MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

CENG 491
COMPUTER ENGINEERING DESIGN I

FINAL DESIGN REPORT

YAYA BILISIM

YASIN ALPEN	1297431
KAAN Y. CEYLAN	1347269
YUNUS ESENCAYI	1347459
AHMET TAHIR UCKUN	1298371

INDEX

1. Project Description

This report describes the process of RCSim Software which is a product of YAYA Biliþim in initial design level. In this report, we will clarify the process and our work on RCSÝM Software.

Yaya Bilisim is a DEVEMB project group which is supposed to develop a software emulator for CEng Embedded Card briefly. One who wants to understand RCSim Software must have a background on PICs, embedded systems, emulators and development boards. In order to understand our work better, we should deal with these topics briefly.

1.1 PIC:

A microcontroller is a compact standalone computer, optimized for control applications. Entire processor, memory and the I/O interfaces are located on a single piece of silicon so, it takes less time to read and write to extrernal devices.

Following are the reasons why microcontrollers are incorporated in control systems:

a. *Cost:* Microcontrollers with the supplemantary circuit components are much cheaper than a computer with an analog and digital I/O
b. *Size and Weight:* Microcontrollers are compact and light compared to computers
c. *Simple applications:* If the application requires very few number of I/O and the code is relatively small, which do not require extended amount of memory and a simple LCD display is sufficient as a user interface, a microcontroller would be suitable for this application.
d. *Reliability:* Since the architecture is much simpler than a computer it is less likely to fail.
e. *Speed:* All the components on the microcontroller are located on a singe piece of silicon. Hence, the applications run much faster than it does on a computer.

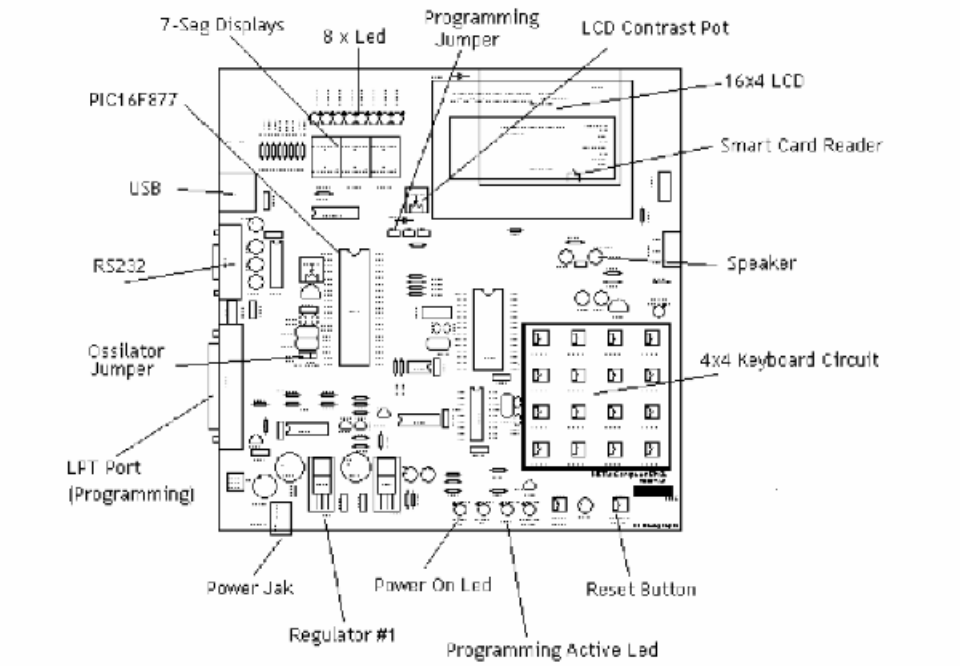
There are a lot of microcontroller manufacturers and they are named according to their manufacturers. PIC (Peripheral Interface controller) is the one produced by Microchip. PICs have Harvard architecture but not Von Neuman.

Since our product is only about PIC16F877, we will deal with it. PIC16F877 is one of the most commonly used microcontroller especially in automotive, industrial, appliances and consumer applications. The core features of PIC16F877 are:

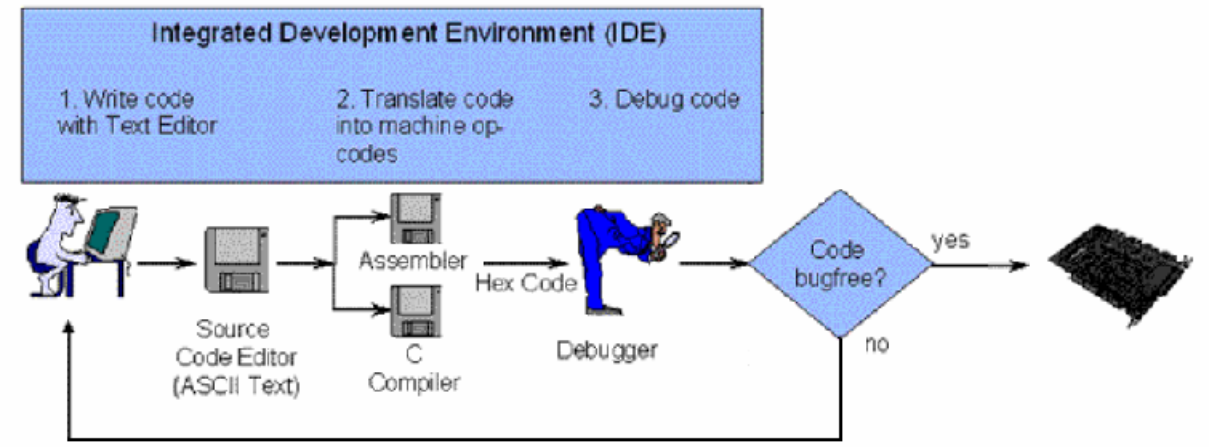
- 14 bit cores with 35 instructions.
- 200 ns instruction time
- 8092 14 bit Flash program memory
- 368 8 bit data memory or registers(RAM)
- 256 8 bit EEPROM data registers
- 8 level hardware stack
- Up to 14 interrupt capability
- 33 I/O pin
- 3 timer/ counter modules
- 10 bit 8 channel A/D converter
- Parallel and Serial ports

1.2 CEng Embedded Card

CEng embedded system card is the card that is used in CEng 336 "Embedded Systems" course. It includes two PIC processors and various interfaces like LCD, Parallel, Serial, USB ports, smartcard reader, LED's etc.



1.3 The Process:



Developing a project have some steps as:

1- Writing the code:

Software Code for a microcontroller is written in a programming language of choice (often Assembler or C). This source code is written with a standard **ASCII text editor** and saved as an ASCII text file. Programming in assembler involves learning a microcontroller's specific instruction set (assembler mnemonics), but results in the most compact and fastest code. A higher level language like C is for the most part independent of a microcontroller's specific architecture, but still requires some controller specific extensions of the standard language to be able to control all of a chip's peripherals and functionality.

2- Translating the code:

Next the source code needs to be translated into instructions the microcontroller can actually execute. A microcontrollers instruction set is represented by "op codes". Op codes are a unique sequence of bits ("0" and "1") that are decoded by the controller's instruction decode logic and then executed. Instead of writing opcodes in bits, they are commonly represented as hexadecimal numbers, whereby one hex number represents 4 bits within a byte, so it takes two hex numbers to represent 8 bits or 1 byte. For that reason a microcontroller's firmware in machine readable form is also called Hex-Code and the file that stores that code Hex-File.

3- Debugging the code:

Since the process of burning the code to the card takes a long time, it is unwanted to burn an error including code. In order to prevent such situations, it is

better to check and debug the code before burning. This is in software level. Although it can be in hardware level (for example setting break points and inspecting the changes on the card), it is out of our topic.

4- Burning to the Card:

The final step is to burn the bug-free code to the Card. Burning means to transfer data from the computer to the Card and investigating the results.

1.4 Emulators:

Even it is a simple project, uploading it to the Card takes a long time. So, for a user it will be very time consuming to work on PICs. In order to reduce this, emulators are developed. Emulators help user to upload their high level or assemble language code to development boards. Moreover, these software provide user to simulate their code's response without burning it to the Card.

1.5 Our Product:

Emulators(or simulators) which are available are suitable for many types of PICs and development boards. It can be seen as an advantage, however it is not the case sometimes.

CEng336 Embedded Course is a must course of computer engineering department in Metu. In this course, the students are supposed to do some work with CEng Embedded Card which is also created by this department. MPLab is the software used in this lesson.

We think that, taking Logic Design course which is the only prerequisite course for CEng336 do not make it easy to do their works on CEng Embedded Card for a junior student. They need to be instructed by assistants or teachers. This is a time-consuming process for students, assistants and instructors. This is where our project is burned.

Our product RCSim is designed just for CEng Embedded Card. Goals and objectives of RCSim Software is as follows:

Easy to Use: We motivate ourselves as if our product will be using in CEng336 course spring 2007. So, it is very important for us to develop a product such that CEng336 students will easily use RCSim Software.

As mentioned before, RCSim Software is specific on CEng Embedded Card. So, it should be adapted on this card and PIC16F877 only. It keeps us from selection

of PIC type and development board type. For example, because of this, there will be less steps for creating a new project. Such simplicities will make it easier to use RCSim Software for CEng336 students.

Moreover, we believe the importance of a user-friendly GUI for easy to use. So, an easy to use and simple GUI is one of our objectives. On the other hand, we plan to put a satisfactory help menu and a self learning tool in our GUI.

Responding to Requirements: Although we design a software as simple as possible, we will provide all requirements of CEng Embedded Card. Satisfying this balance is very important for us.

Most Realistic: Simulation is a very important part of our product. Users will be able to simulate their codes without burning it to the card. These simulations should be done as realistic as possible in order to make those simulations reliable.

2. Team Process

Since we spend much time on preparing the initial design report, we think that we had reached a better understanding on our subject. On the other hand, this process helped us to see the needs of our project.

After delivering the initial design report, we were sure that we should continue on working without giving a break. To be honest, on this point we wanted to have the feedback on the initial design report. Up to that day, we did our work without being sure. When we had the feedback of the initial design report from our assistant Mr. Orkan, we had much more time on preparing the final report design. We set many meetings when the team members had no exams or other projects.

We started to implementation part in this age. The first implementation part of RCSim is the text editor part. Concurrently, we had started to implement the main interface of RCSim. The next step was embedding MPASM to RCSim. Now, we have the main interface and text editor part of RCSim, besides assembler.

While implementing such a big project, it is ordinary to have some bugs. This lead us to release updates of RCSim. Up to this time, we have RCSim, RCSim 0.1.1, RCSim 0.1.2, RCSim 0.1.3 and RCSim 0.1.4 versions.

For the second term, we intend to divide our project into main components. We will set time intervals to each component. Since, we are not so experienced on this area, we are not sure that, which component takes how much time exactly. This situation made us to make decisions about the implementation of RCSim in the second term as follows:

1- If we finish a component in the expected time, nothing to do but continue to implement to the next component.

2- If we are in front of a dead line and we finish to implement a component, do not wait until the dead line but start to the next implementation part and re-schedule the rest time.

3- If we could not finish one component up to its dead line, hasten the process. When we finish that component, again re-schedule the rest time.

We have given the main components and their present situations below:

Assembler : COMPLETE FOR demo

TextEditor COMPLETE FOR demo

GUIManager %50 COMPLETE FOR demo

Debugger : INCOMPLETE

BurnToCard : INCOMPLETE

Simulator : INCOMPLETE

Compiler : INCOMPLETE

Moreover we have given the next semester's plan of implementation.

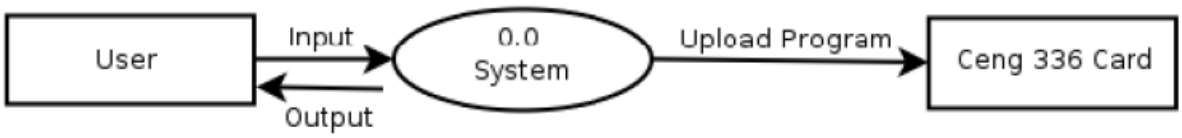
FROM	TO	PARTS IMPLEMENTED	PARTS TO COMPLETE
Beginning	Present	Text Editor , GUI Manager , Assembler ,Simulation(Simple)	Demo
20-Jan	1-Mar	Text Editor , GUI Manager , Assembler ,Simulation(Simple)	Simulation , Debugger
1-Mar	21-Mar	Text Editor , GUI Manager , Assembler ,Simulation,	BurnToCard

		Debugger ,	
21-Mar	1-Apr	Text Editor , GUI Manager , Assembler ,Simulation, Debugger , BurnToCard	Compiler
1-Apr	15-Apr	Text Editor , GUI Manager , Assembler ,Simulation, Debugger , BurnToCard , Compiler	Ready For Demo

3. Schematic Layout Of the Software

3.1 Represantation of the Flow of the Data

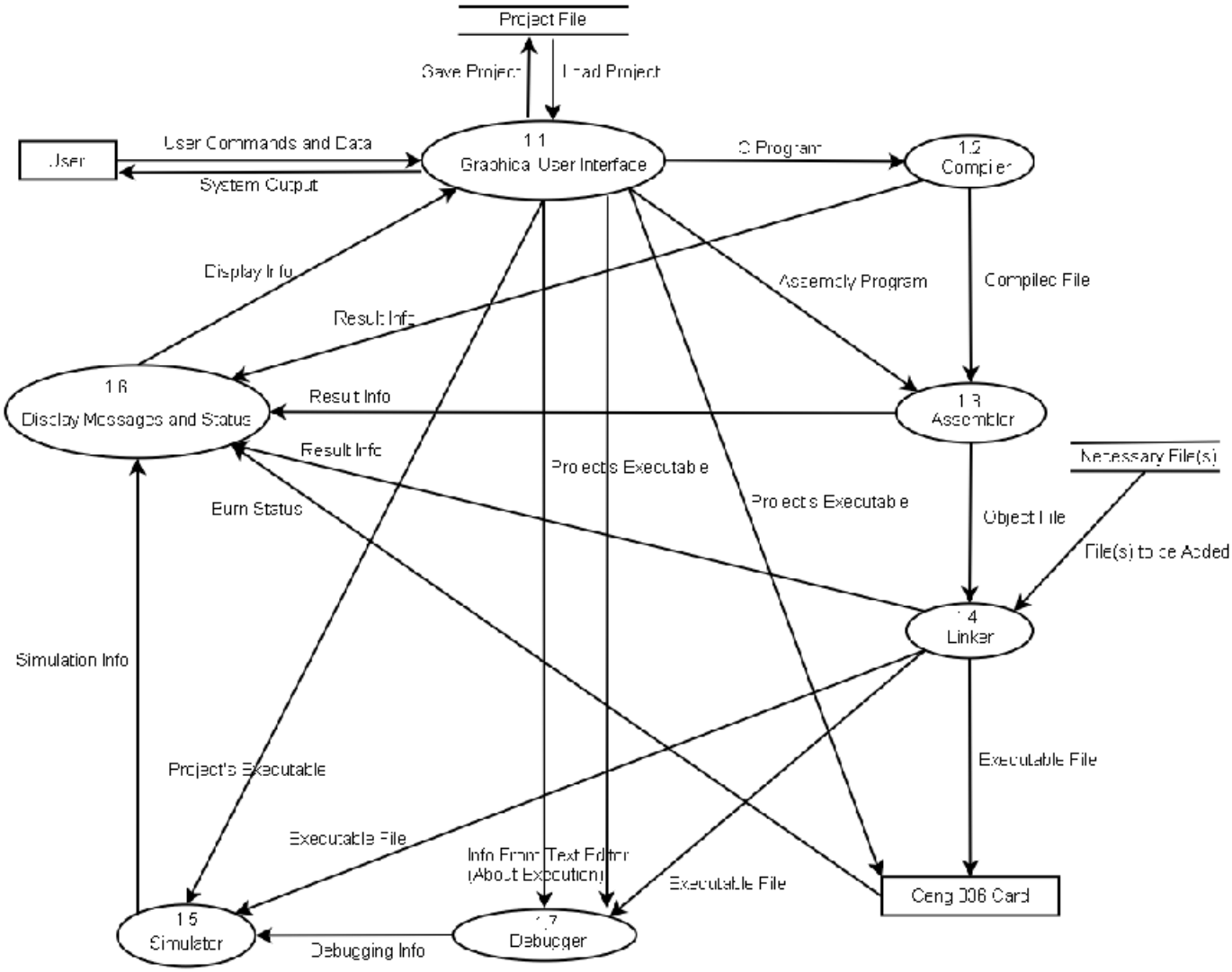
3.1.1 Simple Data Flow Diagram



LEVEL 0 DFD

User gives inputs to the system and takes response from the system. Also the user can upload the program to the Ceng 336 Card.

3.1.2 Detailed Data Flow Schema



LEVEL 1 DFD

3.2 Data Dictionary

Name:	User Commands and Data

Aliases:	None
Product of:	User
Where used:	Graphical User Interface (Process 1.1)
Description:	User controls by mouse clicks, keyboard keys, or writing text through editor.

Name:	System Output
Aliases:	None
Product of:	Graphical User Interface (Process 1.1)
Where used:	User
Description:	User sees the present situation of the program through a user interface.

Name:	C Program
Aliases:	None
Product of:	Graphical User Interface (Process 1.1)
Where used:	Compiler (Process 1.2)
Description:	A program written in C language through the text editor in the program.

Name:	Compiled File
Aliases:	None
Product of:	Compiler (Process 1.2)
Where used:	Assembler (Process 1.3)
Description:	Assembly code of the program which is converted by the compiler.

Name:	Assembly Program
Aliases:	None
Product of:	Graphical User Interface (Process 1.1)
Where used:	Assembler (Process 1.3)
Description:	A program written in assembly language through the text editor in the program.

Name:	Object file
Aliases:	None
Product of:	Assembler (Process 1.3)
Where used:	Linker (Process 1.4)
Description:	Object code of the program which is converted by the assembler.
Name:	Display Info
Aliases:	None
Product of:	Display Messages and Status (Process 1.6)
Where used:	Graphical User Interface (Process 1.1)
Description:	Command line outputs or information

	about the state of the card (e.g. value of the registers, memory etc.)

Name:	Result Info
Aliases:	None
Product of:	Compiler(Process 1.2) , Assembler (Process 1.3), Linker (Process 1.4)
Where used:	Display Messages and Status (Process 1.6)
Description:	Command line outputs of the compiler, assembler or linker.

Name:	Simulation Info
Aliases:	None
Product of:	Simulator (Process 1.5)
Where used:	Display Messages and Status (Process 1.6)
Description:	Information about the simulated parts of the card.

Name:	Executable File
Aliases:	None
Product of:	Linker (Process 1.4)
Where used:	Simulator (Process 1.5), Ceng 336 Card, Debugger (Process 1.7)
Description:	Binary file that can be executed on Ceng 336 Card or simulated by the program.

Name:	File(s) to be added
Aliases:	None
Product of:	Necessary File(s)
Where used:	Linker (Process 1.4)
Description:	Necessary library files to be able to execute the program for the type of PIC processor that is used in Ceng 336 Card

Name:	Project’s Executable
Aliases:	None
Product of:	GUI (Process 1.1)
Where used:	Simulator (Process 1.5), Ceng 336 Card, Debugger (Process 1.7)
Description:	Binary file that can be executed on Ceng 336 Card or simulated by the program that belongs to a previously written

	project.

Name:	Info From Text Editor
Aliases:	None
Product of:	GUI (Process 1.4)
Where used:	Debugger (Process 1.7)
Description:	Necessary information about the code text file (line numbers, breakpoints etc.) for the debugger to run properly.

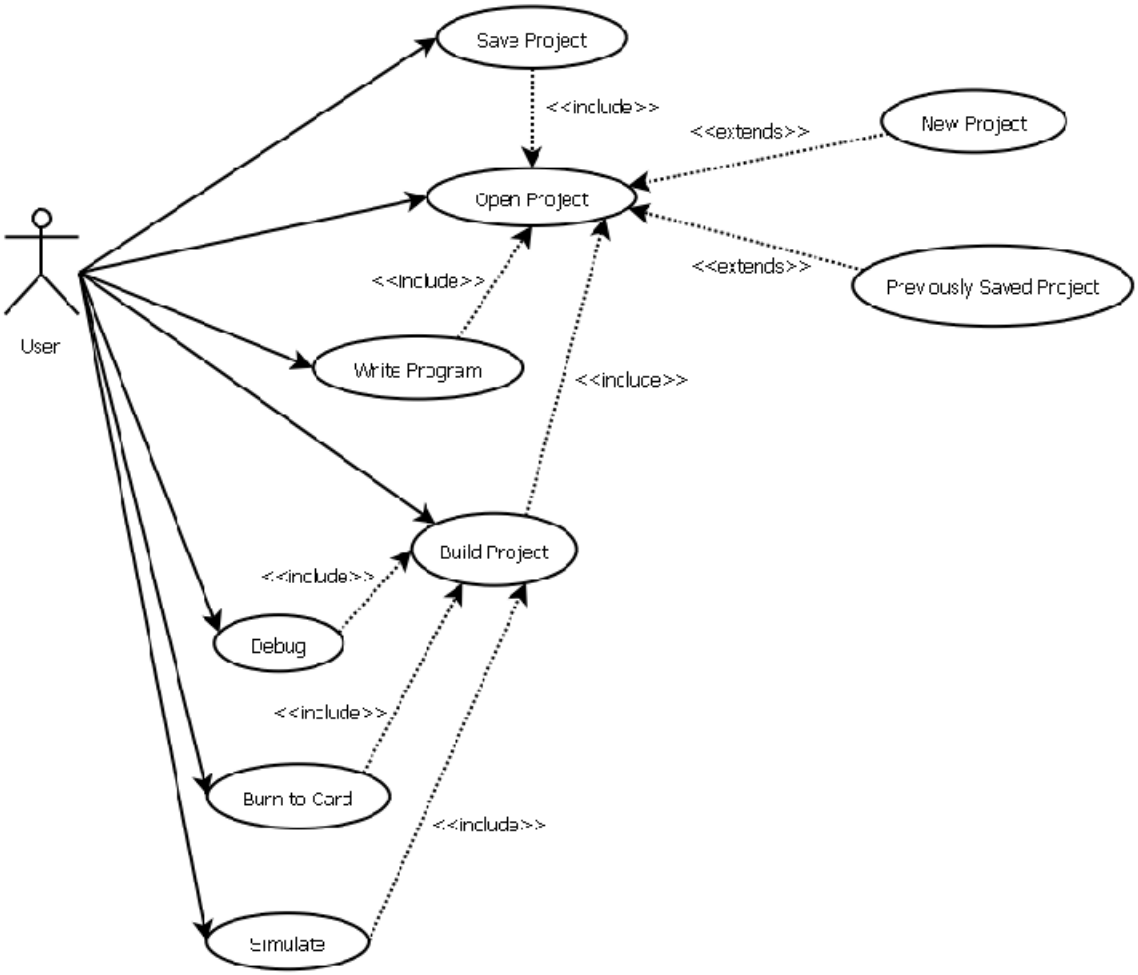
Name:	Burn Status
Aliases:	None
Product of:	Ceng 336 Card
Where used:	Display Messages and Status (Process 1.6)
Description:	Message indicating whether the executable file is successfully written to the card or not.

Name:	Debugging Info
Aliases:	None
Product of:	Debugger (Process 1.7)
Where used:	Simulator (Process 1.5)
Description:	Current state of the execution. (The values of the registers, memory etc.)

Name:	Save Project
Aliases:	None
Product of:	Graphical User Interface (Process 1.1)
Where used:	Project File
Description:	Code and the settings of the current project to be saved.

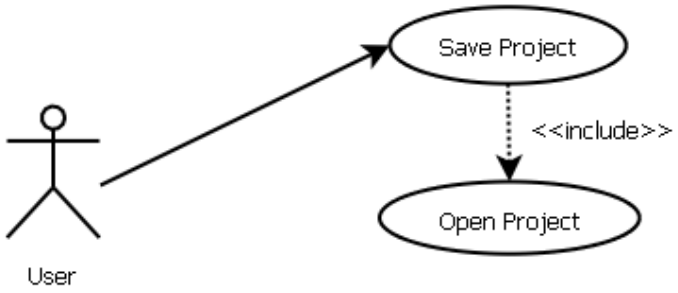
Name:	Load Project
Aliases:	None
Product of:	Project File
Where used:	Graphical User Interface (Process 1.1)
Description:	Code and the settings of the project to be loaded.

3.3 Interaction of the User with the System



Use Case Diagram

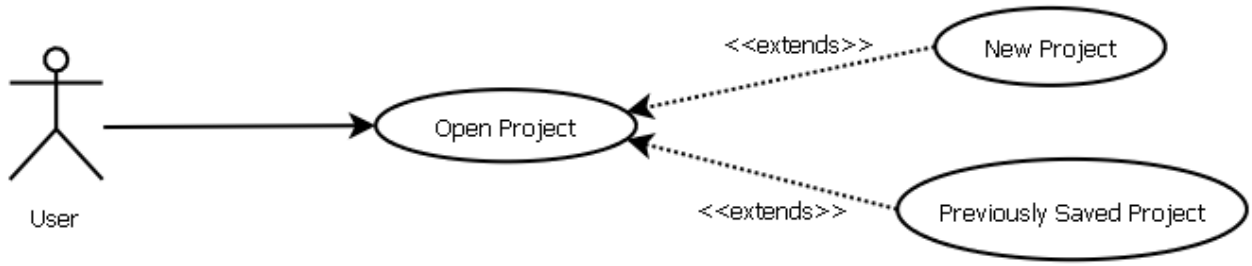
3.3.1 Save Project



Use Case Diagram (Save Project)

User can save the project at any time but for this to be done a project must be opened before.

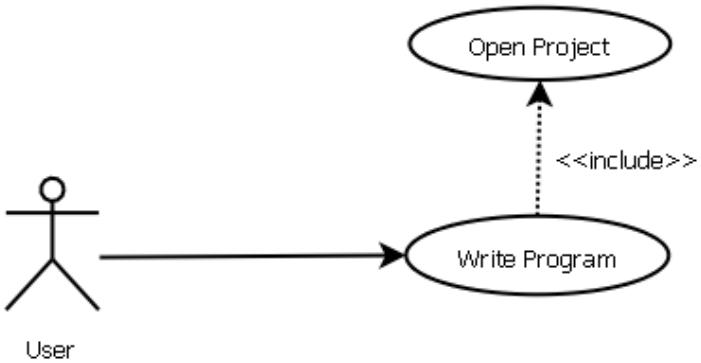
3.3.2 Open Project



Use Case Diagram (Open Project)

User can open a project when the system is idle. This project can be either a new blank project or a previously written and saved project.

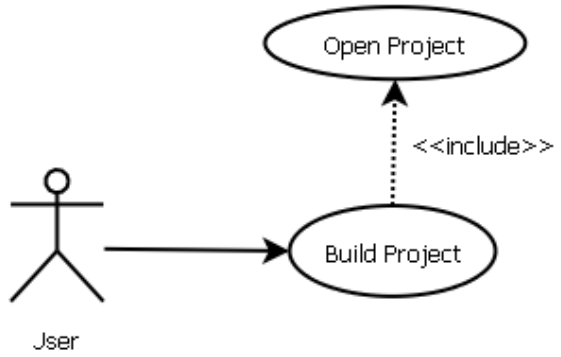
3.3.3 Write Program



Use Case Diagram (Write Program)

User can write program using the text editor in C or assembly language. But for this to be done a project must be opened before.

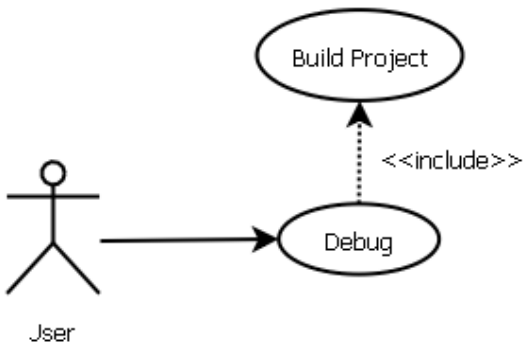
3.3.4 Build Project



Use Case Diagram (Build Project)

User can build a project via the user interface. Building includes compiling, assembling and linking stages for a program written in C language, whereas for a program written in Assembly language building includes assembling and linking steps. For a project to be builded, a project file must be opened before.

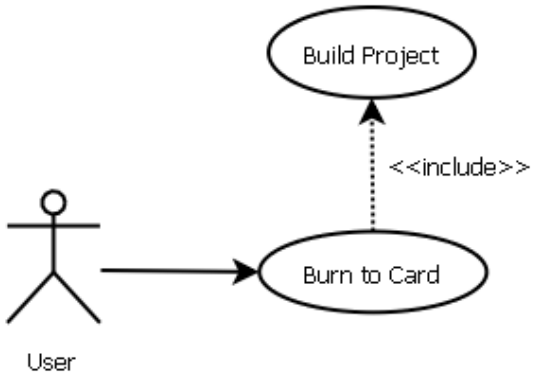
3.3.5 Debug



Use Case Diagram (Debug)

The system includes a software-level debugger, in which the user will be able to execute the current program step by step, put breakpoints, see the contents of the registers, memory etc.
But for this to be enabled, a new or previously written project must be builded successfully before and by the way, a hex file must exist for that project.

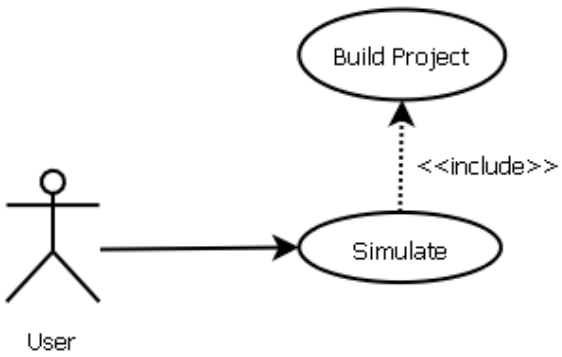
3.3.6 Burn to Card



Use case Diagram (Burn to Card)

User can burn programs that he/she wrote or are previously written. But for this to be enabled, a new or previously written project must be builded successfully before and by the way, a hex file must exist for that project.

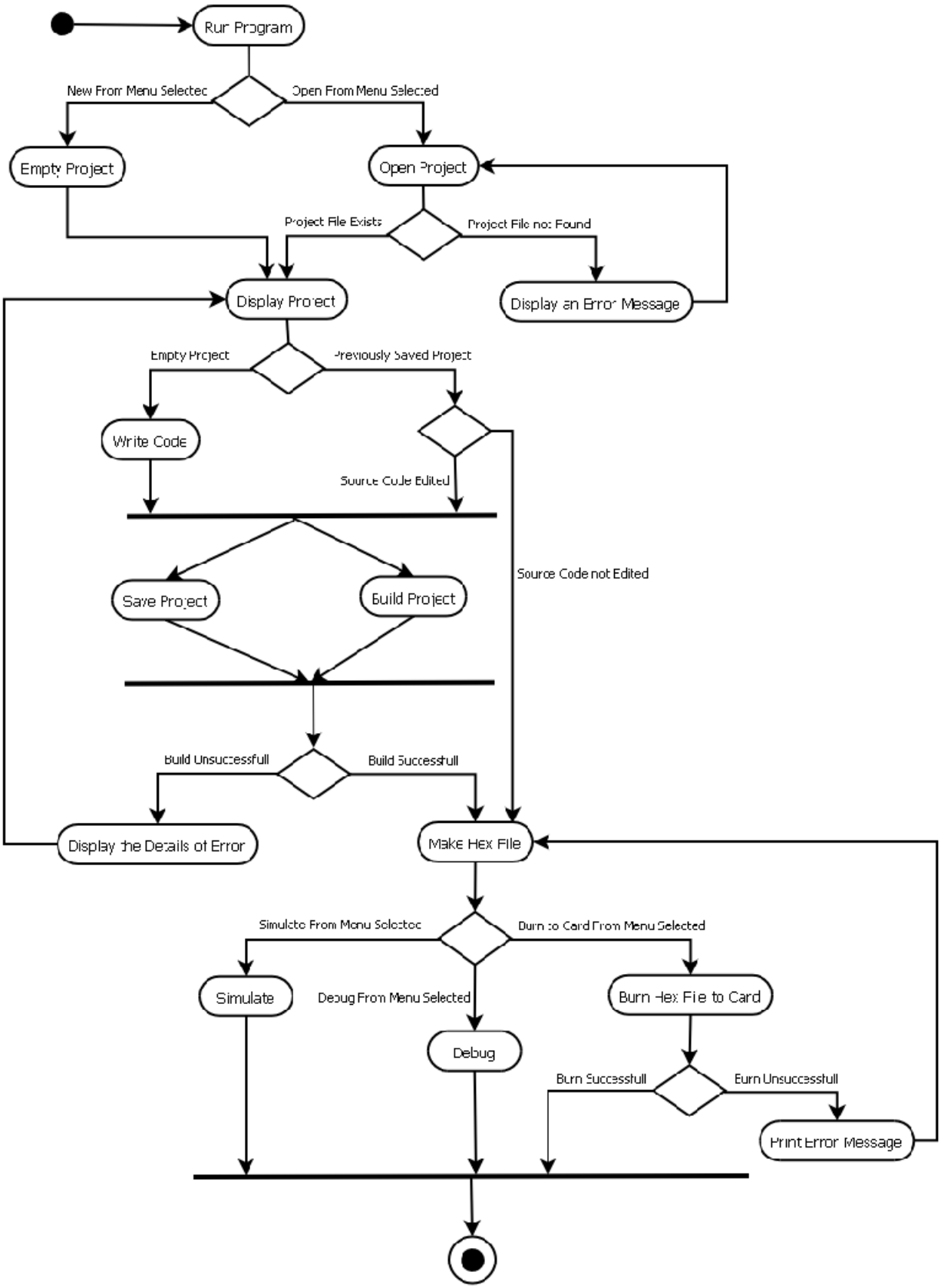
3.3.7 Simulate



Use Case Diagram (Simulate)

User can simulate a program that is previously written or newly written by him/her. But for this to be enabled, a new or previously written project must be builded successfully before and by the way, a hex file must exist for that project.

3.4 Schematic Representation of System Activities



Activity Diagram

Run Program:

The program is opened via the Windows user interface. After that, the program starts and the graphical user interface opens without any project or file opened. According to the user's choice:

- New from Menu selected: So an empty project and the text editor opens automatically to enable the user to write code.
- Open from Menu selected: So an open file dialog opens up to enable the user to explore the directories to find and open a previously saved project. After the project file is selected and OK is pressed, the selected project file opens, its settings are loaded and the source code is shown by the text editor automatically to enable the editing of the code.

Empty Project:

New from menu is selected and an empty project and the text editor opens automatically, showing an empty code file.

Open Project:

Open from menu is selected and an open file dialog opens. After that, the user explores the directories to find and open a previously saved project. According to the user's action:

- Project exists: The user explores the directories and selects the project file via mouse or typing the name of the project. The project exists and it is loaded and displayed. In here the user has two choices:
    - Edit the source code, build and run, and
    - Do not edit the source code, and run the project's executable directly.
- Project does not exist: The user explores the directories and selects the project file via mouse or typing the name of the project. The selected file does not exist or is corrupted. So, an error message indicating that the file that is selected does not exist or may be corrupted is shown to user. After this message box is closed, the system returns to its first state without opening any projects.

Display an Error Message:

After selecting open project from menu, if there exists an error about opening the project file (file does not exist or is corrupted), a message box opens indicating that there has occured an error while trying to open the specified project and the two possible reasons for this error.

Display Project:

New from menu is selected or open project from menu is selected. Then, a new project file is opened and displayed or the selected project is opened and displayed if it is opened successfully.

Write Code:

If a new project is opened, the user can write source code from scratch. At any time he/she can save or build the project.

Save Project:

In fact, this process can be done at the times when the program is idle from the beginning of the execution of the software until stopping of the execution of the software. But if the user opens a new project, writes the source code without making a save at any time, and tries to build it, the system will firstly ask the user to save it using an open file dialog. After this is done and the project file is saved successfuly, the system will let the user to build the program.

Build Project:

After the user writes/edits the source code and wants to convert it to an executable file, he/she builds it by pressing the build button shown in the graphical user interface of the system.
If the project is previously saved but the source code is not saved, the system saves it automatically and builds it. But if the project file is not saved, a file dialog will open up and ask the user to save the project file before building. According to the result of the build two actions are taken:

- Display the details of error
- Make hex file

Display the Details of Error:

After the build button is clicked or build is selected from the menu, if the build is not successfull, the details of the error and what caused it, is shown in an output window, which exists inside the main program window.

Make Hex File:

Contains the action of conversion of the source code to the executable or not at all if a previously saved project file is opened. After the build button is clicked or build is selected from the menu, if the build is successfull, an executable file having .hex extension will be created. This file can be:

- Simulated by using the software,
- Debugged, or
- Written to the Ceng 336 card.

Debug:

After obtaining the executable having the extension hex, the user can observe its behaviour by using the debugger with selecting debug from the menu. The debugger enables the user to execute the program to a specified line number, pause and resume execution, and see the contents of the different parts of the PIC processor (memory, registers, etc.).

Simulate:

After obtaining the executable having the extension hex, the user can load, execute and see the results on the simulator by selecting simulate from the menu without burning the hex code to the Ceng 336 card in the real life. At one step, one line of code is executed, the values are updated, and finally this data is sent to the graphical user interface to be shown.

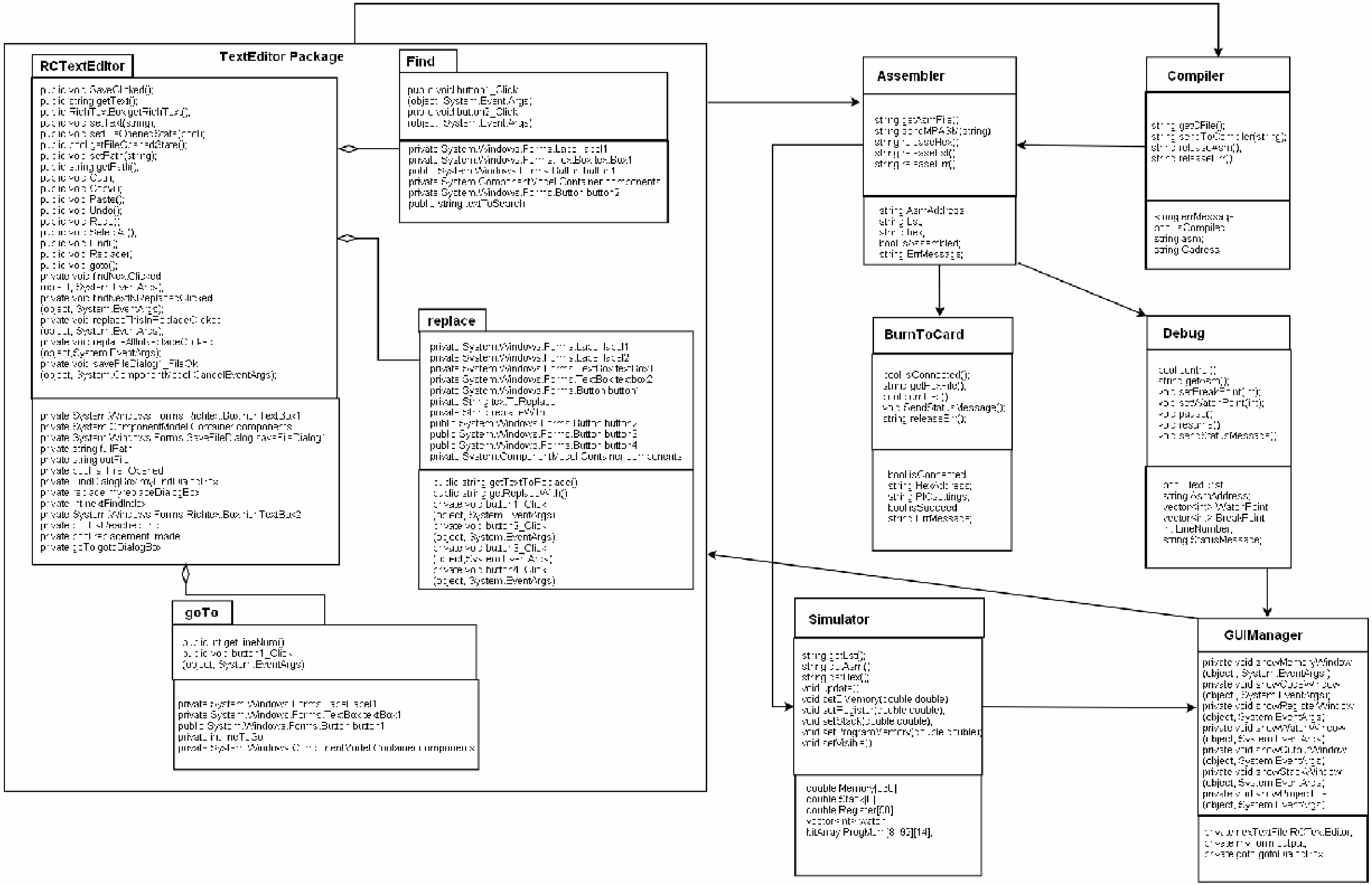
Burn Hex File to Card:

The file having the extension hex can be burned to the card to enable the execution on that card by selecting burn hex file to the card from the menu. If this process is successfull, the system silently returns. But if not, an error message is printed indicating that the burn process has failed.

Print Error Message:

An error message shown in a message box indicates that the burn process of the executable on the Ceng 336 card has failed.

3.5 Main Components and Their Relations in the System


TextEditor Package
RCTextEditor
Find
replace
goTo
Assembler
Compiler
BurnToCard
Debug
Simulator
GUIManager

3.5.1 Assembler Class

Methods

getAsmFile();

This function takes no argument and returns the Asm address as string. By using this method we have the .asm file.

sendMPASM(string);

This function takes the Asm address as argument and returns .hex file. Sends the address to the MPASM as an argument .

releaseHex();

This function takes no argument and return .hex file.

releaseLst();

This function takes no argument and return .lst file.

releaseErr();

This function takes no argument and return .err file.

Attributes

string AsmAddress;

This is the variable of sendMPASM() function.

string lst the address of the lst file

string hex the address of the hex file

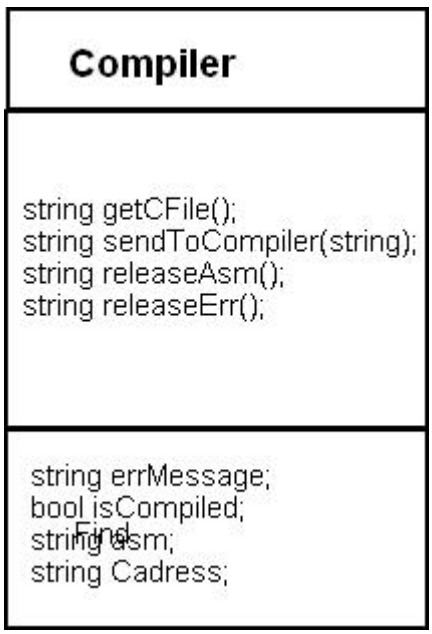
bool isAssembled;

This is set to false initially. It will return true when the C file is compiled into asm file.

string ErrMessage;

This is the attribute of sendMPASM() function. This will be returned when the C file is detected to include bugs.

3.5.2 Compiler Class



Methods

getCFile();

This function takes no argument and returns the address of C file as string. By using this method we have the .c file.

sendToCompiler(string);

This function takes the address of C file as argument and returns .asm file.

releaseAsm();

If isCompiled is true, this function gets the address of the compiled asm file . We will use at this stage a third party software for compilation . This method return .asm file.

releaseErr();

This function takes no argument and return .err file

Attributes

bool isCompiled;

This is set to false initially. It will return true when the C file is compiled into asm and hex files.

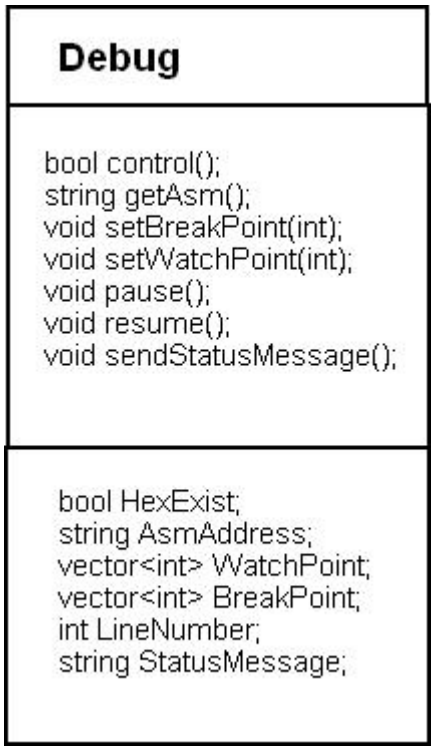
string ErrMessage;

The error message got from 3rd party software .

string Asm the address of the asm file .

string Caddress the address of the C file .

3.5.3 Debug Class



Methods

control();

This function takes no argument. It will check whether the hex file exists or not. It will return boolean such that if hex file exists true, otherwise false.

getAsmFile();

This function takes no argument and returns the Asm address as string. By using this method we have the .asm file.

setBreakpoint(int);

This function takes the line number as int. The mission of this method is to set break points into the indicated line.

setWatchpoint(int);

This function takes the line number as int. The mission of this method is to set watch points into the indicated line.

pause();

This function takes no argument. When this method is called, the process of debugging stops. Debugging stays in this position unless the method resume is called.

resume();

This function takes no argument. When this method is called, the process of debugging starts from the state where it was before calling the method pause;

sendStatusMessage();

This function takes no argument. It returns a message in which the situation of debugging is explained. The type of this message is string.

Attributes

bool HexExist;

This is set to false initially. It will return true when the hex file exists. Otherwise it will return false.

string AsmAddress; the address of the asm file .

vector<int> breakNumber;

This indicates the number of breakpoints that is going to be set. Since multiple numbers possible we use vector.

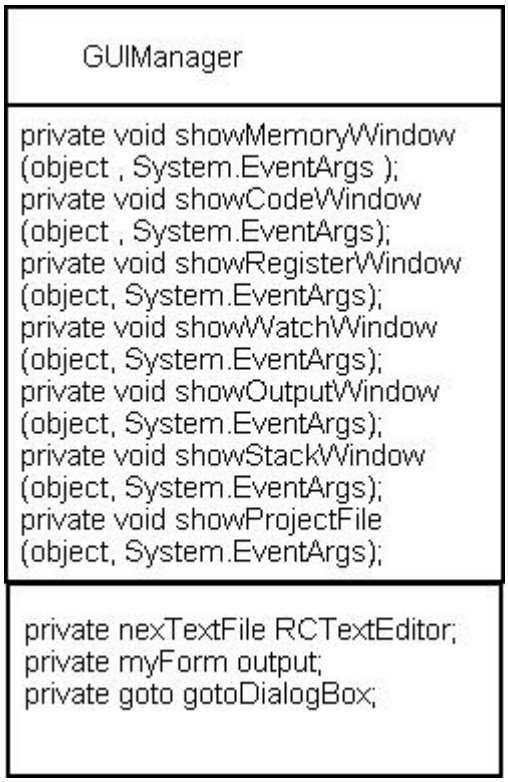
vector<int> watchPoint;

This indicates the number of watchpoints that is going to be set. Since multiple numbers possible we use vector.

int lineNumber; a variable used by getting breakpoint ,watchpoint .

string StatusMessage; The message expressing the current status to the GUI .

3.5.4 GUIManager Class :



Variables :

private newTextFile RCTextEditor

Text Editor variable from RCTextEditor class.

private myForm output

Output window variable from output class.

private goTo goToDialogBox

Go to dialog box variable from goTo class.

Methods :

private void showMemoryWindow(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Memory” choice from the “View” menu. If the memory window is opened, it is closed; otherwise, it is opened.

private void showCodeWindow(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Code Editor” choice from the “View” menu. If the code window is opened, it is closed; otherwise, it is opened.

private void showRegisterWindow(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Registers” choice from the “View” menu. If the registers window is opened, it is closed; otherwise, it is opened.

private void showWatchWindow(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Watch” choice from the “View” menu. If the watch window is opened, it is closed; otherwise, it is opened.

private void showOutputWindow(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Output” choice from the “View” menu. If the output window is opened, it is closed; otherwise, it is opened.

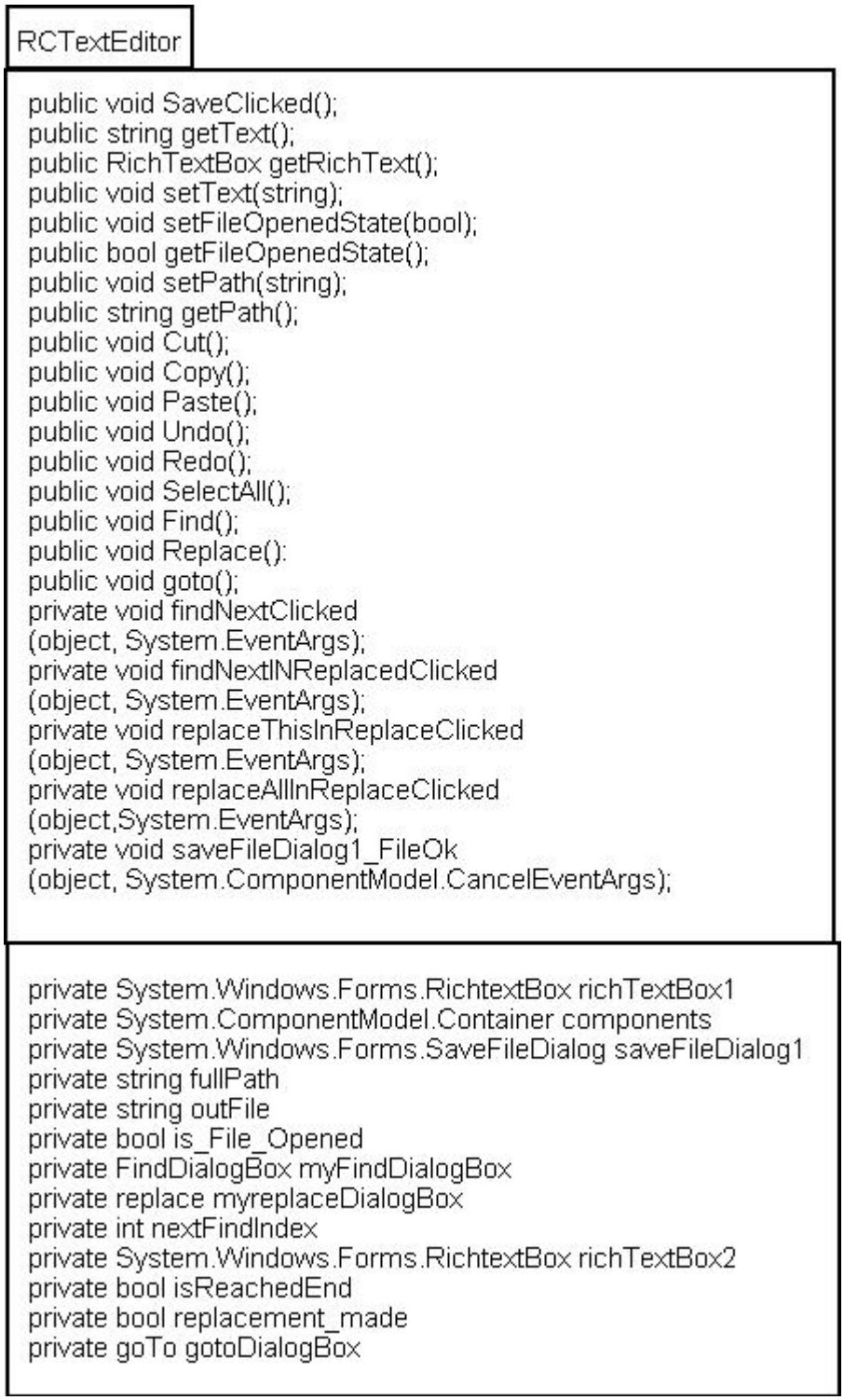
private void showStackWindow(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Stack” choice from the “View” menu. If the stack window is opened, it is closed; otherwise, it is opened.

private void showProject Files(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Project Files” choice from the “View” menu. If the project files window is opened, it is closed; otherwise, it is opened.

3.5.5 Class RCTextEditor:



Variables :

private System.Windows.Forms.RichTextBox richTextBox1

The main component of the text editor in which the user will write his/her code to.

private System.ComponentModel.Container components

Set to null initially. Indicates the container that the GUI objects will be in.

private System.Windows.Forms.SaveFileDialog saveFileDialog1

This dialog box appears when the user wants to save a newly written code text. This enables the user to explore the directories and select a directory to save the code file. If a previously saved code text is edited and wanted to be saved, just the file is saved, this doesn't appear.

private string fullPath

Set to string "" (empty string) initially. This acts as an intermediate storage for the path of the file to be saved. This path information comes from the save file dialog that is shown to the user before.

private string outFile

Set to string "" (empty string) initially. If the fullPath variable is set (that is if it is not an empty string), this variable is assigned to the fullPath variable, since this variable stands for the path of the output file to be saved.

private bool is_File_Opened

Set to boolean false initially. This variable indicates whether the current code file is saved before (in the current session or previous sessions) or not. According to this variable, save file dialog is shown to the user or not.

private FindDialogBox myFindDialogBox

A variable from FindDialogBox class. This stands for the find dialog box when the user clicks "Find".

private replace myReplaceDialogBox

A variable from replace class. This stands for the replace dialog box when the user clicks "Replace".

private int nextFindIndex

Set to int 0 initially. This variable stands for the index that will be used as a starting point for the next string to be found when the user clicks “Find Next” in replace dialog box or find dialog box.

private System.Windows.Forms.RichTextBox richTextBox2

This richTextBox variable will be used to show the line numbers on the left hand side of the richTextBox1 (namely, the text editor). This text box will be read-only (in other words, editing will not be allowed).

private bool isReachedEnd

Set to boolean false initially. Indicates whether the end of file is reached or not during the replace and find sessions. According to this variable, if it is true, a yes-no message box is shown to the user indicating that the end of file is reached and asks to continue from the beginning of the file or not.

private bool replacement_made

Set to boolean false initially. This is a precaution for the insertion of a text again and again when the user clicks “Replace this Occurence” multiple times. Set to true when the user makes a replacement in the text.

Methods:

public void saveClicked() :

Handles the events to occur when the user clicks “Save” from menu.

public string getText() :

Returns the text written in text editor.

public RichTextBox getRichText() :

Returns the richTextBox object in which the user writes his/her code.

public void setText(string output) :

Assigns the given string argument to the editor's richTextBox object's text. This is used in opening a previously saved file. The contents of the argument is copied to the text editor's empty richTextBox and shown to user.

public void setFileOpenedState(bool state) :

Assigns the given boolean state to the is_File_Opened variable which is a boolean state, too. Set to true when the user opens a file, or saves. Otherwise, false. According to this variable's state, a save file dialog is shown to the user or not.

public bool getFileOpenedState() :

Returns the boolean variable is_File_Opened that determines whether the code file is previously saved or not.

public void setPath(string path) :

Assigns the given string, which is a path, to the output file's path. The output path is represented by the variable outFile in class.

public string getPath() :

Returns the path of the currently opened file.

public void Cut() :

Cuts the selected text from the editor.

public void Copy() :

Copies the selected text from the editor.

public void Paste() :

Pastes the copied or cut text from the editor to the point where the code text's cursor stands.

public void Undo() :

Undoes the last action made in text editor.

public void Redo() :

Redoes the last action made in text editor.

public void SelectAll() :

Selects all the text in the text editor.

public void Find() :

Lets the user to search for a string in the code text. Opens a find frame (which is a class FindDialogBox variable named myFindDialogBox) to enable the user to enter what he/she will search.

public void Replace() :

Lets the user to replace a string with another string in the code text. Opens a replace frame (which is a class replace variable named myReplaceDialogBox) to enable the user to search for a given string, replace it or not, or to replace all of the occurences of this text with new one in the code text.

private void findNextClicked(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Find Next” button in the find next frame.

private void findNextInReplaceClicked(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Find Next” button in the replace frame.

private void replaceThisInReplaceClicked(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Replace This Occurence” button in the replace frame.

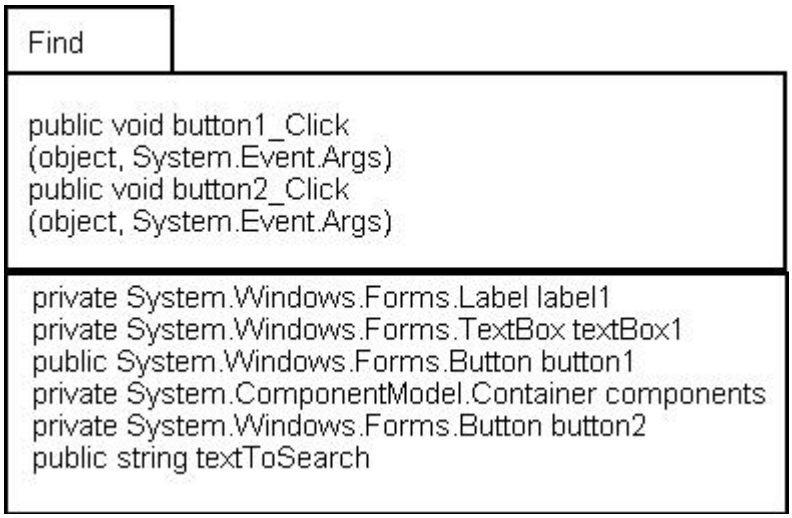
private void replaceAllInReplaceClicked(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Replace All” button in the replace frame.

private void saveFileDialog1_FileOk(object sender,
System.ComponentModel.CancelEventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “OK” in the save file dialog box.

3.5.6 Class FindDialogBox :



Variables :

private System.Windows.Forms.Label label1

The label that is set to “Text to Search” and shown to the user.

private System.Windows.Forms.TextBox textBox1

Stands for the text box that the user enters a text to search in the code text.

public System.Windows.Forms.Button button1

Stands for the button named as “Find Next”.

private System.ComponentModel.Container components

Set to null initially. Indicates the container that the GUI objects will be in.

private System.Windows.Forms.Button button2

Stands for the button named as “Close”.

public string textToSearch

Set to string “” (empty string) initially. Assigned to the text entered in the textBox1 variable in the process.

Methods :

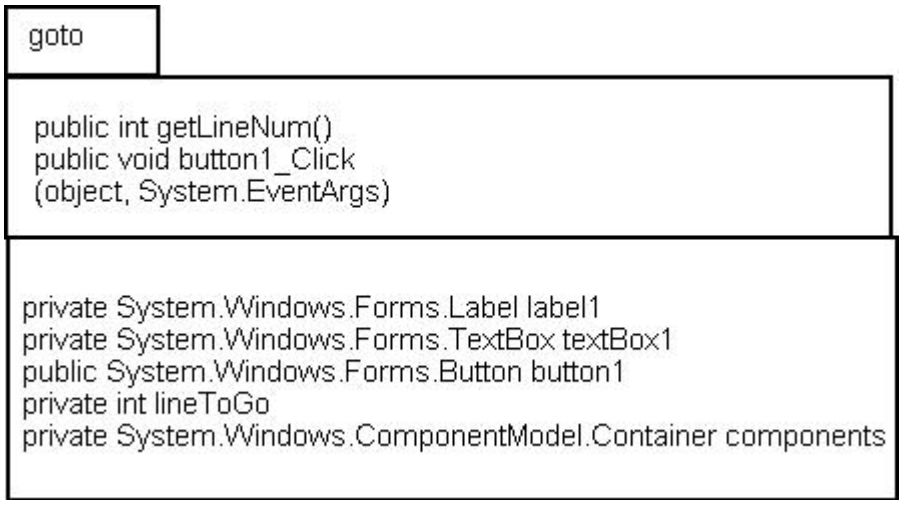
public void button1_Click(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Find Next” button.

private void button2_Click(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Close” button in the find next frame.

3.5.7 Class goTo :



Variables :

private System.Windows.Forms.Label label1

The label that is set to “Enter Line Number to Go” and shown to the user.

private System.Windows.Forms.TextBox textBox1

Stands for the text box that the user enters a line number to go in the code text.

public System.Windows.Forms.Button button1

Stands for the button named as “Go”.

private int lineToGo

Set to int 1 initially. Indicates the line number to go in the textBox1 variable.

private System.ComponentModel.Container components

Set to null initially. Indicates the container that the GUI objects will be in.

Methods :

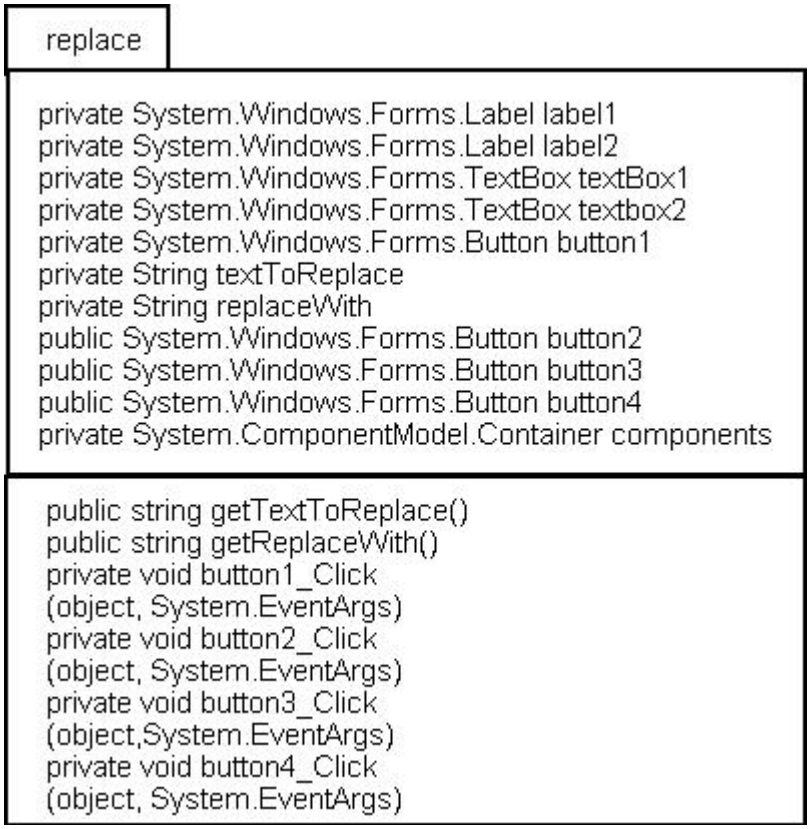
public int getLineNum() :

Returns the line number to go (namely, lineToGo variable).

private void button1_Click(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Go” button in the “go to line” frame.

3.5.8 Class replace :



Variables :

private System.Windows.Forms.Label label1

The label that is set to “Text to Find” and shown to the user.

private System.Windows.Forms.Label label2

The label that is set to “Replace With” and shown to the user.

private System.Windows.Forms.TextBox textBox1

Stands for the text box that the user enters a text to search in the code text.

private System.Windows.Forms.TextBox textBox2

Stands for the text box that the user enters a text to replace with the one entered in textBox1 in the code text.

public System.Windows.Forms.Button button1

Stands for the button named as “Find Next”.

private string textToReplace

Set to string “” (empty string) initially. Indicates the text to find and/or replace in the textBox1 variable.

private string replaceWith

Set to string “” (empty string) initially. Indicates the text to replace with the one entered in the textBox1 variable.

public System.Windows.Forms.Button button2

Stands for the button named as “Replace This Occurence”.

public System.Windows.Forms.Button button3

Stands for the button named as “Replace All”.

private System.Windows.Forms.Button button4

Stands for the button named as “Close”.

private System.ComponentModel.Container components

Set to null initially. Indicates the container that the GUI objects will be in.

Methods :

public string getTextToReplace() :

Returns the textToReplace variable.

public string getReplaceWith() :

Returns the replaceWith variable.

private void button1_Click(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Find Next” button.

private void button2_Click(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Replace This Occurence” button.

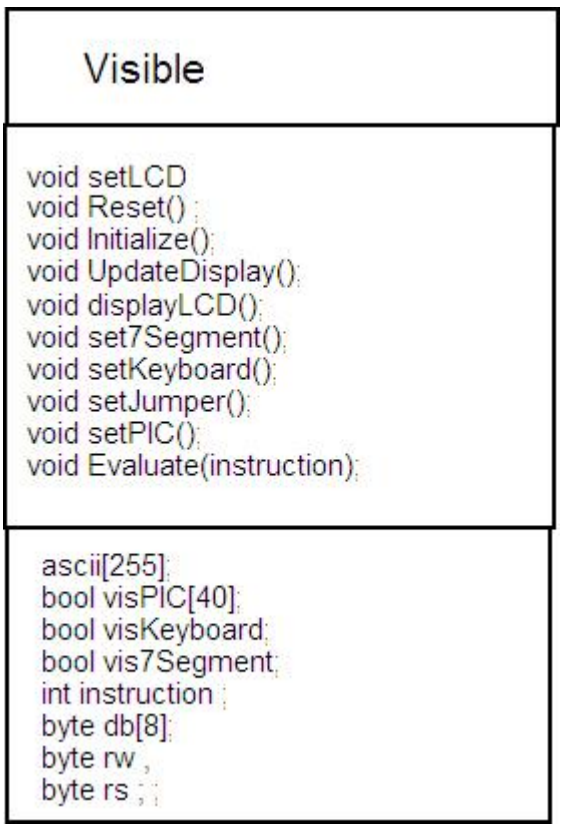
private void button3_Click(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Replace All” button.

private void button4_Click(object sender, System.EventArgs e) :

This function is an event handler. It handles the events occur when the user clicks “Close” button.

3.5.9 Class Visible



As we mentioned in previous progresses we will use an Visual CENG Embedded Card named ‘Visible ‘ . By touching at the buttons on Visible user can change the state of the Visible card. – The Visible figure is shown at GUI part- A program burned to the Visible by our RCSim program will change the state of the Visible . For example the 7-segment display on visible will change its state on Visible.

Void setLCD ()
The datas affecting LCD display are managed by this method . The inner methods for the LCD display is managed by this method.

Void Reset()
resets LCD

Void initialize()
initializes LCD display

Void updateDisplay()
updates the LCD display

Void evaluateInstr(instruction)
LCD instruction evaluation function managed by setLCD method .

displayLCD(byte db[7], byte rw, byte rs)
LCD display method , adapted from the Hitachi HD44780 model .

Void set7segment()
The datas affecting 7 Segment display are arranged by this method .

Void setKeyboard()
The situation of the keyboard buttons are managed by this function .

Void setJumper ()
The situation of the keyboard buttons are managed by this function .

Void setPic()

The situation of the PIC on Visible arranged by this method .

Attributes.

ascii[255]

ascii character needed by LCD display

bool VisPIC [40]

This will be used in setVisible(). The values of the buttons at Visible PIC pins will be kept on this datas. There are 20 left 20 right total 40 pins of our pic.

bool VisKeyboard[16] . This attributes will also be handled by setVisible() method . The keyboard have 16 buttons ; each are dependent to some value in data memory , the related values will be kept in this datas.

bool Vis7segment[3][7]

Visible has also a 3 x 7Segment Display . for all we need 7 lines each are bool –light on light off-

int instruction

instructions used in LCD display

byte db[7]

datas connected to LCD display

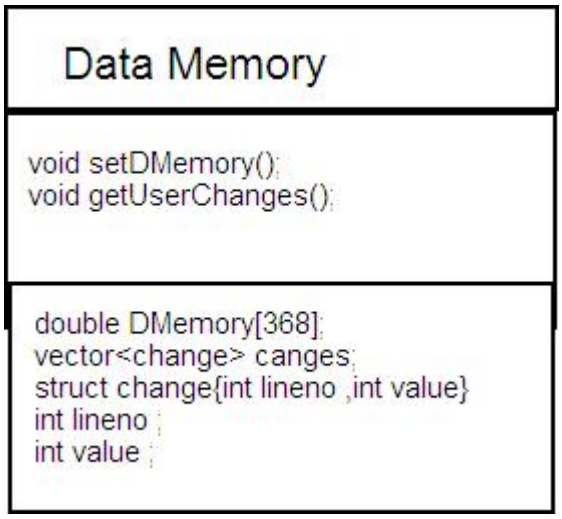
byte rw

LCD reads or writes

byte rs

LCD should reset or set ; when 0 set ,1 reset .

3.5.10 Class DMemory



The data memory class , managing data memory visualization

Void setDMemory() The changes made to the data memory made from simulater , debugger is simultaneously made to this class dMemory[] values by this method .

Void getUserchanges(vector<change>) the changes obtained from the user are made in the data memory .

Attributes

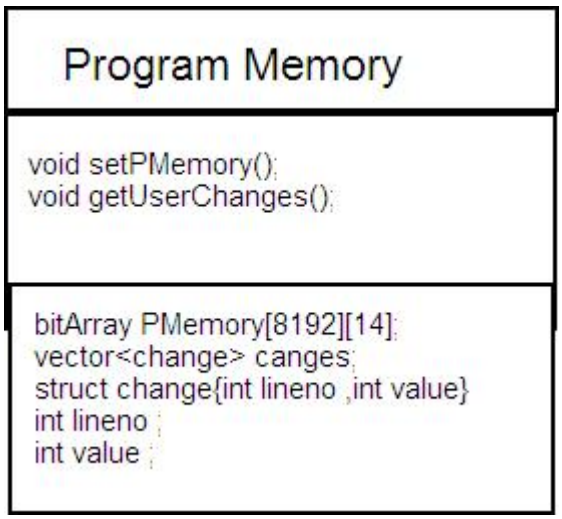
double dMemory[368] ;
the values of data memory

vector <struct> change { int linenumber , int tempvalue }
the changes are kept until user sent this to RCsim in an array.

int linenumber ;
the current linenumber int data memory ;

int tempvalue ;
the value given by the user

3.5.11 Class PMemory



The data memory class , managing data memory visualization

Void setPMemory() The changes made to the data memory made from simulater , debugger is simultaneously made to this class dMemory[] values by this method .

Void getUserchanges(vector<change>) the changes obtained from the user are made in the data memory .

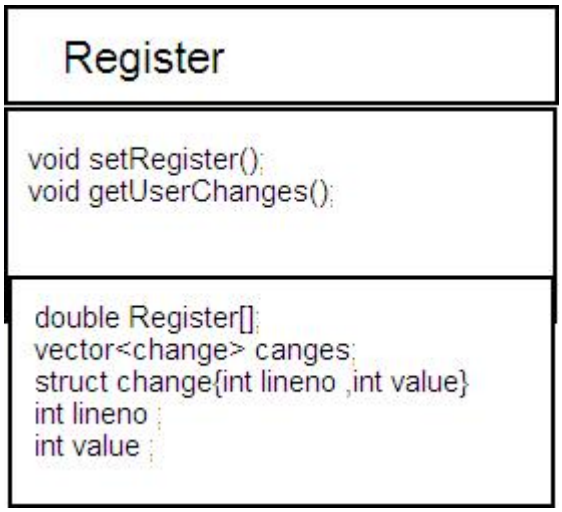
Attributes
double PMemory[8192] ;
the values of data memory

vector <struct> change { int linenumber , int tempvalue }
the changes are kept until user sent this to RCsim in an array.

int linenumber ;
the current linenumber int data memory ;

int tempvalue ; temporary value

3.5.12 Class Register



The data memory class , managing data memory visualization

Void setRegister() The changes made to the data memory made from simulater , debugger is simultaneously made to this register[] values by this method .

Void getUserchanges(vector<change>) the changes obtained from the user are made in the data memory .

Attributes

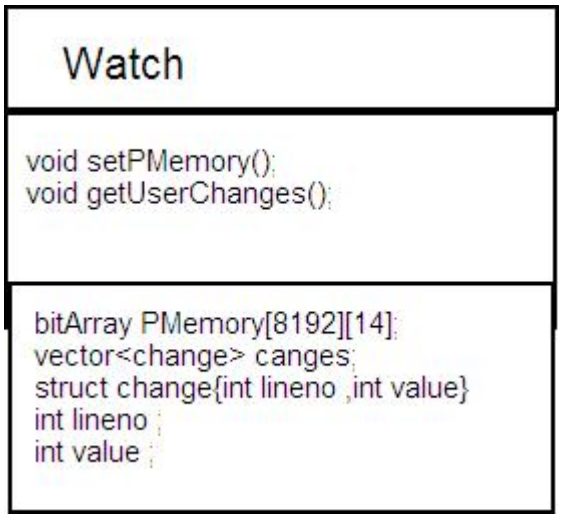
Double register [368] ;
the values of data memory

vector <struct> change { int linenumber , int tempvalue }
the changes are kept until user sent this to RCsim in an array.

int linenumber ;
the current linenumber int data memory ;

int tempvalue ;
the value given by the user

Class Watch



The data memory class , managing data memory visualization

Void setWatch() The changes made to the data memory made from simulater , debugger is simultaneously made to this vector values by this method .

Void getUserchanges(vector<change>) the changes obtained from the user are made in the data memory .

Attributes

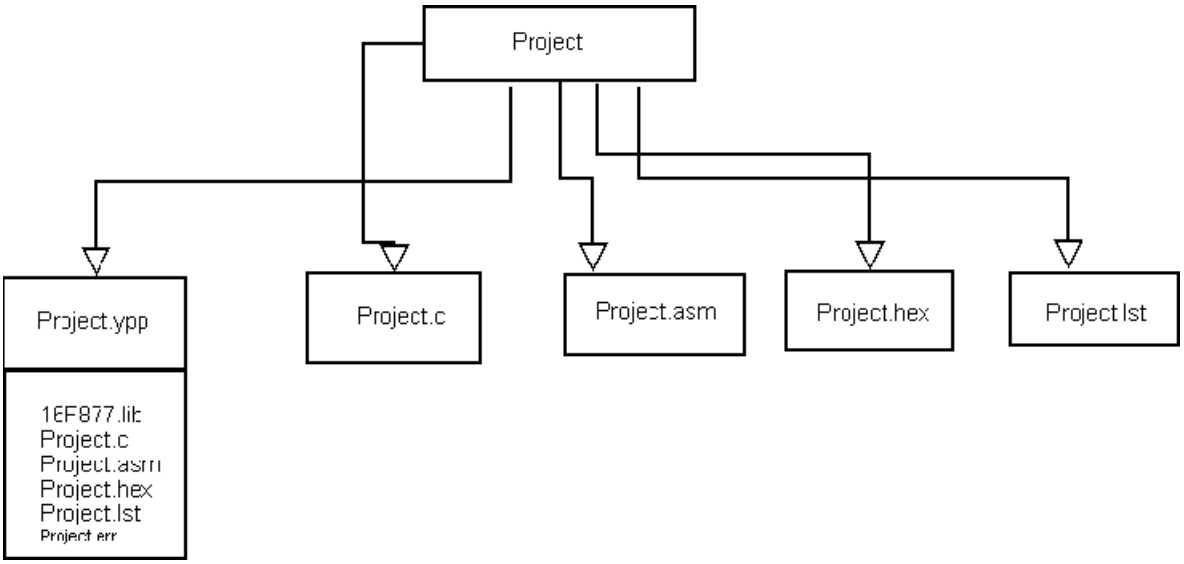
Vector<double >Watch ;
the values of data memory

vector <struct> change { int linenumber , int tempvalue }
the changes are kept until user sent this to RCsim in an array.

int linenumber ;
the current linenumber int data memory ;

int tempvalue ;
the value given by the user

4. Project's File Structure



1.

1. Project.yyp

".yyp" is the file extension of our main project file. This include links to

- project.c – if the code is written in C language
- project.asm
- project.lst
- project .err
- project.hex
- other files

This file also will have records of the settings of the project , like date of last modification.

2. Project.c

This is the source file used in our project, if the program is written in C language but not in assemble.

3. Project.asm

This is the assembly source file used in our project. There are two sources of this file

1. A user can directly write the assembly program
2. The user can write C codes, the compiler converts it into assembly language.

4. Project.lst

This is the file obtained from MPASM assembler program. This file is important for our system that we will use this in our simulation part. In this file format symbol table of the assembly file is given. The user can never give his/her .lst file to our system . Only our program will generate the .lst file.

5. Project.err

This file is generated from our program as an output. If there exists any error during the compiler, assembler , linker , burn period this file will be generated automatically . The user will also see this text in the output file window on the program.

6. Project.hex

This file is the product of the MPASM assembler. The only format that can be sent to the program is this file format. The user can never give his/her .hex file to our system. Only, our program will generate the .hex file.

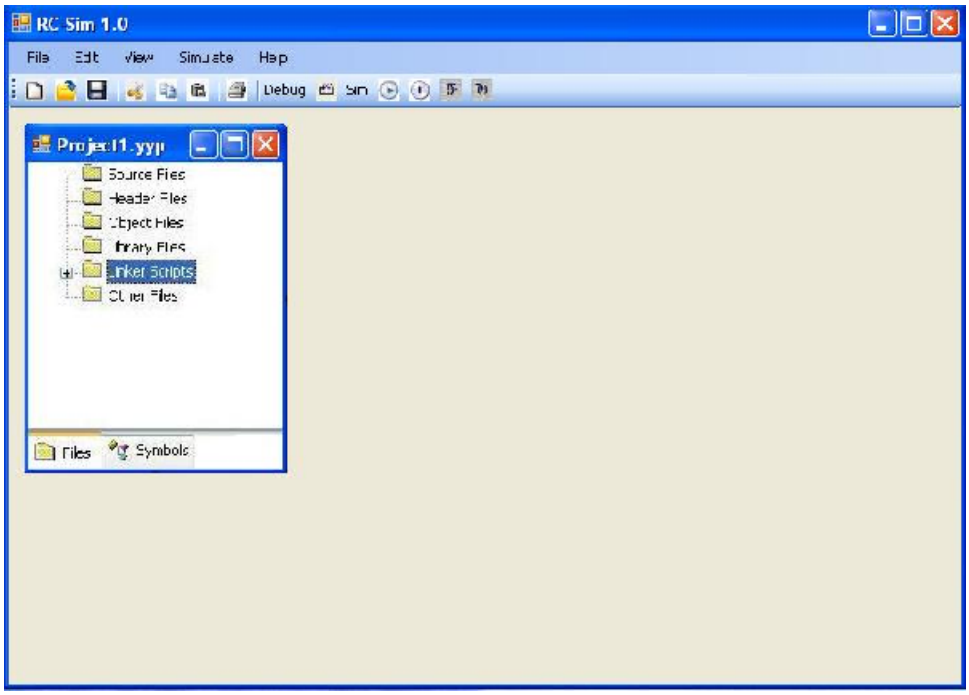
7. Other files

If the user wants to include some other files, we will categorize this files as a part of this format For example properties of PIC is here.

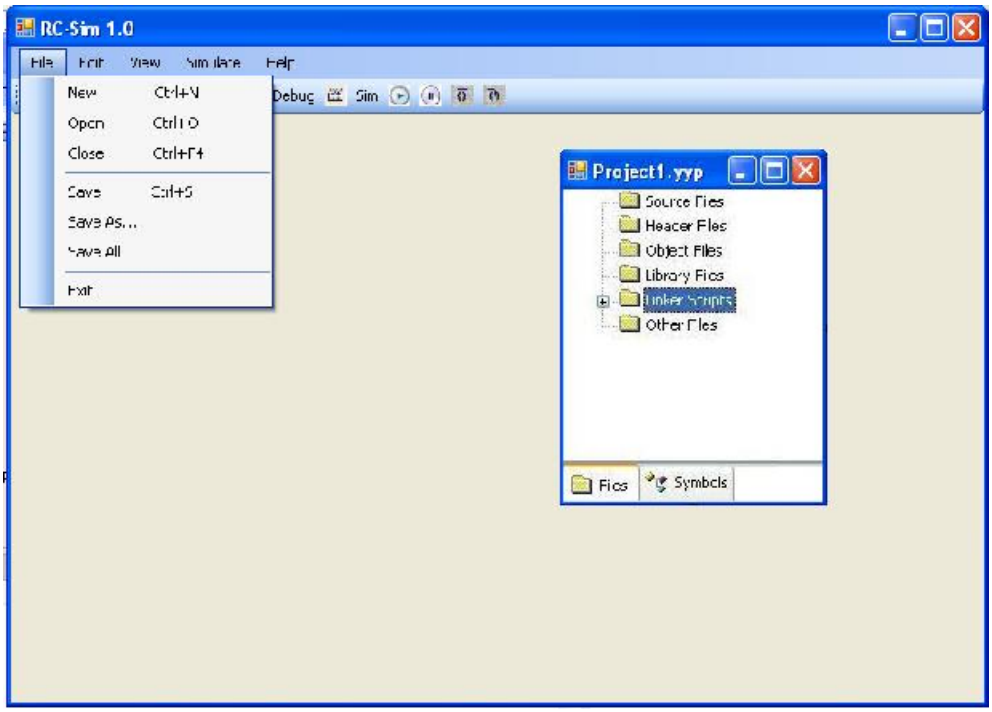
5. User Interface Design (Menus)

In this part, menus of RCSim will be explained with some screenshots.

General view of RCSim will be as follows:



5.1 File Menu



a. New

Creates a new project with blank code editor.

b. Open

Opens the dialog box to select a saved project or source files

c. Close

Closes the current project but not the program

d. Save

Saves changes on current active window.

e. Save As

Saves active window with different names as a different project or file.

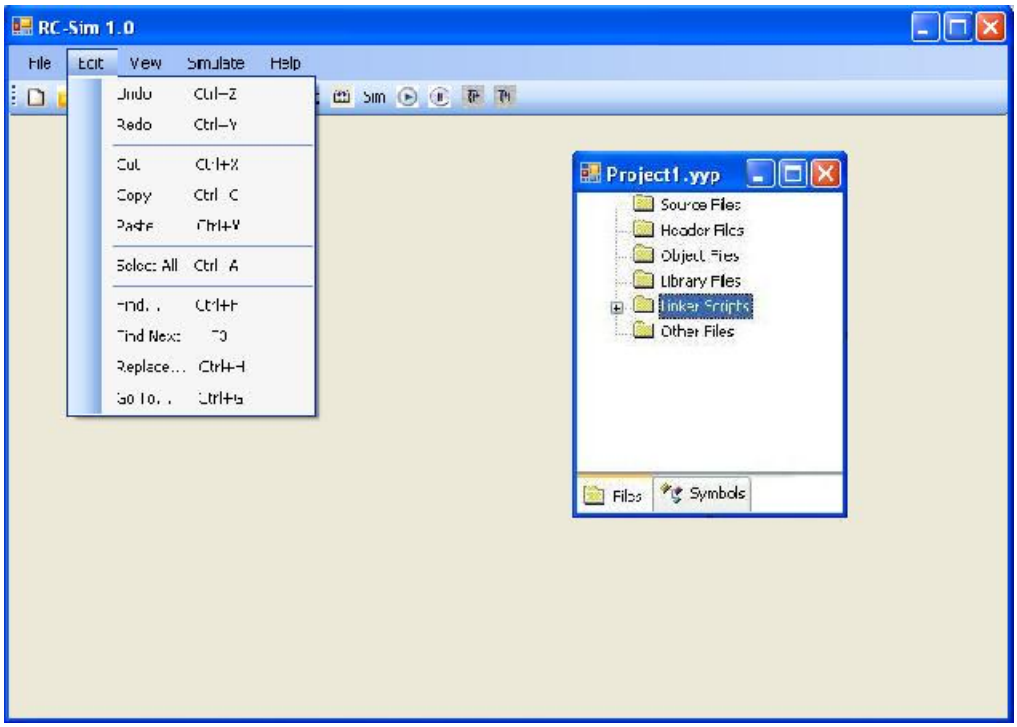
f. Save All

Saves all changes in all windows.

g. Exit

Closes all windows and exits the program.

5.2 Edit Menu



a.Undo

Takes the last action back.

b.Redo

Takes the last undo action back again.

c.Cut

Cuts the selected text and copies to the memory.

d.Copy

Copies the selected text to the memory.

e.Paste

Pastes – writes the copied text.

f.Select All

Selects all the texts in current window.

g.Find

Opens the dialog box to find a word or phrase.

h.Find Next

Finds the next item of the searched word or phrase.

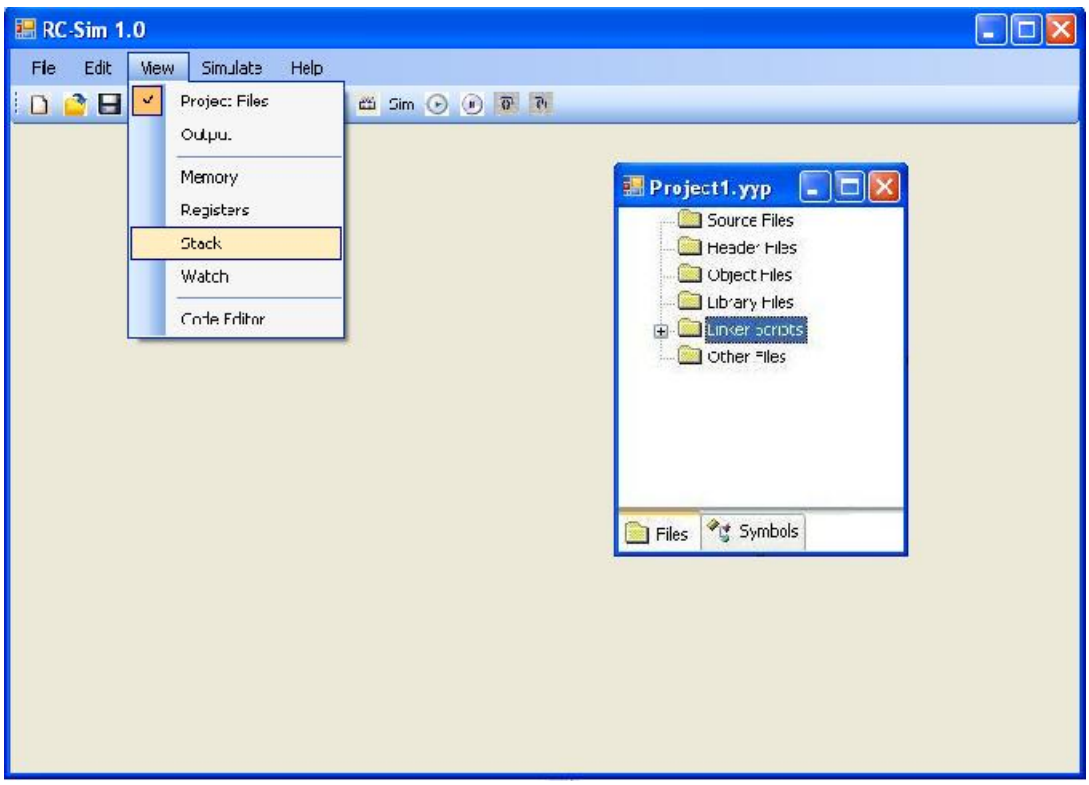
i.Replace

Opens the dialog box to replace the word(s) with the written ones.

j.Go to

Opens a dialog box to go to the line indicated by the user.

5.3 View Menu



a.Project Files

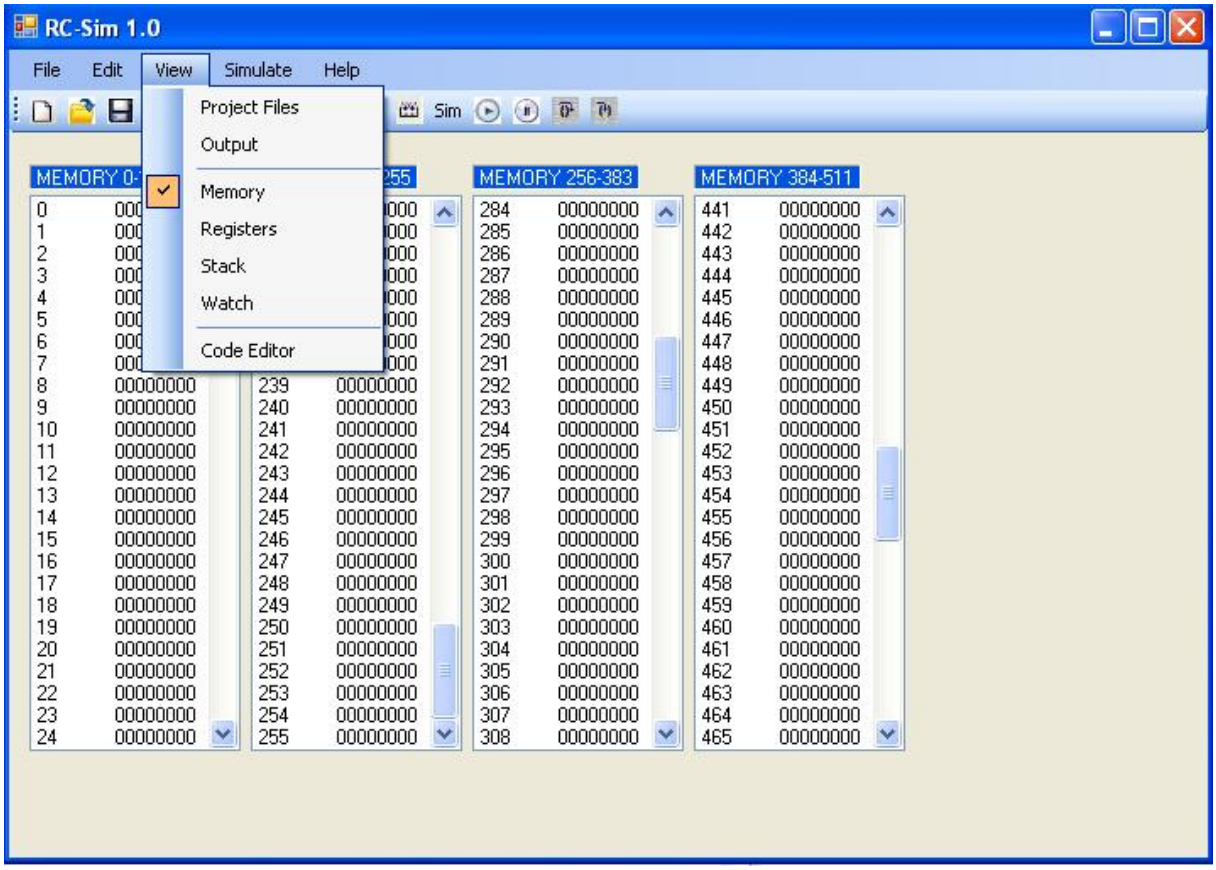
Shows project's files.

b.Output

Shows the output file.

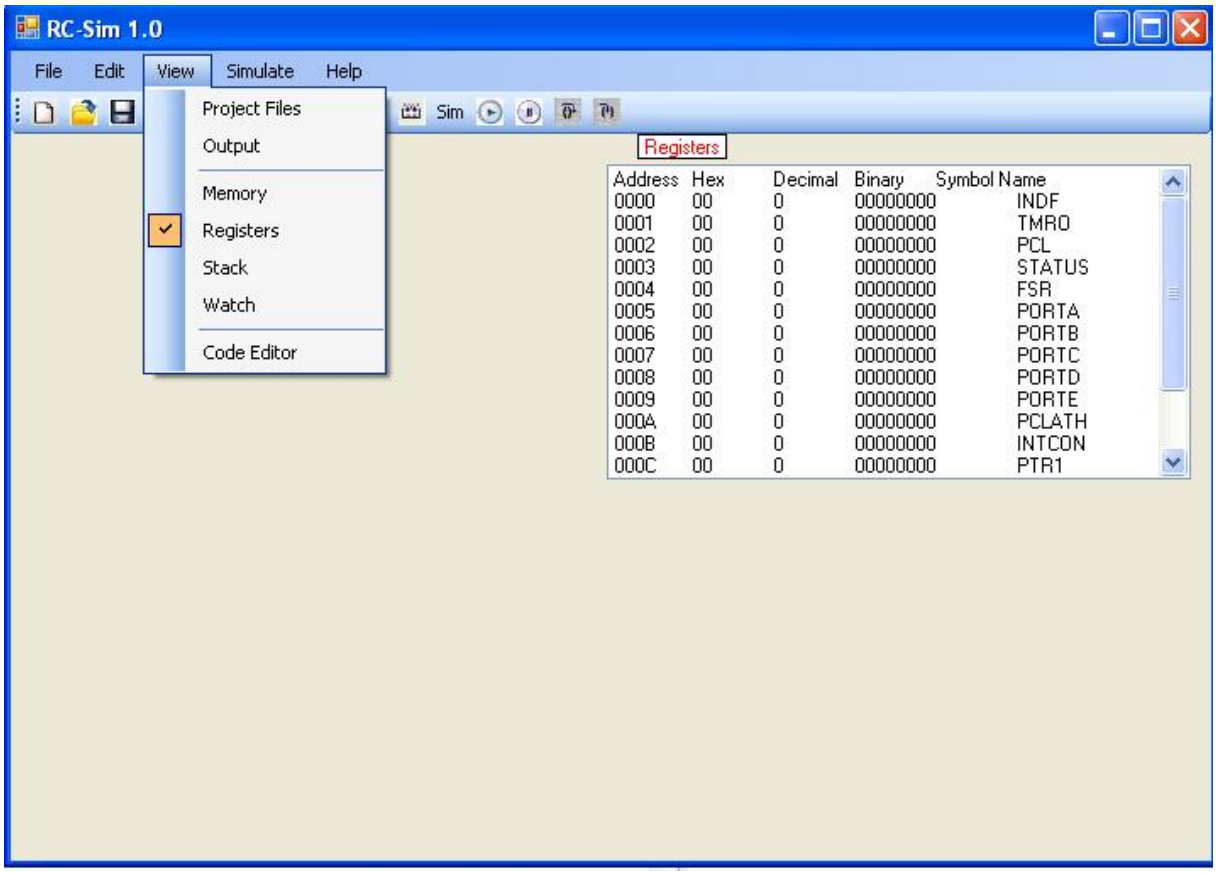
c.Memory

Shows the conditon of the memory.



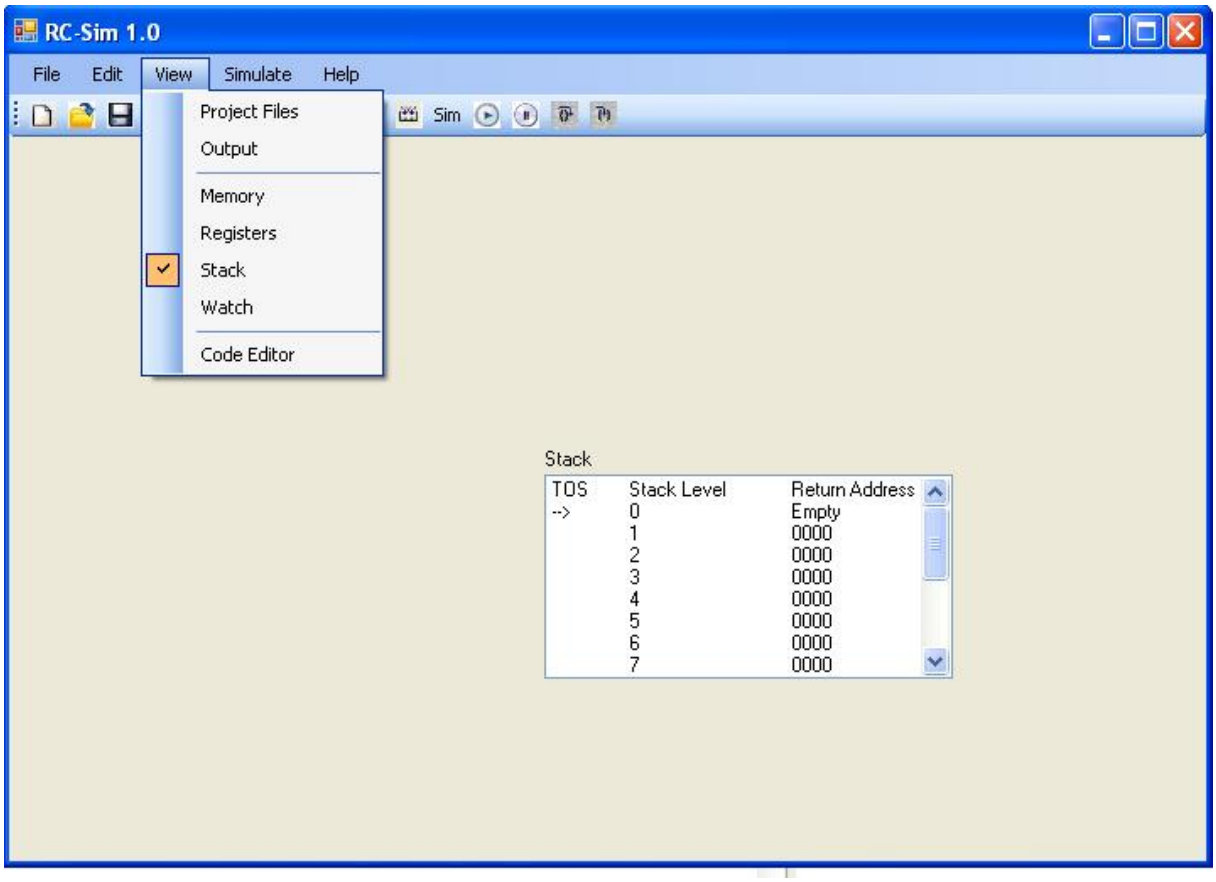
d.Registers

Shows the conditon of registers.



e.Stack

Shows the conditon of the stack of processor.



f.Watch

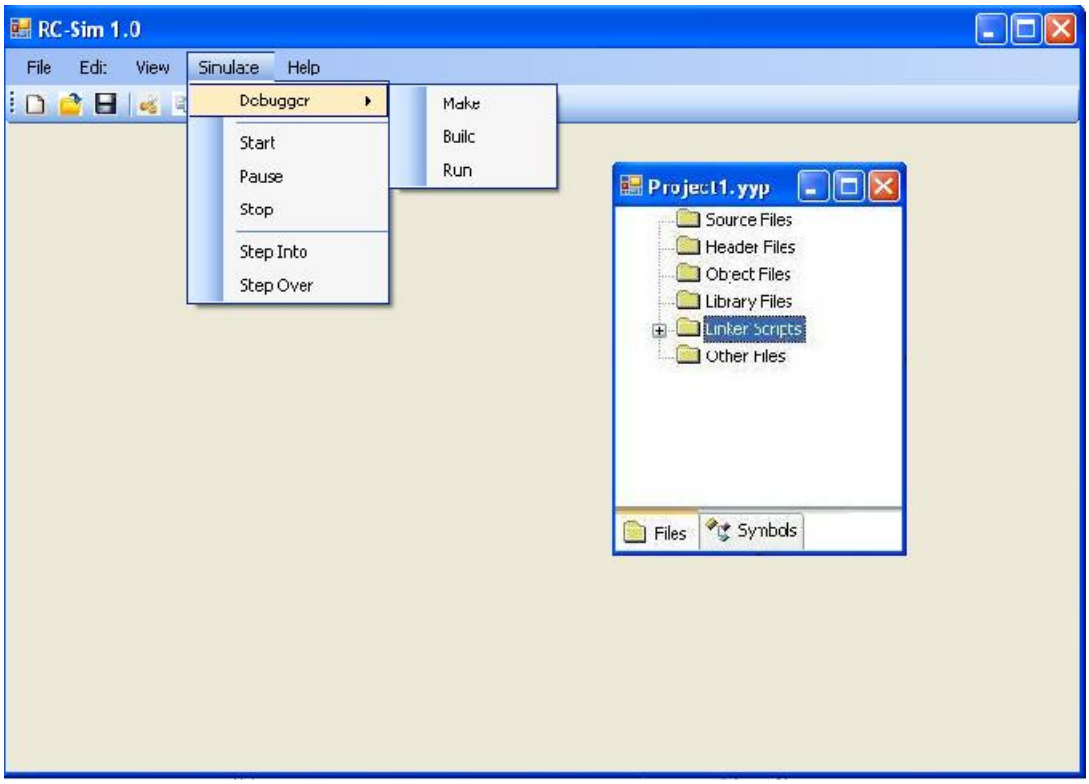
Opens the dialog box to watch the indicated breakpoints.

g.Code Editor

Opens the editor for programmer.

5.4 Simulate Menu

Contains the menu for debugger.



a. Debugger

- Make : Rebuilds an application, re-compiling only those source files that have changed since the last complete compilation
- Build : Compiles the application
- Run : Runs the simulator

b. Start

Starts the simulation or continues paused simulation

c. Pause

Breaks the simulator

d. Stop

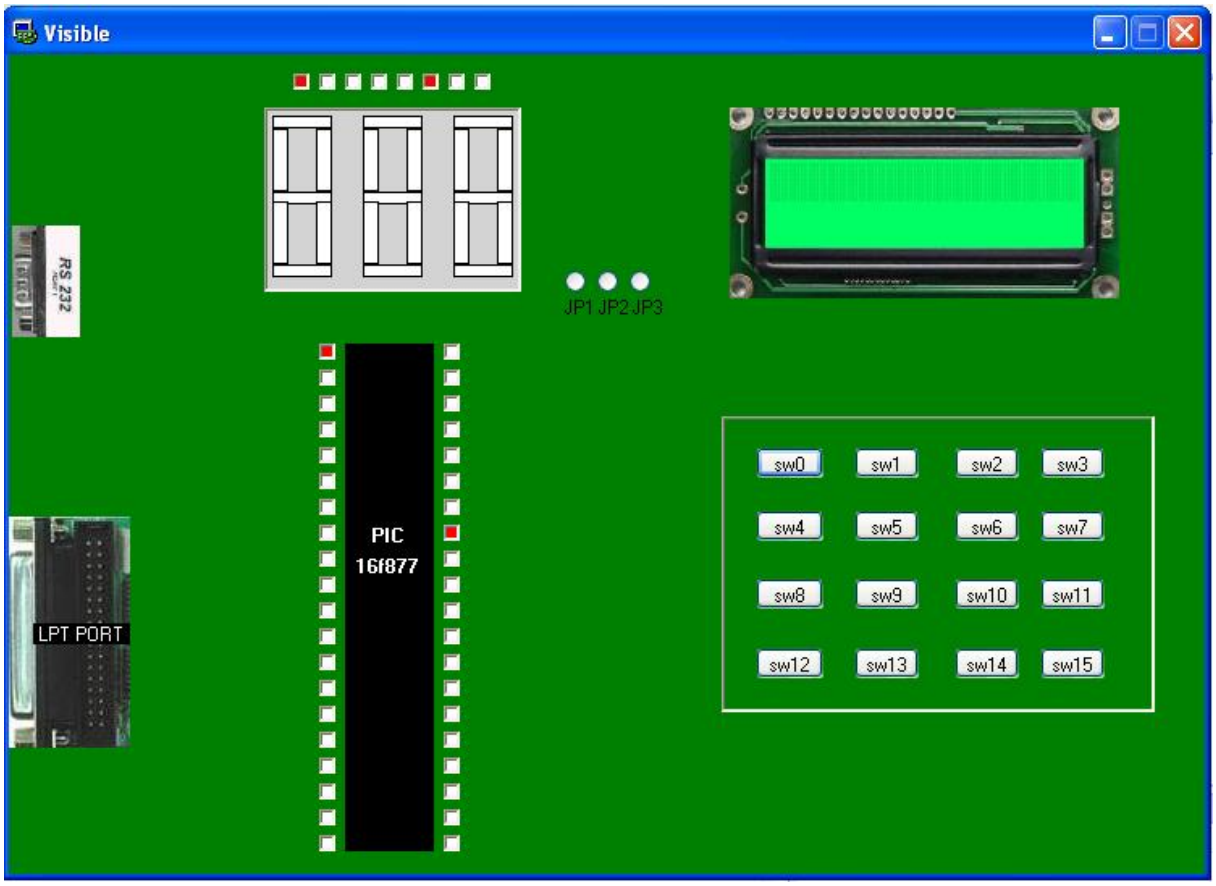
Stops the simulator.

e. Step Into

Steps though code, one instruction at a time.

f. Step Over

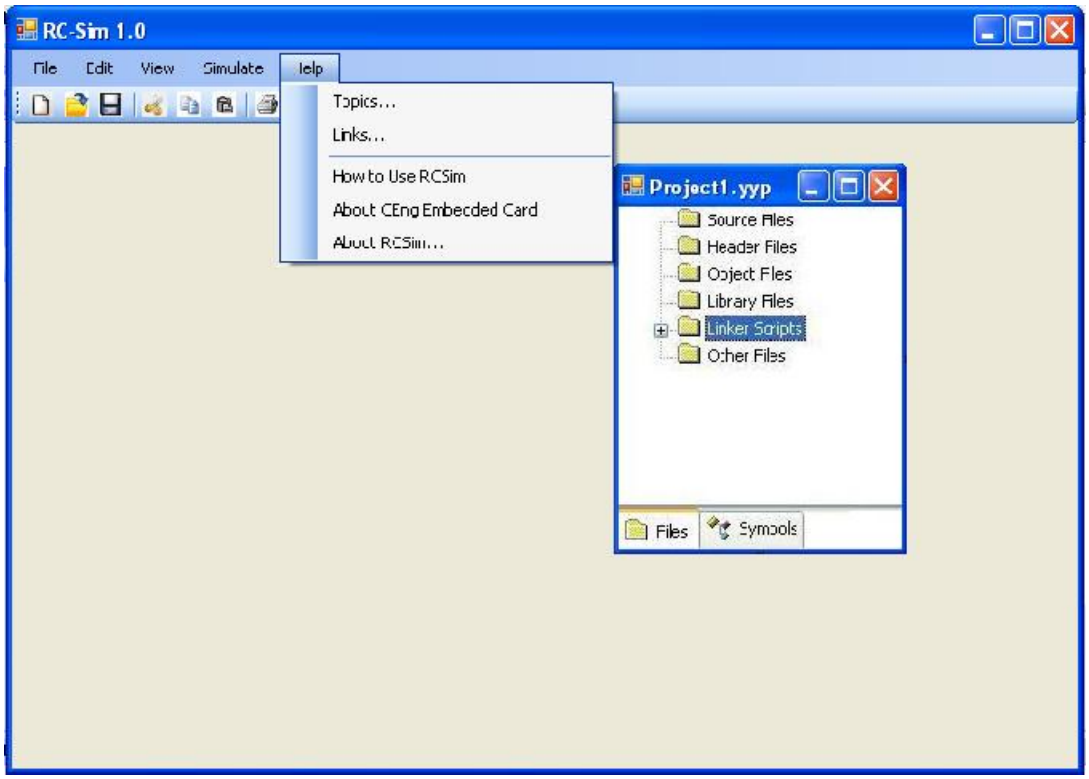
Allows you to step over subroutines. This command executes the code in the subroutine and then stops execution at the return address to the subroutine.



The visible Card is shown above

By using this card user will show the LCD display status on screen . User can give input to our product by using 16 Keyboard buttons . Simultaneously the system will adapt to its situation. PIC and 7 Segment display are also shown above.

5.5 Help Menu



a. Topics

Opens an help menu based on related topics.

b. Links

Opens a box contains links about card, PICs etc...

c. How to Use RCSim

Opens users guide to RCSim. It will be very clear and understandable in order to help Ceng336 students. Moreover, we will provide some example codes. These codes will be asked from instructors of Ceng336 course also.

d. About Ceng Embedded Card

Opens a box contains information about card. Determining this information will be done by the help of instructors of Ceng336 course.

d. About RCSim

It’s all about us and our project.