PROJECT REPORT

On

PULSE RATE MONITOR.

Submitted in partial fulfillment of the Degree of Bachelor of Technology

By

SUNEET MEHTA -041038 SHARANG NAWANI -041067

SUSHAN SHARMA -041065 SAHIL AGGARWAL -041083

Under the Guidance of : Mr. Sunil Bhooshan

Mr. Vinay Kumar

DEPARTMENT OF ELECTRONICS AND COMMUNICATION.

JAYPEE UNIVERSITY OF INFORMATION. TECHNOLOGY-WAKNAGHAT.

MAY-2008

CERTIFICATE

This is to certify that the work entitled, **"Pulse Rate Monitor"** submitted by we four group members in partial fulfillment for the award of degree of Bachelor of Technology in **Electronics And Communication** of **Jaypee University of Information Technology** has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

[illegible]

Mr. Vinay Kumar
Sr.Lecturer,
ECE Department
JUIT

[illegible]

Dr. S.V.Bhooshan
H.O.D,
ECE Department
JUIT

ACKNOWLEDGMENT

The project in this report is an outcome of continual work over a period of one year and intellectual support from various sources. Obligations thus occurred in completing the work have been many. It is therefore almost impossible to express adequately the debts owed to many persons who have been instrumental in imparting this work a successful status. It is however a matter of pleasure to express our gratitude and appreciation to those who have been contributing to bring about this project.

We take this opportunity to thank our esteemed mentor and supervisor Mr. Vinay Kumar, Sr. Lecturer and to Mr. S.V. Bhooshan, Head of Department Of Electronics and Communication Engineering, for lending us stimulating suggestions, innovative quality guidance and creative thinking. Their practicality, constructive criticism, constant encouragement and advice helped us in all the stages of the project. We are grateful to them for the support they provided us in doing things at our pace and for being patient with our mistakes.

We would also like to thank Mr.Ashwini Jain, Sr.Lecturer , NIT, Kurukshetra and Mr.Surendra Gandhi,Officer,IRDE,Dehradun for helping us in the intial thought process.

We were fortunate to have very supportive friends with whom, it was pleasure to work. This unreserved help, adjusting nature and friendly atmosphere will always be cherished and remembered.

[illegible]
SUNEET MEHTA (041038)

[illegible]
SHARANG NAWANI (041067)

Sushan Sharma.
SUSHAN SHARMA (041065)

Sahil.
SAHIL AGARWAL (041083)

Table of Contents:

List of figures

ABSTRACT

This project aims at measuring the heart beat(or pulse rate) of a human being by the use of a simple infra-red sensor whose output goes to a circuit comprising of a current to voltage converter, differentiator and a non-inverting amplifier made using IC LM-324 (quad op-amp ic) and then display it on a LCD in digital or graphical way with the help of a microcontroller, using c programming for microcontrollers.

In this report we have shown the implementation of our circuit showing the output in form of the visual output on an LCD, the circuit consists of basic operational amplifiers, transistor and a microcontroller which has been programmed to give the output on LCD.

CHAPTER 1
INTRODUCTION

In today's era of automated systems and user-friendly equipments the microcontroller has evolved as a device with increasing importance and vast number of utilities .The reason is simple: as the market is expanding, the consumer is getting more and more options with the utility products that he buys and his demand and expectations increase each time a new product is introduced in the market , as the expectations increase so does the number of experiments being carried out using a microcontroller , this further enhances the possibilities of using a microcontroller in other fields. One of the major reasons for using a microcontroller is their cost effectiveness, i.e. they provide many enhancements with only a little increase in the cost.

The Pulse Rate Monitor highlights the use of microcontroller as a counter ,selection device(for age) and then displaying the output on an LCD .This device makes the use of simple circuits like differentiator , current to voltage converter and non-inverting amplifier with an infra red sensor used effectively to count the heart beat and then the microcontroller makes this circuit better by giving us the option of choosing age and then showing the output on an LCD thereby enhancing the visual appeal of the circuit. The use of simple programming logic in the microcontroller helps us to indicate the user about the normal and danger levels of heart beat depending on the age chosen by the user.

The use of c programming for microcontrollers and the Keil c51 compiler enhances the portability of the program and makes the code easier to comprehend.

1.1 : APPROACH

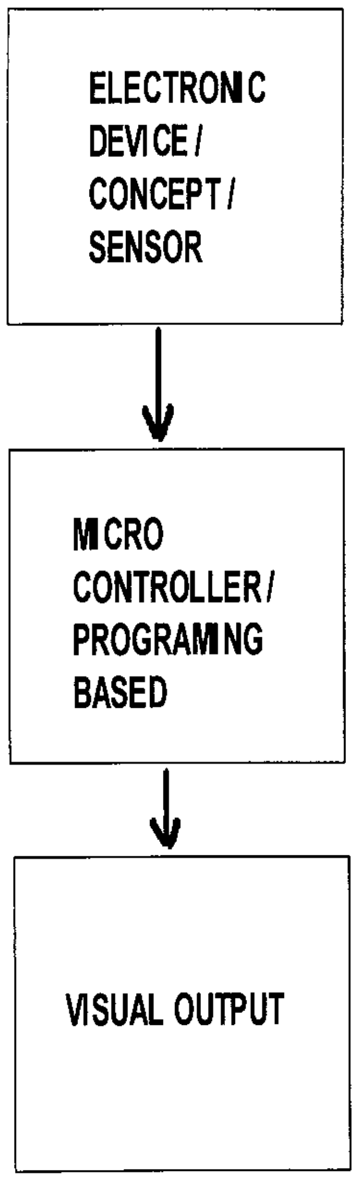

ELECTRONIC DEVICE / CONCEPT / SENSOR
MICRO CONTROLLER / PROGRAMING BASED
VISUAL OUTPUT

Fig 1.1

1.2 : CIRCUIT DIAGRAM

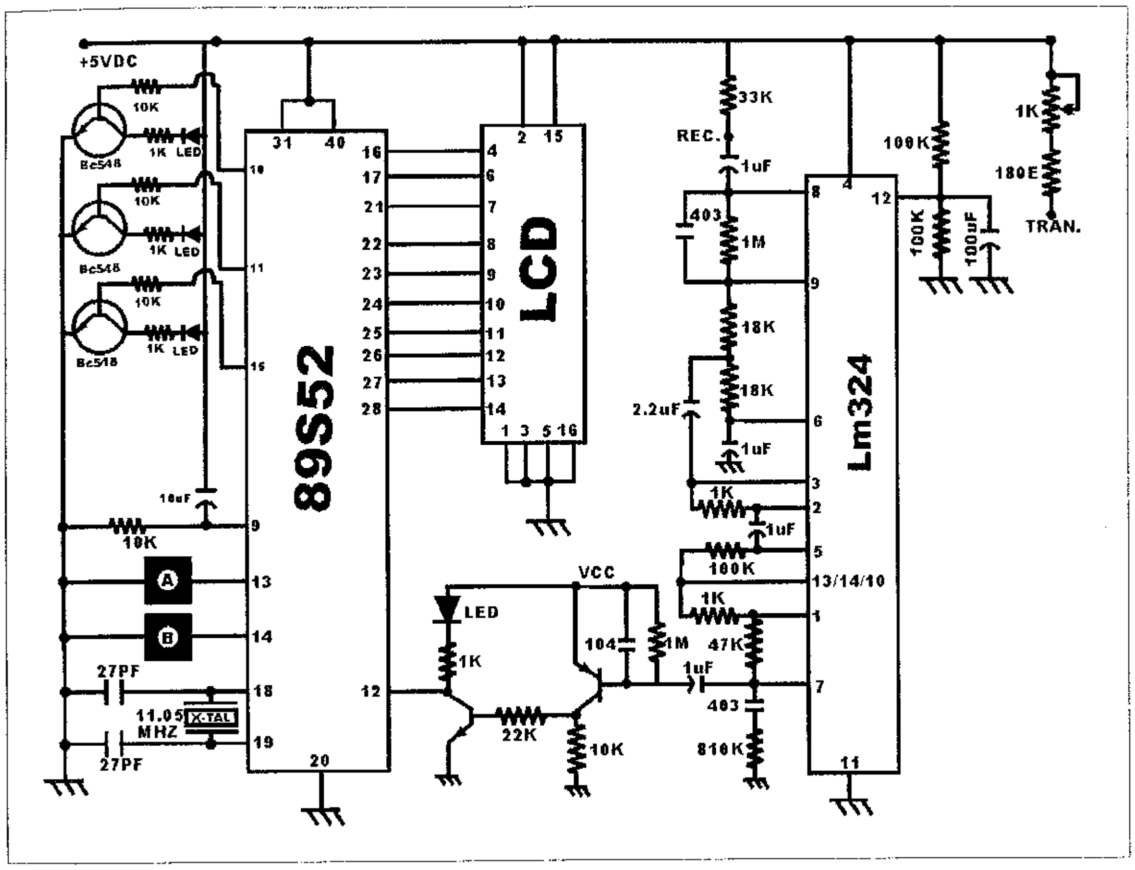


Fig 1.2

1.3 : BLOCK DIAGRAM

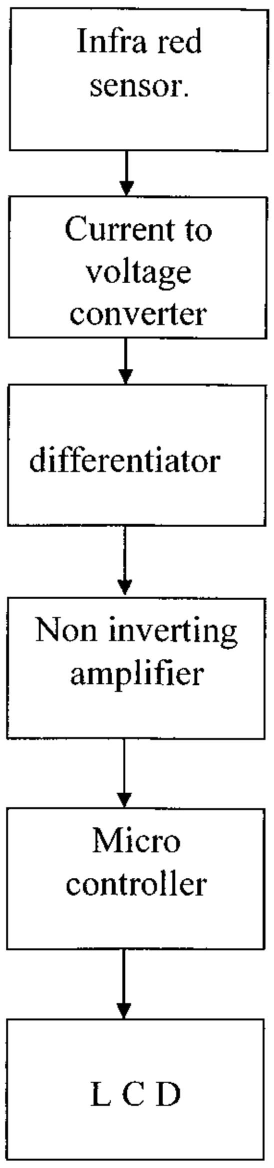


Fig 1.3

1.4 : FLOW DIAGRAM

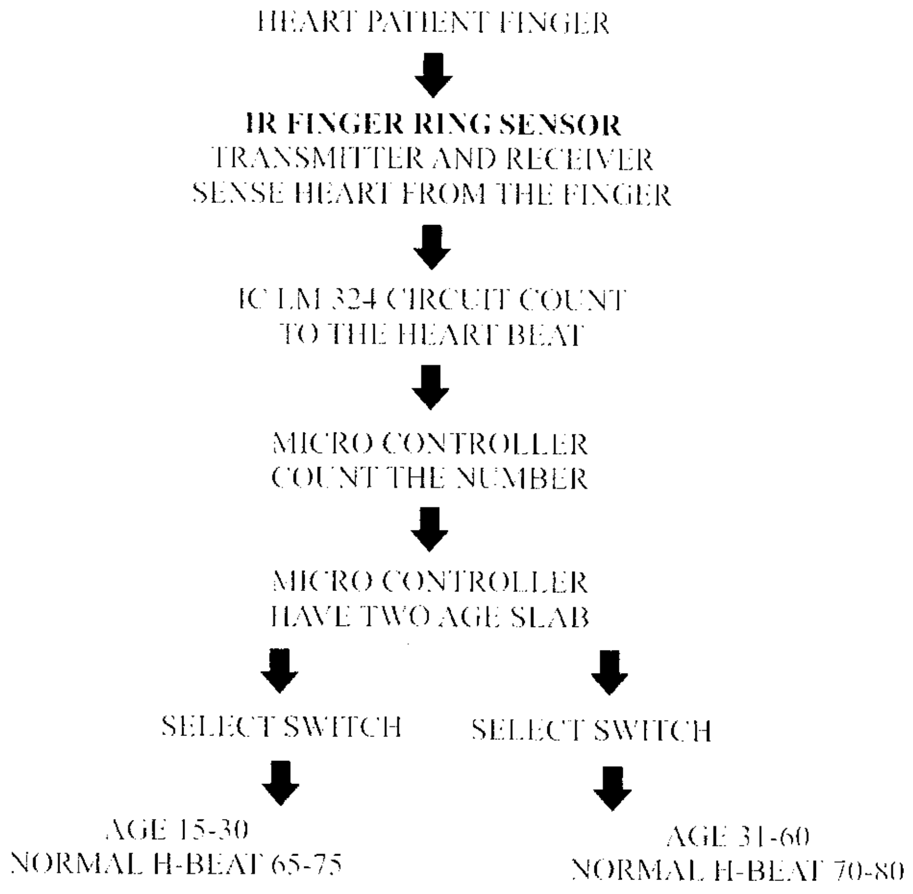


Fig 1.4

CHAPTER 2

MICROCONTROLLERS : INTRODUCTION AND APPLICATIONS

2.1 : DEFINING MICROCONTROLLERS

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

A highly integrated chip that contains all the components comprising a controller. Typically this includes a CPU, RAM, some form of ROM, I/O ports, and timers. Unlike a general-purpose computer, which also includes all of these components, a microcontroller is designed for a very specific task – to control a particular system. As a result, the parts can be simplified and reduced, which cuts down on production costs A {microprocessor} on a single {integrated circuit} intended to operate as an {embedded} system. As well as a {CPU}, a microcontroller typically includes small amounts of {RAM} and {PROM} and timers and I/O ports A single chip that contains the processor (the CPU), non-volatile memory for the program (ROM or flash), volatile memory for input and output (RAM), a clock and an I/O control unit. .A microprocessor on a single integrated circuit intended to operate as an embedded system. As well as a CPU, a microcontroller typically includes small amounts of RAM and PROM and timers and I/O ports. The definitions given by various sources describe microcontroller as an integrated circuit (IC) with processor as well as peripherals on chip.

But the crux of the matter is the widespread uses of microcontrollers in electronic systems. They are hidden inside a surprising number of products such as microwave oven, TV, VCR, digital camera, cell phone, Camcorders, cars, printers, keyboards, to name a few. The last three decades and especially the past few years have witnessed the tremendous growth in the top-of-the-line processors used for personal computing and digital signal processor (DSP) applications. Today, the use of microcontrollers in embedded systems outnumbers the use of processors in both the PC and the workstation market. It is estimated that over 50% of the vast majority of the computer chips sold are microcontrollers. The main reasons behind their huge success are powerful yet cleverly chosen customizable electronics, ease in programming, and low cost. These days microcontrollers find increasing application even in areas where they could not be used previously. With the decreasing costs and footprints, it is now possible to have small-sized and cost-effective microcontroller units (MCUs) for new applications.

The microcontrollers today are small enough to penetrate into the traditional market for 4-bit applications like TV remote controls, toys, and games. For the simplest of all control applications they can offer high value smart switch functionality for applications that previously used electromechanical devices. Also the customers now can add intelligence to their end products for low cost as per the microcontroller market report by Frost & Sullivan research service .

2.2 : MICROCONTROLLERS AND OTHER COMPETING DEVICES

Generally the technical fraternity tries to compare the various devices like microprocessors, PCs, microcontrollers, DSPs, and reconfigurable devices like FPGAs and CPLDs. An interesting point to note is that embedded systems are made using all the above-mentioned devices except PC owing to its general purpose architecture. As programmability is the common feature of all these devices, they have their firm footing in different application domains. On one side of the spectrum, microcontroller-based embedded system design emphasizes on task specific dedicated applications at low power, low cost, high throughput, and highest reliability. On the other extreme of the spectrum, FPGA-based embedded systems dominate their custom computing architectures. Unlike microcontrollers, these systems can be reconfigured on the fly to suit the application with higher computational density and throughput. With the proliferation of density, FPGA-based embedded systems offer higher performance with the only challenging issue of memory required to store their configurations.

The technical community also tends to associate various characteristics of embedded systems with microprocessors and microcontrollers. The microprocessors are typically found to dominate the desktop arena focusing on more and more bit processing capability, with other features such as fairly good amount of cache with memory management and protection schemes supported by the operating system. Although microcontrollers share flexibility aspect of microprocessors through programming, 8-bit versions are more in use (although 16- and 32-bit exist) with RAM and ROM (instead of cache) added with many on chip peripherals like timer/counter, decoder, communication interface, etc.

In the literature many embedded systems products have been reported as microprocessors. On the other side of the processor spectrum, a DSP possesses special architecture that provides ultra-fast instruction sequences, such as shift and add, multiply and add, which are commonly used in math-intensive signal processing applications. The common attributes associated with the DSPs are multiply-accumulate (MAC) operations, deep pipelining, DMA processing, saturation arithmetic, separate program and data memories, and floating point capabilities required most of the time. However, the line of differentiation between all these devices is getting blurred at a rapid pace. With the introduction of fuzzy logic, artificial intelligence and networked communication capabilities in the consumer products like refrigerators, mobile phones, and cars, convergence of the architectures of most of the above-mentioned programmable devices is witnessed by the industry. Today's ideal microcontroller is expected to offer plenty of MIPS, run the DSP programs with the same speed of the DSP processor, integrate all its peripherals and support flash, communicate with the world with I2C or CAN protocols, withstand extremes of environment in a car engine, and cost but a few cents.

2.3 : VIGNETTES: MICROCONTROLLERS

It is interesting to note that the development of microprocessors seems to be an accident out of the microcontroller synthesis. In 1969, Busicom, a Japanese company, approached Intel to convert their special purpose ROM and shift register–based calculator cores into a specialized application specific processor. The objective was the development of microcontrollers rather than a general purpose of CPU chips for keyboard scanning, display control, printer control, and other functions for a Busicom-manufactured calculator. However, the Intel engineers opted for a more flexible programmable microcomputer approach rather than the random logic chip-set originally envisioned by Busicom. The four designs proposed by Federico Faggin, Ted Hoff, and Stan Mazor from Intel were a 2048-bit ROM with a 4-bit programmable input/output port (4001); a 4-registers × 20-locations × 4-bit RAM data memory with a 4-bit output port (4002); an input/output (I/O) expansion chip, consisting of a static shift register with serial input and serial/parallel output (4003); and the 4-bit CPU chip (4004). The 4001, 4002, and 4003 are very close to microcontroller kind of architecture rather than microprocessor. However, the Intel 4004, which was supposed to be the brains of a calculator, turned out to be a general-purpose microprocessor as powerful as ENIAC. The scientific papers and literature published around 1971 reveal that the MP944 digital processor used for the F-14 Tomcat aircraft of the US Navy qualifies as the "first microprocessor". Although interesting, it was not a single-chip processor, and was not general purpose – it was more like a set of parallel building blocks you could use to make a special purpose DSP form . It indicates that today's industry theme of converging DSP-microcontroller architectures was started in 1971. The other companies were also catching up at the same time. The first official claim of filing the patent goes to Texas Instruments under the trade name TMS1000 way back in 1974. This was the first microcontroller which included a 4-bit accumulator, 4-bit Y register and 2- or 3-bit X register, which combined to create a 6- or 7-bit index register for the 64 or 128 nibbles of on-chip RAM. A 1-bit status register was used for various purposes in different contexts. This microcontroller served as the brain of the Texas Instrument's educational toy named "Spark and Spell" shown in the movie *ET: The Extraterrestrial*. In 1976, Intel introduced the first 8-bit microcontroller family

MCS-48 which was so popular that they could ship 251,000 units in that year. After four years of continuous research, the MCS-48 family was upgraded to 8051, an 8-bit microcontroller with on-board EPROM memory. Intel shipped 22 million pieces in 1980. The market requirement was so much that the total units sold in three years later were 91 million. The year 2005 is a special one for the microcontroller 8051. It has celebrated its 25th anniversary. But, also in 2005, Intel notified they would discontinue all automotive versions of their microcontrollers, including 8051. Car engine control units were once perhaps the most prominent application for 8051s. This means only one thing, Intel gives up the microcontrollers for good. This is confirmed by product change notification published in early 2006, announcing that Intel drops its whole microcontroller business.

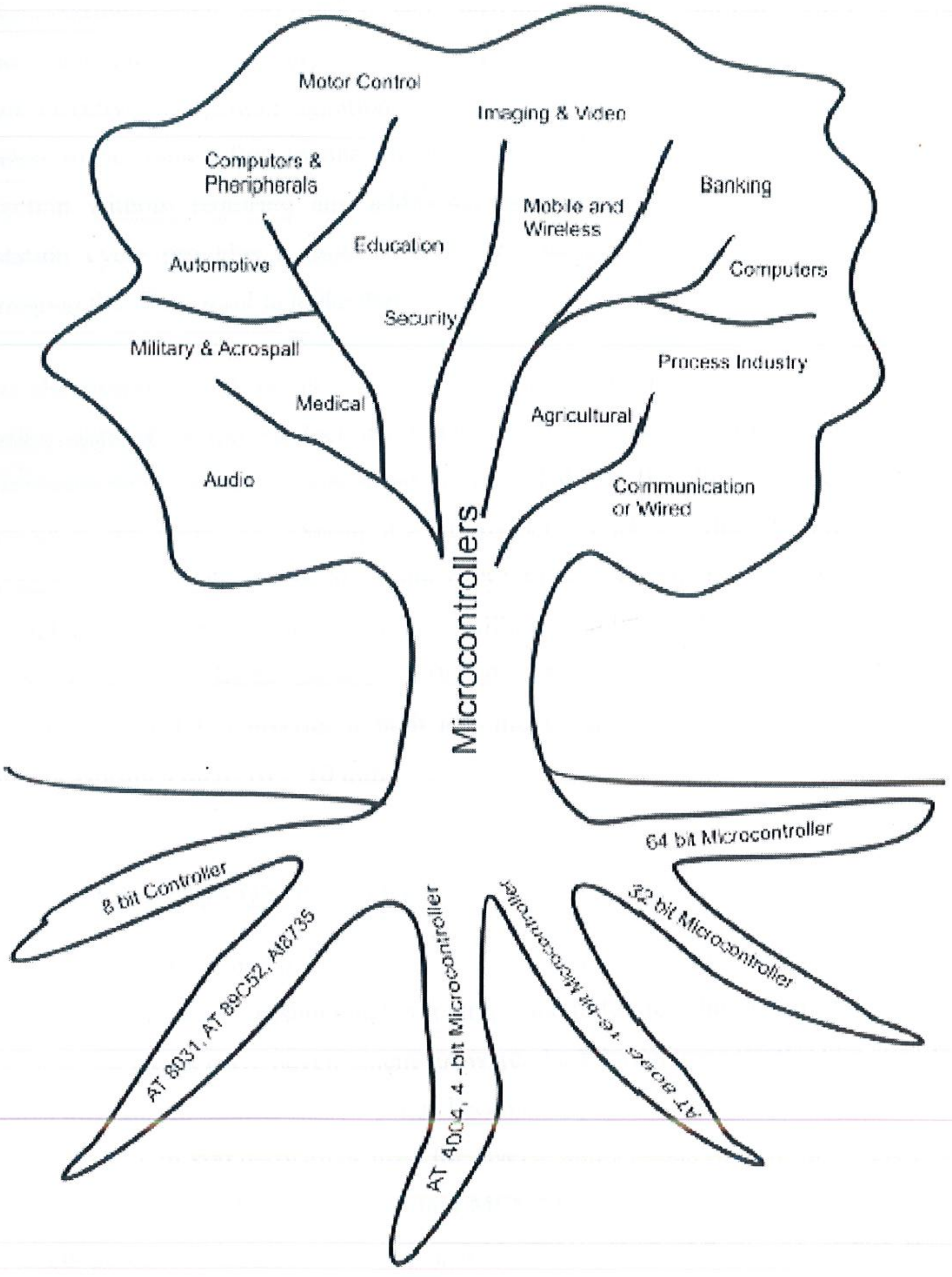
2.4 : MICROCONTROLLER APPLICATIONS

The microcontroller applications are mainly categorized into the following types (see Figure 1.1):

- Audio
- Automotive
- Communication/wired
- Computers and peripherals
- Consumer
- Industrial
- Imaging and video
- Medical
- Military/aerospace
- Mobile/wireless
- Motor control
- Security
- General Purpose
- Miscellaneous

Automobile industry is the main driving force in propelling the growth of microcontrollers. It is estimated that the microcontrollers constitute 33% of the semiconductors used in a vehicle . Requirements of the automobile sector has forced the microcontroller manufacturers to come out with the new bus protocols such as control area network (CAN) and local interconnect network (LIN). Microcontrollers of all bit cores are used in vehicles according to the Frost & Sullivan Industry report. The 8- and 16-bit microcontrollers are used for low-end applications and lower-cost vehicles while the 32-bit microcontrollers are used for high-end application and high-end vehicles. It is estimated that currently 30–40 microcontrollers are used in low-end vehicles and about 70 microcontrollers are used in high-end vehicles. These requirements are continuously increasing and it is highly likely that the count of microcontrollers in vehicles will further increase in the future, quotes World Microcontrollers Market Report by Frost & Sullivan.

MICROCONTROLLER APPLICATION TREE



Embedding microcontrollers in the product offers some unique advantages. For an example, in the latest technology washing machines a transmission is no longer required because a lower-cost AC induction or reluctance motor controlled by sophisticated microcontroller-based electronics can provide all the normal machine cycles Additionally, the electronically controlled induction or reluctance motor provides a more effective and gentler agitation (wash) cycle that allows the drum containing the clothes to be rotated first in one direction, then stopped, and rotated in the opposite direction without requiring any additional mechanical device. This forward/reverse agitation cycle provides a more effective means of cleaning your clothes without damaging the fibers used to make them.

It is also observed that the induction of microcontrollers in a product has increased the market demand of the product itself. One such example is NEC Electronics' 8-bit microcontroller , which is employed in over half of the digital cameras produced throughout the world, thus making it a hit product – albeit one that plays its role behind the scenes. In 2003, the shipment volume of 8-bit microcontrollers for digital camera use achieved monthly shipments of two million. Currently, the most commonly used microcontroller for digital cameras is the µPD78003×. The industry's top-level, low-voltage (1.8 V) A/D converter is built into this compact 64-pin QFP package with an edge measuring a mere 10 × 10 mm.

2.6 : 8-BIT TREADS ON MCU TURF

It is really interesting to note that in this era of 32-bit processors, the 8-bit microcontrollers are flourishing and enjoying a stable future. The reasons behind this are low cost and inexpensive development software. An 8-bit microcontroller like 8051 today enjoys 40% of the market share . It has become so popular that about 40 manufacturers now make it with 800 derivatives used for diverse applications right from simple process control to high-tech telecom. The original MCS-51 family developed by Intel consists of CHMOS processors with features such as 8-bit CPU optimized for event control, Boolean processing, on-chip memory (up to 32 K) and on-chip peripherals (timer/counters, serial ports, PCA, etc.). The other manufacturers like Atmel, Dallas, Philips, Analog Devices, etc., have extensively worked for strengthening the basic core by introducing additional features such as Additional Data Pointers (DPTR), Extended Stack Space, Fast Math Operations and Extended or Reduced Instruction Sets to name a few. Table 1.1 summarizes the features of the popular 8051 derivatives manufactured by various companies. Although industry analysts predicted the saturation of 8051 family and also its death, it turned out to be a rumor. In 2003, Cygnal released world's highest performance 8051 microcontroller [16]. The C8051F120 family devices include 128 Kbyte Flash memory, 8448 byte RAM, and are packaged in either a 100-pin or 64-pin TQFP. On-chip peripherals include a 12-bit 100 ksps ADC, two 12-bit DACs, a 2% internal oscillator, temperature sensor, voltage reference, two UARTs, and many features of a DSP processor.

2.7 : LOW POWER DESIGN

The latest 8-bit devices continue to drive up the performance bar with simplicity for usage and ease of programming. Most of these devices are aimed at low power consumption achieved by using sleep modes and the ability to turn certain peripherals on and off. The best example for this is XE88LC02 from XEMICS , a recently launched microcontroller which features programmable gain/offset amplifier followed by high resolution ADC with four differential or seven pseudo differential inputs, with current consumption as low as 2 μA in real-time clock mode and a typical 300 μA/MIPS in sustained computing mode. Recently, Atmel has also strategically evolved their microcontroller architecture by implementing novel power saving techniques. Atmel's picoPower technology uses a variety of techniques that eliminate unnecessary power consumption in power-down modes. These include an ultra-low-power 32 kHz crystal oscillator, automatic disabling and re-enabling of brownout detection (BOD) circuitry during sleep modes, a power reduction register that completely powers down individual peripherals, and digital input disable registers that reduces the leakage current on digital inputs [18]. PicoPower microcontrollers consume down to 340 μA in active mode, 150 μA in idle mode at 1 MHz, 650 nA in power-save mode and 100 nA in power-down mode.

2.8 : STARTING EMBEDDED SYSTEM PROJECT

The foremost requirement is a lowend PC pre-loaded with the IDE to facilitate the code development, simulation and testing before actual dumping the code in the flash of the microcontroller. The Keil IDE is covered in depth in Chapter 2. Most of
the projects developed in this book have been tested on AT89S52 which has the following features: 8K Bytes of In-System Reprogrammable Flash Memory Fully Static Operation: 0 Hz to 33MHz

- Three-level Program Memory Lock
- 256 × 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes
- 4.0V to 5.5V Operating Range
- Full Duplex UART Serial Channel
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag
- Fast Programming Time
- Flexible ISP Programming (Byte and Page Mode)

2.9 : INTEGRATED DEVELOPMENT ENVIRONMENT

Integrated development environment popularly known as IDE is a suite of software tools that facilitates microcontroller programming. The Keil IDE enables the embedded professional to develop the program in C and assembly as well. The IDE passes through the source code to check the syntax. The compilation leads to a hex file to be dumped in the microcontroller on-chip ROM. A quick session of simulation and debugging using the IDE ensures the working of the program beforehand. The user can verify the results as the package presents screenshots of on-chip resources. This chapter presents in-depth discussion on using the µVision 2 package of Keil IDE on MS Windows platform. It is recommended that while going through the discussion the user should access the µVision 2 package of the Keil. A step-by-step working as discussed in this chapter will empower the user to get familiar with the Keil IDE.

2.10 : GETTING FAMILIAR WITH THE IDE

The microcontroller product development cycle consists of several steps such as

- Development of code either in assembly or C
- Simulation of the code
- Dumping the code in microcontroller
- Prototyping or debugging if required by using in-circuit testing
- Emulation of the code in case of big project
- Refining the code, reprogramming and final testing

A microcontroller-based project generally makes several iterations through the above-mentioned steps before it sees the light of the day.

Almost all the microcontroller manufacturers have come out with the development tools for their products. A suite of such software tools for microcontroller application development is refereed to as IDE. Some examples of the popular IDEs apart from Keil are MPLAB from Microchip, PE micro from Motorola, and AVR studio from Atmel.

Any typical IDE comprises many subcomponents such as a text editor for entering the code, building tools for converting the code into machine level, compiler to convert 'C' code to assembly or hex format, and linker to provide absolute addresses to the relocatable object code. The IDE is generally equipped with powerful code simulator that models operation of the target microcontroller as code execution is in process. IDEs are available as a software suite which runs on a stand-alone PC and displays and allows to modify the contents of virtual I/O and on-chip resources such as registers. Single-stepping and breakpoint facilities are provided to systematically execute and watch the cause/effect relationship of the code. A provision of instruction cycle calculation/display is also provided so as to see the time/memory efficiency of the code.

2.11 : WORKING WITH KEIL IDE

As the entire software development described in this book is based on the Keil 8051 Development Tools, it is worthwhile to study the tool in depth. The Keil IDE is a user-friendly software suite to develop your 'C' code and test the same in a virtual 8051 environment. The main feature of the Keil is that it allows C programmers to adapt to the 8051 environment quickly with little learning curve. It offers the designer a device database of MCS-51 family from which the target device of interest can be chosen. The μVision IDE sets the compiler, assembler, linker, and memory options for the designer. The suite comes with numerous example programs to assist the designer to start his project. With the virtual environment, the available on-chip resources of the microcontroller chosen can be seen working on the PC screen. The simulation window facilitates very realistic simulation of both CPU and embedded peripherals. The graphical window shows the state and configuration of the embedded peripherals and displays the interaction of the microcontrollers with external peripherals. Although
the simulation exercise consumes time, it helps to save the bugs and project failure in the long run.

A natural question that occurs to designer is in what way the conventional C programming for PC is different from the C programming with Keil or C51 programming. The basic objective of the conventional C programming language was to make it work on the PC architecture with a high level interface. Therefore, it does not support direct access to registers. However, unlike the conventional C programming on PC, most of the microcontroller-based embedded systems applications demand reading and setting or resetting of single bits, or Boolean processing. The second main difference is the programming environment. Conventional C operated under the umbrella of an operating system may be Linux or Windows, wherein there is a provision of system calls the program may use to access the hardware. Almost all the programs written with microcontroller are for direct use where the while(1) loop is common. The above-mentioned differences are bridged in Keil compiler by adding several extensions to the C language, e.g., connecting of interrupt handlers.

2.12 : DEVELOPMENT FLOW FOR THE KEIL IDE

The evaluation version of the Keil IDE for MCS-51 family (also known as C51 evaluation software) can be downloaded from the website www.keil.com. One can go up to a limit of 2 Kbytes of object code with the evaluation version. Once the Keil IDE is installed, a short cut appears on the desktop.

STEP 1: INTERFACES OFFERED BY KEIL IDE

After clicking on the above shown shortcut the following steps should be carried out. The first blank window will appear as shown in Figure 2.1. You will be working with three windows presented by the Keil IDE. The first is target toward the extreme left of the screen which is blank at the moment, but will be updated as you go on working with the project. The source file, register header file, and the target microcontroller chosen is displayed here. The bookmarks for the user such as data sheet, user guides, library functions, etc., forms a ready reckoner for the developer. The program window which occupies most of the size of your screen displays the source code in C. At the lowermost end of the screen, an output window is presented which gives information regarding the errors and other output messages during the program compilation. Open a new text file for writing your source code. This file has to be saved with .c extension. This source code file will be ultimately added into your project to be opened as per step 2. The addition should be done as per the instructions given in step 5.

STEP 2: OPENING A NEW PROJECT

Here you can give a meaningful name for your project. After saving it will create a folder which will store your device information and source code, register contents, etc. Figure 2.2 shows a project opened with a name trial.

STEP 3: SELECTING A DEVICE FOR THE TARGET

After completing step 2, Keil will give an alert to select the device. The µVision 2 supports 45 manufacturers and their derivatives. In the exercise given in this book we have selected Atmel's 89S52 microcontroller as a target.

STEP 4: COPYING STARTUP CODE TO YOUR PROJECT

The "startup.a51" will be added automatically to your project from the Keil library "c:\keil\c51\lib" to [c:\keil\c51\EXAMPLES\HELLO\ ctrial\STARTUP.A51].

STEP 5: ADDING YOUR PROGRAM SOURCE CODE

To accomplish this, follow the steps:

- Right click "source group 1" followed by
- Choose "Add Files to Group 'Source Group 1' "
- Set "Files of type" to "All files (*. *)"
- Select "Startup.a51"

Observe the files getting updated in the target window. You will have to double-click on the C source file name displayed in the target window to view it.

STEP 6: CONFIGURING AND BUILDING THE TARGET

Right click on target 1 in the target window, select the option for target 1, a window to choose the options for the target will be displayed. Here you can choose the microcontroller frequency, listing of files, output in hex, debug information, etc.

The important point here is choosing the appropriate memory model. As per the on-line Keil IDE manual C51 currently supports the following memory configurations:

- ROM: currently the largest single object file that can be produced is 64 K, although up to 1MB can be supported with the BANKED model described below.
- All compiler output to be directed to EPROM/ROM, constants, look-up tables, etc., should be declared as "code".
- RAM: There are three memory models, SMALL, COMPACT, and LARGE
- SMALL: all variables and parameter-passing segments will be placed in the 8051's internal memory.
- COMPACT: variables are stored in paged memory addressed by ports 0 and 2. Indirect addressing opcodes are used. On-chip registers are still used for locals and parameters.
- LARGE: variables etc. are placed in external memory addressed by @DPTR. On-chip registers are still used for locals and parameters.
- BANKED: Code can occupy up to 1 MB by using either CPU port pins or memory-mapped latches to page memory above 0×FFFF. Within each 64 KB memory block a COMMON area must be set aside for C library code. Inter-bank function calls are possible.

STEP 7: COMPILE YOUR PROGRAM BY PRESSING F7

Either press F7 or click on build target in the target window to compile your program. The output window will show the errors or warnings if any. You can also see the size of data, code, and external data, which is of immense importance since your on-chip memory is limited.

STEP 8: WORKING IN SIMULATED MODE

Once the program is successfully compiled, you can verify its functionality in the simulated mode by activating the debug window. For this press CTRL + F5 or go to the menu option "Debug" and select "Start and Stop Debug Section". Press F11 for single stepping or F5 for execution in one go. Go to the menu item "Peripheral" and select the appropriate peripherals to view the changes as the program starts executing. Terminating the debug session is equally important. Click on “stop running” or ESC key to halt the program execution. You can make the changes to the program after coming out from the debug session by pressing start/stop debug session. Few iterations through the above-mentioned process will make your program completely bug-free and save your time and other resources on the actual hardware.

CHAPTER 3
ALGORITHM

ALGORITHM

List of variables used

```
sbit EN = P3^7;
sbit RS = P3^6;
#define LCDDATA P2
sbit K_LEFT = P3^3;
sbit K_RIGHT = P3^4;
sbit LED_ALERT = P3^0;
sbit LED_RISK = P3^1;
sbit LED_STOP = P3^5;
sbit PULSE = P3^2;

#define LOW 0
#define HIGH 1
#define LINE1 1
#define LINE2 2

typedef unsigned char uc;
typedef unsigned int ui;

#define FRONTPAGE 0
#define SELECTPAGE 1
#define RUNPAGE 2
#define RUN 3
```

```
Step 1 : declare the functions local to the file as
         Void lcdcommand (unsigned char);
         Void displaychar ( char,unsigned char loc);
Void lcdinit( );
```

Step 2 : define functions external to the file

```
void display(uc mode)
{uc temp;
 switch(mode)
{case FRONTPAGE:   displayline("   Heart Beat   ",LINE1);
                   displayline(" Monitoring      Sys.",LINE2);
                    break;
 case SELECTPAGE:          displayline("Select Age Group",LINE1);
 displayline("15-30      31-60",LINE2);
  break;
  case RUNPAGE:            displayline(" AGE  Nrml  Cnt ",LINE1);
  if(choiceflag == FALSE)
  displayline("15-30 60-90 000 ",LINE2);
  else
  displayline("31-60 65-80 000 ",LINE2);
  case RUN :  temp = count;
  displaychar((temp%10)+48,0xce);
  temp/=10;
  displaychar((temp%10)+48,0xcd);
  temp/=10;
  displaychar((temp%10)+48,0xcc);
  break;
         }
     }
```

Step 3 : define function to display character at particular location

```
void displaychar(uc ascii,uc location)
{       if(location != CURRENT)
        lcdcommand(location);
RS = HIGH;
LCDDATA = ascii;
EN = HIGH;
delay(5);
EN = LOW;
delay(50);
 }
```

Step 4 : define function to display characters (all 16) in one line

```
void displayline(uc* character,uc location)
{       int i;
 switch(location)
{       case LINE1: lcdcommand(0x80);break;
        case LINE2: lcdcommand(0xc0);break;
                }
for(i=0;i<16;i++,character++)
{       displaychar(*character,CURRENT);
 }
}
```

Step 5 : define function to give command to the lcd

```
void lcdcommand(unsigned char command)
{       RS = LOW;
LCDDATA = command;
EN = HIGH;
delay(5);
EN = LOW;
delay(50);
 }
```

Step 6 : define function to generate time delay

```
void delay(ui t)
{
 for(;t>0;t--);
}
```

Step 7 : define the function lcdinit ( ) which is local to the file

```
void lcdinit()
{
lcdcommand(0x38);
delay(200);
lcdcommand(0x38);
delay(400);
 lcdcommand(0x01);
}
```

Step 8 : declaration of variables as :

```
Bitexperiment,displaychangeflag,choiceflag,waittimerflag,counttimerflag,
stopflag,pulse,resetflag;
uc count,wsecond,csecond,timecount,mode;
```

declaration of functions as :

```
void scankeyleft();
void scankeyright();

extern void lcdinit();
extern void delay(int);
extern void display(uc mode);

begining of main( )
 initializing values to four ports as :
 P0 = 0XFF;
 P1 = 0XFF;
 P2 = 0XFF;
 P3 = 0XFF;

 Define the variable again as
 LED_ALERT = LOW;
 LED_RISK = LOW;
 LED_STOP = LOW;
 pulse = FALSE;
 PULSE = HIGH;

 Define the variables
 resetflag = FALSE;
 experiment = FALSE;
 displaychangeflag = TRUE;
 choiceflag = FALSE;                    //BY DEFAULT 15-30
 stopflag = FALSE;
 waittimerflag = FALSE;
 timecount = 180;
 csecond = 0;
 wsecond = 0;
```

step 9 : initialization of timer

```
IE = 0x82;
TH0 = 0X00;
```

```
           TL0 = 0X00;
           timecount = 111;
           TR0 = 1;
Step 10 :  while(1)
           {    if(resetflag == TRUE)
                goto again;
                if(displaychangeflag == TRUE)
                {       display(mode);
                        displaychangeflag = FALSE;
                }
                if(experiment == FALSE)
                {       if(PULSE == LOW)
                        {       if(waittimerflag == FALSE)
                                { mode = SELECTPAGE;
                                  displaychangeflag = TRUE;
                                  wsecond = 5;
                                  waittimerflag = TRUE;
                                 }
                                scankeyleft();
                                scankeyright();
                         }
                }
                else
                {       if(stopflag == FALSE)
                        {       if(pulse == FALSE && PULSE == HIGH)
                                {       pulse = TRUE;
                                        count++;
                                }
                                 display(RUN);
                        }
                        else
                        {       LED_STOP = HIGH;
                                if(choiceflag == FALSE)
                        {       waittimerflag = TRUE;
                                if(PULSE == LOW)
                                {       wsecond = 5;
                                }
                if((count >=50 && count <60) || (count >90 && count <=100))
                                {       LED_ALERT = HIGH;
                                }
                                if(count < 50 || count > 100)
                                {       LED_RISK = HIGH;
                                }
                        }
                        else
                        {       waittimerflag = TRUE;
                if((count >=50 && count <65) || (count >80 && count <=95))
                                {       LED_ALERT = HIGH;
                                }
                                if(count < 50 || count > 95)
                                {       LED_RISK = HIGH;
                                }
                        }
                }
        }
    }
}
```

```
Step 11 :  void timer0isr() interrupt 1 using 1
           { TR0=0;
           TH0 = 0X00;
           TL0 = 0X00;
           if(timecount == 0)
           {  timecount = 111; // ONE SECOND
if(counttimerflag == TRUE)// IF TIME OUT CONDITION IS ON
{  csecond--;                //DECREMENT SECOND
  if(csecond == 0)           //TIME OUT
{   counttimerflag = FALSE;//MAKE READY FOR THE NEXT CALL
    stopflag = TRUE;
    }
 }
   if(waittimerflag == TRUE)
   {          wsecond--;
              if(wsecond == 0)
   {          resetflag = TRUE;
    }
 }
   else
   timecount--;    //NOT A SECOND , HENCE DECREMENT THE
                     INTERRUPT COUNT
   TR0 = 1;
   }
```

Step 12 :

```
void scankeyleft()
{  if(K_LEFT == LOW)
{  experiment = TRUE;
   choiceflag = FALSE;
   csecond = 60;
   counttimerflag = TRUE;
   waittimerflag = FALSE;
   count = 0;
   display(RUNPAGE);
 }
}
 void scankeyright()
```

```
{  if(K_RIGHT == LOW)
{   experiment = TRUE;
    choiceflag = TRUE; //30-60 SELECTED
    csecond = 60;
    count = 0;
    display(RUNPAGE);
  }
 }
```

CHAPTER 4
CONCLUSION

We have successfully implemented the Pulse Rate Monitor by the use of basic circuits using op-amps and shown the output on an LCD by the use of a microcontroller .This project highlighted the bio-medical application of a microcontroller and provided a cheap and simpler substitute to heart beat counting machines .The use of basic programming made it more versatile by providing choices for age and indicating the user about the normal and danger levels.

This device is very simple to use and the results are very precise ,it has no hassles of bulky circuit or complex designing ,it is very convenient to design and implement and finds it's application in our daily life ,we can check our heart beat ourselves anywhere and anytime we want and that too very conveniently.

REFERENCES

- The 8051 Micrcontroller and Embedded System by Mohd. Ali mazidi and Janice Mazidi
- C How to Program by Deitel and Deitel
- Programming Microcontrollers in C by Ted Van Sickle
- www.keil.com
- www.ikalogic.com

BIBLIOGRAPHY

- Op-Amps and Linear Integrated Circuits by Ramkant A.Gayakwad
- Exploring C for Microcontrollers by Jivan S.Parab ,Vinod G.Shelake ,Rajanish K.Kamat and Gourish M.naik
- C for microcontrollers by Dogan Ibrahim