Middle East Technical University
Department of Computer Engineering

CENG 491

Computer Engineering Design I

2006-2007

SimSys Corporation

Requirements Analysis Report

PIDE

Emulator and Development Environment for

CEng Embedded System Card

05.11.2006

TABLE OF CONTENTS

1. INTRODUCTION

1. 1. Purpose of the Document

In this report, the requirement analysis of the PIDE (PIC Integrated Development Environment) Project is presented.

PIDE project involves implementation of a complete IDE compatible with the CENG Embedded Systems Board that is being used in CENG 336 course. The project will be carried out by the SimSys Corporation during the eight-month period starting in October 2006 and ending in May 2007.

In the first part of this report, the general project description is given. Furthermore, the results of the surveys and interviews that are performed by the group to gather information are presented. In the second part, the functional and non-functional requirements are explained to have a complete description of the project. In the third, fourth and fifth parts, the PIDE project is analyzed in detail and scenario-based model, behavioral model and flow-oriented model of the project are given. In the sixth and last part, the Gantt chart is included to present the project plan.

1. 2. Project Description

As the technology evolves, the embedded systems start to find wide area of usage. In most of the devices that people use daily, there exists a core logic which is mostly an embedded microcontroller or microprocessor with some external storage. Besides, those integrated devices also let the implementation and testing of various new controller ideas very easily. This popularity of Embedded Systems is a little overshadowed by the difficulty in developing embedded software due to the lack of a well fitted development environment and pre-testing it on a special independent system prepared just for testing purposes.

An example to the above discussion exists for the CEng336 Embedded Systems course. Among the course contents, development of embedded software and testing on a test board is of primary importance. However, obviously a standalone testing environment that will simulate exactly the same features with high accuracy would greatly simplify the testing procedure.

As a solution to the problem stated above, SimSys Corporation will develop an emulator and development environment for the card used in Ceng336 Embedded Systems course. Considering such a development and simulation environment, the system will support various types of microcontrollers, communicate through various interface standards such as parallel, serial or USB and accommodate some display interfaces such as LCD or LED driving structures. Users will have the chance of compiling their programs and they can test and debug it on the virtual card emulated by the software.

For such a development and simulation environment design project, the implementation areas are unlimited just as the fact that the implementation areas of the embedded systems are unlimited. As a result, such a system, which will simplify the development and testing process, will find great interest from the embedded systems developers. Together with the Ceng336 Card, this software will be useful for computer engineers, electrical engineers, high school students and everyone interested in PIC programming.

1. 3. Survey and Results

To better understand the needs, requirements and expectations of the customers, SimSys Corp. made interviews with the assistants of Ceng336 course, Alper Kılıç and Fatih Gökçe. During the interview, we asked several questions to the assistants and they shared their ideas and expectations from the project. Below are the main topics that are discussed during the interviews and questions we asked in the interviews.

- Widely used embedded systems development software, their useful features and their shortcomings
    - Which software do you use for PIC programming?
    - Which properties do you frequently use? Are there any features they lack?
- Expectations from a software to be called as "a complete PIC programming IDE"
    - What do you expect more from complete PIC programming software?
- Restrictions and program specific features that should be implemented in PIDE
    - Are there any specific requirements of this project regarding the Ceng336 course?
    - Should there be any limitations of the software concerning the assignments of Ceng336 course?
- Feasibility and necessity of the features that are proposed in Project Proposal Report
    - Do you prefer this project to involve a high level (i.e. C) compiler? Or, do you think that assembly will be adequate?
    - What are your opinions about the analysis tools we have stated in the proposal report?
    - Would it be useful to develop software that is OS independent?
    - Will it be possible to implement hardware debugging property? Will it be useful?

Conclusions derived from the interviews can be summarized as below:

In Ceng336 course, the programming tools mainly used are MPLAB IDE [1], PIC Simulator IDE [2] and ISIS [3]. The most used features of these programs are their simulation capabilities, macro definitions that simplify coding.

However they lack a high-level language compiler support that satisfies our customers. Moreover they are OS dependent.

These programs are useful for general PIC programming purposes; however, they do not have one-to-one correspondence with the Ceng336 course. It would be useful to have software that is specifically designed for Ceng336 Embedded Systems Card. Furthermore I2C protocol support is proposed to be a possible feature.

It would be useful to provide a high-level language compiler, which is also capable of doing optimizations and applying code restrictions. It would be preferable that the software is Linux compatible since this is the most common OS in the department. The PIC family used on the board has hardware debugging support, thus further research is conducted on on-board debugging feature and it would be a useful addition to the software.

The survey has provided valuable ideas and helped us determining the topics for further research.

1. 4. Market Research

According to SimSys Corp. market research, there exist software for purposes like development, simulation, etc. but none of them contains everything necessary for a complete IDE for embedded system programming and simulation. Also there are no specific software solutions for the CEng336 Embedded Systems. Hence SimSys Corporation decided to offer a solution to this problem. The company starts off a new project which brings all necessary components for embedded systems development together in single software. The software is supposed to support all the features of CEng336 Embedded Systems board and simulate all of its functionality. Besides, it provides a complete IDE for embedded system programming.

One of the essential features of PIDE is the support for a high-level programming language. Hence, a module, namely a compiler, is needed for the software to handle the process of converting this high-level language into machine code for PIC microprocessor family. So a market research is performed and widely-used compiler packages for PIC family are examined. Below are short descriptions of these compilers and some of their features that could be implemented in Project PIDE.

Hi-TECH C compilers for embedded chipsets

Hi-TECH [4] software provides a variety of compilers for different integrated circuit families. The company has 4 solutions for Microchip PIC family. These are PICC Enterprise, PICC-18, PICC-18 Pro and dsPICC. PICC Enterprise edition gives the widest PIC support and includes all the features that are included in the other three compilers. PICC Enterprise is a C compiler, supports all standard C language. It works under Linux and windows environment and the compiler may be integrated into other software via plug-ins (i.e, MPLab IDE has a plug-in for integrating PICC compiler). It produces optimized code. It allows compiler to be run through command line. Although Hi-TECH PICC C compilers optimized code, they don’t support I2C protocol.

CCS C Windows IDEs

CCS software [5] has several C compilers. PCWH is one of them which support Microchip PIC family. PCWH has a Windows IDE but also can be used with command line. It supports all standard C language and provides other built-in functions. With the powerful IDE, the user can easily write code in C and produce

optimized output to use in PIC. The compiler provides an interface for MPLab IDE, too.

IAR Systems C/C++ compilers:

IAR [6] provides a variety of C/C++ compilers for embedded system programming. Several compilers support a subfamily of Microchip PIC family. One of these compilers is IAR Embedded Workbench for PIC18 which is an integrated development environment for building and debugging embedded applications. An interface common to the assembler, compiler, project manager, editor, builder and debugger tool ensures ease of use. The compiler for PIC18 uses C/C++ and the C-SPY Debugger supports RTOS-aware debugging on hardware or in a simulator. The compiler supports MPLab debugger.

Based on this research, it is a hard task to provide a fully featured and high-quality compiler considering the time and experience constraints. So it is wise to think of a compiler for a high-level programming language as an option that can be implemented in the further steps of the project. An interpreter for a well-defined and reduced macro family to help user in coding process would be more convenient to be included in the project.

Apart from the above research, the sources present in GNU PIC [7] group are examined to find out whether there are useful projects, sources, etc. for us.

GNU PIC website has PIC sources in the below categories:

- Assemblers
- Disassemblers
- Compilers
- Simulators
- Programmers
- Interpreters
- Libraries
- IDEs

Unfortunately, most of the projects under these categories are not open source, but still they are useful for us to observe the implemented features of them and to detect the missing features that would be useful for a PIC IDE.

Among various projects in GNU PIC site, below ones are the ones that would be most useful for us:

miSim DE 2.1[8]: A development environment that is for writing and debugging software for microcontrollers. It includes an editor, assembler and disassembler. The impressive core of this package is a simulator that simulates not only the microcontroller itself, but also devices connected to it in real time - from simple switches and LEDs to video displays and stepper motors. What is nice with miSim is that, its first version is open source and developed in Java. Since java is the language that we are planning to use, e4xamining these sources would be beneficial.

PP06 PIC Programmer Software [9]: Programming software for PICs that uses parallel port. It is also open source and developed in C language. The programmer supports 12 and 14 bit PICs (ie 16CXX, 16FXXX, 12CXXX), and has windows and Linux versions.

PiKdev[10]: A simple graphic IDE for the development of PIC-based applications. It currently supports assembly language. C language is also supported for PIC 18 devices. PiKdev is developed in C++ under Linux and is based on the KDE environment. It has a full featured multiview editor, with lines numbering, blocks folding, bookmarks, syntax highlighting and tabbed access to various buffers; a project manager with standard functionalities; and a programming engine which allow programming various flavors of PIC microcontrollers via classic programming hardware connected to the parallel port or to the serial port.

GNU PIC LIBRARY PROJECT[11]: The interest of this project is to develop a set of Libraries that are released in LGPL License to use to PIC microcontroller programming. It has an LCD and a PC keyboard library for the present.

Gputils, tpasm, PTK4L are other development environments that are not open source but free to download.

These projects will be examined in detail to state out the features expected from a PIC IDE. And during the coding period in second semester, the open source ones will be taken into account to find out whether they contain anything useful for our project PIDE.

Another idea to design a brand new compiler, whose specifications are determined by our own requirements, rather than relying on the compilers already developed for PIC microprocessor family. There are a bunch of projects currently in the market. Most of these projects are open source. I have chosen the ones that I thought we could benefit from as our tendency is to make PIDE being able to compile codes written in an object-oriented language such as C++ or Java.

Yacc (Yet Another Compiler Compiler)

Yacc [12] is a computer program that serves as the standard parser generator on Unix systems. It generates a parser (the part of a compiler that tries to make sense of the input) based on an analytic grammar written in BNF notation. Yacc generates the code for the parser in the C programming language. Since the parser generated by Yacc requires a lexical analyzer, it is often used in combination with a lexical analyzer generator, in most cases either Lex or the free software alternative Flex. The IEEE POSIX P1003.2 standard defines the functionality and requirements to both Lex and Yacc.

Sablecc

SableCC [13] is an object-oriented framework that generates compilers (and interpreters) in the Java programming language. This framework is based on two fundamental design decisions. Firstly, the framework uses object-oriented techniques to automatically build a strictly-typed abstract syntax tree. Secondly, the framework generates tree-walker classes using an extended version of the visitor design pattern which enables the implementation of actions on the nodes of the abstract syntax tree using inheritance. These two design decisions lead to a tool that supports a shorter development cycle for constructing compilers. There are many projects being developed using the main idea of sablecc which include user-defined or co-developed languages as well as specific debuggers.

Javacc

JavaCC [14] (Java Compiler Compiler) is an open source parser generator for the Java programming language. JavaCC is similar to Yacc in that it generates a parser for a grammar provided in EBNF notation, except the output is Java source code. Unlike Yacc, however, JavaCC generates top-down parsers, which limits it to the LL(k) class of grammars (in particular, left recursion cannot be used). The tree builder that accompanies it, JJTree, constructs its trees from the bottom up.

ANTLR (ANother Tool for Language Recognition)

ANTLR [15] is a parser generator that uses LL (k) parsing. ANTLR's predecessor is a parser generator known as PCCTS. ANTLR rules are expressed in a format deliberately similar to EBNF instead of the regular expression syntax employed by other parser generators. At the moment, ANTLR supports generating code in the following languages: C++, Java, Python, C#. ANTLR 3 is under a 3-clause BSD License.

1. 5. Hardware and Software Requirements

1. 5. 1. Development Phase

The PIDE program is supposed to have a platform independent structure. This will allow users with various operating systems to be able to use the program without encountering any problems. To achieve this goal, the project will be developed in Java. Thus, we need Java Run Time Environment, Java SDK and Eclipse.

To test the platform independence feature, we will use Windows and Linux platforms. The application is intended to work on any platform having a compatible Java Runtime Environment.

During the development, we will need CENG Embedded Systems Board and a PC as hardware requirements. The PC should be able to support Java and must have parallel and serial ports that will be used during programming the card.

1. 5. 2. End User

Since the application is to be developed in Java, the end user will only need a Java Runtime Environment. At the end of the project, PIDE distribution package will not include the Java Runtime Environment.

User should also have the CENG Embedded Systems Board to upload and test the program he/she has written.

1. 6. Company Organization

Company Name

SimSys Corporation

Contact

simsysc@yahoogroups.com

Members of the Team

Mahmut Sami Aktaşoğlu	mahmut.aktasoglu@gmail.com
Özgür Çakmak	ozgurceng@yahoo.com
Emre Kültürsay	kultur@mems.eee.metu.edu.tr
Gülhan Serhat	gulhan.serhat@gmail.com

2. REQUIREMENTS

2. 1. Functional requirements

2. 1. 1. Programming the PIC Microcontroller

In order to program a PIC microcontroller product of Microchip Company [16], there are two choices of programming modes. First one is programming the device on an external programming system. Here the controller should be removed from the application circuit and then placed on the programming environment. This programming structure is not feasible for some cases such as removing the controller from the application circuit is not possible. Therefore, a special programming methodology is necessary. The solution to this problem is In Circuit Serial Programming (ICSP), which is some extended version of the standard programming routine.

ICSP [17] requires special modules to be included in the application board design process, namely isolation structures. Isolation is compulsory since the programmer should not be loaded by the application circuitry and the application circuitry should be protected from the relatively higher programming voltages.

The 16C series PIC MCU's [18] is the microcontroller family to be used in the CEng336 board. Pins RB6 and RB7 are used by this micro controller family for serial programming. RB6 is the clock line and RB7 is a bidirectional pin. This bidirectional pin is used by the programmer during the programming process and used by the microcontroller for verification. As a result, isolation of those pins are very important.

2. 1. 2. Create/Open/Save Project and Files

In order to have a mode compact view of the source files, the software should enclose the source files aroud project files. The obvious benefit of having a project file is the application of workspace concept which will allow directly loading of the state of a previously saved environment. This loaded project will launch with its assigned macro file(s) and test files.

The software should be capable of adding or removing existing or new source/test files to/from the project and saving opened files and project. The files

currently included in the project shall be displayed on the workspace pane of the graphical user interface.

2. 1. 3. Text Editor

A successful, operational text editor should include various basic but necessary features such as undo-redo mechanisms, syntax coloring, smart indentation, multiple pane support, bookmarking, find/search/replace commands, matching brackets, and etc. Those features will be included with more possible extensions.

2. 1. 4. Extension to Assembly Language - Macro Source Files

The assembly language is the most powerful programming language choice without any doubt. However, developing exclusive embedded software with the assembly language requires extensive effort. This is because the assembly language lacks various structures such as flow control statements, loops, variables etc. In order to reduce the effort and time consumed by the design process, extra instructions will be added to the standard instruction set. Those statements, called macros, will be transformed into multiple assembly instructions in the compilation process.

2. 1. 5. Test Bench Files to Control Development Board Input Devices

The development board used in the CEng 336 course includes various input devices such as pushbuttons, potentiometers, etc. Therefore, during the real time simulation of the board, the user is responsible for giving the necessary inputs via the user interface with specific timing. However, sometimes the user will just want to concentrate on the outputs, and prefer the inputs be applied automatically from some file, namely test bench file.

Furthermore, there will be cases where a test bench file will be compulsory. Examples are simulation of serial/parallel/usb port connections, smart card reader, high frequency (above maximum human response frequency) input/output, etc.

This test bench file method is widely used in logic simulators (e.g. XILINX WebPack [19]) and is already proved to be successful. In the PIDE software, the

test bench files will have a special format, which will let the user control all input devices with appropriate timing.

2. 1. 6. Simulation

The simulator will make I/O simulation as the name implies applying manual inputs entered by the user. It is optional to make it apply a predefined input waveform at specified time instants. The simulation may well be interrupted at any time instant. Moreover there will also be an analog-to-digital converter inside the simulator. The outputs of a simulation will be shown to the user either by saving the files to the file manager or displaying the results on the screen. The possible outputs are: Observing output waveforms, A/D conversion analysis such as analog input waveform or digital converted representation of the analog input vs. time, and timing analysis such as total time passed during execution or total number of instructions executed.

2. 1. 7. Debugger

Embedded system debugging involves more conceptual layers of a target system than debugging for time-sharing systems. Consider the case of debugging a C program within a time-sharing system. User-debugger interaction occurs almost entirely at a C language level of abstraction. Descent into assembly language and machine code representations of a target program is rare. Suspicions about a compiler bug may require inspection of generated assembly code. In advertent stepping into an optimized library subroutine leads to display of assembly mnemonics and binary numbers. Most programmers can debug their programs exclusively from a source language perspective.

Embedded systems add several dimensions to debugging. Embedded systems include programmable physical devices that have no direct language counterparts at higher levels of abstraction. Their programming requires direct manipulation of registers and state machines. Assembly language programming is common for performance-critical modules. Temporal determinacy is fundamental to a real time embedded system, eliminating the possibility of constraining temporal awareness to a few, isolated regions of code [20].

Thus, implementing a debugger in PIDE will be one of the most effort requiring tasks but still it is a fundamental component of a complete IDE for embedded systems. All the basic functionalities of an embedded debugger, such as setting, enabling and disabling breakpoints, control of the debug process with several stepping options, monitoring contents of registers in different windows and command line debugging utility will be provided in PIDE for different levels of abstractions (namely MACRO and assembly). In addition to the features mentioned, PIDE will be able to manage input operations instead of user, which will save time and effort of the user when he has to type long sequences of input.

2. 2. Non-Functional Requirements

2. 2. 1. User Friendliness

One of the major goals of PIDE project is to implement a user-friendly and easy-to-use IDE for PIC programming. With the help of the market search and the interviews we have performed, it became obvious that there are many development environments for PICs, however most of them have unordered and confusing user interfaces. PIDE will have an uncomplicated and easy to use interface with well-organized menus, toolbars and short cuts. The interface will also be configurable according to user preferences.

2. 2. 2. Easy to Learn

Our market search has revealed that there are many development environments for PICs and most of the embedded system developers use MPLAB IDE, which is a product of Microchip Company. Since PIDE is a newly introduced program, it should be easy to learn to attract the developers and to make them change the program they use.

2. 2. 3. Reliability

As the PIDE program will be used to develop programs, it should be reliable and should have recovery abilities to protect the codes in case of a crash. Furthermore, the simulating tool of the program should be reliable, as it should show the exact behavior of the PIC and the CENG 336 card.

2. 2. 4. Compatibility

The similar programs present in the market are mostly working on Windows. Just a few of the PIC IDEs are Linux compatible but they do not have a satisfying user interface. Consequently, a platform independent program with a nice interface is needed in the market. PIDE project will be developed with Java to accomplish platform independence. This will also be very meaningful considering that PIDE will mainly be used in CENG 336 course and the most common OS in the CENG department is Linux.

2. 2. 5. Performance

The resource usage of the PIDE program should be minimized in order to increase the performance. To achieve this, complexity of the methods used will be taken into account and memory leaks will be avoided.

3. SCENARIO BASED MODEL

This section describes expected features of PIDE from the perspective of the user. Main functionalities of the software are defined through use cases. In the system, two actors and six use cases associated with them are defined. The use case diagram of the system is shown in Figure.

First actor in the system represents the user. It interacts with the system using PIDE's user interface. The second actor in the system represents CEng 336 Embedded System Board. It is the target of system as some outputs of the system is supplied to this actor. Also it supplies the file loaded on the PIC processor to the system and as a result, to the user.

In the next section, the use cases are explained in detail.

Change System Settings (Actor: User)

The user changes system settings. System settings include all kinds of setting options that exists in existing IDEs. By opening a PIDE settings dialog, user will be able to change its preferences, i.e. editor preferences, compilation settings, debugging settings, simulation preferences, directory preferences, communication with CEng 336 Embedded System Board settings and default layout of interfaces.

Manage Files (Actor: User)

The user manages files used by the system. These files may be grouped in three major categories. These are project files, test files and source files. Project files keeps data of projects and workspaces. Test files are used to simulate the CEng 336 Embedded System Board and to automate debugging. Source files are compiled to produce machine code. The user may create, open, edit and save files. The user may also add –possibly external- files to its projects.

Compile Project Files (Actor: User)

The user compiles project files. Project files should be of valid format, i.e. with an extension accepted by the system. Compiling a valid file creates a workspace where user can see compilation status. The user may compile the files individually or together. As the result of compilation, intermediate files (like object files), debug files and files executable by PIC processors are produced.

Debug Project (Actor: User)

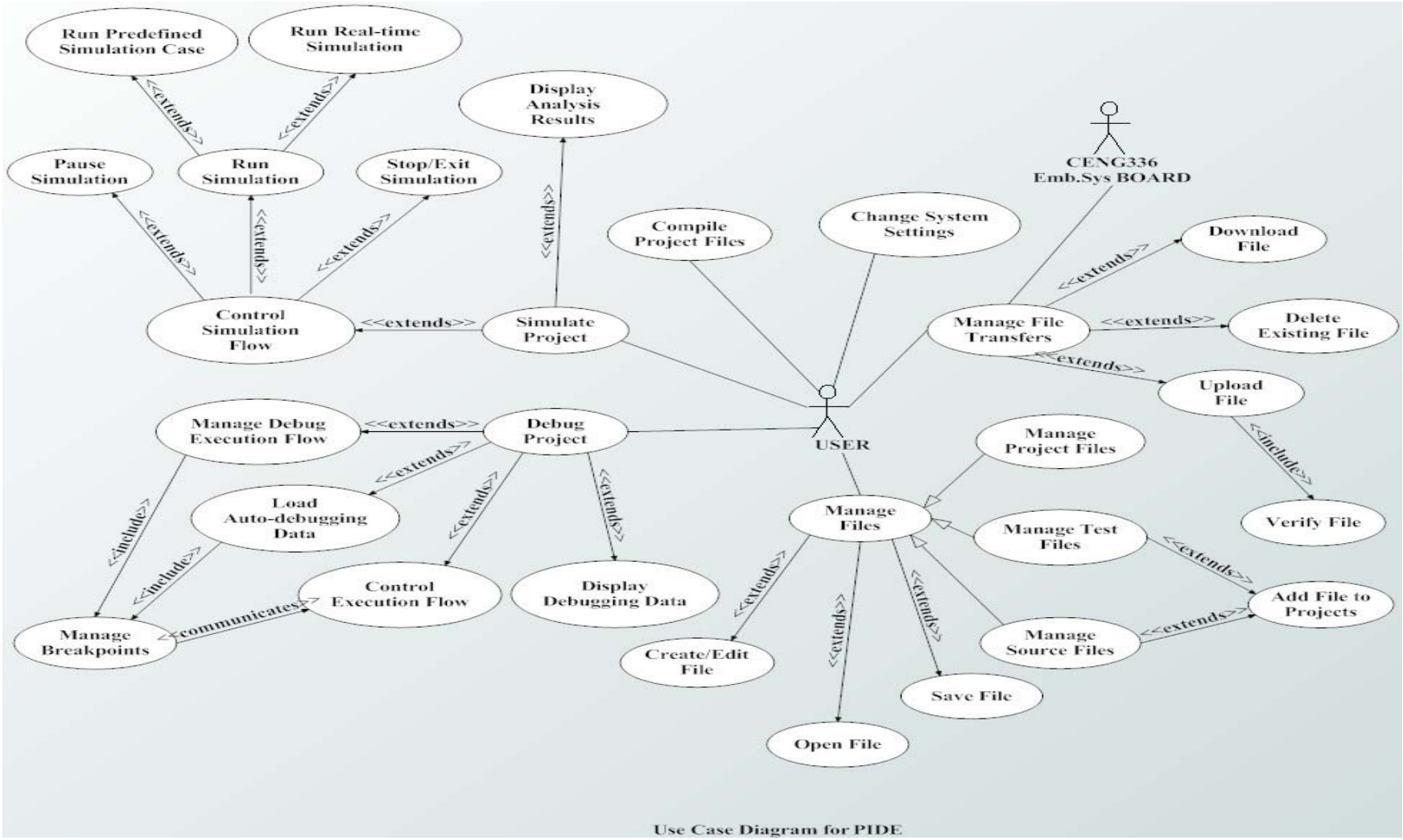
The user debugs projects. To be able to start debugging, all files should be compiled and linked with no error. While debugging, user interrupts execution of the embedded program by adding breakpoints. The user may load auto-debugging files that execute debugging operations and supplies predefined inputs to debugger. The user is able to control the flow of the execution and can see any data related to the embedded system program through PIDE's user interface.

Simulate Project (Actor: User)

The user simulates embedded systems programs. The user may provide files executable by the PIC processors to the system and simulates the real time execution of the CEng 336 Embedded Systems Board. PIDE provides a virtual representation of the board as an interface and runs the simulation by interacting with the interface The user is able to start, pause and stop the simulations. The user may provide simulation files which contain sequences of input data and timing information so that it can simulate desired test cases. The user may monitor useful data and analysis derived from the simulation such as execution time, input/output waveform graphs of the PIC pin legs, etc.

Manage File Transfer (Actors: User, CEng 336 Embedded Systems Board)

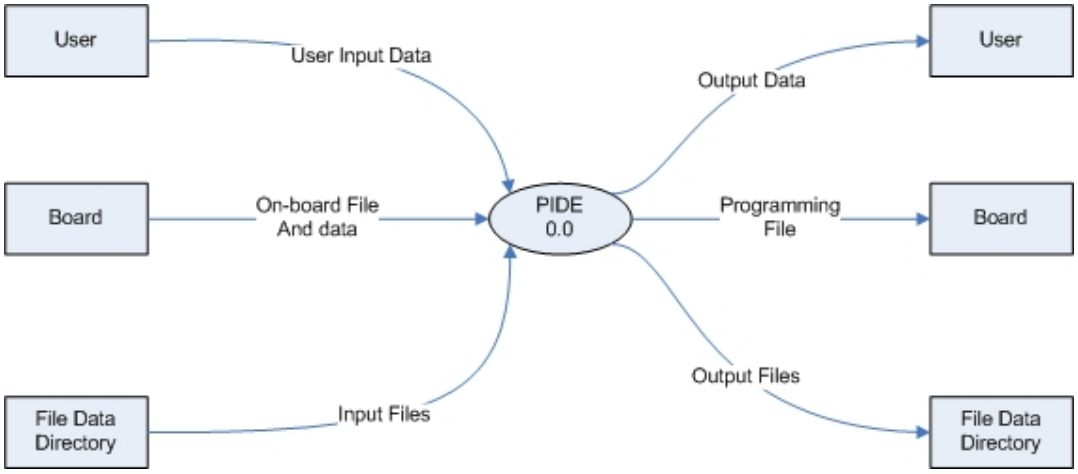
The user manages file transfers between the board and the system. The user may upload files executable by PIC processors to the board. After uploading the file, the system rereads the uploaded file from the board and compares it with the original file to verify that uploading was successful. The user may monitor the machine code which was previously loaded to the board and may erase the current data on the board.



Use Case Diagram for PIDE

4. FLOW ORIENTED MODEL

4. 1. DFD Level 0 - Top View of the Project

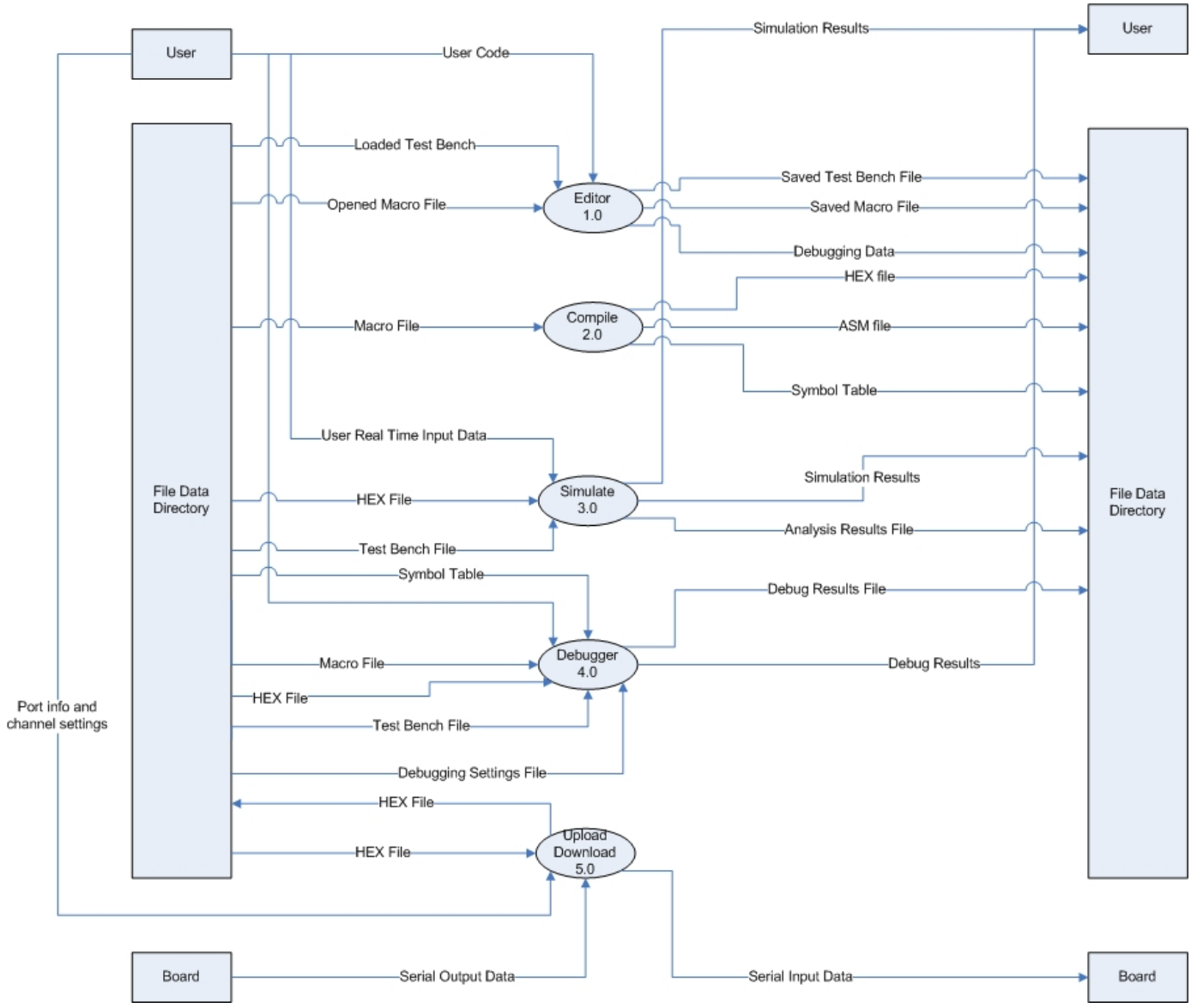


In the topmost level of the data flow diagram, the interaction of the software with the environment is shown.

The software will be highly concentrated around files, that is why the operating system file handling mechanism will be an external interactor. The input data are to be taken from the user, either real time while simulation and debugging, or by means of a test bench file created previously.

The software will communicate with the Simulation Board by means of serial interface.Thus, the software will be capable of programming the microcontroller with the hex file produced after compilation and receiving the hex file inside the microcontroller.

4. 2. DFD Level 1 - PIDE (0.0)



If the top view of the DFD is expanded, the major components of the system are revealed. Those sub-processes are:

Editor(1.0): The text input from the user will be handled by this process. The editor is responsible from viewing the source files, making the corresponding changes to the viewed files in case of user input and redirecting these files to the operating system file handler for saving purposes. The debugging information

such as the breakpoints and memory watches will also be inserted via this process.

Compile(2.0): By definition, compilation is the formation of low level ASM and HEX files from a higher level Macro files. A Macro file is an extended version of an ASM file which has some extra features such as program flow structures(e.g. if-else, while, etc.). Another output of compile process will be a file, containing information about the relation between the Macro and ASM files.

Simulate(3.0): In order to simulate the board, the HEX file need to be interpreted line-by-line. Various inputs to the microcontroller will be given either real time via the graphical user interface or by means of a special file, namely a test bench file. This special file will have its own format, controlling the input devices existing on the development board. The viewable results of the simulation will be displayed via the GUI and special analysis results (e.g. number of instructions executed, waveform on a node, etc.) will be saved to analysis result files.

Debug(4.0): Another feature of the software is the debugger. The debug process will execute the hex file line by line, stopping at the breakpoints defined within the debugging settings file. The execution will stop at those lines and ask for user's "continue" command, displaying the current position on the macro file. Similar to the simulation, the results will be shown and/or saved.

Upload/Download(5.0): The software will also be capable of programming the microcontroller existing on the development board and reading the existing file on the device. This operation will be performed via either parallel or serial port. The port information and channel settings will be entered by the user.

4. 3. DFD Level 2

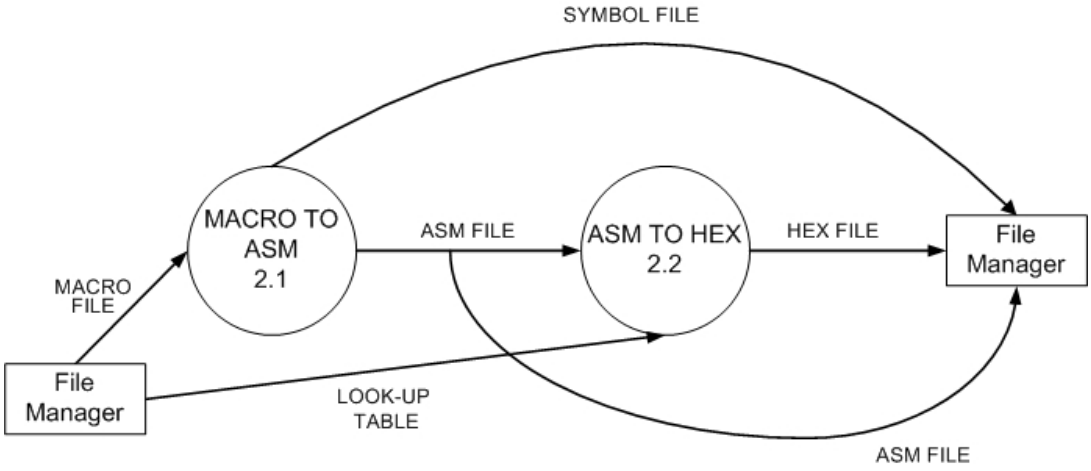
4. 3. 1. Editor Process (1.0)

The editor process is actually the process that handles the major interaction between the user and the computer. Creating, opening, saving, inserting files and inserting source code to those files are performed by the editor. Besides, the settings related to the control information of other processes are updated via the editor.

4. 3. 2. Compile Process (2.0)

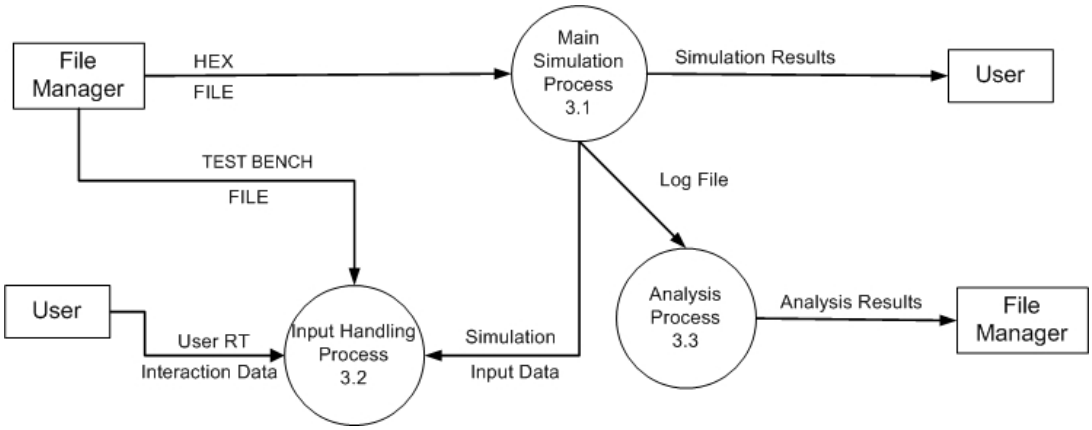
There are two main processes in the second level of this process.

Macro to ASM (2.1) takes a Macro File from the File Manager and produces two files after processing it. These are Table File and ASM file. Table file includes the line numbers of the breakpoints corresponding to the lines in the ASM file. This table file is directly sent to the File Manager. The ASM file is the ASM format of the same Macro file which includes memory address numbers as well as assembly words, and it is sent to the second main process, which is ASM to Hex (2.2).



This process, as the name implies, takes an ASM File and a Look-up Table provided by the file manager and converts it into a Hex File which consists only of the pure numbers. The ASM File is also sent to the File Manager to be saved for the possible future use.

4. 3. 3. Simulate Process (3.0)



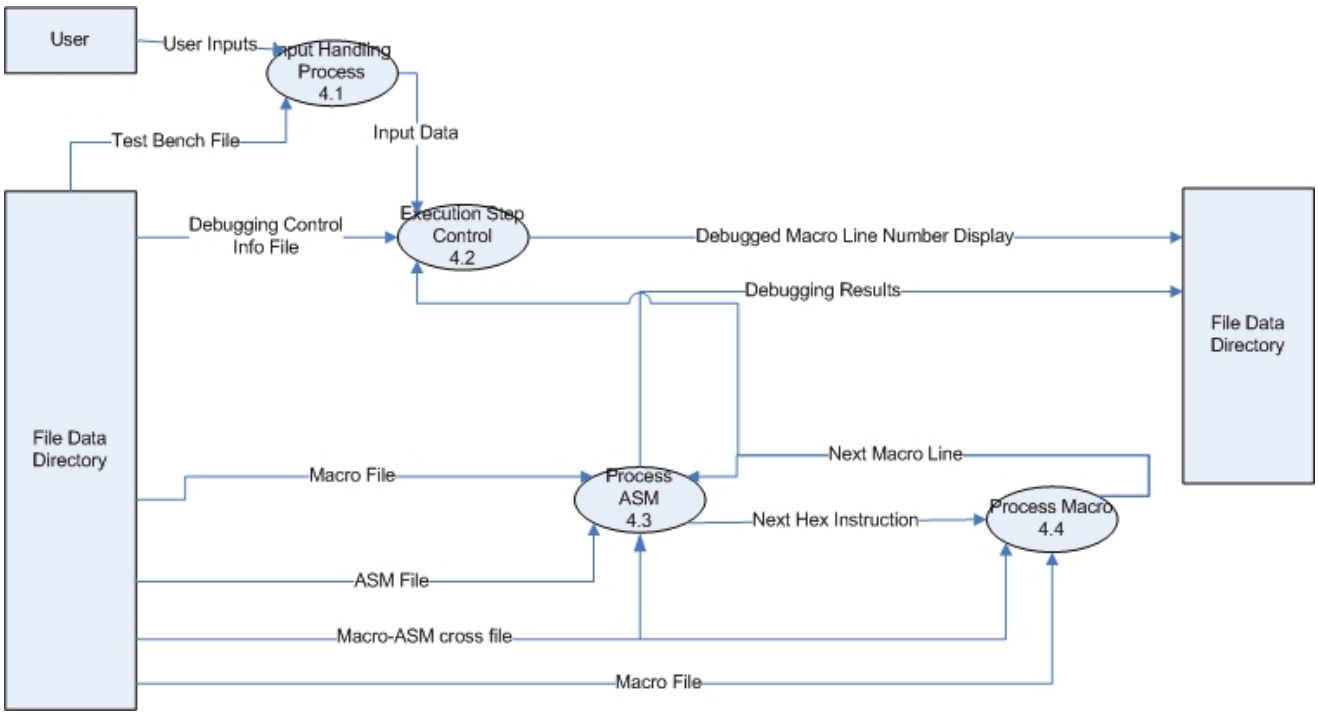
This process includes three inner processes inside which are Main Simulation Process (3.1), Input Handling Process (3.2) and Analysis Process (3.3).

Input Handling Process takes a Test Bench File from the File Manager and the Real-time Interaction Data provided by the user and after processing it produces the Simulation Input Data which is directly sent to the Main Simulation Process.

Main Simulation Process takes a HEX File from the File Manager and produces three different results out of this single file. One of them is Simulation Results which is directly sent to the display in order to provide the user an actual output of the simulation. Another file produced by this process is the Log File which is sent to the Analysis Process.

Analysis Process creates the Analysis Results File by processing the Log File and sends it to the File Manager.

4. 3. 4. Debug Process (4.0)



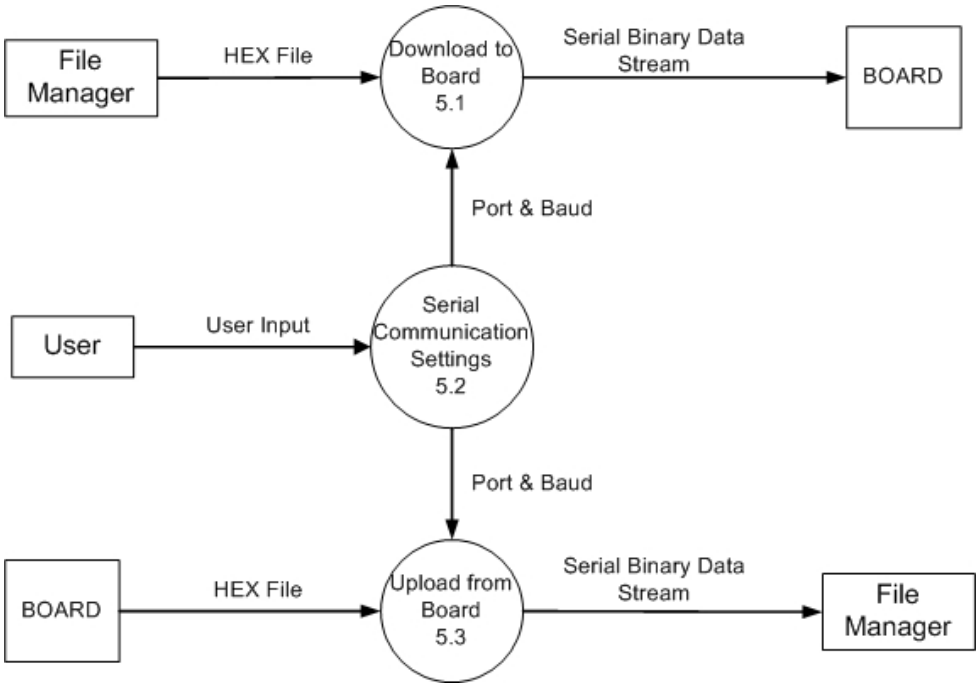
In order to debug a Macro input file, the user should first compile this file and obtain the Macro-ASM cross table file. This file contains a table which maps the instructions in Macro file with the instructions in the ASM file line-by-line in a one-to-many fashion. Using this file, one can find the corresponding ASM instructions to any instruction in the Macro file and vice versa.

Debugging process checks the debugging control file for the next breakpoint and checks if the "next macro line" is a line with a breakpoint. If it is not a breakpoint, nothing happens and execution continues. If it is a breakpoint, then it stops execution and starts waiting for a user "continue debugging" signal. Meanwhile, the line number in the macro file is used to display the current instruction location on the editor.

Once execution starts, "Process Macro" calculates the current macro line number (initialized to 1 at start-up) and sends it to the "Process ASM". This process finds the corresponding ASM instructions and executes them. After execution, the Program Counter location is used to find the line number of the

next ASM instruction. This instruction line number is fed to the "process macro" again. This line number is mapped to a Macro line number by means of "macro-asm cross file" to calculate the next Macro line. This process continues until a breakpoint is encountered.

4. 3. 5. Upload-Download Process (5.0)



Upload – Download Process is the one which deals with the read- write operations with the board. It includes three processes in this level. These are Download-to-Board (5.1), Serial Communication Settings (5.2) and Upload-from-Board (5.3).

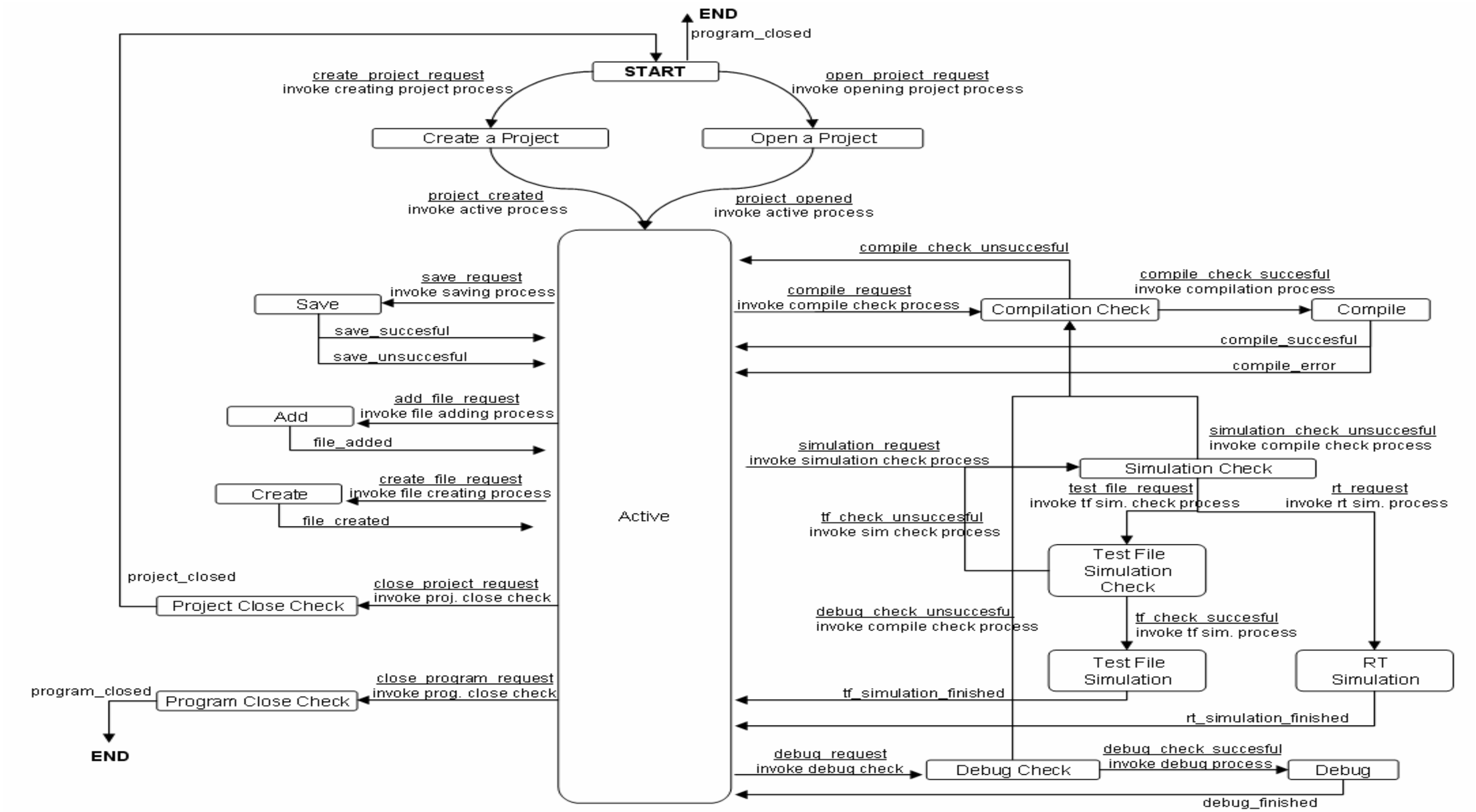
Serial Communication Settings Process takes an input from the user, which specifies the port number of the board and the baud rate, and produces an output which is sent to two other processes.

Download-to-Board takes the Hex File from the File Manager along with the Port & Baud rate data created by the Serial Communication Settings Process, creates the Serial Binary Data Stream and sends it to the board from its buffer.

Upload-from-Board Process takes the Hex File from the Board along with the Port & Baud rate data created by the Serial Communication Settings Process, creates the Serial Binary Data Stream and sends it to the File Manager from its buffer.

5. BEHAVIORAL MODEL

5. 1. State Transition Diagram



5. 2. States

START

When the user runs the PIDE program, it opens in the START state. In this state, the program is running but no active project exists.

Behavior

- With create_project_request, the program passes to CREATE_A_PROJECT state.
- With open_project_request, the program passes to OPEN_A_PROJECT state.
- With program_closed, the program passes to END.

CREATE_A_PROJECT

Here, creating a new project process is performed.

Behavior

- With project_created, the program passes to ACTIVE state.

OPEN_A_PROJECT

Here, opening an existing project process is performed.

Behavior

- With project_opened, the program passes to ACTIVE state.

ACTIVE

This is the main state of the program where exists an active project. User can add existing source/test files to the project, create a new source/test file, enter code to these files, edit the code entered and save these files. User can also compile, simulate and debug the project.

Behavior

- With compile_request, the program passes to COMPILATION_CHECK state.
- With simulation_request, the program passes to SIMULATION_CHECK state.
- With debug_request, the program passes to DEBUG_CHECK state.
- With save_request, the program passes to SAVE state.
- With add_file_request, the program passes to ADD state.
- With create_file_request, the program passes to CREATE state.

- With close_project_request, the program passes to PROJECT_CLOSE_CHECK state.
- With close_program_request, the program passes to PROGRAM_CLOSE_CHECK state.

COMPILATION_CHECK

This state checks whether there exists a macro file or an asm file, and gives an error if not.

Behavior

- With compile_check_succesful, the program passes to COMPILE state.
- With compile_check_unsuccesful, the program passes to ACTIVE state.

COMPILATION

Here, compilation process begins, and an error message is generated if there happens a compile time error.

Behavior

- With compile_succesful, the program passes to ACTIVE state.
- With compile_error, the program passes to ACTIVE state.

SIMULATION_CHECK

This state first checks whether there exists a hex file, and invokes compilation if not. If the hex file exists, user will select the simulation type: using a test file or real time simulation.

Behavior

- With simulation_check_unsuccesful, the program passes to COMPILE CHECK state.
- With test_file_request, the program passes to TF_SIMULATION_CHECK state.
- With rt_request, the program passes to RT_SIMULATION state.

TF_SIMULATION_CHECK

This state checks whether there exists a test file and generates an error message if not.

Behavior

- With tf_check_unsuccesful, the program passes to SIMULATION_CHECK state.
- With tf_check_succesful, the program passes to TF_SIMULATION state.

TF_SIMULATION

Here, the simulation is performed using the test file.

Behavior

- With tf_simulation_finished, the program passes to ACTIVE state.

RT_SIMULATION

Here, real time simulation is performed through the user inputs.

Behavior

- With rt_simulation_finished, the program passes to ACTIVE state.

DEBUG_CHECK

This state checks whether there exists a hex file, and invokes compilation if not.

Behavior

- With debug_check_unsuccesful, the program passes to COMPILE_CHECK state.
- With debug_check_succesful, the program passes to DEBUG state.

DEBUG

Here, the debugging process is performed.

Behavior

- With debug_finished, the program passes to ACTIVE state.

SAVE

This state saves the current file and generates an error message if it cannot be saved.

Behavior

- With save_succesful, the program passes to ACTIVE state.
- With save_unsuccesful, the program passes to ACTIVE state.

ADD

In this state, an existing source/test file is added to the project.

Behavior

- With file_added, the program passes to ACTIVE state.

CREATE

In this state, a new source/test file is created and added to the project.

Behavior

- With file_created, the program passes to ACTIVE state.

PROJECT_CLOSE_CHECK

This state checks whether the active project is already saved, if not, asks the user if he/she wants to save first. If yes, the project is saved and closed. If the user does not want to save, the project is closed.

Behavior

- With project_closed, the program passes to START state.

PROGRAM_CLOSE_CHECK

This state checks whether the active project is already saved, if not, asks the user if he/she wants to save first. If yes, the project is saved and the program is closed. If the user does not want to save, the program is closed without saving.

Behavior

- With program_closed, the program passes to END.

5. 3. Events

The events that cause transitions in the State Transition Diagram are explained below.

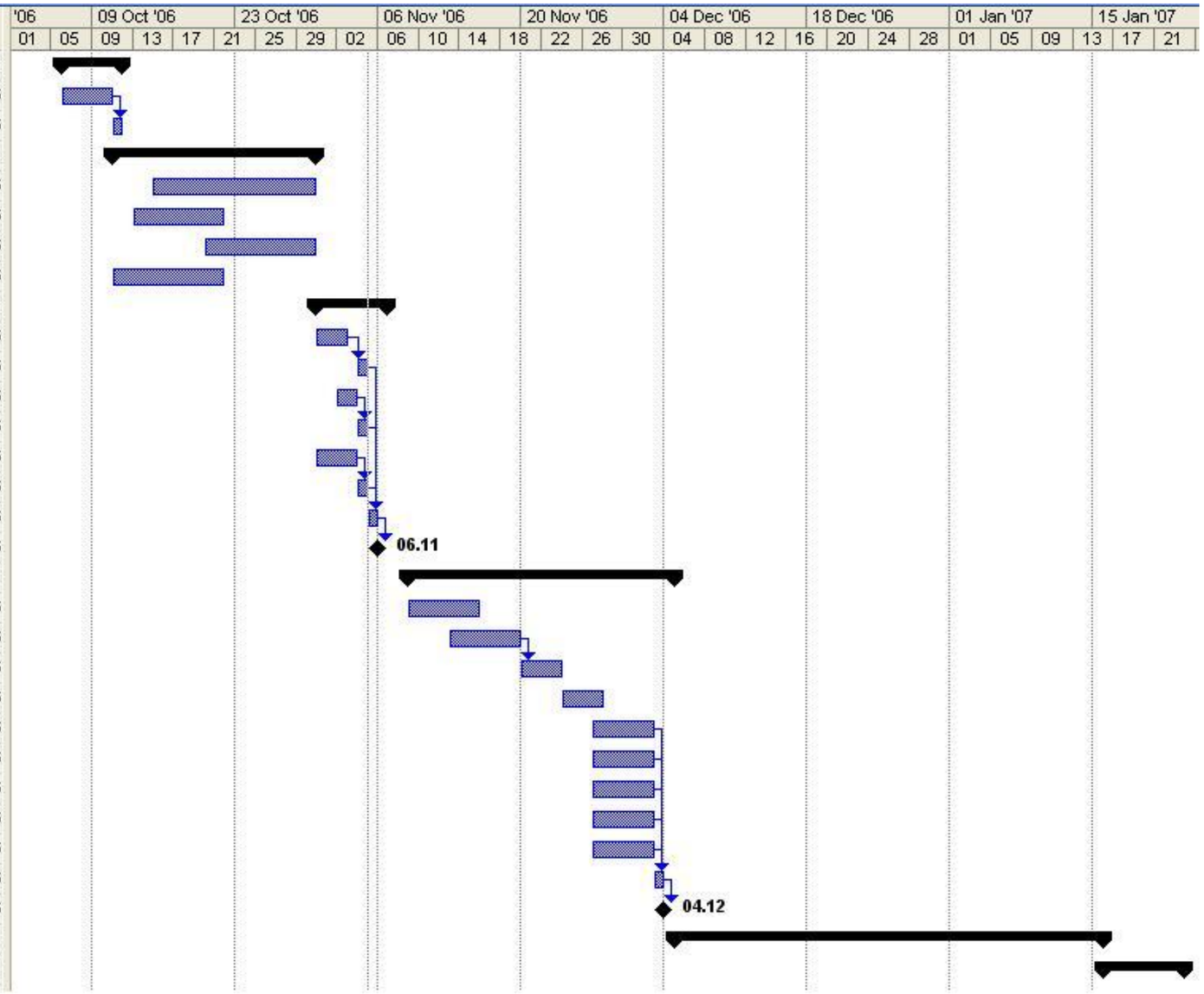
create_project_request	User request to create a new project. (Choosing the 'New Project' option from the menu)
open_project_request	User request to open an existing project. (Choosing the 'Open Project' option from the menu)
project_created	Creating a new project process is completed.
project_opened	Opening a project process is completed.
compile_request	User request to compile the code. (Choosing the 'Compile' option from the menu)

simulation_request	User request to simulate the project. (Choosing the 'Simulate' option from the menu)
debug_request	User request to compile the code. (Choosing the 'Debug' option from the menu)
save_request	User request to save the file. (Choosing the 'Save' option from the menu)
add_file_request	User request to add an existing file to the project. (Choosing the 'Add File' option from the menu)
create_file_request	User request to create a new file. (Choosing the 'New File' option from the menu)
close_project_request	User request to close the project. (Choosing the 'Close Project' option from the menu)
close_program_request	User request to close the program. (Choosing the 'Exit' option from the menu)
compile_check_succesful	Compile check is completed successfully. (A macro file or an asm file is found.)
compile_check_unsuccesful	Compile check is completed unsuccessfully. (A macro file or an asm file cannot be found.)
compile_succesful	Compilation process is completed successfully.
compile_error	Compilation process had compile time errors.
simulation_check_unsuccesful	Simulation check is completed unsuccessfully. (A hex file cannot be found.)
test_file_request	User request to make simulation using a test file. (Choosing the 'Use test file' option from the simulation menu)

rt_request	User request to make real time simulation. (Choosing the 'Real Time Simulation' option from the simulation menu)
tf_check_unsuccesful	Test file simulation check is completed unsuccessfully. (A test file cannot be found.)
tf_check_succesful	Test file simulation check is completed successfully. (A test file is found.)
tf_simulation_finish	Test file simulation process is completed.
rt_simulation_finish	Real time simulation process is completed.
debug_check_unsuccesful	Debug check is completed unsuccessfully. (A hex file cannot be found.)
debug_check_succesful	Debug check is completed successfully. (A hex file is found.)
debug_finished	Debug process is completed.
save_succesful	Saving process is completed successfully.
save_unsuccesful	Saving process is completed unsuccessfully.
file_added	Adding an existing file to the project process is completed.
file_created	Creating a new file process is completed.
project_closed	The project is closed.
program_closed	The PIDE program is closed.

6. SCHEDULE

		Task Name	Start	Finish
1		**Project Proposal**	**Fri 06.10.06**	**Wed 11.10.06**
2		Proposal Report Preparation	Fri 06.10.06	Tue 10.10.06
3		Proposal Report	Wed 11.10.06	Wed 11.10.06
4		**Literature Survey**	**Wed 11.10.06**	**Mon 30.10.06**
5		Information Gathering	Sun 15.10.06	Mon 30.10.06
6		Interview with CEng336 assistants	Fri 13.10.06	Sat 21.10.06
7		Market Research	Fri 20.10.06	Mon 30.10.06
8		User Survey	Wed 11.10.06	Sat 21.10.06
9		**Requirement Analysis**	**Tue 31.10.06**	**Mon 06.11.06**
10		Elaboration on Flow Oriented Model	Tue 31.10.06	Thu 02.11.06
11		Preparation of Data Flow Diagrams	Sat 04.11.06	Sat 04.11.06
12		Elaboration on Scenario Based Model	Thu 02.11.06	Fri 03.11.06
13		Preparation of Use Case Diagrams	Sat 04.11.06	Sat 04.11.06
14		Elaboration on Behavioral Model	Tue 31.10.06	Fri 03.11.06
15		Preparation of State Transition Diagrams	Sat 04.11.06	Sat 04.11.06
16		Analysis Report Preparation	Sun 05.11.06	Sun 05.11.06
17		Analysis Report Deadline	Mon 06.11.06	Mon 06.11.06
18		**Initial Design Report**	**Thu 09.11.06**	**Mon 04.12.06**
19		Use cases	Thu 09.11.06	Wed 15.11.06
20		Defining Structures and Hierarchies	Mon 13.11.06	Sun 19.11.06
21		Preparation of Class Relationship Diagra	Mon 20.11.06	Thu 23.11.06
22		Preparation of State Transition Diagrams	Fri 24.11.06	Mon 27.11.06
23		System Design	Mon 27.11.06	Sat 02.12.06
24		Object Design	Mon 27.11.06	Sat 02.12.06
25		Human Interface Design	Mon 27.11.06	Sat 02.12.06
26		Data Management Design	Mon 27.11.06	Sat 02.12.06
27		Task Management Design	Mon 27.11.06	Sat 02.12.06
28		Initial Design Report Preparation	Sun 03.12.06	Sun 03.12.06
29		Initial Design Report Deadline	Mon 04.12.06	Mon 04.12.06
30		**Final Design Report**	**Tue 05.12.06**	**Mon 15.01.07**
42		**Prototype**	**Tue 16.01.07**	**Tue 23.01.07**



7. REFERENCES

[1] MPLAB IDE –
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002

[2] PICSim IDE – www.oshonsoft.com/pic.html

[3] ISIS – http://www.labcenter.co.uk

[4] HITECH Software Official Web site http://www.htsoft.com/

[5] Computer Customer Services Inc. Official Web Site http://www.ccsinfo.com/

[6] IAR Embedded Workbench-C/C++ compiler and debugger tools
IAR Systems Official web site http://www.iar.com/

[7] GNU PIC - Web page by GNU to share various free and open source PIC programs. http://www.gnupic.org/

[8] miSim DE - A development environment that has both free and charged versions
http://www.feertech.com/misim/homepage.html

[9] PP06 - An open-source programmer for PIC micros developed by Simon Bridger
http://pp06.sourceforge.net/

[10] PiKdev - A KDE based PIC integrated development environment created by Alain Gibaud http://pikdev.free.fr/

[11] GNUPIC LIB - An LCD library written in assembly by Antonio Todobom
http://sourceforge.net/projects/gpiclib

[12] YACC - Parser generator by AT&T for UNIX dinasour.compilertools.net/#yacc

[13] SableCC - an open source compiler generator in Java. www.sablecc.org

[14] Javacc(Java Compiler Compiler) - An open source parser generator
javacc.dev.java.net

[15] ANTLR (Another Tool for Language Recognition) - A parser generator
http://www.antlr.org

[16] Microchip Technology Inc. Official Web Site http://www.microchip.com

[17] ICSP - In Circuit Serial Programming Guide
ww1.microchip.com/downloads/en/DeviceDoc/3027701.pdf

[18] PIC 16 Series
http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1002&mid=10&lang=en&pageId=74

[19] XILINX WebPack www.xilinx.com/ise/logic_design_prod/webpack.htm

[20] Extension Language Automation of Embedded System Debugging - Parson, Schlieder, Beatty DSP16000 LUxWORKS Debugger, Luxdbg Version 1.7.0, Lucent Technologies, December, 1998.