Software Design Specifications

For

8051 Emulator

Version 1.0

Last Updated: June 30, 2013

Under the guidance of

Prof. Deepak B. Phatak

Principal Investigator

Project In-charge:

Dr. Madhuri Sawant

Mentors:

Pankaj Patil

M. Barani

Group Name: 8051 Emulator

T.M.S.K. Alekhya

Prayasee Pradhan

Priyanka Padmanabhan

Sumanth Myrala

Anupam Sanghi

Sweksha Tripathi

Department of Computer Science and Engineering IIT Bombay, Powai

Acknowledgment

We the summer interns of batch 2013 of Aakash R&D Lab, like to thank for Prof D.B.Phatak for giving us an opportunity to work on such a noble project. We want to express our gratitude towards Dr. Madhuri Sawant for her valuable guidance and support. We sincerely thank our mentor Mr. Pankaj Patil for constant motivation and support to put up our ideas into reality. We are overwhelmed to acknowledge his dedication and humbleness. Finally we like to thank all IIT Staff our colleagues for helping us at critical junctures and made things possible.

Abstract

Embedded systems designer deals with circuit designing, embedded programming and implementation of system in a real time environment. The 8051 family of microcontrollers are fundamental controllers used in a wide range of real time embedded systems. Embedded systems development requires various resources, which include circuit designing software, embedded programming tools and hardware resources.

8051 Emulator application is an integrated development environment bringing together a variety of resources required for embedded systems. Intended audience for emulator are students learning embedded systems specifically 8051 microcontroller.

Students can use 8051 Emulator for learning 8051 microcontroller. This application facilitates the student to interface different circuits on ports of the microcontroller, write assembly code as per requirements and execute code written. After execution of code, virtual hardware gives behavioural animations in close resemblance to real 8051 Board. This application helps students to learn basics of 8051 microcontroller without buying actual 8051 development board.

Contents

List of figures

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed design description of 8051 Emulator application.

1.2 Scope

The scope of this document is to provide a detailed description about the implementation, design and architecture of the application. This document explains the detailed description of the project, explained with the help of different class, data flow and sequence diagrams that aptly describes the flow of the application with all the integrated modules.

1.3 References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Design Specifications*.

IEEE Computer Society, 1998

Technical Reference:

1. The 8051 microcontroller and embedded systems by Mazidi, Edition II, Pearson Education

2. The 8051 Microcontroller by Kenneth.J.Ayala, Edition III, Thomson, Delmar Learning.

3. Android Developers Website- developers.android.com

1.4 Overview

The project basically comprises of three modules i.e. the Workbench, the Assembly Editor and the Internals 8051. All the three modules are connected and functions collectively, integrated with all the basic functionalities and workflows. These modules help the user to write the assembly code corresponding to a circuit that the user designs on the Workbench. The execution of the assembly code displays the animation on the workbench like the glowing of the Led Circuit, and the display on the seven segment display etc. along with setting the corresponding values of the registers, program counter etc. in internals registers of 8051.

This application helps the user to have a better understanding and knowledge about 8051 Microcontroller without using real hardware. It hence eliminates the high costs involved in purchasing and maintaining the hardware.

2. Motivation

The basic motivation behind this project is wide application and importance of the 8051 family of microcontrollers. The application is of great help for students of ECE and CSE branches and helps them to understand the basic functionality of the 8051 Microcontroller.

This application is a virtual environment to study 8051 microcontroller, eliminating the high costs involved in the case of hardware. The application can help the user to view the output and the results of experiments related to 8051 Microcontroller. The striking feature is that the output appears in the form of animation and thus makes understanding easier. Apart from the workspace for assembly coding and the animation of virtual hardware, the user can also view the values of the Internals 8051 which gets updated according to the assembly code written by the user.

Thus the basic motivation behind the application is its wide usability in the field of education. It provides the user with a basic level of practical knowledge about the 8051 Microcontroller.

3. Data Consideration

3.1 Assumptions

There are a few assumptions that have been made for the proper functioning and useability of the application.

These assumptions are as follows:

1. The user is restricted to use only the inbuilt circuits that are provided in the circuit panel. The application has not been provided with an option to design the user defined circuits.

2. The application assumes and considers that it is the responsibility of the user to mount the correct device on correct port of the microcontroller.

3. The application assumes that the user will not use the features like timers and external interrupts and serial communication.

3.2 Constraints

The application is designed by the following constraints:

3.2.1 Design constraints

The application is user friendly and helps the user to view the desired output depending on the circuit built on the workbench. But the application does not permit the user

a) To change the brightness of the LED that glows corresponding to a high output value.

b) To change the colours of the animations.

c) To access all the possible circuits on all the ports under certain circumstances.

3.2.2 Functional Constraints

The application even has some functional constraints, like:

a) The application only includes the assembly instructions of MCS - 51 family.

b) The user can execute the assembly code written, only from the WorkBench module.

3.3 System Environment

The application is designed for the Aakash tablet with Android version 4.0.3 (Ice-cream sandwich). The application is built for the resolution 960*510.

3.4 Design Methodology

The system is designed with flexibility for further development and/or modification. The system is divided into manageable processes that are grouped into sub-modules and modules that are built with abstraction. The system is designed such that it is scalable, flexible and user friendly.

4. Architecture

4.1 System Design

4.1.1 Architecture Diagram

The Architecture diagram below shows the principal parts of the system and their interactions.

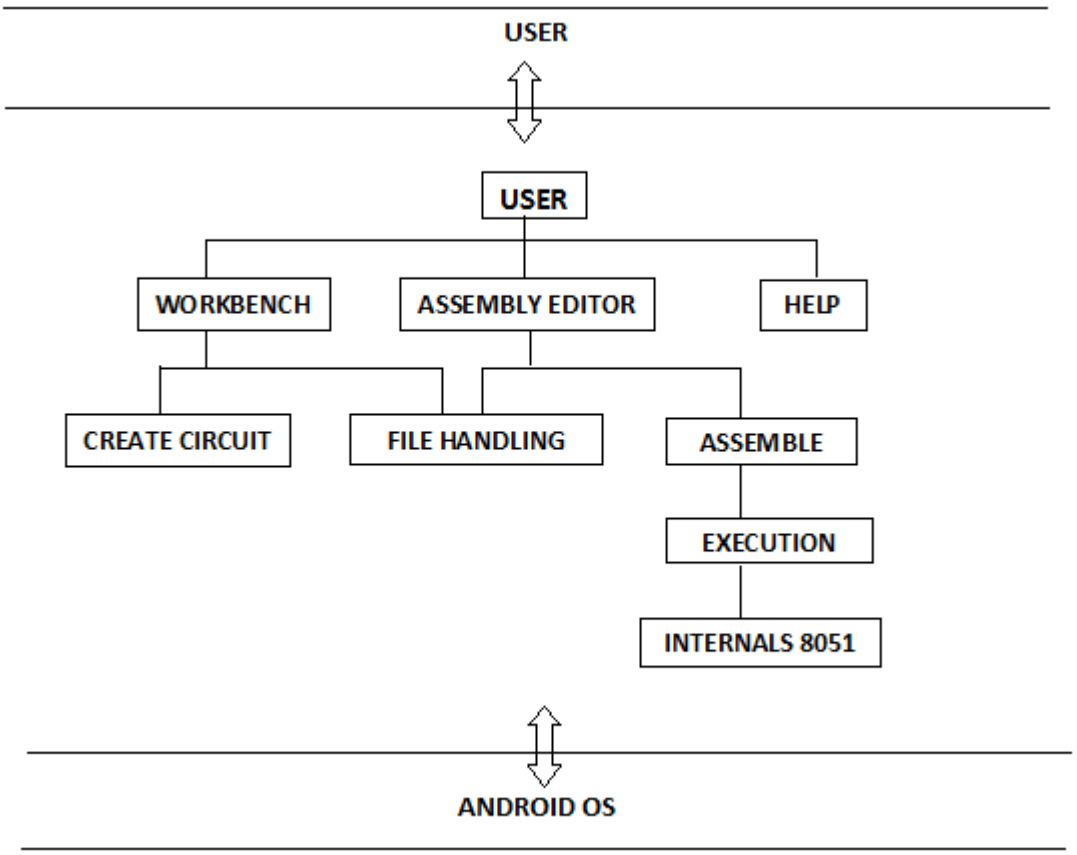

Fig. 4.1.1 Architecture Diagram

4.1.2 Context Diagram

The Context diagram shows actors interaction with the system.

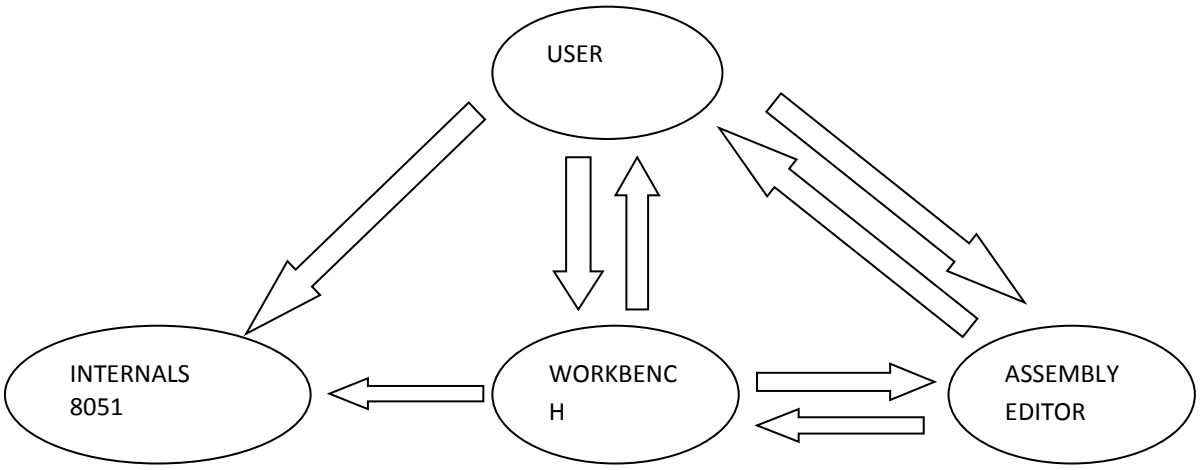

Fig.4.1.2 Context Diagram

The context diagram clearly depicts the major modules involved in the application, i.e. the workbench, the assembly editor, the internals 8051 and the user. It pictures the interaction of user with the modules and even the dependencies of different modules on each other.

4.1.3 Data Flow Diagram

The Dataflow diagram of the 8051 Microcontroller Emulator Application is as follows:

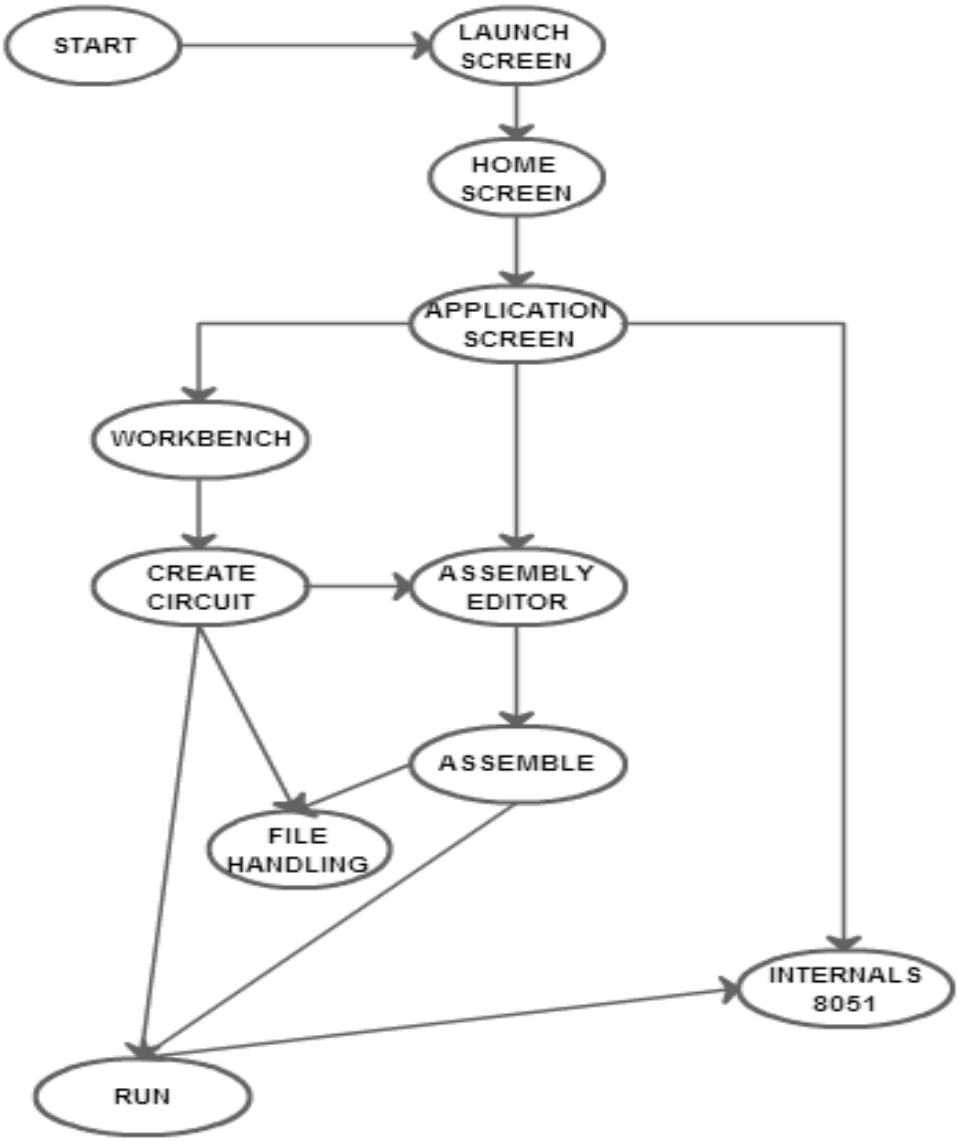

Fig. 4.1.3 Data Flow Diagram

The dataflow diagram above portrays the flow of the application. The first screen that appears on starting the application is the launch screen, followed by the home screen having the introduction of the application.

Next is the application screen which includes three main modules of the application along with an option of help.

1. **Workbench:**

   a) Facilitates the user to build circuits.

   b) Facilitates the user to execute the assembly code.

   c) Facilitates the user to view the graph generated on the oscilloscope.

2. **Assembly Editor**

   Facilitates the user to write the assembly code in the editor, displays the error log corresponding to the written assembly code. It displays the ROM contents generated, after the written code is assembled.

3. **Internals 8051**

   This interface provides the user the values of the registers of the internal RAM.

The application also facilitates file handling i.e. the options to save an existing project, open an existing project, create a new project and delete a project.

The diagram showing the file handling part of the application is as follows:

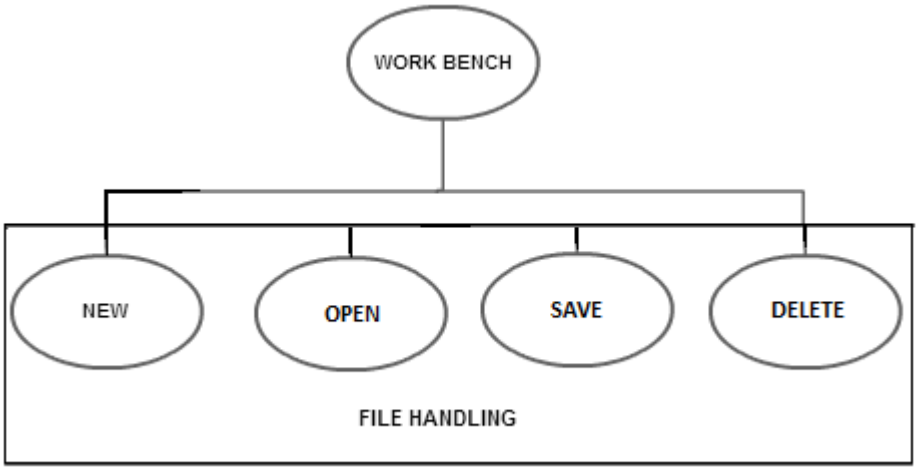

Fig.4.1.4 File handling diagram

4.2 Modelling 8051 Microcontroller

The 8051 is an 8-bit micro-controller. This micro-controller is capable of addressing 64kB of program memory and 64kB of data memory.

The 8051 memory architecture includes 128 bytes of data memory that are directly accessible by its instructions. A 32-byte segment of this 128 byte memory block is addressable by a subset of the 8051 instructions, namely the bit-instructions. External memory of up to 64kB is accessible by a special "move" instruction. Up to 4kB of program instructions can be stored in the internal memory of the 8051, or the 8051 can be configured to use up to 64 Kbytes of external program memory. The majority of the 8051's instructions are executed within 12 clock cycles.

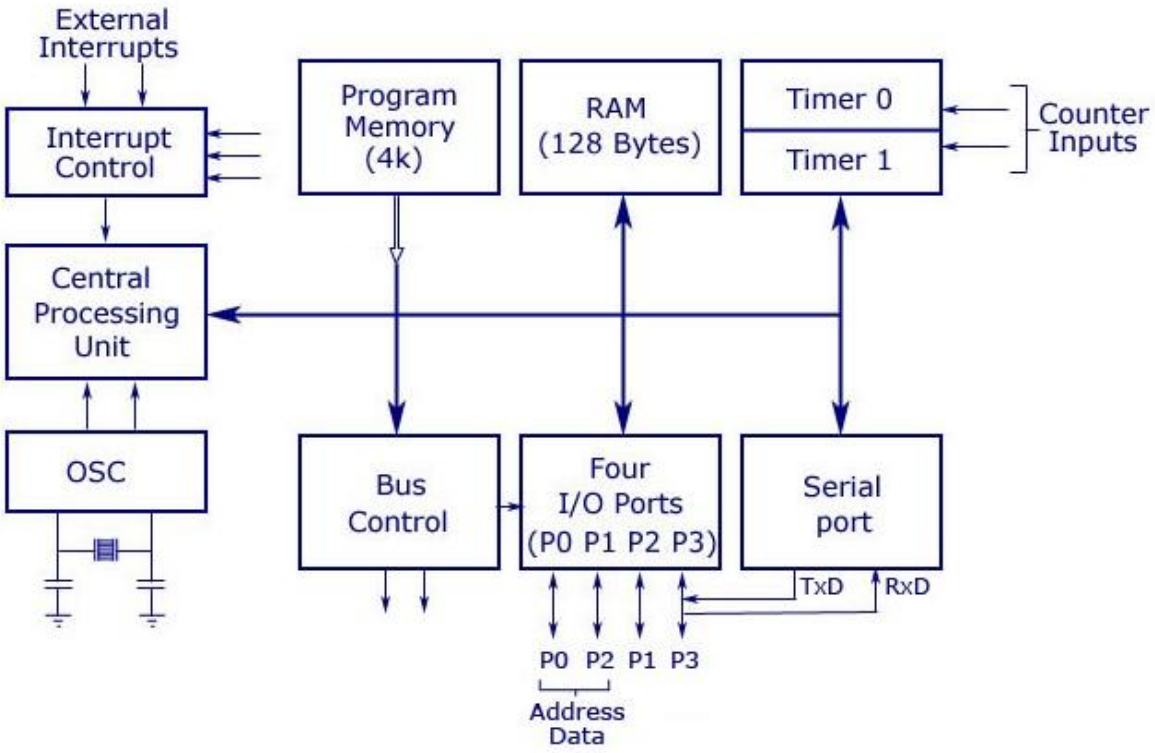

Fig.4.2 Block diagram of 8051 microcontroller

As seen in the figure above, the 8051 microcontroller has the following:

- 4 kB of ROM.
- 128 bytes of RAM (including SFRs) satisfy the user's basic needs.
- 4 ports having a total of 32 input/output lines are in most cases sufficient to make all necessary connections to the peripheral environment.

The mapping from hardware to software has been done in the following ways:

1. The representation of internal 4kB ROM of program memory has been modelled as a String Array “rom” of length 4096 units.

2. The internal RAM of 128 bytes consisting of register banks, bit addressable area and general purpose area is modelled as the first 128 units of the String Array “ram” of 256 units.
   The detailed representation is:
   a) First 32 bytes – Register Banks
   b) Next 16 bytes – Bit Addressable Area
   c) Last 80 bytes – General Purpose Area

3. The next 128 units represent the special functions registers such as ports, accumulator, register B, Program Status Word (PSW), interrupts etc.

4. The flags affected during the arithmetic operations or due to the content of the accumulator, have been handled directly through the corresponding bit values of the PSW (Program Status Word), which is the special function register lying in the second half of the 256 unit String Array RAM.

5. Implementation

5.1 Circuit Design

This module depicts the various circuits included in the circuit panel. These circuits have been designed in Abode Photoshop CS6.

5.1.1 Microcontroller and input circuits

The Microcontroller and the power, clock, pull up and reset circuit are designed as follows:

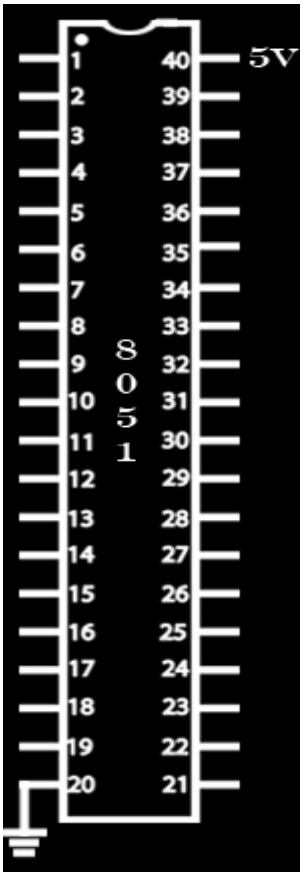

Fig.5.1.1.1 8051 microcontroller

Clock input, Power input and Pull-Up and Reset circuits is as follows:

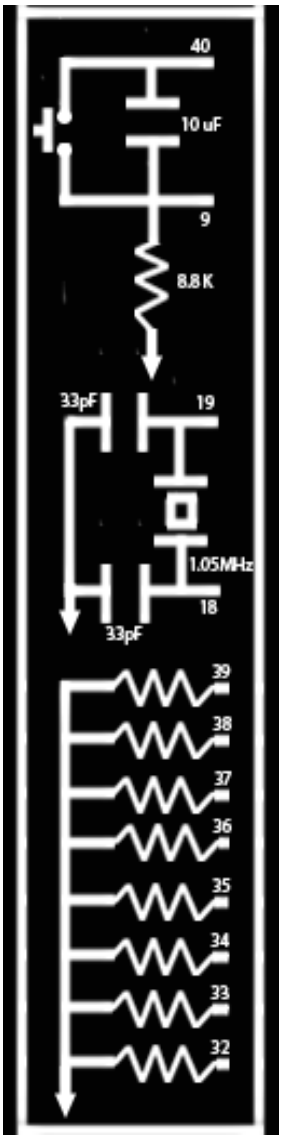

Fig.5.1.1.2 Clock input, power input, pull up and reset circuit

It is assumed that the clock, power, reset and pull up resistor circuits are by default mounted on the respective pins of the microcontroller. For sake of simplicity, these circuits have been placed on the extreme left, away from the microcontroller.

1. **Reset Circuit**:

The reset circuit which is connected to pin 9 and pin 40, resets the internal contents i.e the RAM values of the microcontroller. It resets the Program Counter (PC) value to 0000H, the Stack Pointer (SP) value to 0007H, the port values to 00H and the other registers are also reset to 00H.

2. **Crystal Oscillator circuit**:

The crystal oscillator circuit which is connected to pin 18 and pin 19, provides the microcontroller a fixed clock input at a rate of 12 MHz frequency.

3. **Pull up circuit**:

A pull-up circuit is connected externally to port 0 for it to behave as an output port since it does not have an inbuilt pull-up circuit. The pull up resistors circuit restricts the floating of pin values between high and low logic levels and sets to 5V.

5.1.2 Interface Circuits

Interface circuits are designed in the .png format.

1. **Seven Segment Display Circuit**

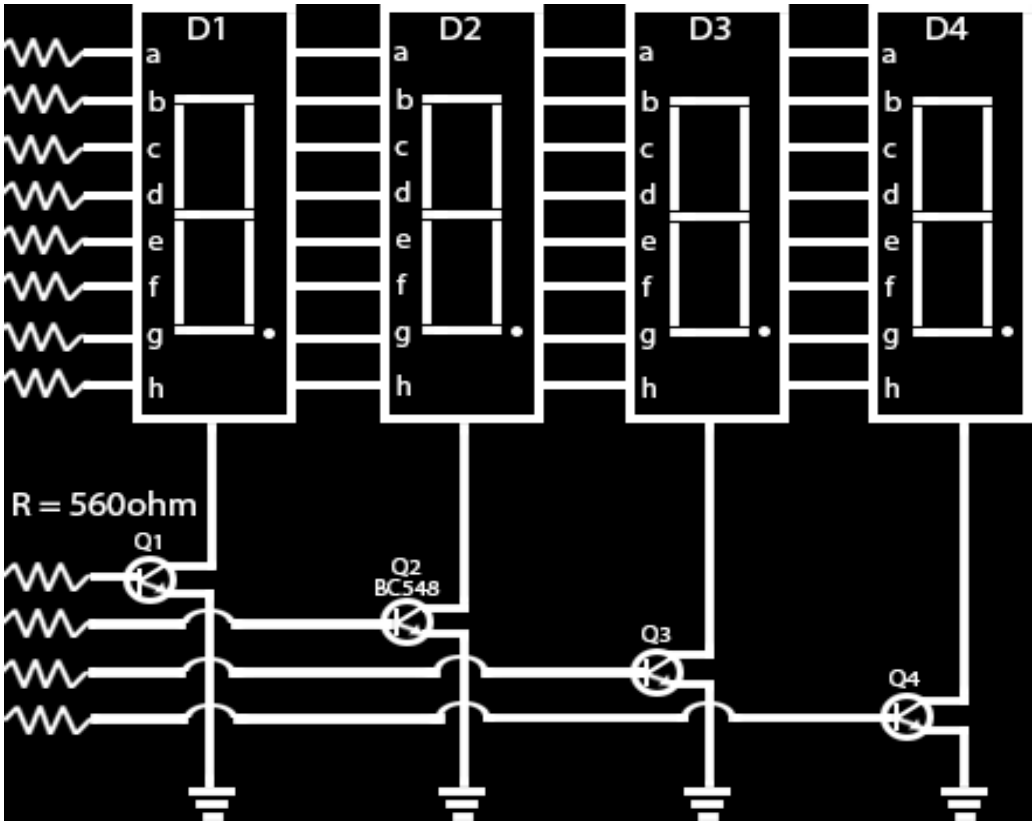

Fig.5.1.2.1 Seven segment display circuit

2. **LED Circuit**

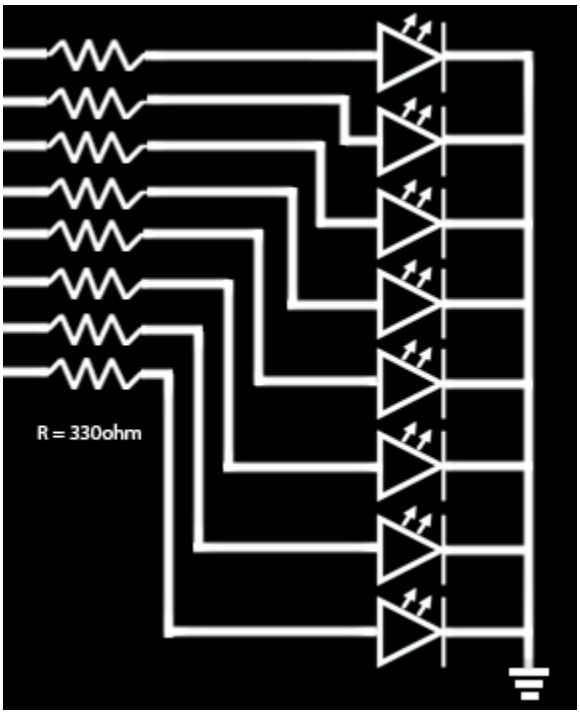

Fig.5.1.2.2 LED circuit

3. **Digital to Analog Convertor Circuit**

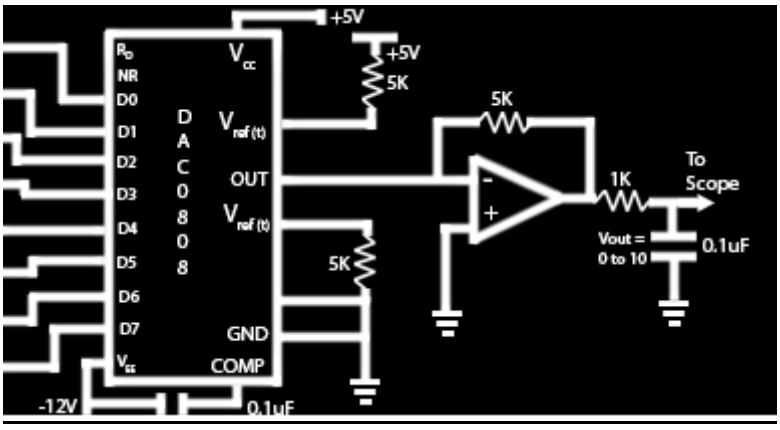

Fig.5.1.2.3 DAC circuit

4. **Stepper Motor Circuit**

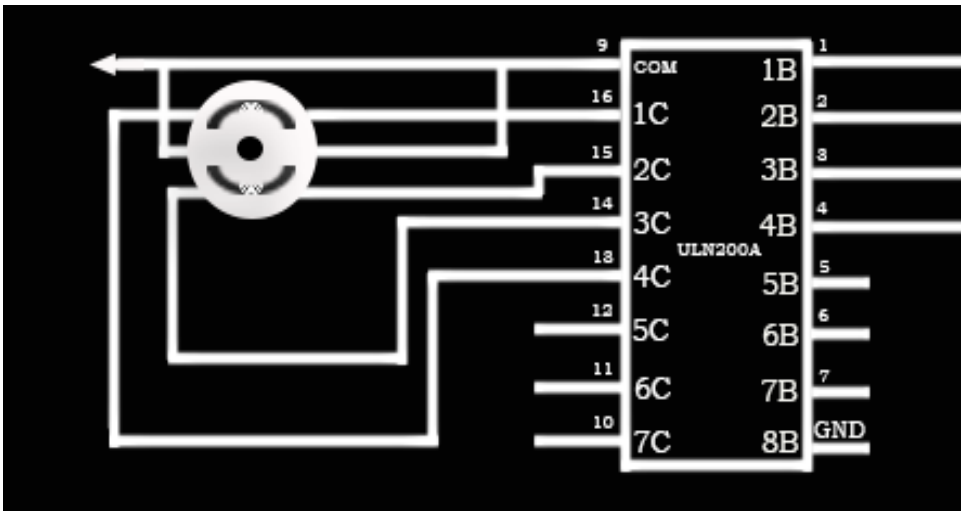

Fig.5.1.2.4 Stepper motor circuit

Thus, all the created circuits were placed in the circuit panel.

These circuits provide the user with general information about the structure of the microcontroller and the power, clock and pull-up circuits which are attached to it.

5.2 Workbench Module

This Module is the main user interface of the application. It provides the user with an option to:

a) Create the desired circuit from the circuits provided in the panel.

b) Run the program and view the final output of the corresponding assembly code in the form of animations.

c) View the voltage-time graph that is generated when one of the ports is attached with the DAC Circuit.

d) Perform the various file handling tasks like saving the existing project, opening previously saved project and opening a new project.

5.2.1 Create circuit panel

The create circuit panel displays a pop-up menu wherein, the user selects specific circuit from a dropdown list, congaing list of all the circuits provided in the circuit panel for each of the ports. The user can choose the circuit which it would want to attach to port 0, port 1, port 2 and port 3.

The different design aspects have been considered while designing dialogue box, like the Seven Segment Display Circuit requires 2 ports of the microcontroller i.e. it utilizes the 10 pins, so 2 ports get engaged. Thus if the user selects Seven Segment Display Circuit in any one of the ports, then the adjacent port is disabled.

5.2.2 Oscilloscope

The Oscilloscope function facilitates the user to view the V vs T graph generated if the DAC circuit is mounted on port of the microcontroller. The generation and formation of the graph helps the user understand the concept behind digital to analog conversion.

5.2.3 Execute

This key function facilitates the execution of the assembled code, and thereby enables the user to view the results in the form of animation on the workbench depending on the circuit mounted and assembly code written. The registers in the Internals 8051 are also updated accordingly at runtime.

5.3 File handling:

This functionality includes options like new, save, open and delete project. It caters to all the file handling cases and thus helps the user to maintain a proper record of the saved projects.

The following cases have been considered in file handling module:

5.3.1 New

1. As soon as the user clicks on the new option, the system checks whether the existing project is saved or not as performed previously.
2. Based on this response the existing project is saved and a new workbench is loaded.

5.3.2. Open

1. When the user clicks on the open option, there would be a prompt to ask the user whether the user wishes to save the existing project.
2. If “Yes”, the project is saved and the list view appears, else the list view directly appears displaying all the saved projects.
3. The project selected by the user is displayed on the workbench and assembly code written for the same project appears in the assembly editor.

5.3.3 Save

1. When the user selects ‘save’, application checks whether the project is a fresh project or not.
2. If it is a new project the user is prompted to enter the name of the project and thereby the project with the given name is saved.
3. If the file name entered by the user already exists then the user would be asked, whether user wish to replace the existing project.
4. If user selects “Yes” the project is replaced and in case of “No”, the user needs to enter a new name for the project.
5. If it is not a fresh project, a dialog box with the previously saved name appears.
6. If the user enters a new name then a project with the entered name is saved else the same project is re-saved.

5.3.4 Delete

1. When the user clicks on the open option, there would be a prompt to ask the user whether the user wishes to save the existing project.
2. If “Yes”, the project is saved and the list view appears, else the list view directly appears displaying all the saved projects.
3. On long press of any object in the list view, a prompt appears asking the user whether the selected project has to be deleted.
4. On “Yes”, the project is deleted.

5.4 Assembly Editor Module

The Assembly Editor Module of the application facilitates the user to write the assembly code. The editor provides the user a workspace to write the assembly instructions.

On selecting the Assemble button, any syntactical errors in the code are displayed along with the line number in the error log.

If the code is error free, assembling of the code is successful and the ROM contents of the 8051 are updated and displayed in the ROM Contents. The total ROM Usage is also specified in bytes.

The editor supports all the instructions belonging to assembly language of 8051 Family. The types of instructions are as follows:

a) **Arithmetic Instructions**, such as ADD, ADDC, SUBB, etc.

b) **Branch Instructions**, such as ACALL, AJMP, LJMP, JMP, etc.

c) **Data Transfer Instructions**, such as MOV, XCH, etc.

d) **Logic Instructions**, such as ANL, ORL, etc.

e) **Bit-oriented Instructions**, such as CLR, SETB, etc.

5.5 Internals 8051

Internals 8051 is an interface that displays the values of the various Special Function Registers (SFRs) of the internal RAM. These values are continuously updated during the execution of the assembly code based on the assembly code written.

The registers displayed in 8051 internals are as follows:

a. CPU registers, accumulator A and Register B

b. 4 Register Banks, each containing 8 registers R0 to R7. These registers are displayed depending on the register bank selected in the PSW.

c. Data Pointers

1. Program Counter(PC)
2. Stack Pointer(SP)
3. Program Status Word(PSW)
4. Data Pointer Higher Byte(DPH)
5. Data Pointer Lower Byte(DPL)

d. Ports, 4 ports P0 to P3 and both the value of the port and the type of port are displayed.

e. Timers and Counters

1. Timer 1 Higher and Lower Byte(TH1 AND TL1)
2. Timer 0 Higher and Lower Byte(TH0 AND TL0)
3. Timer/Counter Mode Control(TMOD)
4. Timer/Counter Control(TCON)

f. Interrupt Control

1. Interrupt Enable Control(IE)
2. Interrupt Priority(IP)

6. Design Structure and Representation

6.1 Dataflow Diagram

The Dataflow Diagrams of different modules of the project is as follows:

6.1.1 Level 0 DFD

Level 0 DFD/Context-level DFD

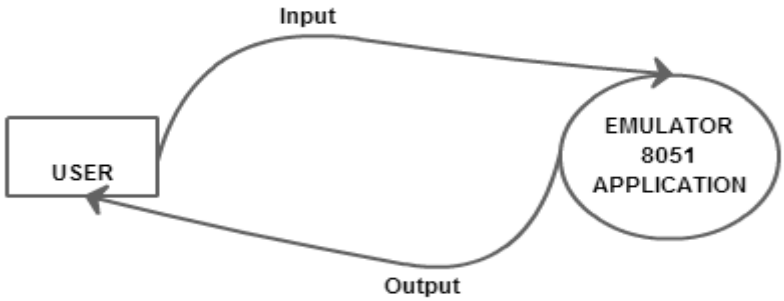

Fig. 6.1.1 Level 0 DFD

6.1.2 Level 1 DFD

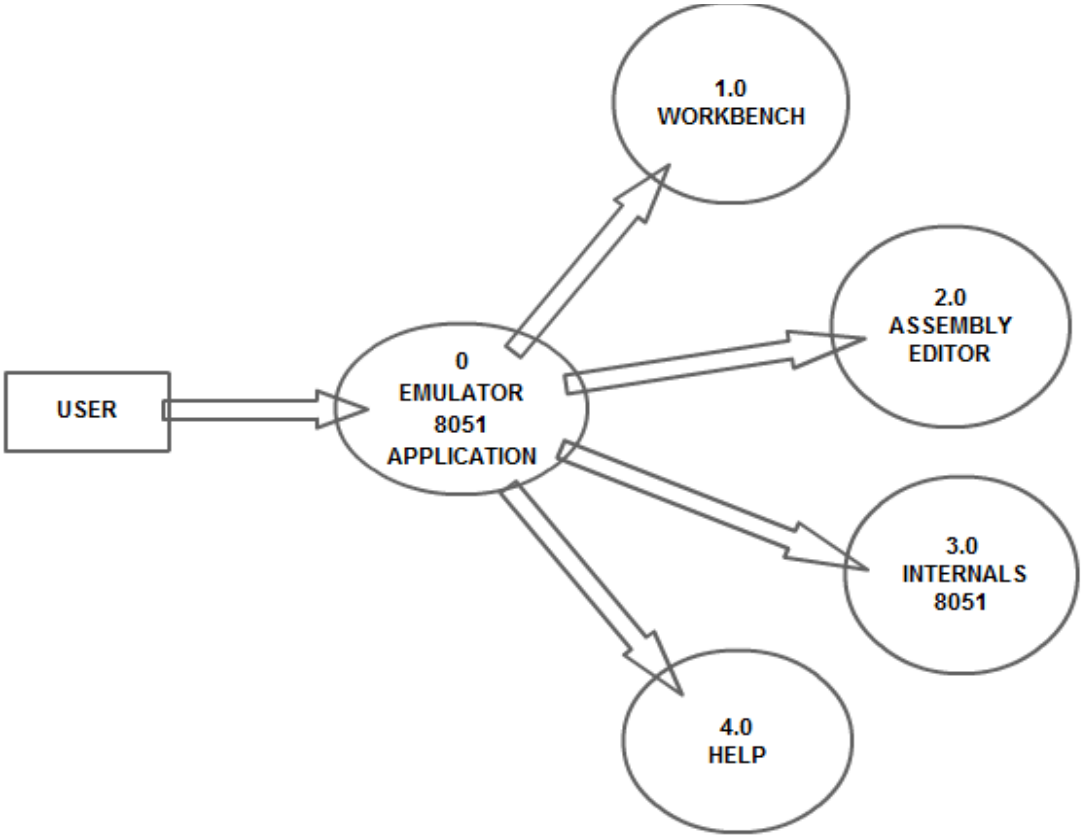

Fig.6.1.2.1 Level 0 DFD

The detailed explanations of the above modules through dataflow diagrams are as follows:

Workbench Module:

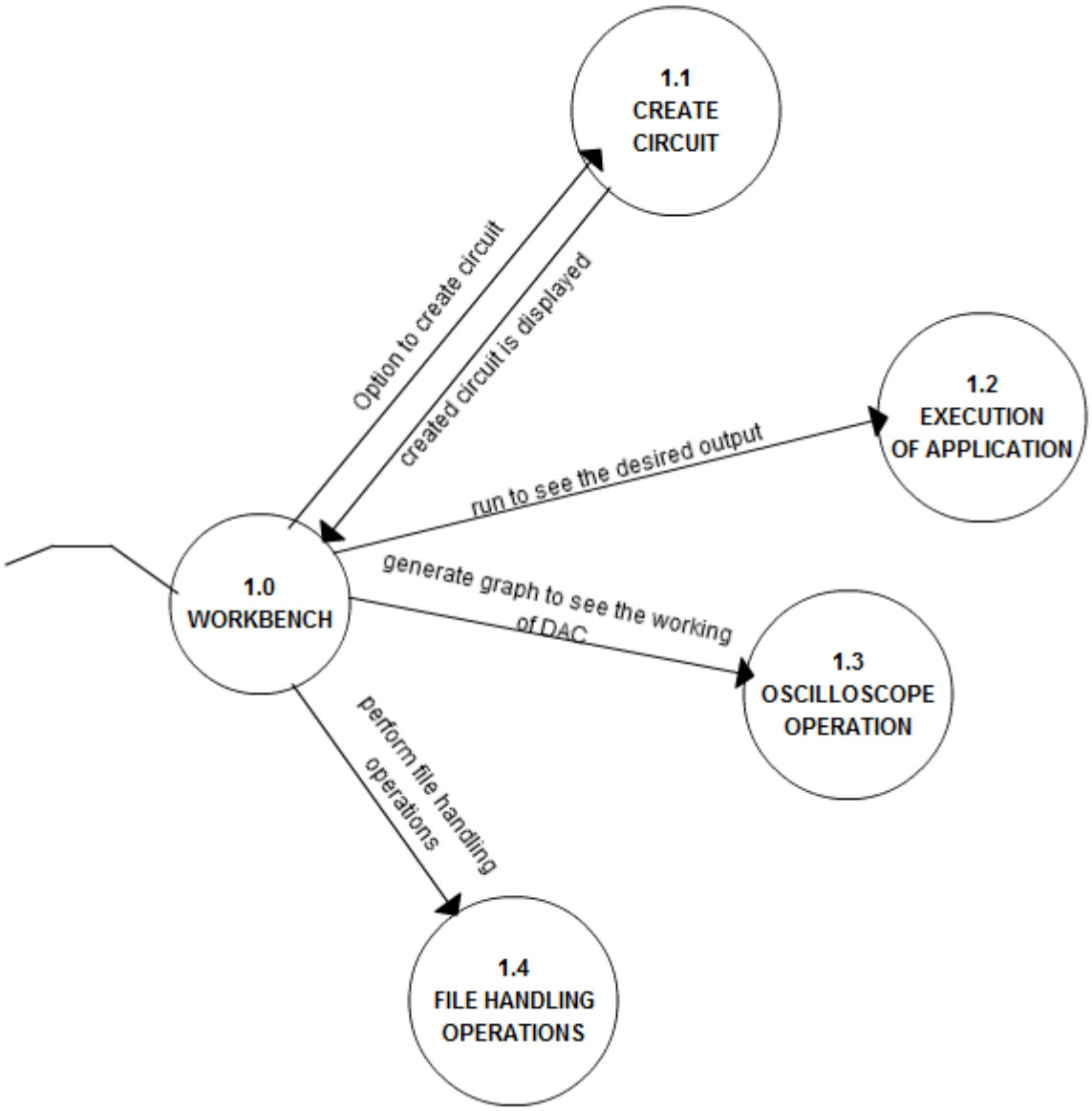

Fig.6.1.2.2 Level 1 DFD-Workbench

Assembly Editor:

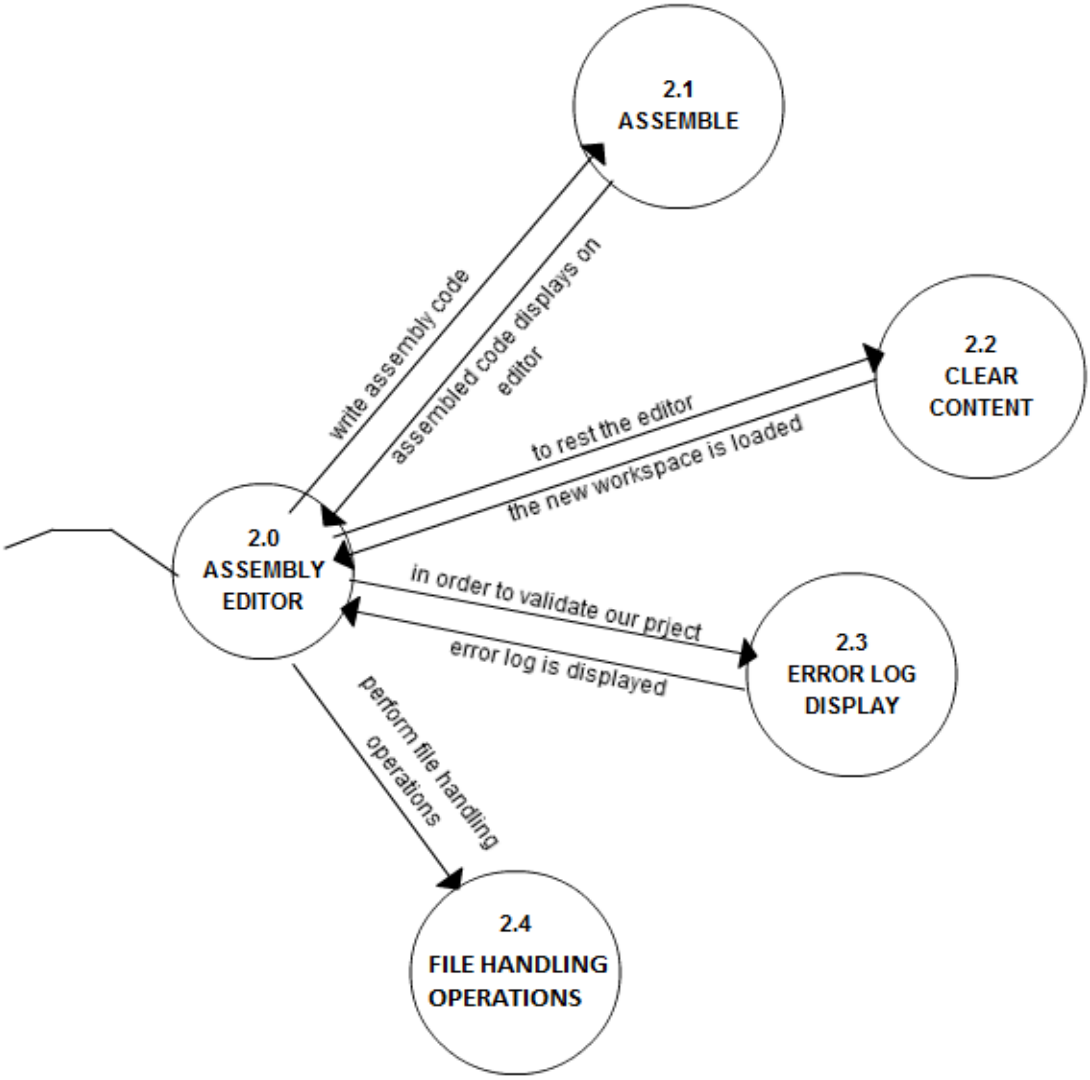

Fig.6.1.2.3 Level 1 DFD-Assembly editor

Internals 8051:

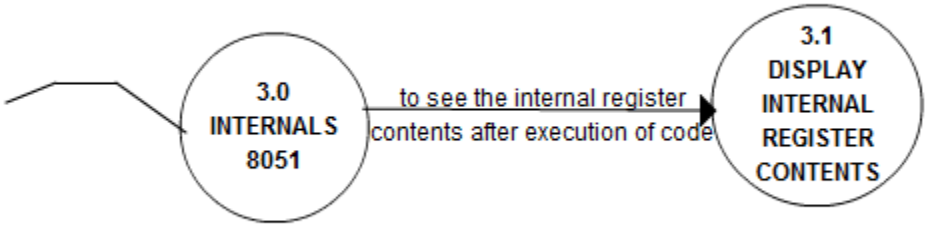

Fig.6.1.2.4 Level 1 DFD-Internals 8051

Help:

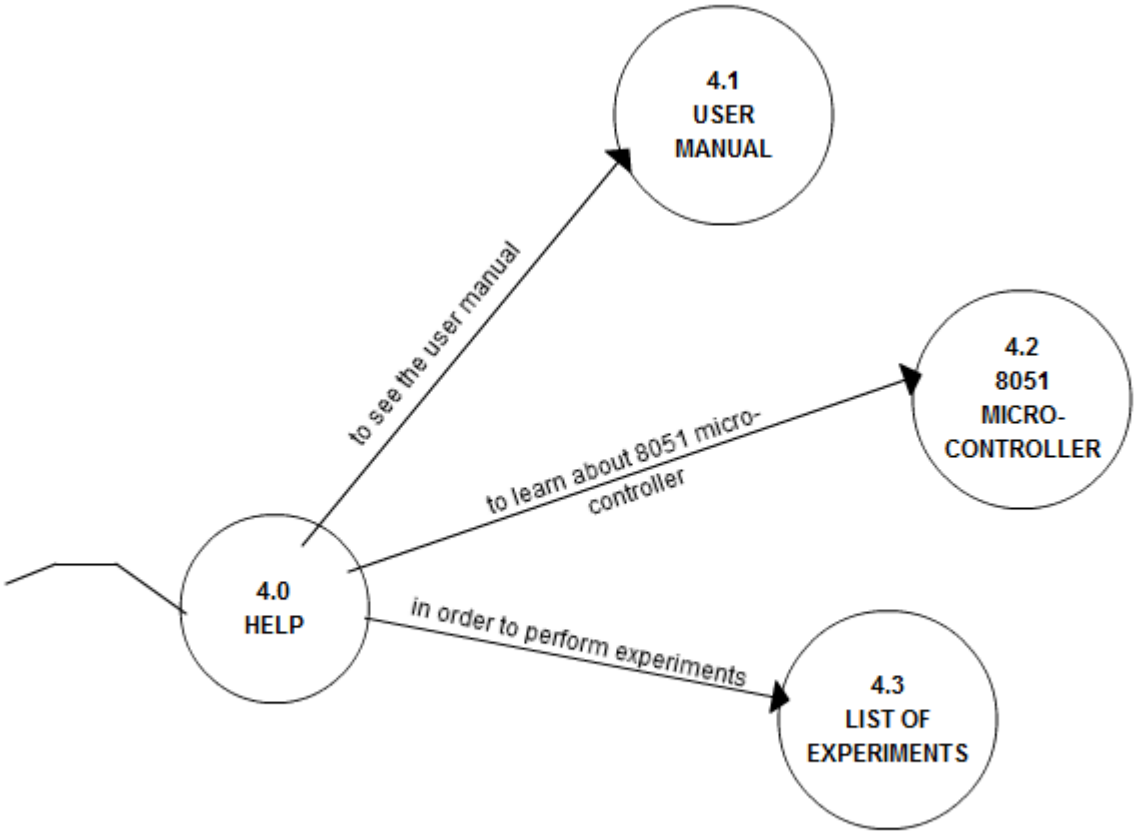

Fig.6.1.2.5 Level 1 DFD-Help

6.1.3 Level 2 DFD

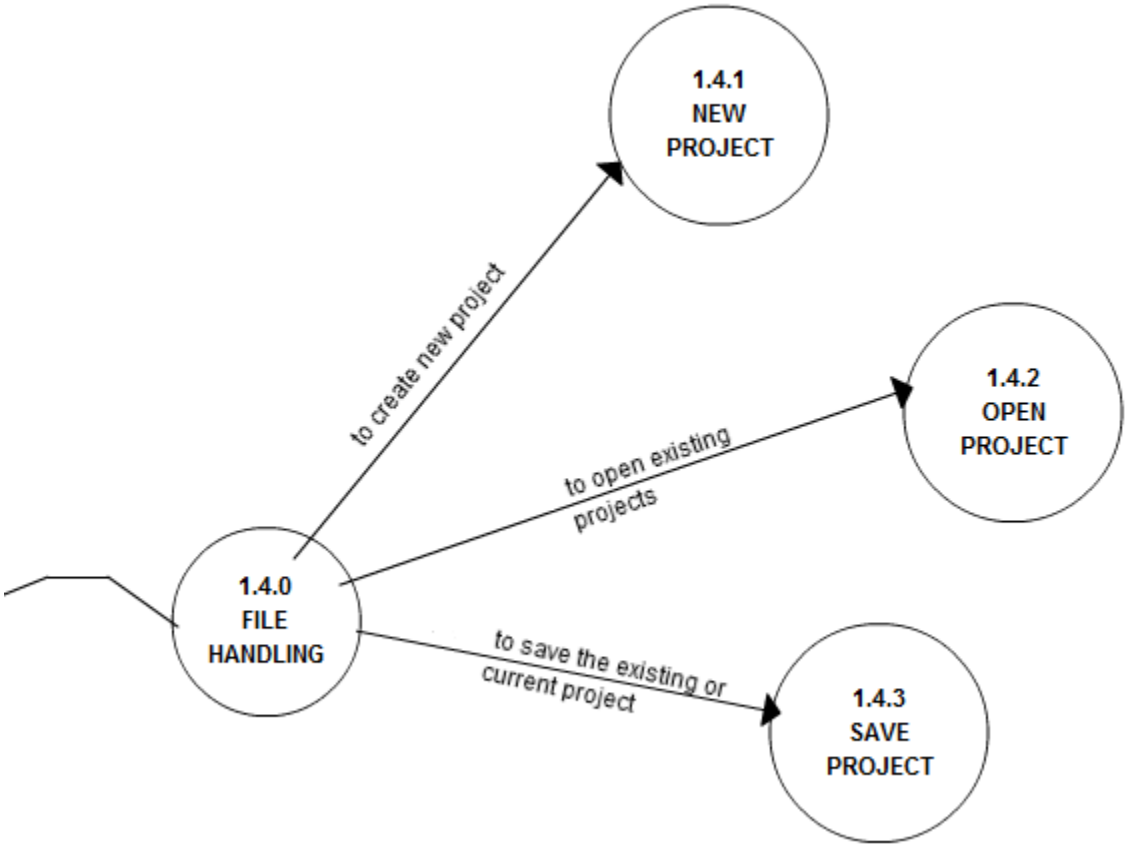

Fig.6.1.3 Level 2 DFD

6.1.4 Level 3 DFD

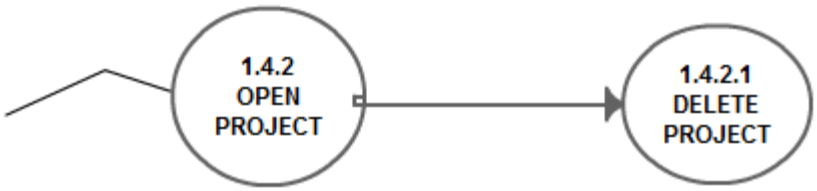

Fig.6.1.4 Level 3 DFD

6.2 Activity Diagram

Activity diagrams, the graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagram of the application is as follow**s:**

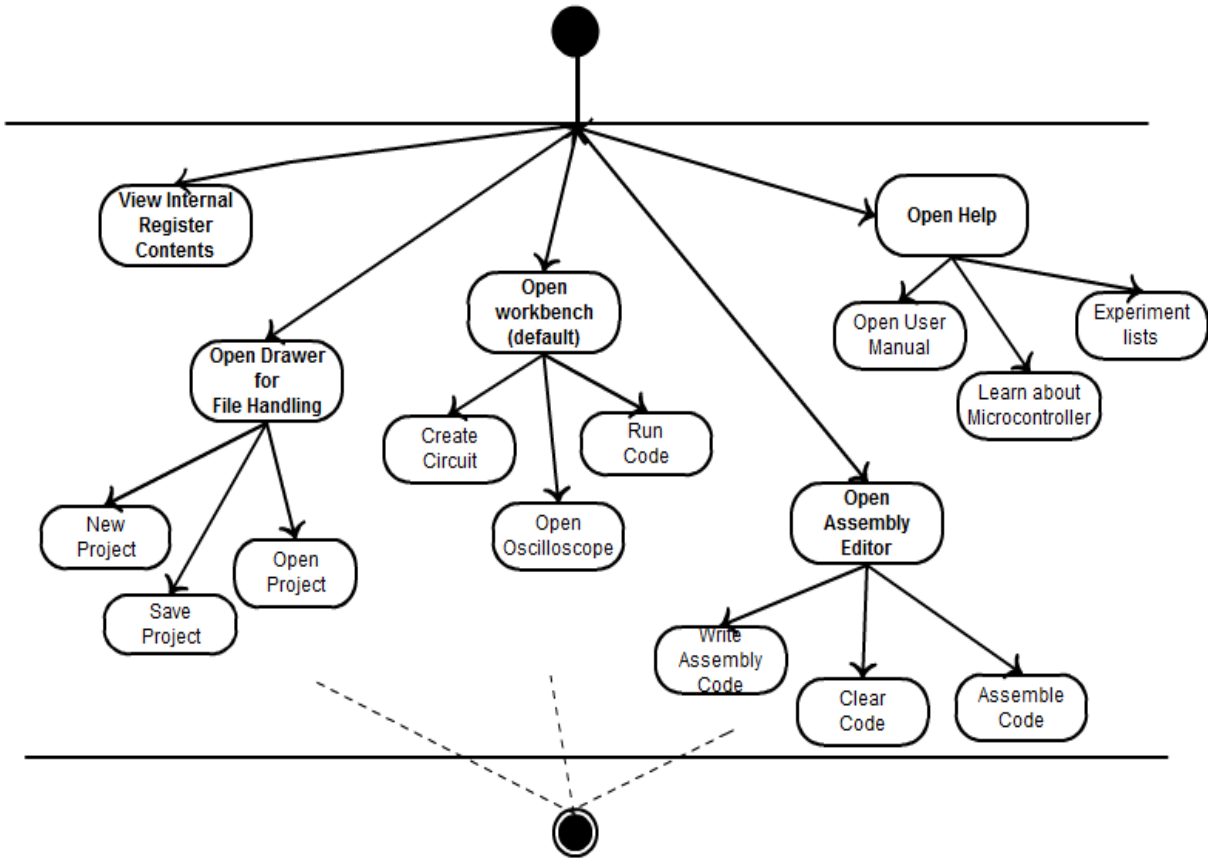

Fig.6.2.1 Activity Diagram

Now, the detailed activity diagram of the modules above has been described below:

Workbench:

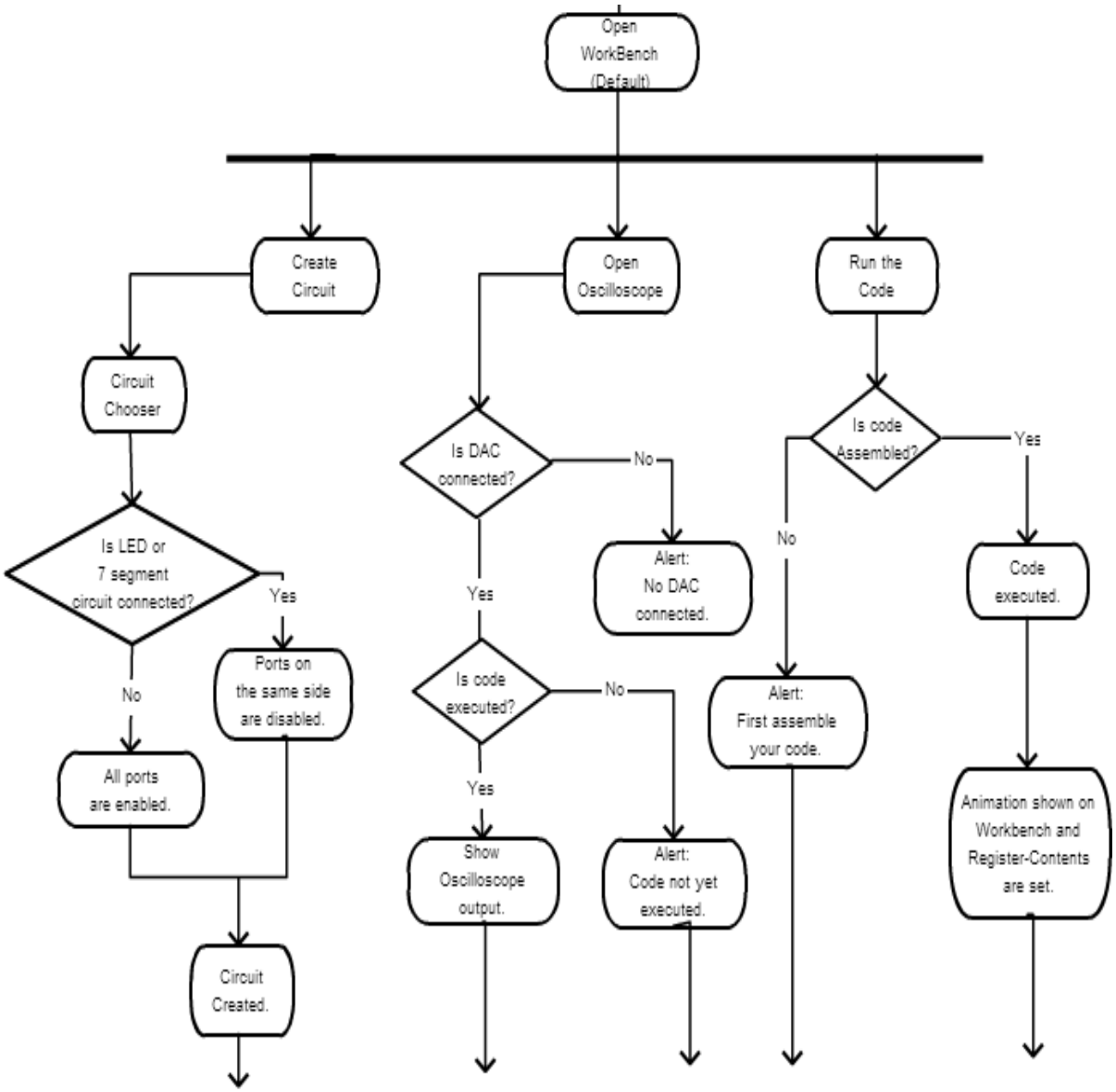

Fig.6.2.2 Activity diagram-WorkBench

Assembly Editor:

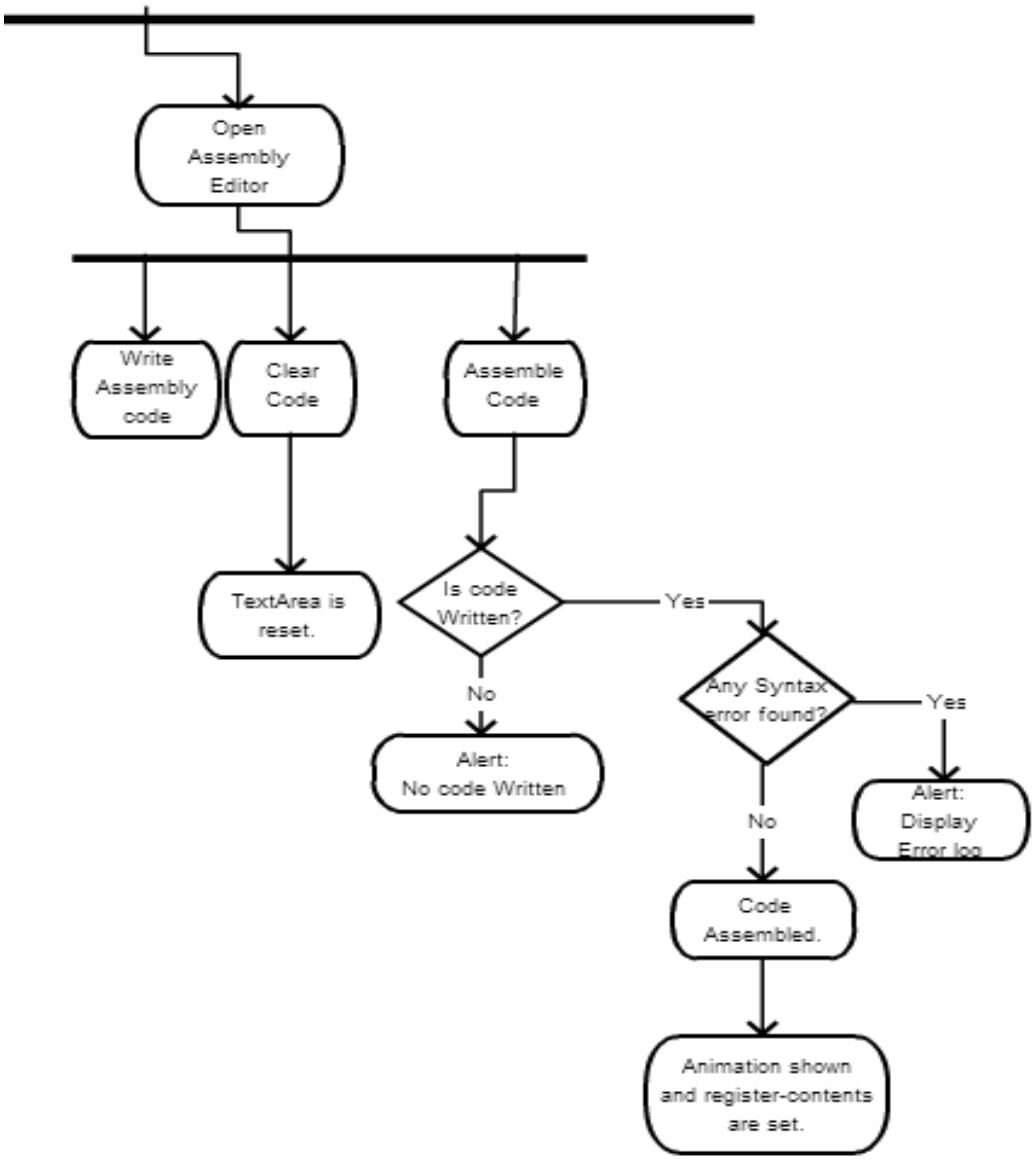

Fig. 6.2.3 Activity diagram-Assembly Editor

File Handling:

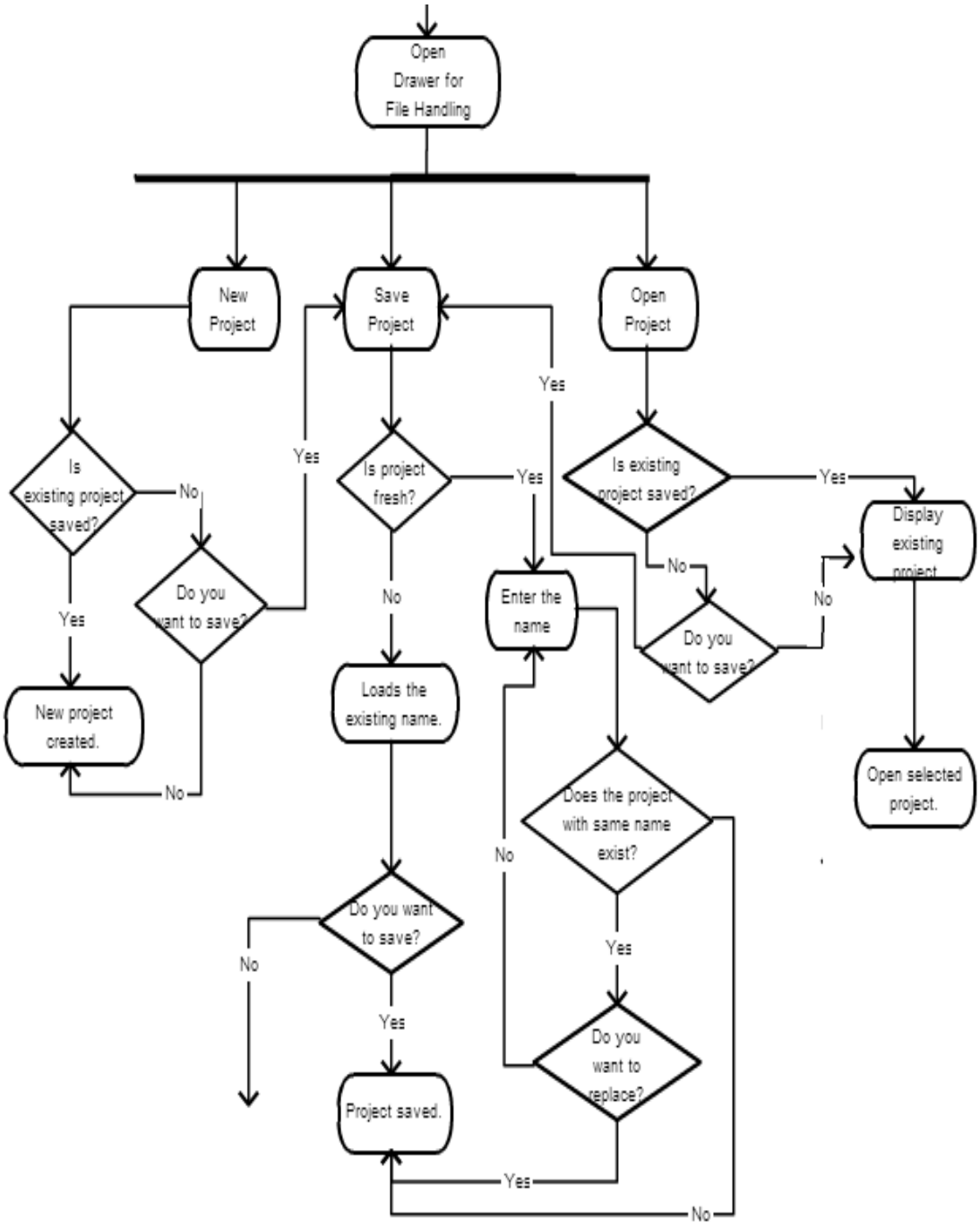

Fig. 6.2.4 Activity diagram-File handling

6.3 Class Diagram

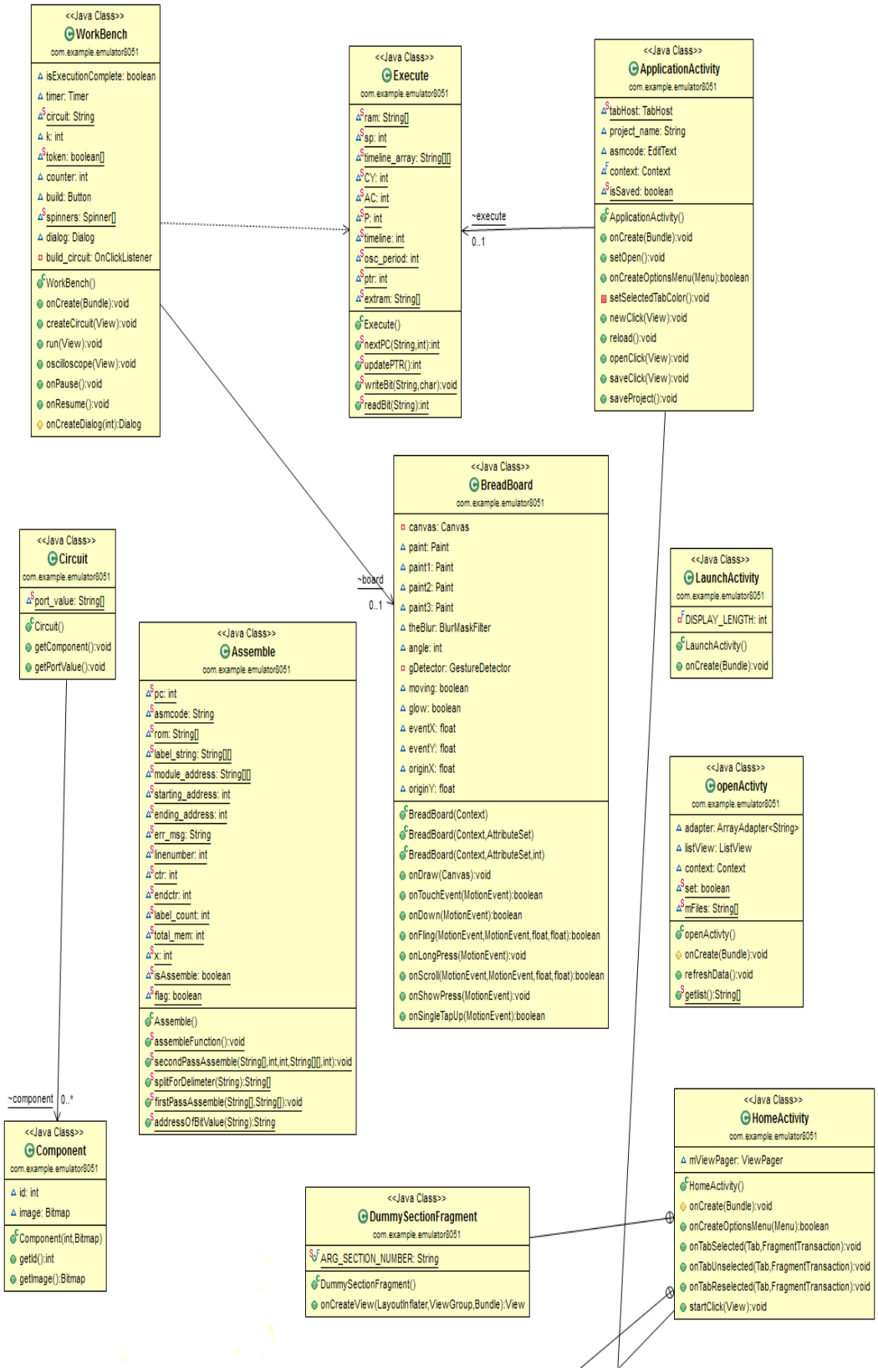

Fig.6.3.1 Class diagram

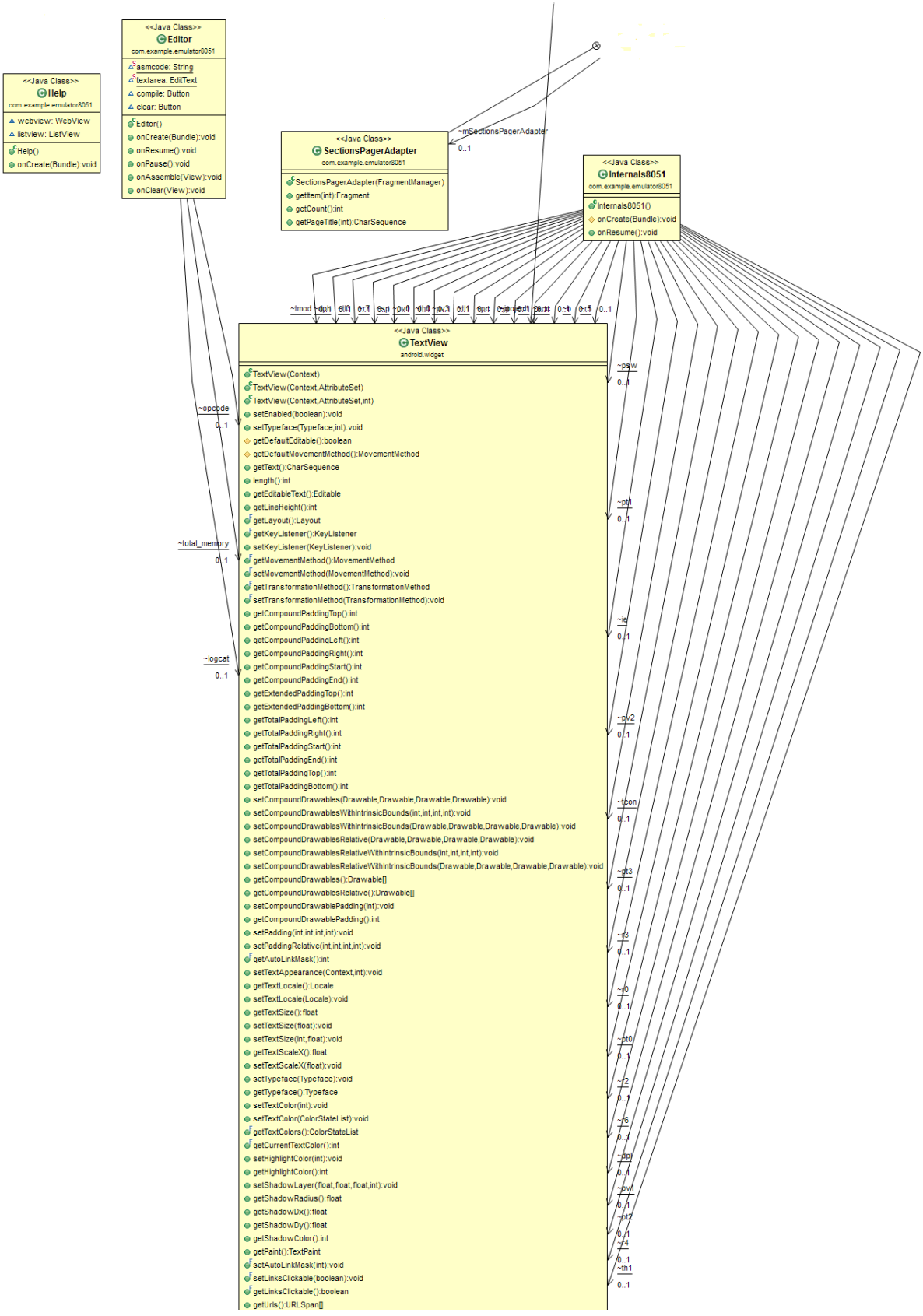

Fig.6.3.2 Class diagram

6.4 Sequence Diagram

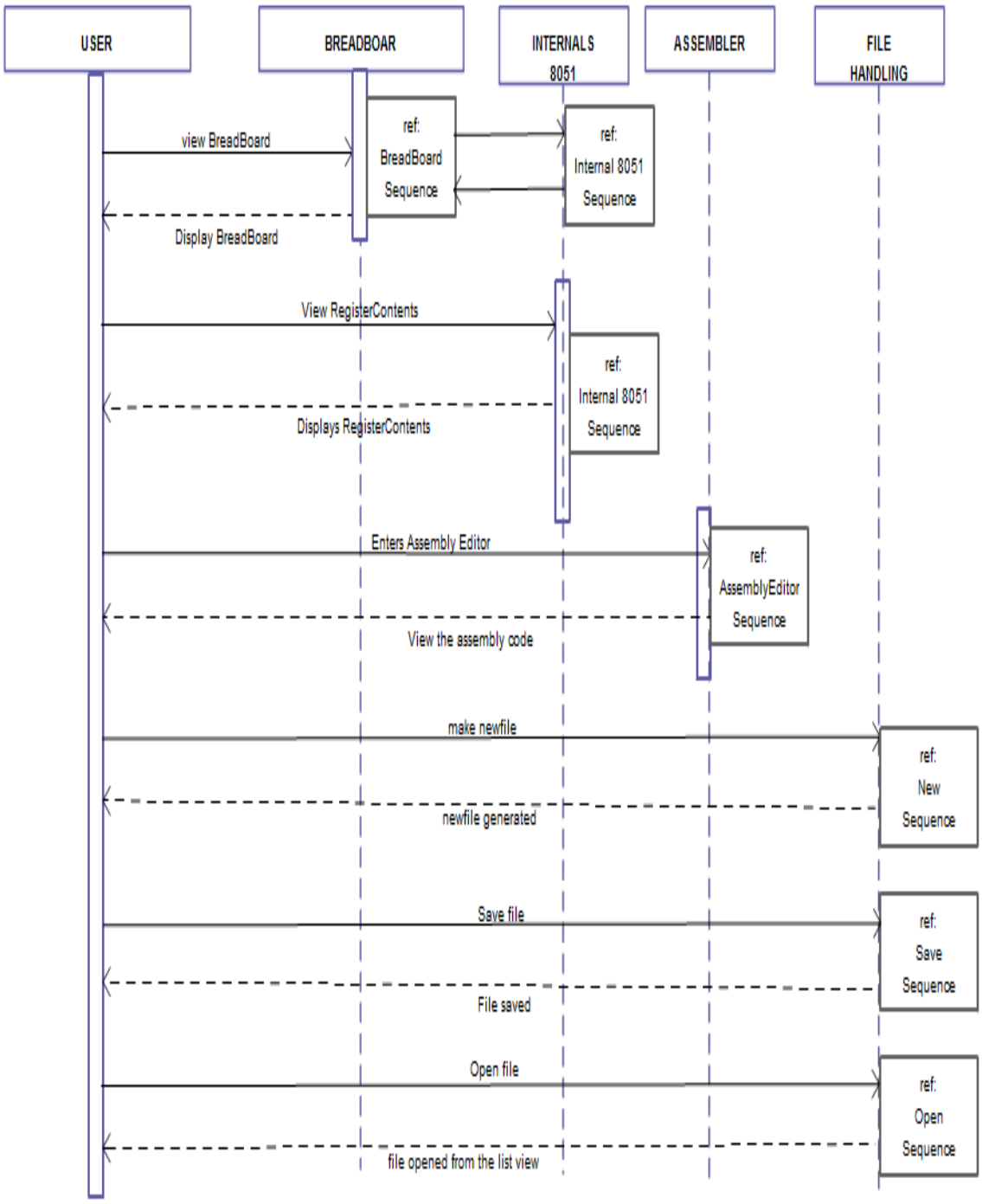

Fig.6.4.1 Sequence diagram

The following is the detailed explanation of various modules shown above:

BreadBoard sequence:

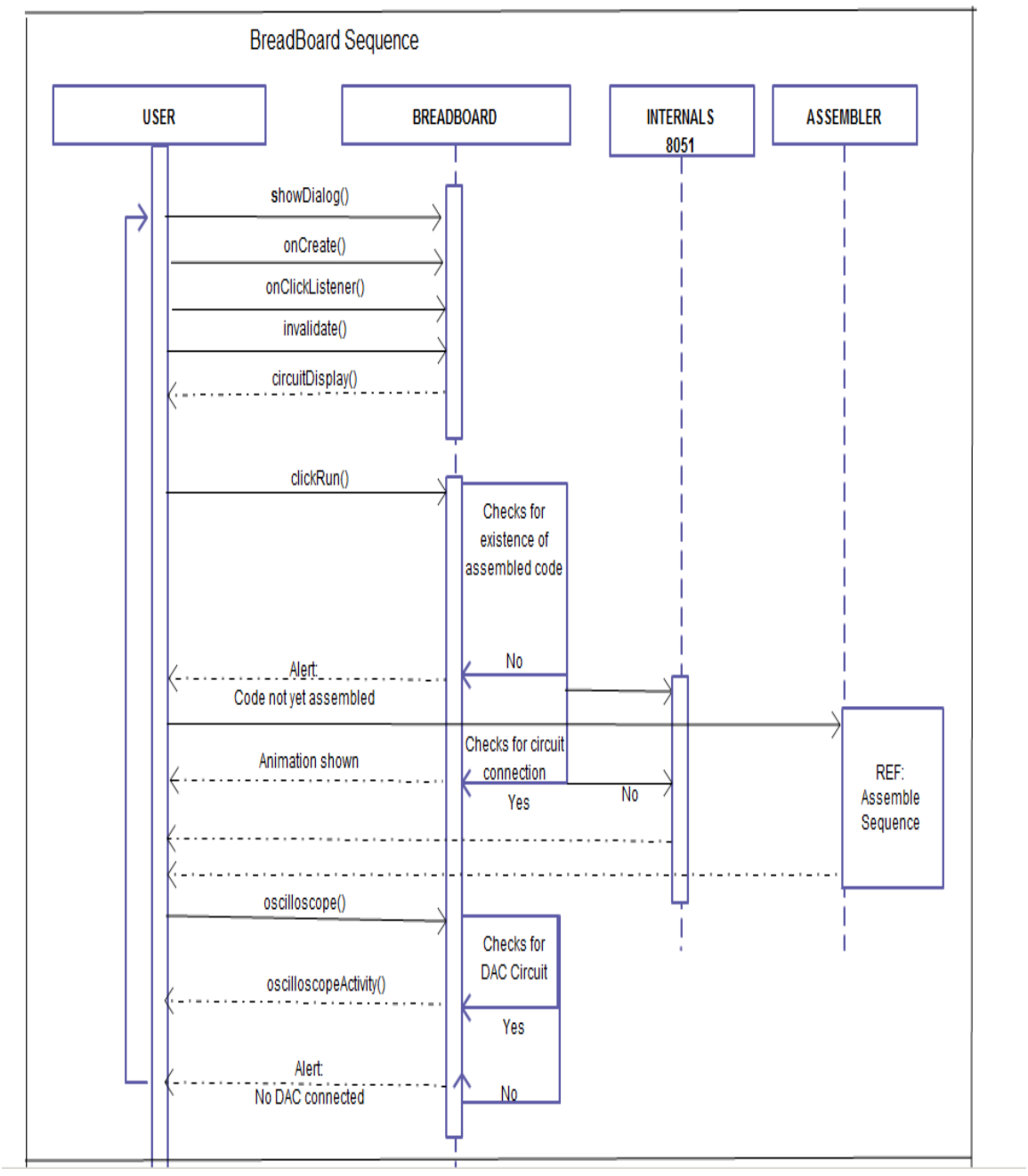

Fig.6.4.2 Sequence diagram- Breadboard

Assemble Sequence:

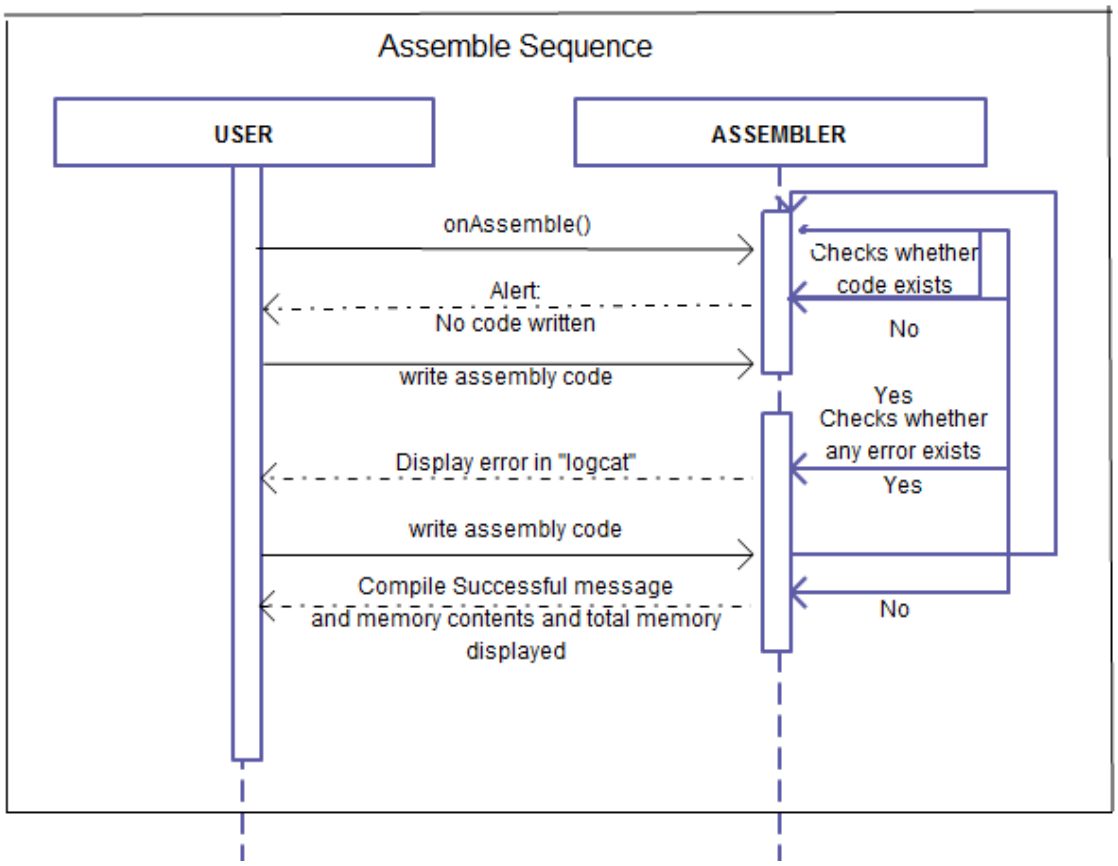

Fig.6.4.3 Sequence diagram- Assemble

Internals8051 Sequence:

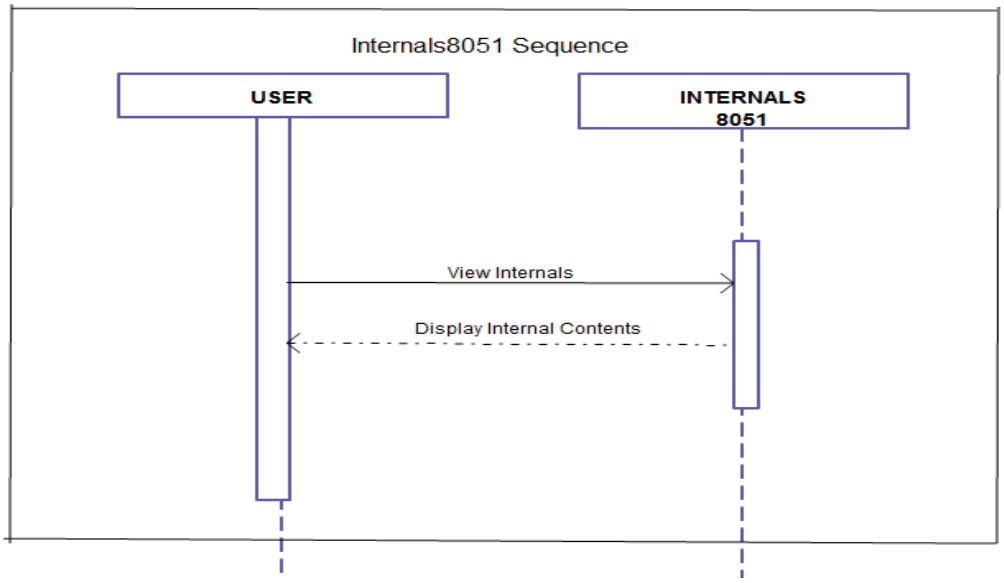

Fig.6.4.4 Sequence diagram – Internals8051

File Handling Sequence:

1. Save sequence

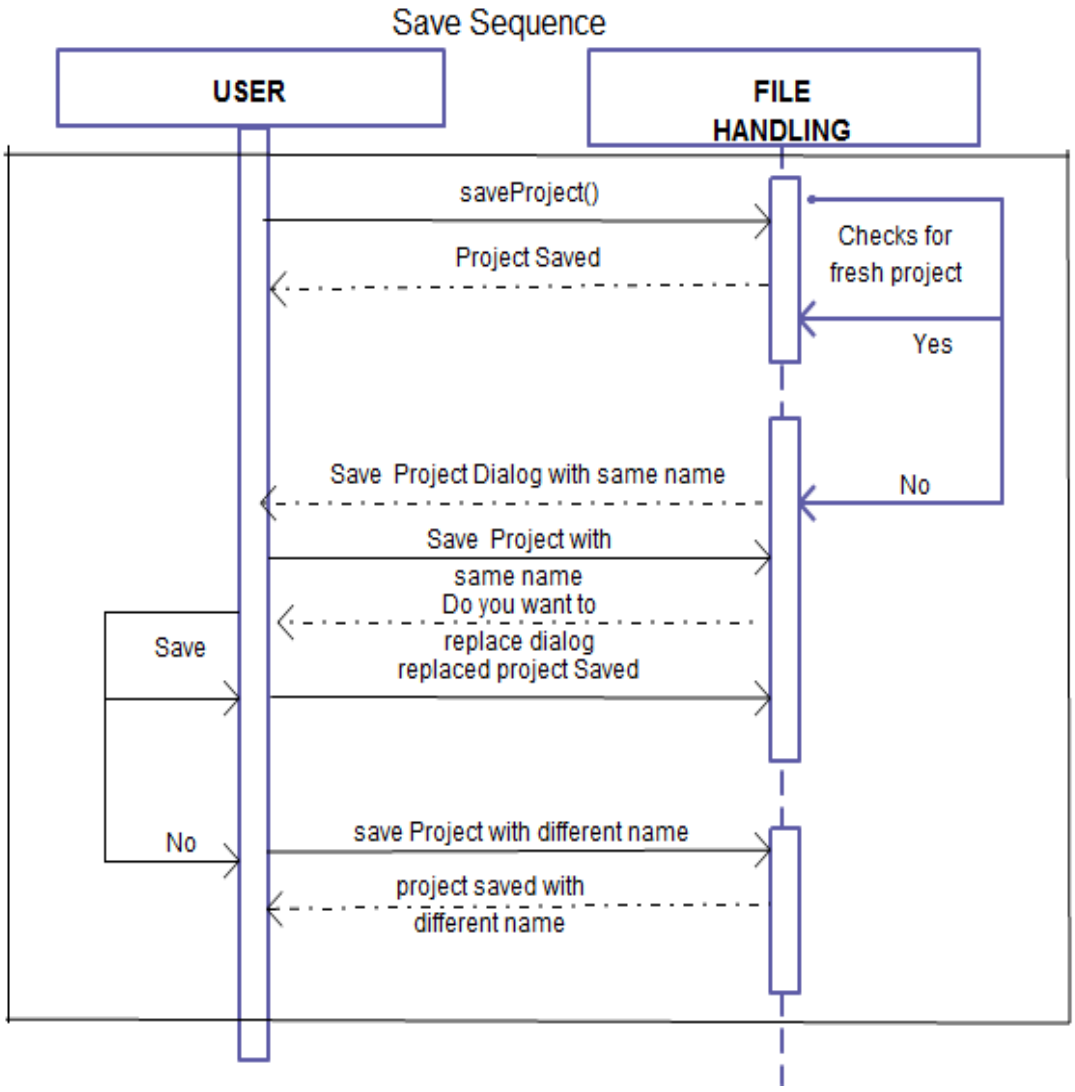

Fig.6.4.5 Sequence diagram for save sequence in file handling

2. New Sequence

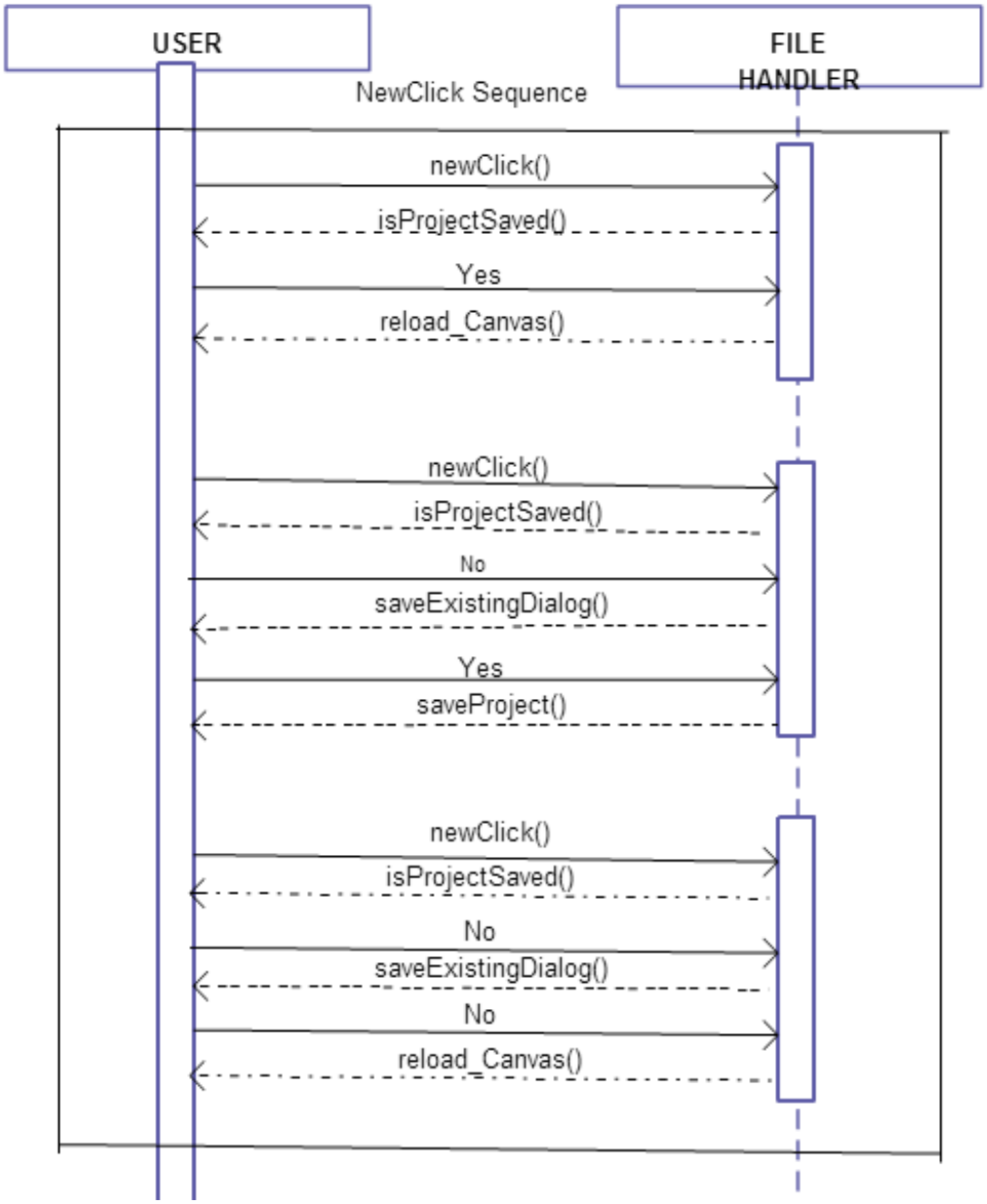

Fig.6.4.6 Sequence diagram for new sequence in file handling

3. Open Sequence

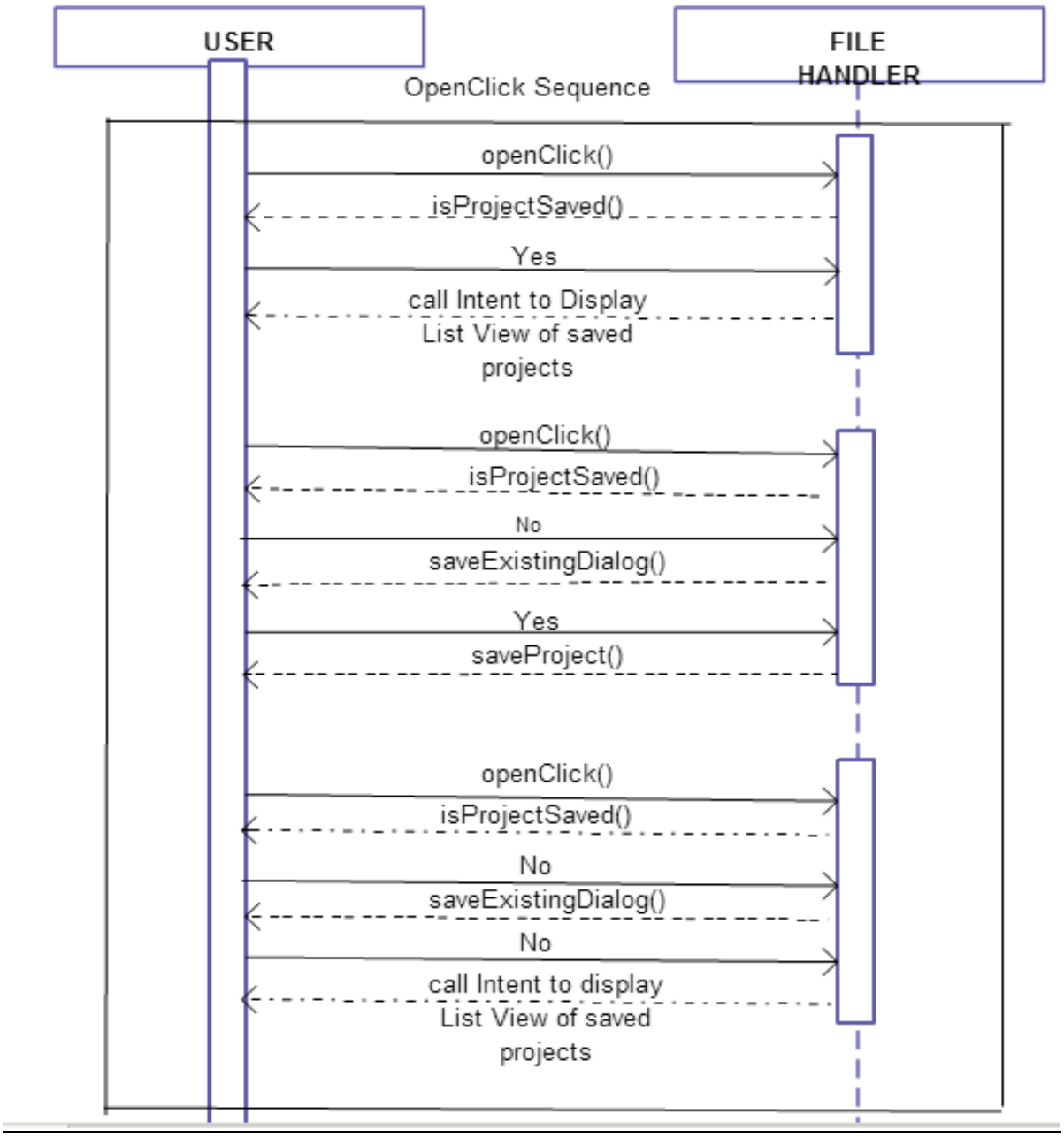

Fig.6.4.7 Sequence diagram for open sequence in file handling

7. Class Description

1. **LaunchActivity**

This is the first activity that appears on launching the application. It displays the splash screen for 5 seconds and then switches to the next activity, the HomeActivity.

2. **HomeActivity**

It has a navigation type of Fixed Tabs + Swipe. The tabs displayed in this activity are Introduction and About Us. It also contains two buttons for Start and Take a Tour.

The start button leads to the work bench module. While the Take a Tour button plays a video which gives a brief description about how to use the application.

3. **ApplicationActivity**

It has a TabHost implemented for the four modules of the application. These modules are WorkBench, Editor, Internals8051 and Help. It facilitates the user to switch easily between these four modules. It also contains a sliding drawer having three buttons. These buttons are New, Open and Save which perform the different file handling operations. The functions for these buttons are also implemented in this activity.

4. **WorkBench**

This activity contains an object of the Breadboard activity. It implements the UI of the workbench. It contains the Build, Execute and Oscilloscope buttons.

The Build button facilitates the selection of the circuits to be connected to the different ports of the microcontroller. The Execute button executes the assembled code written by the user and produces the corresponding animation on the canvas. It also updates the values of the registers in Internals 8051. The oscilloscope button produces a voltage vs time graph in a pop up dialog box when on selection of the DAC circuit.

5. **BreadBoard**

The canvas is created in this activity. It places the 8051 microcontroller and the clock, power, reset and pull up circuits as bitmaps, which are by default mounted on the canvas. The onDraw method of the canvas is also implemented here. The places the interface circuits as bitmaps based on user selection. Changes are made to the canvas in this activity to produce the desired animations. This is done by reading the values of the ports from the Execute activity at runtime. The onTouch event facilitates scrolling of this canvas.

6. **Circuit**

This activity contains a static string array object of Component class and a static string array variable, port_value. This enables multiple classes to access these static objects and variables. It updates the port value continuously as the instructions are executed.

7. **Component**

This activity contains a constructor Component having two parameters which sets the image of the selected interface circuit and its corresponding ID. It contains additional functions so that other classes can retrieve the image and its ID.

8. **Editor**

This activity implements the UI of the editor. It contains two buttons Assemble and Clear. The Assemble button calls the onAssemble method. This function in turn calls a method of the Assemble activity. The clear button clears the contents of the EditText and all the TextViews in the Editor.

9. **Assemble**

This activity contains the method assembleFunction. This method takes the assembly code given by the user as input. It converts it into the corresponding op codes and data and stores it in the string array “rom”. It implements the two pass assembler concept. In case of any errors in the assembly code, it displays the line number of the error in the TextView logcat, else it displays “Compiled Successfully”. It also displays the contents of the ROM with the corresponding addresses and the total ROM usage in the respective TextViews.

10. **Execute**

This activity contains the method nextPC. This method reads the value of the string array “rom” and executes the instruction one by one through an extensive if else statement. It constantly updates the RAM values during runtime.

11**. Internals8051**

It implements the UI of the internals8051. It reads the values of the continuously updated RAM and displays it on the screen with the corresponding register name adjacent to it.

12. **VoltageOutput**

This activity contains two static arrays to store the values of voltage and time. These values are continuously updated during execution generate the graph in the oscilloscope.

13. **oscilloscopeActivity**

This activity contains an object of the OscilloscopeGraph to set its view to a graph.

14. **OscilloscopeGraph**

This activity contains the canvas where the voltage vs time graph is plotted. The values needed to plot this graph are read from VoltageOutput. The onTouch event facilitates scrolling of this canvas.

15. **Help**

This activity implements the UI of the help module. It contains a web view that displays the document selected in the list view. These documents are User Manual, About 8051 and Experiment List.

16. **openActivity**

The Open button in the sliding drawer calls this activity. It displays a dialog box that contains the list view of all the existing experiments previously saved in the SD card. The required experiment is opened on selection. On long clicking any item on this list, deleted the corresponding experiment from the SD card.

8. Conclusion

The application has been modelled to accurately emulate the architecture of the 8051 microcontroller. The method of assembling utilized in the application is similar to the two pass method implemented in reality. In case of real hardware, the user needs to write the code in machine language by constantly referring to the instruction set. However, this application supports assembly language programming, which ensures the user a wide range of flexibility. The timing and synchronization of assembling and execution has been modelled to resemble the timing in the actual hardware. The application is also scalable and allows further improvements and additional features to be added with ease.