MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

(2415522)

SYSTEM DESIGN WITH EMBEDDED LINUX LAB

M.TECH -I YEAR - I SEMESTER (ECE)
M. R24 (MLRS) REGULATION

A.Y : 2024-2025

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1

SYSTEM DESIGN WITH EMBEDDED LINUX

Lab Manual for the Academic Year 2024-25

COURSECODE	**:**	**2415522**
REGULATIONS	**:**	**MLRS-R24**
CLASS	**:**	**I SEMESTER**
BRANCH	**:**	**M.TECH (Embedded Systems)**
INSTRUCTOR	**:**	**Dr. N. SRINIVAS**
PROGRAMMERS	**:**	**R. S. PILLAI**

LAB I/C **HOD - ECE**

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1

CERTIFICATE

This is to certify that this manual is a bonafide record of practical work in the ***System Design with embedded Linux lab*** in I **Semester of I -year M. Tech Sem I (ECE) Programme** during the academic year **2024-2025.** This book is prepared by **Dr. N Srinivas (Associate Professor), Mrs. R Babitha (Assistant Professor), Mrs. B Manjula (Assistant Professor),** Department of Electronics and Communication Engineering.

LAB I/C **Head of the Department**

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1

PREFACE

It is one of the core areas of ECE and constitutes the largest applications in use today. Communication has entered into every part of today's world. This laboratory is intended to make students understand the use of different System Design with Embedded Linux a n d is designed to help students understand the basic principles of design techniques as well as giving them the insight on design, simulation and hardware implementation of circuits. The main aim is to provide hands-on experience to the students so that they are able to put theoretical concepts to practice. The content of this course consists of two parts, 'simulation' and 'hardwired'. . Students will carry out design experiments as a part of the experiments list provided in this lab manual. Students will be given a specific design problem, which after completion they will verify using the simulation software or hardwired implementation.

By,

Dr. N Srinivas (Associate Professor),),
Mrs. R Babitha (Assistant Professor),
Mrs. B Manjula (Assistant Professor),

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1

ACKNOWLEDGEMENT

It was really a good experience, working with ***Systems Design with Embedded Linux Laboratory.*** First, we would like to thank Dr. N. Srinivas, Assoc. Professor, HOD of Department of Electronics and Communication Engineering, Marri Laxman Reddy Institute of technology & Management for his concern and giving the technical support in preparing the document.

We are deeply indebted and gratefully acknowledge the constant support and valuable patronage of Dr. Ravi Prasad, Dean, Marri Laxman Reddy Institute of technology & Management for giving us this wonderful opportunity for preparing the *Analog and Digital Communications Laboratory* manual.

We express our hearty thanks to Dr.R. Murali Prasad, Principal, Marri Laxman Reddy Institute of technology & Management, for timely corrections and scholarly guidance.

At last, but not the least I would like to thanks the entire ECE Department faculty those who had inspired and helped us to achieve our goal.

By,

Dr.N.Srinivas(Associate Professor),

Mrs.R.Babitha(Assistant Professor),

Mrs.B.Manjula(Assistant Professor)

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1

GENERAL INSTRUCTIONS

1. Students should report to the concerned labs as per the timetable schedule.

2. Students who turn up late to the labs will in no case be permitted to perform the experiment scheduled for the day.

3. After completion of the experiment, certification of the concerned staff in-charge in the observation book is necessary.

4. Students should bring a notebook of about 100 pages and should enter the readings/observations into the notebook while performing the experiment.

5. The record of observations along with the detailed experimental procedure of the experiment.

6. Performed in the immediate last session should be submitted and certified by the staff member in-charge.

7.. Not more than one student is permitted to perform the experiment on a setup.

8. When the experiment is completed, students should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.

9. Any damage of the equipment or burnout of components will be viewed seriously by putting penalty.

10. Students should be present in the labs for the total scheduled duration.

11. Students are required to prepare thoroughly to perform the experiment before coming to Laboratory.

12. Procedure sheets/data sheets provided to the student's should be maintained neatly and to be returned after the experiment.

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1

SAFETY PRECAUTIONS

1. No horseplay or running is allowed in the labs.
2. No bare feet or open sandals are permitted.
3. Before energizing any equipment, check whether anyone is in a position to be injured by your actions.
4. Read the appropriate equipment instruction manual sections or consult with your instructor.
5. Before applying power or connecting unfamiliar equipment or instruments into any circuits.
6. Position all equipment on benches in a safe and stable manner.
7. Do not make circuit connections by hand while circuits are energized.
8. Dangerous with high voltage and current circuits.

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1

Vision of the Institute

To be a globally recognized institution that fosters innovation, excellence, and leadership in education, research, and technology development, empowering students to create sustainable solutions for the advancement of society.

Mission of the Institute

To foster a transformative learning environment that empowers students to excel in engineering, innovation, and leadership.

To produce skilled, ethical, and socially responsible engineers who contribute to sustainable technological advancements and address global challenges.
To shape future leaders through cutting-edge research, industry collaboration, and community engagement.

Quality Policy

The management is committed in assuring quality service to all its stakeholders, students, parents, alumni, employees, employers, and the community.

Our commitment and dedication are built into our policy of continual quality improvement by establishing and implementing mechanisms and modalities ensuring accountability at all levels, transparency in procedures, and access to information and actions.

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Department of Electronics and Communication Engineering

Vision of the Department

To provide quality technical education in Electronics and Communication Engineering through research, innovation, striving for global recognition in specified domain, leadership, and sustainable societal solutions.

Mission of the Department

- To create a transformative learning environment that empowers students in electronics and communication engineering, fostering excellence in technical skills and leadership.
- To drive innovation through research, deliver a transformative education grounded in ethical principles, and nurture the development of professionals
- To cultivate strong industry partnerships, and engaging actively with the community for societal and technological progress.

Program educational Objectives (PEOs)

PEO 1: Have Successful career in Industry

Graduates will excel in the Electronics and Communication industry with a strong foundation in technical expertise, continuous learning, and innovation.

PEO 2: Show Excellence in higher studies/Research

Graduates will excel in higher studies and research in Electronics and Communication Engineering (ECE) through a combination of rigorous academic dedication, cutting-edge innovation, and a deep understanding of emerging technologies.

PEO 3: Show Good Competency towards Entrepreneurship

Graduates will have *t*o show good competency towards entrepreneurship in the field of Electronics and Communication Engineering, one must demonstrate an in-depth understanding of emerging technologies, market trends, and the ability to innovate within this rapidly evolving industry.

Program Specific Outcomes (PSOs)

1. Analyze and design analog & digital circuits or systems for a given specification and function.
2. Implement functional blocks of hardware-software co-designs for signal processing and communication applications.

MARRI LAXMAN REDDY
INSTITUTE OF TECHNOLOGY AND MANAGEMENT
(AN AUTONOMOUS INSTITUTION)
(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with 'A' Grade & Recognized Under Section2(f) & 12(B)of the UGC act,1956

Department of Electronics and Communication Engineering

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Lab Objective:

An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular kind of application device. Industrial machines, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines, and toys (as well as the more obvious cellular phone and PDA) are among the myriad possible hosts of an embedded system. Embedded systems that are programmable are provided with a programming interface, and embedded systems programming is a specialized occupation.

An **embedded system** is a special-purpose computer system designed to perform one or a few dedicated functions[1], often with real-time computing constraints. It is usually *embedded* as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems have become very important today as they control many of the common devices we use.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

List of Experiments

S. No	Name of the Experiment
1	Write a Program to a) Read inputs from switches. b) To make LEDs blink.
2	Write a program to interface a switch and a buzzer to two different pins of a Port such that the buzzer should sound as long as the switch is pressed.
3	Write a Program for serial communication.
4	Write a Program for encryption / decryption
5	Develop necessary interfacing circuit to read data from a sensor and process using the 801 and 8051 boards. The data to be displayed on a PC monitor.
6	Write a program to transmit a message from Microcontroller to PC serially using RS232
7	Sort RTOs (mCOS) on to 89CS1 board and Verify.
8	Simulate on elevator movement using RTO's on 89CSI board.

INTRODUCTION ABOUT LAB

There are 12 systems (Compaq Presario) installed in this Lab. Their configurations are as follows :

Processor	**:**	**AMD Athelon ™ 1.67 GHz**
RAM	**:**	**1 GB**
Hard Disk	**:**	**40 GB**
Mouse	**:**	**Optical Mouse**
Network Interface card	**:**	**Present**

Software

1 All systems are configured in DUAL BOOT mode i.e, Students can boot from Windows XP or Linux as per their lab requirement.

This is very useful for students because they are familiar with different Operating Systems so that they can execute their programs in different programming environments.

2 Each student has a separate login for database access

Oracle 9i client version is installed in all systems. On the server, account for each student has been created.

This is very useful because students can save their work (scenarios', pl/sql programs, data related projects ,etc) in their own accounts. Each student work is safe and secure from other students.

3 Latest Technologies like DOT NET and J2EE are installed in some systems. Before submitting their final project, they can start doing mini project from 2nd year onwards.

4 MASM (Macro Assembler), Keil is installed in all the systems

Students can execute their assembly language programs using MASM. MASM is very useful students because when they execute their programs they can see contents of Processor Registers and how each instruction is being executed in the CPU.

1 Rational Rose Software is installed in some systems

Using this software, students can depict UML diagrams of their projects.

2 Software installed : C, C++, JDK1.5, MASM, OFFICE-XP, J2EE and DOT NET, Rational Rose.

3 Systems are provided for students in the 1:1 ratio.

4 Systems are assigned numbers and same system is allotted for students when they do the lab.

LAB CODE

1. Students should report to the concerned labs as per time table schedule.
2. Students who turn up late to the labs will in no case be permited to do the program scheduled for the day.
3. After completion of the program , certification of the concerned staff in-charge in the observation book is necessary.
4. Students should bring a notebook of about 100 pages and should enter the reading/observations into the notebook while performing the experiment.
5. The record of observations along with the detailed experimental procedure of the experiment performed in the immediate last session should be submitted and certified by the staff ember in-charge.
6. The group-wise division made in the beginning should be adhered to and no mix up student among different groups will be permitted later.
7. The components required pertaining to the experiment should be collected from stores in-charge after duly filling in the requisition form.
8. When the experiment is completed, students should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.
9. Any damage of the equipment or burn-out of components will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year.
10. Students should be present in the labs for the total scheduled duration.
11. Students are required to prepare thoroughly to perform the experiment before coming to Laboratory.
12. Procedure sheets/data sheets provided to the student's groups should be maintained neatly and to be returned after the experiment.

Description about ES Concepts:

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reason such as safety and usability; others may have low or no performance requirements,allowing the system hardware to be simplified to reduce costs.

Embedded systems are not always separate devices. Most often they are physically built-in to the devices they control.

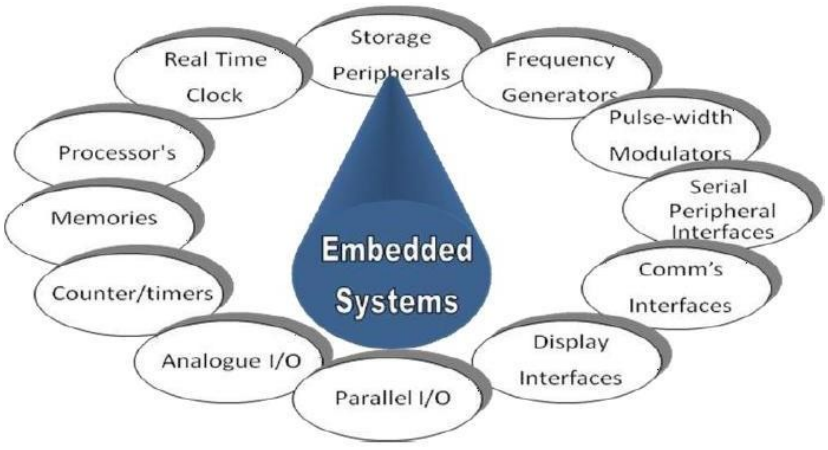
The software written for embedded systems is often called firmware, and is stored in read-only memory or Flash memory chips rather than a disk drive. It often runs with limited computer hardware resources: small or no keyboard, screen, and little memory.

Embedded systems range from no user interface at all — dedicated only to one task — to full user interfaces similar to desktop operating systems in devices such as

PDAs.

Simple embedded devices use buttons, LEDs, and small character- or digit-only displays, often with a simple menu system.

Embedded Systems components:



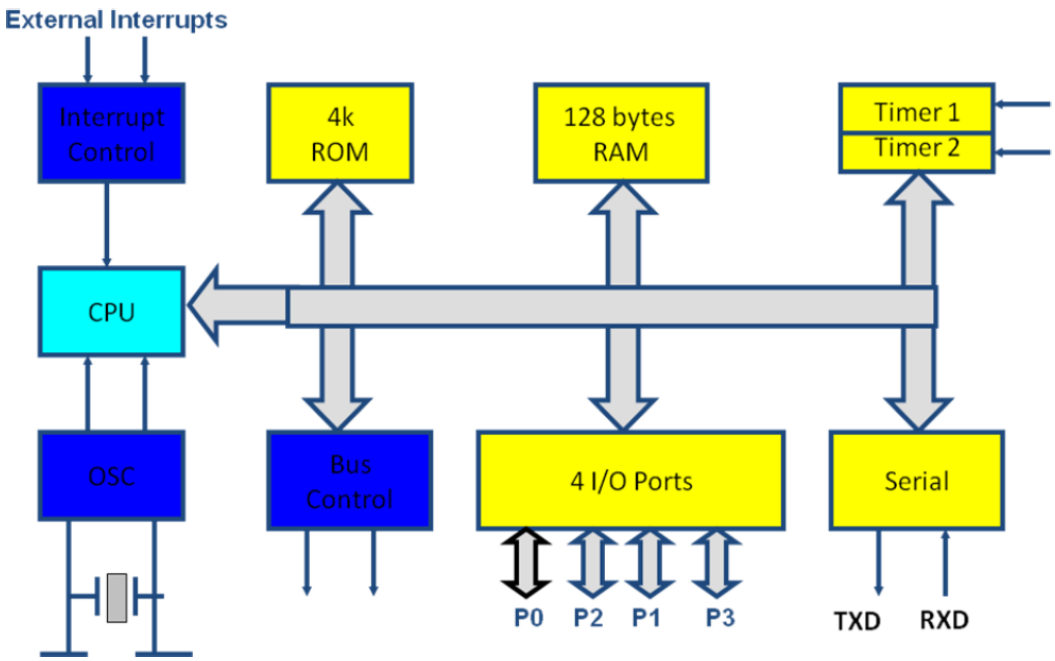
Introduction to 8051 Microcontroller:

- Microcontroller is a device which integrates number of components of a microprocessor system onto a single chip. It typically includes:-

- CPU (Central Processing unit)
- RAM & ROM
- I/O inputs & outputs – Serial & Parallel
- Timers
- Interrupt Controller

By including the features that are specific to the task (Control), Cost is relatively low. Microcontroller are a ―one chip solutions‖ which drastically reduces parts count and design costs.

Block Diagram:



8051 Basic Components:

- 4K bytes internal ROM
- 128 bytes internal RAM
- Four 8-bit I/O ports (P0 - P3).
- Two 16-bit timers/counters
- One serial interface

8051 features:

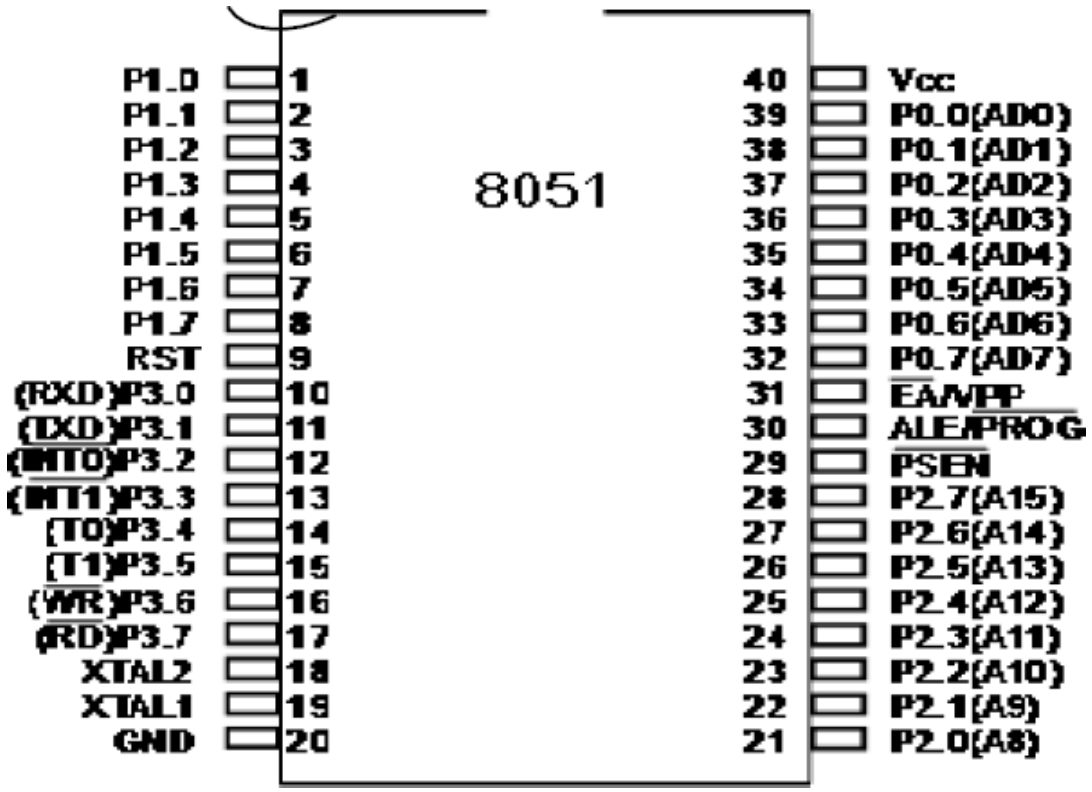
- 4K bytes ROM
- 128 bytes RAM
- Four 8-bit I/O ports
- Two 16-bit timers
- Serial interface
- 64K external code memory space
- ALU
- Working Registers
- Clock Circuits

- Timers and Counters
- Serial Data Communication.

8051 CPU Registers:

- A (8-bit Accumulator)
- B (8-bit register for Mul &Div)
- PSW (8-bit Program Status Word)
- SP (8-bit Stack Pointer)
- PC (16-bit Program Counter)
- DPTR (16-bit Data Pointer)

Pin Description of the 8051:



- The 8051 is a 40 pin device, but out of these 40 pins, 32 are used for I/O
- 24 of these are dual purpose, i.e. they can operate as I/O or a control line or as part of address or date bus.

8051 Development Board (P89V51RD2)

On board Peripherals:

1) hex-key pad
2) seven segment display
3) serial peripheral interface (spi)
4) led‘ display
5) analog to digital converter
6) lm35 temperature sensor
7) digital to analog converter
8) rtc battery
9) eeprom (i2c)
10) rtc
11) lcd display
12) gnd and vcc
13) lcd contrast (potentiometer)
14) p89v51rd2
15) crystal oscillator
16) max232
17) serial port connector
18) stepper motor driver
19) buzzer
20) reset button
21) push button switches
22) slide switches
23) ps/2 connector
24) relay output connector
25) power supply slide switch
26) power jack
27) 7805 voltage regulator
28) bridge rectifier
29) relay

Overview:

The UTS-MC-KIT-M7.3 has got P89V51RD2 microcontroller which has got 64KiloBytes of on chip Flash memory and 1 KiloBytes of RAM. The kit is has got on board 11.0592MHz crystal for generating the on chip clock of 11.0592MHz.

A Key feature of the board is it has got so many interfaces, with different on board peripherals and has got expansion capability to add any further sensor and peripherals in future. This prototype board is very easy to use for 8051 architecture. This board is interfaced with LED‘s, 7 SEG display, LCD display, Pushbutton. This Board can also be interfaced with PC via serial communication and can be viewed through hyper terminal. The LCD display can be connected easily through connectors. No soldering work /No lose contact/ just plug in the berg connectors.

The board has got on chip peripherals like on board 32 KB bytes of RAM, Eight Light Emitting Diodes, four Push Buttons, Four Seven Segment Displays, 16X2 Liquid Crystal Character Display(LCD), Analog to Digital Converter, LM35 Temperature sensor, SPI based ADC, Hex Keypad, Buzzer relay, steeper motor driver interface, Real time clock, RS-232 serial interface.

Component Description:

Microcontroller

The P89V51RD2 device contains a non-volatile 64KB Flash program memory.

In-System Programming (ISP) allows the user to download new code while the microcontroller sits in the application. A default serial loader (boot loader) program in ROM allows serial In-System programming of the Flash memory via the UART without

the need for a loader in the Flash code.

This device executes one machine cycle in 6 clock cycles, hence providing twice the speed of a conventional 80C51. An OTP configuration bit lets the user select conventional 12 clock timing if desired.

This device is a Single-Chip 8-Bit Micro controller manufactured in advanced CMOS process and is a derivative of the 80C51 micro controller family. The instruction set is 100% compatible with the 80C51 instruction set.

The device also has four 8-bit I/O ports, three 16-bit timer/event counters, a multisource, and four- priority-level, nested interrupt structure, an enhanced UART and on-chip oscillator and timing circuits.

The added features of the P89V51RD2 makes it a powerful micro controller for applications that require pulse width modulation, high-speed I/O and up/down counting capabilities such as motor control.

Experimental Procedure for Keil4 IDE

The RVision IDE is, for most developers, the easiest way to create embedded system programs.

This chapter describes commonly used RVision features and explains how to use them.

RVision is a Windows application that encapsulates the Keil microcontroller development tools as well as several third-party utilities. RVision provides everything you need to start creating embedded programs quickly. RVision includes an advanced editor, project manager, and make utility, which work together to ease your development efforts, decreases the learning curve, and helps you to get started with creating embedded applications quickly.

There are several tasks involved in creating a new embedded project:

- Creating a Project File
- Using the Project Windows
- Creating Source Files
- Adding Source Files to the Project
- Using Targets, Groups, and Files
- Setting Target Options, Groups Options, and File Options
- Configuring the Startup Code
- Building the Project
- Creating a HEX File

The below section provides a step-by-step tutorial that shows you how to create an embedded project using the RVision IDE.

Downloading the hex file to the target using Flash magic Software:

Open the Flash Magic tool for downloading into the Microcontroller Board. Click on Device menu select option you will be popped up with a window named choose device. Under choose device options select 8051 and click on Ok button to open flash magic tool to download the hex file in to the MC

Terminal Software for Check the Serial port Data receiving from Microcontroller to PC:

Terminal is a simple serial port (COM) terminal emulation program. It can be used for communication with different devices such as modems, routers, embedded microcontroller systems, GSM phones, GPS modules... It is very useful debugging tool for serial communication applications.

1. Write a Program to

a) Read inputs from switches.
b) To make LEDs blink.

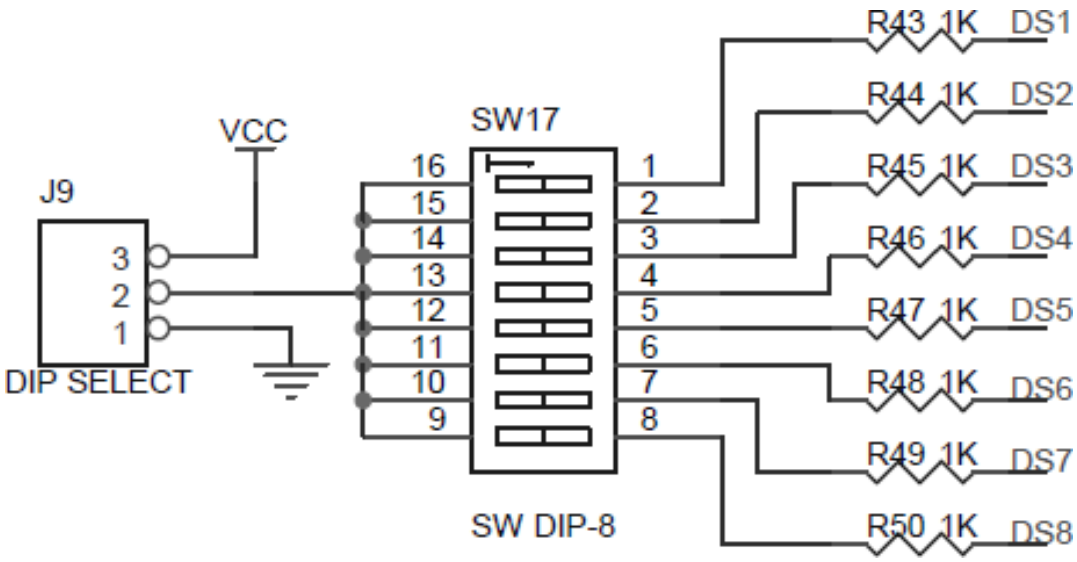
Aim:

To Read inputs from Switches

Program

8-Way DIP Switch

This is another simple interface, 8to the port lines, and for some control application
i) The Development board has one no. of 8used to provide digital inputs to the microcontroller's ports.
ii) User can change the level of digital inputs whatever they want, either high or low by simply selecting the jumper+5V, in order to should be used.

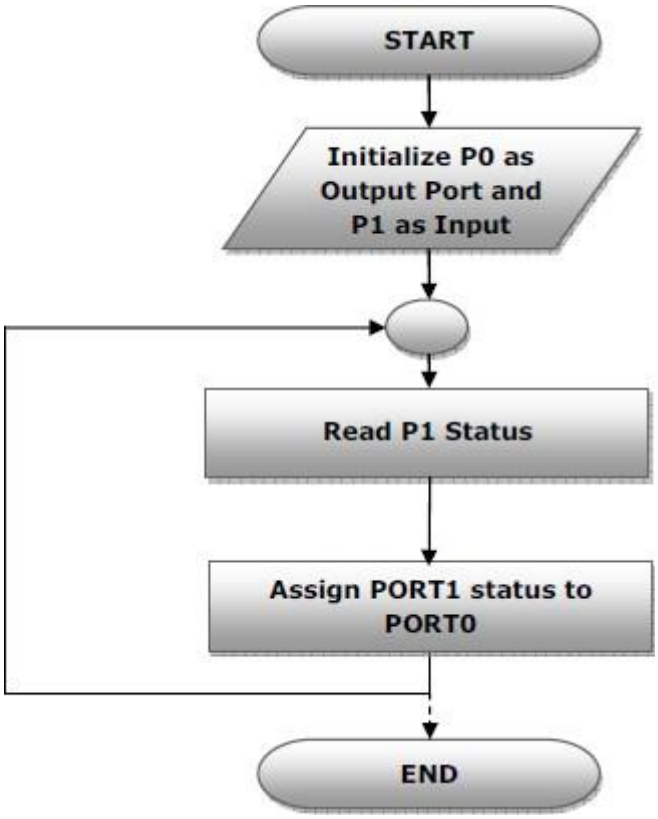


/*----------

Example 1 : Read Switch status & displayed in LED's
Description : Depends upon the dip switch position
the corresponding leds are on (or) off

---------- */

FLOW Chart



CODE:

```
#include <reg51.h> //Define 8051 Registers
#define LED P0 //Assign led to P0
#define SWITCH P1 //Assign switch to P1

/********************************** Main
Function*******************************/
void main(void)
{
SWITCH = 0xff; //Initialize the switch as input port
LED = 0x00; //Initialize the led as output port
while(1)
{
LED = ~SWITCH;
}
}
```

Execution:

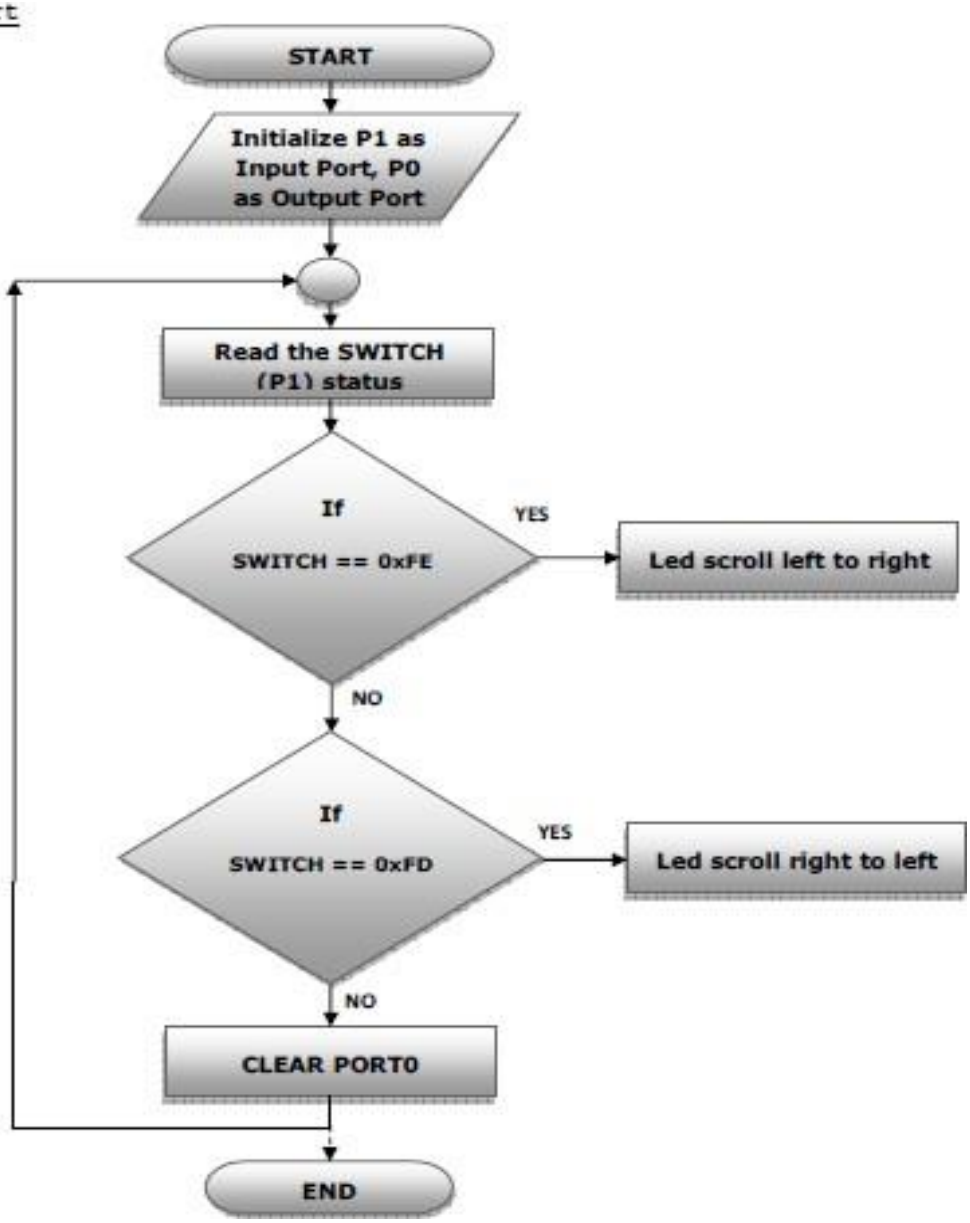
Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

/*----------

Example 2 : Read Switch status & scrolling the LED's
Description : Depends upon the dip switch position
the led will be scrolling

---------- */

Flow Chart



Code:

```
#include<reg51.h> //Define 8051 Registers
#define Led P0 //Assign led as P0
#define Switch P1 //Assign switch as P1
void DelayMs(unsigned int k); //Delay function
void scrollLeft();
void scrollRight();
int loop,port;

/********************** Main Function*****************************/
void main(void)
{
Switch =0xff; //Intialize the port1
Led =0x00; //Intialize the port0
while(1)
{
if(Switch ==0xfe)
{ scrollLeft(); //scrollLeft function
}
else if(Switch ==0xfd)
{ scrollRight(); // scrollRight function
}
else
{ Led =0x00;
}
}
}

/************************** Delay for 1 msec************************/
void DelayMs(unsigned int k)
{
unsigned int i,ms;
for(ms=0;ms<=k;ms++)
{ for(i=0;i<=114;i++); }
}

/************************** LED Scroll Left ************************/
void scrollLeft()
{
port=1;
for(loop=0;loop<8;loop++)
{
Led =port;
port=port*2;
DelayMs(500); //Delay for 500ms
}
}
```

```
/************************** LED Scroll Left ************************/
void scrollRight()
{
port=128;
for(loop=0;loop<8;loop++)
{
Led =port;
port=port/2;
DelayMs(500); //Delay for 500ms
}
}
```

Execution:

Note: After loading corresponding Examples Hex file located in "OUT" Folder to The microcontroller kit, press "RST" Button, user program now executes.

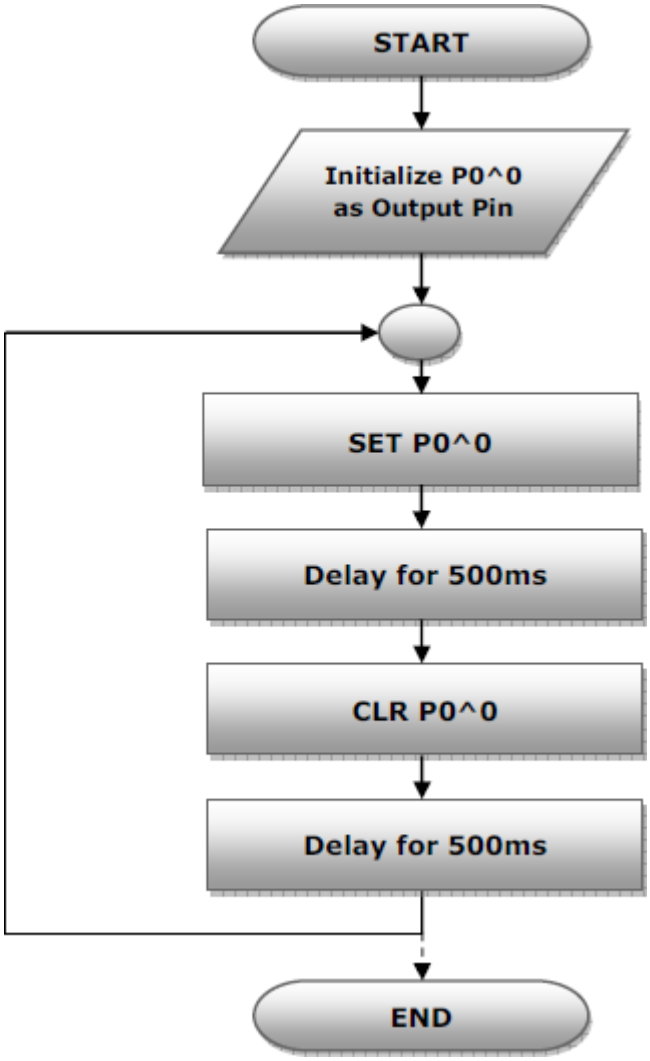
b) To make LEDs blink

----------*/

Example 1 : Program to Blink LED at P0.0
Description : Connect a LED at a port pin and make it
flash at predefined intervals.

----------*/

Flow Chart



Code:

```
#include<reg51.h> //Define 8051 Registers
void DelayMs(unsigned int a); //Delay function
sbit led =P0^0; //Set bit P0^0 for led

/*-------------------------------------------------------------------------------------------
--*/
// Main Program

void main()
{
P0=0x00; //Port0 as output port
while(1) //Loop forever
{
led = 1; //Set the led to high
DelayMs(500); //Delay for 500ms
led = 0; //Set the led to low
DelayMs(500); //Delay for 500ms
}
}

/*****************************
// Delay for 1 msec
*****************************/

void DelayMs(unsigned int k)
{
unsigned int i,ms;
for(ms=0;ms<=k;ms++)
{
for(i=0;i<=114;i++);
}
}
```

Execution:

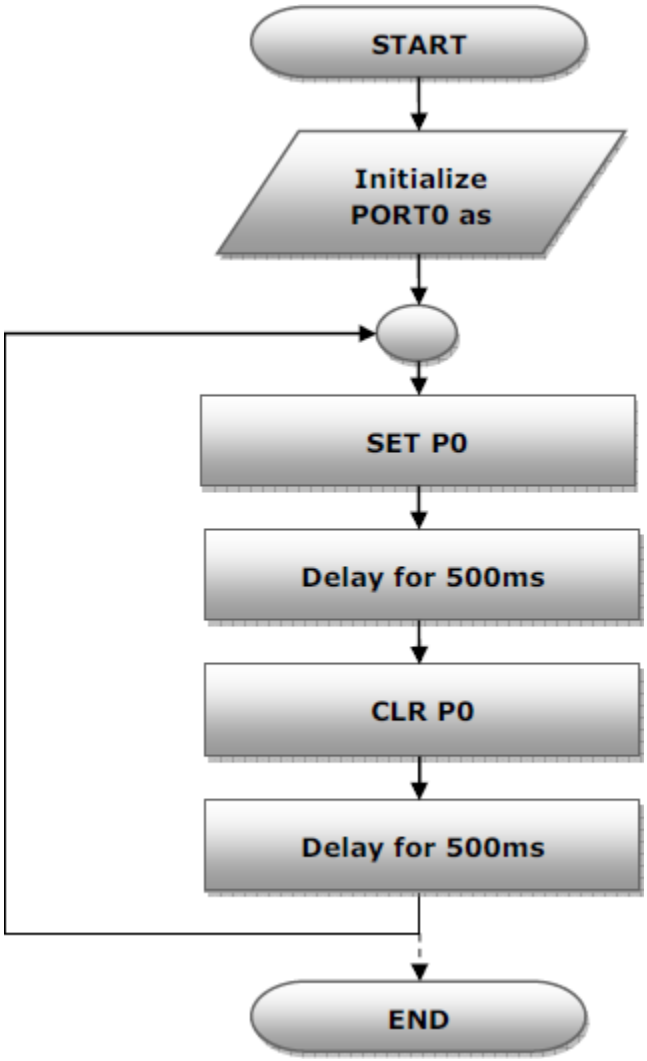
Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

/*----------

Example 2 : Program to Blink LED at Port P0
Description : Connect a LED at a port pins and make it flash at predefined intervals.

---------- */

Flow Chart



Code:

```
#include<reg51.h> //Define 8051 Registers
#define led P0 //Define P0 for Led
void DelayMs(unsigned int a); //Delay function

/*------------------------------------------------------------------
// Main Program
------------------------------------------------------------------ */

void main()
{
P0=0x00; //Port0 as output port
while(1) //Loop forever
{
led = 0xff; //Set the all led to high
DelayMs(500); //Delay for 500ms
led = 0x00; //Set the all led to low
DelayMs(500); //Delay for 500ms
}
}

/*****************************
// Delay for 1 msec
*****************************/

void DelayMs(unsigned int k)
{
unsigned int i,ms;
for(ms=0;ms<=k;ms++)
{
for(i=0;i<=114;i++);
}
}
```

Execution:

Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

2. Write a Embedded C Program to interface a switch and a buzzer to two different pins of a port such that the buzzer should sound as long as the switch is pressed.

Equipment Requirements:

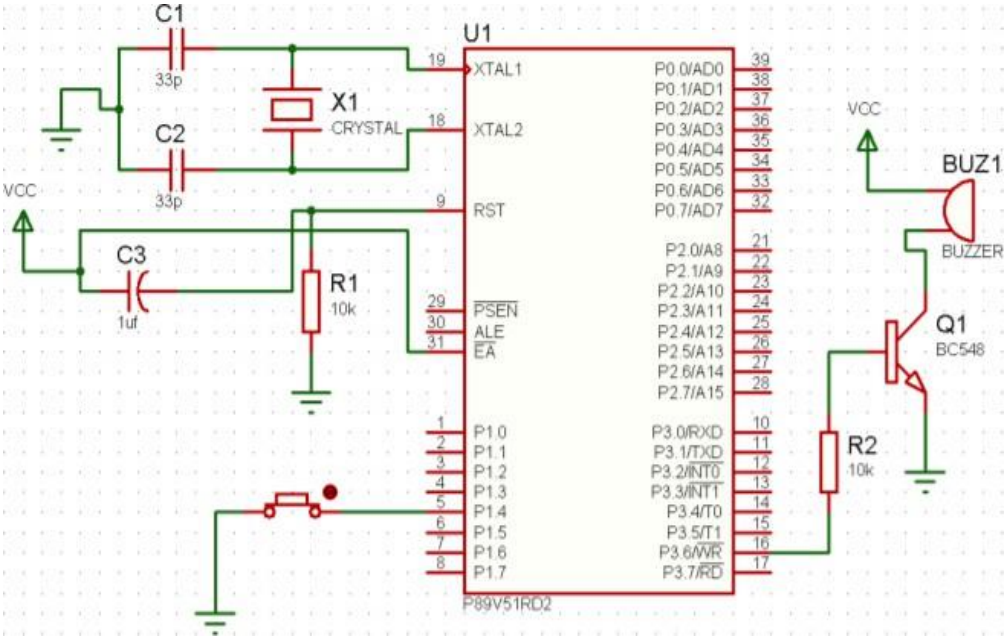
Hardware Requirements:

1. 89V51RD2 Development board
2. A serial 9 pin cable wired one to one from female connector to male connector
3. PC with serial port
4. 9V adaptor
5. Connecting jumper and Connecting Wires.

Software Rquirements :

1. Keil evaluation software
2. Flash Magic tool.

Interfacing Switch & Buzzer with 8051:



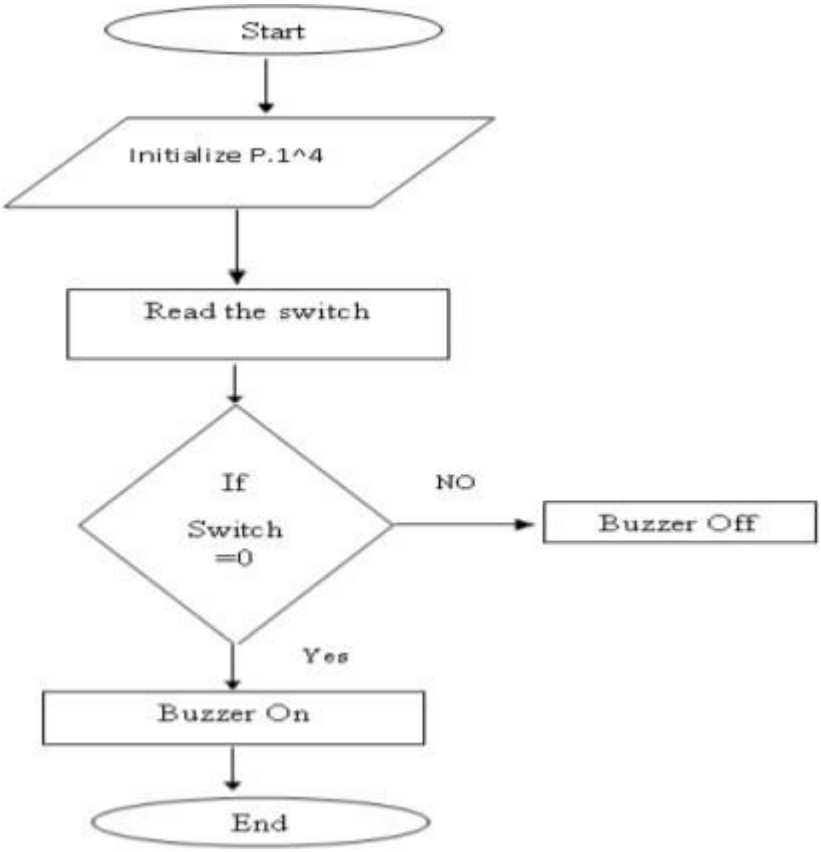
Source code:

/*Program to interface a switch and a buzzer to two different pins of a Port such that the buzzer should sound as long as the switch is pressed.
*/ #include // special function register declarations for the intended 8051 derivative

```
sbit SW1 = P1^4;
sbit BUZZER = P3^6;
void main (void)
 {
BUZZER = 0;
while(1)
{
if(SW1 == 0)
{
BUZZER = 1;
}
else
BUZZER = 0;
```

```
        }
}
```

Flow Chart:

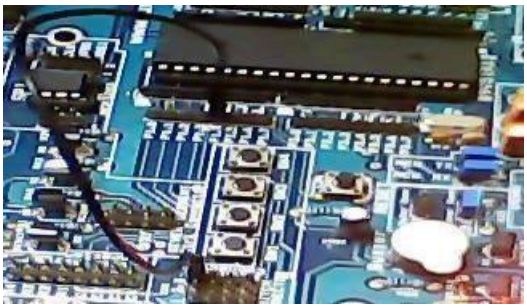


Hardware configuration:

1. Connect a single pin wire from PORT 1.4 to any switch available on board.
2. Place the jumper at jp6 jumper position to connect the buzzer onboard to the controller.
3. Turn ON and OFF or reset the board, to view the output.

Results/Output verification:

After programming the code into the microcontroller just reset the microcontroller.You can listen to the buzzer buzzing.



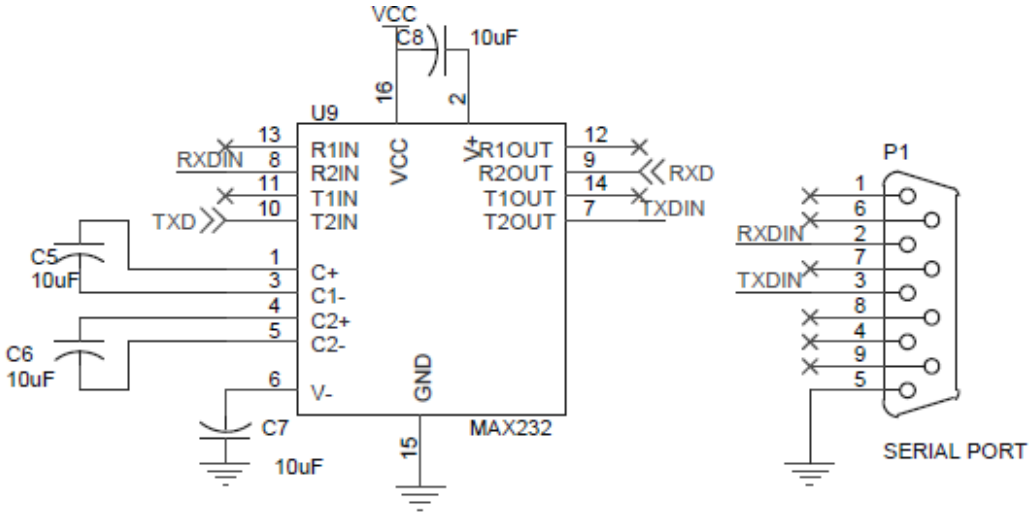
3. Write a Program for serial communication.

Aim : To write a Program for serial communication

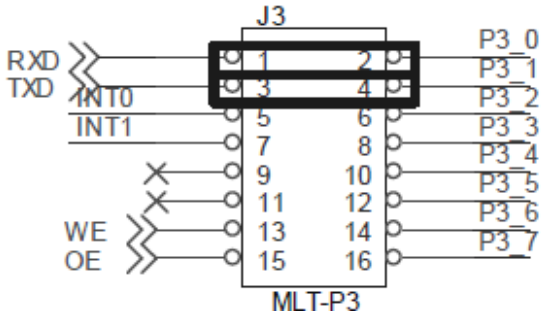
Program

1) RS-232 communication enables point in data acquisition applications, for the transfer of data between the microcontroller and a PC.

2) The voltage levels of a microcontroller and PC are not directly compatible with those of RS-232, a level transition buffer such as MAX232 be used.

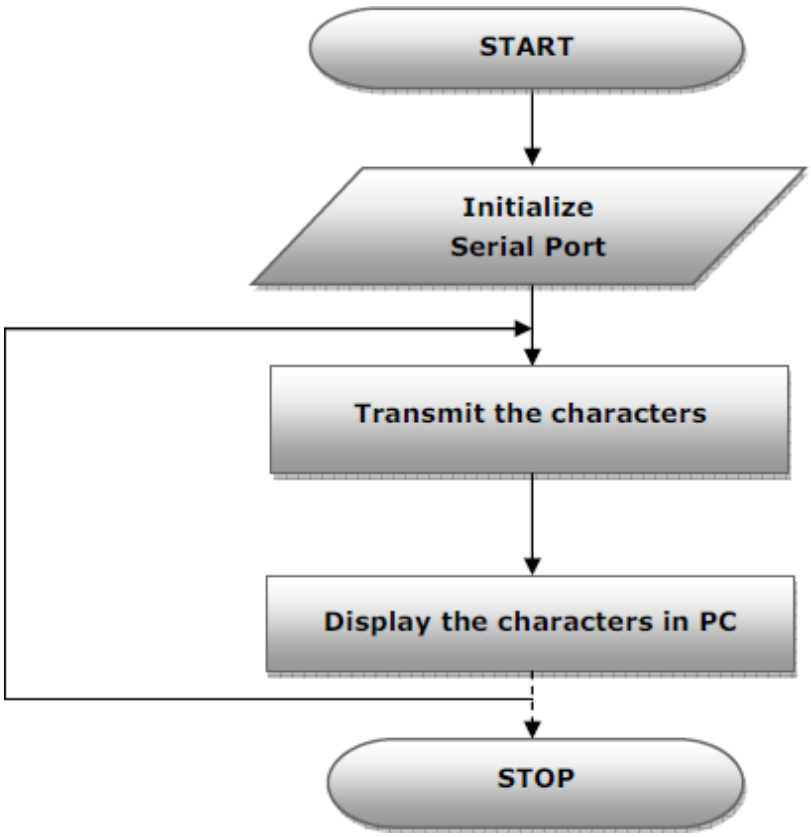


☛: Note : short pins 1 & 2 (RXD) and 3 & 4 (TXD) of J3.



/* -- */
/* Example 1 : Program to send data serially through serial port */
/* */
/* */
/* Description: Output can be viewed through system's hyper terminal */
/* window by setting the baud rate to 9600 */
/* -- */

Flow Chart



Code:

```
#include <REG51.H> /*SFR register declarations */
#include <stdio.h>
void serial_init(void);

//-----------------------------------------------
//Setup the serial port for 9600 baud at 11.0592MHz.
//-----------------------------------------------
void serial_init(void)
{
SCON = 0x50; /* SCON: mode 1, 8-bit UART, enable rcvr */
TMOD |= 0x20; /* TMOD: timer 1, mode 2, 8-bit reload */
TH1 = 0xFD; /* TH1: reload value for 9600 baud,11.0592MHz*/
TR1 = 1; /* TR1: timer 1 run */
TI = 1; /* TI: set TI to send first char of UART */
}

//-------------------------
//Main Program Starts Here
//-------------------------
void main(void)
{
serial_init();
while (1){
printf ("Hello! World\n"); /* Print "Hello World" */
}
}
```

Execution:

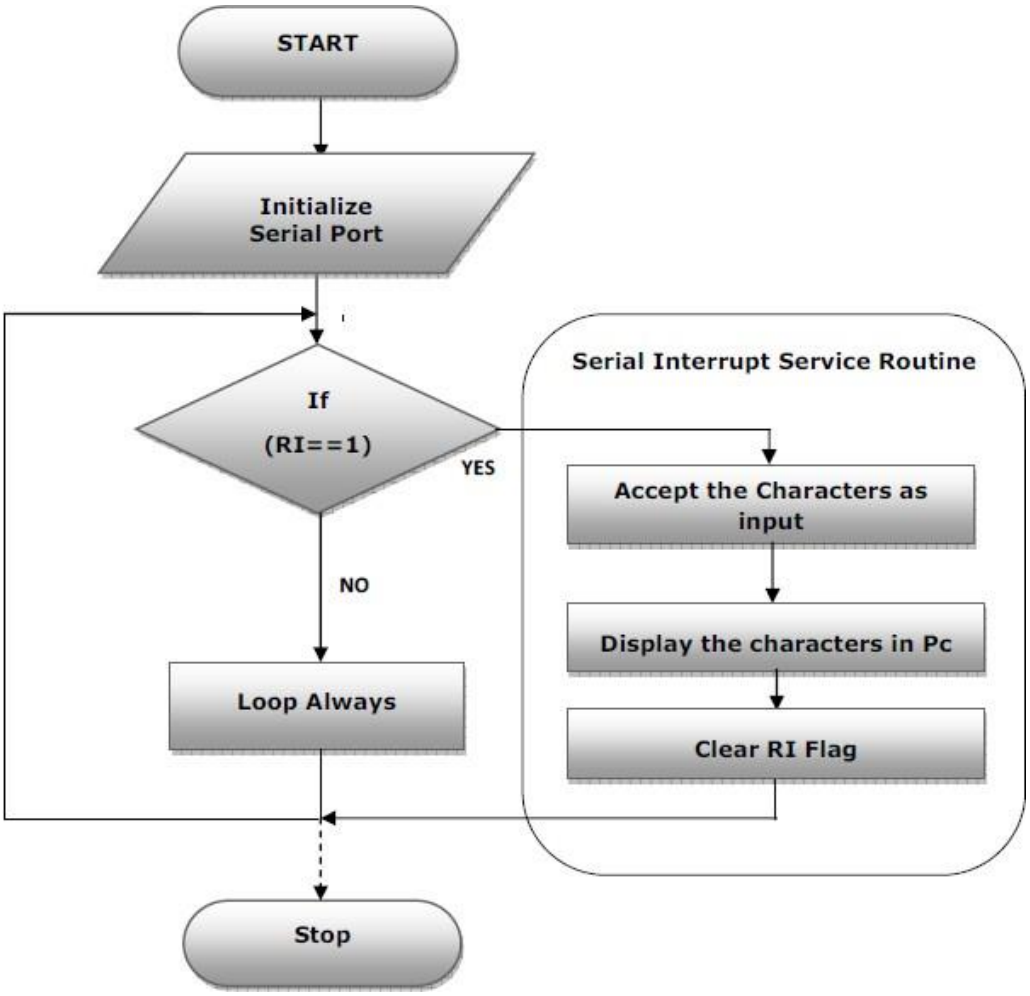
Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

Output:

/*----------

Example 2 : Program for serial receive data echo using interrupt
Description: The program is designed so as to echo the characters typed through the HyperTerminal window
---------- */

Flow Chart



Code:

```
#include <reg51.h> //include 8051 header file
#include <stdio.h>
void delay(void);
void serial_init(void);
int y;
/************************** Receive Interrupt Function
******************/
void serial_int() interrupt 4 // serial interrupt
{
unsigned char y;
if(RI==1){
y=SBUF; // read SBUF
SBUF=y; //send new value to SBUF
delay();
} RI=0; // clear RI flag
}
/*****************************Initialize serial Function****************/
void serial_init(void)
{
TMOD=0x20; //Timer1 8bit autoreload mode
SCON=0x50; // 8bit UART mode
ES=1; //Enable Serial Port Interrupt
EA=1; //Enable All Interrupt
TH1=0xFD; //set Baud Rate 9600 for 11.09MHz
//TH1=(((F-sc/12)/32)/Baudratval)
TR1=1; // Start timer 1
TI=1;
}
/*****************************Delay
Function*********************************/
void delay(void)
{
int i;
for(i=0;i<=10000;i++);
}
/************************** Main Function
*********************************/
void main()
{
serial_init();
while(1); // loop infinitely waiting only for serial interrupts
}
```

Execution:

Note: After Loading corresponding Examples Hex file located in “OUT” Folder to the microcontroller kit, press “RST” Button, user program now executes.

4. Write a Program for encryption / decryption.

Aim

Writing a program for encryption/decryption

Program

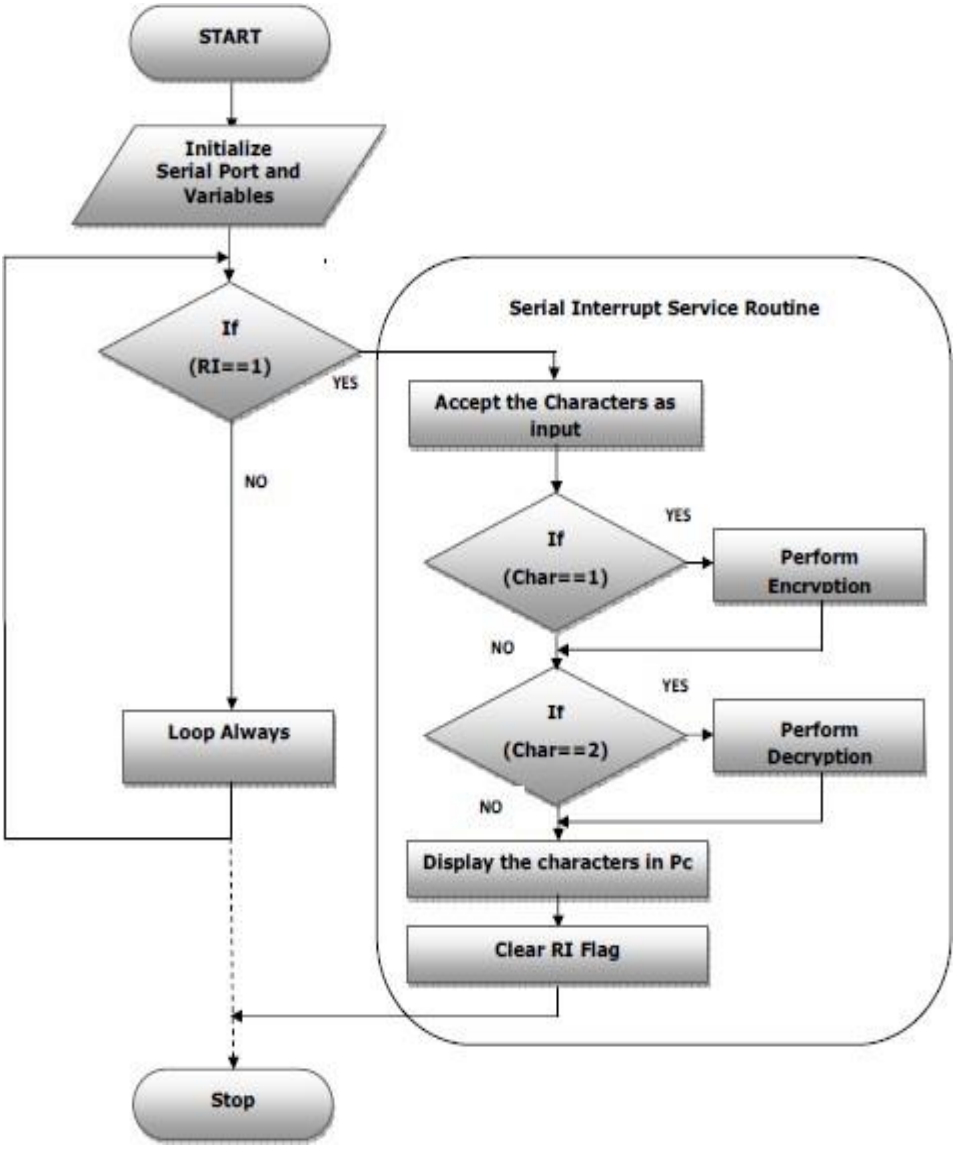
Cryptography(or cryptology; derived from Greek κρύπτω krýpto "hidden" and the verb γράφω gráfo "to write" or λέγειν legein "to speak") is the practice and study of hiding information. In modern times, cryptography is considered to be a branch of both mathematics and computer science, and is affiliated closely with information theory, computer security, and engineering. Cryptography is used in applications present in technologically advanced societies; examples include the security of ATM cards, computer passwords, and electronic commerce, which all depend on cryptography.

Until modern times, cryptography referred almost exclusively to encryption, the process of converting ordinary information (plaintext) into unintelligible gibberish (i.e., ciphertext). Decryption is the reverse, moving from unintelligible ciphertext to plaintext. A cipher
encryption and the reversing decryption. The detailed operation of a cipher is controlled
both by the algorithm and, in each instance, by a (ideally, known only to the communicants) for a specific message exchange context. Keys are important, as ciphers without variable keys are trivially breakable and therefore less than useful for most purposes. Historically, ciphers were often used directly for encryption or decryption, without additional procedures such as authentication or integrity checks.

/*----------

Example : Encryption & Decryption
Description : Get data from serial port, decrypted data displayed in serial window.
---------- */

Flow Chart



Code:

```
#include<reg51.h> //include the 8051 registers
#include<stdio.h> //include the std i/o header files

/******************** Variable Declaration ****************************/
unsigned int i,n,p,z=37;
unsigned char x[10],y[10];
unsigned char msg1[8],enter[]="\nEnter the No of Character : ";
code unsigned char menu[]="\n"
" Encryption Decryption Program "
"\n---------------------- "
"\n Press '1': Get Data from keyboard "
"\n Press '2': Display Decrypted Text ";

/******************** Function Declaration ****************************/
void serial();
void encrypt();
void decrypt();
void delay(unsigned int );

/********************** Main Function ****************************/

void main()
{
EA=1; //Enable the Global interrpt
ES=1; //Enable serial Interrupt
serial(); //call serial routine
delay(100);
while(1); //loop forever
}

/******************** Serial Function ****************************/
void serial()
{
SCON = 0x50; /* SCON: mode 1, 8-bit UART, enable rcvr */
TMOD |= 0x20; /* TMOD: timer 1, mode 2, 8-bit reload */
TH1 = 0xFD; /* TH1: reload value for 9600 baud,11.0592MHz*/
TR1 = 1; /* TR1: timer 1 run */
TI = 1; /* TI: set TI to send first char of UART */
i = 0;
while(menu[i] != '\0')
{
SBUF = menu[i];
delay(20);
i++;
z--;
if(z == 0)
{
z =37; //check end of line
SBUF = 0x0d; //carriage return
delay(50);
}
}
}
```

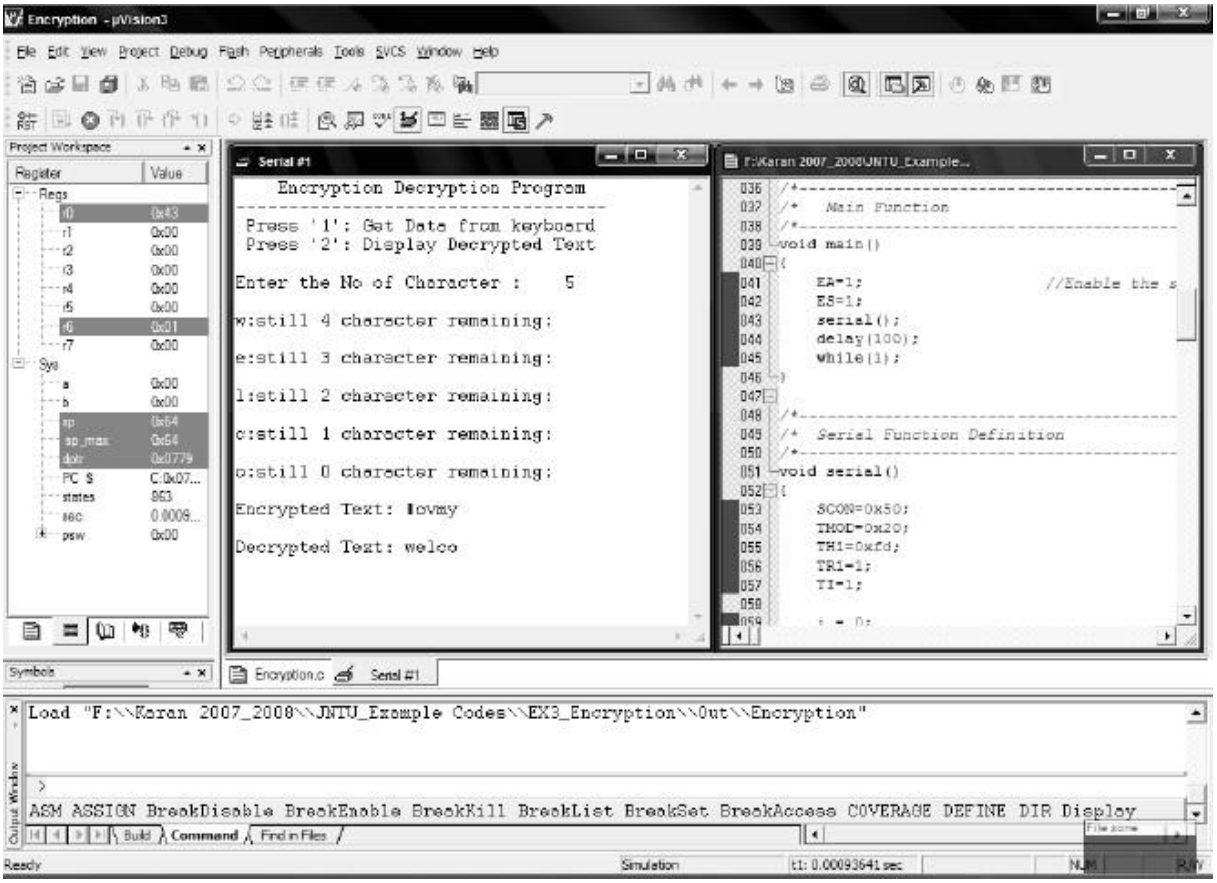
```
/********** Delay Subroutine **********/
void delay(unsigned int k)
{
unsigned int a;
for(a=0;a<k;a++);
}
/********** Encryption **********/
void encrypt()
{
i = 0;
SBUF = 0x0d; //CR
delay(100);
while(enter[i] != '\0')
{
SBUF = enter[i];
delay(100);
i++;
}
while(RI == 0);
n = SBUF;
RI = 0;
SBUF = n;
delay(100);
SBUF = '\n';
delay(100);
SBUF = '\n';
delay(100);
n = n-0x30;
for(i=0;i<n;i++)
{
while(RI==0);
msg1[i]=SBUF;
delay(100);
printf("\n");
SBUF=msg1[i];
delay(100);
p=(n-i)-1;
printf(":still %d character remaining:\n",p);
printf("\n");
RI=0;
}
printf("Encrypted Text: ");
for(i=0;msg1[i]!='\0';i++) //Encryption progress
{
x[i]=(msg1[i]+10);
delay(100);
printf("%c",x[i]);
}
delay(100);
printf("\n");
EA=1; //enable the serial interrupt
ES=1;
}
/********** Decryption **********/
void decrypt()
```

```
{
printf("\nDecrypted Text: ");
for(i=0;i<n;i++) //Decryption progress
{
y[i]=(x[i]-10);
delay(100);
printf("%c",y[i]);
}
printf("\n");
}
/************************** Serial Interrupt Routine
**********************/
void serin() interrupt 4
{
unsigned char z;
if(RI==1)
{
z=SBUF;
RI=0;
switch(z)
{
case 0x31: //Adcii '1'
encrypt(); //encrypt function
break;
case 0x32: //Ascii '2'
decrypt(); //decrypt function
break;
}
}
}
```

Execution:

Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

Output Simulation:



5. Develop necessary interfacing circuit to read data from a sensor and process using the 801 and 8051 boards. The data to be displayed on a PC monitor.

Aim

To Develop necessary interfacing circuit to read data from a sensor and process using the 801 and 8051 boards. The data to be displayed on a PC monitor.

Program

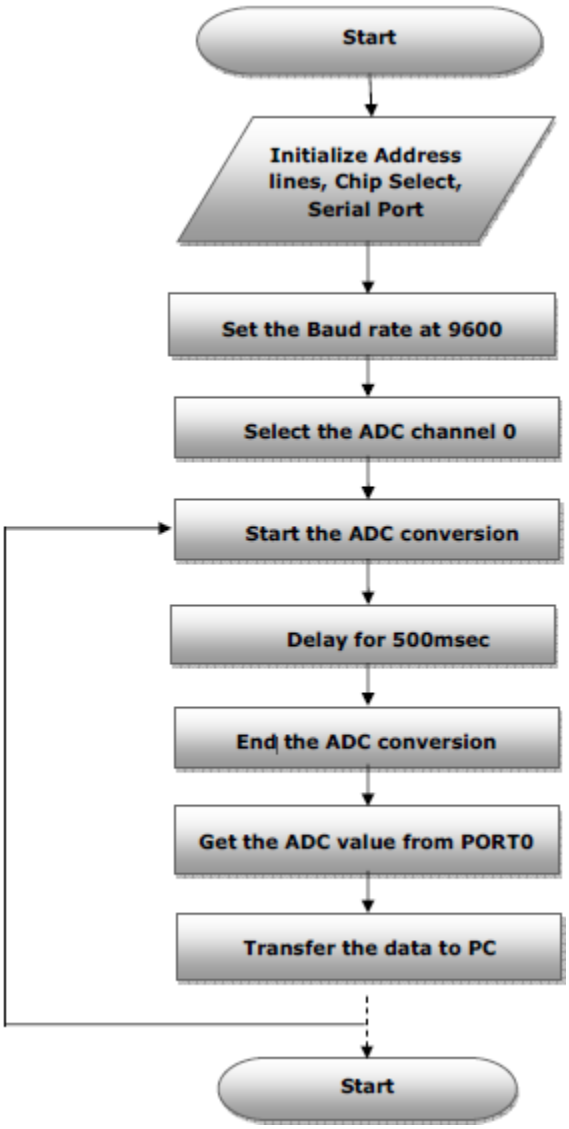
Analog to Digital Converter unit (ADC 0809)

ADC 0809 is an 8-channel 10
Digital form. In ADC section a jumper is provided to select either external analog input from signal conditioning as input source or can select internal 5V generator, which is variable from 0-5V. Th
the any of the port by using the Bus/connector. Reference voltage of 2.5V is given at the reference input so that the analog input span is 5V. In a sample program provided with the module the digital ou
Microcontroller, can be view on the hyper terminal of the PC
Features of ADC0809:

- Resolution: 8 Bits
- Operates ratio metrically or with 5VDC, 2.5VDC, or analog span adjusted voltage reference.
- Differential analog voltage inputs
- Works with 2.5V voltage reference.
- On-chip clock generator.
- 0V to 5V analog input voltage range with single 5V supply.
- No zero adjusts required.

/* -- */
/* Example : Program to read Temperature value from ADC */
/* -- */
/* -- */
/* Note :The ADC data lines are interfaced in the Port1 and the Obtained value in Port1 is converted to decimal value
/* -- */

Flow Chart



```
#include <stdio.h> //Define I/O Functions
#include <reg51.h> //Define 8051 Registers
#include <ctype.h>

//----------------------------
// ADC Control Lines
//----------------------------

sbit A0 = P2^7; //Address lines Initialization
sbit A1 = P2^6;
sbit A2 = P2^5;
sbit CS = P2^4; //Chip Select Pin
void serial(); //Serial Port Initialization
void delay1(int);
void delay2(int);
unsigned char READ_ADC(void);
unsigned char ch;
unsigned int i;

//----------------------------
//Delay Function
//----------------------------

void delay1(int n)
{
int i;
for(i=0;i<n;i++);
}
void delay2(int n)
{
int i;
for(i=0;i<n;i++)
delay1(1000);
}


//-------------------------------------
//Serial Port Initialization
//-------------------------------------

void serial()
{
SCON=0x50; //Serial Mode-1, REN enable
TMOD=0x20; //Timer1, Mode2, 8-bit (Auto Relod mde)
TH1=0xfd; //9600 Baud rate at 11.0592MHz
TR1=1; //Timer 1 ON
TI=1; //Transmit Interrupt Enable
}
```

```
//-------------------------
//ADC Function
//-------------------------

unsigned char READ_ADC()
{
unsigned char ADC_DATA;
CS = 0; delay2(1); //Triggering ADC by chip Slt Pin
CS = 1; delay2(1);
CS = 0; delay2(1);
CS = 1; delay2(1);
ADC_DATA = P1; //Get the value from Port1
return(ADC_DATA);
}

//-------------------------
// Main Program
//-------------------------

void main(void)
{
P1=0xFF;
serial(); //Serial port Initialization
A0 = 0; // channel '0' LM35 Sensor
A1 = 0;
A2 = 0;
printf("ADC Demo - Channel '0' LM35(Temp Sensor) \n");
printf("-----------------------------\n");
while(1)
{
ch = READ_ADC(); //Get the value from Channel-0
printf("\rCH0(Temperature) = %2bd'C",toascii(ch*2));
delay2(2);
}
}
```

Execution:

Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

7. Sort RTOs (mCOS) on to 89CS1 board and Verify.

Aim

To Sort RTOs (mCOS) on to 89CS1 board and Verify.

Program

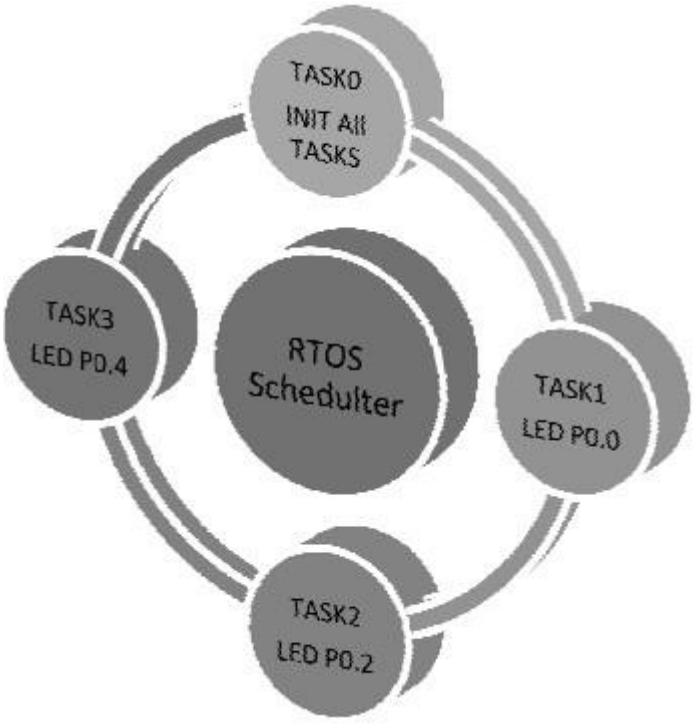
Example 1: Simple RTOS(Multitasking)

Task0 : Task0 Initialize or creates Task1, 2 and 3.

Task1 : Port P0.0(LED0), it blinks pre-defined time intervals by using `os_wait (K_TMO, 50, 0); it waits up to 50 ticks (500msec)`

Task2 : Port P0.2(LED1), it blinks pre-defined time intervals by using `os_wait (K_TMO, 100, 0); it waits up to 100 ticks (1sec)`

Task3 : Port P0.4(LED2), it blinks pre-defined time intervals by using `os_wait (K_TMO, 150, 0); it waits up to 150 ticks(1.5sec)`



```
/*************************************************************************
***/
/* Round Robin Scheduling Multitasking */
/* Note: LED's Blinked in different Delay intervals */
/*************************************************************************
***/

#include <rtx51tny.h> // RTX-51 tiny functions & defines
#include <reg52.h> /* 8051 Register */
sbit LED0 = P0^0; /* LED0 for task1 */
sbit LED1 = P0^2; /* LED0 for task2 */
sbit LED2 = P0^4; /* LED0 for task3 */


/*************************************************************************
****/
/* Task 0 'job0': RTX-51 tiny starts execution with task 0 */
/*************************************************************************
****/

job0 () _task_ 0
{
P1 = 0x00; //P1 make it output port
os_create_task (1); /* start task 1 */
os_create_task (2); /* start task 2 */
os_create_task (3); /* start task 3 */
while (1) { /* endless loop */
os_wait (K_TMO, 5, 0); /* wait for timeout: 5 ticks */
}
}

/*************************************************************************
**/
Task 1 'job1': RTX-51 tiny starts this task with os_create_task (1)
/*************************************************************************
**/

job1 () _task_ 1
{
while (1) { /* endless loop */
LED0 = 1;
os_wait (K_TMO, 30, 0);
LED0 = 0;
os_wait (K_TMO, 30, 0); /* wait for timeout: 10 ticks */
}
}
```

```
/*********************************************************************
**/
/* Task 2 'job2': RTX-51 tiny starts this task with os_create_task (2) */
/*********************************************************************
**/

job2 () _task_ 2
{
while (1) { /* endless loop */
LED1 = 1;
os_wait (K_TMO, 50, 0);
LED1 = 0;
os_wait (K_TMO, 50, 0); /* wait for timeout: 50 ticks */
}
}

/*********************************************************************
****/
/* Task 3 'job3': RTX-51 tiny starts this task with os_create_task (3) */
/*********************************************************************
*****/

job3 () _task_ 3
{
while (1) { /* endless loop */
LED2 = 1;
os_wait (K_TMO, 80, 0);
LED2 = 0;
os_wait (K_TMO, 80, 0); /* wait for timeout: 80 ticks */
}
}
```

Execution:

Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

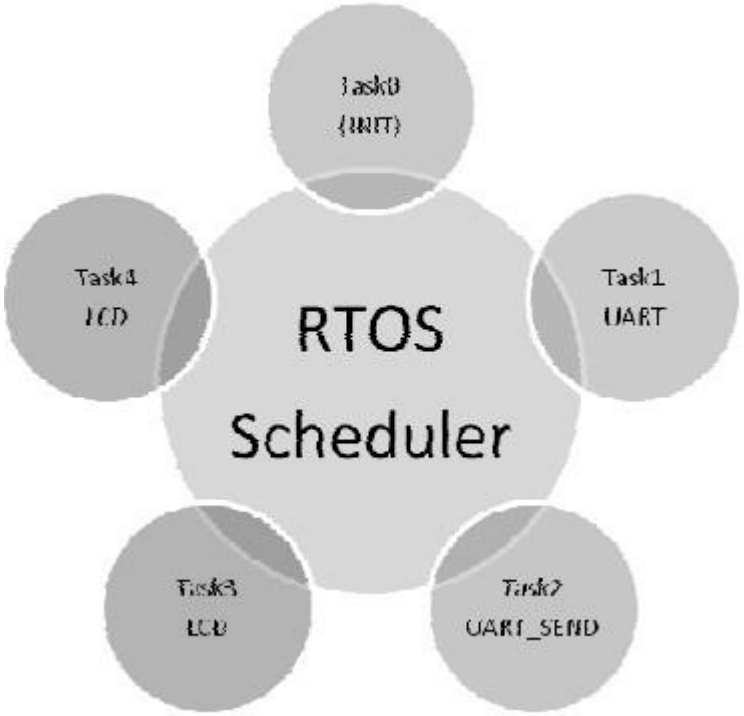
Example 2: Cooperative Scheduling Multitasking

Example 2: Cooperative Scheduling Multitasking

8051 MCU act as task scheduler, by executing task simultaneously by using Co-operative scheduling. It transmits (or receive) message through UART (PC's Hyper terminal), LCD Display and Blinks LED.

This multitasking is distinguished as FIVE tasks (INIT, UART, UART_SEND, LED, and LCD).

Task 0: INIT – Creates all the following FOUR tasks: uart, uart_send, led and lcd.
Task 1: Initialize serial port values for 9600 baud rate
Task2: The RTOS is ready to send data to pc's hyper-terminal window.
Task3: Initialize LCD, commands (clear, display, cursor on/off), and display the Message through lcd's 1st and 2nd row.
Task4: As per the commands LED's status could be ON/OFF, connected with externally pulled-up PORT0



```
/*********************************************************************
**/
/* Cooperative Scheduling Multitasking */
/* Note: LED Blink, UART, LCD (runs simultaneously) */
/*********************************************************************
**/

#include <rtx51tny.h> // RTX-51 tiny functions & defines
#include <stdio.h>
#include <reg52.h> //8051 Register
#define LCD_DATA P1 //Define LCD_DATA Lines to Port(Port1)
#define LED P0 // Define LED for PORT 0

/*****************Define LCD control pins************************/

sbit RS = P3^5; //Register Select
sbit RW = P3^6; //LCD Read/Write
sbit lcd_e = P3^7; //LCD Enable
code unsigned char msg[] = (" 8051 MCOS RTOS "); //Display the message
code unsigned char msg1[] = (" MULTITASKING ");
void lcd_cmd(unsigned char);
void lcd_display(unsigned char);

/*********************************************************************/
/* Task 0 : RTX-51 tiny Initializ task */
/*********************************************************************/

void init(void)_task_ 0
{
os_create_task (1); // start task 1 INIT_UART
os_create_task (2); // start task 2 SEND_UART
os_create_task (3); // start task 2 LED Blink
os_create_task (4); // start task 2 LCD Display
while (1)
{ // endless loop
os_wait (K_TMO, 5, 0); // wait for timeout: 5 ticks
}
}
```

```
/*********************************************************************/
/* Task 1 : RTX-51 tiny starts initialize serial port with task 0 */
/*********************************************************************/

void uart (void) _task_ 1
{
SCON = 0x50; // SCON: mode 1, 8-bit UART, enable rcvr
TMOD |= 0x20; // TMOD: timer 1, mode 2, 8-bit reload
TH1 = 0xFD; // TH1: reload value for 9600/11.0592MHz
TR1 = 1; // TR1: timer 1 run
TI = 1; // TI: set TI to send first char of UART
}

/*********************************************************************
*/
/* Task 2 : RTX-51 tiny starts send UART data with task 0 */
/*********************************************************************/
void uart_send(void) _task_ 2
{
while (1)
{
os_wait (K_TMO, 10, 0);
SBUF = 'A';
}
}

/*********************************************************************/
/* Task 3 : RTX-51 tiny starts LED Blink with task 0 */
/*********************************************************************/

void led(void) _task_ 3
{
while (1) { // endless loop
LED = 0x55;
os_wait (K_TMO, 30, 0);
LED = 0xAA;
os_wait (K_TMO, 30, 0);
}
}
```

```
/*************************************************************************
**/
/* Task 4 : RTX-51 tiny starts LCD Initializaion with task 0 */
/*************************************************************************
**/

void lcd(void) _task_ 4
{
while (1) { // endless loop
unsigned char i;
lcd_cmd(0x38); //2x16 Character 5x7 dot
os_wait (K_TMO, 2, 0);
lcd_cmd(0x0c); //Display On, cursor off
os_wait (K_TMO, 2, 0);
lcd_cmd(0x06); //Shift Cursor to right
os_wait (K_TMO, 2, 0);
lcd_cmd(0x01); //Clear display screen
os_wait (K_TMO, 2, 0);

//----------------------------------------
// First Line Message Display
//----------------------------------------

lcd_cmd(0x80); //First Line Initialization
os_wait (K_TMO, 2, 0);
i=0;
while(msg[i]!='\0')
{
lcd_display(msg[i]);
i++;
}
os_wait (K_TMO, 4, 0);

//----------------------------------------
// Second Line Message Display
//----------------------------------------

lcd_cmd(0xc0); //Second Line Initialization
os_wait (K_TMO, 2, 0);
i=0;
while(msg1[i]!='\0')
{
lcd_display(msg1[i]);
i++;
}
os_wait (K_TMO, 4, 0);
}
}
```

```
//---------------------------------
// LCD command Function
//---------------------------------

void lcd_cmd(unsigned char cmnd)
{
LCD_DATA = cmnd;
RS = 0; RW = 0;
lcd_e = 1;
os_wait (K_TMO, 2, 0);
lcd_e = 0;
}

//---------------------------------
// LCD Data Function
//---------------------------------

void lcd_display(unsigned char dat)
{
LCD_DATA = dat;
RS = 1; RW = 0;
lcd_e = 1;
os_wait (K_TMO, 2, 0);
lcd_e = 0;
}
```

Execution:

Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

8. Simulate on elevator movement using RTO's on 89CSI board.

8051 MCU act as a Real time task scheduler, it executes task simultaneously by using Co-operative scheduling. For an elevator, it displays corresponding floors status in led, status displays in LCD/UART, uses matrix keypads for floors key press.

Totally six tasks (INIT, UART, UART_SEND, LCD_INIT, KEY_OUT, KEY_IN).

Task 0: INIT – Creates all remaining five tasks, uart, lcd, key, it waits 5 ticks

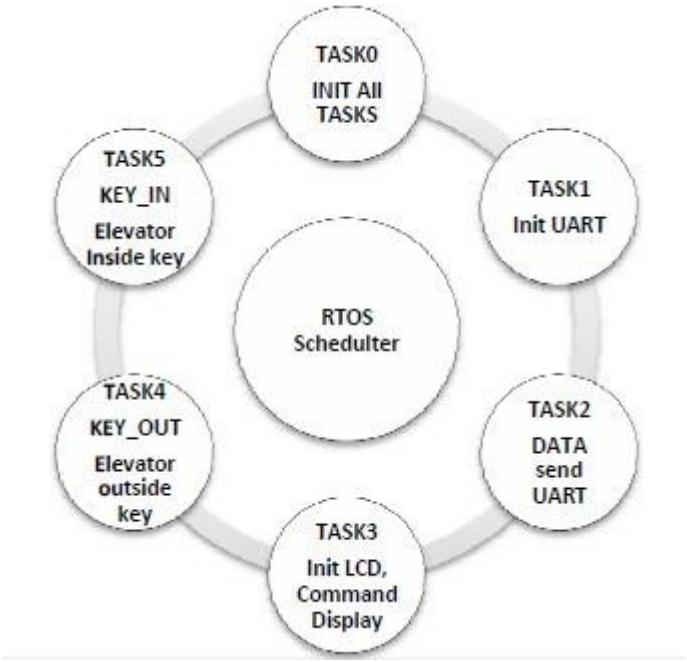
Task 1: Initialize serial port values for 9600 baud rate

Task2: The system is ready to send data to pc's hyper-terminal window.

Task3: Initializes LCD, commands (clear, display, cursor on/off), and display the message to lcd's 1st and 2nd row.

Task4: KEY_OUT – The key outside to the Elevator (Port2). Task scans for an occurrence of Key Press of the floor key, if TRUE serves it by lilting on the LED for corresponding floor. It waits for few ticks and displays message in LCD/UART as (say) "Open" and "Close"

Task5: KEY_IN – The key inside to the Elevator (Port2). System waits for the user to have a press on the keys and on while pressing key, LED status shows scrolling of UP/Down side.



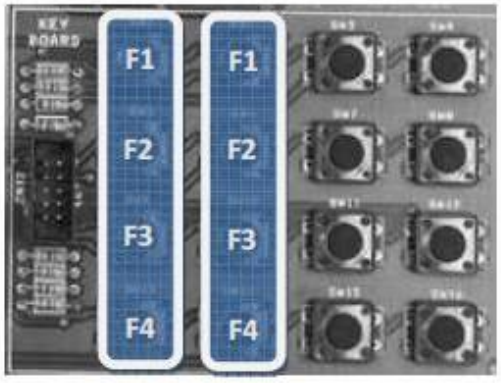
```
/*********************************************************************
****/
/* Task_Init.h:
/* Project specific header for the Elevator example
/*********************************************************************
****/

#define LCD_DATA P1        //Define LCD_DATA Lines to Port(Port1)
#define LED P0             // Define LED for PORT 0

#define INIT 0          //Task0 for Initialize all task
#define UART 1             //Task1 for UART
#define UART_SEND 2        //Task2 for UART_send data
#define LCD_INIT 3         //Task3 for LCD Initialize
#define KEY_OUT 4          //Task4 for Key out for outside elevator
#define KEY_IN 5           //Task5 for Key in for inside elevator

code unsigned char msg0[] = ("ELEVATOR SIMULATION ");     //serial communi
msg
code unsigned char msg[] = (" RTX-51 BASED “);            //Display the message
code unsigned char msg1[] = (" ELEVATOR SIMUL ");
code unsigned char msg2[] = ("OPEN ");                    //lcd/uart msg
code unsigned char msg3[] = ("CLOSE");
```

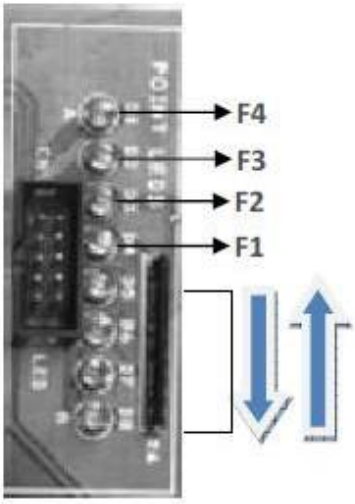
Hardware Details

Floor Indication
Direction Status

Elevator Keys

Inside(up) Outside(down)





```
/*****************************************************************
****/
/* Elevator Simulation by using RTOS(Multitasking) */
/*****************************************************************
****/

#include <rtx51tny.h> // RTX-51 tiny functions & defines
#include <reg51.h> //8051 Register
#include "task_init.h" // project specific header file

/**********************Define LCD control
pins*************************/

sbit RS = P3^5; //Register Select
sbit RW = P3^6; //LCD Read/Write
sbit lcd_e = P3^7; //LCD Enable
unsigned char R,C,ch;
unsigned int i=0,j=0;
code unsigned char Key[4][2] = { '0','5', //Matrix Keypad Character
                                 '1','6', //Initialization
                                 '2','7', //Initialization
                                 '3','8', //Initialization
                                 };

void lcd_cmd(unsigned char);
void lcd_display(unsigned char);
void open(void);
void close(void);
void DelayMs(int);
void upscroll(int n);
void downscroll(int n);

/*****************************************************************
**/
/* Task 0 : RTX-51 tiny Initializ task */
/*****************************************************************
**/

void init(void)_task_ INIT
{
P0 = 0x00;
os_create_task (UART); // start task 4 INIT_UART
os_create_task (UART_SEND); // start task 5 UART_SEND
os_create_task (LCD_INIT); // start task 1 INIT_LCD
os_create_task (KEY_OUT); // start task 3 Floor Status
os_create_task (KEY_IN); // start task 3 Floor Status
while (1)
{ // endless loop
os_wait (K_TMO, 5, 0); // wait for timeout: 5 ticks
}
}
```

```
/*********************************************************************
**/
/* Task 1 : RTX-51 tiny starts initialize serial port with task 0 */
/*********************************************************************
**/

void uart (void) _task_ UART
{
SCON = 0x50; // SCON: mode 1, 8-bit UART, enable rcvr
TMOD |= 0x20; // TMOD: timer 1, mode 2, 8-bit reload
TH1 = 0xFD; // TH1: reload value for 9600 @ 11.0592MHz
TR1 = 1; // TR1: timer 1 run
TI = 1; // TI: set TI to send first char of UART
}

/*********************************************************************
*/
/* Task 2 : RTX-51 tiny starts send UART data with task 0 */
/*********************************************************************
*/

void uart_send(void) _task_ UART_SEND
{
while(msg0[j]!='\0')
{
SBUF=msg0[j];
j++;
os_wait (K_TMO, 2, 0);
} os_wait (K_TMO, 2, 0);
}

/*********************************************************************
***/
/* Task 3 : RTX-51 tiny starts LCD Initializaion with task 0 */
/*********************************************************************
***/

void lcd(void) _task_ LCD_INIT
{
while (1) { // endless loop
unsigned char 1;
lcd_cmd(0x38); //2x16 Character 5x7 dot
os_wait (K_TMO, 2, 0);
lcd_cmd(0x0c); //Display On, cursor off
os_wait (K_TMO, 2, 0);
lcd_cmd(0x06); //Shift Cursor to right
os_wait (K_TMO, 2, 0);
lcd_cmd(0x01); //Clear display screen
os_wait (K_TMO, 2, 0);
```

```
//----------------------------------------
// First Line Message Display
//----------------------------------------

lcd_cmd(0x80); //First Line Initialization
os_wait (K_TMO, 2, 0);
i=0;
while(msg[i]!='\0')
{
lcd_display(msg[i]);
i++;
} os_wait (K_TMO, 4, 0);

//----------------------------------------
// Second Line Message Display
//----------------------------------------

lcd_cmd(0xc0); //Second Line Initialization
os_wait (K_TMO, 2, 0);
i=0;
while(msg1[i]!='\0')
{
lcd_display(msg1[i]);
i++;
}

os_wait (K_TMO, 4, 0);
os_delete_task (LCD_INIT);
}
}

/*****************************Elevator Open
*******************************/

void open(void)
{
lcd_cmd(0xca); //Second Line Initialization
os_wait (K_TMO, 2, 0);
i=0;
while(msg2[i]!='\0')
{
lcd_display(msg2[i]);
os_wait (K_TMO, 1, 0);
SBUF = msg2[i];
i++;
}
os_wait (K_TMO, 4, 0);
}
```

```
/*****************************Elevator Close
*******************************/

void close(void)
{
lcd_cmd(0xca); //Second Line Initialization
os_wait (K_TMO, 2, 0);
i=0;
while(msg3[i]!='\0')
{
lcd_display(msg3[i]);
os_wait (K_TMO, 1, 0);
SBUF = msg3[i];
i++;
}
os_wait (K_TMO, 4, 0);
}

/*****************************LED UP scroll
*******************************/

void upscroll(int n)
{
int i,j=0;
for(i=0;i<n;i++)
{
for(j=0x10;j<=0x80;j<<=1) //shift led one position
{
P0=j; //Initialize Port1
os_wait (K_TMO, 20, 0);
}
}
}

/*****************************LED Downscroll
*******************************/
void downscroll(int n)
{
int i,j=0;
for(i=0;i<n;i++)
{
for(j=0x80;j>=0x10;j>>=1) //shift led one position
{
P0=j; //Initialize Port1
os_wait (K_TMO, 20, 0);
}
}
}
```

```
/*****************************LCD Command
*****************************/

void lcd_cmd(unsigned char cmnd)
{
LCD_DATA = cmnd;
RS = 0; RW = 0;
lcd_e = 1;
os_wait (K_TMO, 2, 0);
lcd_e = 0;
}

/*****************************LCD Display
*****************************/

void lcd_display(unsigned char dat)
{
LCD_DATA = dat;
RS = 1; RW = 0;
lcd_e = 1;
os_wait (K_TMO, 2, 0);
lcd_e = 0;
}

/*************************************************************************
**/
/* Task 4 : RTX-51 tiny starts Elevator outside key with task 0 */
/*************************************************************************
**/

void Key_Scan_out(void) _task_ KEY_OUT
{
while(1){
unsigned int i = 0;
//Scanning for Row Value
P2 = 0x0F; //Initialize Port2 to 0Fh
while(P2 == 0x0F);
if(P2 == 0x0E) //Checking from Row 0 to 3
R = 0;
else if(P2 == 0x0D)
R = 1;
else if(P2 == 0x0B)
R = 2;
else if(P2 == 0x07)
R = 3;
//Scanning for Column Value
P2 = 0xF0;
//Initialize Port2 to F0h
while(P2 == 0xF0);
if(P2 == 0xE0) //Checking from Column 0 to 3
C = 0;
else if(P2 == 0xD0)
C = 1;
else if(P2 == 0xB0)
C = 2;
```

```
else if(P2 == 0x70)
C = 3;
os_wait (K_TMO, 10, 0);
//Floor Status
ch = Key[R][C];
if(ch=='0')
{
downscroll(1);
open(); os_wait (K_TMO, 200, 0);
close(); os_wait (K_TMO, 2, 0);
os_create_task (KEY_IN); //create keyscan inside
P0 =0x08;
}
if(ch=='1')
{
downscroll(2);
open(); os_wait (K_TMO, 200, 0);
close(); os_wait (K_TMO, 1, 0);
os_create_task (KEY_IN); //create keyscan inside
P0 = 0x04;
}
if(ch=='2')
{
downscroll(3);
open(); os_wait (K_TMO, 200, 0);
close(); os_wait (K_TMO, 1, 0);
os_create_task (KEY_IN); //create keyscan inside
P0 = 0x02;
}
if(ch=='3')
{
downscroll(4);
open(); os_wait (K_TMO, 200, 0);
close(); os_wait (K_TMO, 1, 0);
os_create_task (KEY_IN); //create keyscan inside
P0 = 0x01;
}
}
}
```

```
/*************************************************************************
**/
/* Task 5 : RTX-51 tiny starts Elevator inside key with task 0 */
/*************************************************************************
**/

void Key_Scan_in(void) _task_ KEY_IN
{
while(1){
unsigned int i = 0;
//Scanning for Row Value
P2 = 0x0F; //Initialize Port2 to 0Fh
while(P2 == 0x0F);
if(P2 == 0x0E) //Checking from Row 0 to 3
R = 0;
else if(P2 == 0x0D)
R = 1;
else if(P2 == 0x0B)
R = 2;
else if(P2 == 0x07)
R = 3;
//Scanning for Column Value
P2 = 0xF0; //Initialize Port2 to F0h
while(P2 == 0xF0);
if(P2 == 0xE0) //Checking from Column 0 to 3
C = 0;
else if(P2 == 0xD0)
C = 1;
else if(P2 == 0xB0)
C = 2;
else if(P2 == 0x70)
C = 3;
os_wait (K_TMO, 10, 0);
//Elevator Inside
ch = Key[R][C];
if(ch=='5')
{
upscroll(1);
open(); os_wait (K_TMO, 200, 0);
close(); os_wait (K_TMO, 2, 0);
P0 = 0x08;
os_create_task (KEY_OUT); //create keyscan inside
}
if(ch=='6')
{
upscroll(2);
open(); os_wait (K_TMO, 200, 0);
close(); os_wait (K_TMO, 2, 0);
P0 = 0x04;
os_create_task (KEY_OUT); //create keyscan inside
}
if(ch=='7')
{
upscroll(3);
open(); os_wait (K_TMO, 200, 0);
```

```
close(); os_wait (K_TMO, 2, 0);
P0 = 0x02;
os_create_task (KEY_OUT); //create keyscan inside
}
if(ch=='8')
{
upscroll(4);
open(); os_wait (K_TMO, 200, 0);
close(); os_wait (K_TMO, 2, 0);
P0 = 0x01;
os_create_task (KEY_OUT); //create keyscan inside
}
}
}
```

Execution:

Note: After Loading corresponding Examples Hex file located in "OUT" Folder to the microcontroller kit, press "RST" Button, user program now executes.

TEXT BOOKS:

1) Computers and Components, Wayne Wolf, Elseveir.
2) The 8051 Microcontroller, Third Edition, Kenneth J. Ayala, Thmson .

REFERENCES:

1) Embedding system building blocks, Labrosse, via CMP publishers.
2) Embedded Systems, Raj Kamal, TMH.
3) Micro Controllers, Ajay V Deshmukhi, TMH.
4) Embedded System Design, Frank Vahid, Tony Givargis, John Wiley.
5) Microcontrollers, Raj Kamal, Pearson Edition.
6) An Embedded Software Primer, David E. Simon, Pearson Edition.
7) 'Embedded/Real-Time Systems', KVKKF Prasad, Dreamtech, Press.