A Project Report on

CONTROLLING OF HOME APPLIANCES USING TV REMOTE

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

M. SAI ESWAR	**18A31A0447**
K. MUKUNDA NAGA SAIPRIYA	**18A31A0411**
M. MRUDULA	**18A31A0415**
Ch. ROHINI KUMAR	**18A31A0435**
K. SURYA TEJA MANIKANTA	**18A31A0443**

Under the guidance of

Mr. B. GANDHI, M.Tech

Assistant Professor of ECE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

PRAGATI ENGINEERING COLLEGE

(Approved by AICTE, Permanently Affiliated to JNTUK, KAKINADA & Accredited by NBA)

1-378, A.D.B. Road, Surampalem, Near Peddapuram-533437

2018-22

PRAGATI ENGINEERING COLLEGE

(Approved by AICTE, Permanently Affiliated to JNTUK, KAKINADA & Accredited by NBA)

1-378, A.D.B. Road, Surampalem, Near Peddapuram – 533437

CERTIFICATE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

This is to certify that the Project Report entitled **"CONTROLLING OF HOME APPLIANCES USING TV REMOTE"** is being submitted by M. Sai Eswar (18A31A0447), K.Mukunda Naga Saipriya (18A31A0411), M.Mrudula (18A31A0415), Ch.Rohini Kumar (18A31A0435), K.Surya Teja Manikanta (18A31A0443) in partial fulfillment for the award of the Degree of **Bachelor of Technology** in Electronics & Communication Engineering of Pragati Engineering College, for the record of bonafide work carried out by them.

Dr.V.Sailaja, M.E, Ph.D	**Mr. B.Gandhi ,** M.Tech
Professor & HOD of ECE	Assistant Professor

ACKNOWLEDGEMENT

We express our thanks to project guide **B.Gandhi**, Assistant Professor of Electronics and Communication Engineering, who deserves a special note of thanks and gratitude, for having extended their fullest co-operation and guidance, without this, project would never have materialized.

We express our deep sense of gratitude to **Dr.V.Sailaja , HOD of ECE and Mr.V.Prasanth Incharge HOD of ECE**, for having shown keen interest at every stage of development of our project work and for guiding us in every aspect.

We wish to express our special thanks to our beloved **Dr.K.Satyanarayana**, Principal for giving guidelines and encouragement.

We wish to express our special thanks to our beloved **Dr. S. Sambhu Prasad**, Director and

Dr. G. Raghuram, Director-Academics for giving guidelines and knowledge throughout the curriculum.

We wish to express sincere gratitude to our beloved and respected **Dr. P. Krishna Rao**, **Chairman** and **Sri. M. V. Haranatha Babu**, **Director (Management)**, and **Sri. M. Satish**, **Vice-President** for their encouragement and blessing.

We are thankful to all our faculty members of the department for their valuable suggestions. Our sincere thanks also extended to all the teaching and the non-teaching staff of Pragati Engineering College.

M. SAI ESWAR	**18A31A0447**
K. MUKUNDA NAGA SAIPRIYA	**18A31A0411**
M. MRUDULA	**18A31A0415**
Ch. ROHINI KUMAR	**18A31A0435**
K. SURYA TEJA MANIKANTA	**18A31A0443**

LIST OF TABLES	2
LIST OF FIGURES	32
ABSTRACT	
CHAPTER NAME	**PAGE NO**
CHAPTER-1 INTRODUCTION	1-7
1.1 Introduction to Embedded Systems	2-3
1.2 Overview of Embedded System Architecture	3-5
1.3 Characteristics of Embedded system	6
1.4 Applications of Embedded system	6
1.5 Micro Controllers vs Micro Processors	6
1.6 Micro Controllers for Embedded systems	7
CHAPTER-2 INTRODUCTION TO PROJECT	8-9
2.1 Introduction	9
2.2 Block Diagram	9
CHAPTER-3 MICROCONTROLLER	10-25
3.1 AT8952 Microcontroller	11-12
3.2 Features	12
3.3 Architecture of 8052 Microcontroller	12-13
3.4 Pin out diagram of 8052	13-19
3.4.1 CPU Registers	14
3.4.2 Program Counter	14
3.4.3 Data pointer	14
3.4.4 Program Status Word	15-16

3.4.5 Stack and Stack Pointer 16-17

3.4.6 Internal Memory 17-18

3.4.7 Internal Rom 18

3.4.8 Internal Ram 256 Bytes 18-19

3.5 I/O PORTS 19-25

3.5.1 Thirty-two Input/Output pins 19-20

3.5.2 Interrupts 20-21

3.5.3 Serial Interface 22

3.5.4 Two 16-bit timer/counter 22

3.5.5 Oscillator and clock circuits 23-25

3.6 Applications 25

CHAPTER-4 HARDWARE COMPONENTS 26-31

4.1 Power Supply 27-30

4.1.1 Led 27

4.1.2 Capacitor 27-28

4.1.3 Transformer 28-29

4.1.4 Bridge Rectifier 29

4.1.5 Regulator 29-30

4.2 IR Receiver 30

4.3 Relay Driver 30-31

4.4 Relay	31
4.5 TV Remote	31-32
4.6 Load	32
CHAPTER-5 DESIGN IMPLEMENTATION	33-36
5.1 Circuit Diagram	34
5.2 Working	34
5.3 Interfacing of TSOP1738 with 8052	35
5.4 Interface the 8052 with the 2-relay module	36
CHAPTER-6 SOFTWARE USED	37-43
6.1 Keil Micro vision	38-42
6.2 Willar	42-43
CHAPTER-7 DESIGN OUTPUT	44-45
7.1 Result	45
CHAPTER-8 CONCLUSION	47
FUTURE SCOPE	47
BIBLIOGRAPHY	47
ANNEXURE	48-50

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	Layered architecture of embedded system	3
1.2	Basic block diagram of an embedded system	4
2.1	Block diagram	9
3.1	Pin diagram of 8052 microcontroller	11
3.2	Architecture of 8052 microcontroller	12
3.3	Pin diagram of 8052 microcontroller	13
3.4	Register format of Program Status Word (PSW)	15
3.5	Oscillator circuit	24
4.1	Led	27
4.2	Capacitor	28
4.3	Step-down Transformer	28
4.4	Bridge Rectifier	29
4.5	7805 Pinout diagram	30
4.6	IR Receiver	30
4.7	Relay driver	31
4.8	Relay and 2-relay module	31
4.9	TV remote	32
4.10	Different types of loads	32
5.1	Circuit Diagram	34

5.2	Interface of the TSOP1738 with 8052	35
5.3	Interface the 8052 with 2-relay module	36
6.1	Project Creation	38
6.2	Select device from target	39
6.3	To add the file	40
6.4	Writing the code	40
6.5	Creation of the hex file	41
6.6	Building of the target	42
6.7	Built output	42
6.8	Programming kit	42
6.9	Software Window	43
7.1	Photocopy of the project	45

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
3.1	Pin description of PSW	15-16
3.2	Description of register bank selection bits	16

ABSTRACT

The project is designed to operate electrical loads using a TV remote. The remote transmits coded infrared data which is then received by a sensor interfaced to the control unit. The system operates electrical loads depending on the data transmitted from the TV remote. Operating conventional wall switches is difficult for elderly or physically handicapped people. This proposed system solves the problem by integrating house hold appliances to a control unit that can be operated by a TV remote. NEC based coded data sent from the TV remote is received by an IR receiver interfaced to the microcontroller of 8052 family. The program on the microcontroller refers to the NEC code to generate respective output based on the input data to operate a set of relays through a relay driver IC. The loads are interfaced to the control unit through the relays. The system can be used in existing domestic area for either operating the loads through conventional switches or with the tv remote. The project can be enhanced by using radio frequency technology where the operational range shall be independent of line of sight distance as often encountered with IR type of remote control.

CHAPTER 1

1.INTRODUCTION

1.1 INTRODUCTION TO EMBEDDED SYSTEM

Embedded system was in existence even before the IT revolution. One of the very first recognizably modern embedded system was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT instrumentation Laboratory. At the project’s inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Autonetics D-17 guidance computer for the minuteman missile, released in 1961. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits. This program alone reduced prices on quad NAND gate ICs from $1000/each to $3/each, permitting their use in commercial products.

Since these early applications in the 1960s, embedded system have come down in price and there has been a dramatic rise in processing power and functionality. An early microprocessor for example, the Intel 4004, was designed for calculators and other small systems but still required external memory and support chips. In 1978 National Engineering Manufactures Association released a “standard” for programmable microcontrollers, including almost any computer-base controllers, such as single board computers, numerical, and event-based controllers.

As the cost of microprocessors and microcontrollers fell it became feasible to replace expansive knob-based analog components such as potentiometers and variable capacitors with up/down buttons or knob read out by a microprocessor even in consumer products. By the early 1980s, memory, input and output system components had been integrated into same chip as processor forming a microcontroller. Microcontrollers find applications where a general-purpose computer would be too costly.

A comparatively low-cost microcontroller may be programmed to fulfil the same role as a large number of separate components. Although in this context an embedded system is usually more complex than a traditional solution, most of the complexity is contained within the microcontroller itself very few additional components may be needed and most of the design effort is in the software,

Software prototype and test can be quicker compared with the design and construction of a new circuit not using an embedded processor.

1.2 OVERVIEW OF EMBEDDED SYSTEM ARCHITECTURE

Every embedded system consists of custom-built around a Central Processing unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chips is also called the "FIRMWARE".

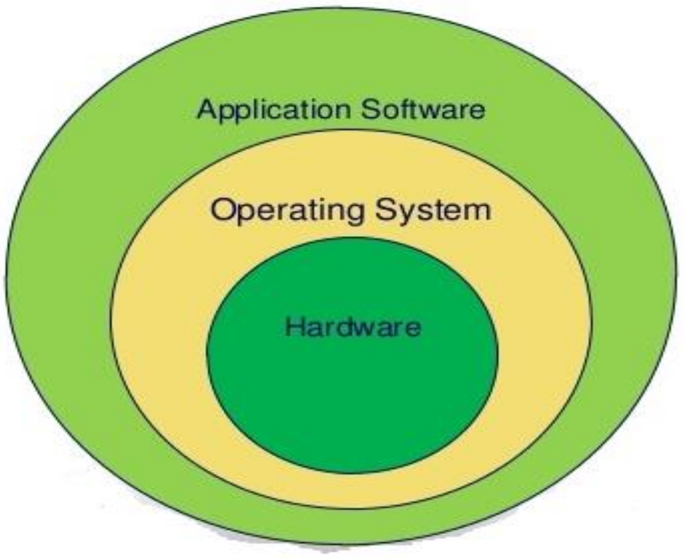


Fig 1.1 Layerd architecture of an embedded system

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For applications involving complex processing, it is advisable to have an operating system. In such case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time and you don't need to reload new software.

Now let us see the detail of the various building blocks of the hardware of an embedded system.

- Central Processing Unit (CPU)
- Memory (Read only memory and Random-access memory)

- Input Devices
- Output Devices
- Communication interfaces
- Application specific circuitry

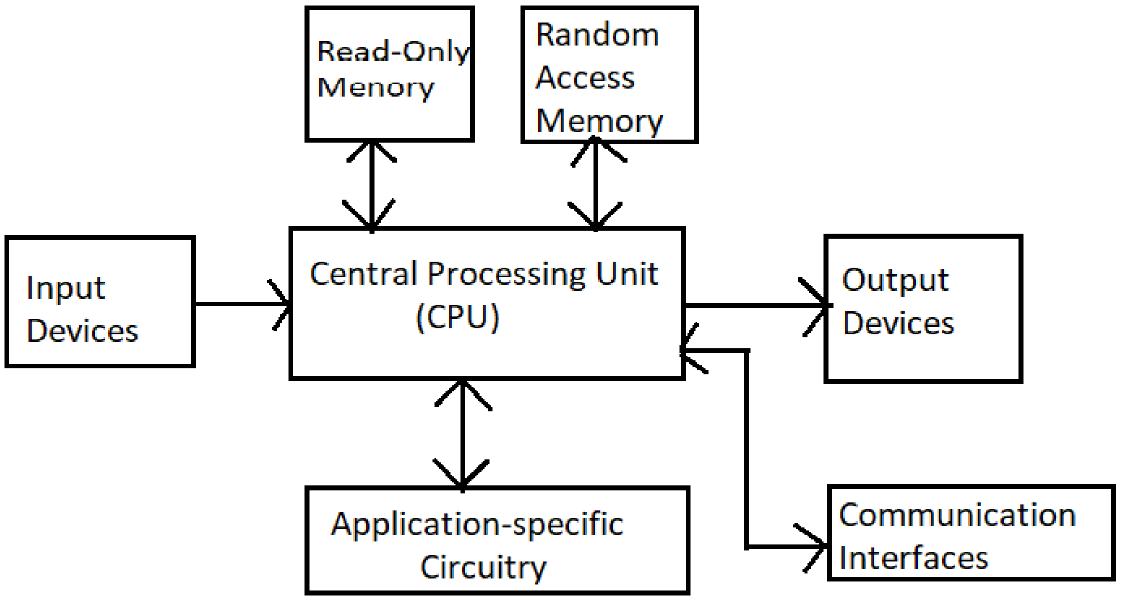


Fig 1.2 Basic block diagram of an embedded system

A general-purpose definition of embedded system is that they are devices used to control, monitor or assist the operation of equipment, machinery or plant. “Embedded” reflects the fact that they are an integral part of the system. An embedded system is a computer system with a dedicated function within a large mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded system controls many devices in common use today. It is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware.

Embedded system contains two main elements they are embedded system hardware, embedded system software.

Embedded system hardware:

As with any electronic system, an embedded system requires a hardware platform on which to run. The hardware will be based around a microprocessor or microcontroller. The embedded system hardware will also contain other elements including memory, input output(I/O) interfaces as well as the user interface, and the display. When using an embedded system there is a choice between the use of a microcontroller or a microprocessor.

- **Microcontroller based system:** A microcontroller is essentially a CPU, central processor unit, or processor with integrated memory or peripheral devices. As fewer external components are needed, embedded system using microcontroller tends to be more widely used.
- **Microprocessor based system:** Microprocessors contains a CPU but use external chips for memory and peripheral interfaces. As they require more devices on board, but they allow more expansion selection of exact peripherals, etc.

Whatever type of processor is used in the embedded system, it may be a very general-purpose type of one of the many highly specialized processors intended for a particular application. In some case custom designed chip may be viable for a particular application if quantities are sufficiently high. One common example of standard class of dedicated processor is digital signal processors. This type of processor is used for processing audio and image file particular. Processing is required very quick as they may be used in application such as mobile phones and the like

Embedded system software:

The embedded system software is written to perform a particular function. It is typically written in a high-level format and then compiled down to provide code that can be lodged with a non-volatile memory with the hardware. One of the key elements of an any embedded system is the software that is used to run the microcontroller. There is a variety of ways that this can be written

- **Machine code:** Machine code is the most basic code that is used for the processor unit. The code is normally in the hex code and provides the basic instructions for each operation of the processor. This form of code is rarely used for embedded system these days.
- **Programming Language:** Writing machine coded is very laborious and time consuming. It is difficult to understand and debug. To overcome this, high level programming language are often used. Language including C, C++, etc. are commonly used.

1.3 CHARACTERISTIC OF EMBEDDED SYSTEM

- Speed (bytes/sec): Should be high speed
- Power (watts): Low power dissipation
- Size and weight: As far as possible small in size and low weight
- Accuracy (%error): Must be very accurate
- Adaptability: High adaptability and accessibility
- Reliability: Must be reliable over a long period of time

1.4 APPLICATIONS OF EMBEDDED SYSTEMS

We are living in the Embedded World. You are surrounded with many embedded products and your daily life largely depends on the proper functioning of these gadgets. Television, Radio, CD player of your living room, Washing Machine or Microwave Oven in your kitchen, Card readers, Access Controllers, Palm devices of your work space enable you to do many of your tasks very effectively. Apart from all these, many controllers embedded in your car take care of car operations between the bumpers and most of the times you tend to ignore all these controllers.

- **Robotics:** Industrial robots, machine tools, Robocop soccer robots.
- **Automotive:** cars, trucks, trains.
- **Aviation:** airplanes, helicopters.
- Home and Building Automation.
- **Aerospace:** rockets, satellites.
- **Energy Systems:** windmills, nuclear plants.
- **Medical Systems:** prostheses, revalidation machine.

1.5 MICROCONTROLLER VERSUS MICROPROCESSOR

The microprocessor is meant the general-purpose Microprocessors such as Intel's X86 family (8086, 80286, 80386, 80486, and the Pentium) or Motorola's 680X0 family (68000, 68010, 68020, 68030, 68040, etc). These microprocessors contain no RAM, no ROM, and no I/O ports on the chip itself. For this reason, they are commonly referred to as general-purpose Microprocessors.

A system designer using a general-purpose microprocessor such as the Pentium or the 68040 must add RAM, ROM, I/O ports, and timers externally to make them functional. Although the addition of external RAM, ROM, and I/O ports makes these systems bulkier and much more expensive, they have the advantage of versatility such that the designer can decide on the amount of RAM, ROM and I/O ports needed to fit the task at hand. This is not the case with Microcontrollers.

A Microcontroller has a CPU (a microprocessor) in addition to a fixed amount of RAM, ROM, I/O ports, and a timer all on a single chip. In other words, the processor, the RAM, ROM, I/O ports and the timer are all embedded together on one chip; therefore, the designer cannot add any external memory, I/O ports, or timer to it. The fixed amount of on chip ROM, RAM, and number of I/O ports in Microcontrollers makes them ideal for many applications in which cost and space are critical.

In many applications, for example a TV remote control, there is no need for the computing power of a 486 or even an 8086 microprocessor. These applications most often require some I/O operations to read signals and turn on and off certain bits.

1.6 MICROCONTROLLERS FOR EMBEDDED SYSTEMS

In the Literature discussing microprocessors, we often see the term Embedded System. Microprocessors and Microcontrollers are widely used in embedded system products. An embedded system product uses a microprocessor (or Microcontroller) to do one task only. A printer is an example of embedded system since the processor inside it performs one task only; namely getting the data and printing it. Contrast this with a Pentium based PC. A PC can be used for any number of applications such as word processor, print-server, bank teller terminal, Video game, network server, or Internet terminal. Software for a variety of applications can be loaded and run. Of course, the reason a pc can perform myriad tasks is that it has RAM memory and an operating system that loads the application software into RAM memory and lets the CPU run it.

In this robot as the fire sensor senses the fire, it senses the signal to microcontroller. In an Embedded system, there is only one application software that is typically burned into ROM. An x86 PC contains or is connected to various embedded products such as keyboard, printer, modem, disk controller, sound card, CD-ROM drives, mouse, and so on. Each one of these peripherals has a Microcontroller inside it that performs only one task.

CHAPTER 2

2.INTRODUCTION TO PROJECT

2.1 INTRODUCTION

The project is designed to operate electrical loads using a TV remote. The remote transmits coded infrared data which is then received by a sensor interfaced to the control unit. The system operates electrical loads depending on the data transmitted from the TV remote. Operating conventional wall switches is difficult for elderly or physically handicapped people. This proposed system solves the problem by integrating house hold appliances to a control unit that can be operated by a TV remote. NEC based coded data sent from the TV remote is received by an IR receiver interfaced to the microcontroller of 8052 family. The program on the microcontroller refers to the NEC code to generate respective output based on the input data to operate a set of relays through a relay driver IC. The loads are interfaced to the control unit through the relays. The system can be used in existing domestic area for either operating the loads through conventional switches or with the tv remote. The project can be enhanced by using radio frequency technology where the operational range shall be independent of line of sight distance as often encountered with IR type of remote control.

2.2 BLOCK DIAGRAM

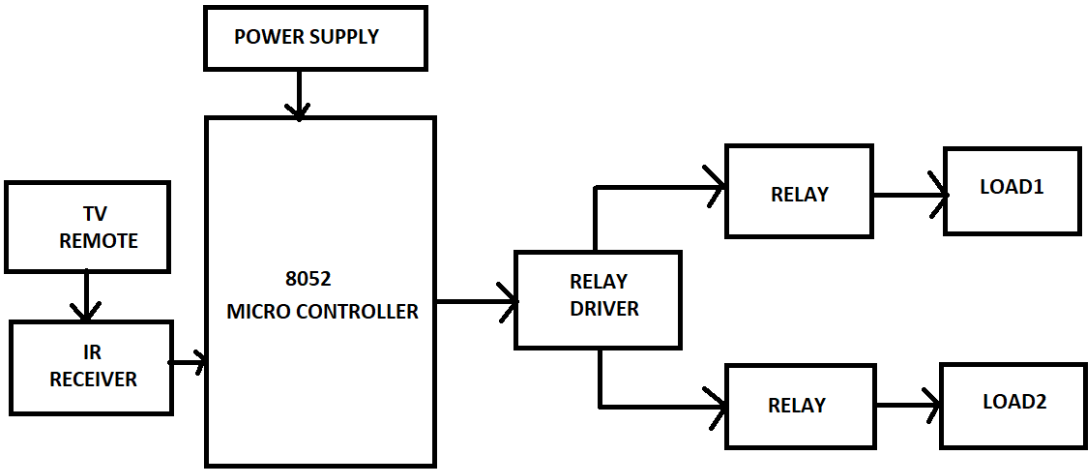


Fig 2.1 Block diagram

CHAPTER 3

3.MICROCONTROLLER

3.1 AT89C52 MICROCONTROLLER

Microcontroller is a general-purpose device, which integrates a number of the components of a microprocessor system on to single chip. It has inbuilt CPU, memory and peripherals to make it as a mini computer. A microcontroller combines on to the same microchip they are CPU core, Memory (both ROM and RAM), Some parallel digital I/o. Microcontrollers are small in size, inexpensive, consumes less power.

Micro controller is a standalone unit, which can perform functions on its own without any requirement for additional hardware like I/O ports and external memory. The heart of the microcontroller is the CPU core. In the past, this has traditionally been based on an 8-bit microprocessor unit. For example, Motorola uses a basic 6800 microprocessor core in their 6805/6808 microcontroller devices.

AT89C51 is the 40 pins, 8-bit Microcontroller. It is the flash type reprogrammable memory. Advantage of this flash memory is we can erase the program with in few minutes. It has 4kb on chip ROM and 128 bytes internal RAM and 32 I/O pin as arranged as port 0 to port 3 each has 8-bit bin. Port 0 contain 8 data line(D0-D7) as well as low order address line (AO-A7).

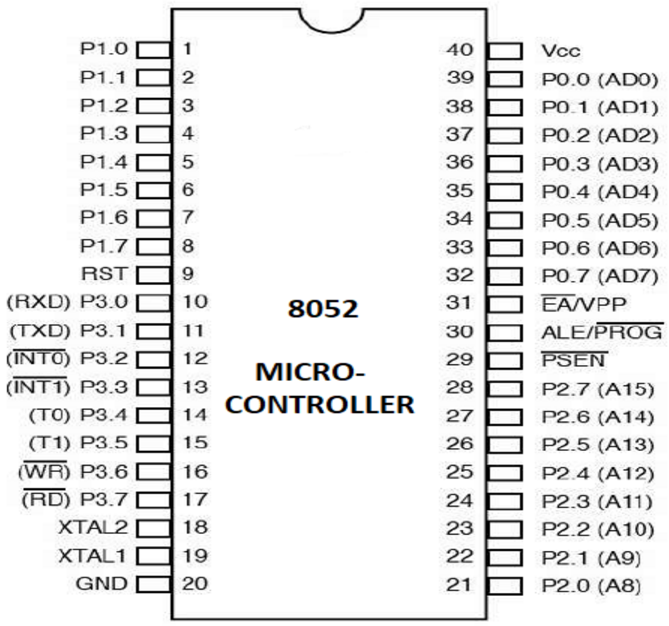


Fig 3.1 Pin diagram of 8052 microcontroller

Port 2 contain higher order address line (A8-A15). Port 3 contains special purpose register such as serial input receiver register SBUF, interrupt INT0, INT1 and timers T_0, T_1 many of the pins have multi functions which can be used as general purpose I/O pins (or) Special purpose function can be decided by the programmer itself.

3.2 FEATURES

- 4KB bytes on-chip program memory (ROM).
- 128 bytes on-chip data memory (RAM).
- Two 16-Bit Timer/Counters.
- 8-bit bidirectional data bus.
- 6-bit unidirectional address bus.

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel is high density non-volatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout.

The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional non-volatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel 89C52 is a powerful microcomputer which provides a highly flexible and cost-effective solution to many embedded control applications.

3.3 ARCHITECTURE OF 8052 MICROCONTROLLER

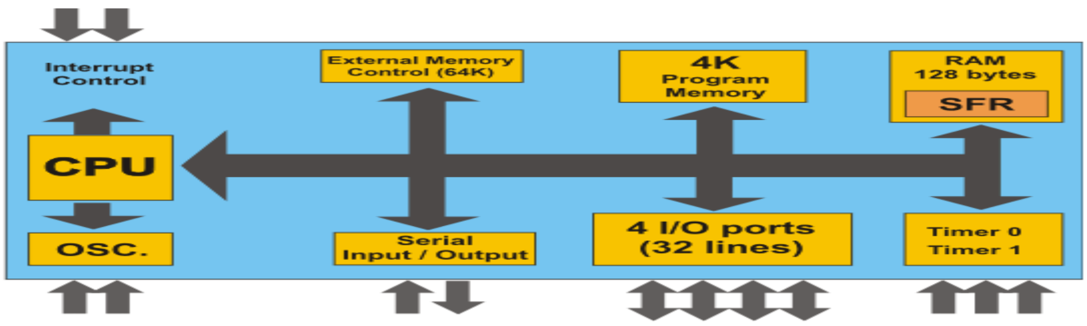


Fig 3.2 Architecture of 8052 microcontroller

The 8052 consists of:

- Eight-bit CPU with registers A (the accumulator) and B
- Program counter (PC)
- Data pointer (DPTR)
- Flags and the Program Status Word (PSW)
- Eight-bit stack pointer (SP)
- Internal ROM or EPROM or FLASH ROM
- Internal RAM of 256bytes (128bytes general purpose):
- Four register banks, each containing eight registers
- Two 16-bit timer / counter: T0 and T1
- Full duplex serial data receiver/transmitter; SBUF
- Interrupts
- Oscillator and clock circuits
- Thirty-two input/output pins arranged as four 8-bit ports P0-P3

3.4 PIN OUT DIAGRAM OF 8052

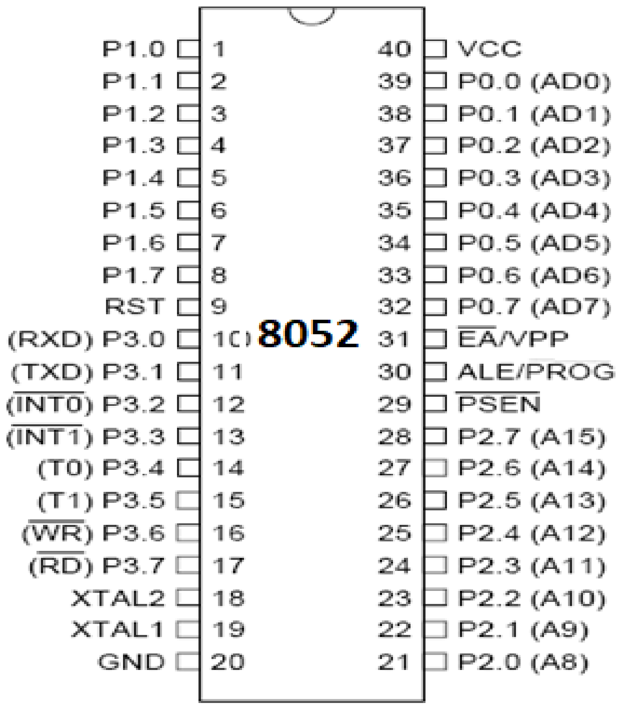


Fig. 3.3 Pin diagram of 8052 microcontroller

3.4.1 CPU Registers:

The 8052 contain 34 general-purpose, or working, registers. Two of these, registers A and B hold results of many instructions, particularly for arithmetical and logical operations. The other 32 are arranged as part of internal RAM in four banks, Bank0-Bank3, of eight registers each.

3.4.2 Program Counter (PC):

The 8052 contain two 16-bit registers: the programs counter (PC) and the data pointer (DPTR). Each is used to hold the address of a word in memory.

Program instruction bytes are fetched from locations in memory that are addressed by the PC. Program ROM may be on the chip at addresses 000h to FFFh, external to the chip for address that exceed FFFh, or totally external for all address from 0000h to FFFFh. The PC is automatically incremented after every instruction byte is fetched and may also be altered by certain instructions. The PC is the only register that does not have an internal address.

The Program Counter (PC) is a 2-byte address that tells the 8052 where the next instruction to execute is found in memory. When the 8052 is initialized, PC always starts at 0000h and is incremented each time an instruction is executed. It is important to note that PC isn’t always incremented by one. Since some instructions require 2 or 3 bytes the PC will be incremented by 2 or 3 in these cases.

3.4.3 Data Pointer (DPTR):

The DPTR register is made up of two 8-bit registers, named DPH and DPL, which are used to furnish memory addresses for internal and external code access and external data access. The DPTR is under the control of program instructions name, DPH and DPL. DPTR does not have a single internal address; DPH and DPL are each assigned an address.

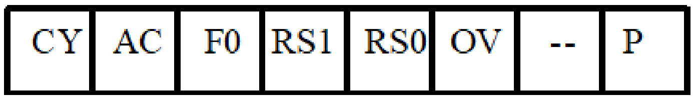
The Data Pointer (DPTR) is the 8052's only user-accessible 16-bit (2-byte) register. It is used by a number of commands that allow the 8052 to access external memory and internal memory. It is often used to store 2-byte values that have nothing to do with memory locations.

3.4.4 Program Status Word (PSW):

Flags are 1-bit registers provided to store the results of certain program instructions. Other instructions can test the condition of the flags and make decisions based on the flag states. In order that the flags may be conveniently addressed, they are grouped inside the program status word (PSW) and the power control (PCON) registers.

The 8052 have four math flags that respond automatically to the outcomes of math operations and three general-purpose user flags that can be set to 1 or cleared to 0 by the programmer as desired. The math flags include Carry (CY), Auxiliary Carry (AC), Overflow (OV), and Parity (P). User flag is named F0; this general-purpose flag that may be used by the programmer to record some event in the program.

Register bank selection may be done by the use of RS0 and RS1 Note that all of the flags can be set and cleared by the programmer at will. The math flags, however, are also affected by math operations.



CY	AC	F0	RS1	RS0	OV	--	P

Fig 3.4 Register format of Program Status Word (PSW)

CY	PS W.7	Carry flag
AC	PS W.6	Auxiliary carry flag
F0	PS W.5	Flag 0 (User Flag)
RS1	PS W.4	Register bank selector bit 1
RS0	PS W.3	Register bank selector bit 0

OV	PS W.2	Overflow flag
--	PS W.1	Reserved for future use
P	PS W.0	Parity flag

Table 3.1 Pins description of PSW

RS1	**RS0**	**Register Bank**	**Address**
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

Table 3.2 Description of Register bank selection bits

The program status word is shown above. The PSW contains the math flags, user program flag F0, and the register select bits RS0, RS1 that identify which of the four general-purpose register banks is currently in use by the program.

3.4.5 Stack and Stack Pointer:

The stack refers to an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly. The 8-bit Stack Pointer (SP) register is used by the 8052 to hold an internal RAM address that is called the top of the stack. The address held in the SP register is the location in internal RAM where the last byte of data was stored by a stack operation.

When data is to be placed on the stack, the SP increments before storing data on the stack so that the stack grows up as data is stored. As data is retrieved from the stack, the byte is read from the stack, and then the SP decrements to point to the next available byte of stored data.

Operation of the stack and the SP is shown above. The SP is set to 07h when the 8052 is reset and can be changed to any internal RAM address by the programmer, I.e., default address location of stack pointer is 07h. Using a data move command, we can change the stack pointer address.

The stack is limited in height to the size of the internal RAM. The stack has the potential (if the programmer is not careful to limit its growth) to over write valuable data in the register banks, bit-addressable RAM, and general purpose (scratchpad) RAM areas. The programmer is responsible for making sure the stack does not grow beyond predefined bounds.

The stack is normally placed high in internal RAM, by an appropriate choice of the number placed in the SP register, to avoid conflict with the register, bit and scratch-pad internal RAM areas.

The Stack Pointer, like all registers except DPTR and PC, may hold an 8-bit (1-byte) value. The Stack Pointer is used to indicate where the next value to be removed from the stack should be taken from.

When you push a value onto the stack, the 8052 first increments the value of SP and then stores the value at the resulting memory location. When you pop a value off the stack, the 8052 returns the value from the memory location indicated by SP, and then decrements the value of SP. That is last in first out method (LIFO). SP is modified directly by the 8052 by six instructions: PUSH, POP, ACALL, LCALL, RET, and RETI. It is also used intrinsically whenever an interrupt is triggered. (More on interrupts later).

3.4.6 Internal Memory:

A functioning computer must have memory for program codes, commonly in ROM, and RAM memory for variable data that can be altered as the program runs. The 8052 has internal RAM and ROM memory for these functions. Additional memory can be added externally using suitable circuits.

Unlike Microcontrollers with Von Neumann architectures, which can use a single memory address for either program code or data, but not for both, the 8052 has a Harvard architecture, which uses the same address, in different memories, for code and data. Internal circuitry accesses the correct memory based on the nature of the operation in progress. Harvard architecture uses separate buses to fetch the data and the address.

3.4.7 Internal ROM:

The 8052 is organized so that data memory and program code memory can be in two entirely different physical memory entities. Each has the same address range. The structure of the internal RAM will be discussed later. Generally, 8052 microcontroller is available with 8Kilo Bytes internal ROM.

A corresponding block of internal program code, contained in an internal ROM, occupies code address space 000h to FFFh. The Program Counter is ordinarily used to address program code bytes from address 0000h to FFFFh. Program addresses higher than 0FFFh, which exceed the internal ROM capacity, will cause the 8052 to automatically fetch code bytes from external program memory.

Code bytes can also be fetched exclusively from an external memory, address 0000h to FFFFh, by connecting the external access pin (EA pin no 31 on the DIP) to ground. The PC does not care where the code is, the circuit designer decides whether the code is found totally in internal ROM, totally in external ROM, or in a combination of internal and external ROM.

8052 microcontrollers have 16-bit address bus and 8-bit data bus, with 16-bit address bus we can address maximum of 64Kilobytes of external memory that is from 0000h to FFFF.

8052 is available with 8kilobytes of internal ROM its derivatives 8752, 8952 are available with EPROM, FLASH ROM respectively with 8kilobytes capacity.

3.4.8 Internal RAM 256 Bytes:

As mentioned at the beginning of this chapter, the 8052 includes a certain amount of on-chip memory. On-chip memory is really one of two types: Internal RAM and Special Function Register (SFR) memory. The layout of the 8052's internal RAM is presented.

As is illustrated in this map, the 8052 has a bank of 128 bytes of Internal RAM. This Internal RAM is found on-chip on the 8052 so it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying its contents. Internal RAM is volatile, so when the 8052 is switched off this memory is cleared.

The 128 bytes of internal ram is subdivided as shown on the memory map. The first 8 bytes (00h - 07h) are "register bank 0". By manipulating certain SFRs, a program may choose to use register banks 1, 2, or 3. These alternative register banks are located in internal RAM in addresses 08h through 1Fh. So, the registers are part of internal RAM.

Bit Memory also lives and is part of internal RAM. Bit memory actually resides in internal RAM, from addresses 20h through 2Fh. It can be bit addressed from 00h to 7fh (totally 128 bits)

The 80 bytes remaining of Internal RAM, from addresses 30h through 7Fh, may be used by user variables that need to be accessed frequently or at high-speed. This area is also utilized by the microcontroller as a storage area for the operating stack.

3.5 I/O PORTS

3.5.1 Thirty - two Input / Output pins:

All four ports in the 8052 are bi-directional each contains a latch, an output driver and input buffer. The output drivers of port0 and 2, and the input buffers of port 0 are used in access to external memory. In this application port 0 is used as a lower byte of the external memory address multiplexed with data bus and port 2 is used as a higher byte of the external memory address when address is sixteen bits wide. Otherwise, it can be used as general purpose I/O.

P0(PORT0):

The P0 (zero) port is characterized by two functions −When the external memory is used then the lower address byte (addresses A0A7) is applied on it, else all bits of this port are configured as input/output. When P0 port is configured as an output then other ports consisting of pins with built-in pull-up resistor connected by its end to 5V power supply, the pins of this port have this resistor left out.

P1(PORT1):

P1 is a true I/O port as it doesn’t have any alternative functions as in P0, but this port can be configured as general I/O only. It has a built-in pull-up resistor and is completely compatible with TTL circuits.

P2(PORT2):

P2 is similar to P0 when the external memory is used. Pins of this port occupy addresses intended for the external memory chip. This port can be used for higher address byte with addresses A8-A15. When no memory is added then this port can be used as a general input/output port similar to Port 1.

P3(PORT3):

In this port, functions are similar to other ports except that the logic 1 must be applied to appropriate bit of the P3 register.

3.5.2 Interrupts:

An interrupt is a special feature, which allows the 8052 to provide the illusion of "multi-tasking," although in reality the 8052 is only doing one thing at a time. The word "interrupt" can often be substituted with the word "event."

An interrupt is triggered whenever a corresponding event occurs. When the event occurs, the 8052 temporarily puts "on hold" the normal execution of the program and executes a special section of code referred to as an interrupt handler. The interrupt handler performs whatever special functions are required to handle the event and then returns control to the 8052 at which point program execution continues as if it had never been interrupted.

A computer program has only two ways to determine the conditions that exist in internal external circuits. One method uses software instructions that jump to subroutines on the states of flags and port pins. The second method responds to hardware signals, called interrupts that force the program to call a subroutine.

Software techniques use up processor time that could be devoted to other tasks; interrupts take processor time only when action by the program is needed. Most applications of microcontrollers involve responding to events quickly enough to control the environment that generates the events (generically termed real-time programming). Interrupts are often the only way in which real-time programming can be done successfully. Interrupts may be generated by internal chip operations or provided by external sources. Any interrupt can cause the 8052 to perform a hardware call to an interrupt, handling subroutine that is located at a predetermined (by the 8052 designers) absolute address in program memory.

Five interrupts are provided in the 8052. Three of these are generated automatically by internal operations: Timer flag 0, Timer flag 1, and the serial port interrupt (RI or TI). Two interrupts are triggered by external signals provided by circuitry that is connected to pins INT0 and INT1 (port pins P3.2 and P3.3).

After the Interrupt has been handled by the interrupt subroutine, which is placed by the programmer at the interrupt location in program memory, the interrupt program must resume operation at the instruction where the interrupt took place. Program resumption is done by storing the interrupted PC address on the stack in RAM before changing the PC to the interrupt address in ROM. The PC address will be restored from the stack after an RETI instruction is executed at the end of the interrupt subroutine.

3.5.3 Serial interface:

Computers must be able to communicate with other computers in modern multiprocessor distributed systems. One cost-effective way to communication is to send and receive data bits serially. The 8052 has a serial data communication circuit that uses register SBUF to hold data. Register SCON controls data communication, register PCON controls data rates, and pins RXD (P3.0) and TXD (P3.1) connect to the serial data network.

SBUF is physically two registers. One is written only and is used to hold data to be transmitted out of the 8052 via TXD. The other is read only and holds received data from external sources via RXD.There are four programmable modes for serial data communication that are chosen by setting the SMX bits in SCON. Baud rates are determined by the mode chose. We will discuss about serial port later.

3.5.4 Two 16-bit TIMER / COUNTER:

Many microcontroller applications require the counting of external events, such as the frequency of a pulse train, or the generation or precise internal time delays between computer actions. Both of these tasks can be accomplished using software techniques, but software loops for counting or timing keep the processor occupied so that other, perhaps more important, functions are not done. To relieve the processor of this burden, two 16-bit up counters, named T0 and T1, are provided for the general use of the programmer. Each counter may be programmed to count internal clock pulses, acting as a timer, or programmed to count external pulses as a counter.

The counters are divided into two-8-bit registers called the timer low (TL0, TL1) and high (TH0, TH1) bytes. All counter action is controlled by bit states in the timer mode control register (TMOD), the timer/counter control register (TCON), and certain program instructions.

3.5.5 Oscillator and Clock circuits:

The heart of the 8052 is the circuitry that generates the clock pulse by which all internal operations are synchronized. Pins XTAL1 and XTAL2 are provided for connecting a resonant network to form an oscillator. Typically, a quartz crystal and capacitors are employed, as shown in Figure 3.7 the crystal frequency is the basic internal clock frequency of the microcontroller. The manufactures make available 8052 designs that can run at specified maximum and minimum frequencies, typically 1 megahertz to 24 megahertz. Minimum frequencies imply that some internal memories are dynamic and must always operate above a minimum frequency or data will be lost.

Serial data communication needs often state the frequency of the oscillator because of the requirement that internal counters must divide the basic clock frequency is not divisible without a remainder, and then the resulting communication frequency is not standard.

Ceramic resonators may be used as low-cost alternative to crystal resonators. However, decreases in frequency stability data accuracy make the ceramic resonator a poor choice if high-speed serial data communication with the systems, or critical timing, is to be done.The oscillator formed by the crystal, capacitors, and an on-chip inverter microcontroller, called the pulse, P, time. The smallest interval of time to accomplish any simple instruction, or part of a complex instruction, however, is the machine cycle. The machine cycle is itself made up of six states. A state is the basic time interval for discrete operations of the microcontroller such as fetching an opcode byte, decoding an opcode, executing an opcode, or writing a data byte. Two oscillator pulses define each state.

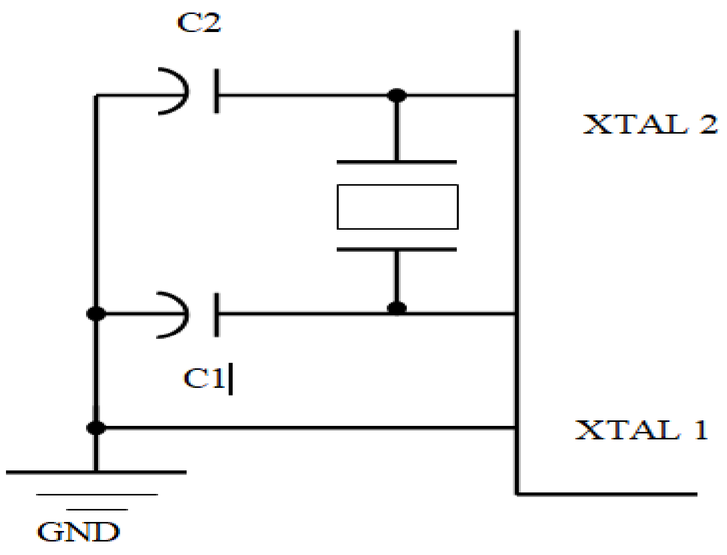


Fig 3.5 Oscillator circuit

Program instructions may require one, two, or four machine cycles to the executed, depending on the type of instruction. Instructions are fetched and executed by the microcontroller automatically, beginning with the instruction located by the microcontroller automatically; beginning with the instruction located at ROM memory address 0000h at the time the microcontroller is first reset.

To calculate the time any particular instruction will take to be executed, find the number of cycles, C, the time to execute that instruction is then found by multiplying C by 12 and dividing the product by the crystal frequency:

For example, if the crystal frequency is 16 megahertz, then the time to execute an ADD A, R1 one-cycle instructions is .75 microseconds. A 12-megahertz crystal yields the convenient time of 1 microsecond per cycle.

An 11.0592-megahertz crystal, although seemingly an odd value, yields a cycle frequency of 921.6 kilohertz, which can be divided evenly by the standard communication baud rates of 19200, 9600, 4800, 2400, 1200, and 300 hertz.

There are two ALE pulse per machine cycle. The ALE pulse, which is primarily used as a timing pulse for external memory access, indicates when every instruction byte is fetched. Two bytes of a single instruction may thus be fetched, and executed, in one machine cycle. Single byte instructions are nor executed in a half cycle, however, Single-byte instructions "throw-away" the second byte (which is the first byte of the next instruction).

3.6 APPLICATIONS

- Light sensing and controlling devices.
- Temperature sensing and controlling devices.
- Fire detections and safety devices.
- Automobile applications.
- Process control devices.
- Defence applications.
- Voltmeter applications.
- Remote Control Home Appliances.
- Voice Controlled Home Appliances

CHAPTER 4

4.HARDWARE COMPONENTS

4.1 POWER SUPPLY

- LED
- Capacitor
- Transformer
- Bridge Rectifier
- Regulator

4.1.1 LED:

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

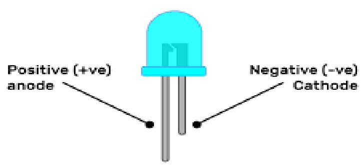


Fig 4.1 LED

The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device.

4.1.2 Capacitor:

A capacitor is a passive electronic component that stores energy in the form of an electrostatic field. In its simplest form, a capacitor consists of two conducting plates separated by an insulating material called the dielectric. The capacitance is directly proportional to the surface areas of the plates, and is inversely proportional to the separation between the plates. Capacitance also depends on the dielectric constant of the substance separating the plates.

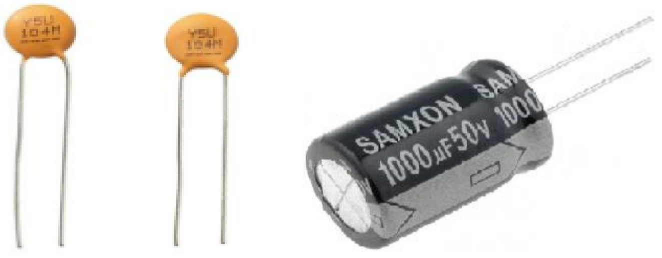


Fig 4.2 Capacitor

4.1.3 Transformer:

A transformer uses the principles of electromagnetism to change one A.C. voltage level to another. Faraday's work in the 19th century showed that a changing current in a conductor (e.g., a transformer primary winding) sets up a changing magnetic field around the conductor. If another conductor (secondary winding) is placed within this changing magnetic field a voltage will be induced into that winding.

A transformer is a static electrical device that transfers electrical energy between two or more circuits. A varying current in one coil of the transformer produces a varying magnetic flux, which, in turn, induces a varying electromotive force across a second coil wound around the same core. Electrical energy can be transferred between the two coils, without a metallic connection between the two circuits. Transformer is a static device used to convert the voltage from one level to another level without change its frequency.

Fig 4.3 Step-down Transformer

There are two types of transformers.

1. Step-up transformer
2. Step-down transformer

Step-up transformer converts low voltage level into high voltage level without change its frequency.

Step-down transformer converts high voltage level into low voltage level without change its frequency.

Here in our project, we are using the step-down transformer which is used to convert the 230v AC to the 12V DC.

4.1.4 Bridge Rectifier:

A diode bridge is an arrangement of four diodes in a bridge circuit configuration that provides the same polarity of output for either polarity of input. It is used in most common application, for conversion of an alternating current input into a direct current output, it is known as bridge rectifier. A bridge rectifier provides full-wave rectification from a two-wire AC input, resulting in lower cost and weight as compared to a rectifier with a 3-wire input from a transformer with a center-tapped secondary winding.

Fig 4.4 Bridge Rectifier

4.1.5 Regulator:

Voltage sources in a circuit may have fluctuations resulting in not providing fixed voltage outputs. The internal current-limiting and thermal-shutdown features of these regulators essentially make them immune to overload. A voltage regulator IC maintains the output voltage at a constant value. 7805 IC, a member of 78xx series of fixed linear voltage regulators used to maintain such fluctuations, is a popular voltage regulator integrated circuit (IC). The XX in 78XX indicates the output voltage it provides. 7805 IC provides +5 volts regulated power supply with provisions to add a heat sink. The regulator we are using in the figure consists of three pins as shown in the below

- **Pin1:** It is used for input pin.
- **Pin2:** This is ground pin regulator.
- **Pin3:** It is used for output pin. Through this pin we get the output.

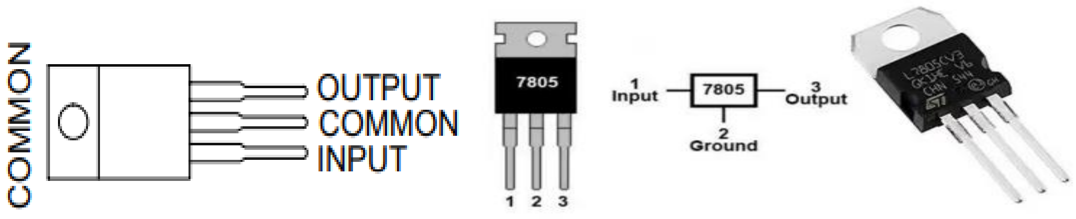


FIG 4.5 7805 PINOUT DIAGRAM

4.2 IR RECEIVER

In this project we are using TSOP1738 receiver and it is standard IR remote control receiver series, supporting all major transmission codes. It uses silicon-based technology, which works at the microlevel and very sensitive and efficient to its functions. In summary, TSOP may be smaller in size but its usage with microcontroller makes it smart and secure.

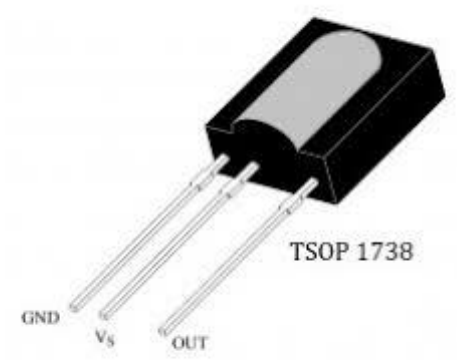


Fig 4.6 IR Receiver

4.3 RELAY DRIVER

In a low power circuit or an output from a Microprocessor is very low. It is sufficient for a LED to glow but to drive a high load you will need a Relay, and to give proper voltage or current to a relay you will need a relay driver. A relay driver circuit is a circuit which can drive, or operate, a relay so that it can function appropriately in a circuit. The driven relay can then operate as a switch in the circuit which can open or close, according to the needs of the circuit and its operation.

Fig 4.7 Relay driver

4.4 RELAY

Relays are switching that open and close circuits electromechanically or electronically. Relays control one electrical circuit by opening and closing contacts in another circuit. As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized. When a relay contact is Normally Closed (NC), there is a closed contact when the relay is not energized. In either case, applying electrical current to the contacts will change their state. In our project we are using relay and relay drive on a single chip.

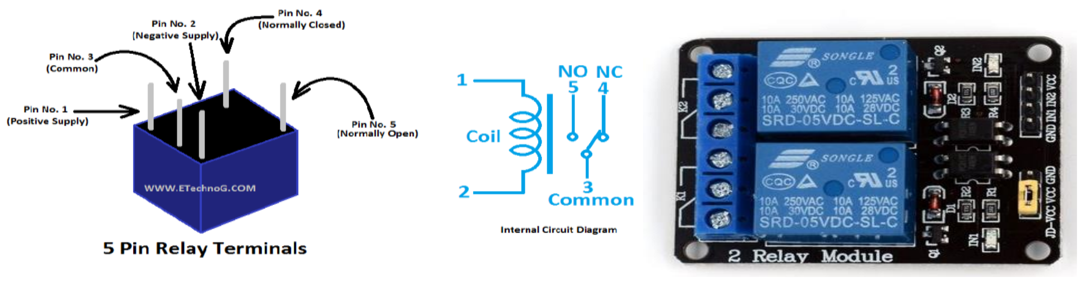


Fig 4.8 Relay and 2-relay module

4.5 TV REMOTE

A remote control (also known as a remote or clicker) is an electronic device used to operate another device from a distance, usually wirelessly. In consumer electronics, a remote control can be used to operate devices such as a television set, DVD player or other home appliance.

Fig 4.9 TV Remote

4.6 LOAD

An electrical load is an electrical component or portion of a circuit that consumes (active) electric power, such as electrical appliances and lights inside the home. The term may also refer to the power consumed by a circuit. This is opposed to a power source, such as a battery or generator, which produces power.

Fig 4.10 Different types of loads

CHAPTER 5

5.DESIGN IMPLEMENTATION

5.1 CIRCUIT DIAGRAM

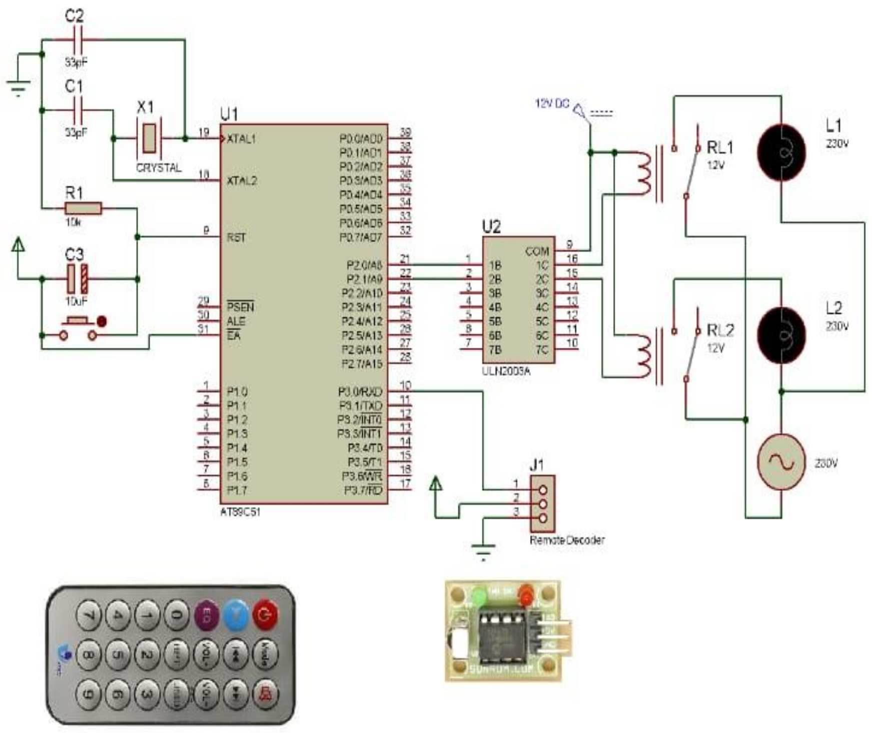


Fig 5.1 Circuit Diagram

5.2 WORKING

The power supply that comes to our house is generally 230v AC, but our requirement of operation of our circuit is 5v, so by using transformer we convert 230v AC to 12v AC, and after converting to 12v the output of the transformer (12v AC) is given to the onboard bridge rectifier. The bridge rectifier is used to convert AC input to the DC output here we are converting 12v AC to 12v DC, But the microcontroller operates at 5v DC so by using the voltage regulator we are converting 12v DC to the 5v AC

Generally, Tv remote transmits the data using infrared rays with the NEC protocol (Numerical Electromagnetic Code). The NEC protocol uses the pulse distance encoding for transmitting the data. By using NEC protocol, the tv remote transmits the 32-bit data, on the other side the signal is received by the receiver which is TSOP1738, and encode the signal. The output of receiver is given to the microcontroller.

Whenever the button is pressed the instructions in the microcontroller gets processed and from microcontrollers the connections are given to the 2-relay module in the relay module consists of the relay drivers, ULN, two relays. The relays are connected to operate the loads. Initially the loads will be in the off state, according to the button pressed in the TV remote the particular load will be activated, they are

- **1-** Load 1 will be ON.
- **2-** Load 1 will be OFF.
- **3-** Load 2 will be ON.
- **4-** Load 2 will be OFF.
- **9-** All loads will be ON.
- **Power button-** All loads will be OFF.

5.3 INTERFACING OF TSOP1738 WITH 8052

The TSOP1738 consists of the three pins they are gnd, vcc, data as shown in the below figure. The interface of the three pins are as follows

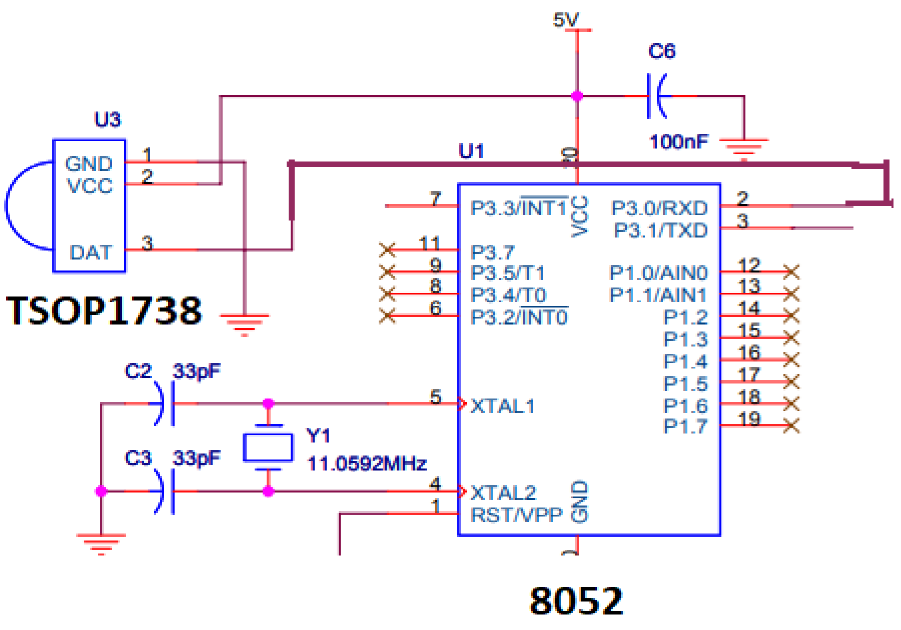


Fig 5.2 Interfacing of the TSOP1738 with 8052

- The DATA PIN is connected to the microcontroller port 3.0. Generally, the port 3.0 is the RXD pin it is nothing but the receiving pin that is receiving the data serially.
- Another pin of the TSOP1738 is the vcc pin. This vcc pin is connected to the 5v.
- The last pin is Gnd pin. The ground pin is connected to the ground on the on chip.

With this interface of the component the data which is being transmitted when we press the TV remote button is received at the receiver and processed to the micro controller.

5.4 INTERFACE THE 8052 WITH THE 2-RELAY MODULE

The 2-Relay module consists of the 4 input pins they are vcc, ground, in1 and in2. The interface of the input four pins and output is as shown in the below figure

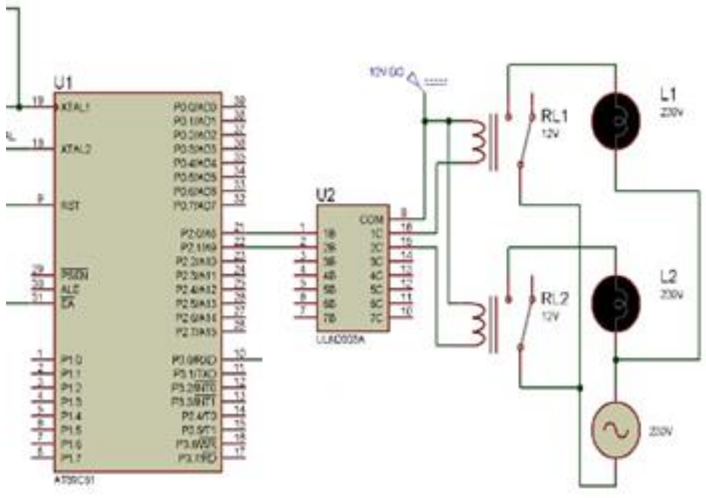


Fig 5.3 Interface the 8052 with 2-relay module

- The ground and vcc pin are connected to the on-chip ground and vcc
- The in1 and in2 are connected to the 20 and 21 of the port 2 that is nothing but the P2.0 and P2.1 respectively.

With the interface of the 8052 with the relay module we can control the loads. The loads can be on and off with respective to the instructions send by the microcontroller to the relay module

CHAPTER 6

6. SOFTWARE USED

6.1 KEIL MICROVISION

Keil Micro Vision software is an integrated development environment (IDE), which integrated a text editor to write programs, a compiler and it will convert your source code to hex files too. In the Keil software we will the program in the embedded c and used to convert the hex file.

In this section, we will learn in easy steps, how to write the code in the embedded c.

Step 1: Click on the Keil micro vision icon and open it and click on the project which is present in the menu bar and click the new micro vision project and name it.

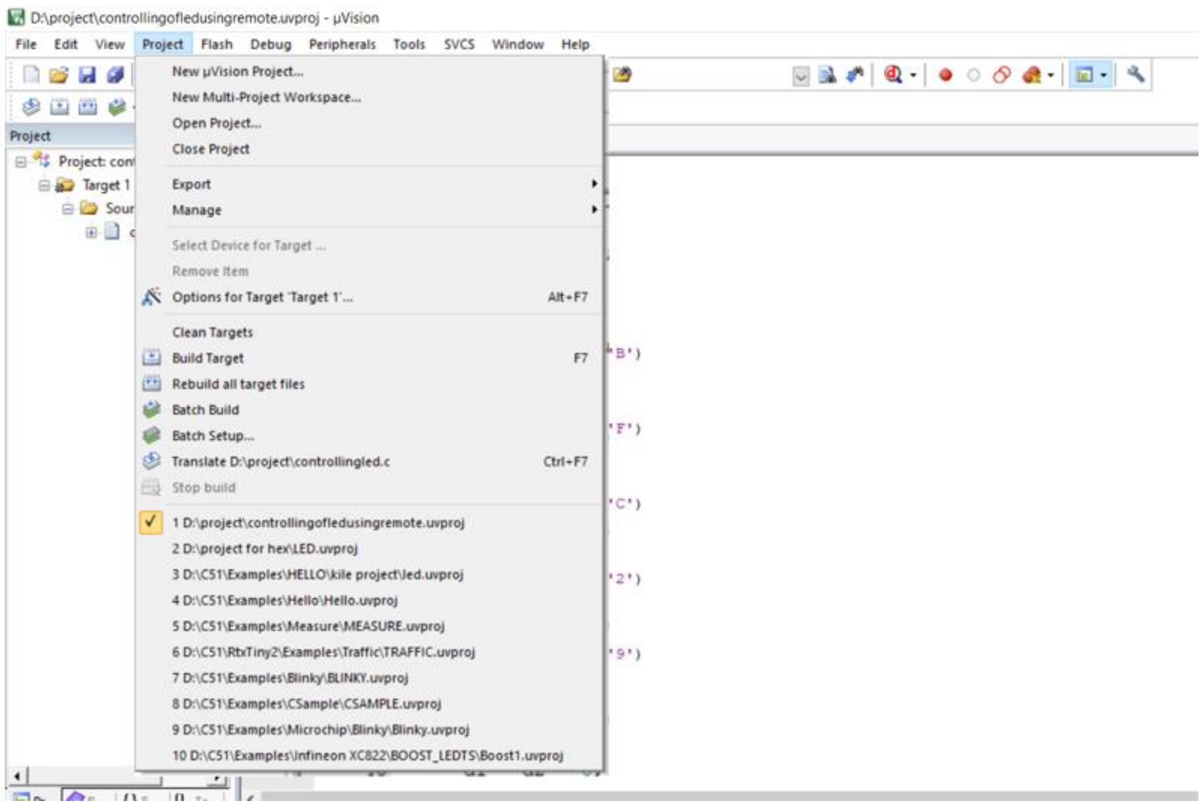
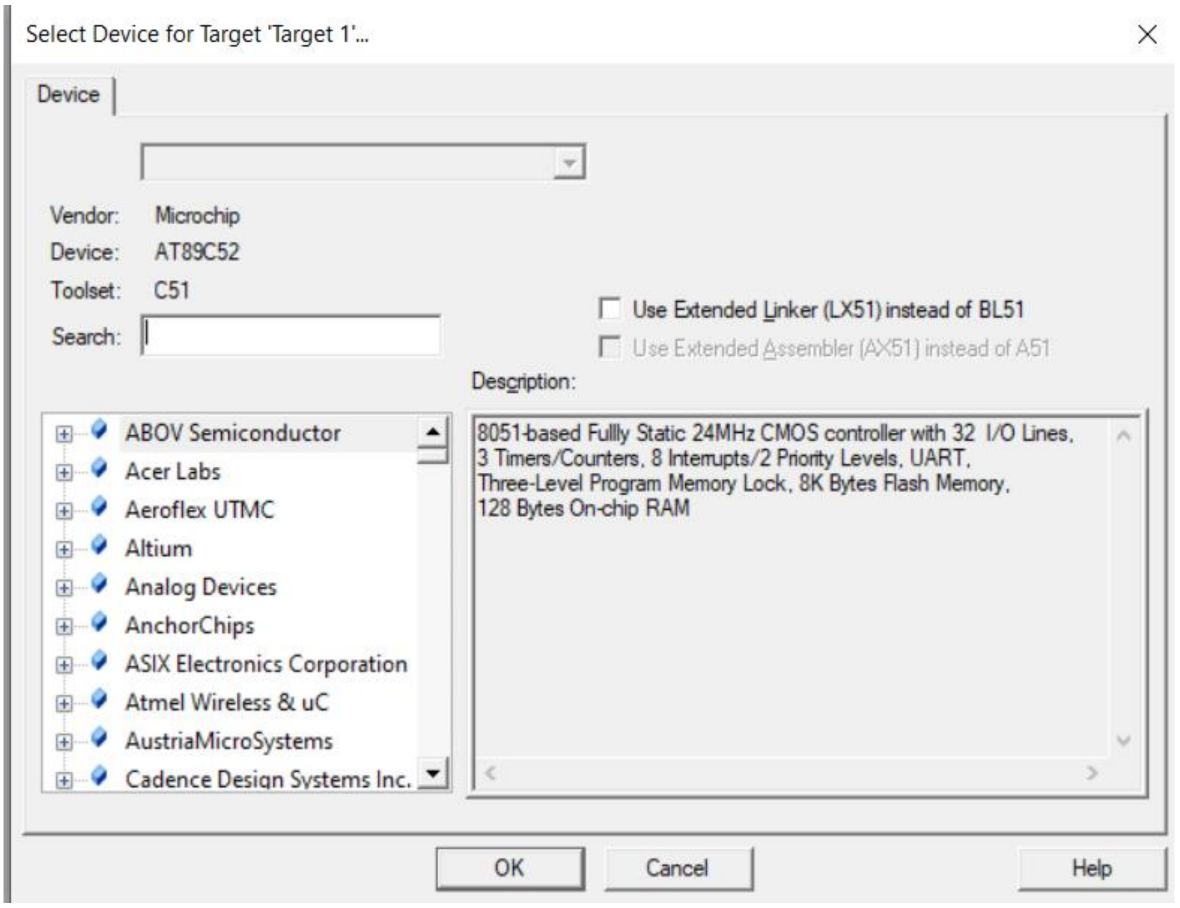

D:\project\controllingofledusingremote.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Project: con
Target 1
Sour
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target ...
Remove Item
Options for Target 'Target 1'... Alt+F7
Clean Targets
Build Target F7
Rebuild all target files
Batch Build
Batch Setup...
Translate D:\project\controllingled.c Ctrl+F7
Stop build
1 D:\project\controllingofledusingremote.uvproj
2 D:\project for hex\LED.uvproj
3 D:\C51\Examples\HELLO\kile project\led.uvproj
4 D:\C51\Examples\Hello\Hello.uvproj
5 D:\C51\Examples\Measure\MEASURE.uvproj
6 D:\C51\RtxTiny2\Examples\Traffic\TRAFFIC.uvproj
7 D:\C51\Examples\Blinky\BLINKY.uvproj
8 D:\C51\Examples\CSample\CSAMPLE.uvproj
9 D:\C51\Examples\Microchip\Blinky\Blinky.uvproj
10 D:\C51\Examples\Infineon XC822\BOOST_LEDTS\Boost1.uvproj

Fig 6.1 Project creation

Step 2: After name the project a window will appear on which we have to select the device for target. In that we have to search AT89C52 and click ok.



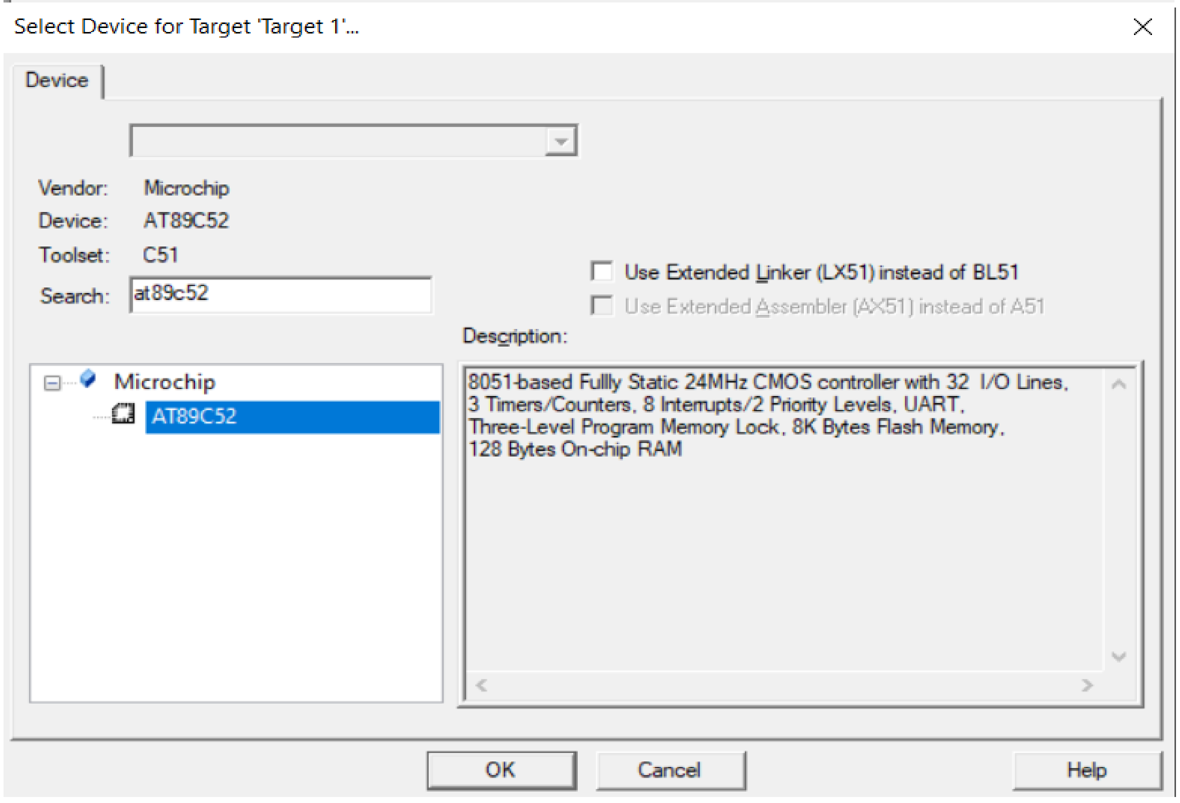
Vendor: Microchip
Device: AT89C52
Toolset: C51
Search:

☐ Use Extended Linker (LX51) instead of BL51
☐ Use Extended Assembler (AX51) instead of A51

Description:

ABOV Semiconductor
Acer Labs
Aeroflex UTMC
Altium
Analog Devices
AnchorChips
ASIX Electronics Corporation
Atmel Wireless & uC
AustriaMicroSystems
Cadence Design Systems Inc.

8051-based Fullly Static 24MHz CMOS controller with 32 I/O Lines, 3 Timers/Counters, 8 Interrupts/2 Priority Levels, UART, Three-Level Program Memory Lock, 8K Bytes Flash Memory, 128 Bytes On-chip RAM



Vendor: Microchip
Device: AT89C52
Toolset: C51
Search: at89c52

☐ Use Extended Linker (LX51) instead of BL51
☐ Use Extended Assembler (AX51) instead of A51

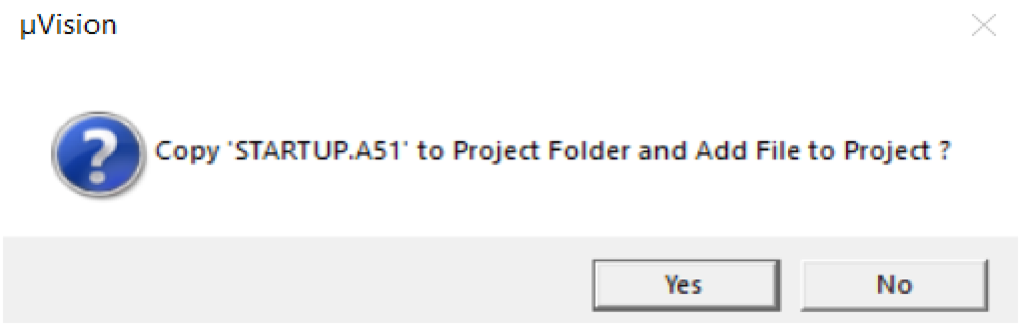
Description:

Microchip
AT89C52

8051-based Fullly Static 24MHz CMOS controller with 32 I/O Lines, 3 Timers/Counters, 8 Interrupts/2 Priority Levels, UART, Three-Level Program Memory Lock, 8K Bytes Flash Memory, 128 Bytes On-chip RAM

Fig 6.2 Select device for target

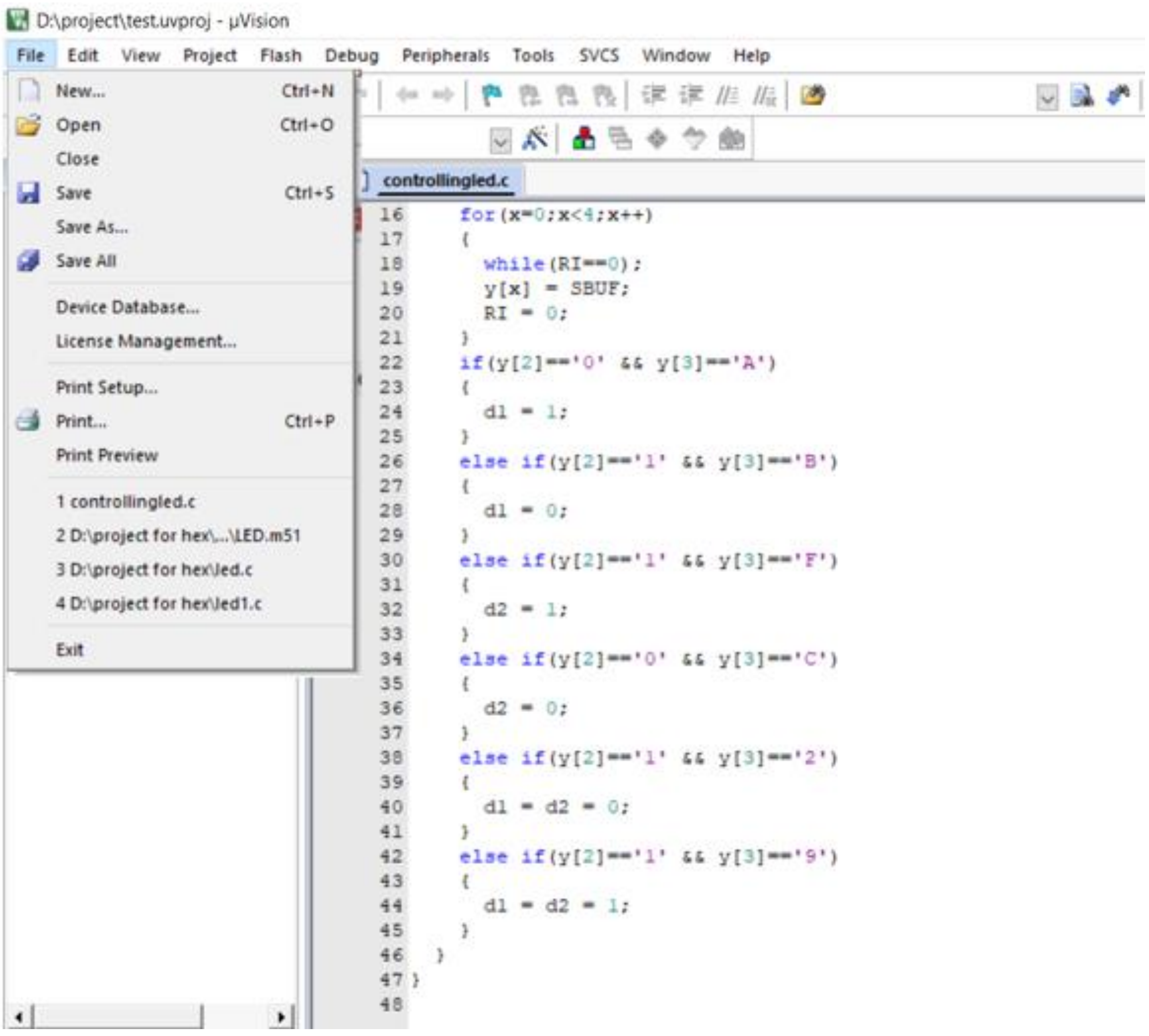
Step 3: so now your project is created and Message window will appear to add start-up file of your Device click on NO so it will be added to your project folder.



Copy 'STARTUP.A51' to Project Folder and Add File to Project ?

Fig 6.3 To add the file

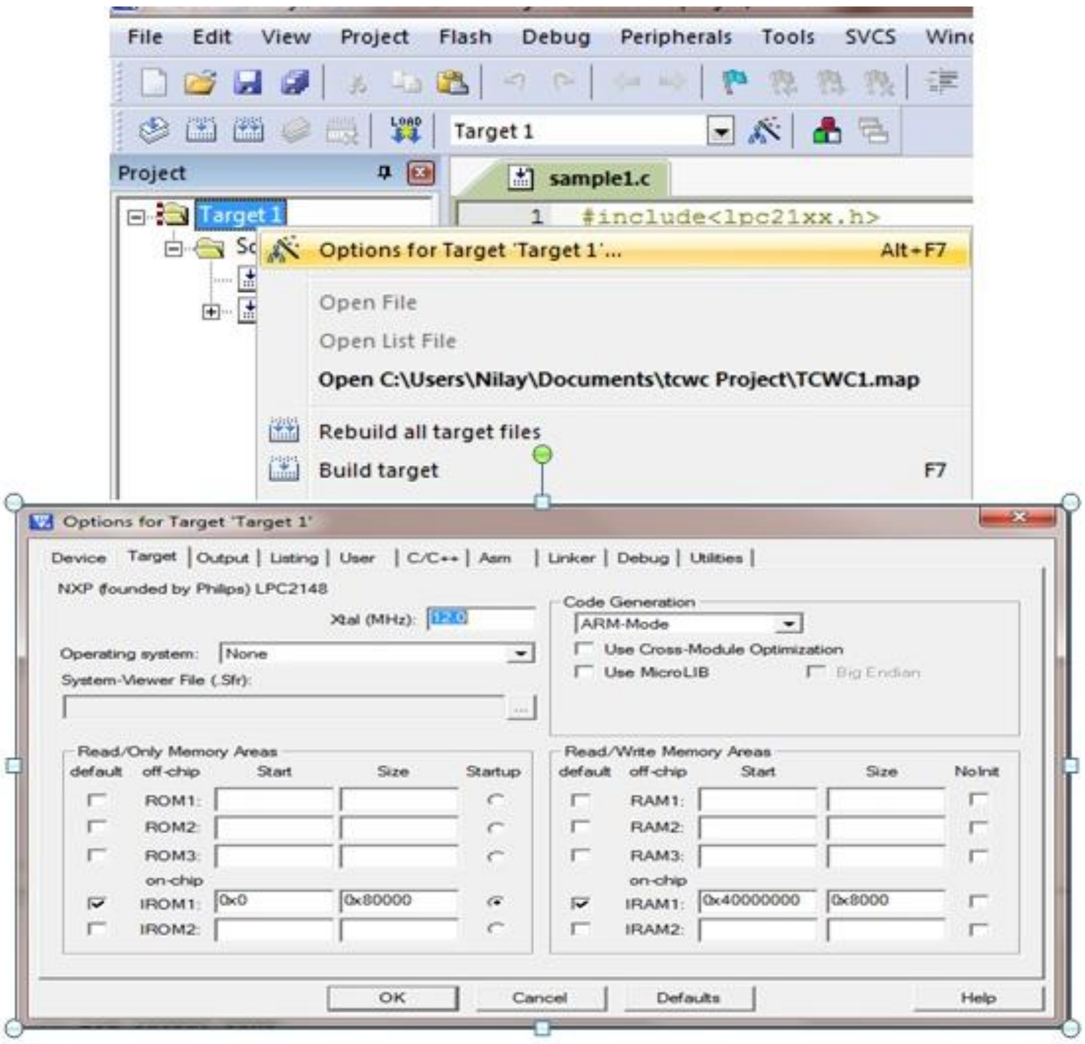
Step 4: Now go to File and create new file and save it with .C extension. After save the file the program and save it again.



```
    for(x=0;x<4;x++)
    {
      while(RI==0);
      y[x] = SBUF;
      RI = 0;
    }
    if(y[2]=='0' && y[3]=='A')
    {
      d1 = 1;
    }
    else if(y[2]=='1' && y[3]=='B')
    {
      d1 = 0;
    }
    else if(y[2]=='1' && y[3]=='F')
    {
      d2 = 1;
    }
    else if(y[2]=='0' && y[3]=='C')
    {
      d2 = 0;
    }
    else if(y[2]=='1' && y[3]=='2')
    {
      d1 = d2 = 0;
    }
    else if(y[2]=='1' && y[3]=='9')
    {
      d1 = d2 = 1;
    }
  }
}
```

Fig 6.4 Writing the code

Step 5: After that on left you see project window. Now come on Project window. Right click on target and click on options for target Here you can change your device also. Click output tab here & check create Hex file if you want to generate hex file Now click on ok so it will save changes.



```
#include<lpc21xx.h>
```

NXP (founded by Philips) LPC2148

Xtal (MHz): 12.0

Operating system: None

System-Viewer File (.Sfr):

Code Generation: ARM-Mode

☐ Use Cross-Module Optimization

☐ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
✓	IROM1:	0x0	0x80000	●
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
✓	IRAM1:	0x40000000	0x8000	☐
☐	IRAM2:			☐

Fig 6.5 Creation of the hex file

Step 6: Now Expand target and you will see source group Right click on group and click on Add files to source group. Now add your program file which you have written in C. You can see program file added under source group. Now Click on Build target. You can find it under Project tab or in toolbar. It can also be done by pressing F7 key.

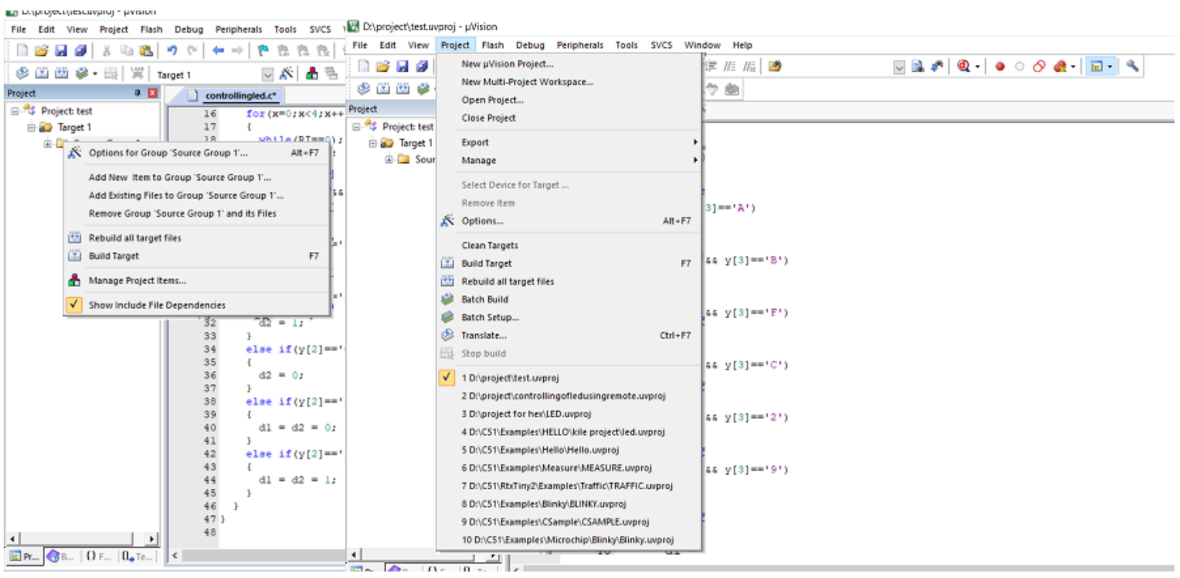
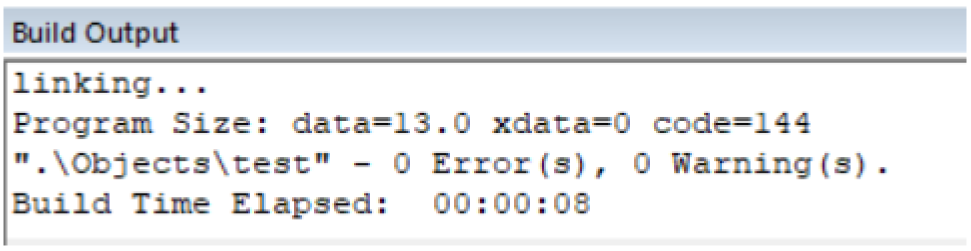

D:\project\test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
controllingled.c*
Project: test
Target 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Rebuild all target files
Build Target F7
Manage Project Items...
Show Include File Dependencies
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target ...
Remove Item
Options... Alt+F7
Clean Targets
Build Target F7
Rebuild all target files
Batch Build
Batch Setup...
Translate... Ctrl+F7
Stop build
1 D:\project\test.uvproj
2 D:\project\controllingofledusingremote.uvproj
3 D:\project for hex\LED.uvproj
4 D:\C51\Examples\HELLO\kile project\led.uvproj
5 D:\C51\Examples\Hello\Hello.uvproj
6 D:\C51\Examples\Measure\MEASURE.uvproj
7 D:\C51\RtxTiny2\Examples\Traffic\TRAFFIC.uvproj
8 D:\C51\Examples\Blinky\BLINKY.uvproj
9 D:\C51\Examples\CSample\CSAMPLE.uvproj
10 D:\C51\Examples\Microchip\Blinky\Blinky.uvproj

Fig 6.6 Building of the target

Step 7: you can see Status of your program in Build output window.



Build Output

```
linking...
Program Size: data=13.0 xdata=0 code=144
".\Objects\test" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:08
```

Fig 6.7 Built output

6.2 WILLAR

Loading the code to the microcontroller is called dumping. The microcontrollers understand only binary language. So, we need to load the hex code into the microcontroller. Here I am using ‘Willer’ programmer software to dump the code to the 8052 microcontrollers.

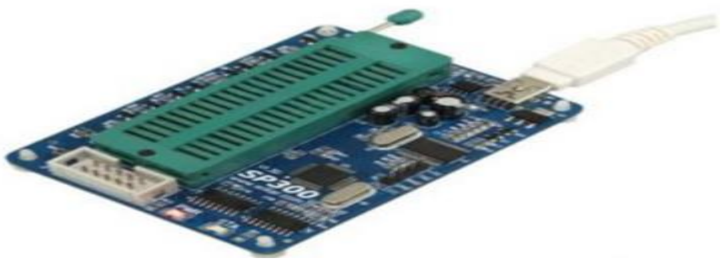
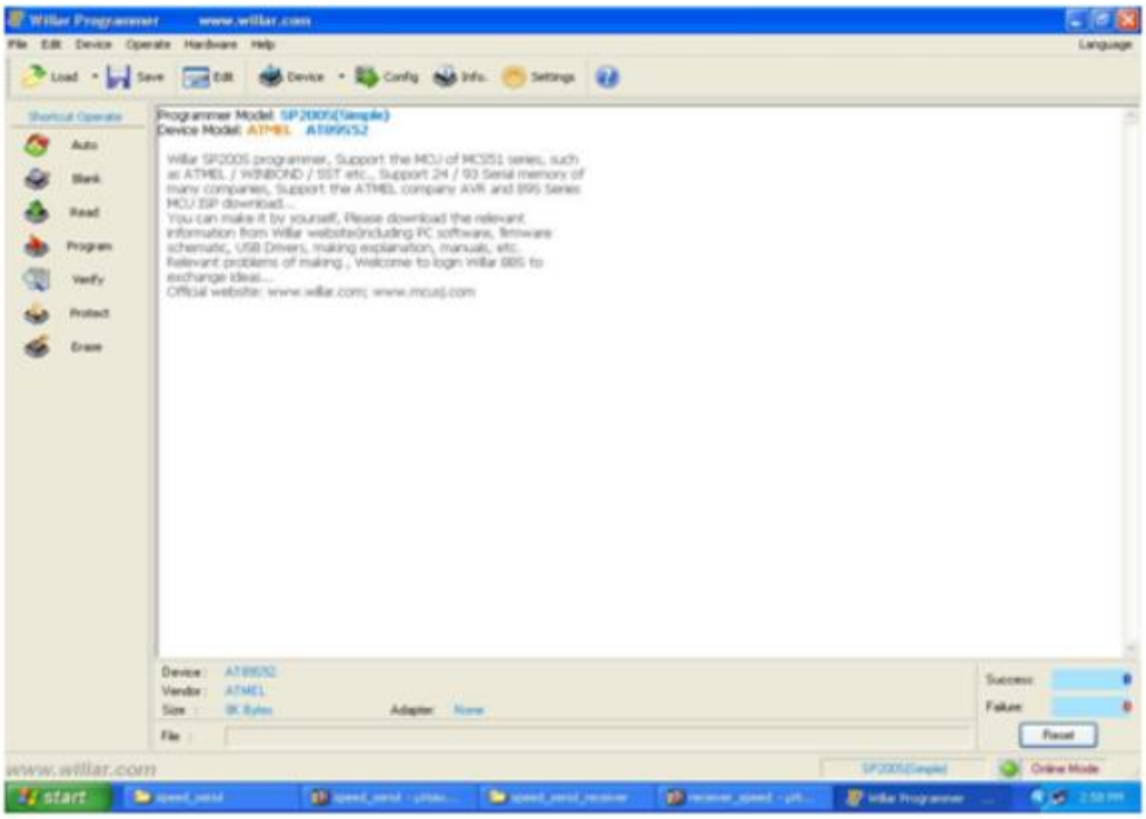


Fig 6.8 Programming kit

The programmer kit comes with software along with the hardware kit. This software needs to be installed onto the computer. The hardware kit comes with a socket, on which the microcontroller is placed. Here are the steps to load the code onto the microcontroller.


Willar Programmer www.willar.com
File Edit Device Operate Hardware Help
Load Save Edit Device Config Info. Settings
Shortcut Operate
Auto
Blank
Read
Program
Verify
Protect
Erase
Reset
start

Programmer Model: SP200S(Simple)
Device Model: ATMEL AT89S52

Willar SP200S programmer, Support the MCU of MCS51 series, such as ATMEL / WINBOND / SST etc., Support 24 / 93 Serial memory of many companies, Support the ATMEL company AVR and 89S Series MCU ISP download...
You can make it by yourself, Please download the relevant information from Willar website(including PC software, firmware schematic, USB Drivers, making explanation, manuals, etc.
Relevant problems of making., Welcome to login Willar BBS to exchange ideas...
Official website: www.willar.com; www.mcusj.com

Device : AT89S52
Vendor : ATMEL
Size : 8K Bytes Adapter: None
File :
Success: 0
Failure: 0

Fig 6.9 Software window

The hardware (programmer kit) is interfaced to the computer through a serial cable. The microcontroller is placed on the socket of the hardware kit. Press the lock button to ensure the microcontroller is connected to the board. Open the software installed in the computer. It will display some operating modes. Select any mode. A window with a menu bar appears. Click on the ‘file’ menu and select ‘load file’ option from the drop-down menu. Click on the ‘auto’ button so that the hex file is loaded to the microcontroller.

CHAPTER 7

7.DESIGN OUTPUT

7.1 RESULT

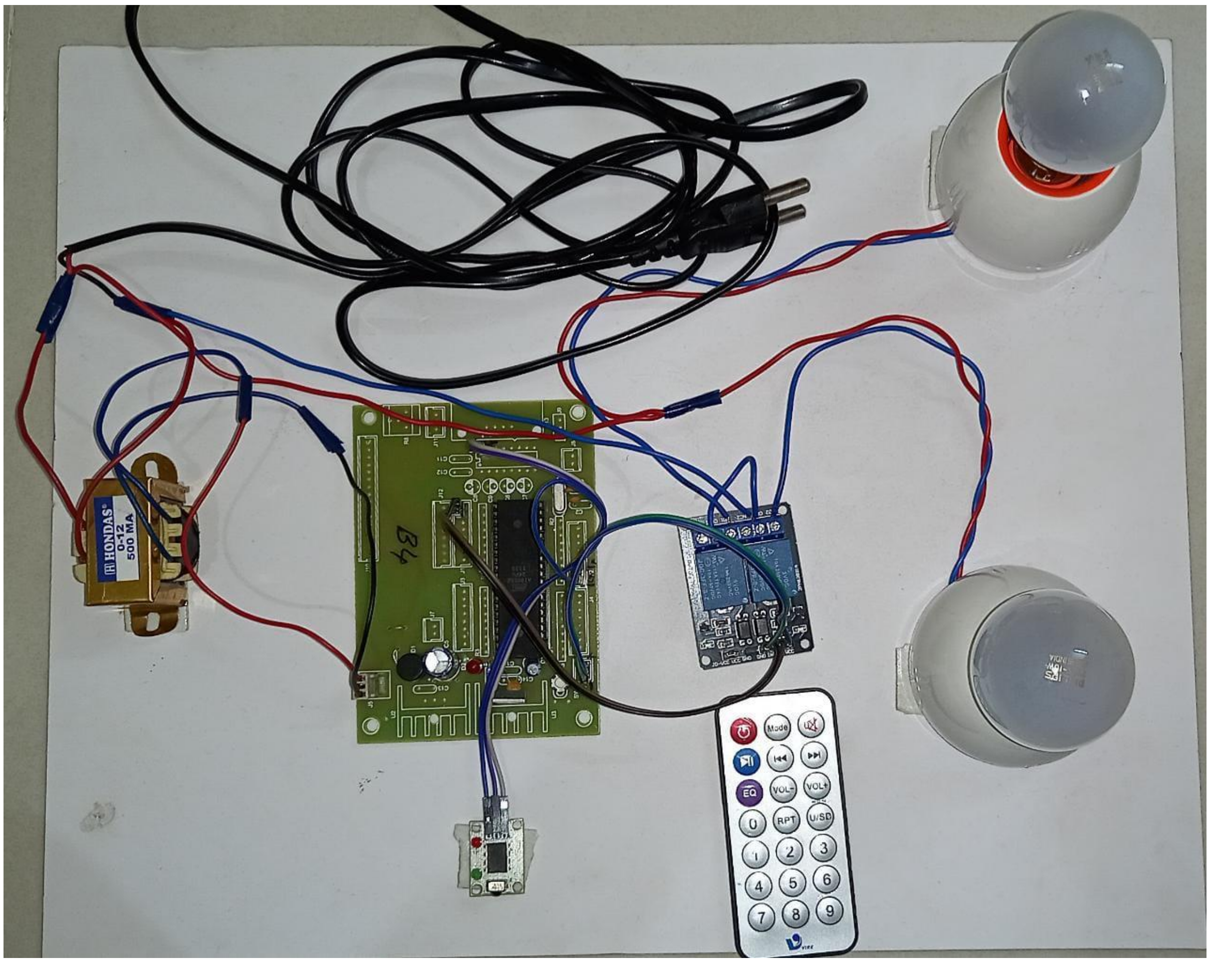


Fig 7.1 Photocopy of project

CHAPTER 8

8.CONCLUSION,FUTURE SCOPE AND BIBLIOGRAPHY

CONCLUSION

Remote controlling of home appliances using tv remote provided wireless control for home and offices. These are very essential in present life style, by this the manual operation is completely eliminated. The home appliances can be switched on/off using IR without actually going near the switch boards or regulators.

The loads like light, motors, heaters, power controlling system and also we controlled all loads at a time from one place without connecting any physical wire between loads and controlling room. This can create whole new dimension of remote controlling as it takes the control of as near as to one's palm for always and everywhere.

FUTURE SCOPE

- This is very smart an intelligent instrument useful for all the age group and has a variety of uses in almost all the areas where instruments need to be automated and controlled as per human need and enhance facility
- A single remote control can be made to operate at different frequencies, each corresponding to a particular task to be performed by the appliance.

BIBLIOGRAPHY

[1] N.Sriskanthan, F. Tan and A. Karande, "Bluetooth based home automation system", Microprocessors and Microsystems, vol. 26, no.6, (2002), pp. 281-289.

[2] K. Gill, S.-H. Yang, F. Yao and X. Lu, "A Zig-Bee-based home automation system", IEEE Transactions on Consumer Electronics, vol. 55, no. 2, (2009), pp. 422–430.

[3] N. Kushalnagar, G. Montenegro and C. Schumacher, "IPv6 over low-power wireless personal area networks (LoWPANs): overview, assumptions, problem statement, and goals", RFC 4919, (2007).

[4] M. Kovatsch, M. Weiss and D. Guinard, “Embedding internet technology for home automation”, In Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA ’10), (2010) September.

[5] B. M. Dorge and T. Scheffler, “Using IPv6 and 6LowPAN for home automation networks,” In Proceedings of the 1st IEEE International Conference on Consumer Electronics (ICCE ’11), (2011) September, pp. 44–47.

[6] D. S. Tudose, A. Voinescu and M.-T. Petrareanu, “Home automation design using 6LoWPAN wireless sensor networks”, Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS ’11), (2011) June, pp. 1–6.

[7] Khusvinder Gill, Shuang-Hua Yang, Fang Yao, and Xin Lu, “A ZigBee-Based Home Automation System” IEEE Transactions on Consumer Electronics, Vol. 55, No. 2, May 2009

[8] R.A.Ramlee, M.H.Leong, R.S.S.Singh, M.M.Ismail, M.A.Othman, H.A.Sulaiman, M.H.Misran, M.A.Meor Said “Bluetooth Remote Home Automation System using Android Application”

ANNEXURE

```
#include <REGX51.H>

sbit d1 = P2^0;

sbit d2 = P2^1;


void main()

{

        unsigned char x,y[4];

        TMOD = 0x20;

        SCON = 0x50;

        TH1 = 0xFD;

        TR1 = 1;
```

```
x = 0;
d1 = d2 = 0;
while(1)
{
        for(x=0;x<4;x++)
        {
                while(RI==0);
                y[x] = SBUF;
                RI = 0;
        }
        if(y[2]=='0' && y[3]=='A')
        {
                d1 = 1;
        }
        else if(y[2]=='1' && y[3]=='B')
        {
                d1 = 0;
        }
        else if(y[2]=='1' && y[3]=='F')
        {
                d2 = 1;
        }
        else if(y[2]=='0' && y[3]=='C')
        {
                d2 = 0;
        }
        else if(y[2]=='1' && y[3]=='2')
        {
                d1 = d2 = 0;
        }
        else if(y[2]=='1' && y[3]=='9')
        {
                d1 = d2 = 1;
        }
    }
}
```