MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

CENG 491
COMPUTER ENGINEERING DESIGN I

INITIAL DESIGN REPORT

YAYA BİLİŞİM

YASİN ALPEN	1297431
KAAN Y. CEYLAN	1347269
YUNUS EŞENÇAYI	1347459
AHMET TAHİR UÇKUN	1298371

INDEX

1. Project Description

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

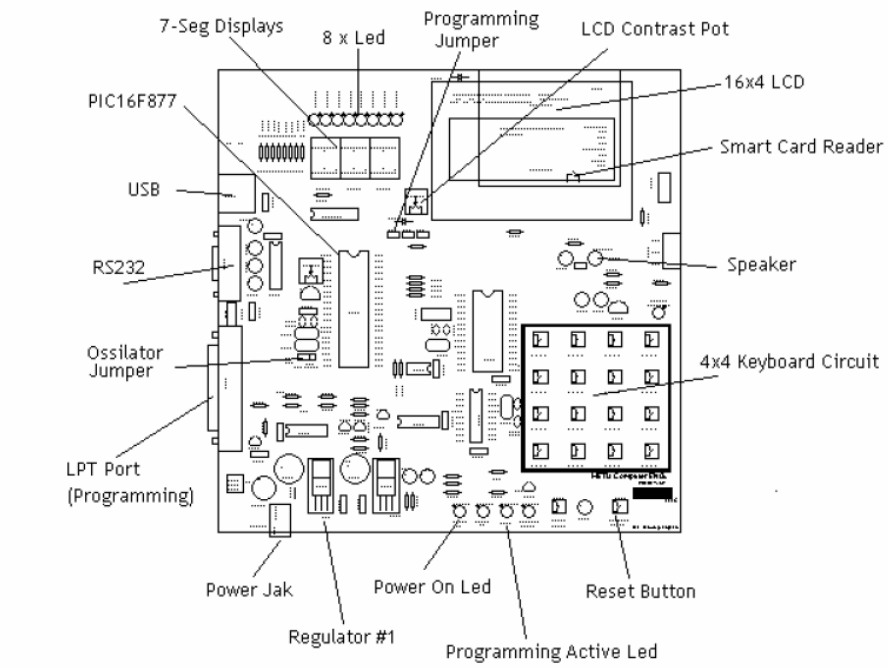
There are a lot of microcontroller manufacturers and they are named according to their manufacturers. PIC (Peripheral Interface controller) is the one produced by Microchip. PICs have Harvard architecture but not Von Neuman.

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:

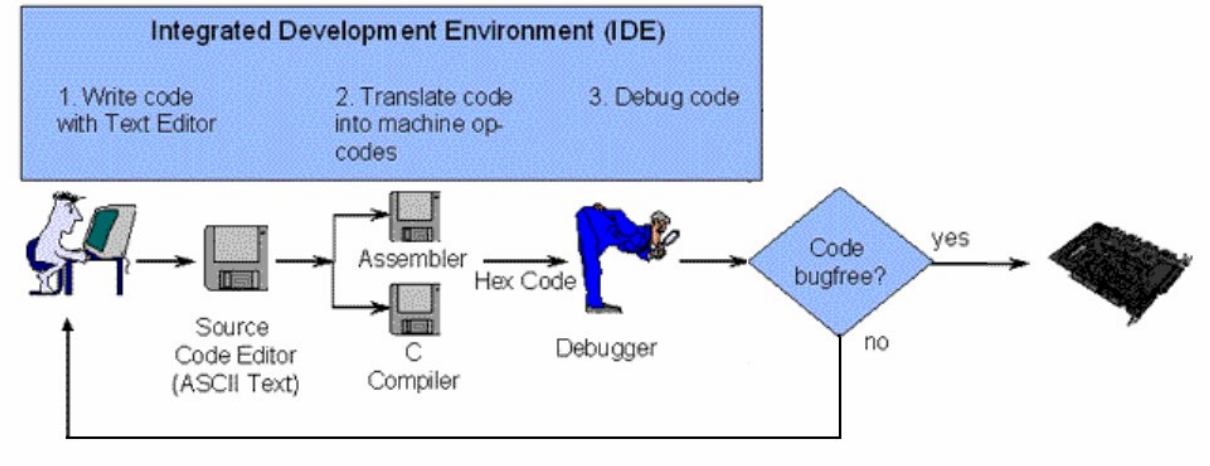
- 14 bit cores with 35 instructions.
- 200 ns instruction time
- 8092 14 bit Flash program memory
- 368 8 bit data memory or registers(RAM)
- 256 8 bit EEPROM data registers
- 8 level hardware stack
- Up to 14 interrupt capability
- 33 I/O pin
- 3 timer/ counter modules
- 10 bit 8 channel A/D converter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

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 were not comfortable with our position up to the requirements report, we spent much more time for initial design. After submitting our requirements report, we first tried to go over requirements process again to understand what is going on better.

Individually we all studied the basic books about PIC processors and useful websites related to our are. Some of them are :

http://www.microchip.com
http://www.ceng.metu.edu.tr/courses/ceng336
http://www.gnupic.org
http://www.mikroelektronika.co.yu/english/product/books/PICBook/
http://www.elec.rdg.ac.uk
http://www.picemulator.com
http://www.piclist.com

After completing this part, we've started to design process. During the process of initial design, we've set some meeting hours according to our syllables. In the last week, we also set some extra meeting hours at the weekend.

Meeting hours:

Monday	15.40 – 17.30	
Wednesday	12.40 – 15.00	(at 13.20 meeting with Mr. Bayer)
Thursday	14.40 – 16.30	
Friday	15.40 – 17.30	

Meeting Places: Library Reserve && Ceng Digital Laboratory

In addition to this, we met with Mr. Bayer every Wednesday as routine. He controlled overall process. Some meetings were vital for the designing process.

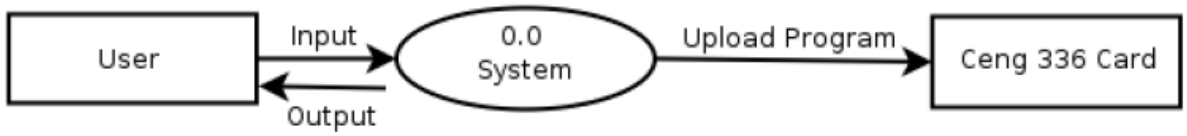
Moreover, we also had a meeting with Mr. Kilic. Although he has many things to do, he did not reject us.

Since Mr. Kilic is the designer of CEng Embedded Card, it is very important for us to have meetings with him. He acts like a costumer for us. Having a costumer in designing stage is very important.

3. Diagrams

3.1 Data Flow Diagrams

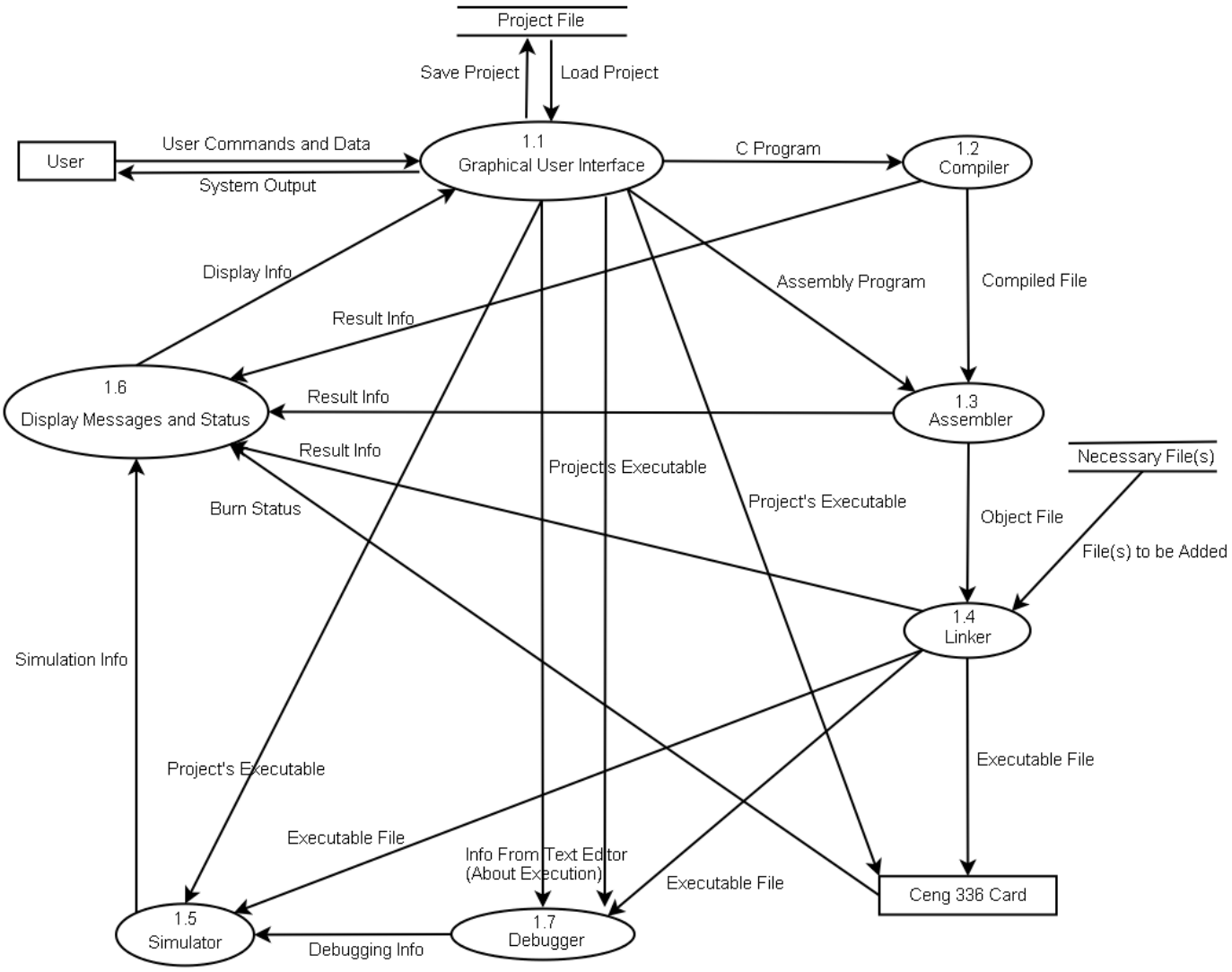
3.1.1Level 0 DFD



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 Level 1 DFD



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.

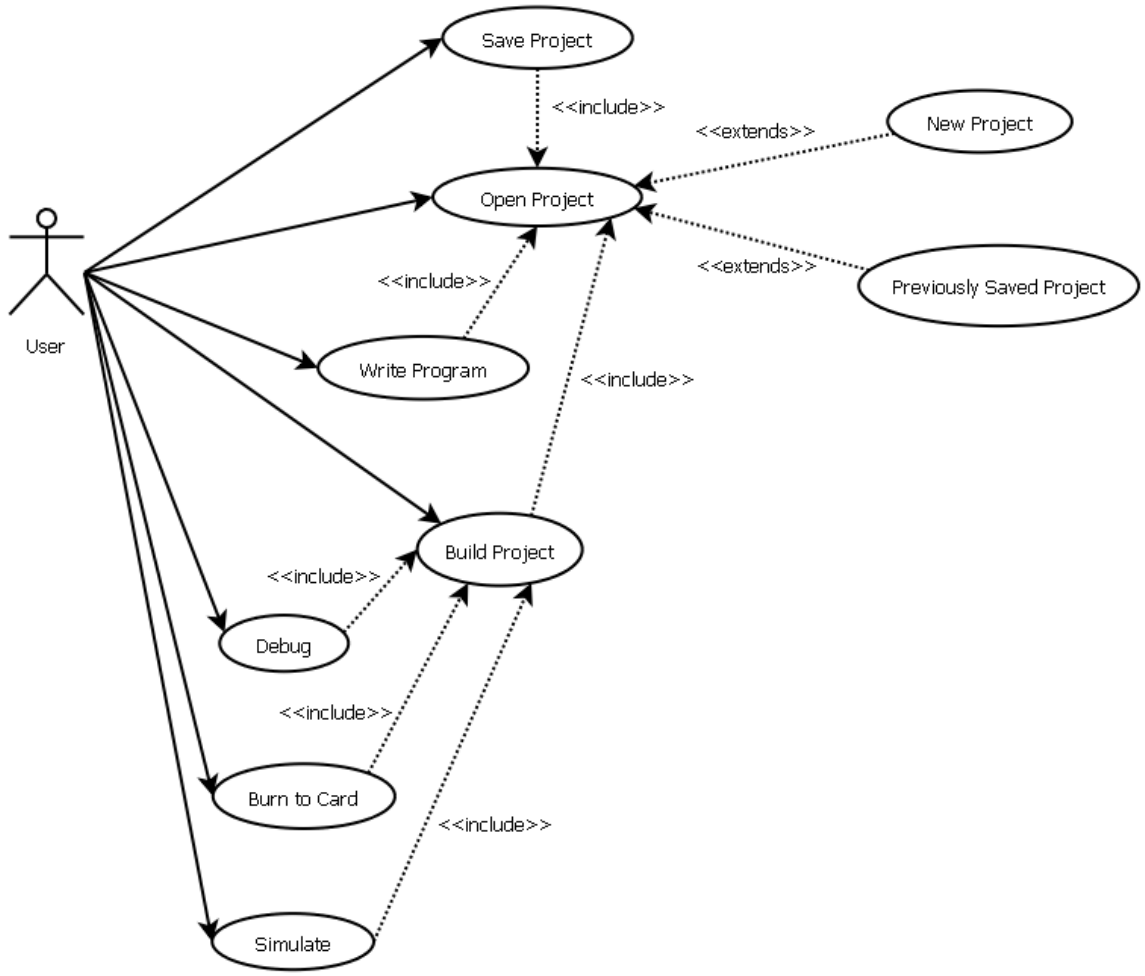
Name:	Burn Status
Aliases:	None
Product of:	Ceng 336 Card
Where used:	Display Messages and Status (Process 1.6)
Description:	Message indicating whether the executable file is successfully written to the card or not.

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.

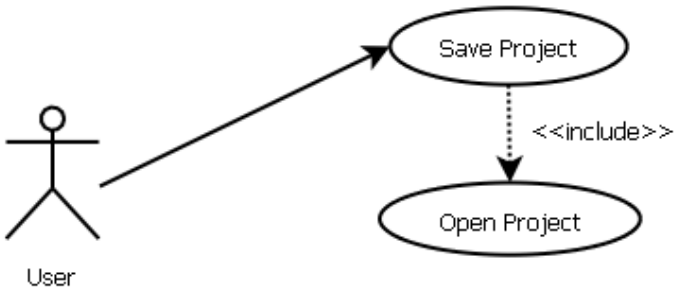
Name:	Load Project
Aliases:	None
Product of:	Project File
Where used:	Graphical User Interface (Process 1.1)
Description:	Code and the settings of the project to be loaded.

3.3 Use Case Diagrams



Use Case Diagram

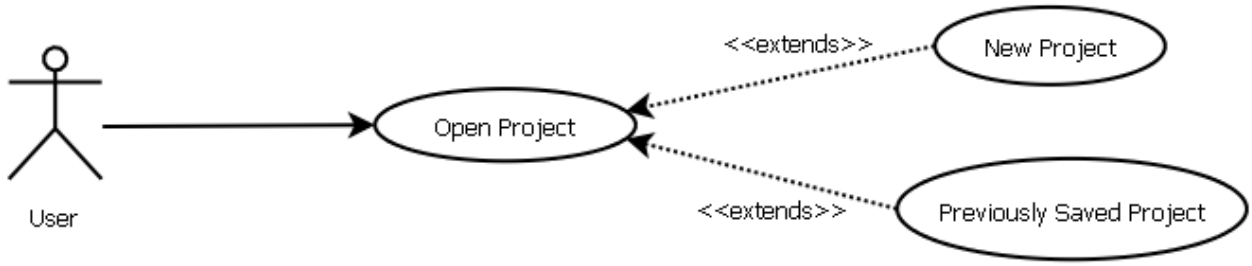
3.3.1 Save Project



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.

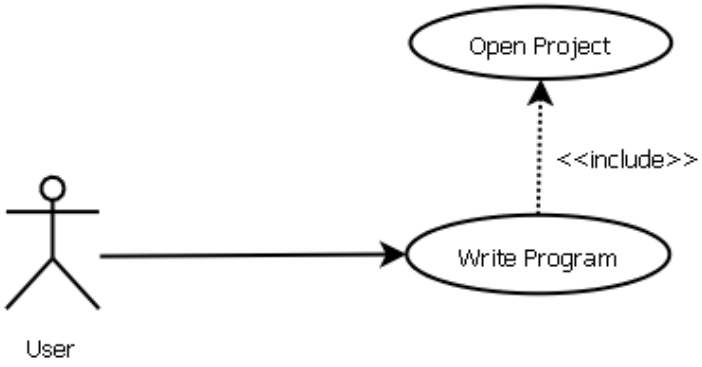
3.3.2 Open Project



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.

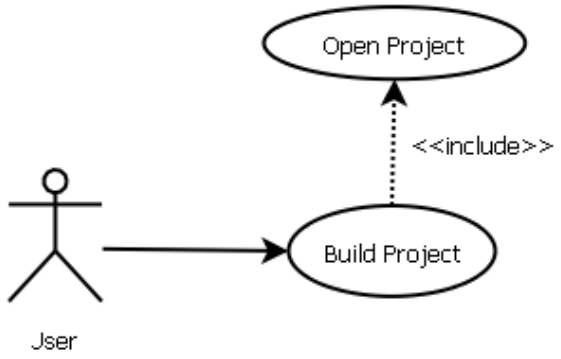
3.3.3 Write Program



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.

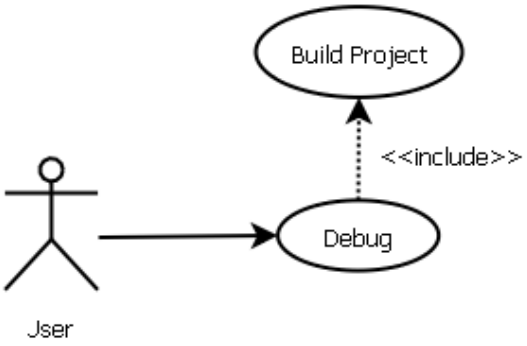
3.3.4 Build Project



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

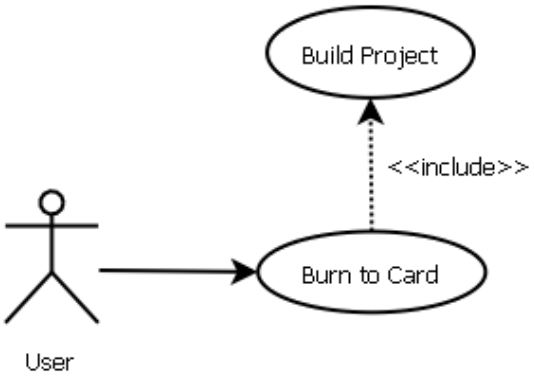


Use Case Diagram (Debug)

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.

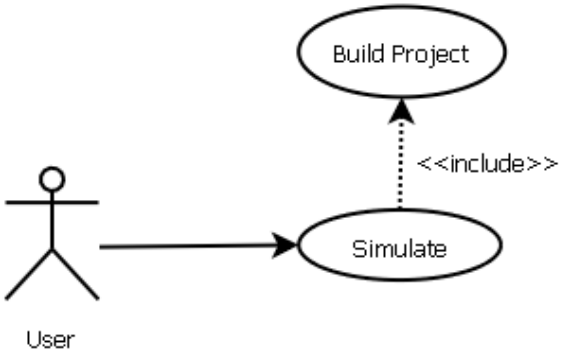
3.3.6 Burn to Card



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.

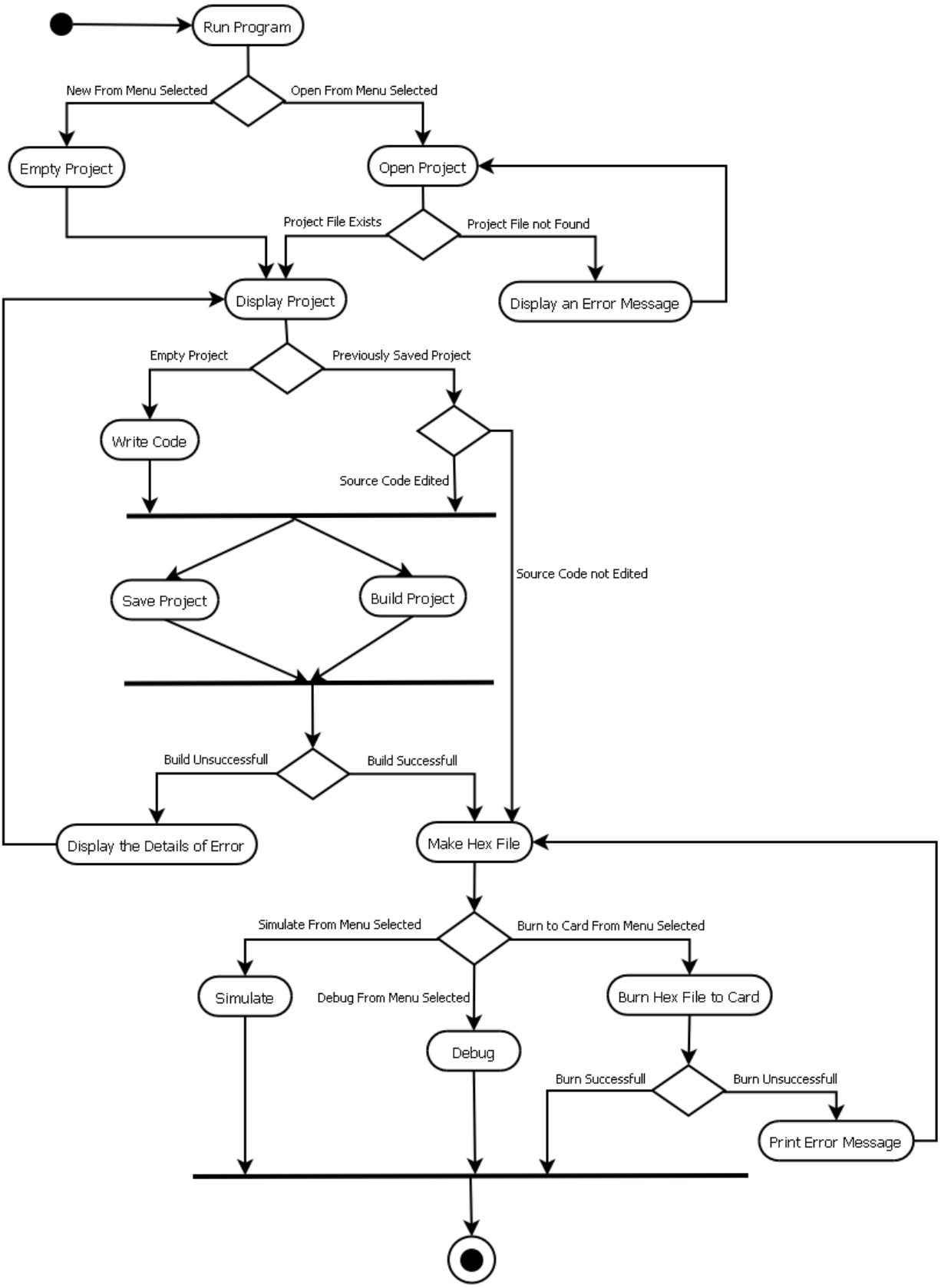
3.3.7 Simulate



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 Activity Diagram



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.

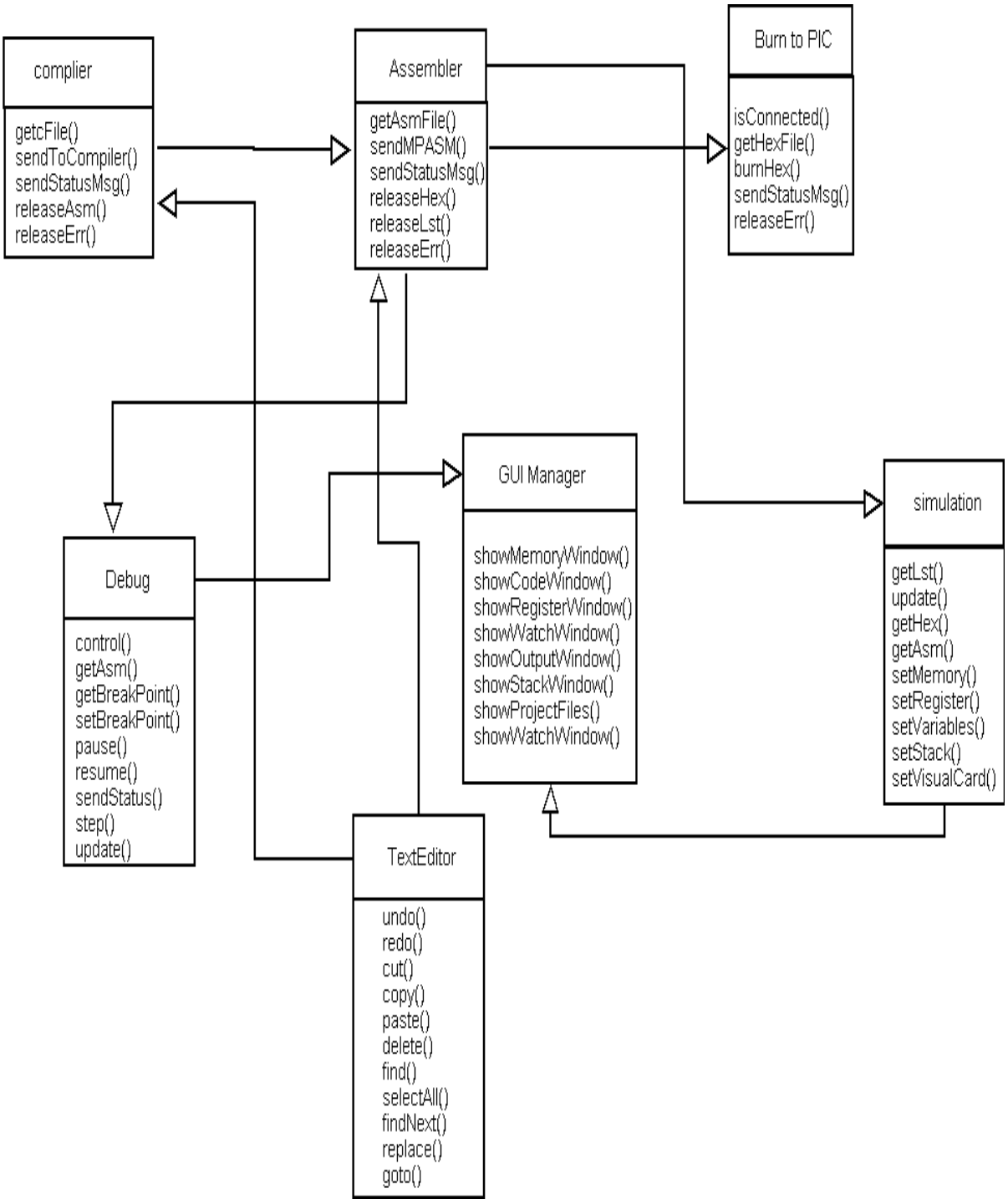
Burn Hex File to Card:

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.

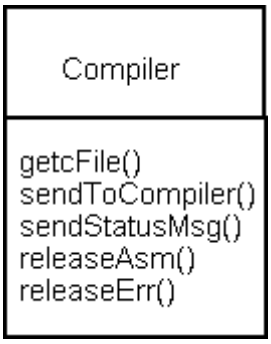
Print Error Message:

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 Class Diagram



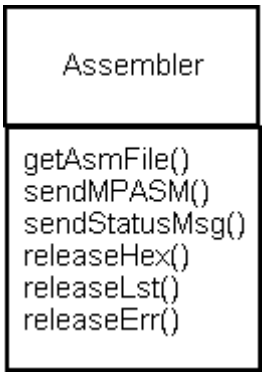
3.5.1 Compiler



Compiler accepts only C codes. This part gets a file, sends it to the compiler we get from outside, tries to compile, show the situation and give .asm file as an output.

- getcFile() : gets a file written in C language .If there is an opened file used in text editor , compiler will get that file .
- sendToCompiler() : this function will send the .c file to compiler used by our program.
- SendStatusMsg() : If there is any error or warning or if the progress is completed successfully this function will send message to GUI.
- releaseAsm(): If the completion progress is completed successfully without any error , this will get the .asm file produced by the compiler release it.
- releaseErr() :In case of any error or warning this functions will send the error message to the .err file ; if there exists no .err file ,this function will generate .err error file.

3.5.2 Assembler



Assembler get .asm file from the user or from the compiler described above. In this stage we will benefit from the MPASM assembler produced by Microchip Company as free. The asm files will be assembled using MPASM .

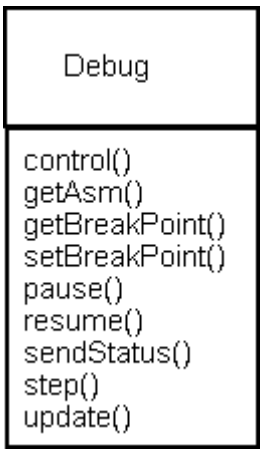
- getAsmFile() : This function will get .asm file from the compiler or directly from the user .
- sendMPASM() : This function will get an asm file and send it to MPASM assembler.
- sendStatusMsg() : In the assembly process, if any error or warning exists, this will record this information to the .err file and shown to the user in output window.
- releaseHex() : If the assembly progress is completed successfully without any errors, this will get the .hex file produced by the compiler and release it.
- releaseLst : This function releases the .lst file of the project after the assembly process is completed successfully without any errors.
- releaseErr() : In case of any error or warning this function will send the error message to the .err file ; if there exists no .err file ,this function will generate .err error file.

3.5.3 Burn to PIC

After the compilation (if the code is written in C language), assembly and linking processes are completed successfully a hex file that can be burned on a PIC microcontroller is created. This part gets this hex file and burns it to the PIC microcontroller that is on the Ceng 336 embedded card.

- isConnected() : This function tries to establish a connection to the Ceng 336 embedded card. The information about the status of the connection process (successful or not) is shown to the user via a message box.
- getHexFile() : If the connection is established successfully, this function gets the hex file produced.
- burnHex() : This function tries to burn the hex file to the PIC microcontroller that is on the Ceng 336 embedded card.
- sendStatusMsg() : If the burn process is failed, the resulting information is recorded to an err file by this function.
- releaseErr() : This function will release the .err error file generated by the sendStatusMsg function.

3.5.4 Debug



- control() : This function controls whether a hex file exists for debugging or not. If not, the user is informed via an error message box.
- getAsm() : The assembly file produced by the system or written by the user is got by this function.
- getBreakPoint() : The breakpoints specified by the user are got by this function.
- setBreakPoint() : The breakpoint info is recorded if the user specifies a new breakpoint.
- pause() : If the user wants to pause the execution of the program, this function gets the control.
- resume() : After the pausing of the executable program written, this function enables the resuming of the execution.

- sendStatus() : The information about the current status of the program is sent to GUIManager.
- step() : Executes one line of source code only.
- update() : The changes made on the registers, memory stack etc. are processed and recorded.

3.5.5 TextEditor

The text editor enables the user to write code, and edit code text.

- undo() : The last change made by the user is undone.
- redo() : The last undo made by the user is redone.
- cut() : The text selected by the user is cut and recorded.
- copy() : The text selected by the user is copied and recorded.
- paste() : The copied or cut text is pasted to the place where the cursor of the mouse is located in the text editor.
- delete() : The text selected by the user is deleted.
- find() : The text that is to be found by the user is searched in the source code.
- selectAll() : The whole text is selected.
- findNext() : The next occurance of the text that is to be founded by the user is shown to the user. If there does not exist any more occurance of this text, the user is informed via a message box that the search is finished.
- replace() : One text specified by the user is replaced in every occurance by another text specified by user again.

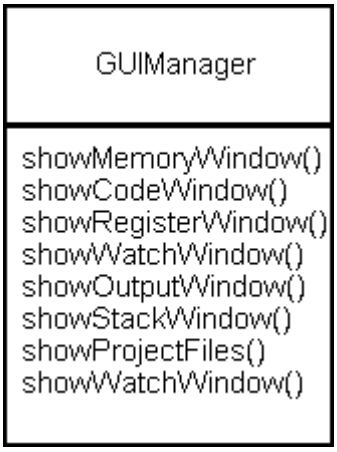
- goto() : goto selected position

3.5.6 Simulator

The simulator enables the user to see the execution of the program without burning it into the Ceng 336 embedded card. That is we supply a graphical user interface for the card in which the user will be able to push buttons, see the outputs of the different parts of the card etc.

- getLst() : Gets the .lst file produced by the system.
- update() : The changes made on the registers, memory stack etc. are processed and recorded.
- getHex() : Gets the hex file produced by the system.
- getAsm : Gets the asm file produced by the system.
- setMemory() : Records the changes made in the memory to the memory.
- setRegister() : Records the changes made in the registers to the registers.
- setVariables() : Records the changes made in the variables to the variables.
- setStack() : Records the changes made in the stack to the stack.
- setVisualCard() : Records the changes made in the visual card to the visual card.

3.5.7 GUIManager



GUIManager is responsible from the visualization of the stack , memory , variables , output window etc.

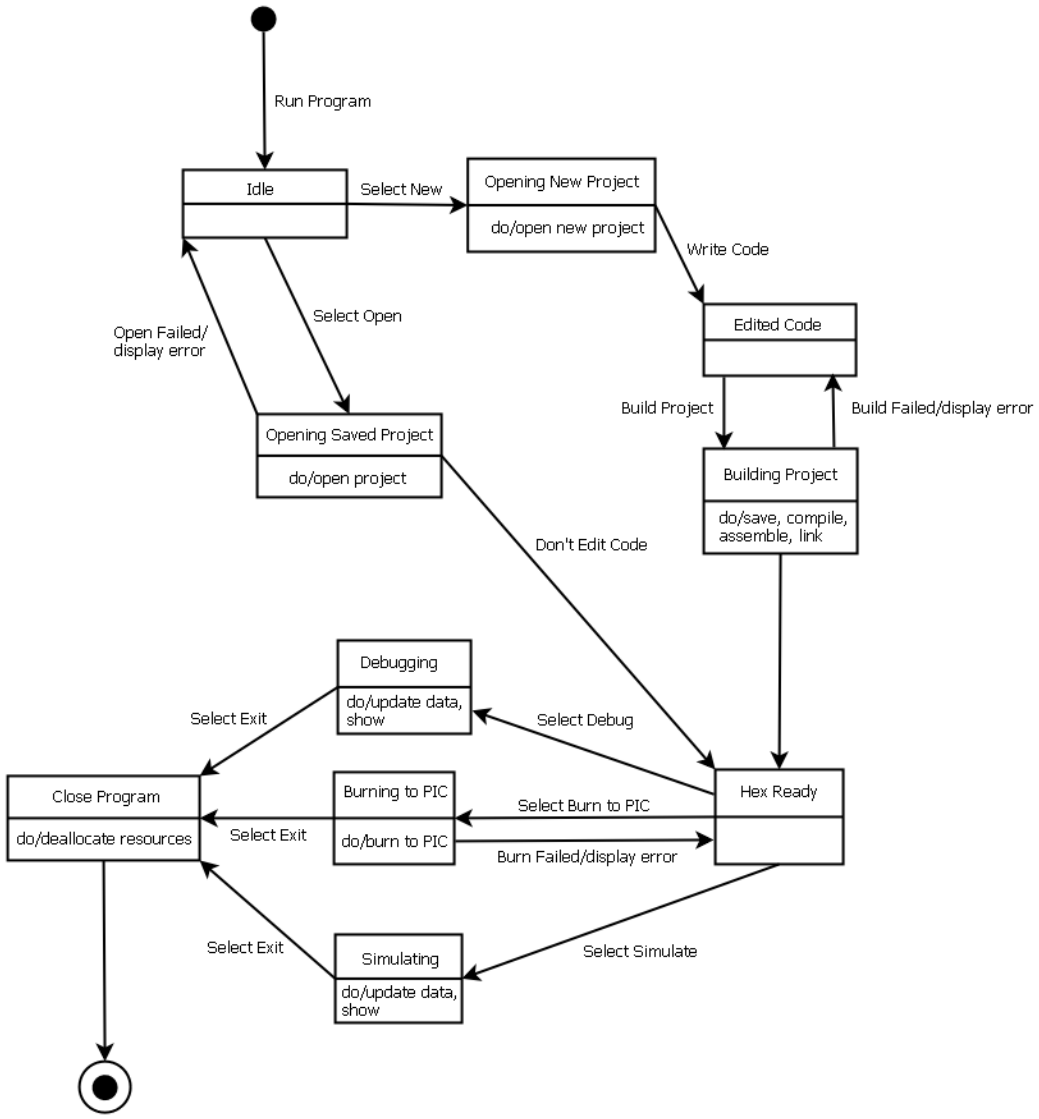
- showMemoryWindow() : activates the memory window
- showStackWindow() : activates the Stack window
- showWatchWindow(): activates the watch window used by debugger
- showOutputWindow() : activates the output window
- showRegisterWindow() : activates the Register window

By the way we are planning to design some data structures as below.

Stack : 10 * 8 sized Array . As default all 0.
Data Memory : 368 * 8 sized Array . As default all 0 .
Register: # register * 8 sized array As default all 0 .

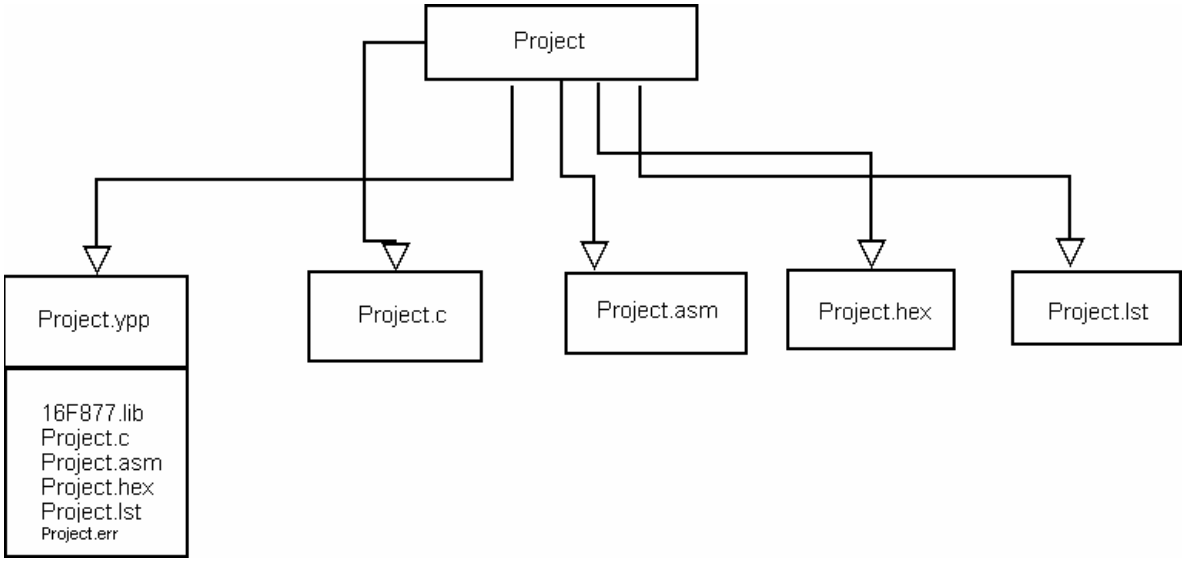
These structures will be used by simulator , GUIManager . Debugger etc. These structures get more clearer in our mind as we get closer to implementation stage.

3.6 State Transition Diagram



State Diagram

4. Project's File Structure



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.

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.

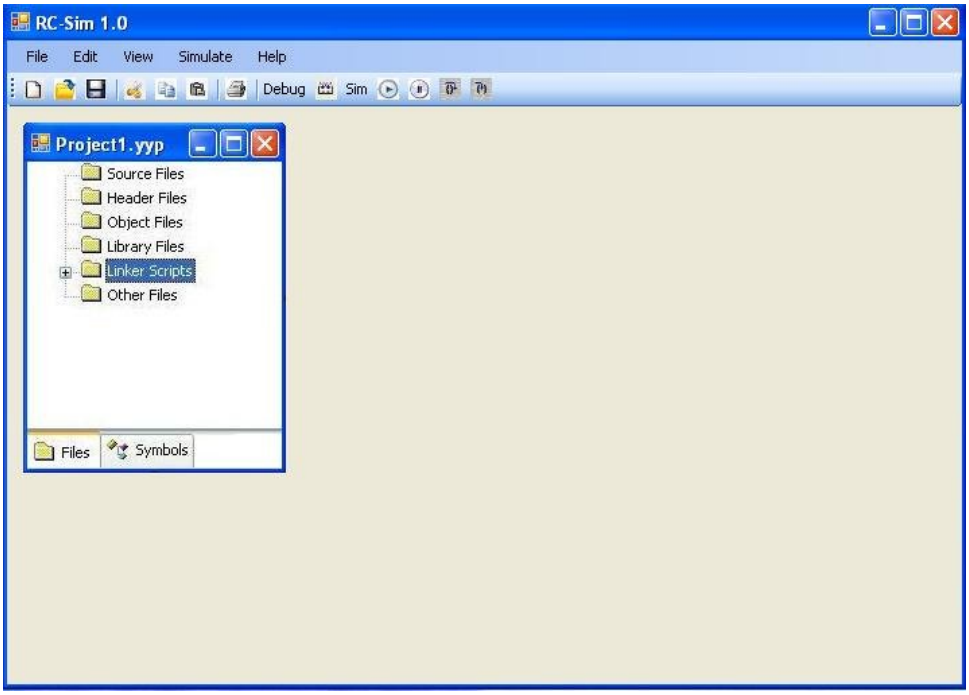
7. Other files

If the user wants to include some other files, we will categorize this files as a part of this format.

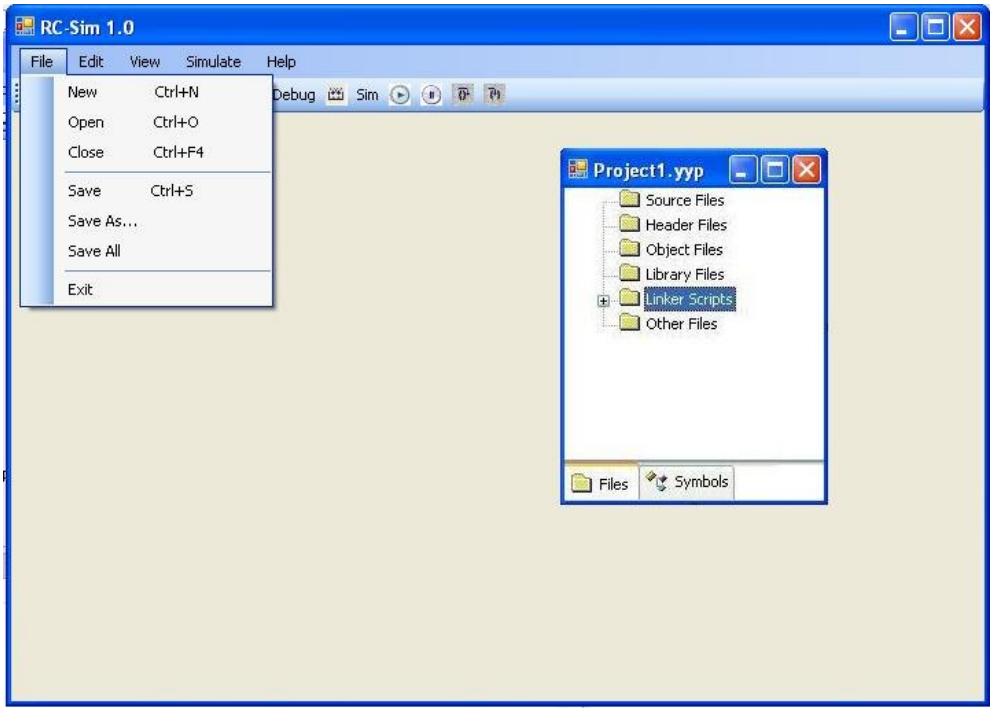
5. User Interface Design (Menus)

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.

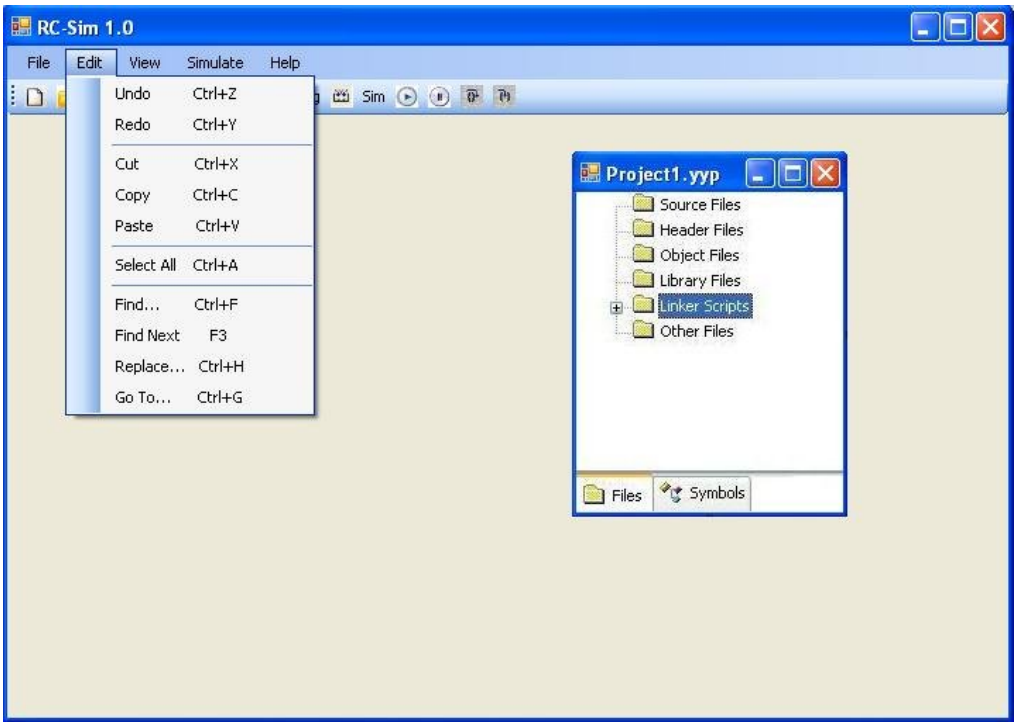
f. Save All

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.

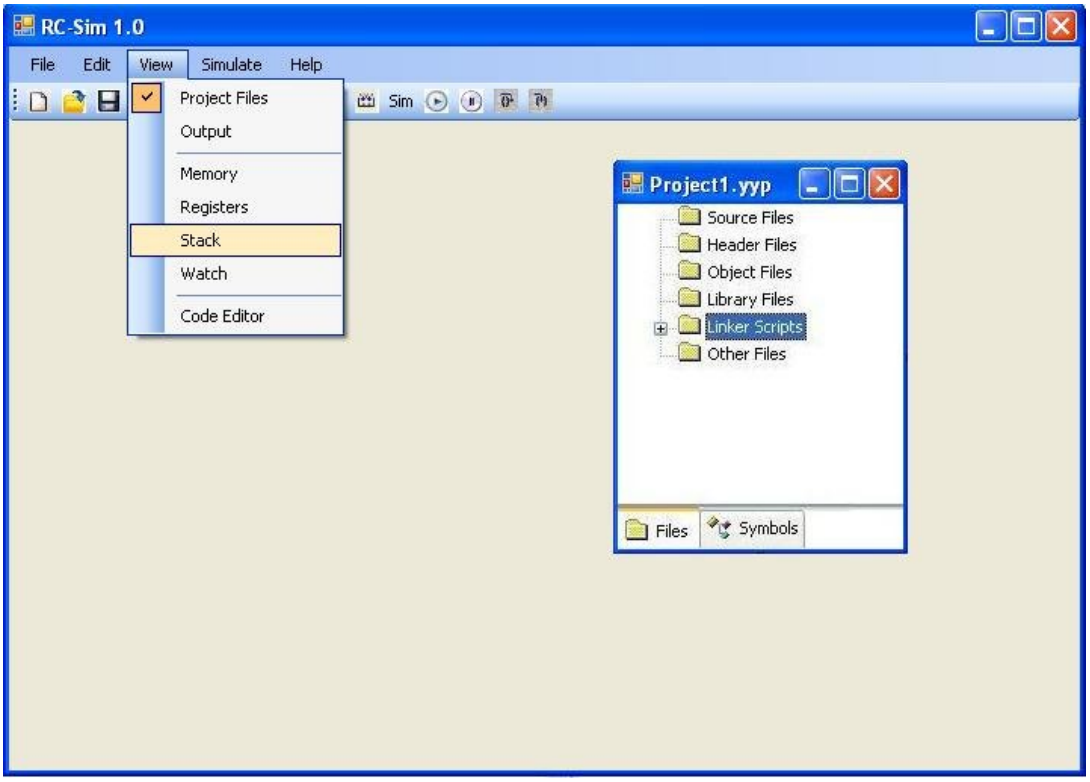
i.Replace

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

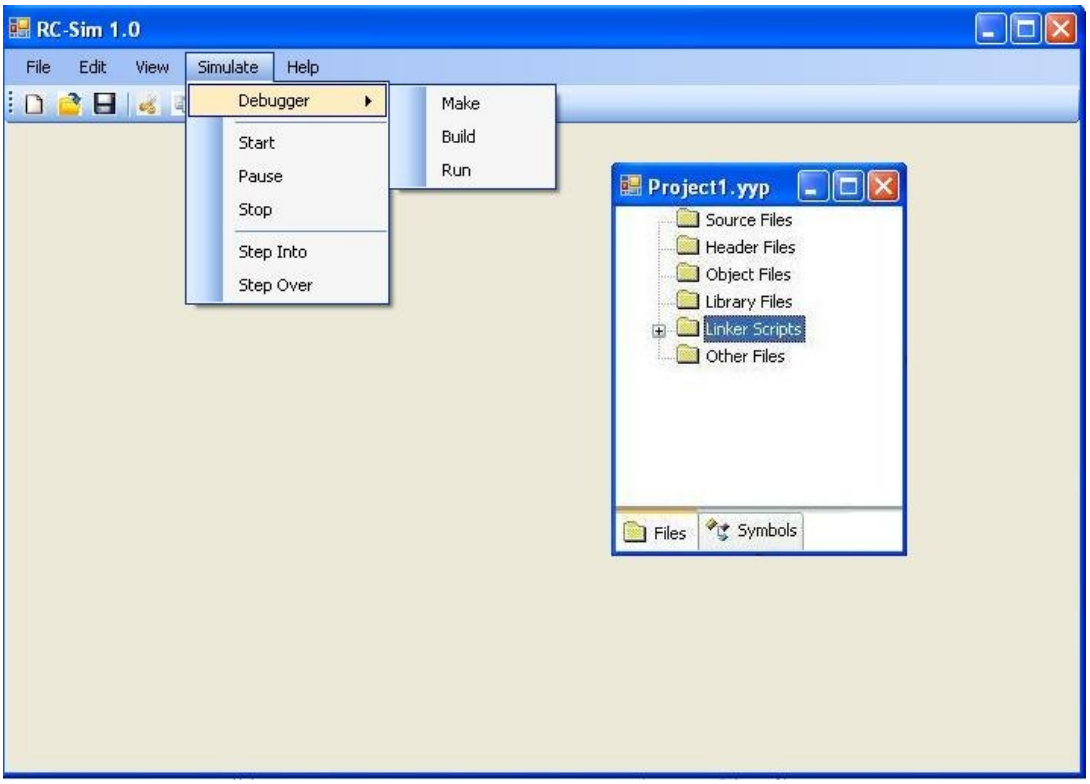
Opens the dialog box to watch the indicated breakpoints.

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.

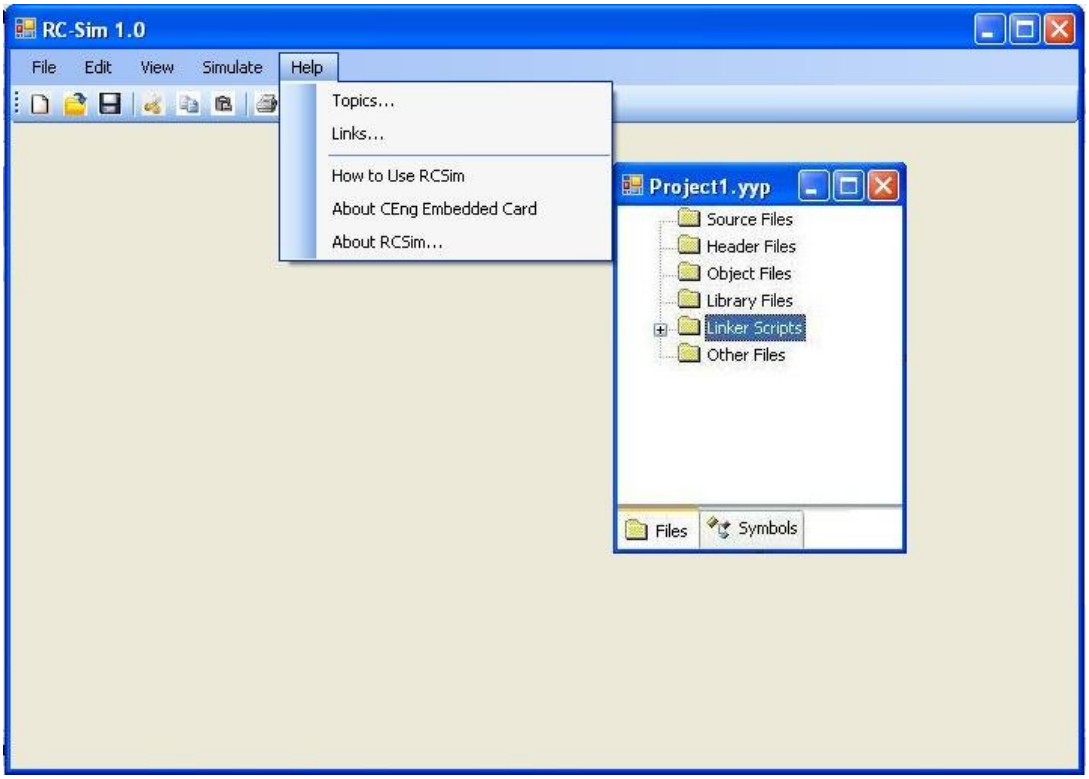
e. Step Into

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.

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.

6. Gantt Chart

This is the gannt chart of our schedule. We have updated it due to the initial design.