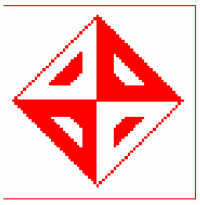
**MIDDLE EAST TECHNICAL UNIVERSITY**



DEPARTMENT OF COMPUTER ENGINEERING

CENG 491 – Computer Engineering Design

# REQUIREMENT ANALYSIS REPORT

Group Name: 

Group Members:


Hatice Beyza KIRBAŞ 1297951
Özgür Ufuk ÖZDEMİR 1347798
Oya TEZEL 1348010
Serdar TUĞCU 1298330


Project Title: Emulator and Development Environment for CENG Embedded System Card

Project Alias: PicSim

# Table of Contents

# 1 Project Description

## *1.1 BACKGROUND*

This document is the requirement analysis report of SoSoft, the DevEmb project group. We are going to state the scope, functional requirements, constraints and structural model of our project in details. This report will provide a basis throughout the solution process of the problem specifying the requirements. The requirements are determined after a detailed analysis of available solutions (similar programs used for the same purpose), the tools that can be used in the solution (language capabilities & available libraries) and users' needs and demands.

DevEmb project includes implementing a compiler and simulator for the Pic Demo2 board (PDB) designed by Alper Kılıç for the CEng336 course. This Pic board has a number of LEDs, an LCD screen, parallel port, serial port, USB port, five jumpers three of which (JP1, JP2, JP3) are used in Programming Mode and two of which (JP4,SPK) are used in Executing Mode, speaker.

In the Programming Mode, one of the JP1, JP2 and JP3 is selected. This selection determines which microcontroller to use, since PDB supports 18, 28, 40-pin microcontroller families. If one of these jumpers is set, the PDB is operates in Execution Mode.

In the Execution Mode, 4MHz or 20 MHz oscillator must be selected for 40-pin microcontroller using JP4. Enabling/disabling speaker is available by using SPK.

| | |
|---|---|
| JP1 JP2 JP3 | 40-pin MCU is selected |
| JP1 JP2 JP3 | 28-pin MCU is selected |
| JP1 JP2 JP3 | 18-pin MCU is selected |
| JP1 JP2 JP3 | No MCU is selected to program, Executing Mode |

## *1.2 Project Definition and Scope*

A microcontroller is a highly integrated chip that contains all the components comprising a controller. Typically, this includes a CPU, RAM, some form of ROM, I/O ports, and timers. Unlike a general-purpose computer, which also includes all of these components, a microcontroller is designed for a very specific task – to control a particular system. As a result, the parts can be simplified and reduced, which cuts down on production costs.

PIC (programmable interface controller)s are microcontrollers manufactured by microchip company. PICs are programmed via various compilers which convert the assembly code or a high level language code to hexadecimal code. Since it is expensive to debug the code to the PIC to see if it works correctly, some simulator programs are available in the market.

These programs mainly show internal microcontroller architecture. Using such PIC programs, it may be painful to control the program. These tools show the internal

circuit programmed and uploaded to the PIC, but does not show the external effects of this circuit. The CENG embedded systems card shows the effect of the program running in the PICs on the external circuit (i.e. the LEDs, LCD). The problem arises from the fact that available programs cannot show what happens to this circuit. So these tools cannot meet users' demands.

There are a number of Pic compilers available recently. But most of these programs can only compile the source file and load it to the Pic board. By using these programs, user can see if the source file is compiled successfully, but a lot of programmer complains that their programs are being compiled successfully but running in a way different than they want. In order to overcome this problem, user needs a simulator. A simulator which simulates the source file that is to be compiled before loading the compiled file to the Pic helps user see if they are going to get what they want. User can see if the program is working correctly and if he/she is satisfied, he/she loads it to the Pic. There are also some programs which have the ability to simulate, but our software will be designed specifically for the PDB and user will not have to handle a lot of extra features of other programs which are unnecessary for the PDB.

We are going to implement a software emulator, compiler and development environment for embedded systems card in the project. By our software, users will be able to compile, upload and debug their programs on the card. Testing the programs in the virtual card emulated by the software will also be possible. This emulator will show the external effects (the output of the circuit implemented in the Pics) of the programs the users compile and debug using it.

Since our software will simulate the embedded system card, it will have all the features that are present on the card and also the user will be able to do everything that he/she can do by the card virtually. In addition our PIC compiler will compile C source codes into hexadecimal code which can be uploaded to the PIC. It will also have a debugger that simulates the code step by step during run process.

# 2 Literature Survey

## *2.1 Applications*

When designing software, one of the most important parts is to determine the needs of the user. The software we are going to design is usually for experienced users. And such users usually have experience with similar programs. So, to examine the other products that can do the same job is a very efficient way to develop the software. Due to this fact, we have examined a number of Pic compilers and simulators. Here, we are going to give details of our observations.

The list of the programs we are going to write in this part of the report are:

- MPLAB
- PROTHEUS
- mikroC
- Pic Basic Compiler

### 2.1.1 MPLAB

MPLAB Integrated Development (IDE) is a free integrated toolset for the development of embedded applications. It runs as a 32-bit application on MS Windows. This program is relatively easy to use and since it is free, a lot of users prefer this program. Since MPLAB has a good interface, it is easy to move between tools. We are going to have a C compiler for Pic programming, so we have examined its C compiler. MPLAB C18 is an optimized C compiler for the PIC18 series microcontrollers. Also MPLAB has MPLAB C30 for PIC24 and dsPIC digital signal controllers.

MPLAB has a debugger which has capabilities like auto alignment of breakpoints, mouse-over variable inspection, drag and drop variables to watch windows, automatic single-step animate feature etc.

The most important advantage of this program is that, it has a very powerfull help support. Its manual is prepared very good and also its web site, www.microchip.com is a referrence for every PIC programmer.

Another advantage is that it is easy to browse the files using its windows. Also the tools are placed to toolbar so that you can reach the tools which are most frequently needed very easily.

MPLAB does not have a graphical simulation of the code. One of the properties of our project is to simulate the written code, so MPLAB is does not demand the needs of our users totally.

We have added a screenshot of MPLAB to the Appendix.

### 2.1.2 PROTHEUS

Protheus is a software by which user can design an electronic circuit using its tools, test the designed circuits and design the Printed Circuit Board (PCB) schematic. This is also a common program used by circuit designer. Unlike MPLAB, Protheus is a program mainly used for simulation. User designs a circuit and sees what he/she did on the computer screen without loading to the PIC.

Protheus has a very good-looking interface but we think that its components are very complex. In order to add a component to the circuit, user has to do a number of actions. Also user is expected to have a very good memory, because user has to know the name of the components to find.

The above paragraph listed some disadvantages but we have to say that its simulator is very successful. Once you have designed a correct circuit, you can see what it does on the screen.

Using view options, user can have a custom view of the program. By the help of this property, the program is very good-looking. As a computer engineer, we think that the view of the screen we are working on is important for motivation.

Protheus also has a good-working debugger. By using this debugger, programming becomes easier for the user. In addition to this, like other programs, Protheus has libraries. The components of the libraries are arranged according to their properties

such as serial numbers (e.g. 74xx), producer companies (e.g. Parallax), and usage (e.g. LEDs).

Appendix includes a screenshot of Protheus.

### 2.1.3 mikroC

MicroC is a PIC C compiler for dsPIC applications. It is not a very common program since it is commercial but we have examined it and have some ideas about the interface and some other tools.

This program has an advanced code assistant. By hitting CTRL + SPACE, user can get the list of all available variables, constants, and routines which user can insert at cursor position (Figure a)

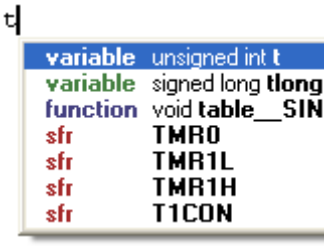



Figure a

There is a Parameter assistant for routines. When user hits CTRL + SHIFT + SPACE, the parameters of the function is displayed. (Figure b)


Figure b

Another property is debugger. This source-level debugger helps troubleshoot and repair the application. The debugger has options like ‘Breakpoints’ (Figure c) and ‘Watch Window’ (Figure d) options.

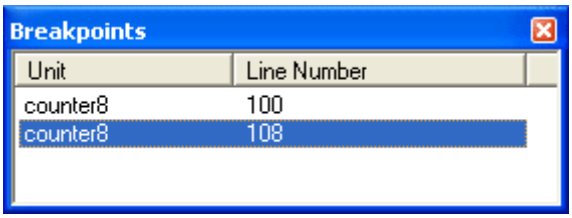



Figure c

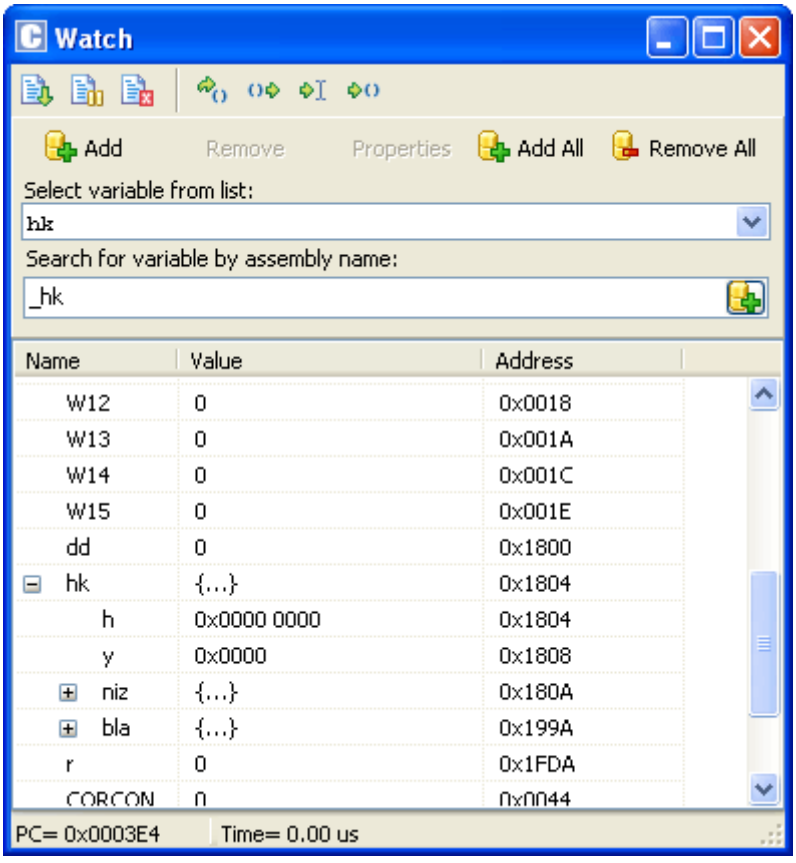



Figure d

Another advantage is Code Explorer. Code explorer finds any variable or function within the code easily. (Figure e)

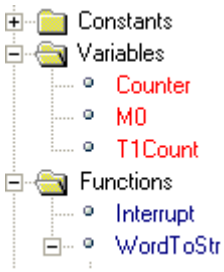



Figure e

### 2.1.4 PIC BASIC COMPILER

This program is a Basic compiler. We are going to have a C compiler, so we could not have enough benefit from this program. But this program has the properties which we are planning to include to our own program. These properties are user-friendly graphical development environment, integrated simulator (emulator), compiler and debugger.

Pic Simulator IDE has the following properties:

-Main simulation interface showing internal microcontroller architecture,

- FLASH program memory viewer, EEPROM data memory editor, hardware stack viewer,

- Microcontroller pinout interface for simulation of digital I/O and analog inputs,

- Variable simulation rate, simulation statistics,

- Breakpoints manager for code debugging with breakpoints support,

- PIC assembler, interactive assembler editor for beginners, PIC disassembler,

- Powerful PIC Basic compiler with smart Basic source editor,

- PIC Basic compiler features: three basic data types (1-bit, 1-byte, 2-byte), optional 4-byte (32-bit) data type with 32-bit arithmetic, arrays, all standard PIC Basic language elements, optional support for structured language (procedures and functions), high level language support for using internal EEPROM memory, using internal A/D converter module, using interrupts, serial communication using internal hardware UART, software UART implementation, I2C communication with external I2C devices, Serial Peripheral Interface (SPI) communication, interfacing character LCDs, interfacing graphical LCDs with 128x64 dot matrix, R/C servos, 1-Wire devices, DS18S20, using internal PWM modules ...
- Configuration bits editor,
- PC's serial port terminal for communication with real devices connected to serial port,
- LCD module simulation interface for character LCD modules,
- Graphical LCD module simulation interface for 128x64 graphical LCD modules,
- Hardware UART simulation interface,
- Software UART simulation interface for software implemented UART routines,
- Oscilloscope and signal generator simulation tools,
- 7-segment LED displays simulation interface,
- Support for external simulation modules,
- Extensive program options, colour themes, ...

## *2.2 TECHNOLOGIC RESEARCH*

In order to develop our program, we need to use programming languages and from the beginning of term to this date, we have discussed on the most suitable

programming language for us. As a result of our discussions, we have decided to use C and C# for development.

Our program will have an interface for editing the source code, an interface for simulation and a compiler. We have searched for some Pic compilers and we have seen that there are a lot of compilers written by C language. Also the interface is very important because one of our aims is to make a user-friendly program. So we shall use C#, which is a language of .Net technology.

For the simulation part, the situation is a bit complex. We have to understand what the code wants to do, and show it on the screen. So there are two parts to handle. We have decided to use both C and C# for the simulation process.

The reason for choosing these two languages is that four members of our group is familiar with C from the courses we have taken during our education and C# is an easy language for learning and it is very powerful at making interfaces.

We will definitely need some other languages in the internal steps of the implementation, but since the analysis period is mainly reserved for the determination of the requirements, we have worked on mainly the requirements. Implementation details may change during the design process.

# 3 Project Requirements

## *3.1 User Interface Requirements*

The most important part of the requirements analysis report is functional and non-functional properties issue. So, we had to decide on these requirements. To decide on these properties, literature survey helped us mostly, because during the literature survey we made, we have did some internet researches, asked questions to people who use similar programs and we have examined a number of similar programs. And as a result of this research period, we have seen the usability of tools and necessity of these tools. So, by the help of the literature survey, we collected information about the tools and we have decided what kind of tools our program will include.

While choosing the tools, we argued on how we can meet both beginner and professional user's needs. To do so, we consider the simplicity to understand and use the program. For understanding the program, the most important thing is that the first idea of the user about the program. To make the user have positive idea, we decided to have tools, which are placed conveniently. For using the program, the most important thing is to understand the user's needs and to put the tools accordingly. This part of the report is going to give the details of the points we have explained above.

### 3.1.1 Functional Requirements

#### 3.1.1.1 Hot Keys

Every programmer wants to use the interface tools in the most efficient way. When a user uses a program like the one we are going to develop, he/she wants to feel comfortable. In order to achieve this, hot keys have an importance for us. By using hot keys, user can use the tools shortly. We decided to use hot keys to make the user's life easy while using the program.

Almost every program has hot key sets. For example when a user writes a code, he/she spends very long times using the keyboard. So when he/she wants to save this, programmer usually use the hot keys for the Save process instead of clicking on some buttons of the interface. A person who has never written a source code and who is not very familiar with computer may see this option very unnecessary but we all think that hot keys make users feel comfortable. So we have decided to make a hot key set and we have explained them in this part of the report. Here are these hotkeys:

Ctrl+S: this is used to save the changes in the work.

Ctrl+C: this is used to copy a part of a code segment.

Ctrl+V: this is used to paste a part of a code segment.

Ctrl+X: this is used to cut a part of a code segment. In other words, to copy and to paste.

The above hot keys are the generally used hot keys. Every computer user knows these ones. So we decided not to change these. The hot keys which are special for our program are below:

Ctrl+O: this will be used to open a new file to work on.

Ctrl+D: this will be used to stop the program. For example, if the code segment enters an infinite loop, by using this hot key the user can stop the program.

F1: this will be used to simulate the program.

F2: this will be used to compile the program.

F3: this will be used to debug the code to the Pic board.

In addition to all these hot keys, according to the needs our program has, we can add some other hot keys. But this is something we are going to decide during the design period of our project.

#### 3.1.1.2 Menu Bar

Menu bar is going to be placed on the top of the screen as a horizontal bar. This bar is going have File, Edit, Project, View and Help options. Here are the details and sub options of these menus.

**File menu:** File menu has the following components:

- **Open File:** The user can open a file by selecting "**Open file**" component. When he/she clicks this component, user will be able to choose the file to be edited from a file browser and an editor will be opened with the work in it.
- **New File:** The user can open a new file by selecting "**New file**" component. When the user clicks this component, an empty editor will be opened and ready to write.
- **Save File:** The user can save the recently written file by selecting **"Save File"**. When this component is selected, the opened file will be saved to the same place. If a new file is wanted to be saved, this component will work like a "Save File As" component.
- **Save File As:** By selecting this component, user can save the file to a custom place with a custom name using the file browser window.
- **Quit:** The user can exit the program by selecting the "**Quit**" component. When the user selects this component, the program asks to save the work he/she works on, and all windows are closed.

**Edit menu:** Edit menu has the following components:

- **Undo:** The user can undo his work by selecting the "**Undo**" component. When the user clicks this component, last change, he/she has done, is discarded.

- **Redo:** The user can redo his work by selecting the "**Redo**" component. If he/she undoes some code segment, when the user selects this component, last undo operation is discarded.
- **Copy:** The user can copy any text by selecting "**Copy**" component. If some text is selected, when the user selects this component, that text is copied to be pasted.
- **Paste:** The user can paste any text by selecting "**Paste**" component. If some text is copied, when this component is selected, that copied part is pasted to the place the user wants.
- **Cut:** The user can cut any text by selecting "**Cut**" component. If some text is selected, when the user selects this component, that text is copied and then is pasted where the user wants.

**Project menu:** Project menu has the following components:

- **Compile:** The user can compile his/her code by selecting "**Compile**" component. To debug the code into the Pic board, the user compiles the code by using this component.
- **Simulate:** The user can simulate his/her code by selecting "**Simulate**" component. To see the simulation of his/her code on the screen, the user selects this component.
- **Debug:** The user can debug the code to the Pic board by selecting "**Debug**" component. After the compilation of the code, compiled code can be debugged to the Pic board by using this component.

**View menu:** View menu has the following components:

- **Project:** By selecting "**Project**" component, the user can see the project he/she works on.
- **Toolbar:** By selecting "**Toolbar**" component, the user can place the needed toolbars for him according to the project requirements.
- **Registers:** The user can observe the changes in the registers(general purpose) by selecting "**Registers**" component.
- **Local Variables:** The user can observe the changes in the local variables(special use registers) by selecting "**Local variables**" component.
- **Program Memory:** The user can observe the program memory by selecting "**Program memory**" component.
- **Layout:** The user can change the background colour and text colour by selecting the "**Layout**" component.

**Help menu:** Help menu is similar like hot keys. Inexperienced computer users see this tool unnecessary but a professional user uses help menus very often. The aim of help menu is very clear: to help the user. The user uses help menu when he/she wants to know something about the program or when he/she has some problem about the work. As a part of the project, we are going to prepare a user manual. The user will be able to access the user manual by using "**User manual**" component of this menu. We hope to make this manual so good that a user will find whatever he/she looks for.

Also the user can search any word by using "**Search**" component of help menu. For this property, we are planning to make an interface by which user can search for a word or phrase from the manual. At the end of the menu list, we are going to make user visit our web site by **"About Us"** component.

As we have stated before, this report is mainly for deciding the requirements. During the design of the project, if we think that any other menu is needed, we can add needed part conveniently.

#### 3.1.1.3 Tool Bar

Almost every program has a tool bar. On the tool bar, a shortcut to most frequently used components are added. So we have decided to make a toolbar for our software. What we are going to add on this toolbar is decided according to general use.

In the tool bar, a new file can be opened and the user can save the file he/she works on. By using toolbar components, the user can open a new project and can save the project he/she works on. Also searching for a phrase on the editor will be available using the toolbar.

Although simulation, compilation and debugging can be done by using program menu, these can also be done by using tool bar. In addition to this, the user can stop the project and he/she can run the project step by step by the help of the tool bar. This will be useful for debugging process.

### 3.1.2 Non-Functional Requirements

At the 1.1 part of the report, we have listed the functional requirements. Functional requirements are the ones specific for the software. But there are also non-functional requirements of a program. These requirements are not only about how to make the program functionalities better while using. They are about how to make user feel comfortable, not to make them wait too long for an action and so on.

Here is the list of these requirements.

#### 3.1.2.1 User Friendliness

The interface of the program is one of the most important part of the project. For the user, working on the project without being lost in an interface is important and the user prefers a program with most convenient interface for his/her work. As we have stated at literature survey, there are some programs which have very powerful functionalities but difficult to use because of their complex interfaces. Accordingly, the importance of the design of the interface is one of the most important issues for our team.

The components of the interface will be well-defined and the user will be able to access the item he/she wants easily. The menus and the tool bar will also be clear and an access to them will be convenient.

### 3.1.2.2 Stability

The stability of the program is one of the most important issues that the user is interested in. Because our program will compile the C code , when the user wants to compile his/her code, it will only be compiled according to the Ansi C standards. Any other programming language cannot be compiled.

Furthermore, our program will simulate the source code and will have an ability to debug the compiled code to the Pic board. So, our program will have the same results when the program is executed with both ways.

### 3.1.2.3 Performance

Graphical programs can have time problems. In addition to this, the project we are working on has to do a lot of machine-based jobs and these jobs may take long times for a computer.

Not to make the user wait too long for simulation, we are going to find algorithms which will make the simulation faster and more efficient. Moreover, by using some algorithms and data structures, we are going to develop a compiler which can do its job in smallest time interval.

Since .Net is a product of Microsoft, we will also make the interfaces which are developed by .Net technology works efficiently on windows platform.

#### 3.1.2.4 Portability

Portability is important for a product to have a wide demand. If we were preparing software for a larger purpose, we had to consider different platform and operating system choices, but because we will make the program specific for ceng336 students and almost every student has Windows XP operating system, we will develop our program for Windows XP.

### 3.1.3 Software Requirements

#### 3.1.3.1 Software Requirements for SoSoft

As we have stated before, we are going to develop the interface of our program using C#. Both C and C# will be used for simulator, and we will use C for the compiler. So our software demands are based on these two.

First of all, .Net is a technology which can work on only Windows XP. So our computers should have Windows XP operating system installed on them. Since C# is a part of Visual Studio, we will need Visual Studio. Visual Studio has compiler and debugger for C# and from our previous experiences, we know that especially the debugger will help us during the implementation.

For the C code part, the only thing that we need is a C compiler. In fact we can compile our C files by Visual Studio, we prefer to use DevC++ which four of us are

already familiar to. Just like Linux environment DevC++ uses gcc and gcc is one of the most successful compilers.

As a result, here is a list of our software requirements:

- Windows XP
- Visual Studio.Net
- C#
- DevC++

#### 3.1.3.2 Software Requirements for the End User

Since one of our major aims is to make our program easy to use, we are going to develop it in a way that user will only concentrate on his/her own code. So, we are going to make an install package and this package is going to have all of the software requirements for the user. The only thing the user is going to do is to install the program to his/her computer. In addition to this, because our program works only on Windows XP operating system, the user also has to have Windows XP operating system on his computer.

### 3.1.4 Hardware Requirements

Users will use this program to see what their codes do on the computer screen. So, our attention is on the software side of the process. The only thing our program is going to do as a hardware process is to load the final compiled file to PDB. As a result of this, user is not going to have any hardware device other than a computer and PDB.

Of course, in order to connect PDB to computer, user is going to use cables for parallel port, serial port and USB port.

For our side, the procedure is the same. We are going to develop our program by totally using programming languages. There is no need for hardware for the implementation. At the end of the implementation, in order to test our scenarios, we will use the PDB just like the user will.

As a result, both we and the user will only need a computer, PDB and the required cables.

# 4 Data Models

## *4.1 Data Dictionary*

| Name | Source Code |
|---|---|
| From | User |
| To | Graphical Interface |
| Format | Text |
| Description | The user inputs a text as source code, using keyboard |

| Name | Source File |
|---|---|
| From | File Directory |
| To | Input File Manager |
| Format | Text File(.c file) |
| Description | The user inputs a text file which includes previously written code, using mouse, keyboard or hotkey |

| Name | Interface Commands |
|---|---|
| From | User |
| To | Graphical Interface |
| Format | action/mouse |
| Description | The user commands the Graphical interface using the mouse |

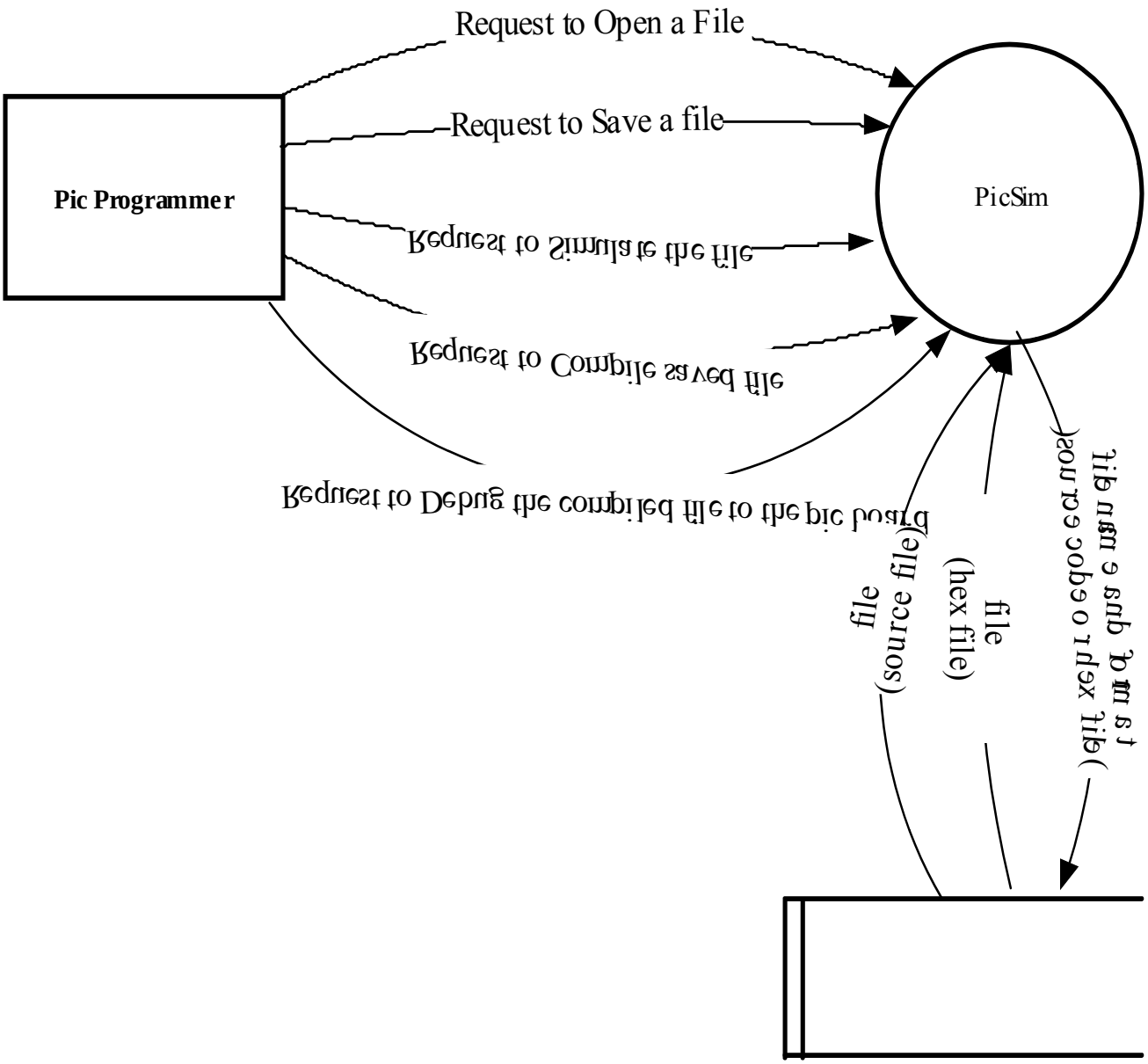
| Name | Simulate |
|---|---|
| From | Graphical Interface |
| To | Simulator |
| Format | Simulation file |
| Description | The simulation of the written source code will simulated by the program |

| Name | Compile |
|---|---|
| From | I/O manager |
| To | Compiler |
| Format | .c file |
| Description | The compiler will compile the written source code and make a HEX file |

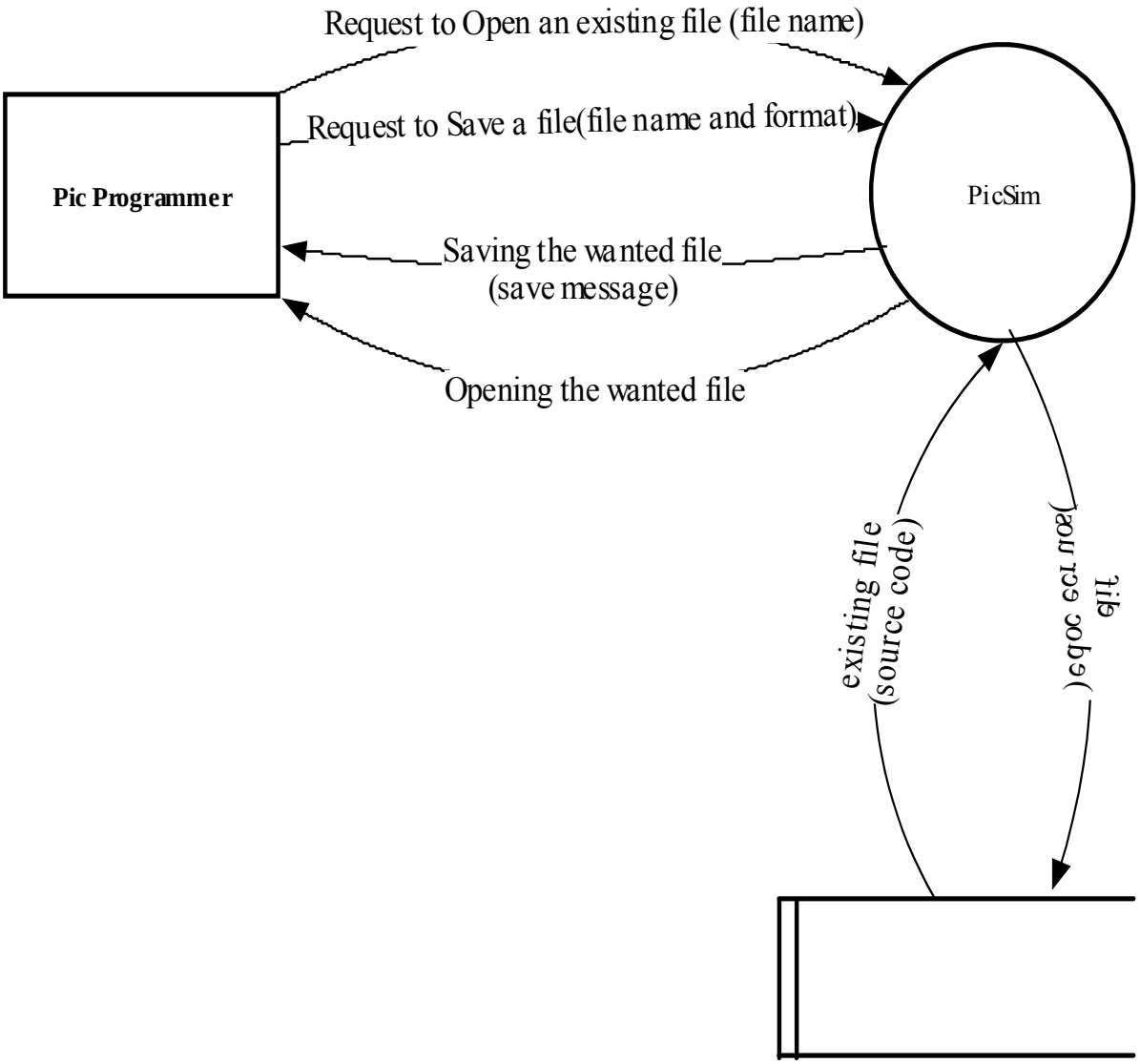
| Name | Debug |
|---|---|
| From | File manager |
| To | PDB(Pic board) |
| Format | HEX file |
| Description | The output of compilation will be loaded to the PIC board |

## *4.2 Data Flow Diagrams*

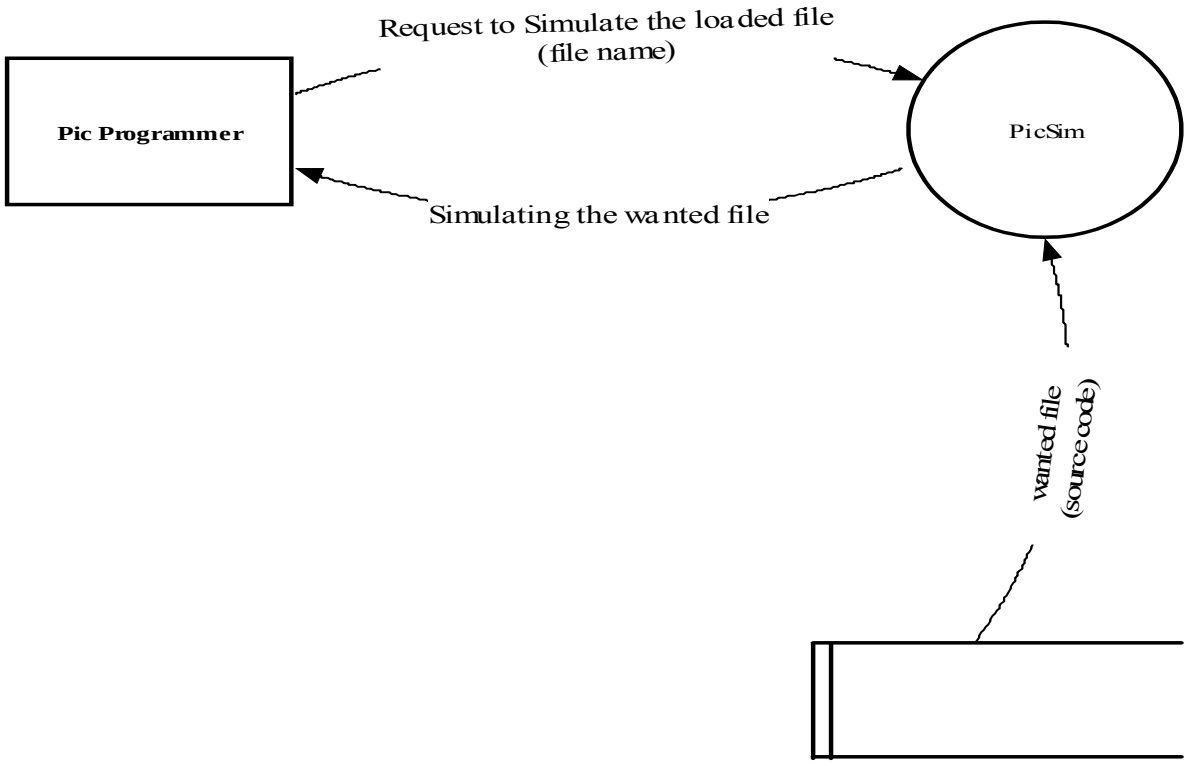
### 4.2.1 Level 0 DFD



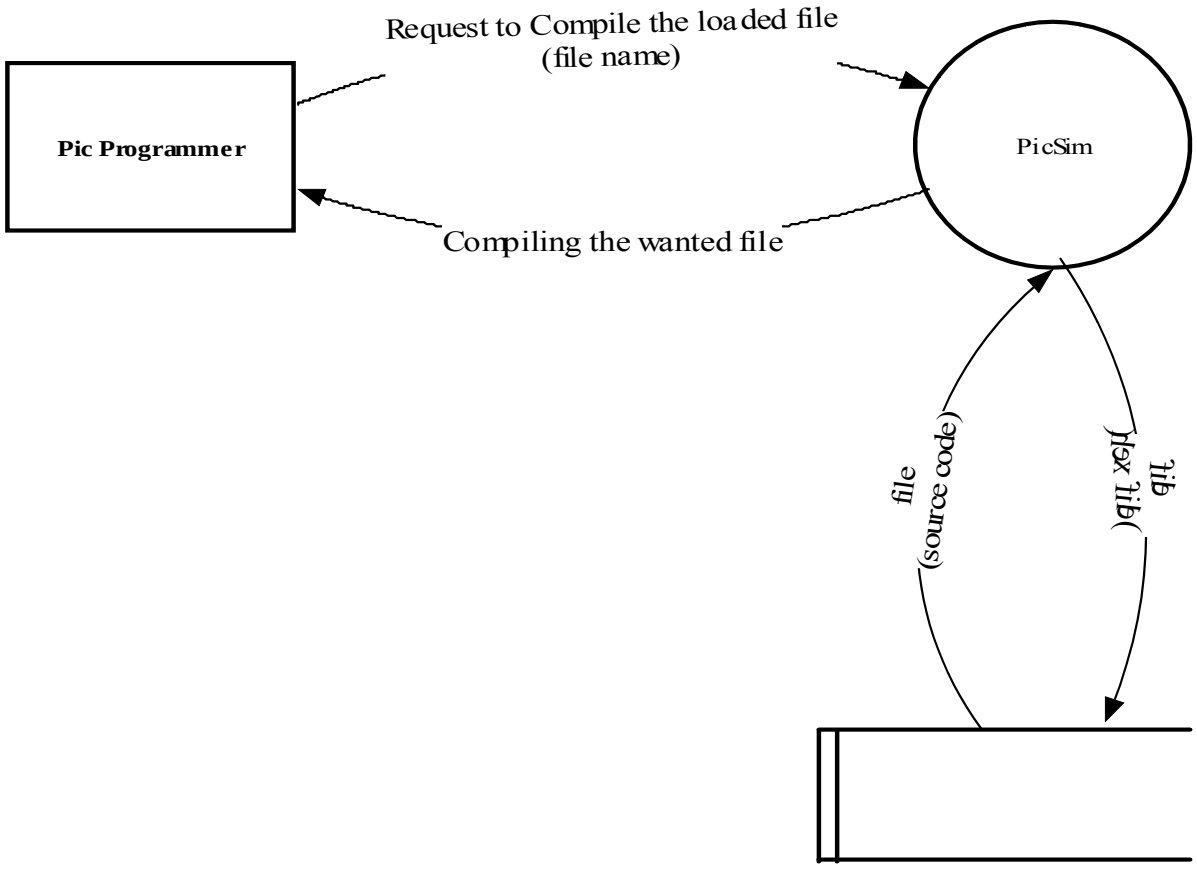
### 4.2.2 Level 1 DFD



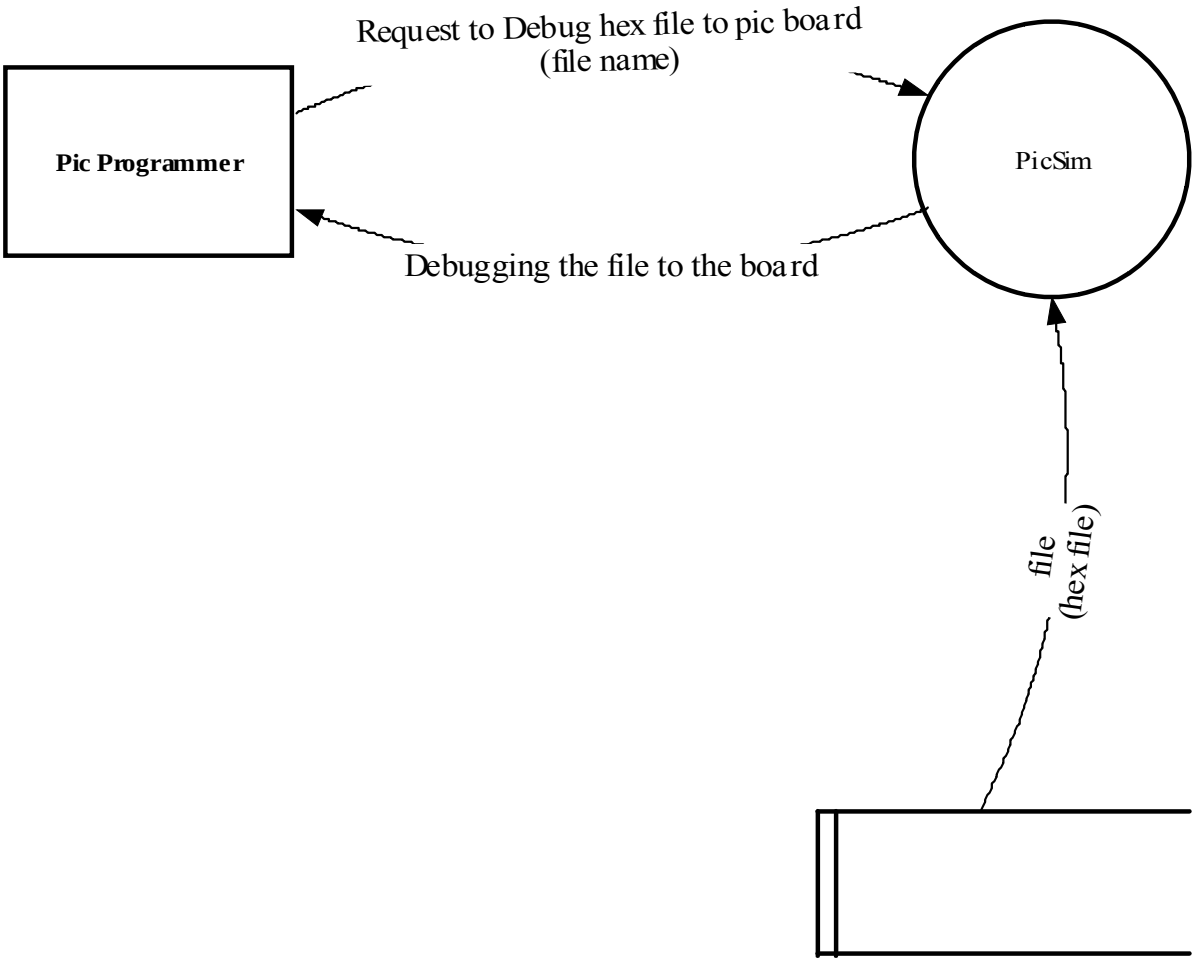

Level 1 DFD for Open/Save file request




Level 1 DFD for simulation

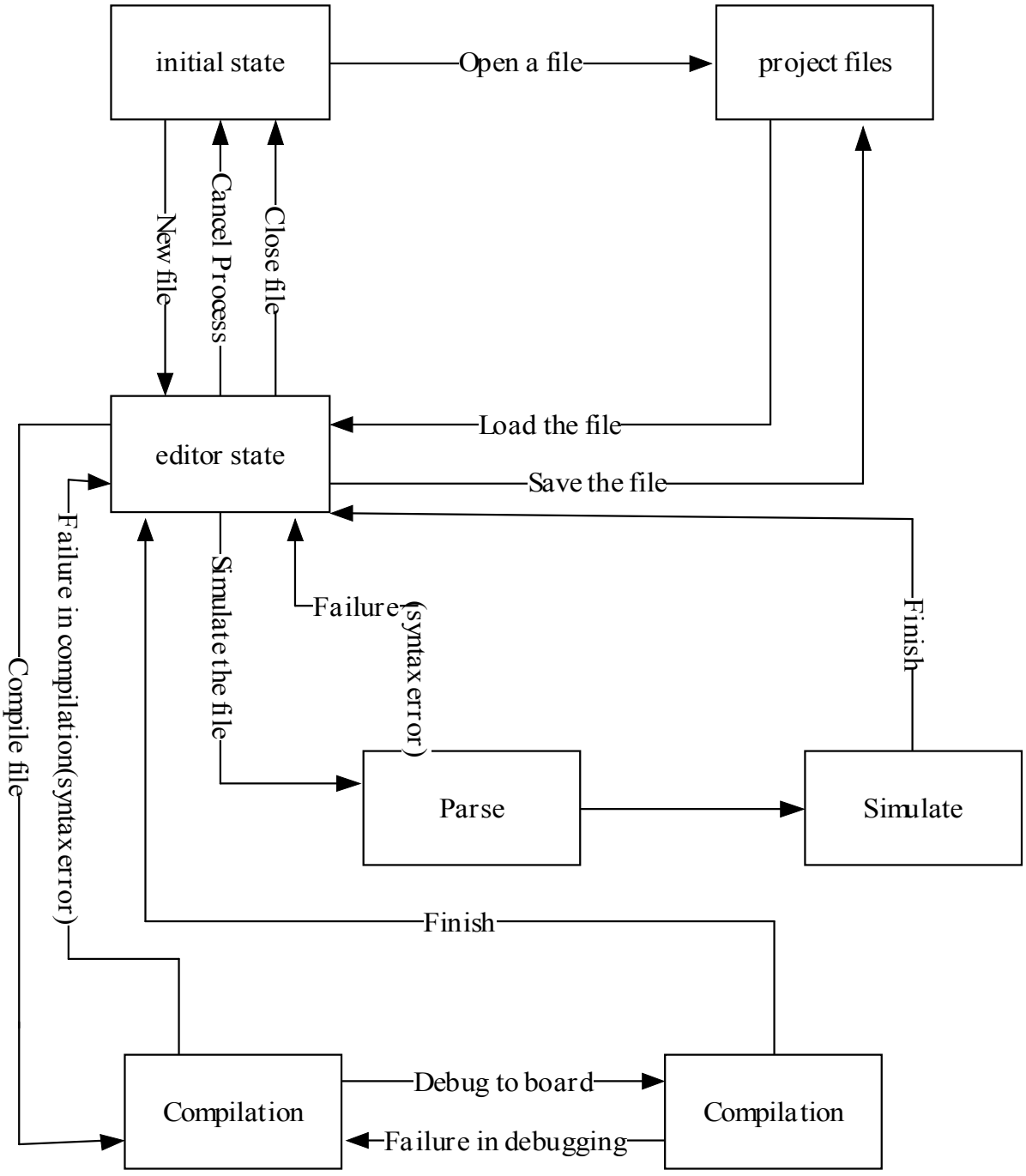



Level 1 DFD for compilation




Level 1 DFD for Debugging

## 4.3 States of the Program




State Transition Diagram

# 5 Usage Model

## *5.1 Usage Scenario*

In our project, user will write a Pic C source code, simulate the code, compile it and debug the code. So usage scenario will have editing, simulation, compilation and debug parts.

In order to run our program, user firstly needs a source file. User may open a new file and write his/her own source code on this file. To do this, user has to open a new file using File tab of the menu or the shortcut on the tool bar. While writing the code, since source code is a text file, user will probably need to cut copy or paste some phrases. After writing the source file, or at any step of editing the file, user may want to save the file. This action has two alternatives: ‘Save’ and ‘Save As’. If the user wants to use an already saved file, he/she may load this source file to the program by using the ‘Open File’ option.

After writing the source code or loading a previously written one, simulation part is coming. For the simulation, first the source file should be saved. When the simulation component is selected, a new screen will be opened for the simulation. If the source file is a valid source file, the simulation may be done for either the whole job expected by the code at once or step by step. Since user may want to see the simulation and the source file at the same time, simulation screen is a new window, which will be able to move on the computer screen. When the simulation process is completed, user will close the simulation window manually.

If the user is satisfied with the simulation, he/she will need to compile it in order to make the source file available for the Pic board. After closing the simulation window, user will use compile component of the program either from the menu bar or the tool bar. Compilation is not a graphical process, so during compilation, user will only see a message informing about the compilation. If the compilation is successful, a message box will appear and this means that program is ready to debug the file on the PDB.

At the end of all these process, we have a HEX file and by using the debug component of the program, this HEX file will be loaded to the PDB. After this step, our program has nothing to do. The rest is up to the PDB.

## *5.2 Use Cases*

We have described how a user may use the file at the usage scenario. In this part, we are going to list the use cases:

1. open a new/existing source code
2. edit the code (cut/copy/paste)
3. save the code
4. select how to simulate the code (totally or step by step)
5. simulate the code
6. correct the errors (if any) of the source code
7. if there were any errors, retry to simulate the code after correction of the code
8. close the simulation window
9. compile the file
10. close the message box which tells that the compilation process is ended

11. debug the file (load it to the PDB)

# 6 Project Management

## *6.1 Process Model*

During the implementation process of our project, we are going to use Rapid Application Development (RAD) Model. The RAD model is an incremental software process model that emphasizes a short development cycle. The RAD model is a ”high-speed” adaptation of the waterfall model in which rapid development is achieved by using a component-based construction approach. If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within a very short time period. We have a four months period for implementation process of our project, but since we are not experienced engineers, the time is not very long for RAD model.

## *6.2 Team Organisation*

Organization of a group plays important role for the success of the project. We are aware of this, so firstly we have chosen the project leader, who is Serdar Tuğcu. Actually, the leader of our team has the responsibility to arrange the deadlines of project parts for our team schedule and arrange meetings of the team members.

Because every member of our team has almost equal technical experience and knowledge about team work and developing the product, we are applying democratic decentralized team organization. According to this, we are sharing the work load and continuously we are gathering and sharing the works done individually.

## *6.3 Schedule*

### 6.3.1 Milestones

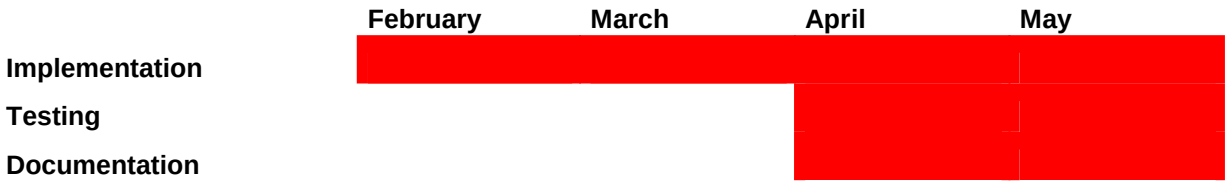
Here is the list of milestones of the project:

- Proposal October 8th 2006
- Requirement Analysis Report November 7th 2006
- Initial Design Report November 28th 2006
- Team Presentations December 5th 2006 – December 22nd 2006
- Final Design Report January 15th 2006
- Prototype Demo January 23rd 2006

### 6.3.2 Gantt Chart

| Current Week | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Weeks / Tasks | 10/09 to 10/16 | 10/16 to 10/23 | 10/23 to 10/30 | 10/30 to 11/06 | 11/06 to 11/13 | 11/13 to 11/20 | 11/20 to 11/27 | 11/27 to 12/04 | 12/04 to 12/11 | 12/11 to 12/18 | 12/18 to 12/25 | 12/25 to 01/01 | 01/01 to 01/08 | 01/08 to 01/15 | 01/15 to 01/22 |
| Problem Definition | | | | | | | | | | | | | | | |
| Proposal | | | | | | | | | | | | | | | |
| Literature and User Survey | | | | | | | | | | | | | | | |
| Interview | | | | | | | | | | | | | | | |
| Functional Requirement Decision | | | | | | | | | | | | | | | |
| Software Requirement Decision | | | | | | | | | | | | | | | |
| Requirement Analysis Report | | | | | | | | | | | | | | | |
| Practising On Tools | | | | | | | | | | | | | | | |
| Discussion On Initial Design | | | | | | | | | | | | | | | |
| Design Decision | | | | | | | | | | | | | | | |
| Interface Decision | | | | | | | | | | | | | | | |
| Prototype Decision | | | | | | | | | | | | | | | |
| Prototype Implementation | | | | | | | | | | | | | | | |
| Initial Design Report | | | | | | | | | | | | | | | |
| Tasks / Weeks | 10/09 to 10/16 | 10/16 to 10/23 | 10/23 to 10/30 | 10/30 to 11/06 | 11/06 to 11/13 | 11/13 to 11/20 | 11/20 to 11/27 | 11/27 to 12/04 | 12/04 to 12/11 | 12/11 to 12/18 | 12/18 to 12/25 | 12/25 to 01/01 | 01/01 to 01/08 | 01/08 to 01/15 | 01/15 to 01/22 |

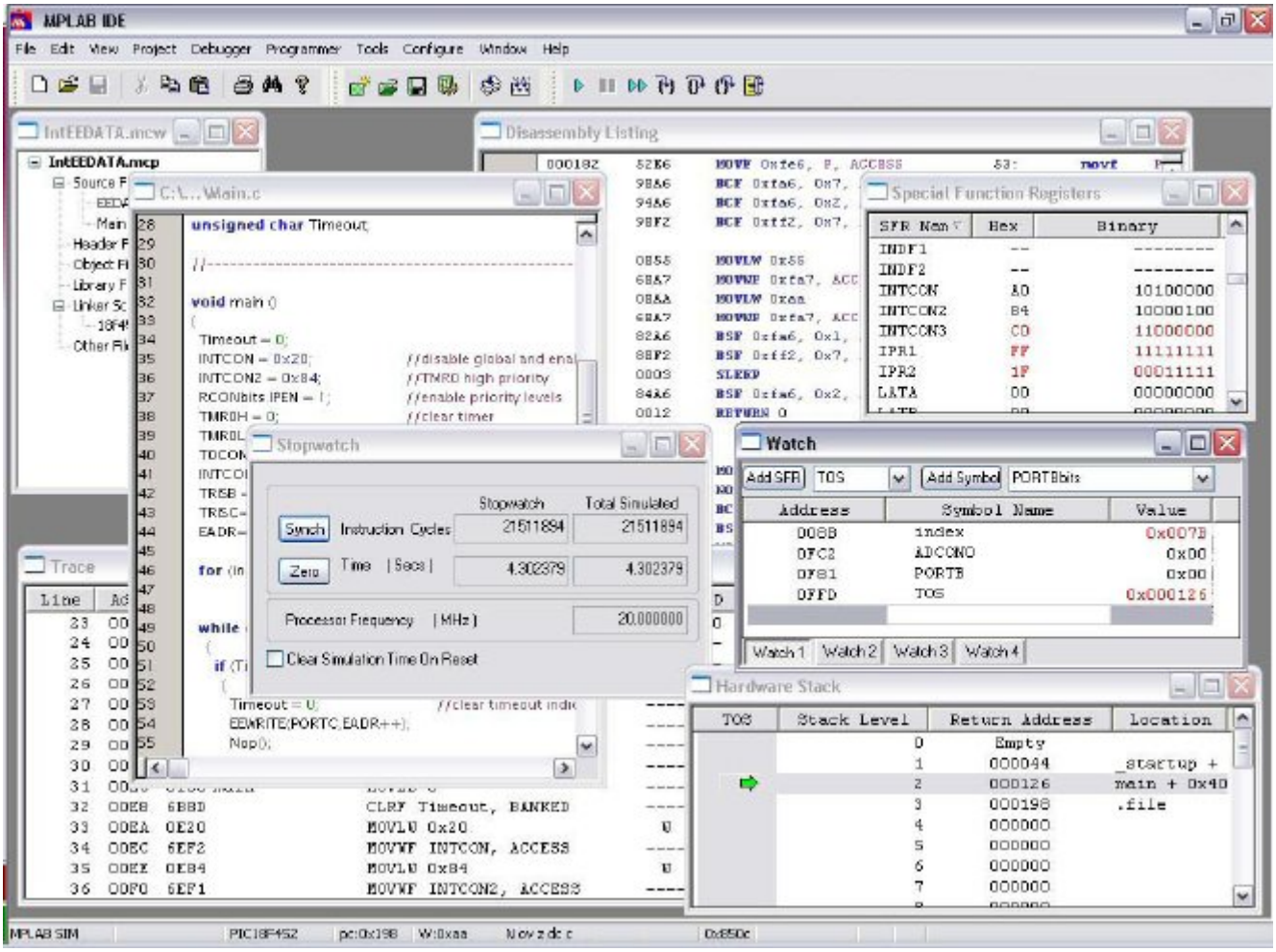
Plan for second term



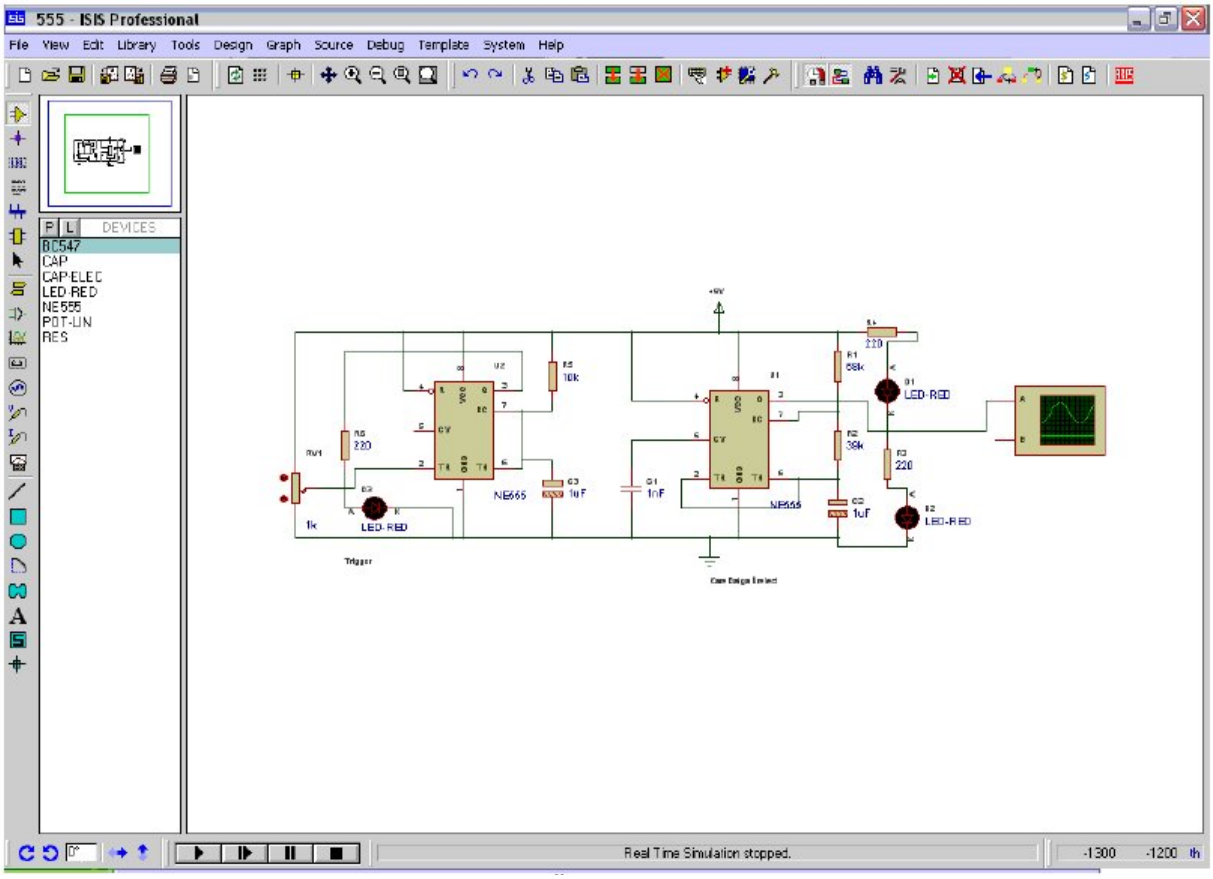
## References


1. http://www.htsoft.com/
2. http://www.oshonsoft.com/index.html
3. http://www.dattalo.com/gnupic/gpsim.html
4. http://www.antrak.org.tr/
5. http://www.microchip.com/
6. PDB user manual

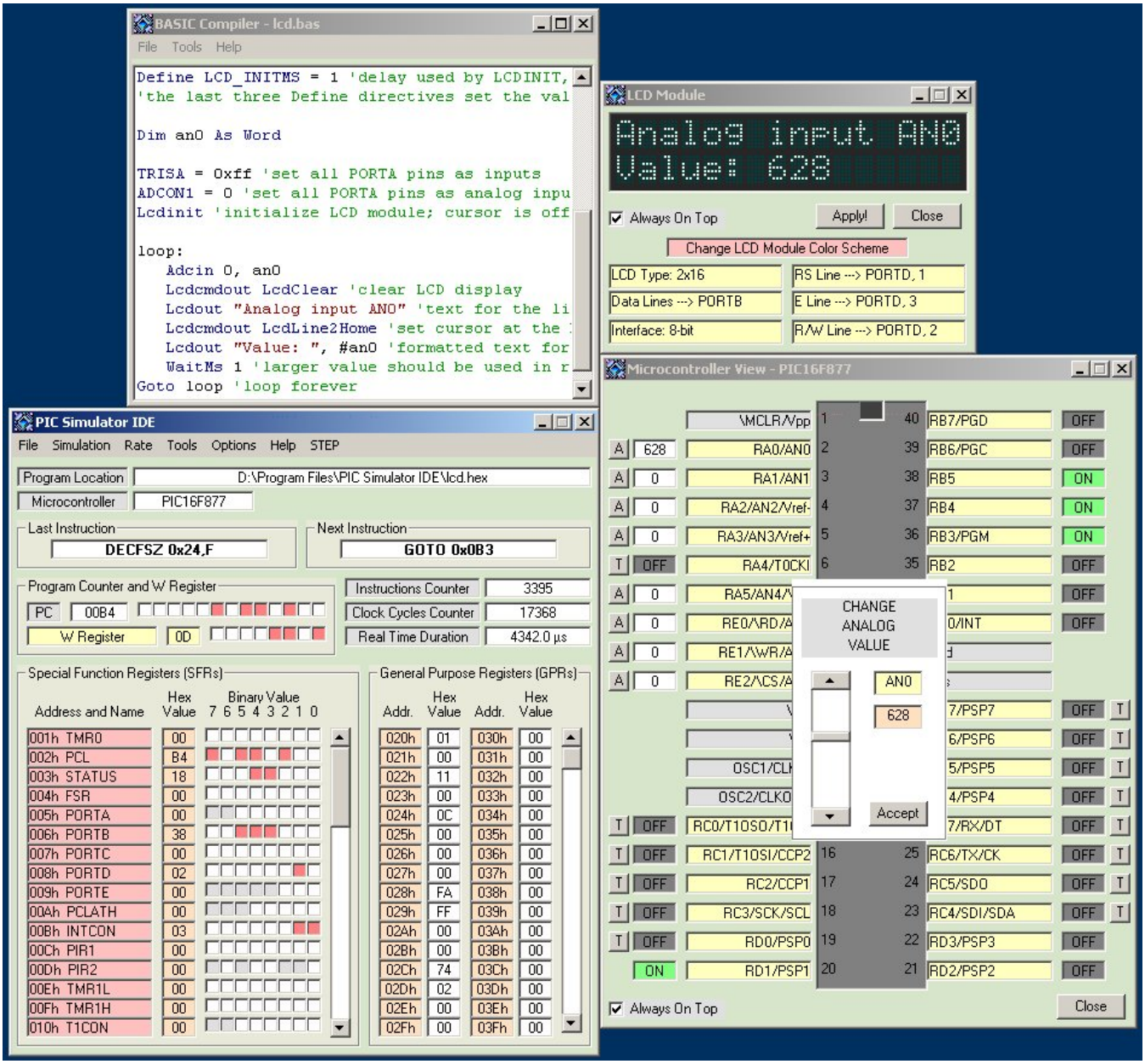
# Appendix




Screenshot of MPLAB




Screenshot of Protheus




Screenshot of PIC Basic Compiler