

Palestine Polytechnic University



College of Engineering and Technology
Electrical Engineering Department

Graduation Project

Smart Car Parking System

Project Team

Riham Walled Zalloum Maysa Al_Mohtaseb

Jihan Khamayseh

Project Supervisor

Eng. Mazen Zalloum

Hebron-Palestine

June-2007

جامعة بوليتكنك فلسطين
الخليل – فلسطين
كلية الهندسة والتكنولوجيا
دائرة الهندسة الكهربائية والحاسوب

Smart Car Parking System

ميساء المحتسب

رهام زلوم

جيهان خميسة

بناء على نظام كلية الهندسة والتكنولوجيا وإشراف ومتابعة المشرف المباشر على المشروع و موافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية والحاسوب وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص أنظمة حاسوب.

توقيع المشرف

.....

توقيع اللجنة الممتحنة

.....

.....

.....

توقيع رئيس الدائرة

.....

Dedication

*To our parents who
spent nights and days doing their best
to give us the best...*

*To all students and who
Wish to look for
the future...*

*To who love the knowledge and
Looking for the new
in this world...*

*To who carry candle of science
To light his avenue
of life...*

To our beloved country Palestine...

To all of our friends...

Riham Walled Zalloum

Maysa AL_Mohtaseb

Jihan Khamayseh

Acknowledgments

To our great supervisor, who offered his best for this project to see light through his instructions and advices, Eng. Mazen Zalloum with all his kindness and wisdom we thank him.

We would also like to thank every person who offered anything to success this work; we sincerely believe that this work wouldn't exist without his inspiration. Great thanks to our college for his support and help, and any one who help us in our project.

Abstract

This project is involved in design and implementation of a smart parking system. The system will direct the driver to the available parking spaces in each floor when entering the parking. Then, from the floor the driver will be derived to the parking available slots.

The project will be implemented with the 8051 microcontroller for all the necessary controlled hardware components are used such as sensors display and motor for our system.

هذا المشروع مهتم في تصميم وتطبيق نظام ذكي لإيقاف السيارات. النظام سوف يوجه السائق إلى أماكن الوقوف المتوفرة في كل أرضية عندما تدخل السيارات للإيقاف. ثم من الأرضية يتوجه السائق إلى الإيقاف في احد الأماكن المتوفرة.

المشروع سيتم تطبيقه باستخدام ٨٠٥١ ميكروكونترولر للتحكم بجميع الأجزاء الأخرى التي سوف يتم استخدامها بالمشروع مثل المجسات، وأجهزة العرض، والموتور.

TABLE OF CONTENTS

<u>DEDICATION</u>	<u>III</u>
<u>ACKNOWLEDGMENT</u>	<u>IV</u>
<u>ABSTRACT</u>	<u>V</u>
<u>TABLE OF CONTENTS</u>	<u>ERROR! BOOKMARK NOT DEFINED.</u>
<u>LIST OF TABLES</u>	<u>XI</u>
<u>LIST OF FIGURES</u>	<u>XII</u>
<u>CHAPTER ONE</u>	<u>1</u>
<u>INTRODUCTION</u>	<u>1</u>
1.1 Preface..	2
1.2 Project Importance.....	3
1.3 Review of Literature.....	4
1.4 Project Scheduling.....	5
1.5 Project Cost	7
1.6 Road Map.....	8
<u>CHAPTER TWO</u>	<u>11</u>
<u>THEORETICAL BACKGROUND</u>	<u>11</u>
2.1 Preface.....	11
2.2 Project Components	11
2.2.1 8051 Microcontroller.....	12
2.2.1.1 Introduction to Microcontroller	12
2.2.1.2 Advantages of a Microcontroller.....	13
2.2.1.3 The Intel 8051.....	14
2.2.1.3.1 Why we use choose 8051 microcontroller board.....	16
2.2.1.3.2 Programmable Peripheral Interface (PPI).	16
2.2.2 Sensor (Switch).....	17

2.2.3 Motor.....	18
2.2.4 LCD.....	19
2.2.5 Gate.....	21
2.2.6 Spring.....	21
2.2.7 Force on Spring.....	22
2.2.8 General Parking Diagram.....	22
<u>CHAPTER THREE</u>	25
<u>DESIGN CONCEPTS</u>	25
3.1 Project Objectives.....	26
3.2 Project Main Hardware Components.....	26
3.3 General Block Diagram.....	27
3.3.1 Floor Subsystems.....	28
3.3.2 Entrance System	28
3.3.3 8051 Microcontroller System	29
3.4 Interfacing	30
3.5 Motors	31
3.6 How System Work	32
3.6.1 Microcontroller with Switch, LCDs and Motors.....	32
3.6.2 The Switch with Microcontroller.....	35
3.6.3 The LCDs with Microcontroller.....	36
3.6.4 The Motor with Microcontroller.....	36
3.6.5 The Spring with Switch.....	37
3.6.6 The Gate with The Motor.....	37
<u>CHAPTER FOUR</u>	36
<u>HARDWARE SYSTEM DESIGN</u>	38
4.1 Preface.....	39
4.2 The Units Design.....	39

4.2.1 Sensors (Interlocking Push Buttons).....	40
4.2.2 DC_motor and H_Bridges	42
4.2.2.1 LCDs	42
4.2.4 Control Unit.....	41
4.2.5 Serial Ports	44
4.2.4.1.1 Pins and Wires.....	45
4.2.4.1.2 Data Flow.....	46
4.2.4.2 Controlling the Motor.....	47
4.2.4.3 Electrolytic Capacitor.....	48
4.2.4.4 (74244) Buffer	48
4.2.5 Over All System Unit (Application Unit).....	49
4.2.5.1 Interfacing Circuit	49
4.2.5.1.1 Interfacing Sensors with Microcontroller (8051) board	50
4.2.5.1.2 Microcontroller 8051 Interfacing with LCD Display.....	51
4.2.5.1.3 Microcontroller 8051 Interfacing with Motors.....	53
4.2.5.1.4 Microcontroller System Interface	54
4.2.6 Parallel Port	55
4.2.6.1 Hardware.....	56
4.2.6.2 Parallel Port Registers.....	57
4.2.6.3 Where this Registers	58
4.3 Overall System Design	59

CHAPTER FIVE **62**

SOFTWARE SYSTEM DESIGN **62**

5.1 Preface.....	63
5.2 Software Requirement.....	63
5.2.1 Win 95/98/Me/NT/2000/XP Compatible Software	63
5.2.2 HyperTerminal.....	64
5.2.3 Standard Serial Cable (Straight Through).....	67

5.2.4 Assembler or 'C' Compiler, Usually As31 or SDCC.....	68
5.2.5 Text Editor Program.....	70
5.3 Function Description.....	70
5.3.1 To Use PPI set Ports and System Parameters.....	70
5.3.2 Timer Initialization.....	71
5.3.2.1 Code for Enable Timer	72
5.3.3 Display Data on LCD.....	72
5.4 Serial Interfacing between the Microcontroller and PC.....	73
5.5 General System Flowchart.....	73
5.6 System Operational Flowchart	75
5.6.1 Sensors of Gates Flowchart	75
5.6.2 Floor One Flowchart	77
5.6.3 Floor Two Flowchart	79
5.6.4 Main LCD Flowchart	81
5.6.5 Motor1 and Motor2 Flowchart	82
5.7 Algorithms and Pseudocode	84
<u>CHAPTER SIX</u>	101
<u>IMPLEMENTATION AND TESTING</u>	101
6.1 Preface	102
6.2 Implementation	102
6.3 Testing	103
6.3.1 Testing 8051 Downloading Programs.....	103
6.3.2 Motors and H-Bridge Testing.....	104
6.3.2.1 Option One Using C Language for Motor Testing	106
6.3.2.2 Option Two Using VB.net for Motor Testing	107
6.3.3 Switches Testing.....	107
6.3.3.1 Option One Using C Language for Switching Testing.....	108
6.3.3.2 Option Two Using VB.net for Switching Testing	108

6.3.4 LCDs Testing	109
6.3.4.1 Option One Using C Language for LCD Testing	109
6.3.4.2 Option Two Using VB.net for LCD Testing	109
6.3.5 Access the Parallel Port Using VB.net	109
6.3.5.1 Testing Output Ports in VB.net	109
6.3.5.2 Testing Input Port in VB.net	110
6.4 Implementation and Testing for Integrated System	111
<u>CHAPTER SEVEN</u>	112
<u>CONCLOUSIONS AND FUTURE WORK</u>	112
7.1 Preface.....	113
7.2 Conclusions.....	114
7.2.1 Problems.....	115
7.2.1.1 Hardware Problems.....	115
7.2.1.1 Software Problems.....	115
7.3 Future Work.....	116
<u>REFERENCES</u>	117
<u>APPENDICES</u>	118

LIST OF TABLES

TABLE	PAGE
TABLE 1.1: PROJECT ACTIVITY BAR CHART (FIRST SEMESTER)	6
TABLE 1.2 PROJECT ACTIVITY BAR CHART (SECOND SEMESTER).....	6
TABLE 4.1: SERIAL PORT PINS.....	46
TABLE 4.2: OPERATION OF THE DC MOTOR DRIVING CIRCUIT	47
TABLE 4.3: CONTROL WORD OF THE PPI	54
TABLE 4.4: PARALLEL PORT SIGNAL LINE	57
TABLE 4.5 REGISTER ADDRESSES OF LPT1 AND LPT2:	58

LIST OF FIGURES

FIGURE	PAGE
FIGURE 2.1: 8051.....	13
FIGURE 2.2: THE 8051 MICROCONTROLLER KIT	16
FIGURE 2.3: SWITCH	17
FIGURE 2.4 SWITCH SENSOR	18
FIGURE 2.5: MOTOR	19
FIGURE 2.6: LCD DISPLAY	20
FIGURE 2.7: THE GATE	21
FIGURE 2.8: THE SPRINGS	22
FIGURE 2.9: THE FIRST FLOOR	23
FIGURE 2.10: THE SECOND FLOOR	24
FIGURE 3.1: GENERAL BLOCK DIAGRAM	28
FIGURE 3.2: FLOOR SUBSYSTEM	28
FIGURE 3.3: ENTRANCE SYSTEM	29
FIGURE 3.4: THE 8051 MICROCONTROLLER	30
FIGURE 3.5: INTERFACING MAIN SYSTEM	31
FIGURE 3.6: INTERFACING FLOOR 1.....	31
FIGURE 3.7: DC MOTOR	32
FIGURE 4.1: SWITCH.....	40
FIGURE 4.2: SERIAL PORT CABLE	44
FIGURE 4.3: SERIAL PORT PINS	45
FIGURE 4.4: PC COM PORT	46
FIGURE 4.5: DC MOTOR CIRCUIT	47
FIGURE 4.6: SINGLE CAPACITOR	48
FIGURE 4.7: RESISTORS	48
FIGURE 4.8: 74244 BUFFER	49
FIGURE 4.9: INTERFACING FLOORS SENSORS WITH 8051 MICROCONTROLLER	50

FIGURE 4.10: INTERFACING MAIN ENTRANCE AND EXIT GATE SENSORS WITH 8051 MICROCONTROLLER	51
FIGURE 4.11: INTERFACING LCDS WITH 8051 MICROCONTROLLER	52
FIGURE 4.12: INTERFACING MOTORS WITH 8051 MICROCONTROLLER	53
FIGURE 4.13: PARALLEL PORT REGISTER	56
FIGURE 4.14: THE SYSTEM DESIGN CIRCUIT 1.....	60
FIGURE 4.15: THE SYSTEM DESIGN CIRCUIT 2.....	61
FIGURE 5.1: CONNECTION DESCRIPTION SCREEN.....	64
FIGURE 5.2: CONNECT TO SCREEN	65
FIGURE 5.3: COM1 PROPERTIES	66
FIGURE 5.4: DIAL UP SCREEN	67
FIGURE 5.5: MS_DOS	69
FIGURE 5.6: GENERAL FLOWCHART	74
FIGURE 5.7: SENSORS OF GATES FLOW CHART.....	76
FIGURE 5.8: FLOOR ONE SENSORS FLOW CHART	78
FIGURE 5.9: FLOOR TWO SENSORS FLOW CHART	80
FIGURE 5.10: MAIN LCD FLOW CHART.....	81
FIGURE 5.11: MOTOR1 FLOW CHART	83
FIGURE 5.12: MOTOR2 FLOW CHART	83
FIGURE 6.1: PORT TESTING EXAMPLE (LEDS)	103
FIGURE 6.2: MOTOR CIRCUIT	106
FIGURE 6.3A: SWITCH CIRCUIT OFF	107
FIGURE 6.3B: SWITCH CIRCUIT ON	108
FIGURE 6.4: LCD TESTING	109
FIGURE 6.5: OUTPUT PORT TESTING	110
FIGURE 6.6: INPUT PORT TESTING	110

1

Introduction

- 1.1. Preface
- 1.2. Project Importance
- 1.3. Review of Literature
- 1.4. Project Scheduling
- 1.5. Project Cost
- 1.6. Road Map

Chapter One

Introduction

1.1 Preface

In this chapter introduces the general idea of the project, importance and discuss some of the related projects.

The project smart car parking system is a microcontroller based smart parking system. Usually, in large parking areas we need to search through the whole place for space and then park the car. Our project aims to overcome this trouble. We want that some sensors should be placed at space for each car. These sensors will be connected to screen like LCDs through the microcontroller at the parking entrance via microcontroller. The display would show whether or not space is there for more cars, and if there is any, what the exact location of that space is. This would be carried out by programming through the microcontroller.

So, the project take about general parking for all the people, consists of two floors, and in each one the park has a place form which the car to enter the park and another to leave it, the output of the system is displayed on LCDs the free places for the cars that enter the park, we use for this push button switch, we will connect them to the microcontroller to get the number of car in the park.

There will be a screen (LCD) on the place from which the car enter show if the park is full or have places in it for another car, also show the number of empty places and which places in each floor is empty, so the person can go to any empty place and then put his car on it.

On each place in each car park we will have sensors (switch) that show if the place is empty or available, the screen of that park show that.

The system is consists of the microcontroller, the sensors, and the displays (LCDs), all of them will be connected to each other in the project to have a complete smart car parking system.

1.2 Project Importance

The importance of this project that the smart parking system will be more comfortable, easier to locate empty parking spaces, without driving around the park. Also, a chargeable system at the entrance will be implemented as a final stage of the project. The system easily can be expanded to accommodate more floors as each of parking floor is treated separately and report only the number of the available spaces for parking.

The importance of the project view on more security, safe and more intelligent parking that not require employee to work in it and the customers will have enough information about how to use the park and in which space, then he chose the place in the park he will go to and know the exact location of that space.

This help parking especially large parking to use it in very easy and perfect way and it will encourage the customers to put their cars in it. We don't need to search through the whole place for space and then park the car. Our project aims to overcome this.

1.3 Review of Literature

Transit-based smart parking in the San Francisco, bay area: an assessment of user demand and behavioral effects.

The parking guidance information component of this system uses loop detectors to monitor available parking spaces in facilities and then transmits messages via VMS signs. The software uses historical data by time to predict parking facility occupancy status. Planned improvements include forecasts of available metered on-street parking and a parking reservation system via the Internet, phone, or in-car terminal.

Another example of an advanced smart parking system is the Frottmaning U-Bahn station park-and-ride lot (with 1,270 parking spaces) in Munich, Germany, on the A9 Autobahn.

This system boasts three dynamic VMS screens along the nearby highway, which indicate the number of parking spaces, real-time transit schedules, and traffic news. Once motorists enter the parking facility, they are guided to the closest empty parking space by a real-time surveillance and control system. The smart “directing” system uses laser-scan detectors at entrance and exit lanes and ultrasound detectors at each parking space.

Smart parking management systems that provide real-time information to motorists about the number of available parking spaces in park-and-ride lots, the departure time of the next train, and downstream roadway traffic conditions (e.g., accidents and delays) have been implemented in many cities in Europe and Japan. More recently, several transits based smart parking management programs have been proposed in the U.S.

1.4 Project Scheduling

The project activities here depend on each other, so the task durations and dependencies are the following:-

T1: Preparing to the project: here we introduce to start in the project and discuss with the advisor to initialize the project, and preparing the group and evaluate the project tasks and levels. At this period choosing project.

T2: The project searching and analysis: at this period we start the first step to search and analysis the project and allocate information and data about the project levels and sublevels, tasks and subtasks, there are many resources to searching and analyzing the concepts.

T3: The project requirements analysis: the project has many equipments must be provided and explained to implement the final project to achieve the system requirements. The system has a hardware and software requirements which must be achieved through the prototype and final presentation.

T4: Introduction to project and study the 8051 microcontroller system.

T5: Study and find the type of sensor we want and other hardware we require it.

T6: Theory background.

T7: Design concept.

T8: Writing the software and the implementation of the project.

1.5 Project Cost

We purchase as a work group all of its equipments to complete the project architecting and designing.

The project need both of hardware equipments and software programs that runs on the microcontroller, so we will purchasing all needed electronic components and parts and software programs.

- The Hardware Components, there are many electrical Chips and equipments have to be provided:-
 1. Resistors.
 2. Capacitors.
 3. Diodes.
 4. Wires (10\$).
 5. 2 Optocouplers (2\$).
 6. 2 DC Motor (100\$).
 7. 30 sensor (micro-switch) (60\$).
 8. 8051 microcontroller development kit (300\$).
 9. 3 LCDs (9000\$).
 10. 3 PPI (15\$).

- Software programs:-
 1. Assembly, C Program it cost and visual Basic (700\$).
 2. Windows 98version or more it cost (500\$).
 3. Microsoft PowerPoint, word and Visio (300\$).

- **Human cost:-**

The team of the project consists of three students, work in 27 week, 70 hour at a day and the work's hour costs 10\$. So, the human cost equal $(3 \times 27 \times 70 \times 10 = 40500\$)$.

The total cost: The total cost contains the hardware equipments and software programs, the total cost reach about approximately (51487\$).

Note: the Electronic equipments price is varied depending on the component efficiency and the purchasing source and as the performance of the project increase the cost increase.

1.6 Road Map

Report consists of seven chapters; the following is a brief description of the topics that are covered in each chapter.

Chapter 1: Introduction

This chapter present general idea about the project and its importance, and also literature review, system requirement, group dependency, project scheduling, estimated cost.

Chapter 2: Theoretical Back Ground

This chapter talks in more details about the basic component used in the project and theoretical back ground.

Chapter 3: Design Concepts

This chapter details the design concepts, introduces project objectives, shows the general block diagram of the system and explains how system works.

Chapter 4: Hardware System Design

This chapter presented out lines formal procedure for design, discuss design options and justify those chosen for the project.

Chapter 5: Software System Design

This chapter handles the software related to our system, depicts flowcharts about system operation.

Chapter Six: System Implementation and Testing

This chapter includes the implementation phase with the testing of these phase. General hardware and software component tested and shown in this chapter.

Chapter Seven: Conclusion and Future Work

This chapter will provides the conclusions that will be concluded after working the system, and suggestion for future work.

2

Theoretical Background

2.1 Preface

2.2 Project Component.

Chapter Two

Theoretical Background

2.1 Preface

This working relates to a general smart parking system and in particular an intelligent and electrical parking system based on working many objects in that park without any human employee in it. Such as, to open the gate of the park when any customer want to enter, and to know exactly where he will park his car from LCDs at the entrance.

Gate has a motor to open and close it, LCDs to tell the customer which floor, number of available spaces and what the exact location to park the car. So, we will use a microcontroller, many LCDs, sensors, motor for each gate, and other hardware devices.

This chapter will illustrate theoretical background for our project applications in general and for each component in particularly, and how each component communicate with other components, like the communication between the microcontroller, sensors and LCDs.

2.2. Project Components

As we have mentioned in chapter one before, this project is fully constructed over a smart car parking, that behave in an intelligent system, we have used many hardware devices, the input comes from sensor and the output to the gate at the entrance and at exit and to the LCDs .The basic unit is that the unit to control all of

the application that we will use, it will be the microcontroller. One need of a microcontroller is to perform and control all of that application that will be needed in our smart car parking system.

In the following sections we will give an explanation of each component (hardware device) that we will use in smart car parking system.

2.2.1. 8051 Microcontroller

2.2.1.1. Introduction to Microcontroller

The microprocessor is the little (single) chip is the heart of a computer; it does all the computations like adding, subtracting, multiplying, and dividing.

The microprocessor might be a Pentium, a K6, a PowerPC, a Sparc or any of the many other brands and types of microprocessors, but they all do approximately the same thing in approximately the same way.

A microcontroller is an entire computer manufactured on a single chip. The I/O and a memory subsystems contained in a microcontroller specialize these devices so that they can be interfaced with hardware and control functions of the applications. Since microcontrollers are powerful digital processor, the degree of control and programmability they provide significantly enhances the effectiveness of the application. Microcontrollers usually dedicated devices embedded within an application.

Single-chip microcomputer indicates that the complete microcomputer system. Microcontrollers are capable of storing and running the program that was written, compiled and downloaded into it. The main parts of a microcontroller in

generally consist of the Central Processing Unit (CPU), Read Only Memory (ROM), Random Access Memory (RAM), input/output lines, serial and parallel ports, registers, peripherals such as timers and watchdog circuits and signal conversion circuits, counters, digital (A/D) converter and others...

P1.0	1	40	Vcc
P1.1	2	39	P0.0/AD0
P1.2	3	38	P0.1/AD1
P1.3	4	37	P0.2/AD2
P1.4	5	36	P0.3/AD3
P1.5	6	35	P0.4/AD4
P1.6	7	34	P0.5/AD5
P1.7	8	33	P0.6/AD6
RESET	9	32	P0.7/AD7
RXD/P3.0	10	31	EA/Vpp
TXD/P3.1	11	30	ALE/PROG
INT0/P3.2	12	29	PSEN
INT1/P3.3	13	28	P2.7/AD15
T0/P3.4	14	27	P2.6/AD14
T1/P3.5	15	26	P2.5/AD13
WR/P3.6	16	25	P2.4/AD12
RD/P3.7	17	24	P2.3/AD11
XTAL2	18	23	P2.2/AD10
XTAL1	19	22	P2.1/AD9
PDIP Vss	20	21	P2.0/AD8

Figure (2.1): 8051

2.2.1.2. Advantages of a Microcontroller

A microcontroller is a computer-on-a-chip used to control electronic devices. It is a type of microprocessor emphasizing self-sufficiency and cost-effectiveness, in contrast to a general-purpose microprocessor. A typical microcontroller contains all the memory and interfaces needed for a simple application, whereas a general purpose microprocessor requires additional chips to provide these functions.

A microcontroller is a single integrated circuit. It can store and run unique programs very flexible and capability to carry out mathematical and logic functions allows it to imitate complicated logic and electronic circuits.

2.2.1.3 The Intel 8051

This section is specializing for 8051 and we will present its details. Next, different tasks related to this microcontroller, plus the problems faced and how they were solved will follow.

The 8051 is the first microcontroller of the family introduced by Intel Corporation at the end of the 1970s. The 8051 family are 8-bit controllers capable of addressing 64K of program memory and separate 64K of data memory.

In our project we will use the 8051 development board which the 8051 development board provides an easy and low-cost way to develop 8051 based microcontroller projects, without purchasing any other equipment, such as IC programmers or emulators. The board comes loaded with PAULMON2, (is available as assembly language source code or ready-to-run binary object code). This provides a simple menu-based system that enables to download code into the RAM or Flash ROM on the board. So the board will then run application instead of booting into the PAULMON2 menu system. A jumper is provided, should you need to erase the Flash ROM to make any changes.

The board features two 82C55 chips that provide 50 I/O lines and 8 LEDs, in addition to the 10 lines from the 8051's port #1, and the 8051's bus lines. A second serial connector is available, with a simple switching circuit, to make it easier to develop applications.

8051 Features

- Standard 87C52 CPU clocked at 22.1184 MHz.

- 50 I/O lines, All I/O lines are clearly labeled and available at the edge of the prototype construction area.
- 32k SRAM, program variables and code (24k usable for code download).
- 30k Flash ROM, non-volatile program storage and data logging.
- High speed baud rates: 115200, 75600, 38400, etc. All standard baud rates are supported (except 300 baud).
- Display port, works with standard character-based LCDs.
- Eight LEDs, controlled by 8 dedicated I/O lines (not shared with the 50 I/O lines).
- Bus expansion with 4 chip select signals, for adding UARTs, A/D converters and other bus-based peripheral chips.
- Unregulated, polarity-protected DC voltage input with 2 position terminal block.
- PAULMON2 monitor for easy code development without additional equipment.
- LCD Display Port
- The 8051 development board's LCD port provides the 14 signals needed for standard character based LCD modules. A 20x2 display is available from PJRC.

All of these features make it ideal for more advanced level A/D applications in automotive, industrial, appliances and consumer applications. Also we can collect many of our needs for the project in one chip, since we need memory to hold digitized identification data, and modulation to modulate the carrier signal with the identification data.

2.2.1.3.1 Why we use choose 8051 microcontroller board?

The 8051 development board provides an easy-to-use to develop 8051 based microcontroller projects, without purchasing any other special equipment, such as IC programmers or emulators.

We choose 8051 after search and comparison between it and other microcontrollers that can be used in our project.

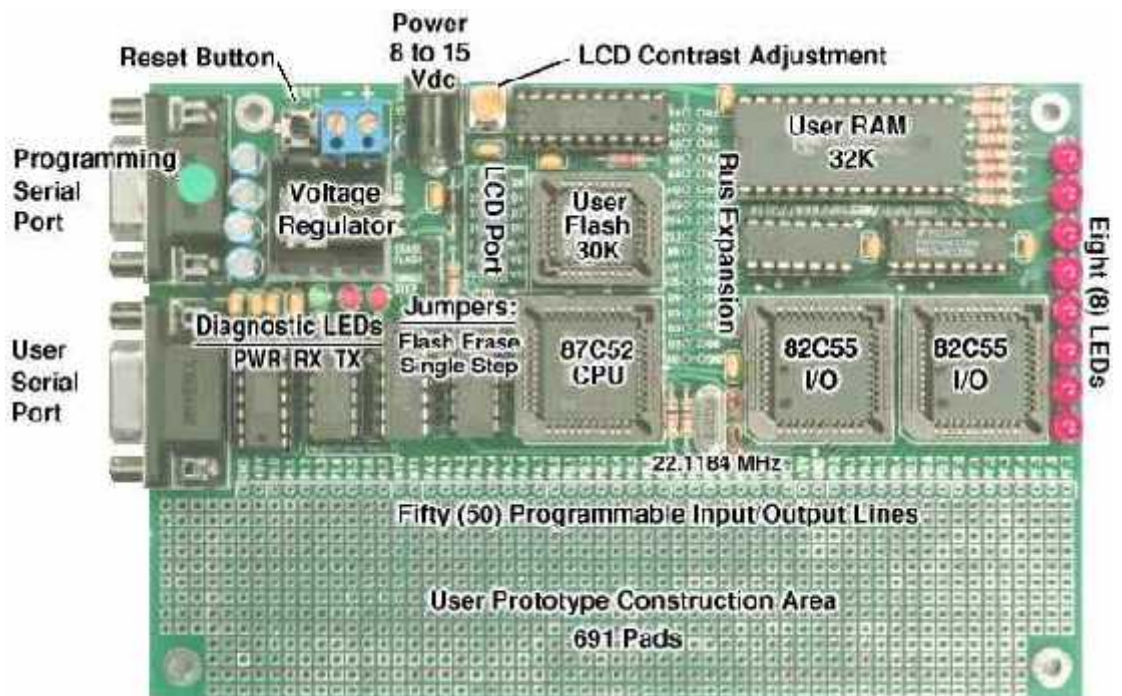


Figure (2.2): The 8051 Microcontroller Kit

2.2.1.3.2 Programmable Peripheral Interface (PPI)

The 82C55A is a high performance version of the industry standard 8255A and is manufactured using a self-aligned silicon gate process. It is a general purpose programmable I/O device which may be used with many different microprocessors.

There are 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The high performance and industry standard configuration of the 82C55A make it compatible with the 80C86, 80C88 and other microprocessors.

We need to use it to connect the input and output, such as, sensors and LCDs, devices to the 8051 microcontroller via its ports.

2.2.2 Sensor (Switch)

In this project we will use a kind of sensor, that is, a switch to sense in every park that the car found or not, and to sense the car at the entrance or exit, for controlling the open or close of the gates park, by make it as the input part to the microcontroller.

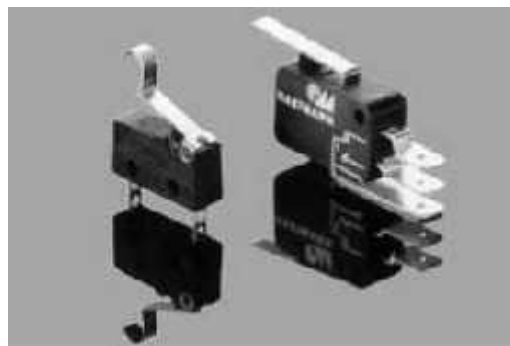


Figure (2.3): Switch

The pin labeled ``+5v supply" may be used to power an active sensor (e.g., the transmitter LED of a reflective optosensor). The pin labeled ``sensor signal" is the input to the Handy Board circuitry; this must be in the range of 0 to 5 volts. The pin labeled ``ground" is the system ground.

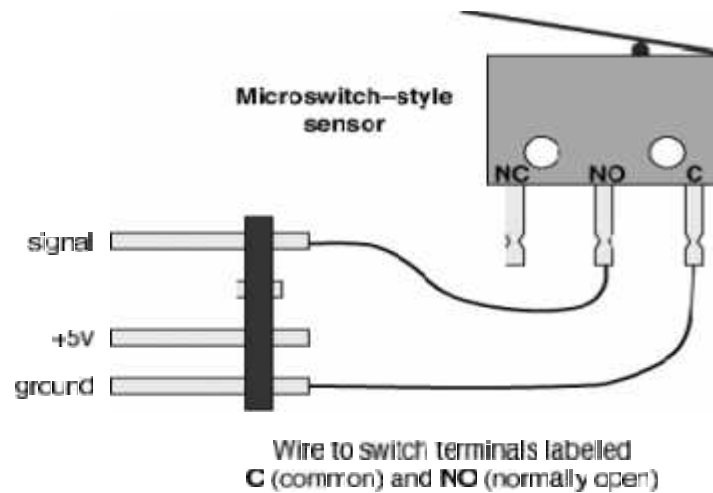


Figure (2.4): Switch Sensor

The above diagram shows how to wire a switch-style sensor to the Handy Board. As indicated in the diagram, the switch terminals labeled "C" (common) and "NO" (normally open) should be connected to the sensor plug.

This wiring creates a switch sensor that is normally open, or disconnected, except when the switch is pressed. The standard software for reading the state of a switch interprets this logic high value as "not pressed" or false. When the switch is closed, the sensor line is connected to ground, and the software reads a logic low value, which is interpreted as "pressed" or true.

2.2.3 Motors

In this project we will use two DC motors for the two gates, to open the gate for the park when the cars will enter, and close it when the cars leave.

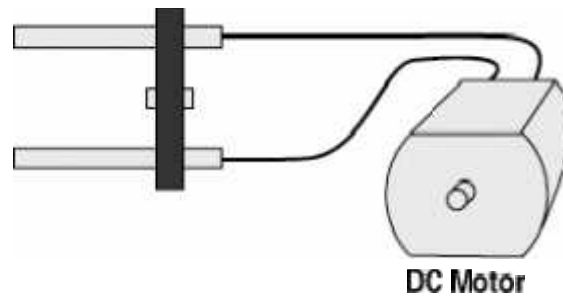


Figure (2.5): Motor

The DC motor connector uses two male pins on 0.2 inch spacing; i.e., the outer two of three pins. The center pin can be clipped away from the assembly.

We will use DC motor because it is give good speed for open the gate, on the other hand, its price cheaper than other kind of motors.

2.2.4. LCD

LCD, A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. It is prized by engineers because it uses very small amounts of electric power, and is therefore suitable for use in battery-powered electronic devices.

We will use it to display if there is an empty location in the park for the customer to put his car in it and to know exactly where to go form the main entrance, for that, we need more than one LCD, the main LCD, and one for each floor.

In this project we will use LCD its size will be at least 34 inch and over, because every one who at 10 meters distance will be able to see the data on the LCDs, and below an option of 40 inch LCD.



Figure (2.6): LCD Display

We will take about SONY LCD. This LCD video performance advanced networking options, or its 40" of display area. This large format LCD display. Receive WXGA 1366 x 768 resolution paired with a 16:9 aspect ratio and forget about blurry or imperfect images. Wide 178 degree viewing angles, with its 450 cd/m2 brightness. The FWD-40LX1/S has a minimum LCD panel life of 60,000 hours.

SONY LCD Features

1. Viewable Image Size: "40".
2. Resolution: WXGA 1366 x 768.
3. Aspect Ratio: 16:9.
4. Brightness (Typical): 450 cd/m2.
5. Connectivity: RGB/Component (HD-15), DVI-D HDCP, Composite (BNC), YUV (Composite).
6. Cabinet Color(s): Silver.
7. Environments: PC Compatible.

2.2.5. Gate

Gate, The combination of our proven and reliable electric motor with a lever system represents a simple and extremely reliable drive solution. It permits short opening and closing times without the barrier boom bouncing in the end positions. The lever system locks the barrier boom at both end positions.

The motor at each gate use to open and close it, and use as the output to the microcontroller.

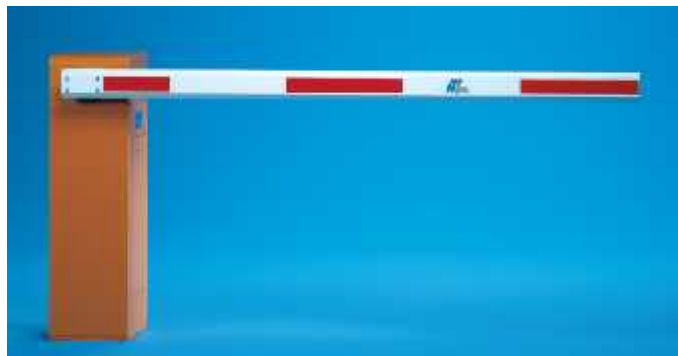


Figure (2.7): The Gate

2.2.6. Spring

we will put a switch between two springs, each spring will be compressed by 200Kg, and over it a piece of iron, therefore, the switch will not closed until a 200Kg over it like a car.

Springs are fundamental mechanical components which form the basis of many mechanical systems. A spring can be defined to be an elastic member who exerts a resisting force when its shape is changed. Most springs are assumed linear and obey the Hooke's Law,

$$F = k\Delta$$

Where F is the resisting force, D is the displacement, and the k is the spring constant. There are many basic spring types, the most plentiful of which are shown as follows, we will use a compression spring.

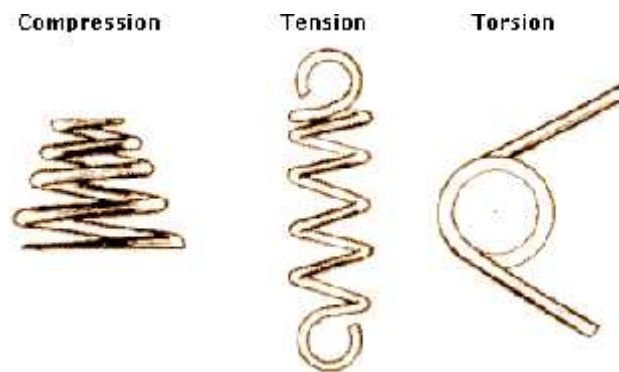


Figure (2.8): The Springs

2.2.7 Force on Spring

The switch found between two springs. Each car location effected with the mass of the car and the mass of the piece of iron. After searching and study we found that the weight of the car is between 400_2000 Kg, the weight of the piece of iron approximately 25 Kg and the weight of five person if the car approximately (5X70), from that we must ensure that the switch become on when weight over it at least 400 Kg .

2.2.8 General Parking Diagram

The figure (2.9) shows the first floor of the park area, the cars are in blue colors and this park contains a large number of cars, the circle in the bottom of the

figure is the place from which the cars go to the second floor and from the left the entrance of the park and the exit from the right.

In this park a piece of iron put in each location in the park, exactly, in the half length, that is the first two wheels only press it, if the location length 5m the piece of iron put after the first 2.5m form the location length.

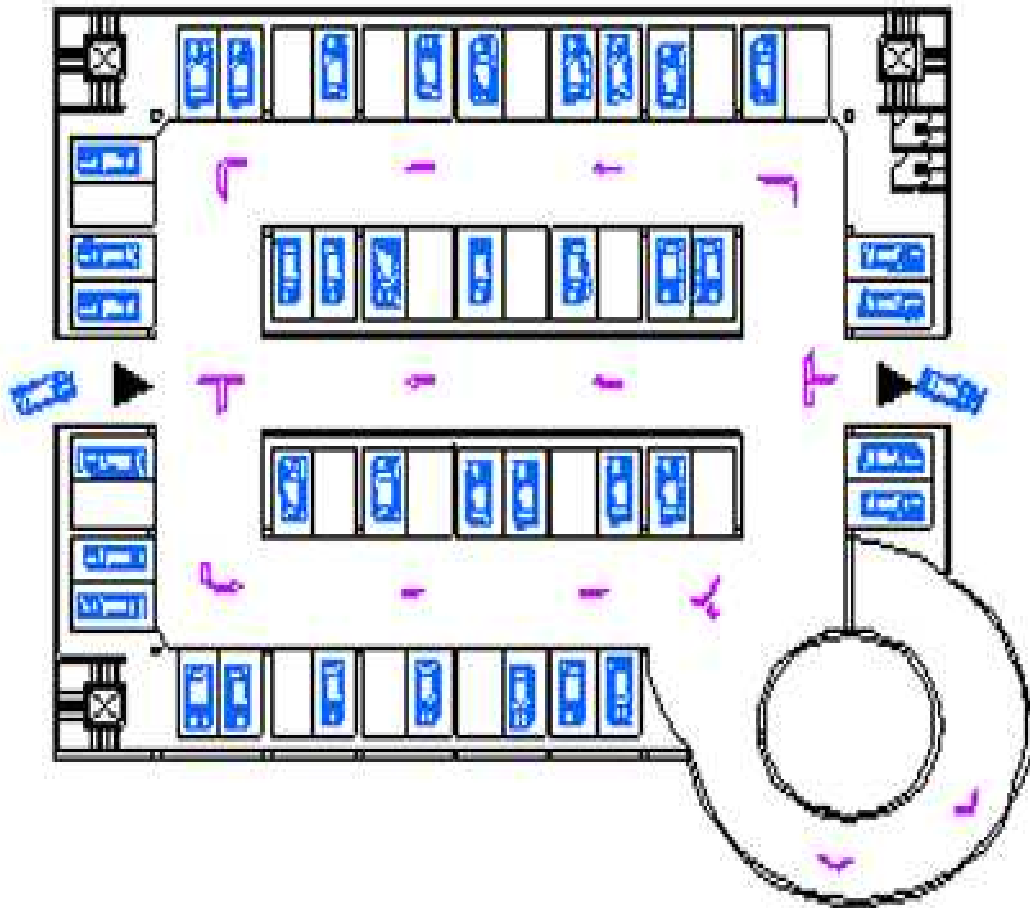


Figure (2.9): The first floor

The figure (2.10) shows the second floor of the park area, cars in blue like the figure of the first floor, but the difference between this figure and the figure (2.9) that this floor without an entrance and exit places but it has a circler for the cars to go to the first floor and to leave the park.

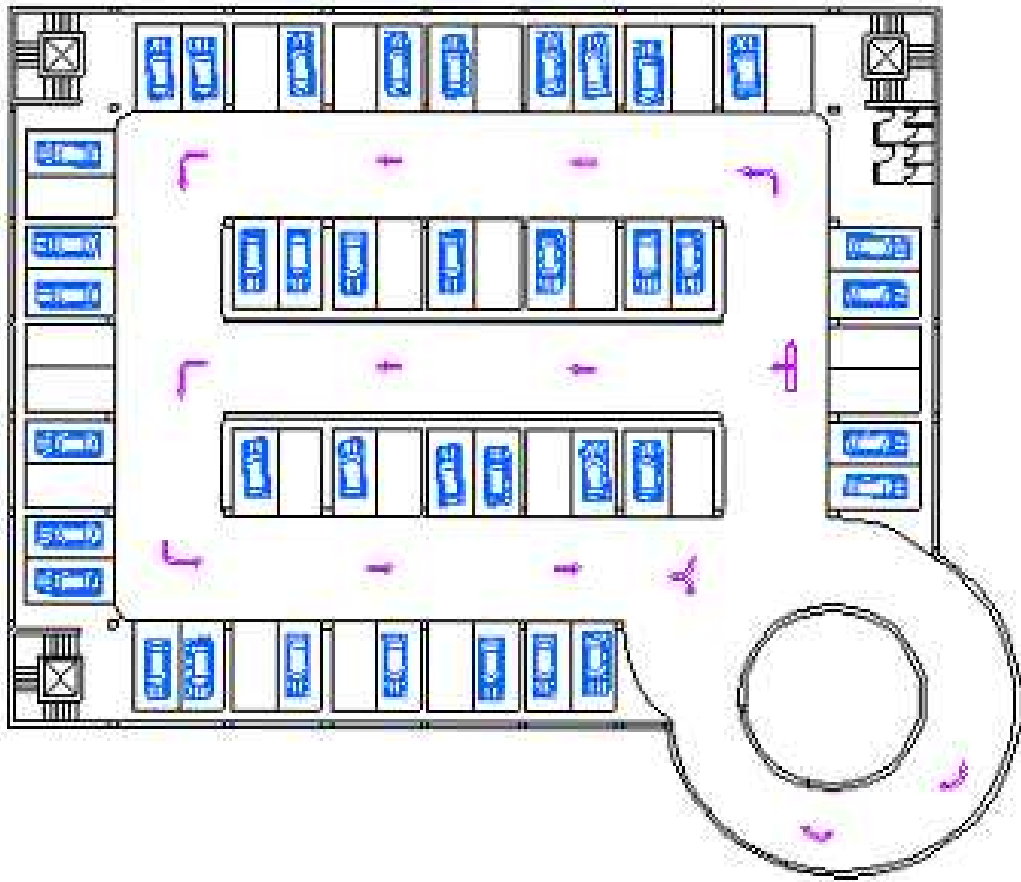


Figure (2.10): The second floor

3

Design Concepts

- 3.1 Project Objectives
- 3.2 Project main hardware components
- 3.3 General Block Diagram
- 3.4 Interfacing
- 3.5 Motors
- 3.6 How System Works

Chapter Three

Design Concepts

3.1 Project Objective

- To design and implement a smart parking car system using 8051 microcontroller.
- Implement a charging system at the entrance to the parking.
- Implement a subsystem for each floor showing the available parking slots, writing the necessary software.

3.2 Project Main Hardware Components

In this project we use a microcontroller 8051 to control the parking in our city and to have more development, smart, security and easy to use parking.

This system consists of many basic components, as we mention in chapter two, the main part of the system given below.

- 8051 microcontroller development kit.
- Sensor (switch).
- Display (LCDs).
- Motor.
- Gate.

All of these component connected together to make smart parking, the 8051 connected to micro-switch as input and also connect it to the LCDs as output from

the microcontroller, and the two gates will open when the car enters or leaves the park according to the motor movement, its movement will happen when the sensor (switch) has closed and so send signal to the microcontroller, so we have full system to control all of the cars that enter, leave and in the parking. Therefore, every free position is known to every customer who wants to put his car in that park.

The smart car parking system aim to make a full control system engineering that control many details in this park, its control opening the gates and close it from the motor movement that move according to the microcontroller signal. So, it is very accurate system that open and close at specific time.

LCD displays free spaces in each floor and the exact location. Therefore, this will be very organized, arranged park and comfortable to the customer use.

In this park the machine will be the employee not the human.

3.3 General Block Diagram

The general block diagram of our project (not including the charging system) is shown in figure (3.1), there are two floor connected to the main system board.

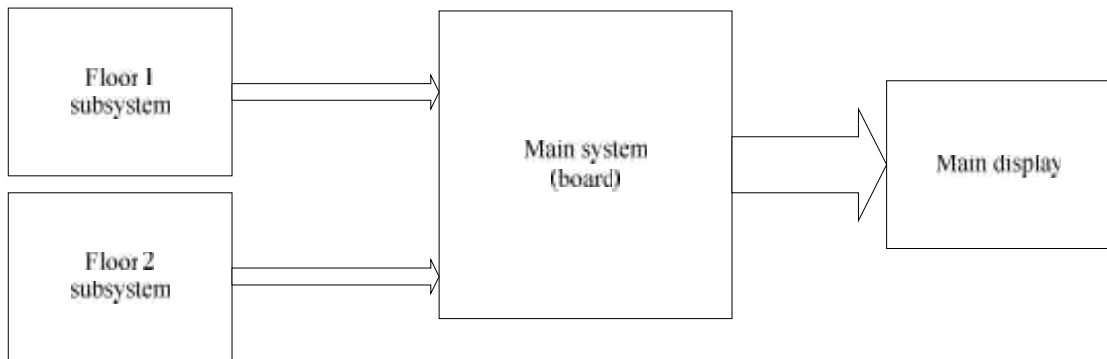


Figure (3.1): General Block Diagram

3.3.1 Floor Subsystems

The system consists of microcontroller 8051, sensors and display.

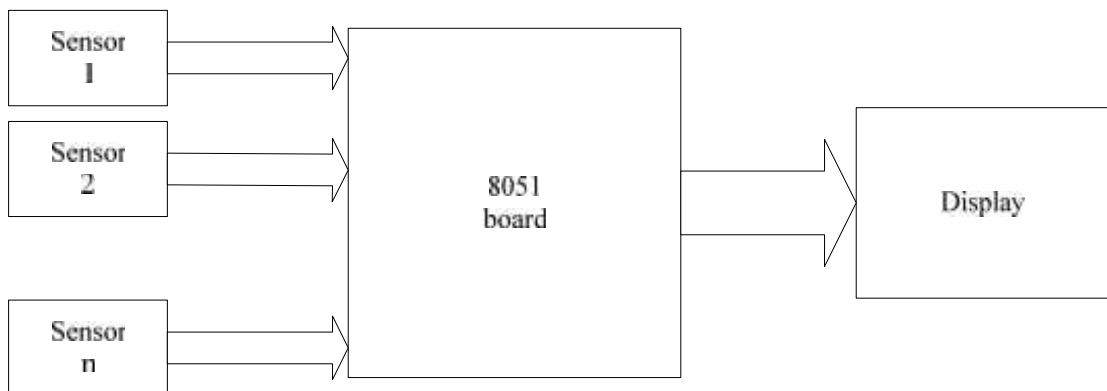


Figure (3.2): Floor Subsystem

3.3.2 Entrance System

Consist of 8051 microcontroller display and floors.

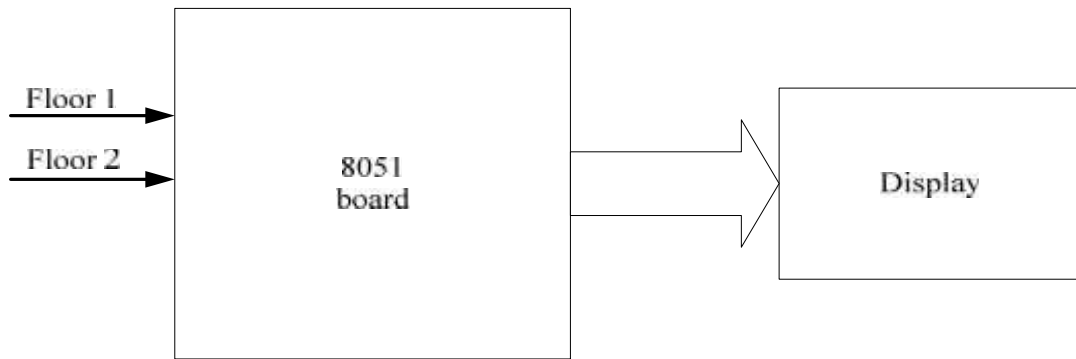


Figure (3.3): Entrance System

3.3.3 8051 Microcontroller System

This block diagram show the 8051, RAM, ROM, Decoder, ALU and others.

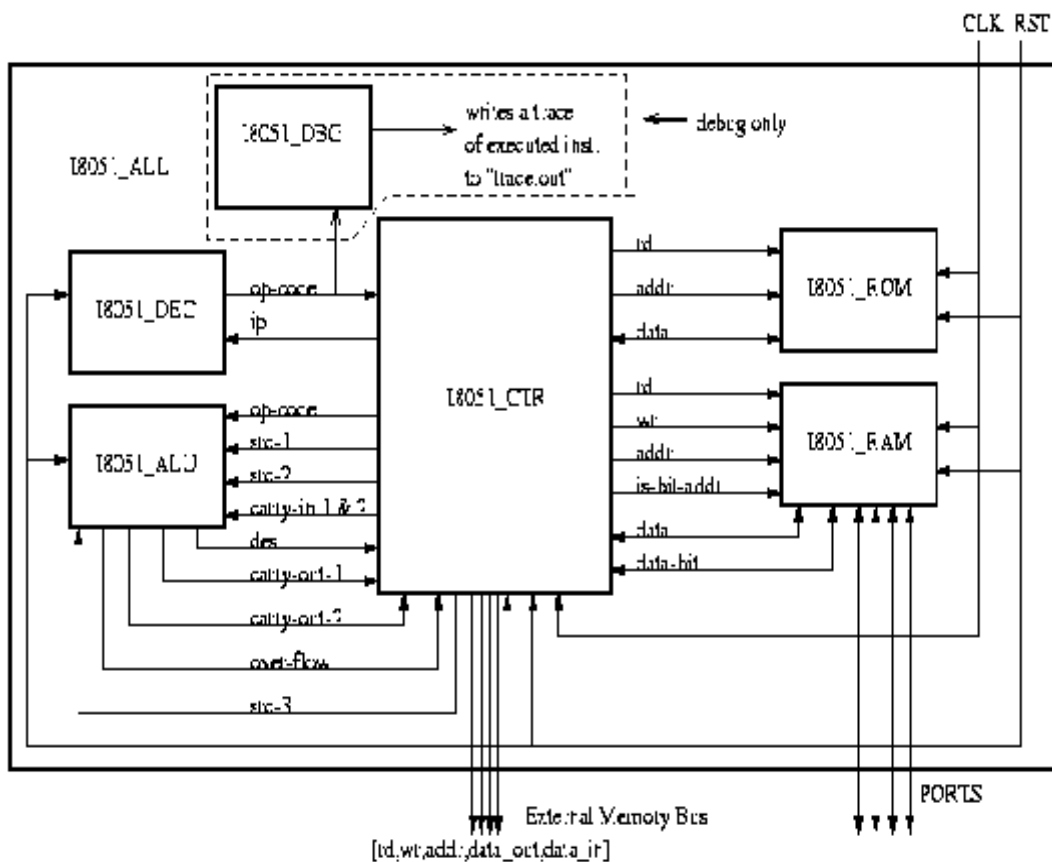


Figure (3.4): The 8051 Microcontroller

3.4 Interfacing

Here we will take about interfacing of the floor system with the entrance main system. The main system makes interfacing between the two floors as input, and the main LCD and the two motors as output. Figure (3.5)

The two subsystems for the two floors make interfacing between the switches as input, and the LCD for each floor as output as shown in figure (3.6). We must draw attention to the interfacing between the main system and the subsystems,

therefore, the whole system share the data between the main and the sub which explained at the end of this chapter in section (3.5) and in chapter two.

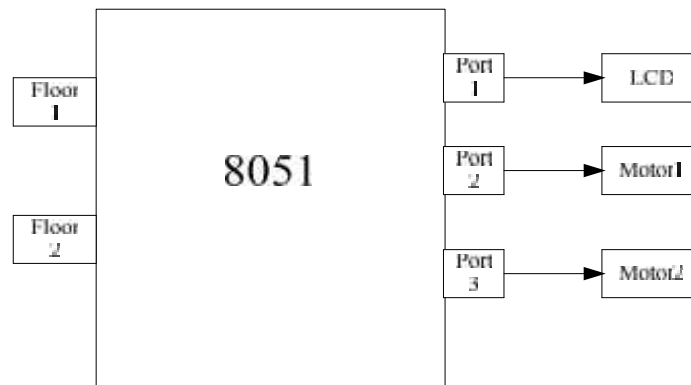


Figure (3.5): Interfacing Main System

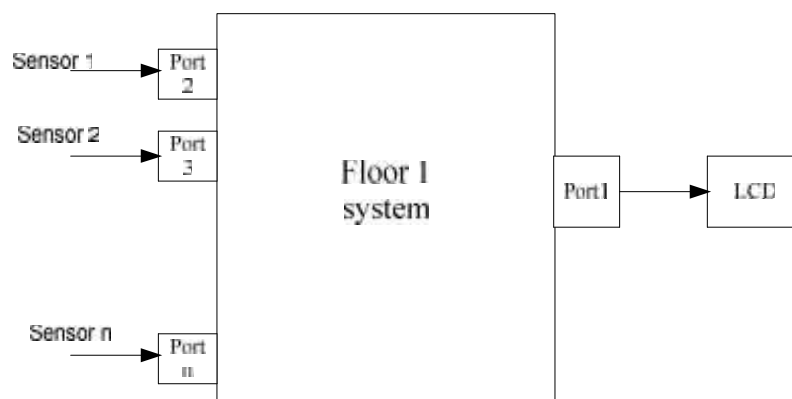


Figure (3.6): Interfacing Floor 1

3.5 Motors

In this project we can use two options of two kinds of motors, AC motor or DC motor. We preface to use DC as we mention before, figure (3.7).

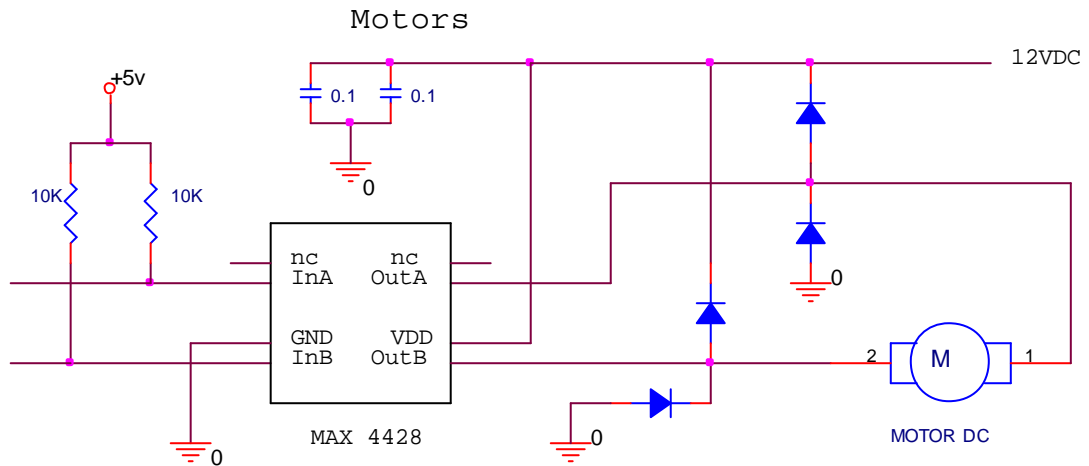


Figure (3.7): DC Motor

3.6 How system Works

3.6.1 Microcontroller with Switch, LCDs and Motors

The microcontroller system consists of hardware and firmware. The firmware of general purpose experimental systems is usually a monitor program that lets users inspect and modify system attributes such as memory and ports. In addition, a monitor program should allow downloading and running other applications program. Once the application software has been fully developed and tested, it may be placed in ROM and the microcontroller system be used as an embedded controller.

The microcontroller system consists of three major blocks: CPU, memory, and input /output ports.

Grouping the sub systems of microcontroller system into blocks is convenient way to describe circuitry that is too large to be displayed on a single sheet. System blocks diagrams also describe the important signals that connect the sub systems.

A bus is a collection of several related signals. There are four buses: the address bus, the data bus, and the two port buses. The buses are represented by names, followed the range of indices. The address bus is given the name A with the index range [0...15]. The individual lines of the address bus are therefore named A0, A1... A15. The address bus is 16 bits wide, where as the data bus only 8 bits wide. Port 1 bits are P1.0, P1.1... P1.7. Similarly P3 bits are connected to a bus, even though only 4 bits, P3.2 to P3.5 are available as general-purpose input/output ports.

The 8051 microcontroller is at the center of the subsystem. The two capacitors C1 and C11 and the crystal Y1 are used by the microcontroller to generate the oscillator clock. The oscillator allows many popular Baud rates to be generated. The jumper JP3, when installed, grounds the EA# signal to allow code to be fetched from external memory. When pressed, pushbutton s2 connects the rest input to VCC. Port0 and Port 2 are used for external memory access. Port0 first emits the low byte of the address. The ALE signal is used to latch the address low byte. Then Port0 emits or receives the data byte. The octal latch is used to extract the address low byte. The output of the latch is always enabled. Eight resistors are connected to port 0, and the ninth resistor is used as a pull-up resistor for the EA# input. The 5 volt supply must be connected to the VCC pin, pin 40; similarly the ground of the power supply, 0 volts, must be connected to the GND pin, pin 20.

The external code and data memory blocks overlap. The ANDed signal, called READ#, is activated (made low) when ever PSEN# or RD# signals are low. The WR# signal is generated by the microcontroller during external memory fetches. Four of the bits of port 3 are used by the system. P3.0 and P3.1 are used for serial communication, and P3.6 and P3.7 for the WR# and RD# signal. Port P1 and 4 bits of port P3 are available as general purpose input/output bits. Port P3 bits are also used in conjunction with the timers and as external interrupts.

The 64K of system memory is organized in two 32K halves. One of the 32K blocks is an EPROM and the other is a RAM device. The most significant bit of the address bus A15 determines which half of memory is addressed. A15 and its complement are used as the decoding signals to activate the two memory devices. The complement of A15 is obtained by one of the NAND gate. The double-pole-double-throw (DPDT) switch determines which memory device is decoded as the low block of memory. With a monitor program in EPROM, the EPROM should be selected as the low memory block so that upon reset, the monitor program initializes and runs the system. An application program that is downloaded into the RAM is executed by toggling S1 while the RESET button is pressed. The RESET button is held pressed, no instructions are fetched. Toggling S1 decodes the RAM containing the application program to be decoded as the low block of memory. Once the reset button is released, the microcontroller starts fetching the instructions from location 0 now decoded as the first byte of the RAM, thus executing the downloaded program. The digital input/output lines are connected to a terminal block. A separate two terminal connector JP2 is used to bring the supply voltage to the volt.

The output of the decoder such as Y0 pin 15 will be connected to the chip select (CS) pin7 of the PPI, and the decoder takes the input from the microcontroller port, and the output of the decoder will be connected to the chip select of RAM and EPROM.

The unused ports of both PPI will be connected to another hardware device, LCD takes bits from PPI ports to make interface between it and the system, the motor will take 1 bit and since we have two motors we need 2 bits from ports. Each micro switch need 1 bit from the ports to make interface between the switch and the system.

In this system we will use a switches sensor for each location in the park and the other for the entrance and exit place.

And here the description steps of working system from the moment of car entrance to the park until to leave. When the car reach our smart park entrance, the sensor (switch) placed in that place will be closed, since the customer put his car in the entrance, then the sensors sends signal to the microcontroller, which in turns sends signal to the motor to open the gate, the same sensor after a delay like 5 seconds close the, this sensor will send a signal to the microcontroller, which in reverse will send another signal to the motor to close the gate. After that, the customer will choose the place where he wants to park his car from the main LCD, which displays the floors (where the first or second floor has empty places or not), which how many places are empty in it (10, 16 ...). Then he chose the location from a sub LCD, and then the car will be parked. Next, the switch at that location will be closed and send a signal to the microcontroller, to change the LCDs display, the location that the customer parks his car in it is not empty.

When the car leave the location, the circuit between the switch and the microcontroller port open, so the LCDs to change its displays which show that this location is empty for another car to use.

Therefore, when the car reaches the other gate to leave from the park, the sensor which precedes the gate will be closed and send a signal to the microcontroller, in respect the microcontroller sends a signal to the motor to open the gate. Finally, the car leaves the park then, closes the gate after a required delay.

3.6.2 The Switch with Microcontroller

The switch connected to the microcontroller by PPI or microcontroller ports, from one of its input (this called a signal terminal as we mention in chapter two), when the signal pin become on (switch closed) it will sent a signal to microcontroller port. The microcontroller receives the signal, and knows that there is a car in that

place. And when a car left that place that port become zero, and because of that, the microcontroller know that this place in the park is free for another car.

3.6.3 The LCDs with Microcontroller

The LCDs connected to the microcontroller with the output ports of PPI. It works as an output of the processed data which come from the microcontroller.

The main LCD show how many free places in each floor. The other shows the exactly free places in that floor. In every update the information changes dependently to switch signal.

The LCDs work dependent to the micro-switches signals, when the micro-switches closed send signals to the microcontroller as we mention before.

3.6.4 The Motor with Microcontroller

The two motors connected with two bits to the microcontroller, one bit for each one, according to PPI. The motor of entrance gate open when the switch sends signal to microcontroller, so the microcontroller sends another signal to motor to open that gate. And the motor will close the gate when another signal the microcontroller sends it.

The second motor (for the exit gate), as motor for entrance gate, except it will open according to the signal send to the microcontroller from switch before the exit gate.

3.6.5 The spring with switch

In each location in the park there is two spring in the ground, between them switch. When the car takes its place in the park, the wheel of that car will press a piece of iron over the two springs. The switch after five centimeter below the iron. This five centimeter become zero when the spring pressed by at lest 200Kg (the mass of car). So the switch closed and sends a signal to microcontroller.

3.6.6 The gate with the motor

The gates will open and close according to the motors movement.

4

Hardware System Design

- 4.1 Introduction
- 4.2 The units design
- 4.3 Overall System Design

Chapter Four

Hardware System Design

4.1 Preface

After explaining the theoretical background, the general block diagram of the system, and how the system works, there is a need to view what is the design of this system in more specific, powerful and more formal terms. So this chapter describes the final system design with all its features which are necessary to make the system works well and achieves the objectives of the system. This chapter shows the interfaces between the equipments of the theoretical background, and the more suitable chips that advanced the design.

4.2 The Units Design

The park consists of five physical modules (sensors, LCDs, motors, Microcontroller and power). All the components are assembled on a single board.

The system has mainly four parts; these parts must interface with each others to achieve the project goals. The team has to deal with each one as separate unit, study its features and prepare it to operate successfully with other units. These units are:

- Sensors.
- Control unit (8051 board).
- Display (LCDs).
- Motors.

4.2.1 Sensors (Interlocking Push Buttons)

The reasons of choosing interlocking push buttons refer to:

- Two through used for entrance and leaving.
- Easy to use since External control circuit is unnecessary.
- Availability.
- Acceptable.

The sensors are used to:

- Sense the entrance of the car, to become closed and send signal to the microcontroller ports (this will be explained later in control unit), and then this location will become not available for another user. When the car leaves the location, another new drivers can see that there is free really location in the same place.



Figure (4.1): Switch

- The system needs to transfer data between the sensors (in all places) and the microcontroller and in reverse.

- The term "switch" typically refers to electrical power or electronic telecommunication circuits. In applications where multiple switching options are required, mechanical switches have long been replaced by electronic variants which can be intelligently controlled and automated.
- Each sensor (push button switch) has two bits one for VCC and the other for the ground, when the sensor pushed to the first time it becomes on, but on the second push it becomes off.
- For this project, the sensor put between two springs, and above it a piece of iron, when the driver parked his car in any location in the park and push the button to the first time it give signal "1" or on, and send it to the microcontroller and all of the time that the driver still in that location the signal does not change, all the time stay on. And when the driver leaves the location, the sensor will be pushed to the second time and becomes off or "0", and stay off until another driver parked his car in this location.
- The project has to use many sensors; each location must have one sensor plus the sensors at the entrance and at the exit of the park.
- The microcontroller has A, B, C, D, E, and F ports and each port has 8 bits, we use port A (0..7) for the sensors in the first floor, and port B (0..7) for the sensors in the second floor, and bit1, bit2 from port c for the sensors at the entrance and exit, we connect each sensor (push button switch) from its VCC port to the microcontroller I/O ports, each one to one bit of the microcontroller as we say before.

4.2.2 DC_motor and H_Bridges

The H-bridge circuit consists of a set of four transistors in IC packages that are arranged in an “H” orientation. This layout allows for current to flow bi-directionally through the circuit thus allowing for directional control for our motors. Additionally, logic inputs signals can be used to determine which direction the motors are spinning. Depending on the paired combination of logic 1’s and 0’s the motor shaft can turn left, turn right, and brake. Speed control is another feature of the h-bridges, when given a pulse-width-modulated (PWM) input signals, depending on the length of the duty cycle; the speed can be varied accordingly. H-Bridges that will be used are Max4428. The H-bridges will act as interfaces between microcontroller and the motors.

This IC (H_Bridges) used to control the DC motor at the gate of the park and make the DC motor move forward (right) to open the gate of the park and backward (left) to close the door of the park.

The DC_motor will receive signal from the microcontroller to open the gate and another signal to close it.

4.2.3 LCDs

The M1632 is a low-power-consumption dot-matrix liquid crystal display (LCD) module with a high-contrast wide-view TN LCD panel and a CMOS LCD drive controller built in. The controller has a built-in character generator ROM/RAM, and display data RAM. All the display functions are controlled by instructions and the module can easily be interfaced with an MPU. This makes the module applicable to a wide range of purposes including terminal display units for microcomputers and display units for measuring gages.

Properties of the LCD used in virtual project:

- 16-character, two line TN liquid crystal display of 5 x 7 dot matrix + cursor.
- Duty ratio: 1/16.
- Character generator ROM for 192 character types (character font: 5 x 7 dot matrix).
- Character generator RAM for 8 character types (program writes) (character font: 5 x 7 dot matrix).
- 80 x 8 bit display data RAM (80 character maximum).
- Interface with 4 bit and 8bit MPUs possible.
- Display data RAM and character generator RAM readable from MPU.
- Many instruction functions: Display Clear, Cursor Home, Display ON/OFF, and Cursor ON/OFF.
- +5 volt single power supply.

There are three LCDs used to show the display data on the screen:

- The main LCD used to display the number of free places in each floor and puts at the entrance of the park before the driver reach floor one.
- The first secondary LCD in floor one used to display the specific free places in that floor like (floor one: A (1, 2), B (1, 3, 4)), this mean location A in floor one has 1 and 2 free places; location B in floor two has 1, 3 and 4 free places.
- The second secondary LCD in floor two used to display the specific free places in that floor like (floor two: A (1, 2), B (1, 3, 4)), this mean location A in floor two has 1 and 2 free places; location B in floor two has 1, 3 and 4 free places.

4.2.4 Control Unit

The system need control unit to achieve all the operation of the design. The control unit represented by two main parts; the first one represented by computer that connect the units of project together by serial port and other hardware design such as LCDs and sensors.

The second part represented by software driver programmed using C programming Language that install from the computer to the microcontroller board from serial port then decode and analyze it after that execute and perform the specific task refer to it (we will discuss this part in detailed at chapter five).

Now we will illustrate the ports that connect the project subsystem.

4.2.4.1 Serial Ports

We have to use the serial port on the computer receives asynchronous data at suitable speeds for our data. The serial port expects asynchronous data; the serial port can only accept words of length 5 or 7 bits. It also expects start and stop bits. The serial port cable has to be used to connect the computer with the circuit board and load programs into the 8051 microcontroller as shown in figure (4.2).



Figure (4.2): Serial Port Cable

The serial port is an I/O (Input/Output) device. An I/O device is just a way to get data into and out of a computer. Most PC's have one or two serial ports. Each has a 9-pin connector (sometimes 25-pin) on the back of the computer. Computer programs can send data (bytes) to the transmit pin (output) and receive bytes from the receive pin (input). The other pins are for control purposes and ground.

The serial port is much more than just a connector. It converts the data from parallel to serial and changes the electrical representation of the data. Inside the computer, data bits flow in parallel (using many wires at the same time). Serial flow is a stream of bits over a single wire (such as on the transmit or receive pin of the serial connector). For the serial port to create such a flow, it must convert data from parallel (inside the computer) to serial on the transmit pin and conversely.

4.2.4.1.1 Pins and Wires

Old PC's used 25 pin connectors but only about 9 pins were actually used so today most connectors are only 9-pin. Each of the 9 pins usually connects to a wire. Besides the two wires used for transmitting and receiving data, another pin (wire) is signal ground. The voltage on any wire is measured with respect to this ground. Thus the minimum number of wires to use for 2-way transmission of data is 3. Except that it has been known to work with no signal ground wire but with degraded performance and sometimes with errors. As shown in figure (4.3).



Figure (4.3): Serial port pins

4.2.4.1.2 Data flow

Data flows into and out of your serial port. Flow rates (such as 56k (56000) bits/sec) are (incorrectly) called "speed". But almost everyone says "speed" instead of "flow rate".

It's important to understand that the average speed is often less than the specified speed. Waits (or idle time) result in a lower average speed. These waits may include long waits of perhaps a second due to flow control. At the other extreme there may be very short waits (idle time) of several micro-seconds between bytes. If the device on the serial port (such as a modem) can't accept the full serial port speed, then the average speed must be reduced.

PC Com Port - EIA-574 RS-232/V.24 pin out on a DB-9 pin used for asynchronous data shown in figure (4.4).

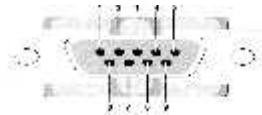


Figure (4.4): PC Com Port

Table(4.1): Serial Port Pins

Pin	Signal
1	Data Carrier detector
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready
7	Request to Send
8	Clear to Send
9	Ring Inductor

4.2.4.2 Controlling the Motor

The following table indicates the operation of the circuit below:

Table (4.2): Operation of the DC motor driving circuit

PPI pins		A	B	motion
P1	P2			
0	0	G	G	No motion
0	1	G	12v	Left rotation
1	0	12v	G	Right rotation
1	1	12v	12v	No motion

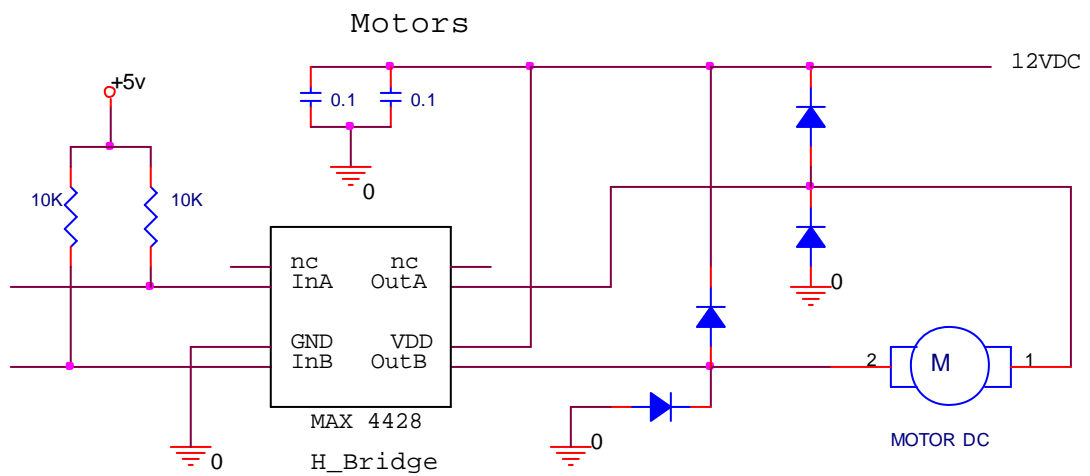


Figure (4.5): DC motor circuit

The DC_motor and H_bridge circuit used to open the gate of the park and close it according to DC_motor movements, when the motor move forward the gate will be opened and when its move backward the gate will be closed, it will be moved forward or backward according to the signal send to the H_bridge.

4.2.4.3 Electrolytic Capacitor and Resistor

The capacitor we used is about 0.1 micro for the every IC to protect it as shown in figure (4.6).

Resistor will be used in every IC in the project to give less current to the circuits as shown in figure (4.6). We will be using 100K Ω for every IC used in push button switch.



Figure (4.6): Single Capacitor



Figure (4.7): Resistors

4.2.4.4 (74244) Buffer

This buffer will be used as storage between the sensors and the microcontroller ports take the input from the sensor and send it to the microcontroller, as shown in figure (4.8).

Feature:

1. State outputs drive bus lines or buffer memory address registers.
2. PNP inputs reduce DC loading.
3. Package options include both plastic and ceramic chip carriers in addition to plastic and ceramic DIPs.

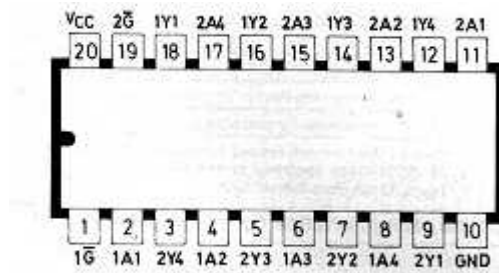


Figure (4.8): 74244 Buffer

4.2.5 Over All System Unit (Application Unit)

This unit is connected to the microcontroller unit, it consists of the circuit that drive the DC motor; which control the gate, and the sensor as input. And control the LCDs display as output.

4.2.5.1 Interfacing Circuits

The system consists of four parts, and each part has an important role in this system. But to achieve the system objectives and operates as one unit, there's a need to integrate each unit with others through the interfacing circuits .The interfacing circuits are:

4.2.5.1.1 Interfacing Sensors with microcontroller (8051) board

The following circuit in figure (4.9) represents the interfacing sensors in the floors with I/O ports of the microcontroller board using the PPI chip.

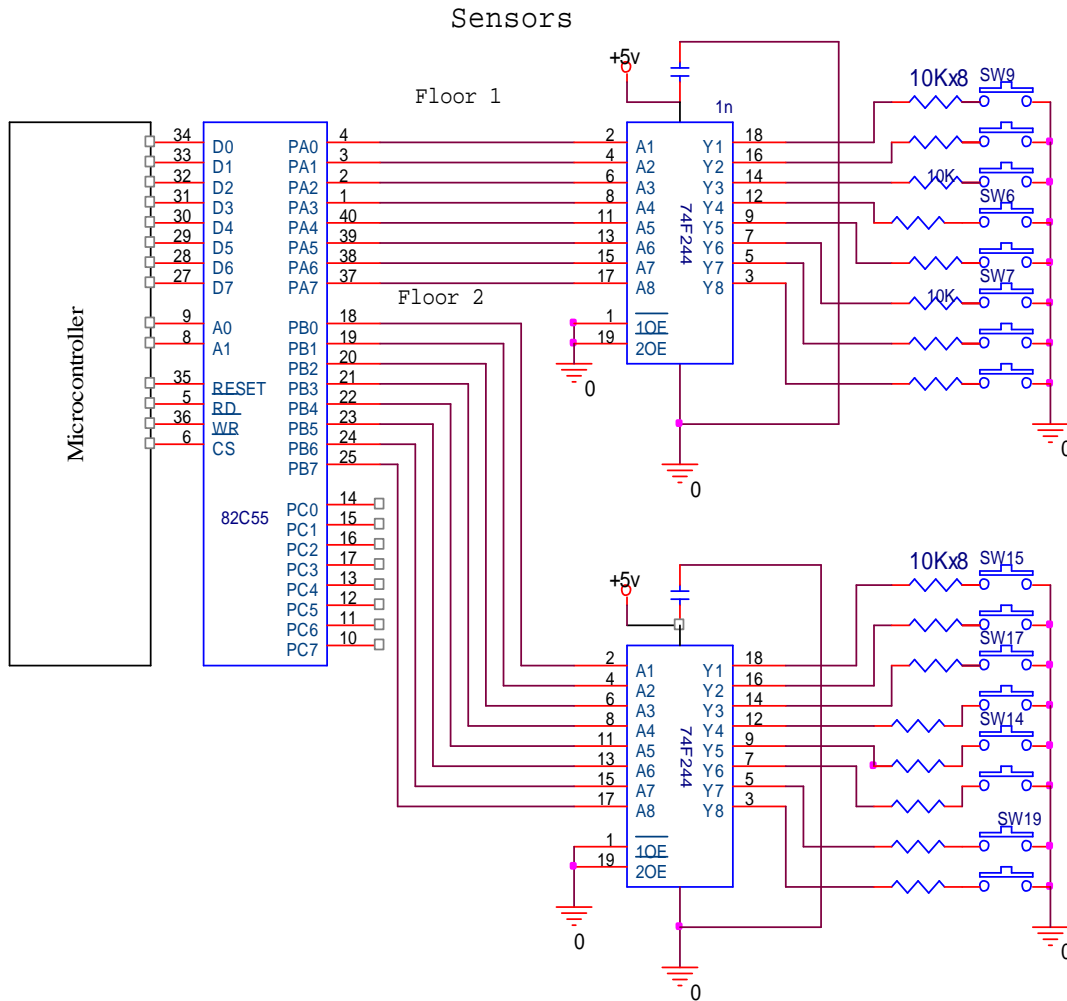


Figure (4.9): Interfacing Floors Sensors with 8051 Microcontroller

The following circuit represents the interfacing sensors of the gates with I/O ports of the microcontroller board using the PPI chip. Then it will be buffered by 74F244 buffer. In this PPI we will use port A (0_7) and port B (0_7) connected with

switches, for floor1 port A and for floor2 port B. Then, the signal will be received through these ports A and B and know which switch is on and which is off.

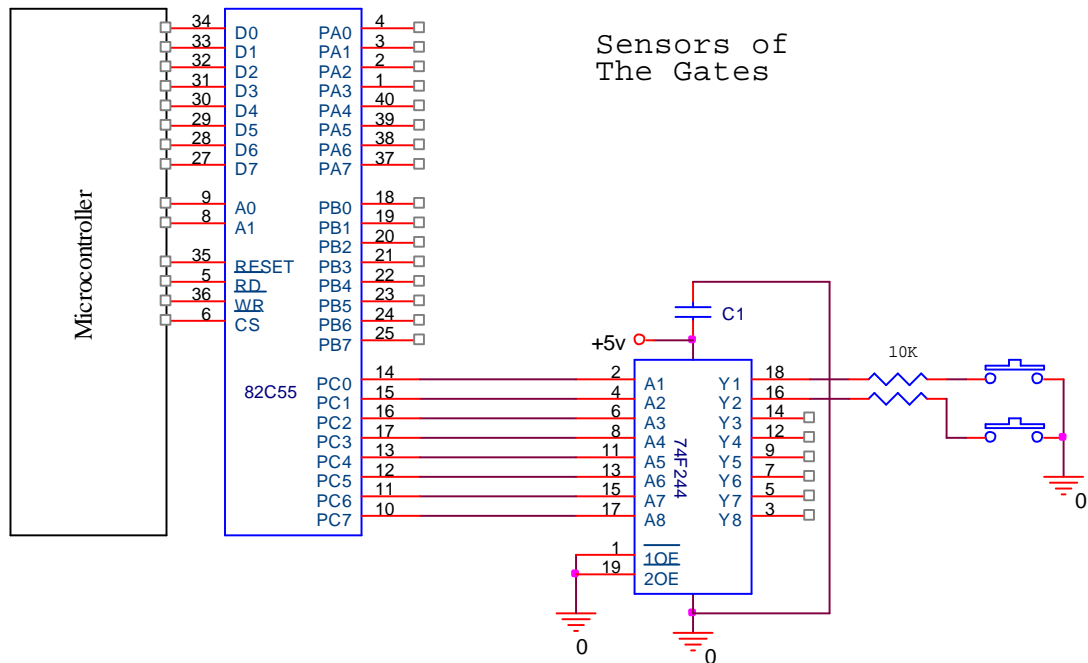


Figure (4.10): Interfacing Main entrance and Exit Gate Sensors with 8051 Microcontroller

4.2.5.1.2 Microcontroller 8051 Interfacing with LCD Display

The Microcontroller is interfaced with the LCD in figure (4.10). The following circuit represents the interfacing LCDs with I/O ports of the microcontroller board using the PPI chip. Then it will be buffered by 74F244 buffer.

In this PPI we will use port A (0_7) and port C (0_2) connected with first LCD, and port B (0_7) and port C (3_5) connected with second LCD. Then, the signal will be sending through these ports A and C to display the output data though these ports.

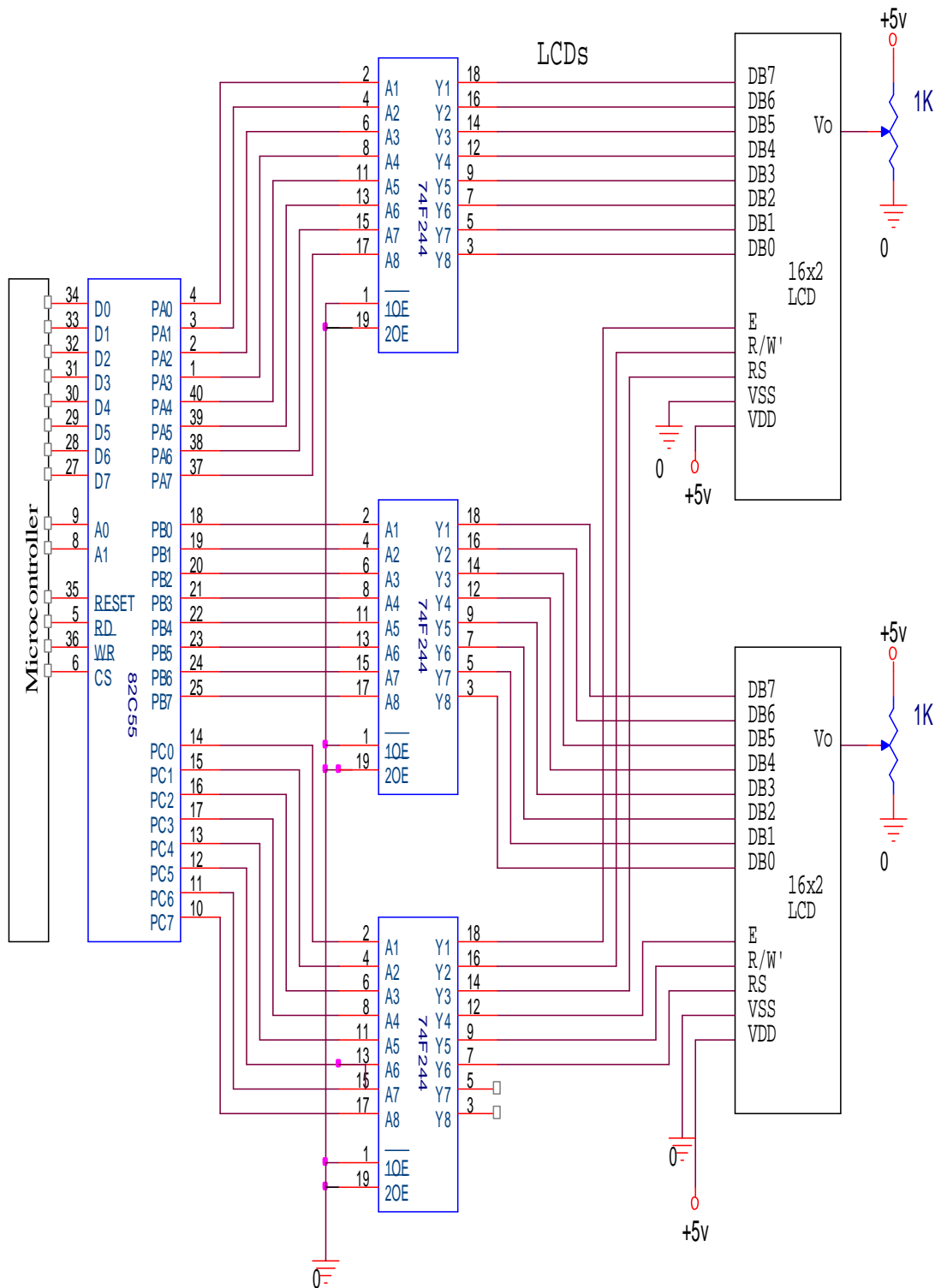


Figure (4.11): Interfacing LCDs with 8051 Microcontroller

4.2.5.1.3 Microcontroller 8051 Interfacing with Motors

The Microcontroller is interfaced with the motors during the PPI in figure (4.10). The following circuit represents the interfacing motors with I/O ports of the microcontroller board using the PPI chip. Then it will be buffered by 74F244 buffer.

In this PPI we will use three bits in port C connected with motors, 2 bits for the first motor and 2 bits for the second motor. Then, the signal will be sending through ports C to display the control the time when motor move and how (left or right).

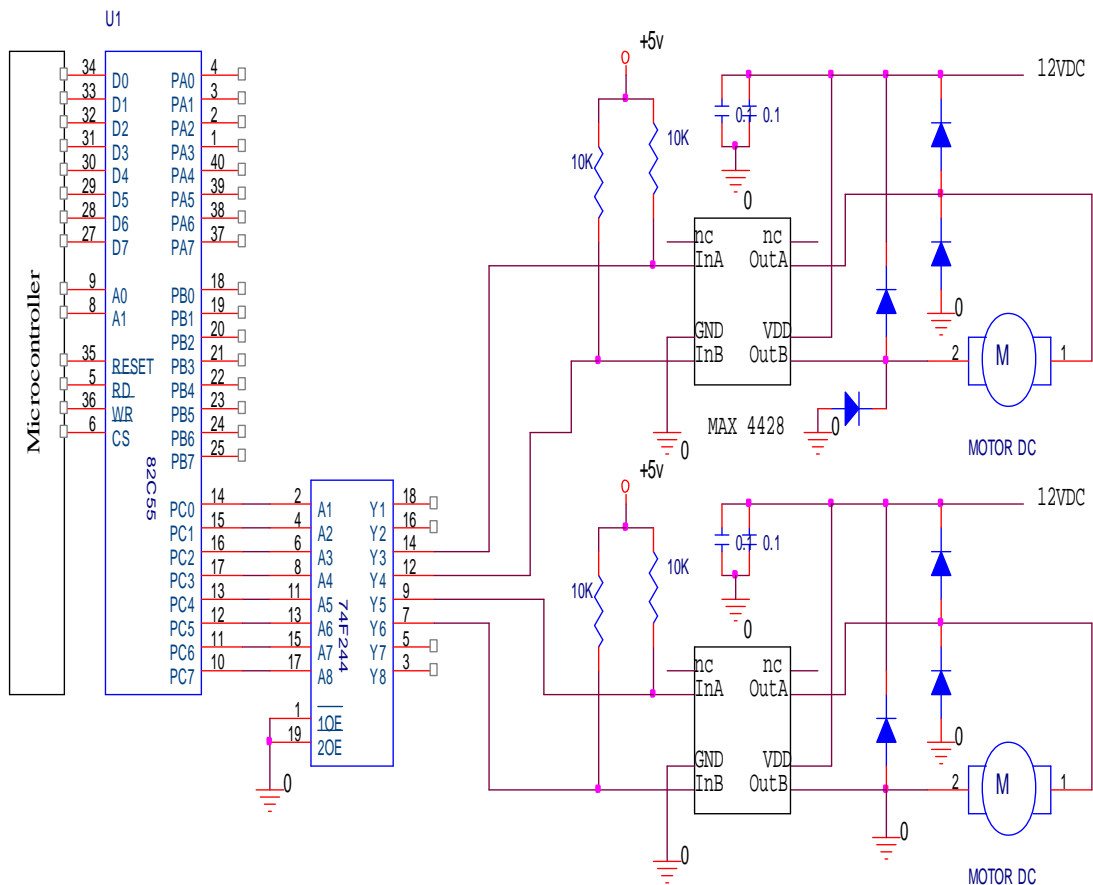


Figure (4.12): Interfacing Motors with 8051 Microcontroller

4.2.5.1.4 Microcontroller System Interface

The system need microcontroller board unit to achieve all operations of the design. The microcontroller is used since all the components that needed for the system were built onto one chip.

This system contains the following main components:

- 8051 Microcontroller.
- 74ls373 latch.
- EPROM
- 74ls138 Decoder.
- PPI (8255A).
- LCD.

The ports of the PPI are configured using control word that written to the control registers as the following:

Port A input.

Port B out put.

Port C not used.

This configuration is specified in the control word as the shown in the following table:

Table (4.3) control word of the PPI

BSR	Mode		PA	PCH	Mode	PB	PCL
1	0	0	1	0	0	0	0

4.2.6 Parallel Ports

A problem occurred in the project. This is, the 8051 microcontroller burned (for more information see chapter seven). That makes the team to find another solution and then used the parallel port to continue the project.

In parallel port the control unit is the computer that connects the units of project together by parallel. The second part represented by software driver programmed using visual basic .net that install on computer to fetch the signals, after that execute and perform the specific task refer to it (we will discuss this part in detailed at chapter five).

Parallel port is a simple and inexpensive tool for building computer controlled devices and projects. The simplicity and ease of programming makes parallel port popular in electronics world. The parallel port is often used in computer controlled robots, Atmel/PIC programmers, etc.

The primary use of parallel port is to connect printers to computer and is specifically designed for this purpose. Thus it is often called as printer Port. You can see the parallel port connector in the rear panel of your PC. It is a 25 pin female (DB25) connector. On almost all the PCs only one parallel port is present, but you can add more by buying and inserting ISA/PCI parallel port cards.

4.2.6.1 Hardware

The pin outs of DB25 connector is shown in the figure (4.13).

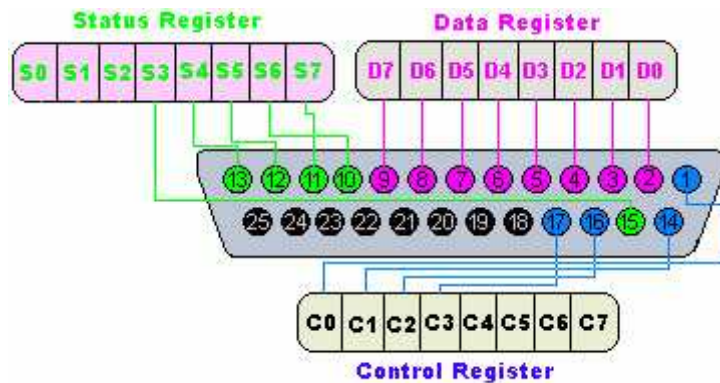


Figure 4.13: Parallel Port Register

The lines in DB25 connector are divided in to three groups:

- Data lines (data bus).
- Control lines.
- Status lines.

As the name refers, data is transferred over data lines, control lines are used to control the peripheral and of course, the peripheral returns status signals back computer through Status lines. These lines are connected to Data, Control and Status registers internally. The details of parallel port signal lines are given in table (4.4)

Table 4.4: Parallel Port Signal Line

Pin No (DB25)	Signal name	Direction	Register - bit	Inverted
1	nStrobe	Out	Control-0	Yes
2	Data0	In/Out	Data-0	No

3	Data1	In/Out	Data-1	No
4	Data2	In/Out	Data-2	No
5	Data3	In/Out	Data-3	No
6	Data4	In/Out	Data-4	No
7	Data5	In/Out	Data-5	No
8	Data6	In/Out	Data-6	No
9	Data7	In/Out	Data-7	No
10	nAck	In	Status-6	No
11	Busy	In	Status-7	Yes
12	Paper- Out	In	Status-5	No
13	Select	In	Status-4	No
14	Linefeed	Out	Control- 1	Yes
15	nError	In	Status-3	No
16	nInitialize	Out	Control- 2	No
17	nSelect- Printer	Out	Control- 3	Yes
18-25	Ground	-	-	-

4.6.3 Parallel Port Registers

The Data, Control and status lines are connected to their corresponding registers inside the computer. So by manipulating these registers in program, one can easily read or write to parallel port with programming languages like 'C' and BASIC.

The registers found in standard parallel port are:

- Data register

- Status registers
- Control register

As their names specify, Data register is connected to Data lines, Control register is connected to control lines and Status register is connected to Status lines. So whatever writes to these registers, will appear in corresponding lines as voltages, by measure it with a millimeter. And whatever give to parallel port as voltages can be read from these registers. For example, if we write '1' to Data register, the line Data0 will be driven to +5v. We can programmatically turn on and off any of the data lines and Control lines.

4.6.3.1 Where these registers?

In an IBM PC, these registers are IO mapped and will have unique address. We have to find these addresses to work with parallel port. For a typical PC, the base address of LPT1 is 0x378 and of LPT2 is 0x278. The data register resides at this base address, status register at base address + 1 and the control register is at base address + 2. So once we have the base address, we can calculate the address of each registers in this manner. The table (4.4) below shows the register addresses of LPT1 and LPT2.

Table 4.5: Register Addresses of LPT1 and LPT2

Register	LPT1	LPT2
data register(base address + 0)	0x378	0x278
Status register (base address + 1)	0x379	0x279
control register (base address + 2)	0x37a	0x27a

4.3 Overall System Design

The system design represents the complete interface between the units designed using 8051 kit.

In this system the inputs is the switches (sensors) that connected to the first PPI after it's buffered, the switches of the floors and gates take port A (0_7), port B (0_7) and two bits from port C (0_1).

The outputs will send to the motors in the first PPI, port C (2_5). Also to the LCDs for the floors in the second PPI, port A (0_7) and port C (0_2) for the first LCD, port B (0_7) and port C (3_5) for the second LCD.

Note: The third LCD is internally found on the 8051 development kit itself. The complete characteristic of the system is shown below in figure (4.14) (not implemented).

The system design represents the complete interface between the units designed using parallel port. There for, to control parking system the following design had been built and tested carefully to achieve the needed objectives. The complete characteristic of the system is shown below in figure (4.15).

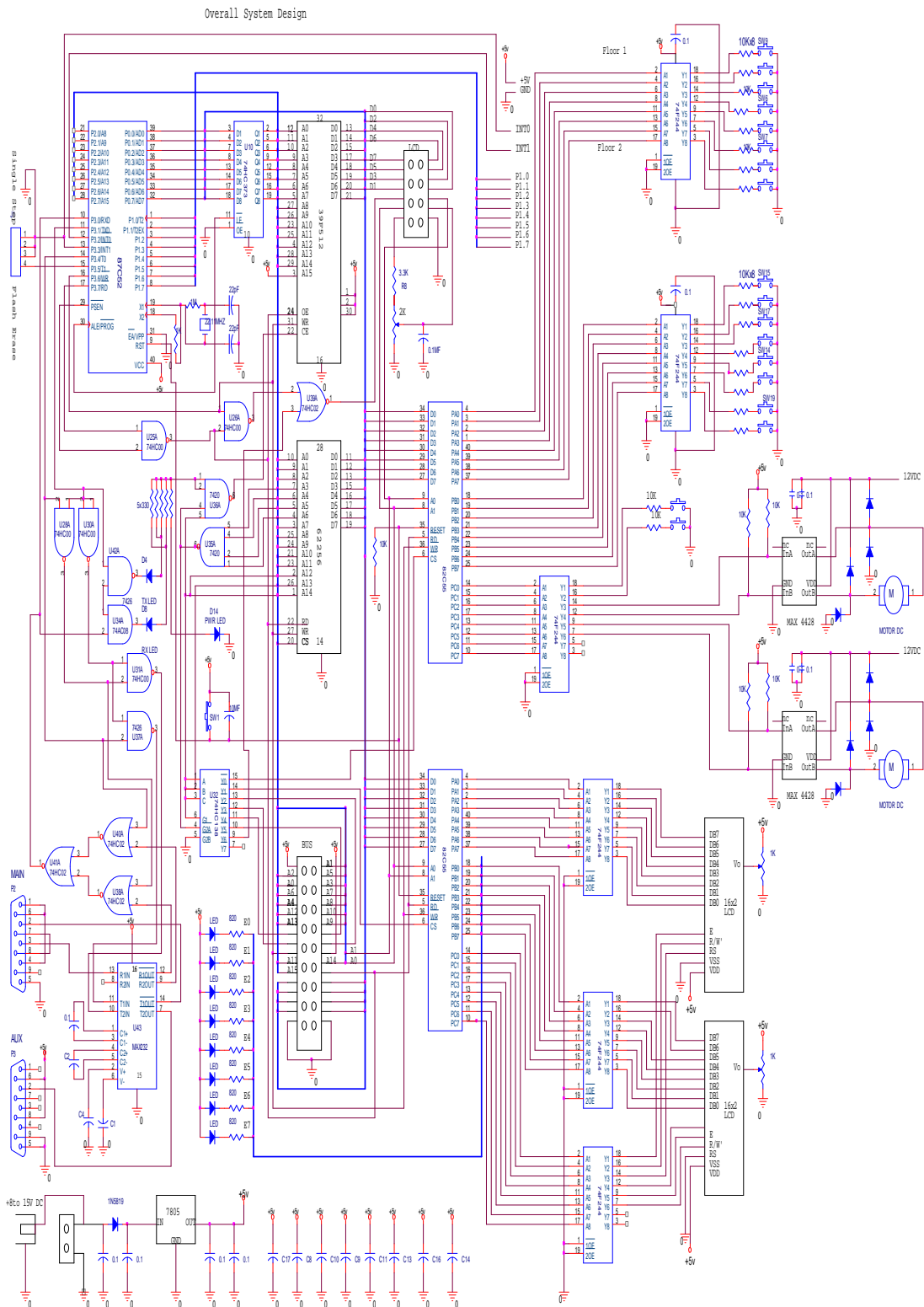


Figure (4.14): The System Design Circuit 1

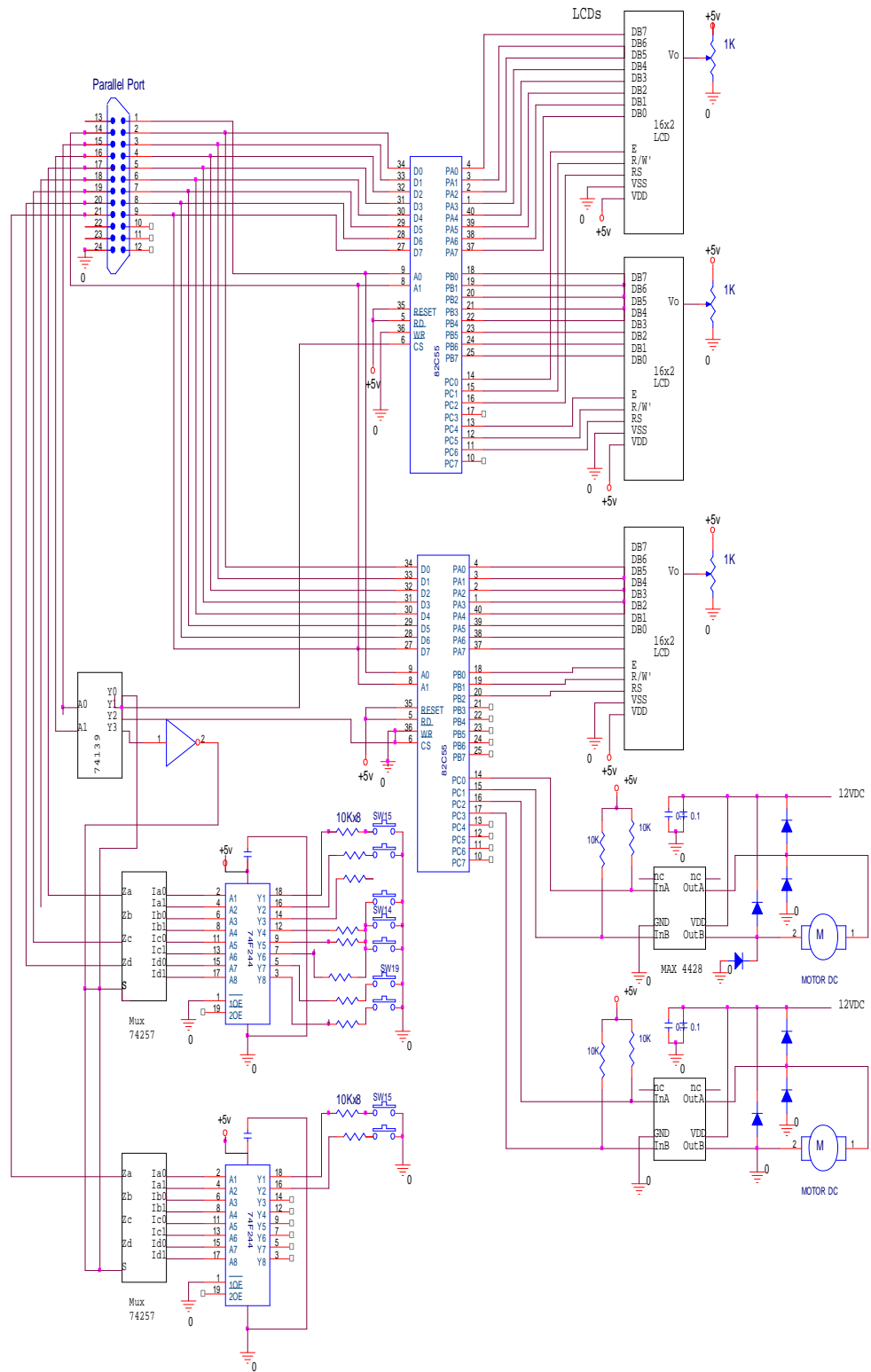


Figure (4.15): The System Design Circuit 2

5

Software System Design

5.1 Preface

5.2 Software Requirements Specifications

5.3 Function description

5.4 Serial Interface between the Microcontroller and PC

5.5 General System Flowchart

5.6 System Operational Flowchart

Chapter Five

Software System Design

5.1 Preface

In this chapter, we are going to describe the software system design which includes an explaining of the programming environment, programming tools, and to describe some of the using methods and algorithm design as followed.

And it contains the flowchart for all processes in this project from first step in the project, to the final step.

The overall software is programming in C programming language.

5.2 Software Requirements Specifications

After analyzing all software requirements, the following software functions and modules are needed:

5.2.1 Win 95/98/ME/NT/2000/XP Compatible Software

This system needs these types of windows in order to write all assembly and C codes on Text Editor Program, e.g. Notepad, also to used HyperTerminal as a program for transfer the assembly and C code, where these programs are already found on these windows.

5.2.2 HyperTerminal

HyperTerminal, which comes with Windows95, 98, Me, NT, 2K and XP can be used to obtain information from any external device such as microcontroller or modem, or to send any data or information to these external device .Also it can be Set up ,format, memory access and erase memory for any external device.

In this system there is a need for this program to make the connection between PC and the 8051 microcontroller, where this program is used to transfer all programs that will be writing in C language via serial cable to the 8051 microcontroller ,but first the procedure of setup this program on PC according to 8051 microcontroller must be understood.

To setup the connection of the HyperTerminal, the following procedure should be performed:

1. Click on HyperTerminal icon, which found in the program file on PC then Connection description screen will appear as shown in figure (5.1).

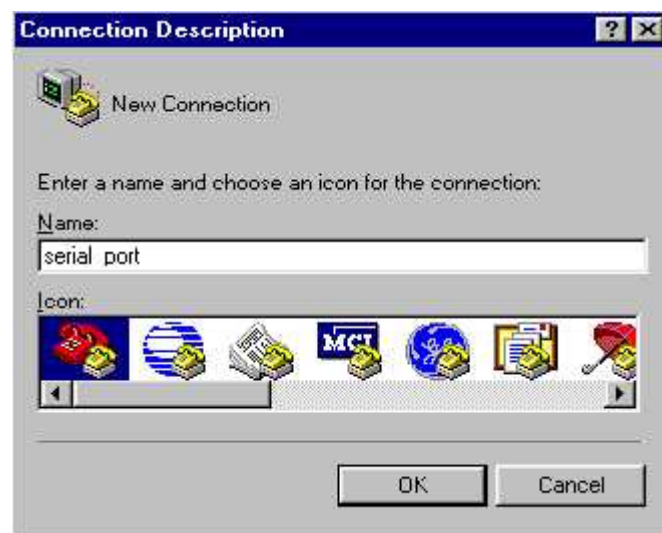


Figure (5.1): Connection Description Screen

2. Put the name of the connection and choose COM1 or COM2 dependent where serial cable is connected then you will see the Connect to screen as shown in the figure (5.2).



Figure (5.2): Connect to Screen

3. Type any country/region, area code, phone number, and connection in the previous screen then you will see the COM1 properties screen as shown in the figure (5.3).

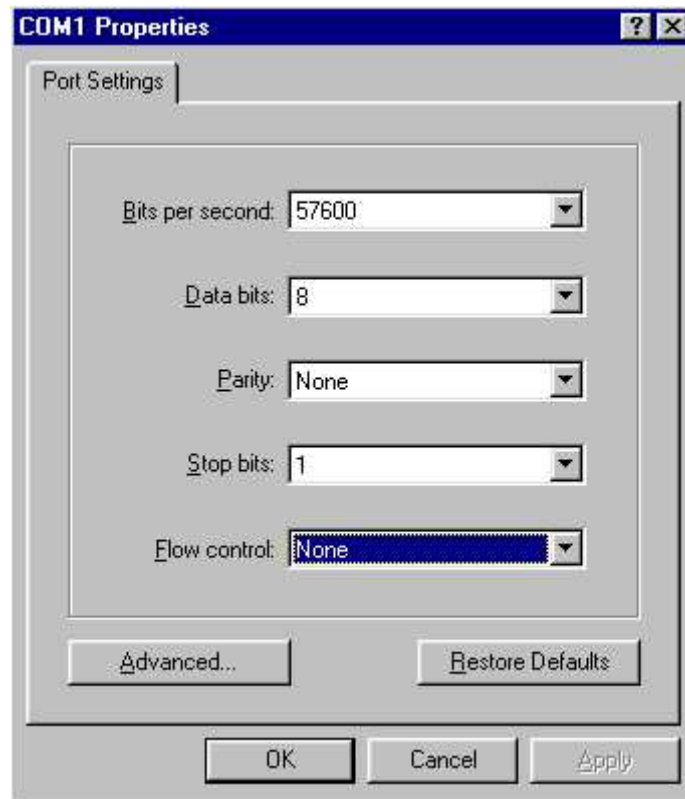


Figure (5.3):COM1 Properties

4. Choose 115200 for bits per second, 8 bit for data bits, 1 bit for stop bits and none for flow control also press on OK icon in the previous screen then you will see the HyperTerminal empty screen.
5. Press enter to dial up connection between the HyperTerminal program and the 8051 microcontroller then you will see the following screen, in figure (5.4).

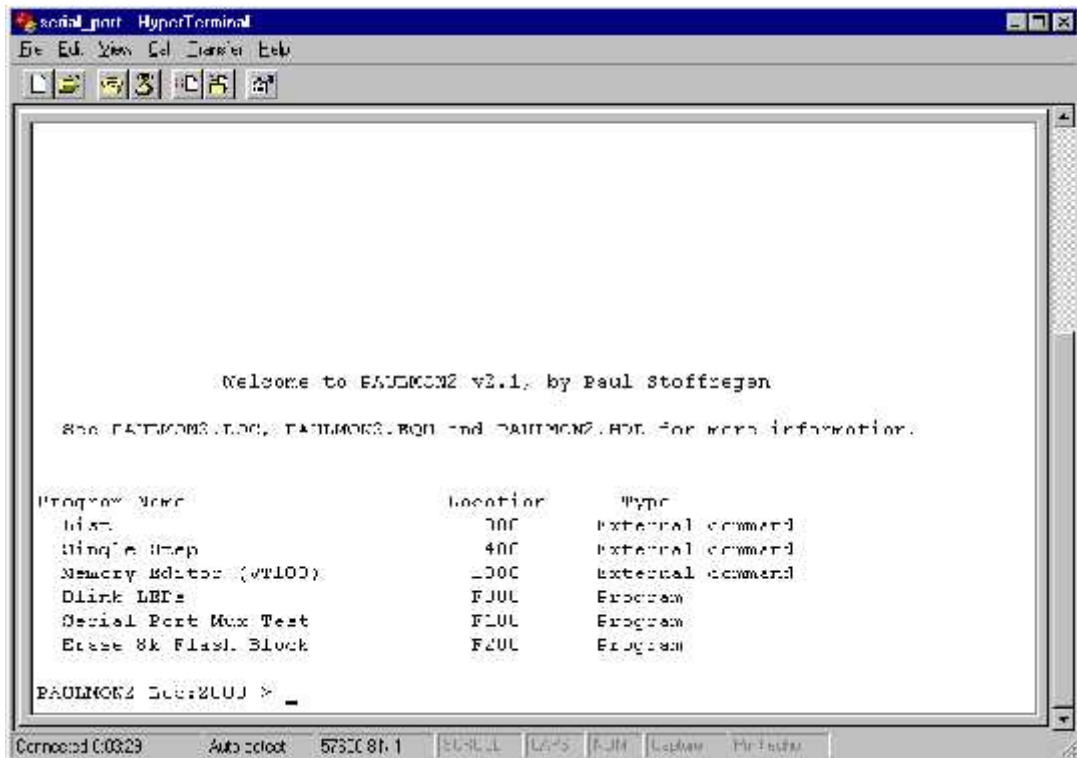


Figure (5.4): Dial up Screen

After performing all the previous procedures, the connection will be ready between the PC and the 8051 microcontroller.

5.2.3 Standard serial cable (straight through)

In this system, the serial cable is needed to make the connection between the PC and 8051 microcontroller, this cable is used to send data between PC and 8051 microcontroller, and this can be connected either with COM1 or with COM2 for both PC and microcontroller.

5.2.4 Assembler or 'C' Compiler, usually AS31 or SDCC.

The 8051 microcontroller deals with the hexadecimal code of the C language or assembly language, so that there is a need for the 'C' Compiler to convert the program from language code to machine code (hexadecimal code).

The system had been programmed in C language related to the requirements of the project, where the SDCC and AS31 compilers are used to convert from the assembly and C code into the hexadecimal code.

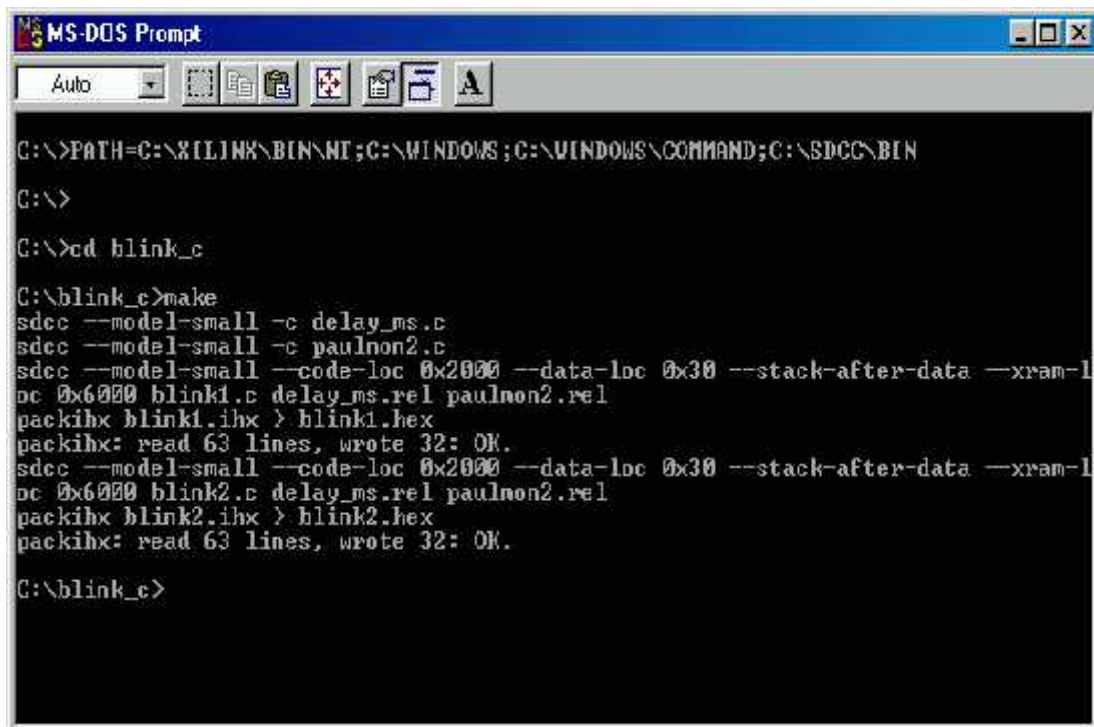
The following steps are explaining how to convert from assembly code to machine code using AS31 compilers:

1. First, install SDCC and AS31 compilers into PC on a specific directory like C Directory.
2. Install the autostart program into PC on the same specific directory like D directory.
3. Go to CMD screen "black screen" then go to the specific directory which the SDCC compiler is found.
4. Type `cd sdcc` then press enter.
5. Type `cd bin` then press enter.
6. Type `as31 c:\name of the assembly program .ASM` then press enter.

After performed all the previous procedures successfully you will see (Begin Pass1 and Pass2) as the result in the black screen.

The following steps are explaining how to convert from C code to machine code using SDCC compiler:

1. First, install SDCC and AS31 compilers into PC on a specific directory like C Directory.
2. Install the autostart program into PC on the same specific directory like D directory.
3. Go to CMD screen "black screen" then go to the specific directory which the SDCC compiler is found.
4. Type path=c:\xilinx\bin\nt;c:\windows;c:\windows\command;c:\sdcc\bin then press enter
5. Type cd <folder name> (in which program file as .c and make file is exist) then press enter.
6. Type make then press enter.



```
MS-DOS Prompt
Auto
C:\>PATH=C:\XILINX\BIN\NT;C:\WINDOWS;C:\WINDOWS\COMMAND;C:\SDCC\BIN
C:\>
C:\>cd blink_c
C:\blink_c>make
sdcc --model-small -c delay_ms.c
sdcc --model-small -c paulnon2.c
sdcc --model-small --code-loc 0x2000 --data-loc 0x30 --stack-after-data --xram-loc 0x6000 blink1.c delay_ms.rel paulnon2.rel
packihx blink1.ihx > blink1.hex
packihx: read 63 lines, wrote 32: OK.
sdcc --model-small --code-loc 0x2000 --data-loc 0x30 --stack-after-data --xram-loc 0x6000 blink2.c delay_ms.rel paulnon2.rel
packihx blink2.ihx > blink2.hex
packihx: read 63 lines, wrote 32: OK.
C:\blink_c>
```

Figure (5.5): MS_DOS

5.2.5 Text Editor Program, e.g. notepad

The Editor Program like notepad is necessary in order to write all assembly and C programs. The assembly program should be saved with extension .ASM on a specific directory or as .C if the program is written in C language and after compiling the program, the hexadecimal program will be saved automatically on the same specific directory.

5.3 Function description

Function description included the code needed to perform the following tasks.

5.3.1 To Use PPI Set ports and system parameters

```
xdata at 0xF900 unsigned char p82c55_port_d;
xdata at 0xF901 unsigned char p82c55_port_e;
xdata at 0xF902 unsigned char p82c55_port_f;
xdata at 0xF903 unsigned char p82c55_def_config;
xdata at 0xF800 unsigned char p82c55_port_a;
xdata at 0xF801 unsigned char p82c55_port_b;
xdata at 0xF802 unsigned char p82c55_port_c;
xdata at 0xF803 unsigned char p82c55_abc_config;
// the following function are written to all over the system
void putchar(char c); // for char written to LCD
void motor1_right(void); // to move the motor1 write
void motor1_left(void); // to move the motor1 left
void motor2_right(void);
void motor2_left(void);
```

```
void timer0_isr(void) interrupt 1; // to initialize the timer
void switch_gates (void); // to take data from gates switches
void floor1_switch (void); // to take data from floors switches
void floor2_switch (void);
volatile unsigned char hours; // definitions for timer
volatile unsigned char minutes;
volatile unsigned char seconds;
volatile bit time_change_flag;
unsigned char a,aa,c1;
unsigned char b,bb,c2;
unsigned char cc;
bit print_to_lcd=0; // to print data on LCD
```

5.3.2 Timer initialization

The 87C52 chip on the development board includes three built in timers, two of which can you easily. (Timer 1 generates the serial port baud rate and usually cannot be used). Here the total time of the system must be less than 1 sec.

❖ How to configured Timer 0

Timer 0 usually configured using these basic steps:

1. Stop the timer and clear the overflow flag
2. Set the mode of operation
3. Write the timer's initial starting value
4. Enable the interrupt (if interrupt also be used)
5. Start the timer

5.3.2.1 Code for Enable Timer

```
IE = 0;                // turn off all interrupts
hours = minutes = seconds = 0; // zero hours, minutes, seconds
TR0 = 0;              // make sure timer 0 is stopped
TF0 = 0;             // clear the overflow flag
TMOD &= 0xF0;        // set to mode 0 (timer1 unchanged)
TL0 = TH0 = 0;       // clear the timer 0 value
time_changed_flag = 0;
TR0 = 1;             // start the timing
IP = 0;              // set interrupt priorities (all low)
IE = 0x82;           // enable timer0 interrupt

while (1) {
    hours = minutes = seconds = 0;
    while (seconds!=0){
        }
    if (time_changed_flag) {
        time_changed_flag = 0;
        print_fast(",");
    }
}
```

5.3.3 Display data on LCD

Finally, the system will display data on LCD through serial interface.

```
lcd_init();
lcd_clear();
```

```
lcd_home();  
print_to_lcd=1;  
lcd_set_xy(1, 1);  
print_fast(" FLOOR ONE: FREE PLACES %d",count1); // for main lcd  
floor1  
lcd_set_xy(2, 1);  
print_fast(" FLOOR TWO: FREE PLACES %d",count2); //for main lcd floor2  
print_to_lcd=0;
```

5.4 Serial Interface between the Microcontroller and PC

The microcontroller become able to make a serial connection with external PC, so the user can see the system variables and store data to analyze it. This can achieve by doing the following steps:

1. Connect the PC to the microcontroller using serial cable.
2. Open the HyperTerminal and set it to com1 and 11520 bit/sec.
3. Press Enter, now you can see system variable.

5.5 General System Flowchart

- The general flowchart for the system can be described in following steps:
- The switches will send signals to the microcontroller via its ports.
- The microcontroller will takes these signals, and process them to perform the action related with it.
- The action will be sent to the LCDs and motor to take the action required. For more details see the flowchart (figure 5.6).

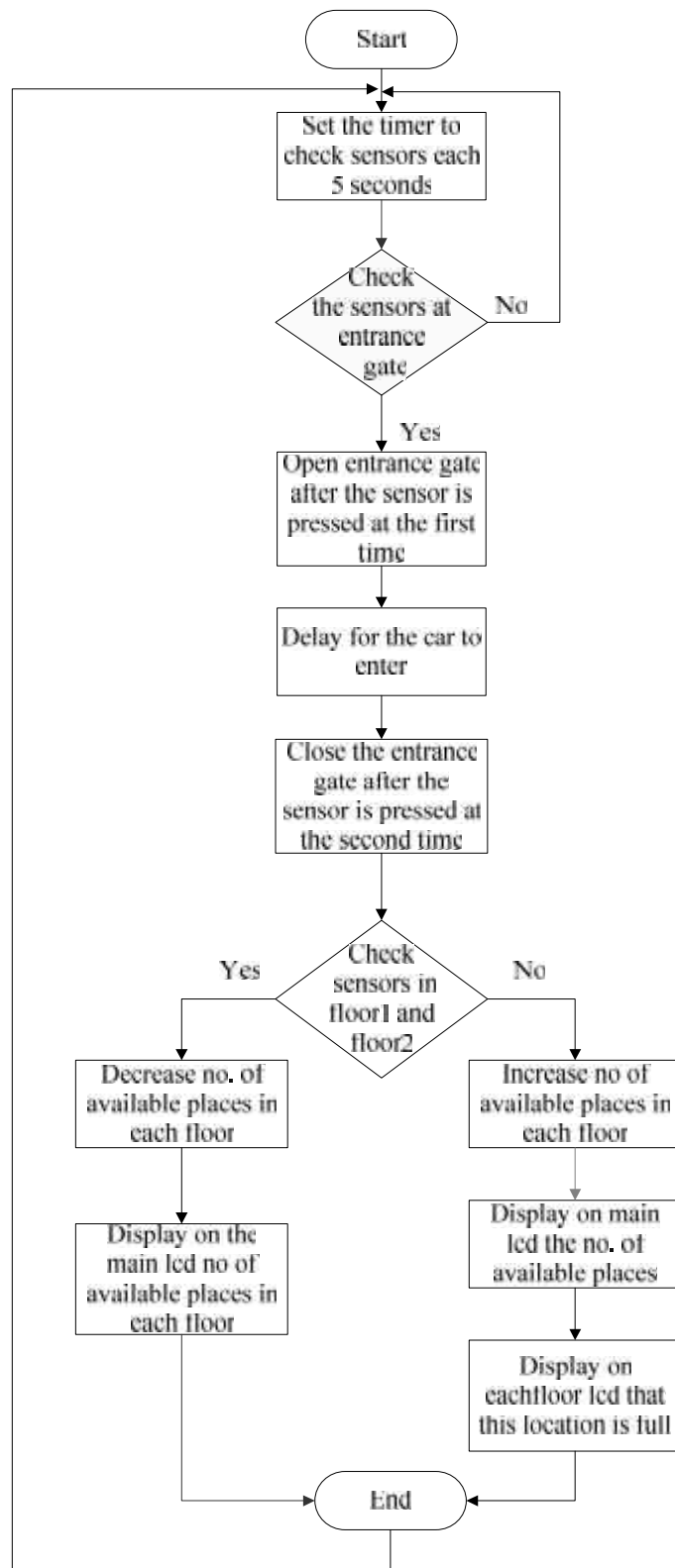


Figure (5.6): General Flowchart

5.6 System Operational Flowchart

These flowcharts shows the functions of the programs and algorithms written to make the system work properly, for each part of the system there is an algorithm written to control the function of this part and all these algorithms are joined in one program to control the overall behavior of the system. In this section, we will show the system operational flowchart as following.

5.6.1 Sensors of Gates Flow Chart

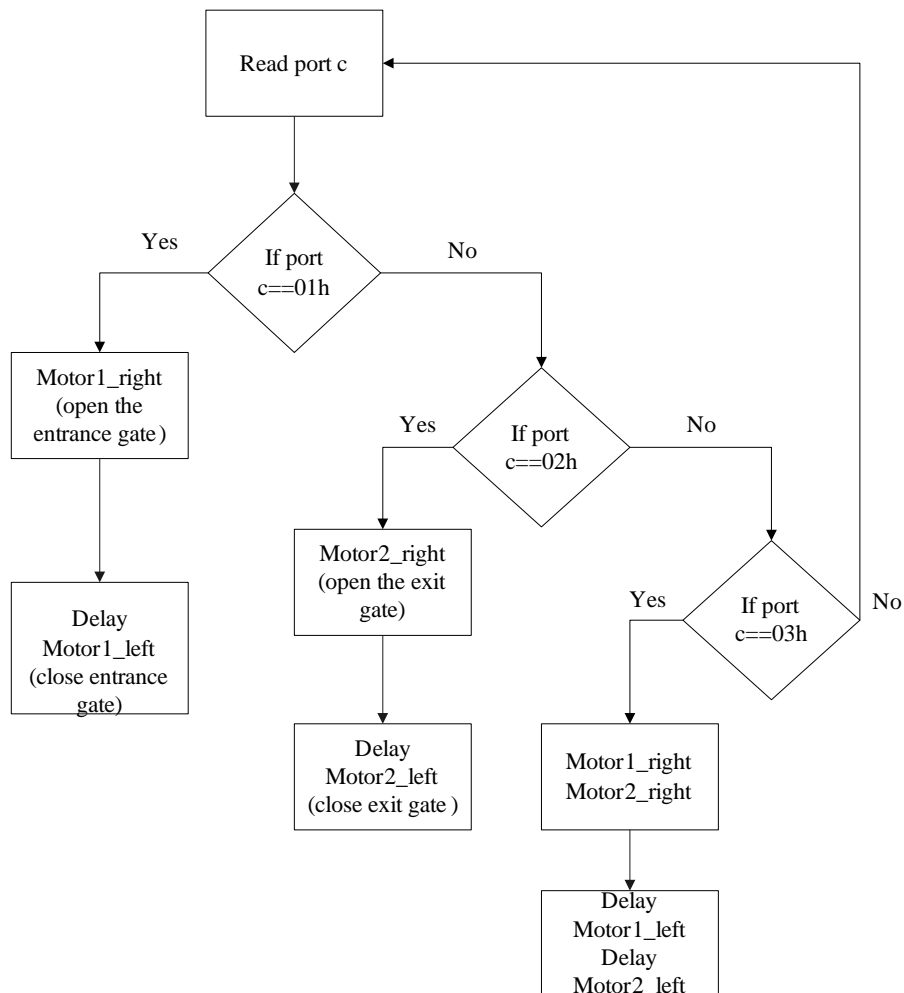


Figure (5.7): Sensors of Gates Flow Chart

As shown in previous (figure 5.7) the remote control initialization consists of many steps such as:

- Set port c as input port.
- Check port c reading and anding it with the value 0xff.

- If the resulting value was 0x01, the sensor at entrance gate is pressed after the first two wheels of the car passed over it and gives signal to the microcontroller to open the gate (motor1_right), then after a period of time (delay) the sensor become off after the back wheels of the car passed over it and then close the gate (motor1_left).
- If the resulting value was 0x02, the sensor at exit gate is pressed after the first wheels of the car passed over it and gives signal to the microcontroller to open the gate (motor2_right), then after a period of time (delay) the sensor becomes off after the second wheels of the car passed over it , and then close the gate (motor2_left).
- If the resulting value was 0x03, the sensors at entrance and exit gates are pressed and give signals to the microcontroller to open both of the gates (motor1_right, motor2_right), then after a period of time (delay) close both of the gates (motor1_left, motor2_left).

5.6.2 Floor One Flow Chart

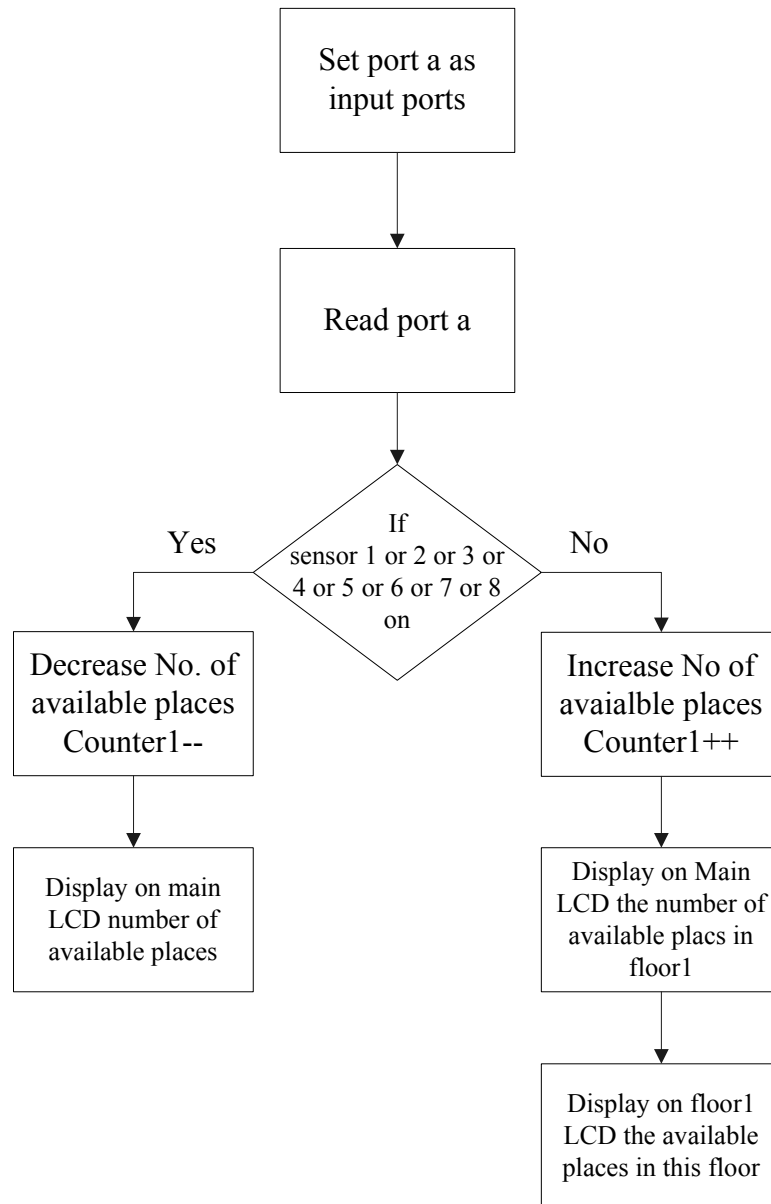


Figure (5.8): Floor One Sensors Flow Chart

As shown in previous (figure 5.8) floor one has many signals which perform specific operation and the previous (figure 5.7) demonstrate how the signal perform its specific action, and consist of many steps such as:

- Initialize port a as input port.

- Read the signal from port a, then anding the reading with the value 0xff.
- If the reading signal was 0x01 or 0x02 or 0x03 or 0x04 or 0x05 or 0x06 or 0x07, sensors at location (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8) are on, and this means that one or more of these places is reserved or not available, and according to this decreasing the number of free places in this floor, and then display on the main LCD the number of available places in floor1, and display on floor1 LCD the available places in this floor.
- If the reading signal was not any of 0x01 or 0x02 or 0x03 or 0x04 or 0x05 or 0x06 or 0x07, sensors at location (1 or 2 or 3 or 4 or 5 or 6 or 7 or 8) are off, and this means that one or more of these places is available, and according to this increasing the number of free places in this floor, and then display on the main LCD the number of available places in floor1, and display on floor1 LCD the available places in this floor.

5.6.3 Floor Two Flow Chart

Floor two is the same as floor one instead of using port b as input port.

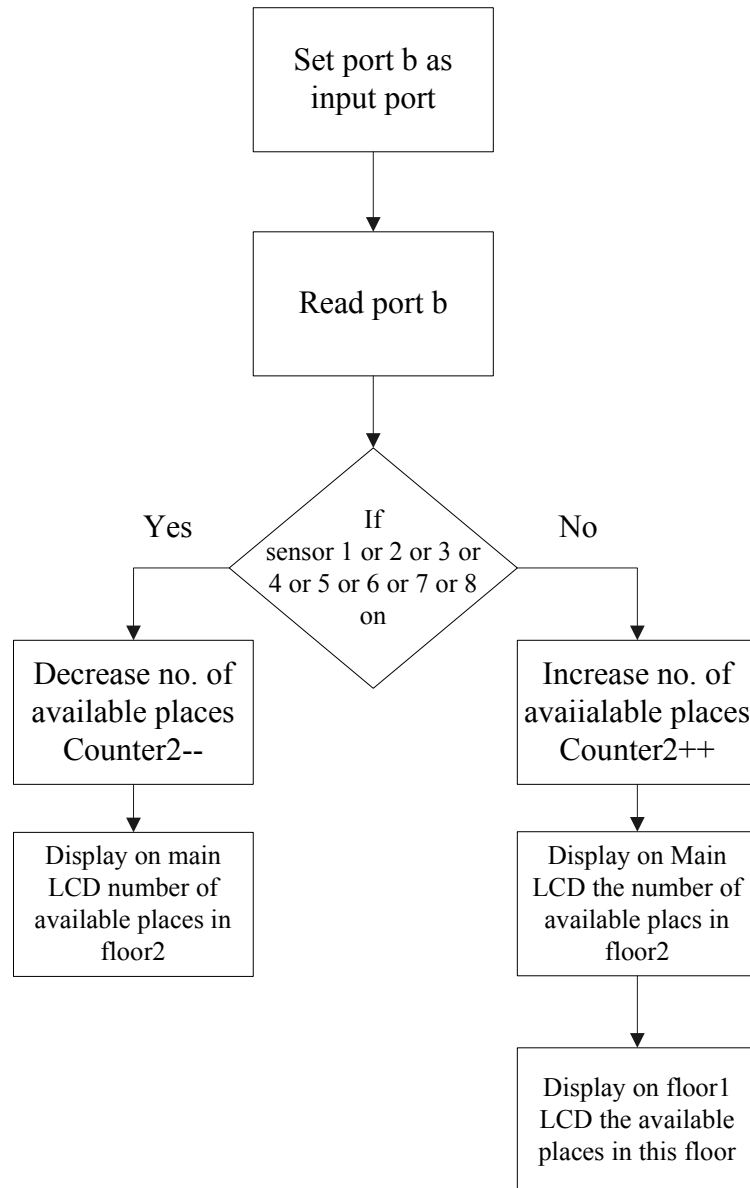


Figure (5.9): Floor Two Sensors Flow Chart

5.6.4 Main LCD Flow Chart

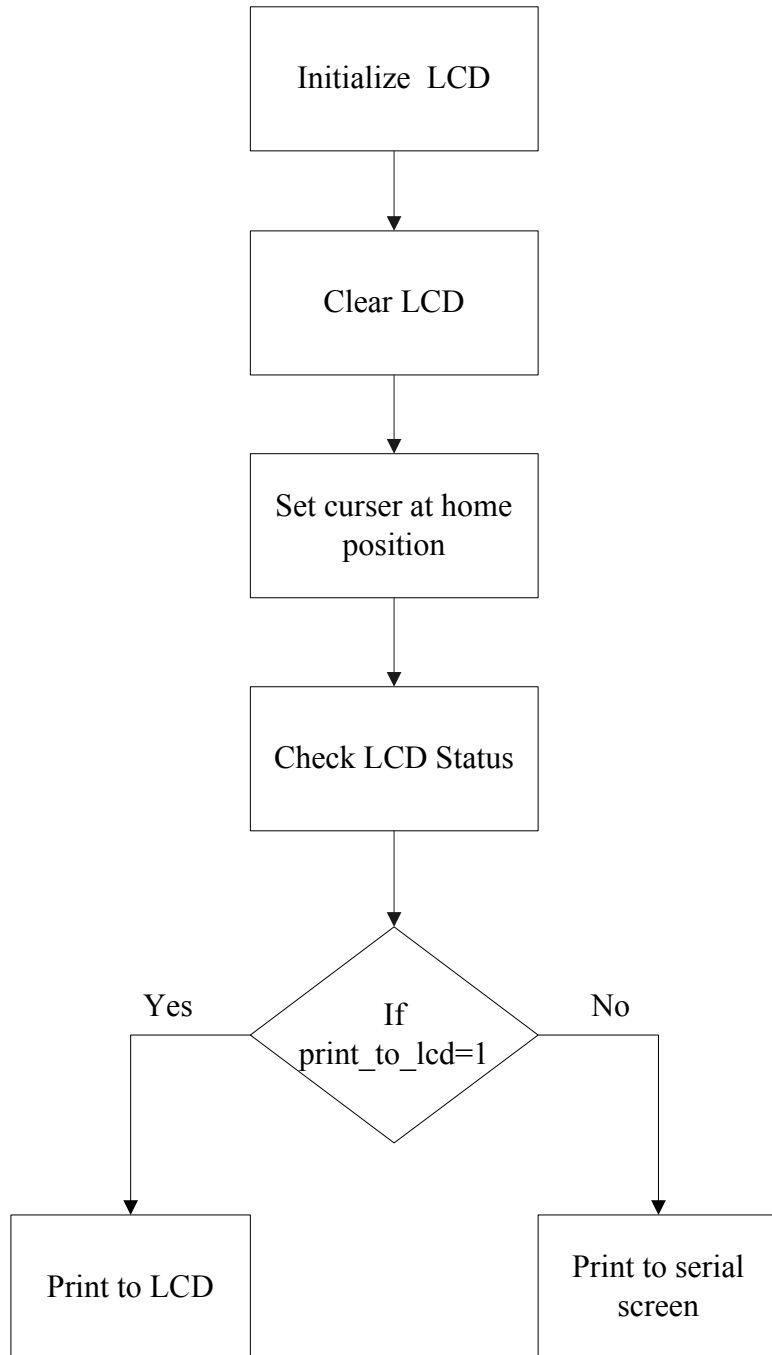


Figure (5.10): Main LCD Flow Chart

As shown in previous (figure 5.10) the LCD has many signals to perform its operation, and the previous (figure 5.9) demonstrate how the LCD perform its specific action, and consist of many steps such as:

- Initialize main LCD, to turn the LCD and cursor on.
- Clear the LCD to write to it.
- Set the cursor of the LCD at home position or at the beginning of line one of the LCD.
- Then check if `print_to_lcd 1` display on the LCD, and if `print_to_lcd 0` then display on the serial port.

5.6.5 Motor1 and Motor2 Flow Chart

Motor needed to open and close the entrance and exit gates. Motor function gets its information from the sensors of the gates and according to them decides what to do, as shown in figure (5.11) when sensors at entrance or exit gates become on, the motors will be turn right to open the gates, and after a delay time it will turn left to close the gate.

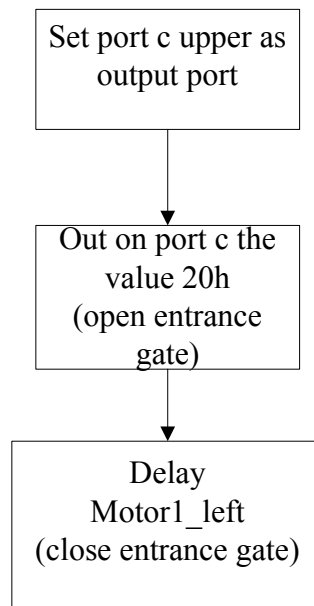


Figure (5.11) Motor1 Flow Chart

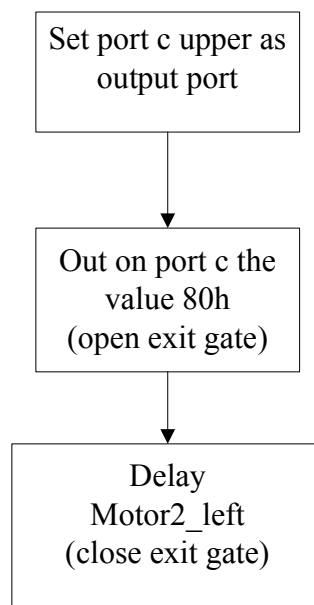


Figure (5.12):Motor2 Flow Chart

5.7 Algorithms and Pseudocode

```
Public Class Form1
```

```
    Inherits System.Windows.Forms.Form
```

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Button1.Click
```

```
        Timer1.Interval = 30000 'set the timer to check the sensors each 30 second
```

```
        Timer1.Start() 'initialization of timer 1
```

```
        Gates_Sensors() 'sensor of the gate initialization
```

```
        Floor1_Sensors() 'sensor of floor1 initialization
```

```
        Floor2_Sensors() 'sensor of floor2 initialization
```

```
    End Sub
```

```
    Private Sub Timer1_Tick_1(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Timer1.Tick
```

```
    End Sub
```

```
Sub Gates_Sensors() 'gate of sensor sub code
```

```
    Dim c As Short
```

```
    Out(&H37AS, 8S) 'h=hex, s=short, 37A=control register
```

```
    c = Inp(&H379) 'read parallel port status register
```

```
    If c And 8 Then 'if entrance gate sensor pressed
```

```
        Motor1_Right() 'open the gate move the motor to the right
```

```
        delay(50) 'delay after gate close
```

```
        Motor1_left() 'close the gate
```

```
    ElseIf c = 16 Then 'if exit gate sensor pressed
```

```
        Motor2_Right() 'open the gate
```

```
        delay(50) 'delay after gate close
```

```

    Motor2_left() 'close the gate move the motor to the left
ElseIf c = 24 Then 'if both entrance and exit gates sensors pressed
    Motor1_Right() 'open entrance and exit gates
    Motor2_Right()
    delay(50)

    Motor1_left() 'close entrance and exit gates
    Motor2_left()
End If
End Sub

Sub Motor1_Right() 'sub code for movement the motor to the right=open
    Out(&H37AS, 0) 'write control word to set ports a,b,c as output ports
    Out(&H378, 128)
    Out(&H37A, 1)
    Out(&H378, 1)
    Out(&H37AS, 3)
    ' delay(10)

    'Motor1_left()
End Sub

Sub Motor1_left() 'left movement to the motor
    Out(&H37A, 1)
    Out(&H378, 0)
    Out(&H37AS, 3)
End Sub

Sub Motor2_Right() 'right movement for the motor
    Out(&H37AS, 0)

```

```
Out(&H378, 128)
Out(&H37A, 1)
Out(&H378, 2)
Out(&H37AS, 2)
'delay(50)
'Motor2_left()
End Sub
```

```
Sub Motor2_left() 'left movement to the motor
Out(&H37A, 1)
Out(&H378, 0)
Out(&H37AS, 2)
End Sub
```

```
Sub Floor1_Sensors() 'read the signal in each sensor in floor 1
Dim a As Short
Dim c1 As Integer
c1 = 0

Out(&H37AS, 8S)
a = Inp(&H379S)
If (a And 32) Then
    ' c1 -= 1
    Main1_lcd(c1) 'put the number of free places in the main LCD

Else
    c1 += 1
    Main1_lcd(c1)
    Floor1_lcd(1) 'put the exactly free location in sub LCD
End If
```

If (a And 64) Then

 c1 -= 1

 Main1_lcd(c1)

Else

 c1 += 1

 Main1_lcd(c1)

 Floor1_lcd(2)

End If

If (a And 128) Then

 c1 -= 1

 Main1_lcd(c1)

Else

 c1 += 1

 Main1_lcd(c1)

 Floor1_lcd(1)

End If

Out(&H37AS, 4S)

If (a And 8) Then

 c1 -= 1

 Main1_lcd(c1)

Else

 c1 += 1

 Main1_lcd(c1)

```
Floor1_lcd(2)
End If
End Sub
```

```
Sub Floor2_Sensors() 'the same as floor1
```

```
Dim b As Short
Dim c2 As Integer = 0
Out(&H37AS, 4)
b = Inp(&H379S)
```

```
If (b And 16) Then
```

```
    Main2_lcd(c2)
```

```
Else
```

```
    c2 += 1
```

```
    Main2_lcd(c2)
```

```
    Floor2_lcd(1)
```

```
End If
```

```
If (b And 32) Then
```

```
    c2 -= 1
```

```
    Main2_lcd(c2)
```

```
Else
```

```
    c2 += 1
```

```
    Main2_lcd(c2)
```

```
    Floor2_lcd(2)
```

```
End If
```

If (b And 64) Then

 c2 -= 1

 Main2_lcd(c2)

Else

 c2 += 1

 Main2_lcd(c2)

 Floor2_lcd(1)

End If

If (b And 128) Then

 c2 -= 1

 Main2_lcd(c2)

Else

 c2 += 1

 Main2_lcd(c2)

 Floor2_lcd(2)

End If

End Sub

Sub Main1_lcd(ByVal c11 As Integer)

 Dim s As String

 Dim arr1 As String() = New String() {"F", "L", "O", "O", "R", "1", ":",
"c11"}

 Dim j As Integer

 Main1_lcd_init() 'initialization to main LCD

```
'Main1_lcd_cursor(1, 0) 'put the cursor in the first position
For j = 0 To arr1.Length
    Main1_lcd_write(arr1(j))
Next
End Sub
```

```
Sub Main1_write_control(ByVal control As Short)
    Out(&H37AS, 0)    'control register
    Out(&H378S, 128) 'control Word
    Out(&H37AS, 2)   'port b
    Out(&H378S, 0)   'clear Rs
    Out(&H37AS, 1)
    Out(&H37AS, 3)   'port a
    Out(&H378S, control)
    Out(&H37AS, 1)
    Out(&H37AS, 2)
    Out(&H378S, 1)
    Out(&H37AS, 1)
    delay(600)
    Out(&H37AS, 2)
    Out(&H378S, 0)
    Out(&H37AS, 1)
End Sub
```

```
Public Sub Main1_lcd_init() 'sub LCD initialization
    Main1_write_control(1)
    delay(60000)
    Main1_write_control(56)
    delay(60000)
    Main1_write_control(14)
```

```
    delay(6000)
    Main1_write_control(6)
    delay(6000)
    Main1_write_control(128)
End Sub
```

```
Sub Main1_lcd_write(ByVal out1 As String)
```

```
'Write character on LCD
```

```
    Out(&H37AS, 0) 'control register
```

```
    Out(&H378S, 128) 'control Word
```

```
    Out(&H37AS, 2) 'port B
```

```
    Out(&H378S, 4) 'set Rs
```

```
    Out(&H37AS, 1)
```

```
    Out(&H37AS, 3) 'port a
```

```
    Out(&H378S, Asc(out1))
```

```
    Out(&H37AS, 1)
```

```
    Out(&H37AS, 2)
```

```
    Out(&H378S, 5)
```

```
    Out(&H37AS, 1)
```

```
    delay(600)
```

```
    Out(&H37AS, 2)
```

```
    Out(&H378S, 4)
```

```
    Out(&H37AS, 1)
```

```
End Sub
```

```
'Sub Main1_lcd_cursor(ByVal row As Short, ByVal column As Short)
```

```
' If (row = 1) Then
```

```
'    Main1_write_control(128 + column)
```

```
' Else
```

```
'    Main1_write_control(192 + column)
```



```

'End If
'
' End Sub
'Sub Main1_lcd_home()

Sub Main2_lcd(ByVal c22 As Integer)
'Sub LCD initialization
  Dim j1 As Integer
  Dim arr2 As String() = New String() {"F", "L", "O", "O", "R", "2", ":",
"c22"}

  Main2_lcd_init()
  ' Main2_lcd_cursor(2, 0)
  'Main2_lcd_home()
  For j1 = 0 To arr2.Length
    Main2_lcd_write(arr2(j1))
  Next
End Sub

Sub Main2_write_control(ByVal control As Short)
  Out(&H37AS, 0) 'control register
  Out(&H378S, 128) 'control Word
  Out(&H37AS, 2) 'port b
  Out(&H378S, 0) 'clear Rs
  Out(&H37AS, 1)
  Out(&H37AS, 3) 'port a
  Out(&H378S, control)
  Out(&H37AS, 1)
  Out(&H37AS, 2)
  Out(&H378S, 1)

```

```
Out(&H37AS, 1)
delay(600)
Out(&H37AS, 2)
Out(&H378S, 0)
Out(&H37AS, 1)
End Sub
```

```
Public Sub Main2_lcd_init()
Main2_write_control(1)
delay(60000)
Main2_write_control(56)
delay(60000)
Main2_write_control(14)
delay(6000)
Main2_write_control(6)
delay(6000)
Main2_write_control(192)
End Sub
```

```
Sub Main2_lcd_write(ByVal out1 As String)
```

```
Out(&H37AS, 0) 'control register
Out(&H378S, 128) 'control Word
Out(&H37AS, 2) 'port B
Out(&H378S, 4) 'set Rs
Out(&H37AS, 1)
Out(&H37AS, 3) 'port a
Out(&H378S, Asc(out1))
Out(&H37AS, 1)
Out(&H37AS, 2)
```

```
Out(&H378S, 5)
Out(&H37AS, 1)
delay(600)
Out(&H37AS, 2)
Out(&H378S, 4)
Out(&H37AS, 1)
End Sub
```

```
'Sub Main2_lcd_cursor(ByVal row As Short, ByVal column As Short)
' If (row = 1) Then
'   Main2_write_control(128 + column)
' Else
'   Main2_write_control(192 + column)
'End If
'
' End Sub
```

```
Sub Floor1_lcd(ByVal aa1 As Short)
```

```
    Floor1_lcd_init()

    If (aa1 = 1 Or aa1 = 2) Then
        Floor1_lcd_write("A")
        Floor1_lcd_write(":")
        Floor1_lcd_write(aa1)
    Else
        Floor1_lcd_write("B")
        Floor1_lcd_write(":")
        Floor1_lcd_write(aa1)
    End If
```

End Sub

Sub Floor1_write_control(ByVal control As Short)

Out(&H37AS, 12) 'control register

Out(&H378S, 128) 'control Word

Out(&H37AS, 13) 'port c

Out(&H378S, 0) 'clear Rs

Out(&H37AS, 14)

Out(&H37AS, 15) 'port a

Out(&H378S, control)

Out(&H37AS, 14)

Out(&H37AS, 13)

Out(&H378S, 1)

Out(&H37AS, 14)

delay(600)

Out(&H37AS, 13)

Out(&H378S, 0)

Out(&H37AS, 14)

End Sub

Public Sub Floor1_lcd_init()

Floor1_write_control(1)

delay(60000)

Floor1_write_control(56)

delay(60000)

Floor1_write_control(14)

delay(6000)

Floor1_write_control(6)

delay(6000)

End Sub

```
Sub Floor1_lcd_write(ByVal out1 As String)
```

```
    If (out1 = 1 Or out1 = 2) Then
```

```
        Floor1_lcd_cursor(1, out1 + 2)
```

```
    Else
```

```
        Floor1_lcd_cursor(2, out1 + 2)
```

```
    End If
```

```
    Out(&H37AS, 12) 'control register
```

```
    Out(&H378S, 128) 'control Word
```

```
    Out(&H37AS, 13) 'port B
```

```
    Out(&H378S, 4) 'set Rs
```

```
    Out(&H37AS, 14)
```

```
    Out(&H37AS, 15) 'port a
```

```
    Out(&H378S, Asc(out1))
```

```
    Out(&H37AS, 14)
```

```
    Out(&H37AS, 13)
```

```
    Out(&H378S, 5)
```

```
    Out(&H37AS, 14)
```

```
    delay(600)
```

```
    Out(&H37AS, 13)
```

```
    Out(&H378S, 4)
```

```
    Out(&H37AS, 14)
```

```
End Sub
```

```
Sub Floor1_lcd_cursor(ByVal row As Short, ByVal column As Short)
```

```
    If (row = 1) Then
```

```
        Floor1_write_control(128 + column)
```

```
    Else
```

```
        Floor1_write_control(192 + column)
```

```
End If
End Sub

Sub Floor2_lcd(ByVal bb1 As Short)
```

```
    Floor2_lcd_init()
    Floor2_lcd_cursor(1, 0)
    'Floor2_lcd_home()
    If (bb1 = 1 Or bb1 = 2) Then
        Floor2_lcd_write("A:" & bb1)
        Floor1_lcd_write(":")
        Floor1_lcd_write(bb1)
```

```
Else
    Floor2_lcd_write("B:" & bb1)
    Floor1_lcd_write(":")
    Floor1_lcd_write(bb1)
End If
End Sub
```

```
Sub Floor2_write_control(ByVal control As Short)
```

```
    Out(&H37AS, 12) 'control register
    Out(&H378S, 128) 'control Word
    Out(&H37AS, 13) 'port c
    Out(&H378S, 0) 'clear Rs
    Out(&H37AS, 15)
    Out(&H37AS, 14) 'port a
    Out(&H378S, control)
    Out(&H37AS, 15)
    Out(&H37AS, 13)
```

```
Out(&H378S, 16)
Out(&H37AS, 15)
delay(600)
Out(&H37AS, 13)
Out(&H378S, 0)
Out(&H37AS, 15)
End Sub
```

```
Public Sub Floor2_lcd_init()
    Floor2_write_control(1)
    delay(60000)
    Floor2_write_control(56)
    delay(60000)
    Floor2_write_control(14)
    delay(6000)
    Floor2_write_control(6)
    delay(6000)
End Sub
```

```
Sub Floor2_lcd_write(ByVal out2 As String)
    If (out2 = 1 Or out2 = 2) Then
        Floor1_lcd_cursor(1, out2 + 2)

    Else
        Floor1_lcd_cursor(2, out2 + 2)
    End If
```

```
Out(&H37AS, 12) 'control register
Out(&H378S, 128) 'control Word
Out(&H37AS, 13) 'port c
```

```
Out(&H378S, 64) 'set Rs
Out(&H37AS, 15)
Out(&H37AS, 14) 'port b
Out(&H378S, Asc(out2))
Out(&H37AS, 15)
Out(&H37AS, 13)
Out(&H378S, 80)
Out(&H37AS, 15)
delay(600)
Out(&H37AS, 13)
Out(&H378S, 64)
Out(&H37AS, 15)
End Sub
```

```
Sub Floor2_lcd_cursor(ByVal row As Short, ByVal column As Short)
    If (row = 1) Then
        Floor2_write_control(128 + column)

    Else
        Floor2_write_control(192 + column)
    End If

    delay(60)
End Sub
```

```
'Sub Floor2_lcd_home()
' Floor2_write_control(2)
End Sub
```

```
Sub delay(ByVal time As Integer)
    Dim i As Integer
```



```
For i = 0 To time
Next
End Sub
End Class
```

6

Implementation and Testing

6.1 Preface.

6.2 Implementation.

6.3 Testing.

6.4 Implementation and testing for Integrated System.

Chapter Six

Implementation and Testing

6.1 Preface

In this chapter shows the implementation and testing processes for our system. And demonstrates the methods and procedures used for testing and examine the system operation and behavior.

The implementation and testing was done by using the following tools and components:

- Connectors with different colors.
- All the ICs that are depicted in the design chapter (see chapter 4).
- Wrapper tool for wrapping the connectors on the ICs stands.
- A wire grabber and a wire cutter.
- A digital Millimeter for testing.

This system has more than one issue to be tested. Some testing parts reflect a software, hardware .Also, testing procedures concentrate on a single device independent from the over whole system. Here are the testing issues.

6.2 Implementation

The implementation process is done synchronized with the testing operation, because each implementation phase will take many testing to ensure that there are no errors.

6.3 Testing

In this section we will demonstrate the testing of each subsystem separately.

6.3.1 Testing 8051 Downloading Programs:

Testing the ability to download a program on the 8051 development kit.



Figure (6.1): Port Testing Example (LEDs)

We test this circuit by C program language that reads the signal and store it, this circuit had been tested many times to emphasize better results, and we use this program to testing this circuit.

We tested the port E and the LEDs by using the following C program, the port E was acts as output port.

```
#include "delay_ms.h"
#include "paulmon2.h"
/* these are the memory-mapped locations used to access */
/* the two 82C55 chips */
xdata at 0xF800 unsigned char p82c55_port_a;
xdata at 0xF801 unsigned char p82c55_port_b;
```

```
xdata at 0xF802 unsigned char p82c55_port_c;  
xdata at 0xF803 unsigned char p82c55_abc_config;  
xdata at 0xF900 unsigned char p82c55_port_d;  
xdata at 0xF901 unsigned char p82c55_port_e;  
xdata at 0xF902 unsigned char p82c55_port_f;  
xdata at 0xF903 unsigned char p82c55_def_config;
```

```
#define VERBOSE
```

```
code unsigned char pattern_table[] = {  
0x7F, /* 01111111 */  
0x3F, /* 00111111 */  
0x1F, /* 00011111 */  
0x8F, /* 10001111 */  
0xC7, /* 11000111 */  
0xE3, /* 11100011 */  
0xF1, /* 11110001 */  
0xF8, /* 11111000 */  
0xFC, /* 11111100 */  
0xFE, /* 11111110 */  
0xFC, /* 11111100 */  
0xF8, /* 11111000 */  
0xF1, /* 11110001 */  
0xE3, /* 11100011 */  
0xC7, /* 11000111 */  
0x8F, /* 10001111 */  
0x1F, /* 00011111 */  
0x3F, /* 00111111 */  
255};  
  
code unsigned char delay_table[] = {  
90, 70, 50, 40, 40, 40, 40, 50, 70, 90, 70, 50, 40, 40, 40, 40, 50, 70, 0};
```

```

/* zero marks end of table */
void main()
{
    unsigned char i =0;
    p82c55_abc_config = 128;    /* all ports outputs */
    p82c55_def_config = 128;
    while (1) {
        if (delay_table[i] > 0) {
            p82c55_port_e = pattern_table[i];
            delay_ms(delay_table[i]);
#ifdef VERBOSE
            pm2_pstr("Pattern=0x");
            pm2_phex(pattern_table[i]);
            pm2_pstr(" for delay=");
            pm2_pint8u(delay_table[i]);
            pm2_newline();
#endif
            i++;
        } else {
            i = 0;
        }
        if (pm2_esc()) pm2_restart();
    }
}

```

This C program can use for other ports and test them.

6.3.2 Motors and H-Bridges Testing

Testing the motors and H-Bridges are implemented by connecting them to shown in figure (6.2), and applying the following code to activate the motors in forward and reverse direction.

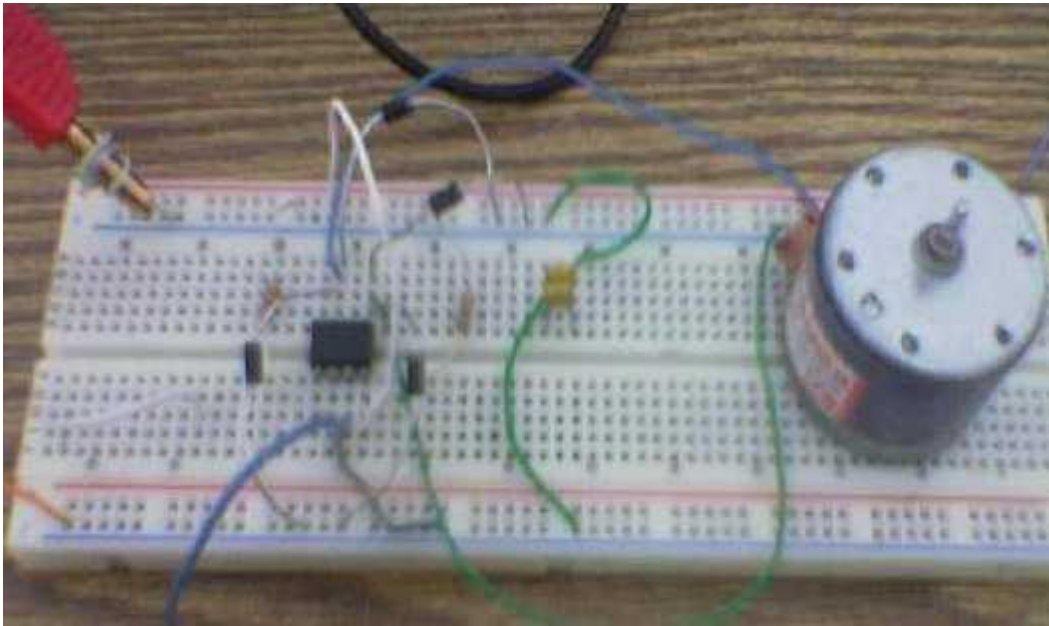


Figure (6.2): Motor circuit

We test this circuit by C program language, and we use this program to testing this circuit. See sample (1.a) written in C language and sample (1.b) written in VB.net at Appendix A.

6.3.2.1 Option One Using C Language for Motor Testing

See sample (1.a) at Appendix A.

6.3.2.2 Option Two Using VB.net for Motor Testing

See sample (1.a) at Appendix A.

6.3.3 Switches Testing

Testing the switches is implemented by connecting them to as shown in figure (6.3) and applying the following software.



Figure (6.3.a): Switch circuit off

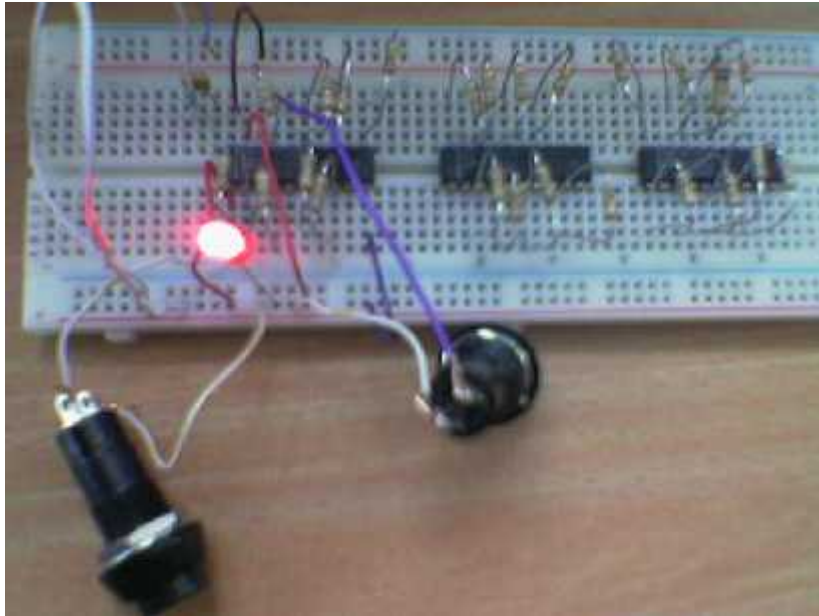


Figure (6.3.b): Switch circuit on

We test this circuit by C program language, this circuit had been tested many times to emphasize better results, and we use this program to testing this circuit.

6.3.3.1 Option One Using C Language for Switch Testing

See sample (2.a) at Appendix A.

6.3.3.2 Option Two Using VB.net for Switch Testing

See sample (2.b) at Appendix A.

6.3.4 LCDs Testing

Testing the LCDs is implemented by connecting them with the parallel port. The LCD circuit tested by C program and VB.net program.

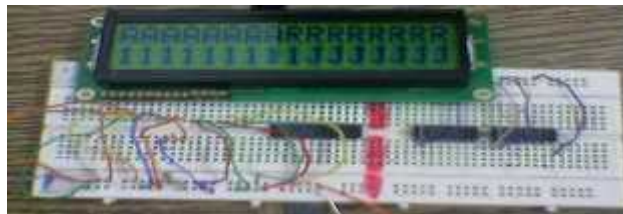


Figure (6.4): LCD Testing

6.3.4.1 Option One Using C Language for LCD Testing

See sample (3.a) at Appendix A.

6.3.4.2 Option Two Using Vb.net for LCD Testing

See sample (3.b) at Appendix A.

6.3.5 Access the parallel port using VB.net

To access the parallel port using the VB.net we get an input.dll library, which must add to system32, and then we define a parallel port that will be used as an object to open and access the ports, as the followed:

6.3.5.1 Testing Output Ports in VB.net

Testing the output on parallel port implemented by connecting them as shown in figure (6.4), and applying code. See sample (4.a) at Appendix A.

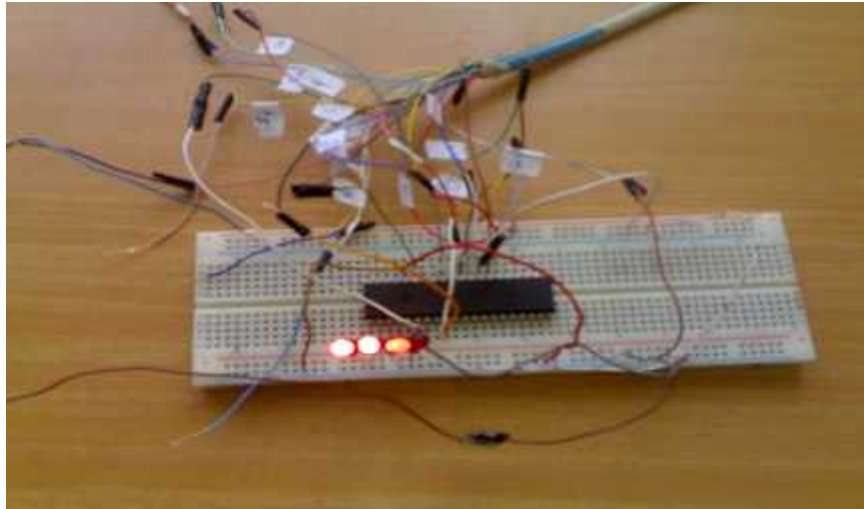


Figure (6.5): Output Port Testing

6.3.5.2 Testing input ports in VB.net

Testing the input on parallel port implemented by connecting them as shown in figure (6.5), and applying program. See sample (4.b) at Appendix A.

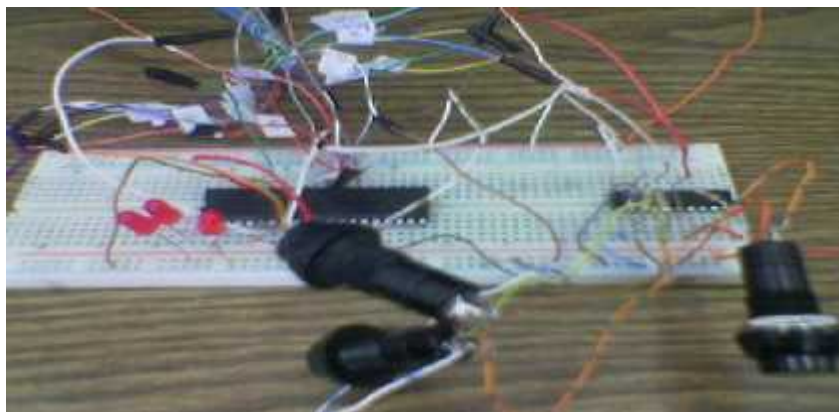


Figure (6.6): Input Port Testing

6.4 Implementation and Testing for Integrated System

As we mentioned previously after completing any circuit that we described it in chapter four we use the C language And VB.net language to test each subsystem and test every input and output with different cases and speed.

After completing VB.net program with test the whole system on the program, the finally results are successfully on software and hardware control.

7

Conclusion and Future Work

7.1 Preface

7.2 Conclusion

7.3 Future work

Chapter Seven

Conclusion and Future Work

7.1 Preface

This chapter represents the conclusions extracted during designing and implementing it and illustrates the system implementation achievements and output.

- Team spent 30-40 hours a week in the university lab working on this project. We learned a lot and used everything we had to solve the problems. We had many limitations on this project that made it even more challenging.
- Building this system was a great way of combining all of the knowledge that we have learned over the five years at university. We were able to create a project and build it. Building the system taught us how to accomplish a goal. We really wanted to have our system completed before handing this documentation, but unfortunately things didn't go as we wanted.
- This project proved that our ideas are available for implementation in a project that resembles a real-life.
- Our system is smart car parking system. This system aims to make Smart Park controlled electrically.
- The main device in the system is the 8051 microcontroller; all other devices are controlled through programming it. While the other devices are maintained in various subsystems which are:
 - Sensor subsystem: contains interlocking push buttons for each location.
 - LCDs subsystem: contains three LCDs (main LCD and two LCDs for one each floor).
 - Motors Subsystem: contains the motors and their H-Bridges circuits.

- For programming the 8051 microcontroller, we used C++ program language and VB.NET program language.
- Each device was tested individually in its own circuit to study its behavior and make sure it works properly and can do its expected job.

The Smart Car Parking System has achieved the main ideas prepared to. It is now ready to be applied on the pc. The main features that achieved are:-

- 1- The team enables to control the hardware applications connected with pc through parallel port.
- 2- The team enables to make the system ease to use by everyone.
- 3- Controlling the hardware applications connected with pc through a parallel port, the team enables to send data from switches connected with pc through parallel port, then analyze the data and send the related voltage signal to the parallel port to run the application related to that signal and then set the specified data on LCDs.
- 4- The team enables to send signal to motors connected with pc through parallel port and controlling its movements.

7.2 Conclusions

After the team finish the design and implementation of the project and integrated overall system.

The team concludes that, there is ability to control a park through parallel port connected with PC or microcontroller like 8051.

7.2.1 Problems

Here are problems faced the project team during the system implementation:

7.2.1.1 Hardware Problems

- The basic problem is that the 8051 microcontroller in 8051 development kit was burned and another one is not available in this country, so the team was bought through the internet from the main company in Germany, and it was sent by the buyer to the Palestinian in Bethlehem.
- A problem occurred in programming the 8051 microcontroller, there is no programmer to program this microcontroller.
- The team took an 8051 from another team to continue the team project work but unfortunately was burned.
- The last choice is to use the parallel port in PC and because of that we covert all our software from C programming Language to VB.net to solve the problem.
- Internal damage in some devices because of wrong connections, or high voltages or currents supplied to the devices during the implementation.

7.2.1.2 Software problems

In 8051 programming a lot of considerations should be taken in using this kit to connect it with PC through serial port and during writing and running the programs on the microcontroller we faced some initializations and declarations problems.

In parallel port lot of initialization and declaration problems during writing LCD code.

7.3 Future work

During the system period the team enables to achieve the main goals he started to do them. Although there are sum techniques that could be used to reach the objects in less time but we started our system from the zero and we used the most available and simple equipments to emphasize the result and prove this of controlling the hardware application connected with pc.

1. Use a robot that follows a particular line and determines whether there is any availability of space. Like a watchman.
2. Use cameras for each location (not sensors), to give information a bout free places.
3. To wireless technology in this system to connects all over the system.
4. Use charging system for every customer enters the park, to control the hours for each customer put his car in the park and its related cost.

References

Books

- Sencer Yeralan and Ashutosh Ahluwalia, Programming Interfacing 8051 Microcontroller, Addison Wesley, 1995.
- David M Calcutt and Frederick J Cowan and G Hassan Parchizadeh, 8051 Microcontroller, A member of the Hodder Headline Group, 1998.
- Hans_Peter Messmer, The Independence PC Hardware Book 3rd Edition, Addison Wesley, 1997.

Internet

- http://www.hillelectronics.com/html/pro_hiltron_HM810R.htm
- <http://www.winpicprog.co.uk>
- <http://www.ptc2.com/vb/showthread.php?t=93>
- <http://www.plcs.net/>
- <http://rapidshare.de/files/22449655/GR2-CPACImagingPro.rar.html>
- <http://www.ptc2.com/vb/showthread.php?t=118>
- <http://chaokhun.kmitl.ac.th/~kswichit/rtc/rtc.htm>
- <http://chaokhun.kmitl.ac.th/~kswichit/>
- <http://www.epanorama.net/links/microprocessor.html#8051>
- <http://www.tkne.net/vb/register.php>
- <http://www.pjrc.com/>
- <http://www.Interfacing the Standard Parallel Port.htm>
- <http://www.Interfacing the LCD module to PC parallel port ElectroSOfts com.htm>

Appendix

Appendix A

Code Samples

1. Motor

(1.a) Option One Using C Language for Motor Testing

```
void motor1_right()
{
    p82c55_abc_config=146; // set port c output
    p82c55_port_c=0x08; // open entrance gate
    delay_ms(5);
    motor1_left()
}

void motor1_left()
{
    p82c55_port_c=0x00; // close entrance gate
}

void motor2_right()
{
    p82c55_abc_config=146; // set port c output
    p82c55_port_c=0x20; // open exit gate
    delay_ms(5);
    motor2_left()
}

void motor2_left()
{
    p82c55_port_c=0x00; // close exit gate
}
```

(1.b) Option Two Using VB.net for Motor Testing

```
Sub Motor1_Right()  
    Out(&H37AS, 0) 'write control word to set ports a,b,c as output ports  
    Out(&H378, 128)  
    Out(&H37A, 1)  
    Out(&H378, 2)  
    Out(&H37AS, 3)  
    delay(10)  
    Motor1_left()  
End Sub
```

2. Switch

(2.a) Option One Using C Language for Switch Testing

```
void floor1_switch ()  
{  
    p82c55_abc_config=155;  
    aa=p82c55_port_a;  
    a=(aa & 0x01);  
    if(a==0x01)  
        c1--;  
    Main1_Lcd(c1);  
    floor_lcd();  
    else c1++;  
    Main1_lcd(c1);  
    floor1_lcd();  
    a=(aa & 0x02);
```

```
if(a==0x02)
c1--;
Main1_lcd(c1);
floor1_lcd()
else c1++;
Main1_lcd(c1);
floor1_lcd();
a=(a & 0x03);
if(a==0x03)
c1--;
Main1_lcd(c1);
floor1_lcd()
else c1++;
Main1_lcd(c1);
floor1_lcd();
a=(a & 0x04);
if(a==0x04)
c1--;
Main1_lcd(c1);
floor1_lcd()
else c1++;
Main1_lcd(c1);
floor1_lcd();
a=(a & 0x05);
if(a==0x05)
c1--;
Main1_lcd(c1);
floor1_lcd()
else c1++;
Main1_lcd(c1);
```

```
floor1_lcd();
a=(a & 0x06);
if(a==0x06)
c1--;
Main1_lcd(c1);
floor1_lcd()
else c1++;
Main1_lcd(c1);
floor1_lcd();
a=(a & 0x07);
if(a==0x07)
c1--;
Main1_lcd(c1);
floor1_lcd()
else c1++;
Main1_lcd(c1);
floor1_lcd();
a=(a & 0x08);
if(a==0x08)
c1--;
Main1_lcd(c1);
floor1_lcd()
else c1++;
Main1_lcd(c1);
floor1_lcd();
}
```


(2.b) Option Two Using VB.net for Switch Testing

```
Sub Floor1_Sensors()
```

```
    Dim a As Short
```

```
    Dim c1 As Integer
```

```
    a = Inp(&H379)
```

```
    Out(&H37AS, 9)
```

```
    If (a And 0) Then
```

```
        c1 -= 1
```

```
        Main1_lcd(c1)
```

```
    Else : c1 += 1
```

```
        Main1_lcd(c1)
```

```
        Floor1_lcd(1)
```

```
    End If
```

```
    If (a And 1) Then
```

```
        c1 -= 1
```

```
        Main1_lcd(c1)
```

```
    Else : c1 += 1
```

```
        Main1_lcd(c1)
```

```
        Floor1_lcd(2)
```

```
    End If
```

```
    If (a And 2) Then
```

```
        c1 -= 1
```

Main1_lcd(c1)

Else : c1 += 1

Main1_lcd(c1)

Floor1_lcd(3)

End If

If (a And 3) Then

c1 -= 1

Main1_lcd(c1)

Else : c1 += 1

Main1_lcd(c1)

Floor1_lcd(4)

End If

If (a And 4) Then

c1 -= 1

Main1_lcd(c1)

Else : c1 += 1

Main1_lcd(c1)

Floor1_lcd(5)

End If

If (a And 6) Then

c1 -= 1

Main1_lcd(c1)

```
Else : c1 += 1
      Main1_lcd(c1)
      Floor1_lcd(6)
End If
```

```
End Sub
```

3. LCD

(3.a) Option One Using C Language for LCD Testing

```
#include "lcd_driver.h"
#include <stdio.h>
#include <8051.h>
#include <paulmon2.h>
#include "delay_ms.h"
/*****/
void initial_lcd();
void clear_lcd();
void lcd_write(char cc);
void lcd_cursor(char row,char column);
void write_control(char control );
xdata at 0xF900 unsigned char p82c55_port_d;
xdata at 0xF901 unsigned char p82c55_port_e;
xdata at 0xF902 unsigned char p82c55_port_f;//E, R/W, RS
xdata at 0xF903 unsigned char p82c55_def_config;
void main(){
p82c55_def_config=128;
```

```
initial_lcd();
clear_lcd();
lcd_write('H');
}

void write_control(char control){
p82c55_port_f=0x00;
p82c55_port_d=control;
p82c55_port_f=0x01;
p82c55_port_f=0x00;
delay_ms(5);
}

void initial_lcd(){
write_control(0x38)
delay_ms(5);
write_control(0x0e);
delay_ms(5);
write_control(0x06);
delay_ms(5);
}

void clear_lcd(){
write_control(0x01);
delay_ms(5);
}

void lcd_write( char cc){
lcd_cursor(1,1);
p82c55_port_f=0x04;
p82c55_port_d=cc;
p82c55_port_f=0x01;
```

```

p82c55_port_f=0x00;
delay_ms(5);
}

void lcd_cursor(char row,char column){
    if (row==1)
write_control(0x80+column);
    else
write_control(0xc0+column)
delay_ms(5);
}

```

(3.b) Option Two Using VB.net for LCD Testing

```

Public Class Form1
    Inherits System.Windows.Forms.Form
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim s As String
        lcd_init()
        lcd_cursor(1, 0)
        ' write_control(192)
        lcd_write("A")
        'lcd_write("I")
        'lcd_write("H")
        'lcd_write("A")
        'lcd_write("N")
    End Sub
    Sub write_control(ByVal control As Short)

```

```
Out(&H37AS, 8) 'control register
Out(&H378S, 128) 'control Word
Out(&H37AS, 10) 'port B
Out(&H378S, 0) 'clear Rs
Out(&H37AS, 9)
Out(&H37AS, 11) 'port a
Out(&H378S, control)
Out(&H37AS, 9)
Out(&H37AS, 10)
Out(&H378S, 1)
Out(&H37AS, 9)
delay(50000)
Out(&H37AS, 10)
Out(&H378S, 0)
Out(&H37AS, 9)
End Sub
```

```
Public Sub lcd_init()
    write_control(1)
    delay(60000000)
    write_control(56)
    delay(60000000)
    write_control(14)
    delay(60000000)
    write_control(6)
    delay(60000000)
End Sub
```

```
' Sub write_char(ByVal c As String)
'   Dim i As Integer
```

```
' For i = 0 To c.length
"   lcd_write(Asc(c))
'Next
'End Sub
```

```
Sub lcd_write(ByVal out1 As String)
```

```
    Out(&H37AS, 8) 'control register
    Out(&H378S, 128) 'control Word
    Out(&H37AS, 10) 'port B
    Out(&H378S, 4) 'set Rs
    Out(&H37AS, 9)
    Out(&H37AS, 11) 'port a
    Out(&H378S, Asc(out1))
    Out(&H37AS, 9)
    Out(&H37AS, 10)
    Out(&H378S, 5)
    Out(&H37AS, 9)
    delay(50)
    Out(&H37AS, 10)
    Out(&H378S, 4)
    Out(&H37AS, 9)
```

```
End Sub
```

```
Sub lcd_cursor(ByVal row As Short, ByVal column As Short)
```

```
    If (row = 1) Then
        write_control(128 + column)
    Else
        write_control(192 + column)
    End If
```

```
End Sub
```

```

Sub delay(ByVal time As Integer)
    Dim i As Integer = 0
    For i = 0 To time
        Next
    End Sub
End Class

```

(4.a) Output Ports in VB.net Using Parallel Port

```

Public Class Form1
    Inherits System.Windows.Forms.Form
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Out(&H37AS, 8) 'control register
        Out(&H378S, 128) 'control Word
        out(&H37AS, 11) 'port a
        Out(&H378S, 255)
        Out(&H37AS, 9)
    End Sub
End Class

```

(4.b) Input ports in VB.net Using Parallel Port

```

Public Class Form1
    Inherits System.Windows.Forms.Form
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

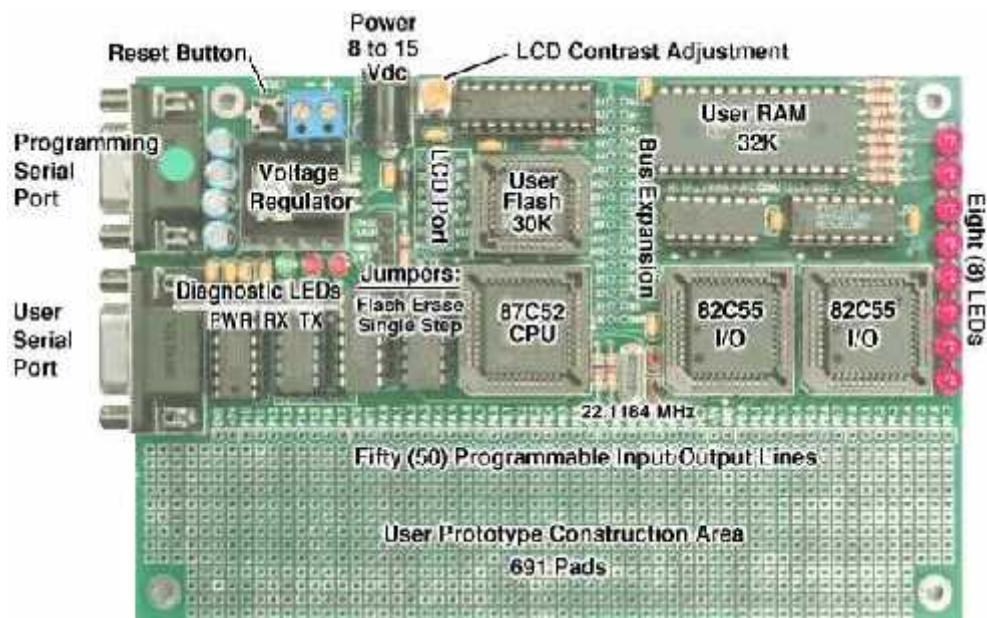
```



```
    TextBox1.Text = Inp(&H379S)
End Sub
End Class
```

8051 Development Board Data Sheet

The 8051 development board provides an easy-to-use and low-cost way to develop your 8051 based microcontroller projects, without purchasing any other special equipment, such as IC programmers or emulators. All required software is available as a free download, including a C compiler !



Features:

- Standard 87C52 CPU clocked at 22.1184 MHz
- 50 I/O lines!! All I/O lines are clearly labeled and available at the edge of the prototype construction area.
- 32k SRAM, program variables and code (24k usable for code download)
- 30k Flash ROM, non-volatile program storage and data logging

- High speed baud rates: 115200, 75600, 38400, etc. All standard baud rates are supported (except 300 baud)
- Display port, works with standard character-based LCDs. A 16x2 display will be available from PJRC (see photo below)
- Eight LEDs, controlled by 8 dedicated I/O lines (not shared with the 50 I/O lines)
- Bus expansion with 4 chip select signals, for adding UARTs, A/D converters and other bus-based peripheral chips.
- Unregulated, polarity-protected DC voltage input with 2 position terminal block.
- PAULMON2 monitor for easy code development without additional equipment.

LCD Display Port

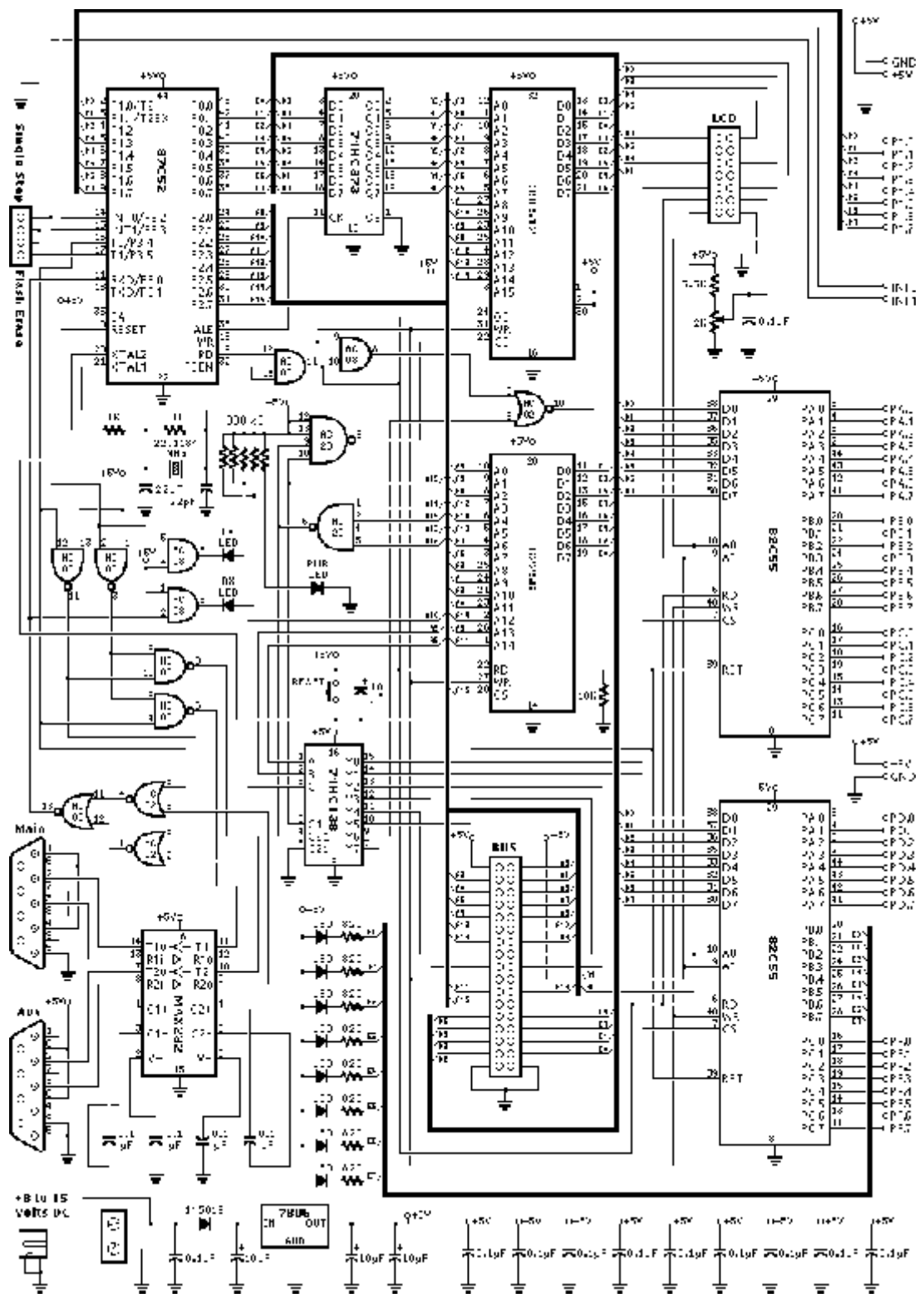
The 8051 development board's LCD port provides the 14 signals needed for standard character based LCD modules. A 20x2 display is available from PJRC :



Requirements

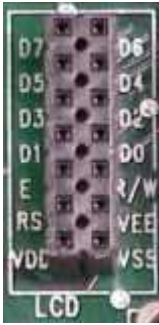
- DC Voltage, 8 to 15 volts (regulated), 50 mA or 300 mA with LCD.
- PC Computer with Serial Port, Linux or Microsoft Windows 95-OSR2, 98, 98SE, NT4, 2000.
- Assembler or Compiler, usually AS31 or SDCC
- Terminal Emulation Program, e.g. HyperTerminal (windows), Minicom (linux)
- Text Editor Program, e.g. Notepad, Vi, Emacs
- Standard 9 pin serial cable (straight through, not null-modem!).

Schematic



Ports

1. LCD Port



The LCD port provides the 14 standard signals required to interface to nearly all standard alpha-numeric character mode displays from 8 to 80 characters. **VDD** provides +5 volt regulated power to the display (**VSS** is Ground). **VEE** ranges from 0 (maximum intensity) to 2 volts (minimum intensity) with the adjustment of the variable resistor located just above next to the power input.

The **E** signal is an active high enable, which is asserted when the processor makes a memory access within the address range of 0xFE00 to 0xFEFF. **RS** and **R/W** are control lines for the display. In order to meet the timing requirements for all standard LCD displays, these are connected to the processor's address lines so they are asserted and remain stable while **E** is asserted. Because of this, separate locations are used to read and write to the LCD. See the memory map page for details.

2. I/O Lines

The 8051 development board provides 50 dedicated I/O lines, which are accessible along the top edge of the board's prototype construction area. Together with the 50 I/O lines, 4 pads provide easy access to the regulated +5 volt power from the board's voltage regulator.



The **P1.0** through **P1.7** are connected directly to the 87C52's port #1. These pins are the easiest to use as single bits. In assembly, they are written using "CLR P1.4" or "SETB P1.4", and they are read using "MOV C, P1.4" (moves the bit value into the carry bit). In C (using SDCC, with #include <8051.h>), they are accessed using

names such as "P1_4". For example: `if (!P1_3) printf("pin P1.3 is low");` The 8051 port pins are quasi-bidirectional, which essentially means that you must write a 1 (which is the default) to the pin to cause it to act as an input.

Signals INT0 and INT1 connect to the 87C52's two interrupt pins. The 87C52 can be configured to execute an interrupt routine associated with each pin when it is low or when a falling edge occurs. In the low level sensitive setting, the interrupt service code usually takes some action which causes the hardware to stop driving the pin low, so that another interrupt does not immediately occur when the interrupt service code returns to the main program. If interrupts are not enabled, these pins can be accessed as ordinary P3.2 and P3.3 port pins. INT1 is also connected to the SINGLE STEP jumper and will be shorted to ground if that jumper is installed.

PA.0 to PA.7, PB.0 to PB.7, and PC.0 to PC.7 are connected to a 82C55 chip mapped at 0xF800. PD.0 to PD.7 and PF.0 to PF.7 are connected to a second 82C55 chip mapped at 0xF900, and they correspond to ports A and C on that second chip, but that second chip's ports are labeled D, E, and F to avoid confusion with the ports from the first 82C55 chip (port E connects to the 8 LEDs). Each 82C55 chip is controlled with 4 memory mapped locations, one to read or write each 8-bit port, and a 4th register to configure the 82C55 chip. See the memory map page for details. The 82C55 must first be configured by writing a byte to its config register, and then the three locations which access the ports may be used.

Jumpers



A 4-pin header may be used for two optional jumpers. The upper two pins may be shorted to erase the flash rom, and the lower two pins may be shorted to enable the single-step feature.

The **FLASH ERASE** jumper causes T1 (also P3.5, pin 17 on the 82C52) to be shorted to ground. During normal operation, this does not erase the flash. When PAULMON2 boots, it reads this pin and if it remains shorted for 256 consecutive reads, the PAULMON2 will erase the flash rom chip. Normally the flash rom is erased from the PAULMON2 menu using the 'Z' command. However, if you have loaded a program into the flash rom and used an "auto-start" header on it, PAULMON2 will jump to your code instead of presenting the normal menus on the serial port. If your code does not return back to the monitor, then you will be unable to get to the normal PAULMON2 menus and this pin will allow you to erase the chip you can return to the menus and download a new version of your program.

The **SINGLE STEP** jumper shorts the INT1 interrupt pin to ground. This is required to use PAULMON2's single-step feature. The single-step operates by enabling interrupt #1 and using the 8051's feature where 1 instruction is always executed after an interrupt. This can be a nice way to "see" your code run, particularly if you are learning assembly. Due to the interrupt usage, it is rarely useful for debugging sophisticated applications.

Bus Expansion Signals



The 8051 bus signals are available on 34 pins in the center of the board. These signals are intended to be used with bus-style peripherals, such as UARTs and A/D converters. All 16 address and 8 data signals are provided. Two **GND** and two **5V** pins provide 5 volt regulated power from the board's voltage regulator.

The **WR** and **RD** signals are active low strobes for write and read. **WR** is connected directly to the 87C52's WR pin, but **RD** is connected to the 74AC08 AND gate. **RD** is asserted low with either the 87C52's PSEN or RD signal is asserted. This means that either MOVX or MOVC may be used to read your connected peripheral chips. Some peripheral chips call their read pin OE (output enable). Typically, the **RD** pin can connect directly to the peripheral's OE pin.

Four chip select signals, **CS2**, **CS3**, **CS4**, and **CS5** are provided to allow easy connection of most bus-style peripheral chips. Each of these signals is asserted low when and access is made within its 256 byte range. See the memory map page for details.

Power Input



The board accepts unregulated DC voltage, between 8 to 12 volts. A terminal block with screws allows a wide range of wires sizes to easily attach to the board without the need for a connector. A 1N5819 diode protects against reverse polarity, and a standard 7805 linear voltage regulator creates the regulated 5 volts needed by the board's circuitry.

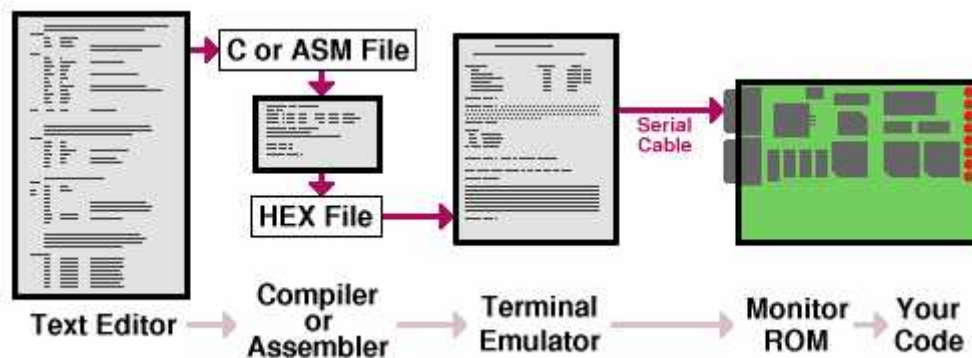
Though the printed maximum voltage is only 12 volts, the board can actually accept up to 30 volts DC. Higher voltages will cause the 7805 voltage regulator to become

hot. The 7805 includes automatic thermal shutdown, but it can become very hot before this upper limit is reached, so caution should be observed if a higher input voltage is used.

The board requires approximately 50 mA when executing code from the flash rom and communicating with a PC on the serial port. Each LED adds about 4 mA. If a LCD with a backlight is used, the backlight will consume considerable current. The 16x2 LCD from PJRC uses approximately 250 mA for its backlight. Additional current also causes the 7805 to heat up, so the board should not be run with more than 12 volts if a LCD backlight is used.

8051 Software Tools Overview

To develop your 8051-based project, you will need three PC-based programs, and a development board with a monitor ROM. Here is the typical data flow using these tools:



For each tool, there you may choose on of several options. The basic tools are:

1. Text Editor

This is where you will compose the code that ultimately will run on your 8051 board and make you project function. All PC operating

systems include a text editor, and there are many free text editors available on the internet. PJRC does not provide a text editor. All Microsoft Windows systems have NOTEPAD, which is a very simple editor. Linux systems usually have VI and EMACS installed, as well as several others.

2. Compiler or Assembler

Your source code will be turned into a .HEX file by either an Assembler or Compiler, depending on your choice of programming language. PJRC provides free downloads of the AS31 assembler and SDCC C Compiler. These free tools are available for Linux-based Systems and Microsoft Windows. It is also possible to use other compilers, such as the Keil C compiler.

3. Terminal Emulator

To communicate with your 8051 board, you must run a terminal emulator program. You will be able to transmit your .HEX file to the board and run its code, observe its results and information it may send to the serial port, and you can also use the terminal emulator to examine and manipulate memory with the Monitor ROM. Microsoft provides HyperTerminal with windows (often it must be installed from the windows cdrom using "add/remove programs"). Linux distributions usually provide minicom.

4. Monitor ROM

The Monitor ROM is 8051 code that runs when your board boots. It provides interactive menus that allow you to download code, run it, manipulate memory and perform other functions. All 8051 development boards from PJRC come with PAULMON2 loaded in

the non-erasable internal memory of the 87C52 chip. Using the monitor, you can cause your code to run on the board. It is also possible to download your code to non-volatile memory on the board together with a "auto-start header" that causes PAULMON2 to run your code automatically when the board boots.

DM74LS244

Octal 3-STATE Buffer/Line Driver/Line Receiver

General Description

These buffers/line drivers are designed to improve both the performance and PC board density of 3-STATE buffers/drivers employed as memory-address drivers, clock drivers, and bus-oriented transmitters/receivers. Featuring 400 mV of hysteresis at each low current PNP data line input, they provide improved noise rejection and high fanout outputs and can be used to drive terminated lines down to 133Ω.

Features

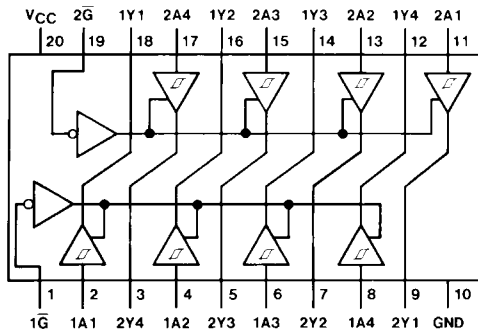
- 3-STATE outputs drive bus lines directly
- PNP inputs reduce DC loading on bus lines
- Hysteresis at data inputs improves noise margins
- Typical I_{OL} (sink current) 24 mA
- Typical I_{OH} (source current) -15 mA
- Typical propagation delay times
 - Inverting 10.5 ns
 - Noninverting 12 ns
- Typical enable/disable time 18 ns
- Typical power dissipation (enabled)
 - Inverting 130 mW
 - Noninverting 135 mW

Ordering Code:

Order Number	Package Number	Package Description
DM74LS244WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
DM74LS244SJ	M20D	20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS244N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram



Function Table

Inputs		Output
\bar{G}	A	Y
L	L	L
L	H	H
H	X	Z

L = LOW Logic Level
H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
Z = High Impedance

DM74LS244 Octal 3-STATE Buffer/Line Driver/Line Receiver

Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	-65°C to +150°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V _{CC}	Supply Voltage	4.75	5	5.25	V
V _{IH}	HIGH Level Input Voltage	2			V
V _{IL}	LOW Level Input Voltage			0.8	V
I _{OH}	HIGH Level Output Current			-15	mA
I _{OL}	LOW Level Output Current			24	mA
T _A	Free Air Operating Temperature	0		70	°C

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 2)	Max	Units	
V _I	Input Clamp Voltage	V _{CC} = Min, I _I = -18 mA			-1.5	V	
HYS	Hysteresis (V _{T+} - V _{T-}) Data Inputs Only	V _{CC} = Min	0.2	0.4		V	
V _{OH}	HIGH Level Output Voltage	V _{CC} = Min, V _{IH} = Min V _{IL} = Max, I _{OH} = -1 mA	2.7			V	
		V _{CC} = Min, V _{IH} = Min V _{IL} = Max, I _{OH} = -3 mA	2.4	3.4			
		V _{CC} = Min, V _{IH} = Min V _{IL} = 0.5V, I _{OH} = Max	2				
V _{OL}	LOW Level Output Voltage	V _{CC} = Min V _{IL} = Max V _{IH} = Min	I _{OL} = 12 mA I _{OL} = Max		0.4 0.5	V	
I _{OZH}	Off-State Output Current, HIGH Level Voltage Applied	V _{CC} = Max V _{IL} = Max	V _O = 2.7V		20	μA	
I _{OZL}	Off-State Output Current, LOW Level Voltage Applied	V _{IH} = Min	V _O = 0.4V		-20	μA	
I _I	Input Current at Maximum Input Voltage	V _{CC} = Max	V _I = 7V		0.1	mA	
I _{IH}	HIGH Level Input Current	V _{CC} = Max	V _I = 2.7V		20	μA	
I _{IL}	LOW Level Input Current	V _{CC} = Max	V _I = 0.4V	-0.5	-200	μA	
I _{OS}	Short Circuit Output Current	V _{CC} = Max (Note 3)		-40	-225	mA	
I _{CC}	Supply Current	V _{CC} = Max, Outputs Open	Outputs HIGH		13	23	mA
			Outputs LOW		27	46	
			Outputs Disabled		32	54	

Note 2: All typicals are at V_{CC} = 5V, T_A = 25°C.

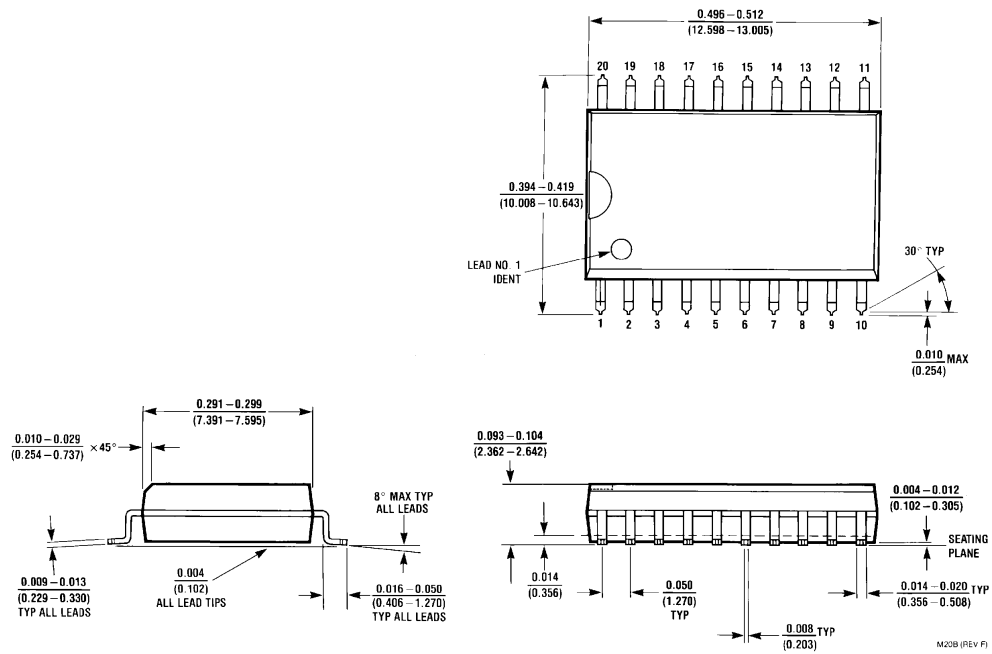
Note 3: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Switching Characteristics

at $V_{CC} = 5V$, $T_A = 25^\circ C$

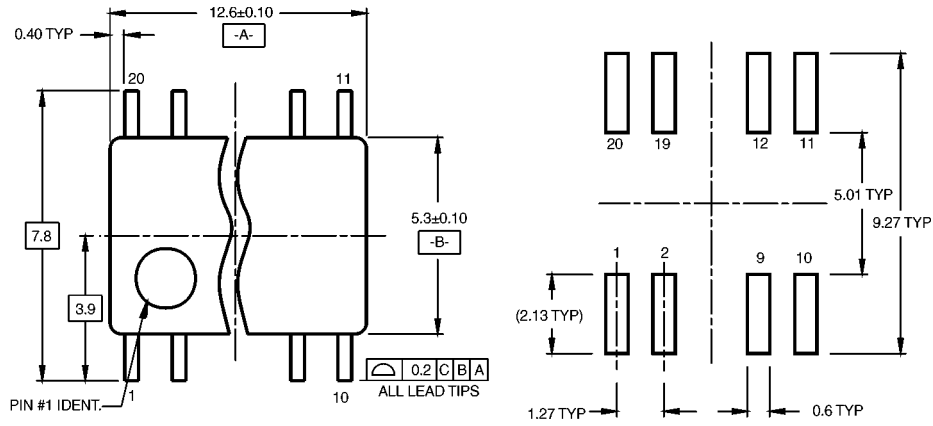
Symbol	Parameter	Conditions	Max	Units
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	$C_L = 45 \text{ pF}$ $R_L = 667\Omega$	18	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	$C_L = 45 \text{ pF}$ $R_L = 667\Omega$	18	ns
t_{PZL}	Output Enable Time to LOW Level	$C_L = 45 \text{ pF}$ $R_L = 667\Omega$	30	ns
t_{PZH}	Output Enable Time to HIGH Level	$C_L = 45 \text{ pF}$ $R_L = 667\Omega$	23	ns
t_{PLZ}	Output Disable Time from LOW Level	$C_L = 5 \text{ pF}$ $R_L = 667\Omega$	25	ns
t_{PHZ}	Output Disable Time from HIGH Level	$C_L = 5 \text{ pF}$ $R_L = 667\Omega$	18	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	$C_L = 150 \text{ pF}$ $R_L = 667\Omega$	21	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	$C_L = 150 \text{ pF}$ $R_L = 667\Omega$	22	ns
t_{PZL}	Output Enable Time to LOW Level	$C_L = 150 \text{ pF}$ $R_L = 667\Omega$	33	ns
t_{PZH}	Output Enable Time to HIGH Level	$C_L = 150 \text{ pF}$ $R_L = 667\Omega$	26	ns

Physical Dimensions inches (millimeters) unless otherwise noted

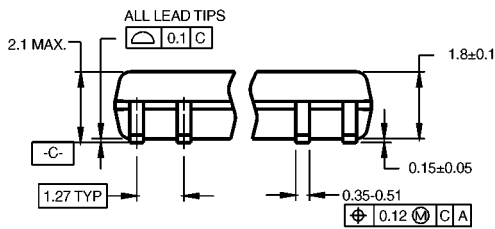


20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide Package Number M20B

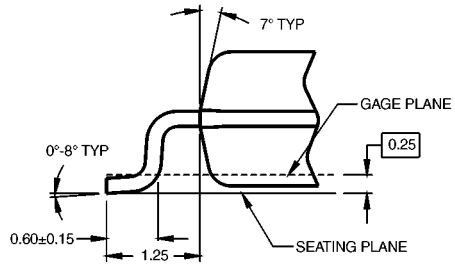
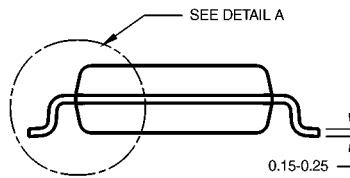
Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



LAND PATTERN RECOMMENDATION



DIMENSIONS ARE IN MILLIMETERS

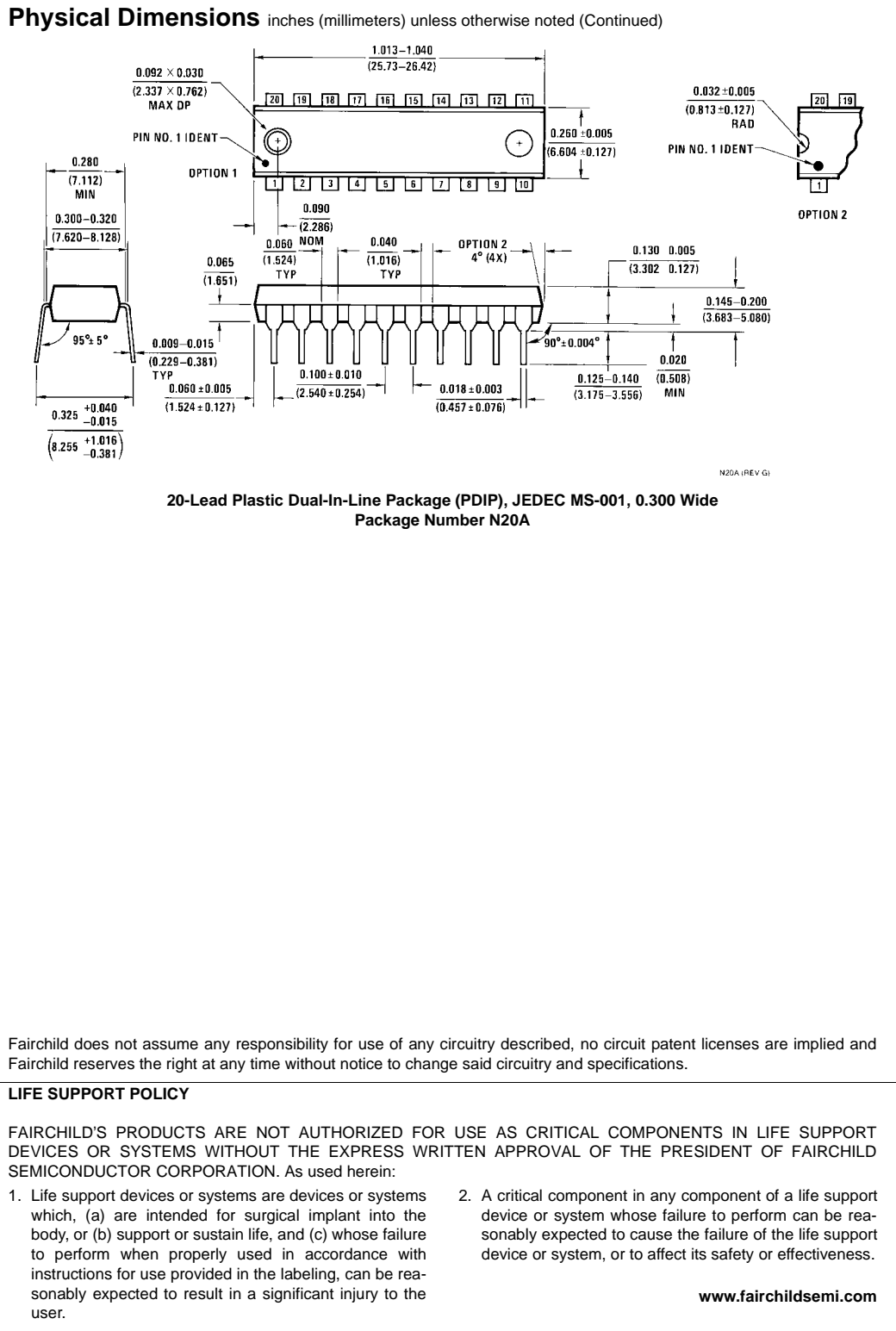


DETAIL A

- NOTES:
- A. CONFORMS TO EIAJ EDR-7320 REGISTRATION, ESTABLISHED IN DECEMBER, 1998.
 - B. DIMENSIONS ARE IN MILLIMETERS.
 - C. DIMENSIONS ARE EXCLUSIVE OF BURRS, MOLD FLASH, AND TIE BAR EXTRUSIONS.

M20DRevB1

**20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
Package Number M20D**



This datasheet has been downloaded from:

www.DatasheetCatalog.com

Datasheets for electronic components.

General Description

The MIC4426/4427/4428 family are highly-reliable dual low-side MOSFET drivers fabricated on a BiCMOS/DMOS process for low power consumption and high efficiency. These drivers translate TTL or CMOS input logic levels to output voltage levels that swing within 25mV of the positive supply or ground. Comparable bipolar devices are capable of swinging only to within 1V of the supply. The MIC4426/7/8 is available in three configurations: dual inverting, dual noninverting, and one inverting plus one noninverting output.

The MIC4426/4427/4428 are pin-compatible replacements for the MIC426/427/428 and MIC1426/1427/1428 with improved electrical performance and rugged design (Refer to the Device Replacement lists on the following page). They can withstand up to 500mA of reverse current (either polarity) without latching and up to 5V noise spikes (either polarity) on ground pins.

Primarily intended for driving power MOSFETs, MIC4426/7/8 drivers are suitable for driving other loads (capacitive, resistive, or inductive) which require low-impedance, high peak current, and fast switching time. Other applications include driving heavily loaded clock lines, coaxial cables, or piezoelectric transducers. The only load limitation is that total driver power dissipation must not exceed the limits of the package.

Note See MIC4126/4127/4128 for high power and narrow pulse applications.

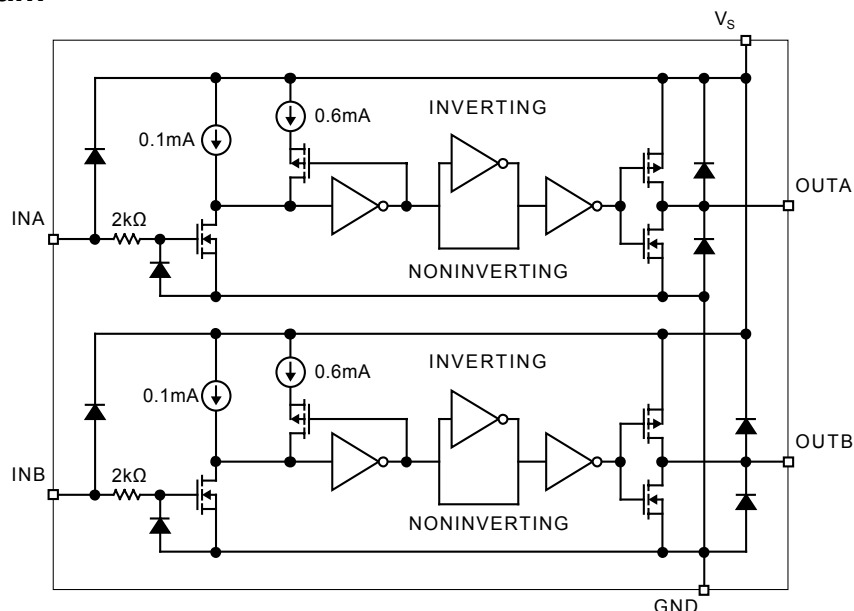
Features

- Bipolar/CMOS/DMOS construction
- Latch-up protection to >500mA reverse current
- 1.5A-peak output current
- 4.5V to 18V operating range
- Low quiescent supply current
 - 4mA at logic 1 input
 - 400µA at logic 0 input
- Switches 1000pF in 25ns
- Matched rise and fall times
- 7Ω output impedance
- <40ns typical delay
- Logic-input threshold independent of supply voltage
- Logic-input protection to -5V
- 6pF typical equivalent input capacitance
- 25mV max. output offset from supply or ground
- Replaces MIC426/427/428 and MIC1426/1427/1428
- Dual inverting, dual noninverting, and inverting/noninverting configurations
- ESD protection

Applications

- MOSFET driver
- Clock line driver
- Coax cable driver
- Piezoelectric transducer driver

Functional Diagram



Ordering Information

Part Number		Temperature Range	Package	Configuration
Standard	Pb-Free			
MIC4426AM*	Contact Factory	-55°C to +125°C	8-Pin SOIC	Dual Inverting
MIC4426BM	MIC4426YM	-40°C to +85°C	8-Pin SOIC	Dual Inverting
MIC4426CM	MIC4426ZM	-0°C to +70°C	8-Pin SOIC	Dual Inverting
MIC4426BMM	MIC4426YMM	-40°C to +85°C	8-Pin MSOP	Dual Inverting
MIC4426BN	MIC4426YN	-40°C to +85°C	8-Pin PDIP	Dual Inverting
MIC4426CN	MIC4426ZN	-0°C to +70°C	8-Pin PDIP	Dual Inverting
MIC4427AM*	Contact Factory	-55°C to +125°C	8-Pin SOIC	Dual Non-Inverting
MIC4427BM	MIC4427YM	-40°C to +85°C	8-Pin SOIC	Dual Non-Inverting
MIC4427CM	MIC4427ZM	-0°C to +70°C	8-Pin SOIC	Dual Non-Inverting
MIC4427BMM	MIC4427YMM	-40°C to +85°C	8-Pin MSOP	Dual Non-Inverting
MIC4427BN	MIC4427YN	-40°C to +85°C	8-Pin PDIP	Dual Non-Inverting
MIC4427CN	MIC4427ZN	-0°C to +70°C	8-Pin PDIP	Dual Non-Inverting
MIC4428AM*	Contact Factory	-55°C to +125°C	8-Pin SOIC	Inverting + Non-Inverting
MIC4428BM	MIC4428YM	-40°C TO +85°C	8-Pin SOIC	Inverting + Non-Inverting
MIC4428CM	MIC4428ZM	-0°C to +70°C	8-Pin SOIC	Inverting + Non-Inverting
MIC4428BMM	MIC4428YMM	-40°C to +85°C	8-Pin MSOP	Inverting + Non-Inverting
MIC4428BN	MIC4428YN	-40°C to +85°C	8-Pin PDIP	Inverting + Non-Inverting
MIC4428CN	MIC4428ZN	-0°C to +70°C	8-Pin PDIP	Inverting + Non-Inverting

*Special order, contact factory.

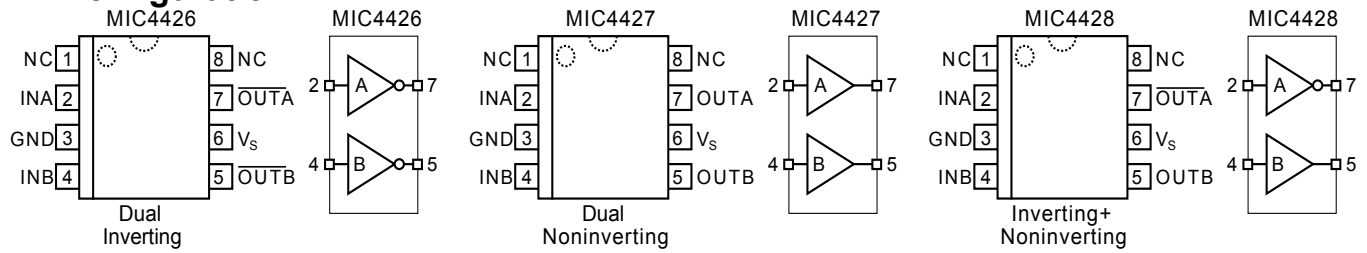
MIC426/427/428 Device Replacement

Discontinued Number	Replacement
MIC426CM	MIC4426BM
MIC426BM	MIC4426BM
MIC426CN	MIC4426BN
MIC426BN	MIC4426BN
MIC427CM	MIC4427BM
MIC427BM	MIC4427BM
MIC427CN	MIC4427BN
MIC427BN	MIC4427BN
MIC428CM	MIC4428BM
MIC428BM	MIC4428BM
MIC428CN	MIC4428BN
MIC428BN	MIC4428BN

MIC1426/1427/1428 Device Replacement

Discontinued Number	Replacement
MIC1426CM	MIC4426BM
MIC1426BM	MIC4426BM
MIC1426CN	MIC4426BN
MIC1426BN	MIC4426BN
MIC1427CM	MIC4427BM
MIC1427BM	MIC4427BM
MIC1427CN	MIC4427BN
MIC1427BN	MIC4427BN
MIC1428CM	MIC4428BM
MIC1428BM	MIC4428BM
MIC1428CN	MIC4428BN
MIC1428BN	MIC4428BN

Pin Configuration



Pin Description

Pin Number	Pin Name	Pin Function
1, 8	NC	not internally connected
2	INA	Control Input A: TTL/CMOS compatible logic input.
3	GND	Ground
4	INB	Control Input B: TTL/CMOS compatible logic input.
5	OUTB	Output B: CMOS totem-pole output.
6	V_S	Supply Input: +4.5V to +18V
7	OUTA	Output A: CMOS totem-pole output.

Absolute Maximum Ratings⁽¹⁾

Supply Voltage (V_S).....	+22V
Input Voltage (V_{IN}).....	$V_S + 0.3V$ to GND – 5V
Junction Temperature (T_J).....	150°C
Storage Temperature.....	–65°C to +150°C
Lead Temperature (10 sec.).....	300°C
ESD Rating ⁽³⁾	

Operating Ratings⁽²⁾

Supply Voltage (V_S).....	+4.5V to +18V
Temperature Range (T_A)	
(A).....	–55°C to +125°C
(B).....	–40°C to +85°C
Package Thermal Resistance	
PDIP θ_{JA}	130°C/W
PDIP θ_{JC}	42°C/W
SOIC θ_{JA}	120°C/W
SOIC θ_{JC}	75°C/W
MSOP θ_{JA}	250°C/W

Electrical Characteristics⁽⁴⁾

4.5V $\leq V_S \leq 18V$; $T_A = 25^\circ C$, **bold** values indicate full specified temperature range; unless noted.

Symbol	Parameter	Condition	Min	Typ	Max	Units
Input						
V_{IH}	Logic 1 Input Voltage		2.4 2.4	1.4 1.5		V V
V_{IL}	Logic 0 Input Voltage			1.1 1.0	0.8 0.8	V V
I_{IN}	Input Current	$0 \leq V_{IN} \leq V_S$	–1		1	μA
Output						
V_{OH}	High Output Voltage		$V_S - 0.025$			V
V_{OL}	Low Output Voltage				0.025	V
R_O	Output Resistance	$I_{OUT} = 10mA, V_S = 18V$		6 8	10 12	Ω Ω
I_{PK}	Peak Output Current			1.5		A
I	Latch-Up Protection	withstand reverse current	>500			mA
Switching Time						
t_R	Rise Time	test Figure 1		18 20	30 40	ns ns
t_F	Fall Time	test Figure 1		15 29	20 40	ns ns
t_{D1}	Delay Time	test Figure 1		17 19	30 40	ns ns
t_{D2}	Delay Time	test Figure 1		23 27	50 60	ns ns
t_{PW}	Pulse Width	test Figure 1	400			ns
Power Supply						
I_S	Power Supply Current	$V_{INA} = V_{INB} = 3.0V$		1.4 1.5	4.5 8	mA mA
I_S	Power Supply Current	$V_{INA} = V_{INB} = 0.0V$		0.18 0.19	0.4 0.6	mA mA

Notes:

1. Exceeding the absolute maximum rating may damage the device.
2. The device is not guaranteed to function outside its operating rating.
3. Devices are ESD sensitive. Handling precautions recommended.
4. Specification for packaged product only.

Test Circuits

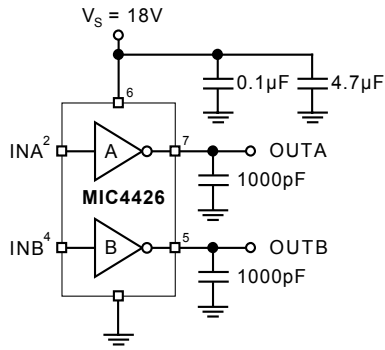


Figure 1a. Inverting Configuration

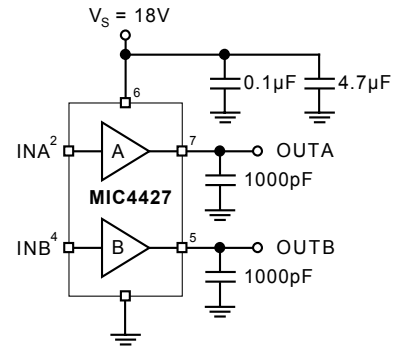


Figure 2a. Noninverting Configuration

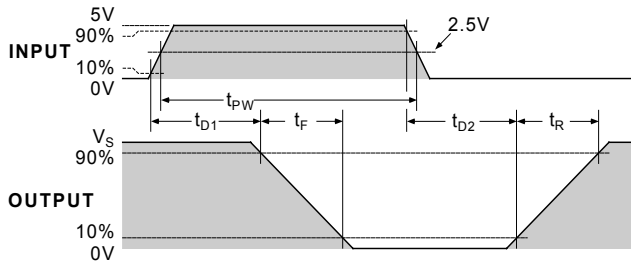


Figure 1b. Inverting Timing

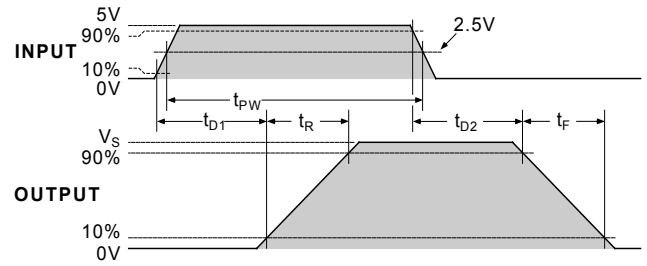
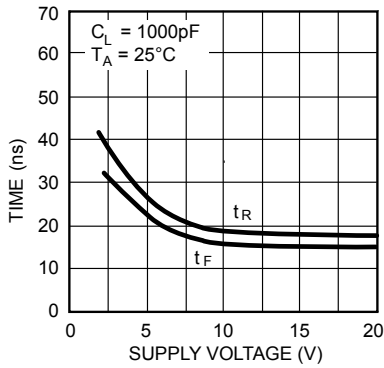


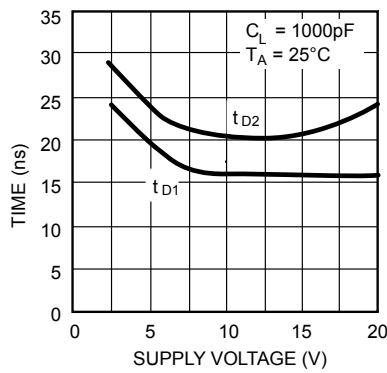
Figure 2b. Noninverting Timing

Electrical Characteristics

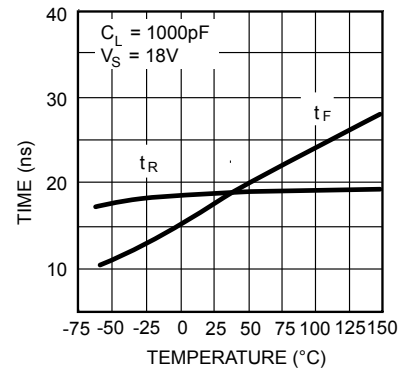
Rise and Fall Time vs. Supply Voltage



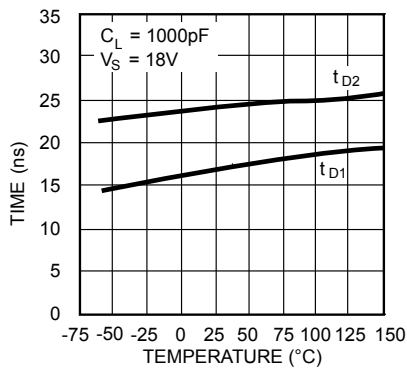
Delay Time vs. Supply Voltage



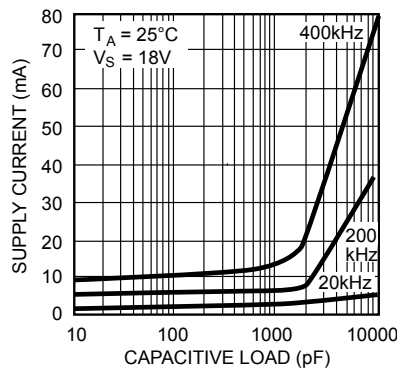
Rise and Fall Time vs. Temperature



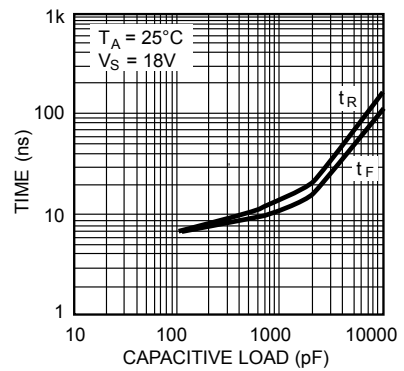
Delay Time vs. Temperature



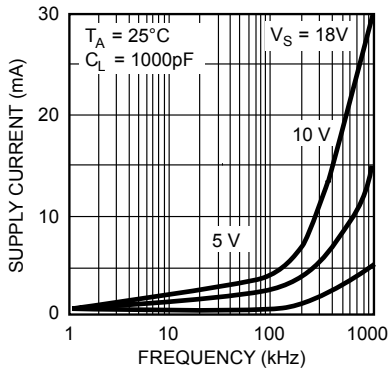
Supply Current vs. Capacitive Load



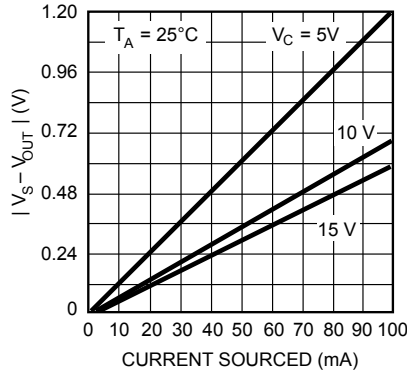
Rise and Fall Time vs. Capacitive Load



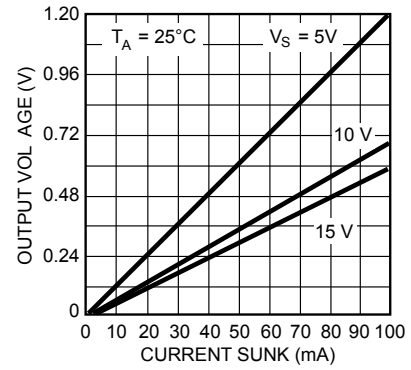
Supply Current vs. Frequency



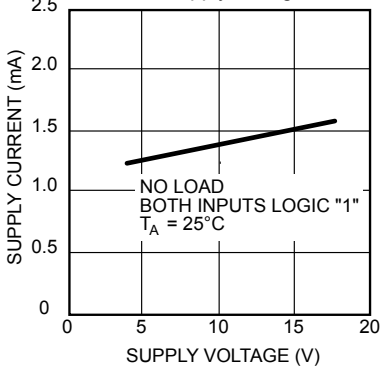
High Output vs. Current



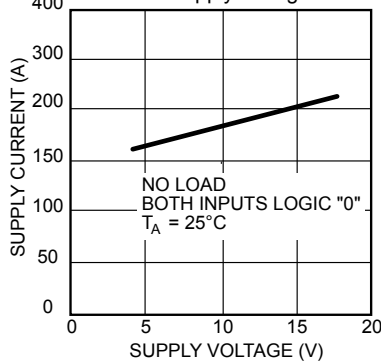
Low Output vs. Current



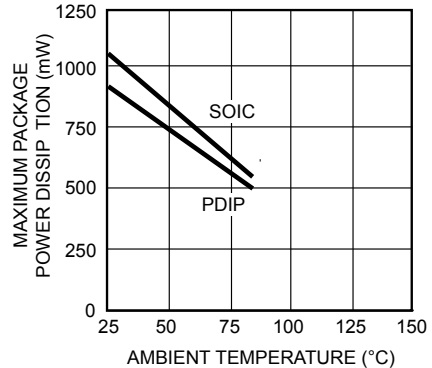
Quiescent Power Supply Current vs. Supply Voltage



Quiescent Power Supply Current vs. Supply Voltage



Package Power Dissipation



Applications Information

Supply Bypassing

Large currents are required to charge and discharge large capacitive loads quickly. For example, changing a 1000pF load by 16V in 25ns requires 0.8A from the supply input.

To guarantee low supply impedance over a wide frequency range, parallel capacitors are recommended for power supply bypassing. Low-inductance ceramic MLC capacitors with short lead lengths (< 0.5") should be used. A 1.0μF film capacitor in parallel with one or two 0.1μF ceramic MLC capacitors normally provides adequate bypassing.

Grounding

When using the inverting drivers in the MIC4426 or MIC4428, individual ground returns for the input and output circuits or a ground plane are recommended for optimum switching speed. The voltage drop that occurs between the driver's ground and the input signal ground, during normal high-current switching, will behave as negative feedback and degrade switching speed.

Control Input

Unused driver inputs must be connected to logic high (which can be V_S) or ground. For the lowest quiescent current (< 500μA), connect unused inputs to ground. A logic-high signal will cause the driver to draw up to 9mA.

The drivers are designed with 100mV of control input hysteresis. This provides clean transitions and minimizes output stage current spikes when changing states. The control input voltage threshold is approximately 1.5V. The control input recognizes 1.5V up to V_S as a logic high and draws less than 1μA within this range.

The MIC4426/7/8 drives the TL494, SG1526/7, MIC38C42, TSC170 and similar switch-mode power supply integrated circuits.

Power Dissipation

Power dissipation should be calculated to make sure that the driver is not operated beyond its thermal ratings. Quiescent power dissipation is negligible. A practical value for total power dissipation is the sum of the dissipation caused by the load and the transition power dissipation ($P_L + P_T$).

Load Dissipation

Power dissipation caused by continuous load current (when driving a resistive load) through the driver's output resistance is:

$$P_L = I_L^2 R_O$$

For capacitive loads, the dissipation in the driver is:

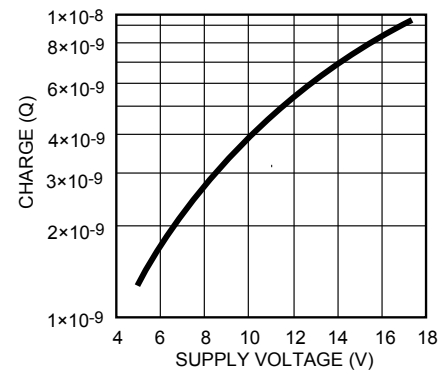
$$P_L = f C_L V_S^2$$

Transition Dissipation

In applications switching at a high frequency, transition power dissipation can be significant. This occurs during switching transitions when the P-channel and N-channel output FETs are both conducting for the brief moment when one is turning on and the other is turning off.

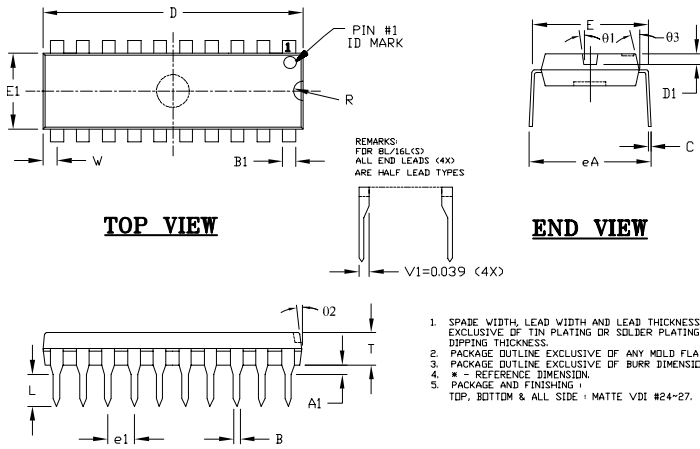
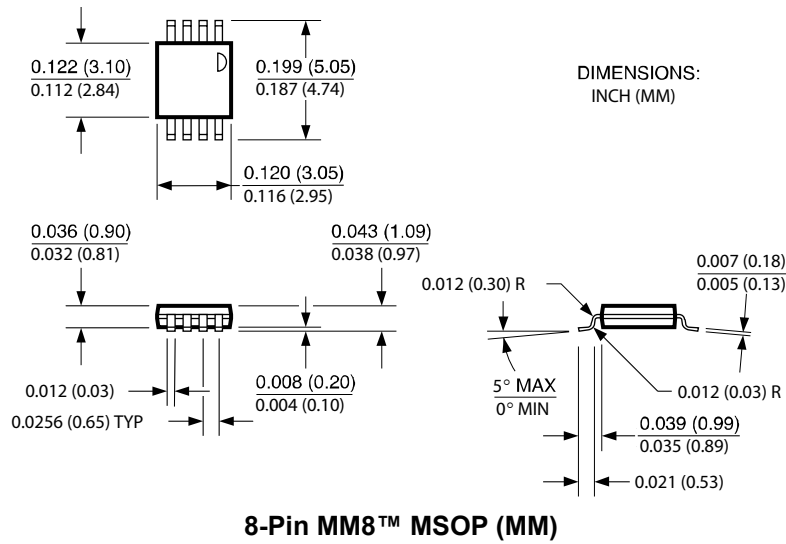
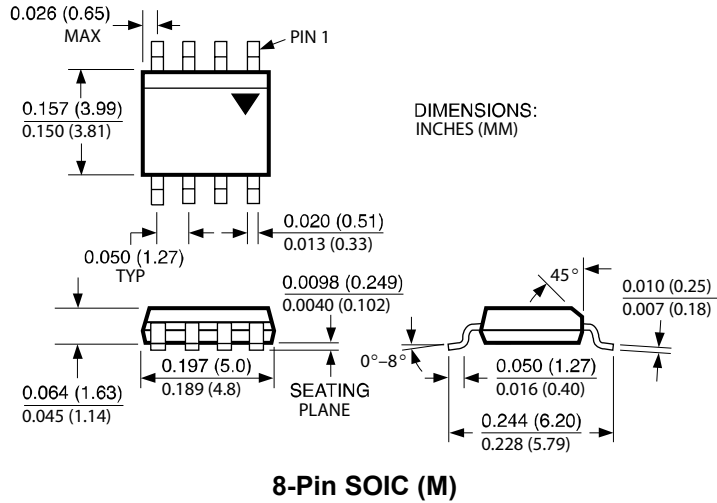
$$P_T = 2 f V_S Q$$

Charge (Q) is read from the following graph:



Crossover Energy Loss per Transition

Package Information



LEAD TYPE		8LD	14/16LD	18LD	20LD
STAND-OFF	A1	0.015 MIN	0.015 MIN	0.015 MIN	0.015 MIN
LEAD WIDTH *	B	0.018	0.018	0.018	0.018
SPADE WIDTH *	B1	0.060	0.060	0.060	0.060
LEAD THICKNESS *	C	0.010	0.010	0.010	0.010
LENGTH TOL ±0.004	D	0.375	0.750	0.890	1.020
IDENT DEPTH	D1	0.030 ~ 0.060	0.030 ~ 0.060	0.030 ~ 0.060	0.030 ~ 0.060
SHOULDER WIDTH OUTER TO OUTER	E	0.300 ~ 0.325	0.300 ~ 0.325	0.300 ~ 0.325	0.300 ~ 0.325
WIDTH TOL ±0.004	E1	0.250	0.250	0.250	0.250
LEAD SPREAD OUTER TO OUTER	eA	0.320 ~ 0.370	0.320 ~ 0.370	0.320 ~ 0.370	0.320 ~ 0.370
LEAD PITCH *	e1	0.100	0.100	0.100	0.100
LEAD LENGTH TOL ±0.004	L	0.125	0.125	0.125	0.125
IDENT RADIUS	R	0.030	0.030	0.030	0.030
TOTAL THICKNESS TOL ±0.004	T	0.130	0.130	0.130	0.130
LEAD TO END PACKAGE	W	0.025REF	0.075REF14LD 0.025REF16LD	0.045REF	0.060REF
IDENT DRAFT TOL ±3°	Ø1	7°	7°	7°	7°
END ANGLE (4x) TOL ±3°	Ø2	7°	7°	7°	7°
SIDE ANGLE (4x) TOL ±3°	Ø3	7°	7°	7°	7°

MICREL INC. 2180 FORTUNE DRIVE SAN JOSE, CA 95131 USA
TEL + 1 (408) 944-0800 FAX + 1 (408) 474-1000 WEB <http://www.micrel.com>

This information furnished by Micrel in this data sheet is believed to be accurate and reliable. However no responsibility is assumed by Micrel for its use.
Micrel reserves the right to change circuitry and specifications at any time without notification to the customer.

Micrel Products are not designed or authorized for use as components in life support appliances, devices or systems where malfunction of a product can reasonably be expected to result in personal injury. Life support devices or systems are devices or systems that (a) are intended for surgical implant into the body or (b) support or sustain life, and whose failure to perform can be reasonably expected to result in a significant injury to the user. A Purchaser's use or sale of Micrel Products for use in life support appliances, devices or systems is a Purchaser's own risk and Purchaser agrees to fully indemnify Micrel for any damages resulting from such use or sale.

© 2003 Micrel, Incorporated.

8051 board

