections of Data Receiver

that are the Signal Receiver 2, the Interface Circuit 2 and the Socket as shown in Figure 3.10.

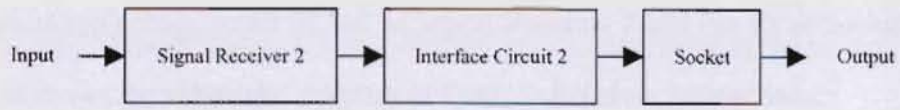


Figure 3.10 : Data Receiver Block Diagram

Signal Receiver is a device that converts electromagnetic waves to electrical signals. It receives electromagnetic waves from Data Transmitter. It is labelled as Signal Receiver 2 because it communicates only with Signal Transmitter 2 and work together as a communication system. The Interface Circuit 2 is an electrical-controlled switch for the socket. The Socket is the same socket of Section 2.

List of components :

1) A 330nF ceramic capacitor
2) A 100nF ceramic capacitor
3) A 78L05 voltage regulator
4) A 2N2222A NPN transistor
5) A Signal Receiver 2

Signal Receiver 2 consist of two sources that are Sc1 and Sc2. Both sources are connected to the transistor Q5 of Interface Circuit 2.

Interface Circuit 2 consist of a 2N2222A NPN transistors and a voltage regulator circuit. For the transistor, the base b is connected to the voltage terminal Sc1 of Signal Receiver 1. The collector c is connected to voltage regulator circuit and the emitter e is connected to voltage terminal Sc2 of Signal Receiver 2 and Pin 12 of Socket. Figure 3.11 below shows the schematic diagram of Data Receiver connections.

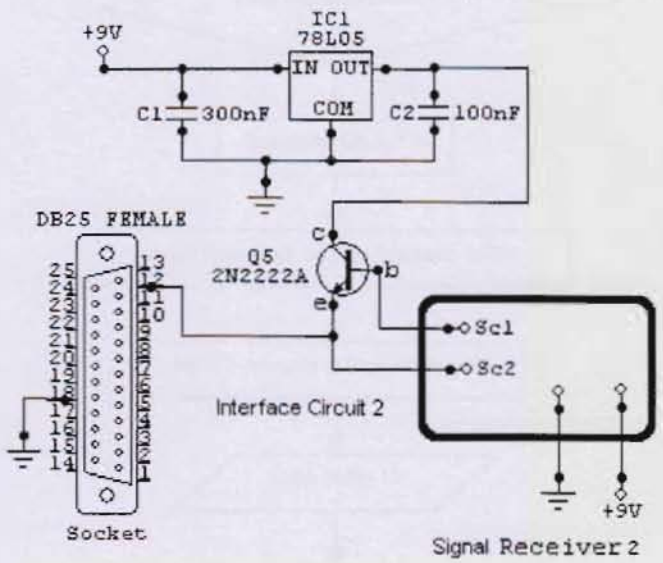


Figure 3.11 : Data Receiver Schematic Diagram

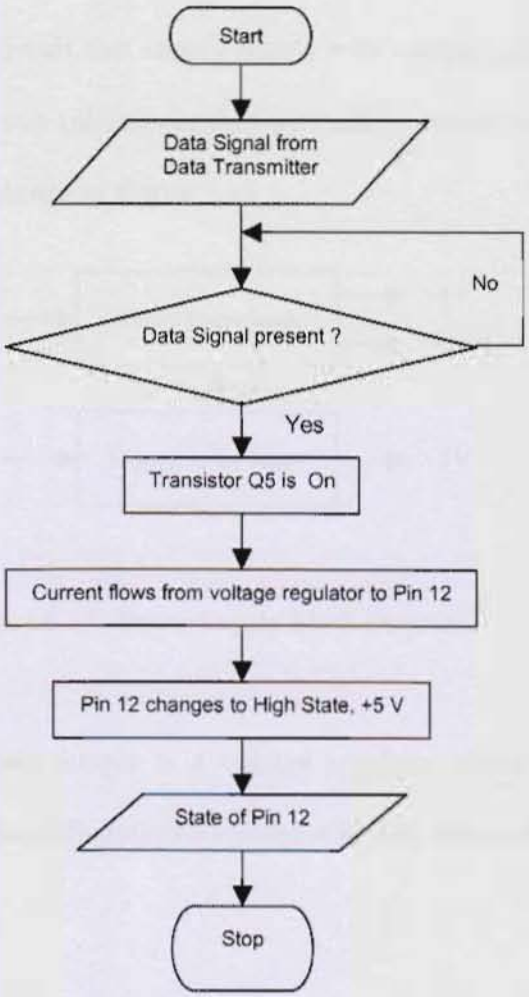


Figure 3.12 : Data Receiver Operation Flow Chart

Figure 3.12 shows the Flow Chart of Data Receiver Operation. The input parameter is the Data Signals from Data Transmitter. And the output is the Pin 12 state of the Socket.

Firstly, the Signal Receiver 2 is waiting the Data Signals. If the device receives the Data Signals, source Sc1 flows current to transistor Q5. As a result the transistor on and allows current flows from voltage regulator circuit to Pin 12. If no Data Signals present, the Signal Receiver will stays in waiting condition.

3.3.4 Task 7 : Power Supply was developed

Power Supply is a circuit that supply supply +5V voltage and -5V voltage. The Power Supply is formed by two subsections that are Positive Power Supply and the Negative Power Supply as shown in Figure 3.13.

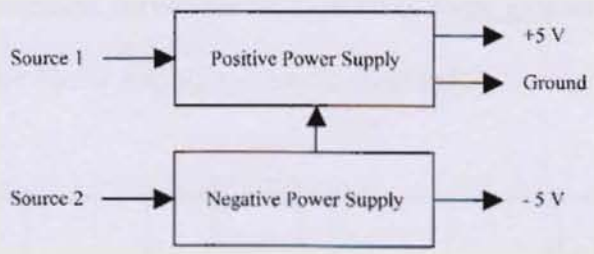


Figure 3.13 : Power Supply Block Diagram

Positive Power Supply is a voltage regulator circuit that convert +9V voltage to +5V. Negative Power Supply is a voltage regulator circuit that convert –9V voltage to –5V.

List of components :

1) Two 9V batteries
2) Two SPDT switches
3) Four 100nF ceramic capacitors
4) Two 1000μF electrolytic capacitors
5) Two 47μF electrolytic capacitors
6) A 7805 voltage regulator
7) A 7905 voltage regulator
8) Two red LED
9) Two 470Ω resistors

Positive Power Supply consist of a voltage regulator 7805 and the common is connected to negative terminal of battery Bat2. Negative Power Supply consist of a voltage regulator 7905 and the common is connected to positve terminal of battery Bat3. Both common are connected together to form ground terminal Gnd. Figure 3.14 below shows the Power Supply schematic diagram[1].

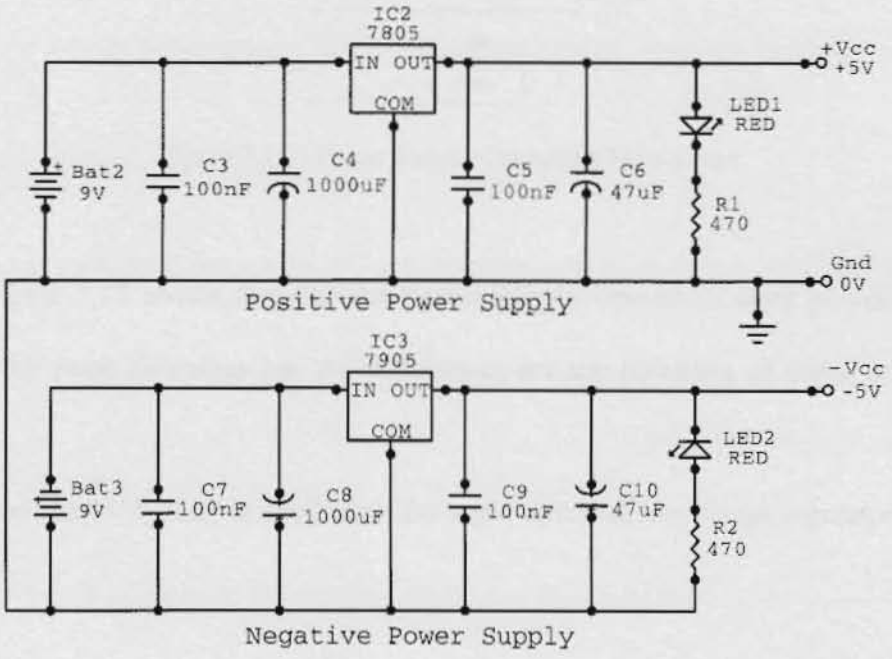


Figure 3.14 : Power Supply Schematic Diagram

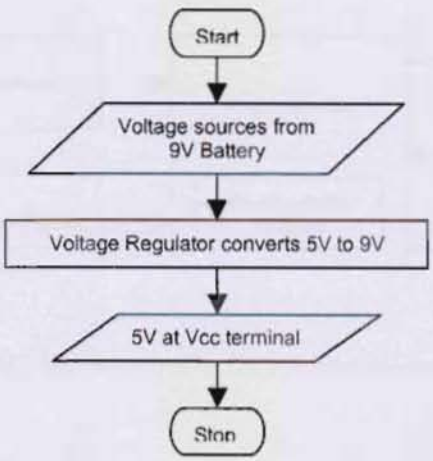


Figure 3.15 : Power Supply Operation Flow Chart

Figure 3.15 above describe the Power Supply Operation. Both power supplies perform the same functions but the differences are the polarities of input and output voltages.

When the 9V battery is applied to the input terminal, the volage regulator changes it into 5V.

3.3.5 Task 8 : Microcontroller was setup

Microcontroller circuit is a circuit that control the Proximity Detector and the Data Transmitter. The Microcontroller circuit consist of hardware and software. It is divided into 4 subsections that are Reset circuit, Oscillator circuit, the Microcontroller and the Microcontroller Program. Figure 3.16 shows the Microcontroller circuit block diagram.

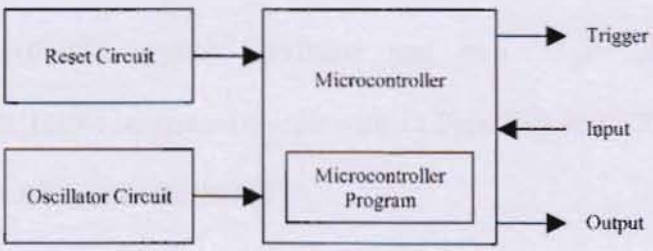


Figure 3.16: Microcontroller Circuit Block Diagram

Reset circuit enable the microcontroller for self-resetting every time when power is applied to the microcontroller voltage source terminal V_{DD} to run the microcontroller. Oscillator circuit provides 10Mhz frequency clock to the *microcontroller. The Reset circuit and the Oscillator circuit are compulsory circuits to* operate the microcontroller. Both circuit do not contribute directly to both operations of this project. The microcontroller receives one input signal and transmit two output signals. The Microcontroller Program is a set of instructions to operate the microcontroller.

List of components :

1) A 1N4001 general purpose diode
2) A 470Ω resistors
3) A 10kΩ resistors
4) A 100nF ceramic capacitor
5) Two 33pF ceramic capacitors
6) A 10MHz crystal oscillator
7) A PIC16F84 microcontroller

Reset circuit consist of a diode, resistors and a capacitor. Oscillator circuit consist of a 10MHz crystal oscillator and two 33pF capacitors. The microcontroller is a PIC16F84 integrated circuit with 18 Pins. Figure 3.17 shows the microcontroller circuit schematic diagram[5].

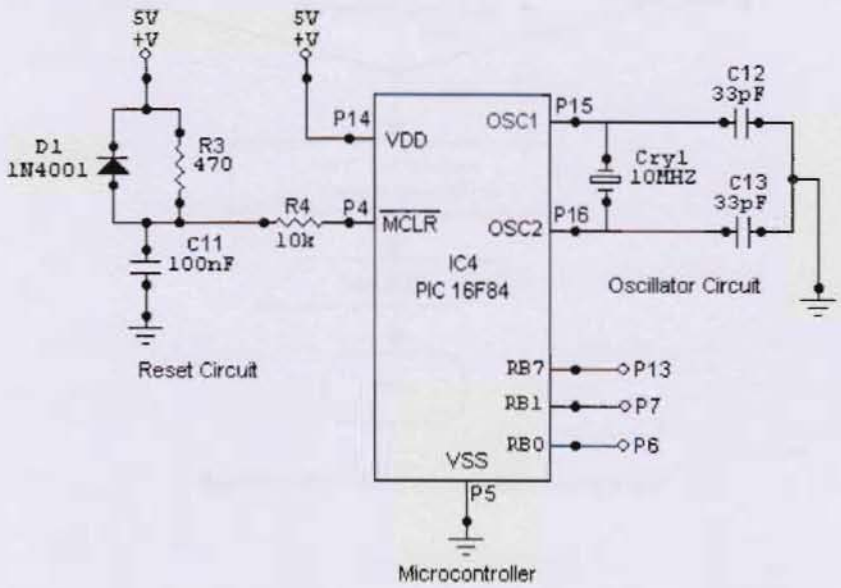


Figure 3.17 : Microcontroller Circuit Schematic Diagram

The Microcontroller Program is hexadecimal machine code that is stored inside the Microcontroller EEPROM. The program name is MicroProg.hex and Appendix D list down the all the Microcontroller Programs that is written originally in BASIC source code MicroProg.bas, the source code that has been converted into assembly code MicroProg.asm and the same promram in hexadecimal code that is Program1.hex.

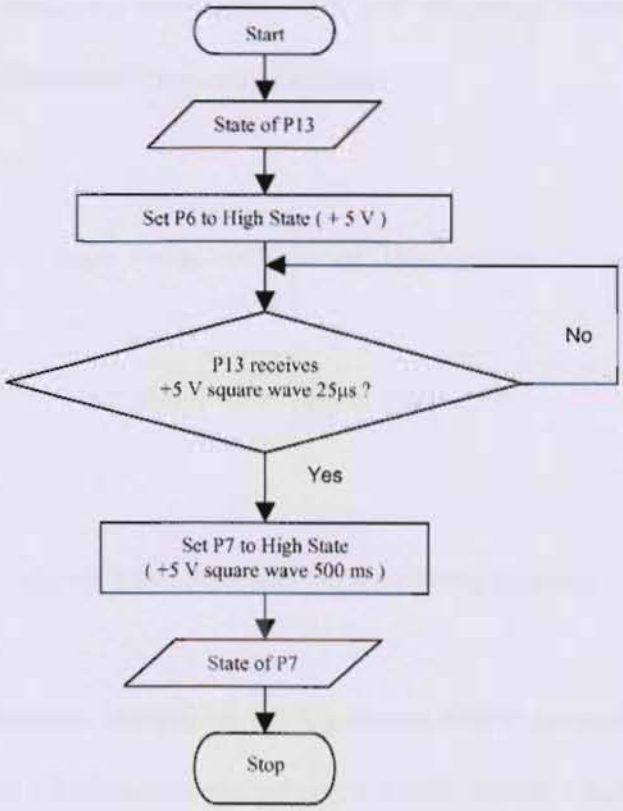


Figure 3.18 : Microcontroller Flow Chart

Figure 3.18 shows the Microcontroller Operations. When pin P6 is in High state, it will switch on and trigger the ultrasonic transmitter UTX at Proximity Detector. If P13 receives +5V square waves 25μs, it will transmit +5 V square wave 500 ms to Data Transmitter through pin P7. Otherwise, Microcontroller stays in waiting condition.

3.3.6 Task 9 : Proximity Detector was developed

Proximity Detector is a device that detect object infront of it. This device operates using an ultrasonic wave principle. The Proximity Detector is divided into two

subsections that are ultrasonic transmitter UTX and ultrasonic receiver URX. Figure 3.19 shows the block diagram of Proximity Detector.

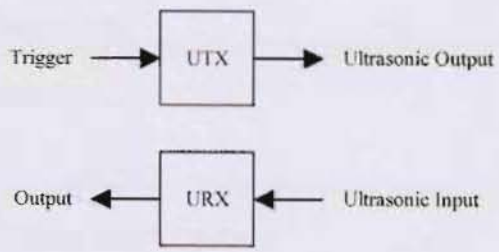


Figure 3.19 : Proximity Detector Block diagram

The ultrasonic transmitter UTX generate 40kHz ultrasonic square waves. The ultrasonic receiver URX receive the reflected 40kHz square waves from ultrasonic transmitter UTX.

List of components :

1) Two 100Ω resistors
2) A 390Ω resistor
3) Two 1kΩ resistors
4) Three 10kΩ resistors
5) Two 22kΩ resistors
6) A 33kΩ resistor
7) A 56kΩ resistor
8) A 1MΩ resistor
9) A 50 kΩ variable resistor
10) A 100 kΩ variable resistor

11) A 220pF ceramic capacitor

12) A 390pF ceramic capacitor

13) A 10nF ceramic capacitor

14) Five 100nF ceramic capacitors

15) A 1μF ceramic capacitor

16) A 555 timer

17) A TL074 Quad Low Noise CMOS OpAmp

18) A LM311

19) An Ultrasonic Transmitter module

20) An Ultrasonic Receiver module

The ultrasonic transmitter UTX is consist of an oscillator circuit that generates 40kHz square waves and an ultrasonic transmitter module that converts electrical signals into ultrasonic waves. The ultrasonic receiver URX is consist of an ultrasonic receiver module that sense the reflected 40kHz square waves from ultrasonic transmitter UTX and converts it into electrical signals, an amplifier to amplify the elecrical signasl and also a comparator that generates square waves. Figure 3.20 below shows the Proximity Detector connections[3].

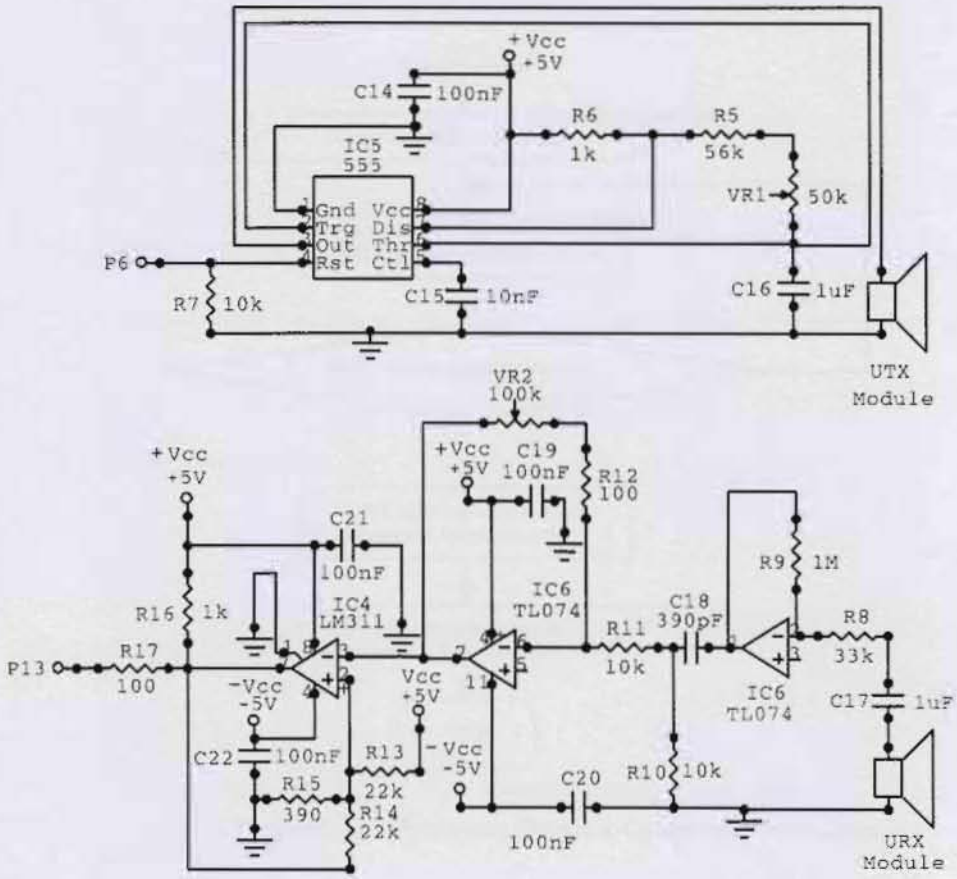


Figure 3.20 : Proximity Detector Schematic Diagram[7]

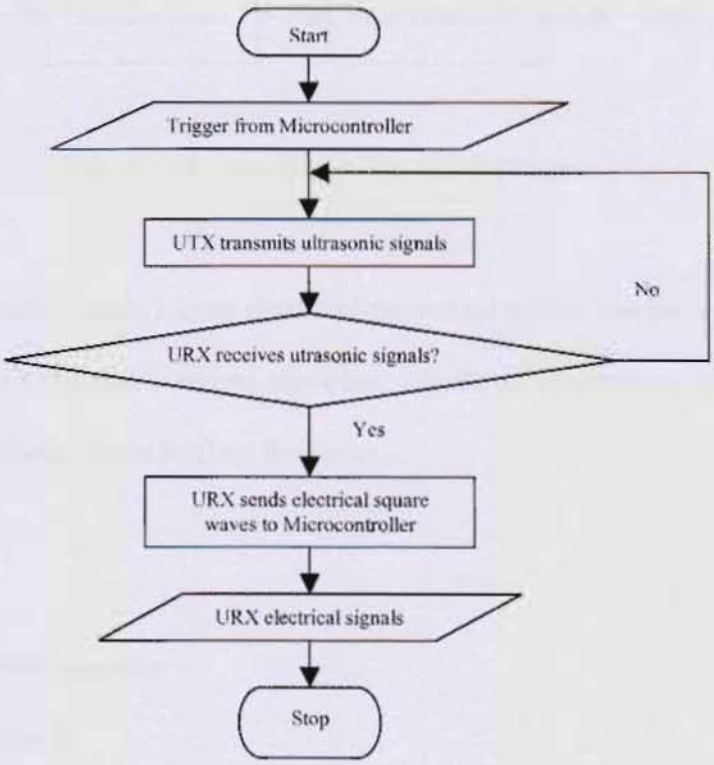


Figure 3.18 : Proximity Detector Operation Flow Chart

Figures 3.21 shows the operation of Proximity Detector. The ultrasonic transmitter UTX transmit ultrasonic waves after receive trigger signal from microcontroller. If there is an object infront of the this device the waves will be reflected back to the ultrasonic receiver URX. Ultrasonic Receiver URX sense the waves and generates +5V square waves 25μs to the Microcontroller.

3.3.7 Task 10 : Data Transmitter was configured

Data Transmitter is a device that transmits Object Detection Signals from *Division 2(car) to Division 1(computer). The two subsections of Data Trnansmitter* are the Interface Circuit 3 and the Signal Transmitter 2 as shown in Figure 3.22.

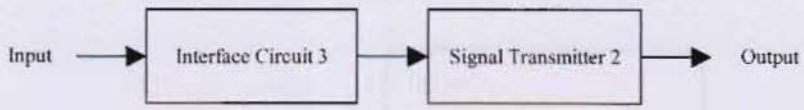


Figure 3.22: Data Transmitter Block Diagram

Interface Circuit 3 is an electrical-controlled switch for the socket. Signal Transmitter 2 is a device that converts electrical signals to electromagnetic waves. It transmits electromagnetic waves to Data Receiver.

List of components :

1) A 2N2222A NPN transistor
2) A Signal Receiver 2

Interface Circuit 3 consist of a 2N2222A NPN transistors. the base b is connected to the pin P13 of the Microcontroller. The collector c and the emitter e are connected to the Signal Transmitter 2.

Signal Transmitter 2 consist of two sources that are Src1 and Src2 and also four points that are Pnt1, Pnt2, Pnt3 and Pnt4. Only the source Src1 and the point Pnt1 are connected to the Interface Circuit 3. Figure 3.23 below shows the schematic diagram of Data Transmitter connections.

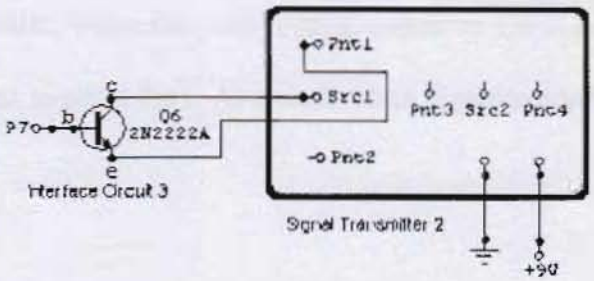


Figure 3.23 : Data Transmitter Schematic Diagram

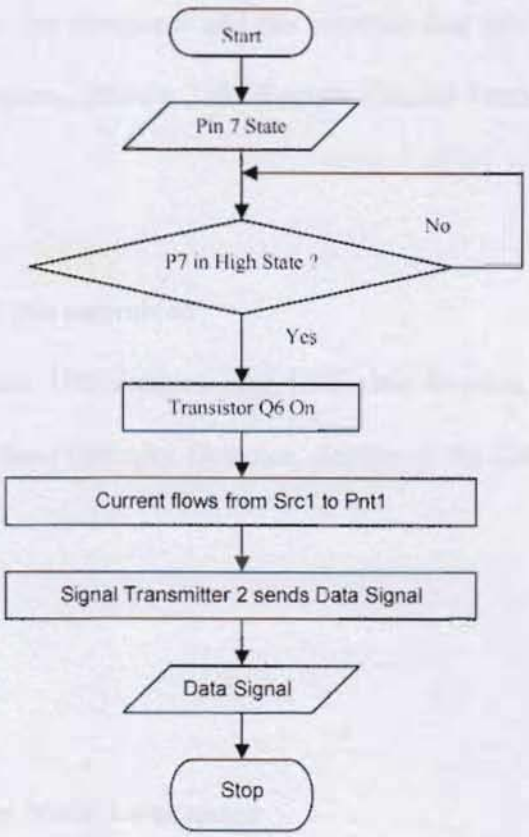


Figure 3.24 : Data Transmitter Operation Flow Chart

Figure 3.24 illustrates the operation of Data Transmitter. The input is the state of pin P7 of microcontroller. When the state is high, transistor Q6 is on and allows current flows from sourse Src1 to point Pnt1. As a result Data Signal sends Data Signals to Data Receiver.

3.4 ASSEMBLING

3.4.1 Task 11 : Division 1 was assembled

Division 1 is the computer and the sections that are atthached to it are Section 1 the Computer Program, Section 2 the Remote Control Transmitter and Section 3 Data Receiver.

3.4.2 Task 12 : Division 2 was assembled

Division 2 is the car. The sections that form this division are Section 4 the Microcontroller, Section 5 the Proximity Detector, Section 6 the Data Transmitter and the Power Supply.

3.5 TESTING

3.5.1 Task 13 : Operation Mode 1 was tested

In this operation mode, the inputs are the buttons that are pressed by the user at the GUI of Computer Program and the output are the movements and directioms of the car. To test this operation mode, the user pressed all the buttons at the GUI of the Computer Program at a time for each button and the car movements and directions that

caused by the specific button pressed was indicated. The sequence of button pressed are the Forward button, Backward button, Right button, Left button and Stop button.

3.5.2 Task 14 : Test Operation Mode 2 was tested

The second operation mode is the car can detect an object infront of it. An object was placed infront of the car. The output dislay of Computer Program GUI was indicated.

Chapter 4

RESULTS AND DISCUSSIONS

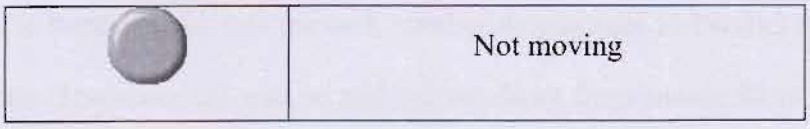
All tasks that have been described in Chapter 3 Methodology. This chapter stated the results of all tasks. Task 1 to Task 12 have been carried out without any modification. In this chapter, the result that was described here are result of Task113 and Task 14.

4.1 Task 13 : MODE 1 OPERATION TEST

Table 4.1 shows the results of this operation and Appendix D shows the operation Flow Chart that described the overall operation of this operation mode.

Table 4.1 : Result of Operation Mode 1

BUTTON PRESS	CAR MOVEMENT / DIRECTION
	Move forward
	Reverse
	Turn right
	Turn left



For the first action when the user pressed the button, the Computer Program was ready to send a number 1_d to the Parallel Port. Before the number was transferred to the Parallel Port, the number was converted into a byte of a number in binary that was 0000 0001. Next, the Computer Program sent the number to a specific register in the Parallel Port that was Data Port. The byte was transferred to Parallel Port Pins and Pin 2 that was connected to Data Port Bit 0 was in High state. The state of the pin caused transistor Q1 on and allowed current flowed from source S1 to point Pt1. When the current flows into point t1, the Signal Transmitter 1 transmitted forward signal to the car and as the result, the car move forward. Button Backward, button Right, button Left were performed same steps as button forward actions as described before but differences are the numbers that were used, the transistor involved, the current flow and the transmitted signals. For the Stop button the was slightly different from the button mechanisms.

When the button Backward was pressed, number 2_d was sent to Parallel Port, Pin 3 was in high state. Transistor Q2 was on and current flows from source S1 to point Pt2. The transmitter transmitted backward signal. Forward and Backward operation shared the same Source of Signal Transmitter. Both operations were indicated the car movements.

When the button Left was pressed, number 8_d was sent to Parallel Port, Pin 5 was in high state. Transistor Q4 was on and current flows from source S2 to point Pt4. The transmitter transmitted left signal.

When the button Right was pressed, number 4_d was sent to Parallel Port, Pin 4 was in high state. Transistor Q3 was on and current flows from source S2 to point Pt3. The transmitter transmitted right signal. Turning Right and Left operations shared the same Source of Signal Transmitter. Both operation were indicated the car directions.

When the button Stop was pressed, number 0_d was sent to Parallel Port, Al Pins were in Low state. None of transistor allowed any current flows from both sources S1 and S2 to all points Pt1, Pt2, Pt3 and Pt4. No transmitted signal to the car and as a result the car stopped.

4.2 MODE 2 OPERATION TEST

Table 4.1 shows the results of this operation and Appendix D shows the operation Flow Chart that described the overall operation of this operation mode.

Table 4.2 : Result of Operation Mode 2

SENSOR CONDITION	DISPLAY
No object detected	No sign and warning text
Detect object	

In this operation mode the input was reflected ultrasonic waves that sensed by the ultrasonic receiver URX and the output was output display at the Computer Program. Firstly, the microcontroller transmit trigger signal to the ultrasonic transmitter UTX to switch on the device. The ultrasonic transmitter, transmitted the ultrasonic waves. When

there are no object the ultrasonic waves were not reflected back the signal, so no input for this operation. When there was an object infront of the car, the waves that are transmitted by the ultrasonic waves were reflected by the object to the ultrasonic receiver URX. Ultrasonic receiver URX transmit a +5V square-wave 25μs electrical signal to the microcontroller. When microcontroller received that signal, it transmit a +5V 500ms square-wave to the transistor Q6 at Signal Transmitter 2. The transistor on and allowed current flowed from source Src1 to point Pnt1 and the Signal Transmitter 2 transmit an electromagnetic wave signals to Signal Receiver 2. When the Signal Receiver 2 received that signals, it transmit a current to transistor Q5. Transistor Q5 allowed current flowed from voltage regulator to Pin 12 of Parallel Port. The Computer Program read the signal from Pin 12 and Display the sign and text at the GUI of the Program.

As a conclusion Task 3 to Task 12 that have been completed driven the results of Task 13 and Task 14. For Task 13, button Forward, Backward, Right and Left had a same mechanism of operation. Their differences are, the number sent to Parallel Port, the used transistor, the point at Signal Transmitter 1 that received current and control signal. For task 14, there two possible input and output. In the test, the system can only detect object infront of it.

Chapter 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

The main objective of this project is to interface the remote control car with the computer and this objective was achieved. The computer can controls all movements and directions of the car.

The second objective is to build a remote control car that capable object infront of it. This objective has been achieved by placing a proximity detector that was controlled by a microcontroller at the remote control car.

The third objective is to develop a remote control car that can give feedback to the computer. This objective is achieved by attach Data Transmitter to the microcontroller and attach Data Receiver to the computer.

5.2 RECOMMENDATION

a) Change sensor operation to distance measurement. In this project, the proximity detector just sense object in front of the car but in distance measurement operation, the proximity detector measure the distance between the car and object in front of it. The result that display computer program in numerical form.

b) Create database for the car activities. The database consists of historical records of the car movements and directions and also the proximity detector

sensing activities. Each activity was recorded with their own time interval. The purpose of creating the database is to save all the activities and can be reviewed and playback in the future for analysis purposes or to perform the same task next time.

c) Use joystick as an alternative controller. The limitation of GUI buttons and mouse usage limitation has been described before in Chapter 4. When using joystick user doesn't has look at the device when controlling the car because his fingers can easily push the suitable joystick button by sensing suitable buttons. To perform this task, the program must be capable to interface the game port.

d) Add camera video to the car. This accessory to the car can make user much easier to control it. The video must be small and light because the car is in small size and also must be cordless system. The computer program will be quite complicated to display the output of this camera. DirectX can used to develop video output oriented program.

e) Interface other types of remote control system. In this project the car can move in four directions per time. Other remote control system, the car can combine the movement and direction of the car. As a result it can move in six directions per time. To carry out this job, the car must be substitute with the car that cable move in six directions and a few commands in computer program must be changed. Another type of remote control system that mimic the real car steering system. The heart of the controller actually consists of potentiometer. To carry out this job DAC can be used to interface the computer and the remote controller.

REFERENCES

1. Reddy, B.R.S., Reddy, G.A.V. and Sriram, V. *Surveillance Car*. Retrieved 3 July 2003 from http :// gdit.iiit.net / ~ravi_b / projects / embeddedforcv.htm.

2. Sirah, Phil. *RoboCar*. Retrieved 4 July 2003 from http :// website.lineone.net / ~iomarcus / robocar.html.

3. RC Car Construction and Interface. Retrieved 4 July 2003 : http:/ /www. drivemeinsane. Com / project / rccar / rccar.html

4. Dhananjay V.Gadre (1998) Programming the Parallel Port, Miller Freeman Inc.

5. Ibrahim, Dogan(2001). *PIC BASIC PROGRAMMING AND PROJECTS*. Newnes. Oxford, England.

6. Microchip Technology Inc. PICF84A Data Sheet Retreived at 5 July 2003 : http://wwww.microchip.com

7. *Application Module DT35 User Manual*. LJ Electronics Inc.

8. An Introduction to 80286 Microprocessor Applications D3000 Laboratory Manual 8.62 Part 2 of 2.

9. KIT 62, 5V Regulated voltage. Circuit Description. The Electronics Hobby Kit.

10. Jones, Richard M.(2000). *Introduction to MFC Programming with Visual C++*. Prentice Hall PTR. Upper Saddle River, New Jersey, USA.

11. Pappas, Chris H. & Murray, William H. (1998). *Visual C++ 6: The Complete Reference*. Osborne / McGraw-Hill. Berkeley, California, USA.

12. *EE401 Project Archive Fall 2001*. Retrieved 4 July 2003 from http :// www.ee.ualberta.ca / ~ee401 / archive_fall_2001.html.

APPENDIX A

Program MFCExample

File name : **stdafx.h**

```
LINE
   1  // stdafx.h : include file for standard system include files,
   2  //  or project specific include files that are used frequently, but
   3  //      are changed infrequently
   4  //
   5
   6  #if !defined(AFX_STDAFX_H__F67BDACD_F816_11D7_B1A3_F35D77579832__INCLUDED_)
   7  #define AFX_STDAFX_H__F67BDACD_F816_11D7_B1A3_F35D77579832__INCLUDED_
   8
   9  #if _MSC_VER > 1000
  10  #pragma once
  11  #endif // _MSC_VER > 1000
  12
  13  #define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows headers
  14
  15  #include <afxwin.h>         // MFC core and standard components
  16  #include <afxext.h>         // MFC extensions
  17  #include <afxdisp.h>        // MFC Automation classes
  18  #include <afxdtctl.h>               // MFC support for Internet Explorer 4 Common Controls
  19  #ifndef _AFX_NO_AFXCMN_SUPPORT
  20  #include <afxcmn.h>                 // MFC support for Windows Common Controls
  21  #endif // _AFX_NO_AFXCMN_SUPPORT
  22
  23
  24  //{{AFX_INSERT_LOCATION}}
  25  // Microsoft Visual C++ will insert additional declarations immediately before the previous line.
  26
  27  #endif // !defined(AFX_STDAFX_H__F67BDACD_F816_11D7_B1A3_F35D77579832__INCLUDED_)
```

File name : **stdafx.cpp**

```
LINE
   1  // stdafx.cpp : source file that includes just the standard includes
   2  //          MFCExample.pch will be the pre-compiled header
   3  //          stdafx.obj will contain the pre-compiled type information
   4
   5  #include "stdafx.h"
   6  // stdafx.cpp : source file that includes just the standard includes
```

File name : **MainFrm.h**

```
LINE
   1  // MainFrm.h : interface of the CMainFrame class
   2  //
   3  /////////////////////////////////////////////////////////////////////////////
   4
   5  #if !defined(AFX_MAINFRM_H__F67BDACF_F816_11D7_B1A3_F35D77579832__INCLUDED_)
   6  #define AFX_MAINFRM_H__F67BDACF_F816_11D7_B1A3_F35D77579832__INCLUDED_
   7
   8  #if _MSC_VER > 1000
   9  #pragma once
```

```
10      #endif // _MSC_VER > 1000
11
12      class CMainFrame : public CMDIFrameWnd
13      {
14              DECLARE_DYNAMIC(CMainFrame)
15      public:
16              CMainFrame();
17
18      // Attributes
19      public:
20
21      // Operations
22      public:
23
24      // Overrides
25              // ClassWizard generated virtual function overrides
26              //{{AFX_VIRTUAL(CMainFrame)
27              virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
28              //}}AFX_VIRTUAL
29
30      // Implementation
31      public:
32              virtual ~CMainFrame();
33      #ifdef _DEBUG
34              virtual void AssertValid() const;
35              virtual void Dump(CDumpContext& dc) const;
36      #endif
37
38      protected:  // control bar embedded members
39              CStatusBar  m_wndStatusBar;
40              CToolBar    m_wndToolBar;
41
42      // Generated message map functions
43      protected:
44              //{{AFX_MSG(CMainFrame)
45              afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
46                      // NOTE - the ClassWizard will add and remove member functions here.
47                      //    DO NOT EDIT what you see in these blocks of generated code!
48              //}}AFX_MSG
49              DECLARE_MESSAGE_MAP()
50      };
51
52      /////////////////////////////////////////////////////////////////////////////
53
54      //{{AFX_INSERT_LOCATION}}
55      // Microsoft Visual C++ will insert additional declarations immediately before the previous line.
56
57      #endif // !defined(AFX_MAINFRM_H__F67BDACF_F816_11D7_B1A3_F35D77579832__INCLUDED_)
```

File name : **MainFrm.cpp**

LINE	SOURCE CODE

```
1       // MainFrm.cpp : implementation of the CMainFrame class
2       //
3
4       #include "stdafx.h"
5       #include "MFCExample.h"
6
7       #include "MainFrm.h"
8
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        ON_WM_CREATE()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
        ID_SEPARATOR,           // status line indicator
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM,
        ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
        // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
                return -1;

        if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP
                | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
                !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
        {
                TRACE0("Failed to create toolbar\n");
                return -1;      // fail to create
        }

        if (!m_wndStatusBar.Create(this) ||
                !m_wndStatusBar.SetIndicators(indicators,
                  sizeof(indicators)/sizeof(UINT)))
        {
                TRACE0("Failed to create status bar\n");
                return -1;      // fail to create
        }

        // TODO: Delete these three lines if you don't want the toolbar to
```

```
71              // be dockable
72              m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
73              EnableDocking(CBRS_ALIGN_ANY);
74              DockControlBar(&m_wndToolBar);
75
76              return 0;
77      }
78
79      BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
80      {
81              if( !CMDIFrameWnd::PreCreateWindow(cs) )
82                      return FALSE;
83              // TODO: Modify the Window class or styles here by modifying
84              //  the CREATESTRUCT cs
85
86              return TRUE;
87      }
88
89      /////////////////////////////////////////////////////////////////////////////
90      // CMainFrame diagnostics
91
92      #ifdef _DEBUG
93      void CMainFrame::AssertValid() const
94      {
95              CMDIFrameWnd::AssertValid();
96      }
97
98      void CMainFrame::Dump(CDumpContext& dc) const
99      {
100             CMDIFrameWnd::Dump(dc);
101     }
102
103     #endif //_DEBUG
104
105     /////////////////////////////////////////////////////////////////////////////
106     // CMainFrame message handlers
```

File name : **Resource.h**

LINE	SOURCE CODE
1	//{{NO_DEPENDENCIES}}
2	// Microsoft Visual C++ generated include file.
3	// Used by MFCEXAMPLE.RC
4	//
5	#define IDD_ABOUTBOX 100
6	#define IDR_MAINFRAME 128
7	#define IDR_MFCEXATYPE 129
8	
9	// Next default values for new objects
10	//
11	#ifdef APSTUDIO_INVOKED
12	#ifndef APSTUDIO_READONLY_SYMBOLS
13	#define _APS_3D_CONTROLS 1
14	#define _APS_NEXT_RESOURCE_VALUE 130
15	#define _APS_NEXT_CONTROL_VALUE 1000
16	#define _APS_NEXT_SYMED_VALUE 101
17	#define _APS_NEXT_COMMAND_VALUE 32771
18	#endif
19	#endif

File name : **MFCExample.h**

LINE	SOURCE CODE
1	// MFCExample.h : main header file for the MFCEXAMPLE application
2	//
3	
4	#if !defined(AFX_MFCEXAMPLE_H__F67BDACB_F816_11D7_B1A3_F35D77579832__INCLUDED_)
5	#define AFX_MFCEXAMPLE_H__F67BDACB_F816_11D7_B1A3_F35D77579832__INCLUDED_
6	
7	#if _MSC_VER > 1000
8	#pragma once
9	#endif // _MSC_VER > 1000
10	
11	#ifndef __AFXWIN_H__
12	#error include 'stdafx.h' before including this file for PCH
13	#endif
14	
15	#include "resource.h" // main symbols
16	
17	///
18	// CMFCExampleApp:
19	// See MFCExample.cpp for the implementation of this class
20	//
21	
22	class CMFCExampleApp : public CWinApp
23	{
24	public:
25	CMFCExampleApp();
26	
27	// Overrides
28	// ClassWizard generated virtual function overrides
29	//{{AFX_VIRTUAL(CMFCExampleApp)
30	public:
31	virtual BOOL InitInstance();
32	virtual int ExitInstance();
33	//}}AFX_VIRTUAL
34	
35	// Implementation
36	protected:
37	HMENU m_hMDIMenu;
38	HACCEL m_hMDIAccel;
39	
40	public:
41	//{{AFX_MSG(CMFCExampleApp)
42	afx_msg void OnAppAbout();
43	afx_msg void OnFileNew();
44	// NOTE - the ClassWizard will add and remove member functions here.
45	// DO NOT EDIT what you see in these blocks of generated code !
46	//}}AFX_MSG
47	DECLARE_MESSAGE_MAP()
48	};
49	
50	///
51	
52	//{{AFX_INSERT_LOCATION}}
53	// Microsoft Visual C++ will insert additional declarations immediately before the previous line.
54	
55	#endif //

```
56      !defined(AFX_MFCEXAMPLE_H__F67BDACB_F816_11D7_B1A3_F35D77579832__INCLUDED_)
```

File name : **MFCExample.cpp**

LINE	SOURCE CODE

```
1       // MFCExample.cpp : Defines the class behaviors for the application.
2       //
3
4       #include "stdafx.h"
5       #include "MFCExample.h"
6
7       #include "MainFrm.h"
8       #include "ChildFrm.h"
9
10      #ifdef _DEBUG
11      #define new DEBUG_NEW
12      #undef THIS_FILE
13      static char THIS_FILE[] = __FILE__;
14      #endif
15
16      /////////////////////////////////////////////////////////////////////////////
17      // CMFCExampleApp
18
19      BEGIN_MESSAGE_MAP(CMFCExampleApp, CWinApp)
20              //{{AFX_MSG_MAP(CMFCExampleApp)
21              ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
22                      // NOTE - the ClassWizard will add and remove mapping macros here.
23                      //    DO NOT EDIT what you see in these blocks of generated code!
24              ON_COMMAND(ID_FILE_NEW, OnFileNew)
25              //}}AFX_MSG_MAP
26      END_MESSAGE_MAP()
27
28      /////////////////////////////////////////////////////////////////////////////
29      // CMFCExampleApp construction
30
31      CMFCExampleApp::CMFCExampleApp()
32      {
33              // TODO: add construction code here,
34              // Place all significant initialization in InitInstance
35      }
36
37      /////////////////////////////////////////////////////////////////////////////
38      // The one and only CMFCExampleApp object
39
40      CMFCExampleApp theApp;
41
42      /////////////////////////////////////////////////////////////////////////////
43      // CMFCExampleApp initialization
44
45      BOOL CMFCExampleApp::InitInstance()
46      {
47              AfxEnableControlContainer();
48
49              // Standard initialization
50              // If you are not using these features and wish to reduce the size
51              //  of your final executable, you should remove from the following
52              //  the specific initialization routines you do not need.
53
54      #ifdef _AFXDLL
55              Enable3dControls();                     // Call this when using MFC in a shared DLL
```

```
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif

	// Change the registry key under which our settings are stored.
	// TODO: You should modify this string to be something appropriate
	// such as the name of your company or organization.
	SetRegistryKey(_T("Local AppWizard-Generated Applications"));


	// To create the main window, this code creates a new frame window
	// object and then sets it as the application's main window object.

	CMDIFrameWnd* pFrame = new CMainFrame;
	m_pMainWnd = pFrame;

	// create main MDI frame window
	if (!pFrame->LoadFrame(IDR_MAINFRAME))
		return FALSE;

	// try to load shared MDI menus and accelerator table
	//TODO: add additional member variables and load calls for
	//	additional menu types your application may need.

	HINSTANCE hInst = AfxGetResourceHandle();
	m_hMDIMenu  = ::LoadMenu(hInst, MAKEINTRESOURCE(IDR_MFCEXATYPE));
	m_hMDIAccel = ::LoadAccelerators(hInst, MAKEINTRESOURCE(IDR_MFCEXATYPE));



	// The main window has been initialized, so show and update it.
	pFrame->ShowWindow(m_nCmdShow);
	pFrame->UpdateWindow();

	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMFCExampleApp message handlers

int CMFCExampleApp::ExitInstance()
{
	//TODO: handle additional resources you may have added
	if (m_hMDIMenu != NULL)
		FreeResource(m_hMDIMenu);
	if (m_hMDIAccel != NULL)
		FreeResource(m_hMDIAccel);

	return CWinApp::ExitInstance();
}

void CMFCExampleApp::OnFileNew()
{
	CMainFrame* pFrame = STATIC_DOWNCAST(CMainFrame, m_pMainWnd);

	// create a new MDI child window
	pFrame->CreateNewChild(
		RUNTIME_CLASS(CChildFrame), IDR_MFCEXATYPE, m_hMDIMenu, m_hMDIAccel);
}
```

```
118  /////////////////////////////////////////////////////////////////////////////
119  // CAboutDlg dialog used for App About
120
121  class CAboutDlg : public CDialog
122  {
123  public:
124          CAboutDlg();
125
126  // Dialog Data
127          //{{AFX_DATA(CAboutDlg)
128          enum { IDD = IDD_ABOUTBOX };
129          //}}AFX_DATA
130
131          // ClassWizard generated virtual function overrides
132          //{{AFX_VIRTUAL(CAboutDlg)
133          protected:
134          virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
135          //}}AFX_VIRTUAL
136
137  // Implementation
138  protected:
139          //{{AFX_MSG(CAboutDlg)
140                  // No message handlers
141          //}}AFX_MSG
142          DECLARE_MESSAGE_MAP()
143  };
144
145  CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
146  {
147          //{{AFX_DATA_INIT(CAboutDlg)
148          //}}AFX_DATA_INIT
149  }
150
151  void CAboutDlg::DoDataExchange(CDataExchange* pDX)
152  {
153          CDialog::DoDataExchange(pDX);
154          //{{AFX_DATA_MAP(CAboutDlg)
155          //}}AFX_DATA_MAP
156  }
157
158  BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
159          //{{AFX_MSG_MAP(CAboutDlg)
160                  // No message handlers
161          //}}AFX_MSG_MAP
162  END_MESSAGE_MAP()
163
164  // App command to run the dialog
165  void CMFCExampleApp::OnAppAbout()
166  {
167          CAboutDlg aboutDlg;
168          aboutDlg.DoModal();
169  }
170
171  /////////////////////////////////////////////////////////////////////////////
172  // CMFCExampleApp message handlers
```

File name : **ChildView.h**

LINE	SOURCE CODE

```
1   // ChildView.h : interface of the CChildView class
2   //
3   /////////////////////////////////////////////////////////////////////////////
4
5   #if !defined(AFX_CHILDVIEW_H__F67BDAD3_F816_11D7_B1A3_F35D77579832__INCLUDED_)
6   #define AFX_CHILDVIEW_H__F67BDAD3_F816_11D7_B1A3_F35D77579832__INCLUDED_
7
8   #if _MSC_VER > 1000
9   #pragma once
10  #endif // _MSC_VER > 1000
11
12  /////////////////////////////////////////////////////////////////////////////
13  // CChildView window
14
15  class CChildView : public CWnd
16  {
17  // Construction
18  public:
19          CChildView();
20
21  // Attributes
22  public:
23
24  // Operations
25  public:
26
27  // Overrides
28          // ClassWizard generated virtual function overrides
29          //{{AFX_VIRTUAL(CChildView)
30          protected:
31          virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
32          //}}AFX_VIRTUAL
33
34  // Implementation
35  public:
36          virtual ~CChildView();
37
38          // Generated message map functions
39  protected:
40          //{{AFX_MSG(CChildView)
41          afx_msg void OnPaint();
42          //}}AFX_MSG
43          DECLARE_MESSAGE_MAP()
44  };
45
46  /////////////////////////////////////////////////////////////////////////////
47
48  //{{AFX_INSERT_LOCATION}}
49  // Microsoft Visual C++ will insert additional declarations immediately before the previous line.
50
51  #endif // !defined(AFX_CHILDVIEW_H__F67BDAD3_F816_11D7_B1A3_F35D77579832__INCLUDED_)
```

File name : **ChildView.cpp**

LINE	SOURCE CODE

```
1   // ChildView.cpp : implementation of the CChildView class
2   //
3
4   #include "stdafx.h"
5   #include "MFCExample.h"
6   #include "ChildView.h"
7
8   #ifdef _DEBUG
9   #define new DEBUG_NEW
10  #undef THIS_FILE
11  static char THIS_FILE[] = __FILE__;
12  #endif
13
14  /////////////////////////////////////////////////////////////////////////////
15  // CChildView
16
17  CChildView::CChildView()
18  {
19  }
20
21  CChildView::~CChildView()
22  {
23  }
24
25
26  BEGIN_MESSAGE_MAP(CChildView,CWnd )
27          //{{AFX_MSG_MAP(CChildView)
28          ON_WM_PAINT()
29          //}}AFX_MSG_MAP
30  END_MESSAGE_MAP()
31
32
33  /////////////////////////////////////////////////////////////////////////////
34  // CChildView message handlers
35
36  BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
37  {
38          if (!CWnd::PreCreateWindow(cs))
39                  return FALSE;
40
41          cs.dwExStyle |= WS_EX_CLIENTEDGE;
42          cs.style &= ~WS_BORDER;
43          cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
44                  ::LoadCursor(NULL, IDC_ARROW), HBRUSH(COLOR_WINDOW+1), NULL);
45
46          return TRUE;
47  }
48
49  void CChildView::OnPaint()
50  {
51          CPaintDC dc(this); // device context for painting
52
53          // TODO: Add your message handler code here
54
55          // Do not call CWnd::OnPaint() for painting messages
56  }
```

File name : **ChildFrm.h**

LINE	SOURCE CODE

```
1   // ChildFrm.h : interface of the CChildFrame class
2   //
3   /////////////////////////////////////////////////////////////////////////////
4
5   #if !defined(AFX_CHILDFRM_H__F67BDAD1_F816_11D7_B1A3_F35D77579832__INCLUDED_)
6   #define AFX_CHILDFRM_H__F67BDAD1_F816_11D7_B1A3_F35D77579832__INCLUDED_
7
8   #if _MSC_VER > 1000
9   #pragma once
10  #endif // _MSC_VER > 1000
11
12  #include "ChildView.h"
13
14  class CChildFrame : public CMDIChildWnd
15  {
16          DECLARE_DYNCREATE(CChildFrame)
17  public:
18          CChildFrame();
19
20  // Attributes
21  public:
22
23  // Operations
24  public:
25
26  // Overrides
27          // ClassWizard generated virtual function overrides
28          //{{AFX_VIRTUAL(CChildFrame)
29          public:
30          virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
            virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO*
31  pHandlerInfo);
32          //}}AFX_VIRTUAL
33
34  // Implementation
35  public:
36          // view for the client area of the frame.
37          CChildView m_wndView;
38          virtual ~CChildFrame();
39  #ifdef _DEBUG
40          virtual void AssertValid() const;
41          virtual void Dump(CDumpContext& dc) const;
42  #endif
43
44  // Generated message map functions
45  protected:
46          //{{AFX_MSG(CChildFrame)
47                  // NOTE - the ClassWizard will add and remove member functions here.
48                  //    DO NOT EDIT what you see in these blocks of generated code!
49          afx_msg void OnFileClose();
50          afx_msg void OnSetFocus(CWnd* pOldWnd);
51          afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
52          //}}AFX_MSG
53          DECLARE_MESSAGE_MAP()
54  };
55
56  /////////////////////////////////////////////////////////////////////////////
57
```

```
58      //{{AFX_INSERT_LOCATION}}
59      // Microsoft Visual C++ will insert additional declarations immediately before the previous line.
60
61      #endif // !defined(AFX_CHILDFRM_H__F67BDAD1_F816_11D7_B1A3_F35D77579832__INCLUDED_)
```

File name : **ChildFrm.cpp**

LINE	SOURCE CODE

```
1       // ChildFrm.cpp : implementation of the CChildFrame class
2       //
3
4       #include "stdafx.h"
5       #include "MFCExample.h"
6
7       #include "ChildFrm.h"
8
9       #ifdef _DEBUG
10      #define new DEBUG_NEW
11      #undef THIS_FILE
12      static char THIS_FILE[] = __FILE__;
13      #endif
14
15      /////////////////////////////////////////////////////////////////////////////
16      // CChildFrame
17
18      IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)
19
20      BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
21              //{{AFX_MSG_MAP(CChildFrame)
22                      // NOTE - the ClassWizard will add and remove mapping macros here.
23                      //    DO NOT EDIT what you see in these blocks of generated code !
24                      ON_COMMAND(ID_FILE_CLOSE, OnFileClose)
25                      ON_WM_SETFOCUS()
26                      ON_WM_CREATE()
27              //}}AFX_MSG_MAP
28      END_MESSAGE_MAP()
29
30      /////////////////////////////////////////////////////////////////////////////
31      // CChildFrame construction/destruction
32
33      CChildFrame::CChildFrame()
34      {
35              // TODO: add member initialization code here
36
37      }
38
39      CChildFrame::~CChildFrame()
40      {
41      }
42
43      BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
44      {
45              // TODO: Modify the Window class or styles here by modifying
46              //  the CREATESTRUCT cs
47
48              if( !CMDIChildWnd::PreCreateWindow(cs) )
49                      return FALSE;
50
51              cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
52              cs.lpszClass = AfxRegisterWndClass(0);
```

```
53
54              return TRUE;
55      }
56
57
58
59      /////////////////////////////////////////////////////////////////////////
60      // CChildFrame diagnostics
61
62      #ifdef _DEBUG
63      void CChildFrame::AssertValid() const
64      {
65              CMDIChildWnd::AssertValid();
66      }
67
68      void CChildFrame::Dump(CDumpContext& dc) const
69      {
70              CMDIChildWnd::Dump(dc);
71      }
72
73      #endif //_DEBUG
74
75      /////////////////////////////////////////////////////////////////////////
76      // CChildFrame message handlers
77      void CChildFrame::OnFileClose()
78      {
79              // To close the frame, just send a WM_CLOSE, which is the equivalent
80              // choosing close from the system menu.
81
82              SendMessage(WM_CLOSE);
83      }
84
85      int CChildFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
86      {
87              if (CMDIChildWnd::OnCreate(lpCreateStruct) == -1)
88                      return -1;
89
90              // create a view to occupy the client area of the frame
91              if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
92                      CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL))
93              {
94                      TRACE0("Failed to create view window\n");
95                      return -1;
96              }
97
98              return 0;
99      }
100
101     void CChildFrame::OnSetFocus(CWnd* pOldWnd)
102     {
103             CMDIChildWnd::OnSetFocus(pOldWnd);
104
105             m_wndView.SetFocus();
106     }
107
108     BOOL CChildFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo)
109     {
110             // let the view have first crack at the command
111             if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
112                     return TRUE;
113
114             // otherwise, do default handling
```

```
115             return CMDIChildWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
116     }
```

File name : **ReadMe.txt**

```
========================================================================
    MICROSOFT FOUNDATION CLASS LIBRARY : MFCExample
========================================================================
```

AppWizard has created this MFCExample application for you. This application not only demonstrates the basics of using the Microsoft Foundation classes but is also a starting point for writing your application.

This file contains a summary of what you will find in each of the files that make up your MFCExample application.

MFCExample.dsp
This file (the project file) contains information at the project level and is used to build a single project or subproject. Other users can share the project (.dsp) file, but they should export the makefiles locally.

MFCExample.h
This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares the CMFCExampleApp application class.

MFCExample.cpp
This is the main application source file that contains the application class CMFCExampleApp.

MFCExample.rc
This is a listing of all of the Microsoft Windows resources that the program uses. It includes the icons, bitmaps, and cursors that are stored in the RES subdirectory. This file can be directly edited in Microsoft Visual C++.

MFCExample.clw
This file contains information used by ClassWizard to edit existing classes or add new classes. ClassWizard also uses this file to store information needed to create and edit message maps and dialog data maps and to create prototype member functions.

res\MFCExample.ico
This is an icon file, which is used as the application's icon. This icon is included by the main resource file MFCExample.rc.

res\MFCExample.rc2
This file contains resources that are not edited by Microsoft Visual C++. You should place all resources not editable by the resource editor in this file.

///

For the main frame window:

MainFrm.h, MainFrm.cpp
These files contain the frame class CMainFrame, which is derived from CMDIFrameWnd and controls all MDI frame features.

res\Toolbar.bmp
This bitmap file is used to create tiled images for the toolbar. The initial toolbar and status bar are constructed in the CMainFrame class. Edit this toolbar bitmap using the resource editor, and update the IDR_MAINFRAME TOOLBAR array in MFCExample.rc to add toolbar buttons.

///

For the child frame window:

ChildFrm.h, ChildFrm.cpp
These files define and implement the CChildFrame class, which supports the child windows in an MDI application.

///

///

Other standard files:

StdAfx.h, StdAfx.cpp
These files are used to build a precompiled header (PCH) file named MFCExample.pch and a precompiled types file named StdAfx.obj.

Resource.h
This is the standard header file, which defines new resource IDs. Microsoft Visual C++ reads and updates this file.

///
Other notes:

AppWizard uses "TODO:" to indicate parts of the source code you should add to or customize.

If your application uses MFC in a shared DLL, and your application is in a language other than the operating system's current language, you will need to copy the corresponding localized resources MFC42XXX.DLL from the Microsoft Visual C++ CD-ROM onto the system or system32 directory, and rename it to be MFCLOC.DLL. ("XXX" stands for the language abbreviation.

For example, MFC42DEU.DLL contains resources translated to German.) If you don't do this, some of the UI elements of your application will remain in the language of the operating system.

///

ParPortExample Program

This program use DialogBox as its' interface. It has same files like MFCExample in Appendix A except for these files :

1. MainFrm.h
2. MainFrm.cpp
3. ChildFrm.h
4. ChildFrm.cpp

There are 2 additional files :

1. ParPortExampleDlg.h
2. ParPortExampleDlg.cpp

These 2 files were the application of the project.

File name : ParPortExampleDlg.h

```
LINE                                                    SOURCE CODE

  1     // ParPortExampleDlg.h : header file
  2     //
  3
  4     #if !defined(AFX_PARPORTEXAMPLEDLG_H__6ECEBBA7_F885_11D7_B1A3_AF4FB94E6A35__INCLUDED_)
  5     #define AFX_PARPORTEXAMPLEDLG_H__6ECEBBA7_F885_11D7_B1A3_AF4FB94E6A35__INCLUDED_
  6
  7     #if _MSC_VER > 1000
  8     #pragma once
  9     #endif // _MSC_VER > 1000
 10
 11     /////////////////////////////////////////////////////////////////////////////
 12     // CParPortExampleDlg dialog
 13
 14     class CParPortExampleDlg : public CDialog
 15     {
 16     // Construction
 17     public:
 18             int Signal;
 19             int StatusIn;
 20             CParPortExampleDlg(CWnd* pParent = NULL);    // standard constructor
 21     // Dialog Data
 22             //{{AFX_DATA(CParPortExampleDlg)
 23             enum { IDD = IDD_PARPORTEXAMPLE_DIALOG };
 24             Cbutton btnCheck;
 25             Cbutton btnSend;
 26             // NOTE: the ClassWizard will add data members here
 27             //}}AFX_DATA
 28
 29             // ClassWizard generated virtual function overrides
 30             //{{AFX_VIRTUAL(CParPortExampleDlg)
 31             protected:
 32             virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV support
 33             //}}AFX_VIRTUAL
 34
 35     // Implementation
 36     protected:
 37             HICON m_hIcon;
 38
 39             // Generated message map functions
 40             //{{AFX_MSG(CParPortExampleDlg)
 41             virtual BOOL OnInitDialog();
```

```
42              afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
43              afx_msg void OnPaint();
44              afx_msg HCURSOR OnQueryDragIcon();
45              afx_msg void OnSignalIn();
46              afx_msg void OnCheck();
47              afx_msg void OnSend();
48              virtual void OnCancel();
49              //}}AFX_MSG
50              DECLARE_MESSAGE_MAP()
51      };
52
53      //{{AFX_INSERT_LOCATION}}
54      // Microsoft Visual C++ will insert additional declarations immediately before the previous line.
55
56      #endif //
57      !defined(AFX_PARPORTEXAMPLEDLG_H__6ECEBBA7_F885_11D7_B1A3_AF4FB94E6A35__INCLUDED_)
```

File name : ParPortExampleDlg.cpp

LINE **SOURCE CODE**

```
1       // ParPortExampleDlg.cpp : implementation file
2       //
3
4       #include "stdafx.h"
5       #include "ParPortExample.h"
6       #include "ParPortExampleDlg.h"
7
8       #include "conio.h"
9
10      #ifdef _DEBUG
11      #define new DEBUG_NEW
12      #undef THIS_FILE
13      static char THIS_FILE[] = __FILE__;
14      #endif
15
16      /////////////////////////////////////////////////////////////////////////////
17      // CAboutDlg dialog used for App About
18
19      class CAboutDlg : public CDialog
20      {
21      public:
22              CAboutDlg();
23
24      // Dialog Data
25              //{{AFX_DATA(CAboutDlg)
26              enum { IDD = IDD_ABOUTBOX };
27              //}}AFX_DATA
28
29              // ClassWizard generated virtual function overrides
30              //{{AFX_VIRTUAL(CAboutDlg)
31              protected:
32              virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
33              //}}AFX_VIRTUAL
34
35      // Implementation
36      protected:
37              //{{AFX_MSG(CAboutDlg)
38              //}}AFX_MSG
39              DECLARE_MESSAGE_MAP()
40      };
41
42      CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
```

```
43      {
44              //{{AFX_DATA_INIT(CAboutDlg)
45              //}}AFX_DATA_INIT
46      }
47
48      void CAboutDlg::DoDataExchange(CDataExchange* pDX)
49      {
50              CDialog::DoDataExchange(pDX);
51              //{{AFX_DATA_MAP(CAboutDlg)
52              //}}AFX_DATA_MAP
53      }
54
55      BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
56              //{{AFX_MSG_MAP(CAboutDlg)
57                      // No message handlers
58              //}}AFX_MSG_MAP
59      END_MESSAGE_MAP()
60
61      /////////////////////////////////////////////////////////////////////////////
62      // CParPortExampleDlg dialog
63
64      CParPortExampleDlg::CParPortExampleDlg(CWnd* pParent /*=NULL*/)
65              : CDialog(CParPortExampleDlg::IDD, pParent)
66      {
67              //{{AFX_DATA_INIT(CParPortExampleDlg)
68                      // NOTE: the ClassWizard will add member initialization here
69              //}}AFX_DATA_INIT
70              // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
71              m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
72      }
73
74      void CParPortExampleDlg::DoDataExchange(CDataExchange* pDX)
75      {
76              CDialog::DoDataExchange(pDX);
77              //{{AFX_DATA_MAP(CParPortExampleDlg)
78                      // NOTE: the ClassWizard will add DDX and DDV calls here
79              DDX_Control(pDX, IDC_CHECK, btnCheck);
80              DDX_Control(pDX, IDC_SEND, btnSend);
81              //}}AFX_DATA_MAP
82      }
83
84      BEGIN_MESSAGE_MAP(CParPortExampleDlg, CDialog)
85              //{{AFX_MSG_MAP(CParPortExampleDlg)
86              ON_WM_SYSCOMMAND()
87              ON_WM_PAINT()
88              ON_WM_QUERYDRAGICON()
89              ON_BN_CLICKED(IDC_SIGNALIN, OnSignalIn)
90              ON_BN_CLICKED(IDC_CHECK, OnCheck)
91              ON_BN_CLICKED(IDC_SEND, OnSend)
92              //}}AFX_MSG_MAP
93      END_MESSAGE_MAP()
94
95      /////////////////////////////////////////////////////////////////////////////
96      // CParPortExampleDlg message handlers
97
98      BOOL CParPortExampleDlg::OnInitDialog()
99      {
100             CDialog::OnInitDialog();
101
102             // Add "About..." menu item to system menu.
103
104             // IDM_ABOUTBOX must be in the system command range.
```

```
105		ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
106		ASSERT(IDM_ABOUTBOX < 0xF000);
107
108		CMenu* pSysMenu = GetSystemMenu(FALSE);
109		if (pSysMenu != NULL)
110		{
111			CString strAboutMenu;
112			strAboutMenu.LoadString(IDS_ABOUTBOX);
113			if (!strAboutMenu.IsEmpty())
114			{
115				pSysMenu->AppendMenu(MF_SEPARATOR);
116				pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
117			}
118		}
119
120		// Set the icon for this dialog.  The framework does this automatically
121		//  when the application's main window is not a dialog
122		SetIcon(m_hIcon, TRUE);			// Set big icon
123		SetIcon(m_hIcon, FALSE);		// Set small icon
124
125		// TODO: Add extra initialization here
126
127		return TRUE;  // return TRUE  unless you set the focus to a control
128	}
129
130	void CParPortExampleDlg::OnSysCommand(UINT nID, LPARAM lParam)
131	{
132		if ((nID & 0xFFF0) == IDM_ABOUTBOX)
133		{
134			CAboutDlg dlgAbout;
135			dlgAbout.DoModal();
136		}
137		else
138		{
139			CDialog::OnSysCommand(nID, lParam);
140		}
141	}
142
143	// If you add a minimize button to your dialog, you will need the code below
144	//  to draw the icon.  For MFC applications using the document/view model,
145	//  this is automatically done for you by the framework.
146
147	void CParPortExampleDlg::OnPaint()
148	{
149		if (IsIconic())
150		{
151			CPaintDC dc(this); // device context for painting
152
153			SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
154
155			// Center icon in client rectangle
156			int cxIcon = GetSystemMetrics(SM_CXICON);
157			int cyIcon = GetSystemMetrics(SM_CYICON);
158			CRect rect;
159			GetClientRect(&rect);
160			int x = (rect.Width() - cxIcon + 1) / 2;
161			int y = (rect.Height() - cyIcon + 1) / 2;
162
163			// Draw the icon
164			dc.DrawIcon(x, y, m_hIcon);
165		}
166		else
```

```
167             {
168                     CDialog::OnPaint();
169             }
170     }
171
172     // The system calls this to obtain the cursor to display while the user drags
173     //  the minimized window.
174     HCURSOR CParPortExampleDlg::OnQueryDragIcon()
175     {
176             return (HCURSOR) m_hIcon;
177     }
178
179     void CParPortExampleDlg::OnSignalIn()
180     {
181             // TODO: Add your control notification handler code here
182             Signal = _inp( 0x0379 );
183             StatusIn = Signal & 32;
184
185             if( StatusIn == 32)
186                     btnCheck.SetCheck(1);
187     }
188
189     void CParPortExampleDlg::OnCheck()
190     {
191             // TODO: Add your control notification handler code here
192
193     }
194
195     void CParPortExampleDlg::OnSend()
196     {
197             // TODO: Add your control notification handler code here
198             _outp( 0x0378, 1 );
199     }
200
201     void CParPortExampleDlg::OnCancel()
202     {
203             // TODO: Add extra cleanup here
204             _outp( 0x0378, 0 );
205             CDialog::OnCancel();
206     }
```

File name : **ReadMe.txt**

```
========================================================================
                    MICROSOFT FOUNDATION CLASS LIBRARY : ParPortExample
========================================================================
```

AppWizard has created this ParPortExample application for you. This application not only demonstrates the basics of using the Microsoft Foundation classes but is also a starting point for writing your application.

This file contains a summary of what you will find in each of the files that make up your ParPortExample application.

ParPortExample.dsp
This file (the project file) contains information at the project level and is used to build a single project or subproject. Other users can share the project (.dsp) file, but they should export the makefiles locally.

ParPortExample.h

This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares the CParPortExampleApp application class.

ParPortExample.cpp
This is the main application source file that contains the application class CParPortExampleApp.

ParPortExample.rc
This is a listing of all of the Microsoft Windows resources that the program uses. It includes the icons, bitmaps, and cursors that are stored in the RES subdirectory. This file can be directly edited in Microsoft Visual C++.

ParPortExample.clw
This file contains information used by ClassWizard to edit existing classes or add new classes. ClassWizard also uses this file to store information needed to create and edit message maps and dialog data maps and to create prototype member functions.

res\ParPortExample.ico
This is an icon file, which is used as the application's icon. This icon is included by the main resource file ParPortExample.rc.

res\ParPortExample.rc2
This file contains resources that are not edited by Microsoft Visual C++. You should place all resources not editable by the resource editor in this file.

///

AppWizard creates one dialog class:

ParPortExampleDlg.h, ParPortExampleDlg.cpp - the dialog
These files contain your CParPortExampleDlg class. This class defines the behavior of your application's main dialog. The dialog's template is in ParPortExample.rc, which can be edited in Microsoft Visual C++.

///
Other standard files:

StdAfx.h, StdAfx.cpp
These files are used to build a precompiled header (PCH) file named ParPortExample.pch and a precompiled types file named StdAfx.obj.

Resource.h
This is the standard header file, which defines new resource IDs. Microsoft Visual C++ reads and updates this file.

///
Other notes:

AppWizard uses "TODO:" to indicate parts of the source code you should add to or customize.

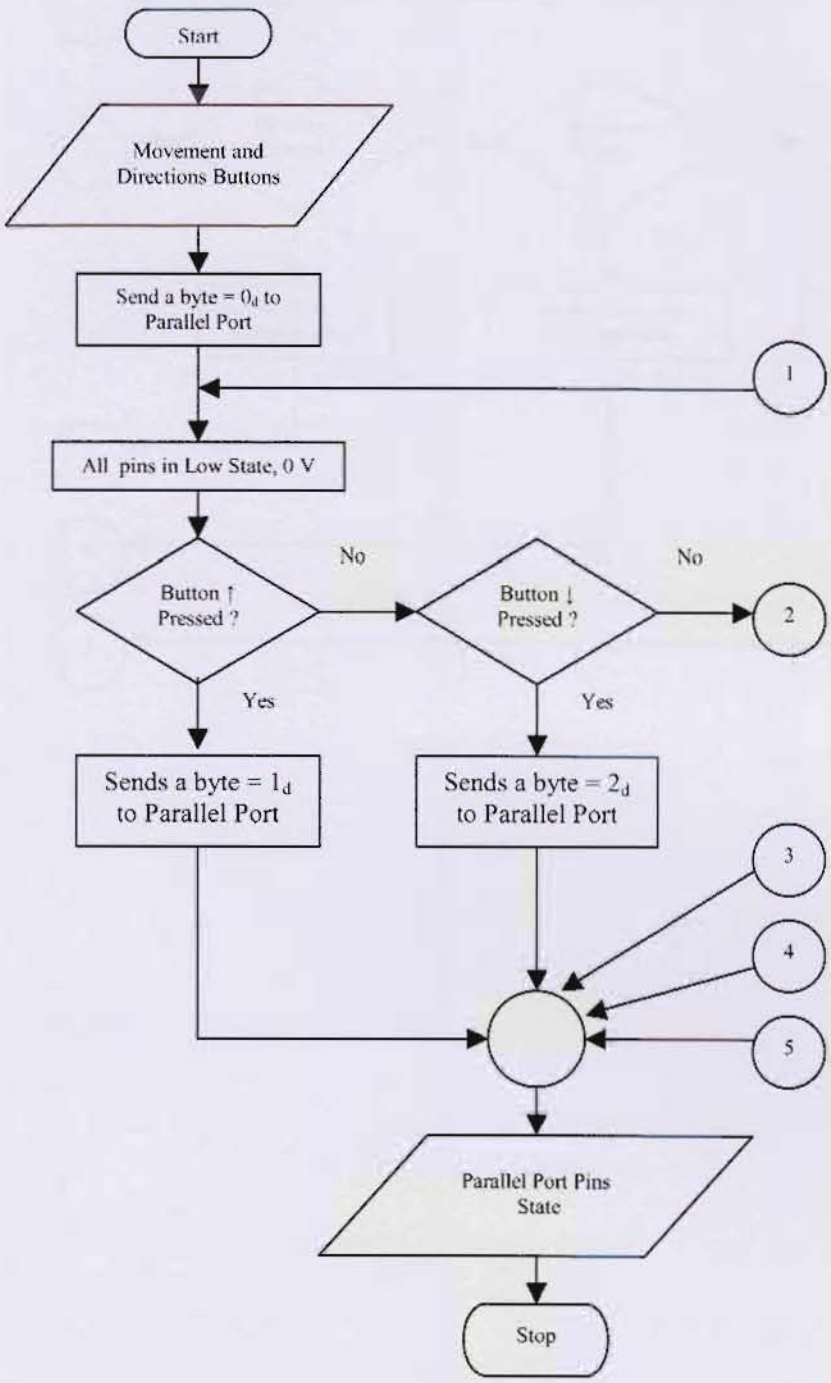
If your application uses MFC in a shared DLL, and your application is in a language other than the operating system's current language, you will need to copy the corresponding localized resources MFC42XXX.DLL from the Microsoft Visual C++ CD-ROM onto the system or system32 directory, and rename it to be MFCLOC.DLL. ("XXX" stands for the language abbreviation.

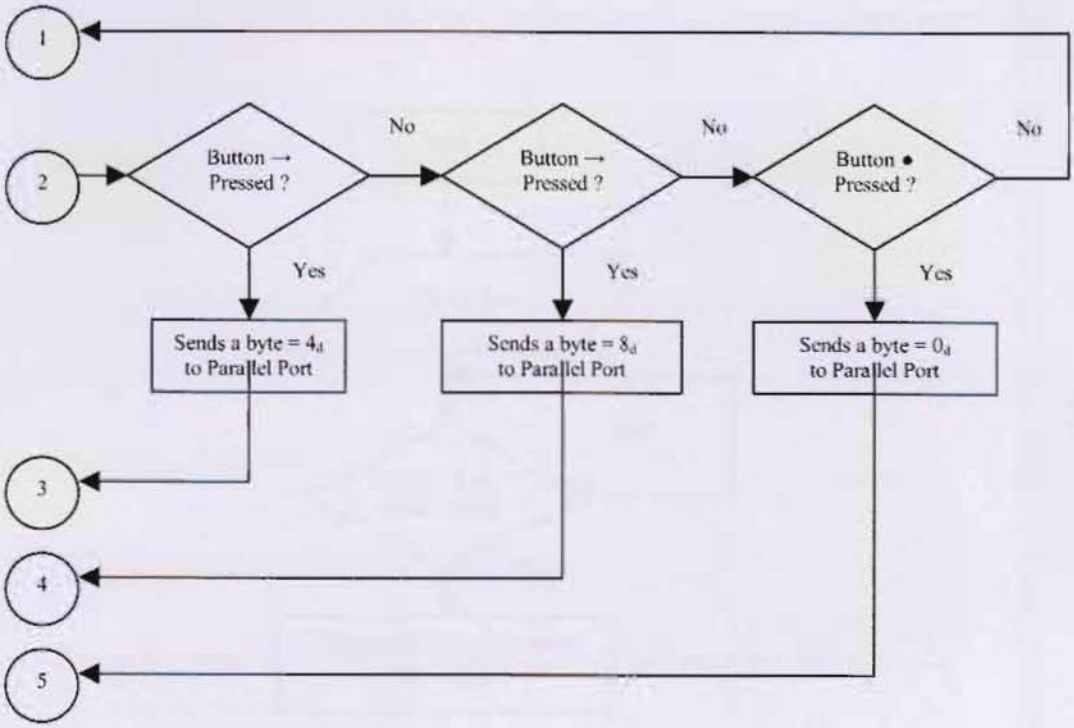
For example, MFC42DEU.DLL contains resources translated to German.) If you don't do this, some of the UI elements of your application will remain in the language of the operating system.

///

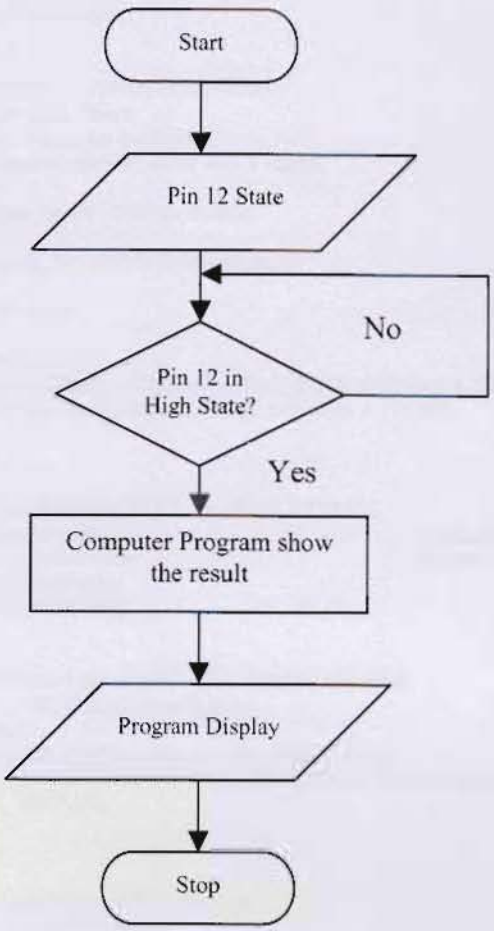
APPENDIX B

Movements and Directions Control Operation of Computer Program




1
2
Button →
Pressed ?
No
Button →
Pressed ?
No
Button •
Pressed ?
No
Yes
Yes
Yes
Sends a byte = 4d
to Parallel Port
Sends a byte = 8d
to Parallel Port
Sends a byte = 0d
to Parallel Port
3
4
5

Object Detection Operation of Computer Program



Computer Program

File name : HoverButton.h

```
LINE  SOURCE CODE
   1
   2  #if !defined(AFX_HOVERBUTTON_H__16C6D980_BD45_11D3_BDA3_00104B133581__INCLUDED_)
   3  #define AFX_HOVERBUTTON_H__16C6D980_BD45_11D3_BDA3_00104B133581__INCLUDED_
   4
   5  #if _MSC_VER > 1000
   6  #pragma once
   7  #endif // _MSC_VER > 1000
   8  // HoverButton.h : header file
   9  //
  10
  11  /////////////////////////////////////////////////////////////////////////
  12  // CHoverButton by Niek Albers
  13  // Thanks to some people for the tooltip.
  14  // A cool CBitmapButton derived class with 3 states,
  15  // Up/Down/Hover.
  16  class CHoverButton : public CBitmapButton
  17  {
  18          DECLARE_DYNAMIC(CHoverButton);
  19
  20          // Construction
  21  public:
  22          CHoverButton();
  23          void SetToolTipText(CString* spText, BOOL bActivate = TRUE);
  24          void SetToolTipText(int nId, BOOL bActivate = TRUE);
  25
  26  // Attributes
  27  protected:
  28          void ActivateTooltip(BOOL bActivate = TRUE);
  29          BOOL m_bHover;                          // indicates if mouse is over the button
  30          CSize m_ButtonSize;                     // width and height of the button
  31          CBitmap mybitmap;
  32          BOOL m_bTracking;
  33
  34  // Overrides
  35          // ClassWizard generated virtual function overrides
  36          //{{AFX_VIRTUAL(CHoverButton)
  37          protected:
  38          virtual BOOL PreTranslateMessage(MSG* pMsg);
  39          virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
  40          //}}AFX_VIRTUAL
  41
  42  // Implementation
  43  public:
  44          BOOL LoadBitmap(UINT bitmapid);
  45          virtual ~CHoverButton();
  46
  47          // Generated message map functions
  48  protected:
  49          CtoolTipCtrl m_ToolTip;
  50          void InitToolTip();
  51          //{{AFX_MSG(CHoverButton)
  52          afx_msg void OnMouseMove(UINT nFlags, CPoint point);
  53          afx_msg LRESULT OnMouseLeave(WPARAM wparam, LPARAM lparam);
  54          afx_msg void OnMouseHover(WPARAM wparam, LPARAM lparam) ;
  55          //}}AFX_MSG
  56
  57          DECLARE_MESSAGE_MAP()
  58  };
  59
  60  /////////////////////////////////////////////////////////////////////////
  61
  62  //{{AFX_INSERT_LOCATION}}
  63  // Microsoft Visual C++ will insert additional declarations immediately before the previous line.
  64
  65  #endif // !defined(AFX_HOVERBUTTON_H__16C6D980_BD45_11D3_BDA3_00104B133581__INCLUDED_)
```

File name : HoverButon.cpp

LINE	SOURCE CODE

```
1   // HoverButton.cpp : implementation file
2   //
3
4   #include "stdafx.h"
5   #include "HoverButton.h"
6
7   #ifdef _DEBUG
8   #define new DEBUG_NEW
9   #undef THIS_FILE
10  static char THIS_FILE[] = __FILE__;
11  #endif
12
13  /////////////////////////////////////////////////////////////////////////
14  // ChoverButton
15
16  CHoverButton::CHoverButton()
17  {
18          m_bHover = FALSE;
19          m_bTracking = FALSE;
20  }
21  CHoverButton::~CHoverButton()
22  {
23  }
24  IMPLEMENT_DYNAMIC(CHoverButton, CBitmapButton)
25
26  BEGIN_MESSAGE_MAP(CHoverButton, CBitmapButton)
27          //{{AFX_MSG_MAP(CHoverButton)
28          ON_WM_MOUSEMOVE()
29  ON_MESSAGE(WM_MOUSELEAVE, OnMouseLeave)
30  ON_MESSAGE(WM_MOUSEHOVER, OnMouseHover)
31          //}}AFX_MSG_MAP
32  END_MESSAGE_MAP()
33
34  /////////////////////////////////////////////////////////////////////////
35   //         CHoverButton message handlers
36
37  void CHoverButton::OnMouseMove(UINT nFlags, CPoint point)
38  {
39          //         TODO: Add your message handler code here and/or call default
40
41          if (!m_bTracking)
42          {
43                  TRACKMOUSEEVENT tme;
44                  tme.cbSize = sizeof(tme);
45                  tme.hwndTrack = m_hWnd;
46                  tme.dwFlags = TME_LEAVE|TME_HOVER;
47                  tme.dwHoverTime = 1;
48                  m_bTracking = _TrackMouseEvent(&tme);
49          }
50          CBitmapButton::OnMouseMove(nFlags, point);
51  }
52
53  BOOL CHoverButton::PreTranslateMessage(MSG* pMsg)
54  {
55          // TODO: Add your specialized code here and/or call the base class
56          InitToolTip();
57          m_ToolTip.RelayEvent(pMsg);
58          return CButton::PreTranslateMessage(pMsg);
59  }
60
61  // Set the tooltip with a string resource
62  void CHoverButton::SetToolTipText(int nId, BOOL bActivate)
63  {
64          CString sText;
65
66          // load string resource
67          sText.LoadString(nId);
68          // If string resource is not empty
69          if (sText.IsEmpty() == FALSE) SetToolTipText(&sText, bActivate);
70
71  }
```

```
72
73      // Set the tooltip with a Cstring
74      void CHoverButton::SetToolTipText(CString *spText, BOOL bActivate)
75      {
76              // We cannot accept NULL pointer
77              if (spText == NULL) return;
78
79              // Initialize ToolTip
80              InitToolTip();
81
82              // If there is no tooltip defined then add it
83              if (m_ToolTip.GetToolCount() == 0)
84              {
85                      CRect rectBtn;
86                      GetClientRect(rectBtn);
87                      m_ToolTip.AddTool(this, (LPCTSTR)*spText, rectBtn, 1);
88              }
89
90              // Set text for tooltip
91              m_ToolTip.UpdateTipText((LPCTSTR)*spText, this, 1);
92              m_ToolTip.Activate(bActivate);
93      }
94
95      void CHoverButton::InitToolTip()
96      {
97              if (m_ToolTip.m_hWnd == NULL)
98              {
99                      // Create ToolTip control
100                     m_ToolTip.Create(this);
101                     // Create inactive
102                     m_ToolTip.Activate(FALSE);
103             }
104     } // End of InitToolTip
105
106     // Activate the tooltip
107     void CHoverButton::ActivateTooltip(BOOL bActivate)
108     {
109             // If there is no tooltip then do nothing
110             if (m_ToolTip.GetToolCount() == 0) return;
111
112             // Activate tooltip
113             m_ToolTip.Activate(bActivate);
114     } // End of EnableTooltip
115
116     void CHoverButton::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
117     {
118             // TODO: Add your code to draw the specified item
119
120             CDC *mydc=CDC::FromHandle(lpDrawItemStruct->hDC);
121
122             CDC * pMemDC = new CDC;
123             pMemDC -> CreateCompatibleDC(mydc);
124
125             CBitmap * pOldBitmap;
126             pOldBitmap = pMemDC -> SelectObject(&mybitmap);
127
128             CPoint point(0,0);
129
130             if(lpDrawItemStruct->itemState & ODS_SELECTED)
131             {
132                     Mydc-
    >BitBlt(0,0,m_ButtonSize.cx,m_ButtonSize.cy,pMemDC,m_ButtonSize.cx,0,SRCCOPY);
133             }
134             Else
135             {
136                     if(m_bHover)
137                     {
138                             mydc->BitBlt(0,0,m_ButtonSize.cx,m_ButtonSize.cy,pMemDC,
                            m_ButtonSize.cx*2,0,SRCCOPY);
139                     }else
140                     {
141                             mydc->BitBlt(0,0,m_ButtonSize.cx,m_ButtonSize.cy,pMemDC,0,0,SRCCOPY);
142                     }
143             }
144
145             // clean up
```

```
146             pMemDC -> SelectObject(pOldBitmap);
147             delete pMemDC;
148     }
149
150     // Load a bitmap from the resources in the button, the bitmap has to have 3 buttonsstates next to each other:
        Up/Down/Hover
151     BOOL CHoverButton::LoadBitmap(UINT bitmapid)
152     {
153             mybitmap.Attach(::LoadImage(::AfxGetInstanceHandle(),MAKEINTRESOURCE(bitmapid),
                IMAGE_BITMAP,0,0,LR_LOADMAP3DCOLORS));
154             BITMAP  bitmapbits;
155             mybitmap.GetBitmap(&bitmapbits);
157             m_ButtonSize.cy=bitmapbits.bmHeight;
158             m_ButtonSize.cx=bitmapbits.bmWidth/3;
159             SetWindowPos( NULL, 0,0, m_ButtonSize.cx,m_ButtonSize.cy,SWP_NOMOVE  |SWP
                _NOOWNERZORDER  );
160             return TRUE;
161     }
162
163     void CHoverButton::OnMouseHover(WPARAM wparam, LPARAM lparam)
164     {
165             // TODO: Add your message handler code here and/or call default
166             m_bHover=TRUE;
167             Invalidate();
168     }
169
170     LRESULT CHoverButton::OnMouseLeave(WPARAM wparam, LPARAM lparam)
171     {
172             m_bTracking = FALSE;
173             m_bHover=FALSE;
174             Invalidate();
175             return 0;
176     }
```

File name : Computerize Remote Control CarDlg.h

LINE	SOURCE CODE
1	// Computerize Remote Control CarDlg.h : header file
2	//
3	
4	#if !defined(AFX_COMPUTERIZEREMOTECONTROLCARDLG_H__7C0EB2E7_63B8_11D8_B798 _0050BA5F7625__INCLUDED_)
5	#define AFX_COMPUTERIZEREMOTECONTROLCARDLG_H__7C0EB2E7_63B8_11D8_B798 _0050BA5F7625__INCLUDED_
6	
7	#if _MSC_VER > 1000
8	#pragma once
9	#endif // _MSC_VER > 1000
10	
11	#include "HoverButton.h"
12	#include <conio.h>
13	
14	//
15	// CComputerizeRemoteControlCarDlg dialog
16	
17	class CComputerizeRemoteControlCarDlg : public CDialog
18	{
19	// Construction
20	public:
21	CComputerizeRemoteControlCarDlg(CWnd* pParent = NULL); // standard constructor
22	CString text;
23	int count;
24	int count1;
25	int inStatusP;
26	int StatusP;
27	
28	//short _stdcall Inp32(short PortAddress);
29	void _stdcall Out32(short PortAddress, short data);
30	
31	// Dialog Data
32	//{{AFX_DATA(CComputerizeRemoteControlCarDlg)

```
33              enum { IDD = IDD_COMPUTERIZEREMOTECONTROLCAR_DIALOG };
34              CHoverButton        m_BTN_STOP;
35              CHoverButton        m_BTN_RIGHT;
36              CHoverButton        m_BTN_LEFT;
37              CHoverButton        m_BTN_FORWARD;
38              CHoverButton        m_BTN_BACKWARD;
39              //}}AFX_DATA
40
41              // ClassWizard generated virtual function overrides
42              //{{AFX_VIRTUAL(CComputerizeRemoteControlCarDlg)
43              protected:
44              virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV support
45              //}}AFX_VIRTUAL
46
47      // Implementation
48      protected:
49              HICON m_hIcon;
50
51              // Generated message map functions
52              //{{AFX_MSG(CComputerizeRemoteControlCarDlg)
53              virtual BOOL OnInitDialog();
54              afx_msg void OnPaint();
55              afx_msg HCURSOR OnQueryDragIcon();
56              afx_msg void OnBtnBackward();
57              afx_msg void OnBtnForward();
58              afx_msg void OnBtnLeft();
59              afx_msg void OnBtnRight();
60              afx_msg void OnBtnStop();
61              virtual void OnCancel();
62              afx_msg void OnTimer(UINT nIDEvent);
63              //}}AFX_MSG
64              DECLARE_MESSAGE_MAP()
65      };
61
62      //{{AFX_INSERT_LOCATION}}
63      // Microsoft Visual C++ will insert additional declarations immediately before the previous line.
64
65      #endif // !defined(AFX_COMPUTERIZEREMOTECONTROLCARDLG_H__7C0EB2E7_63B8_11D8_B798
        _0050BA5F7625__INCLUDED_)
```

File name : Computerize Remote Control CarDlg.cpp

LINE	SOURCE CODE

```
1       // Computerize Remote Control CarDlg.cpp : implementation file
2       //
3
4       #include "stdafx.h"
5       #include "Computerize Remote Control Car.h"
6       #include "Computerize Remote Control CarDlg.h"
7
8       #ifdef _DEBUG
9       #define new DEBUG_NEW
10      #undef THIS_FILE
11      static char THIS_FILE[] = __FILE__;
12      #endif
13
14      /////////////////////////////////////////////////////////////////////////////
15      // CComputerizeRemoteControlCarDlg dialog
16
17      CComputerizeRemoteControlCarDlg::CComputerizeRemoteControlCarDlg(CWnd* pParent /*=NULL*/)
18              : CDialog(CComputerizeRemoteControlCarDlg::IDD, pParent)
19      {
20              //{{AFX_DATA_INIT(CComputerizeRemoteControlCarDlg)
21              //}}AFX_DATA_INIT
22              // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
23              m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
24      }
25
26      void CComputerizeRemoteControlCarDlg::DoDataExchange(CDataExchange* pDX)
27      {
28              CDialog::DoDataExchange(pDX);
```

```
29              //{{AFX_DATA_MAP(CComputerizeRemoteControlCarDlg)
30              DDX_Control(pDX, IDC_BTN_STOP, m_BTN_STOP);
31              DDX_Control(pDX, IDC_BTN_RIGHT, m_BTN_RIGHT);
32              DDX_Control(pDX, IDC_BTN_LEFT, m_BTN_LEFT);
33              DDX_Control(pDX, IDC_BTN_FORWARD, m_BTN_FORWARD);
34              DDX_Control(pDX, IDC_BTN_BACKWARD, m_BTN_BACKWARD);
35              //}}AFX_DATA_MAP
36      }
37
38      BEGIN_MESSAGE_MAP(CComputerizeRemoteControlCarDlg, CDialog)
39              //{{AFX_MSG_MAP(CComputerizeRemoteControlCarDlg)
40              ON_WM_PAINT()
41              ON_WM_QUERYDRAGICON()
42              ON_BN_CLICKED(IDC_BTN_BACKWARD, OnBtnBackward)
43              ON_BN_CLICKED(IDC_BTN_FORWARD, OnBtnForward)
44              ON_BN_CLICKED(IDC_BTN_LEFT, OnBtnLeft)
45              ON_BN_CLICKED(IDC_BTN_RIGHT, OnBtnRight)
46              ON_BN_CLICKED(IDC_BTN_STOP, OnBtnStop)
47              ON_WM_TIMER()
48              //}}AFX_MSG_MAP
49      END_MESSAGE_MAP()
50
51      /////////////////////////////////////////////////////////////////////////////
52      // CComputerizeRemoteControlCarDlg message handlers
53
54      BOOL CComputerizeRemoteControlCarDlg::OnInitDialog()
55      {
56              CDialog::OnInitDialog();
57
58              // Set the icon for this dialog.  The framework does this automatically
59              //  when the application's main window is not a dialog
60              SetIcon(m_hIcon, TRUE);                         // Set big icon
61              SetIcon(m_hIcon, FALSE);                // Set small icon
62
63              // TODO: Add extra initialization here
64              m_BTN_FORWARD.LoadBitmap(IDB_BTN_FORWARD);
65              text=_T("Forward");
61              m_BTN_FORWARD.SetToolTipText(&text);
62              GetDlgItem( IDC_TEXT_MOVE_FORWARD )->ShowWindow( FALSE );
63
64              //---------------------------------------------------------------
65
66              m_BTN_BACKWARD.LoadBitmap(IDB_BTN_BACKWARD);
67              text=_T("Backward");
68              m_BTN_BACKWARD.SetToolTipText(&text);
69              GetDlgItem( IDC_TEXT_MOVE_BACKWARD )->ShowWindow( FALSE );
70
71              //---------------------------------------------------------------
72
73              m_BTN_RIGHT.LoadBitmap(IDB_BTN_RIGHT);
74              text=_T("Right");
75              m_BTN_RIGHT.SetToolTipText(&text);
76              GetDlgItem( IDC_TEXT_MOVE_RIGHT )->ShowWindow( FALSE );
77
78              //---------------------------------------------------------------
79
80              m_BTN_LEFT.LoadBitmap(IDB_BTN_LEFT);
81              text=_T("Left");
82              m_BTN_LEFT.SetToolTipText(&text);
83              GetDlgItem( IDC_TEXT_MOVE_LEFT )->ShowWindow( FALSE );
84
85              //---------------------------------------------------------------
86
87              m_BTN_STOP.LoadBitmap(IDB_BTN_STOP);
88              text=_T("Stop");
89              m_BTN_STOP.SetToolTipText(&text);
90
91              StatusP = 0;
92              count = 0;
93              count1 = 0;
94
95              SetTimer( 1, 100, NULL );
96
97              return TRUE;  // return TRUE  unless you set the focus to a control
98      }
99
```

```
100	// If you add a minimize button to your dialog, you will need the code below
101	//  to draw the icon.  For MFC applications using the document/view model,
102	//  this is automatically done for you by the framework.
103
104	void CComputerizeRemoteControlCarDlg::OnPaint()
105	{
106		if (IsIconic())
107		{
108			CPaintDC dc(this); // device context for painting
109
110			SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
111
112			// Center icon in client rectangle
113			int cxIcon = GetSystemMetrics(SM_CXICON);
114			int cyIcon = GetSystemMetrics(SM_CYICON);
115			CRect rect;
116			GetClientRect(&rect);
117			int x = (rect.Width() - cxIcon + 1) / 2;
118			int y = (rect.Height() - cyIcon + 1) / 2;
119
120			// Draw the icon
121			dc.DrawIcon(x, y, m_hIcon);
122		}
123		Else
124		{
125			CDialog::OnPaint();
126		}
127	}
128
129	// The system calls this to obtain the cursor to display while the user drags
130	//  the minimized window.
131	HCURSOR CComputerizeRemoteControlCarDlg::OnQueryDragIcon()
132	{
133		return (HCURSOR) m_hIcon;
134	}
135
136	void CComputerizeRemoteControlCarDlg::OnBtnForward()
137	{
138		// TODO: Add your control notification handler code here
139		GetDlgItem( IDC_TEXT_MOVE_FORWARD )->ShowWindow( TRUE );
140		GetDlgItem( IDC_TEXT_MOVE_BACKWARD )->ShowWindow( FALSE );
141		GetDlgItem( IDC_TEXT_MOVE_RIGHT )->ShowWindow( FALSE );
142		GetDlgItem( IDC_TEXT_MOVE_LEFT )->ShowWindow( FALSE );
143		_outpw( 0x0378, 1 );
144	}
145
146	void CComputerizeRemoteControlCarDlg::OnBtnBackward()
147	{
148		// TODO: Add your control notification handler code here
149		GetDlgItem( IDC_TEXT_MOVE_FORWARD )->ShowWindow( FALSE );
150		GetDlgItem( IDC_TEXT_MOVE_BACKWARD )->ShowWindow( TRUE );
151		GetDlgItem( IDC_TEXT_MOVE_RIGHT )->ShowWindow( FALSE );
152		GetDlgItem( IDC_TEXT_MOVE_LEFT )->ShowWindow( FALSE );
153		_outpw( 0x0378, 2 );
154	}
155
156	void CComputerizeRemoteControlCarDlg::OnBtnRight()
157	{
158		// TODO: Add your control notification handler code here
159		GetDlgItem( IDC_TEXT_MOVE_FORWARD )->ShowWindow( FALSE );
160		GetDlgItem( IDC_TEXT_MOVE_BACKWARD )->ShowWindow( FALSE );
161		GetDlgItem( IDC_TEXT_MOVE_RIGHT )->ShowWindow( TRUE );
162		GetDlgItem( IDC_TEXT_MOVE_LEFT )->ShowWindow( FALSE );
163		_outpw( 0x0378, 4 );
164	}
165
166	void CComputerizeRemoteControlCarDlg::OnBtnLeft()
167	{
168		// TODO: Add your control notification handler code here
169		GetDlgItem( IDC_TEXT_MOVE_FORWARD )->ShowWindow( FALSE );
170		GetDlgItem( IDC_TEXT_MOVE_BACKWARD )->ShowWindow( FALSE );
171		GetDlgItem( IDC_TEXT_MOVE_RIGHT )->ShowWindow( FALSE );
172		GetDlgItem( IDC_TEXT_MOVE_LEFT )->ShowWindow( TRUE );
173		_outpw( 0x0378, 8 );
174	}
175
```

```
176	void CComputerizeRemoteControlCarDlg::OnBtnStop()
177	{
178		// TODO: Add your control notification handler code here
179		GetDlgItem( IDC_TEXT_MOVE_FORWARD )->ShowWindow( FALSE );
180		GetDlgItem( IDC_TEXT_MOVE_BACKWARD )->ShowWindow( FALSE );
181		GetDlgItem( IDC_TEXT_MOVE_RIGHT )->ShowWindow( FALSE );
182		GetDlgItem( IDC_TEXT_MOVE_LEFT )->ShowWindow( FALSE );
183		_outpw( 0x0378, 0 );
184	}
185
186	void CComputerizeRemoteControlCarDlg::OnTimer(UINT nIDEvent)
187	{
188		// TODO: Add your message handler code here and/or call default
189		inStatusP = _inpw( 0x0379 );
190		StatusP = inStatusP & 32;
191		StatusP = 32;
192
193		if( StatusP != 32 )
194		{
195			if( count == 0 )
196			{
197				GetDlgItem( IDC_ALERT_PICTURE )->ShowWindow( TRUE );
198				GetDlgItem( IDC_ALERT_TEXT )->ShowWindow( TRUE );
199				count++;
200				count1 = 0;
201			}
202		}
203		Else
204		{
205			if( count1 == 0 )
206			{
207				GetDlgItem( IDC_ALERT_PICTURE )->ShowWindow( FALSE );
208				GetDlgItem( IDC_ALERT_TEXT )->ShowWindow( FALSE );
209				count1++;
210				count = 0;
211			}
212		}
213
214		CDialog::OnTimer(nIDEvent);
215	}
216
217	void CComputerizeRemoteControlCarDlg::OnCancel()
218	{
219		// TODO: Add extra cleanup here
220		_outpw( 0x0378, 0 );
221
222		CDialog::OnCancel();
223	}
```

APPENDIX C

Microcontroller Program

File name : MicroProg.bas

Line	Source Code
1	DEVICE 16F84
2	DEFINE PORTB = 10000000
3	SYMBOL TRIG=B.0
4	SYMBOL OUT=B.1
5	SYMBOL IN=B.7
6	
7	LOOP: SET TRIG
8	SET IN
9	BUTTON IN
10	GOTO ACTION
11	
12	ACTION: Clear TRIG
13	SET OUT
14	DELAYMS (500)
15	Clear OUT
16	GOTO LOOP
17	
18	End

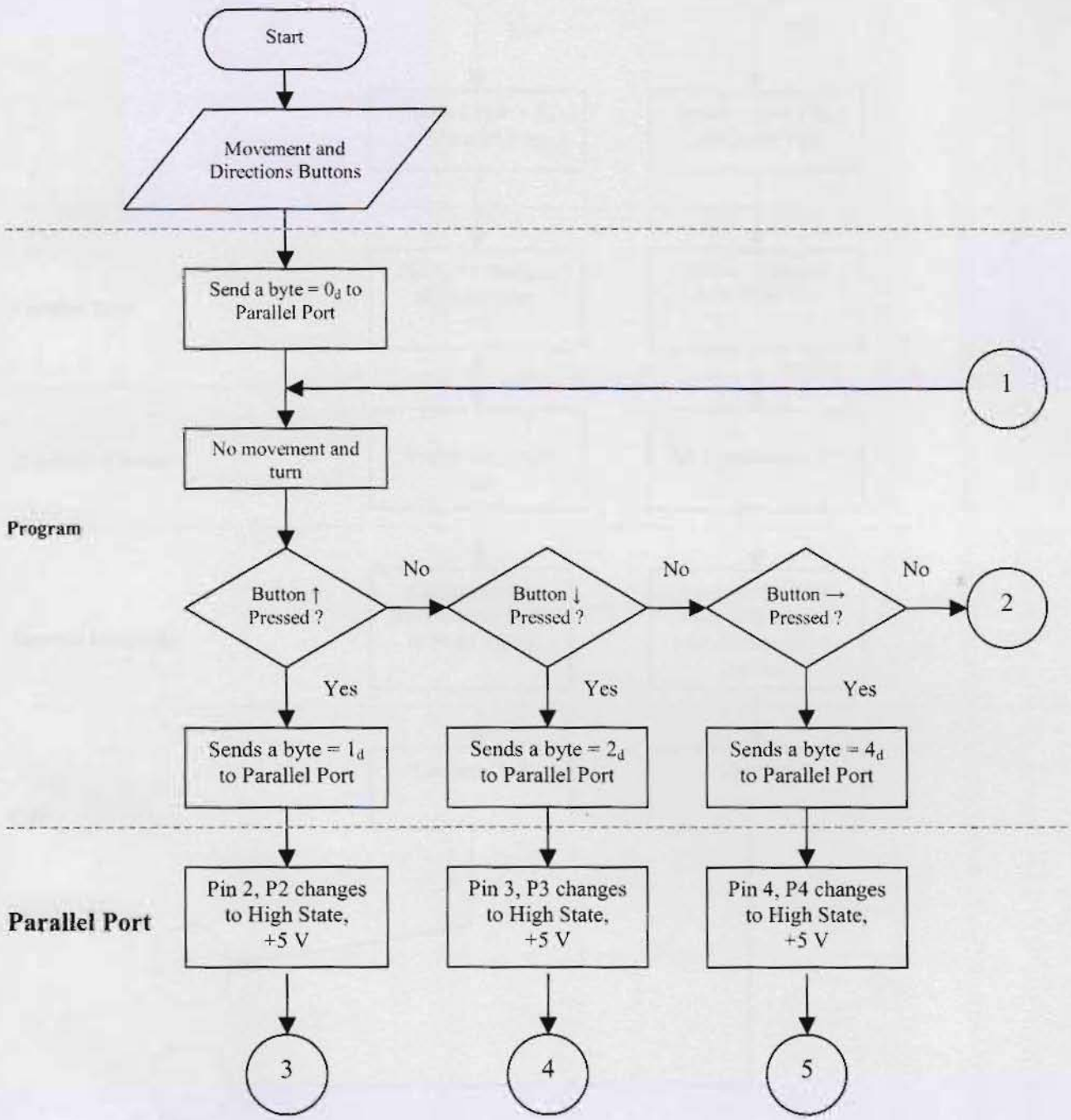
File name : MicroProg.asm

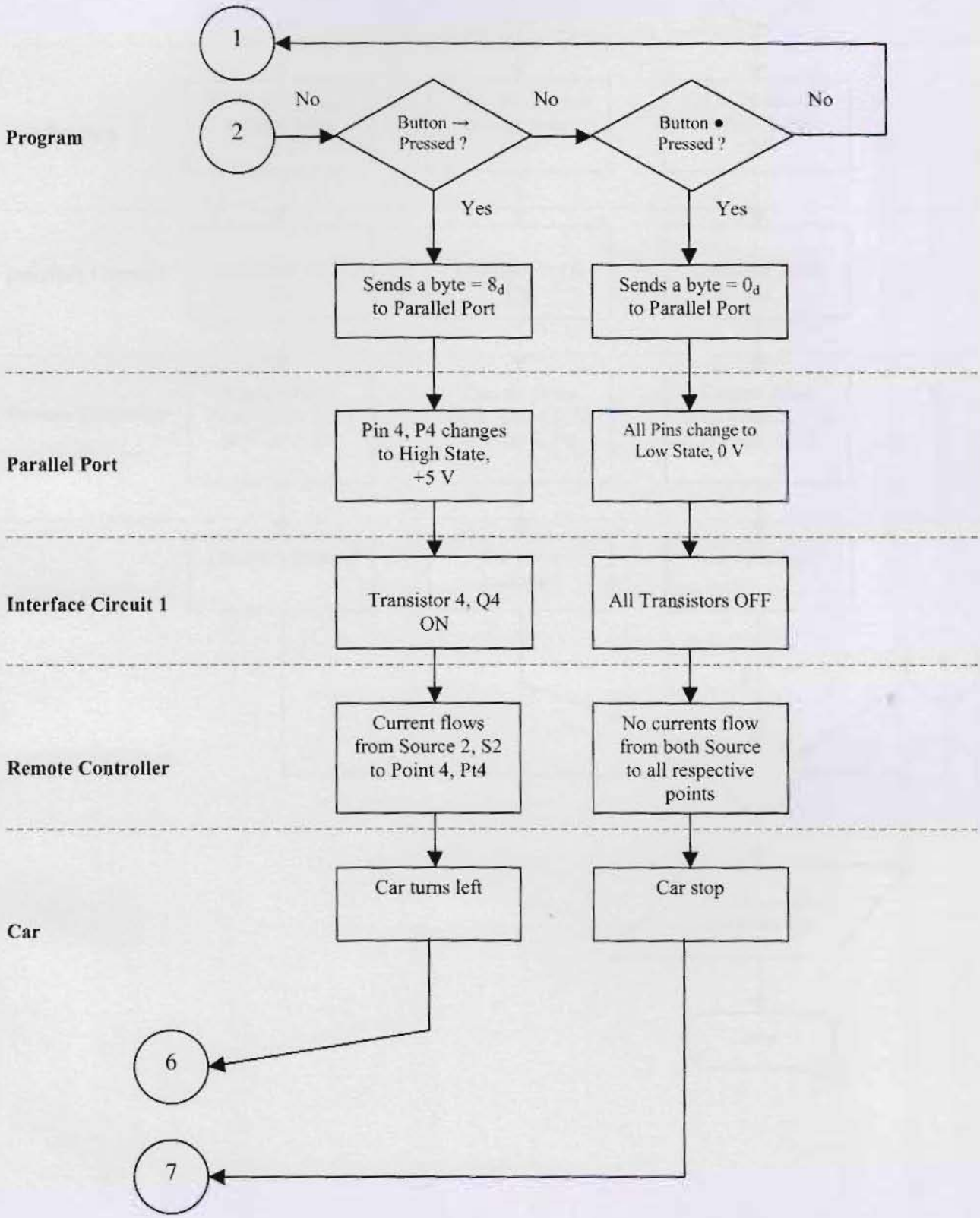
Line	Source Code
1	;----------
2	; (C) Leading Edge Technology Ltd
3	; --- Pic Basic Compiler V4.6 ---
4	; Website: http://LET.cambs.net
5	; EMail: johnmorr@mail.keyworld.net
6	;----------
7	
8	W equ 00
9	F equ 01
10	
11	LIST p=16F84 , r=DEC
12	; ----- DEFINEPORTB=10000000 -----
13	movlw 128

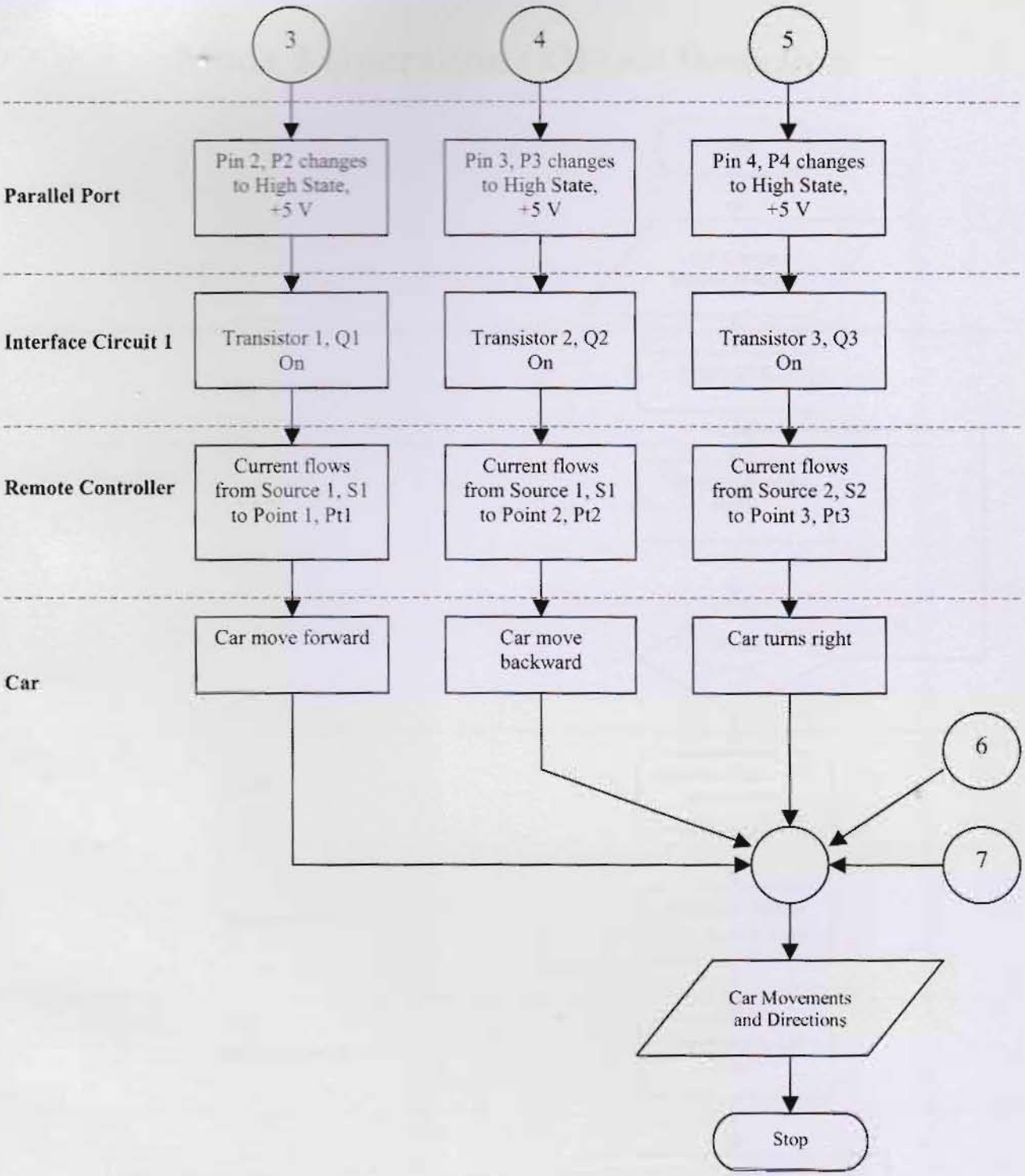
```
14          tris    06
15      LOOP ; ----- LOOP:          SET TRIG -----
16          bsf     06,00
17      ; ----- SET IN -----
18          bsf     06,07
19      ; ----- BUTTON IN -----
20          btfsc   06,07
21          goto    $-1
22          btfss   06,07
23          goto    $-1
24          goto    ACTION
25      ACTION      ; ----- ACTION:       CLEAR TRIG -----
26          bcf     06,00
27      ; ----- SET OUT -----
28          bsf     06,01
29      ; ----- DELAYMS( 500 ) -----
30          movlw       500
31          movwf       13
32          clrf    12
33      pb_lab2
34          nop
35          nop
36          decfsz  12,F
37          goto    pb_lab2
38          decfsz  13,F
39          goto    pb_lab2
40      ; ----- CLEAR OUT -----
41          bcf     06,01
42          goto    LOOP
43          END
```

APPENDIX D

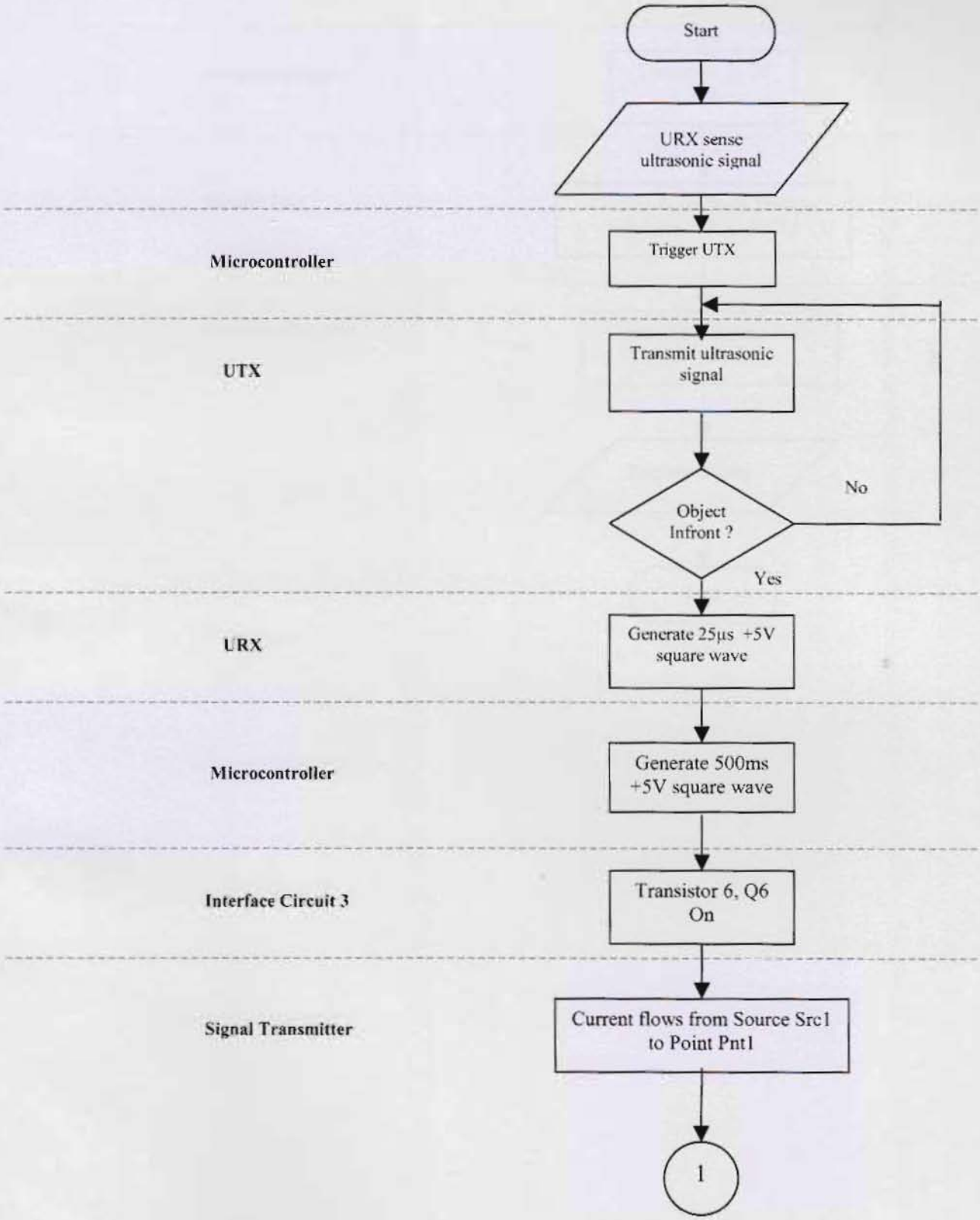
Mode 1 Operation : Movements And Directions Control

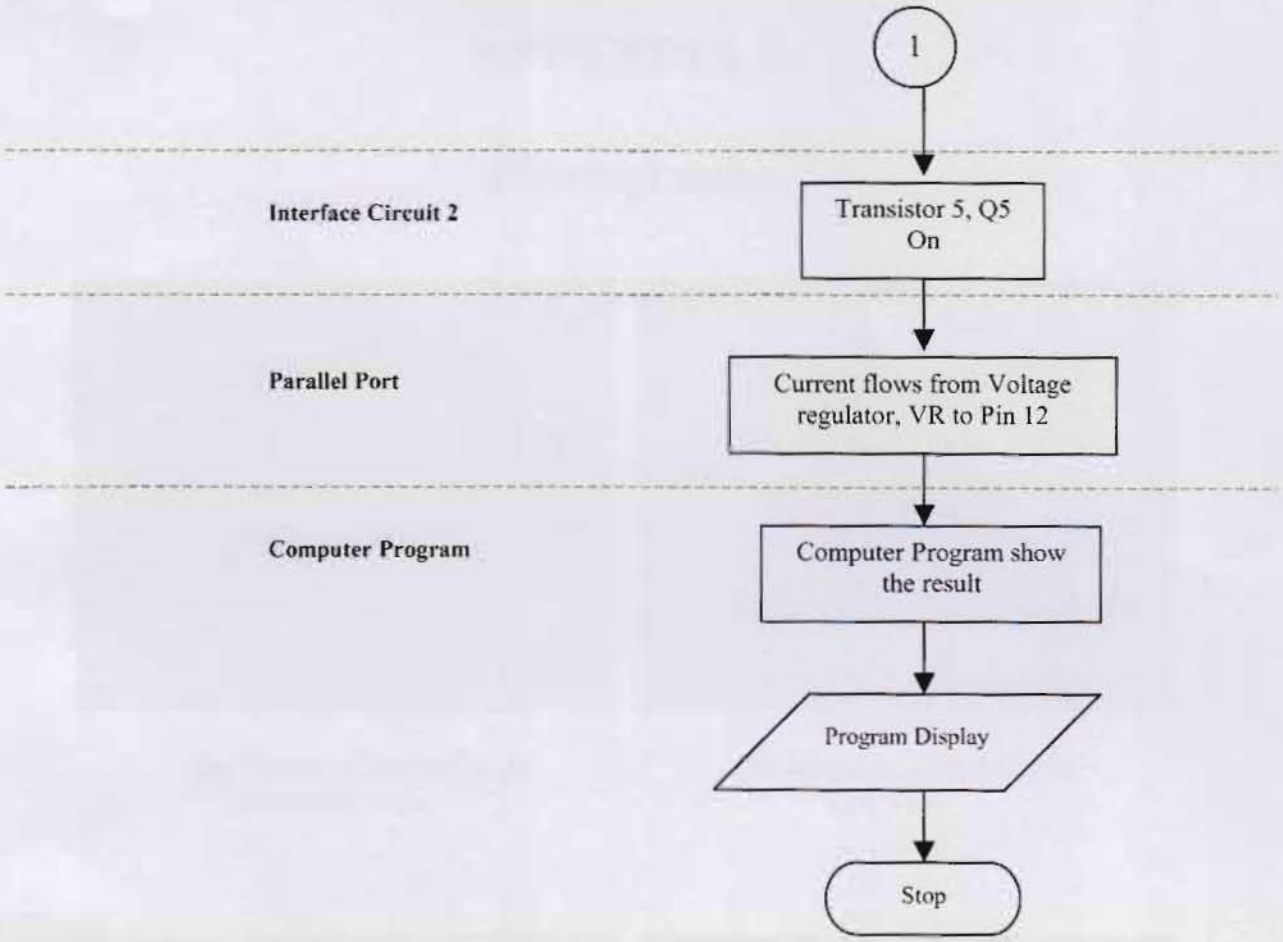



Program
Parallel Port
Interface Circuit 1
Remote Controller
Car
1
2
No
Button → Pressed ?
No
Button • Pressed ?
No
Yes
Yes
Sends a byte = 8d to Parallel Port
Sends a byte = 0d to Parallel Port
Pin 4, P4 changes to High State, +5 V
All Pins change to Low State, 0 V
Transistor 4, Q4 ON
All Transistors OFF
Current flows from Source 2, S2 to Point 4, Pt4
No currents flow from both Source to all respective points
Car turns left
Car stop
6
7


3
4
5
Parallel Port
Pin 2, P2 changes to High State, +5 V
Pin 3, P3 changes to High State, +5 V
Pin 4, P4 changes to High State, +5 V
Interface Circuit 1
Transistor 1, Q1 On
Transistor 2, Q2 On
Transistor 3, Q3 On
Remote Controller
Current flows from Source 1, S1 to Point 1, Pt1
Current flows from Source 1, S1 to Point 2, Pt2
Current flows from Source 2, S2 to Point 3, Pt3
Car
Car move forward
Car move backward
Car turns right
6
7
Car Movements and Directions
Stop

Mode 2 Operation : Object Detection




1
Interface Circuit 2
Transistor 5, Q5
On
Parallel Port
Current flows from Voltage
regulator, VR to Pin 12
Computer Program
Computer Program show
the result
Program Display
Stop

APPENDIX E

Photographs

The Remote Control Car in
Isometric View

The Remote Control Car in
Left View

The Remote Control Car in
Back View

The Remote Control Transmitter
and Data Receiver View